KV provider

KEYENCE KV Series

Version 1.0.4

User’s guide

October 18, 2021

[Remarks]

[Revision history]

Version	Date	Content
1.0.0	2016-06-27	First edition
1.0.1	2017-03-06	Bug fixes: Default data format of Index register was incorrect.
1.0.2	2017-03-27	Bug fixes: Solve the problem that was unable to specify hexadecimal of Device number. Change the specified method of Address option from a decimal number to strings.
	2017-07-21	Error fixes: Ethernet/IP -> TCP/IP
1.0.3	2017-10-06	Bug fix.
1.0.4	2021-10-18	Fixed the processing when receiving data. Fixed option parsing process.

[Operation check completed model]

Model	Version	Remarks
KV-5500		Connected through the internal TCP/IP unit

Content

1. Introduction

This is a user’s guide of CAO provider that reads/writes data from/to a PLC (programmable controller) manufactured by KEYENCE. Hereafter, this provider (CaoProvKEYENCEKV.dll) is called KV provider.

Chapter 2 describes the outline of KV provider, detailed information of variables and commands..

The support status of communication commands and data strings that are implemented in KV provider differs depending on the PLC, which is the communication destination.

For detailed information about communicaton, please refer to "Host Link" of the KEYENCE PLC user's manual.

2. Outline of this provider

2.1. Outline

KV provider is a CAO provider that writes/reads data to/from KEYENCE PLC by using host-link communication through TCP/IP or serial connection.

The file format is DLL (Dynamic Link Library) and it is dynamically uploaded from the CAO engine. To use KV provider, you need to install ORiN2SDK or manually perform the registration based on the following information.

Table 2-1 KV provider

File name	CaoProvKEYENCEKV.dll
ProgID	CaoProv.KEYENCE.KV
Registration	regsvr32 CaoProvKEYENCEKV.dll
Deregistration	regsvr32 /u CaoProvKEYENCEKV.dll

2.2. Method and Property

2.2.1. CaoWorkspace::AddController method

KV provider establishes a connection by referring to the communication connection parameter at the timing of AddController.

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

bstrCtrlName	: [in] Controller name
bstrProvName	: [in] Provider name. Fixed to = " CaoProv.KEYENCE.KV"
bstrPcName	: [in] Computer name where this provider runs
bstrOption	: [in] Option strings

Table 2-2 shows the list of option strings.

Table 2-2 Option strings of CaoWorkspace::AddController

Option ([1])	Description
Conn=<connection parameter>	Required. Communication format and connection parameter. (See 2.2.1.1)
MyIP[=<own IP address>]	Own IP address. (for several NICs) (Default : not specified)
ConnTimeout [=<Timeout period>]	Timeout period for TCP connection. (millisecond) (Default : 3000)
Timeout[=<Timeout period>]	Specify a timeout period at the receiving and sending. (millisecond) (Default : 3000)
StationNumber[=<Station number>]	Station number specification for serial connection (not specified, 0 to 9) (See 2.2.1.2)

[1] Items enclosed with square brackets ("[]") are omittable. Underlined part shows the default value when the option is not specified..

2.2.1.1. Conn option

Specify the communication format and the connection parameters.

The following shows connection parameter strings for Conn option.

TCP/IP device

“Conn=ETH:<Dest IP Address>[:<Dest Port No>[:<Src IP Address>[:<Src Port No>]]]”

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

< Dest IP Address >	:	TCP/IP connection destination IP address Example : “127.0.0.1”, ”192.168.0.1”
<Dest Port No>	:	TCP/IP connection port number. Example : 8501, 5006, 5007
<Src IP Address>	:	Own IP address (For several NICs) ([2])
<Src Port No>	:	Own port number. (For several NICs)

Serial device

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

<COM Port>	:	COM port number Example : 1, 2, 3
<BaudRate>	:	Baud rate. Example :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 (, 19200bps, None, 8bits, 1bit)
(example 2)	“com:2:19200"	Communication port COM2 and 19200bps (, None, 8bits, 1bit)
(example 3)	“com:3:38400:N:8:2"	Communication port COM3, 38400bps, and None, 8bits, and 2bit

2.2.1.2. StationNumber option

Specifying the StationNumber option enables to specify the station number station number (0 to 9) at the serial connection. The station number will be not set if station number is not specified. (Station number option is not used in TCP/IP connection.)

Specified station number is assigned to the Start Communication command (CR) that is sent after the serial

[2] An error occurs if own IP address is specified both in the Conn option and in the MyIP option. When you specify an IP address, be sure to specify it in one option only.

connection is established. For details, refer to “Serial Communication Unit User’s Manual” of KEYENCE.

2.2.2. CaoController::AddVariable method

AddVariable method of CaoController class creates a variable object that reads/writes data from/to a PLC’s device.

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

<bstrVariableName>	:	[in] Variable name
<bstrOption>	:	[in] option strings

Table 2-3 shows option strings for this method

Table 2-3 Option strings for CaoController::AddVariable

Option ([3])	Description
DeviceType=<Device type>	Required. Enter the value of a device type with a code. (See 2.2.2.1)
Address[=<Device number>]	Enter the device number with decimal strings or hexadecimal strings. When access to Link relay, Virtual relay or Link register, it needs to be specified with hexadecimal strings; when access to any other devices, it needs to be specified with decimal strings. If enter incorrect strings, an error will be returned at the time of Put/Get variables. (Default : 0000)
VT[=<Data type>]	Enter the data type to Put/Get. (See 2.2.2.2)
Elem[=<Number of element>]	Enter the number of element to Put/Get with a decimal number. (Default : 1)
Array[=TRUE/FALSE]	This option specifies whether the data to Put/Get is handled as an array or not when the data to Put/Get is one element. (Default : FALSE)

[3] Items enclosed with square brackets (“[]”) are omittable. Underlined part shows the default value when the option is not specified..

2.2.2.1. DeviceType option

An access-target device type is specified by a string code.

Table 2-4 shows codes for the available device types. ([4])

Table 2-4 Code list for each device

Device type	Code	Default data type
Relay	R	Bit
Link relay	B	Bit
Internal auxiliary relay	MR	Bit
Latch relay	LR	Bit
Control relay	CR	Bit
Virtual relay	VB	Bit
Data memory	DM	2 bytes
Expanded data memory	EM	2 bytes
File register	FM	2 bytes
	ZF	2 bytes
Link register	W	2 bytes
Temporary data memory	TM	2 bytes
Index register	Z	2 bytes
Timer (Current value)	TC	4 bytes
Timer (Set value)	TS	4 bytes
Counter (Current value)	CC	4 bytes
Counter (Set value)	CS	4 bytes
Digital trimmer	AT	4 bytes
Control memory	CM	2 bytes
Virtual memory	VM	2 bytes

[Example]

Relay	:	DeviceType=R
Data memory	:	DeviceType=DM

[4] Accessible device differs depending on the PLC model used. For information about supported device models, refer to the user’s manual of each model.

2.2.2.2. VT option

VT option specifies the data type to Put/Get.

To specify a VT option, use VT strings or corresponding VARTYPE value (decimal number).

If VT option is not specified, default data format for each device will be used.

Table 2-5 lists available data types.

Table 2-5 Data type available for VT option

VT	Data type	Description
BIT ([5])	VT_UI1	Convert data to binary data (0 or 1) and then read/write it.
BOOL	VT_BOOL	Convert data to binary data (0 or 1) and then read/write it.
BSTR	VT_BSTR	Write/Read data as ASCII format by the byte count of Elem
I1	VT_I1	Write/Read data as one byte data (signed)
I2	VT_I2	Write/Read data as 2-byte data (signed)
I4	VT_I4	Write/Read data as 4-byte data (signed)
I8	VT_I8	Write/Read data as 8-byte data (signed)
UI1	VT_UI1	Write/Read data as one byte data (unsigned)
UI2	VT_UI2	Write/Read data as 2-byte data (unsigned)
UI4	VT_UI4	Write/Read data as 4-byte data (unsigned)
UI8	VT_UI8	Write/Read data as 8-byte data (unsigned)
R4	VT_R4	Write/Read data as 4-byte data (floating point)
R8	VT_R8	Write/Read data as 8-byte data (double precision floating point)

Table 2-6 Data type when VT option is not specified

Default data format	VT	Data type
Bit	BIT	VT_UI1
2-byte	UI2	VT_UI2
4-byte	UI4	VT_UI4

[5] VT=BIT can be specified only for the relay-type devices (DeviceType=R, B, MR, LR, CR, VB).

2.2.3. CaoVariable:put_Value property

According to the option specification, convert values that are passed by arguments, and then send a command (Write Continuous Data : WRS) that executes writing into the access-target device.

2.2.4. CaoVariable:get_Value property

Send a command (Read Continuous Data : RDS) that reads data from the access-target device by the specified number, and then convert the result to the option-specified data type and return.

2.2.5. CaoControllere::Execute

Execute method of CaoController class executes a command.

Syntax [<vntRet:VARIANT> =] Execute(<bstrCmd:BSTR > [,<vntParam:VARIANT>])

< bstrCmd>	:	[in] Command name
< vntParam>	:	[in] Parameter

2.3. Variable list

2.3.1. CaoController class

Table 2-7 CaoController class System variable list

Variable name	Data type	Description	Attribute	
			get	put
@MAKER_NAME	VT_BSTR	KEYENCE is returned.	✓	–
@VERSION	VT_BSTR	Return the provider version information.	✓	–
@PLC_MODEL	VT_I4	Send “[Mode Query] command (?K) ” and return the result.	✓	–

@PLC_MODE	get: VT_I4
	put: VT_BOOL
		Send “get:[Check Operation Mode] command (?M)” and return the result.
		Send “put:[Change Mode] command (Mn)”.
		TRUE=RUN mode
		FALSE= PROGRAM mode
			✓	✓

@PLC_ERROR	get: VT_I4 put: -	Send “get:[Check Error Number] command (?E)” and return the result. Send “put:[Error Clear] command (ER)”.	✓	✓

Table 2-8 CaoController class User variable list

Variable name	Data type	Description	Attribute	
			get	put
(any name)	Variable type-dependent	Access to a device in PLC.	✓	✓

2.4. Command list

2.4.1. CaoController class

Table 2-9 CaoController class command list

Command name	Data type	Description
ExecuteCommand	VT_BSTR	Send a user command.

2.4.1.1. ExecuteCommand

In the end of a string handed as a parameter, add "(CR)" and then send it as a command. The result is returned as a string (after deleting "CR" and "LF" in the end of string).

The execution of Execute method is returned as normal even if the response shows an error. In this case, the result of Execute method will be the format of strings, such as "E0" or "E1".

2.5. Error code

In KV provider, the following original error codes are defined.

For about ORiN2 common errors, refer to the error code section of ORiN2 Programming guide.

Table 2-10 Original error code

Error name	Error number	Description
Received data format error	0x80100000	This error is returned when received data has an error and is not analyzed properly.
Received data incomplete	0x80100001	This error is returned when received data is incomplete, such as the data size is less than the minimum data size.
Error response	0x801001XX	This error is returned when an error is returned as a command response result. The number of error code ([6]) is inserted in "XX". Example) E0 → 0x80100100 E1 → 0x80100101

[6] For details about error codes, refer to the user's manual of each model.

Appendix A. Data access for each device

Data access is different according to the data type (VT option) and the number of element (Elem option),which are specified by variable option.

■ BIT-device of relay type (R, B, MR, LR, CR, VB)

- BIT, BOOL, I1, UI1

  Access as a BIT (0/1)

  Number of access data = Elem option.
- I2, UI2

  Access as a "Decimal number, 16-bit unsigned (.U)"

  Number of access data = Elem option.
- I4, UI4, R4

  Access as a "Decimal number, 32-bit unsigned (.D)"

  Number of access data = Elem option.
- I8, UI8, R8

  Access as a "Decimal number, 32-bit unsigned (.D)"

  Number of access data =Elem option×2.
- BSTR

  Access as a "Decimal number, 16-bit unsigned (.U)"

  Treat ASCII code of string (one byte for each) as a numeric value.

  Treat the ASCII code as the number of byte of Elem option, and then if the data is odd byte at the writing, fill it with zero.

  Number of access data = Elem option÷2.

■ 2-byte device of data memory/register type (DM, EM, FM, ZF, W, TM, Z, CM, VM)

- BOOL, I1, UI1

  Access as a "Decimal number, 16-bit unsigned (.U)"

  Access the lower 8 bits of a device.

  When data writing, the upper 8 bits are zero-filled

  Number of access data = Elem option.

- I2, UI2

  Access as a "Decimal number, 16-bit unsigned (.U)"

  Number of access data = Elem option.

- I4, UI4, R4

  Access as a "Decimal number, 32-bit unsigned (.D)"

  Number of access data = Elem option.

- I8, UI8, R8

  Access as a "Decimal number, 32-bit unsigned (.D)"

  Number of access data = Elem option × 2.

- BSTR

  Access as a "Decimal number, 16-bit unsigned (.U)"

  Treat ASCII code of string (one byte for each) as a numeric value.

  Treat the ASCII code as the number of byte of Elem option, and then if the data is odd byte at the writing, fill it with zero

  Number of access data = Elem option ÷ 2.

■ 4-byte device of timer/counter type (TC, TS, CC, CS, AT)

- BOOL, I1, UI1

  Access as a "Decimal number, 16-bit unsigned (.U)"

  Access the lower 8 bits of a device.

  When data writing, the upper 24 bits are zero-filled

  Number of access data = Elem option.

- I2, UI2

  Access as a "Decimal number, 16-bit unsigned (.U)"

  Access the lower 16 bits of a device.

  When data writing, the upper 16 bits are zero-filled

  Number of access data = Elem option.

- I4, UI4, R4

  Access as a "Decimal number, 32-bit unsigned (.D)"

  Number of access data = Elem option.

- I8, UI8, R8

  Access as a "Decimal number 32-bit unsigned (.D)"

  Number of access data = Elem option×2.

- BSTR

  Access as a "Decimal number, 32-bit unsigned (.D)"

  Treat ASCII code of string (one byte for each) as a numeric value.

  Treat the ASCII code as the number of byte of Elem option, and then if the data is less than the multiple of four-bytes at the writing, fill it with zero.

  Number of access data = Elem option÷4.