PV provider
Panasonic

Version 1.0.6

User's guide

February 3, 2021

[Remarks]

[Revision history]

Version	Date	Content
1.0.0.0	2011-12-8	First edition.
1.0.0.1	2012-1-16	Correct Command name
1.0.0	2012-7-17	Document versioning rules was changed.
1.0.1	2012-09-24	@ResultDisable was added.
1.0.2	2012-11-07	“RunManual” command was corrected.
1.0.3	2013-01-25	“MemoryRead” and “MemoryWrite” commands were correspondence to PV500.
1.0.4	2015-07-28	Original error (E_COMMAND_EXECUTING) was added. Asynchronous commands were added. Calibration commands were added.(PV260-compatible) Timeout setting/obtainment commands were added.
1.0.5	2017-04-27	MyIP option was added to Option string of CaoWorkspace: : AddController
	2020-11-13	Renamed to PV provider.
1.0.6	2021-02-03	CalibrationStart and CalibrationStartAsync commands was supported for PV260 Ver1.10. (Camera No. argument was added)

Contents

1. Introduction 7

2. Overview of provider 8

2.1. Overview 8
2.2. Method and Property 8
2.2.1. CaoWorkspace: : AddController method 8
2.2.1.1. Conn option 9
2.2.2. CaoController: : Execute method 9
2.2.3. CaoController: : AddVariable method 10
2.2.4. CaoVariable: : put_Value property 10
2.2.5. CaoVariable: : get_Value property 10
2.3. Variable list 10
2.3.1. Controller class 10
2.4. Error Code 11

3. Command reference 12

3.1. General Communication 15
3.1.1. CaoController: : Execute(“Start") command 15
3.1.1.1. Execution Mode is “Execute All” or “Automatic Switch” with Common trigger used 15
3.1.1.2. Execution Mode is “User-Defined” with Common trigger used 16
3.1.2. CaoController: : Execute(“Restart") command 16
3.1.2.1. Execution Mode is "All executions" or "User Defined" 16
3.1.2.2. Execution Mode is "User Defined" 17
3.1.3. CaoController: : Execute (“Xtype") command 17
3.1.4. CaoController: : Execute (“MemoryWrite") command 18
3.1.5. CaoController: : Execute (“CFWrite") command 18
3.1.6. CaoController: : Execute (“MemoryRead") command 18
3.1.7. CaoController: : Execute (“CFRead") command 19
3.1.8. CaoController: : Execute (“CancelData") command 19
3.1.9. CaoController: : Execute (“SDSave") command 19
3.1.10. CaoController: : Execute (“SDReset") command 20
3.1.11. CaoController: : Execute (“PrintScreen") command 20
3.1.12. CaoController: : Execute (“Quit") command 20
3.1.13. CaoController: : Execute (“RunManual") command 21
3.1.14. CaoController: : Execute (“ErrorReset") command 21

3.1.15. CaoController: : Execute (“Cancel") command 22
3.1.16. CaoController: : Execute (“KeyEmulator") command 22
3.1.17. CaoController: : Execute (“Bstop") command 23
3.1.18. CaoController: : Execute (“Bconfirm") command 24
3.1.19. CaoController: : Execute (“LayoutChange") command 24
3.1.20. CaoController: : Execute (“AgainTemplate") command 24
3.1.20.1. Operation for “Area Display” is “No” 24
3.1.21. CaoController: : Execute (“ParameterRead") command 25
3.1.22. CaoController: : Execute (“ParameterReadPair") command 25
3.1.23. CaoController: : Execute (“ParameterWrite") command 26
3.1.24. CaoController: : Execute (“ParameterWritePair") command 26
3.2. PV260 Robot calibration command (Synchronous) 27
3.2.1. CaoController: : Execute(“SetPoint”) command 27
3.2.2. CaoController: : Execute(“Calibrate”) command 27
3.2.2.1. Execution Mode is “Execute All” 28
3.2.2.2. Execution Mode is “User Defined” 28
3.2.3. CaoController: : Execute(“ReCalibrate”) command 28
3.2.3.1. Execution Mode is “Execute All” 29
3.2.3.2. Execution Mode is “User Defined” 29
3.2.4. CaoController: : Execute(“CalibrationStart”) command 30
3.2.5. CaoController: : Execute(“CalibrationEnd”) command 30
3.2.6. CaoController: : Execute(“WorkSet”) command 31
3.2.7. CaoController: : Execute(“WorkReset”) command 31
3.2.8. CaoController: : Execute(“WorkResetEnd”) command 31
3.2.9. CaoController: : Execute(“MoveEnd”) command 32
3.2.10. CaoController: : Execute(“GetTeachPoint”) command 32
3.2.11. CaoController: : Execute(“GetMovePoint”) command 32
3.3. Original command (synchronous) 33
3.3.1. CaoController: : Execute (“Raw") command 33
3.3.2. CaoController: : Execute(“SetTimeout”) command 33
3.3.3. CaoController: : Execute(“GetTimeout”) command 34
3.4. General communication command (Asynchronous) 34
3.4.1. CaoController: : Execute(“StartAsync”) command 34
3.4.1.1. Execution Mode is “Execute All” or “Automatic Switch”, with Common trigger used 34
3.4.1.2. Execution Mode is “User Defined ”, with Common trigger used 35
3.4.2. CaoController: : Execute(“ReStartAsync”) command 35
3.4.2.1. Execution Mode is “Execute All” or “Automatic Switch” 35

3.4.2.2. Execution Mode is “User Defined” 36
3.4.3. CaoController: : Execute(“XTypeAsync”) command 36
3.4.4. CaoController: : Execute(“MemoryWriteAsync”) command 37
3.4.5. CaoController: : Execute (“CFWriteAsync”) command 37
3.4.6. CaoController: : Execute (“MemoryReadAsync”) command 38
3.4.7. CaoController: : Execute (“CFReadAsync”) command 39
3.4.8. CaoController: : Execute (“CancelDataAsync”) command 39
3.4.9. CaoController: : Execute (“SDSaveAsync”) command 40
3.4.10. CaoController: : Execute (“SDResetAsync”) command 40
3.4.11. CaoController: : Execute (“PrintScreenAsync”) command 41
3.4.12. CaoController: : Execute (“QuitAsync”) command 41
3.4.13. CaoController: : Execute (“RunManualAsync”) command 42
3.4.14. CaoController: : Execute (“ErrorResetAsync”) command 42
3.4.15. CaoController: : Execute (“CancelAsync”) command 43
3.4.16. CaoController: : Execute (“KeyEmulatorAsync”) command 43
3.4.17. CaoController: : Execute (“BstopAsync”) command 44
3.4.18. CaoController: : Execute (“BconfirmAsync”) command 44
3.4.19. CaoController: : Execute (“LayOutChangeAsync”) command 45
3.4.20. CaoController: : Execute (“AgainTemplateAsync”) command 45
3.4.20.1. Operation for “Area Display” is “No” 45
3.4.21. CaoController: : Execute (“ParameterReadAsync”) command 46
3.4.22. CaoController: : Execute (“ParameterReadPairAsync”) command 47
3.4.23. CaoController: : Execute (“ParameterWriteAsync”) command 47
3.4.24. CaoController: : Execute (“ParameterWritePairAsync”) command 48
3.5. PV260 Robot calibration command (Asynchronous) 49
3.5.1. CaoController: : Execute (“SetPointAsync”) command 49
3.5.2. CaoController: : Execute (“CalibrateAsync”) command 49
3.5.2.1. Execution Mode is “Execute All” 50
3.5.2.2. Execution Mode is “User Defined” 50
3.5.3. CaoController: : Execute (“ReCalibrateAsync”) command 51
3.5.3.1. Execution Mode is “Execute All” 51
3.5.3.2. Execution Mode is “User Defined” 51
3.5.4. CaoController: : Execute (“CalibrationStartAsync”) command 52
3.5.5. CaoController: : Execute (“CalibrationEndAsync”) command 53
3.5.6. CaoController: : Execute (“WorkSetAsync”) command 53
3.5.7. CaoController: : Execute (“WorkResetAsync”) command 54
3.5.8. CaoController: : Execute(“WorkResetEndAsync”) command 54

3.5.9. CaoController: : Execute (“MoveEndAsync”) command 55
3.5.10. CaoController: : Execute (“GetTeachPointAsync”) command 55
3.5.11. CaoController: : Execute(“GetMovePointAsync”) command 56
3.6. General communication command (Asynchronous) 56
3.6.1. CaoController: : Execute(“RawAsync”) command 56
3.6.2. CaoController: : Execute(“GetResult”) command 57
4. PV260 Robot calibration function..... 59
4.1. PV260 configuration..... 59
4.2. Calibrate, Recalibrate command sequence..... 60
4.3. CalibrationStart command sequence 61
4.4. WorkReset command sequence..... 64
Appendix A. Command corresponding table..... 67

1. Introduction

This book is a user's guide of the PV provider that is the CAO provider for the IMAGECHECKER PV series that is the vision system manufactured by Panasonic.

The PV provider sends and receives commands and acquires images with Ethernet connection. The communication supports Ethernet (TCP/IP) connection (RS232-C connection is currently unsupported.) The PV provider is compatible with IMAGECHECKER PV200 and PV500 made by Panasonic.

2. Overview of provider

2.1. Overview

PV provider offers the CaoController: : Execute-method as a command execution method.

CaoController: : Execute can send and receive a common command to each port to control PV series via Ethernet interface according to a general-purpose communication.

Table2-1 PV provider

File name	CaoProvPV.dll
ProgID	CaoProv.Panasonic.PV
Registry registration	regsvr32 CaoProvPV.dll
Remove registry registration	regsvr32 /u CaoProvPV.dll

2.2. Method and Property

2.2.1. CaoWorkspace: : AddController method

At AddController, PV provider refers communication connection parameter and connects communication. Options specify the communication method and time-out period.

PV provider supports Ethernet connection.

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

bstrCtrlName	:	[in]	Controller name (arbitral name).
bstrProvName	:	[in]	Provider name (Fixed to " CaoProv.Panasonic.PV")
bstrPcName	:	[in]	Provider execution computer name
bstrOption	:	[in]	Option character string

The following is a list of option string items.

Table2-2 Option string of CaoWorkspace: : AddController

Option	Description
Conn =<Connection parameter >	Mandatory. Specify the communication form and connection parameter.

Timeout[=< Time-out period>]	Specify the communication time-out period by millisecond (default: 500) .
PV260[=<Connection parameter >]	Specify this parameter if you use a robot calibration-related command of PV260. 0 : Do not use a robot calibration-related command (default) 1 : Use a robot calibration-related command.
MyIP=[<Local IP address>[:Local port No]]	When using several NICs, NIC can be selected by specifying IP address at this option. NIC will be selected automatically when omitting IP address. Error will be returned when the IP address that is not allocated to a local machine is specified. Local port No. is 0 when omitting IP address.

2.2.1.1. Conn option

The following is the communication parameter string for Conn option. You can omit items enclosed by square blankets ("[]"). Underlined part shows the default value when the option is not specified.

- Ethernet device
  "eth: <IP Address>"
  <IP Address> : IP address of connected PV series.
  Example: "127.0.0.1", "10.5.5.100"

2.2.2. CaoController: : Execute method

Send and receive common commands to each port to control the PV series. Enter a command name in the first argument and the command parameters in the second argument. Please refer to Chapter 3 Command reference for details of each command.

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

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

The return value of the state code from the PV series at the time of Execute method execution is returned as HRESULT.

If the process ends normally, S_OK is returned.

If the process ends abnormally, 0x80100000 + return value (hexadecimal) is returned.

Example: When you execute Start.

Hr = 0x801000C8: Cannot execute due to operation stop state (STOP).

Refer to the PV series manual of Panasonic for the content of the error.

2.2.3. CaoController: : AddVariable method

Create variables used for acquiring images Please refer to Table2-3 Controller class system variable list for the acquisition of the image.

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

bstrVariableName : [in] Variable name

bstrOption : [in] Option character string

Example

```
Dim objBITMAP as object
Dim vntBITMAP as variant
objBITMAP = caoCtrl.AddVariable("@BITMAP", "")
vntBITMAP = objBITMAP.Value

VntBITMAP:  Binary data of camera image
```

2.2.4. CaoVariable: : put_Value property

Variable class currently does not support Put_value property.

2.2.5. CaoVariable: : get_Value property

Images can be acquired in the format of Table2-3 Controller class system variables list.

2.3. Variable list

2.3.1. Controller class

Table2-3 Controller class system variable list

Variable identifier	Data type	Description
@BITMAP	VT_UI1|VT_ARRAY	Get the camera image in the BITMAP format.
@BITMAP_MONITOR	VT_UI1|VT_ARRAY	Get the monitor image of the PV series in the

		BITMAP format.
@ResultDisable	VT_BOOL	True : Do not get results. False : Get results. (default)

- Color camera image acquisition is currently unimplemented.

2.4. Error Code

In PV provider, the following original error code is defined. For about ORiN2 common errors, refer to the error code section on “ORiN2 Programming Guide”.

Table 2-4 Specific error code list

Error	Error number	Description
E_COMMAND_EXECUTING	0x80F00000	Another command is executed during a command execution.
E_COMMAND_CONNECTED	0x80F00001	A command was executed to an unconnected communication port

3. Command reference

This chapter explains each command of the CaoController: : Execute method. For detailed operation of each command, please refer to the general-purpose communication command of the PV series manual made by Panasonic.

For about the executable mode of each command, please refer to the general-purpose communication command list of the PV series manual made by Panasonic.

Table 3-1 CaoController: : Execute command list

PV series command	Command	Description	Reference
Commands for General Communication (Synchronous)			
%S	Start	Inspection start (“Execute All” or “Automatic Switch”)	P.15
		Inspection start (“User-Defined”)	P.16
%R	Restart	Re-inspection execution (Inspect without taking picture by a present memory image.)	P.16
%X	Xtype	Product type switch	P.17
%MW	MemoryWrite	Write the setting data On-board memory	P.18
%CW	CFWrite	Write the setting data SD card memory	P.18
%MR	MemoryRead	Read the setting data On-board memory.	P.18
%CR	CFRead	Read the setting data SD card memory	P.19
%CD	CancelData	Stop Writing/Reading the setting data (cancel)	P.19
%SS	SDSave	Save stored image memory (SD memory card)	P.19
%SR	SDReset	Delete stored image memory	P.20
%PS	PrintScreen	Print screen	P.20
%Q	Quit	Reset statistics	P.20
%RM	RunManual	Switch Run/Stop status (RUN/STOP)	P.21
%E	ErrorReset	Reset the error signal	P.21
%CC	Cancel	Abort the Inspection/Processing (cancel various operation)	P.22
%K	KeyEmulator	Key emulating	P.22
%BS	Bstop	Keypad operation Refuse/Permit	P.23
%BC	Bconfirm	Confirm the keypad operation permission state	P.24
%I	LayOutChange	Change the layout	P.24
%A	AgainTemplate	Re-registration the template	P.24

%PR	ParameterRead	Read the parameter	P.25
%PRP	ParameterReadPair	Readout the parameter pair (various bound pair values etc.)	P.25
%PW	ParameterWrite	Change the Parameter	P.26
%PWP	ParameterWritePair	Change the Parameter pair (various bound pair values etc.)	P.26
PV260 Robot calibration command (Synchronous)			
%P=	SetPoint	Robot coordinates acknowledged	P.27
%CA	Calibrate	Measurement start command (Execute All)	P.28
		Measurement start command (User Defined)	P.28
%RCA	ReCalibrate	Re-measurement start command (Execute All)	P.29
		Re-measurement start command (User Defined)	P.29
%CAS	CalibrationStart	Auto calibration setting start	P.30
―	CalibrationEnd	Auto calibration setting completion notification reception (CalibrationStart-related command)	P.30
%WCS	WorkSet	Work detection base position reregistering	P.31
%WRS	WorkReset	Work detection base position reregistering start	P.31
―	WorkResetEnd	Work detection base position reregistration completion notification reception (WorkReset-related command)	P.31
%MVE	MoveEnd	Movement completion notification	P.32
%TCD	GetTeachPoint	Teaching coordinate request	P.32
―	GetMovePoint	Robot coordinates obtainment (CalibrationStart, WorkReset-related command)	P.32
Original command			
―	Raw	Send/Receive the command message	P.33
―	SetTimeout	Set the communication timeout period	P.33
―	GetTimeout	Obtain the communication timeout period	P.34
General communication command (Asynchronous)			
%S	StartAsync	Inspection execution asynchronously (Execute All/Automatic Switch)	P.34
		Inspection execution asynchronously (User Defined)	P.34
%R	ReStartAsync	Start reinspection asynchronously.(Execute All /Automatic Switch)	P.35
		Start reinspection asynchronously. (User	P.36

		Defined)	
%X	XTypeAsync	Switch product type asynchronously.	P.36
%MW	MemoryWriteAsync	Write the setting data asynchronously. On-board memory	P.37
%CW	CFWriteAsync	Write the setting data asynchronously. SD memory card	P.37
%MR	MemoryReadAsync	Read the setting data asynchronously. On-board memory	P.38
%CR	CFReadAsync	Read the setting data asynchronously. SD memory card	P.39
%CD	CancelDataAsync	Abort writing / reading the setting data asynchronously. /(Cancel)	P.39
%SS	SDSaveAsync	Save stored image memory asynchronously (SD memory card)	P.40
%SR	SDResetAsync	Delete stored image memory asynchronously	P.40
%PS	PrintScreenAsync	Print screen asynchronously.	P.41
%Q	QuitAsync	Reset statistics asynchronously.	P.41
%RM	RunManualAsync	Switch RUN/STOP status asynchronously.	P.42
%E	ErrorResetAsync	Reset an error signal asynchronously.	P.42
%CC	CancelAsync	Cancel Inspection/Processing asynchronously (Cancel various operations)	P.43
%K	KeyEmulatorAsync	Key emulating asynchronously	P.43
%BS	BstopAsync	Asynchronous keypad operation Refuse/Permit	P.44
%BC	BconfirmAsync	Asynchronous keypad operation Confirm status	P.44
%I	LayOutChangeAsync	Change the layout asynchronously	P.45
%A	AgainTemplateAsync	Asynchronous template reregistration	P.45
%PR	ParameterReadAsync	Asynchronous parameter reading	P.46
%PRP	ParameterReadPairAsync	Read parameter pairs asynchronously (Maximum / minimum values)	P.47
%PW	ParameterWriteAsync	Change parameters asynchronously	P.47
%PWP	ParameterWritePairAsync	Change parameter pairs asynchronously (Maximum/ minimum values)	P.48
PV260 Robot calibration command (Asynchronous)			
%P=	SetPointAsync	Asynchronous robot coordinates acknowledgment.	P.49
%CA	CalibrateAsync	Asynchronous measurement start command	P.50

		(Execute All)	
		Asynchronous measurement start command (User Defined)	P.50
%RCA	ReCalibrateAsync	Asynchronous re-measurement start command (Execute All)	P.51
		Asynchronous re-measurement start command (User Defined)	P.51
%CAS	CalibrationStartAsync	Asynchronous auto calibration setting start	P.52
－	CalibrationEndAsync	Asynchrouns auto calibration setting completion notification reception (CalibrationStart-related command)	P.53
%WCS	WorkSetAsync	Asynchronous work detection base position reregistration	P.53
%WRS	WorkResetAsync	Asynchronous work detection base position reregistration start	P.54
－	WorkResetEndAsync	Asynchronous work detection base position reregistration completion notification reception (WorkReset-related command)	P.54
%MVE	MoveEndAsync	Asynchronous movement completion notification	P.55
%TCD	GetTeachPointAsync	Asynchronous teachin coordinate request	P.55
－	GetMovePointAsync	Asynchronous robot coordinates obtainment (CalibrationStart, WorkReset-related command)	P.56
Original command (Asynchronous)			
－	RawAsync	Asynchronous command message sending	P.56
－	GetResult	Obtain the return value of Asynchronous command	P.57

3.1. General Communication

3.1.1. CaoController: : Execute(“Start") command

Execute inspection. The syntax is different from the execution mode, “Execute All”, “Automatic Switch” or “User-Defined”. The image processing result returns a value set by "General-purpose result output" of the PV series by the character string.

3.1.1.1. Execution Mode is “Execute All” or “Automatic Switch” with Common trigger used

Syntax Start ( )

Argument : None

Return value : [Out] Image processing result (VT_BSTR)

The following example shows how to execute inspection.

Example

```
Dim bstrResult as string
bstrResult = caoCtrl.Execute("Start")

' bstrResult:  Image processing result
```

3.1.1.2. Execution Mode is “User-Defined” with Common trigger used

Syntax Start (<lBlockNum>)

< lBlockNum > : [in] Block number to be executed (VT_I4) (0-9)

Return value : [out] Image processing result (VT_BSTR)

The following example shows how to designate the Block number 1 then, execute the inspection.

Example

```
Dim lBlockNum as long
Dim bstrResult as string

lBlockNum = 1

bstrResult = caoCtrl.Execute("Start", lBlockNum)

' bstrResult:  Image processing result
```

3.1.2. CaoController: : Execute(“Restart") command

Execute inspection without importing images (re-inspection). Syntaxes are different from the execution mode, “All execution”, “Automatic Switch” or “User Defined”.

3.1.2.1. Execution Mode is "All executions" or "User Defined"

Syntax Restart ( )

Argument : None

Return value : [out] Image processing result (VT_BSTR)

The following example shows how to execute re-inspection.

Example

```
Dim bstrResult as string

bstrResult = caoCtrl.Execute "Restart"

' bstrResult:  Image processing result
```

3.1.2.2. Execution Mode is "User Defined"

Syntax Restart (< lBlockNum >)

< lBlockNum > : [in] Block number to be executed (VT_I4) (0-9)

Return value : [out] Image processing result (VT_BSTR)

The following example shows how to designate the Block number 1 then, execute re-inspection.

Example

```
Dim lBlockNum as long
Dim bstrResult as string

lBlockNum = 1

bstrResult = caoCtrl.Execute "Restart", lBlockNum

' bstrResult:  Image processing result
```

3.1.3. CaoController: : Execute (“Xtype") command

Switch a product type.

Syntax Xtype (<lProductNum>)

< lProductNum > : [in] Product type number (VT_I4) (0-255)

Return value : None

The following example shows how to switch the type number to 100.

Example

```
Dim lProductNum as long

lProductNum = 100

 caoCtrl.Execute "Xtype", lProductNum
```

3.1.4. CaoController: : Execute (“MemoryWrite") command

Write the setting data into the PV series on-board memory.

Syntax MemoryWrite ([< lWriteNum >])

< lWriteNum >	:	PV200	None
		PV500	[in]Area Number(VT_I4) (0 - 99)
Return value	:	None	

The following example shows how to write the setting data into the PV series on-board memory. .

Example

```
caoCtrl.Execute "MemoryWrite"
```

3.1.5. CaoController: : Execute (“CFWrite") command

Write the setting data into SD memory cards

Syntax CFWrite (< lWriteNum >)

< lWriteNum >	:	[in] Area number where SD memory card is stored. (VT_I4) (0-99)
Return value	:	None

The following example shows how to save the setting number into saving area No.10 of an SD memory card.

Example

```
Dim lWriteNum as long
 lWriteNum = 10
caoCtrl.Execute "CFWrite", lWriteNum
```

3.1.6. CaoController: : Execute (“MemoryRead") command

Read the setting data from the on-board memory of PV series.

Syntax MemoryRead ([<lReadNum>])

<lReadNum>	:	PV200	None
		PV500	[in]Area Number(VT_I4) (0 - 99)
Return value	:	None	

The following example shows how to read out the setting data from the on-board memory.

Example

```
caoCtrl.Execute "MemoryRead"
```

3.1.7. CaoController: : Execute (“CFRead") command

Read the setting data from SD memory cards.

Syntax CFRead (<lReadNum >)

< lReadNum >	:	[in] Reading area number of SD memory cards (VT_I4) (0 to 99)
Return value	:	None

The following example shows how to designate Area number 10, and then read the setting data.

Example

```
Dim lReadNum as long
lReadNum = 10
caoCtrl.Execute "CFRead", lReadNum
```

3.1.8. CaoController: : Execute (“CancelData") command

Abort writing/reading out the setting data.

Syntax CancelData ( )

Argument	:	None
Return value	:	None

The following example shows how to abort writing/reading out the setting data.

Example

```
caoCtrl.Execute "CancelData"
```

3.1.9. CaoController: : Execute (“SDSave") command

Save the image stored in PV series on-board memory into an SD memory card.

Syntax SDSave ( )

Argument : None

Return value : None

The following example shows how to save the image memory to an SD memory card.

Example

```
caoCtrl.Execute "SDSave"
```

3.1.10. CaoController: : Execute (“SDReset") command

Delete the image memory stored in on-board memory.

Syntax SDReset ( )

Argument : None

Return value : None

The following example shows hot to delete the on-board image memory.

Example

```
caoCtrl.Execute "SDReset"
```

3.1.11. CaoController: : Execute (“PrintScreen") command

Capture all of the currently displayed images and then save them to a computer via an SD memory cards or Ethernet interface.

Syntax PrintScreen ( )

Argument : None

Return value : None

The following example shows how to save the current displays.

Example

```
caoCtrl.Execute "PrintScreen"
```

3.1.12. CaoController: : Execute (“Quit") command

Clear the statistical data and the execution count.

Syntax Quit ( )

Argument : None

Return value : None

The following example shows how to clear the statistical data and the execution count.

Example

```
caoCtrl.Execute "Quit"
```

3.1.13. CaoController: : Execute (“RunManual") command

Switch RUN/STOP status of PV series.

Syntax [< lResult > =] RunManual (< lMode >)

< lMode > : [in] Switch RUN/STOP status (VT_I4) (0-1)

0 : Switch to RUN.

1 : Switch to STOP.

< lResult > : [out] Switched value (VT_I4) (0-1)

0 : RUN

1 : STOP

The following example shows how to switch RUN/STOP status of the PV series.

Example

```
Dim lMode as long
Dim lResult as long

lMode = 1

lResult = caoCtrl.Execute("RunManual", lMode)

' lResult: 1
```

3.1.14. CaoController: : Execute (“ErrorReset") command

Reset the error signal.

Syntax ErrorReset ( )

Argument : None

Return value : None

The following example shows how to clear errors

Example

```
caoCtrl.Execute "ErrorReset"
```

3.1.15. CaoController: : Execute (“Cancel") command

Cancel an ongoing motion and then go back to the state immediate before the motion begins.

Syntax Cancel ( )

Argument : None

Return value : None

The following example shows how to cancel motion execution.

Example

```
caoCtrl.Execute "Cancel"
```

3.1.16. CaoController: : Execute (“KeyEmulator") command

Execute same operation as a keypad. No response from the PV series returns.

Syntax KeyEmulator (< lShift > < lKey >)

< lShift > : [in] Shift key ON･OFF (VT_I4) (0-1)

0 : OFF

1 : ON

< lKey > : [in] Assigned values for each keys (VT_I4) (1-16).

- Regarding values assigned to keys, refer to the attached table.

Return value : None

Table 3-2 Value assignment list to various keys

No	Assigned key	No	Assigned key
1	ENTER Left below	9	ENTER Upper right
2	ENTER Down	10	TRIG
3	ENTER Upper right	11	CANCEL
4	ENTER Left	12	FUNC
5	ENTER Center	13	F1
6	ENTER Right	14	F2
7	ENTER Upper left	15	F3

8	ENTER Up	16	OPE/SET switch

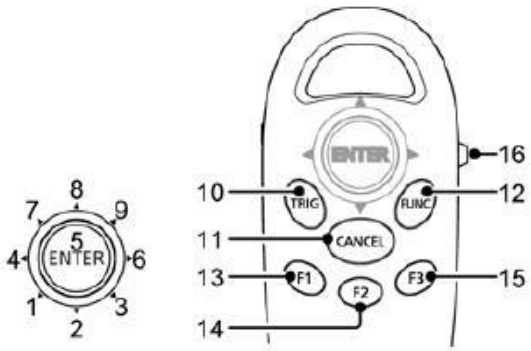

Figure3-1 Correspondence of keypad to value assignment

The following shows how to operate a keypad to switch RUN/SETUP menu.

Example

```
Dim lShift as long
Dim lKey as long

lShift = 0
lKey = 16

caoCtrl.Execute "KeyEmulator", Array(lShift, lKey)
```

3.1.17. CaoController: : Execute (“Bstop") command

Refuse/Permit the operation by a keypad on the RUN menu.

Syntax Bstop (< lEnabled >)

< lEnabled > : [in] Keypad operation permission (VT_I4) (0-1)

0 : Permit

1 : Refuse

Return value : None

The following shows how to refuse the keypad operation.

Example

```
Dim lEnabled as long

lEnabled = 1

caoCtrl.Execute "Bstop", lEnabled
```

3.1.18. CaoController: : Execute (“Bconfirm") command

Get the current state of a keypad operation permission.

Syntax Bconfirm ()

Argument : None

Return value : [out] Keypad operation permission state (VT_I4) (0-1)

0 : Permission

1 : Refuse

The following example shows how to get the permission state (permit) of the keypad operation.

Example

```
Dim lResult as long
lResult = caoCtrl.Execute("Bconfirm")
' lResult: 0
```

3.1.19. CaoController: : Execute (“LayoutChange") command

On the RUN menu, this command is used when the layout displayed in the monitor is switched by the signal from an external device.

Syntax LayoutChange (< lLayOut >)

< lLayOut > : [in] Layout number (VT_I4) (0 to 15)

Return value : None

The following example shows how to switch the layout to 1.

Example

```
Dim lLayOut as long
lLayOut = 1
caoCtrl.Execute "LayoutChange", lLayOut
```

3.1.20. CaoController: : Execute (“AgainTemplate") command

3.1.20.1. Operation for “Area Display” is “No”

Re-register the template of the smart matching checker.

Syntax AgainTemplate (< lChecker > ,< lTemplate >)

< lChecker > : [in] Checker number (VT_I4) (0 to 999)

< lTemplate > : [in] Template number (VT_I4) (0 to 63)

Return value : None

Note: Re-registrable smart matching is the smart matching locating under [Checker]. The smart matching used for the position correction or the area adjustment cannot re-register the template.

The following example shows how to re-register the template of the smart matching checker.

Example

```
Dim lChecker as long
Dim lTemplate as long

lChecker = 1
lTemplate = 10

caoCtrl.Execute "AgainTemplate", Array(lChecker, lTemplate)
```

3.1.21. CaoController: : Execute (“ParameterRead") command

Read the setting values and the system values of the PV series on-board memory.

Please refer to the PV series manual of Panasonic for readable data and each command parameters.

Syntax ParameterRead (< bstrParam >)

< bstrParam > : [in] Command parameter of reading object (VT_BSTR)

Return value : [out] Value of specified parameter (VT_BSTR)

The following example shows how to readout the current time.

Example

```
Dim bstrParam as string
Dim bstrResult as string

bstrParam = “SYS_TIME”

bstrResult = caoCtrl.Execute("ParameterRead", bstrParam)

‘ bstrResult: 12: 29: 30
```

3.1.22. CaoController: : Execute (“ParameterReadPair") command

Read two data of the PV series on-board memory.

Please refer to the PV series manual of Panasonic for readable data and each command parameters.

Syntax ParameterReadPair (< bstrParam >)

< bstrParam > : [in] Command parameter of the reading object (VT_BSTR)

Return value : [out] Value of specified parameter (VT_BSTR | VT_ARRAY)

The following example shows how to read the upper/lower limits of the binary level group "A" of camera 0.

Example

```
Dim bstrParam as string
Dim vntResult as variant

bstrParam = "BLV: PAIRA"

vntResult = caoCtrl.Execute("ParameterReadPair", bstrParam)

' vntResult: 80,255 (the lowest value, the highest value)
```

3.1.23. CaoController: : Execute ("ParameterWrite") command

Change the setting data and the system value of the PV series on-board memory.

Please refer to the PV series manual of Panasonic for changeable data and various command parameters.

Syntax ParameterWrite (< bstrParam >, < vntData >)

< bstrParam >	:	[in] Command parameter of changing target (VT_BSTR)
< vntData >	:	[in] Value to be changed (VT_VARIANT)
Return value	:	None

The following example shows how to change the value 0 of the general-purpose register to "3.14".

Example

```
Dim bstrParam as string
Dim vntData as variant

bstrParam = "SYS: REG0"
vntData = 3.14

caoCtrl.Execute "ParameterWrite", Array(bstrParam, vntData)
```

3.1.24. CaoController: : Execute ("ParameterWritePair") command

Change values of two data of the PV series on-board memory.

Please refer to the PV series manual of Panasonic for readable data and various command parameters.

Syntax ParameterWritePair (< bstrParam >, < vntData1 >, < vntData2 >)

< bstrParam >	:	[in] Command parameter of changing target (VT_BSTR)
< vntData1 >	:	[in] Value 1 to be changed (VT_VARIANT)
< vntData2 >	:	[in] Value 2 to be changed (VT_VARIANT)
Return value	:	None

The following example shows how to change the upper/lower limit of numeric operation No.10 to upper limit “100”, lower limit “50” respectively.

Example

```
Dim bstrParam as string
Dim vntData1 as variant
Dim vntData2 as variant

bstrParam = "CAC010: LPAIR"
vntData1 = 50
vntData2 = 100

caoCtrl.Execute "ParameterWritePair", Array(bstrParam, vntData1, vntData2)
```

3.2. PV260 Robot calibration command (Synchronous)

3.2.1. CaoController: : Execute(“SetPoint”) command

Notify PV of robot coordinates

Syntax 1. SetPoint (< dblX >, < dblY >, < dblZ >, < dblRx >, < dblRy >, < dblRz >, < lFig >)

< dblX >	:	[in] Robot coordinate (X)(VT_R8)
< dblY >	:	[in] Robot coordinate (Y)(VT_R8)
< dblZ >	:	[in] Robot coordinate (Z)(VT_R8)
< dblRx >	:	[in] Robot coordinate (Rx)(VT_R8)
< dblRy >	:	[in] Robot coordinate (Ry)(VT_R8)
< dblRz >	:	[in] Robot coordinate (Rz)(VT_R8)
< lFig >	:	[in] Robot coordinate (Fig)(VT_I4)
Return value	:	None

The following shows how to notify a PV of a current robot position.

Example

```
caoCtrl.Execute "SetPoint", Array(POSX(CURPOS), POSY(CURPOS), POSZ(CURPOS),
                                  POSRX(CURPOS), POSRY(CURPOS), POSRZ(CURPOS), FIG(CURPOS))
```

3.2.2. CaoController: : Execute(“Calibrate”) command

Execute the measurement. The syntax differs depending on the execution mode; “Execute All”, or “User Defined”. To execute the measurement, you need to prepare programs designed for Robot-PV cooperation. Refer to 4.2 Calibrate, Recalibrate for the command sequence.

3.2.2.1. Execution Mode is “Execute All”

Syntax Calibrate (< lCalibrationNum >)

< lCalibrationNum > : [in] Calibration number (VT_I4)(0 to 5)

Return value : [out] Robot coordinates array (X, Y, Rz, Fig) (VT_VARIANT)

The following example shows how to execute measurement for the Calibration number 0.

Example

```
Dim lCalibrationNum as long
Dim vntResult as variant

lCalibrationNum = 0

vntResult = caoCtrl.Execute("Calibrate", lCalibrationNum)

' vntResult :  Robot coordinates information (X, Y, Rz, Fig)
```

3.2.2.2. Execution Mode is “User Defined”

Syntax Calibrate (< lCalibrationNum >, <lBlockNum>)

< lCalibrationNum > : [in] Calibration number (VT_I4) (0 to 5)

< lBlockNum > : [in] Block number to execute (VT_I4) (0 to 9)

Return value : [out] Robot coordinates array (X, Y, Rz, Fig) (VT_VARIANT)

The following example shows how to execute measurement with specifying the Calibration number 0 and the Block number 1.

Example

```
Dim lCalibrationNum as long
Dim lBlockNum as long
Dim vntResult as variant

lCalibrationNum = 0
lBlockNum = 1

vntResult = caoCtrl.Execute("Calibrate", Array(lCalibrationNum, lBlockNum))

' vntResult :  Robot coordinates information (X, Y, Rz, Fig)
```

3.2.3. CaoController: : Execute(“ReCalibrate”) command

Execute inspection without importing images (re-measurement). The syntax differs depending on the Execution Mode; “Execute All” or “User Defined”.

To execute re-measurement, you need to prepare programs designed for Robot-PV cooperation. Refer to 4.2 Calibrate, Recalibrate for the command sequence.

3.2.3.1. Execution Mode is “Execute All”

Syntax ReCalibrate (< lCalibrationNum >)

< lCalibrationNum > : [in] Calibration number (VT_I4)(0 to 5)

Return value : [out] Robot coordinates array (X, Y, Rz, Fig) (VT_VARIANT)

The following shows hot to execute re-measurement for the Calibration number 0.

Example

```
Dim lCalibrationNum as long
Dim vntResult as variant

lCalibrationNum = 0

vntResult = caoCtrl.Execute("ReCalibrate", lCalibrationNum)

' vntResult :  Robot coordinates information (X, Y, Rz, Fig)
```

3.2.3.2. Execution Mode is “User Defined”

Syntax ReCalibrate (< lCalibrationNum >, <lBlockNum>)

< lCalibrationNum > : [in] Calibration number (VT_I4) (0 to 5)

< lBlockNum > : [in] Block number to execute (VT_I4) (0 to 9)

Return value : [out] Robot coordinates array (X, Y, Rz, Fig) (VT_VARIANT)

The following shows how to execute re-measurement with specifying the Calibration number 0 and the Block number 1.

Example

```
Dim lCalibrationNum as long
Dim lBlockNum as long
Dim vntResult as variant

lCalibrationNum = 0
lBlockNum = 1

vntResult = caoCtrl.Execute("ReCalibrate", Array(lCalibrationNum, lBlockNum))

' vntResult :  Robot coordinates information (X, Y, Rz, Fig)
```

3.2.4. CaoController: : Execute(“CalibrationStart”) command

Start Auto calibration. To execute the measurement, you need to prepare programs designed for Robot-PV cooperation. Refer to 4.3CalibrationStart command sequence for the command sequence.

If there is no entry in the “Camera No.” argument, it is deemed that the Camera No. is not designated. (Format of version 1.1.0 or before)

Syntax CalibrationStart (< lCalibrationNum > [, < lCameraNum >])

< lCalibrationNum >	:	[in] Calibration number (VT_I4)(0 to 5)
< lCameraNum >	:	[in] Camera No. (VT_I4) (0 to 1)
Return value	:	None

The following example shows how to start Auto calibration for the Calibration number 0.

Example

```
Dim lCalibrationNum as long
Dim lCameraNum as long

lCalibrationNum = 0
lCameraNum = 1

‘Inform PV that the Camera No.is not designated (%CAS0)
caoCtrl.Execute "CalibrationStart", lCalibrationNum

‘Inform PV that the Camera No.is designated (%CAS0,1)
caoCtrl.Execute "CalibrationStart", Array(lCalibrationNum, lCameraNum)
```

3.2.5. CaoController: : Execute(“CalibrationEnd”) command

Obtain the notification of Auto calibration completion. To execute the measurement, you need to prepare programs designed for Robot-PV cooperation. Refer to 4.3 CalibrationStart command sequence for the command sequence.

Syntax CalibrationEnd ( )

Argument	:	None
Return value	:	None

The following shows how to obtain the notification of Auto calibration completion.

Example

```
caoCtrl.Execute "CalibrationEnd"
```

3.2.6. CaoController: : Execute(“WorkSet”) command

Re-register a work detection base position (without taking pictures).

If you change the calibration configuration after the base position registration, you need to register the base position again. You can re-calculates the base position automatically with this command.

Syntax WorkSet ( )

Argument	:	None
Return value	:	None

The following shows how to re-register the work detection base position.

Example

```
caoCtrl.Execute "WorkSet"
```

3.2.7. CaoController: : Execute(“WorkReset”) command

Re-register a work detection base position (with taking pictures).

If you change the calibration configuration after the base position registration, you need to register the base position again. If all the setting values of Number of fields, Number of markings, and Robot position information at the base position registration are the same as the earlier registration, you can re-calculate the base position automatically by executing this command.

To register the base position again with taking pictures, you need to prepare programs designed for Robot-PV cooperation. Refer to 4.4 WorkReset command sequence for the command sequence.

Syntax WorkReset (< lWorkNum >)

< lWorkNum >	:	[in] Work detection number (VT_I4)(0 to 15)
Return value	:	None

The following shows how to re-register the work detection base position.

Example

```
Dim lWorkNum as long
lWorkNum = 0
caoCtrl.Execute "WorkReset", lWorkNum
```

3.2.8. CaoController: : Execute(“WorkResetEnd”) command

Obtain the notification of the work detection base position re-registration completion. To register the work detection base position again, you need to prepare programs designed for Robot-PV cooperation. Refer to 4.4 WorkReset command sequence for the command sequence.

Syntax WorkResetEnd ( )

Argument : None

Return value : None

The following shows how to obtain the notification of the work detection base position re-registration completion.

Example

```
caoCtrl.Execute "WorkResetEnd"
```

3.2.9. CaoController: : Execute("MoveEnd") command

Notify PV of robot movement completion.

Syntax MoveEnd ( )

Argument : None

Return value : None

The following shows how to notify PV of robot movement completion.

Example

```
caoCtrl.Execute "MoveEnd"
```

3.2.10. CaoController: : Execute("GetTeachPoint") command

Obtain all teaching coordinate configured in PV.

Syntax GetTeachPoint ( )

Argument : None

Return value : Robot coordinates array (X, Y, Rz, Fig)
(VT_VARIANT | VT_ARRAY)

The following shows how to request the teaching coordinates.

Example

```
Dim vntResult as variant
vntResult = caoCtrl.Execute("GetTeachPoint")
' vntResult :  Robot coordinates information array (X, Y, Rz, Fig)
```

3.2.11. CaoController: : Execute("GetMovePoint") command

Obtain robot coordinates sent from PV during Auto calibration (CalibrationStart) or during re-registration

of the work detection base position (WorkReset). To execute Auto calibration and re-registration of the work detection base position, you need to prepare programs designed for Robot-PV cooperation. Refer to 4.3 CalibrationStart command sequence and 4.4 WorkReset command sequence.

Syntax GetMovePoint ( )

Argument	:	None
Return value	:	Robot coordinates (X, Y, Rz, Fig)(VT_VARIANT)

The following shows how to obtain robot coordinates sent from PV.

Example

```
Dim vntMovePos as variant
vntMovePos = caoCtrl.Execute("GetMovePoint")
' vntMovePos :  Robot coordinates
```

3.3. Original command (synchronous)

3.3.1. CaoController: : Execute ("Raw") command

Send and receive a command message. BCC is calculated internally automatically.

Syntax ExecRaw(< bstrCmdMessage >)

< bstrCmdMessage >	:	[in] Command message to be transmitted (VT_BSTR)
Return value	:	[out] Command message to receive (VT_BSTR)

The following example shows how to execute inspection with Common trigger and with the execution mode of "All executions" or "User Defined".

Example

```
Dim bstrParam as string
Dim bstrResult as string
bstrCmdMessage = "%S"
vntResult = caoCtrl.Execute "Raw", bstrCmdMessage
```

3.3.2. CaoController: : Execute("SetTimeout") command

Specify a communication timeout period. In default, the value is the same as the one configured in AddController.

Syntax SetTimeout (< lTimeout >)

< lTimeout >	:	[in] Timeout period msec. (VT_I4)
Return value	:	None

The following shows how to specify the timeout period to 1 second (1000 msec.).

Example

```
caoCtrl.Execute "SetTimeout", 1000
```

3.3.3. CaoController: : Execute("GetTimeout") command

Obtain the communication timeout period. In default, the value is the same as the one configured in AddController.

Syntax GetTimeout ( )

Argument	:	None
Return value	:	[out] Timeout period msec.(VT_I4)

The following shows how to obtain the timeout period (1000 msec.).

Example

```
Dim lResult as long
lResult = caoCtrl.Execute("GetTimeout")
' lResult: 1000
```

3.4. General communication command (Asynchronous)

3.4.1. CaoController: : Execute("StartAsync") command

Start inspection asynchronously. The syntax differs depending on the execution mode; "Execute All", "Automatic Switch", or "User Defined".

To obtain and check the return value of the command, use GetResult command. For details about GetResult command, please refer to 3.6.2.CaoController: : Execute("GetResult") command) command.

3.4.1.1. Execution Mode is "Execute All" or "Automatic Switch", with Common trigger used

Syntax StartAsync ( )

Argument	:	None
Return value	:	None

The following shows how to execute inspection asynchronously.

Example

```
Dim vntResult as variant

caoCtrl.Execute "StartAsync"

' Obtain the return value of StartAsync command
vntResult = caoCtrl.Execute("GetResult")

' vntResult : Inspection result
```

3.4.1.2. Execution Mode is "User Defined ", with Common trigger used

Syntax StartAsync (<lBlockNum>)

< lBlockNum > : [in] Block number to execute (VT_I4) (0 to 9)

Return value : None

The following shows how to execute inspection asynchronously with specifying the Block number 1.

Example

```
Dim lBlockNum as long
Dim bstrResult as string

lBlockNum = 1

caoCtrl.Execute "StartAsync", lBlockNum

'Obtain the return value of StartAsync command
bstrResult = caoCtrl.Execute("GetResult")

' bstrResult : Inspection result
```

3.4.2. CaoController: : Execute("ReStartAsync") command

Execute inspection asynchronously without taking pictures (re-inspection). The syntax differs depending on the execution mode; "Execute All", "Automatic Switch", or "User Defined".

To obtain and check the return value of the command, use GetResult command.For details about GetResult command, please refer to 3.6.2.CaoController: : Execute("GetResult") command command.

3.4.2.1. Execution Mode is "Execute All" or "Automatic Switch"

Syntax RestartAsync ( )

Argument : None

Return value : None

The following shows how to execute re-inspection asynchronously.

Example

```
Dim vntResult as variant

caoCtrl.Execute "RestartAsync"

' Obtain the return value of RestartAsync command
vntResult = caoCtrl.Execute("GetResult")

' vntResult : Inspection result
```

3.4.2.2. Execution Mode is "User Defined"

Syntax RestartAsync (< lBlockNum >)

< lBlockNum > : [in] Block number to execute (VT_I4) (0 to 9)

Return value : None

The following shows how to set the Block number to 1 and execute re-inspection, asynchronously.

Example

```
Dim lBlockNum as long
Dim vntResult as variant

lBlockNum = 1

caoCtrl.Execute "RestartAsync", lBlockNum

' Obtain the return value of RestartAsync command
vntResult = caoCtrl.Execute("GetResult")

' vntResult : Image processing result
```

3.4.3. CaoController: : Execute("XTypeAsync") command

Switch a product type asynchronously.

To obtain and check the return value of the command, use GetResult command.For details about GetResult command, please refer to 3.6.2.CaoController: : Execute("GetResult") command command.

Syntax XtypeAsync (<lProductNum>)

< lProductNum > : [in] product type number (VT_I4) (0 to 255)

Return value : None

The following shows how to switch the product type number to 100.

Example

```
Dim lProductNum as long
Dim vntResult as variant

lProductNum = 100

caoCtrl.Execute "XtypeAsync", lProductNum

' Obtain the return value of XtypeAsync command
vntResult = caoCtrl.Execute("GetResult")

' vntResult :  Empty
```

3.4.4. CaoController: : Execute("MemoryWriteAsync") command

Write the setting data into PV series storage area asynchronously.

To obtain and check the return value of the command, use GetResult command.For details about GetResult command, please refer to 3.6.2.CaoController: : Execute("GetResult") command command.

Syntax MemoryWriteAsync ([< lWriteNum >])

< lWriteNum > : This parameter differs depending on the PV series used.

PV200	None
PV500	[in]Storage area number of PV500 (VT_I4) 0 to 99)

Return value : None

The following shows how to store the setting data in the memory storage area of PV asynchronously.

Example

```
Dim vntResult as variant

caoCtrl.Execute "MemoryWriteAsync"

' Obtain the return value of MemoryWriteAsync command
vntResult = caoCtrl.Execute("GetResult")

' vntResult :  Empty
```

3.4.5. CaoController: : Execute ("CFWriteAsync") command

Write the setting data to an SD memory card asynchronously.

To obtain and check the return value of the command, use GetResult command.For details about

GetResult command, please refer to 3.6.2.CaoController: : Execute(“GetResult”) command command.

Syntax CFWriteAsync (< lWriteNum >)

< lWriteNum > : [in] Storage are number of SD area number (VT_I4) (0 to 99)

Return value : None

The following shows how to save the setting data into the Storage area number 10 of SD memory card asynchronously.

Example

```
Dim lWriteNum as long
Dim vntResult as variant

 lWriteNum = 10

caoCtrl.Execute "CFWriteAsync", lWriteNum

‘ Obtain the return value of CFWriteAsync command
vntResult = caoCtrl.Execute("GetResult")

‘ vntResult :  Empty
```

3.4.6. CaoController: : Execute (“MemoryReadAsync”) command

Read the setting data from the memory of PV series asynchronously.

To obtain and check the return value of the command, use GetResult command.For details about GetResult command, please refer to 3.6.2.CaoController: : Execute(“GetResult”) command command.

Syntax MemoryReadAsync ([<lReadNum>])

<lReadNum> : This parameter differs depending on the PV series used.

PV200	None
PV500	[in]Storage area number of PV500 (VT_I4) (0 to 99)

Return value : None

The following shows how to read the setting data from the memory of PV asynchronously

Example

```
Dim vntResult as variant
caoCtrl.Execute "MemoryReadAsync"

‘ Obtain the return value of  MemoryReadAsync command
vntResult = caoCtrl.Execute("GetResult")

‘ vntResult :  Empty
```

3.4.7. CaoController: : Execute (“CFReadAsync”) command

Read the setting data from an SD memory card asynchronously.

To obtain and check the return value of the command, use GetResult command.For details about GetResult command, please refer to 3.6.2.CaoController: : Execute(“GetResult”) command command.

Syntax CFReadAsync (<lReadNum >)

< lReadNum >	:	[in] Area number of SD memory card to read (VT_I4) (0 to 99)
Return value	:	None

The following shows how to specify Area number 10 and read the data asynchronously.

Example

```
Dim lReadNum as long
Dim vntResult as variant

lReadNum = 10

caoCtrl.Execute "CFReadAsync", lReadNum

' Obtain the return value of CFReadAsync command
vntResult = caoCtrl.Execute("GetResult")

' vntResult :  Empty
```

3.4.8. CaoController: : Execute (“CancelDataAsync”) command

Cancel saving/reading of the setting data asynchronously.

To obtain and check the return value of the command, use GetResult command.For details about GetResult command, please refer to 3.6.2.CaoController: : Execute(“GetResult”) command command.

Syntax CancelDataAsync ( )

Argument	:	None
Return value	:	None

The following shows how to cancel saving/reading the setting data asynchronously.

Example

```
Dim vntResult as variant

caoCtrl.Execute "CancelDataAysnc"

' Obtain the return value of CancelDataAsync command
```

```
vntResult = caoCtrl.Execute("GetResult")

' vntResult :  Empty
```

3.4.9. CaoController: : Execute ("SDSaveAsync") command

Save the image memory data stored in PV into an SD memory card.

To obtain and check the return value of the command, use GetResult command.For details about GetResult command, please refer to 3.6.2.CaoController: : Execute("GetResult") command command.

Syntax SDSaveAsync ( )

Argument : None

Return value : None

The following shows how to save the image memory data into an SD memory card asynchronously.

Example

```
Dim vntResult as variant

caoCtrl.Execute "SDSaveAsync"

' Obtain the return value of SDSaveAsync command
vntResult = caoCtrl.Execute("GetResult")

' vntResult :  Empty
```

3.4.10. CaoController: : Execute ("SDResetAsync") command

Delete the image memory data stored in the PV series asynchronously.

To obtain and check the return value of the command, use GetResult command.For details about GetResult command, please refer to 3.6.2.CaoController: : Execute("GetResult") command command.

Syntax SDResetAsync ( )

Argument : None

Return value : None

The following shows how to delete the image memory data stored in the PV series asynchronously.

Example

```
Dim vntResult as variant

caoCtrl.Execute "SDResetAsync"

' Obtain the return value of SDReadAsync command
vntResult = caoCtrl.Execute("GetResult")
```

```
' vntResult :  Empty
```

3.4.11. CaoController: : Execute (“PrintScreenAsync”) command

Capture the current displays (all items to be displayed) and then save the data into an SD memory card or into a computer via Ethernet interface, asynchronously.

To obtain and check the return value of the command, use GetResult command.For details about GetResult command, please refer to 3.6.2.CaoController: : Execute(“GetResult”) command command.

Syntax PrintScreenAsync ( )

Argument : None

Return value : None

The following shows how to save the current display asynchronously.

Example

```
Dim vntResult as variant

caoCtrl.Execute "PrintScreenAsync"

' Obtain the return value of PrintScreenAsync command
vntResult = caoCtrl.Execute("GetResult")

' vntResult :  Empty
```

3.4.12. CaoController: : Execute (“QuitAsync”) command

Clear the statistics data and scan count asynchronously.

To obtain and check the return value of the command, use GetResult command.For details about GetResult command, please refer to 3.6.2.CaoController: : Execute(“GetResult”) command command.

Syntax QuitAsync ( )

Argument : None

Return value : None

The following shows how to clear the statistics data and scan data asynchronously.

Example

```
Dim vntResult as variant

caoCtrl.Execute "QuitAsync"

' Obtain the return value of QuitAsync comand
vntResult = caoCtrl.Execute("GetResult")
```

```
' vntResult :  Empty
```

3.4.13. CaoController: : Execute (“RunManualAsync”) command

Switch the PV series operation state between RUN and STOP asynchronously.–

To obtain and check the return value of the command, use GetResult command.For details about GetResult command, please refer to 3.6.2.CaoController: : Execute(“GetResult”) command command.

Syntax RunManualAsync (< lMode >)

< lMode > : [in] Switch RUN/STOP (VT_I4) (0 to 1)

0 : Switch to RUN-state.

1 : Switch to STOP-state

Return value : None

The following shows how to switch the PV series from RUN to STOP, asynchronously.

Example

```
Dim lMode as long
Dim vntResult as variant

lMode = 1

caoCtrl.Execute "RunManualAsync", lMode

' Obtain the return value of RunManualAsync command
vntResult = caoCtrl.Execute("GetResult")

' vntResult: 1
```

3.4.14. CaoController: : Execute (“ErrorResetAsync”) command

Reset an Errror signal asynchronously.–

To obtain and check the return value of the command, use GetResult command.For details about GetResult command, please refer to 3.6.2.CaoController: : Execute(“GetResult”) command command.

Syntax ErrorResetAsync ( )

Argument : None

Return value : None

The following shows how to clear an error asynchronously.

Example

```
Dim vntResult as variant
caoCtrl.Execute "ErrorResetAsync"
```

```
' Obtain the return value of ErrorResetAsync command
vntResult = caoCtrl.Execute("GetResult")

' vntResult :  Empty
```

3.4.15. CaoController: : Execute ("CancelAsync") command

Cancel an ongoing motion and then go back to the state immediate before the motion begins, asynchronously.

To obtain and check the return value of the command, use GetResult command.For details about GetResult command, please refer to 3.6.2.CaoController: : Execute("GetResult") command command.

Syntax CancelAsync ( )

Argument	:	None
Return value	:	None

The following shows how to cancel an ongoing motion asynchronously.

Example

```
Dim vntResult as variant

caoCtrl.Execute "CancelAsync"

' Obtain the return value of CancelAsync command
vntResult = caoCtrl.Execute("GetResult")

' vntResult :  Empty
```

3.4.16. CaoController: : Execute ("KeyEmulatorAsync") command

Execute same operation as a keypad asynchronously. No response from the PV series returns.

To obtain and check the return value of the command, use GetResult command.For details about GetResult command, please refer to 3.6.2.CaoController: : Execute("GetResult") command command.

Syntax KeyEmulator (< lShift > < lKey >)

< lShift >	:	[in] Shift key ON・OFF (VT_I4) (0 to 1) 0 : OFF 1 : ON
< lKey >	:	[in] Assigned values for each keys (VT_I4) (1 to 16) Regarding values assigned to keys, refer to KeyEmulator command.
Return value	:	None

The following shows how to operate the keypad to switch RUN/SETUP menu, asynchronously.

Example

```
Dim lShift as long
Dim lKey as long
Dim vntResult as variant

lShift = 0
lKey = 16

caoCtrl.Execute "KeyEmulatorAsync", Array(lShift, lKey)

' Obtain the return value of KeyEmulatorAsync command
vntResult = caoCtrl.Execute("GetResult")

' vntResult :  Empty
```

3.4.17. CaoController: : Execute (“BstopAsync”) command

Refuse/Permit the operation by a keypad on the RUN menu, asynchronously.

To obtain and check the return value of the command, use GetResult command.For details about GetResult command, please refer to 3.6.2.CaoController: : Execute(“GetResult”) command command.

Syntax BstopAsync (< lEnabled >)

< lEnabled > : [in] Keypad operation permission (VT_I4) (0 to 1)

0 : Permit

1 : Refuse

Return value : None

The following shows how to refuse the keypad operation, asynchronously.

Example

```
Dim lEnabled as long
Dim vntResult as variant

lEnabled = 1

caoCtrl.Execute "BstopAsync", lEnabled

' Obtain the return value of BstopAsync command
vntResult = caoCtrl.Execute("GetResult")

' vntResult :  Empty
```

3.4.18. CaoController: : Execute (“BconfirmAsync”) command

Get the current state of keypad operation permission, asynchronously.

To obtain and check the return value of the command, use GetResult command.For details about GetResult command, please refer to 3.6.2.CaoController: : Execute(“GetResult”) command command.

Syntax BconfirmAsync ()

Argument : None

Return value : None

The following example shows how to get the permission state (permit) of the keypad operation, asynchronously.

Example

```
Dim vntResult as long

caoCtrl.Execute "BconfirmAsync"

' Obtain the return value of BconfirmAsync command
vntResult = caoCtrl.Execute("GetResult")

' vntResult: 0
```

3.4.19. CaoController: : Execute (“LayOutChangeAsync”) command

On the RUN menu, this command is used when the layout displayed in the monitor is switched by the signal from an external device, asynchronously.

To obtain and check the return value of the command, use GetResult command.For details about GetResult command, please refer to 3.6.2.CaoController: : Execute(“GetResult”) command command.

Syntax LayoutChangeAsync (< lLayOut >)

< lLayOut > : [in] Layout number (VT_I4) (0 to 15)

Return value : None

The following example shows how to switch the layout to 1, asynchronously.

Example

```
Dim lLayOut as long
Dim vntResult as variant

lLayOut = 1

caoCtrl.Execute "LayoutChangeAsync", lLayOut

' Obtain the return value of LayoutChangeAsync command
vntResult = caoCtrl.Execute("GetResult")

' vntResult :  Empty
```

3.4.20. CaoController: : Execute (“AgainTemplateAsync”) command

3.4.20.1. Operation for “Area Display” is “No”

Re-register the template of the smart matching checker, asynchronously.

To obtain and check the return value of the command, use GetResult command.For details about GetResult command, please refer to 3.6.2.CaoController: : Execute(“GetResult”) command command.

Syntax AgainTemplateAsync (< lChecker > , < lTemplate >)

< lChecker >	:	[in] Checker number (VT_I4) (0 to 999)
< lTemplate >	:	[in] Template number (VT_I4) (0 to 63)
Return value	:	None

Note: Re-registerable smart matching is the smart matching locating under [Checker]. The smart matching used for the position correction or the area adjustment cannot re-register the template

The following example shows how to re-register the template of the smart matching checker, asynchronously.

Example

```
Dim lChecker as long
Dim lTemplate as long
Dim vntResult as variant

lChecker = 1
lTemplate = 10

call caoCtrl.Execute("AgainTemplateAsync", Array(lChecker, lTemplate))

‘ Obtain the return value of AgainTemplateAsync command
vntResult = caoCtrl.Execute("GetResult")

‘ vntResult :  Empty
```

3.4.21. CaoController: : Execute (“ParameterReadAsync”) command

Read the setting values and the system values of the PV series on-board memory, asynchronously.

Please refer to the PV series manual of Panasonic for readable data and each command parameters.

To obtain and check the return value of the command, use GetResult command.For details about GetResult command, please refer to 3.6.2.CaoController: : Execute(“GetResult”) command command.

Syntax ParameterReadAsync (< bstrParam >)

< bstrParam >	:	[in] Command parameter of reading object (VT_BSTR)
Return value	:	[out] Value of specified parameter (VT_BSTR)

The following example shows how to readout the current time, asynchronously.

Example

```
Dim bstrParam as string
Dim vntResult as string
```

```
bstrParam = “SYS_TIME”

caoCtrl.Execute ″ParameterReadAsync″, bstrParam

‘ Obtain the return value of ParameterReadAsync command
vntResult = caoCtrl.Execute(″GetResult″)

‘ vntResult: 12: 29: 30
```

3.4.22. CaoController: : Execute (“ParameterReadPairAsync”) command

Read two data of the PV series on-board memory, asynchronously.

Please refer to the PV series manual of Panasonic for readable data and each command parameters.

To obtain and check the return value of the command, use GetResult command.For details about GetResult command, please refer to 3.6.2.CaoController: : Execute(“GetResult”) command command.

Syntax ParameterReadPairAsync (< bstrParam >)

< bstrParam >	:	[in] Command parameter of the reading object (VT_BSTR)
Return value	:	[out] Value of specified parameter (VT_BSTR | VT_ARRAY)

The following example shows how to read the upper/lower limits of the binary level group "A" of camera 0, asynchronously.

Example

```
Dim bstrParam as string
Dim vntResult as variant

bstrParam = “BLV: PAIRA”

caoCtrl.Execute ″ParameterReadPairAsync″, bstrParam

‘ Obtain the return value of ParameterReadPairAsync command
vntResult = caoCtrl.Execute(″GetResult″)

‘ vntResult: 80,255(the lowest value, the highest value)
```

3.4.23. CaoController: : Execute (“ParameterWriteAsync”) command

Change the setting data and the system value of the PV series on-board memory, asynchronously.

Please refer to the PV series manual of Panasonic for changeable data and various command parameters.

To obtain and check the return value of the command, use GetResult command.For details about GetResult command, please refer to 3.6.2.CaoController: : Execute(“GetResult”) command command.

Syntax ParameterWriteAsync (< bstrParam >, < vntData >)

< bstrParam >	:	[in] Command parameter of changing target (VT_BSTR)

< vntData >	:	[in] Value to be changed (VT_VARIANT)
Return value	:	None

The following example shows how to change the value 0 of the general-purpose register to “3.14”, asynchronously.

Example

```
Dim bstrParam as string
Dim vntData as variant
Dim vntResult as variant

bstrParam = “SYS: REG0”
vntData = 3.14

caoCtrl.Execute "ParameterWriteAsync", Array(bstrParam, vntData)

‘ Obtain the return value of ParameterWriteAsync command
vntResult = caoCtrl.Execute("GetResult")

‘ vntResult :  Empty
```

3.4.24. CaoController: : Execute (“ParameterWritePairAsync”) command

Change values of two data of the PV series on-board memory, asynchronously.

Please refer to the PV series manual of Panasonic for readable data and various command parameters.

To obtain and check the return value of the command, use GetResult command.For details about GetResult command, please refer to 3.6.2.CaoController: : Execute(“GetResult”) command command.

Syntax ParameterWritePairAsync (< bstrParam >, < vntData1 >, < vntData2 >)

< bstrParam >	:	[in] Command parameter of changing target (VT_BSTR)
< vntData1 >	:	[in] Value 1 to be changed (VT_VARIANT)
< vntData2 >	:	[in] Value 2 to be changed (VT_VARIANT)
Return value	:	None

The following example shows how to asynchronously change the upper/lower limit of numeric operation No.10 to upper limit “100”, lower limit “50”, respectively.

Example

```
Dim bstrParam as string
Dim vntData1 as variant
Dim vntData2 as variant
Dim vntResult as variant

bstrParam = “CAC010: LPAIR”
vntData1 = 50
vntData2 = 100
```

```
caoCtrl.Execute "ParameterWritePairAsync", Array(bstrParam, vntData1, vntData2)

'Obtain the return value of ParameterWritePairAsync command
vntResult = caoCtrl.Execute("GetResult")

' vntResult :  Empty
```

3.5. PV260 Robot calibration command (Asynchronous)

3.5.1. CaoController: : Execute ("SetPointAsync") command

Notify PV of the robot coordinates asynchronously.

To obtain and check the return value of the command, use GetResult command.For details about GetResult command, please refer to 3.6.2.CaoController: : Execute("GetResult") command command.

Syntax SetPointAsync (< dblX >, < dblY >, < dblZ >, < dblRx >, < dblRy >, < dblRz >, < lFig >)

< dblX >	:	[in] Robot coordinate (X)(VT_R8)
< dblY >	:	[in] Robot coordinate (Y)(VT_R8)
< dblZ >	:	[in] Robot coordinate (Z)(VT_R8)
< dblRx >	:	[in] Robot coordinate (Rx)(VT_R8)
< dblRy >	:	[in] Robot coordinate (Ry)(VT_R8)
< dblRz >	:	[in] Robot coordinate (Rz)(VT_R8)
< lFig >	:	[in] Robot coordinate (Fig)(VT_I4)
Return value	:	None

The following shows how to notify PV of the current robot position asynchronously.

Example

```
Dim vntResult as variant

caoCtrl.Execute "SetPointAsync", Array(POSX(CURPOS), POSY(CURPOS), POSZ(CURPOS),
                                  POSRX(CURPOS), POSRY(CURPOS), POSRZ(CURPOS), FIG(CURPOS))

' Obtain the return value of SetPointAsync command
vntResult = caoCtrl.Execute("GetResult")

' vntResult :  Empty
```

3.5.2. CaoController: : Execute ("CalibrateAsync") command

Execute the measurement asynchronously. The syntax differs depending on the execution mode; "Execute All", or "User Defined".

To execute the measurement, you need to prepare programs designed for Robot-PV cooperation. Refer to 4.2 Calibrate, Recalibratefor command sequence.

To obtain and check the return value of the command, use GetResult command.For details about GetResult command, please refer to 3.6.2.CaoController: : Execute("GetResult") command command.

3.5.2.1. Execution Mode is “Execute All”

Syntax CalibrateAsync (< lCalibrationNum >)

< lCalibrationNum > : [in] Calibration number (VT_I4)(0 to 5)

Return value : None

The following shows how to execute the measurement asynchronously for the Calibration number 0.

Example

```
Dim lCalibrationNum as long
Dim vntResult as variant

lCalibrationNum = 0

caoCtrl.Execute "CalibrateAsync", lCalibrationNum

' Obtain the return value of CalibrateAsync command
vntResult = caoCtrl.Execute("GetResult")

' vntResult :  Robot coordinates information (X, Y, Rz, Fig)
```

3.5.2.2. Execution Mode is “User Defined”

Syntax CalibrateAsync (< lCalibrationNum >, <lBlockNum>)

< lCalibrationNum > : [in] Calibration number (VT_I4) (0 to 9)

< lBlockNum > : [in] Block number to execute (VT_I4) (0 to 9)

Return value : None

The following shows how to execute measurement asynchronously with specifying the Calibration number 0 and the Block number 1.

Example

```
Dim lCalibrationNum as long
Dim lBlockNum as long
Dim vntResult as variant

lCalibrationNum = 0
lBlockNum = 1

call caoCtrl.Execute("CalibrateAsync", Array(lCalibrationNum, lBlockNum))

' Obtain the return value of CalibrateAsync command
vntResult = caoCtrl.Execute("GetResult")

' vntResult :  Robot coordinates information (X, Y, Rz, Fig)
```

3.5.3. CaoController: : Execute (“ReCalibrateAsync”) command

Execute the measurement asynchronously without taking pictures (re-measurement). The syntax differs depending on the execution mode; “Execute All”, or “User Defined”.

To execute re-measurement, you need to prepare programs designed for Robot-PV cooperation. Refer to 4.2 Calibrate，Recalibrate for command sequence.

To obtain and check the return value of the command, use GetResult command.For details about GetResult command, please refer to 3.6.2.CaoController: : Execute(“GetResult”) command command.

3.5.3.1. Execution Mode is “Execute All”

Syntax ReCalibrateAsync (< lCalibrationNum >)

< lCalibrationNum > : [in] Calibration number (VT_I4)(0 to 5)

Return value : None

The following shows how to execute re-measurement asynchronously for the Calibration number 0.

Example

```
Dim lCalibrationNum as long
Dim vntResult as variant

lCalibrationNum = 0

caoCtrl.Execute "ReCalibrateAsync", lCalibrationNum

‘ Obtain the return value of ReCalibrateAsync command
vntResult = caoCtrl.Execute("GetResult")

‘ vntResult :  Robot coordinates information (X, Y, Rz, Fig)
```

3.5.3.2. Execution Mode is “User Defined”

Syntax ReCalibrateAsync (< lCalibrationNum >, <lBlockNum>)

< lCalibrationNum > : [in] Calibration number (VT_I4) (0 to 5)

< lBlockNum > : [in] Block number to execute (VT_I4) (0 to 9)

Return value : None

The following shows how to execute re-measurement asynchronously with specifying the Calibration number 0 and the Block number 1.

Example

```
Dim lCalibrationNum as long
Dim lBlockNum as long
Dim vntResult as variant

lCalibrationNum = 0
lBlockNum = 1

call caoCtrl.Execute("ReCalibrateAsync", Array(lCalibrationNum, lBlockNum))

' Obtain the return value of ReCalibrateAsync command
vntResult = caoCtrl.Execute("GetResult")

' vntResult :  Robot coordinates information (X, Y, Rz, Fig)
```

3.5.4. CaoController: : Execute ("CalibrationStartAsync") command

Start asynchronous auto calibration. To execute the measurement, you need to prepare programs designed for Robot-PV cooperation. Refer to 4.3CalibrationStart command sequence for the command sequence.

If there is no entry in the "Camera No." argument, it is deemed that the Camera No. is not designated. (Format of version 1.1.0 or before)

To obtain and check the return value of the command, use GetResult command.For details about GetResult command, please refer to 3.6.2.CaoController: : Execute("GetResult") command command.

Syntax CalibrationStartAsync (< lCalibrationNum >)

< lCalibrationNum >	:	[in] Calibration number (VT_I4)(0 to 5)
< lCameraNum >	:	[in] Camera No. (VT_I4) (0 to 1)
Return value	:	None

The following shows how to start Auto calibration for the Calibration number 0.

Example

```
Dim lCalibrationNum as long
Dim lCameraNum as long
Dim vntResult as variant

lCalibrationNum = 0
lCameraNum = 1

'Inform PV that the Camera No.is not designated (%CAS0)
caoCtrl.Execute "CalibrationStartAsync", lCalibrationNum

'Obtain the return value of CalibrationStartAsync command
vntResult = caoCtrl.Execute("GetResult")


'Inform PV that the Camera No.is designated (%CAS0,1)
caoCtrl.Execute "CalibrationStartAsync", Array(lCalibrationNum, lCameraNum)

'Obtain the return value of CalibrationStartAsync command.
```

```
vntResult = caoCtrl.Execute("GetResult")

' vntResult :  Empty
```

3.5.5. CaoController: : Execute ("CalibrationEndAsync") command

Obtain the notification of Auto calibration completion asynchronously. To execute Auto calibration, you need to prepare programs designed for Robot-PV cooperation. Refer to 4.3CalibrationStart command sequence. To execute the measurement, you need to prepare programs designed for Robot-PV cooperation. Refer to 4.3CalibrationStart command sequence for the command sequence.

To obtain and check the return value of the command, use GetResult command.For details about GetResult command, please refer to 3.6.2.CaoController: : Execute("GetResult") command command.

Syntax CalibrationEndAsync ( )

Argument : None

Return value : None

The following shows how to obtain the notification of Auto calibration asynchronously.

Example

```
Dim vntResult as variant

caoCtrl.Execute "CalibrationEndAsync"

' Obtain the return value of CalibrationEndAsync command
vntResult = caoCtrl.Execute("GetResult")

vntResult :  Empty
```

3.5.6. CaoController: : Execute ("WorkSetAsync") command

Re-register a work detection base position asynchronously (without taking pictures).

If you change the calibration configuration after the base position registration, you need to register the base position again. You can re-calculates the base position automatically with this command

To obtain and check the return value of the command, use GetResult command.For details about GetResult command, please refer to 3.6.2.CaoController: : Execute("GetResult") command command.

Syntax WorkSetAsync ( )

Argument : None

Return value : None

The following shows how to re-register the work detection base position asynchronously.

Example

```
Dim vntResult as variant
```

```
caoCtrl.Execute "WorkSetAsync"

' Obtain the return value of WorkSetAsync command
vntResult = caoCtrl.Execute("GetResult")

' vntResult :  Empty
```

3.5.7. CaoController: : Execute ("WorkResetAsync") command

Re-register a work detection base position asynchronously (with taking pictures).

If you change the calibration configuration after the base position registration, you need to register the base position again.

If all the setting values of Number of fields, Number of markings, and Robot position information at the base position registration are the same as the earlier registration, you can re-calculate the base position automatically by executing this command.

To register the base position again with taking pictures, you need to prepare programs designed for Robot-PV cooperation. Refer to 4.4 WorkReset command sequence for the command sequence.

To obtain and check the return value of the command, use GetResult command.For details about GetResult command, please refer to 3.6.2.CaoController: : Execute("GetResult") command command.

Syntax WorkResetAsync (< lWorkNum >)

< lWorkNum > : [in] Work detection number (VT_I4)(0 to 15)

Return value : None

The following shows how to re-register the work detection base position asynchronously.

Example

```
Dim lWorkNum as long
Dim vntResult as variant

lWorkNum = 0

caoCtrl.Execute "WorkResetAsync", lWorkNum

' Obtain the return value of  WorkResetAsync command
vntResult = caoCtrl.Execute("GetResult")

' vntResult :  Empty
```

3.5.8. CaoController: : Execute("WorkResetEndAsync") command

Obtain the notification of the work detection base position re-registration completion asynchronously. To register the work detection base position again, you need to prepare programs designed for Robot-PV cooperation. Refer to 4.4 WorkReset command sequence for the command sequence.

To obtain and check the return value of the command, use GetResult command.For details about

GetResult command, please refer to 3.6.2.CaoController: : Execute(“GetResult”) command command.

Syntax WorkResetEndAsync ( )

Argument : None

Return value : None

The following shows how to obtain the notification of the work detection base position re-registration completion asynchronously.

Example

```
Dim vntResult as variant

caoCtrl.Execute "WorkResetEndAsync"

' Obtain the return value of  WorkResetEndAsync command
vntResult = caoCtrl.Execute("GetResult")

' vntResult :  Empty
```

3.5.9. CaoController: : Execute (“MoveEndAsync”) command

Notify PV of robot movement completion asynchronously.

To obtain and check the return value of the command, use GetResult command.For details about GetResult command, please refer to 3.6.2.CaoController: : Execute(“GetResult”) command command.

Syntax MoveEndAsync ( )

Argument : None

Return value : None

The following shows how to notify PV of robot movement completion asynchronously.

Example

```
Dim vntResult as variant

caoCtrl.Execute "MoveEndAsync"

' Obtain the return value of MoveEndAsync command
vntResult = caoCtrl.Execute("GetResult")

' vntResult :  Empty
```

3.5.10. CaoController: : Execute (“GetTeachPointAsync”) command

Obtain all teaching coordinate configured in PV asynchronously.

To obtain and check the return value of the command, use GetResult command.For details about GetResult command, please refer to 3.6.2.CaoController: : Execute(“GetResult”) command command.

Syntax GetTeachPointAsync ( )

Argument : None

Return value : None

The following shows how to request the teaching coordinates asynchronously.

Example

```
Dim vntResult as variant

caoCtrl.Execute "GetTeachPointAsync"

' Obtain the return value of GetTeachPointAsync command
vntResult = caoCtrl.Execute("GetResult")

' vntResult :  Robot coordinates information array (X, Y, Rz, Fig)
```

3.5.11. CaoController: : Execute("GetMovePointAsync") command

Obtain robot coordinates sent from the PV series during Auto calibration (CalibrationStart) or during re-registration of the work detection base position (WorkReset), asynchronously. To execute Auto calibration and re-registration of work detection base position, you need to prepare programs designed for Robot-PV cooperation. Refer to 4.3 CalibrationStart command sequence and 4.4 WorkReset command sequence.

To obtain and check the return value of the command, use GetResult command.For details about GetResult command, please refer to 3.6.2.CaoController: : Execute("GetResult") command command.

Syntax GetMovePoint ( )

Argument : None

Return value : Robot coordinates(X, Y, Rz, Fig)(VT_VARIANT)

The following shows how to obtain robot coordinates sent from PV asynchronously.

Example

```
Dim vntMovePos as variant

caoCtrl.Execute "GetMovePointAsync"

' Obtain the return value of GetMovePointAsync command
vntMovePos = caoCtrl.Execute("GetResult")

' vntMovePos :  Robot coordinates
```

3.6. General communication command (Asynchronous)

3.6.1. CaoController: : Execute("RawAsync") command

Send a command message asynchronously. BCC is calculated internally automatically.

To obtain and check the return value of the command, use GetResult command.For details about GetResult command, please refer to 3.6.2.CaoController: : Execute(“GetResult”) command command.

Syntax RawAsync (< bstrCmdMessage >)

< bstrCmdMessage >	:	[in] Command message to be transmitted (VT_BSTR)
Return value	:	[out] None

The following shows how to execute inspection with Common trigger, and with the execution mode of “Execute All” or “Automatic Switch”, asynchronously.

Example

```
Dim bstrParam as string
Dim vntResult as variant

bstrCmdMessage = "%S"

caoCtrl.Execute "RawAsync", bstrCmdMessage

' Obtain the return value of RawAsync command
vntResult = caoCtrl.Execute("GetResult")

' vntResult : Inspection result
```

3.6.2. CaoController: : Execute(“GetResult”) command

Wait the completion of an asynchronous command and obtain the return value.

There is no return value if the executed asynchronous command has no return value.

If an error occurs at an asynchronous command execution, the error is not issued during the asynchronous command execution. The error is issued at the GetResult command execution.

If there is no response within the specified timeout period during asynchronous command completion waiting, a timeout error (0x80000900) is issued. If this timeout error occurs, set longer timeout period by using SetTimeout command or an option of AddController.

Syntax GetResult ( )

Argument	:	None
Return value	:	[out]
		Return value of asynchronous command（VT_VARIANT）
		The return value depends on the executed command.

The following shows how to obtain the return value of asynchronous inspection.

Example

```
Dim vntResult as variant
caoCtrl.Execute "StartAsync"
```

```
vntResult = caoCtrl.Execute("GetResult")

vntResult : Inspection result
```

4. PV260 Robot calibration function

4.1. PV260 configuration

PV provider offers a command that realizes the cooperation with the Robot calibration function of PV260. To use this command, configure PV260 as follows.

1. [2.Robot setting] - [Robot communication] – Select [DENSO] from [Maker] (See Figure 2-1)

2. [2.Robot setting] – [Robot communication] – [Coordinate format] – [Hand system] – Select [0] from [Right-Hand] (See figure 2-2)

3. [2.Robot setting]-[Robot communication]-[Coordinate format]-[Hand system] – Select [1] from [Left-Hand] (See Figure 2-2)

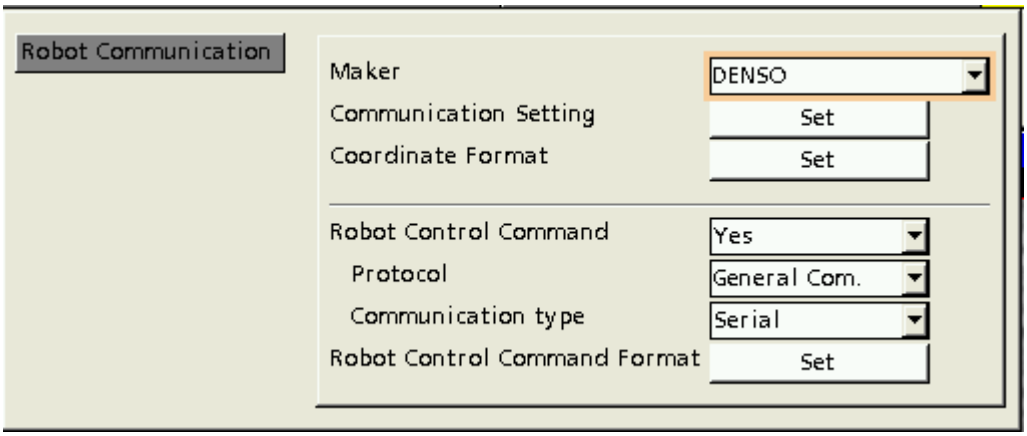

Figure 2-1. Robot communication – Maker

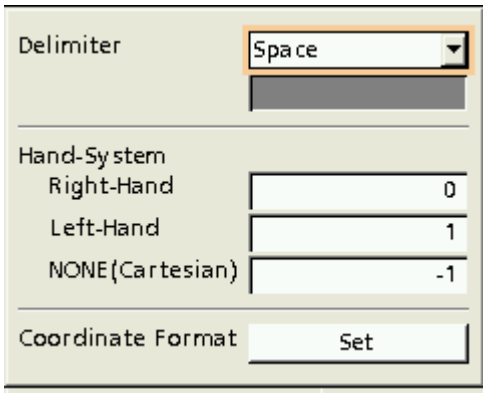

Figure 2-2. Robot communication – Hand system

4.2. Calibrate，Recalibrate command sequence

The following figure shows the sequence that executes Calibrate command and Recalibrate command in cooperation with PV and a robot.

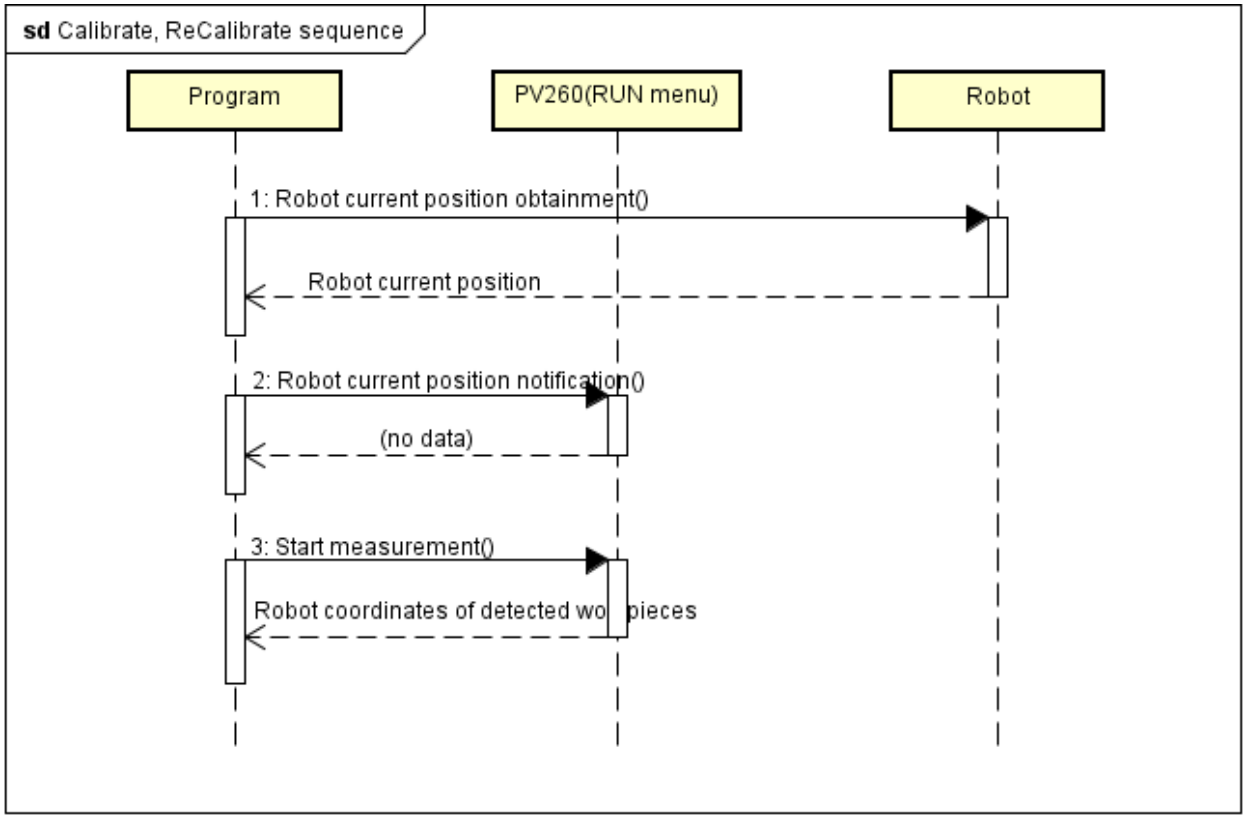

Figure 4-1 Calibrate，Recalibrate command sequence

Table 4-1 Calibrate，Recalibrate command sequence processing

Number in Figure 4-1	Processing	Compatible command	Remark
1	Robot current position obtainment	CurPos	PAC command
2	Robot current position notification	SetPoint	―
3	Start measurement	Calibrate / Recalibrate	―

The following shows a sample program written in PacScript. For an IP address, use the IP address specified by PV. This sample program uses the following IP address for connection.

IP address (PV) : 192.168.0.62

Sample ProCA.pcs

```
'!TITLE "ProCA"
#Include "Variant.h"
```

```
#Define Address "192.168.0.62"

Sub Main
      TakeArm

      Dim objPV As Object
      Dim vntRet As Variant
      Dim vntPos As Variant
      Dim li As Long
      Dim lPosBase As Position
      Dim lPosMove As Position

      lPosBase = CurPos
      Set objPV = Cao.AddController( "pv", "CaoProv.Panasonic.PV", "", "PV260=1,Conn=eth:" & Address )

      ' SetPoint
      Call objPV.SetPoint( VarChangeType( CurPos, VT_R4+VT_ARRAY ) )

      ' Calibrate
      vntRet = objPV.Calibrate( 0 )

      If UBound( vntRet ) >= 0 Then
          For li = 0 To UBound( vntRet )
              vntPos = vntret( li )
              If ( ( vntPos( 0 ) = 0 ) And ( vntPos( 1 ) = 0 ) And ( vntPos( 2 ) = 0 ) ) Then
                  PrintDbg "NotFound"
                  Exit Sub
              End If

              lPosMove = lPosBase
              LetX lPosMove = vntPos( 0 )
              LetY lPosMove = vntPos( 1 )
              LetRz lPosMove = vntPos( 2 )
              LetF lPosMove = vntPos( 3 )

              Move P, @E lPosMove
              Delay 500
          Next
      End If

      GiveArm
End Sub
```

4.3. CalibrationStart command sequence

The following figure shows the sequence that executes CalibrationStart command in cooperation with PV and a robot.

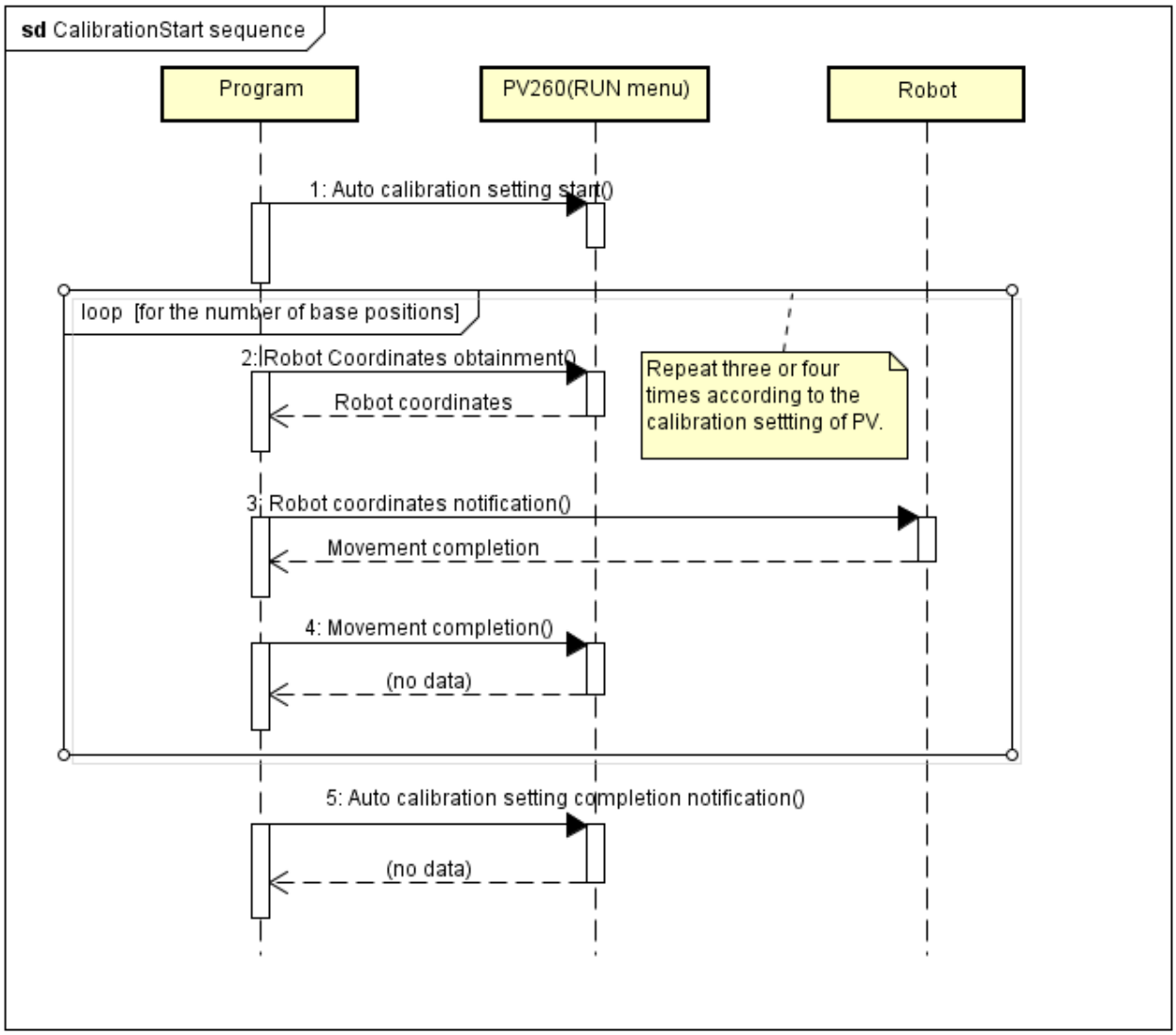

Figure 4-2 CalibrationStart command sequence

Table 4-2 CalibrationStart command sequence processing

Number in Figure 4-2	Processing	Compatible command	Remark
1	Auto calibration setting start	CalibrationStart	—
2	Robot coordinates obtainment	GetMovePoint	—
3	Robot coordinates notification	Move	PAC command
4	Movement completion	MoveEnd	—
5	Obtain the notification of Auto calibration completion	CalibrationEnd	—

The following shows a sample program written in PacScript. For an IP address, use the IP address specified by

PV. This sample program uses the following IP address for connection.

IP address (PV) : 192.168.0.62

Sample ProCAS.pcs

```
'!TITLE "ProCAS"
#include "Variant.h"

#define ADDRESS "192.168.0.62"
#define CAL_NO 0

Sub Main
	takearm

	Dim objPV as Object
	Dim vntVal as Variant
	Dim li as long
	Dim lpBase as Position
	Dim lpMove(3) as Position

	lpBase = CurPos

	set objPV = cao.AddController("pv", "CaoProv.Panasonic.PV", "", "PV260=1,Conn=eth:" & ADDRESS)

	' CalibrationStart
	call objPV.Execute("CalibrationStart", CAL_NO)

	' Obtain 3 points
	for li = 0 to 3
	    ' GetMovePoint
	    vntVal = objPV.Execute("GetMovePoint")

	    ' Copy the base position
	    lpMove(li) = lpBase

	    ' Specify the coordinates data
	    if (vartype(vntVal) And VT_ARRAY) then
	        LETX lpMove(li) = vntVal(0)
	        LETY lpMove(li) = vntVal(1)
	        LETRZ lpMove(li) = vntVal(2)
	        LETF lpMove(li) = vntVal(3)

	        Move P, @E lpMove(li)
	        delay 500
	    end if

	    ' MoveEnd
	    call objPV.Execute("MoveEnd")
	    delay 1000
	next

	' CalibrationEnd
	call objPV.Execute("CalibrationEnd")

	givearm
End Sub
```

4.4. WorkReset command sequence

The following figure shows the sequence that executes WorkReset command in cooperation with PV and a robot.

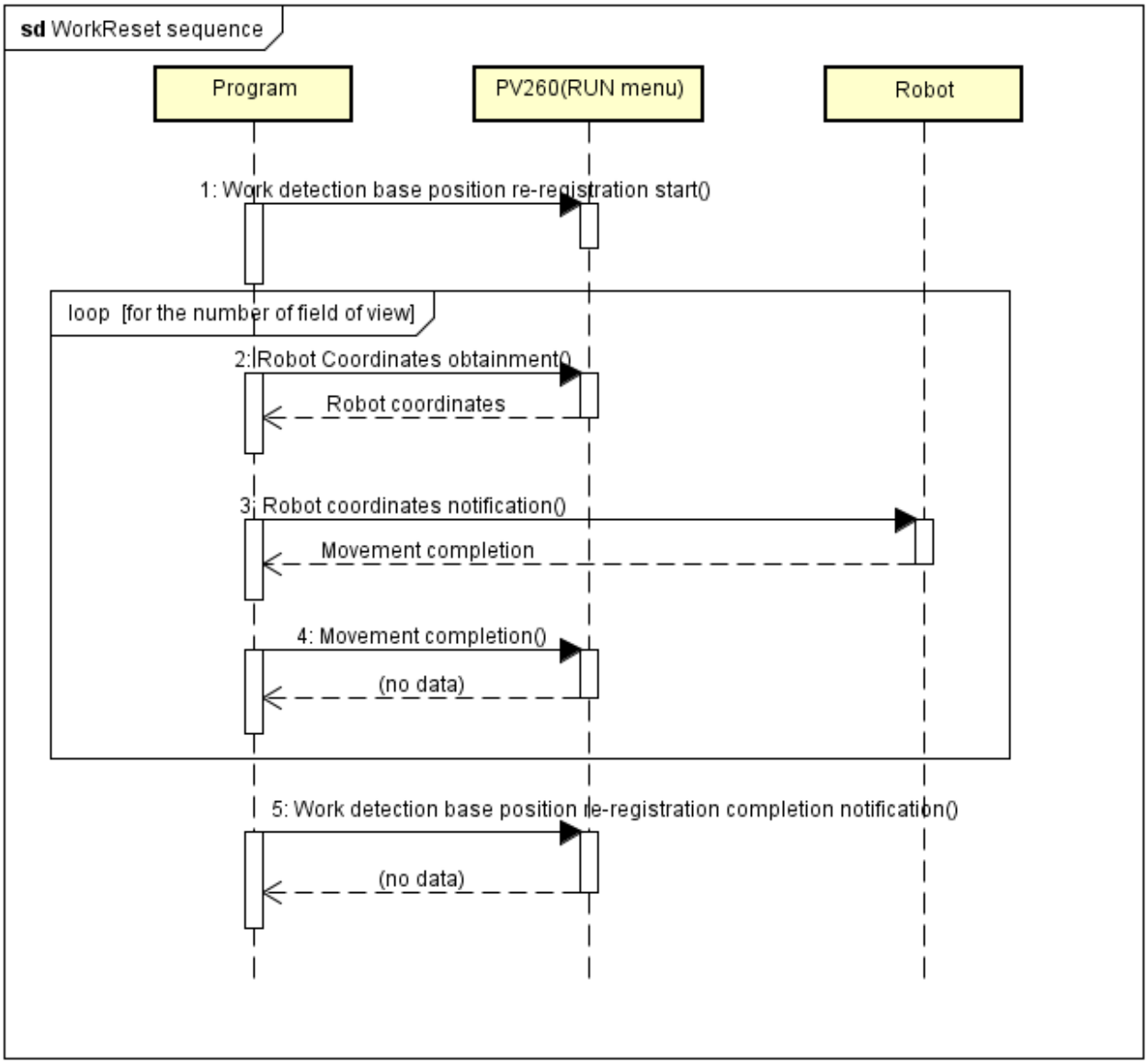

Figure 4-3 WorkReset command sequence

Table 4-3 WorkReset command sequence processing

Number in Figure 4-3	Processing	Compatible command	Remark
1	Work detection base position reregistration start	WorkReset	—
2	Robot coordinates obtainment	GetMovePoint	—
3	Robot coordinates notification	Move	PAC command

4	Movement completion	MoveEnd	—
5	Obtain the notification of the work detection base position reregistration	WorkResetEnd	—

The following shows a sample program written in PacScript. For an IP address, use the IP address specified by PV. This sample program uses the following IP address for connection.

IP address (PV) : 192.168.0.62

Sample ProWRS.pcs

```
'!TITLE "ProWRS"

#Include "Variant.h"

#Define ADDRESS "192.168.0.62"
#Define WORK_NO 0

Sub Main
    TakeArm

    Dim objPV As Object
    Dim vntVal As Variant
    Dim li As Long
    Dim lpBase As Position
    Dim lpMove( 1 ) As Position

    lpBase = CurPos

    Set objPV = Cao.AddController( "pv", "CaoProv.Panasonic.PV", "", " PV260=1,Conn=eth:" & ADDRESS )

    ' WorkReset
    Call objPV.Execute( "WorkReset", WORK_NO )

    ' Obtain 2 points
    For li = 0 To 1
        ' GetMovePoint
        vntVal = objPV.Execute( "GetMovePoint" )

        ' Copy the base position
        lpMove( li ) = lpBase

        ' Specify the coordinates data
        If ( VarType( vntVal ) And VT_ARRAY ) Then
            LetX lpMove( li ) = vntVal( 0 )
            LetY lpMove( li ) = vntVal( 1 )
            LetRz lpMove( li ) = vntVal( 2 )
            LetF lpMove( li ) = vntVal( 3 )

            Move P, @E lpMove( li )
            Delay 500
        End If

        ' MoveEnd
        Call objPV.Execute( "MoveEnd" )
        Delay 1000
    Next

    ' CalibrationEnd
```

```
        Call objPV.Execute( "WorkResetEnd" )

        GiveArm
    End Sub
```

Appendix A. Command corresponding table

Table A-1 Command corresponding table

		Image processing device		
PV command	Provider command	PV200	PV500	PV260
PV series compatible command				
%S	Start	✓	✓	✓
%R	Restart	✓	✓	✓
%X	Xtype	✓	✓	✓
%MW	MemoryWrite	✓	✓	✓
%CW	CFWrite	✓	✓	✓
%MR	MemoryRead	✓	✓	✓
%CR	CFRead	✓	✓	✓
%CD	CancelData	✓	✓	✓
%SS	SDSave	✓	✓	✓
%SR	SDReset	✓	✓	✓
%PS	PrintScreen	✓	✓	✓
%Q	Quit	✓	✓	✓
%RM	RunManual	✓	✓	✓
%E	ErrorReset	✓	✓	✓
%CC	Cancel	✓	✓	✓
%K	KeyEmulator	✓	✓	✓
%BS	Bstop	✓	✓	✓
%BC	Bconfirm	✓	✓	✓
%I	LayoutChange	✓	✓	✓
%A	AgainTemplate	✓	✓	✓
%PR	ParameterRead	✓	✓	✓
%PRP	ParameterReadPair	✓	✓	✓
%PW	ParameterWrite	✓	✓	✓
%PWP	ParameterWritePair	✓	✓	✓
%P=	SetPoint	-	-	✓
%CA	Calibrate	-	-	✓
%RCA	Recalibrate	-	-	✓
%CAS	CalibrationStart	-	-	✓

―	CalibrationEnd	-	-	✓
%WCS	WorkSet	-	-	✓
%WRS	WorkReset	-	-	✓
―	WorkResetEnd	-	-	✓
%MVE	MoveEnd	-	-	✓
%TCD	GetTeachPoint	-	-	✓
―	GetMovePoint	-	-	✓
Original command				
―	Raw	✓	✓	✓
―	SetTimeout	✓	✓	✓
―	GetTimeout	✓	✓	✓
―	GetResult	✓	✓	✓
PV series compatible command (Asynchronous)				
%S	StartAsync	✓	✓	✓
%R	RestartAsync	✓	✓	✓
%X	XtypeAsync	✓	✓	✓
%MW	MemoryWriteAsync	✓	✓	✓
%CW	CFWriteAsync	✓	✓	✓
%MR	MemoryReadAsync	✓	✓	✓
%CR	CFReadAsync	✓	✓	✓
%CD	CancelDataAsync	✓	✓	✓
%SS	SDSaveAsync	✓	✓	✓
%SR	SDResetAsync	✓	✓	✓
%PS	PrintScreenAsync	✓	✓	✓
%Q	QuitAsync	✓	✓	✓
%RM	RunManualAsync	✓	✓	✓
%E	ErrorResetAsync	✓	✓	✓
%CC	CancelAsync	✓	✓	✓
%K	KeyEmulatorAsync	✓	✓	✓
%BS	BstopAsync	✓	✓	✓
%BC	BconfirmAsync	✓	✓	✓
%I	LayoutChangeAsync	✓	✓	✓
%A	AgainTemplateAsync	✓	✓	✓
%PR	ParameterReadAsync	✓	✓	✓
%PRP	ParameterReadPairAsync	✓	✓	✓

%PW	ParameterWriteAsync	✓	✓	✓
%PWP	ParameterWritePairAsync	✓	✓	✓
%P=	SetPointAsync	-	-	✓
%CA	CalibrateAsync	-	-	✓
%RCA	RecalibrateAsync	-	-	✓
%CAS	CalibrationStartAsync	-	-	✓
―	CalibrationEndAsync	-	-	✓
%WCS	WorkSetAsync	-	-	✓
%WRS	WorkResetAsync	-	-	✓
―	WorkResetEndAsync	-	-	✓
%MVE	MoveEndAsync	-	-	✓
%TCD	GetTeachPointAsync	-	-	✓
―	GetMovePointAsync	-	-	✓
Original command (Asynchronous)				
―	RawAsync	✓	✓	✓