RC9 Provider

for DENSO Robot RC9

Version 1.1.0

User's Guide

May 20, 2023

[Remarks]

[Revision History]

Version	Date	Content
1.0.0	2020-07-09	First edition.
	2020-11-09	Added Appendix A (POSEDATA Type Definition)
	2021-02-11	Fixed typos in “3. AddController Command Arguments”
	2021-05-11	Added “4. Clear the STO State”
1.0.1	2021-12-14	Modify License Check
	2022-05-16	Fixed typos in “5.3.3. Task class”
1.1.0	2023-03-20	Added commands for RC9 Safety Motion specification Added commands for COBOTTA PRO Added commands for OnRobot hand

[Hardware]

Model	Version	Notes
RC9	1.0.0～	

[Command]

Model	Version	Notes

Contents

1. Introduction....11
2. Set the activation authority of the robot controller 12
3. AddController Command Arguments 13
4. Clear the STO state (Safety state) 14
5. Command Reference 15
5.1. List of commands....15
5.2. Methods and properties 16
5.2.1. CaoWorkspace::AddController method 16
5.2.1.1. When you establish multiple connections with RC9 controller 18
5.2.2. CaoController::AddFile method 19
5.2.3. CaoController::AddRobot method....20
5.2.4. CaoController::AddTask method....20
5.2.5. CaoController::AddVariable method 20
5.2.6. CaoController::get_Name property....21
5.2.7. CaoController::get_FileNames property 22
5.2.8. CaoController::get_TaskNames property....22
5.2.9. CaoController::get_VariableNames property 22
5.2.10. CaoController::Execute method 23
5.2.10.1. CaoController::Execute("ClearError") command....24
5.2.10.2. CaoController::Execute("GetErrorDescription") command....25
5.2.10.3. CaoController::Execute("KillAll") command....25
5.2.10.4. CaoController::Execute("KillAllTsr") command....25
5.2.10.5. CaoController::Execute("RunAllTsr") command....26
5.2.10.6. CaoController::Execute("SuspendAll") command 26
5.2.10.7. CaoController::Execute("StepStopAll") command....27
5.2.10.8. CaoController::Execute("ContinueStartAll") command....27
5.2.10.9. CaoController::Execute("GetErrorLogCount") command 27
5.2.10.10. CaoController::Execute("GetErrorLog") command....28
5.2.10.11. CaoController::Execute("GetOprLogCount") command 29
5.2.10.12. CaoController::Execute("GetOprLog") command....29
5.2.10.13. CaoController::Execute(“GetPublicValue”) command 30
5.2.10.14. CaoController::Execute(“SetPublicValue”) command 32
5.2.10.15. CaoController::Execute("SysState") command 34
5.2.10.16. CaoController::Execute("SysInfo") command 34
5.2.10.17. CaoController::Execute(“SetAllDummyIO”) command 36
5.2.10.18. CaoController::Execute(“GetCurErrorCount”) command 36

5.2.10.19. CaoController::Execute(“GetCurErrorInfo”) command 37
5.2.10.20. CaoController::Execute(“StoState”) command 37
5.2.10.21. CaoController::Execute(“ResetStoState”) command 37
5.2.10.22. CaoController::Execute("GetSlaveButtonState") command 38
5.2.10.23. CaoController::Execute("ClearSlaveButtonState") command 38
5.2.10.24. CaoController::Execute("ManualReset") command 39
5.2.10.25. CaoController::Execute("RunBlockProgram") command 39
5.2.10.26. CaoController::Execute("GetEbpState") command 39
5.2.11. CaoFile::AddFile method 40
5.2.12. CaoFile::AddVariable method 40
5.2.13. CaoFile::get_VariableNames property 41
5.2.14. CaoFile::get_FileNames property 41
5.2.15. CaoFile::get_Size property 41
5.2.16. CaoFile::get_Value property 41
5.2.17. CaoFile::put_Value property 41
5.2.18. CaoRobot::Accelerate method 41
5.2.19. CaoRobot::AddVariable method 42
5.2.20. CaoRobot::get_VariableNames property 42
5.2.21. CaoRobot::Halt method 42
5.2.22. CaoRobot::Change method 42
5.2.23. CaoRobot::Drive method 43
5.2.24. CaoRobot::Move method 43
5.2.25. CaoRobot::Rotate method 47
5.2.26. CaoRobot::Speed method 48
5.2.27. CaoRobot::Execute method 49
5.2.27.1. CaoRobot::Execute("TMul") command 54
5.2.27.2. CaoRobot::Execute("TInv") command 54
5.2.27.3. CaoRobot::Execute("TNorm") command 55
5.2.27.4. CaoRobot::Execute("J2T") command 55
5.2.27.5. CaoRobot::Execute("T2J") command 56
5.2.27.6. CaoRobot::Execute("J2P") command 56
5.2.27.7. CaoRobot::Execute("P2J") command 56
5.2.27.8. CaoRobot::Execute("T2P") command 57
5.2.27.9. CaoRobot::Execute("P2T") command 57
5.2.27.10. CaoRobot::Execute("Dev") command 58
5.2.27.11. CaoRobot::Execute("DevH") command 58
5.2.27.12. CaoRobot::Execute("OutRange") command 58

5.2.27.13. CaoRobot::Execute("MPS") command....59
5.2.27.14. CaoRobot::Execute("RPM") command....59
5.2.27.15. CaoRobot::Execute("DPS") command....60
5.2.27.16. CaoRobot::Execute("CurPos") command....60
5.2.27.17. CaoRobot::Execute("DestPos") command....61
5.2.27.18. CaoRobot::Execute ("CurPosEx") command....61
5.2.27.19. CaoRobot::Execute("DestPosEx") command....61
5.2.27.20. CaoRobot::Execute("CurJnt") command....62
5.2.27.21. CaoRobot::Execute("DestJnt") command....62
5.2.27.22. CaoRobot::Execute("CurJntEx") command....62
5.2.27.23. CaoRobot::Execute("DestJntEx") command....63
5.2.27.24. CaoRobot::Execute("CurTrn") command....63
5.2.27.25. CaoRobot::Execute("DestTrn") command....64
5.2.27.26. CaoRobot::Execute("CurTrnEx") command....64
5.2.27.27. CaoRobot::Execute("DestTrnEx") command....65
5.2.27.28. CaoRobot::Execute("CurFig") command....65
5.2.27.29. CaoRobot::Execute("CurSpd") command....65
5.2.27.30. CaoRobot::Execute("CurAcc") command....66
5.2.27.31. CaoRobot::Execute("CurDec") command....66
5.2.27.32. CaoRobot::Execute("CurJSpd") command....66
5.2.27.33. CaoRobot::Execute("CurJAcc") command....67
5.2.27.34. CaoRobot::Execute("CurJDec") command....67
5.2.27.35. CaoRobot::Execute("CurExtSpd") command....67
5.2.27.36. CaoRobot::Execute("CurExtAcc") command....68
5.2.27.37. CaoRobot::Execute("CurExtDec") command....68
5.2.27.38. CaoRobot::Execute("StartLog") command....68
5.2.27.39. CaoRobot::Execute("StopLog") command....69
5.2.27.40. CaoRobot::Execute("ClearLog") command....69
5.2.27.41. CaoRobot::Execute("Motor") command....69
5.2.27.42. CaoRobot::Execute("ExtSpeed") command....70
5.2.27.43. CaoRobot::Execute("TakeArm") command....70
5.2.27.44. CaoRobot::Execute("GiveArm") command....71
5.2.27.45. CaoRobot::Execute("Draw") command....72
5.2.27.46. CaoRobot::Execute("Approach") command....72
5.2.27.47. CaoRobot::Execute("Depart") command....73
5.2.27.48. CaoRobot::Execute("DriveEx") command....74
5.2.27.49. CaoRobot::Execute("DriveAEx") command....75

5.2.27.50. CaoRobot::Execute("RotateH") command 75
5.2.27.51. CaoRobot::Execute("Arrive") command 76
5.2.27.52. CaoRobot::Execute("MotionSkip") command 76
5.2.27.53. CaoRobot::Execute("MotionComplete") command 77
5.2.27.54. CaoRobot::Execute("CurTool") command 78
5.2.27.55. CaoRobot::Execute("GetToolDef") command 78
5.2.27.56. CaoRobot::Execute("SetToolDef") command 78
5.2.27.57. CaoRobot::Execute("CurWork") command 79
5.2.27.58. CaoRobot::Execute("GetWorkDef") command 79
5.2.27.59. CaoRobot::Execute("SetWorkDef") command 79
5.2.27.60. CaoRobot::Execute("WorkAttribute") command 80
5.2.27.61. CaoRobot::Execute("GetAreaDef") command 80
5.2.27.62. CaoRobot::Execute("SetAreaDef") command 80
5.2.27.63. CaoRobot::Execute("SetArea") command 81
5.2.27.64. CaoRobot::Execute("ResetArea") command 81
5.2.27.65. CaoRobot::Execute("AreaSize") command 82
5.2.27.66. CaoRobot::Execute("GetAreaEnabled") command 82
5.2.27.67. CaoRobot::Execute("SetAreaEnabled") command 82
5.2.27.68. CaoRobot::Execute("GetRobotTypeName") command 83
5.2.27.69. CaoRobot::Execute("CrtMotionAllow") command 83
5.2.27.70. CaoRobot::Execute("EncMotionAllow") command 84
5.2.27.71. CaoRobot::Execute("EncMotionAllowJnt") command 84
5.2.27.72. CaoRobot::Execute("ErAlw") command 85
5.2.27.73. CaoRobot::Execute("GetSrvData") command 85
5.2.27.74. CaoRobot::Execute("GetSrvJntData") command 86
5.2.27.75. CaoRobot::Execute("GrvCtrl") command 87
5.2.27.76. CaoRobot::Execute("HighPathAccuracy") command 87
5.2.27.77. CaoRobot::Execute("MotionTimeout") command 87
5.2.27.78. CaoRobot::Execute("SingularAvoid") command 88
5.2.27.79. CaoRobot::Execute("SpeedMode") command 88
5.2.27.80. CaoRobot::Execute("PayLoad") command 89
5.2.27.81. CaoRobot::Execute("GenerateNonStopPath ") command 89
5.2.27.82. CaoRobot::Execute("RobInfo") command 90
5.2.27.83. CaoRobot::Execute("SetBaseDef") command 91
5.2.27.84. CaoRobot::Execute("GetBaseDef") command 91
5.2.27.85. CaoRobot::Execute("SetHandIO") command 91
5.2.27.86. CaoRobot::Execute("GetHandIO") command 92

5.2.27.87. CaoRobot::Execute("StartServoLog") command....92
5.2.27.88. CaoRobot::Execute("ClearServoLog") command....92
5.2.27.89. CaoRobot::Execute("StopServoLog") command....93
5.2.27.90. CaoRobot::Execute("GetCtrlLogMaxTime") command....93
5.2.27.91. CaoRobot::Execute("SetCtrlLogMaxTime") command....93
5.2.27.92. CaoRobot::Execute("GetCtrlLogInterval") command....93
5.2.27.93. CaoRobot::Execute("SetCtrlLogInterval ") command....94
5.2.27.94. CaoRobot::Execute("GetPluralServoData ") command....94
5.2.27.95. CaoRobot::Execute("GetLogCount") command....95
5.2.27.96. CaoRobot::Execute("GetLogRecord") command....95
5.2.27.97. CaoRobot::Execute("MoveMasterPos") command....96
5.2.27.98. CaoRobot::Execute("GetHandAnalogInput") command....96
5.2.27.99. CaoRobot::Execute("SafetyInfo") command....96
5.2.27.100. CaoRobot::Execute("GetSafetyPayload") command....98
5.2.27.101. CaoRobot::Execute("SetSlowSpdChk") command....99
5.2.27.102. CaoRobot::Execute("ChangeScene") command....99
5.2.27.103. CaoRobot::Execute("CurScene") command....99
5.2.27.104. CaoRobot::Execute("CurSubScene") command....100
5.2.27.105. CaoRobot::Execute("SceneInfo") command....100
5.2.28. CaoRobot::Execute method (for OnRobot hand)....101
5.2.28.1. CaoRobot::Execute("HandGrip") command....104
5.2.28.2. CaoRobot::Execute("HandRelease") command....104
5.2.28.3. CaoRobot::Execute("CBGetIO") command....105
5.2.28.4. CaoRobot::Execute("RgxIsConn") command....105
5.2.28.5. CaoRobot::Execute("RgxIsRg2") command....105
5.2.28.6. CaoRobot::Execute("RgxIsRg6") command....106
5.2.28.7. CaoRobot::Execute("RgxIsBusy") command....106
5.2.28.8. CaoRobot::Execute("RgxIsGrip") command....107
5.2.28.9. CaoRobot::Execute("RgxIsSafetyOn") command....107
5.2.28.10. CaoRobot::Execute("RgxGetWidth") command....107
5.2.28.11. CaoRobot::Execute("RgxGetDepth") command....108
5.2.28.12. CaoRobot::Execute("RgxGetRelativeDepth") command....108
5.2.28.13. CaoRobot::Execute("RgxSetFTOffset") command....108
5.2.28.14. CaoRobot::Execute("RgxMove") command....108
5.2.28.15. CaoRobot::Execute("RgxStop") command....109
5.2.28.16. CaoRobot::Execute("RgxResetPower") command....109
5.2.28.17. CaoRobot::Execute("VgxIsConn") command....109

5.2.28.18. CaoRobot::Execute("VgxIsVG10") command 110
5.2.28.19. CaoRobot::Execute("VgxIsVgc10") command 110
5.2.28.20. CaoRobot::Execute("VgxGetVacA") command 111
5.2.28.21. CaoRobot::Execute("VgxGetVacB") command 111
5.2.28.22. CaoRobot::Execute("VgxGetLimit") command 111
5.2.28.23. CaoRobot::Execute("VgxSetLimit") command 111
5.2.28.24. CaoRobot::Execute("VgxGrip") command 112
5.2.28.25. CaoRobot::Execute("VgxIdle") command 112
5.2.28.26. CaoRobot::Execute("VgxRelease") command 113
5.2.28.27. CaoRobot::Execute("SgIsConn") command 113
5.2.28.28. CaoRobot::Execute("SgIsBusy") command 113
5.2.28.29. CaoRobot::Execute("SgIsInitialized") command 114
5.2.28.30. CaoRobot::Execute("SgGetOpenMin") command 114
5.2.28.31. CaoRobot::Execute("SgGetOpenMax") command 114
5.2.28.32. CaoRobot::Execute("SgHasError") command 115
5.2.28.33. CaoRobot::Execute("SgGetDepth") command 115
5.2.28.34. CaoRobot::Execute("SgGetRelativeDepth") command 115
5.2.28.35. CaoRobot::Execute("SgGetSiliconDepth") command 116
5.2.28.36. CaoRobot::Execute("SgGetWidth") command 116
5.2.28.37. CaoRobot::Execute("SgHome") command 116
5.2.28.38. CaoRobot::Execute("SgInit") command 117
5.2.28.39. CaoRobot::Execute("SgGrip") command 117
5.2.28.40. CaoRobot::Execute("SgGentleGrip") command 117
5.2.28.41. CaoRobot::Execute("SgRelease") command 118
5.2.28.42. CaoRobot::Execute("SgStop") command 118
5.2.28.43. CaoRobot::Execute("TwofgIsConn") command 119
5.2.28.44. CaoRobot::Execute("TwofgIsBusy") command 119
5.2.28.45. CaoRobot::Execute("TwofgIsGrip") command 119
5.2.28.46. CaoRobot::Execute("TwofgGetErrorCode") command 120
5.2.28.47. CaoRobot::Execute("TwofgGetWidth") command 120
5.2.28.48. CaoRobot::Execute("TwofgGetMinWidth") command 120
5.2.28.49. CaoRobot::Execute("TwofgGetMaxWidth") command 121
5.2.28.50. CaoRobot::Execute("TwofgGetForce") command 121
5.2.28.51. CaoRobot::Execute("TwofgGetOrientation") command 121
5.2.28.52. CaoRobot::Execute("TwofgGetFingerLength") command 122
5.2.28.53. CaoRobot::Execute("TwofgGetFTWidth") command 122
5.2.28.54. CaoRobot::Execute("TwofgGetMode") command 122

5.2.28.55. CaoRobot::Execute("TwofgSetOrientation") command 123
5.2.28.56. CaoRobot::Execute("TwofgSetFingerLength") command 123
5.2.28.57. CaoRobot::Execute("TwofgSetFTWidth") command 123
5.2.28.58. CaoRobot::Execute("TwofgSwitchToExternal") command 123
5.2.28.59. CaoRobot::Execute("TwofgSwitchToInternal") command....124
5.2.28.60. CaoRobot::Execute("TwofgGrip") command 124
5.2.28.61. CaoRobot::Execute("TwofgRelease") command....125
5.2.28.62. CaoRobot::Execute("TwofgStop") command....125
5.2.28.63. CaoRobot::Execute("TfgIsConn") command....125
5.2.28.64. CaoRobot::Execute("TfgIsBusy") command 126
5.2.28.65. CaoRobot::Execute("TfgIsGrip") command....126
5.2.28.66. CaoRobot::Execute("TfgIsForceGrip") command 126
5.2.28.67. CaoRobot::Execute("TfgIsCalibrationValid") command....127
5.2.28.68. CaoRobot::Execute("TfgHasError") command....127
5.2.28.69. CaoRobot::Execute("TfgHasGeneralError") command 127
5.2.28.70. CaoRobot::Execute("TfgHasSafetyDCError") command....128
5.2.28.71. CaoRobot::Execute("TfgGetDiameterFT") command 128
5.2.28.72. CaoRobot::Execute("TfgGetDiameterRaw") command....128
5.2.28.73. CaoRobot::Execute("TfgGetDiameterMin") command 129
5.2.28.74. CaoRobot::Execute("TfgGetDiameterMax") command 129
5.2.28.75. CaoRobot::Execute("TfgSetFingerPos") command 129
5.2.28.76. CaoRobot::Execute("TfgSetFingerLength") command....130
5.2.28.77. CaoRobot::Execute("TfgSetFTOffset") command....130
5.2.28.78. CaoRobot::Execute("TfgMove") command 130
5.2.28.79. CaoRobot::Execute("TfgGrip") command 131
5.2.28.80. CaoRobot::Execute("TfgFlexGrip") command....131
5.2.28.81. CaoRobot::Execute("TfgStop") command....132
5.2.29. CaoTask::AddVariable method 132
5.2.30. CaoTask::get_VariableNames property 132
5.2.31. CaoTask::Start method 132
5.2.32. CaoTask::Stop method 132
5.2.33. CaoTask::Execute method....133
5.2.33.1. CaoTask::Execute("GetStatus") command....134
5.2.33.2. CaoTask::Execute("GetThreadPriority") command 134
5.2.33.3. CaoTask::Execute("SetThreadPriority") command....134
5.2.34. CaoVariable::get_Value property....135
5.2.35. CaoVariable::put_Value property....135

5.3. Variable list....135
5.3.1. Controller class....135
5.3.2. Robot class....140
5.3.3. Task class....142
5.3.4. File class....143
5.4. Event list....144
Appendix A. POSEDATA Type Definition....146

1. Introduction

With ORiN2 SDK Ver 2.1.51 or later, RC9 provider is set up by an installer.

The usage of RC9 provider is the same as RC8 provider.

This document describes only the differences between RC9 provider and RC8 provider.

・The setting of Executable Token is different.

・Argument of AddController command is different.

・In order to turn on the motor, the robot controller needs to clear the STO state (Safety state).

・Commands is different.

2. Set the activation authority of the robot controller

Follow the steps below to set the activation authority.

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

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

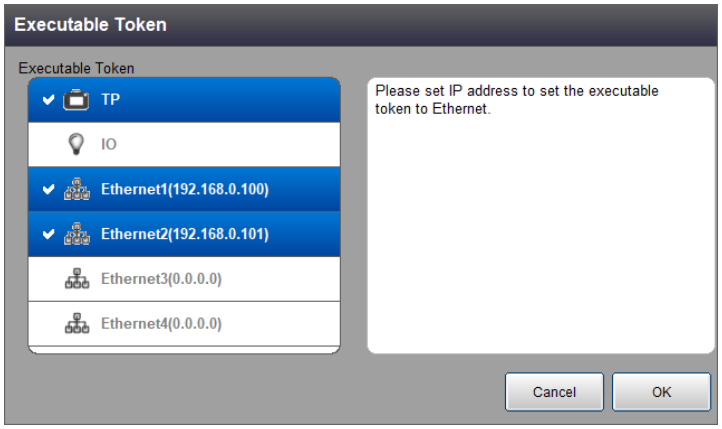
To use Ethernet, select the teach pendant's [F6 Setup] menu -> [F5 Communication and Token] -> [F1 Executable Token] and set the activation authority to Ethernet. In the settings below, TP, Ethernet1(192.168.0.100) and Ethernet2(192.168.0.101) have acquired the activation authority.

* RC9's activation authority is no longer set to ANY that RC8 had, and you can select multiple devices instead.

[F6] ⇒

[F5] ⇒

[F1] ⇒



3. AddController Command Arguments

AddController command arguments are different from RC8.

Connection to Controller

RC9

```
Set g_ctrl = g_eng.Workspaces(0).AddController("Controller", "caoProv.DENSO.RC9", "",
"Server=192.168.0.1")
```

RC8

```
Set g_ctrl = g_eng.Workspaces(0).AddController("Controller", "caoProv.DENSO.RC8", "",
"Server=192.168.0.1")
```

Connection to Project

RC9

```
Set g_ctrl = g_eng.Workspaces(0).AddController("Controller", "caoProv.DENSO.RC9", "",
"RCPJ=C:\test\$Current\default.rcpj")
```

RC8

```
Set g_ctrl = g_eng.Workspaces(0).AddController("Controller", "caoProv.DENSO.RC8", "",
"WPJ=C:\test\test.WPJ")
```

4. Clear the STO state (Safety state)

In order to turn on the motor, the robot controller needs to clear the STO state (safety state). See RC9 User Manuals for more information on STO states and how to release them. You can also clear the STO state by using "CaoController::Execute(“ResetStoState”) command" and "CaoController::Execute("ManualReset") command".

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.16
CaoController			
	AddFile	Connect to a file or folder (PAC, system file).	P.19
	AddRobot	Connect the robot.	P.20
	AddTask	Connect the task (PAC).	P.20
	AddVariable	Connect the user/system variable.	P.20
	get_Name	Get the controller name.	P.21
	get_FileNames	Get a list of file names.	P.22
	get_TaskNames	Get a list of tasks (PAC).	P.22
	get_VariableNames	Get a list of user/system variables.	P.22
	Execute	Execute a command of the controller class.	P.23
CaoFile			
	AddFile	Connect a PAC file.	P.34
	AddVariable	Connect a system variable of files.	P.40
	get_VariableNames	Get a list of system variable names.	P.41
	get_FileNames	Get a list of files.	P.41
	get_Size	Get the size of a file.	P.41
	get_Value	Get the value of a file.	P.41
	put_Value	Rewrite the value of a file.	P.41
CaoRobot			
	Accelerate	Set the internal acceleration and deceleration ratio of the robot.	P.41
	AddVariable	Connect a system variable.	P.42
	get_VariableNames	Get a list of system variable names.	P.42
	Halt	Stop the robot in asynchronous motion.	P.42

	Change	Change the tool/user coordinate system of the robot.	P.42
	Drive	This method is not supported directly in this provider.	P.43
	Move	Robot moves.	P.43
	Rotate	Rotate around the specified axis.	P.43
	Speed	Set the internal movement speed of the robot.	P.48
	Execute	Execute a command of the robot.	P.49
CaoTask			
	AddVariable	Connect a system variable of the robot.	P.132
	get_VariableNames	Get a list of system variable names.	P.132
	Start	Start the PAC program.	P.132
	Stop	Stop the PAC program.	P.132
	Execute	Execute a command of the task class.	P.132
CaoVariable			
	get_Value	Get a value.	P.135
	put_Value	Set a value.	P.135

5.2. Methods and properties

5.2.1. CaoWorkspace::AddController method

RC9 provider establishes the connection to the target controller by referring to the parameters that have been passed at the AddController method execution.

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.RC9")
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 RC9 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.
RCPJ={<Project file>}	To perform simulation, specify a full path of a "*.rcpj" 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: "RCPJ= {C:\ Test\$Current\default.rcpj}" "RCPJ= {*}"
Message=<event notice>	Notify event occurrences. To inform an event, set TRUE. Events will not informed when FALSE is selected. It will be TRUE if it is omitted. If the notify event is not required, it is recommended to specify False for this option.

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.

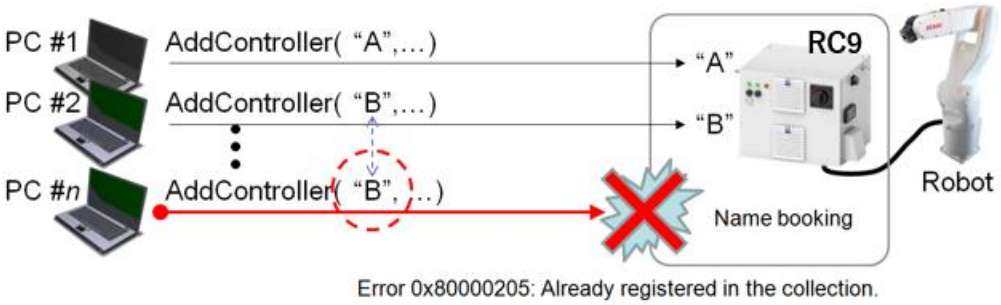
Specifying an empty string ("") for the name, the @IfNotMember option will be ignored.

Example Create CaoController

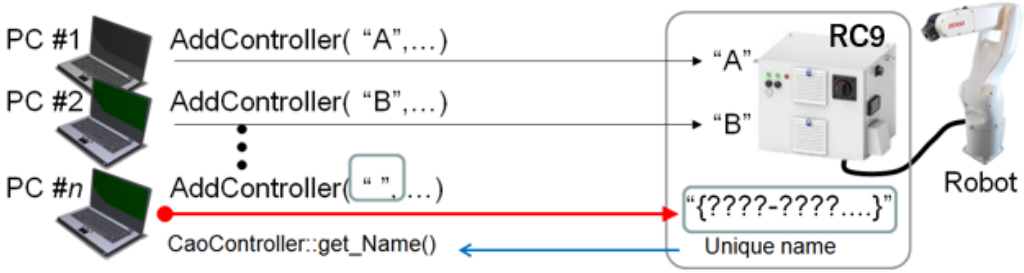
```
Private caoEng As CaoEngine             ' Engine object
Private caoWs As CaoWorkspace           ' WorkSpace object
Private caoCtrl As CaoController        ' Controller object

Set caoEng = New CaoEngine
Set caoWS = caoEng.CaoWorkspaces.Item(0)
Set caoCtrl = CaoWS.AddController("rc9","CaoProv.DENSO.RC9","・","Server=192.168.0.1,Timeout=1000")
```

5.2.1.1. When you establish multiple connections with RC9 controller

When the RC9 controller connects with devices through RC9 provider, clients (PCs) and a server (RC9 controller) communicates with b-CAP protocol in each connection (by AddController unit). Note that the controller name, which is the first argument of CaoWorkspace::AddController, needs to be the unique value. A duplicate controller name causes connection failure.



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 a PAC program name, system reserved file name, or directory name. You can specify a directory by specifying only file path for the argument. If the path is not specified, files in the project root, which is 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	*.PNS9	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 any string and there is no restriction for naming. For example, you can specify AddRobot ("Robot1").

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

bstrName	:	[in]	Robot name

Example

```
Dim CaoRob as CaoRobot
Set Rob = caoCtrl.AddRobot("Robot0" )
```

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 RC9 provider, both the user variable and the system variable can be specified for the variable name.

User variables support the following variables, i.e., RC9 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.

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::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.RC9"," " ,"Server=192.168.0.1")

Debug.Print caoCtrl. Name
```

5.2.7. CaoController::get_FileNames property

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

Example List the file names that are below 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
```

5.2.8. 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
```

5.2.9. CaoController::get_VariableNames property

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

5.2.10. CaoController::Execute method

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

For about arguments, 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 by 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.

List of commands of CaoController::Execute method

Category	Command name	Function	Support Version	
Error processing				
	ClearError	Reset an error.	1.0.0	P.24
	GetErrorDescription	Get an error string.	1.0.0	P.25
	GetCurErrorCount	Get the number of errors currently occurred.	1.0.0	P.36
	GetCurErrorInfo	Get the information of currently issued error.	1.0.0	P.37
Task control				
	KillAll	Terminate all tasks.	1.0.0	P.25

	SuspendAll	Suspend all tasks.	1.0.0	P.26
	StepStopAll	Step-stop all tasks	1.0.0	P.27
	ContinueStartAll	Continue-start all tasks.	1.0.0	P.27
	KillAllTsr	Initialized stop of all the supervisory tasks under execution.	1.0.0	P.25
	RunAllTsr	Run supervisory tasks specified by Mode.	1.0.0	P.26
	RunBlockProgram	Run the program created with Easy Block Programming	1.7.0	P.39
Log				
	GetErrorLogCount	Get count of error log.	1.0.0	P.27
	GetErrorLog	Get data of the error.	1.0.0	P.28
	GetOprLogCount	Get count of operation log.	1.0.0	P.29
	GetOprLog	Get data of the operation.	1.0.0	P.29
Variable access				
	GetPublicValue	Read a Public variable value.	1.0.0	P.30
	SetPublicValue	Write a Public variable value.	1.0.0	P.32
Misc.				
	SysState	Get controller status.	1.0.0	P.34
	SysInfo	Get the system information of the controller.	1.0.0	P.34
	SetAllDummyIO	Set the I/Os to dummy I/Os.	1.0.0	P.36
	StoState	Get the STO state (safety state).	1.0.0	P.37
	ResetStoState	Reset the STO state (safety state).	1.0.0	P.37
	ManualReset	Reset the STO state (safety state).	1.7.0	P.39
	GetSlaveButtonState	Get button status when Master/Slave.	1.0.0	P.38
	ClearSlaveButtonState	Get button status when Master/Slave	1.0.0	P.38
	GetEbpState	Get the Easy Block Programming state.	1.7.0	P.39

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

Clear an error that occurs in the controller.

Syntax ClearError ( )

Argument : None

Return value : None

Example

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

5.2.10.2. 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 )
```

5.2.10.3. 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"
```

5.2.10.4. 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"
```

5.2.10.5. 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"
```

5.2.10.6. 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 suspended status. However, supervisory tasks do not stop.

Example

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

5.2.10.7. 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"
```

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

Start all the running tasks which are being suspended.

Syntax ContinueStartAll ( )

Argument : None

Return value : None

Example

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

5.2.10.9. 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" )
```

5.2.10.10. 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
		[15]:IP address when Client ID is PC(VT_I4)

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
```

5.2.10.11. 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" )
```

5.2.10.12. 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

[11]:Operation description (VT_BSTR)

[12]:Login User ID (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
```

5.2.10.13. 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)>	:	Index numbers for each dimension (VT_I4)
<lIndex(n+1)>		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 pbIValue As Long
Public pbIArrays(1, 2, 3) As Long
Public pbIArray(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", "pbIValue")
iVal = caoCtrl.Execute("GetPublicValue", vParam)
```

VisualBasic – Reading one element of array

```
Dim vParam As Variant
Dim iVal As Long

vParam = Array("Pro1", "pbIArrays", 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", "pbIArray")
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)
```

5.2.10.14. 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)>	:	Index number for each dimension (VT_I4)
<lIndex(n+1)>		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 pbIValue As Long
Public pbIArrays(1, 2, 3) As Long
Public pbIArray(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", "pbIValue")
caoCtrl.Execute "SetPublicValue", vParam
```

VisualBasic – Writing one element of array

```
Dim iVal As Long
Dim vParam As Variant

iVal = 1234
vParam = Array(iVal, "Pro1", "pbIArrays", 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", "pbIArray")
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
```

5.2.10.15. 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")
```

5.2.10.16. 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	Reservation 0	Integer type
5	Login user level 1000: Operator 2000: Programmer 3000: Maintainer 4000: Risk assessor 5000 or 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
29	Controller Type 0:RC9	Integer type
30	Safety Type	

	0:Standard 1:Safety Motion	

Example

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

5.2.10.17. CaoController::Execute("SetAllDummyIO") command

Set the I/Os to dummy I/Os.

Syntax SetAllDummyIO (<lMode>)

<IMode> : Number (VT_I4)

Return value : none

Index number	Robot information
0	Reset all dummy IOs
1	Set user IOs to dummy IOs
2	Set system IOs to dummy IOs
3	Set both user and system IOs to dummy IOs

Example

```
g_caoCtrl.Execute "SetAllDummyIO", 0    'Reset all dummy IOs
```

5.2.10.18. CaoController::Execute("GetCurErrorCount") command

Return the number of errors currently occurred. The number of error will be 0 if error is cleared.

Syntax GetCurErrorCount ( )

Argument : none

Return value : [out] The number of errors being occurred [VT_I4]

Example

```
Dim ErrCount as long
ErrCount = caoCtrl.Execute("GetCurErrorCount")
```

5.2.10.19. CaoController::Execute(“GetCurErrorInfo”) command

Return the information of currently issued error. Number zero (0) in Index represents the latest error.

Syntax GetCurErrorInfo (<lIndex>)

< lIndex>	:	Index number(VT_I4)
Return value	:	[out] Information of the currently issued error.
		1: Error code (VT_I4)
		2: Error message (VT_BSTR)
		3: Sub code (VT_I4)
		4: File ID + line number (VT_I4)
		5: Program name (VT_BSTR)
		6: Line number (VT_I4)
		7: File ID (VT_I4)

A return value 4 consists of (((file ID << 20) & 0xFFF00000) | (line number & 0x000FFFFF)).

Example

```
Dim Errinfo as Variant
Errinfo = caoCtrl.Execute("GetCurErrorInfo",0)
```

5.2.10.20. CaoController::Execute(“StoState”) command

Return the STO state (safety state).

Syntax StoState( )

Argument	:	none
Return value	:	[out] state(VT_BOOL)

Example

```
Dim state as Variant
state = caoCtrl.Execute("StoState")
```

5.2.10.21. CaoController::Execute(“ResetStoState”) command

Reset the STO state (safety state).

[Note] This command is for backward compatibility. Usually use the CaoController::Execute("ManualReset") command.

Syntax ResetStoState( )

Argument	:	None

Return value : None

Example

```
caoCtrl.Execute "ResetStoState"
```

5.2.10.22. CaoController::Execute("GetSlaveButtonState") command

Get the button status notified from the master robot. This command can be used when the Slave Control function is enabled.

Syntax GetSlaveButtonState (<lBtnType>)

<lBtnType> : Specify the target to get the status (VT_I4). See table below.

Return value : [out] Array of button states. There are the following two elements.

0: Number of times the button has been pressed so far (VT_I4)

1: Whether the button is currently pressed (VT_I4)

0 is not pressed, 1 is pressed

Index number	Button type
0	COBOTTA function button
1	COBOTTA hand plus button
2	COBOTTA hand minus button

Example

```
Dim vntButtonState as Variant
vntButtonState = caoCtrl.Execute("GetSlaveButtonState", 0)
```

5.2.10.23. CaoController::Execute("ClearSlaveButtonState") command

When the Slave Control function is enabled, the button status notified from the master robot to the slave controller is cleared. Executing this command resets the count of the number of times the button has been pressed to 0.

Syntax ClearSlaveButtonState(<lBtnType>)

<lBtnType> : Specify the target to get the status (VT_I4). See table above.

Return value : None

Example

```
caoCtrl.Execute"ClearSlaveButtonState"
```

5.2.10.24. CaoController::Execute("ManualReset") command

Reset the STO state (safety state).

Syntax ManualReset( )

Argument	:	None
Return value	:	None

Example

```
caoCtrl.Execute "ManualReset"
```

5.2.10.25. CaoController::Execute("RunBlockProgram") command

Run the program created with Easy Block Programming.

Syntax RunBlockProgram (<bstrTaskName>)

<bstrTaskName>	:	Name of the program to run(VT_BSTR)
Return value	:	None

Example

```
caoCtrl.Execute "RunBlockProgram", "Pro1"
```

5.2.10.26. CaoController::Execute("GetEbpState") command

Get the Easy Block Programming state.

Syntax GetEbpState (<lIndex> [, <vntParam>])

<lIndex>	:	Index number of the state to get (VT_I4)
<vntParam>	:	Additional information (VT_VARIANT)
		Set according to index number
Return value	:	State value (VT_VARIANT)
		Return value varies by index number

Example

```
vntStatus = caoCtrl.Execute("GetEbpState", lIndex)
vntStatus = caoCtrl.Execute("GetEbpState", Array(lIndex, vntParam))
```

See the table below for the corresponding index numbers, additional information, and return values as arguments.

Index number	Additional information	Return value(type)
0	None	EasyBlockProgramming initialization state(VT_BOOL) True: Initialization completed False: Incomplete Initialization
4	None	EasyBlockProgramming all program execution state(VT_BOOL) True: One or more programs are running False: All programs stopped
20	Program name (VT_BSTR)	Execution status of the named program (VT_I4) 0: Stopping 1: Running

5.2.11. 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
```

5.2.12. CaoFile::AddVariable method

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

Refer to Table 5-13 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
```

5.2.13. CaoFile::get_VariableNames property

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

5.2.14. 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.

5.2.15. 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
```

5.2.16. 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
```

5.2.17. CaoFile::put_Value property

Set the contents of the file corresponding to the object.

5.2.18. 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 when a license key of b-CAP Slave is added and the robot is controlled by a client as a b-CAP 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%
```

5.2.19. CaoRobot::AddVariable method

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

Refer to Table 5-11 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
```

5.2.20. CaoRobot::get_VariableNames property

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

5.2.21. CaoRobot::Halt method

Stop the robot motion.

A runtime error occurs if a task in the RC9 controller has robot control authority (when Takearm has been executed). Use the CaoTask::Stop method to control the stop of a task.

5.2.22. CaoRobot::Change method

Change the tool/user coordinate system of the robot.

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

The following shows the argument specifications of Change.

Syntax Change <bstrName:BSTR>

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

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

Example

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

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

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

caoRob.Execute"GiveArm"
```

5.2.23. 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.

5.2.24. 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 " Appendix A. 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 the 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 ", " "@< Value >"or "@A<Value>"

<Extended-joints> : The syntax of an extended-joints option is shown below.[1]
(Specify the extended-joints option after a pose data and a blank.)

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

Example 1	Move 1,"@P P1" ,"NEXT"	‘ MOVE P, @P P1, NEXT
Example 2	Move 3,"P1,@E P2"	‘ MOVE C, P1,@E P2
Example 3	Move 2,"@0 P(307.1856,-157.8244,107.0714,160,0,0,1)"	‘ MOVE L,@0 P(307.1856,-157.8244,107.0714,160,0,0,1)
Example 4	Move 4,"@E 2"	‘ MOVE S, @E 2
Example 5	Move 1,"@P P10 EX((7, 30.5))" ,"NEXT"	‘ MOVE P, @P P10 EX((7,30.5)), NEXT
Example 6	Move 2,"@E P20 EXA((7, 30.8), (8, 90.5))	‘MOVE L, @E P20 EXA((7, 30.8), (8, 90.5))"
Example 7	Move 1,"P1,@A[50] P2"	‘ MOVE P, P1,@A[50] P2

This method is not available when a license key of b-CAP Slave is added and the robot is controlled by a client as a b-CAP SlaveMode.

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

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

Table 5-5 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*>"
Misc.	MOVE P, P<*n1*> +(*x*,*y*,*z*,*rx*,*ry*,*rz*)	Move 1, DEV ("<*n1*>","P(*x*,*y*,*z*,*rx*,*ry*,*rz*)")
	MOVE P, P<*n1*> +(*x*,*y*,*z*,*rx*,*ry*,*rz*)H	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>, …)"

5.2.25. CaoRobot::Rotate method

Rotate the robot around the specified axis.

This method corresponds to ROTATE instruction of PacScript language.

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 "Appendix A. 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.

5.2.26. 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.

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.

5.2.27. 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. For commands for the OnRobot hand, see the CaoRobot::Execute method (for OnRobot hand).

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

Category	Command name	Function	Support Version	
Operation				
	TMul	Calculate the product of two homogeneous transformation type data.	1.0.0	P.54
	TInv	Calculate the inverse matrix of homogeneous transformation type data.	1.0.0	P.54
	TNorm	Calculate the inverse matrix of homogeneous transformation type data.	1.0.0	P.55
	J2T	Transform J type data to T type data.	1.0.0	P.55
	T2J	Transform T type data to J type data.	1.0.0	P.56
	J2P	Transform J type data to P type data.	1.0.0	P.56
	P2J	Transform P type data to J type data.	1.0.0	P.56

	T2P	Transform T type data to P type data.	1.0.0	P.57
	P2T	Transform P type data to T type data.	1.0.0	P.57
	Dev	Calculate the offset in the base coordinates.	1.0.0	P.58
	DevH	Calculate the offset in the tool coordinates.	1.0.0	P.58
	OutRange	Judge whether the motion range is exceeded.	1.0.0	P.58
	MPS	Calculate the value of the SPEED command from data in Mps.	1.0.0	P.59
	RPM	Calculate the value of the SPEED command from data in rpm.	1.0.0	P.59
	DPS	Calculate the value of SPEED command from deg/s unit	1.0.0	P.60
Positioning				
	CurPos	Get the current position as P type data.	1.0.0	P.60
	DestPos	Get the target position as P type data.	1.0.0	P.61
	CurJnt	Get the current position as J type data.	1.0.0	P.62
	DestJnt	Get the target position as J type data.	1.0.0	P.62
	CurTrn	Get the current position as T type data.	1.0.0	P.63
	DestTrn	Get the target position as T type data.	1.0.0	P.64
	CurFig	Get the current posture Fig value.	1.0.0	P.65
	CurPosEx	Get the current position data by P type data with time stamp.	1.0.0	P.61
	DestPosEx	Get the target position data with time stamp as P type data.	1.0.0	P.61
	CurJntEx	Get the current position data by J type data with time stamp.	1.0.0	P.62
	DestJntEx	Get the target position data with time stamp as J type data.	1.0.0	P.63
	CurTrnEx	Get the current position data by T type data with time stamp.	1.0.0	P.64
	DestTrnEx	Get the target position data with time stamp as T type data.	1.0.0	P.65
Speed				
	SpeedMode	Change the optimal speed setting function.	1.0.0	P.88
	CurSpd	Return an internal speed setting value.	1.0.0	P.65

	CurAcc	Return an internal acceleration setting value.	1.0.0	P.66
	CurDec	Return an internal deceleration setting value.	1.0.0	P.66
	CurJSpd	Return internal axis speed.	1.0.0	P.66
	CurJAcc	Return internal axis acceleration.	1.0.0	P.67
	CurJDec	Return internal axis deceleration.	1.0.0	P.67
Log				
	StartLog	Start log recording.	1.0.0	P.68
	StopLog	Stop log recording.	1.0.0	P.69
	ClearLog	Clear log data.	1.0.0	P.69
	StartServoLog	Start servo logging.	1.0.0	P.92
	ClearServoLog	Delete obtained servo log data.	1.0.0	P.92
	StopServoLog	Stop servo logging.	1.0.0	P.93
	GetCtrlLogMaxTime	Get the duration of control logging.	1.0.0	P.93
	SetCtrlLogMaxTime	Set the duration of control logging.	1.0.0	P.93
	GetCtrlLogInterval	Get the interval of control logs.	1.0.0	P.93
	SetCtrlLogInterval	Set the interval of control logs.	1.0.0	P.94
	GetLogCount	Get count of control log.	1.0.0	P.95
	GetLogRecord	Get data of control log.	1.0.0	P.95
Robot operation				
	Motor	Turn ON/OFF the motor.	1.0.0	P.69
	ExtSpeed	Set the external speed.	1.0.0	P.70
	TakeArm	Request to get control authority.	1.0.0	P.70
	GiveArm	Request to release control authority.	1.0.0	P.71
	Draw	Execute the relative movement specified in the work coordinate system.	1.0.0	P.72
	Approach	Execute the absolute movement specified in the tool coordinate system.	1.0.0	P.72
	Depart	Execute the relative movement specified in the tool coordinate system.	1.0.0	P.73
	DriveEx	Execute the relative motion of each axis.	1.0.0	P.74
	DriveAEx	Execute the absolute motion of each axis.	1.0.0	P.75
	RotateH	Execute rotary motion by taking an approach vector as an axis.	1.0.0	P.75

	Arrive	Wait for the robot to reach the defined motion ratio.	1.0.0	P.76
	MotionSkip	Abort the robot motion in progress.	1.0.0	P.76
	MotionComplete	Judge whether the robot motion is complete.	1.0.0	P.77
	MoveMasterPos	Move each joint to angle of master robot joint.	1.0.0	P.104
Tool				
	CurTool	Get the current tool number.	1.0.0	P.78
	GetToolDef	Get the tool definition specified by the tool number.	1.0.0	P.78
	SetToolDef	Set the tool definition.	1.0.0	P.78
Work				
	CurWork	Get the current work number.	1.0.0	P.79
	GetWorkDef	Get the work definition specified by the work number.	1.0.0	P.79
	SetWorkDef	Set the work definition.	1.0.0	P.79
	WorkAttribute	Get work attribute specified by a work number.	1.0.0	P.80
Base				
	SetBaseDef	Specify the base definition.	1.0.0	P.91
	GetBaseDef	Obtain the base definition.	1.0.0	P.91
Area				
	GetAreaDef	Get the area definition specified by the area number.	1.0.0	P.80
	SetAreaDef	Set the area parameter.	1.0.0	P.80
	SetArea	Enable the area check.	1.0.0	P.81
	ResetArea	Disable the area check.	1.0.0	P.81
	AreaSize	Return the size (each side length) of a check area as the vector type.	1.0.0	P.82
	GetAreaEnabled	Get the area enabled or disabled status.	1.0.0	P.82
	SetAreaEnabled	Set the area enabled or disabled status.	1.0.0	P.82
Hand IO				
	SetHandIO	Set I/O number, values, and range of Hand I/O.	1.0.0	P.91
	GetHandIO	Obtain IO number, range, and values of HandI/O.	1.0.0	P.92
	GetHandAnalogInput	Get the voltage/current of the analog input.	1.7.0	P.96
Accuracy				

	CrtMotionAllow	Change the stop precision settings.(@C)	1.0.0	P.83
	EncMotionAllow	Change the "Allowable angle in stop state" of robot axis.(@E)	1.0.0	P.84
	EncMotionAllowJnt	Change the "Allowable angle in stop state" of other than robots' axis.(@E)	1.0.0	P.84
	HighPathAccuracy	Switch "True/False" of the high tracing control function.	1.0.0	P.87
Status/Value Obtainment				
	GetSrvData	Get servo internal data of the robot axis.	1.0.0	P.85
	GetSrvJntData	Get servo internal data of the other than robots' axis.	1.0.0	P.86
	RobInfo	Return the robot information.	1.0.0	P.90
	SafetyInfo	Get information about the Safety Motion.	1.7.0	P.96
	GetSafetyPayload	Get the load characteristics used by the Safety Motion.	1.7.0	P.98
Settings of Functions and Conditions				
	GrvCtrl	Switch "True/False" of Gravity Compensation Control Function.	1.0.0	P.87
	MotionTimeout	Change a timeout setting value of the motion instruction.	1.0.0	P.87
	ErAlw	Configure the deviation tolerance function.	1.0.0	P.85
	PayLoad	Change the setting value of internal load conditions.	1.0.0	P.89
	SingularAvoid	Enable or disable singularity avoidance function.	1.0.0	P.88
	SetSlowSpdChk	Enable Auto mode slow check state.	1.7.0	P.99
Scene				
	ChangeScene	Switch the current scenes and subscenes.	1.7.0	P.99
	CurScene	Get the current scene number.	1.7.0	P.99
	CurSubScene	Get the current subscene number.	1.7.0	P.100
	SceneInfo	Get information about the scene specified by the scene number.	1.7.0	P.100

Misc.			
GetRobotTypeName	Get the robot type.L.	1.0.0	P.83
GetPluralServoData	Obtain multiple servo data at one time.	1.0.0	P.94
GenerateNonStopPath	This command is exclusive to “the Non-stop motion calculator option.”	1.0.0	P.89

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
```

5.2.27.1. 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
```

5.2.27.2. 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
```

5.2.27.3. 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
```

5.2.27.4. 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
```

5.2.27.5. 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
```

5.2.27.6. 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
```

5.2.27.7. 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
```

5.2.27.8. 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
```

5.2.27.9. 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
```

5.2.27.10. 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)
```

5.2.27.11. 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)
```

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

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

The tool and work definition are ignored if <Pose> is specified as J type data.The tool and work definition specified as POSEDATA (P type) is avaiable since Controller version 2.0.*.

Syntax OutRange(<Pose>[, <ToolDef> [, <WorkDef>]])

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

<ToolDef> : [in] Tool definition. POSEDATA (P type) or VT_I4
POSEDATA (P type) : Tool definition.
VT_I4 : Tool number.-1 (default) is the current tool number VT_I4.

<WorkDef> : [in] Work definition. POSEDATA (P type) or VT_I4
POSEDATA (P type) : Work definition.
VT_I4 : 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)" )
```

5.2.27.13. 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
```

5.2.27.14. 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
```

5.2.27.15. 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
```

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

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

5.2.27.17. 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
---------------------------------------------------------------------------------------------------------------------------
```

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

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

5.2.27.19. 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
```

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

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

5.2.27.21. 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
```

5.2.27.22. 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
```

5.2.27.23. 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
```

5.2.27.24. 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
```

5.2.27.25. 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
```

5.2.27.26. 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
```

5.2.27.27. 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
```

5.2.27.28. 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
```

5.2.27.29. 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)
```

5.2.27.30. 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)
```

5.2.27.31. 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)
```

5.2.27.32. 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)
```

5.2.27.33. 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)
---------------------------------------------------------------------------------------------------------------------
```

5.2.27.34. 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)
---------------------------------------------------------------------------------------------------------------------
```

5.2.27.35. CaoRobot::Execute("CurExtSpd") command

Return an external speed setting value.

Syntax CurExtSpd()

Argument	:	None
Return value	:	External speed (VT_R4)

Example

```
---------------------------------------------------------------------------------------------------------------------
    Dim vSpd As Variant
    vSpd = caoRob.Execute("CurExtSpd" )
---------------------------------------------------------------------------------------------------------------------
```

5.2.27.36. CaoRobot::Execute("CurExtAcc") command

Return an external acceleration setting value.

Syntax CurExtAcc()

Argument : None

Return value : External acceleration (VT_R4)

Example

```
Dim vAcc As Variant
vAcc = caoRob.Execute("CurExtAcc")
```

5.2.27.37. CaoRobot::Execute("CurExtDec") command

Return an external deceleration setting value.

Syntax CurExtDec()

Argument : None

Return value : External deceleration (VT_R4)

Example

```
Dim vDec As Variant
vDec = caoRob.Execute("CurExtDec")
```

5.2.27.38. 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.

Syntax StartLog( )

Argument : None

Return value : None

Execute the ClearLog command before this command to enable recording.

Example

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

5.2.27.39. 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"
```

5.2.27.40. 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"
```

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

Turn ON/OFF the motor.

This method is not available when a license key of b-CAP Slave is added and the robot is controlled by a client as a b-CAP 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
```

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

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

This method is not available when a license key of b-CAP Slave is added and the robot is controlled by a client as a b-CAP SlaveMode.

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

Speed : [in] External speed (VT_R4)

Accel : [in] External acceleration (VT_R4)
-1 (default): Keep the current setting (Not change the current setting)
"-2" is entered if it is omitted.

Decel : [in] External deceleration (VT_R4)
-1 (default): Keep the current setting (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.

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

Request to get control authority.

This command corresponds to TAKEARM instruction of PacScript language.

This method is not available when a license key of b-CAP Slave is added and the robot is controlled by a client as a b-CAP 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.
```

5.2.27.44. 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.'
```

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

Execute the relative movement specified in the work coordinate system

This command corresponds to DRAW instruction of PacScript language.

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

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

5.2.27.46. 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.

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")
```

5.2.27.47. 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.

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")
```

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

Execute the relative motion of each axis.

This command corresponds to DRIVE instruction of PacScript language.

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

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

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

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

Execute the absolute motion of each axis.

This command corresponds to DRIVEA instruction of PacScript language.

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

vntPoses	:	[in] Axis number and distance (POSEDATA type, C3 format) Specify the desired axes and axis coordinates in POSEDATA type for eight axes at the maximum.
strOpt	:	[in] Motion option (VT_BSTR) [SPEED=n][,ACCEL=n][,DECEL=n][,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")
```

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

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

This command corresponds to ROTATEH instruction of PacScript language.

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" )
```

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

Wait for the robot to reach the defined motion ratio.

This command corresponds to ARRIVE instruction of PacScript language.

Syntax Arrive (<Motion ratio>)

Argument	:	[in] (VT_R4) Motion ratio
Return value	:	None

Example

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

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

Abort the robot motion in progress.

This command corresponds to MOTIONSKIP instruction of PacScript language.

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

ArmGroup	:	[in] Arm group number (VT_I4) -1 (default): Current arm group under control
Parameter	:	[in] Operation continuation pattern (VT_I4) 0 (default): Specify the pass start displacement as @0 and connect it with the maximum deceleration. 1: Specify the pass start displacement as @P and connect it with the maximum deceleration. 2: Specify the pass start displacement as @0 and connect it with the set deceleration. 3:1: Specify the pass start displacement as @P and connect it with the set deceleration.
Return value	:	None

Example

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

5.2.27.53. 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
```

5.2.27.54. 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")
```

5.2.27.55. 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("GetToolDef", 1)
Debug.Print"X=・ & vVal(0) &", Y=・ & vVal(1) &", Z=・ & vVal(2)
Debug.Print"RX=・ & vVal(3) &", RY=・ & vVal(4) &", RZ=・ & vVal(5)
```

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

Set the tool definition.

This method is not available when a license key of b-CAP Slave is added and the robot is controlled by a client as a b-CAP 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)")
```

5.2.27.57. 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( "CurWork")
```

5.2.27.58. 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)
```

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

Set the work definition.

This method is not available when a license key of b-CAP Slave is added and the robot is controlled by a client as a b-CAP 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)")
```

5.2.27.60. 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)
```

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

Get the area definition with the specified area number.

Syntax GetAreaDef (<AreaNo>)

Argument	:	[in] Area 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)
```

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

Set the area parameter.

This method is not available when a license key of b-CAP Slave is added and the robot is controlled by a client as a b-CAP 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)
```

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

Enable the area check.

This method is not available when a license key of b-CAP Slave is added and the robot is controlled by a client as a b-CAP SlaveMode.

Syntax SetArea (<AreaNum>)

<AreaNum> : Area number (VT_I4)

Return value : None

Example

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

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

Disable the area check.

This method is not available when a license key of b-CAP Slave is added and the robot is controlled by a client as a b-CAP SlaveMode.

Syntax ResetArea (<AreaNum>)

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

Example

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

5.2.27.65. 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)
```

5.2.27.66. 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
```

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

Set the area enabled or disabled status.

This method is not available when a license key of b-CAP Slave is added and the robot is controlled by a client as a b-CAP 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
```

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

Get the robot type.

Syntax GetRobotTypeName ( )

Argument	:	None
Return value	:	Robot type (VT_BSTR)

Example

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

5.2.27.69. 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 when a license key of b-CAP Slave is added and the robot is controlled by a client as a b-CAP 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
```

5.2.27.70. 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 when a license key of b-CAP Slave is added and the robot is controlled by a client as a b-CAP 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
```

5.2.27.71. 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 when a license key of b-CAP Slave is added and the robot is controlled by a client as a b-CAP 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)
```

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

Set the deviation tolerance function and True/False the function.

This command corresponds to ERALW instruction of PacScript language.

This method is not available when a license key of b-CAP Slave is added and the robot is controlled by a client as a b-CAP 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)
```

5.2.27.73. 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(1,2,4,5,7,8,17,18,19,20) (VT_I4)

Return value : Servo internal data

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

1: Current motor speed value (rpm)

2: Motor angle deviation (mm or deg)
4: Absolute motor current value (Rated ratio %)
5: Motor torque command position (excluding dead weight compensation) (Rated ratio %)
7: Load factor (%)
8: Each axial position and angle command position (mm or deg)
17: Tool tip speed [work coordinates] (mm/s)
* Position: Only 3 Components are obtained.
18: Tool tip deviation [work coordinates] (mm)
* Position: Only 3 Components are obtained.
19: Tool tip speed [tool coordinates] (mm/s)
* Position: Only 3 Components are obtained.
20: Tool tip deviation [tool coordinates] (mm)
* Position: Only 3 Components are obtained.

Example

```
Dim vntVal As Variant
vntVal = caoRobot.Execute("GetSrvData", 2)
```

5.2.27.74. 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)
1: Current motor speed value (rpm)
2: Motor angle deviation (mm or deg)
4: Absolute motor current value (Rated ratio %)
5: Motor torque command position (excluding dead weight compensation) (Rated ratio %)
8: Each axial position and angle command position (mm or deg)

Example

```
Dim fData As Single
```

```
fData = caoRobot.Execute("GetSrvJntData", 2, 1)
```

5.2.27.75. 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 when a license key of b-CAP Slave is added and the robot is controlled by a client as a b-CAP 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
```

5.2.27.76. 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 when a license key of b-CAP Slave is added and the robot is controlled by a client as a b-CAP 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
```

5.2.27.77. 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 when a license key of b-CAP Slave is added and the robot is controlled by a client as a b-CAP 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
```

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

Enable or disable singularity avoidance function.

This command corresponds to SINGULARAVOID instruction of PacScript language.

This method is not available when a license key of b-CAP Slave is added and the robot is controlled by a client as a b-CAP 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
```

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

Change the optimal speed setting function.

This command corresponds to SPEEDMODE instruction of PacScript language.

This method is not available when a license key of b-CAP Slave is added and the robot is controlled by a client as a b-CAP 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
```

5.2.27.80. 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 when a license key of b-CAP Slave is added and the robot is controlled by a client as a b-CAP SlaveMode.

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

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

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

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

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

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

< Teaching Points >	:	[in]Teaching Points [<Position > | VT_ARRAY]
< Area Information >	:	[in] Area Information [<Area > | VT_ARRAY]
< Teaching Point Number >		[in] Teaching Point Number [VT_I4]
<Total Speed Rate>		[in] Total Speed Rate [VT_R8] 0.0 - 1.0 This ratio is used to change the entire speed of Non-stop motion. This is equivalent to the external speed of robot.
<Convergence Coefficient>	:	[in] Convergence Coefficient [VT_R8] 0.0 to 1.0. This factor is used to calculate the motion point by convergent calculation. Specify “0.7” for normal use.
Return value	:	[out]Motion Points [<Position > | VT_ARRAY]

Example

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

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

Return the robot information

This command 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
3	Robot ID	Integer type

Example

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

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

Specify the base definition.

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)”)
```

5.2.27.84. 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)
```

5.2.27.85. 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)
```

5.2.27.86. 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)).
```

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

Start servo logging.

Syntax StartServoLog()

Argument	:	none
Return value	:	none

Example

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

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

Delete obtained servo log data.

Syntax ClearServoLog()

Argument	:	none
Return value	:	none

Example

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

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

Stop servo logging

Syntax StopServoLog()

Argument : none

Return value : none

Example

```
caoRobot.Execcute “StopServoLog
```

5.2.27.90. 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”)
```

5.2.27.91. 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
```

5.2.27.92. 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”)
```

5.2.27.93. 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
```

5.2.27.94. CaoRobot::Execute("GetPluralServoData ") command

Obtain multiple servo data at one time.

Syntax GetPluralServoData ()

Argument : none
Return value : Servo data (VT_VARIANT|VT_ARRAY)
Current motor speed, Motor angle deviation, Absolute value of motor current, Instruction value of motor torque, Each axes position, Angle instruction value, Tool-end speed, Tool-end deviation

Example

```
Dim vVal As Variant
vVal = caoRobot.Execcute (“GetPluralServoData”)
```

5.2.27.95. CaoRobot::Execute("GetLogCount") command

Get count of control log.

Syntax GetLogCount ()

Argument	:	None
Return value	:	Norn

Example

```
Dim caoRob As CaoRobot
Set caoRob = caoCtrl.AddRobot(“Arm0",0)

caoRob.Execute"TakeArm", Array(0, 0)
caoRob.Execute "Motor", Array(1, 0)

CaoRob.Execute“StartBackTraceLog”

caoRob.Move 2, "@E P1","S = 30”

caoRpb.Move 2, "@E P2","S = 100”

CaoRob.Execute“StopBackTraceLog”
caoRob.Execute"GiveArm"
```

5.2.27.96. CaoRobot::Execute("GetLogRecord") command

Get data of control log.

Syntax GetLogRecord (<lIndex>)

<lIndex>	:	Index number (VT_I4) 0 to <Count>-1
Return value	:	Details (VT_VARIANT[VT_VARIANT|VT_ARRAY:39 elements])
		[0-7]:J1-J8 Instruction value (VT_R8)
		[8-15]:J1-J8 Encoder value (VT_R8)
		[16-23]: J1-J8 Current value (VT_R8)
		[24-31]: J1-J8 Load factor (VT_R8)
		[32]: User data (VT_R8)
		[33]: Trace data (VT_I4)
		[34]: Program name (VT_BSTR)
		[35]: Line number (VT_I4)
		[36]: Elapsed time (start time)[ms] (VT_I4)
		[37]: Tool number (VT_I4)
		[38]: Work number (VT_I4)

Example

```
Dim lCnt As Long
lCnt = caoRobot.Execute("GetLogCount")
Dim i As Long
For i=0 To lCnt-1
  vDat = caoRobot.Execute("GetLogRecord", i)
  ‘ vDat(0) : J1 Inst, …,vDat(7) : J8 Inst
  ‘ vDat(8) : J1 Enc, …,vDat(15) : J8 Enc
  ‘ :
Next
```

5.2.27.97. CaoRobot::Execute("MoveMasterPos") command

Move each joint of the slave robot to angle of the master robot joint.

This command corresponds to MOVEMASTERPOS instruction of PacScript language.

Syntax MoveMasterPos ()

Argument	:	None
Return value	:	Norn

Example

```
caoRob.Execute"TakeArm", Array(0, 0)
caoRob.Execute "Motor", Array(1, 0)
caoRob.Execute"MoveMasterPos"
```

5.2.27.98. CaoRobot::Execute("GetHandAnalogInput") command

Get the voltage/current of the analog input.

This command corresponds to GetHandAnalogInput instruction of PacScript language.

Syntax GetHandAnalogInput(<lChNo>)

<lChNo>	:	Channel No. (VT_I4)
Return value	:	Voltage [V] / Current [A] (VT_R4)

Example

```
Dim fVal As Single
fVal = caoRob.Execute("GetHandAnalogInput", 0)
```

5.2.27.99. CaoRobot::Execute("SafetyInfo") command

Get information about the Safety Motion.

This command corresponds to SafetyInfo instruction of PacScript language.

Syntax SafetyInfo(<lIndex>)

<lIndex>	:	Index (VT_I4)

Return value : Information specified by the index.

Index	Information	Data type
0	The state of the Safety Motion. -1: Indefinite 0: Safe state 1: Normal 2: Recovery 3: Standby 4: Check	VT_I4
1	RC9 Safety Motion specification: Tool number recognized by the Safety Motion COBOTTA PRO: Current tool number	VT_I4
2	RC9 Safety Motion specification: Tool definition of the tool number that recognized by the Safety Motion COBOTTA PRO: Tool definition for the current tool number	VT_I4 | VT_ARRAY
3	RC9 Safety Motion specification: Speed limit recognized by the Safety Motion. 0: None 1: RLS 2: SS2 COBOTTA PRO: Always 0	VT_I4
4	Input state of the Safety Motion I/O RC9 Safety Motion specification: 1st bit: SS2 input 2nd bit: RLS input 5th bit: TOOL0 input 6th bit: TOOL1 input 7th bit: TOOL2 input 8th bit: TOOL3 input COBOTTA PRO: 1st–8th bit: Universal safety input 1–8	VT_I4

5	Output state of the Safety Motion I/O RC9 Safety Motion specification: 1st bit: STO output 2nd bit: SOS output 9th bit: SLP output 10th bit: RLS output 13th bit: RLP output COBOTTA PRO: 1st –8th bit: Universal safety input 1–8	VT_I4
6	Each joint angle recognized by the Safety Motion	VT_R4 | VT_ARRAY
7	Safety parameter ID	VT_I4
8	Whether manual reset is required 0: Manual reset not required 1: Manual reset required	VT_I4

Example

```
Dim lState As Long
lState = caoRob.Execute("SafetyInfo", 0)
```

5.2.27.100. CaoRobot::Execute("GetSafetyPayload") command

Returns the load characteristics used by the Safety Motion.

This command corresponds to GetSafetyPayload instruction of PacScript language.

Syntax GetSafetyPayload(<lIndex>)

<lIndex> : Tool number (VT_I4)
If -1 is specified, the load characteristics of the tool number currently recognized by the Safety Motion are returned.

Return value : Load characteristics (VT_VARIANT | VT_ARRAY)
[0]: Mass [g] (VT_I4)
[1]: Center of gravity [mm] (VT_R4 | VT_ARRAY)
[0]: Center of gravity X [mm] (VT_R4)
[1]: Center of gravity Y [mm] (VT_R4)
[2]: Center of gravity Z [mm] (VT_R4)
[2]: Moment of inertia [$kgcm^2$] (VT_R4 | VT_ARRAY)

[0]: Moment of inertia [kgcm^2] (VT_R4)

[1]: Moment of inertia [kgcm^2] (VT_R4)

[2]: Moment of inertia [kgcm^2] (VT_R4)

Example

```
Dim vntVal As Variant
vntVal = caoRob.Execute("GetSafetyPayload", -1)
```

5.2.27.101. CaoRobot::Execute("SetSlowSpdChk") command

Enable the Auto mode slow check feature. When this function is enabled, the speed of the monitoring point on the robot is reduced so that it does not exceed the value set in the parameter "Auto mode slow check speed". Before running your program at high speed, be sure to slow it down with this function and check its operation. This function is disabled when the CaoRobot that executed the command is removed. In COBOTTA PRO, monitoring is also performed by the monitored-speed function of the safety function.

Syntax SetSlowSpdChk( )

Argument : None

Return value : None

Example

```
caoRob.Execute "SetSlowSpdChk"
caoRob.Move 1, “P0”     ‘Slowed down so as not to exceed auto mode slow check speed.
```

5.2.27.102. CaoRobot::Execute("ChangeScene") command

Switch the current scenes and subscenes.

This command corresponds to ChangeScene instruction of PacScript language.

Syntax ChangeScene(<lSceneNo>, <lSubsceneNo>)

<lSceneNo> : Scene number (VT_I4)

<lSubsceneNo> : Subscene number (VT_I4)

Return value : None

Example

```
caoRob.Move 1, “SafetyP0”
caoRob.Execute “ChangeScene”, Array(0, 0)
```

5.2.27.103. CaoRobot::Execute("CurScene") command

Get the current scene number.

This command corresponds to CurScene instruction of PacScript language.

Syntax CurScene( )

Argument : None

Return value : Scene number (VT_I4)

Example

```
Dim lSceneNo As Long
lSceneNo = caoRob.Execute("CurScene")
```

5.2.27.104. CaoRobot::Execute("CurSubScene") command

Get the current subscene number.

This command corresponds to CurSubScene instruction of PacScript language.

Syntax CurSubScene( )

Argument : None

Return value : Subscene number (VT_I4)

Example

```
Dim lSubSceneNo As Long
lSubSceneNo = caoRob.Execute("CurSubScene")
```

5.2.27.105. CaoRobot::Execute("SceneInfo") command

Get information about the scene specified by the scene number.

This command corresponds to SceneInfo instruction of PacScript language.

Syntax SceneInfo(<lSceneNo>, <lSubsceneNo>)

<lSceneNo> : Scene number (VT_I4)

<lSubsceneNo> : Subscene number (VT_I4)

Return value : Scene settings (VT_I4 | VT_ARRAY)

[0]: Work number

[1]: Tool number

[2]: Workpiece number

[3]: SafetyP number of the switching condition

[4]: RLS [mm/s]

[5]: RLF [N]

Example

```
Dim vntVal As Variant
vntVal = caoRob.Execute("SceneInfo", Array(0, 1))
```

5.2.28. CaoRobot::Execute method (for OnRobot hand)

The table lists the commands available for the OnRobot hand of the Execute method. The meaning of the "<lInstance>: Target instance (VT_I4)" value given to the command is as follows:

0: Hand set to current tool number

1: A single or primary in dual configuration

2: secondary in dual configuration

The default is "0: Hand set to current tool number."

Table 5-7 List of commands for the OnRobot hand in the CaoRobot::Execute method

Category	Command name	Function	Support Version	
All hands				
	HandGrip	Execute the grip command according to the value set in the parameter	1.7.0	P.104
	HandRelease	Execute the release command according to the value set in the parameter	1.7.0	P.104
	CBGetIO	Get the state of the Compute Box's digital input	1.7.0	P.105
RG2/RG6				
	RgxIsConn	Return connection state of RG2 or RG6	1.7.0	P.105
	RgxIsRg2	Return connection state of RG2	1.7.0	P.105
	RgxIsRg6	Return connection state of RG6	1.7.0	P.106
	RgxIsBusy	Return busy state	1.7.0	P.106
	RgxIsGrip	Return grip state	1.7.0	P.107
	RgxIsSafetyOn	Return safety switch status	1.7.0	P.107
	RgxGetWidth	Return current width value	1.7.0	P.107
	RgxGetDepth	Return the current absolute depth	1.7.0	P.108
	RgxGetRelativeDepth	Return actual relative depth from beginning of last actuation	1.7.0	P.108
	RgxSetFTOffset	Set the fingertip offset	1.7.0	P.108
	RgxMove	Move the fingers to the desired position	1.7.0	P.108
	RgxStop	Stop the fingers motion in progress	1.7.0	P.109
	RgxResetPower	Reset tool power and safety switch	1.7.0	P.109

VG10/VGC10				
	VgxIsConn	Return connection state of VG10 or VGC10	1.7.0	P.109
	VgxIsVg10	Return connection state of VG10	1.7.0	P.110
	VgxIsVgc10	Return connection state of VGC10	1.7.0	P.110
	VgxGetVacA	Return current vacuum of channel A	1.7.0	P.111
	VgxGetVacB	Return current vacuum of channel B	1.7.0	P.111
	VgxGetLimit	Return actual current limit	1.7.0	P.111
	VgxSetLimit	Set the internal current limit, which is proportional to the airflow	1.7.0	P.111
	VgxGrip	Execute gripping with the specified vacuum level	1.7.0	P.112
	VgxIdle	Turn off the pumps on the desired channels. The valves on the affected channels will be still closed. This is a low energy state that can keep the object gripped until several seconds.	1.7.0	P.112
	VgxRelease	Turn off the pumps and open the valves on the affected channels to release Workpiece	1.7.0	P.113
Soft Gripper				
	SgIsConn	Return connection state	1.7.0	P.113
	SgIsBusy	Return busy state	1.7.0	P.113
	SgIsInitialized	Return initialization state	1.7.0	P.114
	SgGetOpenMin	Return the minimum width of the current tool	1.7.0	P.114
	SgGetOpenMax	Return the maximum width of the current tool	1.7.0	P.114
	SgHasError	Return error state	1.7.0	P.115
	SgGetDepth	Return current depth	1.7.0	P.115
	SgGetRelativeDepth	Return actual relative depth from beginning of last actuation	1.7.0	P.115
	SgGetSiliconDepth	Return the depth of the currently used tool	1.7.0	P.116
	SgGetWidth	Return current width	1.7.0	P.116
	SgHome	Move Soft Gripper to the home position	1.7.0	P.116
	SgInit	Move Soft Gripper to the home position and set the tool type to SG	1.7.0	P.117
	SgGrip	Grip to the desired position	1.7.0	P.117

	SgGentleGrip	Gentle grip to the desired position and slow down when the gripper is close to the target width	1.7.0	P.117
	SgRelease	Release the object by moving to the desired width	1.7.0	P.118
	SgStop	Stop the motion in progress	1.7.0	P.118
2FG7				
	TwofgIsConn	Return the connection state	1.7.0	P.119
	TwofgIsBusy	Return the busy state	1.7.0	P.119
	TwofgIsGrip	Return the grip state	1.7.0	P.119
	TwofgGetErrorCode	Return the current error value	1.7.0	P.120
	TwofgGetWidth	Return the current width in the current mode	1.7.0	P.120
	TwofgGetMinWidth	Return the minimum width of the gripper in the current setup and mode	1.7.0	P.120
	TwofgGetMaxWidth	Return the maximum width of the gripper in the current setup and mode	1.7.0	P.121
	TwofgGetForce	Return the current gripping force	1.7.0	P.121
	TwofgGetOrientation	Return the finger attachment orientation	1.7.0	P.121
	TwofgGetFingerLength	Return the finger length	1.7.0	P.122
	TwofgGetFTWidth	Return the fingertip width	1.7.0	P.122
	TwofgGetMode	Return the current grip mode	1.7.0	P.122
	TwofgSetOrientation	Set the orientation of the fingers	1.7.0	P.123
	TwofgSetFingerLength	Set the finger length	1.7.0	P.123
	TwofgSetFTWidth	Set the fingertip width	1.7.0	P.123
	TwofgSwitchToExternal	Switch to external grip mode	1.7.0	P.123
	TwofgSwitchToInternal	Switch to internal grip mode	1.7.0	P.124
	TwofgGrip	Move the fingers to the desired position with the target force and speed	1.7.0	P.124
	TwofgRelease	Move the fingers to the desired position	1.7.0	P.125
	TwofgStop	Stop the finger motion in progress	1.7.0	P.125
3FG15				
	TfgIsConn	Return connection state	1.7.0	P.125
	TfgIsBusy	Return busy state	1.7.0	P.126
	TfgIsGrip	Return grip state	1.7.0	P.126

TfgIsForceGrip	Return force grip state	1.7.0	P.126
TfgIsCalibrationValid	Return calibration state	1.7.0	P.127
TfgHasError	Return error state	1.7.0	P.127
TfgHasGeneralError	Return general error state	1.7.0	P.127
TfgHasSafetyDcError	Return DC motor error state	1.7.0	P.128
TfgGetDiameterFT	Return the current diameter with fingertip offset	1.7.0	P.128
TfgGetDiameterRaw	Return the current raw diameter	1.7.0	P.128
TfgGetDiameterMin	Return the current minimum diameter	1.7.0	P.129
TfgGetDiameterMax	Return the current maximum diameter	1.7.0	P.129
TfgSetFingerPos	Set the finger position	1.7.0	P.129
TfgSetFingerLength	Set the finger length	1.7.0	P.130
TfgSetFTOffset	Set the fingertip offset	1.7.0	P.130
TfgMove	Move the fingers to the desired position	1.7.0	P.130
TfgGrip	Move the fingers to the desired position and with the target force	1.7.0	P.131
TfgFlexGrip	The fingers will move from the current diameter towards the target diameter, and do a grip with the desired force. Maximum force is 140 N when 100% is selected, and payload is maximum 8 kg.	1.7.0	P.131
TfgStop	Stop the finger motion in progress	1.7.0	P.132

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

Execute the grip command according to the value set in the parameter.

This command corresponds to HandGrip instruction of PacScript language.

Syntax HandGrip( )

Argument : None

Return value : None

Example

```
caoRob.Execute “HandGrip”
```

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

Execute the release command according to the value set in the parameter.

This command corresponds to HandRelease instruction of PacScript language.

Syntax HandRelease( )

Argument : None

Return value : None

Example

```
caoRob.Execute “HandRelease”
```

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

Get the state of the Compute Box's digital input.

This command corresponds to CBGetIO instruction of PacScript language.

Syntax CBGetIO(<lID>)

<lID> : ID of the digital input of Compute Box (VT_I4)

Return value : The state of the specified input (VT_I4)

0: Low

1: High

Example

```
Dim lState As Long
lState = caoRob.Execute(“CBGetIO”, 1)
```

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

Return connection state of RG2 or RG6.

This command corresponds to RgxIsConn instruction of PacScript language.

Syntax RgxIsConn([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Connection state (VT_I4)

0: Not connected

1: Connected

Example

```
Dim lIsConn As Long
lIsConn = caoRob.Execute(“RgxIsConn”)
```

5.2.28.5. CaoRobot::Execute("RgxIsRg2") command

Return connection state of RG2.

This command corresponds to RgxIsRg2 instruction of PacScript language.

Syntax RgxIsRg2([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Connection state with RG2 (VT_I4)

0: Not connected to RG2

1: Connected to RG2

Example

```
Dim lIsRg2 As Long
lIsRg2 = caoRob.Execute("RgxIsRg2")
```

5.2.28.6. CaoRobot::Execute("RgxIsRg6") command

Return connection state of RG6.

This command corresponds to RgxIsRg6 instruction of PacScript language.

Syntax RgxIsRg2([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Connection state with RG6 (VT_I4)

0: Not connected to RG6

1: Connected to RG6

Example

```
Dim lIsRg6 As Long
lIsRg6 = caoRob.Execute("RgxIsRg6")
```

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

Return busy state.

This command corresponds to RgxIsBusy instruction of PacScript language.

Syntax RgxIsBusy([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Busy state (VT_I4)

0: Device is idle

1: Device is busy

Example

```
Dim lIsBusy As Long
lIsBusy = caoRob.Execute("RgxIsBusy")
```

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

Return grip state.

This command corresponds to RgxIsGrip instruction of PacScript language.

Syntax RgxIsGrip([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Grip state (VT_I4)

0: No grip detected

1: Grip detected

Example

```
Dim lIsGrip As Long
lIsGrip = caoRob.Execute("RgxIsGrip")
```

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

Return safety switch status.

This command corresponds to RgxIsSafetyOn instruction of PacScript language.

Syntax RgxIsSafetyOn([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Status of safety switch (VT_I4)

0: Off

1: On

Example

```
Dim lIsSafetyOn As Long
lIsSafetyOn = caoRob.Execute("RgxIsSafetyOn")
```

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

Return current width value.

This command corresponds to RgxGetWidth instruction of PacScript language.

Syntax RgxGetWidth([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Current width [mm] (VT_R4)

Example

```
Dim fWidth As Single
fWidth = caoRob.Execute("RgxGetWidth")
```

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

Return the current absolute depth.

This command corresponds to RgxGetDepth instruction of PacScript language.

Syntax RgxGetDepth([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Current depth [mm] (VT_R4)

Example

```
Dim fDepth As Single
fDepth = caoRob.Execute("RgxGetDepth")
```

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

Return actual relative depth from beginning of last actuation.

This command corresponds to RgxGetRelativeDepth instruction of PacScript language.

Syntax RgxGetRelativeDepth([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Actual relative depth from beginning of last actuation [mm] (VT_R4)

Example

```
Dim fRelativeDepth As Single
fRelativeDepth = caoRob.Execute("RgxGetRelativeDepth")
```

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

Set the fingertip offset.

This command corresponds to RgxSetFTOffset instruction of PacScript language.

Syntax RgxSetFTOffset(<fFTOffset>)

<fFTOffset> : Fingertip offset [mm] (VT_R4)

Return value : None

Example

```
caoRob.Execute "RgxSetFTOffset", 10.0
```

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

Move the fingers to the desired position.

This command corresponds to RgxMove instruction of PacScript language.

Syntax RgxMove(<fWidth>, <fForce>, <lWaiting>)

<fWidth>	:	Target width [mm] (VT_R4)
<fForce>	:	Target force [N] (VT_R4)
<lWaiting>	:	Wait setting (VT_I4) 0: Return after command is executed (without waiting for execution to be complete) 1: Return after fingers reached the position
Return value	:	None

Example

```
caoRob.Execute “RgxMove”, Array(20.0, 30.0, 1)
```

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

Stop the fingers motion in progress.

This command corresponds to RgxStop instruction of PacScript language.

Syntax RgxStop( )

Argument	:	None
Return value	:	None

Example

```
caoRob.Execute “RgxStop”
```

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

Reset tool power and safety switch.

This command corresponds to RgxResetPower instruction of PacScript language.

Syntax RgxResetPower( )

Argument	:	None
Return value	:	None

Example

```
caoRob.Execute “RgxResetPower”
```

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

Return connection state of VG10 or VGC10.

This command corresponds to VgxIsConn instruction of PacScript language.

Syntax VgxIsConn([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Connection state (VT_I4)

0: Not connected

1: Connected

Example

```
Dim lIsConn As Long
lIsConn = caoRob.Execute("VgxIsConn")
```

5.2.28.18. CaoRobot::Execute("VgxIsVG10") command

Return connection state of VG10.

This command corresponds to VgxIsVg10 instruction of PacScript language.

Syntax VgxIsVg10([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Connection state with VG10 (VT_I4)

0: Not connected to VG10

1: Connected to VG10

Example

```
Dim lIsVg2 As Long
lIsVg2 = caoRob.Execute("VgxIsVg10")
```

5.2.28.19. CaoRobot::Execute("VgxIsVgc10") command

Return connection state of VGC10.

This command corresponds to VgxIsVgc10 instruction of PacScript language.

Syntax VgxIsVgc10([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Connection state with VGC10 (VT_I4)

0: Not connected to VGC10

1: Connected to VGC10

Example

```
Dim lIsVgc10 As Long
lIsVgc10 = caoRob.Execute("VgxIsVgc10")
```

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

Return current vacuum of channel A.

This command corresponds to VgxGetVacA instruction of PacScript language.

Syntax VgxGetVacA([<lInstance>])

<lInstance>	:	Target instance (VT_I4)
Return value	:	Current vacuum of channel A [%] (VT_R4)

Example

```
Dim fVacA As Single
fVacA = caoRob.Execute("VgxGetVacA")
```

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

Return current vacuum of channel B.

This command corresponds to VgxGetVacB instruction of PacScript language.

Syntax VgxGetVacB([<lInstance>])

<lInstance>	:	Target instance (VT_I4)
Return value	:	Current vacuum of channel B [%] (VT_R4)

Example

```
Dim fVacB As Single
fVacB = caoRob.Execute("VgxGetVacB")
```

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

Return actual current limit.

This command corresponds to VgxGetLimit instruction of PacScript language.

Syntax VgxGetLimit([<lInstance>])

<lInstance>	:	Target instance (VT_I4)
Return value	:	Current limit [mA] (VT_R4)

Example

```
Dim fLimit As Single
fLimit = caoRob.Execute("VgxGetLimit")
```

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

Set the internal current limit, which is proportional to the airflow.

This command corresponds to VgxSetLimit instruction of PacScript language.

Syntax VgxSetLimit(<fLimit>)

<fLimit> : Current limit [mA] (VT_R4)

Return value : None

Example

```
caoRob.Execute “VgxSetLimit”, 500.0
```

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

Execute gripping with the specified vacuum level.

This command corresponds to VgxGrip instruction of PacScript language.

Syntax VgxGrip(<fVacuumA>, <fVacuumB>, <lWaiting>)

<fVacuumA> : Desired vacuum level of A [%] (VT_R4)

<fVacuumB> : Desired vacuum level of B [%] (VT_R4)

<lWaiting> : Wait setting (VT_I4)

0: Return after command is executed (without waiting for execution to be complete)

1: Return after fingers reached the position

Return value : None

Example

```
caoRob.Execute “VgxGrip”, Array(30.0, 30.0, 1)
```

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

Turn off the pumps on the desired channels. The valves on the affected channels will be still closed. This is a low energy state that can keep the object gripped until several seconds.

This command corresponds to VgxIdle instruction of PacScript language.

Syntax VgxIdle(<lChannelA>, <lChannelB>)

<lChannelA> : State of channel A (VT_I4)

0: No change

1: Change to idle state

<lChannelB> : State of channel B (VT_I4)

0: No change

1: Change to idle state

Return value : None

Example

```
caoRob.Execute “VgxIdle”, Array(1, 1)
```

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

Turn off the pumps and open the valves on the affected channels to release workpiece.

This command corresponds to VgxRelease instruction of PacScript language.

Syntax VgxRelease(<lChannelA>, <lChannelB>)

<lChannelA>	:	State of channel A (VT_I4) 0: No change 1: Release vacuum of channel A
<lChannelB>	:	State of channel B (VT_I4) 0: No change 1: Release vacuum of channel A
Return value	:	None

Example

```
caoRob.Execute “VgxRelease”, Array(1, 1)
```

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

Return connection state.

This command corresponds to SgIsConn instruction of PacScript language.

Syntax SgIsConn([<lInstance>])

<lInstance>	:	Target instance (VT_I4)
Return value	:	Connection state (VT_I4) 0: Not connected 1: Connected

Example

```
Dim lIsConn As Long
lIsConn = caoRob.Execute(“SgIsConn”)
```

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

Return busy state.

This command corresponds to SgIsBusy instruction of PacScript language.

Syntax SgIsBusy([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Busy state (VT_I4)

0: Device is idle

1: Device is busy

Example

```
Dim lIsBusy As Long
lIsBusy = caoRob.Execute("SgIsBusy")
```

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

Return initialization state.

This command corresponds to SgIsInitialized instruction of PacScript language.

Syntax SgIsInitialized([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Initialization state (VT_I4)

0: SG is not initialized

1: SG is initialized

Example

```
Dim lIsInitialized As Long
lIsInitialized = caoRob.Execute("SgIsInitialized")
```

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

Return the minimum width of the current tool.

This command corresponds to SgGetOpenMin instruction of PacScript language.

Syntax SgGetOpenMin([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Minimum width [mm] (VT_R4)

Example

```
Dim fOpenMin As Single
fOpenMin = caoRob.Execute("SgGetOpenMin")
```

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

Return the maximum width of the current tool.

This command corresponds to SgGetOpenMax instruction of PacScript language.

Syntax SgGetOpenMax([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Maximum width [mm] (VT_R4)

Example

```
Dim fOpenMax As Single
fOpenMax = caoRob.Execute("SgGetOpenMax")
```

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

Return error state.

This command corresponds to SgHasError instruction of PacScript language.

Syntax SgHasError([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Error state (VT_I4)

0: No error detected

1: Error detected

Example

```
Dim lHasError As Long
lHasError = caoRob.Execute("SgHasError")
```

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

Return current depth.

This command corresponds to SgGetDepth instruction of PacScript language.

Syntax SgGetDepth([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Current depth [mm] (VT_R4)

Example

```
Dim fDepth As Single
fDepth = caoRob.Execute("SgGetDepth")
```

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

Return actual relative depth from beginning of last actuation.

This command corresponds to SgGetRelativeDepth instruction of PacScript language.

Syntax SgGetRelativeDepth([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Actual relative depth from beginning of last actuation [mm] (VT_R4)

Example

```
Dim fRelativeDepth As Single
fRelativeDepth = caoRob.Execute("SgGetRelativeDepth")
```

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

Return the depth of the currently used tool.

This command corresponds to SgGetSiliconDepth instruction of PacScript language.

Syntax SgGetSiliconDepth([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Current silicon depth [mm] (VT_R4)

Example

```
Dim fSiliconDepth As Single
fSiliconDepth = caoRob.Execute("SgGetSiliconDepth")
```

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

Return current width.

This command corresponds to SgGetWidth instruction of PacScript language.

Syntax SgGetWidth([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Current width [mm] (VT_R4)

Example

```
Dim fWidth As Single
fWidth = caoRob.Execute("SgGetWidth")
```

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

Move Soft Gripper to the home position.

This command corresponds to SgHome instruction of PacScript language.

Syntax SgHome( )

Argument : None

Return value : None

Example

```
caoRob.Execute “SgHome”
```

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

Move Soft Gripper to the home position and set the tool type to SG.

This command corresponds to SgInit instruction of PacScript language.

Syntax SgInit(<lToolID>)

<lToolID>	:	Tool type (VT_I4) 1: None 2: SG-a-H 3: SG-a-S 4: SG-b-H
Return value	:	None

Example

```
caoRob.Execute “SgInit”, 2
```

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

Grip to the desired position.

This command corresponds to SgGrip instruction of PacScript language.

Syntax SgGrip(<fWidth>, <lWaiting>)

<fWidth>	:	Target width [mm] (VT_I4)
<lWaiting>	:	Wait setting (VT_I4) 0: Return after command is executed (without waiting for execution to be complete) 1: Return after fingers reached the position
Return value	:	None

Example

```
caoRob.Execute “SgGrip”, Array(20.0, 1)
```

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

Gentle grip to the desired position and slow down when the gripper is close to the target width.

This command corresponds to SgGentleGrip instruction of PacScript language.

Syntax SgGentleGrip(<fWidth>, <lWaiting>)

<fWidth> : Target width [mm] (VT_I4)
<lWaiting> : Wait setting (VT_I4)
0: Return after command is executed (without waiting for execution to be complete)
1: Return after fingers reached the position
Return value : None

Example

```
caoRob.Execute “SgGentleGrip”, Array(20.0, 1)
```

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

Release the object by moving to the desired width.

This command corresponds to SgRelease instruction of PacScript language.

Syntax SgRelease(<fWidth>, <lWaiting>)

<fWidth> : Target width [mm] (VT_I4)
<lWaiting> : Wait setting (VT_I4)
0: Return after command is executed (without waiting for execution to be complete)
1: Return after fingers reached the position
Return value : None

Example

```
caoRob.Execute “SgRelease”, Array(60.0, 1)
```

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

Stop the motion in progress.

This command corresponds to SgStop instruction of PacScript language.

Syntax SgStop( )

Argument : None
Return value : None

Example

```
caoRob.Execute “SgStop”
```

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

Return the connection state.

This command corresponds to TwofgIsConn instruction of PacScript language.

Syntax TwofgIsConn([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Connection state (VT_I4)

0: Not connected

1: Connected

Example

```
Dim lIsConn As Long
lIsConn = caoRob.Execute("TwofgIsConn")
```

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

Return busy state.

This command corresponds to TwofgIsBusy instruction of PacScript language.

Syntax TwofgIsBusy([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Busy state (VT_I4)

0: Device is idle

1: Device is busy

Example

```
Dim lIsBusy As Long
lIsBusy = caoRob.Execute("TwofgIsBusy")
```

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

Return grip state.

This command corresponds to TwofgIsGrip instruction of PacScript language.

Syntax TwofgIsGrip([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Grip state (VT_I4)

0: No grip detected

1: Grip detected

Example

```
Dim lIsGrip As Long
```

```
lIsGrip = caoRob.Execute("TwofgIsGrip")
```

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

Return the current error value.

This command corresponds to TwofgGetErrorCode instruction of PacScript language.

Syntax TwofgGetErrorCode([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Error state (VT_I4)

0: No error

1: Calibration error

2: Linear sensor error

3: Calibration and linear sensor error

Example

```
Dim lError As Long
lError = caoRob.Execute("TwofgGetErrorCode")
```

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

Return the current width in the current mode.

This command corresponds to TwofgGetWidth instruction of PacScript language.

Syntax TwofgGetWidth([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Current width in the current mode [mm] (VT_R4)

Example

```
Dim fWidth As Single
fWidth = caoRob.Execute("TwofgGetWidth")
```

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

Return the minimum width of the gripper in the current setup and mode.

This command corresponds to TwofgGetMinWidth instruction of PacScript language.

Syntax TwofgGetMinWidth([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Minimum width [mm] (VT_R4)

Example

```
Dim fWidth As Single
fWidth = caoRob.Execute("TwofgGetMinWidth")
```

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

Return the maximum width of the gripper in the current setup and mode.

This command corresponds to TwofgGetMaxWidth instruction of PacScript language.

Syntax TwofgGetMaxWidth([<lInstance>])

<lInstance>	:	Target instance (VT_I4)
Return value	:	Maximum width [mm] (VT_R4)

Example

```
Dim fWidth As Single
fWidth = caoRob.Execute("TwofgGetMaxWidth")
```

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

Return the current gripping force.

This command corresponds to TwofgGetForce instruction of PacScript language.

Syntax TwofgGetForce([<lInstance>])

<lInstance>	:	Target instance (VT_I4)
Return value	:	Current gripping force [N] (VT_I4)

Example

```
Dim lForce As Long
lForce = caoRob.Execute("TwofgGetForce")
```

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

Return the finger attachment orientation.

This command corresponds to TwofgGetOrientation instruction of PacScript language.

Syntax TwofgGetOrientation([<lInstance>])

<lInstance>	:	Target instance (VT_I4)
Return value	:	Current orientation (VT_I4)
		1: Inwards
		2: Outwards

Example

```
Dim lOrientation As Long
lOrientation = caoRob.Execute("TwofgGetOrientation")
```

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

Return the finger length.

This command corresponds to TwofgGetFingerLength instruction of PacScript language.

Syntax TwofgGetFingerLength([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Finger length [mm] (VT_R4)

Example

```
Dim fLength As Single
fLength = caoRob.Execute(“TwofgGetFingerLength”)
```

5.2.28.53. CaoRobot::Execute("TwofgGetFTWidth") command

Return the fingertip width.

This command corresponds to TwofgGetFTWidth instruction of PacScript language.

Syntax TwofgGetFTWidth([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Fingertip width [mm] (VT_R4)

Example

```
Dim fWidth As Single
fWidth = caoRob.Execute("TwofgGetFTWidth")
```

5.2.28.54. CaoRobot::Execute("TwofgGetMode") command

Return the current grip mode.

This command corresponds to TwofgGetMode instruction of PacScript language.

Syntax TwofgGetMode([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Current grip mode (VT_I4)

1: External

2: Internal

Example

```
Dim lMode As Long
lMode = caoRob.Execute("TwofgGetMode")
```

5.2.28.55. CaoRobot::Execute("TwofgSetOrientation") command

Set the orientation of the fingers.

This command corresponds to TwofgSetOrientation instruction of PacScript language.

Syntax TwofgSetOrientation(<lOrientation>)

<lOrientation> : Orientation of the fingers (VT_I4)
1: Inwards
2: Outwards

Return value : None

Example

```
caoRob.Execute "TwofgSetOrientation", 1
```

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

Set the finger length.

This command corresponds to TwofgSetFingerLength instruction of PacScript language.

Syntax TwofgSetFingerLength(<fLength>)

<fLength> : Finger length [mm] (VT_R4)

Return value : None

Example

```
caoRob.Execute "TwofgSetFingerLength", 8.5
```

5.2.28.57. CaoRobot::Execute("TwofgSetFTWidth") command

Set the fingertip width.

This command corresponds to TwofgSetFTWidth instruction of PacScript language.

Syntax TwofgSetFTWidth(<fWidth>)

<fWidth> : Fingertip width [mm] (VT_R4)

Return value : None

Example

```
caoRob.Execute "TwofgSetFTWidth", 5.0
```

5.2.28.58. CaoRobot::Execute("TwofgSwitchToExternal") command

Switch to external grip mode.

This command corresponds to TwofgSwitchToExternal instruction of PacScript language.

Syntax TwofgSwitchToExternal( )

Argument : None

Return value : None

Example

```
caoRob.Execute "TwofgSwitchToExternal"
```

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

Switch to internal grip mode.

This command corresponds to TwofgSwitchToInternal instruction of PacScript language.

Syntax TwofgSwitchToInternal( )

Argument : None

Return value : None

Example

```
caoRob.Execute "TwofgSwitchToInternal"
```

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

Move the fingers to the desired position with the target force and speed.

This command corresponds to TwofgGrip instruction of PacScript language.

Syntax TwofgGrip(<fWidth>, <lForce>, <lSpeed>, <lWaiting>)

<fWidth> : Desired width [mm] (VT_R4)

<lForce> : Desired gripping force [N] (VT_I4)

<lSpeed> : Desired grip speed [%] (VT_I4)

<lWaiting> : Wait setting (VT_I4)

0: Return after command is executed (without waiting for execution to be complete)

1: Return after fingers reached the position

Return value : None

Example

```
caoRob.Execute "TwofgGrip", Array(35.0, 40, 80, 1)
```

5.2.28.61. CaoRobot::Execute("TwofgRelease") command

Move the fingers to the desired position.

This command corresponds to TwofgRelease instruction of PacScript language.

Syntax TwofgRelease(<fWidth>, <lWaiting>)

<fWidth>	:	Desired width [mm] (VT_R4)
<lWaiting>	:	Wait setting (VT_I4) 0: Return after command is executed (without waiting for execution to be complete) 1: Return after fingers reached the position
Return value	:	None

Example

```
caoRob.Execute "TwofgRelease", Array(70.0, 1)
```

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

Stop the finger motion in progress.

This command corresponds to TwofgStop instruction of PacScript language.

Syntax TwofgStop( )

Argument	:	None
Return value	:	None

Example

```
caoRob.Execute "TwofgStop"
```

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

Return connection state.

This command corresponds to TfgIsConn instruction of PacScript language.

Syntax TfgIsConn([<lInstance>])

<lInstance>	:	Target instance (VT_I4)
Return value	:	Connection state (VT_I4) 0: Not connected 1: Connected

Example

```
Dim lIsConn As Long
lIsConn = caoRob.Execute(“TfgIsConn”)
```

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

Return busy state.

This command corresponds to TfgIsBusy instruction of PacScript language.

Syntax TfgIsBusy([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Busy state (VT_I4)

0: Device is idle

1: Device is busy

Example

```
Dim lIsBusy As Long
lIsBusy = caoRob.Execute("TfgIsBusy")
```

5.2.28.65. CaoRobot::Execute("TfgIsGrip") command

Return grip state.

This command corresponds to TfgIsGrip instruction of PacScript language.

Syntax TfgIsGrip([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Grip state (VT_I4)

0: No grip detected

1: Grip detected

Example

```
Dim lIsGrip As Long
lIsGrip = caoRob.Execute("TfgIsGrip")
```

5.2.28.66. CaoRobot::Execute("TfgIsForceGrip") command

Return force grip state.

This command corresponds to TfgIsForceGrip instruction of PacScript language.

Syntax TfgIsForceGrip([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Force grip state (VT_I4)

0: No force grip detected

1: Force grip detected

Example

```
Dim lIsGrip As Long
lIsGrip = caoRob.Execute(“TfgIsForceGrip”)
```

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

Return calibration state.

This command corresponds to TfgIsCalibrationValid instruction of PacScript language.

Syntax TfgIsCalibrationValid([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Calibration state (VT_I4)

0: Calibration is not valid

1: Calibration is valid

Example

```
Dim lIsValid As Long
lIsValid = caoRob.Execute("TfgIsCalibrationValid")
```

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

Return error state.

This command corresponds to TfgHasError instruction of PacScript language.

Syntax TfgHasError([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Error state (VT_I4)

0: No error detected

1: Error detected

Example

```
Dim lHasError As Long
lHasError = caoRob.Execute("TfgHasError”)
```

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

Return general error state.

This command corresponds to TfgHasGeneralError instruction of PacScript language.

Syntax TfgHasGeneralError([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : General error state (VT_I4)

0: No general error detected

1: General error detected

Example

```
Dim lHasError As Long
lHasError = caoRob.Execute("TfgHasGeneralError")
```

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

Return DC motor error state.

This command corresponds to TfgHasSafetyDCError instruction of PacScript language.

Syntax TfgHasSafetyDCError([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : DC motor error state (VT_I4)

0: No error detected on DC motor

1: Error detected on DC motor

Example

```
Dim lHasError As Long
lHasError = caoRob.Execute("TfgHasSafetyDCError")
```

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

Return the current diameter with fingertip offset.

This command corresponds to TfgGetDiameterFT instruction of PacScript language.

Syntax TfgGetDiameterFT([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Current diameter with fingertip offset [mm] (VT_R4)

Example

```
Dim fDiameterFT As Single
fDiameterFT = caoRob.Execute("TfgGetDiameterFT")
```

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

Return the current raw diameter.

This command corresponds to TfgGetDiameterRaw instruction of PacScript language.

Syntax TfgGetDiameterRaw([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Current raw diameter [mm] (VT_R4)

Example

```
Dim fDiameterRaw As Single
fDiameterRaw = caoRob.Execute("TfgGetDiameterRaw")
```

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

Return the current minimum diameter.

This command corresponds to TfgGetDiameterMin instruction of PacScript language.

Syntax TfgGetDiameterMin([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Current minimum diameter [mm] (VT_R4)

Example

```
Dim fDiameterMin As Single
fDiameterMin = caoRob.Execute("TfgGetDiameterMin")
```

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

Return the current maximum diameter.

This command corresponds to TfgGetDiameterMax instruction of PacScript language.

Syntax TfgGetDiameterMax([<lInstance>])

<lInstance> : Target instance (VT_I4)

Return value : Current maximum diameter [mm] (VT_R4)

Example

```
Dim fDiameterMax As Single
fDiameterMax = caoRob.Execute("TfgGetDiameterMax")
```

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

Set the finger position.

This command corresponds to TfgSetFingerPos instruction of PacScript language.

Syntax TfgSetFingerPos(<lFingerPos>)

<lFingerPos> : Finger position (1-3) (VT_I4)

Return value : None

Example

```
caoRob.Execute "TfgSetFingerPos", 1
```

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

Set the finger length.

This command corresponds to TfgSetFingerLength instruction of PacScript language.

Syntax TfgSetFingerLength(<fFingerLength>)

<fFingerLength> : Finger length [mm] (VT_R4)

Return value : None

Example

```
caoRob.Execute "TfgSetFingerLength", 49.0
```

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

Set the fingertip offset.

This command corresponds to TfgSetFTOffset instruction of PacScript language.

Syntax TfgSetFTOffset(<fFTOffset>)

<fFTOffset> : Fingertip offset [mm] (VT_R4)

Return value : None

Example

```
caoRob.Execute "TfgSetFTOffset", 13.5
```

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

Move the fingers to the desired position.

This command corresponds to TfgMove instruction of PacScript language.

Syntax TfgMove(<fDiameter>, <lWaiting>)

<fDiameter> : Desired raw diameter [mm] (VT_R4)

<lWaiting> : Wait setting (VT_I4)

0: Return after command is executed (without waiting for execution to be complete)

1: Return after fingers reached the position

Return value : None

Example

```
caoRob.Execute "TfgMove", Array(20.0, 1)
```

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

Move the fingers to the desired position and with the target force.

This command corresponds to TfgGrip instruction of PacScript language.

Syntax TfgGrip(<fDiameter>, <fForce>, <lGripType>, <lWaiting>)

<fDiameter>	:	Desired diameter with fingertip offset [mm] (VT_R4)
<fForce>	:	Desired gripping force [%] (VT_R4)
<lGripType>	:	Grip type (VT_I4) 1: Internal grip 2: External grip
<lWaiting>	:	Wait setting (VT_I4) 0: Return after command is executed (without waiting for execution to be complete) 1: Return after fingers reached the position
Return value	:	None

Example

```
caoRob.Execute "TfgGrip", Array(20.0, 20.0, 2, 1)
```

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

The fingers will move from the current diameter towards the target diameter, and do a grip with the desired force. Maximum force is 140 N when 100% is selected, and payload is maximum 8 kg.

This command corresponds to TfgFlexGrip instruction of PacScript language.

Syntax TfgFlexGrip(<fDiameter>, <fForce>, <lGripType>, <lWaiting>)

<fDiameter>	:	Desired diameter with fingertip offset [mm] (VT_R4)
<fForce>	:	Desired gripping force [%] (VT_R4)
<lGripType>	:	Grip type (VT_I4) 1: Internal grip 2: External grip
<lWaiting>	:	Wait setting (VT_I4) 0: Return after command is executed (without waiting for execution to be complete) 1: Return after fingers reached the position
Return value	:	None

Example

```
caoRob.Execute "TfgFlexGrip", Array(20.0, 20.0, 2, 1)
```

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

Stop the finger motion in progress.

This command corresponds to TfgStop instruction of PacScript language.

Syntax TfgStop( )

Argument	:	None
Return value	:	None

Example

```
caoRob.Execute "TfgStop"
```

5.2.29. CaoTask::AddVariable method

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

Refer to Table 5-12 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
bstrOpt	:	[in]	Option (not used)

5.2.32. CaoTask::Stop method

Stop the PAC program that supports the object.

The following shows the argument specifications of Stop.

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

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

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

5.2.33. CaoTask::Execute method

Execute the command.

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

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

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

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

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

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

Example

```
Dim vRes As Variant
Dim caoTsk As CaoTask

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

The list shows available commands.

Table 5-8 Command list of CaoTask::Execute

Category	Command name	Function	
Task status			
	GetStatus	Get the task status.	P.134
Priority			
	GetThreadPriority	Get the priority.	P.134
	SetThreadPriority	Set the priority.	P.134

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 is not exist.
		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".

5.3. Variable list

5.3.1. Controller class

Table 5-9 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.	√	√
SafetyP	VT_ARRAY | VT_R4	SafetyP type variable. The variable number is specified after the variable name.	√	√

IARRAY	VT_ARRAY |VT_I4	Handles an I type variable as an I4 type array. "IARRAY*" (* is a value) represents a variable name. A variable is defined by specifying the start number and the array size. After that, it is accessed as an array variable. See the example below:	√	√
FARRAY	VT_ARRAY |VT_R4	Handles an F type variable as a R4 type array. "FARRAY*" (* is a value) represents a variable name. A variable is defined by specifying the start number and the array size. After that, it is accessed as an array variable. See the example below:	√	√
DARRAY	VT_ARRAY |VT_R8	Handles a D type variable as a R8 type array. "DARRAY*" (* is a value) represents a variable name. A variable is defined by specifying the start number and the array size. After that, it is accessed as an array variable. See the example below:	√	√
VARRAY	(VT_ARRAY| VT_R4)* array size	Handles a V type variable as an array. V type variable is specified by a R4 type array. "VARRAY*" (* is a value) represents a variable name. A variable is defined by specifying the start number and the array size. After that, it is accessed as an array variable. See the example below:	√	√
PARRAY	(VT_ARRAY| VT_R4)* array size	Handles a P type variable as an array. P type variable is specified by a R4 type array. "PARRAY*" (* is a value) represents a variable name. A variable is defined by specifying the start number and the array size. After that, it is accessed as an array variable. See the example below:	√	√
JARRAY	(VT_ARRAY| VT_R4)* array size	Handles a J type variable as an array. J type variable is specified by a R4 type array. "JARRAY*" (* is a value) represents a variable name. A variable is defined by specifying the start number and the array size. After that, it is accessed as an array variable. See the example below:	√	√
TARRAY	(VT_ARRAY| VT_R4)* array size	Handles a T type variable as an array. T type variable is specified by a R4 type array. "TARRAY*" (* is a value) represents a variable name. A variable is defined by specifying the start number and the array size. After that, it is accessed as an array variable. See the example below:	√	√
SARRAY	VT_ARRAY |VT_BSTR	Handles an S type variable as a BSTR type array. "SARRAY*" (* is a value) represents a variable name. A variable is defined by specifying the start number and the array size. After that, it is accessed as an array variable. See the example below:	√	√
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.	√	√
IOARRAY	VT_ARRAY |VT_BOOL	Handles an IO as a BOOL type array. "IOARRAY*" (* is a value) represents a variable name. A variable is defined by specifying the start number and the array size. After that, it is accessed as an array variable. See the example below:	√	√
IOBARRAY	VT_ARRAY |VT_I1	Handles an IO as an I1 type array. "IOBARRAY*" (* is a value) represents a variable name. A variable is defined by specifying the start number and the array size. After that, it is accessed as an array variable. See the example below:	√	√
IOWARRAY	VT_ARRAY |VT_I2	Handles an IO as an I2 type array. "IOWARRAY*" (* is a value) represents a variable name. A variable is defined by specifying the start number and the array size. After that, it is accessed as an array variable. See the example below:	√	√
IODARRAY	VT_ARRAY |VT_I4	Handles an IO as an I4 type array. "IODARRAY*" (* is a value) represents a variable name. A variable is defined by specifying the start number and the array size. After that, it is accessed as an array variable. See the example below:	√	√
IOFARRAY	VT_ARRAY |VT_R4	Handles an IO as an R4 type array. "IOFARRAY*" (* is a value) represents a variable name. A variable is defined by specifying the start number and the array size. After that, it is accessed as an array variable. See the example below:	√	√

Example

```
'IARRAY
' A variable which is handled as the array of three elements from the 0th to the second element of the variable definition of I type variable
Dim IArray0 as CaoVariable
Set IArray0 = caoCtrl.AddVariable("IArray0", "StartNo = 0, ArraySize = 3")
IArray0 = Array(1,2,3)    I0=1, I1=2, and I2=3.
```

```
'PARRAY
' A variable which is handled as the array of two elements from the first to the second element of the variable definition of P type variable.
Dim PArray0 as CaoVariable
Set PArray0 = caoCtrl.AddVariable("PArray0", "StartNo = 1, ArraySize = 2")
PArray0 = Array(Array(1,2,3,4,5,6,-1), Array(1,2,3,4,5,6,-1))

'IOARRAY
' Variable definition VT_BOOL type, starting from IO number 128, array of three elements.
Dim IOArray0 As CaoVariable
Set IOArray0 = g_caoCtrl.AddVariable("IOArray0", "StartIoNo = 128, ArraySize = 3")
' Value setting
IOArray0 = Array(True, False, True)
' Value obtainment
vntVal = IOArray0

'IOFARRAY
' Variable definition VT_R4 type, starting from IO number 160, array of four elements.
Dim IOFArray0 As CaoVariable
Set IOFArray0 = g_caoCtrl.AddVariable("IOFArray0", "StartIoNo = 160, ArraySize = 4")
'値設定
IOFArray0 = Array(123.45, 234.56, 0.88, 345.67)
'値取得
vntVal = IOFArray0
```

Table 5-10 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_SAFETYP_LEN	VT_I4	Size of global SafetyP type variable	√	-
@VAR_IO_LEN	VT_I4	I/O point number (number of bits)	√	-
@MODE	VT_I4	1: manual, 2: teach check, 3: auto	√	√[2]
@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.	√	-
@ENABLE_SW	VT_BOOL	Enable switch status	√	-

[2] This command is available only for VRC (Virtual Robot Controller).

@AUTO_ENABLE	VT_BOOL	Auto enable status	√	-
@MAKER_NAME	VT_BSTR	"DENSO CORPORATION"	√	-
@TYPE	VT_BSTR	"RC9 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 5-11 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	√	√
@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	√	√
Base*	VT_ARRAY | VT_R8	Enter the definition of Base1 coordinate. X, Y, Z, RX, RY, RZ, Attribute For Attribute, enter 0.	√	√

5.3.3. Task class

Table 5-12 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 task. [0] = Line number [1] = Running 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 5-13 File class system variable list

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

5.4. Event list

Receive events that notified the error or the status of the controller via OnMessage.

Table 5-14 OnMessage event list

Event	Event Number	Value		Explanation
		Data type	Value	
Occurrence of an error	3	VT_I4	Error code	Also occurs error of level 0
Emergency stop	5	VT_I4	ON:-1 OFF:0	-
Protection stop	6	VT_I4	ON:-1 OFF:0	-
Auto enable	7	VT_I4	ON:-1 OFF:0	-
Mode switching	9	VT_I4	MANUAL:1 AUTO:3	-

[Notes]

Other events of the above are not supported.

Example

```
Dim g_eng As CaoEngine
Dim WithEvents g_ctrl As CaoController
Dim lEventNo As Long
Dim vntVal As Variant

Private Sub Form_Load()
    Set g_eng = New CaoEngine
    ' Change the IP address for your controler
    Set g_ctrl = g_eng.Workspaces(0).AddController("RC8", "CaoProv.DENSO.RC8", "",
"Server=192.168.0.1")
End Sub

Private Sub Form_Unload(Cancel As Integer)
    ' Release CaoController
    g_eng.Workspaces(0).Controllers.Remove g_ctrl.Index
    Set g_ctrl = Nothing
    ' Release CaoEngine
    Set g_eng = Nothing
End Sub

Private Sub g_ctrl_OnMessage(ByVal pICaoMess As CAOLib.ICaoMessage)
    lEventNo = pICaoMess.Number
    vntVal = pICaoMess.Value
End Sub
```

Appendix A. POSEDATA Type Definition

In 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[3] "[<Pass>] [<Variable type>]<Index> [<ExJnt>]" or "[<Pass>] [<Variable type>](<Element 1>,<Element 2>,...) [<ExJnt>]"
- VT_R4|VT_ARRAY[4] <Raw value> = (<Element 1:VT_R4>,<Element 2:VT_R4>,...)[5]
- 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, SafetyP 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	SafetyP	None
VT_BSTR	"P"	"T"	"J"	"V"	"SafetyP"	""
VT_I4	0	1	2	3	4	-1

<Index> : <Value:VT_R4>

<Element n> : <Value:VT_R4>

[3] In case of VT_BSTR, more than one POSEDATA type separated by commas can be specified.
[4] Because <Variable type> and <Pass> cannot be specified, variable type is treated as P type and pass type is treated as @0 by default.
[5] Because <Variable type> and <Pass> cannot be specified, variable type is treated as P type and pass type is treated as @0 by default.

<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 expressed 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 A.1. Examples

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

ex1. T200

String	"T200"
VARIANT type array (Variable type specified by string)	Array(200,"T")[6]
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)

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

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

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

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

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

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

ex1. @P V20

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

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

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

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

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

ex1. @P 1

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

ex2. @P 1.56

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

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

ex1. @P (1, 30.0)

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

Other examples

ex1. V1,V2,V3

(Rotation plane for CaoRobot::Rotate())

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

ex2. APPROACH P,P70, 60, NEXT

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

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

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

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

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

[Notes]

If you specify immediate values directly by POSEDATA type by VT_R4|VT_ARRAY form, the values will treated as 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>,...)
```