RC8 Provider

for DENSO Robot RC8

Version 1.1.2

User's Guide

November 06, 2014

[Remarks]

[Revision History]

Version	Date	Content
1.1.0	2012-09-10	First edition.
1.1.1	2012-10-31	Hand object interface has been added.
	2013-02-06	Motion option in CaoRobot::{Move, Rotate, Draw, Approach, Depart, DriveEx, DriveAEx, RotateH} method has been modified.
	2013-06-27	Added a postscript about @IfNotMember option.
	2013-07-09	Added description about log interface.
	2013-07-24	Added Synchronizing flag to KillAll, SuspendAll, and StepStopAll. Added CurJntEx,HighCurJntEx,DestJntEx,CurPosEx,HighCurPosEx, DestPosEx,CurTrnEx,HighCurTrnEx,DestTrnEx
	2013-08-01	Added following arguments to Force Param ・ Control rate ・ Maximum translation speed ・ Maximum rotation speed Added following items ・ KillAllTsr, RunAllTsr ・ ForceValue ・ ForceWaitCondition ・ ForceChangeTable
	2013-12-06	Added DPS command
1.1.2	2014-02-24	Added Appendix C
	2014-03-05	Added SysInfo,RobInfo.
	2014-03-06	Added "Non-Stop Motion Calculator"
	2014-04-02	Added "GetPublicValue, SetPublicValue"
	2014-09-08	Added " SyncTimeStart,SyncTimeEnd,SyncMoveStart,SyncMoveEnd,SetBaseDef, GetBaseDef,SetHandIO、 GetHandIO,StartServoLog,ClearServoLog,StopServoLog"
	2014-11-06	Added " GetCtrlLogMaxTime,SetCtrlLogMaxTime,GetCtrlLogInterval,SetCtrlLog Interval"

[Hardware]

Model	Version	Notes
RC8	1.2.4	

[Command]

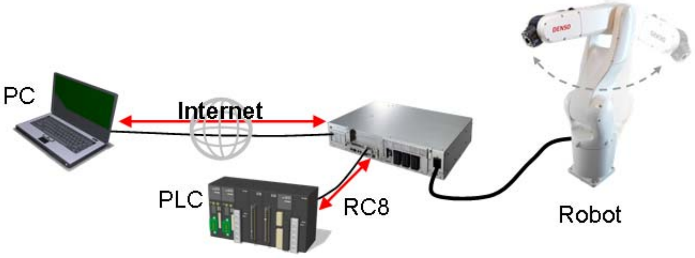
Model	Version	Notes
RC8	1.3.6	Added Spline-related commands

Contents

1. Introduction

This document describes external specifications of the Cao provider for RC8 controller of the Denso robot (including VRC[1]) 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.

[1] VRC (Virtual Robot Controller) is manufactured by DENSO WAVE INCORPORATED. To use the VRC, you need to prepare VRC license separately.

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

To use the Cao Engine module, you need to register a regetimate license key for each PC. 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.

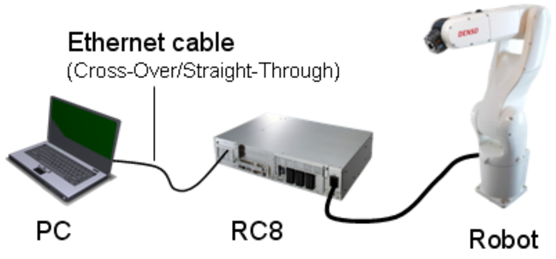
(3) A robot emergency stop switch should be set up separately from the power switch.

2.2.2. Preparation of hardware

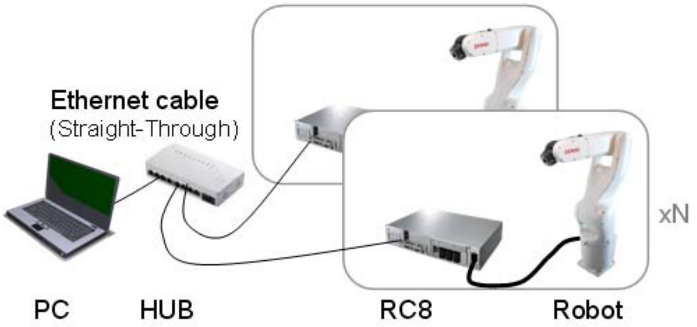
The following shows the basic hardware configurations that can be used for the robot controller clients. When designing equipment, consider the system configuration for the software required by the customer and prepare hardware accordingly.

(1) PC-based robot system

- Configuration with one RC8 unit



- Configuration with more than one RC8 unit



(2) RC8-based robot system

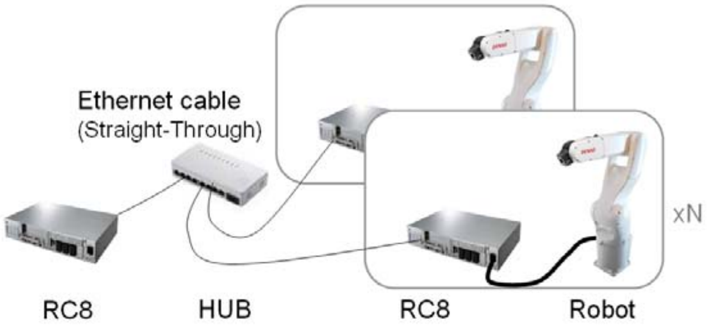


Table 2-2 Configurations of robot systems

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.

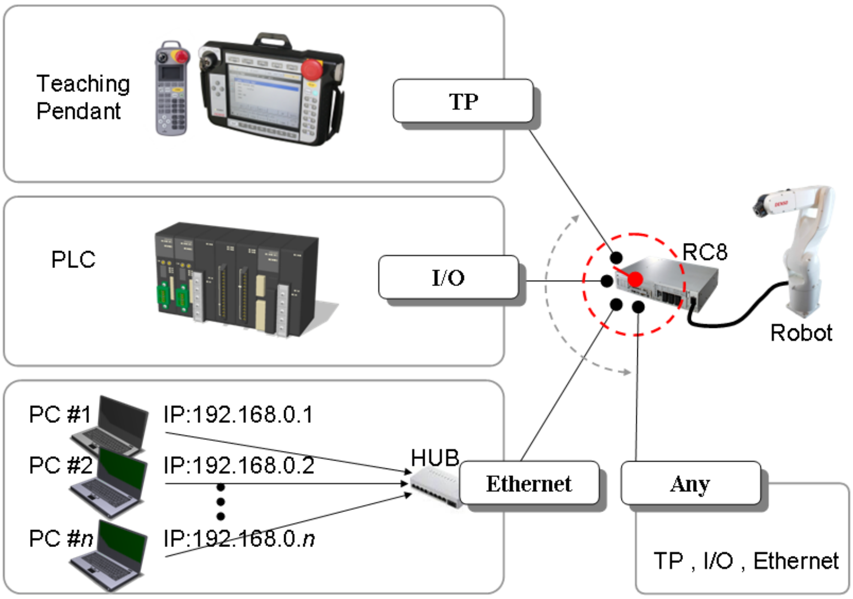


Figure 2-1 Setup of devices with activation authority

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.

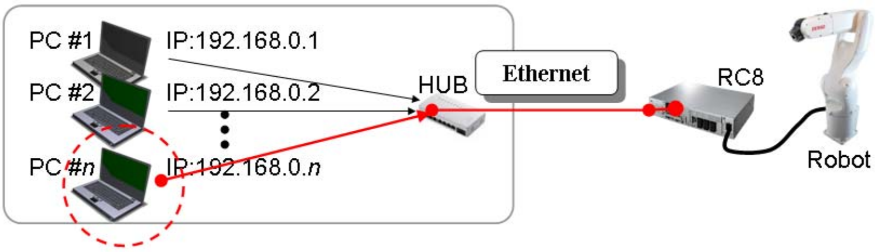


Figure 2-2 Setup of clients with activation authority

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

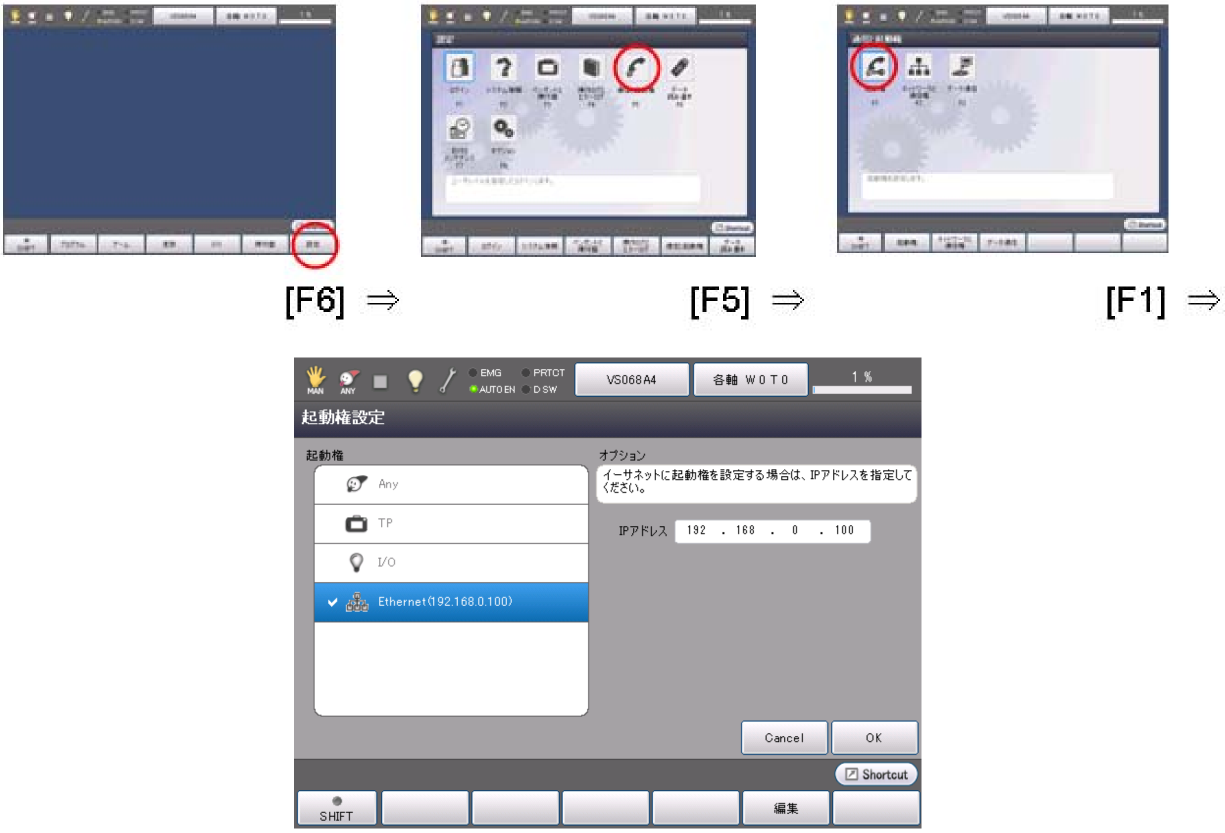
2.2.3.1. Setup using a teach pendant

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

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

(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)

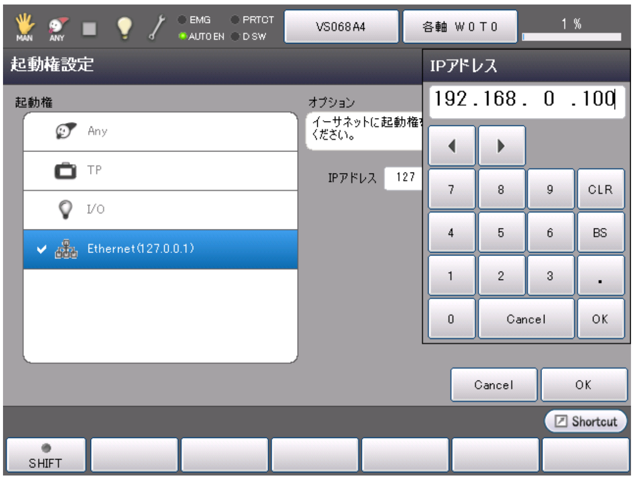
オプション

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
ください。

IPアドレス 127

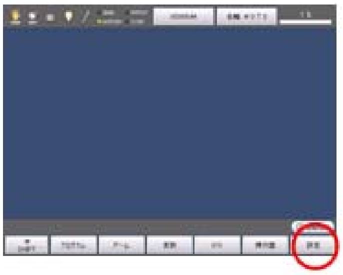
IPアドレス

192.168. 0 .100

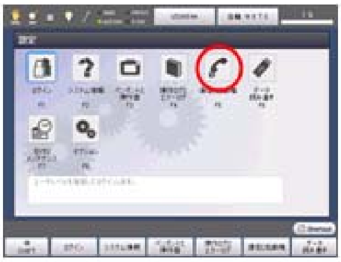
Figure 2-4 Setting of IP addresses of clients

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

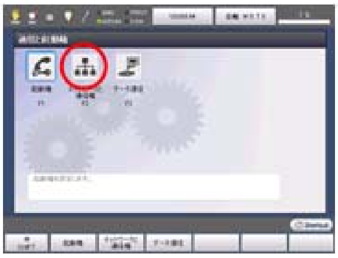
To use Ethernet, select the teach pendant's [F6 Setup] menu -> [F5 Communication and Token] -> [F2 Network and Permission] and set the read/write permissions to Ethernet.



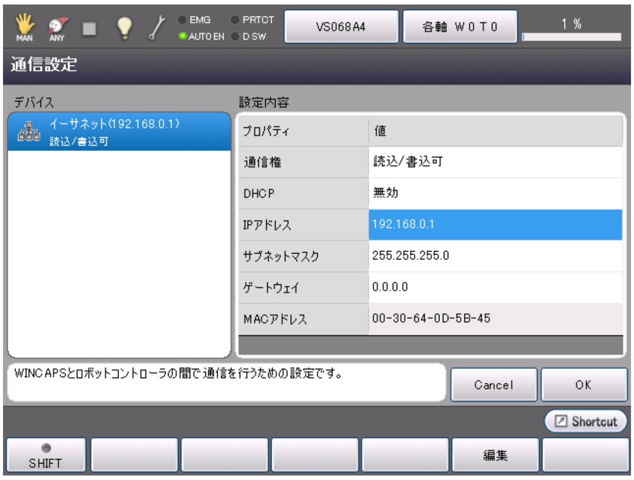
[F6] ⇒



[F5] ⇒



[F2] ⇒



通信設定

デバイス

イーサネット(192.168.0.1)
読込/書込可

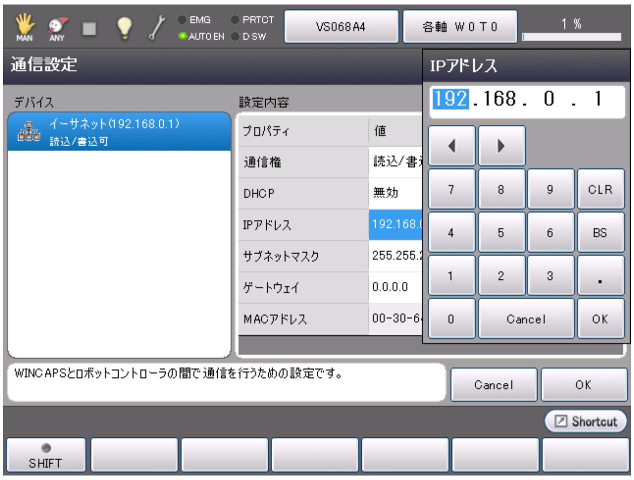
設定内容

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.



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
DHCP	無効
IPアドレス	192.168.(
サブネットマスク	255.255.2
ゲートウェイ	0.0.0.0
MACアドレス	00-30-6

IPアドレス

192.168. 0 . 1

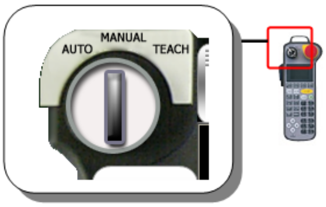
WINCAPSとロボットコントローラの間で通信を行うための設定です。

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.

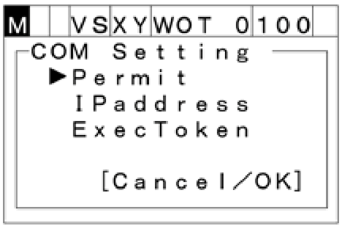


Figure 2-7 List of communications settings

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.

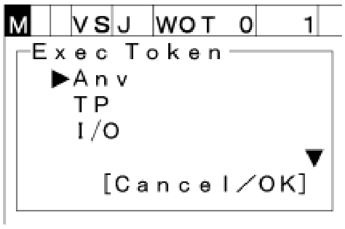
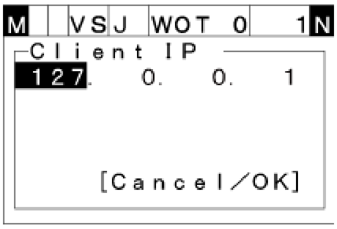


Figure 2-8 List of activation authority settings

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

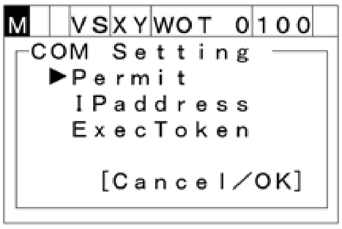
Figure 2-9 Setting of IP addresses of clients

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.



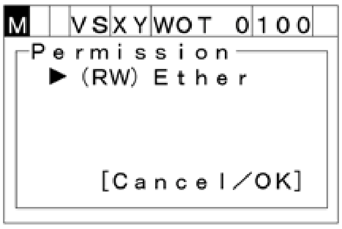
```
M  VSXYWOT 0100
COM Setting
 ▶Permit
  IPaddress
  ExecToken

    [Cancel／OK]
```

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.

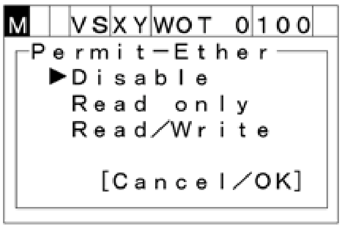


```
M  VSXYWOT 0100
Permission
 ▶ (RW) Ether



    [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]
```

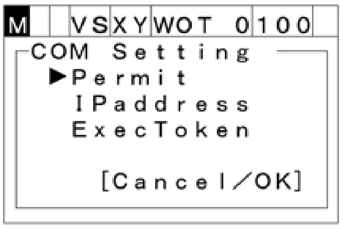
Figure 2-12 List of communication options

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.

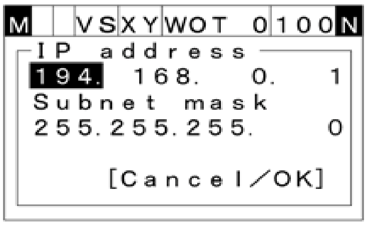


```
M  VSXYWOT 0100
COM Setting
 ►Permit
  IPaddress
  ExecToken

    [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.

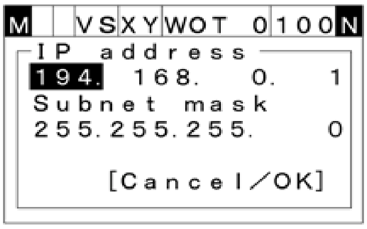
Press [Cancel] to exit the communications setting.



```
M  VSXYWOT 0100N
IP address
194. 168.   0.   1
Subnet mask
255.255.255.    0

    [Cancel/OK]
```

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.



```
M  VSXYWOT 0100N
IP address
194. 168.   0.   1
Subnet mask
255.255.255.    0

    [Cancel/OK]
```

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

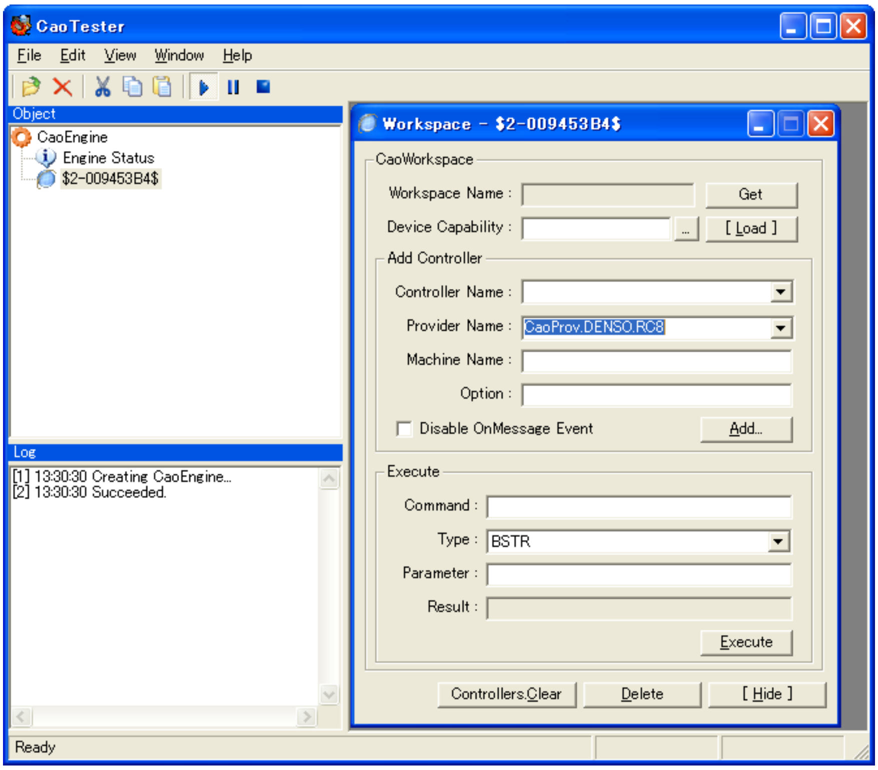
Before running a developed client application, check that the RC8 robot controller to be controlled has been set up correctly using CaoTester, an ORiN2 SDK standard tool.

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.



CaoEngine
Engine Status
$2-009453B4$

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

Provider Name : CaoProv.DENSO.RC8

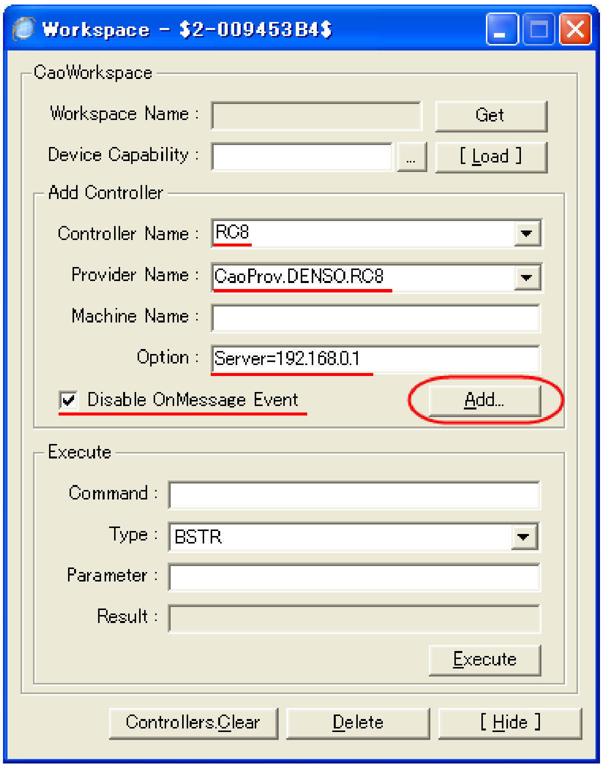
Type : BSTR

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

Machine Name :

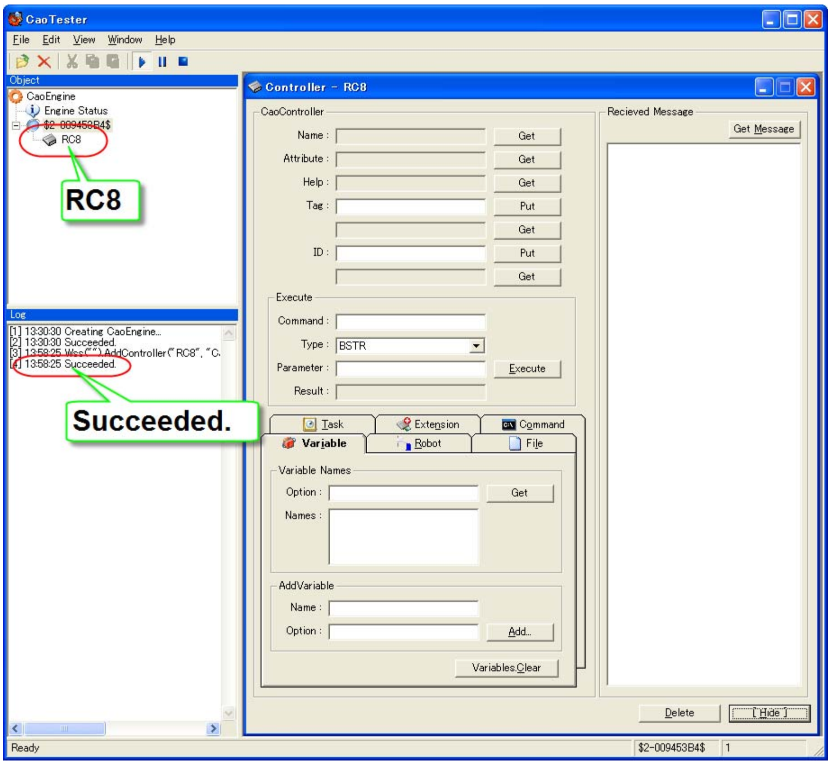
Option : Server=192.168.0.1

☑ Disable OnMessage Event

Type : BSTR

Figure 2-17 [Workspace] window

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



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

Type : BSTR

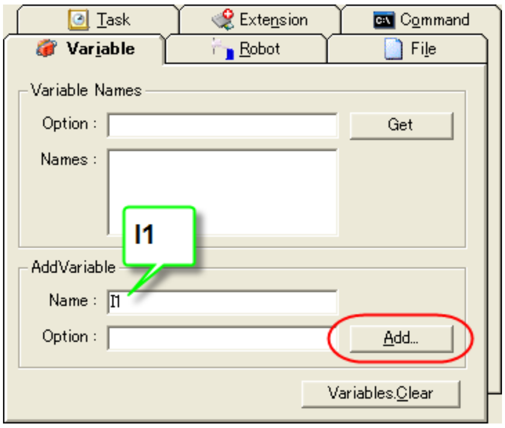
$2-009453B4$ 1

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.

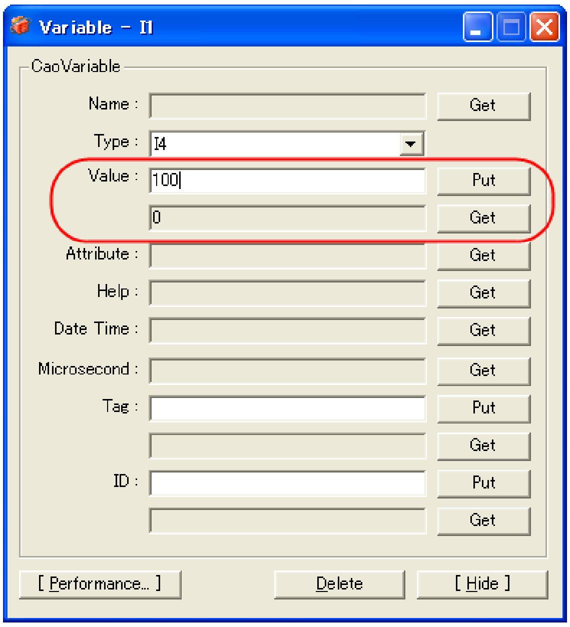


Name : I1

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

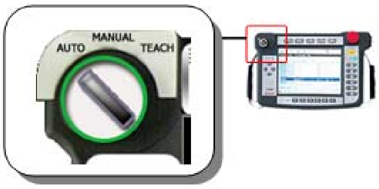
0

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 to 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 over the target robot controller may not be correctly set on the client PC. In that case, check the activation authority setting again.

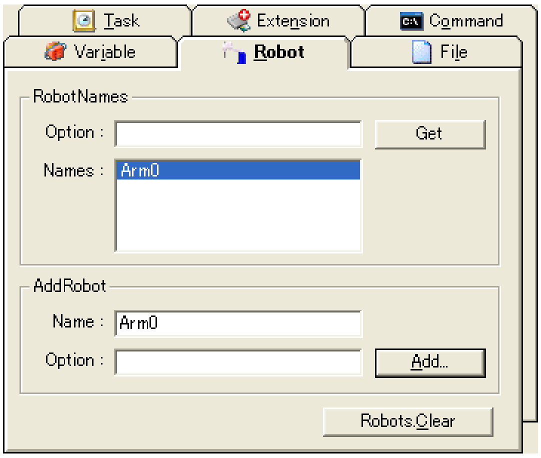
(1) **<u>Set</u>** the robot controller **<u>to the Auto mode</u>**.



(2) 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

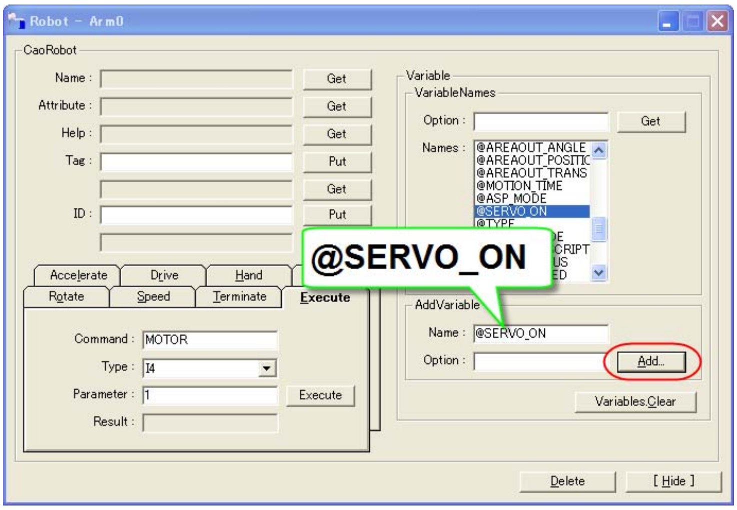
AddRobot
Name : Arm0
Option :

Figure 2-21 [Robot] tab settings

(3) 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.



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

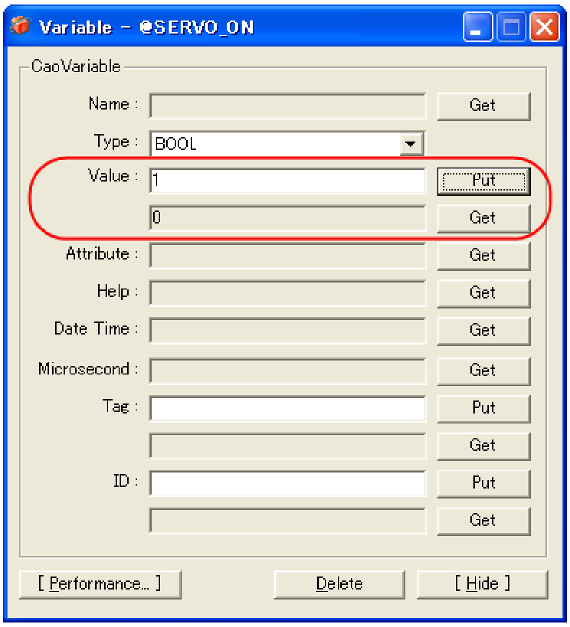
AddVariable
Name : @SERVO_ON
Option :

Command : MOTOR
Type : I4
Parameter : 1
Result :

Figure 2-22 [Robot] window settings

(4) 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.



Name :

Type : BOOL

Value : 1

0

Attribute :

Help :

Date Time :

Microsecond :

Tag :

ID :

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.
Expansion Board	CaoExtension	Can set and acquire parameters of the electric gripper, aquire status, and control motion commands.

3.1.2. System configuration of RC8 provider

The RC8 provider is a core module independent of the hardware of the robot controller. The RC8 provider establishes the compatibility between the simulation and the RC8 robot controller; this will solve the inconsistency of the robot motion between programs created by the simulation and programs created by the RC8 robot controller.

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

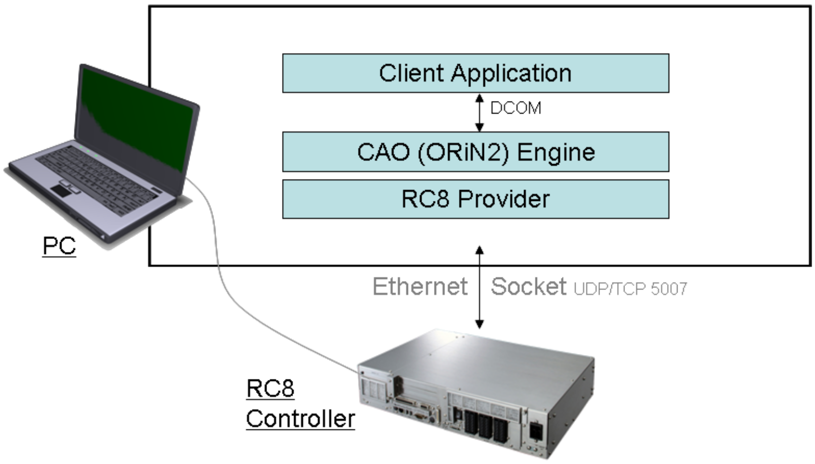


Figure 3-1 System configuration of PC and RC8

This connection route is determined depending on the connection parameter that is specified when Client Application connects to the RC8 provider (AddController). If you assign IP address ("Server=...") of the robot controller to the RC8 provider, it will connect to the robot controller; if you assign a project file ("wpj=...") of WINCAPS3 to the RC8 provider, it will connect to the VRC.

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.

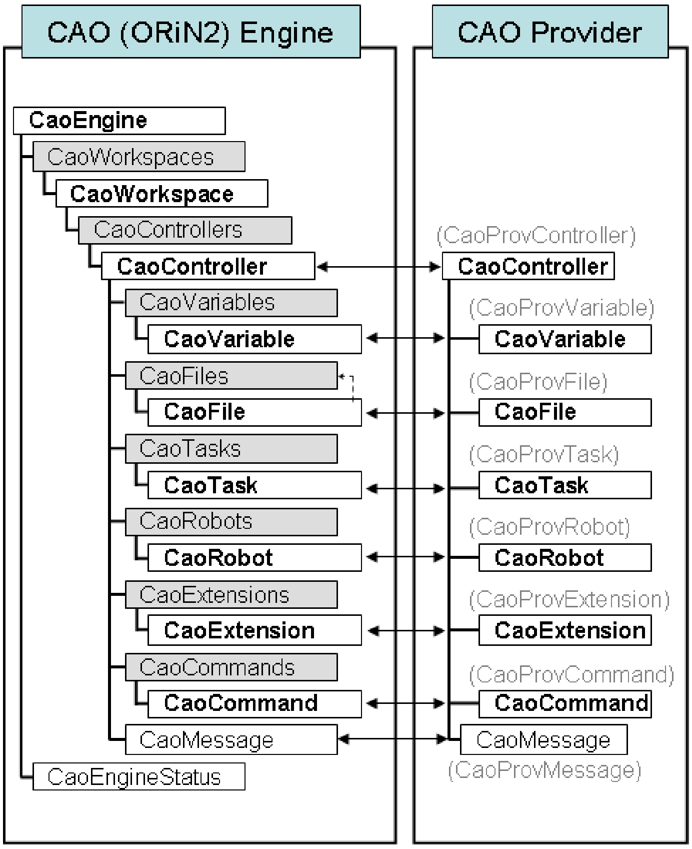


Figure 3-2 Configuration of Cao engine and provider

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.

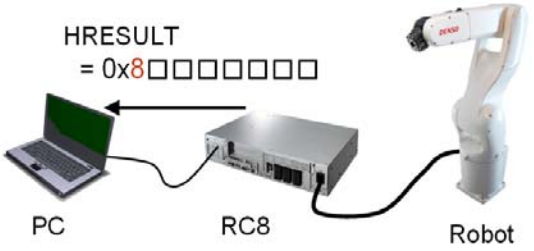


Figure 3-3 Error in call from PC

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.

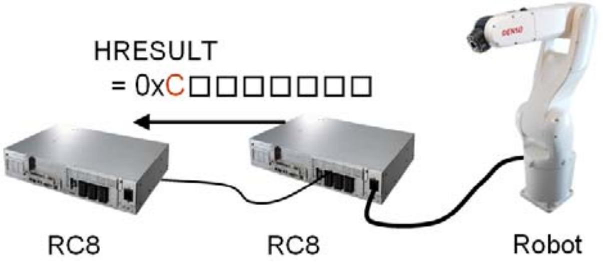


Figure 3-4 Error in call from RC8

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".

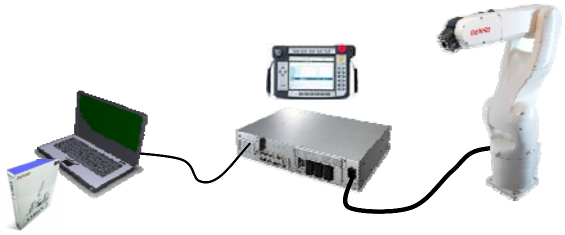


Figure 4-1 Robot connection

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

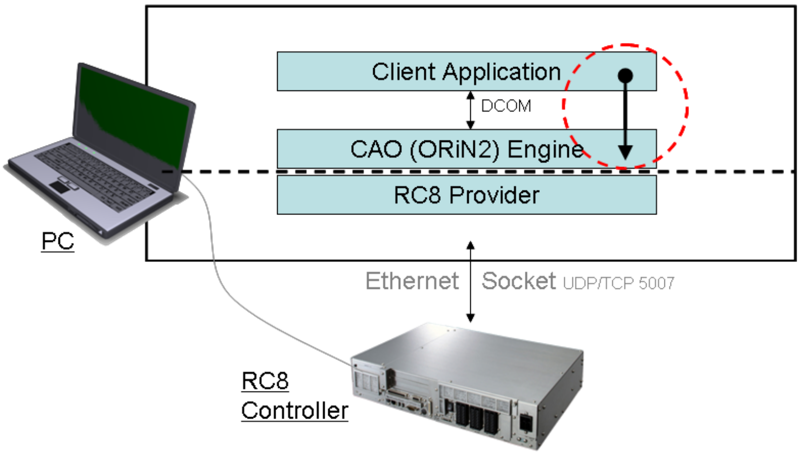


Figure 4-2 Outline of programing

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.

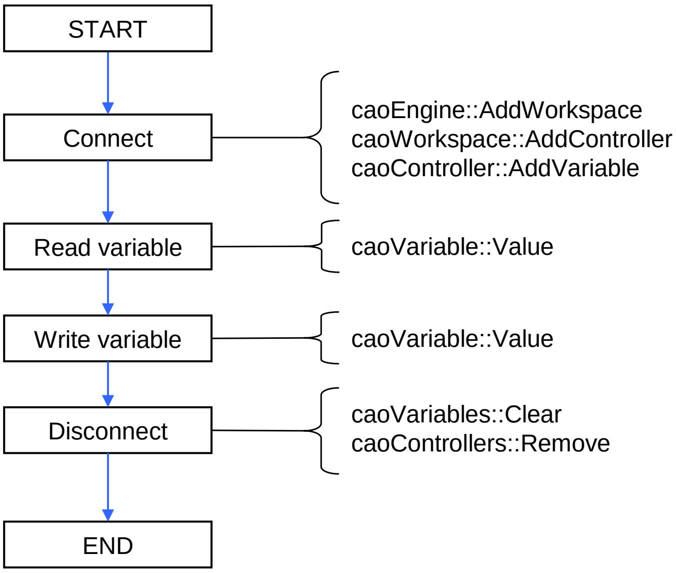


Figure 4-3 Variable access

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 for a variable that you intend to connect. 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)
            ' Delete variable object
            g_ctrl.Variables.Clear
            Set g_val = Nothing

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

            ' Delete 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 more details, refer to "2.2.3 Setup of system parameters".

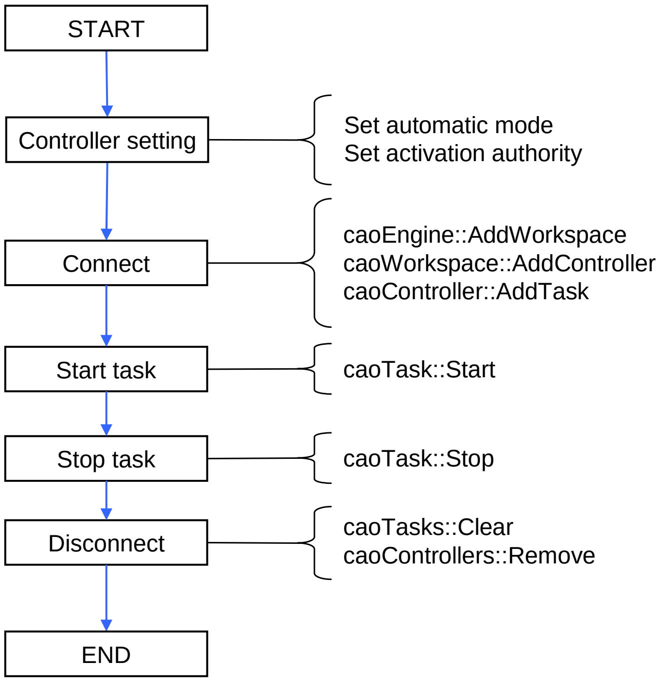


Figure 4-4 Task control flow

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 or 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.

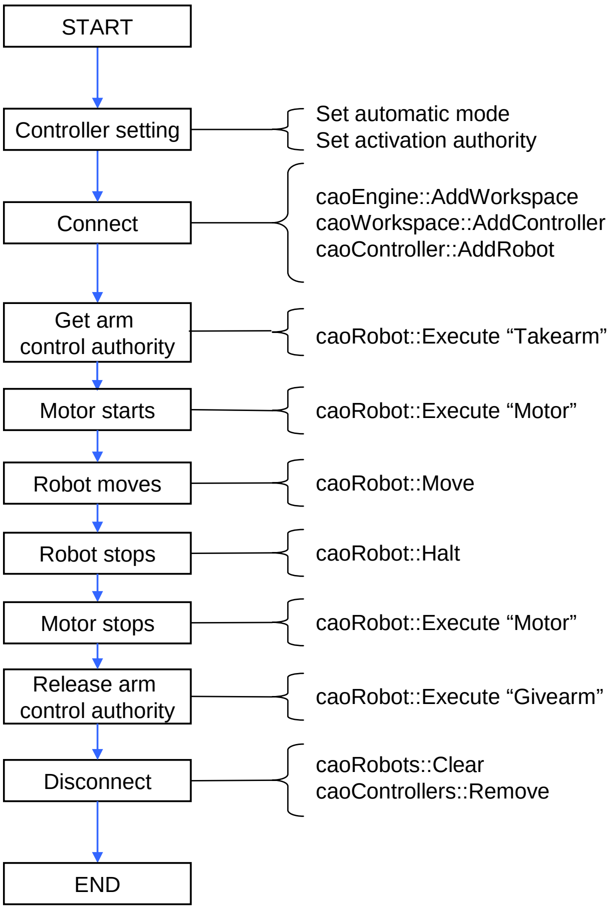


Figure 4-5 Robot control flow

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 over the robot. Furthermore, release the arm control authority over 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 if robot has 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.46
CaoController			
	AddFile	Connect to a file or folder (PAC, system file).	P.48
	AddRobot	Connect the robot.	P.49
	AddTask	Connect the task (PAC).	P.50
	AddVariable	Connect the user/system variable.	P.51
	get_Name	Get the controller name.	P.53
	get_FileNames	Get a list of file names.	P.53
	get_TaskNames	Get a list of tasks (PAC).	P.54
	get_VariableNames	Get a list of user/system variables.	P.54
	Execute	Execute a command of the controller class.	P.54
CaoFile			
	AddFile	Connect a PAC file.	P.66
	AddVariable	Connect a system variable of files.	P.69
	get_VariableNames	Get a list of system variable names.	P.69
	get_FileNames	Get a list of files.	P.69
	get_Size	Get the size of a file.	P.69
	get_Value	Get the value of a file.	P.70
	put_Value	Rewrite the value of a file.	P.70
CaoRobot			
	Accelerate	Set the internal acceleration and deceleration ratio of the robot.	P.70
	AddVariable	Connect a system variable.	P.71
	get_VariableNames	Get a list of system variable names.	P.71
	Halt	Stop the robot in asynchronous motion.	P.71

	Change	Change the tool/user coordinate system of the robot.	P.72
	Drive	This method is not supported directly in this provider.	P.72
	Move	Robot moves.	P.73
	Rotate	Rotate around the specified axis.	P.73
	Speed	Set the internal movement speed of the robot.	P.79
	Execute	Execute a command of the robot.	P.80
CaoTask			
	AddVariable	Connect a system variable of the robot.	P.139
	get_VariableNames	Get a list of system variable names.	P.139
	Start	Start the PAC program.	P.139
	Stop	Stop the PAC program.	P.139
	Execute	Execute a command of the task class.	P.139
CaoVariable			
	get_Value	Get a value.	P.142
	put_Value	Set a value.	P.142

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 3000 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.
WPJ={<Project file>}	To perform simulation, specify a full path of a "*.wpj" file of WINCAPS3 for a project file. To connect to the latest virtual controller that have been activated, specify "*(asterisk)" for a project file. If the project file name is identical with the name of the virtual controller that has been activated, connect to that controller simultaneously. It will be "*(asterisk)" if it is omitted. Example: "WPJ= {C:\ Program Files\WINCAPS3\Test\Test.wpj}" "WPJ= {*}"

If "@IfNotMember" is specified to an option string, an existing object corresponding to the controller name will be returned. The object of the specified name is newly made when there is no object of the same name.

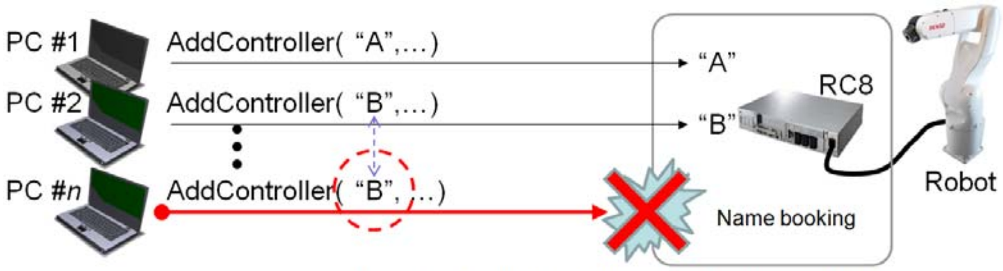
Specifying an empty string ("") for the name, the @IfNotMember option will be ignored.

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")
-----------------------------------------------------------------------------------------------------------------------------
```

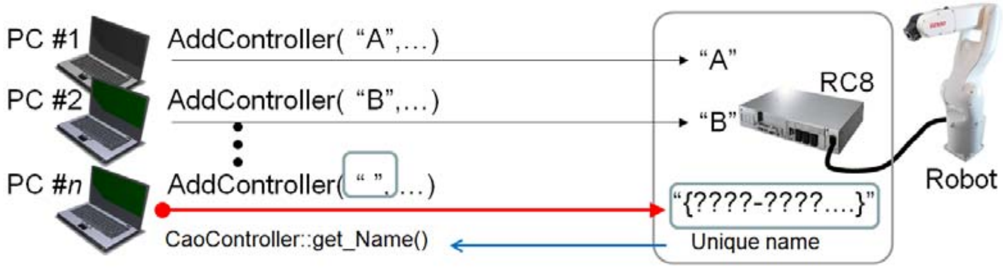
5.2.1.1. When you establish multiple connections with RC8 controller

When the RC8 controller connects with devices through RC8 provider, clients (PCs) and a server (RC8 controller) communicates with b-CAP protocol in each connection (by AddController unit). Note that the controller name, which is first argument of CaoWorkspace::AddController, needs to be the unique value. A duplicate controller name causes connection failure.



Error 0x80000205: Already registered in the collection.

These controller names should be handled so that these do not duplicate among clients. If it is impossible, you can request the system side to issue a unique name by means of entering empty string ("") to a controller name. This automatically assigned name can be obtained by CaoController::Name property.



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

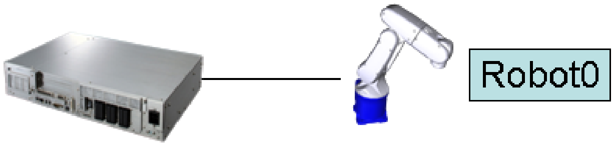
A CaoRobot object is retrieved by calling the AddRobot method. The argument of the AddRobot method of the CaoController class specifies the robot name (BSTR type). "Robot name" specified here is an arbitrary string. For example, specify AddRobot ("Robot1").

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

bstrName	:	[in]	Robot name
bstrOption	:	[in]	Option character string ID=<Robot number> By default, ID=0, i.e., the master arm (Arm0) is specified.

Example

```
Dim CaoRob as CaoRobot
Dim SlaveRob as CaoRobot
Set Rob = caoCtrl.AddRobot("Robot0","ID=0“ )   ‘Specify Robot0
Set SlaveRob = caoCtrl.AddRobot("Robot1","ID=1“ ) ‘Use this command when slave robot is used
                                                  ‘in the coordinate function
```



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 you specify 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<n>[2]	Object for electric gripper <n>[3]

Example:

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

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

[3] Before using an electric gripper object, it is necessary to install an optional electric gripper 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."

5.2.7. 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
--------------------------------------------------------------------------------
```

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. 1. The command name is passed as a BSTR string to the first argument.
2. 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 Lists of commands of CaoController::Execute method

Category	Command name	Function	
Error handling			
	ClearError	Reset an error.	P.55
	GetErrorDescription	Get an error string.	P.55
Task control			
	KillAll	Terminate all tasks.	P.56

	SuspendAll	Suspend all tasks.	P.56
	StepStopAll	Step-stop all tasks	P.58
	ContinueStartAll	Continue-start all tasks.	P.58
Log			
	GetErrorLogCount	Get count of error log.	P.59
	GetErrorLog	Get data of the error.	P.59
	GetOprLogCount	Get count of operation log.	P.60
	GetOprLog	Get data of the operation.	P.61
Variable access			
	GetPublicValue	Read a Public variable value.	P.62
	SetPublicValue	Write a Public variable value.	P.64
Misc.			
	SysState	Get controller status.	P.66

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 : [in] Synchronizing flag(VT_I4)
0: Exit processing without waiting the stop of robots or program.
1: Exit processing after robots or program has stopped.
If the argument is omitted, 0 is assumed to be specified.

Return value : None

All the tasks are sent into an initialized stop status. However, supervisory tasks do not stop.

Example

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

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

Perform initialized stop of all the supervisory tasks under execution

Syntax KillAllTsr ([<lSync>])

Argument : [in] Synchronizing flag(VT_I4)
0: Exit processing without waiting the stop of supervisory tasks.
1: Exit processing after supervisory tasks has stopped.
If the argument is omitted, 0 is assumed to be specified.

Return value : None

Example

```
-----------------------------------------------------------------------------------------------------------------------------
        caoCtrl.Execute"KillAllTsr"
----------------------------------------------------------------------------------------------------------------------------
```

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

Run supervisory tasks specified by Mode

Syntax RunAllTsr ([<Mode>])

Argument : [in] Mode (VT_I4)
0: Run supervisory tasks in the root
1: Run all the supervisory tasks
If the argument is omitted, 0 is assumed to be specified.

Return value : None

Example

```
--------------------------------------------------------------------------------
        caoCtrl.Execute"RunAllTsr"
--------------------------------------------------------------------------------
```

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

Suspend all the running tasks.

Syntax SuspendAll ( )

Argument : [in] Synchronizing flag(VT_I4)
0: Exit processing without waiting the stop of robots or program.
1: Exit processing after robots or program has stopped.
If the argument is omitted, 0 is assumed to be specified.

Return value : None

All the tasks are sent into a suspended status. However, supervisory tasks do not stop.

Example

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

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

Perform step stop of all the running tasks.

Syntax StepStopAll ( )

Argument : [in] Synchronizing flag(VT_I4)
0: Exit processing without waiting the stop of robots or program.
1: Exit processing after robots or program has stopped.
If the argument is omitted, 0 is assumed to be specified.

Return value : None

Supervisory tasks do not stop.

If Synchronizing flag is 1, StepStopAll command exits the processing when the program stops before the step-stop has achieved.

Example

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

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

Start all the running tasks which are being suspended.

Syntax ContinueStartAll ( )

Argument : None

Return value : None

Example

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

5.2.11.9. CaoController::Execute("GetErrorLogCount") command

Get count of error log.

Syntax GetErrorLogCount ()

Argument	:	None
Return value	:	Count (VT_I4)

Example

```
--------------------------------------------------------------------------------
        Dim lErrCnt As Long
        lErrCnt = caoCtrl.Execute("GetErrorLogCount" )
--------------------------------------------------------------------------------
```

5.2.11.10. CaoController::Execute("GetErrorLog") command

Get data of the error.

Syntax GetErrorLog (<lIndex>)

<lIndex> : Index number (VT_I4) 0 to <Count>-1

Return value : Details (VT_VARIANT[VT_VARIANT|VT_ARRAY:15 elements])

[0]:Error Code (VT_I4)
[1]:Year (VT_I4)
[2]:Month (VT_I4)
[3]:Day of week (VT_I4)
[4]:Day (VT_I4)
[5]:Hour (VT_I4)
[6]:Minute (VT_I4)
[7]:Second (VT_I4)
[8]:Milliseconds (VT_I4)
[9]: Tick count of the controller [ms] (VT_I4)
[10]:Program Name (VT_BSTR)
[11]:Line Number (VT_I4)
[12]:Error Description (VT_BSTR)
[13]:Original Error Code (VT_I4)
[14]:Client ID (VT_I4)
-1:Unknown
0:System

1:TP

2:IO

3:PC

4:PAC

5:COM2

6:COM3

7:COM4

Example

```
--------------------------------------------------------------------------------------------------------------------------------
        Dim lErrCnt As Long
        lErrCnt = caoCtrl.Execute("GetErrorLogCount")
        Dim i As Long
        For i=0 To lErrCnt-1
           vDat = caoCtrl.Execute("GetErrorLog", i)
           ‘ Hex(vDat(0)) Error Code
           ‘ vDat(12) Error Description
           ‘ :
        Next
--------------------------------------------------------------------------------------------------------------------------------
```

5.2.11.11. CaoController::Execute("GetOprLogCount") command

Get count of operation log.

Syntax GetOprLogCount ()

Argument	:	None
Return value	:	Count (VT_I4)

Example

```
--------------------------------------------------------------------------------------------------------------------------------
        Dim lOprCnt As Long
        lOprCnt = caoCtrl.Execute("GetOprLogCount" )
--------------------------------------------------------------------------------------------------------------------------------
```

5.2.11.12. CaoController::Execute("GetOprLog") command

Get data of the operation.

Syntax GetOprLog (<lIndex>)

<lIndex> : Index number (VT_I4) 0 to <Count>-1

Return value : Details (VT_VARIANT[VT_VARIANT|VT_ARRAY:12 elements])

[0]:Error Code (VT_I4)

[1]:Year (VT_I4)

[2]:Month (VT_I4)

[3]:Day of week (VT_I4)

[4]:Day (VT_I4)

[5]:Hour (VT_I4)

[6]:Minute (VT_I4)

[7]:Second (VT_I4)

[8]:Milliseconds (VT_I4)

[9]: Tick count of the controller [ms] (VT_I4)

[10]:Client ID (VT_I4)

-1:Unknown

0:System

1:TP

2:IO

3:PC

4:PAC

5:COM2

6:COM3

7:COM4

[11]:Operation description (VT_BSTR)

Example

```
------------------------------------------------------------------------------------------------------------------------
        Dim lOprCnt As Long
        lOprCnt = caoCtrl.Execute("GetOprLogCount")
        Dim i As Long
        For i=0 To lOprCnt-1
           vDat = caoCtrl.Execute("GetOprLog", i)
           ‘ Hex(vDat(0)) Operation code
           ‘ vDat(11) Operation description
           ‘ :
        Next
------------------------------------------------------------------------------------------------------------------------
```

5.2.11.13. CaoController::Execute("GetPublicValue") command

Read a Public variable value.

Syntax GetPublicValue (<bstrTask> , <bstrName>[, <lIndex1>, <lIndex2>, ･･･])

<bstrTask>	:	Task name (VT_BSTR)
<bstrName>	:	Variable name (VT_BSTR)
<lIndex(n)> <lIndex(n+1)> ･･･	:	Index numbers for each dimension (VT_I4) Assume that "Public pbIArrays(1,2,3) As Long" is specified. Specifying (bstrTask, bstrName, 1, 2, 2) will read one element of pbIArrays(1, 2, 2). If this item is omitted when the target is one dimensional array, the array will be read as a batch.
Return value	:	Values of Public variable

Read a Public variable value in the specified task.

If a Public variable is an array, specifying index number of each dimension will read a value in the specified array.

If an index number is omitted when a Public variable is one-dimensional array, whole elements in the array will be read as a batch.

V type: Read as a F-type (VT_R4) array with three elements.

[0]X, [1]Y, [2]Z

P type: Read as a F-type (VT_R4) array with seven elements.

[0]X, [1]Y, [2]Z, [3]Rx, [4]Ry, [5]Rz, [6]Fig

J type: Read as a F-type (VT_R4) array with eight elements.

[0]J1, [1]J2, [2]J3, [3]J4, [4]J5, [5]J6, [6]J7, [7]J8

T type: Read as a F-type (VT_R4) array with ten elements.

[0]X, [1]Y, [2]Z, [3]Ox, [4]Oy, [5]Oz, [6]Ax, [7]Ay, [8]Az, [9]Fig

[Attention]

A batch reading of a Public variable with two or more dimensional arrays is not supported.

If a batch reading is carried out for a multi-dimensional array, only the elements in one-dimensional array will be read as a batch.

V-, P-, J-, and T-type can be read only when a target Public variable is a Scala variable.

Example

PacScript – Pro1.pcs

```
--------------------------------------------------------------------------------
    Public pblValue As Long
    Public pblArrays(1, 2, 3) As Long
    Public pblArray(2) As Long
    Public pbPValue As Position

    Sub Main

    End Sub
--------------------------------------------------------------------------------
```

VisualBasic – Reading Variable

```
--------------------------------------------------------------------------------
    Dim vParam As Variant
    Dim iVal As Long

    vParam = Array("Pro1", "pblValue")
    iVal = caoCtrl.Execute("GetPublicValue", vParam)
--------------------------------------------------------------------------------
```

VisualBasic – Reading one element of array

```
--------------------------------------------------------------------------------
    Dim vParam As Variant
    Dim iVal As Long

    vParam = Array("Pro1", "pblArrays", 1, 2, 2)
    iVal = caoCtrl.Execute("GetPublicValue", vParam)
--------------------------------------------------------------------------------
```

VisualBasic – Reading one dimensional array as a batch

```
--------------------------------------------------------------------------------
    Dim vParam As Variant
    Dim vArray As Variant

    vParam = Array("Pro1", "pblArray")
    vArray = caoCtrl.Execute("GetPublicValue", vParam)
--------------------------------------------------------------------------------
```

VisualBasic – Reading P-type variable

```
--------------------------------------------------------------------------------
    Dim vParam As Variant
    Dim vVal As Variant

    vParam = Array("Pro1", "pbPValue")
    vVal = caoCtrl.Execute("GetPublicValue", vParam)
--------------------------------------------------------------------------------
```

5.2.11.14. CaoController::Execute("SetPublicValue") command

Write a Public variable value.

Syntax SetPublicValue (<vValue>, <bstrTask> , <bstrName>[, <lIndex1>, <lIndex2>, ･･･])

<vValue>	:	Value of Public variable to be written (given value)
<bstrTask>	:	Task name (VT_BSTR)
<bstrName>	:	Variable name (VT_BSTR)
<lIndex(n)> <lIndex(n+1)> ･･･	:	Index number for each dimension (VT_I4) Assume that "Public pbIArrays(1,2,3) As Long" is specified. Specifying(vValue, bstrTask, bstrName, 1, 2, 2) will write one element of pbIArrays(1, 2, 2). If this item is omitted when the target is one dimensional array, the array will be written as a batch.
Return value	:	None

Write a Public variable value in the specified task.

If a Public variable is an array, specifying index number of each dimension will write a value in the specified array.

If an index number is omitted when a Public variable is one-dimensional array, whole elements in the array will be written as a batch.

V type: Write as a F-type (VT_R4) array with three elements.

[0]X, [1]Y, [2]Z

P type: Write as a F-type (VT_R4) array with seven elements

[0]X, [1]Y, [2]Z, [3]Rx, [4]Ry, [5]Rz, [6]Fig

J type: Write as a F-type (VT_R4) array with eight elements

[0]J1, [1]J2, [2]J3, [3]J4, [4]J5, [5]J6, [6]J7, [7]J8

T type: Write as a F-type (VT_R4) array with ten elements.

[0]X, [1]Y, [2]Z, [3]Ox, [4]Oy, [5]Oz, [6]Ax, [7]Ay, [8]Az, [9]Fig

[Attention]

A batch writing of a Public variable with two or more dimensional arrays is not supported.

If a batch writing is carried out for a multi-dimensional array, an error occurs.

A batch writing of a Public variable with one-dimensional array is available only when the number of elements and the element type are the same in the source array and the destination array.

V-, P-, J-, and T-type can be written only when a target Public variable is a Scala variable.

Example

PacScript – Pro1.pcs

```
Public pblValue As Long
Public pblArrays(1, 2, 3) As Long
Public pblArray(2) As Long
Public pbPValue As Position

Sub Main

End Sub
```

VisualBasic – Writing Variable

```
Dim iVal As Long
Dim vParam As Variant

iVal = 1234
vParam = Array(iVal, "Pro1", "pblValue")
caoCtrl.Execute "SetPublicValue", vParam
```

VisualBasic – Writing one element of array

```
Dim iVal As Long
Dim vParam As Variant

iVal = 1234
vParam = Array(iVal, "Pro1", "pblArrays", 1, 2, 2)
caoCtrl.Execute "SetPublicValue", vParam
```

VisualBasic – Writing one dimensional array as a batch

```
Dim i As Long
Dim iArray(2) As Long
Dim vParam As Variant

For i = 0 To 2
    iArray(i) = i
Next i
vParam = Array(iArray, "Pro1", "pblArray")
caoCtrl.Execute "SetPublicValue", vParam
```

VisualBasic – Writing P-type variable

```
Dim fVals(6) As Single
Dim vParam As Variant

fVals(0) = 1.0
fVals(1) = 2.0
fVals(2) = 3.0
fVals(3) = 1.0
fVals(4) = 2.0
```

```
        fVals(5) = 3.0
        fVals(6) = -1
        vParam = Array(fVals, "Pro1", "pbPValue")
        caoCtrl.Execute "SetPublicValue", vParam
-----------------------------------------------------------------------------------------------------------------------------
```

5.2.11.15. CaoController::Execute("SysState") command

Return the controller status.

This command corresponds to SYSSTATE instruction of PacScript language.

Syntax SysState ( )

Argument : None

Return value : Controller status (VT_I4)

Example

```
-----------------------------------------------------------------------------------------------------------------------------
          Dim state as long
          State = caoCtrl.Execute("SysState")
-----------------------------------------------------------------------------------------------------------------------------
```

5.2.11.16. CaoController::Execute("SysInfo") command

Return the system information of the controller

This corresponds to the SysInfo command of PacScript.

Syntax SysInfo (<1Index>)

< lIndex > : Index number (VT_I4)

Return value : [out] System information of the controller

Index number	System information	Data type
0	Manufacturing number (serial number of the controller)	String type
1	MAC address of built-in network card in the controller	String type
2	Pendant connection state 0: Not connected 1: Teach pendant is connected 2: Mini-pendant is connected	Integer type
3	Global variables information The numbers from 0 to 7 correspond to the variable types as shown below. 0 : I type 1 : F type	Integer type array

	2:D type 3:V type 4:P type 5:J type 6:T type 7:S type	
4	CPU information 0: Not identified (Virtual environment) 2: ATOM 5: i7	Integer type
5	Login user level 1000: Operator 2000: Programmer 3000: Maintainer 3001or more: Reserved for System	Integer type
6	Supervisory tasks status -1:Running 0:Not running	Integer type
7	TP panel display -1:Displayed 0:Not displayed	Integer type
8	Total operation (min.)	Integer type
9	Total runnung (min.)	Integer type
10	Cumulative operation (min.)	Integer type
11	Cumulative running (min.)	Integer type
12	Operation (min.)	Integer type
13	Running (min.)	Integer type
14	Motor-ON count	Integer type
15	Encoder battery maintenance date -1: Maintenance date has passed 0 : Maintenance date has not arrived yet	Integer type
16	Controller battery maintenance date -1: Maintenance date has passed 0 :Maintenance date has not arrived yet	Integer type

Example

```
Dim sysinfo as long
sysinfo = caoCtrl.Execute("SysInfo",0)
```

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 ("Pro1.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.

This method is not available in SlaveMode.

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

```
    Accelerate 0, 50.0, -1    ' Acceleration = 50%, deceleration = no change
    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 (when 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.

This method is not available in SlaveMode.

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
[SPEED=n][,ACCEL=n][,DECEL=n][,TIME=n][,NEXT]
SPEED (S): Specify the movement speed. The meaning is the same as the SPEED statement.
ACCEL: Specify the acceleration. The meaning is the same as the ACCEL statement.
DECEL: Specify the deceleration. The meaning is the same as the DECEL statement.
TIME: Specify the time to activate the motion.
NEXT: Asynchronous execution option.

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>, [<@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.[4]
(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

[4] 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.

Example 6	Move 2,"@E P20 EXA((7, 30.8), (8, 90.5))	‘MOVE L, @E P20 EXA((7, 30.8), (8, 90.5))"

This method is not available in SlaveMode.

The following table shows the PacScript MOVE commands supported by Move method.

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 P, P<*n1*> EX((*j1*, *v1*),(*j2*, *v2*)) MOVE P, P<*n1*> EXA((*j1*, *v1*)) MOVE P, P<*n1*> EXA((*j1*, *v1*),(*j2*, *v2*))	Move 1,"P<*n1*> EX((*j1*,*v1*))" Move 1,"P<*n1*> EX((*j1*,*v1*),(*j2*, *v2*))" Move 1,"P<*n1*> EXA((*j1*,*v1*))" 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.

This method is not available in SlaveMode.

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] Pass

[@0][@P][@E][@ <Value>]

Rotary option

[,Pose=n]

Pose=1: The posture changes along with rotation.

Pose=2: The posture at the current position (start point) is maintained and rotation motion is performed for the track only.

Motion option

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

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

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

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

TIME: Specify the time to activate the motion.

NEXT: Asynchronous execution option.

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 deterimined by three points of V type variables in base coordinates. For arguments of vntRotSuf, specify 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.

Actual speed is calculated by multiplying the external speed by the internal speed.

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

This method is not available in SlaveMode.

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        ‘ External speed 50%,   Actual speed 85*50 = 42.5%

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

Internal speed can be changed temporarily per unit, by using SPEED option of the robot motion command, such as Move, Rotate, Drive and Draw. However, the internal speed will be reset to the value specified by CaoRobot::Speed after the motion completes. Normally, internal speed is initialized to 100% by TakeArm command of CaoRobot::Execute.

5.2.28. CaoRobot::Execute method

The "Execute method" defines peculiar operation commands to the robot that is not 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.84
	TInv	Calculate the inverse matrix of homogeneous transformation type data.	P.84
	TNorm	Calculate the inverse matrix of homogeneous transformation type data.	P.85
	J2T	Transform J type data to T type data.	P.85
	T2J	Transform T type data to J type data.	P.86
	J2P	Transform J type data to P type data.	P.86
	P2J	Transform P type data to J type data.	P.87
	T2P	Transform T type data to P type data.	P.87
	P2T	Transform P type data to T type data.	P.88

	Dev	Calculate the offset in the base coordinates.	P.88
	DevH	Calculate the offset in the tool coordinates.	P.89
	OutRange	Judge whether the motion range is exceeded.	P.89
	MPS	Calculate the value of the SPEED command from data in Mps.	P.90
	RPM	Calculate the value of the SPEED command from data in rpm.	P.90
	DPS	Calculate the value of SPEED command from deg/s unit	P.74
Positioning			
	CurPos	Get the current position as P type data.	P.91
	DestPos	Get the target position as P type data.	P.92
	CurJnt	Get the current position as J type data.	P.92
	DestJnt	Get the target position as J type data.	P.94
	CurTrn	Get the current position as T type data.	P.94
	DestTrn	Get the target position as T type data.	P.97
	CurFig	Get the current posture Fig value.	P.97
Log			
	StartLog	Start log recording.	P.99
	StopLog	Stop log recording.	P.101
	ClearLog	Clear log data.	P.102
Robot operation			
	Motor	Turn ON/OFF the motor.	P.102
	ExtSpeed	Set the external speed.	P.103
	TakeArm	Request to get control authority.	P.103
	GiveArm	Request to release control authority.	P.103
	Draw	Execute the relative movement specified in the work coordinate system.	P.105
	Approach	Execute the absolute movement specified in the tool coordinate system.	P.106
	Depart	Execute the relative movement specified in the tool coordinate system.	P.107
	DriveEx	Execute the relative motion of each axis.	P.108
	DriveAEx	Execute the absolute motion of each axis.	P.109

	RotateH	Execute rotary motion by taking an approach vector as an axis.	P.110
	Arrive	Wait for the robot to reach the defined motion ratio.	P.110
	MotionSkip	Abort the robot motion in progress.	P.111
	MotionComplete	Judge whether the robot motion is complete.	P.111
	ArchMove	Perform an arch motion.	P.121
Tool			
	CurTool	Get the current tool number.	P.112
	GetToolDef	Get the tool definition specified by the tool number.	P.112
	SetToolDef	Set the tool definition.	P.113
Work			
	CurWork	Get the current work number.	P.113
	GetWorkDef	Get the work definition specified by the work number.	P.113
	SetWorkDef	Set the work definition.	P.114
Area			
	GetAreaDef	Get the area definition specified by the area number.	P.114
	SetAreaDef	Set the area parameter.	P.115
	SetArea	Enable the area check.	P.116
	ResetArea	Disable the area check.	P.116
	AreaSize	Return the size (each side length) of a check area as the vector type.	P.117
	GetAreaEnabled	Get the area enabled or disabled status.	P.117
	SetAreaEnabled	Set the area enabled or disabled status.	P.117
Spline			
	AddPathPoint	Add a path point to the path data.	P.118
	ClrPathPoint	Clear the whole path points.	P.118
	GetPathPoint	Get a path point to the path data.	P.119
	LoadPathPoint	Load a path data.	P.119
	GetPathPointCount	Get the number of path points.	P.120
Misc.			
	GetRobotTypeName	Get the robot type.L.	P.120
	CrtMotionAllow	Change the stop precision settings.(@C)	P.121

EncMotionAllow	Change the "Allowable angle in stop state" of robot axis.(@E)	P.122
EncMotionAllowJnt	Change the "Allowable angle in stop state" of other than robots' axis.(@E)	P.122
ErAlw	Configure the deviation tolerance function.	P.123
ForceCtrl	Enable/Disable the force control function.	P.123
ForceParam	Set parameters for force control function.	P.124
GetSrvData	Get servo internal data of the robot axis.	P.125
GetSrvJntData	Get servo internal data of the other than robots' axis.	P.128
GrvCtrl	Switch "True/False" of Gravity Compensation Control Function.	P.128
CurLmt	Configure the current limiting function.	P.129
Zforce	Specify the thrust force for current limiting function.	P.129
GrvOffset	Configure "True/False" of dead weight correction function.	P.130
HighPathAccuracy	Switch "True/False" of the high tracing control function.	P.130
MotionTimeout	Change a timeout setting value of the motion instruction.	P.130
SingularAvoid	Enable or disable singularity avoidance function.	P.131
SpeedMode	Change the optimal speed setting function.	P.131
PayLoad	Change the setting value of internal load conditions.	P.132

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("DPS") command

Convert the rotation speed (deg/s) of the specified axis to the ratio against the maximum internal speed at PTP motion. If the axis number is not specified, convert the rotation speed (deg/s) to the ratio against the maximum internal speed at CP motion.

Syntax DPS(<Axis>, <deg/s>)

<Axis>	:	[in] Axis number (VT_I4)
<deg/s>	:	[in] Value of rotation speed by "deg/s" (VT_R4)
Return value	:	Value of SPEED command by "%" (VT_R4)

Example Conversion from the rotation speed (deg/s) to the relative speed (%)

```
Dim vSp As Variant
' Move by 50(Deg/sec) (when in rotation)
vSp = g_caoRobot.Execute("DPS", 50)
          caoRob.Speed -1,vSp

          ' Move the first axis by 50(deg/sec)
vSp = g_caoRobot.Execute("DPS", Array(1, 50))
caoRob.Speed 0,vSp
```

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

Get the current position, which is updated in every 8milliseconds in the controller, by 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
```

5.2.28.17. 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
--------------------------------------------------------------------------------------------------------------------------------
```

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

Get the current position data, which is updated in every 8milliseconds in the controller, by P type data with time stamp.

Time stamp：Elapsed time from turning the controller power supply on (msec)

Syntax CurPosEx( )

Argument	:	None
Return value	:	Time stamp＋P type (VT_VARIANT[VT_R8|VT_ARRAY:1+7 elements]) {Time, X, Y, ...}

Example

```
--------------------------------------------------------------------------------------------------------------------------------
        Dim vResult As Variant
        vResult = caoRob.Execute("CurPosEx" )     'Time stamp+ Get current position
--------------------------------------------------------------------------------------------------------------------------------
```

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

Get the target position data with time stamp as P type data.

Time stamp：Elapsed time from turning the controller power supply on (msec)

Syntax DestPosEx( )

Argument : None

Return value : Time stamp＋P type

(VT_VARIANT[VT_R8|VT_ARRAY:1+7 elements])

{Time, X, Y, ...}

Example

```
-----------------------------------------------------------------------------------------------
        Dim vResult As Variant
        vResult = caoRob.Execute("DestPosEx" )     'Time stamp+ Get target position
-----------------------------------------------------------------------------------------------
```

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

Get the current position data in real time with time stamp as P type data.

Encoder value is returned if the robot is not machine-locked.

This command might cause a high load on the controller.

Time stamp：Elapsed time from turning the controller power supply on (msec)

Syntax HighCurPosEx( )

Argument : None

Return value : Time stamp＋P type

(VT_VARIANT[VT_R8|VT_ARRAY:1+7 elements])

{Time, X, Y, ...}

Example

```
-----------------------------------------------------------------------------------------------
        Dim vResult As Variant
        vResult = caoRob.Execute("HighCurPosEx" )     'Time stamp+ Get current position
-----------------------------------------------------------------------------------------------
```

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

Get the current position data, which is updated in every 8milliseconds in the controller, by 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
------------------------------------------------------------------------------------------------------------------------------------
```

5.2.28.22. 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
------------------------------------------------------------------------------------------------------------------------------------
```

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

Get the current position data, which is updated in every 8milliseconds in the controller, by J type data with time stamp.

Time stamp : Elapsed time from turning the controller power supply on (msec)

Syntax CurJntEx( )

Argument	:	None
Return value	:	Time stamp＋J type (VT_VARIANT[VT_R8|VT_ARRAY:1+8 elements]) {Time, J1, J2, ...}

Example

```
------------------------------------------------------------------------------------------------------------------------
        Dim vResult As Variant
        vResult = caoRob.Execute("CurJntEx" )     'Time stamp+Get current position
------------------------------------------------------------------------------------------------------------------------
```

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

Get the target position data with time stamp as J type data.

Time stamp : Elapsed time from turning the controller power supply on (msec)

Syntax DestJntEx( )

Argument	:	None
Return value	:	Time stamp＋J type (VT_VARIANT[VT_R8|VT_ARRAY:1+8 elements]) {Time, J1, J2, ...}

Example

```
------------------------------------------------------------------------------------------------------------------------
        Dim vResult As Variant
        vResult = caoRob.Execute("DestJntEx" )     'Time stamp+Get target position
------------------------------------------------------------------------------------------------------------------------
```

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

Get the current position data in real time with time stamp as J type data.

Encoder value is returned if the robot is not machine-locked.

This command might cause a high load on the controller.

Time stamp：Elapsed time from turning the controller power supply on (msec)

Syntax HighCurJntEx( )

Argument : None

Return value : Time stamp＋J type
(VT_VARIANT[VT_R8|VT_ARRAY:1+8 elements])
{Time, J1, J2, ...}

Example

```
------------------------------------------------------------------------------------------------------------------------
        Dim vResult As Variant
        vResult = caoRob.Execute("HighCurJntEx" )     'Time stamp+Get current position
------------------------------------------------------------------------------------------------------------------------
```

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

Get the current position data, which is updated in every 8milliseconds in the controller, by 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
------------------------------------------------------------------------------------------------------------------------
```

5.2.28.27. 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
-----------------------------------------------------------------------------------------------------------------------------
```

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

Get the current position data, which is updated in every 8milliseconds in the controller, by T type data with time stamp.

Time stamp：Elapsed time from turning the controller power supply on (msec)

Syntax CurTrnEx( )

Argument	:	None
Return value	:	Time stamp＋T type (VT_VARIANT[VT_R8|VT_ARRAY:1+10 elements]) {Time, X, Y, ...}

Example

```
-----------------------------------------------------------------------------------------------------------------------------
        Dim vResult As Variant
        vResult = caoRob.Execute("CurTrnEx" )     'Time stamp+Get current position
-----------------------------------------------------------------------------------------------------------------------------
```

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

Get the target position data with time stamp as T type data.

Time stamp：Elapsed time from turning the controller power supply on (msec)

Syntax DestTrnEx( )

Argument	:	None
Return value	:	Time stamp＋T type (VT_VARIANT[VT_R8|VT_ARRAY:1+10 elements]) {Time, X, Y, ...}

Example

```
--------------------------------------------------------------------------------
        Dim vResult As Variant
        vResult = caoRob.Execute("DestTrnEx" )     'Time stamp+Get target position
--------------------------------------------------------------------------------
```

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

Get the current position data in real time with time stamp as T type data.

Encoder value is returned if the robot is not machine-locked.

This command might cause a high load on the controller.

Time stamp：Elapsed time from turning the controller power supply on (msec)

Syntax HighCurTrnEx( )

Argument	:	None
Return value	:	Time stamp＋T TYPE (VT_VARIANT[VT_R8|VT_ARRAY:1+10 elements]) {Time, X, Y, ...}

Example

```
--------------------------------------------------------------------------------
        Dim vResult As Variant
        vResult = caoRob.Execute("HighCurTrnEx" )     'Time stamp+Get current position
--------------------------------------------------------------------------------
```

5.2.28.31. 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
------------------------------------------------------------------------------------------
```

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

Return an internal speed setting value.

Syntax CurSpd([arm group])

Argument : Arm group(VT_I4)

Return value : Internal speed (VT_R4)

Example

```
------------------------------------------------------------------------------------------
        Dim vSpd As Variant
        vSpd = caoRob.Execute("CurSpd" ,0)
------------------------------------------------------------------------------------------
```

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

Return an internal acceleration setting value.

Syntax CurAcc([arm group])

Argument : Arm group(VT_I4)

Return value : Internal acceleration (VT_R4)

Example

```
------------------------------------------------------------------------------------------
        Dim vSpd As Variant
        vSpd = caoRob.Execute("CurAcc" ,0)
------------------------------------------------------------------------------------------
```

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

Return an internal deceleration setting value.

Syntax CurDec([arm group])

Argument	:	Arm group(VT_I4)
Return value	:	Internal deceleration (VT_R4)

Example

```
--------------------------------------------------------------------------------
        Dim vSpd As Variant

        vSpd = caoRob.Execute("CurDec" ,0)
--------------------------------------------------------------------------------
```

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

Return internal axis speed.

Syntax CurJSpd([arm group])

Argument	:	Arm group(VT_I4)
Return value	:	Internal axis speed (VT_R4)

Example

```
--------------------------------------------------------------------------------
        Dim vSpd As Variant

        vSpd = caoRob.Execute("CurJSpd" ,0)
--------------------------------------------------------------------------------
```

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

Return internal axis acceleration.

Syntax CurJAcc([arm group])

Argument	:	Arm group(VT_I4)
Return value	:	Internal axis acceleration (VT_R4)

Example

```
--------------------------------------------------------------------------------
        Dim vSpd As Variant

        vSpd = caoRob.Execute("CurJAcc" ,0)
--------------------------------------------------------------------------------
```

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

Return internal axis deceleration.

Syntax CurJDec([arm group])

Argument : Arm group(VT_I4)

Return value : Internal axis deceleration (VT_R4)

Example

```
Dim vSpd As Variant

vSpd = caoRob.Execute("CurJDec" ,0)
```

5.2.28.38. 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.

This method is not available in SlaveMode.

Syntax StartLog( )

Argument : None

Return value : None

Execute the ClearLog command before this command to enable recording.

Example

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

5.2.28.39. 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"
```

5.2.28.40. 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.

This method is not available in SlaveMode.

Example

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

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

Turn ON/OFF the motor.

This method is not available in SlaveMode.

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

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

Set the external speed, external acceleration, and external deceleration.

This method is not available in SlaveMode.

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 "-2" is entered if it is omitted.
Decel	:	[in] External deceleration (VT_R4) -1 (default) Do not change the current setting "-2" is entered if it is omitted.
Return value	:	None

Example

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

        caoRob.Execute"ExtSpeed", Array(50.0)    ‘ External speed=50% (Acceleration, deceleration are set
automatically)

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

Actual speed is calculated by multiplying the external speed by the internal speed. Internal speed is specified by Speed command.

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

Request to get control authority.

This command corresponds to TAKEARM instruction of PacScript language.

This method is not available in SlaveMode.

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

ArmGroup	:	[in] Arm group number (VT_I4) 0 to 31 (0 by default) (0 is an Armgroup that includes robot joints only and does not include any Extended-joints)
Keep	:	[in] Default value (VT_I4) 0: Set the internal speed to 100, the current tool number to 0, and the current work number to 0. 1: Keep the current internal speed, the current tool number and the current work number without change.

(0 by default)

※If Keep option is not specified, the internal speed, the current tool number and the current work number are initialized to 100, 0 and 0, respectively.

Return value : None

Example

```
------------------------------------------------------------------------------------------------------------------------------
          caoRob.Execute"Takearm",Array(0,0)     'Intialize the internal speed to 100, the current tool to 0, and the
current wor to 0.
          :

          'When the internal speed is 50, the current tool is 1, and the current work is 0
          caoRob.Execute"Takearm",Array(0,1) 'Not initialize the internal speed, current tool and current work,
          'and then get the control authority only.
------------------------------------------------------------------------------------------------------------------------------
```

5.2.28.44. 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

```
------------------------------------------------------------------------------------------------------------------------------
          ' When a program that has executed Takearm is terminated,
          ' the robot halts before Givearm is executed.
          ' To terminate the program after completion of Next motion,
          ' explicitly execute Givearm command.

          caoRob.Execute"Takearm",Array(0,0)
          caoRob.Move 1,"@0 J(0,45,90,0,45,0)"
          caoRob.Move 1,"@0 J(90,45,90,0,45,0)","Next"
          Set IO[129]
          Set IO[130]
          caoRob.Execute"GiveArm"       ' Wait until Next motion completes.'

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

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

Execute the relative movement specified in the work coordinate system

This command corresponds to DRAW instruction of PacScript language.

This method is not available in SlaveMode.

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][,TIME=n][,NEXT] SPEED (S): Specify the movement speed. The meaning is the same as the SPEED statement. ACCEL: Specify the acceleration. The meaning is the same as the ACCEL statement. DECEL: Specify the deceleration. The meaning is the same as the DECEL statement. TIME: Specify the time to activate the motion. NEXT: Asynchronous execution option.
Return value	:	None

Example

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

5.2.28.46. 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.

This method is not available in SlaveMode.

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][,TIME=n][,NEXT]
SPEED (S): Specify the movement speed. The meaning is the same as the SPEED statement.
ACCEL: Specify the acceleration. The meaning is the same as the ACCEL statement.
DECEL: Specify the deceleration. The meaning is the same as the DECEL statement.
TIME: Specify the time to activate the motion.
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")
--------------------------------------------------------------------------------------------------------------------------------
```

5.2.28.47. 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.

This method is not available in SlaveMode.

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][,TIME=n][,NEXT]
SPEED (S): Specify the movement speed. The meaning is the same as the SPEED statement.
ACCEL: Specify the acceleration. The meaning is the same as the ACCEL statement.
DECEL: Specify the deceleration. The meaning is the same as the DECEL statement.
TIME: Specify the time to activate the motion.
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")
------------------------------------------------------------------------------------------------------------------------
```

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

Execute the relative motion of each axis.

This command corresponds to DRIVE instruction of PacScript language.

This method is not available in SlaveMode.

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][,TIME=n][,NEXT]
SPEED (S): Specify the movement speed. The meaning is the same as the SPEED statement.
ACCEL: Specify the acceleration. The meaning is the same as the ACCEL statement.
DECEL: Specify the deceleration. The meaning is the same as the DECEL statement.
TIME: Specify the time to activate the motion.
NEXT: Asynchronous execution option.

Return value : None

Example

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

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

Execute the absolute motion of each axis.

This command corresponds to DRIVEA instruction of PacScript language.

This method is not available in SlaveMode.

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][,TIME=n][,NEXT]
SPEED (S): Specify the movement speed. The meaning is the same as the SPEED statement.
ACCEL: Specify the acceleration. The meaning is the same as the ACCEL statement.
DECEL: Specify the deceleration. The meaning is the same as the DECEL statement.
TIME: Specify the time to activate the motion.
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")
--------------------------------------------------------------------------------
```

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

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

This command corresponds to ROTATEH instruction of PacScript language.

This method is not available in SlaveMode.

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][,TIME=n][,NEXT]
SPEED (S): Specify the movement speed. The meaning is the same as the SPEED statement.
ACCEL: Specify the acceleration. The meaning is the same as the ACCEL statement.
DECEL: Specify the deceleration. The meaning is the same as the DECEL statement.
TIME: Specify the time to activate the motion.
NEXT: Asynchronous execution option.

Return value : None

Example

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

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

Wait for the robot to reach the defined motion ratio.

This command corresponds to ARRIVE instruction of PacScript language.

This method is not available in SlaveMode.

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

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

Abort the robot motion in progress.

This command corresponds to MOTIONSKIP instruction of PacScript language.

This method is not available in SlaveMode.

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)
------------------------------------------------------------------------------------------------------------------------------
```

5.2.28.53. 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
```

5.2.28.54. 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")
```

5.2.28.55. 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)
```

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

Set the tool definition.

This method is not available in SlaveMode.

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)")
----------------------------------------------------------------------------------------------------------------
```

5.2.28.57. 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( "urWork")
----------------------------------------------------------------------------------------------------------------
```

5.2.28.58. 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)
----------------------------------------------------------------------------------------------------------------
```

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

Set the work definition.

This method is not available in SlaveMode.

Syntax SetWorkDef (<WorkNo>, <WorkDef><WorkAttribute>)

WorkNo : [in] Work number (VT_I4)

WorkDef : [in] Work definition (P type Fig is ignored.)
X, Y, Z, RX, RY, RZ

WorkAttribute : [in] Work attribute (VT_I4) (If it is omitted, 0 is specified.)
0:Standard(default)
1:FixedTool(Fix)

Return value : None

Example

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

5.2.28.60. CaoRobot::Execute("WorkAttribute") command

Get work attribute specified by a work number

Syntax WorkAttribute (<WorklNo>)

WorkNo : [in] Work number (VT_I4)

Return value : Work attribute (VT_I4)

Example

```
--------------------------------------------------------------------------------
        Dim vVal As Variant
        vVal = caoRob.Execute("WorkAttribute", 1)
--------------------------------------------------------------------------------
```

5.2.28.61. 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)
--------------------------------------------------------------------------------
```

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

Set the area parameter.

This method is not available in SlaveMode.

Syntax1 SetAreaDef (<Area number>, <Center>, <Size>, <I/O number>, <Variable storage number>[,<Area detection setting>])

Syntax2 SetAreaDef (<Area number>, <Area definition>)

Argument : Syntax 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)
Syntax2:
[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)
------------------------------------------------------------------------------------------------------------------------
```

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

Enable the area check.

This method is not available in SlaveMode.

Syntax SetArea (<AreaNum>)

<AreaNum>	:	Area number (VT_I4)
Return value	:	None

Example

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

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

Disable the area check.

This method is not available in SlaveMode.

Syntax ResetArea (<AreaNum>)

<AreaNum>	:	Area number (VT_I4)
Return value	:	None

Example

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

5.2.28.65. 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)
```

5.2.28.66. 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
```

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

Set the area enabled or disabled status.

This method is not available in SlaveMode.

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
```

5.2.28.68. CaoRobot::Execute("AddPathPoint") command

Add a path point to the path data.

This method is not available in SlaveMode.

If the path number that you have specified already has different path points, the newly specified path points will be added after the existing path points.

This command corresponds to ADDPATHPOINT instruction of PacScript language

Syntax AddPathPoint (<PathNum>, < Pose >)

< PathNum >	:	[in] Path number（1 to 20） (VT_I4)
< Pose >	:	[in] POSEDATA value (one of P, J, and T types)
Return value	:	None

Example

```
------------------------------------------------------------------------------------------------------------------------
        caoRobot.Execute "AddPathPoint", Array(2, "P(400, 250, 140, 180, 0, 180)")
------------------------------------------------------------------------------------------------------------------------
```

5.2.28.69. CaoRobot::Execute("ClrPathPoint") command

Clear the whole path points at the specified path.

This method is not available in SlaveMode.

This command corresponds to CLRPATHPOINT instruction of PacScript language

Syntax ClrPathPoint (<PathNum>)

< PathNum >	:	[in] Path number（1 to 20） (VT_I4)
Return value	:	None

Example

```
------------------------------------------------------------------------------------------------------------------------
        caoRobot.Execute "ClrPathPoint", 2
------------------------------------------------------------------------------------------------------------------------
```

5.2.28.70. CaoRobot::Execute("GetPathPoint") command

Return a position data of specified path point.

This command corresponds to GETPATHPOINT instruction of PacScript language

Syntax GetPathPoint (<PathNum>, <PointNum>)

< PathNum > : [in] Path number（1 to 20）(VT_I4)

< PointNum > : [in] Path point number（1 to 5000）(VT_I4)

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

Example

```
Dim vntPos as Variant
vntPos = caoRobot.Execute("GetPathPoint", Array(2, 1))
```

5.2.28.71. CaoRobot::Execute("LoadPathPoint") command

Load a path data.

This method is not available in SlaveMode.

Clear the path data of the specified path number, and then read out the path data stored in the data storage memory.

This command corresponds to LOADPATHPOINT instruction of PacScript language

Syntax LoadPathPoint (<PathNum>)

< PathNum > : [in] Path number（1 to 20）(VT_I4)

Return value : None

Example

```
caoRobot.Execute "LoadPathPoint", 2
```

5.2.28.72. CaoRobot::Execute("GetPathPointCount ") command

Return the number of path points at the specified path.

This command corresponds to GETPATHPOINTCOUNT instruction of PacScript language

Syntax GetPathPointCount (<PathNum>)

< PathNum >	:	[in] Path number（1 to 20） (VT_I4)
Return value	:	Number of path points (VT_I4)

Example

```
------------------------------------------------------------------------------------------------------------------------
        Dim lCount As Long
        lCount = caoRobot.Execute("GetPathPointCount", 2)
------------------------------------------------------------------------------------------------------------------------
```

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

Get the robot type.

Syntax GetRobotTypeName ( )

Argument	:	None
Return value	:	Robot type (VT_BSTR)

Example

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

5.2.28.74. CaoRobot::Execute("ArchMove") command

Perform an arch motion.

This command corresponds to ARCHMOVE instruction of PacScript language

This method is not available in SlaveMode.

Syntax ArchMove (<TgtPose>,<Hight>[,<ArchStartPos>[,<ArchEndPos>]])

< TgtPose >	:	[in] Target position (VT_I4)
< Hight >	:	[in] Height[mm] (VT_R4)
< ArchStartPos >	:	[in] Arch start position[mm] (VT_R4) If the argument is omitted, 0 is assumed to be specified.
< ArchEndPos >	:	[in] Arch finish position[mm] (VT_R4) If the argument is omitted, 0 is assumed to be specified.
Return value	:	None

Example

```
--------------------------------------------------------------------------------------------------------------------------
        caoRobot.Execute "ArchMove",Array( "P10",50,30,30 )
--------------------------------------------------------------------------------------------------------------------------
```

5.2.28.75. CaoRobot::Execute("CrtMotionAllow") command

Change the stop positional precision and postural precision settings of Move @C command.

This command corresponds to CRTMOTIONALLOW instruction of PacScript language.

This method is not available in SlaveMode.

Syntax1 CrtMotionAllow (<True>,<Position>[,<Posture>])

Syntax2 CrtMotionAllow (<False>)

< True/False >	:	[in] True/False (VT_I4) True (other than 0) or False (0)
< Position >	:	[in] Positional precision [mm] (VT_R4)
< Posture >	:	[in] Postural precision [degree] (VT_R4)
Return value	:	None

Example

```
--------------------------------------------------------------------------------------------------------------------------
        caoRobot.Execute "CrtMotionAllow", Array(True, 1, 1 )
        caoRobot.Move 1, "@C J2"
        caoRobot.Execute "CrtMotionAllow", False
--------------------------------------------------------------------------------------------------------------------------
```

5.2.28.76. CaoRobot::Execute("EncMotionAllow") command

With Move @E, change the "Allowable angle in stop state" of each axis for robot axis used for stop judgement.

This command corresponds to ENCMOTIONALLOW instruction of PacScript language.

This method is not available in SlaveMode.

Syntax1 EncMotionAllow (<True>,<Angle>[,<Mode>])

Syntax2 EncMotionAllow (<False>)

< True/False > : [in] True/False (VT_I4)
True (other than 0) or False (0)

< Angle > : [in] Allowable angle (VT_R4)

< Mode > : [in] Mode value (VT_I4)
0: [degree] or [mm](default)
1: [pulse]

Return value : None

Example

```
--------------------------------------------------------------------------------------------------------------------------------
        caoRobot.Execute "EncMotionAllow", Array(True, 1, 1 )
        caoRobot.Move 1, "@E J2"
        caoRobot.Execute "EncMotionAllow", False
--------------------------------------------------------------------------------------------------------------------------------
```

5.2.28.77. CaoRobot::Execute("EncMotionAllowJnt") command

With Move @E, change the "Allowable angle in stop state" of the axis for other than robots' axis used for stop judgement.

This command corresponds to ENCMOTIONALLOWJNT instruction of PacScript language.

This method is not available in SlaveMode.

Syntax1 EncMotionAllowJnt (<True>,<Axis>,<Angle>[,<Mode>])

Syntax2 EncMotionAllowJnt (<False>,<Axis>)

< True/False > : [in] True/False (VT_I4)
True (other than 0) or False (0)

< Axis > : [in] Axis number (VT_R4)

< Angle > : [in] Allowable angle (VT_R4)

< Mode > : [in] Mode value (VT_I4)
0: [degree] or [mm](default)
1: [pulse]

Return value : None

Example

```
caoRobot.Execute "EncMotionAllowJnt", Array(True, 7, 0.01, 1)
caoRobot.Move 1, "@E J2 EXA(7, 30.5)"
caoRobot.Execute "EncMotionAllowJnt", Array(False, 7)
```

5.2.28.78. CaoRobot::Execute("ErAlw") command

Configure the deviation tolerance function and True/False.

This command corresponds to ERALW instruction of PacScript language.

This method is not available in SlaveMode.

Syntax1 ErAlw (<True>,<Axis>,<Value>)

Syntax2 ErAlw (<False>,<Axis>)

< True/False > : [in] True/False (VT_I4)
True (other than 0) or False (0)

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

< Value > : [in] Setting value ([degree] or [mm]) (VT_R4)

Return value : None

Example

```
caoRobot.Execute "ErAlw", Array(True, 1, 0.01)
caoRobot.Execute "ErAlw", Array(True, 2, 0.01)
caoRobot.Execute "ErAlw", Array(False, 0)
```

5.2.28.79. CaoRobot::Execute("ForceCtrl") command

Enable/disable the force control function (compliance function).

This command corresponds to FORCECTRL instruction of PacScript language.

This method is not available in SlaveMode.

Syntax1 ForceCtrl (<True>,<CtrlNum>)

Syntax2 ForceCtrl (<False>)

< True/False > : [in] True/False (VT_I4)
True (other than 0) or False (0)

< CtrlNum > : [in] Force Control Number(1 to 10) (VT_I4)

< Mode> [in]Control mode [VT_I4]
0: Compliance Function
1: Compliance Function with Force sensor

If it is omitted, the value is determined according to the parameter table number for compliance function.

Return value : None

Example

```
caoRobot.Execute "ForceCtrl", Array(True, 1)
caoRobot.Execute "ForceCtrl", False
```

5.2.28.80. CaoRobot::Execute("ForceParam") command

Set parameters for force control function (compliance function).

This command corresponds to FORCEPARAM instruction of PacScript language.

This method is not available in SlaveMode.

Syntax ForceParam (<CtrlNum>,<Coordinates>,<Force>,[<PosEralw>,[<Spring>,[<Damp>,[<Mass>,
[<CurLmt>,[<OffSet>,[<Eralw>, [<lMode>,[<Rate>,[<SpMax>,<RSpMax>]]]]]]]]])

< CtrlNum > : [in] Force Control Number(1 to 10) (VT_I4)

< Coordinates > : [in] Coordinates (VT_I4)
0: base coordinates
1: tool coordinates
2: work coordinates

< Force > : [in] Force
P type(VT_VARIANT[VT_R8|VT_ARRAY:6 element])

< PosEralw > : [in] Allowable Position Deviation
P type(VT_VARIANT[VT_R8|VT_ARRAY:6 element])

< Spring > : [in] Compliance
P type(VT_VARIANT[VT_R8|VT_ARRAY:6 element])

< Damp > : [in] Damping
P type(VT_VARIANT[VT_R8|VT_ARRAY:6 element])

< Mass > : [in] Inertia
P type(VT_VARIANT[VT_R8|VT_ARRAY:6 element])

< CurLmt > : [in] Current Limit Value
J type(VT_VARIANT[VT_R8|VT_ARRAY:6 element])

< OffSet > : [in] Offset Value
P type(VT_VARIANT[VT_R8|VT_ARRAY:6 element])

< Eralw > : [in] Allowable Axis Deviation
J type(VT_VARIANT[VT_R8|VT_ARRAY:6 element])

<lMode>		[in] Speed control mode [VT_I4] (Reserved area for future use. Specify "0")
<Rate>	:	[in]Control rate [P type]
<SpMax>	:	[in]Maximum translation speed [VT_R8]
<RSpMax>	:	[in]Maximum rotation speed [VT_R8]
Return value	:	None

Example

```
--------------------------------------------------------------------------------
        caoRobot.Execute "ForceParam", Array(1, 1, "P10")
--------------------------------------------------------------------------------
```

5.2.28.81. CaoRobot::Execute("ForceValue") command

Get values of force control specified by arguments.

Syntax ForceValue (<DataNo> [,<Mode>])

< DataNo > : [in]Data number [VT_I4]

0 : Sensor value [N|Nm]. Get the sensor value on the control coordinatesystem as P type..

1 : Sensor value [pulse] Get the sensor output value as P type.

2 :The positive peak of the force and moment on the control coordinate system [N|Nm]

3 : The negative peak of the force and moment on the control coordinate system [N|Nm]

4 : Travel distance of the tool end from the control start position on the control coordinate system. (command value) [mm|deg]

5 : Positive peak of the tool end's travel distance from the control start position on the control coordinate system (command value) [mm|deg]

6 : Negative peak of the tool end's travel distance from the control start position on the control coordinate system (command value)[mm|deg]

7 : Travel distance of the tool end from the control start position on the control coordinate system. (current value) [mm|deg]

8 : Positive peak of the tool end's travel distance from the control start position on the control coordinate system (current value) [mm|deg]

9 : Negative peak of the tool end's travel distance from the control start position on the control coordinate system (current value) [mm|deg]

10 : Deviation between the command value and the force control command value that occurs from the force control start, on the control coordinate system. [mm|deg]

11 : Positive peak in the deviation between the command value and the force control command value that occurs from the force control start, on the control coordinate system. [mm|deg]

12 : Negative peak in the deviation between the command value and the force control command value that occurs from the force control start, on the control coordinate system. [mm|deg]

< Mode > : [in]Mode [VT_I4]

0 : Get data

-1 : Reset the peal value

If it is omitted, 0 is assumed to be specified.

Return value : [out] Value of force control [VT_VARIANT]

If Mode is "0": A force control value according to the data number

If Mode is "-1" : Value immediately before the reset

Example

```
vntVal = caoRobot.Execute("ForceValue", Array(1, 0))
```

5.2.28.82. CaoRobot::Execute("ForceWaitCondition") command

Wait until the condition specified by the force control is satisfied.

This method is not available in SlaveMode.

Syntax ForceWaitCondition ([<Position> [,<Force> [,<Time> [,<Mode> [,<Timeout>]]]]])

< Position > : [in]Travel distance [P type]

Set the travel distance of the tool end from the control start position. [mm], [deg]

If it is omitted, "-1" is assumed to be specified.

<Force> : [in]Force and morment [P type]

Set the force and moment converted to the force control coordinate system. [N],[Nm]

If it is omitted, "-1" is assumed to be specified.

<Time> : [in]Elapsed time [VT_I4]

Set the elapsed time from the control start.[ms]

If it is omitted, "-1" is assumed to be specified.

<Mode> : [in]Termination mode

Set the termination mode for the robot and force control when the condition is met.

0 : Neither the robot motion nor force control is terminated.

1 : The robot immediately stops. (Halt) Force control is not terminated.

2 : Both the robot motion and force control are terminated.

If it is omitted, 0 is assumed to be specified.

<Timeout> : [in] Timeout period [VT_I4]

Specify the timeout period [ms].

Example

```
caoRobot.Execute "ForceWaitCondition", Array("P0","P1")
```

5.2.28.83. CaoRobot::Execute("ForceSensor") command

Control the force sensor

This method is not available in SlaveMode.

Syntax ForceSensor (<FuncNo>)

< FuncNo> : [in] Function number [VT_I4]

0 : Force sensor reset

Example

```
caoRobot.Execute "ForceSensor", 0
```

5.2.28.84. CaoRobot::Execute("ForceChangeTable") command

Change the force control table which is currently controlled.

This method is not available in SlaveMode.

Syntax ForceChangeTable (< TableNo >)

< Table No > : [in] Table number (1-10) [VT_I4]

Example

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

5.2.28.85. CaoRobot::Execute("GetSrvData") command

Return the servo internal data of the robot axis.

This command corresponds to GETSRVDATA instruction of PacScript language

Syntax GetSrvData (<DataNum>)

< DataNum > : [in] Data number(1,2,4,5,8,17,18,19,20) (VT_I4)

Return value : Servo internal data

1,2,4,5,8: (VT_R4)

17,18,19,20: J type(VT_VARIANT[VT_R8|VT_ARRAY:7 element])

Example

```
--------------------------------------------------------------------------------
        Dim fData As Single
        fData = caoRobot.Execute("GetSrvData", 2)
--------------------------------------------------------------------------------
```

5.2.28.86. CaoRobot::Execute("GetSrvJntData") command

Return the servo internal data of the specified axis.

This command corresponds to GETSRVJNTDATA instruction of PacScript language

Syntax GetSrvJntData (<DataNum>,<Axis>)

< DataNum > : [in] Data number(1,2,4,5,8) (VT_I4)

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

Return value : Servo internal data (VT_R4)

Example

```
--------------------------------------------------------------------------------
        Dim fData As Single
        fData = caoRobot.Execute("GetSrvJntData", 2, 1)
--------------------------------------------------------------------------------
```

5.2.28.87. CaoRobot::Execute("GrvCtrl") command

Configure True/False of Gravity Compensation Control Function.

This command corresponds to GRVCTRL instruction of PacScript language

This method is not available in SlaveMode.

Syntax GrvCtrl (<True/False>)

< True/False >	:	[in] True/False (VT_I4) True (other than 0) or False (0)
Return value	:	None

Example

```
--------------------------------------------------------------------------------
        caoRobot.Execute "GrvCtrl", True
        caoRobot.Execute "GrvCtrl", False
--------------------------------------------------------------------------------
```

5.2.28.88. CaoRobot::Execute("CurLmt") command

Configure the current limiting function and True/False.

This command corresponds to CURLMT instruction of PacScript language.

This method is not available in SlaveMode.

Syntax1 CurLmt (<True>,<Axis>,<Value>)

Syntax2 CurLmt (<False>,<Axis>)

< True/False >	:	[in] True/False (VT_I4) True (other than 0) or False (0)
< Axis >	:	[in] Axis number (VT_R4) 0: All axes(When set to false)
< Value >	:	[in] Setting value(1 to 100[%]) (VT_R4)
Return value	:	None

Example

```
--------------------------------------------------------------------------------
        caoRobot.Execute "GrvCtrl", True
        caoRobot.Execute "CurLmt", Array(True, 1, 10.5)
        caoRobot.Execute "CurLmt", Array(True, 2, 50.3)
        caoRobot.Execute "CurLmt", Array(False, 0)
        caoRobot.Execute "GrvCtrl", False
--------------------------------------------------------------------------------
```

5.2.28.89. CaoRobot::Execute("Zforce") command

Specify the thrust force for current limiting function of the third axis (Z axis) in H Series Robot.

This command corresponds to ZFORCE instruction of PacScript language.

This method is not available in SlaveMode.

Syntax Zforce (<Thrust>)

< Thrust >	:	[in] Thrust force (VT_R4)
Return value	:	None

Example

```
----------------------------------------------------------------------------------------------------------------------------
        caoRobot.Execute "GrvCtrl", True
        caoRobot.Execute "Zforce", 50
        caoRobot.Execute "GrvCtrl", False
----------------------------------------------------------------------------------------------------------------------------
```

5.2.28.90. CaoRobot::Execute("GrvOffset") command

Configure True/False of gravity offset function.

This command corresponds to GRVOFFSET instruction of PacScript language.

This method is not available in SlaveMode.

Syntax GrvOffset (<True/False>)

< True/False >	:	[in] True/False (VT_I4) True (other than 0) or False (0)
Return value	:	None

Example

```
----------------------------------------------------------------------------------------------------------------------------
        caoRobot.Execute "GrvOffset", True
        caoRobot.Execute "GrvOffset", False
----------------------------------------------------------------------------------------------------------------------------
```

5.2.28.91. CaoRobot::Execute("HighPathAccuracy") command

Switch True/False of the high tracing control function.

This command corresponds to HIGHPATHACCURACY instruction of PacScript language.

This method is not available in SlaveMode.

Syntax1 HighPathAccuracy (<True/False>)

< True/False >	:	[in] True/False (VT_I4) True (other than 0) or False (0)
Return value	:	None

Example

```
----------------------------------------------------------------------------------------------------------------------------
        caoRobot.Execute "HighPathAccuracy", True
        caoRobot.Execute "HighPathAccuracy", False
----------------------------------------------------------------------------------------------------------------------------
```

5.2.28.92. CaoRobot::Execute("MotionTimeout") command

Change a timeout setting value of the motion instruction.

This command corresponds to MOTIONTIOMEOUT instruction of PacScript language.

This method is not available in SlaveMode.

Syntax1 MotionTimeout (<True>,<Timeout>)

Syntax2 MotionTimeout (<False>)

< True/False >	:	[in] True/False (VT_I4) True (other than 0) or False (0)
< Timeout >	:	[in] Timeout period(1 to 30000[msec]) (VT_I4)
Return value	:	None

Example

```
--------------------------------------------------------------------------------
        caoRobot.Execute "MotionTimeout", Array(True, 1000)
        caoRobot.Execute "MotionTimeout", False
--------------------------------------------------------------------------------
```

5.2.28.93. CaoRobot::Execute("SingularAvoid") command

Enable or disable singularity avoidance function.

This command corresponds to SINGULARAVOID instruction of PacScript language.

This method is not available in SlaveMode.

Syntax SingularAvoid (<Mode>)

< Mode >	:	[in] Mode (VT_I4) 0: Disable 2: Enable
Return value	:	None

Example

```
--------------------------------------------------------------------------------
        caoRobot.Execute "SingularAvoid", 2
        caoRobot.Move 1, "@0 P2"
        caoRobot.Execute "SingularAvoid", 0
--------------------------------------------------------------------------------
```

5.2.28.94. CaoRobot::Execute("SpeedMode") command

Change the optimal speed setting function.

This command corresponds to SPEEDMODE instruction of PacScript language.

This method is not available in SlaveMode.

Syntax SpeedMode (<Mode>)

< Mode >	:	[in] Mode number (VT_I4) 0: Disable

1: Enable(PTP motion)

2: Enable(CP motion)

3: Enable(PTP and CP motion)

Return value : None

Example

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

5.2.28.95. CaoRobot::Execute("PayLoad") command

Change the setting value of internal load conditions.

This command corresponds to PAYLOAD instruction of PacScript language.

This method is not available in SlaveMode.

Syntax PayLoad (<Payload>[,<Gravity>[,<Inertia>]])

< Payload > : [in] Mass of payload (VT_I4)

True (other than 0) or False (0)

< Gravity > : [in] Payload center of gravity

V type(VT_VARIANT[VT_R8|VT_ARRAY:3 element])

If the argument is omitted, 0 Vector(0,0,0) is assumed to be specified.

< Inertia > : [in] Payload center of gravity inertia

V type(VT_VARIANT[VT_R8|VT_ARRAY:3 element])

If the argument is omitted, 0 Vector(0,0,0) is assumed to be specified.

Return value : None

Example

```
-------------------------------------------------------------------------------------------------------------------------
        caoRobot.Execute "PayLoad", Array(2000, "V(0, 100, 150)" , "V(0, 10, 10)")
-------------------------------------------------------------------------------------------------------------------------
```

5.2.28.96. CaoRobot::Execute("GenerateNonStopPath ") command

This command is exclusive to "the Non-stop motion calculator option."

See "Appendix C. Non-Stop Motion Calculator - Trajectory Generator Command for Non Stop Inspection " in detail.

Syntax PayLoad (<Teaching Points>, <Area Information>, <Teaching Point Number>, <Total Speed Rate>, <Convergence Coefficient>)

< Teaching Points > : [in]Teaching Points [VT_ARRAY]

< Area Information > : [in] Area Information [VT_ARRAY]

< Teaching Point Number > [in] Teaching Point Number [VT_I4]

<Total Speed Rate> [in] Total Speed Rate [VT_R4] 0.0 - 1.0

<Convergence Coefficient> : [in] Convergence Coefficient [VT_R4] 0.0- 1, "0.7" is usual.

Return value : [out]Motion Points [VT_ARRAY]

Example

```
------------------------------------------------------------------------------------------------------------------------
vntMovePos = caoRobot..Execute( "GenerateNonStopPath", Array(vntTeachPos, vntAreaInfo,
Ubound(vntTeachPos) + 1, 100.0 * 0.01, 0.7))
------------------------------------------------------------------------------------------------------------------------
```

5.2.28.97. CaoRobot::Execute("RobInfo") command

Return the robot information

This corresponds to the RobInfo command of PacScript.

Syntax RobInfo (<1Index>)

< lIndex > : Index number (VT_I4)

Return value : [out] Robot information

Index number	Robot information	Data type
0	A unique value assigned to each robot type	Integer type
1	Robot type	String type
2	The total distance of each axis traversed after shipment	Joint type

Example

```
------------------------------------------------------------------------------------------------------------------------
Dim RobotInfo as long
RobotInfo = caoRobot.Execute("RobInfo",0)
------------------------------------------------------------------------------------------------------------------------
```

5.2.28.98. CaoRobot::Execute("SetBaseDef") command

Specify the base definisiont .

Syntax SetBaseDef(<1BaseNo>,<BaseDef>,<1BaseAttribute>)

<1BaseNo> : Base number (VT_I4)

<BaseDef> : Base definition (P-type)
X, Y, Z, RX, RY, RZ

<1BaseAttribute> : At present, this item is not used

Return value : None

Example

```
-----------------------------------------------------------------------------------------------------------------------------
    caoRobot.Execcute “SetBaseDef”, Array(1,”P2”)
    caoRobot.Execute “SetWorkDef”,Array(1,”P(100,200,300,180,0,180)”)
-----------------------------------------------------------------------------------------------------------------------------
```

5.2.28.99. CaoRobot::Execute("SyncTimeStart") command

Declare that multiple robots start and stop their motion simultaneously. Hereafter, this operation is called “synchronous motion”.

Syntax SyncTimeStart()

Argument : none

Return value : none

Example

```
-----------------------------------------------------------------------------------------------------------------------------
      Dim caoRobot0 As CaoRobot
      Dim caoRobot1 As CaoRobot

      Set CaoRobot0 = caoCtrl.AddRobot(“Robot0”,”ID=0”)
      Set CaoRobot1 = caoCtrl.AddRobot(“Robot1”,”ID=1”)

      CaoRobot0.Execute “SyncTimeStart”
                  caoRobot0.Move 1,”P1”     ‘Instruct the master robot to move to P1
                  caoRobot1.Move 1,”P3”     ‘Instruct the slave robot to move to P3
      caoRobot0.Execute “SyncTimeEnd”     ‘Start the synchronous motion of Robot0 and Robot1.
-----------------------------------------------------------------------------------------------------------------------------
```

5.2.28.100. CaoRobot::Execute("SyncTimeEnd") command

Start the synchronous motion with multiple robots.

Syntax SyncTimeEnd(<1Option>)

<1Option> : Motion option (VT_I4)

0: none (This should be “0” if it is omitted.)

1: NEXT

Return value : none

Example

```
-----------------------------------------------------------------------------------------------------------------------------
      Dim caoRobot0 As CaoRobot
      Dim caoRobot1 As CaoRobot

      Set CaoRobot0 = caoCtrl.AddRobot(“Robot0”,”ID=0”)
      Set CaoRobot1 = caoCtrl.AddRobot(“Robot1”,”ID=1”)
```

```
        CaoRobot0.Execute “SyncTimeStart”
                caoRobot0.Move   1,”P1”   ‘Instruct the master robot to move to P1
                caoRobot1.Move   1,”P3”   ‘Instruct the slave robot to move to P3
        caoRobot0.Execute “SyncTimeEnd”,1   ‘Start synchronous motion of Robot0 and Robot1 with Next option.
--------------------------------------------------------------------------------------------------------------------------
```

5.2.28.101. CaoRobot::Execute("SyncMoveStart") command

Declare that multiple robots will work together to complete a task. Hereafter, this operation is called “cooperative motion”. One robot is designated to the leader robot, others are designated to the leader robots. PTP motion is not available.

Syntax SyncMoveStart(<1FollowerRobotID>,<1FollowerRobotID>...)

<1FollowerRobotID>	:	Robot ID of the follower robot (VT_I4|VT_ARRAY)
Return value	:	none

Example

```
--------------------------------------------------------------------------------------------------------------------------
        Dim caoRobot0 As CaoRobot
        Dim caoRobot1 As CaoRobot

        Set CaoRobot0 = caoCtrl.AddRobot(“Robot0”,”ID=0”)
        Set CaoRobot1 = caoCtrl.AddRobot(“Robot1”,”ID=1”)

        CaoRobot0.Execute “SyncMoveStart”,Array(1)
           caoRobot0.Move 2,”J(0,45,90,0,45,0,0,0)” ‘Instruct the leader robot to move to the specified position.
        caoRobot0.Execute “SyncMoveEnd”       ‘Start the cooperative motion of Robot0 and Robot1
--------------------------------------------------------------------------------------------------------------------------
```

5.2.28.102. CaoRobot::Execute("SyncMoveEnd") command

Start the cooperative motion with multiple robots.

Syntax SyncMoveEnd(<1Option>)

<1Option>	:	Motion option (VT_I4)
		0: none (This should be “0” if it is omitted.)
		1: Next
Return value	:	none

Example

```
--------------------------------------------------------------------------------------------------------------------------
      Dim caoRobot0 As CaoRobot
      Dim caoRobot1 As CaoRobot

      Set CaoRobot0 = caoCtrl.AddRobot(“Robot0”,”ID=0”)
      Set CaoRobot1 = caoCtrl.AddRobot(“Robot1”,”ID=1”)

      CaoRobot0.Execute “SyncMoveStart”,Aarray(1)
        caoRobot0.Move 2,”J3” ‘Instruct the leader robot to move to J3.
```

```
caoRobot0.Execute “SyncMoveEnd”,1 ‘Start the cooperative motion of Robot0 and Robot1with Next option.
```

5.2.28.103. CaoRobot::Execute("GetBaseDef") command

Obtain the base definition.

Syntax GetBaseDef(<1BaseNo>)

<1BaseNo>	:	Base number (VT_I4)
Return value	:	Base definition (VT_R8|VT_ARRAY)
		X, Y, Z, RX, RY, RZ, attraibute

Example

```
Dim vVal As Variant
vVal = caoRobot.Execcute (“GetBaseDef”,1)
```

5.2.28.104. CaoRobot::Execute("SetHandIO") command

Set I/O number, values, and range of Hand I/O.

Syntax SetHandIO(<1IONo>,<1value>,<1range>)

<1IONo>	:	The smallest Hand I/O number.(VT_I4)
<1Value>	:	Values to be set. (VT_I4)
<1Range>		Setting range (VT_I4)
Return value	:	None

Example

```
caoRobot.Execcute “SetHandIO”,Array(48,8,4)
```

5.2.28.105. CaoRobot::Execute("GetHandIO") command

Obtain IO number, range, and values of HandI/O.

Syntax GetHandIO(<1IONo>,<1Range>)

<1IONo>	:	The smallest Hand I/O number (VT_I4)
<1Range>	:	Setting range (VT_I4)
Return value	:	Values of Hand IO in the setting range (VT_I4)

Example

```
-----------------------------------------------------------------------------------------------------------------
        Dim lVal As Integer
        lVal = caoRobot.Execcute (“GetHandIO”,Array(48,,4)).
-----------------------------------------------------------------------------------------------------------------
```

5.2.28.106. CaoRobot::Execute("StartServoLog") command

Start servo logging.

Syntax StartServoLog()

Argument	:	none
Return value	:	none

Example

```
-----------------------------------------------------------------------------------------------------------------
        caoRobot.Execcute “StartServoLog”
-----------------------------------------------------------------------------------------------------------------
```

5.2.28.107. CaoRobot::Execute("ClearServoLog") command

Delete obtained servo log data.

Syntax ClearServoLog()

Argument	:	none
Return value	:	none

Example

```
-----------------------------------------------------------------------------------------------------------------
        caoRobot.Execcute “ClearServoLog”
-----------------------------------------------------------------------------------------------------------------
```

5.2.28.108. CaoRobot::Execute("StopServoLog") command

Stop servo logging

Syntax StopServoLog()

Argument	:	none
Return value	:	none

Example

```
-----------------------------------------------------------------------------------------------------------------
        caoRobot.Execcute “StopServoLog
-----------------------------------------------------------------------------------------------------------------
```

5.2.28.109. CaoRobot::Execute("GetCtrlLogMaxTime") command

Get the duration of control logging.

Syntax GetCtrlLogMaxTime ( )

Argument : none

Return value : Logging duration of the control log (VT_I4）

Example

```
Dim lVal As Integer
lVal = caoRobot.Execcute (“GetCtrlLogMaxTime”)
```

5.2.28.110. CaoRobot::Execute("SetCtrlLogMaxTime") command

Set the duration of control logging.

Syntax SetCtrlLogMaxTime (lTime)

Argument : Logging duration to be set (VT_I4)

Return value : none

Example

```
caoRobot.Execcute “SetCtrlLogMaxTime”,10
```

5.2.28.111. CaoRobot::Execute("GetCtrlLogInterval") command

Get the interval of control logs.

Syntax GetCtrlLogInterval ( )

Argument : none

Return value : Logging interval of the control log（VT_I4）

Example

```
Dim lVal As Integer
lVal = caoRobot.Execcute (“GetCtrlLogInterval”)
```

5.2.28.112. CaoRobot::Execute("SetCtrlLogInterval ") command

Set the interval of control logs.

Syntax SetCtrlLogInterval (lTime)

Argument : Logging interval to be set (VT_I4)

Return value : none

Example

```
caoRobot.Execcute “SetCtrlLogInterval”,8
```

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, ...))

3. 1. The command name is passed as a BSTR string to the first argument.
4. 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.140
Priority			
	GetThreadPriority	Get the priority.	P.141
	SetThreadPriority	Set the priority.	P.141

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 values 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, ...))

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

6. 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 gripper 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.144
	UnChuck	Execute unchuck operation.	P.144
	Motor	Turn ON/OFF the motor power.	P.144
	Org	Execute origin return.	P.145
	MoveP	Execute point operation.	P.145
	MoveA	Execute absolute position movement.	P.146
	MoveR	Execute relative position movement.	P.146
	MoveAH	Execute hold operation in acceleration/deceleration absolute position movement.	P.146
	MoveRH	Execute hold operation in acceleration/deceleration relative position movement.	P.147
	MoveH	Execute hold operation in constant speed movement.	P.147
	MoveZH	Execute hold operation in zone-specific constant speed movement.	P.148
	Stop	Stop the operation.	P.148
	CurPos	Get the current position.	P.149
	GetPoint	Get the point data element.	P.149
	get_EmgState	Get the emergency stop input status.	P.149
	get_ZonState	Get the ZON signal status.	P.150
	get_OrgState	Get the origin return status.	P.150
	get_HoldState	Get the hold status.	P.151
	get_InposState	Get the INPOS status.	P.151
	get_Error	Get the electric gripper error information.	P.152
	get_BusyState	Get the operation status.	P.152
	get_MotorState	Get the motor power status.	P.152

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 gripper 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 gripper. While the electric gripper is in an emergency stop status, executing the motor-ON command does not turn ON the motor. While the electric gripper is already in motor-ON status, executing the motor-ON command has no effect and the electric gripper'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 gripper 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 gripper 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 gripper 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 gripper 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 gripper 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	[VT_I4]
		0: Open direction		
		Other than 0: Close direction		
Return value	:	None		

Execute the constant speed movement and hold operation of the electric gripper 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 gripper 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 gripper.

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 gripper.

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 gripper 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 gripper 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 gripper.

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 gripper.

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 gripper.

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 gripper.

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 gripper.

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 gripper.

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	√	-
@TSR_BUSY_STATUS	VT_BOOL	true = Supervisory tasks running, false = Supervisory tasks 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	√	-
@MAKER_NAME	VT_BSTR	"DENSO CORPORATION"	√	-
@TYPE	VT_BSTR	"RC8 Controller"	√	-
@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[5]               "[<Pass>]  [<Variable type>]<Index>  [<ExJnt>]"
  │                           or
  │                           "[<Pass>]  [<Variable type>](<Element 1>,<Element 2>,...)
  ├─                          [<ExJnt>]"
  └─ VT_R4|VT_ARRAY[6]        <Raw value> = (<Element 1:VT_R4>,<Element 2:VT_R4>,...)[7]
     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>

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

[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.

[7] 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")[8]
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)

[8] 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)

<table>
<tr><td>2nd argument: string
3rd argument: string</td><td>.Execute "APPROACH", Array(2, "J(60.5,30.3,400,90)", "@100 70", "NEXT")</td></tr>
<tr><td>2nd argument: VARIANT array
(Raw value,
variable type specified by string)
3rd argument: VARIANT array
(Variable type and pass type specified by string)</td><td>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")</td></tr>
<tr><td>2nd argument: VARIANT array
(Raw value,
variable type specified by string)
3rd argument: VARIANT array
(Variable type and pass type specified by value)</td><td>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")</td></tr>
</table>

[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

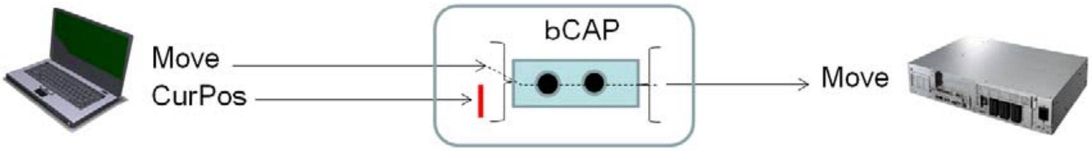
```
Dim vJ as Variant
vJ=CaoCtrl.Variables("J100").Value   ‘VT_R4|VT_ARRAY
Robot.Move 1, vJ                     ' Wrong!! = MOVE P, P(<j1>,<j2>,<j3>,…)
```

The correct code is as follows.

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

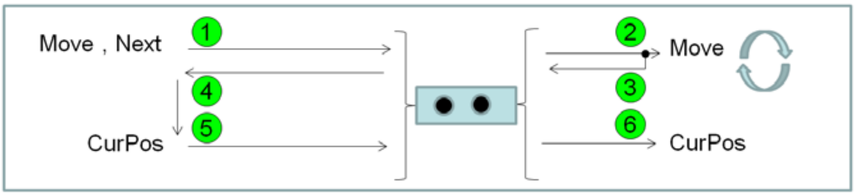
Appendix C. Simultaneous Command Issuance to RC8 controller

When the RC8 controller connects with devices through RC8 provider, clients (PCs) and a server (RC8 controller) communicates with b-CAP protocol in each connection (by AddController unit). At that time, the commands transmitted from the client are serialized by the server. Therefore, you need to keep in mind that the following points for the programming on the client side, if you intend to write program that issues multiple commands simultaneously (e.g: monitoring the robot position during robot motion).



(1) For single-thread

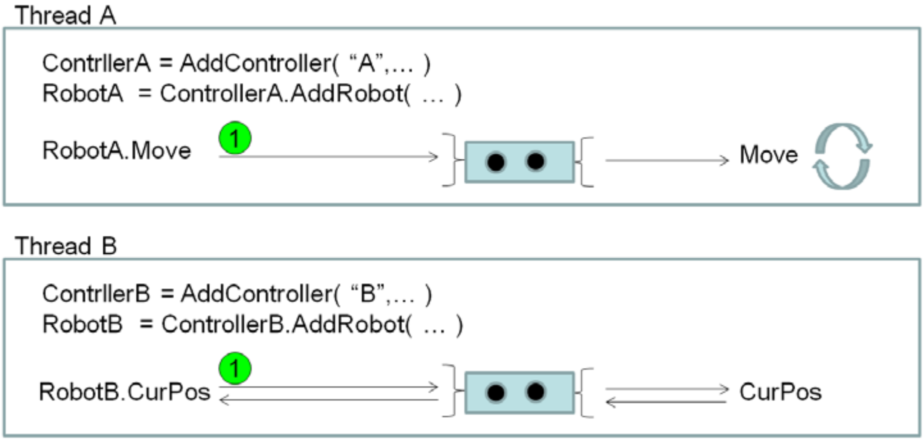
Desynchronize Move command by means of Next option, and then insert CurPos command during the motion termination wait loop.



(2) For multi-thread

Create respective threads for the motion execution and the current position monitoring.

Execute CaoWorkspace::AddController for each thread to establish respective connections.


Thread A
ContrllerA = AddController(“A”,…)
RobotA = ControllerA.AddRobot(…)
RobotA.Move
1
Move
Thread B
ContrllerB = AddController(“B”,…)
RobotB = ControllerB.AddRobot(…)
RobotB.CurPos
1
CurPos

In this example, two threads are used; Thread A executes Move command whereas Thread B executes CurPos command that gets robot’s current position. Because Move and CurPos run in the different threads, these can be executed at the same time. Note that you should avoid using the main thread as Thread A, because the main thread controls GUI. If the main thread is occupied by the motion command, drawing is impeded. Creating a new thread for Theread A is recommended.

Appendix D. Non-Stop Motion Calculator - Trajectory Generator Command for Non Stop Inspection

To use the Non-stop motion calculator option, you need to install ORiN2 SDK, and also need to input "Non-stop motion calculator" license information in RC8. Please refer "Displaying and Adding/Deleting Function Extension Screen" in DENSO ROBOT USERS MANUALS, in which indicate the method how to add the license in RC8. Please see "ORiN2 RC8 Provider "Non-Stop motion calculator" Option User's Guide."

Appendix D.1. Parameters

Following is details of <Position > parameter and <Area> parameter of GenerateNonStopPath command.

```
<Teaching Positions:VT_VARIANT | VT_ARRAY> =
    <X:VT_R4>,
    <Y:VT_R4>,
    < Z:VT_R4>,
    < RX:VT_R4>,
    < RY:VT_R4>,
    < RZ:VT_R4>,
    < Fig:VT_I4>,
    < J7:VT_R4>,
    < J8:VT_R4>,
    < Motion Speed:VT_R4>=(0.0～1.0),
    < Motion Pattern:VT_I4> = (0:@P,  1:@0,  2:@E),
    < Tool Number : VT_I4>

< Area Information:VT_R4 | VT_ARRAY > =
    < Area Size X:VT_R4>,
    < Area Size Y:VT_R4>,
    < Area Size Z:VT_R4>,
    < Area Angle:VT_R4>,
    < Area Size  J7:VT_R4>,
    < Area Size  J8:VT_R4>
```

Appendix D.2. Error Codes

The error code of the "GenerateNonStopPath" command defined in the provider is "0x8150015E." The custom error codes for the command are shown in the below table. The custom code is shown in the detail of the error.

Number (original number)	Contents
0x2000****	Conversion Error from Position to Joint (Teaching Data)
0x2100****	Software limit Error (Teaching Data)
0x2200****	Out of Speed Rate Range
0x2300****	Conversion Error from Position to Joint (Revised Data)
0x2400****	Software limit Error (Revised Data)
0x2500****	Revision operation convergence is impossible
0x2600****	Too near teaching points (Position and Posture)
0x2A00****	Joint Flag must be set as "limited rotation".
0x2F000000	Out of total speed rate range
0x2F010000	Out of Coefficients Range
0x2F020000	Trajectory Pass Generation fail
0x2F030000	Low Memory
0x2F0A0000	Out of Data Number

Error Code: 0x2000**** - 0x2800****

The symbol of "**" indicates the error occurred position number. The value is "the error occurred position number + 1".

Error Code: 0x2A00****

The symbol of "*" indicates the number of the axis which is set as "unlimited rotation".

Appendix D.3. Restrictions

Restrictions of GenerateNonStopPath Command are as follows:

- Max. Number of Teaching Points = 200
- Available for 6-axis robot only
- Area size for additional axis should be assigned in [degree] (for rotational axis) or [mm] (for linear axis) according to the axis setting.
- Unavailable for Unlimited rotation of the extra-joint

Appendix D.4. Sample Program

The following sample program is described in CaoScript. The following sample teaching data is for VS-6577G-BA robot. Assign appropriate IP Address for the target controller. This sample assumes the target controller IP addres is 10.6.235.72.

Sample	NonStopPath.vbs

```
' Generate NonStopPath
Const RC_ADDRESS = "10.6.235.72"

Sub Main
        Dim rc
        Dim vntTeachPos()
        Dim vntAreaInfo()
        Dim vntMovePos
        Dim vntParam
        Dim lIndex

        dbg.ClearLog

        set rc = cao.AddController("RC", "CaoProv.DENSO.RC8", "", "server=" & RC_ADDRESS)
        set rob = rc.AddRobot("Robot")

        ' GenerateNonStopPath Command Parameter（Pos, Area, Size, SpdRate, Coef）
        ' Pos : TeachPoint Data（x, y, z, rx, ry, rz, fig, J7, J8, SpdRate, attr, ToolNum）
        redim vntTeachPos(7)

        vntTeachPos(0) = Array(300.0, 100.0, 600.0, 180.0, 0.0, 180.0, 5, 0.0, 0.0, 100 * 0.01, 1, 0)
        vntTeachPos(1) = Array(300.0, 91.0, 600.0, 180.0, 0.0, -180.0, 5, 0.0, 0.0, 100 * 0.01, 0, 0)
        vntTeachPos(2) = Array(310.0, 30.0, 600.0, 180.0, 0.0, -180.0, 5, 0.0, 0.0, 100 * 0.01, 1, 0)
        vntTeachPos(3) = Array(315.5, 24.5, 600.0, 180.0, 0.0, -180.0, 5, 0.0, 0.0, 100 * 0.01, 0, 0)
        vntTeachPos(4) = Array(300.0, 10.0, 600.0, 180.0, 0.0, 173.0, 5, 0.0, 0.0, 100 * 0.01, 1, 0)
        vntTeachPos(5) = Array(300.0, 10.0, 600.0, 180.0, 0.0, 176.0, 5, 0.0, 0.0, 100 * 0.01, 0, 0)
        vntTeachPos(6) = Array(300.0, 10.0, 600.0, 180.0, 0.0, 171.0, 5, 0.0, 0.0, 100 * 0.01, 0, 0)
        vntTeachPos(7) = Array(300.0, 10.0, 600.0, 180.0, 0.0, -180.0, 5, 0.0, 0.0, 100 * 0.01, 1, 0)

        ' Area : Area Info（x, y, z, angle, J7, J8）
        redim vntAreaInfo(7)

        vntAreaInfo(0) = Array(4,4,4,4,0,0)
        vntAreaInfo(1) = Array(4,4,4,4,0,0)
        vntAreaInfo(2) = Array(4,4,4,4,0,0)
        vntAreaInfo(3) = Array(4,4,4,4,0,0)
        vntAreaInfo(4) = Array(4,4,4,4,0,0)
        vntAreaInfo(5) = Array(4,4,4,4,0,0)
        vntAreaInfo(6) = Array(4,4,4,4,0,0)
        vntAreaInfo(7) = Array(4,4,4,4,0,0)

        dbg.Output "Teach Points"
        for lIndex = 0 to Ubound(vntTeachPos)
                dbg.Output lIndex & "：" & dat.BstrFromVariant(vntTeachPos(lIndex))
        next

        ' ---------------------------------------
        ' Generate NonStopPath
        ' ---------------------------------------
        vntMovePos = rob.Execute("GenerateNonStopPath", Array(vntTeachPos, vntAreaInfo,
Ubound(vntTeachPos) + 1, 100.0 * 0.01, 0.7))
```

```
        dbg.Output "Move Points"
        for lIndex = 0 to Ubound(vntMovePos)
            dbg.Output lIndex & " : " & dat.BstrFromVariant(vntMovePos(lIndex))
        next
    End Sub
```

Appendix D.5. Workaround at the time of the Adjustment Failure (Error Code [original number]: 0x8150015E[0x2300****])

The GenerateNonStopPath command fails when the revised angle is out of maximum adjustment angle range. If the failure occurs and error code [original number] 0x8150015E[0x2300xxxx] (xxxx represents teaching position number) is shown, change the teaching position, or adjust parameters for the selected Adjustment Method as shown in below.

Adjust the following parameters in [F2 Arm] - [F6 Aux] - [F1 Config].

Parameter	Outline	Input Limitation
FIGCHECK.MAX_DISPLACEMENT_dJ1	Offset of Maximum Adjustment Angle for 1st axis (deg)	-21474.8 ～ 21474.8
FIGCHECK.MAX_DISPLACEMENT_dJ2	Offset of Maximum Adjustment Angle for 2nd axis (deg)	-21474.8 ～ 21474.8
FIGCHECK.MAX_DISPLACEMENT_dJ3	Offset of Maximum Adjustment Angle for 3rd axis (deg)	-21474.8 ～ 21474.8
FIGCHECK.MAX_DISPLACEMENT_dJ4	Offset of Maximum Adjustment Angle for 4th axis (deg)	-21474.8 ～ 21474.8
FIGCHECK.MAX_DISPLACEMENT_dJ5	Offset of Maximum Adjustment Angle for 5th axis (deg)	-21474.8 ～ 21474.8
FIGCHECK.MAX_DISPLACEMENT_dJ6	Offset of Maximum Adjustment Angle for 6th axis (deg)	-21474.8 ～ 21474.8
FIGCHECK.MAX_DISPLACEMENT_dJ7	Offset of Maximum Adjustment Angle for 7th axis (deg)	-21474.8 ～ 21474.8
FIGCHECK.MAX_DISPLACEMENT_dJ8	Offset of Maximum Adjustment Angle for 8th axis (deg)	-21474.8 ～ 21474.8