RC8 Provider

for DENSO Robot RC8

Version 1.1.1

User's Guide

October 31, 2012

[Remarks]

[Revision History]

Version	Date	Content
1.1.0	2012-09-10	First edition.
1.1.1	2012-10-31	Hand object wad added.

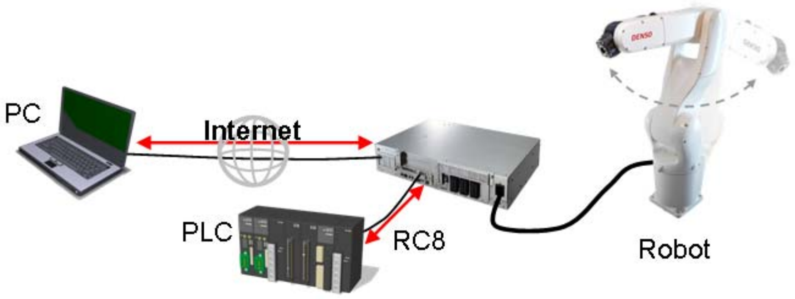
[Hardware]

Model	Version	Notes
RC8	1.2.4	

Contents

1. Introduction

This document describes external specifications of the Cao provider for RC8 controller of the Denso robot compliant to ORiN Ver. 2 specifications. In this document, Cao provider for RC8 is called RC8 provider.
This document describes the RC8 provider specifications on connection procedures, variables, I/O access, file manipulation, task control, robot control, hand control and original enhancement.



1.1. System requirements and versions assumed in this document

As the system requirements, the client PC is assumed to run on Windows and the target robot controller is assumed to be RC8 or later.
The required development environment on the PC is a programming environment that supports Component Object Model (COM).

1.2. Information sources for your reference

Although the programming examples in this document are written in Visual Basic 6.0, development is possible using various programming languages such as C++, Java, .NET, LabVIEW, and Delphi. For details about usages, refer to the "ORiN 2 Programming Guide".

"ORiN2 Programming Guide" is provided as the following file in the ORiN2 SDK installation folder.

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

* Read the <lang> part as characters that represent the language used in each environment.
This guide describes with examples the basic knowledge and technology of ORiN2 and COM/DCOM required to develop an application using the provider.

Refer also to the following documents if required.

b-CAP Provider User's Guide

・ORiN2¥CAO¥ProviderLib¥b-CAP¥Doc¥b-CAP_ProvGuide_<lang>.pdf

NetwoRC Provider User's Guide (Provider for RC7 Controller)

・ORiN2¥CAO¥ProviderLib¥DENSO¥NetwoRC¥Doc¥NetwoRC_ProvGuide_<lang>.pdf

2. Environment Setup for Application Development

2.1. Setup of PC development environment

2.1.1. Automatic installation of RC8 provider

With ORiN2 SDK Ver 2.1.9 or later, RC8 provider is set up by an installer.

If ORiN2 SDK Ver 2.1.9 or later is installed, the operation environment (runtime) for connecting to the RC8 robot controller (hereinafter referred to as the robot controller) is ready.

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

2.1.2. Manual installation of RC8 provider

To set up RC8 provider without using the installer, registry need to be manually registered according to the table below.

Table 2-1 RC8 provider

File name	CaoProvRC8.dll
ProgID	CaoProv.DENSO.RC8
Registry registration	Regsvr32 CaoProvRC8.dll
Remove registry registration	Regsvr32 /u CaoProvRC8.dll

A license key is required to use the Cao Engine module. Refer to “License registration” section of "ORiN2 SDK User's Guide".

2.2. Setup of RC8 controller

2.2.1. Emergency stop device position

A robot emergency stop switch should be prepared and set up near a robot operator before operating the robot, so that the switch can immediately stop the robot motion in an emergency situation.

(1) The emergency stop switch should be red-colored.

(2) After the emergency stop switch is activated, the switch should not return to normal (robot operating) position automatically or by other operator’s careless action.

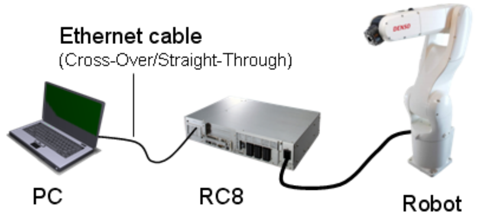
(3) A robot emergency stop switch should be set up separately from the power switch.

2.2.2. Preparation of hardware

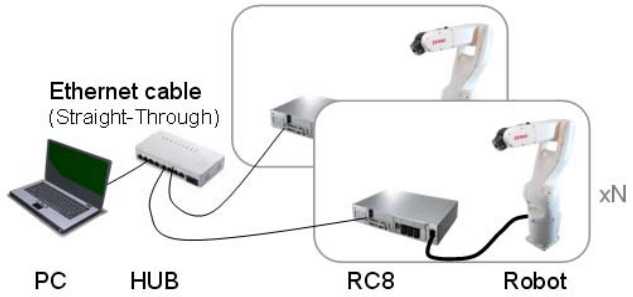
The following shows the basic hardware configurations that can be used for the robot controller clients. When designing equipment, consider the system configuration for the software required by the customer and prepare hardware accordingly.

(1) PC-based robot system

- Configuration with one RC8 unit



- Configuration with more than one RC8 unit



(2) RC8-based robot system

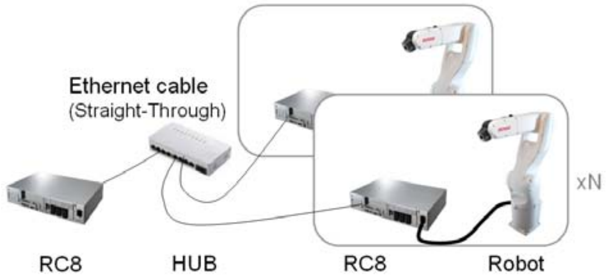


Table 2-2 Configurations of robot systems

Hardware		Software			
Client	Connection type	OS	Programming language	Dependent module	Remarks
(1) PC-based	Ethernet (TCP/IP)	Windows	C,C++,C#,VB,VBA, Java,LabVIEW Delphi, Python, Ruby,... (Environment that supports DCOM technology)	ORiN2 SDK (Cao, RC8/ b-CAP/VRC providers)	- Using ORiN2 technology, all APIs supported by RC8 are available for use. - ORiN2 SDK is required for the client PC.
		Linux (others)	C, C++ (Environment that supports socket communications)	Socket library	- All APIs supported by RC8 are available for use because b-CAP protocol is supported using the socket communications technology.
(2) RC8-based	Ethernet, I/O	RC8-dependent Windows	PacScript (VBA-based)	Standard equipment of ORiN2 SDK (Cao and RC8/b-CAP/VRC providers)	- With the standard equipment functions, All APIs supported by RC8 are available for use.

2.2.3. Setup of system parameters

Before using the RC8 provider, the robot controller to be controlled must be set up.

Either a teach pendant (TP) or mini pendant (MiniTP) is required to set up the system parameters. The systems parameters that need to be set up are (1) communication permission and (2) activation authority.

A communication permission is provided to assign data read and write permissions to a robot controller. Assign a write permission in order to write variable data or control a robot.

An activation authority is a setting used to assign a communication device the authority to activate (run) a program task on the robot controller, turn ON the motor, and control the robot (motion command). Either (1) TP, (2) I/O, (3) Ethernet, or (4) Any can be set. Setting "Any" gives activation authority regardless of the communication routing. When setting "Any," execute exclusive processing between communication devices to prevent collisions between the client PCs and PLCs.

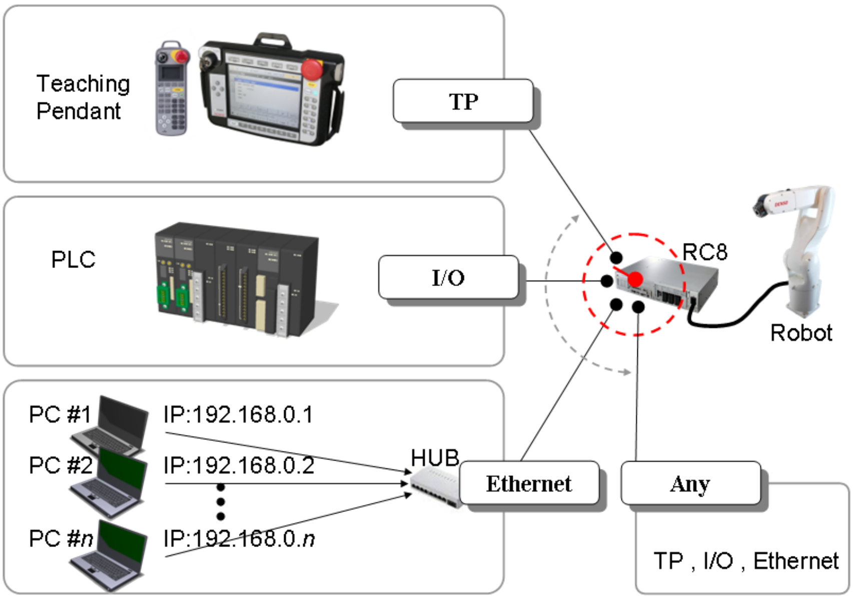


Figure 2-1 Setup of devices with activation authority

When using Ethernet as the connection method, the IP addresses of client PCs must be set. When this setting is selected, the robot controller allows only specific client PCs to activate a program task or control the robot.

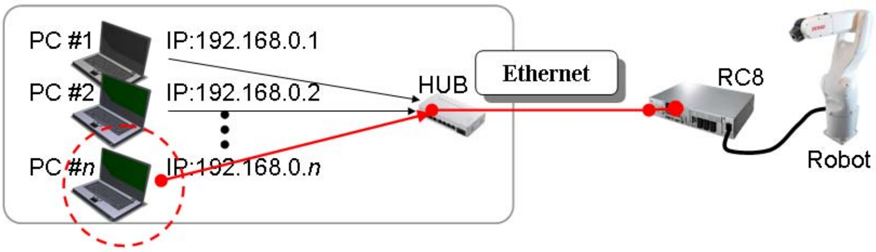


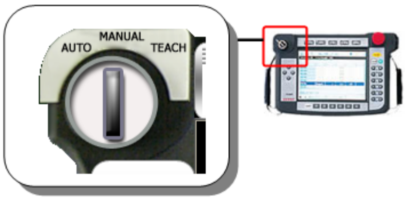
Figure 2-2 Setup of clients with activation authority

The following sections describe the setup methods using each of these settings.

2.2.3.1. Setup using a teach pendant

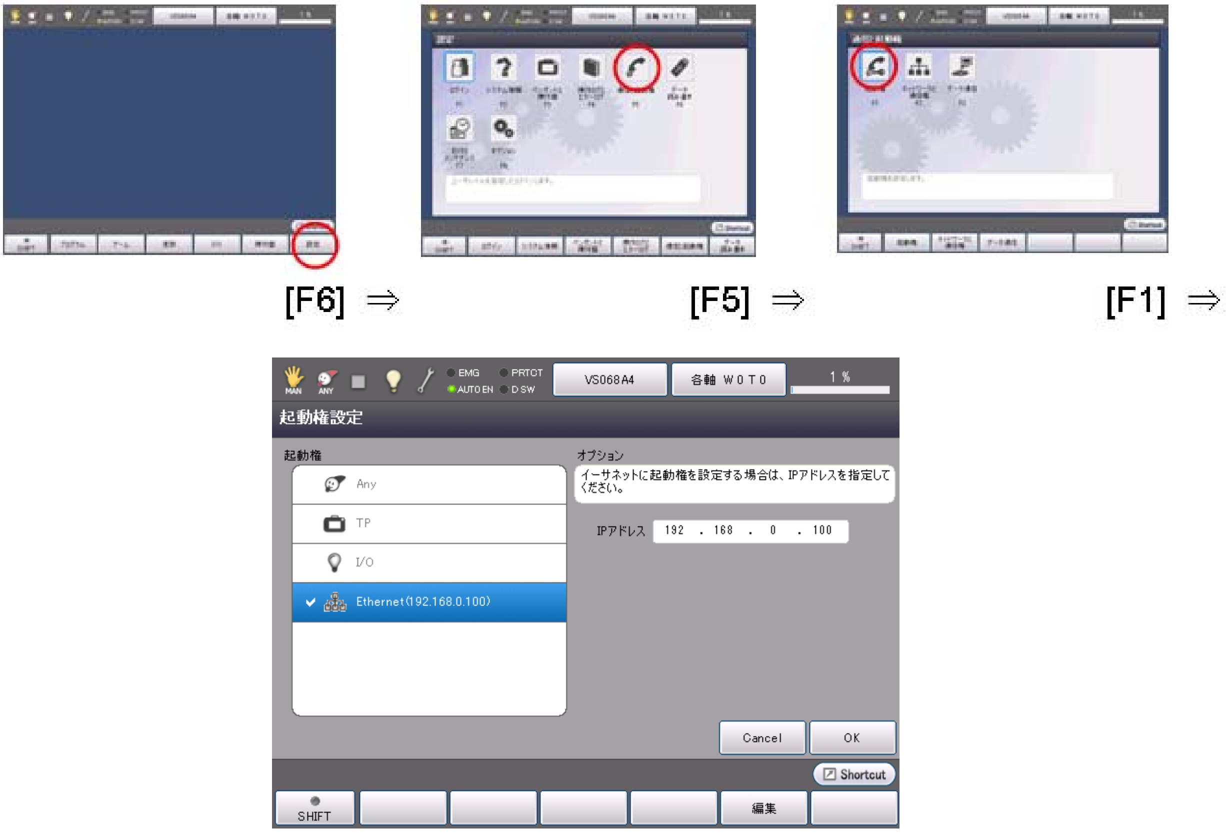
Set the IP address of a robot controller using a teach pendant according to the following procedure.

(1) **<u>Set</u>** the robot controller **<u>to the Manual mode</u>**.



(2) **<u>Set the activation authority</u>** of the robot controller.

To use Ethernet, select the teach pendant's [F6 Setup] menu -> [F5 Communication and Token] -> [F1 Executable Token] and set the activation authority to Ethernet.



起動権設定

起動権

- Any
- TP
- I/O
- ✓ Ethernet(192.168.0.100)

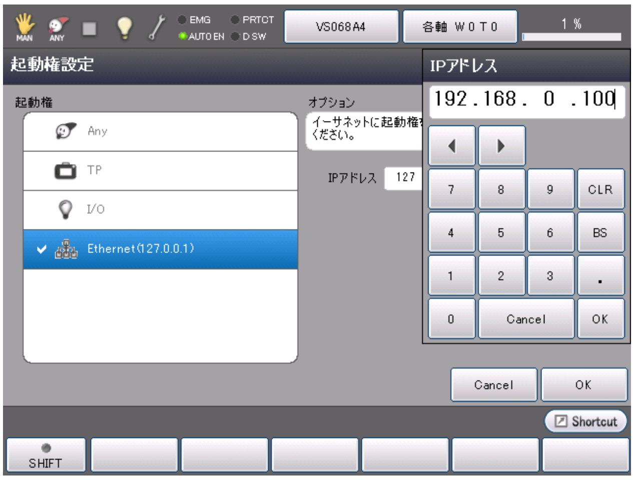
オプション

イーサネットに起動権を設定する場合は、IPアドレスを指定してください。

IPアドレス 192 . 168 . 0 . 100

Figure 2-3 Setting of activation authority

Then, press [F5 Edit] and set the IP address of the client that assigns activation authority to the robot controller.



起動権設定

起動権
- Any
- TP
- I/O
- ✓ Ethernet(127.0.0.1)

オプション
イーサネットに起動権を…ください。

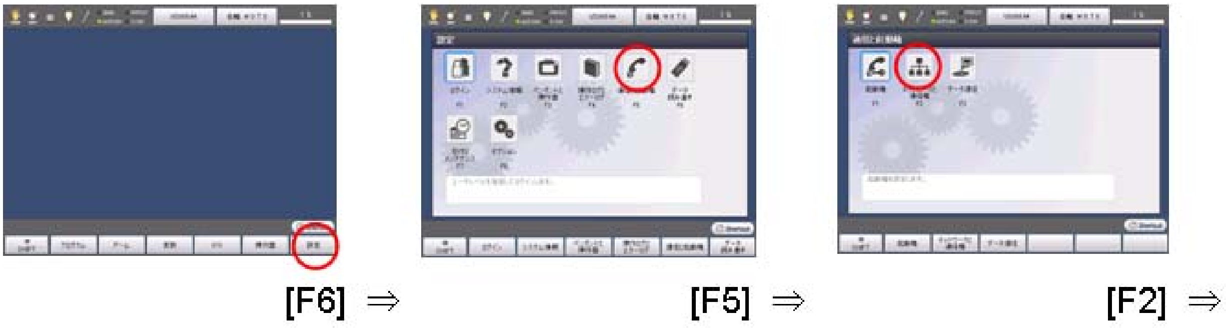
IPアドレス 127

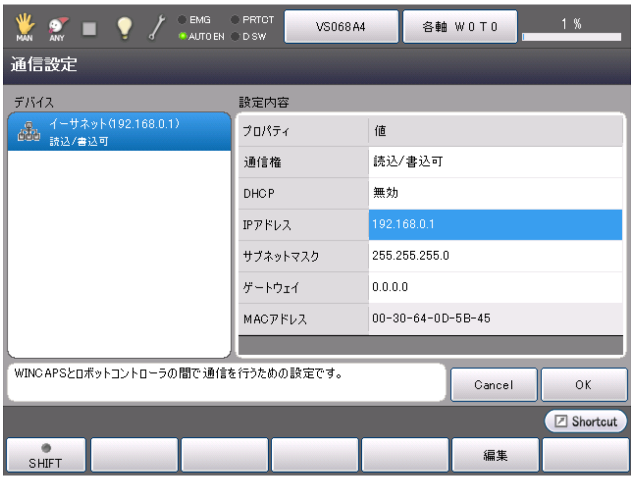
IPアドレス 192.168. 0 .100

Figure 2-4 Setting of IP addresses of clients

(3) Set the network and communication permissions of the robot controller.

To use Ethernet, select the teach pendant's [F6 Setup] menu -> [F5 Communication and Token] -> [F2 Network and Permission] and set the read/write permissions to Ethernet.





通信設定

デバイス
イーサネット(192.168.0.1)
読込/書込可

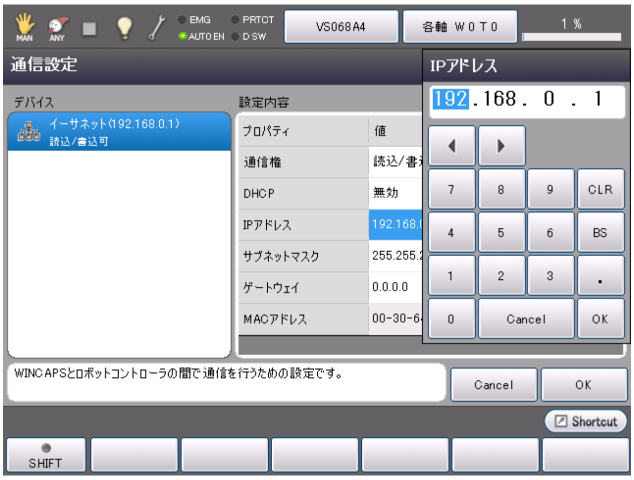
設定内容

プロパティ	値
通信権	読込/書込可
DHCP	無効
IPアドレス	192.168.0.1
サブネットマスク	255.255.255.0
ゲートウェイ	0.0.0.0
MACアドレス	00-30-64-0D-5B-45

WINCAPSとロボットコントローラの間で通信を行うための設定です。

Figure 2-5 Communication settings

Next, press [F5 Edit] and set the IP addresses and subnet masks of the robot controllers. Set the gateway address if required.



192.168. 0 . 1

プロパティ	値
通信権	読込/書
DHCP	無効
IPアドレス	192.168.
サブネットマスク	255.255.
ゲートウェイ	0.0.0.0
MACアドレス	00-30-6

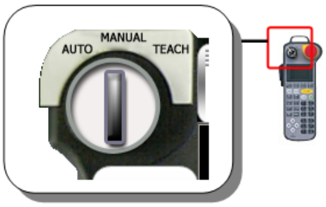
WINCAPSとロボットコントローラの間で通信を行うための設定です。

Figure 2-6 Setting of IP addresses of robot controllers

2.2.3.2. Setup using a mini pendant

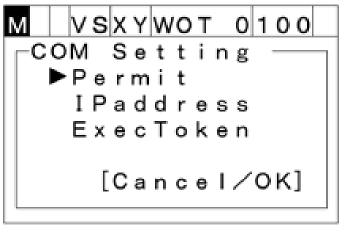
Set the IP address of a robot controller using a mini pendant according to the following procedure.

(1) **Set** the robot controller **to the Manual mode**.



(2) **Set the activation authority** of the robot controller.

Press [COM] to display the [COM Setting] screen shown below which lists communications settings for the robot controller.



```
M  VSXYWOT 0100
COM Setting
 ▶Permit
  IPaddress
  ExecToken

    [Cancel/OK]
```

Figure 2-7 List of communications settings

Select "Exec Token" with up and down cursor keys, and then press [OK] to display the [Exec Token] screen shown below which lists activation authority settings.

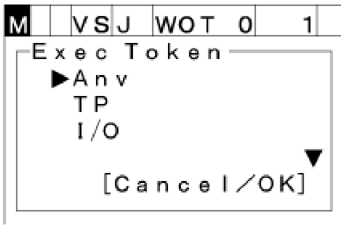
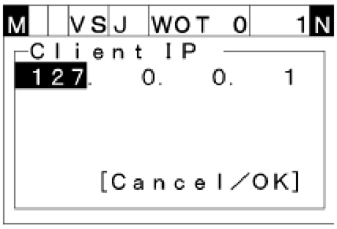


Figure 2-8 List of activation authority settings

Select "Ether" with up and down cursor keys, and then press [OK]. The [Client IP] screen is displayed as shown below. Set the IP address of the client that assigns activation authority to the robot controller.



Client IP
127. 0. 0. 1

Figure 2-9 Setting of IP addresses of clients

Press [OK] to confirm the change.
Press [Cancel] to cancel the change.

(3) **Set the communication permission** of the robot controller.
Press [COM] to display the [COM Setting] screen shown below which lists communications settings for the robot controller.

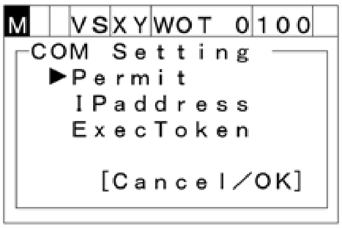
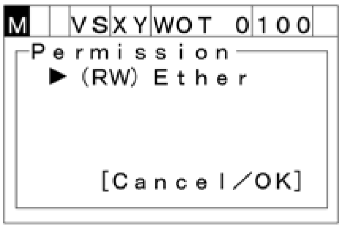


Figure 2-10 List of communications settings

Select "Permit" with up and down cursor keys, and then press [OK] to display the [Permission] screen which lists port options as shown below.
(Off): Not available, (R): Read only, (RW): Read/write available
Press [Cancel] to exit the communications setting.



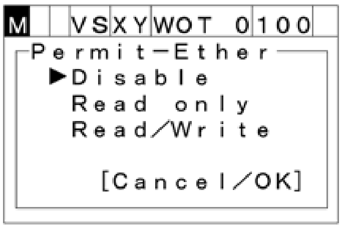
```
M    VSXYWOT 0100
Permission
 ► (RW) Ether

        [Cancel/OK]
```

Figure 2-11 List of port options

Select "Ether" and press [OK]. The [Permit-Ether] screen is diplayed as shown below which lists communication options.

Press [Cancel] to exit the communications setting.



```
M    VSXYWOT 0100
Permit-Ether
 ►Disable
  Read only
  Read/Write

        [Cancel/OK]
```

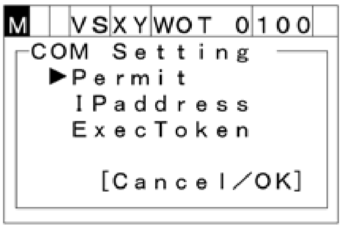
Figure 2-12 List of communication options

Using the up and down cursor keys, select one of "Disable," "Read only," and "Read/write" and press [OK] to change the communication permission.

Press [Cancel] to cancel the change of the communication permission.

(4) **Set the network** of the robot controller.

Press [COM] to display the [COM Setting] screen shown below which lists communications settings.



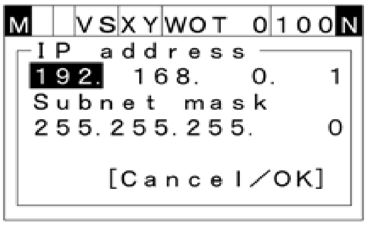
```
M    VSXYWOT 0100
COM Setting
 ►Permit
  IPaddress
  ExecToken

        [Cancel/OK]
```

Figure 2-13 List of communications settings

Using the up and down cursor keys, select "IP address" and press [OK] to display the [IP address] setting screen as shown below.

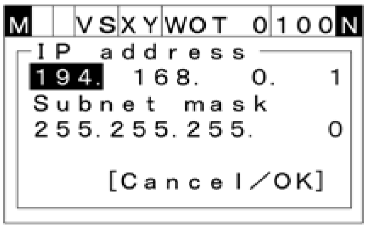
Press [Cancel] to exit the communications setting.



IP address
192. 168. 0. 1
Subnet mask
255.255.255. 0

Figure 2-14 IP address setting screen

Select an item using the up/down/left/right cursor keys. The value can be changed using the numeric entry keys.



IP address
194. 168. 0. 1
Subnet mask
255.255.255. 0

Figure 2-15 Change of IP addresses

Press [OK] to confirm the change.
Press [Cancel] to cancel the change.

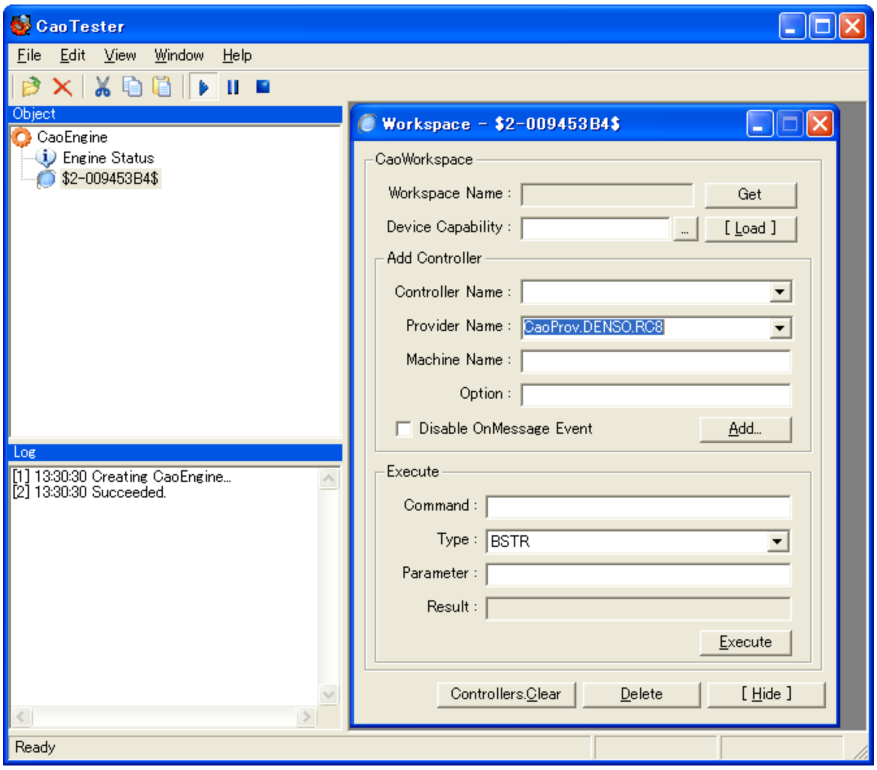
2.3. Operation check using CaoTester

Before running a developed client application, check that the RC8 robot controller to be controlled has been set up correctly using CaoTester, an ORiN2 SDK standard tool.

2.3.1. Check of variable access

Perform the variable access operation using CaoTester and check that the client PC has a basic connection with the target robot controller according to the procedure shown below. If this operation cannot be correctly performed, the client PC installation environment or the network environment and settings of the target robot controller may be faulty and therefore perform setup again.

(1) Activate CaoTester.
To activate CaoTester, select [ORiN2¥CAO¥Tools¥CaoTester¥Bin¥CaoTester.exe] in the ORiN2 SDK installation folder.



```
[1] 13:30:30 Creating CaoEngine...
[2] 13:30:30 Succeeded.
```

Provider Name : CaoProv.DENSO.RC8

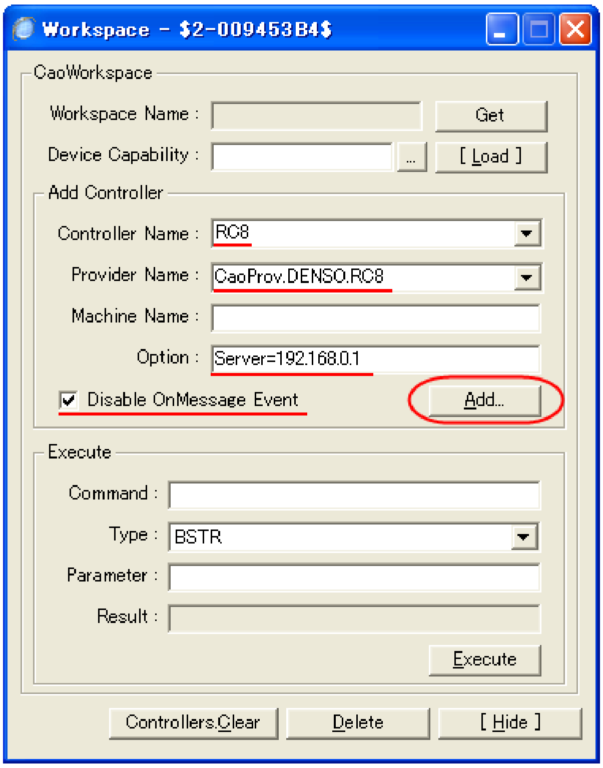
Type : BSTR

Figure 2-16 Initial screen of CaoTester

(2) Select the [Workspace] window and set parameters in [Add Controller].

For the purpose of explanation, the target controller is assumed to have an IP address of 192.168.0.1. Read the settings as those in your actual environment.

Controller Name	: RC8	
Provider Name	: CaoProv.DENSO.RC8	
Machine Name	: <Blank>	
Option	: Server=*192.168.0.1*	* IP address of the target controller



Controller Name : RC8

Provider Name : CaoProv.DENSO.RC8

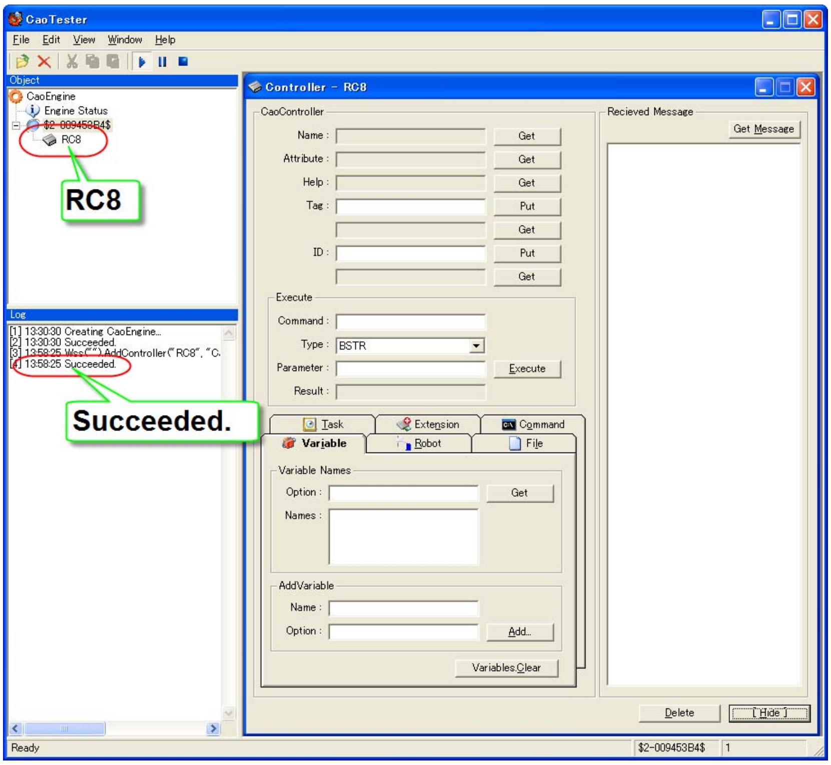
Option : Server=192.168.0.1

☑ Disable OnMessage Event

Type : BSTR

Figure 2-17 [Workspace] window

(3) Press the [Add] button in the [Workspace] window to display the [CaoController] window.



Type : BSTR

```
[1] 13:30:30 Creating CaoEngine...
[2] 13:30:30 Succeeded.
[3] 13:58:25 Wss("").AddController("RC8", "C.
[4] 13:58:25 Succeeded.
```

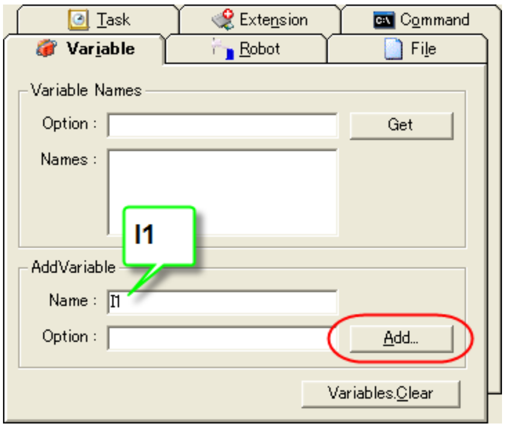
Figure 2-18 [CaoTester] screen while creating [Controller] window

(4) In the [Controller] window, access the [Variable] tab and create a [Variable] window for I1 variable in [AddVariable].

Name : I1

Option : <Blank>

In [AddVariable], set the parameters shown above and press the [Add..] button.



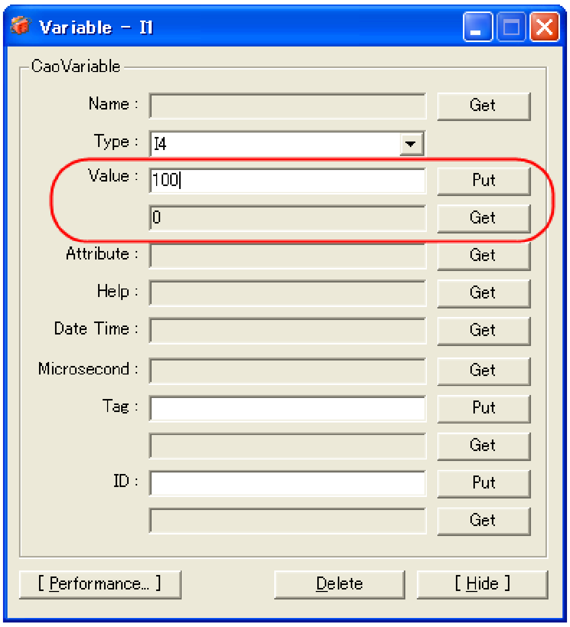
Name : I1

Option :

Figure 2-19 [Variable] tab settings

(5) Access the variable in [Value] in the [Variable] window.

Press the [Get] and [Put] buttons to access the value of the target controller.



Type : I4

Value : 100

0

Figure 2-20 [Value] setting in [Variable] window

2.3.2. Check that the motor is ON

Turn ON and OFF the motor power using CaoTester and check that the client PC can control the motor power with the target robot controller according to the procedure shown below. If this operation cannot be correctly performed, the activation authority of the target robot controller may not be correctly set on the client PC and therefore check the activation authority setting again.

(6) **Set** the robot controller **to the Auto mode**.

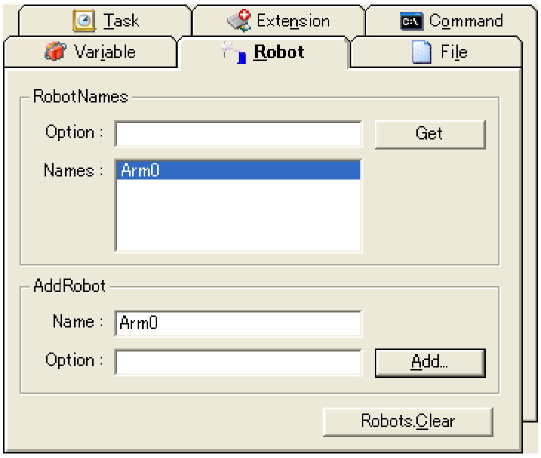
AUTO
MANUAL
TEACH

(7) Select the [Controller] window of CaoTester, access the [Robot] tab, and create a [Robot] window.

Name : Arm0

Option : <Blank>

In [AddRobot], set the parameters shown above and press the [Add..] button.



RobotNames

Option :

Names : Arm0

AddRobot

Name : Arm0

Option :

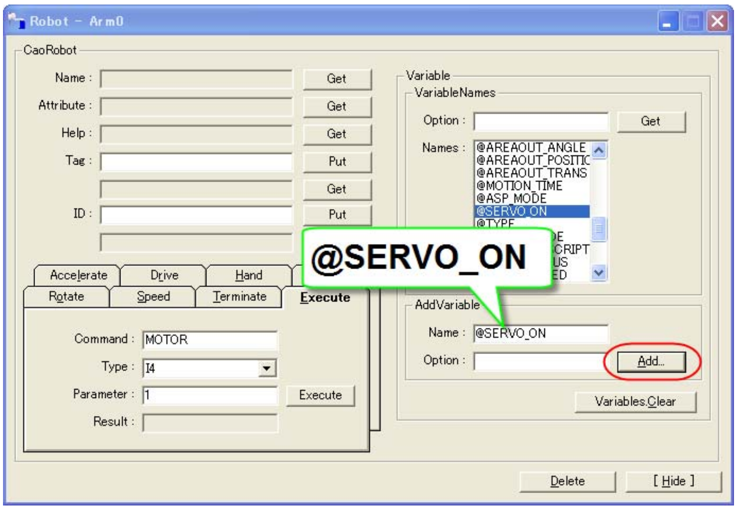
Figure 2-21 [Robot] tab settings

(8) In the [Robot] window, access the [Variable] tab and create a [Variable] window for @SERVO_ON in [AddVariable].

Name : @SERVO_ON

Option : <Blank>

In [AddVariable], set the parameters shown above and press the [Add..] button.



Robot - Arm0

CaoRobot

Command : MOTOR

Type : I4

Parameter : 1

Variable

VariableNames

Names : @AREAOUT_ANGLE, @AREAOUT_POSITIC, @AREAOUT_TRANS, @MOTION_TIME, @ASP_MODE, @SERVO_ON, @TYPE

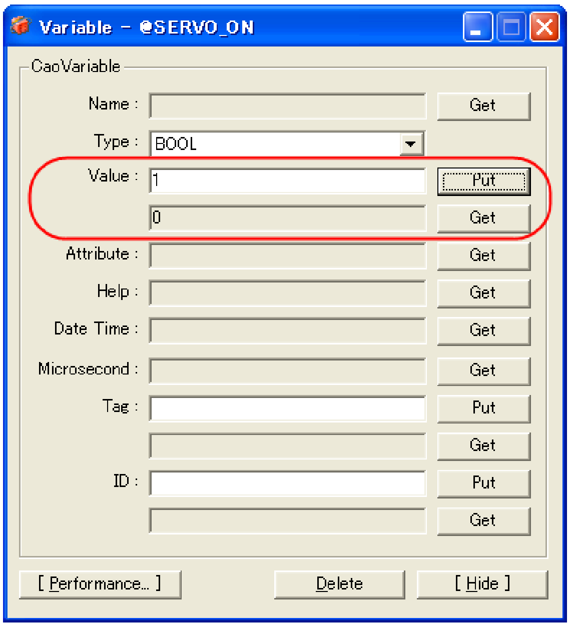
AddVariable

Name : @SERVO_ON

Figure 2-22 [Robot] window settings

(9) Turn ON or OFF the motor power in [Value] in the [Variable] window.

Press the [Get] and [Put] buttons to turn ON (1) and OFF (0) the motor power of the target controller.



Variable - @SERVO_ON

CaoVariable

Type : BOOL

Value : 1 (Put)

0 (Get)

Figure 2-23 [Value] setting in [Variable] window

3. Basic Knowledge on RC8 Programming

3.1. Outline of RC8 provider

3.1.1. Functions provided by RC8 provider

The RC8 provider provides a wide range of APIs compliant with ORiN2 to enable calling of all the functions provided by the robot controller to external devices.

The following table shows the outline of functions provided by the RC8 provider. For the details, refer to "5. Command Reference".

Table 3-1 Outline of RC8 provider functions

Function name	Category	Remarks
Event notification	CaoController	Can receive error notifications and status changes of the controller as OnMessage event asynchronously.
Variables access	CaoVariable	Can read/write I/O, global variables, and local variables as well as system parameters. Can also acquire information and statuses of a wide range of controller resources.
File manipulation	CaoFile	Can acquire information on and manipulate files and folders.
Task control	CaoTask	Can control the status acquisition, activation, and stop of tasks to be executed. Also can perform task-to-task communications using message queues of tasks.
Robot control	CaoRobot	Can control robots using turn ON/OFF of motor power, operation speeds/operation commands of robots, and TOOL/WORK/AREA settings, etc.

3.1.2. System configuration of RC8 provider

The RC8 provider is a core module independent of the hardware of the robot controller.

The following shows the system configuration for connecting a PC and an RC8 robot controller.

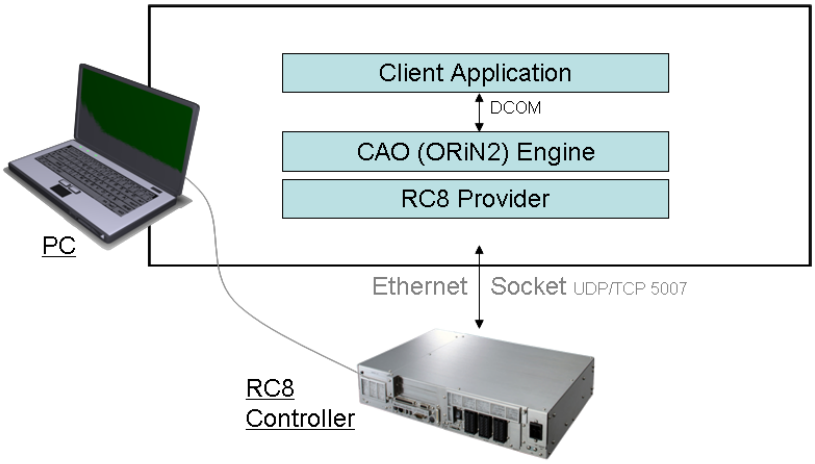


Figure 3-1 System configuration of PC and RC8

3.1.2.1. Configuration of Cao engine and Cao provider

Cao providers such as the RC8 provider are plug-ins of the Cao engine of ORiN2. Therefore, understanding of class configuration of the Cao engine is required to create a client application.

The following figure shows the class configuration of the Cao engine and the Cao provider.

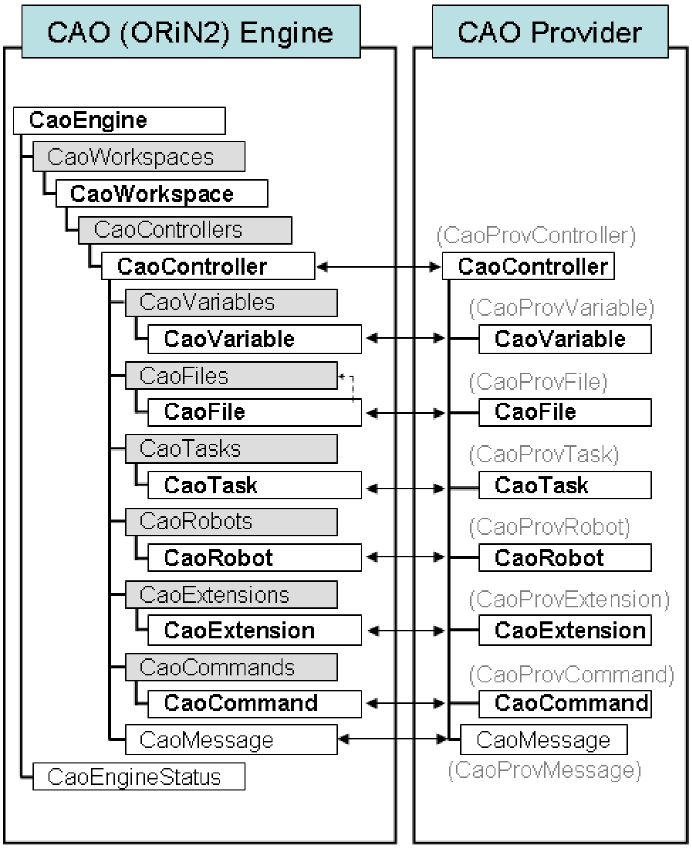


Figure 3-2 Configuration of Cao engine and provider

The class configuration of the Cao engine is a model of resources owned by general devices including robot controllers. A client application, by accessing the classes provided by the Cao engine, can indirectly access the devices to be connected.

3.1.3. HRESULT and handling of errors

If a value of HRESULT that represents a response of the methods and properties of classes of the Cao provider is 0 or higher, it means that the processing has been successfully completed. On the other hand, a negative value signifies that the call failed.

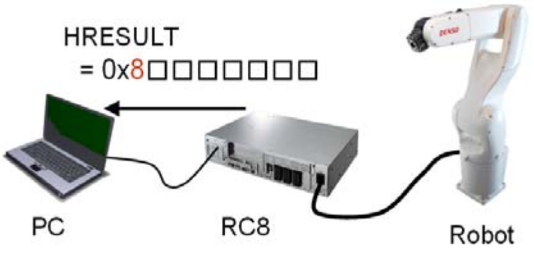


Figure 3-3 Error in call from PC

If the error of HRESULT is 0x8□□□□□□□, look up the error in the table of error codes in the manual provided with the robot controller.

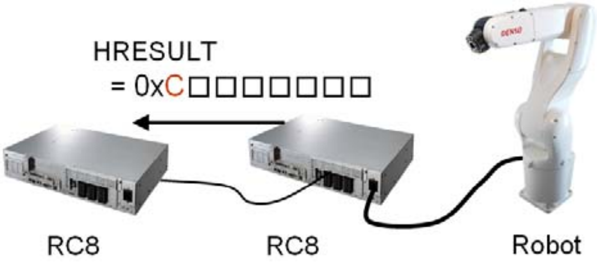


Figure 3-4 Error in call from RC8

If the error of HRESULT is 0xC□□□□□□□, read 0xC as 0x8 and look up the error in the table of error codes in the manual provided with the robot controller.

3.1.4. Handling of property definitions

For the purpose of explanation of the property specifications of classes of the Cao provider, the following conventions are used throughout this manual.

Property acquisition <Variable to be substituted> = Obj.PropertyName

Handled as <Variable to be substituted> = Obj.get_Property()

get_ PropertyName	Acquisition of the value of <PropertyName> property

Property setting Obj.PropertyName = <Setting value>

Handled as Obj.put_Property (<Setting value>).

put_ PropertyName	Setting of the value of <PropertyName> property

3.1.5. Execute method and runtime binding

If a method not defined in the target class is called using the runtime binding function, the Execute method is automatically called according to the following specifications:

```
vntRet = Obj.CommandName Param1, Param2, ...
  ↓
vntRet = Obj.Execute("CommandName", Array(Param1, Param2, ... ) )
```

1. The command name is passed as a BSTR string to the first argument.
2. All the parameters are passed as a VARIANT array to the second argument.

To realize these specifications, the Execute method of classes of the Cao provider is defined as follows:

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

bstrCmd	:	[in]	Command name, BSTR type string
vntParam	:	[in]	Parameter, VARIANT type array (or singular)
vntRet		[out]	Return value, VARIANT type

The arguments of the Execute method specify a command as a BSTR and a parameter as a VARIANT array.

4. RC8 Programming Using the Provider

To perform robot control with RC8 provider, communication between an ORiN installed PC and the robot controller should be established with Ethernet. Some commands also require the robot controller setup. For details of setup, refer to "2 Environment Setup for Application Development" and for details of commands, refer to "5 Command Reference".

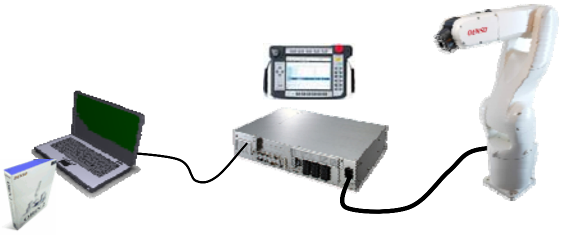


Figure 4-1 Robot connection

The developed program uses RC8 provider to communicate with the robot controller, by generating a socket (UDP/TCP).

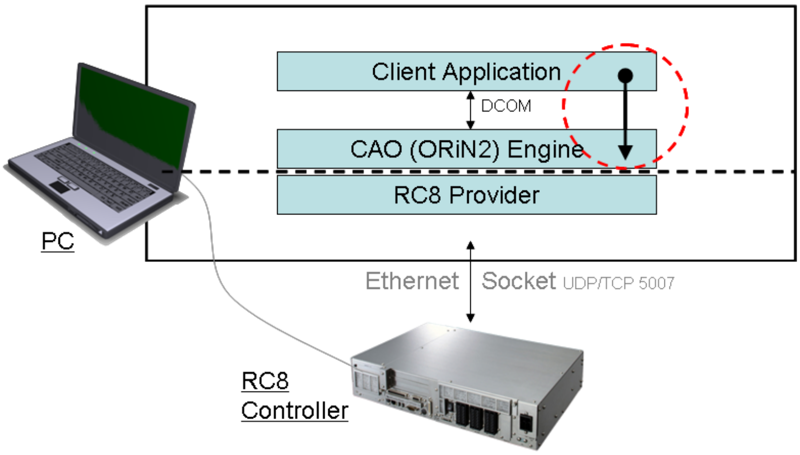


Figure 4-2 Outline of programing

RC8 provider establishes communication between the PC and the robot controller by the following procedure:

- Create CaoEngine
- Create CaoWorkspace
- Create CaoController

After the communication is established, variables in the controller will be accessed by creating a CaoVariable object, and robot motions will be initiated by creating a CaoRobot object. Examples in the following section explain the procedure of robot programming.

4.1. RC8 controller variable access

Figure 4-3 shows the procedure to access variables.

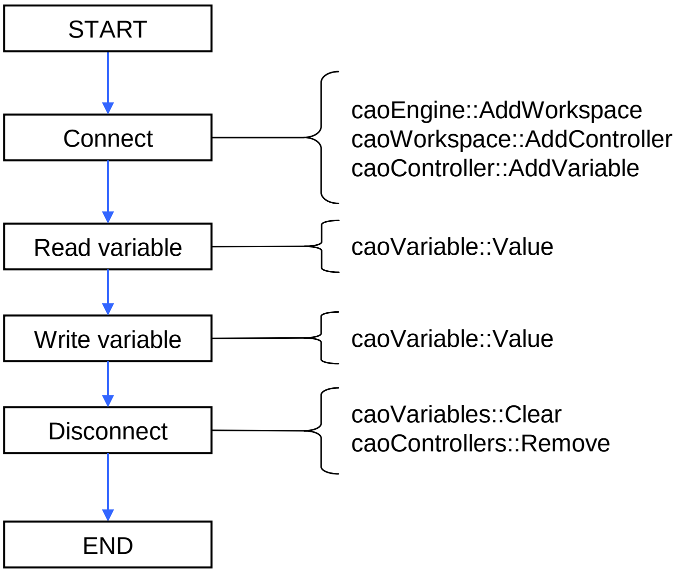


Figure 4-3 Variable access

4.1.1. Connection

Following is the procedure to establish connection to the robot controller.

(1) Create variables to store objects. CaoEngine object, CaoWorkspace object, and CaoController object are required to establish communication to the robot controller. CaoWorkpace object does not need to prepare a variable if CaoController object is acquired from CaoWorkspaces. CaoVariable object is also necessary to access variables. Following is an example code in VB6.

```
Dim g_eng as CaoEngine              ' CaoEngine object variable
Dim g_wrks as caoWorkspace          ' CaoWorkspace object variable
Dim g_ctrl as CaoController         ' CaoController object variable
Dim g_val as CaoVariable            ' CaoVariable object variable
```

(2) Create a CaoEngine object. CaoEngine object is created with New keyword.

```
Set g_eng = New CaoEngine           ' CaoEngine object creation
```

(3) Acquire or create a CaoWorkspace object. When created, CaoEngine object automatically creates one Caoworkspaces object and one Caoworkspace object. The next sample program uses the automatically created workspace. Following is an example code for creating a new CaoWorkspace object.

```
        Set g_wrks = g_eng.Addworkspace("NewWrks", "")
```

(4) Create a CaoController object. To create a CaoController object, specify the provider name and its parameters. RC8 provider specifies the destination controller IP address as an option. Following is an example code.

```
        Set g_ctrl = g_wrks.AddController("RC8", "CaoProv.DENSO.RC8", "", "Server=192.168.0.1")
```

(5) Create a CaoVariable object. Create a CaoVariable object for the connected variable. Following is an example code for accessing the 10th element of P-type variable.

```
        Set g_val = g_ctrl.AddVariable("P10", "")
```

4.1.2. Variable read/write access

To read and write the connected variable value, use Value property of CaoVariable object. To read and write value, another variable with the suitable type for the connected variable should be prepared. Following is an example code.

```
        Dim vntPotision as Variant
        vntPotision = g_val.Value                         ' Get value
        g_val.Value = Array(50, 50, 50, 0, 0, 0, -1)      ' Set value
```

4.1.3. Disconnection

To disconnect from the controller, delete not only the created object itself, but also delete the object from a collection class that manages the object. Following is an example code.

```
        g_ctrl.Variables.Clear                        ' Delete all objects from CaoVariables
        Set g_val = Nothing                           ' Delete CaoVariable
        g_wrks.Controllers.Remove g_ctrl.Index        ' Delete CaoController from CaoControllers
        Set g_ctrl = Nothing                          ' Delete CaoCtonroller
        g_eng.Workspaces.Remove g_wrks.Index          ' Delete CaoWorkspace from CaoWorkspaces
        Set g_wrks = Nothing                          ' Delete CaoWorkspace
        Set g_eng = Nothing                           ' Delete CaoEngine
```

4.1.4. Sample program

Following is an example program written in VB6. The sample program uses the automatically created workspace and reads/writes the variable IO150 (the 150th I/O variable). IP should be set to the value for the target controller. This sample program uses following value.

IP:192.168.0.1

List 4-1 **Variable.frm**

```
Dim g_eng As CaoEngine
Dim g_ctrl As CaoController
Dim g_val As CaoVariable

Private Sub Command1_Click()
    ' Read variable
    Text1.Text = g_val.Value
End Sub

Private Sub Command2_Click()
    ' Write variable
    g_val.Value = CBool(Text2.Text)
End Sub

Private Sub Form_Load()
    Set g_eng = New CaoEngine

    ' Connect RC: IP setting depends on your RC setting.
    Set g_ctrl = g_eng.Workspaces(0).AddController("RC8", "CaoProv.DENSO.RC8", "",
"Server=192.168.0.1")

    ' Variable name "IO150"
    Set g_val = g_ctrl.AddVariable("IO150", "")

End Sub

Private Sub Form_Unload(Cancel As Integer)
    ' Destroy variable object
    g_ctrl.Variables.Clear
    Set g_val = Nothing

    ' Destroy controller object
    g_eng.Workspaces(0).Controllers.Remove g_ctrl.Index
    Set g_ctrl = Nothing

    ' Destroy CaoEngine
    Set g_eng = Nothing
End Sub
```

4.2. Task control with RC8 controller

To perform task control, perform the processing described in Figure 4-4. To run a task, the controller must be in AUTO mode. Furthermore, the activation authority of the controller must be set to the IP of an ORiN installed PC. For further details, refer to "2.2.3 Setup of system parameters".

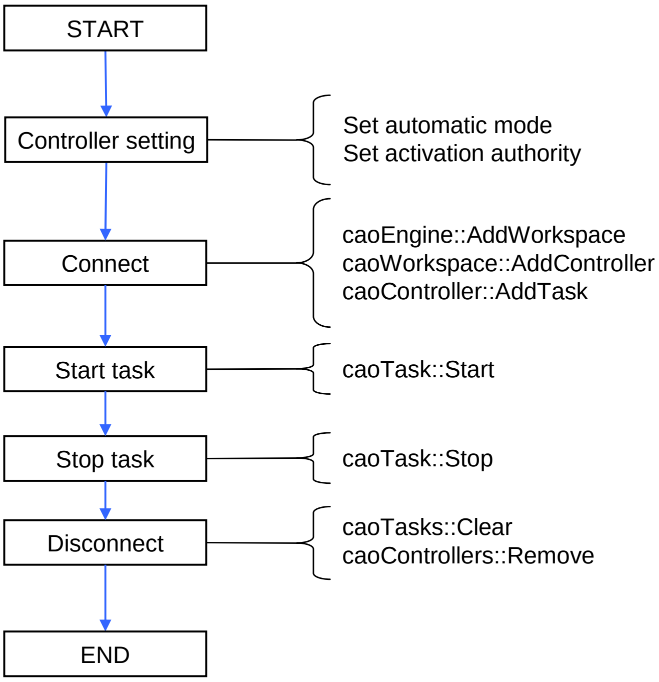


Figure 4-4 Task control flow

4.2.1. Connection

For details about the procedure for creating a CaoController object, refer to "4.1.1 Connection". To control a task, create a CaoTask object. Following is an example code for creating a task object.

```
Dim g_task as CaoTask        ' Variable that stores a CaoTask object
Set g_task = g_ctrl.AddTask("PRO01", "")
```

4.2.2. Start/stop of a task

To start and stop a task, use Start method and Stop method of a CaoTask object. Following is an example of continuous execution and cycle stop of a task.

```
g_task.Start 2               ' Continuous execution
g_task.Stop 3                ' Cycle stop
```

4.2.3. Sample program

The sample program uses the automatically created workspace and controls the task "PRO01" (continuous execution and cycle stop).

List 4-2 Task.frm

```
Dim g_eng As CaoEngine
Dim g_ctrl As CaoController
Dim g_task As CaoTask

Private Sub Command1_Click()
     ' Start task
    g_task.Start 2
End Sub

Private Sub Command2_Click()
     ' Stop task
    g_task.Stop 3
End Sub

Private Sub Form_Load()
    Set g_eng = New CaoEngine

    ' Connect RC: IP setting depends on your RC setting.
    Set g_ctrl = g_eng.Workspaces(0).AddController("RC8", "caoProv.DENSO.RC8", "",
"Server=192.168.0.1")

    ' Task name "PR01"
    Set g_task = g_ctrl.AddTask("PRO1", "")
End Sub

Private Sub Form_Unload(Cancel As Integer)
    g_ctrl.Tasks.Clear
    Set g_task = Nothing

    g_eng.Workspaces(0).Controllers.Remove g_ctrl.Index
    Set g_ctrl = Nothing

    Set g_eng = Nothing
End Sub
```

4.3. Robot control with RC8 controller

To perform robot control, the controller must be set to AUTO mode.

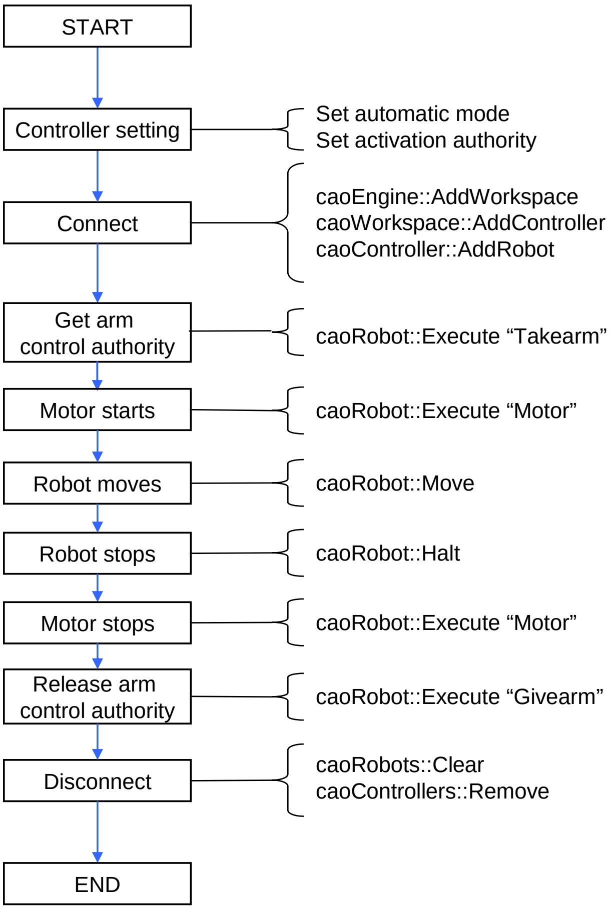


Figure 4-5 Robot control flow

4.3.1. Connection

For details about the procedure for creating a CaoController object, refer to "4.1.1 Connection". To run the robot, create a CaoRobot object. Following is an example code.

```
-----------------------------------------------------------------------------------------------------------------
        Dim g_robot as CaoRobot                       ' Variable that stores a CaoRobot object
        Set g_ robot = g_ctrl.AddRobot("Arm", "")
-----------------------------------------------------------------------------------------------------------------
```

4.3.2. Getting and release of arm control authority

To perform robot control, get the arm control authority for the robot. Furthermore, release the arm control authority for the robot before disconnecting it from the controller. Following is an example code.

```
-----------------------------------------------------------------------------------------------------------------
        g_robot.Execute "Takearm"                     ' Get arm control authority
          ‘ :
        g_robot.Execute "Givearm"                     ' Release arm control authority
-----------------------------------------------------------------------------------------------------------------
```

4.3.3. Start and stop of the motor

To perform robot control, the robot motor must be running. Following is an example code for starting and stopping the motor using the RC8 provider. For further details, refer to "5.2.27.25 CaoRobot::Execute("Motor") command″.

```
------------------------------------------------------------------------------------------
        g_robot.Execute "Motor", Array(1, 0)    ' Start motor
          ':
        g_robot.Execute "Motor", Array(0, 0)    ' Stop motor
------------------------------------------------------------------------------------------
```

4.3.4. Move and stop of the robot

CaoRobot::Move method moves the robot. Refer to "5.2.24 CaoRobot::Move method" for details of Move. By adding NEXT option to Move, CaoRobot::Halt method can stop the robot motion while it is moving.

```
------------------------------------------------------------------------------------------
        g_robot.Move 1,"P(400, 300, 200, 180, 0, 180, 5)","Next"        ' Move robot
          ':
        g_robot.Halt                                                    ' Stop robot
------------------------------------------------------------------------------------------
```

4.3.5. Sample program

The sample program uses the automatically created workspace and moves the robot to a position stored in P10 (10th element of P-type variable) and then moves it to the position stored in P11 (11th element of P-type variable). By adding NEXT option to Move, CaoRobot::Halt method can stop the robot motion while it is moving.

List 4-3 **Robot.frm**

```
Dim g_eng As CaoEngine
Dim g_ctrl As CaoController
Dim g_robot As CaoRobot
Dim g_robotVar As CaoVariable
Dim g_haltFlag As Boolean

Private Sub Command1_Click()
  ' Start motor if arm is stationary
  If g_robotVar.Value = False Then
     g_robot.Execute "Motor", Array(1, 0)
  End If
End Sub

Private Sub Command2_Click()
  ' Stop motor if arm is stationary
  If g_robotVar.Value = False Then
     g_robot.Execute "Motor", Array(0, 0)
  End If
End Sub

Private Sub Command3_Click()
    ' Stop robot
    g_robot.Halt

    ' Record robot stop
    g_haltFlag = True
End Sub

Private Sub Command4_Click()
    ' Do not run new operation instruction if arm is running
    If g_robotVar.Value = True Then
        Exit Sub
    End If

    g_haltFlag = False

    ' Run robot
    g_robot.Move 1, "@P P10", "NEXT"

    ' Do not start next motion until previous motion is completed
    Do Until g_robotVar.Value = False
        DoEvents
    Loop

    ' Do not start next motion after robot is stopped
    If g_haltFlag = True Then
        Exit Sub
    End If

    ' Run robot
    g_robot.Move 1, "@P P11", "NEXT"
End Sub

Private Sub Form_Load()
    Set g_eng = New CaoEngine

' Connect RC: IP setting depends on your RC setting.
    Set g_ctrl = g_eng.Workspaces(0).AddController("RC8", "caoProv.DENSO.RC8", "",
"Server=192.168.0.1")

' Create CaoRobot object
    Set g_robot = g_ctrl.AddRobot("Arm")

    ' Argument used to check arm running status
    Set g_robotVar = g_robot.AddVariable("@BUSY_STATUS")

     ' Get arm control authority
    g_robot.Execute "Takearm"

    ' Start motor
    Command1_Click
End Sub

Private Sub Form_Unload(Cancel As Integer)
    ' Stop motor
    Command2_Click

    ' Release arm control authority
    g_robot.Execute "Givearm"

    g_robot.Variables.Clear
    Set g_robotVar = Nothing
    g_ctrl.Robots.Clear
    Set g_robot = Nothing
    g_eng.Workspaces(0).Controllers.Remove g_ctrl.Index
    Set g_ctrl = Nothing
    Set g_eng = Nothing
End Sub
```

5. Command Reference

5.1. List of commands

Table 5-1 List of commands

Category	Method/property	Function	
CaoWorkspace			
	Addcontroller	Connect communication to the RC.	P.38
CaoController			
	AddFile	Connect to a file or folder (PAC, system file).	P.39
	AddRobot	Connect the robot.	P.40
	AddTask	Connect the task (PAC).	P.41
	AddVariable	Connect the user/system variable.	P.41
	get_Name	Get the controller name.	P.43
	get_FileNames	Get a list of file names.	P.44
	get_TaskNames	Get a list of tasks (PAC).	P.44
	get_VariableNames	Get a list of user/system variables.	P.44
	Execute	Execute a command of the controller class.	P.44
CaoFile			
	AddFile	Connect a PAC file.	P.47
	AddVariable	Connect a system variable of files.	P.48
	get_VariableNames	Get a list of system variable names.	P.48
	get_FileNames	Get a list of files.	P.48
	get_Size	Get the size of a file.	P.48
	get_Value	Get the value of a file.	P.48
	put_Value	Set the value of a file.	P.49
CaoRobot			
	Accelerate	Set the internal acceleration and deceleration ratio of the robot.	P.49
	AddVariable	Connect a system variable.	P.49
	get_VariableNames	Get a list of system variable names.	P.49
	Halt	' Stop the robot in asynchronous motion.	P.50

	Change	Change the tool/user coordinate system of the robot.	P.50
	Drive	This method is not supported directly in this provider.	P.50
	Move	' Robot moves.	P.50
	Rotate	Robot rotates around the specified axis.	P.50
	Speed	Set the internal movement speed of the robot.	P.55
	Execute	Execute a command of the robot.	P.55
CaoTask			
	AddVariable	Connect a system variable of the robot.	P.80
	get_VariableNames	Get a list of system variable names.	P.80
	Start	Start the PAC program.	P.80
	Stop	Stop the PAC program.	P.81
	Execute	Execute a command of the task class.	P.81
CaoVariable			
	get_Value	Get a value.	P.83
	put_Value	Set a value.	P.83

5.2. Methods and properties

5.2.1. CaoWorkspace::AddController method

RC8 provider refers to the parameters passed when the AddController method is executed and connects to the target controller.

The option strings specify the communication method, connection parameters and timeout period. Options are delimited by ",".

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

bstrCtrlName : [in] Controller name
Specify a unique arbitrary string for each connection.
* An error (0x80000205) occurs if the same name is specified from a different application or another PC.
If an empty string ("") is specified, the Cao engine automatically assigns a unique controller name.

bstrProvName : [in] Provider name (Fixed to "CaoProv.DENSO.RC8")

bstrPcName : [in] Provider execution machine name
Specify an empty string ("") for the same machine.

bstrOption : [in] Option character string = "<Option 1>, <Option 2>,…"

Following is a list of option string items.

Table 5-2 Option character string of CaoWorkspace::AddController

Option	Explanation
Server=<IP address>	Specify the IP address of the RC8 controller to be connected. Example: "Server=192.168.0.1"
Timeout=<Time>	Specify the communication timeout period in ms. It is 500 ms by default. This option is enabled only when the Server option is specified.
Interval=<Time>	Specify an interval for getting a message from the connected controller in ms. It is 100 ms by default. This option is enabled only when the Server option is specified.
InvokeTimeout=<Time>	Specify the command invoke timeout period in ms. A timeout error occurs if the command processing takes longer than the specified time. It is 180000 ms by default. This option is enabled only when the Server option is specified.

Example Create CaoController

```
--------------------------------------------------------------------------------------------------------------------------------
        Private caoEng As CaoEngine                ' Engine object
        Private caoWs As CaoWorkspace              ' WorkSpace object
        Private caoCtrl As CaoController           ' Controller object

        Set caoEng = New CaoEngine
        Set caoWS = caoEng.CaoWorkspaces.Item(0)
        Set caoCtrl = CaoWS.AddController(“rc8","CaoProv.DENSO.RC8","・,"Server=192.168.0.1,Timeout=1000")
--------------------------------------------------------------------------------------------------------------------------------
```

5.2.2. CaoController::AddFile method

The argument of the AddFile method of the CaoController class specifies the file name (BSTR type). The specified "File name" is the PAC program name, system reserved file name, or directory name.

A directory can be specified as an argument by designating only a file path.

If the path is not specified, files in the project root, the default directory, are specified.

Following shows the argument specification of AddFile.

Syntax AddFile(<bstrName:BSTR > [,<bstrOption:BSTR>])

bstrName	:	[in]	File/directory name
bstrOption	:	[in]	Option character string

Specify a directory name with a '¥' symbol added to the end of it.

The option uses the following character strings.

Table 5-3 Option character string of CaoController::AddFile

Option	Meaning
@Create[=<0 to 2>]	0: bstrName is not created. An error is returned if bstrName does not exist (default). 1: bstrName is created. The existing bstrName is acquired if it already exists. 2: bstrName is created. An error is returned if the specified bstrName exists.

The table below shows a list of files.

Table 5-4 File implementation status list

	ORiN2 file name	Form	Explanation
1	*.PCS	text	PacScript source
2	*.H	text	PacScript header
3	*.PNS	text	Operation panel source

[Attention]

The CaoFile object does not support simultaneous access to a file.

Be sure to implement an exclusive file access control routine in the application.

Example Get the content of a Pro1.pcs file.

```
--------------------------------------------------------------------------------
        Dim caoFl As CaoFile
        Dim strText As String
        Set caoFl = caoCtrl.AddFile("pro1.pcs"," " )    ' Specify pro1.pcs
        strText = caoFl.Value
--------------------------------------------------------------------------------
```

5.2.3. CaoController::AddRobot method

A CaoRobot object is retrieved by calling the AddRobot method. The argument of the AddRobot method of the CaoController class specifies the robot name (BSTR type). "Robot name" specified here is an arbitrary string. For example, specify AddRobot ("Robot1").

Syntax AddRobot(<bstrName:BSTR > [,<bstrOption:BSTR>])

bstrName : [in] Robot name

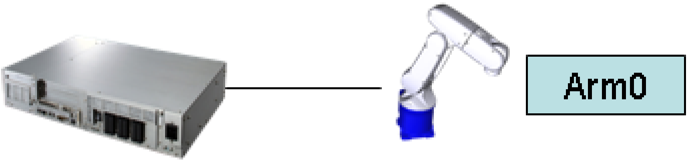
bstrOption : [in] Option character string

ID=<Arm number>

By default, ID=0, i.e., the master arm (Arm0) is specified.

Example

```
--------------------------------------------------------------------------------
        Dim caoRob as CaoRobot
        Set caoRob = caoCtrl.AddRobot("Robot"," " )   ' Specify Arm0
--------------------------------------------------------------------------------
```



5.2.4. CaoController::AddTask method

The argument of the AddTask method of the CaoController class specifies the task name (BSTR type). "Task name" specified here specifies a PAC program name. For instance, the CaoTask object is retrieved in the expression like AddTask("pro1").

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

bstrName : [in] Task name

bstrOption : [in] Option character string (not used)

Example

```
--------------------------------------------------------------------------------
        Dim caoTsk as CaoTask
        Set caoTsk = caoCtrl.AddTask("Pro1"," " )   ' Specify Pro1
--------------------------------------------------------------------------------
```

5.2.5. CaoController::AddVariable method

The AddVariable method of this CaoController class is a method for the access to the variable. In the RC8 provider, both the user variable and the system variable can be specified for the variable name.

User variables support the following variables, i.e., RC8 controller global variables (I, F, V, P, J, D, T, S) and I/O.

The following shows the argument specifications of AddVariable.

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

bstrName : [in] Variable name "<Variable name>[<Number>]"

bstrOption : [in] Option character string "<Option>"

<Variable identifier> : I, F, V, P, J, D, T, S or IO, IOB, IOW, IOD, IOF. The characters are not case-sensitive (uppercase and lowercase have the same meaning).
The I/O values are processed as follows: IO in Bits, IOB in Bytes, IOW in Words, IOD in Double Words (Long), and IOF in Float (Single).

<Number> : Variable's number specified by the identifier or "*" or "*_<Numeric value>"
The number is specified by a decimal number.
The specification of "*" is handled as the initial value of 0. The variable's number can be retrieved and changed by 'ID' property of the variable object.
Specify the numeric value in *_<Numeric value> as a decimal number. Wild card for variables of the same type (*): This is an identification number that enables to specify multiple definitions, and the value has no special meaning.

"[" and ”]” can be omitted.

Example 1	“i0","I[0]"	···	Specify the 0th I type variable.
Example 2	“IO128","io[128]"	···	Specify the 128th I/O variable.
Example 3	“I*","IO[*]"	···	Specify a wild card.
Example 4	“I*_1","I*_2" ,"I*_3"	···	Specify multiple wild cards (I type variables).

When specifying a system variable, add “@" at the beginning of the variable name. All variables without “@” at the beginning of names are treated as user variables.
Refer to "5.3 Variable list" about the system variables implemented in the RC8 provider.

Example Access to the 128th I/O variable

```
Dim caoVar as CaoVariable
Set caoVar = caoCtrl.AddVariable("IO128"," " )    ' Specify I/O128
caoVar.value = 1
MsgBox caoVar.Value
```

```
Dim caoVar as CaoVariable
Set caoVar = caoCtrl.AddVariable("IO*"," " )    ' Specify IO* and the index in ID
caoVar.ID = 128
caoVar.value = 1
MsgBox caoVar.Value
```

5.2.6. CaoController::AddExtension method

The argument of the AddExtension method of the CaoController class specifies the extended function name (BSTR type).

Syntax <caoExt:CaoExtension object> = AddExtension (<bstrName:BSTR> [,<bstrOption:BSTR>])

bstrName	:	[in]	Extended function name
bstrOption	:	[in]	Option character string (not used)
Return value	:	[out]	CaoExtension class object

Following is a list of available extended functions.

Table 5-5 CaoWorkspace::AddExtension extended function name list

Extended function name	Explanation
Hand object<n>[1]	Object for electric end-effector <n>[2]

Example:

```
--------------------------------------------------------------------------------
        Dim caoExt as CaoExtension
        Set caoExt = caoCtrl.AddExtension( “Hand0”)   ' Get Hand0 object
--------------------------------------------------------------------------------
```

5.2.7. CaoController::get_Name property

Get the controller name specified in the AddController method of the CaoWorkspace class.

Example Display the automatically assigned controller name.

```
--------------------------------------------------------------------------------
        Private caoEng As CaoEngine           ' Engine object
        Private caoWs As CaoWorkspace         ' WorkSpace object
        Private caoCtrl As CaoController      ' Controller object

        Set caoEng = New CaoEngine
        Set caoWS = caoEng.CaoWorkspaces.Item(0)
        Set caoCtrl = CaoWS.AddController(" ","CaoProv.DENSO.RC8"," " ,"Server=192.168.0.1")

        Debug.Print caoCtrl. Name
--------------------------------------------------------------------------------
```

[1] <n> specifies a board number (0 to 7).

[2] Before using an electric end-effector object, it is necessary to install an optional electric end-effector control board and make initial settings. For details about making initial settings, refer to "Settings of the Electric Gripper" in "DENSO ROBOT USER MANUALS Controller Model:RC8 Series."

5.2.8. CaoController::get_FileNames property

Get a list of file names that can be specified by the AddFile method.

Example List the following file names in the root folder.

```
Dim ln%, lb%, ub%
Dim var As variant

var = caoCtrl.FileNames

lb = LBound( var )
ub = UBound( var )
For ln = lb To ub
        Debug.Print Str( ln ) &"=" & var( ln )
Next
```

5.2.9. CaoController::get_TaskNames property

Get a list of task names that can be specified by the AddTask method.

Example List task names.

```
Dim ln%, lb%, ub%
Dim var As variant

var = caoCtrl.TaskNames

lb = LBound( var )
ub = UBound( var )
For ln = lb To ub
        Debug.print Str( ln ) &"=" & var( ln )
Next
```

5.2.10. CaoController::get_VariableNames property

Get a list of variable names and system variable names that can be specified by the AddVariable method.

5.2.11. CaoController::Execute method

Execute a provider-specific extended command belonging to the CaoController class.

The arguments of the Execute method specify a command as a BSTR and a parameter as a VARIANT array.

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

bstrCmd	:	[in]	Command name
vntParam	:	[in]	Parameter
vntRet		[out]	Return value

If a method not defined in this class is called using the runtime binding function, the Execute method is automatically called according to the following specifications:

vntRet = Obj.CommandName Param1, Param2, ...
↓
vntRet = Obj.Execute("CommandName", Array(Param1, Param2, ...))

1. The command name is passed as a BSTR string to the first argument.
2. All the parameters are passed as a VARIANT array to the second argument.

Example

```
-----------------------------------------------------------------------------------------------------------------------------
        Dim vRes as Variant
        vRes = caoCtrl.Execute("ClearError" )    ' Clear error of controller
        vRes = caoCtrl.ClearError()
-----------------------------------------------------------------------------------------------------------------------------
```

The list shows available commands.

Table 5-6 List of commands of CaoController::Execute method

Category	Command name	Function	
Error handling			
	ClearError	Reset an error.	P.45
	GetErrorDescription	Get an error string.	P.46
Task control			
	KillAll	Terminate all tasks.	P.46
	SuspendAll	Suspend all tasks.	P.46
	StepStopAll	Step-stop all tasks	P.47
	ContinueStartAll	Continue-start all tasks.	P.47

5.2.11.1. CaoController::Execute("ClearError") command

Clear an error that occurs in the controller.

Syntax ClearError ( )

Argument : None
Return value : None

Example

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

5.2.11.2. CaoController::Execute("GetErrorDescription") command

Get the description of an error with the specified error code.

Syntax GetErrorDescription (<lErrCode>)

<lErrCode>	:	[in] Error code (VT_I4)
Return value	:	Error description (VT_BSTR)

Example

```
Dim strDescription As String
strDescription = caoCtrl.Execute("GetErrorDescription" , &H83500003 )
```

5.2.11.3. CaoController::Execute("KillAll") command

Perform initialized stop of all the running tasks.

Syntax KillAll ( )

Argument	:	None
Return value	:	None

If the robot is running, all robot stop is executed, and all the tasks are sent into an initialized stop status.

Example

```
caoCtrl.Execute"KillAll"
```

5.2.11.4. CaoController::Execute("SuspendAll") command

Suspend all the running tasks.

Syntax SuspendAll ( )

Argument	:	None
Return value	:	None

If the robot is running, all robot suspend is executed, and all the tasks are sent into a suspended status.

Example

```
------------------------------------------------------------------------------------------------------------------------------------
        caoCtrl.Execute"SuspendAll"
------------------------------------------------------------------------------------------------------------------------------------
```

5.2.11.5. CaoController::Execute("StepStopAll") command

Perform step stop of all the running tasks.

Syntax StepStopAll ( )

Argument	:	None
Return value	:	None

Example

```
------------------------------------------------------------------------------------------------------------------------------------
        caoCtrl.Execute"StepStopAll"
------------------------------------------------------------------------------------------------------------------------------------
```

5.2.11.6. CaoController::Execute("ContinueStartAll") command

Start running all the tasks that are being suspended.

Syntax ContinueStartAll ( )

Argument	:	None
Return value	:	None

Example

```
------------------------------------------------------------------------------------------------------------------------------------
        caoCtrl.Execute"ContinueStartAll"
------------------------------------------------------------------------------------------------------------------------------------
```

5.2.12. CaoFile::AddFile method

Create a file object in the same way as 5.2.2. The file path corresponding to the created CaoFile object is "<Path of the parent object>/<File name specified in AddFile>".

Example Display the size of Pro1.pcs file in the User folder.

```
------------------------------------------------------------------------------------------------------------------------------------
        Dim caoFlDir As CaoFile
        Set caoFlDir = caoCtrl.AddFile("User¥","・  )    ' Specify User folder
        Dim caoFl As CaoFile
        Set caoFl = caoFlDir.AddFile("Pro1.pcs" )    ' Specify User¥Pro1.pcs file
        Debug.Print caoFl.Size
----------------------------------------------------------------------------------------------------
```

5.2.13. CaoFile::AddVariable method

The argument of the AddVariable method of the CaoFile class specifies the system variable name.

Refer to Table 5-15 for the list of implemented system variables.

Example Get the CRC of the Pro1.pcs file.

```
----------------------------------------------------------------------------------------------------
        Dim caoFl As CaoFile
        Set caoFl = caoCtrl.AddFile(“ro1.pcs")

        Dim caoCrc As CaoVariable
        Set caoCrc = caoFl.AddVariable(“CRC")

        Debug.Print caoCrc.Value    ' Display CRC of Pro1.pcs
----------------------------------------------------------------------------------------------------
```

5.2.14. CaoFile::get_VariableNames property

Get a list of variable names and system variable names that can be specified by the AddVariable method.

5.2.15. CaoFile::get_FileNames property

This can be executed only when the path corresponding to the object is a directory.

Executing it gets a list of file names in the directory.

5.2.16. CaoFile::get_Size property

Get the size of the file corresponding to the object.

Example Get the size of the Pro1.pcs file.

```
----------------------------------------------------------------------------------------------------
        Dim caoFl As CaoFile
        Set caoFl = caoCtrl.AddFile(“ro1.pcs")

        Debug.Print caoFl.Size    ' Display size of Pro1.pcs
----------------------------------------------------------------------------------------------------
```

5.2.17. CaoFile::get_Value property

Get the contents of the file corresponding to the object.

Example Get the contents of Pro1.pcs file.

```
--------------------------------------------------------------------------------------------------------------------------------
        Dim caoFl As CaoFile
        Set caoFl = caoCtrl.AddFile (“ro1.pcs")

        Debug.Print caoFl.Value    ' Display contents of Pro1.pcs
--------------------------------------------------------------------------------------------------------------------------------
```

5.2.18. CaoFile::put_Value property

Set the contents of the file corresponding to the object.

5.2.19. CaoRobot::Accelerate method

Set the internal acceleration and deceleration ratio of the robot.

This method corresponds to ACCEL and JACCEL instructions of PacScript language.

The following shows the argument specifications of Accelerate.

Syntax Accelerate <lAxis:LONG >, <fAccel:FLOAT> [,<fDecel:FLOAT>]

lAxis	:	[in]	Axis number -1: Tool accel (ACCEL), 0: All axes (JACCEL)
fAccel	:	[in]	Acceleration (-1: keep current setting)
fDecel		[in]	Deceleration (-1: keep current setting)

(Example 1) Accelerate 0, 50.0, -1 ' Acceleration = 50%, deceleration = no change

(Example 2) Accelerate 0, -1, 60.0 ' Acceleration = no change, deceleration = 60%

5.2.20. CaoRobot::AddVariable method

The argument of the AddVariable method of the CaoRobot class specifies the system variable name.

Refer to Table 5-13 for the list of implemented system variables.

Example Refer to the current robot position (P type).

```
--------------------------------------------------------------------------------------------------------------------------------
        Dim caoRob As CaoRobot
        Set caoRob = caoCtrl.AddRobot("Arm0")

        Dim caoCurPos As CaoVariable
        Set caoCurPos = caoRob.AddVariable("@CURRENT_POSITION")

        Dim vVal As Variant
        vVal = caoCurPos.Value
        MsgBox vVal(0) &"," & vVal(1) &"," & vVal(2)    ' Display X, Y, and Z
--------------------------------------------------------------------------------------------------------------------------------
```

5.2.21. CaoRobot::get_VariableNames property

Get a list of variable names and system variable names that can be specified by the AddVariable method.

5.2.22. CaoRobot::Halt method

Stop the robot motion.

A runtime error occurs if a task in the RC8 controller has robot control authority (Takearm has been executed). Use the CaoTask::Stop method to control the stop of a task.

5.2.23. CaoRobot::Change method

Change the tool/user coordinate system of the robot.

This method corresponds to CHANGETOOL and CHANGEWORK instructions of PacScript language.

The following shows the argument specifications of Change.

Syntax Change <bstrName:BSTR>

bstrName : [in] For CHANGETOOL= "Tool<Number>"
For CHANGEWORK= "Work<Number>"

<Number> : Numerical value expressed by decimal number (default: 0)

Example

```
---------------------------------------------------------------------------------------------------------
        Dim caoRob As CaoRobot
        Set caoRob = caoCtrl.AddRobot("Arm0")

        caoRob.Execute"TakeArm", Array(0, 0)

        caoRob.Change"Tool1"          ' Change to Tool1
        caoRob.Change"Work1"          ' Change to Work1
        caoRob.Move 1,"P10"

        caoRob.Execute"GiveArm"
---------------------------------------------------------------------------------------------------------
```

5.2.24. CaoRobot::Drive method

This method is not supported directly in this provider.

Instead, use "DriveEx" or "DriveAEx" command of CaoRobot::Execute that can operate two or more axes all at once.

5.2.25. CaoRobot::Move method

Move the robot to the specified coordinates. This method corresponds to MOVE instruction of PacScript language. The following shows the argument specifications of Move.

Syntax Move <lComp:LONG >, <vntPose:POSEDATA> [,<vntPose:POSEDATA>…] [, < bstrOpt:BSTR>]

lComp	:	[in]	Interpolation 1:MOVE P,... , 2:MOVE L,... , 3:MOVE C,... , 4:MOVE S,...
vntPose	:	[in]	Pose data (POSEDATA type)
bstrOpt	:	[in]	Motion option, "NEXT" = Asynchronous execution

Refer to "POSEDATA Type Definition" for the POSEDATA type.

The form and the meaning when the character string is specified by the POSEDATA type are as follows.

In case of VT_BSTR type (string)

- If Comp = 1, 2

  "[<@pass start displacement>] <Pose> [<Extended-joints>]"

  ex. "P1", "@P T100", "@E J520"

- If Comp = 3

  "<Pose 1> [<Extended-joints>], [<@pass start displacement>] <Pose 2>[<Extended-joints>]"

  - *** Pose 1 and Pose 2 need to be same variable type. *** ***

  ex. "P1,@E P2", "T100,@P T101"

- If Comp = 4

  "[<@pass start displacement>] <Free curve trajectory number> [<Extended-joints>]"

  ex. "1", "@P 20", "@E 5"

<Free curve trajectory number> : A decimal number (spline curve number 1 to 20)

<Pose> : "<Variable type><Variable number>" or "[<Variable type>](<Element 1>,<Element 2>,…)"

: <Variable type> : One of characters 'P', 'T' and 'J' 'P' is assumed to be specified if the variable type is omitted in the specification of an element (raw value).

<Number> : A decimal number

<Element n> : An element of either of variable types 'P', 'T', and 'J'

P type＝P(<x>,<y>,<z>,<rx>,<ry>,<rz>,<fig>)

J type＝J(<j1>,<j2>,<j3>,<j4>,<j5>,<j6>,<j7>,<j8>)

T type＝T(<x>,<y>,<z>,<ox>,<oy>,<oz>,<ax>,<ay>,<az>,<fig>)

[Note] For 4-axis robot, T element of P type variable corresponds to <rz>. <rx> and <ry> are not used.

<@pass start displacement> : “@0", "@P","@E ", or "@<Value>"

<Extended-joints> : The syntax of an extended-joints option is shown below.[3]
(Specify the extended-joints option after the pose data and blank.)

"EX((<JointNumber1>, <RelativeDistance1>)[, (<JointNumber2>,<RelativeDistance2>)…])
or
"EX((<JointNumber1>, <AxisCoordinates1>)[, (<JointNumber2>,<AxisCoordinates2>)…])

Example 1	Move 1,"@P P1" ,"NEXT"	‘ MOVE P, @P P1, NEXT
Example 2	Move 3,"P1,@E P2"	‘ MOVE C, P1,@E P2
Example 3	Move 2,"@0 P(307.1856,-157.8244,107.0714,160,0,0,1)"	‘ MOVE L,@0 P(307.1856,-157.8244,107.0714,160,0,0,1)
Example 4	Move 4,"@E 2"	‘ MOVE S, @E 2
Example 5	Move 1,"@P P10 EX((7, 30.5))" ,"NEXT"	‘ MOVE P, @P P10 EX((7,30.5)), NEXT
Example 6	Move 2,"@E P20 EXA((7, 30.8), (8, 90.5))	‘MOVE L, @E P20 EXA((7, 30.8), (8, 90.5))"

When two or more Move methods are executed consecutively, the latter motion method is in “wait” status until the preceding motion method execution ends, and an application seems to be not responding. In this wait state, OnMessage event #9 of CaoController class is periodically issued, so catch the event and pass the program control authority to application program if necessary.
The following table shows the PacScript MOVE commands supported by Move method.

[3] To use the extended joint option, define extended joint related settings (e.g. arm group definition) on the controller, and use TakeArm command to select an arm group for controlled extended joint.

Table 5-7 List of Move commands

Division	PAC command	Move method
MOVE P,…	MOVE P, P<*n1*>	Move 1,"P<*n1*>“
	MOVE P, @P P<*n1*>	Move 1,"@P P<*n1*>“
	MOVE P, @E P<*n1*>	Move 1,"@E P<*n1*>“
	MOVE P, T<*n1*>	Move 1,"T<*n1*>“
	MOVE P, @P T<*n1*>	Move 1,"@P T<*n1*>“
	MOVE P, @E T<*n1*>	Move 1,"@E T<*n1*>“
	MOVE P, J<*n1*>	Move 1,"J<*n1*>“
	MOVE P, @P J<*n1*>	Move 1,"@P J<*n1*>“
	MOVE P, @E J<*n1*>	Move 1,"@E J<*n1*>“
MOVE L,…	MOVE L, P<*n1*>	Move 2,"P<*n1*>“
	MOVE L, @P P<*n1*>	Move 2,"@P P<*n1*>“
	MOVE L, @E P<*n1*>	Move 2,"@E P<*n1*>“
	MOVE L, T<*n1*>	Move 2,"T<*n1*>“
	MOVE L, @P T<*n1*>	Move 2,"@P T<*n1*>“
	MOVE L, @E T<*n1*>	Move 2,"@E T<*n1*>“
	MOVE L, J<*n1*>	Move 2,"J<*n1*>“
	MOVE L, @P J<*n1*>	Move 2,"@P J<*n1*>“
	MOVE L, @E J<*n1*>	Move 2,"@E J<*n1*>“
MOVE C,…	MOVE C, P<*n1*>, P<*n2*>	Move 3,"P<*n1*>, P<*n2*>“
	MOVE C, P<*n1*>, @P P<*n2*>	Move 3,"P<*n1*>, @P P<*n2*>“
	MOVE C, P<*n1*>, @E P<*n2*>	Move 3,"P<*n1*>, @E P<*n2*>“
	MOVE C, T<*n1*>, T<*n2*>	Move 3,"T<*n1*>, T<*n2*>“
	MOVE C, T<*n1*>, @P T<*n2*>	Move 3,"T<*n1*>, @P T<*n2*>“
	MOVE C, T<*n1*>, @E T<*n2*>	Move 3,"T<*n1*>, @E T<*n2*>“
	MOVE C, J<*n1*>, J<*n2*>	Move 3,"J<*n1*>, J<*n2*>“
	MOVE C, J<*n1*>, @P J<*n2*>	Move 3,"J<*n1*>, @P J<*n2*>“
	MOVE C, J<*n1*>, @E J<*n2*>	Move 3,"J<*n1*>, @E J<*n2*>“
Extended-joints	MOVE P, P<*n1*> EX((*j1*, *v1*))	Move 1,"P<*n1*> EX((*j1*,*v1*))"
	MOVE P, P<*n1*> EX((*j1*, *v1*),(*j2*, *v2*))	Move 1,"P<*n1*> EX((*j1*,*v1*),(*j2*, *v2*))"
	MOVE P, P<*n1*> EXA((*j1*, *v1*))	Move 1,"P<*n1*> EXA((*j1*,*v1*))"
	MOVE P, P<*n1*> EXA((*j1*, *v1*),(*j2*, *v2*))	Move 1,"P<*n1*> EXA((*j1*,*v1*),(*j2*, *v2*))"

Misc.	MOVE P, P<*n1*> +(*x*,*y*,*z*,*rx*,*ry*,*rz*) MOVE P, P<*n1*> +(*x*,*y*,*z*,*rx*,*ry*,*rz*)H	Move 1, DEV ("<*n1*>","P(*x*,*y*,*z*,*rx*,*ry*,*rz*)") Move 1, DEVH ("<*n1*>","P(*x*,*y*,*z*,*rx*,*ry*,*rz*)") *Refer to CaoRobot::Execute for DEV and DEVH.

<*n1*>, <*n2*> : Integers 0 to 65535 or "(<Element 1>, <Element 2>, …)"

5.2.26. CaoRobot::Rotate method

Rotate the robot around the specified axis.

This method corresponds to ROTATE instruction of PacScript language.

The following shows the argument specifications of Rotate.

Syntax Rotate <vntRotSuf:POSEDATA>, <fDeg:FLOAT>, <vntPivot:POSEDATA>, <bstrOpt:BSTR>

vntRotSuf	:	[in]	Rotation surface
fDeg	:	[in]	Angle (deg)
vntPivot	:	[in]	Rotation center
bstrOpt		[in]	Motion option “@0”, “@P”, “@E”, "pose=<n>“, "NEXT", or “@<Value>”

Refer to "POSEDATA Type Definition" for the POSEDATA type. The form and the meaning when the character string is specified by the POSEDATA type are as follows.

In case of VT_BSTR type (string)

- vntRotSuf: [in] rotation surface
  “V<n1>,V<n2>,V<n3>“ or "XY","YZ","ZX","XYH","YZH","ZXH"
  or "V(<x>,<y>,<z>),V(…),V(…)"
  ex."V100,V101,V102"
  However, "XY", "YZ", "ZX", "XYH", "YZH", and "ZXH" are supported only by VT_BSTR.

- vntPivot: [in] rotation center
  “V<n4>“ or "V(<x>,<y>,<z>)"
  ex."V103"

Example 1	Rotate"V1,V2,V3", 45.8,"V4","@E"　‘ ROTATE V1,V2,V3, @E 45.8, V4
Example 2	Rotate"V(0,0,1),V(0,1,0),V(0,0,0)", 30.0,"V(0,0,0)","@E,pose=1,NEXT"
Example 3	Rotate"XY", 90.0,"V(0,0,0)","@P"
Example 4	Rotate"XYH", -45.0,"V(250,0,0)","@150"

Rotation surface is specified by three V type variables. The three points in base coordinates define the surfaces. Argument vntRotSuf specifies three V type variables in BSTR (string) type variable separated by comma, space or tab.

Rotation center point vntPivot is specified by a V type variable expressed in BSTR(string) type.

5.2.27. CaoRobot::Speed method

Set the internal movement speed of the robot.

This method corresponds to SPEED and JSPEED instructions of PacScript language.

About the external movement speed of the robot, use "ExtSpeed" command of CaoRobot::Execute.

The following shows the argument specifications of Speed.

Syntax Speed <lAxis:LONG >, <fSpeed:FLOAT>

lAxis : [in] Axis number -1: Effective to Tool axis (SPEED), 0: Effective to all axes (JSPEED)

fSpeed : [in] speed

Example

```
Dim caoRob As CaoRobot
Set caoRob = caoCtrl.AddRobot("Arm0")

caoRob.Execute"TakeArm", Array(0, 0)

caoRob.Speed -1, 85                     ' Internal speed of 85%
caoRob.Execute"ExtSpeed", 50

caoRob.Execute"GiveArm"
```

5.2.28. CaoRobot::Execute method

The Execute method defines peculiar operation commands to the robot that isn't supported by the CaoRobot class, and offers the function to implement them.

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

bstrCmd : [in] Command

vntParam : [in] Parameter

vntRet [out] Return value

If a method not defined in this class is called using the runtime binding function, the Execute method is automatically called according to the following specifications:

vntRet = Obj.CommandName Param1, Param2, ...

↓

vntRet = Obj.Execute("CommandName", Array(Param1, Param2, ...))

1. The command name is passed as a BSTR string to the first argument.
2. All the parameters are passed as a VARIANT array to the second argument.

The list shows available commands.

Table 5-8 List of commands of CaoController::Execute method

Category	Command name	Function	
Operation			
	TMul	Calculate the product of two homogeneous transformation type data.	P.58
	TInv	Calculate the inverse matrix of homogeneous transformation type data.	P.59
	TNorm	Calculate the inverse matrix of homogeneous transformation type data.	P.59
	J2T	Transform J type data to T type data.	P.59
	T2J	Transform T type data to J type data.	P.60
	J2P	Transform J type data to P type data.	P.60
	P2J	Transform P type data to J type data.	P.61
	T2P	Transform T type data to P type data.	P.61
	P2T	Transform P type data to T type data.	P.61
	Dev	Calculate the offset in the base coordinates.	P.62
	DevH	Calculate the offset in the tool coordinates.	P.62
	OutRange	Judge whether the motion range is exceeded.	P.63
	MPS	Calculate the value of the SPEED command from data in Mps.	P.63
	RPM	Calculate the value of the SPEED command from data in rpm.	P.63
Positioning			
	CurPos	Get the current position as P type data.	P.64
	DestPos	Get the target position as P type data.	P.64
	CurJnt	Get the current position as J type data.	P.64
	DestJnt	Get the target position as J type data.	P.65
	CurTrn	Get the current position as T type data.	P.65

	DestTrn	Get the target position as T type data.	P.65
	CurFig	Get the current posture Fig value.	P.66
Log			
	StartLog	Start log recording.	P.66
	StopLog	Stop log recording.	P.66
	ClearLog	Clear log data.	P.67
Robot operation			
	Motor	Turn ON/OFF the motor.	P.67
	ExtSpeed	Set the external speed.	P.67
	TakeArm	Request to get control authority.	P.67
	ForceTakeArm	Forcibly get control authority.	P.68
	GiveArm	Request to release control authority.	P.68
	Draw	Execute the relative movement designated in the work coordinate system.	P.69
	Approach	Execute the absolute movement designated in the tool coordinate system.	P.70
	Depart	Execute the relative movement designated in the tool coordinate system.	P.71
	DriveEx	Execute the relative motion of each axis.	P.72
	DriveAEx	Execute the absolute motion of each axis.	P.72
	RotateH	Execute rotary motion by taking an approach vector as an axis.	P.73
	Arrive	Wait for the robot to reach the defined motion ratio.	P.74
	MotionSkip	Abort the robot motion in progress.	P.74
	MotionComplete	Judge whether the robot motion is complete.	P.75
Tool			
	CurTool	Get the current tool number.	P.75
	GetToolDef	Get the tool definition specified by the tool number.	P.76
	SetToolDef	Set the tool definition.	P.76
Work			
	CurWork	Get the current work number.	P.76
	GetWorkDef	Get the work definition specified by the work number.	P.77
	SetWorkDef	Set the work definition.	P.77

Area			
	GetAreaDef	Get the area definition specified by the area number.	P.77
	SetAreaDef	Set the area parameter.	P.78
	SetArea	Enable the area check.	P.79
	ResetArea	Disable the area check.	P.79
	AreaSize	Return the size (each side length) of a check area as the vector type.	P.79
	GetAreaEnabled	Get the area enabled or disabled status.	P.79
	SetAreaEnabled	Set the area enabled or disabled status.	P.80
Misc.			
	GetRobotTypeName	Get the robot type.L.	P.80

The arguments of the Execute method of the CaoRobot class specify a command number + parameter as a VARIANT array.

Example

```
Dim vRes as Variant
vRes = caoRob.Execute("GetJntData",Array(1, 6 ) )    ' Current motor speed of 6 axes [rpm]
caoRob.Execute"ExtSpeed",Array(50.0, 25.0, 25.0 )
                                              ‘ External speed = 50%, acceleration = 25%, deceleration = 25%
caoRob.Execute"APPROACH", Array(1,"P11","@P 100","NEXT") ‘APPROACH P,P11,@P 100, NEXT
```

5.2.28.1. CaoRobot::Execute("TMul") command

Calculate the product of two homogeneous transformation type data.

Syntax TMul (<Tn1>, <Tn2>)

<Tn1>	:	[in] T type (POSEDATA)
<Tn2>	:	[in] T type (POSEDATA)
Return value	:	Product of <Tn1> and <Tn2> (VT_VARIANT[VT_R8|VT_ARRAY:10 element])

Example

```
Dim vResult As Variant

vResult = caoRob.Execute("TMul",    Array("T10","T20" ) )     ' Calculate by specifying the T type index

vResult = caoRob.Execute("TMul",    Array("T(400,500,400, 1,0,0, 0,1,0, 5)", _
"(T( 100,0,0, 1,0,0, 0,1,0, -1)" ) )     ' Calculate by specifying the T type element directly
```

5.2.28.2. CaoRobot::Execute("TInv") command

Calculate the inverse matrix of T (homogeneous transformation) type data.

Syntax TInv(<Tn1>)

<Tn1>	:	[in] T type (POSEDATA)
Return value	:	Inverse matrix of <Tn1> (VT_VARIANT[VT_R8|VT_ARRAY:10 element])

Example

```
----------------------------------------------------------------------------------------------------------------------------
        Dim vResult As Variant

        vResult = caoRob.Execute("TInv", "T10" )     ‘ Inverse matrix of T10

        vResult = caoRob.Execute("TInv", "T(400,500,400, 1,0,0, 0,1,0, 5)" )    _
        ' Calculate by specifying the T type element directly
----------------------------------------------------------------------------------------------------------------------------
```

5.2.28.3. CaoRobot::Execute("TNorm") command

Normalize T (homogeneous transformation) type data.

Syntax TNorm(<Tn1>)

<Tn1>	:	[in] T type (POSEDATA)
Return value	:	Normalization of <Tn1> (VT_VARIANT[VT_R8|VT_ARRAY:10 element])

Example

```
----------------------------------------------------------------------------------------------------------------------------
        Dim vResult As Variant

        vResult = caoRob.Execute("TNorm", "T10" )     ‘ Normalization of T10

        vResult = caoRob.Execute("TNorm", "T(400,500,400, 1,0,0, 0,1,0, 5)" )    _
        ' Calculate by specifying the T type element directly
----------------------------------------------------------------------------------------------------------------------------
```

5.2.28.4. CaoRobot::Execute("J2T") command

Transform J type data to T type data.

Syntax J2T (<Jn1>)

<Jn1>	:	[in] J type (POSEDATA)
Return value	:	T type (VT_VARIANT[VT_R8|VT_ARRAY:10 element])

Example

```
Dim vResult As Variant

vResult = caoRob.Execute("J2T", "J10" )      ' Transform J10 value to T type data

vResult = caoRob.Execute("J2T", "J(90,90,90, 0,0,0)" )     _
' Transform by specifying the J type element directly
```

5.2.28.5. CaoRobot::Execute("T2J") command

Transform T type data to J type data.

Syntax T2J (<Tn1>)

<Tn1> : [in] T type (POSEDATA)

Return value : J type
(VT_VARIANT[VT_R8|VT_ARRAY:8 element])

Example

```
Dim vResult As Variant

vResult = caoRob.Execute("T2J", "T10" )      ' Transform T10 value to J type data

vResult = caoRob.Execute("T2J", "T(400,400,500, 1,0,0, 0,1,0, 5)" )     _
' Transform by specifying the T type element directly
```

5.2.28.6. CaoRobot::Execute("J2P") command

Transform J type data to P type data.

Syntax J2P (<Jn1>)

<Jn1> : [in] J type (POSEDATA)

Return value : P type
(VT_VARIANT[VT_R8|VT_ARRAY:7 element])

Example

```
Dim vResult As Variant

vResult = caoRob.Execute("J2P", "J10" )      ' Transform J10 value to P type data

vResult = caoRob.Execute("J2P", "J(90,90,90, 0,0,0)" )     _
' Transform by specifying the J type element directly
```

5.2.28.7. CaoRobot::Execute("P2J") command

Transform P type data to J type data.

Syntax P2J (<Pn1>)

<Pn1>	:	[in] P type (POSEDATA)
Return value	:	J type (VT_VARIANT[VT_R8|VT_ARRAY:8 element])

Example

```
--------------------------------------------------------------------------------
        Dim vResult As Variant

        vResult = caoRob.Execute("P2J", "P10" )      ' Transform P10 value to J type data

        vResult = caoRob.Execute("P2J", "P(400,400,500, 180,0,180, 5)" )      _
        ' Transform by specifying the P type element directly
--------------------------------------------------------------------------------
```

5.2.28.8. CaoRobot::Execute("T2P") command

Transform T type data to P type data.

Syntax T2P (<Tn1>)

<Tn1>	:	[in] T type (POSEDATA)
Return value	:	P type (VT_VARIANT[VT_R8|VT_ARRAY:7 element])

Example

```
--------------------------------------------------------------------------------
        Dim vResult As Variant

        vResult = caoRob.Execute("T2P", "T10" )      ' Transform T10 value to P type data

        vResult = caoRob.Execute("T2P", "T(400,400,500, 1,0,0, 0,1,0, 5)" )      _
        ' Transform by specifying the T type element directly
--------------------------------------------------------------------------------
```

5.2.28.9. CaoRobot::Execute("P2T") command

Transform P type data to T type data.

Syntax P2T (<Pn1>)

<Pn1>	:	[in] P type (POSEDATA)
Return value		T type (VT_VARIANT[VT_R8|VT_ARRAY:10 element])

Example

```
---------------------------------------------------------------------------------------------------------
        Dim vResult As Variant

        vResult = caoRob.Execute("P2T", "P10" )     ' Transform P10 value to T type data

        vResult = caoRob.Execute("P2T", "P(400,400,500, 180,0,180, 5)" )     _
        ' Transform by specifying the P type element directly
---------------------------------------------------------------------------------------------------------
```

5.2.28.10. CaoRobot::Execute("Dev") command

Calculate the coordinates of the offset <Pn2> from the reference position <Pn1> in the base coordinates. The Fig value of the offset <Pn2> is ignored.

Syntax Dev (<Pn1>, <Pn2>)

<Pn1>	:	[in] P type (POSEDATA)
<Pn2>	:	[in] P type (POSEDATA)
Return value	:	P type (VT_VARIANT[VT_R8|VT_ARRAY:7 element])

Example

```
---------------------------------------------------------------------------------------------------------
        Dim vResult As Variant

        vResult = caoRob.Execute("Dev", Array("P10","P(100, 200, 300, 180, 0, 180)" ))
        ‘ Calculate the positions of P10 + P(100, 200, 300, 180, 0, 180)
---------------------------------------------------------------------------------------------------------
```

5.2.28.11. CaoRobot::Execute("DevH") command

Calculate the coordinates of the offset <Pn2> from the reference position <Pn1> in the tool coordinates. The Fig value of the offset <Pn2> is ignored.

Syntax DevH (<Pn1>, <Pn2>)

<Pn1>	:	[in] P type (POSEDATA)
<Pn2>	:	[in] P type (POSEDATA)
Return value	:	P type (VT_VARIANT[VT_R8|VT_ARRAY:7 element])

Calculation is performed based on the coordinates of the currently effective tool definition (current tool).

Example

```
---------------------------------------------------------------------------------------------------------
        Dim vResult As Variant

        vResult = caoRob.Execute("DevH", Array("P10","P(100, 200, 300, 180, 0, 180)" ))
        ‘ Calculate the positions of P10 + Tool coordinate P (100, 200, 300, 180, 0, 180)
---------------------------------------------------------------------------------------------------------
```

5.2.28.12. CaoRobot::Execute("OutRange") command

Return a result whether the position data is within the robot's motion range.

The tool and work numbers are ignored if <Pose> is specified as J type data.

Syntax OutRange(<Pose>[, <ToolNo> [, <WorkNo>]])

<Pose> : [in] POSEDATA value (one of P, J, and T types)

<ToolNo> : [in] Tool number -1 (default) is the current tool number VT_I4

<WorkNo> : [in] Work number -1 (default) is the current work number VT_I4

Return value : VT_I4

0: Within motion range

1 to 63: Bit of axis that is software limit

-1: Impossible position due to axis configuration

-2: Singular point

Example Move if the motion range is not exceeded.

```
-----------------------------------------------------------------------------------------------------------------------------
        Dim lRet As Long

        lRet = caoRob.Execute("OutRange", "P(400, 400, 300, 180, 0, 180, 5)" )
-----------------------------------------------------------------------------------------------------------------------------
```

5.2.28.13. CaoRobot::Execute("MPS") command

Transform an operation speed in mm/sec to a SPEED command value in %.

Syntax Mps(<mps>)

<mps> : [in] Speed value in mm/sec (VT_R4)

Return value : SPEED command value in % (VT_R4)

Example Transform an absolute speed to a relative speed.

```
-----------------------------------------------------------------------------------------------------------------------------
        Dim vSp As Variant

        vSp = caoRob.Execute("MPS",   200.0 ) ‘ 200.0 mm/sec
        caoRob.Speed -1, vSP
-----------------------------------------------------------------------------------------------------------------------------
```

5.2.28.14. CaoRobot::Execute("RPM") command

Transform a rotation speed in rpm to a SPEED command value in %.

Syntax Rpm(<Axis>, <rpm>)

<Axis> : [in] Axis number (VT_I4)

<rpm> : [in] Rotation speed in rpm (VT_R4)

Return value : SPEED command value in % (VT_R4)

Example Transform a rotation speed in RPM to a relative speed in %.

```
Dim vSp As Variant

vSp = g_caoRobot.Execute("RPM", Array(1, 60)) ‘ Axis 1, 60.0 rpm
caoRob.Speed -1, vSP
```

5.2.28.15. CaoRobot::Execute("CurPos") command

Get the current position as P type data.

Syntax CurPos( )

Argument : None

Return value : P type (VT_VARIANT[VT_R8|VT_ARRAY:7 element])

This is equivalent to a value that can be acquired in the system variable "@Current_Position".

Example

```
Dim vResult As Variant

vResult = caoRob.Execute("CurPos" )     ' Get current position
```

5.2.28.16. CaoRobot::Execute("DestPos") command

Get the target position as P type data.

Syntax DestPos( )

Argument : None

Return value : P type (VT_VARIANT[VT_R8|VT_ARRAY:7 element])

This is equivalent to a value that can be acquired in the system variable "@Dest_Position".

Example

```
Dim vResult As Variant

vResult = caoRob.Execute("DestPos" )     ' Get target position
```

5.2.28.17. CaoRobot::Execute("CurJnt") command

Get the current position as J type data.

Syntax CurJnt( )

Argument : None

Return value : J type (VT_VARIANT[VT_R8|VT_ARRAY:8 element])

This is equivalent to a value that can be acquired in the system variable "@Current_Angle".

Example

```
Dim vResult As Variant
vResult = caoRob.Execute("CurJnt" )     ' Get current position
```

5.2.28.18. CaoRobot::Execute("DestJnt") command

Get the target position as J type data.

Syntax DestPos( )

Argument	:	None
Return value	:	P type (VT_VARIANT[VT_R8|VT_ARRAY:8 element])

This is equivalent to a value that can be acquired in the system variable "@Dest_Angle".

Example

```
Dim vResult As Variant
vResult = caoRob.Execute("DestJnt" )     ' Get target position
```

5.2.28.19. CaoRobot::Execute("CurTrn") command

Get the current position as T type data.

Syntax CurTrn( )

Argument	:	None
Return value	:	T type (VT_VARIANT[VT_R8|VT_ARRAY:10 element])

This is equivalent to a value that can be acquired in the system variable "@Current_Trans".

Example

```
Dim vResult As Variant
vResult = caoRob.Execute("CurTrn" )     ' Get current position
```

5.2.28.20. CaoRobot::Execute("DestTrn") command

Get the target position as T type data.

Syntax DestTrn( )

Argument	:	None
Return value	:	T type (VT_VARIANT[VT_R8|VT_ARRAY:10 element])

This is equivalent to a value that can be acquired in the system variable "@Dest_Trans".

Example

```
----------------------------------------------------------------------------------------------------------
        Dim vResult As Variant
        vResult = caoRob.Execute("DestTrn" )    ' Get target position
----------------------------------------------------------------------------------------------------------
```

5.2.28.21. CaoRobot::Execute("CurFig") command

Get the Fig value that indicates the current posture.

Syntax CurFig( )

Argument	:	None
Return value	:	Fig value (VT_14)

Example

```
----------------------------------------------------------------------------------------------------------
        Dim vFig As Variant
        vFig = caoRob.Execute("CurFig" )    ' Get current Fig
----------------------------------------------------------------------------------------------------------
```

5.2.28.22. CaoRobot::Execute("StartLog") command

Stop recording logs when the allowable number of logs for sampling is reached since the execution of StartLog after control log recording is started by CLearLog.

Syntax StartLog( )

Argument	:	None
Return value	:	None

Execute the ClearLog command before this command to enable recording.

Example

```
----------------------------------------------------------------------------------------------------------
        caoRob.Execute"StartLog"
----------------------------------------------------------------------------------------------------------
```

5.2.28.23. CaoRobot::Execute("StopLog") command

The execution of StopLog stops recording control logs after control log recording is started by CLearLog.

Syntax StopLog( )

Argument	:	None
Return value	:	None

Execute the ClearLog command before this command to enable recording.

Example

```
------------------------------------------------------------------------------------------------
          caoRob.Execute"StopLog"
------------------------------------------------------------------------------------------------
```

5.2.28.24. CaoRobot::Execute("ClearLog") command

Start control log recording.

Syntax ClearLog( )

Argument	:	None
Return value	:	None

Clear the control log data and start sampling to the ring buffer.

Example

```
------------------------------------------------------------------------------------------------
          caoRob.Execute"ClearLog"
------------------------------------------------------------------------------------------------
```

5.2.28.25. CaoRobot::Execute("Motor") command

Turn ON/OFF the motor.

Syntax Motor (<State> [,<NoWait>])

State	:	[in] Motor status (VT_I4) 0: Motor OFF 1: Motor ON
NoWait	:	[in] Completion wait (VT_I4) 0:Wait for completion (default) 1: Do not wait for completion
Return value	:	None

Example

```
------------------------------------------------------------------------------------------------
          caoRob.Execute"Motor",Array(1,0) Turn on motor and wait for completion of motor ON process
------------------------------------------------------------------------------------------------
```

5.2.28.26. CaoRobot::Execute("ExtSpeed") command

Set the external speed, acceleration, and deceleration.

Syntax ExtSpeed (<Speed> [,<Accel> [,<Decel>]])

Speed	:	[in] External speed (VT_R4)
Accel	:	[in] External acceleration (VT_R4) -1 (default) Do not change the current setting
Decel	:	[in] External deceleration (VT_R4) -1 (default) Do not change the current setting
Return value	:	None

Example

```
----------------------------------------------------------------------------------------------------------------
        caoRob.Execute"ExtSpeed", Array(50.0, 25.0, 25.0 )
                                        ‘ External speed = 50%, acceleration = 25%, deceleration = 25%
----------------------------------------------------------------------------------------------------------------
```

5.2.28.27. CaoRobot::Execute("TakeArm") command

Request to get control authority.

This command corresponds to TAKEARM instruction of PacScript language.

Syntax TakeArm ([<ArmGroup> , [<Keep>]])

ArmGroup	:	[in] Arm group number (VT_I4) 0 to 31 (0 by default)
Keep	:	[in] Default value (VT_I4) 0:Set the speed to 100 and the tool and work numbers to 0. 1:Maintain the current speed and tool and work numbers. (0 by default)
Return value	:	None

Example

```
----------------------------------------------------------------------------------------------------------------
        caoRob.Execute"Takearm",Array(0,0)
----------------------------------------------------------------------------------------------------------------
```

5.2.28.28. CaoRobot::Execute("ForceTakeArm") command

Request to get control authority.

This command corresponds to TAKEARM instruction of PacScript language.

The control authority is forcibly acquired if Takearm is executed in other than PacScript. A runtime error occurs if Takearm is executed in PacScript.

Syntax ForceTakeArm ([<ArmGroup> , [<Keep>]])

ArmGroup	:	[in] Arm group number (VT_I4) 0 to 31 (0 by default)
Keep	:	[in] Default value (VT_I4) 0: Set the speed to 100 and the tool and work numbers to 0. 1: Maintain the current speed and tool and work numbers. (0 by default)
Return value	:	None

Example

```
caoRob.Execute"ForceTakearm",Array(0,0)
```

5.2.28.29. CaoRobot::Execute("GiveArm") command

Request to release control authority.

This command corresponds to GIVEARM instruction of PacScript language.

Syntax GiveArm ( )

Argument	:	None
Return value	:	None

Example

```
caoRob.Execute"GiveArm"
```

5.2.28.30. CaoRobot::Execute("Draw") command

Execute the relative movement designated in the work coordinate system

This command corresponds to DRAW instruction of PacScript language.

Syntax Draw (<lComp>,<vntPose>[,<strOpt>])

lComp	:	[in] Interpolation method (VT_I4) 1: PTP motion 2: CP motion
vntPose	:	[in] Distance (POSEDATA type, C1 format) "[<@pass start displacement>]<Parallel movement distance>"
strOpt	:	[in] Motion option (VT_BSTR) [SPEED=n][,ACCEL=n][,DECEL=n][,NEXT]

SPEED (S): Designate the movement speed. The meaning is the same as the SPEED statement.

ACCEL: Designate the acceleration. The meaning is the same as the ACCEL statement.

DECEL: Designate the deceleration. The meaning is the same as the DECEL statement.

NEXT: Asynchronous execution option

Return value : None

Example

```
caoRob.Execute"Draw", Array(1,"V0")
caoRob.Execute"Draw", Array(2,"V(100, 100, 100)")
```

5.2.28.31. CaoRobot::Execute("Approach") command

Move to the approach position that is as far to the reference position as the specified distance.

This command corresponds to APPROACH instruction of PacScript language.

Syntax Approach (<lComp>,<vntPoseBase>,<vntPoseLen>[,<strOpt>])

lComp : [in] Interpolation method (VT_I4)
1: PTP motion
2: CP motion

vntPoseBase : [in] Reference position (POSEDATA type, C0 format)
"<Position: P, T, or J type>"
An error occurs if the pass start displacement is specified in POSEDATA type C0 format.

vntPoseLen : [in] Approach length (POSEDATA type, C2 format)
"[Pass start displacement]<Value (mm)>"

strOpt : [in] Motion option (VT_BSTR)
[SPEED=n][,ACCEL=n][,DECEL=n][,NEXT]

SPEED (S): Designate the movement speed. The meaning is the same as the SPEED statement.

ACCEL: Designate the acceleration. The meaning is the same as the ACCEL statement.

DECEL: Designate the deceleration. The meaning is the same as the DECEL statement.

NEXT: Asynchronous execution option

Return value : None

Example

```
--------------------------------------------------------------------------------------------------------------------------------
        caoRob.Execute"Approach",Array(1," P1","@P 100","S=50")
        caoRob.Execute"Approach",Array(2," P(400, 200, 350, 180, 0, 180, 5)","@E 56.8","S=30, NEXT")
--------------------------------------------------------------------------------------------------------------------------------
```

5.2.28.32. CaoRobot::Execute("Depart") command

Move from the current position along the Z axis in the tool coordinates.

This command corresponds to DEPART instruction of PacScript language.

Syntax Depart (<lComp>,<vntPoseLen>[,<strOpt>])

lComp : [in] Interpolation method (VT_I4)

1: PTP motion

2: CP motion

vntPoseLen : [in] Depart length (POSEDATA type, C2 format)

"[Pass start displacement]<Value (mm)>"

strOpt : [in] Motion option (VT_BSTR)

[SPEED=n][,ACCEL=n][,DECEL=n][,NEXT]

SPEED (S): Designate the movement speed. The meaning is the same as the SPEED statement.

ACCEL: Designate the acceleration. The meaning is the same as the ACCEL statement.

DECEL: Designate the deceleration. The meaning is the same as the DECEL statement.

NEXT: Asynchronous execution option

Return value : None

Example

```
--------------------------------------------------------------------------------------------------------------------------------
        caoRob.Execute"Depart",Array(1,"@P 100","S=50")
        caoRob.Execute"Depart",Array(2"@E 56.8","S=30, NEXT")
--------------------------------------------------------------------------------------------------------------------------------
```

5.2.28.33. CaoRobot::Execute("DriveEx") command

Execute the relative motion of each axis.

This command corresponds to DRIVE instruction of PacScript language.

Syntax DriveEx (<vntPoses> [, < strOpt >])

vntPoses : [in] Axis number and distance (POSEDATA type, C3 format)
Specify the desired axes and distances in POSEDATA type for eight axes at the maximum.

strOpt : [in] Motion option (VT_BSTR)
[SPEED=n][,ACCEL=n][,DECEL=n][,NEXT]
SPEED (S): Designate the movement speed. The meaning is the same as the SPEED statement.
ACCEL: Designate the acceleration. The meaning is the same as the ACCEL statement.
DECEL: Designate the deceleration. The meaning is the same as the DECEL statement.
NEXT: Asynchronous execution option

Return value : None

Example

```
--------------------------------------------------------------------------------
        vntPoses = "@0 (1, 10), (2, 10)"
        caoRob.Execute "DriveEX", Array(v, "S=10, NEXT")
--------------------------------------------------------------------------------
```

5.2.28.34. CaoRobot::Execute("DriveAEx") command

Execute the absolute motion of each axis.

This command corresponds to DRIVEA instruction of PacScript language.

Syntax DriveAEx (<vntPoses> [, < strOpt >])

vntPoses : [in] Axis number and distance (POSEDATA type, C3 format)
Specify the desired axes and axis coordinates in POSEDATA type for eight axes at the maximum.

strOpt : [in] Motion option (VT_BSTR)

[SPEED=n][,ACCEL=n][,DECEL=n][,NEXT]

SPEED (S):Designate the movement speed. The meaning is the same as the SPEED statement.

ACCEL:Designate the acceleration. The meaning is the same as the ACCEL statement.

DECEL:Designate the deceleration. The meaning is the same as the DECEL statement.

NEXT:Asynchronous execution option

Return value : None

Example

```
--------------------------------------------------------------------------------------------------------------------------
        vntPose1 = Array(Array(1, 10), -1, "@0")
        vntPose2 = Array(Array(2, 10), -1)
        vntPoses = Array(vntPose1, vntPose2)
        caoRob.Execute "DriveAEX", Array(vntPoses, "S=10, NEXT")
        caoRob.Execute "DriveAEX", Array("@0 (1,10), (2,10)", "S=10, NEXT")
--------------------------------------------------------------------------------------------------------------------------
```

5.2.28.35. CaoRobot::Execute("RotateH") command

Execute rotary motion by taking an approach vector as an axis.

This command corresponds to ROTATEH instruction of PacScript language.

Syntax RotateH (<vntPoseAxis> [,<strOpt>])

vntPoseAxis : [in] Relative rotation angle around approach vector (POSEDATA type, C2 format)

"[Pass start displacement]<Value (degree)>"

strOpt : [in] Motion option (VT_BSTR)

[SPEED=n][,ACCEL=n][,DECEL=n][,NEXT]

SPEED (S):Designate the movement speed. The meaning is the same as the SPEED statement.

ACCEL:Designate the acceleration. The meaning is the same as the ACCEL statement.

DECEL:Designate the deceleration. The meaning is the same as the DECEL statement.

NEXT:Asynchronous execution option

Return value : None

Example

```
-----------------------------------------------------------------------------------------------------------------
        caoRob.Execute"RotateH", Array("@P 32.5" ,"S=50" )
-----------------------------------------------------------------------------------------------------------------
```

5.2.28.36. CaoRobot::Execute("Arrive") command

Wait for the robot to reach the defined motion ratio.

This command corresponds to ARRIVE instruction of PacScript language.

Syntax Arrive (<Motion ratio>)

Argument : [in] (VT_R4) Motion ratio

Return value : None

Example

```
-----------------------------------------------------------------------------------------------------------------
        caoRob.Move 1,"P1","Next"      ' Asynchronous execution
        caoRob.Execute"Arrive", 50     ' Wait for 50% completion
-----------------------------------------------------------------------------------------------------------------
```

5.2.28.37. CaoRobot::Execute("MotionSkip") command

Abort the robot motion in progress.

This command corresponds to MOTIONSKIP instruction of PacScript language.

Syntax MotionSkip ([<ArmGroup>[, <Parameter>]])

ArmGroup : [in] Arm group number (VT_I4)

-1 (default): Current arm group under control

Parameter : [in] Operation continuation pattern (VT_I4)

0 (default): Specify the pass start displacement as @0 and connect it with the maximum deceleration.

1: Specify the pass start displacement as @P and connect it with the maximum deceleration.

2: Specify the pass start displacement as @0 and connect it with the set deceleration.

3:1: Specify the pass start displacement as @P and connect it with the set deceleration.

Return value : None

Example

```
        caoRob.Execute "MotionSkip", Array(0, 1)
```

5.2.28.38. CaoRobot::Execute("MotionComplete") command

Judge whether the robot motion command or robot motion is complete.

This command corresponds to MOTIONCOMPLETE instruction of PacScript language.

Syntax MotionComplete ([<ArmGroup> [,<Mode>]])

ArmGroup : [in] Arm group number (VT_I4)
-1 (default): Current arm group under control

Mode : [in] Mode
0 (default): Get motion command completion status
1: Get motion completion status

Return value : [out] Status <VT_BOOL>
in Mode 0
Operation command is complete: VARIANT_TRUE,
Running, suspended, continue-stopped: VARIANT_FALSE
in Mode 1
Robot stopped: VARIANT_TRUE,
Robot running: VARIANT_FALSE

Example Asynchronous motion and wait for completion

```
        caoRob.Move 1,"P1","Next" ' Asynchronous motion to P1
        Do
          ' <Processing during movement>

        Loop While( Not caoRob.Execute("MotionComplete", Array(-1, 1) )) ' Operation completion check
```

5.2.28.39. CaoRobot::Execute("CurTool") command

Get the current tool number.

Syntax CurTool ( )

Argument : None

Return value : Current tool number (VT_I4)

Example

```
-------------------------------------------------------------------------------------------------------
        Debug.Print caoRob.Execute("CurTool")
-------------------------------------------------------------------------------------------------------
```

5.2.28.40. CaoRobot::Execute("GetToolDef") command

Get the tool definition specified by the tool number.

Syntax GetToolDef (<ToolNo>)

ToolNo	:	[in] Tool number (VT_I4)
Return value	:	Tool definition (VT_R8|VT_ARRAY)
		X, Y, Z, RX, RY, RZ

Example

```
-------------------------------------------------------------------------------------------------------
        Dim vVal As Variant
        vVal = caoRob.Execute(敵 etToolDef", 1)
        Debug.Print"X=・  & vVal(0) &", Y=・  & vVal(1) &", Z=・  & vVal(2)
        Debug.Print"RX=・  & vVal(3) &", RY=・  & vVal(4) &", RZ=・  & vVal(5)
-------------------------------------------------------------------------------------------------------
```

5.2.28.41. CaoRobot::Execute("SetToolDef") command

Set the tool definition.

Syntax SetToolDef (<ToolNo>, <ToolDef>)

ToolNo	:	[in] Tool number (VT_I4)
ToolDef		[in] Tool definition (P type Fig is ignored.)
		X, Y, Z, RX, RY, RZ
Return value	:	None

Example

```
-------------------------------------------------------------------------------------------------------
        caoRobot.Execute "SetToolDef", Array(1, "P2")
        caoRobot.Execute "SetToolDef", Array(2, "P(100, 200, 300, 180, 0, 180)")
-------------------------------------------------------------------------------------------------------
```

5.2.28.42. CaoRobot::Execute("CurWork") command

Get the current work number.

Syntax CurWork ( )

Argument	:	None
Return value	:	Current work number (VT_I4)

Example

```
        Debug.Print caoRob.Execute(鼎 urWork")
```

5.2.28.43. CaoRobot::Execute("GetWorkDef") command

Get the work definition specified by the work number.

Syntax GetWorkDef (<WorklNo>)

WorkNo	:	[in] Work number (VT_I4)
Return value	:	Work definition (VT_R8|VT_ARRAY)
		X, Y, Z, RX, RY, RZ, attributes

Example

```
        Dim vVal As Variant
        vVal = caoRob.Execute("GetWorkDef", 1)
        Debug.Print"X= " & vVal(0) &", Y= " & vVal(1) &", Z= " & vVal(2)
        Debug.Print"RX= " & vVal(3) &", RY= " & vVal(4) &", RZ= " & vVal(5)
        Debug.Print"ATTR= " & vVal(6)
```

5.2.28.44. CaoRobot::Execute("SetWorkDef") command

Set the work definition.

Syntax SetWorkDef (<WorkNo>, <WorkDef>)

WorkNo	:	[in] Work number (VT_I4)
WorkDef	:	[in] Work definition (P type Fig is ignored.)
		X, Y, Z, RX, RY, RZ
Return value	:	None

Example

```
        caoRobot.Execute "SetWorkDef", Array(1, "P2")
        caoRobot.Execute "SetWorkDef", Array(2, "P(100, 200, 300, 180, 0, 180)")
```

5.2.28.45. CaoRobot::Execute("GetAreaDef") command

Get the area definition with the specified area number.

Syntax GetAreaDef (<AreaNo>)

Argument : [in] Work number (VT_I4)

Return value : Area definition (VT_R8|VT_ARRAY)
X,Y,Z,RX,RY,RZ,DX,DY,DZ,IO,Position,Error,Time,DRX,DRY, DRZ,Margin,Position1,Margin1,Position2,Margin2,Position3, Margin3,Position4,Margin4,Position5,Margin5,Position6,Margin6, Position7,Margin7,Position8,Margin8,Enable

Example

```
--------------------------------------------------------------------------------------------------------------------------------
        Debug.Print caoRob.Execute("GetAreaDef", 1)
--------------------------------------------------------------------------------------------------------------------------------
```

5.2.28.46. CaoRobot::Execute("SetAreaDef") command

Set the area parameter.

Syntax SetAreaDef (<Area number>, <Center>, <Size>, <I/O number>, <Variable storage number>[,<Area detection setting>])

SetAreaDef (<Area number>, <Area definition>)

Argument : Format 1:
[in] Area number (VT_I4)
[in] Position and rotation (inclination) of center point (P type)
[in] Area size (V type)
[in] I/O number (VT_I4)
[in] Variable storage number (VT_I4)
[in] Area detection setting (VT_I4)
Format 2:
[in] Area definition (VT_R8|VT_ARRAY)
X,Y,Z,RX,RY,RZ,DX,DY,DZ,IO,Position[,Error,Time,DRX,DRY, DRZ,Margin,Position1,Margin1,Position2,Margin2,Position3, Margin3,Position4,Margin4,Position5,Margin5,Position6,Margin6, Position7,Margin7,Position8,Margin8,Enable]

Return value : None

Example

```
--------------------------------------------------------------------------------------------------------------------------------
        caoRobot.Execute "SetAreaDef", Array(1, "P0", "V0", 24, 0, 0)
        caoRobot.Execute "SetAreaDef", Array(2, "P(400, 250, 140, 180, 0, 180)", "V(200, 125, 70)", 24, 0, 0)
--------------------------------------------------------------------------------------------------------------------------------
```

5.2.28.47. CaoRobot::Execute("SetArea") command

Enable the area check.

Syntax SetArea (<AreaNum>)

<AreaNum> : Area number (VT_I4)

Return value : None

Example

```
------------------------------------------------------------------------------------------------------------------------
        caoRobot.Execute "SetArea", 1
------------------------------------------------------------------------------------------------------------------------
```

5.2.28.48. CaoRobot::Execute("ResetArea") command

Disable the area check.

Syntax ResetArea (<AreaNum>)

<AreaNum> : Area number (VT_I4)

Return value : None

Example

```
------------------------------------------------------------------------------------------------------------------------
        caoRobot.Execute "ResetArea", 1
------------------------------------------------------------------------------------------------------------------------
```

5.2.28.49. CaoRobot::Execute("AreaSize") command

Return the size (each side length) of a check area as the vector type.

Syntax AreaSize (<AreaNum>)

<AreaNum> : Area number (VT_I4)

Return value : Area size (VT_R8|VT_ARRAY）
X,Y,Z

Example

```
------------------------------------------------------------------------------------------------------------------------
        Dim vVal As Variant
        vVal = caoRob.Execute("AreaSize", 1 )    ' Get size of Area1
        Debug.Print"X=・  & vVal(0) &", Y=・  & vVal(1) &", Z=・  & vVal(2)
------------------------------------------------------------------------------------------------------------------------
```

5.2.28.50. CaoRobot::Execute("GetAreaEnabled") command

Get the area enabled or disabled status.

Syntax GetAreaEnabled (<AreaNum>)

<AreaNum>	:	[in] Area number (VT_I4)
Return value	:	Enabled/disabled (VT_BOOL)

Example

```
--------------------------------------------------------------------------------------------------------------------------
        Debug.Print caoRob.Execute("GetAreaEnabled", 1 )    ' Get enabled/disabled status of Area1
--------------------------------------------------------------------------------------------------------------------------
```

5.2.28.51. CaoRobot::Execute("SetAreaEnabled") command

Set the area enabled or disabled status.

Syntax SetAreaEnabled (<AreaNum>, <Enable/disable>)

<AreaNum>	:	[in] Area number (VT_I4)
<Enable/disable>	:	[in] Area number (VT_BOOL)
Return value	:	None

Example

```
--------------------------------------------------------------------------------------------------------------------------
        caoRob.Execute"SetAreaEnabled", Array( 1, True )    ' Set Area1 enabled
--------------------------------------------------------------------------------------------------------------------------
```

5.2.28.52. CaoRobot::Execute("GetRobotTypeName") command

Get the robot type.

Syntax GetRobotTypeName ( )

Argument	:	None
Return value	:	Robot type (VT_BSTR)

Example

```
--------------------------------------------------------------------------------------------------------------------------
        Debug.Print caoRob.Execute("GetRobotTypeName" )
--------------------------------------------------------------------------------------------------------------------------
```

5.2.29. CaoTask::AddVariable method

The argument of the AddVariable method of the CaoTask class specifies the system variable name.

Refer to Table 5-14 for the list of implemented system variables.

5.2.30. CaoTask::get_VariableNames property

Get a list of variable names and system variable names that can be specified by the AddVariable method.

5.2.31. CaoTask::Start method

Run the PAC program that supports the object.

The following shows the argument specifications of Start.

Syntax Start <lMode:LONG>, <bstrOpt:BSTR>

lMode	:	[in]	Start mode 1: One cycle execution, 2: Continuous execution, 3: Step forward, 4: Step backward
bstrOpt	:	[in]	Option (not used)

5.2.32. CaoTask::Stop method

Stop the PAC program that supports the object.

The following shows the argument specifications of Stop.

Syntax Stop <lMode:LONG>, <bstrOpt:BSTR>

lMode	:	[in]	Stop mode 0: Default stop, 1: Instant stop, 2: Step stop, 3: Cycle stop, 4: Initialized stop
bstrOpt	:	[in]	Option (not used)

"0: default stop" is the same as "1: Instant stop".

5.2.33. CaoTask::Execute method

Execute the command.

The arguments of the Execute method specify a command as a BSTR and a parameter as a VARIANT array.

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

bstrCmd	:	[in]	Command name
vntParam	:	[in]	Parameter
vntRet		[out]	Return value

If a method not defined in this class is called using the runtime binding function, the Execute method is automatically called according to the following specifications:

```
 vntRet = Obj.CommandName Param1, Param2, ...
   ↓
 vntRet = Obj.Execute("CommandName", Array(Param1, Param2, ... ) )
```

1. The command name is passed as a BSTR string to the first argument.

2. All the parameters are passed as a VARIANT array to the second argument.

Example

```
-----------------------------------------------------------------------------------------------------------------------------
        Dim vRes As Variant
        Dim caoTsk As CaoTask

        Set caoTsk = caoCtrl.AddTask("pro1" )
        vRes = caoTsk.Execute("GetStatus" )    ' Get task status
-----------------------------------------------------------------------------------------------------------------------------
```

The list shows available commands.

Table 5-9 List of commands of CaoTask::Execute

Category	Command name	Function	
Task status			
	GetStatus	Get the task status.	P.82
Priority			
	GetThreadPriority	Get the priority.	P.82
	SetThreadPriority	Set the priority.	P.83

5.2.33.1. CaoTask::Execute("GetStatus") command

Get the status of a task.

Syntax GetStatus( )

Argument	:	None	
Return value	:	Status（VT_I4）	
		0:TASK_NON_EXISTENT,	Task non-existent
		1:TASK_SUSPEND,	Hold-stopped
		2:TASK_READY,	Ready
		3:TASK_RUN,	Running
		4:TASK_STEPSTOP,	Step-stopped

Example

```
-----------------------------------------------------------------------------------------------------------------------------
        Dim lStatus As Long
        lStatus = caoTsk.Execute("GetStatus" )
-----------------------------------------------------------------------------------------------------------------------------
```

5.2.33.2. CaoTask::Execute("GetThreadPriority") command

Get the execution priority of a task.

Syntax GetThreadPriority( )

Argument : None

Return value : Priority (VT_I4)
2:THREAD_PRIORITY_HIGHEST
1:THREAD_PRIORITY_ABOVE_NORMAL
0:THREAD_PRIORITY_NORMAL
-1:THREAD_PRIORITY_BELOW_NORMAL
-2:THREAD_PRIORITY_LOWEST

5.2.33.3. CaoTask::Execute("SetThreadPriority") command

Set the execution priority of a task.

Syntax SetThreadPriority([<lPriority>])

<lPriority> : Priority (VT_I4)
2:THREAD_PRIORITY_HIGHEST
1:THREAD_PRIORITY_ABOVE_NORMAL
0:THREAD_PRIORITY_NORMAL
-1:THREAD_PRIORITY_BELOW_NORMAL
-2:THREAD_PRIORITY_LOWEST
If the argument is omitted, 0 is assumed to be specified.

Return value : None

5.2.34. CaoVariable::get_Value property

Get the value of the variable corresponding to the object.

For the details about the variable implementation status and data type, refer to"5.3 Variable list".

5.2.35. CaoVariable::put_Value property

Set the value of the variable corresponding to the object.

For the details about the variable implementation status and data type, refer to"5.3 Variable list".

5.2.36. CaoExtension::Execute method

Execute the command of an extended function.

The arguments of the Execute method specify a command as a BSTR and a parameter as a VARIANT array.

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

bstrCmd : [in] Command name

vntParam : [in] Parameter

vntRet : [out] Return value

If a method not defined in this class is called using the runtime binding function, the Execute method is automatically called according to the following specifications:

```
vntRet = Obj.CommandName Param1, Param2, ...
    ↓
vntRet = Obj.Execute( “CommandName”, Array(Param1, Param2, ... ) )
```

1. The command name is passed as a BSTR string to the first argument.
2. All the parameters are passed as a VARIANT array to the second argument.

Example Hand object operation

```
-----------------------------------------------------------------------------------------------------------------------------
        Dim caoExt As CaoExtension

        Set caoExt = caoCtrl.AddExtension( “Hand0” )
        CaoExt.Execute “Motor”, true Electric end-effector motor
        caoExt.Execute “Org”                    ' Origin return

        caoExt.Execute “Chuck”, 0               ' Execute chuck operation

        caoExt.Execute “UnChuck”, 1             ' Execute unchuck operation

-----------------------------------------------------------------------------------------------------------------------------
```

The list shows available commands.

Table 5-10 CaoController::Execute method command list

Category	Command name	Function	
Hand object			
	Chuck	Execute chuck operation.	P.85
	UnChuck	Execute unchuck operation.	P.85
	Motor	Turn ON/OFF the motor power.	P.86
	Org	Execute origin return.	P.86
	MoveP	Execute point operation.	P.87
	MoveA	Execute absolute position movement.	P.87
	MoveR	Execute relative position movement.	P.88
	MoveAH	Execute hold operation in acceleration/deceleration absolute position movement.	P.88
	MoveRH	Execute hold operation in acceleration/deceleration relative position movement.	P.88
	MoveH	Execute hold operation in constant speed movement.	P.89

MoveZH	Execute hold operation in zone-specific constant speed movement.	P.89
Stop	Stop the operation.	P.90
CurPos	Get the current position.	P.90
GetPoint	Get the point data element.	P.91
get_EmgState	Get the emergency stop input status.	P.91
get_ZonState	Get the ZON signal status.	P.91
get_OrgState	Get the origin return status.	P.92
get_HoldState	Get the hold status.	P.92
get_InposState	Get the INPOS status.	P.93
get_Error	Get the electric end-effector error information.	P.93
get_BusyState	Get the operation status.	P.94
get_MotorState	Get the motor power status.	P.94

5.2.36.1. Hand object - CaoExtension::Execute("Chuck") command

Execute chuck operation according to the specified point data.

Syntax Chuck(<No>)

Argument : No [in] Point number (0 to 31) [VT_I4]

Return value : None

Execute the work hold operation according to the settings in the specified point data.

The hold operation must be set in the point data in advance.

The command cannot be executed if the operation status is busy (unless get_BusyState is 0).

Example

```
------------------------------------------------------------------------------------------------------------------------------------
        caoExt.Execute “Chuck”, 0
------------------------------------------------------------------------------------------------------------------------------------
```

5.2.36.2. Hand object - CaoExtension::Execute("UnChuck") command

Execute unchuck operation according to the specified point data.

Syntax UnChuck(<No>)

Argument	:	No [in] Point number (0 to 31) [VT_I4]
Return value	:	None

Move the electric end-effector from the hold status to the preset position according to the settings in the specified point data.

The movement operation must be set in the point data in advance.

The command cannot be executed if the operation status is busy (unless get_BusyState is 0).

Example

```
--------------------------------------------------------------------------------
        caoExt.Execute “UnChuck”, 1
--------------------------------------------------------------------------------
```

5.2.36.3. Hand object - CaoExtension::Execute("Motor") command

Turn ON/OFF the motor power.

Syntax Motor (<State>)

Argument	:	State [in] Motor status [VT_I4] 0: Motor OFF Other than 0: Motor ON
Return value	:	None

Turn ON or OFF the motor of the electric end-effector. While the electric end-effector is in an emergency stop status, executing the motor-ON command does not turn ON the motor. While the electric end-effector is already in motor-ON status, executing the motor-ON command has no effect, and the electric end-effector's motor remains ON.

The command cannot be executed if the operation status is busy (unless get_BusyState is 0).

Example

```
--------------------------------------------------------------------------------
        caoExt.Execute “Motor”, 1
--------------------------------------------------------------------------------
```

5.2.36.4. Hand object - CaoExtension::Execute("Org") command

Execute origin return.

Syntax Org( )

Argument	:	None
Return value	:	None

Execute origin return.

This command must be executed at least once after the electric end-effector power is turned ON. If an error occurs, origin return must also be executed after the error is reset.

Before origin return is completed, executing an operation command of the electric end-effector causes an error.

The command cannot be executed if the operation status is busy (unless get_BusyState is 0).

Example

```
--------------------------------------------------------------------------------
        caoExt.Execute “Org”
--------------------------------------------------------------------------------
```

5.2.36.5. Hand object - CaoExtension::Execute("MoveP") command

Execute point operation.

Syntax MoveP (<No>)

Argument	:	No [in] Point number (0 to 31) [VT_I4]
Return value	:	None

Execute the end-effector operation according to the settings in the specified point data.

The operation must be set in the point data in advance.

The command cannot be executed if the operation status is busy (unless get_BusyState is 0).

Example

```
--------------------------------------------------------------------------------
        caoExt.Execute “MoveP” , 1
--------------------------------------------------------------------------------
```

5.2.36.6. Hand object - CaoExtension::Execute("MoveA") command

Execute absolute position movement operation.

Syntax MoveA (<Pos>, <Speed>)

Argument	:	Pos	[in] Position (-999.90 to 999.90 [mm])	[VT_R4]
	:	Speed	[in] Speed (20 to 100[%])	[VT_I4]
Return value	:	None		

Execute the absolute position movement operation of the end-effector to the specified position at the specified speed.

The command cannot be executed if the operation status is busy (unless get_BusyState is 0).

Example

```
--------------------------------------------------------------------------------
        caoExt.Execute “MoveA” , Array(5.00, 20)
--------------------------------------------------------------------------------
```

5.2.36.7. Hand object - CaoExtension::Execute("MoveR") command

Execute absolute position movement operation.

Syntax MoveR (<Pos>, <Speed>)

Argument	:	Pos	[in] Position (-999.90 to 999.90 [mm])	[VT_R4]
	:	Speed	[in] Speed (20 to 100[%])	[VT_I4]
Return value	:	None		

Execute the relative position movement operation of the end-effector to the specified position at the specified speed.

The command cannot be executed if the operation status is busy (unless get_BusyState is 0).

Example

```
------------------------------------------------------------------------
        caoExt.Execute “MoveR” , Array(-3.00, 100)
------------------------------------------------------------------------
```

5.2.36.8. Hand object - CaoExtension::Execute("MoveAH") command

Execute the absolute position hold operation with acceleration/deceleration.

Syntax MoveAH (<Pos>, <Speed>, <Force>)

Argument	:	Pos	[in] Position (-999.90 to 999.90 [mm])	[VT_R4]
	:	Speed	[in] Speed (20 to 100[%])	[VT_I4]
	:	Force	[in] Hold force (30 to 100[%])	[VT_I4]
Return value	:	None		

Execute the absolute position movement and hold operation of the electric end-effector at the specified position and speed with the specified hold force.

The command cannot be executed if the operation status is busy (unless get_BusyState is 0).

Example

```
------------------------------------------------------------------------
        caoExt.Execute “MoveAH” , Array(2.50, 100, 100)
------------------------------------------------------------------------
```

5.2.36.9. Hand object - CaoExtension::Execute("MoveRH") command

Execute the relative position hold operation with acceleration/deceleration.

Syntax MoveRH (<Pos>, <Speed>, <Force>)

Argument	:	Pos	[in] Position (-999.90 to 999.90 [mm])	[VT_R4]
	:	Speed	[in] Speed (20 to 100[%])	[VT_I4]
	:	Force	[in] Hold force (30 to 100[%])	[VT_I4]
Return value	:	None		

Execute the relative position movement and hold operation of the electric end-effector at the specified position and speed with the specified hold force.

The command cannot be executed if the operation status is busy (unless get_BusyState is 0).

Example

```
------------------------------------------------------------------------------------------------
        caoExt.Execute “MoveRH” , Array(2.50, 100, 100)
------------------------------------------------------------------------------------------------
```

5.2.36.10. Hand object - CaoExtension::Execute("MoveH") command

Execute hold operation in constant speed movement.

Syntax MoveH (<Speed>, <Force>, <Direct>)

Argument	:	Speed [in] Speed (20 to 50[%])	[VT_I4]
	:	Force [in] Hold force (30 to 100[%])	[VT_I4]
	:	Direct [in] Movement direction 0: Open direction Other than 0: Close direction	[VT_I4]
Return value	:	None	

Execute the constant speed movement and hold operation of the electric end-effector at the specified speed in the specified movement direction with the specified hold force.

The command cannot be executed if the operation status is busy (unless get_BusyState is 0).

Example

```
------------------------------------------------------------------------------------------------
        caoExt.Execute “MoveH” , Array(50, 100, 1)
------------------------------------------------------------------------------------------------
```

5.2.36.11. Hand object - CaoExtension::Execute("MoveZH") command

Execute hold operation in zone-specific constant speed movement.

Syntax MoveZH (<Speed>, <Force>, <Direct>)

Argument	:	ZON1 [in] ZON range 1 (-999.90 to 999.90 [mm])	[VT_R4]
	:	ZON2 [in] ZON range 2 (-999.90 to 999.90 [mm])	[VT_R4]
	:	Speed [in] Speed (20 to 50[%])	[VT_I4]
	:	Force [in] Hold force (30 to 100[%])	[VT_I4]
	:	Direct [in] Movement direction 0: Open direction Other than 0: Close direction	[VT_I4]

Return value : None

Execute the constant speed movement and hold operation of the electric end-effector in the specified ZON range at the specified speed in the specified movement direction with the specified hold force.

Once the end-effector is within the ZON range 1 and ZON range 2, get_ZonState becomes an in-range status (other than 0).

The command cannot be executed if the operation status is busy (unless get_BusyState is 0).

Example

```
caoExt.Execute “MoveZH” , Array(1.00, 4.00, 50, 100, 1)
```

5.2.36.12. Hand object - CaoExtension::Execute("Stop") command

Stop the operation.

Syntax Stop ()

Argument : None

Return value : None

While the electric end-effector is running, execute this command to stop the operation immediately.

Example

```
caoExt.Execute “Stop”
```

5.2.36.13. Hand object - CaoExtension::Execute("CurPos") command

Return the current position.

Syntax CurPos ()

Argument : None

Return value : [out] Current position [mm] [VT_R4]

Return the current position [mm] of the electric end-effector.

Depending on the timing, it takes 10 ms at the maximum.

Example

```
Dim handPos as Single
handPos = caoExt.Execute( “CurPos” )
```

5.2.36.14. Hand object - CaoExtension::Execute("GetPoint") command

Return the point data elements.

Syntax GetPoint (<No>, <Index>)

Argument	:	No	[in] Point number (0 to 31)	[VT_I4]
	:	Index	[in] Point data element (0 to 5)	[VT_I4]
Return value	:	[out]	Value of the specified element of the specified point data	
		0: Operation mode		
		1: Distance [mm]	[VT_R4]	
		2: Speed [mm]	[VT_I4]	
		3: Hold force [%]	[VT_I4]	
		4: ZON range 1 [mm]	[VT_R4]	
		5: ZON range 2 [mm]	[VT_R4]	

Return the value of the specified element of the specified point data.

Example

```
Dim Speed as Long
```

5.2.36.15. Hand object - CaoExtension::Execute("get_EmgState") command

Inform the emergency stop signal input status.

Syntax get_EmgState ()

Argument	:	None	
Return value	:	Emergency stop signal input status	[VT_I4]
		0: Emergency stop status	
		Other than 0: Emergency stop cleared status (The emergency stop input is short-circuited.)	

Return the emergency stop status of the electric end-effector.

Example

```
Dim State as Long
State = caoExt.Execute( “get_EmgState”)
```

5.2.36.16. Hand object - CaoExtension::Execute("get_ZonState") command

Inform the status whether the electric end-effector is positioned within the set range.

Syntax get_EmgState ()

Argument	:	None	
Return value	:	ZON status	[VT_I4]
		0: Positioned out of the range specification	
		Other than 0: Positioned between the range specifications 1 and 2	

Return the status whether the electric end-effector is positioned within the set range.

Example

```
--------------------------------------------------------------------------------
        Dim State as Long
        State = caoExt.Execute( “get_ZonState”)
--------------------------------------------------------------------------------
```

5.2.36.17. Hand object - CaoExtension::Execute("get_OrgState") command

Inform the origin return status.

Syntax get_OrgState ()

Argument	:	None	
Return value	:	Origin return status	[VT_I4]
		0: Origin return is not completed	
		Other than 0: Origin return is completed	

Inform the origin return status.

Example

```
--------------------------------------------------------------------------------
        Dim State as Long
        State = caoExt.Execute( “get_OrgState”)
--------------------------------------------------------------------------------
```

5.2.36.18. Hand object - CaoExtension::Execute("get_HoldState") command

Inform the hold status of the electric end-effector.

Syntax get_HoldState ()

Argument	:	None	
Return value	:	Hold status	[VT_I4]
		0: Not holding	
		Other than 0: Holding the work with the specified hold force	

Return the hold status of the electric end-effector.

Example

```
--------------------------------------------------------------------------------
        Dim State as Long
        State = caoExt.Execute( “get_HoldState”)
--------------------------------------------------------------------------------
```

5.2.36.19. Hand object - CaoExtension::Execute("get_InposState") command

Inform whether the end-effector is in the target position (INPOS status).

Syntax get_InposState ()

Argument	:	None	
Return value	:	INPOS status	[VT_I4]
		0: Out of the target position or currently moving	
		Other than 0: Within the target position range after origin return or positioning operation	

Return whether the end-effector is in the target position (INPOS status).

The target position range is determined by the "positioning completion distance" parameter.

Example

```
--------------------------------------------------------------------------------
        Dim State as Long
        State = caoExt.Execute( “get_InposState”)
--------------------------------------------------------------------------------
```

5.2.36.20. Hand object - CaoExtension::Execute("get_Error") command

Inform the error status of the electric end-effector.

Syntax get_Error ()

Argument	:	None	
Return value	:	Error code (decimal format data)	[VT_I4]
		0: Normal status	
		Other than 0: An error occurred. The value represents an error code.	

Return the error status of the electric end-effector.

Example

```
--------------------------------------------------------------------------------
        Dim State as Long
        State = caoExt.Execute( “get_Error”)
--------------------------------------------------------------------------------
```

5.2.36.21. Hand object - CaoExtension::Execute("get_BusyState") command

Inform the operation status.

Syntax get_BusyState ()

Argument	:	None	
Return value	:	Operation status	[VT_I4]
		0: An operation command can be received.	
		Other than 0: Running. An operation command came in and was received.	

Return the operation status of the electric end-effector.

Example

```
--------------------------------------------------------------------------------
        Dim State as Long
        State = caoExt.Execute( “get_BusyState”)
--------------------------------------------------------------------------------
```

5.2.36.22. Hand object - CaoExtension::Execute("get_MotorState") command

Inform the motor power status.

Syntax get_MotorState ()

Argument	:	None	
Return value	:	Motor power status	[VT_I4]
		0: Motor power OFF	
		Other than 0: Motor power ON	

Return the motor power status of the electric end-effector.

Example

```
--------------------------------------------------------------------------------
        Dim State as Long
        State = caoExt.Execute( “get_MotorState”)
--------------------------------------------------------------------------------
```

5.3. Variable list

5.3.1. Controller class

Table 5-11 Controller class user variable list

Variable identifier	Data type	Explanation	Attribute	
			get	put
I	VT_I4	I type variable. The variable number is specified after the variable name.	√	√
F	VT_R4	F type variable. The variable number is specified after the variable name.	√	√
D	VT_R8	D type variable. The variable number is specified after the variable name.	√	√
V	VT_ARRAY | VT_R4	V type variable. The variable number is specified after the variable name. The data type has three elements.	√	√
P	VT_ARRAY | VT_R4	P type variable. The variable number is specified after the variable name. The data type has seven elements.	√	√
J	VT_ARRAY | VT_R4	J type variable. The variable number is specified after the variable name. The data type has eight elements.	√	√
T	VT_ARRAY | VT_R4	T type variable. The variable number is specified after the variable name. The data type has ten elements.	√	√
S	VT_BSTR	S type variable. The variable number is specified after the variable name.	√	√
IO	VT_BOOL	IO type variable. The variable number is specified after the variable name.	√	√
IOB	VT_I1	IO type variable. The variable number is specified after the variable name.	√	√
IOW	VT_I2	IO type variable. The variable number is specified after the variable name.	√	√
IOD	VT_I4	IO type variable. The variable number is specified after the variable name.	√	√
IOF	VT_R4	IO type variable. The variable number is specified after the variable name.	√	√

Table 5-12 Controller class system variable list

Variable identifier	Data type	Explanation	Attribute	
			get	put
@VAR_I_LEN	VT_I4	Size of global I type variable	√	√
@VAR_F_LEN	VT_I4	Size of global F type variable	√	√
@VAR_D_LEN	VT_I4	Size of global D type variable	√	√
@VAR_V_LEN	VT_I4	Size of global V type variable	√	√
@VAR_J_LEN	VT_I4	Size of global J type variable	√	√
@VAR_P_LEN	VT_I4	Size of global P type variable	√	√
@VAR_T_LEN	VT_I4	Size of global T type variable	√	√
@VAR_S_LEN	VT_I4	Size of global S type variable	√	√
@VAR_IO_LEN	VT_I4	I/O point number (number of bits)	√	-
@MODE	VT_I4	1: manual, 2: teach check, 3: auto	√	-
@LOCK	VT_BOOL	true: Machine lock ON, false: Machine lock OFF	√	√
@TIME	VT_I4	Actual time elapsed since machine activation (msec)	√	-
@CURRENT_TIME	VT_DATE	Current time	√	-
@BUSY_STATUS	VT_BOOL	true = Program running, false = Program stopped	√	-
@NORMAL_STATUS	VT_BOOL	true = Normal, false = Abnormal (An error has occurred.)	√	-
@ERROR_CODE	VT_I4	Code of an error that has occurred as a decimal number. 0 is returned if no error has occurred. Setting 0 clears the error.	√	√
@ERROR_CODE_HEX	VT_BSTR	Code of an error that has occurred as a hexadecimal character string. "00000000" is returned if no error has occurred.	√	-
@ERROR_DESCRIPTION	VT_BSTR	Description of an error that has occurred	√	-

@EMERGENCY_STOP	VT_BOOL	true = Emergency stop is active. false = Emergency stop is not active.	√	-
@DEADMAN_SW	VT_BOOL	Deadman status	√	-
@AUTO_ENABLE	VT_BOOL	Auto enable status	√	-
@VERSION	VT_BSTR	Controller's version	√	-
@SERIAL_NO	VT_BSTR	Controller's serial number	√	-
@PROTECTIVE_STOP	VT_BOOL	Protective stop	√	-

5.3.2. Robot class

Table 5–**13 Robot class system variable list**

Variable identifier	Data type	Explanation	Attribute	
			get	put
@CURRENT_POSITION	VT_ARRAY | VT_R8	Current robot position. The unit is arbitrary. P type variable.	√	-
@CURRENT_ANGLE	VT_ARRAY | VT_R8	Current robot position (each axis value). The unit is arbitrary. J type variable	√	-
@SERVO_ON	VT_BOOL	true = Servo ON, false = Servo OFF	√	√
@BUSY_STATUS	VT_BOOL	true = Arm moving, false = Arm stopped	√	-
@TYPE_NAME	VT_BSTR	Robot type name	√	-
@TYPE	VT_I4	Robot type data	√	-
@CURRENT_TRANS	VT_ARRAY | VT_R8	Current robot position expressed in T type	√	-
@CURRENT_TOOL	VT_I4	Currently used tool number	√	√
@CURRENT_WORK	VT_I4	Currently used work number	√	√
@SPEED	VT_R4	Internal speed	√	√
@ACCEL	VT_R4	Internal acceleration	√	√

@DECEL	VT_R4	Internal deceleration	√	√
@JSPEED	VT_R4	Internal joint speed	√	√
@JACCEL	VT_R4	Internal joint acceleration	√	√
@JDECEL	VT_R4	Internal joint deceleration	√	√
@EXTSPEED	VT_R4	External speed	√	√
@EXTACCEL	VT_R4	External acceleration	√	√
@EXTDECEL	VT_R4	External deceleration	√	√
@HIGH_CURRENT_POSITION	VT_ARRAY | VT_R8	Current robot position. P type variable. Function specification: When the controller is not in machine-lock mode, the current encoder value is returned.	√	-
@HIGH_CURRENT_ANGLE	VT_ARRAY | VT_R8	Current robot position (each axis value). J type variable. For function specification, refer to @HIGH_CURRENT_POSITION.	√	-
@HIGH_CURRENT_TRANS	VT_ARRAY | VT_R8	Current robot position expressed in T type. For function specification, refer to @HIGH_CURRENT_POSITION.	√	-
@DEST_ANGLE	VT_ARRAY | VT_R8	Previous motion command target position. J type variable. While the robot is stopped, the current position (command value) is returned.	√	-
@DEST_POSITION	VT_ARRAY | VT_R8	Previous motion command target position. P type variable. While the robot is stopped, the current position (command value) is returned.	√	-

@DEST_TRANS	VT_ARRAY | VT_R8	Previous motion command target position. T type variable. While the robot is stopped, the current position (command value) is returned.	√	-
Tool*	VT_ARRAY | VT_R8	Tool definition with the number represented by * X,Y,Z,RX,RY,RZ	√	√
Work*	VT_ARRAY | VT_R8	Work definition with the number represented by * X,Y,Z,RX,RY,RZ,Attribute	√	√
Area*	VT_ARRAY | VT_R8	Area definition with the number represented by * X,Y,Z,RX,RY,RZ,DX,DY,DZ,IO,Position,Error,Time,DRX,DRY,DRZ,Margin,Position1,Margin1,Position2,Margin2,Position3,Margin3,Position4,Margin4,Position5,Margin5,Position6,Margin6,Position7,Margin7,Position8,Margin8,Enable	√	√

5.3.3. Task class

Table 5-14 Task class system variable list

Variable identifier	Data type	Explanation	Attribute	
			get	Put
@STATUS	VT_I4	State of task 0: Task not yet generated (NON_EXISTENT) 1: Hold-stopped 2: Stopped 3: Running 4: Step-stopped	√	-
@PRIORITY	VT_I4	Priority of task. Not supported. Refer to SetThreadPriority() and GetThreadPriority().	-	-
@LINE_NO	VT_I4 | VT_ARRAY	Line number and file ID of currently running main program [0] = Line number [1] = File ID（corresponding to CaoFile::get_ID()）	√	-
@CYCLE_TIME	VT_I4	One cycle execution time of task. The unit is ms.	√	-

@START	VT_I4	Start a task. The meaning of the value is the same as the Mode argument of the CaoTask::Start method. The modes are 1: One cycle execution, 2: Continuous execution, 3: One step forward, and 4: One step backward. Unlike the Start method, the option cannot be specified.	-	√
@STOP	VT_I4	Stop a task. The meaning of the value is the same as the Mode argument of the CaoTask::Stop method. The modes are 0: Default stop, 1: Instant stop, 2: Step stop, 3: Cycle stop, and 4: Initialized stop. Unlike the Stop method, the option cannot be specified. Default stop (0) corresponds to Instant stop (1).	-	√
@ELAPSED_TIME	VT_I4	Time elapsed since task started running. The unit is ms.	√	-

@STATUS_DETAILS	VT_I4	Detailed task status information.
		TASK_NON_EXISTENT = 0, Task non-existent
		TASK_SUSPEND = 1, Hold-stopped
		TASK_READY = 2, Ready
		TASK_RUN = 3, Running
		TASK_STEPSTOP = 4, Step-stopped
		TASK_CNTSTP = 5, Continue-stopped
		TASK_PEND = 6, Pending
		TASK_DELAY = 7, Delay
			√	-

5.3.4. File class

Table 5-15 File class system variable list

Variable identifier	Data type	Explanation	Attribute	
			get	Put
@CRC	VT_I4	CRC32	√	-

Appendix A. CaoController Object Creation

Following is the procedure to create CaoController of ORiN.

(1) Create variables to store objects.

(2) Create a CaoEngine object.

(3) Acquire or create a CaoWorkspace object.

(4) Create a CaoController object.

Following is detailed explanation of the procedure. In this example, the language in use is Visual Basic 6.0, and objects are created with New keyword.

(5) First, declare variables to store objects. The objects required to open a controller are CaoEngine and CaoWorkspace. Furthermore, the AddController method creates a CaoController object. Therefore, create variables for these three objects. Following is an example of declaring variables for each object type as private variables.

```
Private caoEng As CaoEngine             ' Engine object
Private caoWs As CaoWorkspace           ' CaoWorkSpace object
Private caoCtrl As CaoController        ' Controller object
```

(6) Next, create a CaoEngine object using New keyword and assign it to the variable using the Set statement.

```
Set caoEng = New CaoEngine
```

(7) The CaoEngine object, when created, creates default CaoWorkspaces and CaoWorkspace objects, one each. To acquire the default CaoWorkspace object, use CaoWorkspaces.Item(0). Following is an example of acquiring the default CaoWorkspace object.

```
Set caoWs = caoEng.CaoWorkspaces.Item(0)
```

(8) A CaoController object can be created using the AddController method of the CaoWorkspace object.

```
‘ CaoCtrl and CaoWS are variables used to store objects.
Set CaoCtrl = caoWs.AddController("RC8","CaoProv.DENSO.RC8"," ・ ,"Server=192.168.0.1")
```

Example

```
--------------------------------------------------------------------------------------------------------------------------
        Private caoEng As CaoEngine             ' Engine object
        Private caoWs As CaoWorkspace           ' WorkSpace object
        Private caoCtrl As CaoController        ' Controller object

        Set caoEng = New CaoEngine
        Set caoWS = caoEng.CaoWorkspaces.Item(0)
        Set caoCtrl = CaoWs.AddController("RC8","CaoProv.DENSO.RC8"," ・ ,"Server=192.168.0.1")

--------------------------------------------------------------------------------------------------------------------------
```

Appendix B. POSEDATA Type Definition

In the RC8 provider, "POSEDATA " is defined so that the pose data type and vector type data of DENSO robots can be handled by VARIANT type variables.

```
POSEDATA type (VARIANT)
  ├ VT_BSTR[4]                 "[<Pass>]  [<Variable type>]<Index>  [<ExJnt>]"
  │                            or
  │                            "[<Pass>]  [<Variable type>](<Element 1>,<Element 2>,...)
  ├                            [<ExJnt>]"
  └ VT_R4|VT_ARRAY[5]          <Raw value> = (<Element 1:VT_R4>,<Element 2:VT_R4>,...)[6]
    VT_VARIANT|VT_ARRAY        (<Value>[,<Variable type>[,<Path>[, <ExJnt>]]])
          ├<Value>             <Index:VT_R4>
          │                    or
          │                    <Raw value:VT_R4|VT_ARRAY>
          ├<Variable type>     P, T, J, V type with VT_I4 or VT_BSTR specified (default = P)
          ├<Pass>              @P, @E, @0, @<Value> with VT_I4 or VT_BSTR specified
          │                    (default = @0)
          └<ExJnt>             <Extended-joints option:VT_VARIANT|VT_ARRAY>
                               (default = None)
```

<Pass> : @P, @E, @0, @<Value>

Mark	@P	@E	@0	@<Value: n>	None
VT_BSTR	"@P"	"@E"	"@0"	"@n"	""
VT_I4	-1	-2	0	n	0

<Variable type> : P type, T type, J type, V type

Mark	P	T	J	V	None
VT_BSTR	"P"	"T"	"J"	"V"	""
VT_I4	0	1	2	3	-1

<Index> : <Value:VT_R4>

<Element n> : <Value:VT_R4>

[4] In case of VT_BSTR, more than one POSEDATA type separated by commas can be specified.

[5] Because <Variable type> and <Pass> cannot be specified, variable type is treated as P type and pass type is treated as @0 by default.

[6] Because <Variable type> and <Pass> cannot be specified, variable type is treated as P type and pass type is treated as @0 by default.

<Extended-joints option> : (<EX or EXA>, (<Joint 1: VT_I4>,<Value 1: VT_R8>)[,(<Joint 2>,<Value 2>)...])

Mark	EX	EXA	None
VT_BSTR	"EX"	"EXA"	""
VT_I4	1	2	0

The following formats of PacScript language can be indicated by POSEDATA type.

[<Pass start displacement>] <Pose:P,T,J type> [<ExJnt>]	(C0-format)
[<Pass start displacement>] <Vector:V type>	(C1-format)
[<Pass start displacement>] <Value> [<ExJnt>]	(C2-format)
[<Pass start displacement>] (<Element 1>,<Element 2>,...) [<ExJnt>]	(C3-format)

Appendix B.1. Examples

[<Pass start displacement>] <Pose> [<ExJnt>] (C0-format)

ex1. T200

String	"T200"
VARIANT type array (Variable type specified by string)	Array(200,"T")[7]
VARIANT type array (Variable type specified by value)	Array(200,1)

ex2. @P J100

String	"@P J100"
VARIANT type array (Variable type and pass type specified by string)	Array(100,"J","@P")
VARIANT type array (Variable type and pass type specified by value)	Array(100,2,-1)

[7] Array(…) is a function to return an array composed of the argument to the function. (Array function of VB6)

ex3. @E P(10.0, 10.5, 34.6, 0.0, 90.0, 0.0, -1.0)

String	"@E P(10.0, 10.5, 34.6, 0.0, 90.0, 0.0, -1.0)"
VARIANT type array (Raw value, with variable type and pass type specified by string)	Dim p(6) as Single Dim vP as Variant p(0) = 10.0 : p(1) = 10.5 : p(2) = 34.6 : p(3) = 0.0 p(4) = 90.0 : p(5) = 0.0 : p(6) = -1.0 vP = p() Array(vP,"P","@E")
VARIANT type array (Raw value, (Variable type and pass type specified by value)	Dim p(6) as Single Dim vP as Variant p(0) = 10.0 : p(1) = 10.5 : p(2) = 34.6 : p(3) = 0.0 p(4) = 90.0 : p(5) = 0.0 : p(6) = -1.0 vP = p() Array(vP, 0, -2)

ex4. @P J100 EXA((7, 30.5), (8, 90.5))

String	"@P J100 EXA((7, 30.5), (8, 90.5))"
VARIANT type array (Variable type, pass type, and extended-joints specified by string)	Array(100,"J","@P", Array("EXA",Array(7,30.5), Array(8,90.5)))
VARIANT type array (Variable type, pass type, and extended-joints specified by value)	Array(100,2,-1, Array(2, Array(7,30.5), Array(8,90.5)))

[<Pass start displacement>] <Vector:V type> (C1-format)

ex1. @P V20

String	"@P V20"
VARIANT type array (Variable type and pass type specified by string)	Array(20,"V","@P")
VARIANT type array (Variable type and pass type specified by value)	Array(20,3,-1)

ex2. @E V(0.0, 125.5, 50.0)

String	"@E V(0.0, 125.5, 50.0)"
VARIANT type array (Raw value, with variable type and pass type specified by string)	Dim v(2) as Single Dim vV as Variant v(0) = 0.0 : v(1) = 125.5 : v(2) = 50.0 vV = v() Array(vV,"V","@E")

VARIANT type array (Raw value, (Variable type and pass type specified by value)	Dim v(2) as Single Dim vV as Variant v(0) = 0.0 : v(1) = 125.5 : v(2) = 50.0 vV = v() ‘ = VT_R4 | VT_ARRAY Array(vV, 3, -2)

[<Pass start displacement>] <Value> [<ExJnt>] (C2-format)

ex1. @P 1

String	"@P 1"
VARIANT type array (Variable type and pass type specified by string)	Array(1," ・ ,"@P")
VARIANT type array (Variable type and pass type specified by value)	Array(1,-1,-1)

ex2. @P 1.56

String	"@P 1.56"
VARIANT type array (Variable type and pass type specified by string)	Array(1.56," ","@P")
VARIANT type array (Variable type and pass type specified by value)	Array(1.56,-1,-1)

[<Pass start displacement>] (<Element 1>,<Element 2>,..) [<ExJnt>] (C3-format)

ex1. @P (1, 30.0)

String	"@P (1, 30.0)"
VARIANT type array (Variable type and pass type specified by string)	Dim v(1) as Single v(0) = 1 : v(1) = 30.0 Dim vV as Variant vV = v() Array(vV," ","@P")
VARIANT type array (Variable type and pass type specified by value)	Dim v(1) as Single v(0) = 1 : v(1) = 30.0 Dim vV as Variant vV = v() Array(vV, -1, -1)

Other examples

ex1. V1,V2,V3

(Rotation plane for CaoRobot::Rotate())

String	"V1,V2,V3"
String array	Array("V1","V2","V3")
VARIANT type array (Variable type specified by string)	Array(Array(1,"V"),Array(2,"V"),Array(3,"V"))
VARIANT type array (Variable type specified by value)	Array(Array(1,3),Array(2,3),Array(3,3))

ex2. APPROACH P,P70, 60, NEXT

(Approach command for CaoRobot::Execute(), without pass specification)

2nd argument: string 3rd argument: string	.Execute "APPROACH", Array(1, "P70", "60", "NEXT")
2nd argument: VARIANT array 3rd argument: VARIANT array	.Execute "APPROACH", Array(1, Array(70, "P"), _ Array(60, "", ""), "NEXT")

ex3. APPROACH L,J(60.5,30.3,400,90),@100 70, NEXT

(Approach command for CaoRobot::Execute(), without pass specification)

2nd argument: string 3rd argument: string	.Execute "APPROACH", Array(2, "J(60.5,30.3,400,90)", "@100 70", "NEXT")
2nd argument: VARIANT array (Raw value, variable type specified by string) 3rd argument: VARIANT array (Variable type and pass type specified by string)	Dim j(3) as Single Dim vJ as Variant j(0) = 60.5 : j(1) = 30.3 : j(2) = 400 : j(3) = 90 vJ = j() ‘ = VT_R4 | VT_ARRAY .Execute "APPROACH", Array(2, Array(vJ, "J"), _ Array(70,"","@100"), "NEXT")
2nd argument: VARIANT array (Raw value, variable type specified by string) 3rd argument: VARIANT array (Variable type and pass type specified by value)	Dim j(3) as Single Dim vJ as Variant j(0) = 60.5 : j(1) = 30.3 : j(2) = 400 : j(3) = 90 vJ = j() ‘ = VT_R4 | VT_ARRAY .Execute "APPROACH", Array(2, Array(vJ, "J"), _ Array(70, -1, 100), "NEXT")

[Notes]

When a raw value is specified directly by POSEDATA type by VT_R4|VT_ARRAY form, it becomes P type and @0 by default. Therefore, data other than P type cannot be specified directly by the VT_R4|VT_ARRAY form. In this case, specify the variable type of the data explicitly by the VT_VARIANT|VT_ARRAY form or VT_BSTR form.

Note that the following codes do not make sense.

‘[PAC] MOVE P, J100

```
Dim vJ as Variant
vJ=CaoCtrl.Variables("J100").Value   ‘VT_R4|VT_ARRAY
Robot.Move 1, vJ                    ' Wrong!! = MOVE P, P(<j1>,<j2>,<j3>,⋯)
```

The correct code is as follows.

```
Robot.Move 1, Array(vJ,"J")      ‘ Variant specification = MOVE P, J(<j1>,<j2>,<j3>,...)
```