MxComponent4 provider

Mitsubishi Electric MxComponent Compatible Devices

Version 1.0.0

User's Guide

July 6, 2022

[Remark]

[Revision History]

Version	Date	Content
1.0.0	2022-07-06	First edition

[Compatible devices]

Model	Version	Notes

Table of Contents

1. Introduction

This manual is a user's guide for MxCompo4 providers that use MELCO's MxComponent ACTcontrollers to communicate with each other.

MxCompo4 providers can communicate over all communication paths supported by MxComponent by using MX Component Version 4 ACTcontrollers "ActUtlType".

For more information about ActUtlType, see MxComponent Version 4 documentation.

Note :

To use MxCompo4 providers, you must install MxComponent Version 4.

It can only be connected to a programmable controller that supports "ActUtlType".

2. Provider Overview

2.1. Overview

MxCompo4 providers execute the corresponding ActUtlType control methods when they execute CAO API. The correspondence between CAO API and MELSEC API is shown below.

Table 2-1 Correspondence between CAO API and MELSEC API

CAO API		ActUtlType Method
CaoController::FinalConstruct		Put_ActLogicalStationNumber Open
CaoController::FinalRelease		Close
CaoTask::Start		SetCpuStatus
CaoTask::Stop		SetCpuStatus
CaoVariable::put_Value	(User variable)	SetDevice (1-point write) WriteDeviceBlock
	(@BUFFER)	WriteBuffer
CaoVariable::get_Value	(User variable)	GetDevice (1-point read) ReadDeviceBlock
	(@BUFFER)	ReadBuffer
	(@CLOCK)	GetClockData
	(@CPUTYPE)	GetCpuType

Also, if the method of ActUtlType executed through CAO API results in an error, you can use the "GetLastError" command in Controller::Execute() to obtain the error number of MxComponent.

Describes the provider overview.

Table 2-2 MxCompo4 provider

File name	CaoProvMELCOMxCompo4.dll
ProgID	CaoProv.MELCO.MxCompo4
Registry registration	Regsvr32 CaoProvMELCOMxCompo4.dll
Deletion of Registry Registration	Regsvr32 /u CaoProvMELCOMxCompo4.dll

2.2. Method properties

2.2.1. CaoWorkspace::AddController method

MxCompo4 providers use ActUtlType::Open() to communicate when creating Controller. In this case, specify the logical station number of the communication destination.

Format

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

BstrCtrlName	:	[in]	Controller name
BstrProvName	:	[in]	Provider name. Fixed value =" CaoProv.MELCO.MxCompo4"
BstrPcName	:	[in]	Provider's running machine name
BstrOption	:	[in]	Option string = "LogicID = <logical station number>"

The following is a list of option strings:

Table 2-3 CaoWorkspace::AddController Option-String

Option	Meaning
LogicID = <logical station number>	Logical station number to specify communication destination (required) The logical station number to be specified must be set in advance in the communication configuration utility of MxComponent. See MxComponent documentation for more information about the Configuration Utility.

2.2.2. CaoController::AddTask method

Creates a Task that executes and stops the CPU status of the communication destination.

The task name can be any name.

Format

AddTask(<bstrName:BSTR >[,<bstrOption:BSTR >])

BstrName	:	[in]	Title of Task
BstrOption	:	[in]	Option string (unused)

2.2.3. CaoController::AddVariable method

Creates a Variable that accesses the communication destination device and obtains and sets thevalue.

Format

AddVariable(<bstrName:BSTR >[,<bstrOption:BSTR >])

BstrName	:	[in]	Variable Name
BstrOption	:	[in]	Option string

In this method, you can specify the following in the option string:

Table 2-4 CaoController::AddVariable Option-String

Option	Meaning
Size = <word size>	Set the read/write size in word units by user variable and "@BUFFER". (Default: 0)[1] However, only when Size=0 is specified, 1-device point is read/written.
VARTYPE = <datatype> (VT=<data type>)	PLC side data type (default: VT_I4)
DATATYPE = <datatype> (DT=<data type>)	Data type to return to the CAO client If this option is specified at the same time as Size option, this option takes precedence.

Table 2-5 Data types that can be specified with VARTYPE,DATATYPE option

Data type	Notation
Bit data	BOOL BIT
1-byte integer (signed)	I1 CHAR
1-byte integer (unsigned)	UI1 BYTE
2-byte integer (signed)	I2 SHORT
2-byte integer (unsigned)	UI2 WORD
4-byte integer (signed)	I4 LONG
4-byte integer (unsigned)	UI4 DWORD

[1] When a bit device is specified as a user variable, 1 word = 16 points is set to 1 for reading and writing.

Single-precision floating point	R4 SINGLE

By specifying VARTYPE option, the PLC-side data structure is determined and the appropriate size is trimmed.

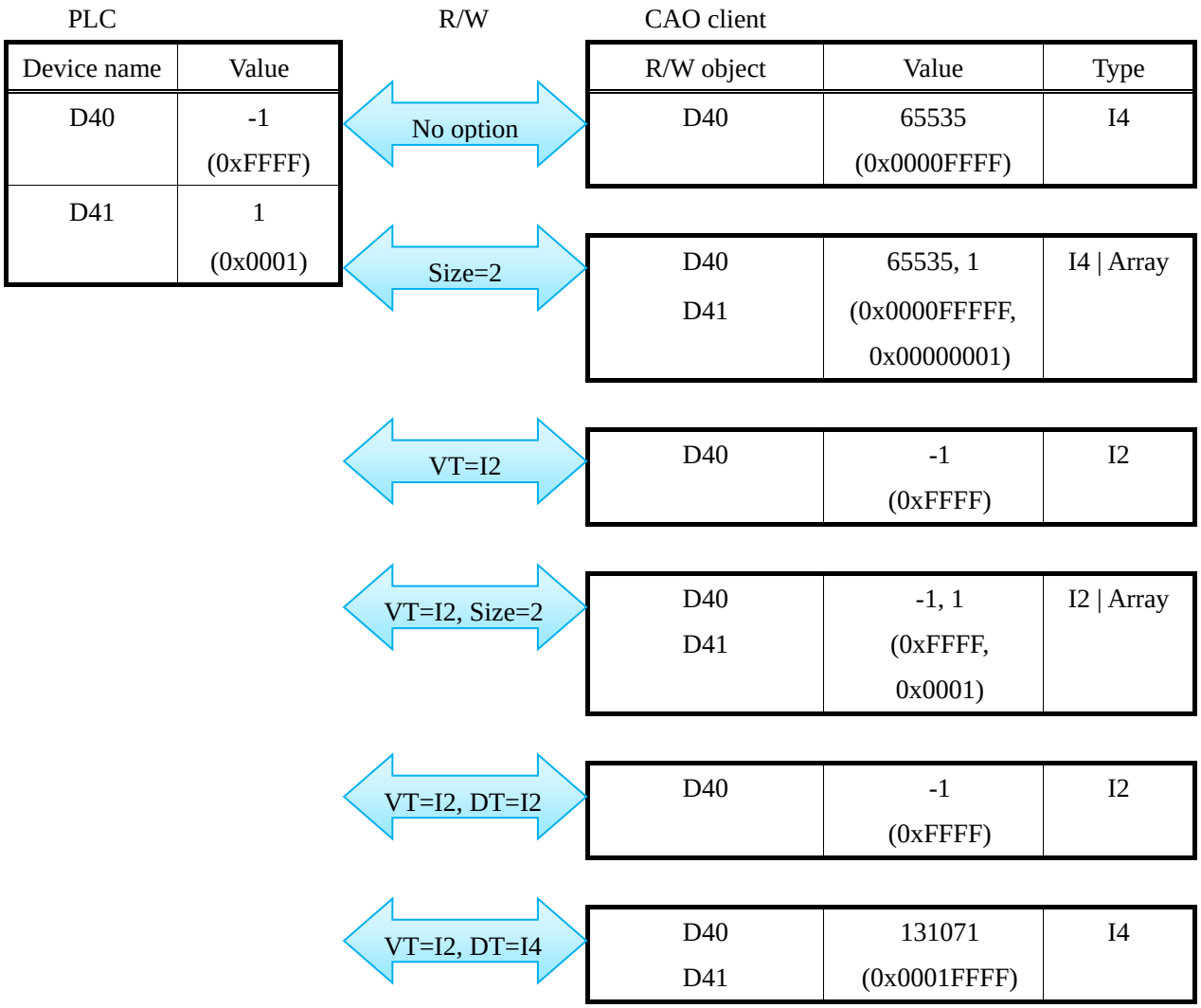
If you specify a type smaller than the PLC-side data in VARTYPE option, only the lower-byte information is acquired. If a large type is specified, the high-order byte is padded with zeros.

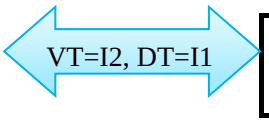
DATATYPE option converts the data to the specified type.

If the type of DATATYPE option is larger than VARTYPE option, the value of the device is stored in the device for which the low-order byte is specified, and the value of the device is stored in the high-order byte at the next address.

If DATATYPE option specifies a type that is less than VARTYPE option, the missing part is filled with zeros.

Fig. 2-1 Optional transformation of AddVariable





D40 D41	-1 (0x000000FF)	I1

2.2.4. CaoController::Execute method

By specifying "GetLastError" as the command name, you get the last error number that occurred in ActUtlType controlI will get it.

Format

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

BstrCmd	:	[in]	Command="GetLastError"
VntParam	:	[in]	Parameter (not used)
VntRet	:	[out]	Return Values

2.2.5. CaoTask::Start method

Set the CPU of the connection destination to the running state.

Format

Start(<lMode:LONG >[,<bstrOption:BSTR >])

lMode	:	[in]	Start mode (not used)
BstrOption	:	[in]	Option (not used)

Both the first and second arguments are ignored. However, the start mode of the first argument is mandatory and should always be set to 0.

2.2.6. CaoTask::Start method

Stops the CPU of the connection destination.

Format

Stop(<lMode:LONG >[,<bstrOption:BSTR >])

lMode	:	[in]	Stop mode
BstrOption	:	[in]	Option (not used)

If "2" (step stop) is specified for the stop mode, the CPU status is set to PAUSE; if any other value is specified, the CPU status is set to STOP.

2.2.7. CaoVariable::put_Value Property

Writes information corresponding to a variable name. Refer to 2.3.1 for details.

2.2.8. CaoVariable::get_Value Property

Reads information corresponding to a variable name. Refer to 2.3.1 for details.

2.3. Variable list

2.3.1. Controller class

Table 2-6 List of controller class user variables

Variable Name	Data type	Description	Attribute	
			Get	Put
<Device name>	Type specified in VARTYPE Or Type specified in DATATYPE	Read/write to/from the specified device. See MxComponent documentation for device-name specification formats. Batch reads and writes the word size specified by Size option of AddVariable.When 0 is specified by Size option, one device specified by the device name is read and written. The method of the corresponding ActUtlType is used to store data when reading or writing. See Table 2-1 for the corresponding functions.	✓	✓

Table 2-7 List of controller class system variables

Variable Name	Data type	Description	Attribute	
			Get	Put
@BUFFER: <IO number>: <Leading address>	VT_I2| VT_ARRAY	Reads and writes data to the buffer memory of the connection destination. Batch reads and writes the size specified by Size option of AddVariable.	✓	✓
@CLOCK	VT_DATE	Gets the time of the connected CPU.	✓	-
@CPUTYPE	VT_VARIANT| VTARRAY	Gets the connection destination CPU information. The CPU type name string is stored in the first element, and the CPU type name number is stored in the second element.	✓	-

2.4. Error code

MxCompo4 providers do not have unique error codes. For ORiN2 common errors, refer to the Error Codes section of ORiN2 Programming Guide.