KVCOM provider
KEYENCE programmable controller
KV series provider

Version 1.0.0

User's guide

Oct 04, 2016

[Remarks]

[Revision history]

Version	Date	Content
1.0.0	2016-10-04	First edition

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 (CaoProvKEYENCEKVCOM.dll) is called KVCOM provider

KVCOM provider uses KV COM + Library in order to communicate with KEYENCE PLC, so you need to install "KVCOM + Library", which is the library from Keyence, separately. For information about the PLC models supported by KVCOM provider and the connection method, please refer to the KVCOM + Library's user's manual.

This provider has been developed for the KV COM + Library of the following version. If you use different version, this provider may not be perform properly, so please install this version's software.

KV COM+ Library Ver.1.3

1.1. Computer environment setup

Before start using KVCOM provider, configure your computer environment as Table 1-1 shows.

Table 1-1 Computer environment setup

Process	Description
(1) Install KV COM + Library	See "1-3 Software installation" of "KV COM+Library User's manual "
(2) Set the environment variable (Path)	See "5-1 DLL installation" of "KV COM + Library User's manual".
(3) Register KVCOM provider	See "2 Outline of this provider" in this document.

2. Outline of this provider

KVCOM provider is a CAO provider that reads/writes data from/to KEYENCE PLC by using KVCOM + Library. The file format is DLL (Dynamic Link Library) and it is dynamically uploaded from the CAO engine. To use this provider, you need to execute Step (1) and (2) in 1.1 Computer environment setup, and then manually install ORiN2SDK or perform the registration based on the following information.

Table 2-1 KVCOM provider

File name	CaoProvKEYENCEKVCOM.dll
ProgID	CaoProv.KEYENCE.KVCOM
Registration	regsvr32 CaoProvKEYENCEKVCOM.dll
Deregistration	regsvr32 /u CaoProvKEYENCEKVCOM.dll

3. Reference

3.1. PLC control

3.1.1. CaoWorkspace::AddController method

This method connects to a PLC to create a controller object that reads/writes data from/to the PLC. The connection method to the PLC is specified by option strings.

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

Return value	<objRet>	:	[out] Controller object
Argument	< bstrCtrlName>	:	[in] Controller name (any name)
	< bstrProvName >	:	[in] Provider name. Fixed to= "CaoProv.KEYENCE.KVCOM"
	< bstrPCName >	:	[in] Computer name where provider runs
	<bstrOption>	:	[in] option strings (See Table 3-1)

Table 3-1 shows the list of option strings.

Table 3-1 Option strings of CaoWorkspace::AddController

Option	Description
PlcId=<PLCID>	This value shows the model of PLC to connect. With a decimal number, enter a DBPlcid (enumerated type data) that has been defined in DBPlcDef.h. DBPlcDef.h is stored in "dlluser¥include" in the install folder of KVCOM + Library.
DestName=<Connection parameter>	Character string that shows the connection method to PLC. For details, refer to "lpszDestName" described in [5-2 DLL function]-[3 DB Connect] of KVCOM+ Library user's manual. If connection parameter itself includes a comma, enclose the connection parameter with inequality signs (See Example 2). For how to create connection parameters with GUI, see Appendix A.

Example 1 Connect to KV5500 from a computer through an USB.

```
Dim caoEng as CaoEngine
Dim caoCtrl as CaoController

Set caoEng = New caoEngine
Set caoCtrl = caoEng.Workspaces(0).AddController("KV5500", "CaoProv.KEYENCE.KVCOM", "",
"PlcId=4611, DestName=USB")
```

Example 2 Connect to KV5500 from a computer through an USB, and then establish further connection from the KV5500 to other device through Ethernet/IP network.

```
Dim caoEng as CaoEngine
Dim caoCtrl as CaoController

Set caoEng = New caoEngine
Set caoCtrl = caoEng.Workspaces(0).AddController("KV5500", "CaoProv.KEYENCE.KVCOM", "",
"PlcId=4611, DestName=<USB;Via,13,2,4,0,4,1,0,1,0,0,192.168.0.10,53>")
```

3.1.2. CaoController::AddVariable method

AddVariable method of CaoController class creates a variable object that reads/writes data from/to a PLC's device. You can read/write data by reading/writing from/to a Value property of the Variable object created.

Table 3-2 shows option strings for this method. With this options, you can specify device type, device number, variable type, number of data for a device in the access-target PLC.

Syntax <objRet:CaoVariable> = AddVariable(<bstrVariableName:VT_BSTR>
,<bstrOption:VT_BSTR>)

Return value	<objRet>	:	[out] Variable object
Argument	<bstrVariableName>	:	[in] Variable name (any name)
	<bstrOption>	:	[in] option strings (See Table 3-2)

Example

```
'Create CaoVariable object that accesses DM00000 to DM00015.
Set caoVar = caoCtrl.AddVariable("DM", "DeviceKind=18, DeviceNo=0, Elem=4")

'Execute value reading
vVal = caoVar.Value
'Execute value writing
caoVar.Value = vVal
```

Table 3-2 shows the list of option string items.

Table 3-2 Option strings of CaoController::AddVariable

Option	Description
DeviceKind=<device type >	Required. Enter the value of a device type by decimal or hexadecimal number. For details about values, refer to [4-2 Property, Method, Event] - [DBCommManager object] - [ReadDevice] of KV COM+Library User's manual.
DeviceNo=<device number>	Required. Enter the top of the device number to access by decimal or hexadecimal number. For details, refer to [4-2 Property, Method, Event] – [DBCommManager object] – [ReadDevice] of KV COM+Library User's manual.
VT=<variable type>	Omittable. Enter the data type to Put/Get. For details, refer to 3.1.2.1. When there is no entry, BIT is selected if the device type is bit, and I2 is selected if the device type is word.
Elem=<number of element>	Omittable. Enter the number of data to Put/Get by decimal or hexadecimal number. If the variable type is BSTR (VT=BSTR), you can assign up to 256 bytes for strings (including Null terminator). When there is no entry, "1" is selected if variable type is other than BSTR, and 256 is selected if VT=BSTR.
Array=<True or False>	Omittable. 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)

You can specify the data size to read/write at one time by specifying VT option and Elem option. If the data size exceeds the data size of device type specified by DeviceKind option, the reading/writing will be done for the devices whose device numbers are the next ones specified by DeviceNo option. Data read/written from/in devices shall be set with the little-endian (set from the lower bit).

If the number of devices read/written at one time exceeds 8192, an error occurs at the data reading/writing.

3.1.2.1. VT option

VT option specifies the data type to Put/Get with Value property. Table 3-3 lists available data types.

Table 3-3 Data types available for VT option

VT strings	Data type	Description
BIT	VT_UI1	Write/Read data as one-bit data (0 or 1).
BOOL	VT_BOOL	Write/Read data as one-bit data (True or False).
BSTR	VT_BSTR	Convert BSTR data to ASCII strings (with Null terminator) and then write/read it. This data type is available only when the device type is word.
I1	VT_I1	Write/Read data as one byte data
I2	VT_I2	Write/Read data as 2-byte data
I4	VT_I4	Write/Read data as 4-byte data
I8	VT_I8	Write/Read data as 8-byte data
UI1	VT_UI1	Write/Read data as one byte data
UI2	VT_UI2	Write/Read data as 2-byte data
UI4	VT_UI4	Write/Read data as 4-byte data
UI8	VT_UI8	Write/Read data as 8-byte data
R4	VT_R4	Write/Read data as 4-byte data
R8	VT_R8	Write/Read data as 8-byte data

3.1.3. CaoVariable:put_Value property

This property converts the handed value according to the option specifications of AddVariable, and then writes it in a device. If the handed value cannot be converted to the data type specified by option, or if the handed value is less than the number of element specified by option, an error occurs.

3.1.4. CaoVariable:get_Value property

This property reads data from a device and converts it to the data type specified by AddVariable option, and then return it.

3.1.5. CaoController::Execute("ReadComment") method

This method reads a comment from a device in PLC.

Syntax <bstrRet:VT_VBSTR> = ReadComment(<iDeviceKind:VT_I4>, <iDeviceNo:VT_I4> [, <iCommnetNo:VT_I4>])

Return value	<bstrRet>	:	[out] Device comment obtained
Argument	<iDeviceKind>	:	[in] Device type For details, refer to [4-2 Property, Method, Event] - [DBCommManager object] - [ReadDevice] of KV COM+Library User's manual.
	< iDeviceNo>	:	[in] Device number
	< iCommnetNo>	:	[in] Comment number (Default : 1) If your controller is KV-7500/7300, to specify a comment number, use any number between 1 through 8.

Example

```
bstrRet = caoCtrl.Execute("ReadComment", Array(18, 10000))
```

3.1.6. CaoController::Execute("ClearError") method

This method clears an error occurred in PLC.

Syntax ClearError()

Return value		:	none
Argument		:	none

Example

```
caoCtrl.Execute "ClearError"
```

3.1.7. CaoController::Execute("SetSystemTime") method

This method sets the real-time clock in PLC.

Syntax SetSystemTime(<iTime:VT_ARRAY | VT_I4>])

Return value : none

Argument <iTime> : [in] Array of time information to set.

[0] year

[1] month

[2] date

[3] hour

[4] minute

[5] second

Example

```
caoCtrl.Execute "SetSystemTime", Array(2016, 10, 4, 12, 0, 0)
```

3.1.8. CaoController::Execute("GetAlarmLog") method

This method obtains the alarm log from the PLC.

Syntax <vntRet:VT_VARAINT | VT_ARRAY> = GetAlarmLog()

Return value <vntRet> : [out] Obtained alarm log. One log data array consists of the following elements. This method returns arrays by the number of obtained alarm logs.
If no alarm log is received, VT_EMPTY will return.
[0] Relay number (VT_I4)
[1] True: rising, False: falling (VT_BOOL)
[2] Time (VT_ARRAY |VT_I4)
[0] year
[1] month
[2] date
[3] hour
[4] minute
[5] second

Argument : none

Example

```
vntRet = caoCtrl.Execute("GetAlarmLog")
```

3.1.9. CaoController::get_VariableNames property

This property returns the system variable name list specified by CaoController::AddVariable. For details about system variable name, refer to Table 3-4 and Table 3-5.

Syntax <bstrRet:VT_BSTR | VT_ARRAY> = VariableNames()

Return value <bstrRet> : [out] System variable name list

Argument : none

Example

```
bstrRet = caoCtrl.VariableNames
```

3.1.10. System variable list

Table 3-4 CaoController class system variable list

Variable name	Data type	Description	Attribute	
			get	put
@TYPE	VT_BSTR	Return the model name of PLC connected. For details, refer to [5-2 DLL function]-[DBQueryType] of KV COM+Library User's manual	✓	—
@MODE	VT_I4	Operation mode of PLC connected. 1:RUN mode, 2:PROG mode	✓	✓
@ERROR_NUMBER	VT_UI1	Error number that is currently occurred in PLC. If there is no error, "0" is returned. For details, refer to User's manual of your PLC.	✓	—
@LIBRARY_ERROR	VT_I4	If each method, property or any variable-call causes an KVCOM+Library error (HRESULT =0x80100000), this variable stores the KVCOM+Library error number. For details about error number, refer to [Appendix 5 Error message list]-[DLL] of KV COM+ Library User's manual. To clear this variable's entry, enter "0".	✓	✓

3.2. Memory card access

3.2.1. CaoController::Execute("MemoryCardFileCopyTo") method

This method copies a file from a computer to a memory card inserted in the PLC.

The maximum file size to be copied is 256MB.

Syntax MemoryCardFileCopyTo(<bstrPathFrom:VT_BSTR>, <bstrPathTo:VT_BSTR>
[,<bstrOption:VT_BSTR>])

Return value		:	none
Argument	< bstrPathFrom>	:	[in] Specify the file path of the copy source on a computer.
	< bstrPathTo>	:	[in] Specify the file path of the copy destination on a memory card.
	< bstrOption>	:	[in] If you specify "OverWrite" in this option, when the same file name exists in the copy destination, the destination's file is overwritten. If you do not specify "OverWrite", an error occurs when the same file name exists in the copy destination.

Example

```
caoCtrl.Execute "MemoryCardFileCopyTo", Array("C:¥Temp¥data1.txt", "data1.txt", "OverWrite")
```

3.2.2. CaoController::Execute("MemoryCardFileCopyFrom") method

This method copies a file from a memory card inserted in PLC to a computer.

The maximum file size to be copied is 256MB.

Syntax MemoryCardFileCopyFrom(<bstrPathFrom:VT_BSTR>, <bstrPathTo:VT_BSTR>
[,<bstrOption:VT_BSTR>])

Return value		:	none
Argument	< bstrPathFrom>	:	[in] Specify the file path of the copy source on a memory card.
	< bstrPathTo>	:	[in] Specify the file path of the copy destination on a computer.
	< bstrOption>	:	[in] If you specify "OverWrite" in this option, when the same file name exists in the copy destination, the destination's file is overwritten. If you do not specify "OverWrite", an error occurs when the same file name exists in the copy destination.

Example

```
caoCtrl.Execute "MemoryCardFileCopyFrom", Array("data1.txt", "C:¥Temp¥data1.txt",
"OverWrite")
```

3.2.3. CaoController::AddFile method

This method creates a CaoFile object that accesses a file or directory in a memory card inserted in PLC.

Syntax <objRet:CaoFile> = AddFile(<bstrName:VT_BSTR>[, <bstrOption:VT_BSTR>])

Return value	<objRet>	:	none
Argument	< bstrName>	:	[in] Specify a file name or path of directory name on a memory card to access. (Specify (IP address)/MMC or later.) For directory, add "¥" sign at the end.
	< bstrOption>	:	[in] If you specify "@Create=2", a directory specified in bstrName is created.

Example

```
Set caoFl = caoCtrl.AddVariable("data1.txt")
```

3.2.4. CaoFile::AddFile method

This method creates a File object similar to 3.2.3. For bstrName, you can only specify a file/directory that locates in the CaoFile object-corresponding directory. If the executed CaoFile object does not correspond with a directory, an error occurs.

3.2.5. CaoController::get_FileNames property

This property obtains the list of file/directory names in a memory card inserted in PLC.

Syntax <bstrRet:VT_BSTR | VT_ARRAY> = FileNames([<bstrOption:VT_BSTR>])

Return value	<bstrRet>	:	[out] Array of file name/directory name obtained. The end of directory name has a "¥" symbol.
Argument	< bstrOption>	:	[in] Specify the directory path of the list to obtain. If this entry is omitted, the list of the root directory is obtained.

Example

```
bstrRet = caoCtrl.FileNames
```

3.2.6. CaoFile::get_FileNames property

This property obtains the list of directory that corresponds to CaoFile object similar to 3.2.5. You cannot specify a directory with bstrOption. If the executed CaoFile object does not correspond with a directory, an error occurs.

3.2.7. CaoFile::get_Path property

This property obtains a file or directory path that corresponds with CaoFile object.

Syntax <bstrRet:VT_BSTR> = Path

Return value	<bstrRet>	:	[out] File or directory path that corresponds with CaoFile object.
Argument		:	none

Example

```
bstrRet = caoFl.Path
```

3.2.8. CaoFile::Delete method

This method deletes a file or directory that corresponds with CaoFile object.

Syntax Delete([<bstrOption:VT_BSTR])

Return value	:	none.
Argument	:	[in] If the target is a file, specifying "Force" will delete the read-only attribute file. If the target is a directory, specifying "Force" allows to delete the directory even if it contains any data. An error occurs if you delete read-only file or if you delete any directory which contains any data without option.

Example

```
caoFl.Delete "Force"
```

3.2.9. CaoFile::get_VariableNames property

This property returns the system variable name list that can be specified by CaoFile::AddVariable. For details about system variable names, refer to Table 3-6.

Syntax <bstrRet:VT_BSTR | VT_ARRAY> = VariableNames()

Return value	<bstrRet>	:	[out] System variable name list
Argument		:	none

Example

```
bstrRet = caoFl.VariableNames
```

3.2.10. System variable list

Table 3-5 CaoController class system variable list

Variable name	Data type	Description	Attribute	
			get	put
@MEMORYCARD_SPACE	VT_I4 | VT_ARRAY	This returns the total size and empty space of the memory card by Byte-unit. Divide 64-bit integer value into 32-bit for each, and then return them with the array shown below. [0] Total size. Upper 32-bit [1] Total size. Lower 32-bit [2] Empty space. Upper 32-bit [3] Empty space. Lower 32-bit	✓	—

Table 3-6 CaoFile class system variable list

Variable name	Data type	Description	Attribute	
			get	put
@MEMORYCARD_ FILE_STATE	VT_VARIANT | VT_ARRAY	This returns the status of a file or directory in the following array format. [0] Last access date (VT_ARRAY | VT_I4) [0] year [1] month [2] date [3] hour [4] minute [5] second [1] Creation date (VT_ARRAY | VT_I4) [0] year [1] month [2] date [3] hour [4] minute [5] second [2] Last modified date (VT_ARRAY | VT_I4) [0] year [1] month [2] date [3] hour [4] minute [5] second [3] File size [Byte] (VT_I4) [4] Attribute (VT_I4 0: file 1: directory)	✓	—

3.3. Error code

In KVCOM provider, original error code described in Table 3-7 are defined.

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

Table 3-7 Original error code

Error name	Error number	Description
KV COM + Library error	0x80100000	This error code is returned when an error occurs at the API-calling of KVCOM+Library. If this error occurs, please obtain the detailed error information by using @LIBRARY_ERROR, which is CaoController class system variable.
Character string code conversion error	0x80100001	This error code is returned if any character strings are not converted at the character string conversion (ASCII<>UNICODE).

Appendix A. How to Create a Connection Parameter

This section describes how to create a connection parameter "DestName" that is necessary for AddController with GUIs.

Step 1 Start [Microsoft Excel].

Step 2 Press [Alt] +[F11].

Step 3 On the [Microsoft Visual Basic for Applications] window, from the [Menu], select [Insert (I)] , select [User form (U)].

Step 4 Select [Tool (T)], select [Additional Controls(A)].

Step 5 On the [Additional Controls] window, from the [Available Controls] list, select [DBComm Manager Class(Library)] check box, and then click [OK] button.

Step 6 On the [Toolbox] window, select [DBComm Manager], and then paste it to [UserForm1].

Step 7 On the [Properties] window, select a property page, and then, click [...] button.

Step 8 Click [Settings] on the [Property page] window.

Step 9 On the [Communication settings] window, select a communication method, and then click [OK].

Step 10 Use character strings of the text box on the [Property page] as a connection parameter.