MEWTOCOL-COM provider

For series made of Panasonic PLC FP

Version 1.2.3

User's guide

February 7, 2023

Remarks:

This document uses the machine translation.

【 revision history 】

Version	Date	Content
1.0.0	2018-11-13	First edition
1.1.0	2020-05-22	FrameSize is added to the options when using AddController.
1.2.0	2020-09-24	Changed the option StationNo in AddController so that it can be specified even in TCP/IP communication.
1.2.1	2021-12-21	Fixed false positives for errors.
1.2.2	2022-03-18	Fixed an issue that prevented index registers from being accessed. A raw command was added. Fix memory corruption bug in @PLC_STATUS and @PLC_ERROR. Fixed a bug that caused memory corruption when specifying large Elem options.
1.2.3	2023-02-07	Fixed a bug that caused memory corruption when a communication error occurred. Added parameters that can be used with AddController's CONN option. Errors have been corrected.

【Operation confirmation model】

Model	Notes
FP7	It corresponds to TCP/IP and the serial connection. There are an inaccessible device and a system variable by the limitation of the format of the MEWTOCOL-COM communication command. Timer setting value (TS) and timer passage value (TE) become read/writing 16 subordinate position bits.
ABXC32T	

Contents

1. Introduction

This book is an user's guide of the CAO provider that write/reads data made of Panasonic PLC. CAO provider (CaoProvPanasonicMEWTOCOL-COM.dll) that treats in this book is called MEWTOCOL-COM provider.

The outline of the MEWTOCOL-COM provider and details of the variable and the command have been described to the chapter 2.

Show the correspondence situation and the data string of the communication command mounted in the MEWTOCOL-COM provider in PLC that becomes a destination. Refer to "MEWTOCOL communication procedure" of the manual of PLC of Panasonic for details of the communication.

2. Outline of provider

2.1. Outline

The MEWTOCOL-COM provider is CAO provider that write/reads data by connecting TCP/IP or the cereal made of Panasonic PLC, and using the computer link function.

The file format is DLL(Dynamic Link Library), and when using it from the CAO engine, it is dynamically loaded. When the MEWTOCOL-COM provider is used, it is necessary to install ORiN2SDK or to register the registry by the hand work referring to the table below.

Table2-1MEWTOCOL-COM provider

File name	CaoProvPanasonicMEWTOCOL-COM.dll
ProgID	CaoProv.Panasonic.MEWTOCOL-COM
Registry registration	regsvr32 CaoProvPanasonicMEWTOCOL-COM.dll
Blotting out of registry registration	regsvr32 /u CaoProvPanasonicMEWTOCOL-COM.dll

2.2. Method property

2.2.1. CaoWorkspace::AddController method

The MEWTOCOL-COM provider connects the communication referring to connected parameter for the communication at AddController.

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

< bstrCtrlName >	:	[in] controller name
< bstrProvName >	:	[in] provider name. Fixed value ="CaoProv.Panasonic.MEWTOCOL-COM"
<bstrPcName>	:	Execution machine name of in provider (unused)
<bstrOption>	:	[in] optional character string

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

Table2-2 Optional character string of CaoWorkspace::AddController

Option	Meaning
Conn=< connected parameter >	Indispensability. Set a communication form and the connected parameter. (refer to chapter 2.2.1.1)
BlockCheck[=<TRUE/FALSE>]	Specify effective/invalidity of the horizontal parity check. (default: TRUE)
StationNo[=< station number >]	Specify the station number. Specify any two characters[1]. (Default: 01)
FrameSize[=<frame size>]	Specify the maximum number of characters that can be sent in one frame[2]. 0:118 characters, 1:2048 characters (Default: 1)
ConnTimeout[=< timeout period >]	Specify the timeout period (millisecond) when it connects it. (default: 3000)
Timeout [=< timeout period >]	Specify the timeout period (millisecond) in the TCP communication. (default: 3000)

[1] The characters that can be specified as a station number vary depending on the specifications of the connected device (decimal characters 0 to 99 in the case of FP7). For details, please refer to the instruction manual of the device to be connected.
[2] The extended size (2048 characters) may not be available depending on the connected device.

2.2.1.1. Conn is optional.

Connected parameter character string of optional Conn is shown as follows. A possible omission is shown here in the square bracket (""). Moreover, the underlined part under the explanation of each parameter shows the default value when the option is not specified.

【 Serial device 】

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

<COM Port>	:	COM port number. ‘1'-COM1,‘2'-COM2, …
<BaudRate>	:	Transmission rate. 300,600,1200,2400,4800,9600, 19200,38400,57600,115200, 230400.
<Parity>	:	Parity bit. 'N'-None, 'O'-Odd parity, 'E'-Even parity.
<DataBits>	:	Data bits. 7, 8.
<StopBits>	:	Stop bits. 1, 2.
<FlowControl>	:	Flow control. '0'-None, '2'-Hardware flow control.

【 Ethernet device 】

TCP/IP

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

< Dest IP Address >	:	Internet Protocol address of connection destination.
< Dest Port No >	:	Port number of connection destination.

2.2.1.2. Sample program

The sample program of AddController is shown below.

List 2-1

```
HRESULT hr = S_OK;
ICaoEngine* pEng = NULL;
ICaoWorkspaces *pWss = NULL;
ICaoWorkspace *pWs = NULL;
ICaoController *pCtrl = NULL;

// Generation of CaoEngine
hr = CoCreateInstance(CLSID_CaoEngine,
                      NULL,
                      CLSCTX_LOCAL_SERVER,
                      IID_ICaoEngine,
                      (void **)&pEng);
if (FAILED(hr)) {
        goto EndProc;
}

// Acquisition of CaoWorkspace collection
hr = pEng->get_Workspaces(&pWss);
if (FAILED(hr)) {
        goto EndProc;
}

// Acquisition of CaoWorkspace
hr = pWss->Item(CComVariant(0L), &pWs);
if (FAILED(hr)) {
        goto EndProc;
}

// Generation of CaoController
hr = pWs->AddController(CComBSTR(L"MEWTOCOL_SAMPLE"),
                      CComBSTR(L"CaoProv.Panasonic.MEWTOCOL-COM"),
                      CComBSTR(L""),
                      CComBSTR(L"Conn=TCP:192.168.1.5:60001"),
                      &pCtrl);
if (FAILED(hr)) {
        goto EndProc;
}

// Setting/acquire the value here at the generation of a variable.

EndProc:
if (pCtrl) pCtrl->Release();
if (pWs) pWs->Release();
if (pWss) pWss->Release();
if (pEng) pEng->Release();
```

2.2.2. CaoController::GetVariableNames property

Acquire the list of the shown system variable identifier of Table2-.

2.2.3. CaoController::AddVariable method

Make the variable object write/to read data to each device in PLC.

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

< bstrVariableName > : [in] variable identifier
<bstrOption> : [in] optional character string

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

Table 2-3 Optional character string of CaoController::AddVariable

Option ([3])	Meaning
DeviceType=<Device type >	Indispensability. Specify the device type at the access destination by the code. (refer to chapter 2.2.3.1)
Address=<Device number>	Indispensability. Specify the initial address of the device at the access destination. (refer to chapter 2.2.3.2) Two or more addresses can be specified in the bit device. (refer to chapter 2.2.3.3)
VT[=< data type >]	Specify the data type that does Put/Get. (refer to chapter 2.2.3.4)
Elem[=<the number of element >]	Specify the number of elements of data that does Put/Get by the decimal number. For bit-by-bit access, the upper limit is 8. (default: 1)
Array[=<TRUE/FALSE>]	Specify whether to make it to the array type at the time of reading one element. (default: FALSE)

[3] Square brackets ("[ ]") can be omitted.

2.2.3.1. DeviceType is optional.

Specify the device to be accessed by the device code. Table 2-4 shows the device code that can be specified. An external input, the timer, and the counter cannot write only reading.

Table 2-4 Device list

Device name	Device code	Address type	Method of addressing
External input	X	One bit	At access for unit of bit (one point) The decimal number and the low rank are hexadecimal numbers. three high rank digits At access for unit of word (16 points) Decimal number
External output	Y		
Internal relay	R		
Link relay	L		
Timer	T		Decimal number
Counter[4]	C		
Data register	DT	16 bits	Decimal number
Link register	LD		
System data register	SD		
Index register	I	16 bits	Hexadecimal number
Timer setting value register[5]	TS		Decimal number
Timer passage value register[5]	TE		

2.2.3.2. Address is optional.

As for the address, the specification method is different depending on the device type and the reading and writing point. Please refer to "Method of addressing" in Table 2-4 for the specification method.

(example)

X1F (access in each bit)	DeviceType=X,Address=1F
Y100 (access in each word)	DeviceType=Y,Address=100,VT=I2
DT250	DeviceType=DT,Address=250

2.2.3.3. Random read/writing

Bit device (X,Y,R,L,T,C) can be accessed to an address not consecutive by specifying two or more addresses for optional Address by colon (:) delimitation. Only one kind of DeviceType can be specified, and other VT,

[4] The counter is only for reading.
[5] The timer setting value register and the timer passage value register can read/write 16 subordinate position bits in FP7. As for 16 high rank bits, "H0000" is written.

Elem, and optional Array are disregarded.

It reads by specified array (VT_UI1|VT_ARRAY) for a few minutes of the address and/is written optional Address it. The addresses that can be specified are 1-8 pieces.

(example) DeviceType=X,Address=0:2:7 Access X0, X2, and X7.

2.2.3.4. VT is optional.

Specify the point a read and written data type and element (1 one point = bit) . Specify it by either of the values of the VT character string or corresponding VARTYPE (decimal number numerical value).

When I4,UI4,R4 is specified, consecutive 32 bits (32 elements for bit addresses and 2 elements for 16-bit addresses) are processed as one element.

When R8 is specified, consecutive 64 bits (64 elements for bit addresses and 4 elements for 16-bit addresses) are processed as one element.

Table 2-5 Data type list

VT	Data type	Point/number of elements	Explanation
BIT	VT_UI1	One point	Data is converted into binary of 0/1 and it reads and writes it. It is possible to specify it only for bit device (X,Y,R,L,T,C).
BOOL	VT_BOOL	One point	Data is converted into binary of 0/1 and it reads and writes it. It is possible to specify it only for bit device (X,Y,R,L,T,C).
BSTR	VT_BSTR	Eight points	It reads and writes it as ASCII of the Elem byte. 0 buries the shortfall from the specified number of elements under the writing value of a short character string when specifying it optional Elem it.
I1	VT_I1	Eight points	Make reading creaky as one byte data (There is a sign). When the number of elements of the odd number piece is specified, it treats as an element of the even number piece, and 0 is buried and eight added points are written optional Elem it.
I2	VT_I2	16 points	Make reading creaky as two byte data (There is a sign).
I4	VT_I4	32 points	Make reading creaky as four byte data (There is a sign).
UI1	VT_UI1	Eight points	Make reading creaky as one byte data (the sign none). When the number of elements of the odd number piece is specified, it treats as an element of the even number piece, and 0 is buried and eight added points are written optional Elem it.
UI2	VT_UI2	16 points	Make reading creaky as two byte data (the sign none).
UI4	VT_UI4	32 points	Make reading creaky as four byte data (the sign none).
R4	VT_R4	32 points	Make reading creaky as four byte data (floatage decimal).

R8	VT_R8	64 points	Make reading creaky as eight byte data (double precision floatage decimal).

Use the data type matched to the address type of the device when you omit the VT option.

Table 2-6 Data type when unspecifying it

Address type	VT	Data type
Bit	BIT	VT_UI1
16 bits	UI2	VT_UI2

2.2.3.5. Sample program

The sample program of AddVariable is shown below.

The example) Set the value to the address "DT0100" by the array (number 10 of elements).

List 2-2

```
HRESULT hr = S_OK;
ICaoEngine* pEng = NULL;
ICaoWorkspaces *pWss = NULL;
ICaoWorkspace *pWs = NULL;
ICaoController *pCtrl = NULL;
ICaoVariable *pVar = NULL;
CComVariant vntGet;

// Generation of CaoEngine
hr = CoCreateInstance(CLSID_CaoEngine,
                      NULL,
                      CLSCTX_LOCAL_SERVER,
                      IID_ICaoEngine,
                      (void **)&pEng);
if (FAILED(hr)) {
        goto EndProc;
}

// Acquisition of CaoWorkspace collection
hr = pEng->get_Workspaces(&pWss);
if (FAILED(hr)) {
        goto EndProc;
}

// Acquisition of CaoWorkspace
hr = pWss->Item(CComVariant(0L), &pWs);
if (FAILED(hr)) {
        goto EndProc;
}

// Generation of CaoController
hr = pWs->AddController(CComBSTR(L"MEWTOCOL_SAMPLE"),
                      CComBSTR(L"CaoProv.Panasonic.MEWTOCOL-COM"),
                      CComBSTR(L""),
           CComBSTR(L"Conn=TCP:192.168.1.5:60001"),
                      &pCtrl);
if (FAILED(hr)) {
        goto EndProc;
}

// Generation of a variable
hr = pCtrl->AddVariable(CComBSTR(L"A"),
                      CComBSTR(L"DeviceType=DT,Address=100,Elem=10"),
                      &pVar);
if (FAILED(hr)) {
        goto EndProc;
}
```

```
        // Setting and acquisition of value
        CComVariant vntPut;
        vntPut.vt = (VT_I2 | VT_ARRAY);
        vntPut.parray = SafeArrayCreateVector(VT_I2, 0, 10);
        pVar->put_Value(vntPut);
        pVar->get_Value(&vntGet);

    EndProc:
        if (pVar) pVar->Release();
        if (pCtrl) pCtrl->Release();
        if (pWs) pWs->Release();
        if (pWss) pWss->Release();
        if (pEng) pEng->Release();
```

2.2.4. CaoController::Execute method

Execute the extended method in Controller.

For the argument, specify the required argument for the method.

The specifi cations of Execute are shown below.

Format

```
Execute
(
        "<method name>",    ' Method name
        "<argument>"        ' Argument
)
```

The following is a list of methods that can be specified in Execute. The usage examples are described in detail for each method.

Table 2-7 CaoController::Execute Method List

Method name	Description
Raw	Send a MEWTOCOL-COM message directly to receive a reply.

2.2.4.1. Raw

Send a MEWTOCOL-COM message directly to receive a reply.

For the argument, specify the message from the header of MEWTOCOL-COM protocol ("%" or "<") to the right before the BCC with VT_BSTR.

If the response is normal, the return value returns as VT_BSTR from immediately after the normal response ("$") of the response to immediately before the BCC.

If the response was divided into multiple frames, the concatenated split frames are returned.
If communication fails or the response is abnormal, the error described in 2.4 is returned.

Usage example

```
String response = ctrl.Execute("Raw", "<01#RDD00010002");
```

2.2.5. CaoVariable::get_Value property

Send the command read to become a size specified from the device to be accessed in the option. Convert into the data type for which the result of reading is specified in the option and return it.

2.2.6. CaoVariable::put_Value property

Send the command written in the device to be accessed after converting the value passed by the argument according to optional specification.

2.3. Variable list

2.3.1. CaoController class

Table2-8 CaoController class system variable list

Variable identifier	Data type	Explanation	Attribute	
			get	put
@MAKER_NAME	VT_BSTR	Return "Panasonic ".	✓	–
@VERSION	VT_BSTR	Return the provider version.	✓	–
@PLC_MODEL	VT_BSTR	Send the PC status lead command, and return the model code and the version of the results.	✓	–
@PLC_MODE	VT_I2	Send the get:PC status lead command, and return the PROGRAM mode or the RUN mode. Put: Send the remote control command, and switch the operational mode of PLC. 0:PROGRUM mode and 1: RUN mode	✓	✓
@PLC_STATUS	VT_ARRAY|VT_I2	Send the PC status lead command, and return the operational mode of the results by arranging 0 and 1. The correspondence of the array element number and each mode is as follows. 0 : Operational mode 0:PROGRAM mode 1:RUN mode 1 : Static test mode 0:Driving and 1 usually: Static test mode 2 : BRK instruction and one step execution 0:Driving usually 1:When BRK/1 step is executed 3 : BRK instruction execution permission 0:BRK instruction invalidity 1:BRK instruction execution permission 4 : External output permission	✓	–

		0:Do not output the external. 1:Output the external. 5 : Execution permission of one step 0:It drives, and 1:1 steps are executable usually. 6 : MSG instruction execution 1:When message is displayed 7 : Operational mode 1:REMOTE		
@PLC_ERROR	VT_ARRAY|VT_I2	Send the PC status lead command, and return the error flag of the results by arranging normality/abnormality (0: normality and 1: abnormality). The correspondence of the array element number and each error is as follows. 0 : Self-diagnosis error 1 : Power failure detection at moment 2 : Fuse determination detection 3 : A high performance unit is abnormal. 4 : The I/O collation is abnormal. 5 : The battery is abnormal. 6 : Battery abnormality maintenance 7 : Operation error flag	✓	−

Table 2-9 CaoController class user variable list

Variable identifier	Data type	Explanation	Attribute	
			get	put
Arbitrariness	Variable dependence	Access the device in PLC.	✓	✓

- Only reading : an external input, the timer, and the counter device.

2.4. Error code

[in] the MEWTOCOL-COM provider, return an error code that as follows and is peculiar.

Table 2-10 Peculiar error code

Error name	Error number	Explanation
There is no response.	0x80100000	Return it when you cannot receive data from PLC. Confirm whether the network transmission setting of PLC is corresponding to an optional connection.
Receive data is abnormal.	0x80100001	Return it when data receives loss [shiteirunado] and data outside assumption.
Block check error	0x80100002	Return it when the block check code of receive data is not corresponding.
The writing data is abnormal.	0x80100003	Return it when the number of accesses is not corresponding to the number of data in the random access.
Error reply	0x801001XX	Return it when you receive the error as a response result of the command. Error code ([6]) of the hexadecimal number received from PLC is inserted in XX. Example) 22 → 0x80100122

[6] Please refer to the MEWTOCOL communication specification manual for details of the error code.