OMRON programmable controller
Host link provider for CJ series

Version 1.0.1

User's guide

August 02, 2022

【 remarks 】

This document has been translated into English using machine translation.

【 revision history 】

Version	Date	Content
1.0.0	2019-07-31	First edition.
1.0.1	2022-08-02	Fixed the value writing process when specifying a channel.

【 communication confirmation equipment 】

Model	Version	Notes
CJ2H-CPU64-EIP	Ver1.5	

Contents

1. Introduction

This book is a user's guide of the CAO provider that write/reads data made of OMRON (CJ series) PLC. CAO provider (CaoProvOMRONCJHostLink.dll) that treats in this book is called CJHostLink provider.

Details of the outline and the variable of the CJHostLink provider have been described to the chapter 2.

The correspondence situation and the data string of the communication command mounted in the CJHostLink provider are shown in PLC (CJ series) that becomes a destination. Serial communications of the FINS command (host link) are used about the communication. Please refer to “sbca-304r-30_cs1_cj1_cp1_com_cmd.pdf" and “sbca-350l-22_cj2h-cpu6-eip_cj2m-cpu.pdf" of OMRON for details.

Moreover, the connection to the serial port must use a specified cable. Please refer to “sbca-349r-1_cj2h-cpu6-eip_cj2m-cpu.pdf" of OMRON for details of the cable.

2. Outline of provider

2.1. Outline

The CJHostLink provider is CAO provider that write/reads data by using the FINS command made of OMRON (CJ series) PLC by the title link (cereal connection). The file format is DLL(Dynamic Link Library), and when using it from the CAO engine, it is dynamically loaded. When the CJHostLink provider is used, it is necessary to install ORiN2SDK or to register the registry by the hand work referring to the table below.

Table2-1CJHostLink provider

File name	CaoProvOMRONCJHostLink.dll
ProgID	CaoProv.OMRON.CJHostLink
Registry registration	regsvr32 CaoProvOMRONCJHostLink.dll
Blotting out of registry registration	regsvr32 /u CaoProvOMRONCJHostLink.dll

2.2. Method property

2.2.1. CaoWorkspace::AddController method

The CJ provider connects the communication referring to connected parameter for the communication at AddController. (Operate as TCP client.)

Format AddController(<bstrCtrlName:BSTR>,<bstrProvName:BSTR>,
<bstrPCName:BSTR>,<bstrOption:BSTR>))

bstrCtrlName : In controller name
bstrProvName : In provider name. fixed value ="CaoProv.OMRON.CJHostLink"
bstrPcName : Execution machine name of in provider
bstrOption : In optional character string

The list specified for an optional character string is shown as follows.

Table2-2Optional character string of CaoWorkspace::AddController

[1]Option ()	Explanation
Conn=< connected parameter >	Indispensability. Communication form and connected parameter. Please refer to chapter 2.2.1.1 for details.
UnitNo =<Unit No for host link>	Unit No. for host link (0-31) Please refer to chapter 2.2.1.2 for details.
FinsParam=<DNA>:<DA1>:<DA2> [:<GCT>[:<SNA>[:<SA1>[:<SA2>:[<SID>]]]]]	Indispensability (Part can be omitted). The header composition parameter of the FINS command. Please refer to chapter 2.2.1.3 for details.
Timeout =< timeout period >	Timeout period when sending and receiving it. (millisecond) (default: 3000)
Retry =< number of retries >	Set the number of communication retries for sending and receiving. (default: 0)
PutOpt:[=< Write operation >]	Specify the data structure when writing to the channel address with BIT, BOOL, UI1, I1. (See 2.4.Appendix C)

[1] Square brackets ("[]") indicate that they can be omitted. The underlined part in the explanation of each parameter is the default value when no option is specified.

2.2.1.1. Conn is optional.

Connected parameter character string of optional Conn is shown as follows. [1].

Cereal device

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

<COM Port>	:	COM port number. '1'-COM1,'2'-COM2, …
<BaudRate>	:	Transmission rate. 300,600,1200,2400,4800,9600,19200,38400,57600,115200.
<Parity>	:	Parity. 'N'-NONE,'E'-EVEN,'O'-ODD.
<DataBits>	:	Number of data bits. '7'-7bit,'8'-8bit.
<StopBits>	:	Number of stop bits. '1'-1bit,'2'-2bit.

(example 1)	"com:1"	Communication port COM1 (, 9600bps, Even, 7bits, 2bit)
(example 2)	"com:2:19200"	Communication port COM2 and 19200bps (,Even, 7bits, 2bit)
(example 3)	"com:3:38400:N:8:2"	Communication port COM3, 38400bps, and None, 8bits, and 2bit

2.2.1.2. UnitNo is optional.

It is a number to specify CPU unit of the other party connected with a host link computer. Title machine No. set by the PLC system construction is set when connected with CPU unit.

If the connection destination is a serial communication board / unit, set the unit number set in the system settings (assigned DM area). (Figure2-1).

(default: 0)

・ Setting of UnitNo (CX-Programmer is used).

Figure2-1Setting of unit-number for host link

2.2.1.3. FinsParam is optional.

Connected parameter character string of optional FinsParam is shown as follows. [1].

"FinsParam=<DNA>:<DA1>:<DA2>[:<GCT>[:<SNA>[:<SA1>[:<SA2>:[<SID>]]]]]"

<DNA>	:	Other party network address (Destination network address)
		0 :Network
		1-127: Other party network address
<DA1>	:	Other party node address (Destination node number)
		0 :PLC secret communication Makoto
		1-32: For the Controller Link network
		1-254: For Ethernet
		255: Broadcast data transmission together
<DA2>	:	Other party title machine address (Destination unit address)
		0 : CPU unit
		254: Correspondence Controller Link unit/Ethernet unit
		16-31: CPU high performance unit
		225: INNER board
<GCT>	:	Number of permissible bridge streets (Gateway Count)
		2, 7
<SNA>	:	From [nettowa-kuadoresu] (Source network address)
		0 :Network
		1-127: Other party network address
<SA1>	:	From [no-doadoresu] (Source node number)

0 :PLC secret communication Makoto

1-32: For the Controller Link network

1-254: For Ethernet

<SA2> : Sending era name machine address (Source unit address)

0 : CPU unit

16-31: CPU high performance unit

<SID> : Service ID(Service ID)

0-255: Given value

2.2.2. CaoController::AddVariable method

The AddVariable method of the CaoController class is a method for making the variable object write/to read data to PLC (CJ series).

In this method, the area type / address of the IO memory to be accessed in the PLC (CJ series) is determined by specifying the variable name in the format shown in Section 2.2.2.1.

Moreover, the type conversion of data and the read writing/number of data can be specified in the options. [2]

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

<bstrVariableName> : In variable identifier

<bstrOption> : In optional character string

The list specified for an optional character string is shown as follows.

Table2-3Optional character string of CaoController::AddVariable

Option	Explanation
Mode =<Access mode>	The mode when it accesses the timer, the counter, and the task flag is specified. (Please refer to chapter 2.2.2.2)
VT =<Variable type >	The data type that does Put/Get is specified. (Please refer to chapter 2.2.2.3)
Elem =< the number of element >	The number of elements of data that does Put/Get is specified. When the data type is VT_BSTR, it becomes byte number specification of the character string. As for the number of elements, specification by another and a hexadecimal number specified by the decimal number is also possible. (0x0A, &h0A, 0AH) (default: 1)
Array[=< True or False >]	Whether it treats as an array when data that does Put/Get is one element is specified. (default: False)

[2] If the packet size of the communication generated by reading and writing data increases, the communication may fail due to the restriction of the FINS command. This can be avoided by reducing the number of data elements that are read and written at one time.

ID =< ID initial value >	ID initial value of the variable is specified. (decimal number) (default: 0)

2.2.2.1. Area type/addressing to be accessed according to variable identifier

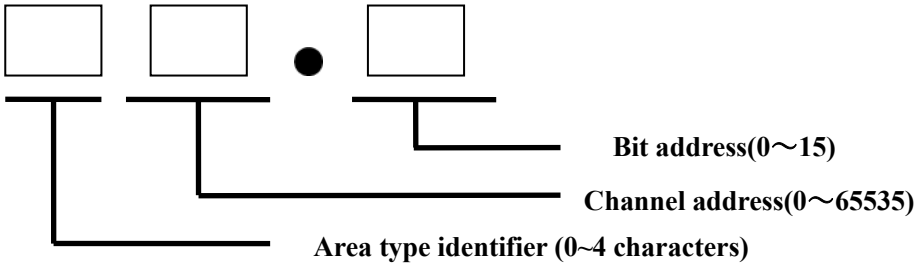
It is composed of the area type identifier (alphabet 0-4 characters), the channel addresses (0 ~ 65535 of the decimal number), and the bit addresses (0 ~ 15 of the decimal number).

When the data kind of the memory for the access is a bit, "." (dot) is placed after the channel address and the bit address is specified. The bit address is not specified for the channel (word).

Only the channel (word) addressing format is used for timers, counters, and task flags, and the data type is specified by the Mode option described later. (Please refer to chapter 2.2.2.2)

The format to specify the memory type and the address is shown below.

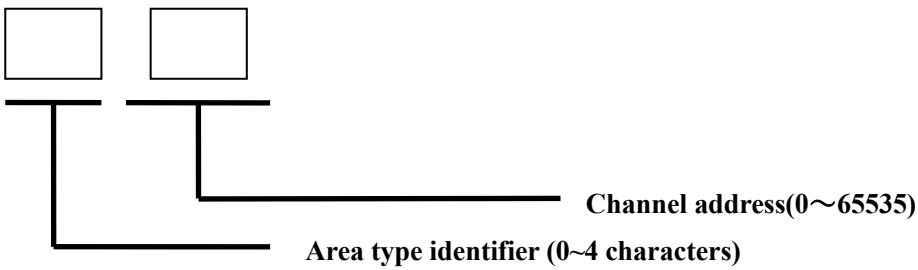
Bit addressing format



ex) channel address 0x0010 and bit 12 of channel I/O(CIO).

⇒ 16.12 or 0016.12 (Treat assuming that the same address is specified regardless of the presence of the digit of 0 burials.)

Channel (word) addressing format



ex) channel address 0x00FF of enhancing data memory EM bank 4.

⇒ E4_255 or E4_0255

The list of the identifier of each IO memory area type is shown below.

Table2-4Area type identifier list

Area type		Identifier
Channel I/O	CIO	None
Internal, supplementary relay	WR	W
Maintenance relay	HR	H
Special, supplementary relay	AR	A
Timer	TIM	T
Counter	CNT	C
Data memory	DM	D
Enhancing data memory	EM bank 0	E0_
	EM bank 1	E1_
	EM bank 2	E2_
	EM bank 3	E3_
	EM bank 4	E4_
	EM bank 5	E5_
	EM bank 6	E6_
	EM bank 7	E7_
	EM bank 8	E8_
	EM bank 9	E9_
	EM bank A	EA_
	EM bank B	EB_
	EM bank C	EC_
	EM bank D	ED_
	EM bank E	EE_
	EM bank F	EF_
	EM bank 10	E10_
	EM bank 11	E11_
	EM bank 12	E12_
	EM bank 13	E13_
	EM bank 14	E14_
	EM bank 15	E15_
	EM bank 16	E16_

Enhancing data memory	EM bank 17	E17_
	EM bank 18	E18_
Task flag	TK	TK

2.2.2.2. Mode is optional.

It uses it to specify the data kind when the timer, the counter, and the task flag were targeted.
(default: NORMAL)

NORMAL : The data type is treated as a bit.
This setting is disregarded, except for the timer, the counter, and the task flag.

CURRENT : The data type is treated as a value now.
It is possible to specify it only at the timer counter.

STATUS : The data type is treated as status.
It is possible to specify it only at the task flag.

2.2.2.3. VT is optional.

The data type that does Put/Get is specified.
The list of the data type that can be specified is shown below.
(default: Bit address specification = BIT, channel address specification = I2)

Table2-5Data type list that can be specified optional VT it

VT	Data type	Kind that can be used	Explanation
BIT	VT_UI1	Bit/channel	Data is converted into binary of 0/1 and write/reading. Val==0:0, Val≠0:1
BOOL	VT_BOOL	Bit/channel	Data is converted into binary of 0/1 and write/reading. Val==VARIANT_FALSE:0, Val= VARIANT_TRUE:1
BSTR	VT_BSTR	Channel	It is write/reading of BSTR as ASCII.
I1	VT_I1	Channel	It is write/reading as one byte data.
I2	VT_I2	Channel	It is write/reading as two byte data.
I4	VT_I4	Channel	It is write/reading as four byte data.
I8	VT_I8	Channel	It is write/reading as eight byte data.
UI1	VT_UI1	Channel	It is write/reading as one byte data.
UI2	VT_UI2	Channel	It is write/reading as two byte data.
UI4	VT_UI4	Channel	It is write/reading as four byte data.
UI8	VT_UI8	Channel	It is write/reading as eight byte data.

R4	VT_R4	Channel	It is write/reading as four byte data.
R8	VT_R8	Channel	It is write/reading as eight byte data.

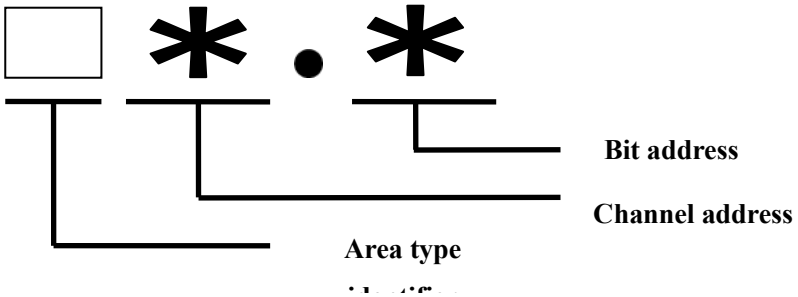
2.2.2.4. Wild-card specification in address

When the variable is registered, the address part can be specified in the wild-card by "*" (asterisk) to be without making the address in the access destination fixed, and dynamically revokable later.

The address is decided with ID of the variable at put/get. (Please refer to chapter 2.2.5)

The format to specify the address in the wild-card is shown below.

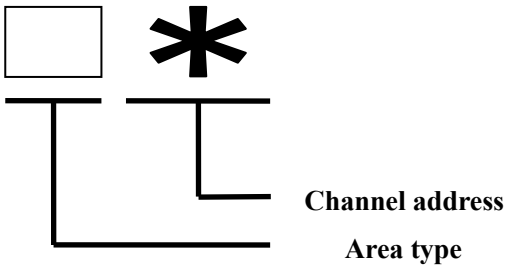
Bit addressing format



- Wild-card specification of address to access internal, supplementary relay (WR) bit

  W*.*

Channel (word) addressing format



- Wild-card specification of address to access maintenance relay (HR) channel

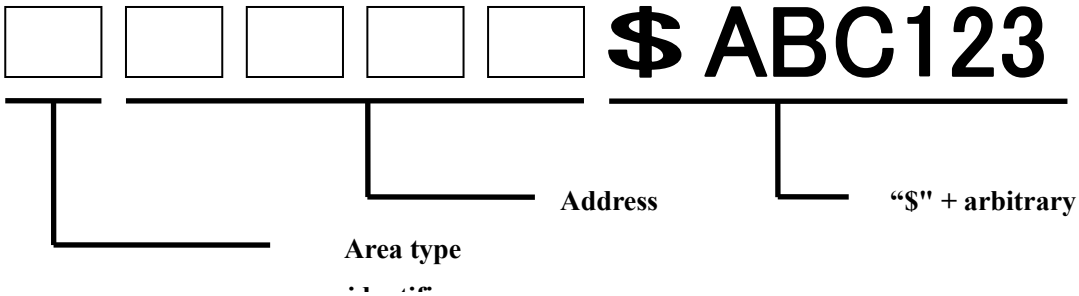
  H*

2.2.2.5. Arbitrary character string giving to variable identifier

An arbitrary character string can be given by placing “$" (dollar) to register two or more variables changing accessed neither area type nor address (usefulness to change only the option).

The format to give an arbitrary character string is shown below.

Bit/channel (word) address commonness and specification format



- To acquire the improvement flag in address 0x00A0 of counter (CNT) and the present value, two variables are registered.
  C0160$flag and C0160$val (After "$", it is not analyzed and can be registered as another variable.)

2.2.3. CaoVariable:put_Value property

After converting the value passed as an argument according to the option specification, send the FINS command (command code: 01 02) to write to the memory area specified by the variable name.

2.2.4. CaoVariable:get_Value property

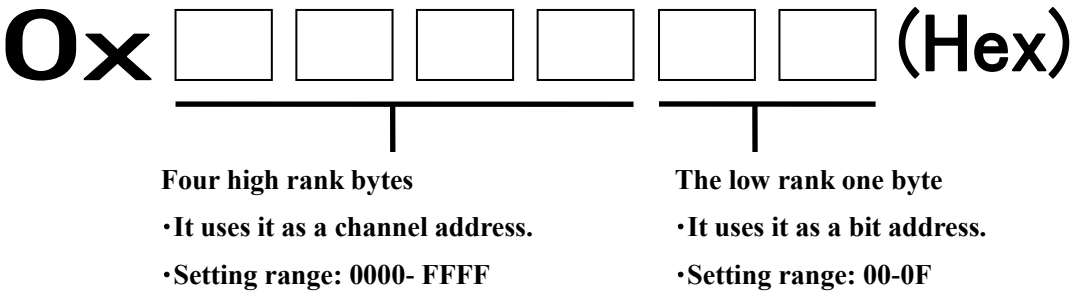
The FINS command (The command code: 01 01) read to become a size specified from the memory area specified by the variable identifier in the option is sent. The result of reading converts into the type specified in the option and is returned.

2.2.5. CaoVariable:put_ID property

ID of the variable is set.

The address by the variable identifier must be referred to for this ID when you specify the wild-card. (Please refer to cahpter 2.2.2.4)

It functions as simple ID when the wild-card is not specified. When the wild-card is specified, the ID value has the following meanings and the setting ranges.



※ To specify only the channel address, the low rank one byte is assumed to be 0x00 and ID is set.

2.2.6. CaoVariable:get_ID property

ID of the variable set now is acquired.

2.3. Variable list

2.3.1. CaoController class

Table2-6CaoController class user variable list

Variable identifier	Data type	Explanation	Attribute	
			get	put
Arbitrariness	Variable dependence	Area type/address of the accessed IO memory is specified. (Please refer to chapter 2.2.2	✓	✓

Table2-7CaoController class system variable list

Variable identifier	Data type	Explanation	Attribute	
			get	put
@MAKER_NAME	VT_BSTR	Returns the manufacturer name = "OMRON".	✓	-
@VERSION	VT_BSTR	Version information.	✓	-
@LAST_ERROR	VT_VARIANT | VT_ARRAY	Communication abnormality response details information immediately before. [0] VT_UI2: Generation command code [1] VT_BOOL: The relay is abnormal. [2] VT_BOOL: The main body stop is abnormal. [3] VT_BOOL: The main body continuance is abnormal. [4] VT_UI1: The main response code [5] VT_UI1: Sub response code [6] VT_UI1: Abnormal generation network address [7] VT_UI1: Abnormal generation node address	✓	-

2.4. Error code

In the CJ provider, the following and peculiar the error code is defined.

Error information can be acquired by reading system variable “@LAST_ERROR" when the error of 0x801001xx is returned.

About the ORiN2 commonness error, Please refer to the chapter of the error code of "ORiN2 SDK programming guide".

Table2-8Peculiar error code

Error name	Error number	Explanation
The host link response is abnormal. (format error)	0x80100000	It returns when the response packet of the host link communication is an abnormal format outside assumption.
Receive data lack	0x80100001	Receive data is missed, and when it is not possible to analyze it, it is returned.
Different command response	0x80100002	When a command response different from the transmitted FINS command is received, it is returned.
Relay abnormality generation	0x80100100	When abnormality is found in the communication route, and an abnormal relay responds, it is returned.
Error reply	0x80100101	When the error reply is received from the connection destination, it is returned.

Appendix A. Example of setting variable identifier

It introduces the example of setting some variable identifiers as follows.

Example 1） It accesses channel address 0x0010 and bit 12 of channel I/O(CIO).

16.12 or 0016.12 (Treat assuming that the same address is specified regardless of the presence of the digit of 0 burials.)

Example 2） It accesses channel address 0x00FF of enhancing data memory EM bank 4.

E4_255 or E4_0255

Example 3） Wild-card specification of address to access internal, supplementary relay (WR) bit

W*.*

ID is set to access address 0x0010 and bit 12.

ID=4108 (0x001012)

Example 4） Wild-card specification of address to access maintenance relay (HR) channel

H*

ID is set to access address 0x0010.

ID=4096 (0x001000)

Example 5） To acquire the improvement flag in address 0x00A0 of counter (CNT) and the present value, two variables are registered.

C0160$flag and C0160$val (It is not analyzed and it can register since "$" as another variable).

Appendix B. Allocation of memory in PLC (CJ series) when variable is written

When the memory writing is done by using the put_Value property, it exemplifies it to the memory in PLC (CJ series) ..how the mapping is done... (Explain by assumption to which all 0 clearing the memory in PLC (CJ series).)

Example 1） Bit5 and Bit7 in maintenance relay (HR) channel address 0x0010 are ON (1)ed.

- Two variables for the setting are registered.

  AddVariable

  < variable 1>

  Name : H0016.05

  Option : Mode=NORMAL, VT=BOOL, Elem=1, Array=FALSE

  < variable 2>

  Name : H0016.07

  Option : Mode=NORMAL, VT=BOOL, Elem=1, Array=FALSE

- The value is written in each variable.

  put_Value

  < variable 1>

  VT_BOOL : TRUE

  < variable 2>

  VT_BOOL : TRUE

- State in memory of PLC (CJ series)(CX-Programmer is used).

The value of H0016 is 00A0(0000000010100000).

- The variable that reads this channel address for the confirmation is registered.

  AddVariable

  < variable 3>

  Name : H0016

  Option : Mode=NORMAL, VT=UI2, Elem=1, Array=FALSE

- The value is read.

  get_Value

  < variable 3>

  VT_UI2 : 160 (0x00A0)

Example 2） Data is written at data memory (DM) channel address 0x0100 by one byte.

- One variable for the setting is registered.

  AddVariable

  < variable 4>

  Name : D0100

  Option : Mode=NORMAL, VT=UI1, Elem=1, Array=FALSE

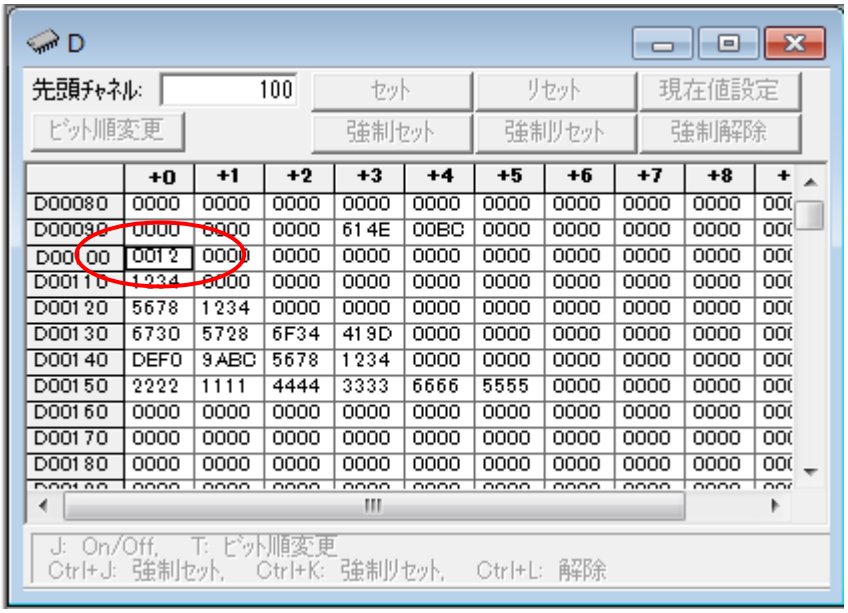
- The value is written.

  put_Value

  < variable 4>

  VT_UI1 :18 (0x12)

- State in memory of PLC (CJ series)(CX-Programmer is used).



The value of D0100 is 0012 (0000000000010010).

- This channel address is read for the confirmation.

  get_Value

  < variable 4>

  VT_UI1 :18 (0x12)

Example 3） Data is written at data memory (DM) channel address 0x0110 by two bytes.

- One variable for the setting is registered.

  AddVariable

  < variable 5>

  Name : D0110

  Option : Mode=NORMAL, VT=UI2, Elem=1, Array=FALSE

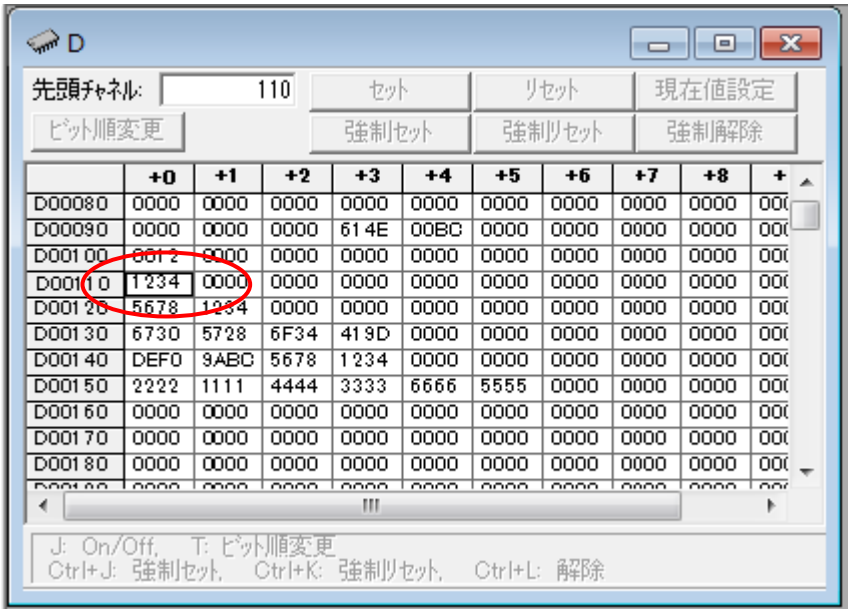
- The value is written.

  put_Value

  < variable 5>

  VT_UI2 :4660 (0x1234)

- State in memory of PLC (CJ series)(CX-Programmer is used).



	+0	+1	+2	+3	+4	+5	+6	+7	+8	+
D00080	0000	0000	0000	0000	0000	0000	0000	0000	0000	000
D00090	0000	0000	0000	614E	00BC	0000	0000	0000	0000	000
D00100	0012	0000	0000	0000	0000	0000	0000	0000	0000	000
D00110	1234	0000	0000	0000	0000	0000	0000	0000	0000	000
D00120	5678	1234	0000	0000	0000	0000	0000	0000	0000	000
D00130	6730	5728	6F34	419D	0000	0000	0000	0000	0000	000
D00140	DEF0	9ABC	5678	1234	0000	0000	0000	0000	0000	000
D00150	2222	1111	4444	3333	6666	5555	0000	0000	0000	000
D00160	0000	0000	0000	0000	0000	0000	0000	0000	0000	000
D00170	0000	0000	0000	0000	0000	0000	0000	0000	0000	000
D00180	0000	0000	0000	0000	0000	0000	0000	0000	0000	000

The value of D0110 is 1234 (0001001000110100).

- This channel address is read for the confirmation.

  get_Value

  < variable 5>

  VT_UI2 :4660 (0x1234)

Example 4） Data is written at data memory (DM) channel address 0x0120 by four bytes.

- One variable for the setting is registered.

  AddVariable

  < variable 6>

  Name : D0120

  Option : Mode=NORMAL, VT=I4, Elem=1, Array=FALSE

- The value is written.

  put_Value

  < variable 6>

  VT_I4 :305419896 (0x12345678)

- State in memory of PLC (CJ series)(CX-Programmer is used).

	+0	+1	+2	+3	+4	+5	+6	+7	+8
D00080	0000	0000	0000	0000	0000	0000	0000	0000	0000
D00090	0000	0000	0000	614E	00BC	0000	0000	0000	0000
D00100	0012	0000	0000	0000	0000	0000	0000	0000	0000
D00110	1234	0000	0000	0000	0000	0000	0000	0000	0000
D00120	5678	1234	0000	0000	0000	0000	0000	0000	0000
D00130	5730	5728	6F34	419D	0000	0000	0000	0000	0000
D00140	DEF0	9ABC	5678	1234	0000	0000	0000	0000	0000
D00150	2222	1111	4444	3333	6666	5555	0000	0000	0000
D00160	0000	0000	0000	0000	0000	0000	0000	0000	0000
D00170	0000	0000	0000	0000	0000	0000	0000	0000	0000
D00180	0000	0000	0000	0000	0000	0000	0000	0000	0000

The value of D0120 is 5678 (0101011001111000).

The value of D0121 is 1234 (0001001000110100).

- This channel address is read for the confirmation.

  get_Value

  < variable 6>

  VT_I4 :305419896 (0x12345678)

Example 5） Data is written at data memory (DM) channel address 0x0140 by eight bytes.

- One variable for the setting is registered.

  AddVariable

  < variable 7>

  Name : D0140

  Option : Mode=NORMAL, VT=I8, Elem=1, Array=FALSE

- The value is written.

  put_Value

  < variable 7>

  VT_I8 :1311768467463790320 (0x123456789ABCDEF0)

- State in memory of PLC (CJ series)(CX-Programmer is used).

The value of D0140 is DEF0(1101111011110000).

The value of D0141 is 9ABC(1001101010111100).

The value of D0142 is 5678 (0101011001111000).

The value of D0143 is 1234 (0001001000110100).

- This channel address is read for the confirmation.

  get_Value

  < variable 7>

  VT_I8 :1311768467463790320 (0x123456789ABCDEF0)

Example 6） Three data is written at data memory (DM) channel address 0x0150 by four bytes.

- One variable for the setting is registered.

  AddVariable

  < variable 8>

  Name : D0150

  Option : Mode=NORMAL, VT=I4, Elem=3, Array=TRUE

- The value is written.

  put_Value

  < variable 8>

  VT_ARRAY | VT_I4 :

  286335522 (0x11112222),

  858997828 (0x33334444),

  1431660134 (0x55556666)

- State in memory of PLC (CJ series)(CX-Programmer is used).

The value of D0150 is 2222 (1101111011110000).
The value of D0151 is 1111 (1001101010111100). } Element

The value of D0152 is 4444 (0101011001111000)
The value of D0153 is 3333 (0001001000110100). } Element

The value of D0154 is 6666 (0001001000110100)
The value of D0155 is 5555 (0001001000110100). } Element

- This channel address is read for the confirmation.

  get_Value

  < variable 8>

  VT_ARRAY | VT_I4 :

  286335522 (0x11112222),

  858997828 (0x33334444),

  1431660134 (0x55556666)

Appendix C. About write processing when VT = BIT, VT = BOOL, VT = UI1, VT = I1 to the channel address is specified

The behavior when VT = BIT, VT = BOOL, VT = UI1, VT = I1 is specified when specifying the channel address can now be selected from v1.0.1. If you create the file after v1.0.1, please specify "True" for this option.

Table 2-9 Option character strings of CaoWorkspace::AddController

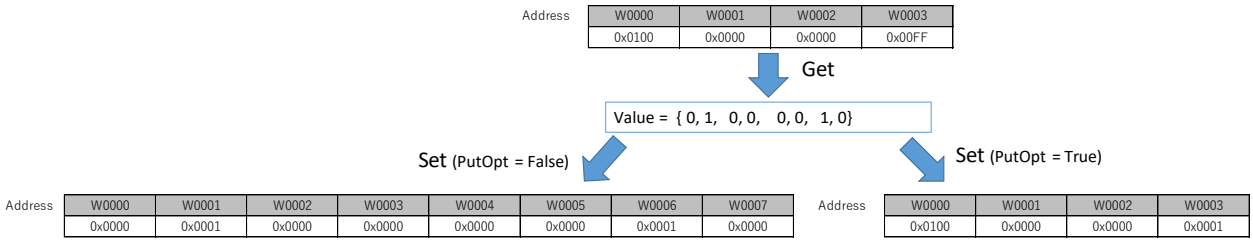
Option	Description
PutOpt[=< True or False >]	・ FALSE (works for versions less than v1.0.1) When writing data, the high-order byte is complemented with 0 and written. ・ TRUE (operation added in v1.0.1 or later) Data is stored and written in order from the lower byte. (Default: FALSE)

The following is an image of each operation.

<Variable>

Name : W0

Option : VT=BIT, Elem=8



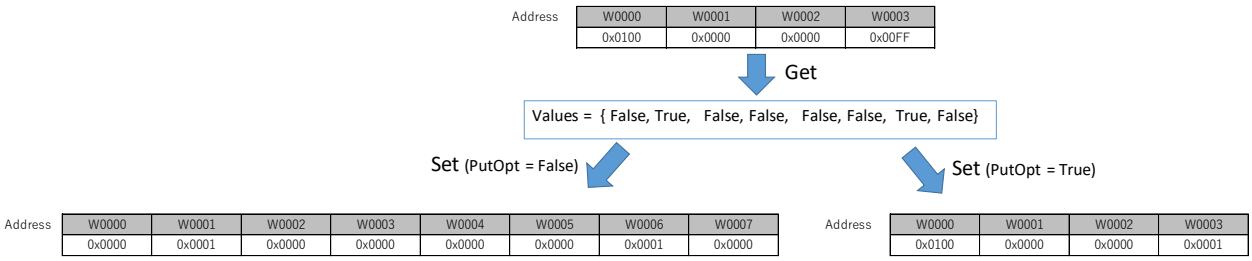
FALSE: When reading, 1-channel data is divided into high-order byte data and low-order byte data, and if each is other than 0, it is set to 1. At the time of writing, 1-bit data is treated as 1-channel data, and the upper byte is complemented with 0 and written to the device.

TRUE: When reading, 1-channel data is divided into high-order byte data and low-order byte data, and if each is other than 0, it is set to 1. At the time of writing, 2-bit data is treated as 1-channel data, and the data is stored in order from the lower byte and written to the device.

<Variable>

Name : W0

Option : VT=BOOL, Elem=8



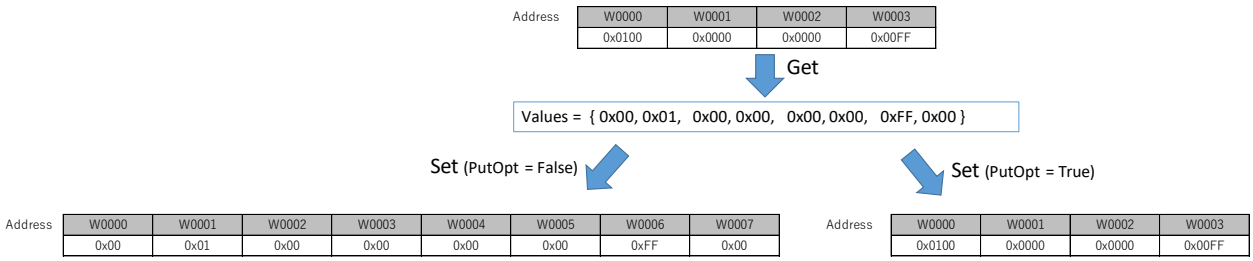
FALSE: When reading, 1-channel data is divided into high-order byte data and low-order byte data, and if each is other than 0, it is set to True. At the time of writing, 1 Boolean data is treated as 1 channel data, and the upper byte is complemented with 0 and written to the device.

TRUE: When reading, 1-channel data is divided into high-order byte data and low-order byte data, and if each is other than 0, it is set to True. At the time of writing, 2-Boolean data is treated as 1-channel data, and the data is stored in order from the lower byte and written to the device.

<Variable>

Name : W0

Option : VT=UI1 or I1, Elem=8



FALSE: At the time of reading, 1 channel data is divided into upper byte data and lower byte data and the value is acquired. At the time of writing, 1-byte data is treated as 1-channel data, and the upper byte is complemented with 0 and written to the device.

TRUE: When reading, 1 channel data is divided into upper byte data and lower byte data and the value is acquired. At the time of writing, 2-byte data is treated as 1-channel data, and the data is stored in order from the lower byte and written to the device.