TOYO MACHINERY & METAL CO., LTD.
TOYO-MM PLCS12 providers

Version 1.1.0

User's Guide

August 6, 2021

NOTE:

[Revision History]

Version	Date	Content
1.0.0	2018-09-11	First edition
1.1.0	2021-08-06	Add the following variables ・Molding conditions ・Molding operation status Fixed an optional CaoWorkspace::AddController method. ・Removing Delay Options ・Adding CmdTimeout and ConnTimeout Options Amendments to Other Terms

[Operation check model]

Model	Version	Notes
J450C	-	[Operation in case of disconnection] Even if disconnection is restored and reconnection is successful, the machine may not respond to the communication command. If this happens, disconnect and try to reconnect after a certain period of time. [Multiple connections supported] Multiple client connections are not supported.

Contens

1. Introduction

This manual is a user's guide for providers who request and notify injection molding machines of Toyo Machinery and Metals, Inc. of data. Hereinafter, the injection molding machine of Toyo Machinery and Metals Co., Ltd. will be referred to as a PLCS-12 injection molding machine. Fig. 1-1 shows the overall configuration of this provider and the device. The providers are referred to as TOYO-MM PLCS12 providers.

TOYO-MM PLCS12 providers transmit and receive data to and from PLCS-12 injection molding machines using TCP/IP protocols.

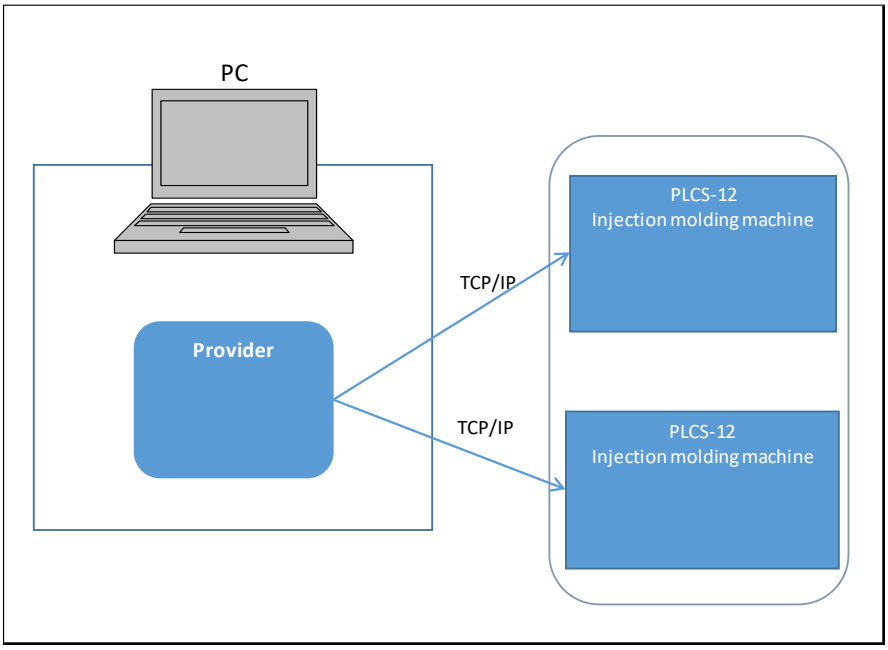

Fig. 1-1 Configuration Diagram

1.1. Environment and version assumed by this document

The environment in which the client PC runs on Windows and the target PLCS-12 injection machine can be connected via Ethernet..PC development environment can be developed as long as the programming environment supports Component Object Model (COM, Component Object Model).

1.2. Informative sources

All examples of programming in this document are described in Visual Basic 6.0, but can be developed in a variety of programming languages, including C++, Java, and .NET. For instructions, refer to ORiN2 Programming Guide.

ORiN2 Programming Guide applies to the following files in ORiN2 SDK installation folder:

- ORiN2¥CAO¥Doc¥ORiN2_ProgrammersGuide_<lang>.pdf

※Replace <lang> with an environment-specific linguistic string and read it.

Describes the basics and techniques of ORiN2, COM/DCOM required to develop applications using providers.

TOYO-MM PLCS12 providers have been developed with reference to the "PLCS-12 Monitoring Software Ethernet Communication Specification (filename: C07058P P-12 Ethernet Communication Specification.pdf)" provided by Toyo Machine Metal Co., Ltd. Subsequently, this specification is called PLCS-12 Ethernet communication specification.

Reference listed below

C07058P P-12 Ethernet Communication Specifications.pdf

C14365P Ethernet Communication Specifications.pdf

2. Setting Up Your Environment for Application Development

2.1. Connecting a PLCS-12 Injection Molding Machine to a ClientPC

Fig. 2-1 is part of the control panel of PLCS-12 injection molding machine.

PLCS-12 Injection Molding Machine and ClientPC are connected by Ethernet communication (TCP-protocol).

Connect the LAN cable from the LAN port of the PC to the LAN port (CMA26) of PLCS-12 Injection Molding Machine (Fig. 2-2).

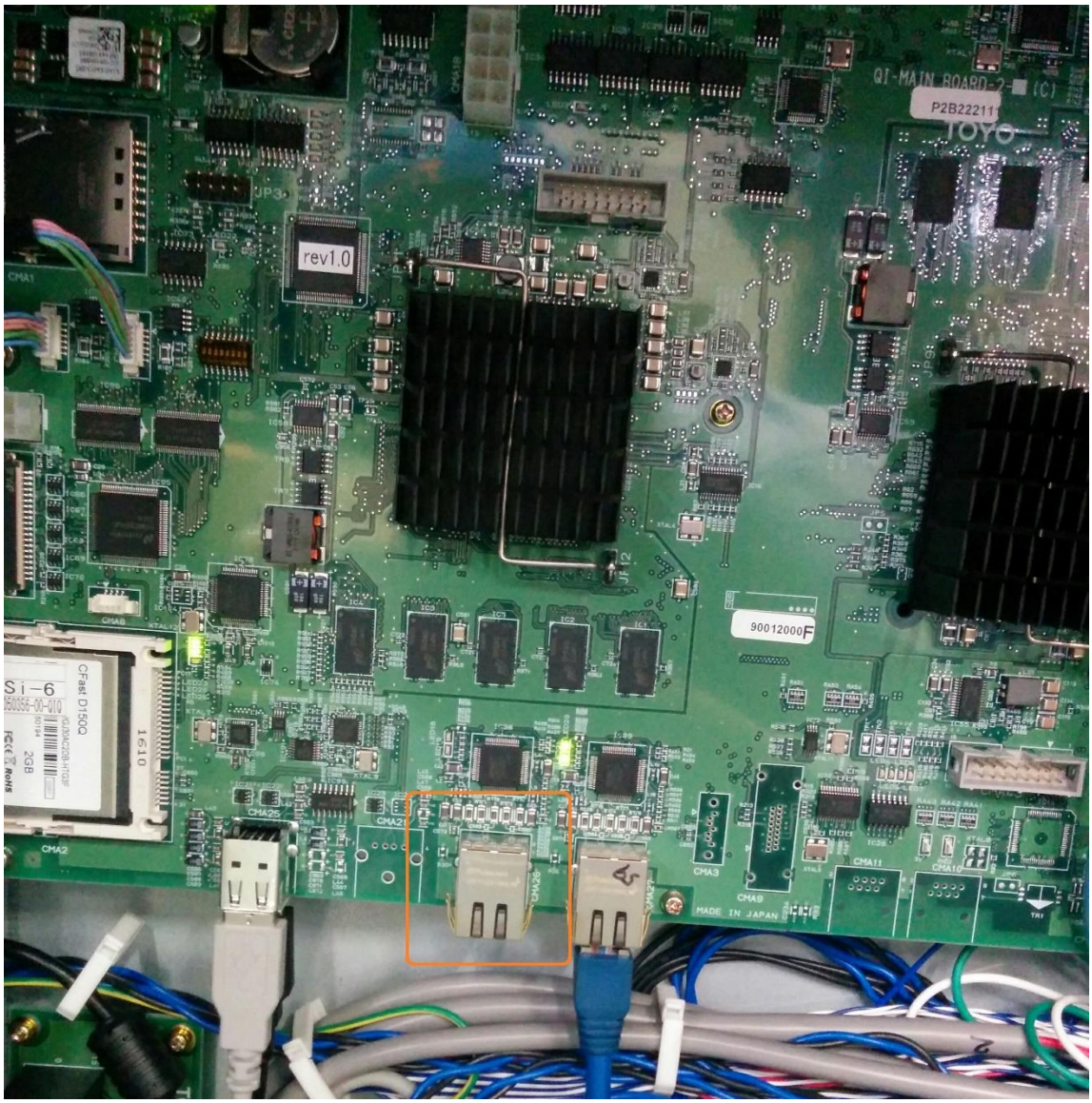

Fig. 2-1 Control panel

After connecting the LAN cable, set the IP address in PLCS-12 Injection Molding Machine Display Settings.

Press MENU1 to open MENU1 as shown in Fig. 2-3.

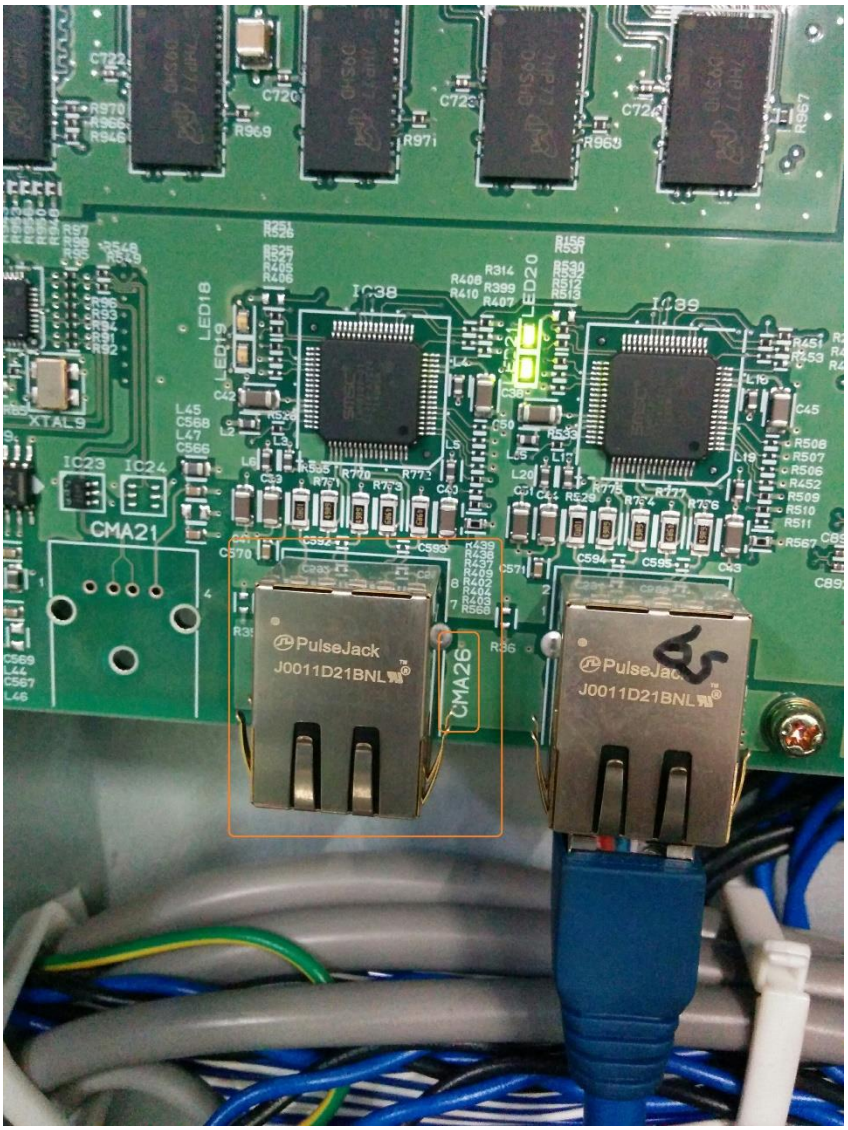

Fig. 2-2 LAN

Select the communication port setting to display Fig. 2-4 of the communication port setting screen.

Fig. 2-3 MENU1 display

Set the IP address on the communication port. Set the same IP address as the Ethernet IP address (left 3 digits) and subnet mask of the client PC. (See Fig. 2-5.)

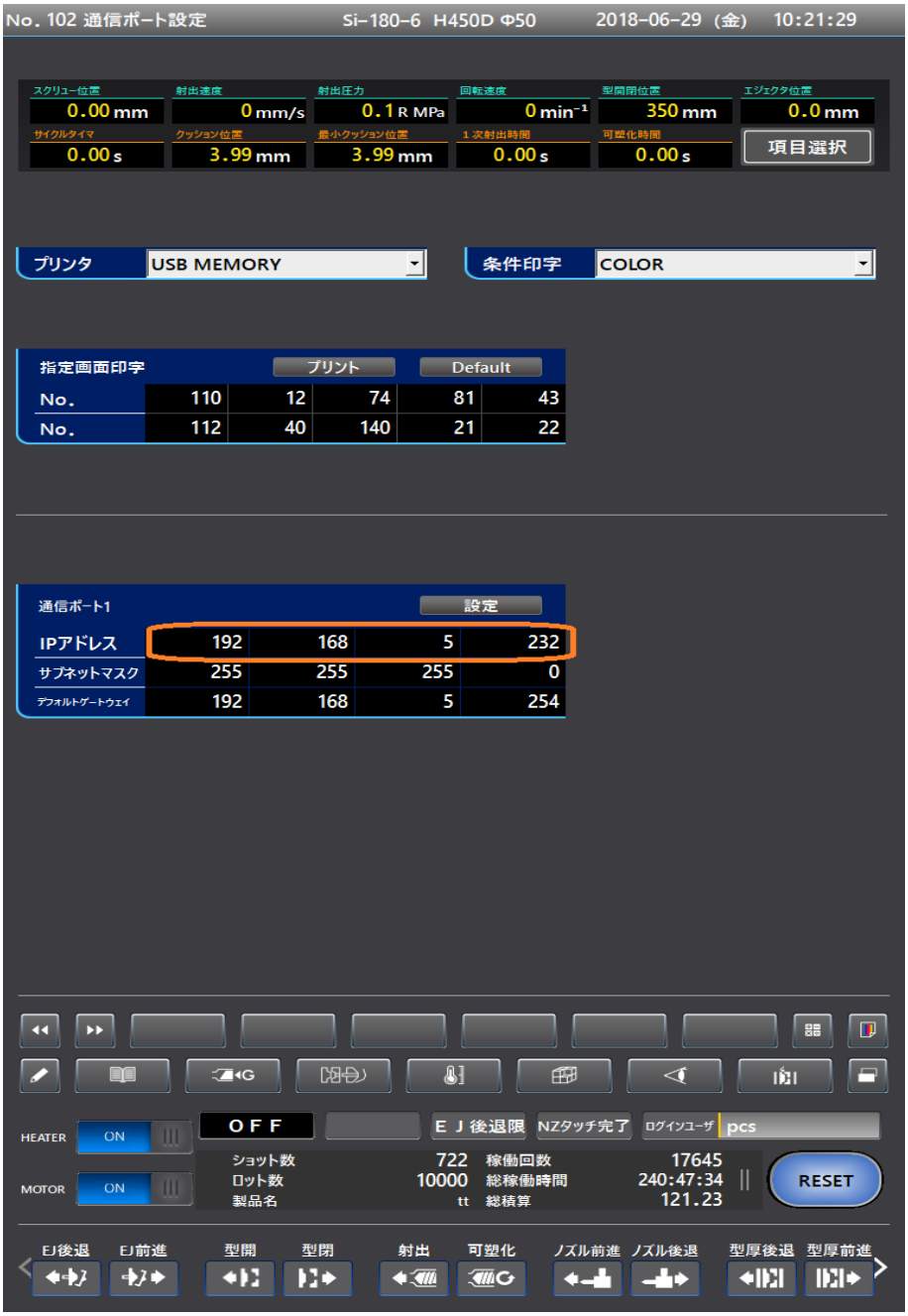

Fig. 2-4 Communication port screen

Same as PLCS-12 injection molding machine.

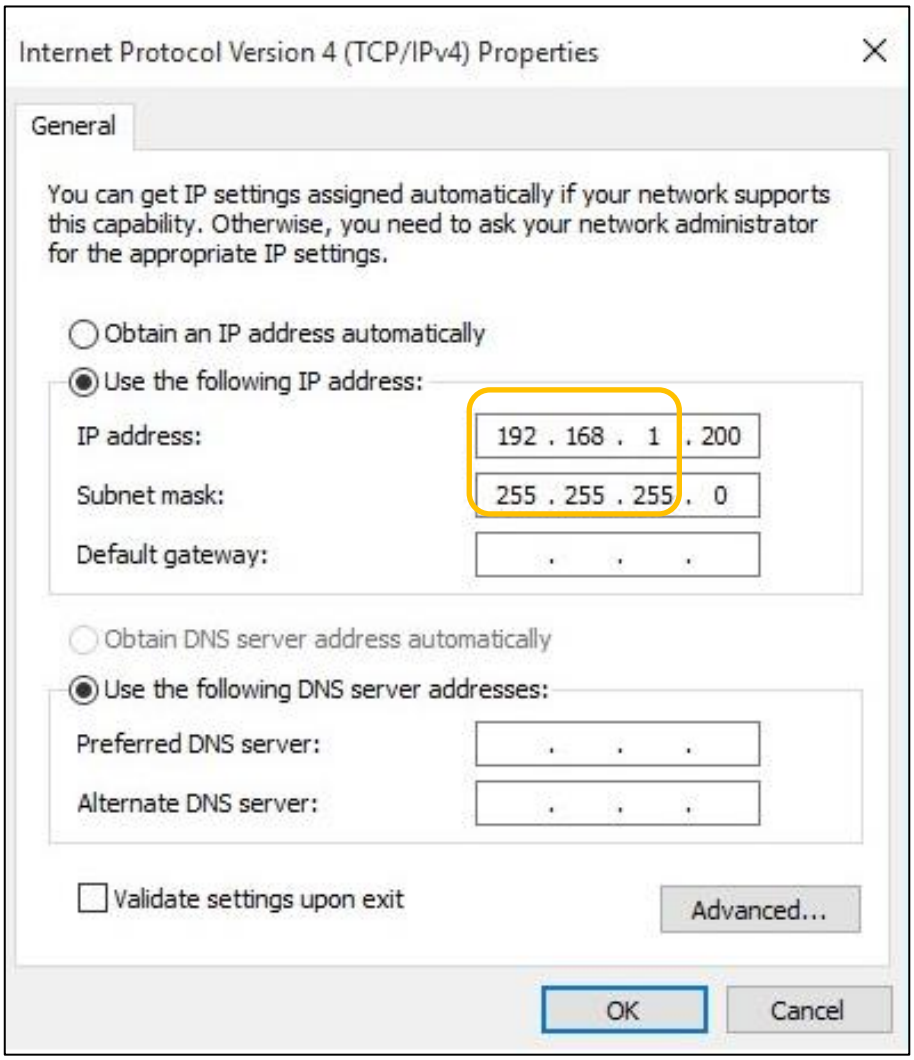

Fig. 2-5 Properties of TCP/IPv4

2.2. Setting up a PC development environment

2.2.1. Installing TOYO-MM PLCS12 Providers Automatically

If you have ORiN2 SDK installed, you are ready to connect to PLCS-12 Injection Machine.

To set up your development environment, please prepare a programming environment that supports Component Object Model (COM, Component Object Model) such as Microsoft Visual Studio 6.0, 2003/2005/2008/2010, LabVIEW, etc.

2.2.2. Installing TOYO-MM PLCS12 Providers Manually

If you install the software manually to use TOYO-MM PLCS12 providers, you must register the registry as shown below. To register the registry, start the command prompt with administrator privileges and execute regsvr32 command.

In order for the CAO engine to operate, one legitimate ORiN2 SDK license must be registered for each PC. Refer to the section "Adding and Removing Licenses" in ORiN2 SDK User's Guide.

Table2-1 TOYO-MM PLCS12 Providers

File name	CaoProvTOYO-MMPLCS12.dll
ProgID	CaoProv.TOYO-MM.PLCS12
Registry registration	Regsvr32 CaoProvTOYO-MMPLCS12.dll
Deletion of Registry Registration	Regsvr32 /u CaoProvTOYO-MMPLCS12.dll

3. Programming by TOYO-MM PLCS12 providers

TOYO-MM PLCS12 providers allow you to connect PLCS-12 Injection Molding Machine to the ClientPC as follows:

- Creating a CaoEngine
- Creating a CaoWorkspace
- Creating a CaoController

After connecting to PLCS-12 Injection Molding Machine, CaoController's OnMessage events provide access to the info of PLCS-12 Injection Molding Machine.

3.1. Sample programming to receive monitor data 1

This example shows a sample program that receives data from a PLCS-12 injection molding machine. Table 3-1 describes the requirements of the sample program, and Fig. 3-1 describes the flow of the sample program.

Table 3-1 Sample program requirements

	Description
Host	The destination IP address is 192.168.1.201.
Process Description	The monitor data 1 from PLCS-12 injection machine is received in OnMessage.

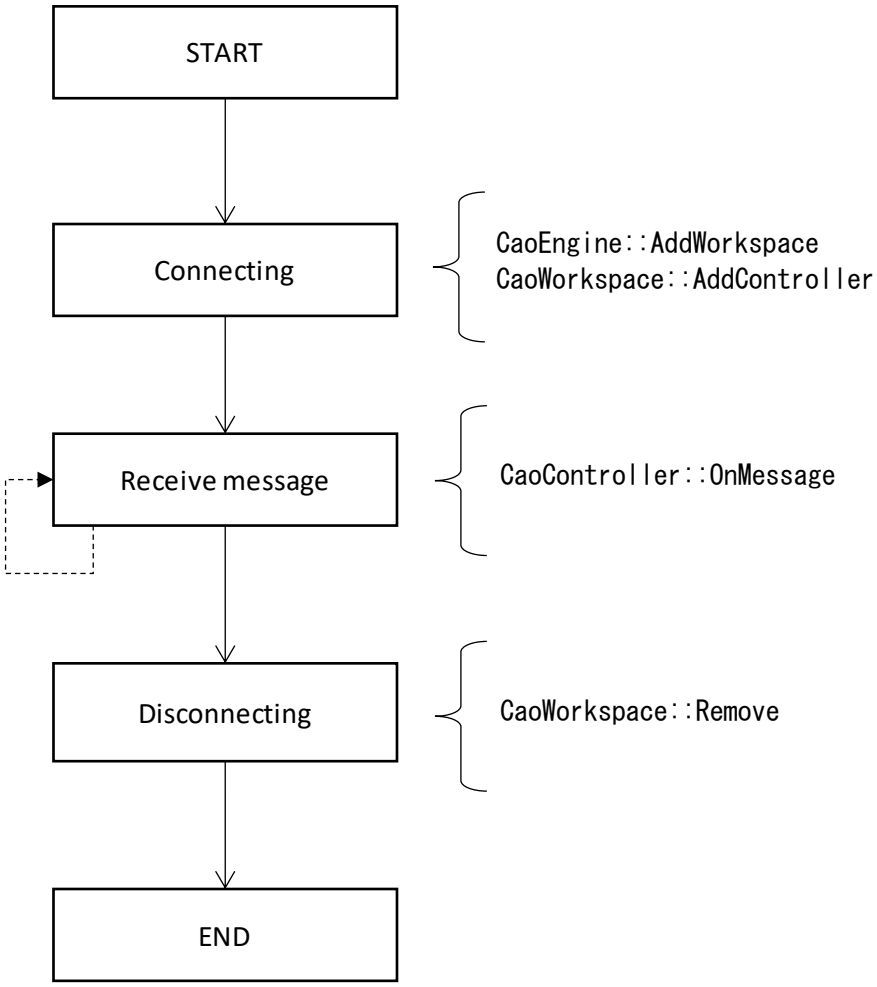

Fig. 3-1 Flow of Monitor Data 1 Reception

Specific codes are given in the following sections.

3.1.1. Sample program

The following is an overview of the sample program.

Sample	ReceiveCheckMonitorData1.vb

```
' Object
Private engine As CaoEngine
Private workspace As CaoWorkspace
Private WithEvents controller As CaoController

' Connection method
Sub Connect()
    ' Generate CaoEngine
    Set engine = New CaoEngine

    ' Generate CaoWorkspace
    Set workspace = engine.AddWorkspace("NewWrks", "")

    ' Generate CaoController
```

```
    Set controller = workspace.AddController("Controller", _
                                             "CaoProv.TOYO-MM.PLCS12", _
                                             "", _
                                             "Conn=Tcp:192.168.1.201")
End Sub

' Disconnect method
Sub Disconnect()
    ' Remove CaoController from CaoWorkspace
    Call workspace.Controllers.Remove(controller.Index)
    ' Clear CaoController
    Set controller = Nothing

    ' Remove CaoWorkspace from CaoEngine
    Call engine.Workspaces.Remove(workspace.Index)
    ' Clear CaoWorkspace
    Set workspace = Nothing

    ' Clear CaoEngine
    Set engine = Nothing
End Sub

' Message reception method
Private Sub controller_OnMessage(ByVal pICaoMess As CAOLib.ICaoMessage)
    ' If the received message number is monitor data 1, the message body is displayed.
    If pICaoMess.Number = 33 Then
        Dim moniValues As Variant
        MoniValues = pICaoMess.Value

        ' Date
        Dim moniDate As String
        MoniDate = moniValues(0) & "/" & moniValues(1) & "/" & moniValues(2)
        Call MsgBox(moniDate)

        ' Hour, minute, and second
        Dim moniTime As String
        MoniTime = moniValues(3) & ":" & moniValues(4) & ":" & moniValues(5)
        Call MsgBox(moniTime)

        ' Number of shots
        Dim moniShots As Integer
        MoniShots = moniValues(6)
        Call MsgBox(moniShots)

        ' Monitor data
        For i = 1 To 96 Step 1
            Dim moniData As Integer
            MoniData = moniValues(6 + i)
            Call MsgBox(moniData)
        Next i

        ' Bad mark (0... OK, 1... bad)
        For i = 1 To 96 Step 1
            Dim moniFailure As Integer
            MoniFailure = moniValues(102 + i)
```

```
            Call MsgBox(moniFailure)
        Next i
    End If
End Sub
```

3.1.1.1. Connection

To connect to a PLCS-12 injection machine, do the following:

(1) Prepare a variable to hold the object. The objects required to connect to the controller are CaoEngine objects, CaoWorkspace objects, and CaoController objects. In VB6, CaoController type variable must be declared with "WithEvents" in order to handle CaoController events.

```
' Variables for CaoEngine Objects
Private engine As CaoEngine
' Variables for CaoWorkspace Objects
Private workspace As CaoWorkspace
' Variables for CaoController Objects
Private WithEvents controller As CaoController
```

(2) Creates a CaoEngine object. CaoEngine object is generated using the New keyword.

```
' Generate CaoEngine
Set engine = New CaoEngine
```

(3) Gets or generates a CaoWorkspace object. When you create a CaoEngine object, it defaults to one CaoWorkspaces object and one object. The following is a sample code/default CaoWorkspace for creating a new CaoWorkspace.

```
' Generate CaoWorkspace
Set workspace = engine.AddWorkspace("NewWrks", "")

' Getting the Default CaoWorkspace Objects
Set workspace = engine.Workspaces.Item(0)
```

(4) Create a CaoController object. To generate a CaoController object, set the provider name to use and the parameters to use. For TOYO-MM PLCS12 providers, optionally specify the IP address to connect to. The following is a code example:

```
' Generate CaoController
Set controller = workspace.AddController("Controller", _
                                         "CaoProv.TOYO-MM.PLCS12", _
                                         "", _
                                         "Conn=Tcp:192.168.1.201")
```

3.1.1.2. Receive message

PLCS-12 Injection Molding Machine data can be obtained by handling CaoController::OnMessage event. The sample program handles CaoController::OnMessage events by providing controller_OnMessage event handlers. For the specifications of each message, see section 4.4, Event List.

```
' If the received message number is monitor data 1, the message body is displayed.
If pICaoMess.Number = 33 Then
    Dim moniValues As Variant
    MoniValues = pICaoMess.Value

    ' Date
    Dim moniDate As String
    MoniDate = moniValues(0) & "/" & moniValues(1) & "/" & moniValues(2)
    Call MsgBox(moniDate)

    ' Hour, minute, and second
    Dim moniTime As String
    MoniTime = moniValues(3) & ":" & moniValues(4) & ":" & moniValues(5)
    Call MsgBox(moniTime)

    ' Number of shots
    Dim moniShots As Integer
    MoniShots = moniValues(6)
    Call MsgBox(moniShots)

    ' Monitor data
    For i = 1 To 96 Step 1
        Dim moniData As Integer
        MoniData = moniValues(6 + i)
        Call MsgBox(moniData)
    Next i

    ' Bad mark (0... OK, 1... bad)
    For i = 1 To 96 Step 1
        Dim moniFailure As Integer
        MoniFailure = moniValues(102 + i)
        Call MsgBox(moniFailure)
    Next i
End If
```

3.1.1.3. Disconnect

To disconnect from the controller, you can erase the generated objects and delete the objects that you want to erase from the collection class that manages the objects. The following is a code example:

```
' Remove CaoController from CaoWorkspace
Call workspace.Controllers.Remove(controller.Index)
' Clear CaoController
Set controller = Nothing
' Remove CaoWorkspace from CaoEngine
Call engine.Workspaces.Remove(workspace.Index)
' Clear CaoWorkspace
Set workspace = Nothing
' Clear CaoEngine
Set engine = Nothing
```

4. Command Reference

4.1. Method/Property List

Table 4-1 List of methods and properties

Category	Methods/Properties[1]		Function	Reference
CaoWorkspace				
	AddController	M	Connected to controller	P.20
CaoController				
	Index	P	Obtaining the Controller Number	P.22
	Name	P	Get Controller Name	P.22
	VariableNames	P	Get a list of variable names that can be connected	P.22
	Variables	P	Retrieving Variable Collections Holded by the Controller	P.23
	AddVariable	M	Adding Variable Objects	P.24
	OnMessage	E	Message reception event	P.24
CaoVariable				
	Index	P	Obtaining Variable Numbers	P.25
	Name	P	Retrieving Variable Names	P.25
	Value	P	Get/set value	P.25

[1] M:Indicates methods, P: properties, and E: events, respectively.

4.2. Method properties

4.2.1. CaoWorkspace classes

4.2.1.1. AddController method

In CaoWorkspace, add a controller object. TOYO-MM PLCS12 providers refer to the parameters passed when AddController method is executed and connect to the appropriate PLCS-12 injection machine. The following are the specifics of AddController method:

Format

CaoController AddController

```
(
        "<controller name>",            // Controller name (optional)
        "CaoProv.TOYO-MM.PLCS12",       // Provider name (fixed)
        "<machine name>",               // Provider execution machine name (unused)
        "<Option>"                      // Option string
)
```

Usage example

```
Dim engine As CaoEngine          ' Engine
Dim workspace As CaoWorkspace            ' Workspace
Dim controller As CaoController  ' Controller

Set engine = New CaoEngine
Set workspace = engine.Workspaces.Item(0)
Set controller = workspace.AddController("Controller", _
                                          "CaoProv.TOYO-MM.PLCS12", _
                                          "", _
                                          "Conn=Tcp:192.168.1.201")
```

Option

The following options are specified in the option string: The option string is a string consisting of the following options separated by a comma (,).

Option	Required	Description	Value Range	Default value
Conn	✓	Specifies the TCP-connection options for connecting to PLCS-12 Injection Molding Machine. See 4.2.1.1.1. Conn Options for more information.	-	-

Option	Required	Description	Value Range	Default value
ConnTimeout[2]	-	Specify the connection, transmission and reception timeout in ms.	1 - 65534	500
CmdTimeout	-	Specifies the timeout in ms when acquiring a value. ConnTimeout The following specifications cannot be specified.	2 - 65535	1000

4.2.1.1.1. Conn Optional

The following is a Conn optional connection parameter string: Here, braces ("[]") are optional, and the underlined part in the description of each parameter indicates the default value when no options are specified.

In PLCS-12 Injection Molding Machine that you checked, you could not confirm the change of the port number, so you do not need to specify the port to connect to.

An underscore indicates the default value.

"Conn=Tcp:<IP>[:<Port>]"

<IP> : Host address.

<Port> : Destination port . (8200)

[2] ConnTimeout option is an alternative to Ver1.0.0 option that existed in Timeout. We recommend that you use ConnTimeout option in Ver1.1.0 and later. If you specify Timeout and ConnTimeout at the same time, ConnTimeout takes precedence.

4.2.2. CaoController classes

4.2.2.1. Index Properties

Gets the controller number as a Long type (4-byte integer type). This number indicates the number of controller collections held by CaoWorkspace by the corresponding controller.

Usage example

```
Dim engine As CaoEngine          ' Engine
Dim workspace As CaoWorkspace            ' Workspace
Dim controller As CaoController  ' Controller

Set engine = New CaoEngine
Set workspace = engine.Workspaces.Item(0)
Set controller = workspace.AddController("Controller", _
                                          "CaoProv.TOYO-MM.PLCS12", _
                                          "", _
                                          "Conn=Tcp:192.168.1.201")

'Get Index
Dim index as Long
Index = controller.Index
```

4.2.2.2. Name Properties

Retrieves the controller name specified by AddController method of CaoWorkspace class.

Usage example

```
Dim engine As CaoEngine          ' Engine
Dim workspace As CaoWorkspace            ' Workspace
Dim controller As CaoController  ' Controller

Set engine = New CaoEngine
Set workspace = engine.Workspaces.Item(0)
Set controller = workspace.AddController("Controller", _
                                          "CaoProv.TOYO-MM.PLCS12", _
                                          "", _
                                          "Conn=Tcp:192.168.1.201")

Debug.Print controller.Name
```

4.2.2.3. VariableNames Properties

Gets a list of variable names that can be connected. Variable names obtained by this property will be described later.

AddVariable method

Usage example

```
Dim engine As CaoEngine          ' Engine
Dim workspace As CaoWorkspace            ' Workspace
Dim controller As CaoController  ' Controller

Set engine = New CaoEngine
Set workspace = engine.Workspaces.Item(0)
Set controller = workspace.AddController("Controller", _
                                          "CaoProv.TOYO-MM.PLCS12", _
                                          "", _
                                          "Conn=Tcp:192.168.1.201")

' Get variable name list
Dim variables as Variant
Variables = controller.VariableNames
```

4.2.2.4. Variables Properties

Gets a collection of variables that the controller holds.

Usage example

```
Dim engine As CaoEngine          ' Engine
Dim workspace As CaoWorkspace            ' Workspace
Dim controller As CaoController  ' Controller

Set engine = New CaoEngine
Set workspace = engine.Workspaces.Item(0)
Set controller = workspace.AddController("Controller", _
                                          "CaoProv.TOYO-MM.PLCS12", _
                                          "", _
                                          "Conn=Tcp:192.168.1.201")

' Variable Collection Retrieval
Dim variables as CaoVariables
Set variables = controller.Variables

' Variable acquisition
Dim variable as CaoVariable
Set variable = variables.Item(0)
```

4.2.2.5. AddVariable method

Adds a variable object to CaoController. You can only use variable names that are listed in 4.3.1.

AddVariable is specifi ed as follows.

Format

```
CaoVariable AddVariable
(
        "<variable name>",          // Variable Name
        "<Option>"                  // Option string (optional)
)
```

4.2.2.6. OnMessage event

You can receive controller error notifications and status changes as OnMessage events. Refer to 4.4 for the events that can be received.

4.2.3. CaoVariable classes

4.2.3.1. Index Properties

Gets the variable number as a Long type (4-byte integer type). This number indicates the number of the variable that CaoController class holds in the variable collection.

Usage example

```
Dim engine As CaoEngine          ' Engine
Dim workspace As CaoWorkspace            ' Workspace
Dim controller As CaoController  ' Controller

Set engine = New CaoEngine
Set workspace = engine.Workspaces.Item(0)
Set controller = workspace.AddController("Controller", _
                                          "CaoProv.TOYO-MM.PLCS12", _
                                          "", _
                                          "Conn=Tcp:192.168.1.201")

' Adding Variables
Dim variable As CaoVariable
Set variable = controller.AddVariable("@MAKER_NAME")

'Get Index
Dim index as Long
Index = variable.Index
```

4.2.3.2. Name Properties

Retrieves the variable name specified by AddVariable method of CaoContrller class.

Usage example

```
Dim engine As CaoEngine          ' Engine
Dim workspace As CaoWorkspace            ' Workspace
Dim controller As CaoController  ' Controller

Set engine = New CaoEngine
Set workspace = engine.Workspaces.Item(0)
Set controller = workspace.AddController("Controller", _
                                          "CaoProv.TOYO-MM.PLCS12", _
                                          "", _
                                          "Conn=Tcp:192.168.1.201")

' Adding Variables
Dim variable As CaoVariable
Set variable = controller.AddVariable("@MAKER_NAME")

Debug.Print variable.Name
```

4.2.3.3. Value Properties

Gets/sets the data. The behavior depends on the variable name. For details, refer to 4.3.

4.3. Variable list

Defines a list of variables that can be used in each class. Variables refer to objects of CaoVariable classes.

4.3.1. CaoController class-variable

Variable Name	Description	Value		Reference
		Get	Put	
@MAKER_NAME	Obtain the manufacturer's name.	✓	-	P.26
@VERSION	Get the DLL version.	✓	-	P.26
@LAST_ERROR	Gets/sets the last error that occurred.	✓	✓	P.27
@MOLDING_CONDITION	Get the forming conditions.	✓	-	P.28
@MOLDING_OPERATION_STATE	Used to acquire the molding operation status.	✓	-	P.28

4.3.1.1. @MAKER_NAME

Obtain the manufacturer's name.

Usage example

```
Dim engine As CaoEngine
Dim workspace As CaoWorkspace
Dim controller As CaoController
Set engine = New CaoEngine
Set workspace = engine.AddWorkspace("NewWrks", "")
Set controller = workspace.AddController("Controller", _
                                         "CaoProv.TOYO-MM.PLCS12", _
                                         "", _
                                         "Conn=Tcp:192.168.1.201")
' Add Variable
Dim var As CaoVariable
Set var = controller.AddVariable("@MAKER_NAME")
' Acquisition of Values
Dim strVal As String
StrVal = var.value
```

Data type

Data type	Description
VT_BSTR	Obtain the manufacturer's name.

4.3.1.2. @VERSION

Gets the DLL version.

Usage example

```
Dim engine As CaoEngine
Dim workspace As CaoWorkspace
Dim controller As CaoController
Set engine = New CaoEngine
Set workspace = engine.AddWorkspace("NewWrks", "")
```

```
Set controller = workspace.AddController("Controller", _
                                         "CaoProv.TOYO-MM.PLCS12", _
                                         "", _
                                         "Conn=Tcp:192.168.1.201")
' Add Variable
Dim var As CaoVariable
Set var = controller.AddVariable("@VERSION")
' Acquisition of Values
Dim value As String
Value = var.value
```

Data type

Data type	Description
VT_BSTR	Get the DLL version. *.*.*

4.3.1.3. @LAST_ERROR

Gets/sets the last error that occurred.

Usage example

```
Dim engine As CaoEngine
Dim workspace As CaoWorkspace
Dim controller As CaoController
Set engine = New CaoEngine
Set workspace = engine.AddWorkspace("NewWrks", "")
Set controller = workspace.AddController("Controller", _
                                         "CaoProv.TOYO-MM.PLCS12", _
                                         "", _
                                         "Conn=Tcp:192.168.1.201")
' Add Variable
Dim var As CaoVariable
Set var = controller.AddVariable("@LAST_ERROR")
' Acquisition of Values
Dim strVal As String
StrVal = var.value

' Value setting
Var.value = ("FFFFFFFF")
' Clear to zero
Var.value = ("00000000")
```

Data type

Data type	Description
VT_BSTR	Gets the last error that occurred. (8-digit hexadecimal string)

4.3.1.4. @MOLDING_CONDITION

Get the forming conditions.

Usage example

```
Dim engine As CaoEngine
Dim workspace As CaoWorkspace
Dim controller As CaoController
Set engine = New CaoEngine
Set workspace = engine.AddWorkspace("NewWrks", "")
Set controller = workspace.AddController("Controller", _
                                         "CaoProv.TOYO-MM.PLCS12", _
                                         "", _
                                         "Conn=Tcp:192.168.1.201")
' Add Variable
Dim var As CaoVariable
Set var = controller.AddVariable("@MOLDING_CONDITION")
' Acquisition of Values
Dim conditions As String()
Conditions = var.value
```

Data type

Data type		Description
VT_BSTR | VT_ARRAY		The forming conditions are retained.
	$0 \leq i \leq 75$	Injection setting
	$76 \leq i \leq 132$	Weighing setting
	$133 \leq i \leq 163$	Temperature setting
	$164 \leq i \leq 221$	OPEN/CLOSE/EJ SETTING
	$222 \leq i \leq 250$	Model Thickness/Timer Counter
	$251 \leq i \leq 284$	Unit setting

4.3.1.5. @MOLDING_OPERATION_STATE

Used to acquire the molding operation status.

Usage example

```
Dim engine As CaoEngine
Dim workspace As CaoWorkspace
Dim controller As CaoController
Set engine = New CaoEngine
Set workspace = engine.AddWorkspace("NewWrks", "")
Set controller = workspace.AddController("Controller", _
                                         "CaoProv.TOYO-MM.PLCS12", _
                                         "", _
                                         "Conn=Tcp:192.168.1.201")
' Add Variable
Dim var As CaoVariable
Set var = controller.AddVariable("@MOLDING_OPERATION_STATE")
' Acquisition of Values
```

```
Dim states As String()
States = var.value
```

Data type

Data type		Description
VT_BSTR | VT_ARRAY		Obtain the manufacturer's name.
	0	Date
	1	Time
	2	Model name
	3	Operation mode code
	4	Operation mode name
	5	Product Name
	6	Mold No.
	7	Number of shots
	8	Cycle timer
	9	Operating time
	10	Stop time
	11	Bad number
	12	Failure rate
	13	Production volume
	14	Production rate
	15	Occupancy rate
	16	Progress
	17 ~ 31	Reserved

4.4. Event list

You can receive controller error notifications and status changes as OnMessage events.

The received message stores the identifier of the communication command in Number property, and the data string associated with the identifier in Value property. The data column is a string array (VT_BSTR | VT_ARRAY) decomposed by the delimiter (,) in the data column.

For more information on identifiers and data strings, refer to PLCS-12 Ethernet Communication Specifications.

Here is the message number to receive:

Message number	Description
30	Alarm data
32	Mode data
33	Monitor data 1
34	Monitor data 2
36	Molding conditions
46	Molding operation status
54	Set value history data
55	Semi-fixed history data
999	Communication error

5. PLCS12 Provider Error Codes

This provider has the following unique error codes masked with the 0x8011****. (Refer to Table 5-1.)

Refer to the Error Codes section of ORiN2 Programming Guide C:¥ORiN2¥CAO¥Doc¥ORiN2_ProgrammersGuide_ja.pdffor information on common ORiN2 errors.

Table 5-1 Unique Error Codes

Error Number	Description
0x80110001	The received packet is invalid.
0x80110002	The checksum of the received packet is invalid.
0x80110003	Failed to create mutex.

Appendix A. Correspondence Table with Communication Commands

Variable

Variable	Communication command identifier
@MOLDING_CONDTION	On request: 35
	Receive: 36
@MOLDING_OPERATION_STATE	On request: 45
	Receive: 46

Message

Message	Communication command identifier
Alarm data	30
Mode data	32
Monitor data 1	33
Monitor data 2	34
Molding conditions	36
Molding operation status	46
Set value history data	54
Semi-fixed history data	55