DENSO

DENSO Robotics

THIRD PARTY PRODUCTS

PROVIDER MANUAL

Maker

Panasonic Industrial Devices SUNX

Products／Series

Vision Sensor

MODEL: PV Series

Vision

Introduction

This document is a user's manual for the provider to use "Panasonic Industrial Devices SUNX Vision Sensor PV Series" connected to the DENSO robot controller RC8 series. Note that some functions may be unavailable on old PV models. For details and handling of the connected device, refer to the user’s manual of "Panasonic Industrial Devices SUNX Vision Sensor PV Series".

Caution: (1) Note that the functions and performance cannot be guaranteed if this product is used without observing instructions in this manual.
(2) All products and company names mentioned are trademarks or registered trademarks of their respective holders.

This manual covers the following product

Panasonic Industrial Devices SUNX PV200/PPV500 Series

Important

To ensure proper and safe operation, be sure to read "Safety Precautions Manual" before using the provider.

Notice to Customers

1. Risks associated with using this product

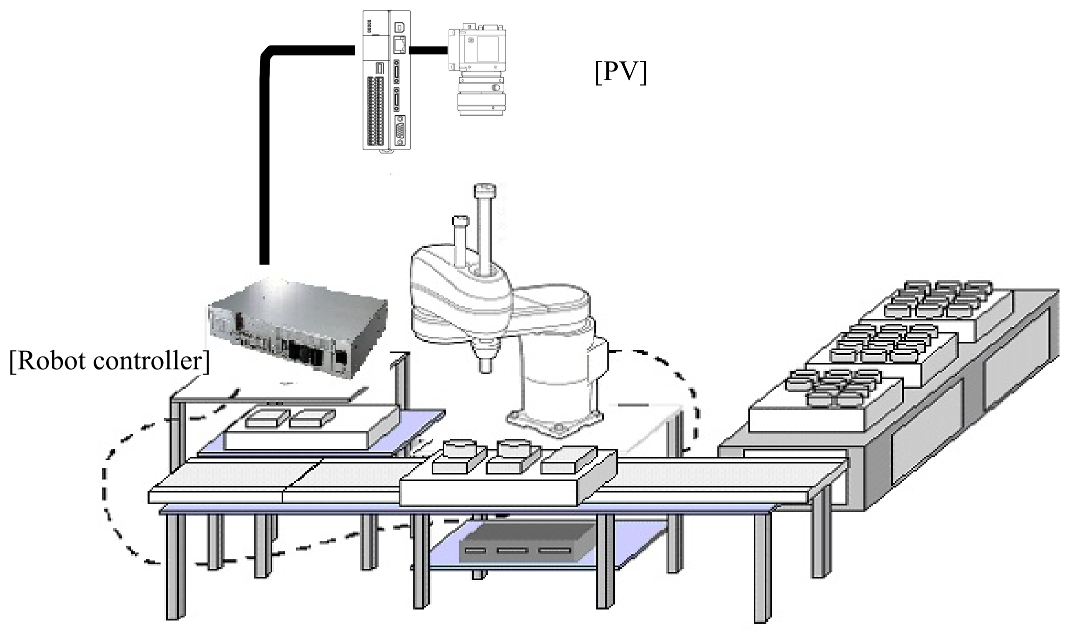
The user of this product shall be responsible for embedding and using the product (software) on a system and any result from using it.

Contents

1. Outline of This Product (Provider)

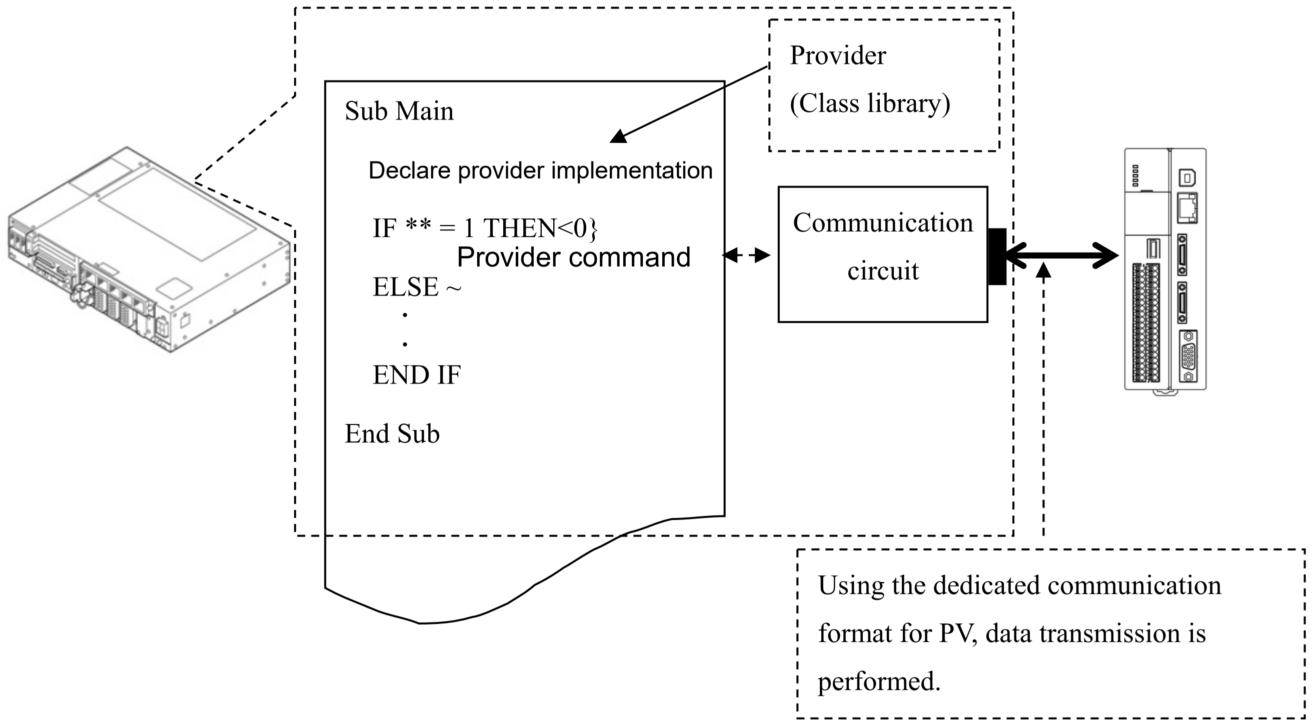
1.1 Target device of provider

This provider can be used only when a DENSO robot controller (RC8 series) is connected to the PV series.



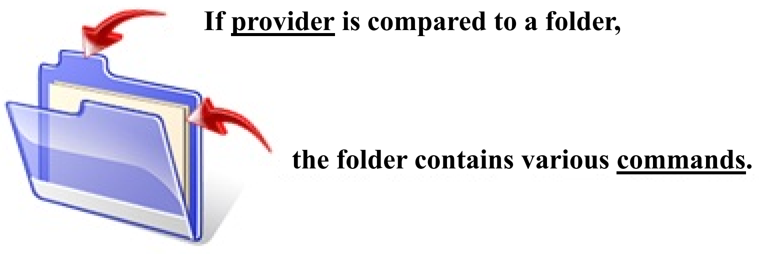
1.2 Features of provider

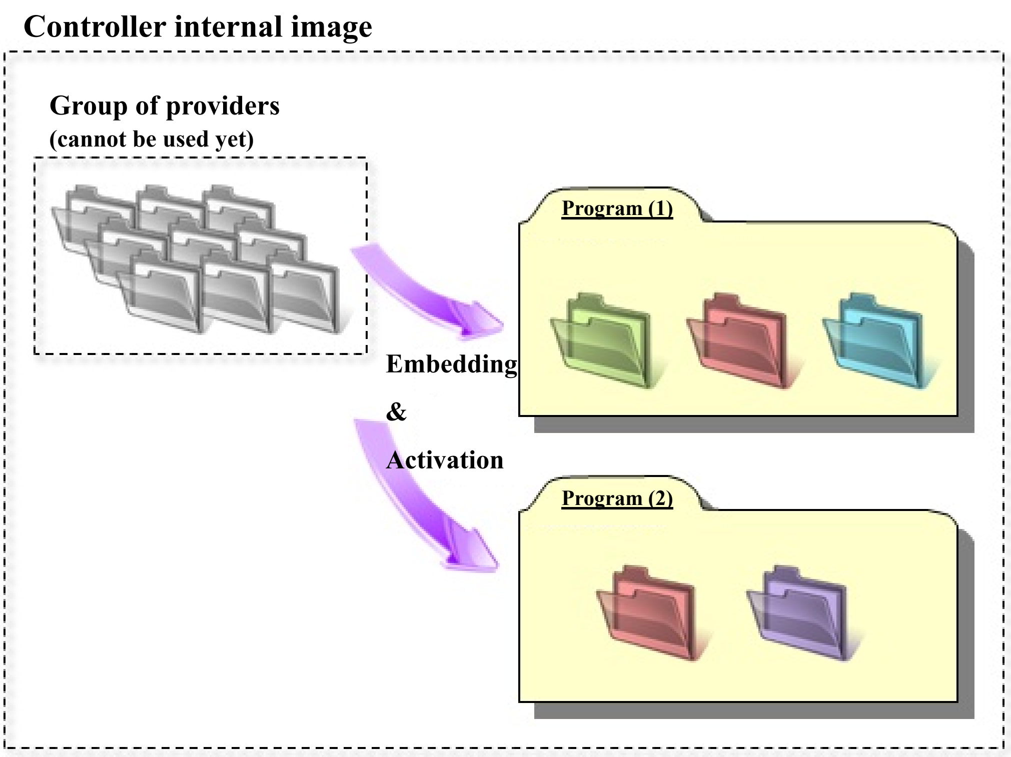
This provider is provided to use the PV native commands required to access PV series in the robot program. Use of this provider allows customers to establish communication with a robot easily without creating a communication program for PV series. The following shows a diagram of provider embedding.


Provider
(Class library)
Sub Main
Declare provider implementation
IF ** = 1 THEN<0}
Provider command
ELSE ~
END IF
End Sub
Communication circuit
Using the dedicated communication format for PV, data transmission is performed.

1.3 Mechanism of provider

This provider offers various programs required to control the target device as a single provider. Just activate the license to use the provider. Once provider implementation is declared on a desired program file, the functions prepared by the provider can be used as commands in the user program. Since the provider is included in the controller, there is no need of installation. Also, it is possible to implement multiple providers of different type. Note that a program (procedure) cannot contain the providers of the same type.


If provider is compared to a folder,
the folder contains various commands.



Provider prepared in the system. This cannot be used yet.

Provider after embedding. This can be used in a provider-embedded program.
Different colors are used to indicate the provider type.

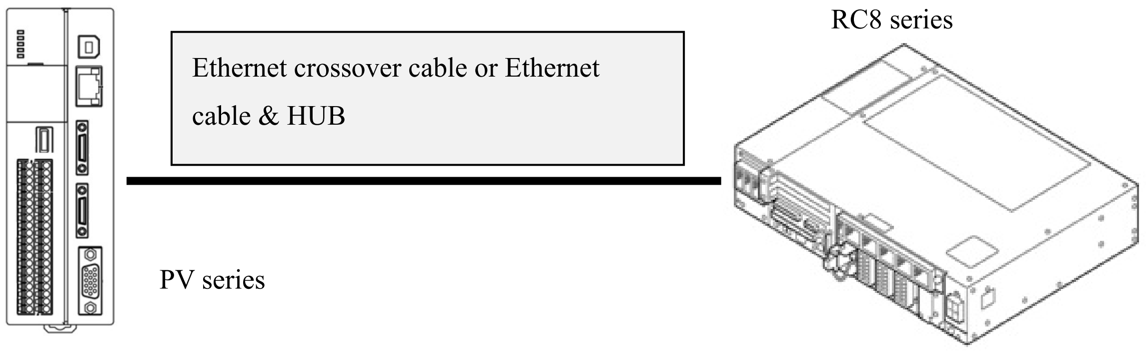
Note: When the same provider exists in different programs like in the above figure, exclusion process is required between the programs (tasks).

* The provider is provided as a dynamic link library (abbreviated as DLL) which can be used from PacScript.

2. How to Connect

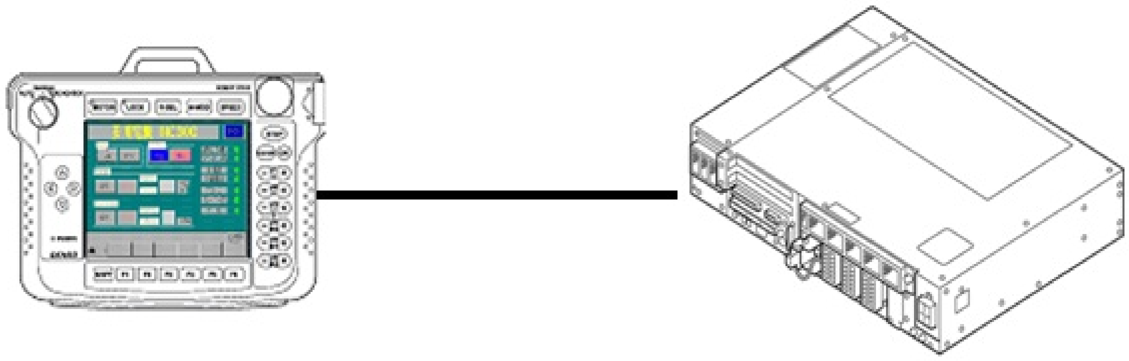
2.1 Ethernet (TCP/IP) connection example

To connect the robot controller to PV series via Ethernet (TCP/IP), use an Ethernet crossover cable. Also, when a switching hub/router is used, use the cable suitable for the switching hub/router specifications.



3. Communication Settings for Robot Controller and Device Used

Use a teach pendant to adjust the communication settings for the device to be used.

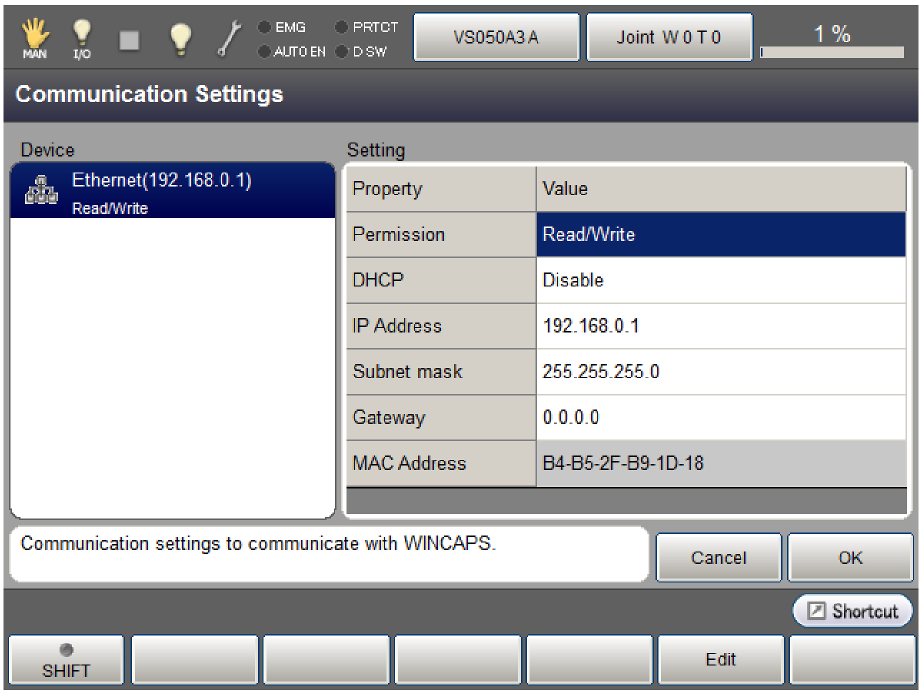


3.1 Communication via Ethernet (TCP/IP)

3.1.1 Ethernet (TCP/IP) communication settings on robot controller

Set the robot controller's IP address.

Press [F6 Setting] - [F5 Communication and Token] - [F2 Network and Permission] to display the [Communication Settings] window. Set the IP address and subnet mask so that the robot controller and PV are within the same subnet mask.



Property	Value
Permission	Read/Write
DHCP	Disable
IP Address	192.168.0.1
Subnet mask	255.255.255.0
Gateway	0.0.0.0
MAC Address	B4-B5-2F-B9-1D-18

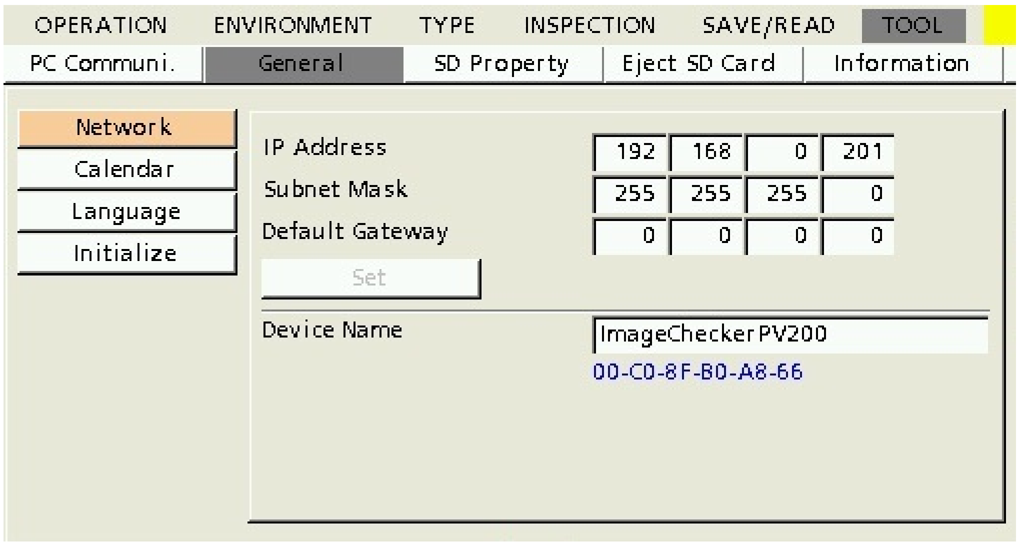
Communication settings to communicate with WINCAPS.

3.1.2 Ethernet (TCP/IP) communication settings for PV

Set the PV's IP address.

Select [TOOL] – [Network]. Set the IP address and subnet mask on the [Network Settings] screen. Set the IP address and subnet mask so that the robot controller and PV are within the same subnet mask.

[Screen of PV]



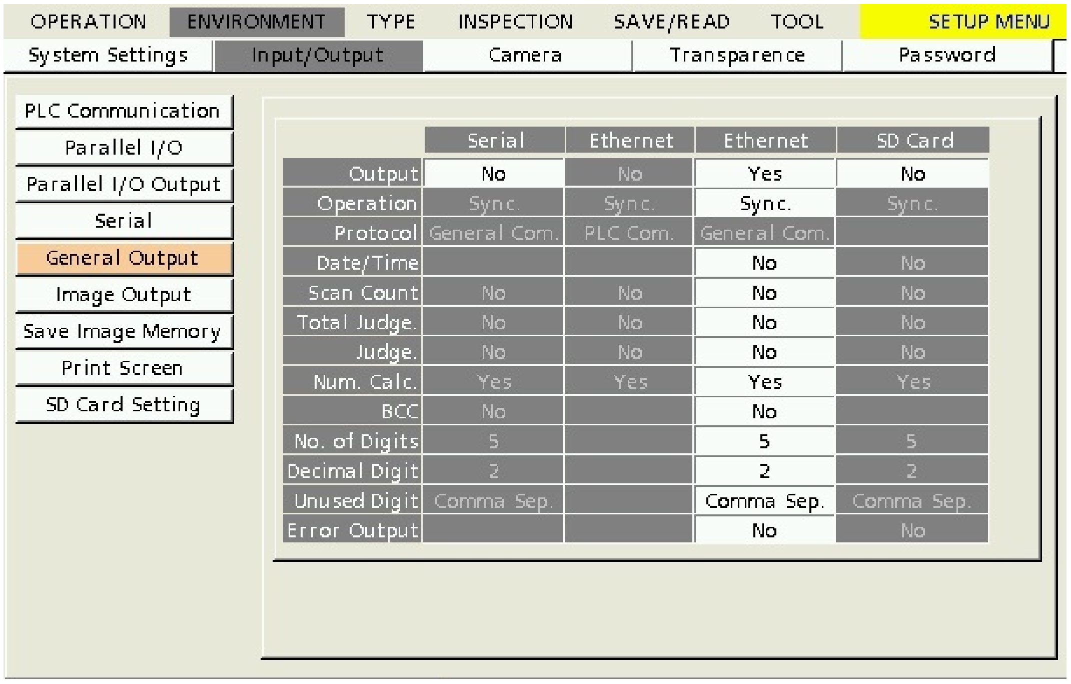
IP Address	192	168	0	201
Subnet Mask	255	255	255	0
Default Gateway	0	0	0	0

Device Name ImageChecker PV200

00-C0-8F-B0-A8-66

3.1.3 Result output settings for PV

Select [ENVIRONMENT] – [Input/Output] – [General Output]. Configure the Ethernet general output (protocol) settings. Refer to the Panasonic PV user's manual for details about settings.



	Serial	Ethernet	Ethernet	SD Card
Output	No	No	Yes	No
Operation	Sync.	Sync.	Sync.	Sync.
Protocol	General Com.	PLC Com.	General Com.	
Date/Time			No	No
Scan Count	No	No	No	No
Total Judge.	No	No	No	No
Judge.	No	No	No	No
Num. Calc.	Yes	Yes	Yes	Yes
BCC	No		No	
No. of Digits	5		5	5
Decimal Digit	2		2	2
Unused Digit	Comma Sep.		Comma Sep.	Comma Sep.
Error Output			No	No

4. Provider Execution Procedure

The basic process of the provider is implementation (declaration) -> execution. This provider takes a connection process at the time of implementation. The operation can be repeated as many times as needed. A program example is shown below.

```
Sub Main

On Error Goto ErrorProc       (1)                              'Declare error process routine
Dim caoCtrl as Object         (2)                              'Declare provider variable
Dim strResult as String       (3)                              'Declare result acquisition variable

caoCtrl = cao.AddController("PV", "caoProv.Panasonic.PV", "", "conn=eth:192.168.0.201") (4)

"State from trigger to data receiving process"      (5)


EndProc:
    'End process
    Exit Sub


ErrorProc:
    'Error process


End Sub
```

(1) Declare the provider error processing routine as needed. (Connection error detection at declaration)
(2) Declare the provider implementation variable as Object type. The variable name can be specified arbitrarily.
(3) Declare the result acquisition variable. The data type depends on the command.
(4) Execute implementation with the provider declaration command cao.AddController. The parameters required for settings vary by provider. From this point the provider commands are available using the implementation variable caoCtrl.
(5) Now the program can be stated using the provider commands.

5. Command Description

This page contains a description of commands. The commands are classified into connection commands, PV commands, and proprietary extension commands. For the detailed operation of PV commands, refer to the manual of general-purpose communication command details for Panasonic PV series.

Table 5-1 List of commands

command	PV command	Usage	PV260	Refer to
Connection commands				
cao.AddController	—	Implements the provider to a variable and makes a connection to PV.		13
Addvariable	—	Creates a variable used exclusively for acquiring image or cell value.		14
Value	—	Acquires data through the variable created by AddVariable.		15
PV commands				
Start	%S	Executes test.		16
Restart	%R	Re-executes test (Test using the current memory image without capturing another image).		17
Xtype	%X	Changes the product type.		18
MemoryWrite	%MW	Saves setting data to main unit storage memory.		19
CFWrite	%CW	Saves setting data to SD card memory.		20
MemoryRead	%MR	Reads setting data from main unit storage memory.		21
CFRead	%CR	Reads setting data from SD card memory.		22
CancelData	%CD	Cancels saving or reading setting data.		23
SDSave	%SS	Saves storage image memory (SD memory card).		24
SDReset	%SR	Deletes storage image memory.		25
PrintScreen	%PS	Prints the screen.		26
Quit	%Q	Resets statistics.		27
RunManual	%RM	Switches between run and stop.		28
ErrorReset	%E	Resets error signal.		29
Cancel	%CC	Cancels test/processing (cancels various operations).		30
KeyEmulator	%K	Emulates keys.		31
Bstop	%BS	Keypad operations available/unavailable.		32
Bconfirm	%BC	Checks that keypad operations are available.		33
LayOutChange	%I	Changes the layout.		34
AgainTemplate	%A	Makes a re-entry of template.		35
ParameterRead	%PR	Reads parameters.		36
ParameterReadPair	%PRP	Reads parameter pairs (such as upper and lower limits).		37
ParameterWrite	%PW	Changes parameters.		38
ParameterWritePair	%PWP	Changes parameter pairs (such as upper and lower limits).		39
PV General communication command (Asynchronous)				
StartAsync	%S	Executes test.		40
ReStartAsync	%R	Re-executes test (Test using the current memory image without capturing another image).		41

XTypeAsync	%X	Changes the product type.		42
MemoryWriteAsync	%MW	Saves setting data to main unit storage memory.		43
CFWriteAsync	%CW	Saves setting data to SD card memory.		44
MemoryReadAsync	%MR	Reads setting data from main unit storage memory.		45
CFReadAsync	%CR	Reads setting data from SD card memory.		46
CancelDataAsync	%CD	Cancels saving or reading setting data.		47
SDSaveAsync	%SS	Saves storage image memory (SD memory card).		48
SDResetAsync	%SR	Deletes storage image memory.		49
PrintScreenAsync	%PS	Prints the screen.		50
QuitAsync	%Q	Resets statistics.		51
RunManualAsync	%RM	Switches between run and stop.		52
ErrorResetAsync	%E	Resets error signal.		53
CancelAsync	%CC	Cancels test/processing (cancels various operations).		54
KeyEmulatorAsync	%K	Emulates keys.		55
BstopAsync	%BS	Keypad operations available/unavailable.		56
BconfirmAsync	%BC	Checks that keypad operations are available.		57
LayOutChangeAsync	%I	Changes the layout.		58
AgainTemplateAsync	%A	Makes a re-entry of template.		59
ParameterReadAsync	%PR	Reads parameters.		60
ParameterReadPairAsync	%PRP	Reads parameter pairs (such as upper and lower limits).		61
ParameterWriteAsync	%PW	Changes parameters.		62
ParameterWritePairAsync	%PWP	Changes parameter pairs (such as upper and lower limits).		63
Original command				
Raw	—	Sends and receives command messages.		64
SetTimeout	—	Set the communication timeout period.		65
GetTimeout	—	Obtain the communication timeout period.		66
Original command (Asynchronous)				
RawAsync	—	Asynchronous command message sending.		67
GetResult	—	Asynchronous command execution result obtainment.		68
PV260 Robot calibration command (Synchronous)				
SetPoint	%P=	Robot coordinates acknowledged.	X	69
Calibrate	%CA	Measurement start command.	X	70
ReCalibrate	%RCA	Re-measurement start command.	X	71
CalibrationStart	%CAS	Auto calibration setting start.	X	72
CalibrationEnd	—	Auto calibration setting completion notification reception .(CalibrationStart-related command)	X	73
WorkSet	%WCS	Work detection base position reregistering.	X	74
WorkReset	%WRS	Work detection base position reregistering start.	X	75
WorkResetEnd	—	Work detection base position reregistration completion notification reception. (WorkReset-related command)	X	76
MoveEnd	%MVE	Movement completion notification.	X	77
GetTeachPoint	%TCD	Teaching coordinate request.	X	78

GetMovePoint	—	Robot coordinates obtainment. (CalibrationStart, WorkReset-related command)	X	79
PV260 Robot calibration command (Asynchronous)				
SetPointAsync	%P=	Robot coordinates acknowledged.	X	80
CalibrateAsync	%CA	Measurement start command.	X	81
ReCalibrateAsync	%RCA	Re-measurement start command.	X	82
CalibrationStartAsync	%CAS	Auto calibration setting start.	X	83
CalibrationEndAsync	—	Auto calibration setting completion notification reception .(CalibrationStart-related command)	X	84
WorkSetAsync	%WCS	Work detection base position reregistering.	X	85
WorkResetAsync	%WRS	Work detection base position reregistering start.	X	86
WorkResetEndAsync	—	Work detection base position reregistration completion notification reception. (WorkReset-related command)	X	87
MoveEndAsync	%MVE	Movement completion notification.	X	88
GetTeachPointAsync	%TCD	Teaching coordinate request.	X	89
GetMovePointAsync	—	Robot coordinates obtainment. (CalibrationStart, WorkReset-related command)	X	90

Commands with "X" on the PV260 column are PV260-dedicated commands.

Following abbreviated expressions are used for the command descriptions in this manual.

<Implementation variable>:<ImplVar>

<Property variable>:<PropVar>

cao.AddController

Usage

Implements the provider to a variable and makes a connection to PV.

Syntax

cao.AddController <Controller name>,<Provider name>,
<Provider running machine name>,[<Option>]

Argument:
<Controller name> Assign a name (The name is used for control).
<Provider name> "CaoProv.Panasonic.PV"
<Provider running machine name> Omit this parameter.
<Option> [PV260 parameter], [Connection parameter], [Timeout period], [IP Address: port]

[PV260 parameter] Specify this parameter if you use a robot calibration-related command of PV260. This Option is available in Ver.1.12.* or later.
0 : Do not use a robot calibration-related command (default)
1 : Use a robot calibration-related command.
"PV260=0" or "PV260=1"
[Connection parameter] "conn=eth:<IP address>"
[Timeout period] Specify the timeout period (msec) for transmission.
"Timeout[=<Time>]" Default: 500
[IP Address: port] 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.
This Option is available in Ver.2.3.* or later.
"MyIP=[<Local IP address>[:Local port No]] "

Description

The provider becomes effective when implemented to a variable. From this point the implemented Object type variable is used to access the provider. (The implemented variable is called "Implementation Variable".)

Example

```
Dim caoCtrl as Object

caoCtrl = cao.AddController("PV", "caoProv.Panasonic.PV", "", "PV260=1, conn=eth:192.168.0.201")


* Specify a timeout period as follows:
caoCtrl = cao.AddController("PV", "CaoProv.Panasonic.PV", "", _
                            "PV260=1, conn=eth:192.168.0.201, Timeout = 1000")
```

<ImplVar>.AddVariable

Usage Creates a property variable used for acquiring images.

Syntax **<ImplVar>.AddVariable** <Specify image>, [<Option>]

Argument: <Specify image> Specify the type of image to be acquired.
@BITMAP: Camera image
@BITMAP_MONITOR: Monitor image
<Option> None

Description An Object type variable is created to acquire image from PV.

Example

The following shows an example of acquiring an image And display it On the operation panel screen.

```
Dim caoCtrl As Object
Dim objBmp As Object
Dim vntResult as Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )

objBmp = caoCtrl.AddVariable( "@BITMAP", "" )
vntResult = objBmp.Value
```

<PropVar>.Value

Usage

Acquires image data through the variable created by AddVariable.

Syntax

<PropVar>.Value( )

Return value: BITMAP formatted data.

Description

Image data is acquired from the provider (implementation variable) through the variable created by AddVariable.

Example

The following shows an example of acquiring an image and display it on the operation panel.

```
Dim caoCtrl As Object
Dim objBmp As Object
Dim vntResult as Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )

objBmp = caoCtrl.AddVariable( "@BITMAP", "" )
vntResult = objBmp.Value
```

<ImplVar>.Start

Usage

Executes testing. The syntax for the "Execute All " or "Automatic Switch " mode is different from that for the "Specified execution" mode. Settings of "General Result Output" on PV series are returned as a character string for image processing result.

Syntax

<ImplVar>.Start(<Block No.> **)**

Argument: [Block No.] Execution target block No. (integer 0 to 9)

Return value: Image processing result (character string)

Description

A block No. is required as an argument only when a batch trigger is used with the execution mode set to " User-Defined".
Argument is not required for the case of " Execute All" or " Automatic Switch" mode.

Example

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

```
Dim caoCtrl As Object
Dim strResult As String

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
strResult = caoCtrl.Start( 1 )                'User-Defined
```

The following example shows how to execute inspection.

```
Dim caoCtrl As Object
Dim strResult As String

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
strResult = caoCtrl.Start                     'Execute All or Automatic Switch
```

<ImplVar>.Restart

Usage

Executes testing without capturing image. Different syntax is used for "Execute All " or "Automatic Switch " mode and "Specified execution" mode.

Syntax

<ImplVar>.Restart(<Block No.> **)**

Argument: [Block No.] Execution target block No. (integer 0 to 9)

Return value: Image processing result (character string)

Description

A block No. is required as an argument only when a batch trigger is used with the execution mode set to " User-Defined".
Argument is not required for the case of " Execute All" or " Automatic Switch" mode.

Example

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

```
Dim caoCtrl As Object
Dim strResult As String

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
strResult = caoCtrl.Restart( 1 )                'User-Defined
```

The following example shows how to execute re-inspection..

```
Dim caoCtrl As Object
Dim strResult As String

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
strResult = caoCtrl.Restart                    'Execute All or Automatic Switch
```

<ImplVar>.Xtype

Usage Changes the product type.

Syntax **<ImplVar>.Xtype** <Product No.>

Argument: <Product No.> (Integer 0 to 255)

Description The product type is changed.

Example

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

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.Xtype 100
```

<ImplVar>.MemoryWrite

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

Syntax
<ImplVar>.MemoryWrite [<Area No.>]

Argument: <Area No.> Specify the saving area No. of SD memory card.

PV200	None	
PV500	<Area No.>	(integers 0 to 99)

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

Example

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

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.MemoryWrite
```

<ImplVar>.CFWrite

Usage Saves the setting data in an SD memory card.

Syntax **<ImplVar>.CFWrite** <Area No.>

Argument: <Area No.> Specify the saving area No. of SD memory card. (integers 0 to 99)

Description Setting data is saved to an SD memory card after specifying the area No.

Example

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

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.CFWrite 10
```

<ImplVar>.MemoryRead

Usage

Reads the setting data from the main unit memory.

Syntax

<ImplVar>.MemoryRead [<Area No.>]

Argument: <Area No.> Specify the reading area No. of SD memory card.

PV200	None	
PV500	<Area No.>	(integers 0 to 99)

Description

Setting data is read from the main unit memory after specifying the area No.

Example

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

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.MemoryRead
```

<ImplVar>.CFRead

Usage

Reads the setting data from an SD memory card.

Syntax

<ImplVar>.CFRead <Area No.>

Argument: <Area No.> Specify the reading area No. of SD memory card.
(Integer 0 to 99)

Description

Setting data is read from an SD memory card after specifying the area No.

Example

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

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.CFRead 10
```

<ImplVar>.CancelData

Usage Cancels saving/reading the setting data.

Syntax **<ImplVar>.CancelData**

Description Saving/reading process of the setting data is cancelled.

Example

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

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.CancelData
```

<ImplVar>.SDSave

Usage Saves the image memory stored in the main unit to an SD memory card.

Syntax **<ImplVar>.SDSave**

Description The image memory stored in the main unit is saved to an SD memory card.
An unused number on the SD memory card is searched for and used as the save destination. (The save destination number cannot be specified.)

Example

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

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.SDSave
```

<ImplVar>.SDReset

Usage

Deletes the image memory stored in the main unit.

Syntax

<ImplVar>.SDReset

Description

The image memory stored in the main unit is deleted.
The operation is the same as when selecting [Save/Read] -> [Delete Image Memory] in the setting screen.

Example

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

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.SDReset
```

<ImplVar>.PrintScreen

Usage Captures the screen currently displayed (all items displayed) and stores it to an SD memory card or to a PC via Ethernet interface.

Syntax **<ImplVar>.PrintScreen**

Description The current screen display is captured and stored to an SD memory card or PC via Ethernet interface.
Data is saved to the location specified in [Output Destination] when selecting [ENVIRONMENT] – [Input/Output] – [Print Screen]. The output destination cannot be specified for this command.

Example

The following example shows how to save the current displays.

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.PrintScreen
```

<ImplVar>.Quit

Usage

Clears the statistics data and scanning count.

Syntax

<ImplVar>.Quit

Description

The statistics data and scanning count are cleared.

Example

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

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.Quit
```

<ImplVar>.RunManual

Usage Switches the operation mode of PV series between run and stop.

Syntax **<ImplVar>.RunManual(** <Mode> **)**

Argument: <Mode> Switching between run and stop (integer).
0: Switches to run mode.
1: Switches to stop mode.
Return value: Selected mode value (integer).
0: Run
1: Stop

Description The operation mode of PV series is switched between run and stop.

Example

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

```
Dim caoCtrl As Object
Dim iResult As Integer

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
iResult = caoCtrl.RunManual( 1 )
```

<ImplVar>.ErrorReset

Usage Resets the Error signal.

Syntax **<ImplVar>.ErrorReset**

Description Resets the Error signal.

Example

The following example shows how to clear errors.

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.ErrorReset
```

<ImplVar>. Cancel

Usage Cancels the process currently being executed.

Syntax **<ImplVar>.Cancel**

Description The process currently being executed is cancelled to return to the state before starting the process.

Example

The following example shows how to cancel motion execution.

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.Cancel
```

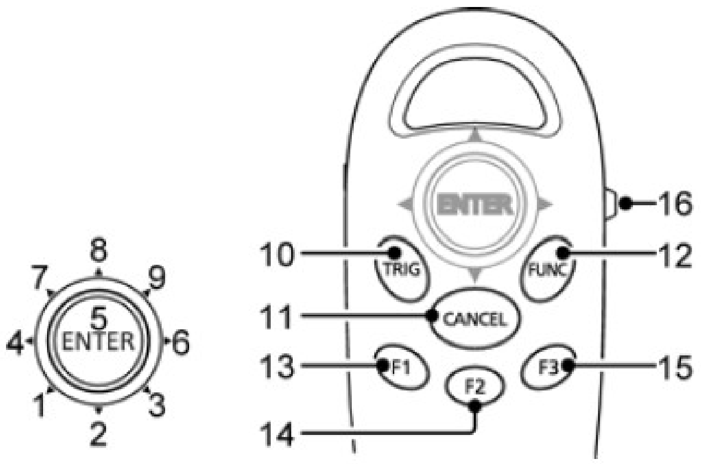
<ImplVar>.KeyEmulator

Usage Executes operations as keypad operations.

Syntax **<ImplVar>.KeyEmulator** <Shift>, <Key>

Argument: <Shift> Shift key ON/OFF (integer 0, 1).
0: OFF
1: ON
<Key> Value allocated to each key (integer 1 to 16).
See the following figure for details.

Description Operations are executed as keypad operations.
No response is made from PV series.



Example

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

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.KeyEmulator 0, 16
```

<ImplVar>.Bstop

Usage

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

Syntax

<ImplVar>.Bstop <Availability>

Argument: <Availability> Keypad operation permission (integer 0, 1).
0: Permit
1: Refuse

Description

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

Example

The following shows how to refuse the keypad operation.

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.Bstop 1
```

<ImplVar>.Bconfirm

Usage Get the current state of a keypad operation permission.

Syntax **<ImplVar>.Bconfirm( )**

Return value: Keypad operation availability status (integer 0, 1).
0: Permission
1: Refuse

Description Get the current state of a keypad operation permission.

Example

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

```
Dim caoCtrl As Object
Dim iResult As Integer

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
iResult = caoCtrl.Bconfirm
```

<ImplVar>.LayoutChange

Usage

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

<ImplVar>.LayoutChange <Layout No.>

Argument: <Layout No.> Specify with an integer (0 to 15).

Description

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

Example

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

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.LayoutChange 1
```

<ImplVar>.AgainTemplate

Usage

Re-register the template of the smart matching checker.

Syntax

<ImplVar>.AgainTemplate <Checker No.> **,** <Template No.>

Argument: <Checker No.> Specify with an integer (0 to 999).
<Template No.> Specify with an integer (0 to 63).

Description

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.

Example

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

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.AgainTemplate 1, 10
```

<ImplVar>.ParameterRead

Usage Reads the settings or system values from the PV series main unit.

Syntax **<ImplVar>.ParameterRead(** <Parameter> **)**

Argument: <Parameter> Specify with a character string.

Return value: Specified parameter value (character string).

Description Settings or system values are read from the PV series main unit. This command is effective during operation only. For the data to read and command parameters, refer to the user's manual of Panasonic PV series.

Example

The following example shows how to readout the current time.

```
Dim caoCtrl As Object
Dim strResult As String

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
strResult = caoCtrl.ParameterRead( "SYS_TIME" )
```

<ImplVar>.ParameterReadPair

Usage

Reads two data items related to the settings or system values from the PV series main unit.

Syntax

<ImplVar>.ParameterReadPair (<Parameter> **)**

Argument: <Parameter> Specify with a character string.

Return value: Specified parameter value (Variant type).

Description

Two data items related to the settings or system values are read from the PV series main unit. A data set, such as upper and lower limits, is read. This command is effective during operation only. For the data to read and command parameters, refer to the user's manual of Panasonic PV series.

Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
vntResult = caoCtrl.ParameterReadPair( "BLV:PAIRA" )
```

<ImplVar>.ParameterWrite

Usage Changes the settings or system values of the PV series main unit.

Syntax **<ImplVar>.ParameterWrite** <Parameter>, <Data>

Argument: <Parameter> Specify with a character string.
<Data> Specify with a Variant type.

Description Settings or system values of the PV series main unit are changed. This command is effective during operation only. For the data to change and command parameters, refer to the user's manual of Panasonic PV series.

Example

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

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.ParameterWrite "SYS:REG0", 3.14
```

<ImplVar>.ParameterWritePair

Usage

Changes two data items related to the settings or system values of the PV series main unit.

Syntax

<ImplVar>.ParameterWritePair <Parameter>, <Data 1>, <Data 2>

Argument: <Parameter> Specify with a character string.
<Data 1> Specify with a Variant type.
<Data 2> Specify with a Variant type.

Description

Two data items related to the settings or system values of the PV series main unit are changed. This command is effective during operation only. For the data to change and command parameters, refer to the user's manual of Panasonic PV series.

Example

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

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.ParameterWritePair "BLV:PAIRA", 50, 100
```

<ImplVar>.StartAsync

Usage

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. Data to obtain is the character string type.

Syntax

<ImplVar>.StartAsync <Block No.>

Argument: <Block No.> Execution target block No. (integer 0 to 9)

Description

A block No. is required as an argument only when a batch trigger is used with the execution mode set to " User-Defined".
Argument is not required for the case of " Execute All" or " Automatic Switch" mode.

Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.StartAsync 1                        ' User-Defined

'Obtain the return value of StartAsync command
vntResult = caoCtrl.GetResult
```

The following shows how to execute inspection asynchronously.

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.StartAsync                          ' Execute All or Automatic Switch"

'Obtain the return value of StartAsync command
vntResult = caoCtrl.GetResult
```

<ImplVar>.ReStartAsync

Usage

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. Data to obtain is the character string type.

Syntax

<ImplVar>.RestartAsync <Block No.>

Argument: <Block No.> Execution target block No. (integer 0 to 9)

Description

A block No. is required as an argument only when a batch trigger is used with the execution mode set to " User-Defined".
Argument is not required for the case of " Execute All" or " Automatic Switch" mode.

Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.ReStartAsync 1                    ' User-Defined

'Obtain the return value of RestartAsync command
vntResult = caoCtrl.GetResult
```

The following shows how to execute inspection asynchronously.

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.ReStartAsync                      ' Execute All or Automatic Switch

'Obtain the return value of RestartAsync command
vntResult = caoCtrl.GetResult
```

<ImplVar>.XtypeAsync

Usage

Switch a product type asynchronously. To obtain and check the return value of the command, use GetResult command.

Syntax

<ImplVar>.XtypeAsync <Product No.>

Argument: <Product No.> (Integer 0 to 255)

Description

Switch a product type asynchronously. To obtain and check the return value of the command, use GetResult command.

Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.XtypeAsync 100

'Obtain the return value of XtypeAsync command
vntResult = caoCtrl.GetResult
```

<ImplVar>.MemoryWriteAsync

Usage

Write the setting data into PV series storage area asynchronously. To obtain and check the return value of the command, use GetResult command.

Syntax

<ImplVar>.MemoryWriteAsync **[**<Area No.> **]**

Argument: <Area No.> Specify the saving area No. of SD memory card.

PV200	None	
PV500	<Area No.>	(integers 0 to 99)

Description

Write the setting data into PV series storage area asynchronously. To obtain and check the return value of the command, use GetResult command.

Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.MemoryWriteAsync

'Obtain the return value of MemoryWriteAsync command
vntResult = caoCtrl.GetResult
```

<ImplVar>.CFWriteAsync

Usage

Write the setting data to an SD memory card asynchronously.
To obtain and check the return value of the command, use GetResult command.

Syntax

<ImplVar>.CFWriteAsync <Area No.>

Argument: <Area No.> Specify the saving area No. of SD memory card.
(integers 0 to 99)

Description

Write the setting data to an SD memory card asynchronously.
To obtain and check the return value of the command, use GetResult command.

Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.CFWriteAsync 10

'Obtain the return value of CFWriteAsync command
vntResult = caoCtrl.GetResult
```

<ImplVar>.MemoryReadAsync

Usage

Read the setting data from the memory of PV series asynchronously.
To obtain and check the return value of the command, use GetResult command.

Syntax

<ImplVar>.MemoryReadAsync **[**<Area No.>**]**

Argument: <Area No.> Specify the reading area No. of SD memory card.

PV200	None	
PV500	<Area No.>	(integers 0 to 99)

Description

Read the setting data from the memory of PV series asynchronously.
To obtain and check the return value of the command, use GetResult command.

Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.MemoryReadAsync

'Obtain the return value of MemoryReadAsync command
vntResult = caoCtrl.GetResult
```

<ImplVar>.CFReadAsync

Usage

Read the setting data from an SD memory card asynchronously.
To obtain and check the return value of the command, use GetResult command.

Syntax

<ImplVar>.CFReadAsync <Area No.>

Argument: <Area No.> Specify the reading area No. of SD memory card.
(Integer 0 to 99)

Description

Read the setting data from an SD memory card asynchronously.
To obtain and check the return value of the command, use GetResult command.

Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.CFReadAsync 10

'Obtain the return value of CFReadAsync command
vntResult = caoCtrl.GetResult
```

<ImplVar>.CancelDataAsync

Usage

Cancel saving/reading of the setting data asynchronously.
To obtain and check the return value of the command, use GetResult command.

Syntax

<ImplVar>.CancelDataAsync

Description

Cancel saving/reading of the setting data asynchronously.
To obtain and check the return value of the command, use GetResult command.

Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl. CancelDataAsync

'Obtain the return value of CancelDataAsync command
vntResult = caoCtrl.GetResult
```

<ImplVar>.SDSaveAsync

Usage 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.

Syntax **<ImplVar>.SDSaveAsync**

Description 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.

Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.SDSaveAsync

'Obtain the return value of SDSaveAsync command
vntResult = caoCtrl.GetResult
```

<ImplVar>.SDResetAsync

Usage Delete the image memory data stored in the PV series asynchronously.
To obtain and check the return value of the command, use GetResult command.

Syntax **<ImplVar>.SDResetAsync**

Description Delete the image memory data stored in the PV series asynchronously.
To obtain and check the return value of the command, use GetResult command.

Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.SDResetAsync

'Obtain the return value of SDResetAsync command
vntResult = caoCtrl.GetResult
```

<ImplVar>.PrintScreenAsync

Usage

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.

Syntax

<ImplVar>.PrintScreenAsync

Description

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.

Example

The following shows how to save the current display asynchronously.

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.PrintScreenAsync

'Obtain the return value of PrintScreenAsync command
vntResult = caoCtrl.GetResult
```

<ImplVar>.QuitAsync

Usage

Clear the statistics data and scan count asynchronously.
To obtain and check the return value of the command, use GetResult command.

Syntax

<ImplVar>.Quit Async

Description

Clear the statistics data and scan count asynchronously.
To obtain and check the return value of the command, use GetResult command.

Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.QuitAsync

'Obtain the return value of QuitAsync comand
vntResult = caoCtrl.GetResult
```

<ImplVar>.RunManualAsync

Usage

Switch the PV series operation state between RUN and STOP asynchronously. To obtain and check the return value of the command, use GetResult command. Data to obtain is the integer type.

Syntax

<ImplVar>.RunManualAsync <Mode>

Argument: <Mode> Switching between run and stop (integer).
0: Switches to run mode.
1: Switches to stop mode.

Description

Switch the PV series operation state between RUN and STOP asynchronously. To obtain and check the return value of the command, use GetResult command. Data to obtain is the character string type.

Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.RunManualAsync 1

'Obtain the return value of RunManualAsync command
vntResult = caoCtrl.GetResult
```

<ImplVar>.ErrorResetAsync

Usage Reset an Errror signal asynchronously. To obtain and check the return value of the command, use GetResult command.

Syntax **<ImplVar>.ErrorResetAsync**

Description Reset an Errror signal asynchronously. To obtain and check the return value of the command, use GetResult command.

Example

The following shows how to clear an error asynchronously.

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.ErrorResetAsync

'Obtain the return value of ErrorResetAsync command
vntResult = caoCtrl.GetResult
```

<ImplVar>. CancelAsync

Usage

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.

Syntax

<ImplVar>.CancelAsync

Description

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.

Example

The following shows how to cancel an ongoing motion asynchronously.

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.CancelAsync

'Obtain the return value of CancelAsync command
vntResult = caoCtrl.GetResult
```

<ImplVar>.KeyEmulatorAsync

Usage

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.

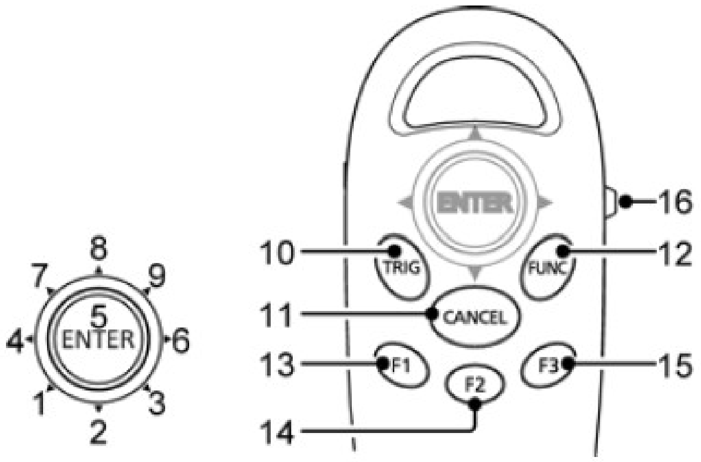
Syntax

<ImplVar>.KeyEmulatorAsync <Shift>, <Key>

Argument: <Shift> Shift key ON/OFF (integer 0, 1).
0: OFF
1: ON
<Key> Value allocated to each key (integer 1 to 16).
See the following figure for details.

Description

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.



Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.KeyEmulatorAsync 0, 16

'Obtain the return value of KeyEmulatorAsync command
vntResult = caoCtrl.GetResult
```

<ImplVar>.BstopAsync

Usage

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.

Syntax

<ImplVar>.BstopAsync <Availability>

Argument: <Availability> Availability of keypad operations (integer 0, 1).
0: Available
1: Unavailable

Description

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.

Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.BstopAsync 1

'Obtain the return value of BstopAsync command
vntResult = caoCtrl.GetResult
```

<ImplVar>.BconfirmAsync

Usage Get the current state of keypad operation permission, asynchronously.
To obtain and check the return value of the command, use GetResult command.

Syntax **<ImplVar>.BconfirmAsync**

Description Get the current state of keypad operation permission, asynchronously.
To obtain and check the return value of the command, use GetResult command.

Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.BconfirmAsync

'Obtain the return value of BconfirmAsync command
vntResult = caoCtrl.GetResult
```

<ImplVar>.LayoutChangeAsync

Usage

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.

Syntax

<ImplVar>.LayoutChangeAsync <Layout No.>

Argument: <Layout No.> Specify with an integer (0 to 15).

Description

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.

Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.LayOutChangeAsync 1

'Obtain the return value of LayoutChangeAsync command
vntResult = caoCtrl.GetResult
```

<ImplVar>.AgainTemplateAsync

Usage

Re-register the template of the smart matching checker, asynchronously. To obtain and check the return value of the command, use GetResult command.

Syntax

<ImplVar>.AgainTemplateAsync <Checker No.> , <Template No.>

Argument: <Checker No.> Specify with an integer (0 to 999).
<Template No.> Specify with an integer (0 to 63).

Description

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.

Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.AgainTemplateAsync 1, 10

'Obtain the return value of AgainTemplateAsync command
vntResult = caoCtrl.GetResult
```

<ImplVar>.ParameterReadAsync

Usage

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. Data to obtain is the character string type.

Syntax

<ImplVar>.ParameterReadAsync <Parameter>

Argument: <Parameter> Specify with a character string.

Description

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. Data to obtain is the character string type.

Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.ParameterReadAsync "SYS_TIME"

'Obtain the return value of ParameterReadAsync command
vntResult = caoCtrl.GetResult
```

<ImplVar>.ParameterReadPairAsync

Usage

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. Data to obtain is the variant type.

Syntax

<ImplVar>.ParameterReadPairAsync <Parameter>

Argument: <Parameter> Specify with a character string.

Description

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. Data to obtain is the character string type.
Data to obtain is the variant type.

Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.ParameterReadPairAsync "BLV:PAIRA"

'Obtain the return value of ParameterReadPairAsync command
vntResult = caoCtrl.GetResult
```

<ImplVar>.ParameterWriteAsync

Usage

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.

Syntax

<ImplVar>.ParameterWriteAsync <Parameter>, <Data>

Argument: <Parameter> Specify with a character string.
<Data> Specify with a Variant type.

Description

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.

Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.ParameterWriteAsync "SYS:REG0", 3.14

'Obtain the return value of ParameterWriteAsync command
vntResult = caoCtrl.GetResult
```

<ImplVar>.ParameterWritePairAsync

Usage

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.

Syntax

<ImplVar>.ParameterWritePairAsync <Parameter>, <Data 1>, <Data 2>

Argument: <Parameter> Specify with a character string.
<Data 1> Specify with a Variant type.
<Data 2> Specify with a Variant type.

Description

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.

Example

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.

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.ParameterWritePairAsync "CAC010:LPAIR", 50, 100

'Obtain the return value of ParameterWritePairAsync command
vntResult = caoCtrl.GetResult
```

<ImplVar>.Raw

Usage Transfers a command message.

Syntax **<ImplVar>.Raw(** <Send command message> **)**

Argument: <Send command message> Specify with a character string.

Return value: Received command message (character string).

Description PV series commands are transferred directly. Automatic calculation is performed for BCC (block check code) internally.
For commands, refer to the user's manual of Panasonic PV series.

Example

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

```
Dim caoCtrl As Object
Dim strResult As String

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
strResult = caoCtrl.Raw( "%S" )
```

<ImplVar>. SetTimeout

Usage

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

Syntax

<ImplVar>. SetTimeout < Timeout period >

Argument: < Timeout period > Specify with an integer.

Description

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

Example

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

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.SetTimeout 1000
```

<ImplVar>. GetTimeout

Usage

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

Syntax

<ImplVar>. GetTimeout( )

Return value: Timeout period (integer).

Description

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

Example

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

```
Dim caoCtrl As Object
Dim iResult As Integer

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
iResult = caoCtrl.GetTimeout
```

<ImplVar>. RawAsync

Usage

Send a command message asynchronously. BCC is calculated internally automatically. To obtain and check the return value of the command, use GetResult command. Data to obtain is the character string type.

Syntax

<ImplVar>. RawAsync < Send command message >

Argument: <Send command message> Specify with a character string.

Description

Send a command message asynchronously. BCC is calculated internally automatically. To obtain and check the return value of the command, use GetResult command. Data to obtain is the character string type.

Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.RawAsync "%S"

'Obtain the return value of RawAsync command
vntResult = caoCtrl.GetResult
```

<ImplVar>. GetResult

Usage

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

<ImplVar>. GetResult ( )

Return value: Return value of asynchronous command (Variant type).
The return value depends on the executed command.

Description

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.

Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.StartAsync

vntResult = caoCtrl.GetResult
```

<ImplVar>. SetPoint

Usage

Notify PV of robot coordinates.

Syntax

<ImplVar>. SetPoint < Robot coordinate (X) >, < Robot coordinate (Y) >, < Robot coordinate (Z) >, < Robot coordinate (Rx) >, < Robot coordinate (Ry) >, < Robot coordinate (Rz) >, < Robot coordinate (Fig)>

Argument: < Robot coordinate (X) >	Specify with a double precision type.
< Robot coordinate (Y) >	Specify with a double precision type.
< Robot coordinate (Z) >	Specify with a double precision type.
< Robot coordinate (Rx) >	Specify with a double precision type.
< Robot coordinate (Ry) >	Specify with a double precision type.
< Robot coordinate (Rz) >	Specify with a double precision type.
< Robot coordinate (Fig) >	Specify with an integer type.

Description

Notify PV of robot coordinates.

Example

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

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "PV260=1, conn=eth:192.168.0.201" )
caoCtrl.SetPoint POSX(CURPOS), POSY(CURPOS), POSZ(CURPOS), POSRX(CURPOS), _
POSRY(CURPOS), POSRZ(CURPOS), FIG(CURPOS)
```

<ImplVar>. Calibrate

Usage

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

Syntax

<ImplVar>. Calibrate (< Calibration No >, **[**< Block No >**])**

Argument: < Calibration No > Specify with an integer type. (0 to 5)
< Block No > Specify with an integer type. (0 to 9)

Return value: Robot coordinate array (X, Y, Rz, Fig) (Variant type).

Description

A block No. is required as an argument only when a batch trigger is used with the execution mode set to " User-Defined". A block No. is not required for the case of " Execute All" mode.

Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "PV260=1, conn=eth:192.168.0.201" )
vntResult = caoCtrl.Calibrate( 0 )          ' Execute All
```

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "PV260=1, conn=eth:192.168.0.201" )

vntResult = caoCtrl.Calibrate( 0, 1 )       ' User-Defined
```

<ImplVar>. ReCalibrate

Usage

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

Syntax

<ImplVar>. ReCalibrate (< Calibration No >, **[**< Block No >**])**

Argument: < Calibration No > Specify with an integer type. (0 to 5)
< Block No > Specify with an integer type. (0 to 9)

Return value: Robot coordinate array (X, Y, Rz, Fig) (Variant type).

Description

A block No. is required as an argument only when a batch trigger is used with the execution mode set to " User-Defined". A block No. is not required for the case of " Execute All" mode.

Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "PV260=1, conn=eth:192.168.0.201" )
vntResult = caoCtrl.ReCalibrate( 0 )            ' Execute All
```

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "PV260=1, conn=eth:192.168.0.201" )
vntResult = caoCtrl.ReCalibrate( 0, 1 )         ' User-Defined
```

<ImplVar>. CalibrationStart

Usage

Start Auto calibration. To execute Auto calibration, it is necessary to prepare a program linked to a robot according to the CalibrationStart command.

Syntax

<ImplVar>. CalibrationStart < Calibration No. **>,** [< Camera No. >]

Argument: < Calibration No. > Specify with an integer type. (0 to 5)
< Camera No. > Specify with an integer type. (0 to 1)

Description

Start Auto calibration.
If the format of PV260 is Ver1.1.0 or earlier, when omitting the argument of camera No., it will be set to "Without camera No. specified".

Example

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

```
Dim lCalibrationNum as long
Dim lCameraNum as long

lCalibrationNum = 0
lCameraNum = 1

  ‘ Notify “Without camera No. specified” (%CAS0) to PV.
caoCtrl.Execute "CalibrationStart", lCalibrationNum

  ‘ Notify “With camera No. specified” (%CAS0,1) to PV.
caoCtrl.Execute "CalibrationStart", Array(lCalibrationNum, lCameraNum)
```

<ImplVar>. CalibrationEnd

Usage Obtain the notification of Auto calibration completion.

Syntax **<ImplVar>. CalibrationEnd**

Description Obtain the notification of Auto calibration completion.

Example

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

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "PV260=1, conn=eth:192.168.0.201" )
caoCtrl.CalibrationEnd
```

<ImplVar>. WorkSet

Usage

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

<ImplVar>. WorkSet

Description

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.

Example

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

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "PV260=1, conn=eth:192.168.0.201" )
caoCtrl.WorkSet
```

<ImplVar>. WorkReset

Usage

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.

Syntax

<ImplVar>. WorkReset < Work detection No >

Argument: < Work detection No > Specify with an integer type. (0 to 15)

Description

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..

Example

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

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "PV260=1, conn=eth:192.168.0.201" )
caoCtrl.WorkReset 0
```

<ImplVar>. WorkResetEnd

Usage Obtain the notification of the work detection base position re-registration completion.

Syntax **<ImplVar>. WorkResetEnd**

Description Obtain the notification of the work detection base position re-registration completion.

Example

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

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "PV260=1, conn=eth:192.168.0.201" )
caoCtrl.WorkResetEnd
```

<ImplVar>. MoveEnd

Usage
Notify PV of robot movement completion.

Syntax
<ImplVar>. MoveEnd

Description
Notify PV of robot movement completion.

Example

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

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "PV260=1, conn=eth:192.168.0.201" )
caoCtrl.MoveEnd
```

<ImplVar>. GetTeachPoint

Usage Obtain all teaching coordinate configured in PV.

Syntax **<ImplVar>. GetTeachPoint ( )**

Return value: Robot coordinate array (X, Y, Rz, Fig) (Variant type).

Description Obtain all teaching coordinate configured in PV.

Example

The following shows how to request the teaching coordinates.

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "PV260=1, conn=eth:192.168.0.201" )
vntResult = caoCtrl.GetTeachPoint
```

<ImplVar>. GetMovePoint

Usage

Obtain robot coordinates sent from PV during Auto calibration (CalibrationStart) or during re-registration of the work detection base position (WorkReset).

Syntax

<ImplVar>. GetMovePoint( )

Return value: Robot coordinate array (X, Y, Rz, Fig) (Variant type).

Description

Obtain robot coordinates sent from PV during Auto calibration (CalibrationStart) or during re-registration of the work detection base position (WorkReset).

Example

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

```
Dim caoCtrl As Object
Dim vntResultPos As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "PV260=1, conn=eth:192.168.0.201" )
vntResultPos = caoCtrl.GetMovePoint
```

<ImplVar>. SetPointAsync

Usage

Notify PV of the robot coordinates asynchronously. To obtain and check the return value of the command, use GetResult command.

Syntax

<ImplVar>. SetPointAsync < Robot coordinate (X) >,
< Robot coordinate (Y) >,
< Robot coordinate (Z) >,
< Robot coordinate (Rx) >,
< Robot coordinate (Ry) >,
< Robot coordinate (Rz) >,
< Robot coordinate (Fig)>

Argument: < Robot coordinate (X) >	Specify with a double precision type.
< Robot coordinate (Y) >	Specify with a double precision type.
< Robot coordinate (Z) >	Specify with a double precision type.
< Robot coordinate (Rx) >	Specify with a double precision type.
< Robot coordinate (Ry) >	Specify with a double precision type.
< Robot coordinate (Rz) >	Specify with a double precision type.
< Robot coordinate (Fig) >	Specify with an integer type.

Description

Notify PV of the robot coordinates asynchronously. To obtain and check the return value of the command, use GetResult command.

Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "PV260=1, conn=eth:192.168.0.201" )
caoCtrl.SetPointAsync PosX( CurPos ), PosY( CurPos ), PosZ( CurPos ), PosRx( CurPos ), _
PosRy( CurPos ), PosRz( CurPos ), Fig( CurPos )

'Obtain the return value of SetPointAsync command
vntResult = caoCtrl.GetResult
```

<ImplVar>. CalibrateAsync

Usage

Execute the measurement asynchronously. The syntax differs depending on the execution mode; "Execute All", or "User Defined". To obtain and check the return value of the command, use GetResult command. Data to obtain is the variant type.

Syntax

<ImplVar>. CalibrateAsync < Calibration No >, < Block No >

Argument: < Calibration No > Specify with an integer type. (0 to 5)
< Block No > Specify with an integer type. (0 to 9)

Description

A block No. is required as an argument only when a batch trigger is used with the execution mode set to " User-Defined". A block No. is not required for the case of " Execute All" mode.

CalibrationStart

Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "PV260=1, conn=eth:192.168.0.201" )
caoCtrl.CalibrateAsync 0                    ' Execute All

'Obtain the return value of CalibrateAsync command
vntResult = caoCtrl.GetResult
```

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "PV260=1, conn=eth:192.168.0.201" )
caoCtrl.CalibrateAsync 0, 1                 ' User-Defined

'Obtain the return value of CalibrateAsync command
vntResult = caoCtrl.GetResult
```

<ImplVar>. ReCalibrateAsync

Usage

Execute the measurement asynchronously without taking pictures (re-measurement). The syntax differs depending on the execution mode; "Execute All", or "User Defined". To obtain and check the return value of the command, use GetResult command. Data to obtain is the variant type.

Syntax

<ImplVar>. ReCalibrateAsync < Calibration No >**,** < Block No >

Argument: < Calibration No > Specify with an integer type. (0 to 5)
< Block No > Specify with an integer type. (0 to 9)

Description

A block No. is required as an argument only when a batch trigger is used with the execution mode set to " User-Defined ". A block No. is not required for the case of " Execute All " mode.

Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "PV260=1, conn=eth:192.168.0.201" )
caoCtrl.ReCalibrateAsync 0                ' Execute All

'Obtain the return value of ReCalibrateAsync command
vntResult = caoCtrl.GetResult
```

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "PV260=1, conn=eth:192.168.0.201" )
caoCtrl.ReCalibrateAsync 0, 1             ' User-Defined

'Obtain the return value of ReCalibrateAsync command
vntResult = caoCtrl.GetResult
```

<ImplVar>. CalibrationStartAsync

Usage

Start asynchronous auto calibration. To execute Auto calibration, it is necessary to prepare a program linked to a robot according to the CalibrationStart command.

Syntax

<ImplVar>. CalibrationStartAsync < Calibration No. >, [< Camera No. >]

Argument: < Calibration No. > Specify with an integer type. (0 to 5)
< Camera No. > Specify with an integer type. (0 to 1)

Description

Start asynchronous auto calibration.
If the format of PV260 is Ver1.1.0 or earlier, when omitting the argument of camera No., it will be set to “Without camera No. specified”.
To obtain and check the return value of the command, use GetResult command.

Example

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

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

lCalibrationNum = 0
lCameraNum = 1

  ‘ Notify “Without camera No. specified” (%CAS0) to PV.
caoCtrl.Execute "CalibrationStartAsync", lCalibrationNum

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


  ‘ Notify “With camera No. specified” (%CAS0,1) to PV.
caoCtrl.Execute "CalibrationStartAsync", Array(lCalibrationNum, lCameraNum)

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

  ‘ vntResult : No return value (Empty)
```

<ImplVar>. CalibrationEndAsync

Usage Obtain the notification of Auto calibration completion asynchronously. To obtain and check the return value of the command, use GetResult command.

Syntax **<ImplVar>. CalibrationEndAsync**

Description Obtain the notification of Auto calibration completion asynchronously. To obtain and check the return value of the command, use GetResult command.

Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "PV260=1, conn=eth:192.168.0.201" )
caoCtrl.CalibrationEndAsync

'Obtain the return value of CalibrationEndAsync command
vntResult = caoCtrl.GetResult
```

<ImplVar>. WorkSetAsync

Usage

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 obtain and check the return value of the command, use GetResult command.

Syntax

<ImplVar>. WorkSetAsync

Description

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 obtain and check the return value of the command, use GetResult command.

Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "PV260=1, conn=eth:192.168.0.201" )
caoCtrl.WorkSetAsync

'Obtain the return value of WorkSetAsync command
vntResult = caoCtrl.GetResult
```

<ImplVar>. WorkResetAsync

Usage

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 obtain and check the return value of the command, use GetResult command.

Syntax

<ImplVar>. WorkResetAsync < Work detection No >

Argument: < Work detection No > Specify with an integer type. (0 to 15)

Description

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 obtain and check the return value of the command, use GetResult command.

Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "PV260=1, conn=eth:192.168.0.201" )
caoCtrl.WorkResetAsync 0

'Obtain the return value of WorkResetAsync command
vntResult = caoCtrl.GetResult
```

<ImplVar>. WorkResetEndAsync

Usage

Obtain the notification of the work detection base position re-registration completion asynchronously. To obtain and check the return value of the command, use GetResult command.

Syntax

<ImplVar>. WorkResetEndAsync

Description

Obtain the notification of the work detection base position re-registration completion asynchronously. To obtain and check the return value of the command, use GetResult command.

Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "PV260=1, conn=eth:192.168.0.201" )
caoCtrl.WorkResetEndAsync

'Obtain the return value of WorkResetEndAsync command
vntResult = caoCtrl.GetResult
```

<ImplVar>. MoveEndAsync

Usage

Notify PV of robot movement completion asynchronously. To obtain and check the return value of the command, use GetResult command.

Syntax

<ImplVar>. MoveEndAsync

Description

Notify PV of robot movement completion asynchronously. To obtain and check the return value of the command, use GetResult command.

Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "PV260=1, conn=eth:192.168.0.201" )
caoCtrl.MoveEndAsync

'Obtain the return value of MoveEndAsync command
vntResult = caoCtrl.GetResult
```

<ImplVar>. GetTeachPointAsync

Usage Obtain all teaching coordinate configured in PV asynchronously. To obtain and check the return value of the command, use GetResult command. Data to obtain is the variant type.

Syntax **<ImplVar>. GetTeachPointAsync**

Description Obtain all teaching coordinate configured in PV asynchronously. To obtain and check the return value of the command, use GetResult command. Data to obtain is the variant type.

Example

The following shows how to request the teaching coordinates asynchronously.

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "PV260=1, conn=eth:192.168.0.201" )
caoCtrl.GetTeachPointAsync

'Obtain the return value of GetTeachPointAsync command
vntResult = caoCtrl.GetResult
```

<ImplVar>. GetMovePointAsync

Usage

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 obtain and check the return value of the command, use GetResult command. Data to obtain is the variant type.

Syntax

<ImplVar>. GetMovePointAsync

Description

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 obtain and check the return value of the command, use GetResult command. Data to obtain is the variant type.

Example

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

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "PV260=1, conn=eth:192.168.0.201" )
caoCtrl.GetMovePointAsync

'Obtain the return value of GetMovePointAsync command
vntResult = caoCtrl.GetResult
```

6. Error code of PV provider

The specific error code of the PV provider is created as shown below, based on the return value.
0x80100010 + Return value

For the error code of each command, refer to the PV series reference manual of Panasonic Industrial Devices SUNX.

Example: When executing Start

0x801000C8: Not executable due to stop state.

The following error codes are defined as original error codes.

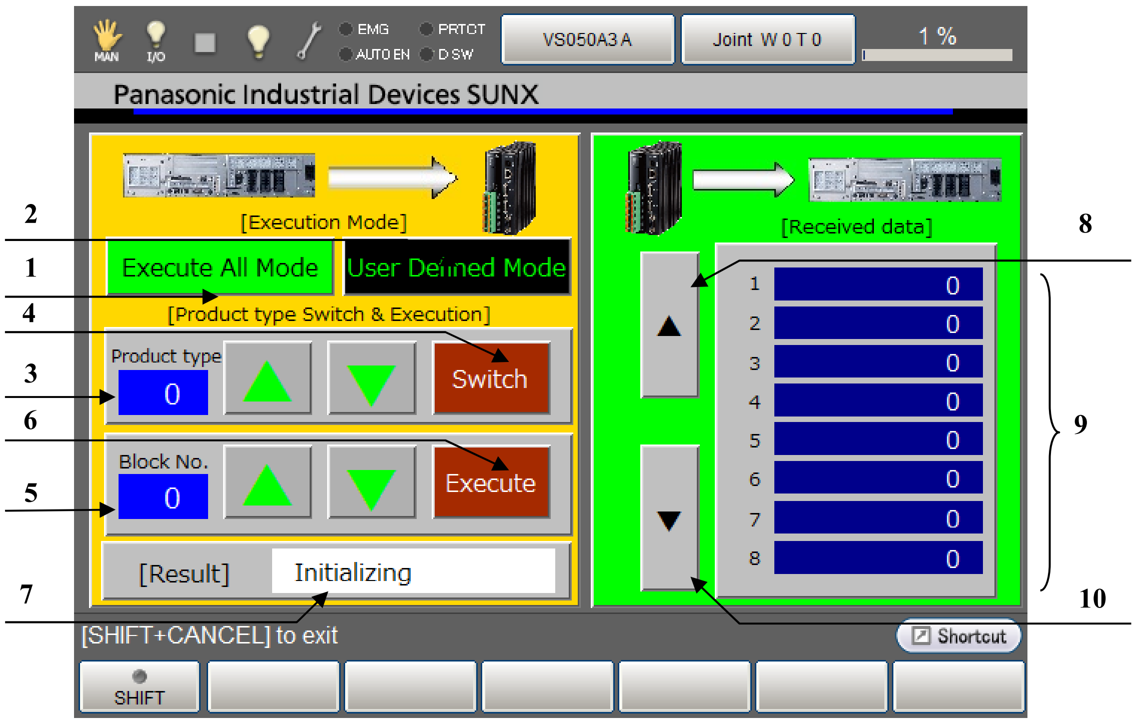
Error	Error number	Description
E_COMMAND_EXECUTING	0x80F00000	Another command is executed during a command execution.
E_COMMAND_CONNECTED (Ver.1.12.* or later)	0x80F00001	A command was executed to an unconnected communication port

About the ORiN2 commonness error, please refer to the chapter of the error code of "ORiN2 Programming guide".

7. Operation Panel Screen

This provider provides the following operation panel screen. This operation panel uses the provider to check operations, etc. after connecting to the device. See the following as an application example of the operation panel. Displaying the operation panel establishes connection to PV (implements the provider). The communication settings need to be configured beforehand. Closing the operation panel terminates the connection (releases the provider).

[Main screen]



Description

Each button functions as follows.

1. Switches to the "All execution" or "Branch execution" mode.
2. Switches to the "Specified execution" mode.
3. A field for setting a product for change. Range: 0 to 255
4. Changes to the product type set in (3).(Xtype)
5. A field for setting a block No. for the "Specified execution" mode. Range: 0 to 9
6. Executes testing according to the settings made in the steps 3 and 5. Received data appears in the data display section (9). (Start)
7. Displays the processing result.
8. Moves up the page displayed for received data.
9. Displays the received data.
10. Moves down the page displayed for received data.

Note 1: When a provider implementation (initialization) is done properly, "Connected" is displayed in the field (7).

Note 2: Do not use the operation panel screen when the PV provider is used by PacScript program.

8. Sample Program

```
Sub Main

    On Error Goto ErrProc                           'Declare error process routine

    Dim caoPV as Object                             'Declare provider variable
    Dim strResult as String                         'Declare character-string variable
    Dim pTargetPos as Position                      'Declare P-type variable

    takearm keep = 0

    pTargetPos = P11

    caoPV = cao.AddController("PV", "CaoProv.Panasonic.PV", "", "Conn=eth:192.168.0.110, Timeout = 1000")
                                                    'Provider implementation
    caoPV.Xtype 2                                   'Change to product 2
    strResult = caoPV.Start                         'Trigger -> wait for process
    letx pTargetPos = posx(P11) + val(strResult)    'Expand X component of received data to position data

    approach p, pTargetPos, @p 20, s = 100          'Go to position after correction
    move l, @e pTargetPos, s = 10
    call Hand.Close
    depart l, @p 50, s = 100

EndProc:                                            'Normal end routine
    "State necessary end process"
    exit sub

ErrProc:                                            'Abnormal end routine
    "State necessary error process"

End Sub
```

Revision History

DENSO Robot
Provider
User’s Manual

Panasonic Industrial Devices SUNX Vision Sensor PV series

Version	Supported RC8	Content
Ver.1.0.0	Ver.1.1.2	First version
Ver.1.0.1	Ver.1.3.6 and later	Addition of variable "@ResultDisable"
Ver.1.0.2	Ver.1.3.7 and later	Correction of RunManual command
Ver.1.0.4	Ver.1.12. *	Original error (E_COMMAND_EXECUTING) was added. Asynchronous commands were added. Calibration commands were added.(PV260-compatible) Timeout setting/obtainment commands were added.
Ver.1.0.5	Ver.2.3. *	MyIP option was added to Option string of AddController
Ver.1.0.6		Modified version.
Ver.1.0.7	Ver.2.15.*	Correction of CalibrationStart command Correction of CalibrationStartAsync command

DENSO WAVE INCORPORATED

DENSO Robotics

THIRD PARTY PRODUCTS

DENSO WAVE INCORPORATED