Open Protocol provider

Torque made by Atlas Copco company controller

Version 1.0.1

User's guide

October 30, 2018

【 remarks 】

・This document uses the machine translation.

【 revision history 】

Version	Date	Content
1.0.0	2018-06-15	First edition.
1.0.1	2018-10-30	Memory leak was corrected.

【 hardware 】

Model	Version	Notes
PF4000	14.12.1	It confirms the operation in the Open Protocol communication specification.

Contents

1. Introduction

This book is an user's guide of torque made by the Atlas Copco company controller's CAO provider that information acquires, and observes the event. CAO provider (CaoProv.ATLASCOPCO.OPENPROTOCOL.dll) that treats in this book is called Open Protocol provider.

Please refer to chapter 2 fir details of the outline and the variable of the Open Protocol provider.

Refer to “OpenProtocol_Specification_R_2_8_0_9836 4415 01" of the Atlas Copco Co. for details of the communication cable mounted in the Open Protocol provider.

2. Outline of provider

2.1. Outline

The Open Protocol provider is CAO provider that sends and receives data by using the torque made by Atlas Copco company controller, Ethernet or the serial communications port. The file format is DLL(Dynamic Link Library), and when using it from the CAO engine, it is dynamically loaded. When the Open Protocol provider is used, it is necessary to install ORiN2SDK or to register the registry by the hand work referring to the table below.

Table2-1Open Protocol provider

File name	CaoProvATLASCOPCOOPENPROTOCOL.dll
ProgID	CaoProv.ATLASCOPCO.OPENPROTOCOL
Registry registration	regsvr32 CaoProvATLASCOPCOOPENPROTOCOL.dll
Blotting out of registry registration	regsvr32 /u CaoProvATLASCOPCOOPENPROTOCOL.dll

2.2. Method property

2.2.1. CaoWorkspace::AddController method

When the Controller object is generated, the Open Protocol provider processes initial of Ethernet (TCP) and serial communications.

Moreover, specify the absolute path of the Levy John configuration file for an optional character string.

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

bstrCtrlName : In controller name

bstrProvName : In provider name. " CaoProv.ATLASCOPCO.OPENPROTOCOL fixed value ="

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

Option	Explanation
Conn=< connected parameter >	Indispensability. Set a communication form and the connected parameter.
ConnTimeOut =[< timeout period >]	Set the timeout period in the TCP communication by the millisecond. (default: 5000) It is effectively optional only to communicate Ethernet.
Timeout =[< timeout period >]	Set the timeout period in sending and receiving by the millisecond. (default: 3000)
Path =[< file path >]	Set the absolute path of the Levy John configuration file. Transmit the message by Levy John =1 when you omit it.
KeepAliveTime =[<Transmission interval>]	Set the interval when the effective maintenance command is transmitted by the millisecond. (default: 10000) =Do not transmit the effective maintenance command when you set 0. - Torque controller's communication time-out is 15 seconds. When messages are not exchanged for 15 seconds, the torque

	controller judges that the connection was cut, and ends. It is necessary to transmit at intervals within 15 seconds to continue the connection with the torque controller.

2.2.1.1. Conn is optional.

Connected parameter character string of optional Conn is shown as follows. The parameter in the square bracket ("") shows a possible omission here. Moreover, the underlined part under the explanation of each parameter shows the default value when optional specification is omitted.

【 Ethernet device 】

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

"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. 4545

【 cereal device 】

"Conn=COM:<COM Port>[:<BaudRate>]"

<COM Port> : COM port number. '1'-COM1,'2'-COM2, …

<BaudRate> : Transmission rate.
2400,4800,9600,19200,38400,57600,115200

2.2.1.2. Path is optional.

Unitary manage Levy John of each message with the xml file (Levy John configuration file).

Specify the absolute path of the Levy John configuration file for optional Path.

Example) Path=C:¥Users¥xxx¥AppData¥Local¥Temp¥IoTDS¥MIDLists.xml

There is Levy John in the message respectively. The same message is different the content of data depending on Levy John. The provider doesn't know Levy John for whom the message that tries to be transmitted is the best. The sender should understand Levy John of each message.

The format of the xml file is as follows.

```
<?xml version="1.0" encoding="SHIFT_JIS"?>
<MIDList>
    <MID id="0001" name="Application Communication start">
 <rev>4</rev>
    </MID>
 <MID id="0003" name="Application Communication stop">
 <rev>1</rev>
 </MID>
 <MID id="0010" name="Parameter set ID upload request">
 <rev>1</rev>
 </MID>
 <MID id="0018" name="Select Parameter set">
 <rev>1</rev>
 </MID>
 <MID id="0060" name="Last tightening result data subscribe">
 <rev>6</rev>
 </MID>
 <MID id="0062" name="Last tightening result data acknowledge">
 <rev>1</rev>
 </MID>
 <MID id="0063" name="Last tightening result data unsubscribe">
 <rev>1</rev>
 </MID>
</MIDList>
```

2.2.1.3. Sample program

The sample program of AddController is shown below.

【Ethernet】

```
 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"ATLASCOPCO_TEST"),
 CComBSTR(L"CaoProv.ATLASCOPCO.OPENPROTOCOL"),
                         CComBSTR(L""),
 CComBSTR(L"Conn=ETH:192.168.1.100:4545, Path=C:¥Users¥xxx¥AppData
                           ¥Local¥Temp¥IoTDS¥MIDLists.xml"),
 &pCtrl);
 if (FAILED(hr)) {
 goto EndProc;
 }

// Put processing necessary for here.
// The value is set, and acquired.

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

【 cereal 】

```
 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"ATLASCOPCO_TEST"),
 CComBSTR(L"CaoProv.ATLASCOPCO.OPENPROTOCOL"),
                         CComBSTR(L""),
 CComBSTR(L"Conn=COM:7, Path=C:¥Users¥xxx¥AppData¥Local¥Temp
                            ¥IoTDS¥MIDLists.xml"),
 &pCtrl);
 if (FAILED(hr)) {
 goto EndProc;
 }

// Put processing necessary for here.
// The value is set, and acquired.

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

2.2.2. CaoController::Execute method

The Execute method of the CaoController class does the sending and receiving message. Specify the command name in the first argument and specify the parameter of the command for the second argument.

Please refer to chapter 3 for details of the command mounted in the Open Protocol provider.

Format Execute(<bstrCommandName:VT_BSTR>[,<vntParam:VT_VARIANT>])

<bstrCommandName > : In command name

<vntParam> : In parameter

2.2.2.1. Sample program

The sample program of Execute is shown below.

The example) Select the parameter set.

```
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"ATLASCOPCO_TEST"),
 CComBSTR(L"CaoProv.ATLASCOPCO.OPENPROTOCOL"),
                          CComBSTR(L""),
 CComBSTR(L"Conn=ETH:192.168.1.100:4545, Path=C:¥Users¥xxx¥AppData
                            ¥Local¥Temp¥IoTDS¥MIDLists.xml"),
 &pCtrl);
 if (FAILED(hr)) {
 goto EndProc;
 }

// Communication beginning
 vntRet.Clear();
 vntParam.Clear();
 hr = pCtrl->Execute(CComBSTR("CommStart"), vntParam, &vntRet);
 if (FAILED(hr)) {
 goto EndProc;
 }

 ////////////////////////////////////////////////////////////////////////////////////////////
 // SelPSet
// Selection of parameter set
 ////////////////////////////////////////////////////////////////////////////////////////////
 vntRet.Clear();
 vntParam.Clear();
 vntParam.vt = VT_UI4;
    vntParam.ulVal = 1;
 hr = pCtrl->Execute(CComBSTR("SelPSet"), vntParam, &vntRet);
 if (FAILED(hr)) {
 goto EndProc;
 }

EndProc:

// Communication stop
 vntRet.Clear();
 vntParam.Clear();
 hr = pCtrl->Execute(CComBSTR("CommStop"), vntParam, &vntRet);
 if (FAILED(hr)) {
 goto EndProc;
 }

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

2.2.3. CaoController::AddVariable method

The AddVariable method of the CaoController class is a method for the access to the variable.

The variable mounted in the Open Protocol providerTable2-3Refer to [wo].

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

<bstrVariableName>	:	In variable identifier
<bstrOption>	:	In optional character string

2.2.3.1. Sample program

The sample program of AddVariable is shown below.

The example) Acquire the final error code of the reception thread.

```
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"ATLASCOPCO_TEST"),
 CComBSTR(L"CaoProv.ATLASCOPCO.OPENPROTOCOL"),
                          CComBSTR(L""),
 CComBSTR(L"Conn=ETH:192.168.1.100:4545, Path=C:¥Users¥xxx¥AppData
                            ¥Local¥Temp¥IoTDS¥MIDLists.xml"),
 &pCtrl);
 if (FAILED(hr)) {
 goto EndProc;
 }

// Generation of a variable
 hr = pCtrl->AddVariable(CComBSTR(L"@LASTERROR_RCV"), CComBSTR(L""), &pVar);
 if (FAILED(hr)) {
 goto EndProc;
 }

// Acquisition of value
 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::OnMessage event

If the event notification from the torque controller is received, the OnMessage event of the CaoController class is generated.

The event mounted in the Open Protocol providerTable4-1Refer to [wo].

2.2.4.1. Sample program

The sample program of the OnMessage event is shown below.

```
private void CtrlLoad()
{
 if (!this.ctrlLoaded)
 {
 try
 {
// CAO engine generation
 this.caoEng = new CaoEngine();
 this.caoWss = this.caoEng.Workspaces;
 this.caoWs = this.caoWss.Item(0);

// Acquisition of controller collection
 this.caoCtrls = this.caoWs.Controllers;

// Connect it with the controller.
 this.caoCtrl = this.caoCtrls.Add(this.textBox_CtrlName.Text,
 this.textBox_ProvName.Text,
 this.textBox_MachineName.Text,
 this.textBox_Option.Text);

// Registration of OnMessage event handler
 this.caoCtrl.OnMessage += new _ICaoControllerEvents_OnMessageEventHandler
                                                                    (caoCtrl_OnMessage);

 this.CtrlEnable(true);

// Window display for controller
 this.Ctrl = new Form2();
 this.Ctrl.ShowDialog(this);
 }
 catch (Exception ex)
 {
 MessageBox.Show(ex.Message);
 }
 }
}
```

```
void caoCtrl_OnMessage(CaoMessage pICaoMess)
{
 try
 {
 if ((pICaoMess.Number == 61) || (pICaoMess.Number == 900))
 {
Call of method of thread according to //
 SetMessageCallback SetMsg = new SetMessageCallback(SetText);
 Invoke(SetMsg, pICaoMess);
 }
 }
 catch (Exception ex)
 {
 MessageBox.Show(ex.Message);
 }
}

// Processing function at OnMessage
private void SetText(CaoMessage pICaoMsg)
{
// Add the processing when the event is received.
}
```

2.3. Variable list

2.3.1. Controller class

Table2-3Controller class variable list

Variable identifier	Data type	Explanation	Attribute	
			get	put
@MAKER_NAME	VT_BSTR	Provider making manufacturer	-	−
@VERSION	VT_BSTR	Provider version	-	−
@LASTERROR_RCV	VT_I4	The final error code of reception thread	-	−

2.4. Error code

In the Open Protocol provider, the following and peculiar the error code is defined. Moreover, Please refer to "ORiN2SDK user's guide" for the error code.

Table2-4Peculiar error code

Error name	Error number	Explanation
Command error	0x801001xx	Put the error code from the torque controller in the part of xx when you were not able to execute the demand for a certain reason and return it. Refer to the Open Protocol reference manual for the content of the error code.
Line unconnection error	0x80100200	Communication beginning command (CommStart) is not executed.

3. Command reference

3.1. Command list

Table3-1Command list

Category	Command	Function	
	CommStart	Communication beginning	P. 19
	CommStop	Communication stop	P. 20
	RegResults	The final tightening result data registration	P. 20
	UnregResults	Cancellation of the final tightening result data registration	P. 20
	SelPSet	Selection of parameter set	P. 21
	RegTraceCurve_Lastest	The final corrugate curve data registration	P. 21
	RegTraceCurve_FromId	Id specified corrugate curve data registration	P. 22
	RegTraceCurve_FromTime	Stamp specification wave type curve data registration of time	P. 22
	RegTraceCurve_SnapId	Register wavy curve data between Id.	P. 23
	RegTraceCurve_SnapTime	Wave type curve data registration between stamps time	P. 23
	UnregTraceCurve	Cancellation of wavy curve data registration	P. 24

3.2. The command is detailed.

CommStart

Syntax *object*.CommStart

Argument None

Return value VT_UI1 | VT_ARRAY: Communication beginning response

Explanation Transmit the communication beginning demand.

Connect the line for the normal termination.

Please refer to chapter 2.4 for the error code when terminating abnormally.

Refer to the manual of Open Protocol for details of the communication beginning response.

CommStop

Syntax *object*.CommStop

Argument None

Return value VT_I4 : 0

Explanation Transmit the communication stop demand.
Cut the line for the normal termination.
Please refer to chapter 2.4 for the error code when terminating abnormally.

RegResults

Syntax *object*.RegResults

Argument None

Return value VT_I4 : 0

Explanation Register the final tightening result data.
Receive the event of final tightening result data up-loading response (MID = 0061) from the torque controller when you execute tightening after it registers.
Please refer to chapter 2.4 for the error code when terminating abnormally.
Refer to the manual of Open Protocol for details of the final tightening result data up-loading response.

UnregResults

Syntax *object*.UnregResults

Argument None

Return value VT_I4 : 0

Explanation Cancel the registration of the final tightening result data.
Even if tightening is executed after this command is executed, the event of the final tightening result data up-loading response is not generated from the torque controller.

Please refer to chapter 2.4 for the error code when terminating abnormally.

SelPSet

Syntax *object*.SelPSet(<PSET>)

Argument < PSET>= VT_UI4: Parameter set number

Return value VT_I4 : 0

Explanation Switch to the parameter set number in which torque controller's parameter set number is specified by the argument.
The specified parameter set number should be a number of the parameter set that the torque controller has now.
Please refer to chapter 2.4 for the error code when terminating abnormally.

RegTraceCurve_Lastest

Syntax *object*.RegTraceCurve_Lastest<TYPE>

Argument < TYPE>= VT_BSTR: Curve type

Return value VT_I4 : 0

Explanation Register wavy curve data by Kon soon about the specified curve type.
Specify a curve type by the bit.

Curve type	Bit position
Angle trace	1
Torque trace	2
Current trace (*)	3
Gradient trace (*)	4
Stroke trace (*)	5
Force trace (*)	6

- Current trace-Force trace is a unsupport in a present version.

The example) Specify TYPE=3 when you acquire wavy curve data of Angle and Torque.
Receive the event of wavy curve data (MID = 0900) from the torque controller when

succeeding in executing this command.
Please refer to chapter 2.4 for the error code when terminating abnormally.
Refer to the manual of Open Protocol for details of wavy curve data.

RegTraceCurve_FromId

Syntax *object*.RegTraceCurve_FromId<DATA>

Argument <DATA> = VT_BSTR | VT_ARRAY: Curve type and tightening ID

Return value VT_I4 : 0

Explanation Register wavy curve data of a curve type since of specified tightening ID.
A curve type is "RegTraceCurve_LastestRefer to the paragraph of".
Specify tightening ID within the range of 1-4294967295.
Receive the event of wavy curve data (MID = 0900) from the torque controller when succeeding in executing this command. The event for a few minutes of data is generated when there is two or more data since of tightening ID.
Please refer to chapter 2.4 for the error code when terminating abnormally.

RegTraceCurve_FromTime

Syntax *object*.RegTraceCurve_FromTime<DATA>

Argument <DATA> = VT_BSTR | VT_ARRAY: Curve type and time stamp

Return value VT_I4 : 0

Explanation Register wavy curve data of a curve type since it stamps of the specified time.
A curve type is "RegTraceCurve_LastestRefer to the paragraph of".
Specify the time stamp by the character string in 19 bytes. (YYYY-MM-DD:HH:MM:SS)
Receive the event of wavy curve data (MID = 0900) from the torque controller when succeeding in executing this command. The event for a few minutes of data is generated when there is two or more data since it stamps of time.
Please refer to chapter 2.4 for the error code when terminating abnormally.

RegTraceCurve_SnapId

Syntax *object*.RegTraceCurve_SnapId<DATA>

Argument <DATA> = VT_BSTR | VT_ARRAY: Curve type and beginning tightening ID end tightening ID

Return value VT_I4 : 0

Explanation Register wavy curve data of a curve type from beginning tightening Id to end tightening Id.
A curve type is "RegTraceCurve_LastestRefer to the paragraph of".
Specify tightening ID within the range of 1-4294967295.
Receive the event of wavy curve data (MID = 0900) from the torque controller when succeeding in executing this command.
Please refer to chapter 2.4 for the error code when terminating abnormally.

RegTraceCurve_SnapTime

Syntax *object*.RegTraceCurve_SnapTime<DATA>

Argument <DATA> = VT_BSTR | VT_ARRAY: Curve type, start time, and finish time

Return value VT_I4 : 0

Explanation Register wavy curve data of a curve type from the start time to the finish time.
A curve type is "RegTraceCurve_LastestRefer to the paragraph of".
Specify the start time/the finish time by the character string in 19 bytes. (YYYY-MM-DD:HH:MM:SS)
Receive the event of wavy curve data (MID = 0900) from the torque controller when succeeding in executing this command.
Please refer to chapter 2.4 for the error code when terminating abnormally.

UnregTraceCurve

Syntax *object*.UnregTraceCurve<TYPE>

Argument < TYPE>= VT_BSTR: Curve type

Return value VT_I4 : 0

Explanation Cancel the registration of wavy curve data of the specified curve type.
A curve type is "RegTraceCurve_LastestRefer to the paragraph of".
Please refer to chapter 2.4 for the error code when terminating abnormally.

4. OnMessage event

4.1. Event list

Table4-1Event list

Event	Event number Number	Value		Remarks
		Type	Value	
The final tightening result data	61	VT_UI1 | VT_ARRAY	Reception Data	RegResults command call It is generated by the tightening execution.
Wavy curve data	900			It is generated by the following command executions. ・RegTraceCurve_Lastest ・RegTraceCurve_FromId ・RegTraceCurve_FromTime ・RegTraceCurve_SnapId ・RegTraceCurve_SnapTime

5. Appendix

5.1. Setting of communication

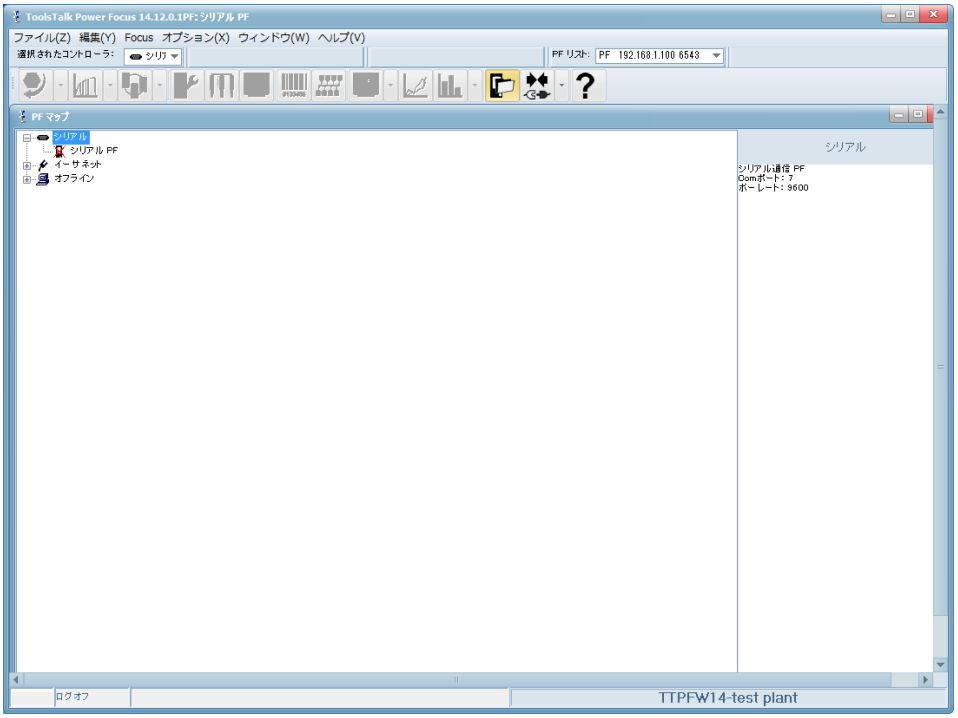
The Open Protocol provider sends and receives data by using the torque controller, Ethernet or the serial communications port.

Setup steps of each communication protocol are as follows.

【 cereal 】(Serial communications port 2 is used.)

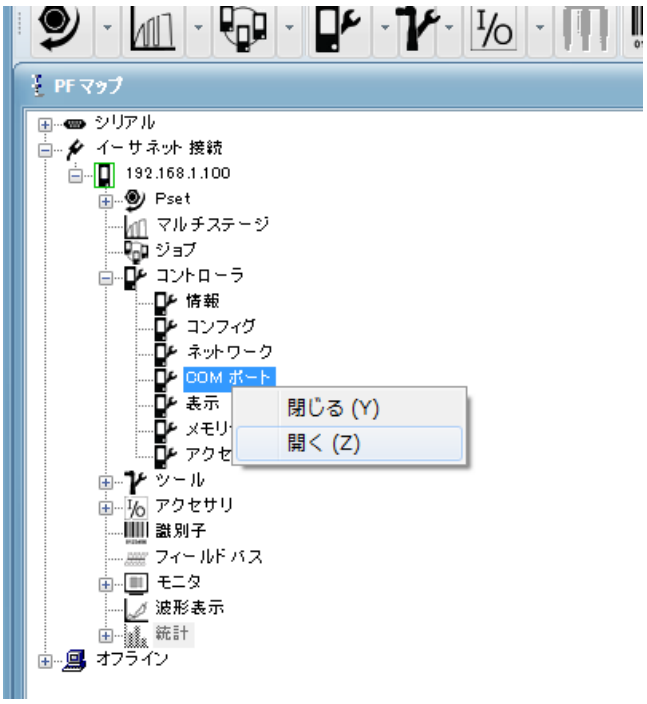
① Connect the torque controller with PC with Ethernet.

② Start set tool ToolsTalk Power Focus.

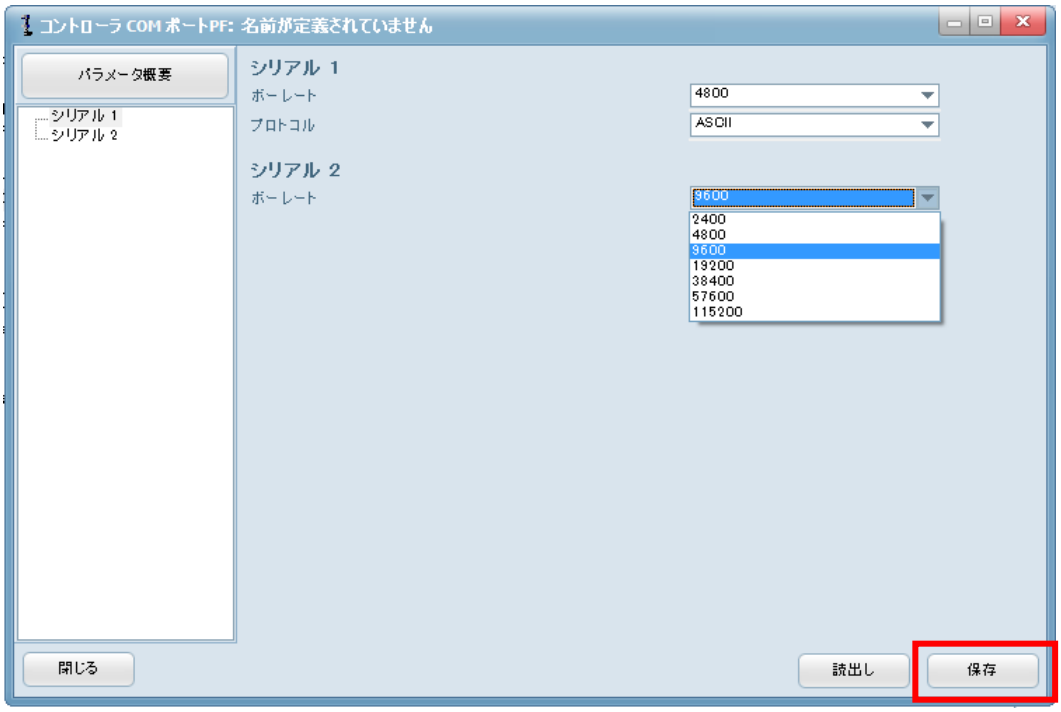


③ Double-click "Ethernet PF" of the PF map and connect the line.

④ Open "Controller" tree of the PF map, right-click in "COM port", and select "Open".



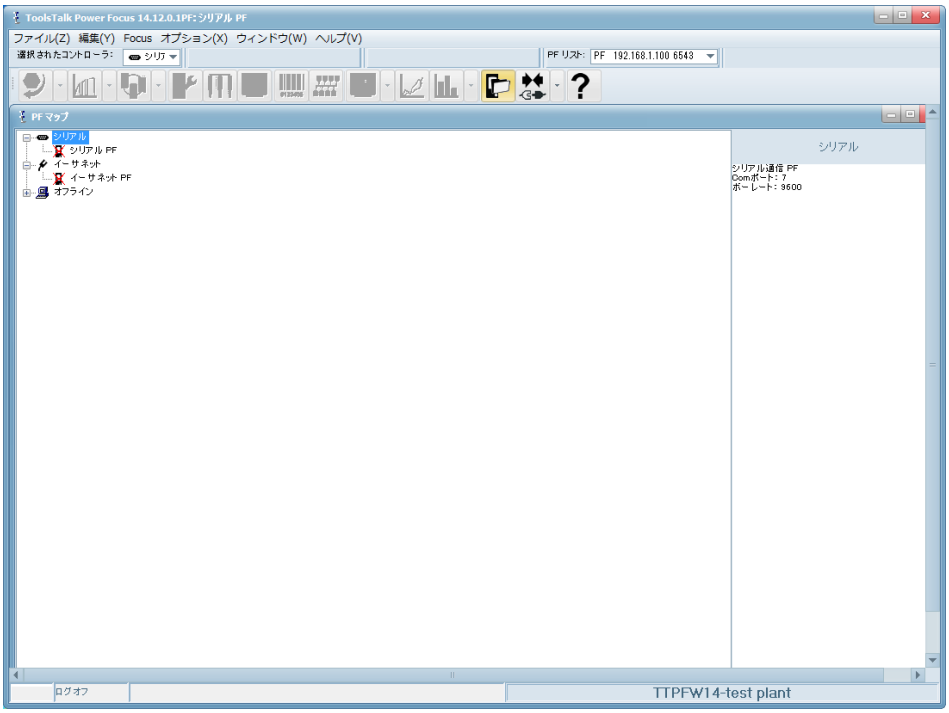
⑤ Set the baud rate of cereal 2, and push the preservation button.
Reactivate because the message is displayed to reactivate the torque controller. Turn on the power supply again for a while after time when the error is displayed when it reactivates.



【Ethernet】

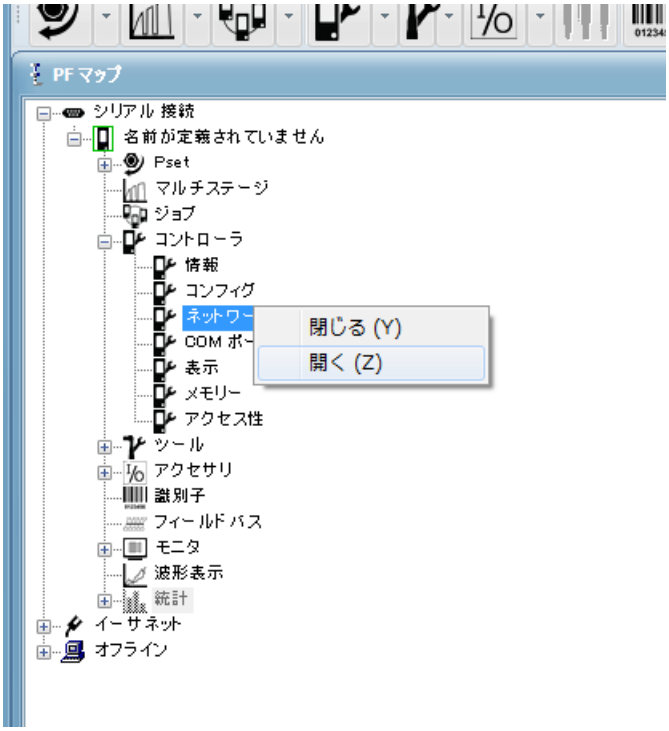
① Connect the torque controller with PC with the cereal.

② Start set tool ToolsTalk Power Focus.



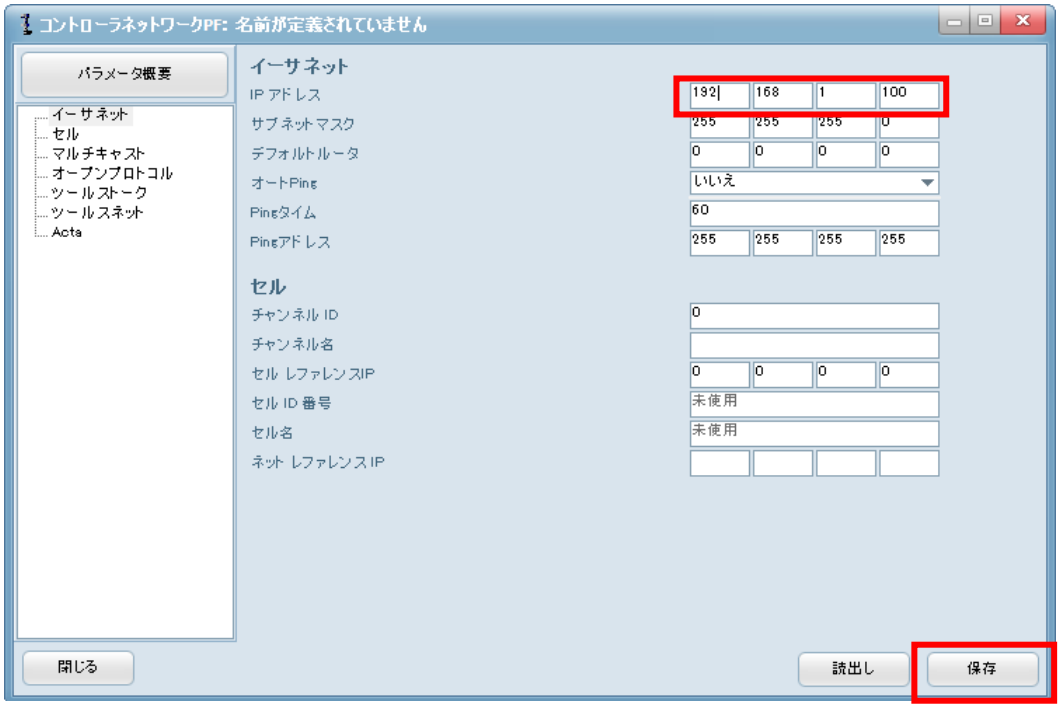
③ Double-click "Cereal PF" of the PF map and connect the line.

④ Open "Controller" tree of the PF map, right-click in "Network", and select "Open".



⑤ Set Internet Protocol address, and push the preservation button.

Reactivate because the message is displayed to reactivate the torque controller. Turn on the power supply again for a while after time when the error is displayed when it reactivates.



This concludes my report.