DENSO

DENSO Robotics

THIRD PARTY PRODUCTS

PROVIDER MANUAL

Maker

KEYENCE

Products／Series

Flexible Image Processing System

MODEL:XG-X Series

Vision

Introduction

This document is a user's manual for the provider to use "KEYENCE Flexible Image Processing System XG-X Series" connected to the DENSO robot controller RC8 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 document targets the following models in XG-X series.

KEYENCE XG-X2000 Series

In this document, the above models are called XG-X 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

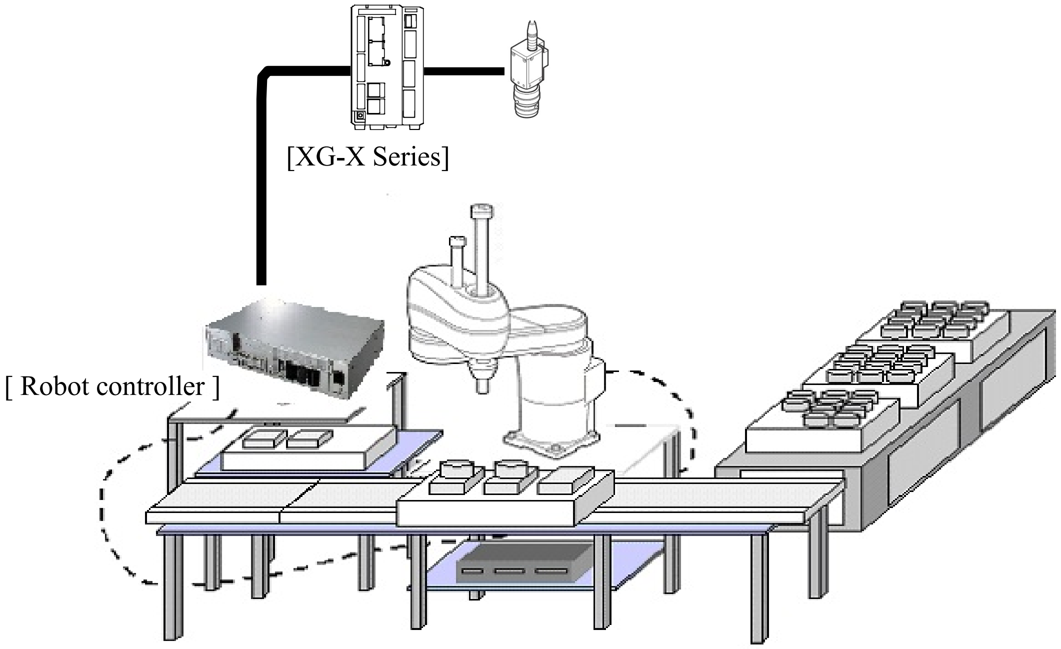
The user of this product shall be responsible for embedding and using the product (software) on a system and any result from using it.

Contents

1. Outline of This Product (Provider)

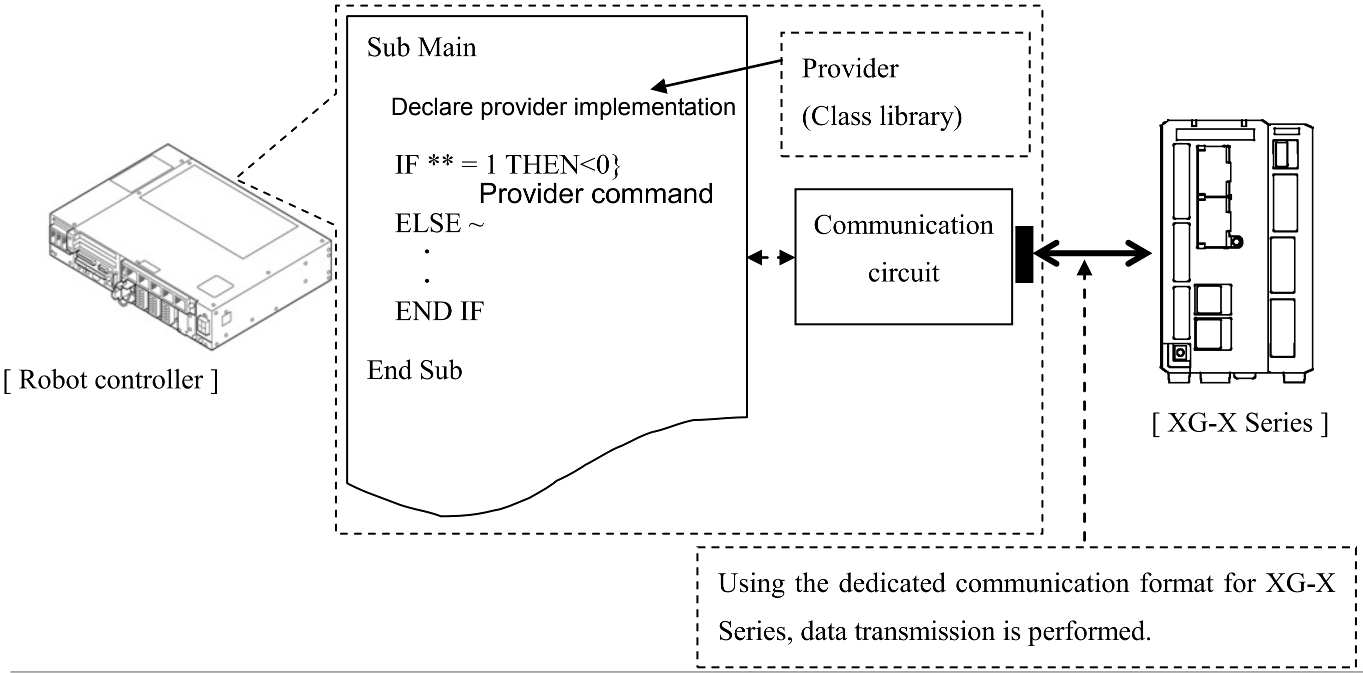
1.1. Target device of provider

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



1.2. Features of provider

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



```
Sub Main
    Declare provider implementation
    IF ** = 1 THEN<0}
                Provider command
    ELSE ~
        .
        .
    END IF
End Sub
```

Using the dedicated communication format for XG-X Series, data transmission is performed.

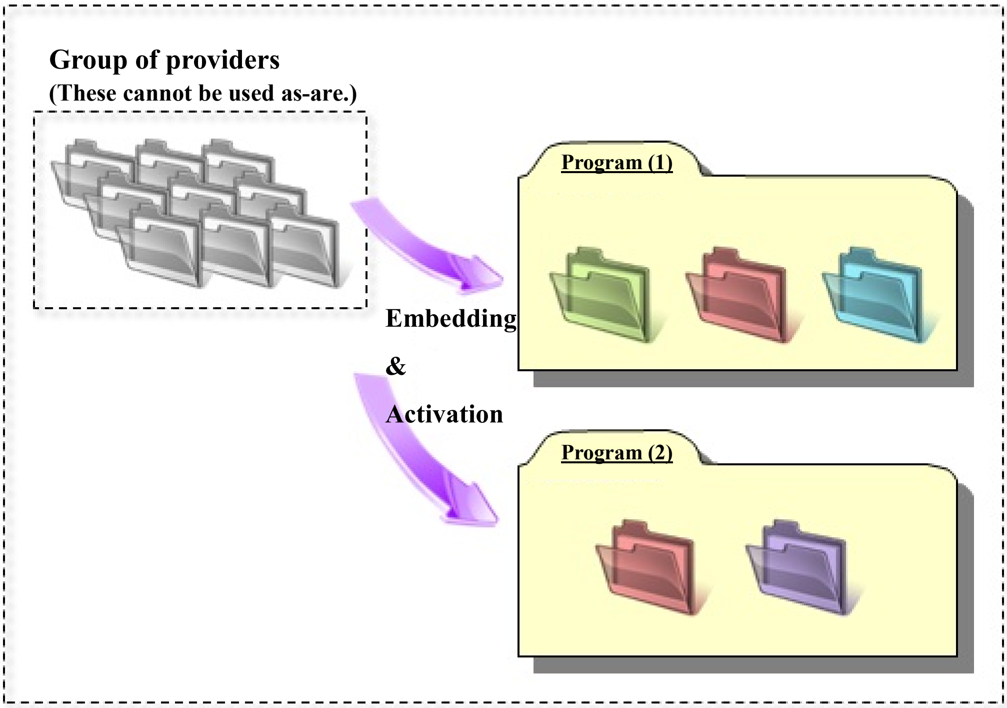
1.3. Mechanism of provider

This provider offers various programs required to control the target device as a single provider. To use this provider, you just need to activate the license. 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.

Assuming if a provider is a folder,

the folder contains various commands.

Controller internal image



▨ **Provider prepared in the system. This cannot be used as-is.**

🟩🟥🟦🟪 **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).

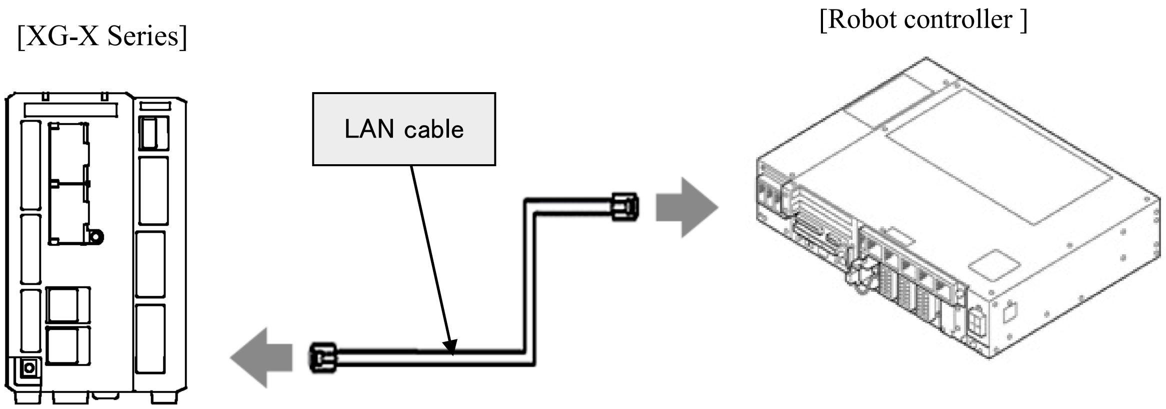
* The provider is provided as a dynamic link library (abbreviated as DLL) which can be used from PacScript.

2. How to Connect

You can use either Ethernet or RS-232C for connection between a robot controller and XG-X series. To establish a connection, use a cable that is compatible with the communication specification you use. For detailed information about each communication cable, refer to the XG-X Series User's Manual of KEYENCE.

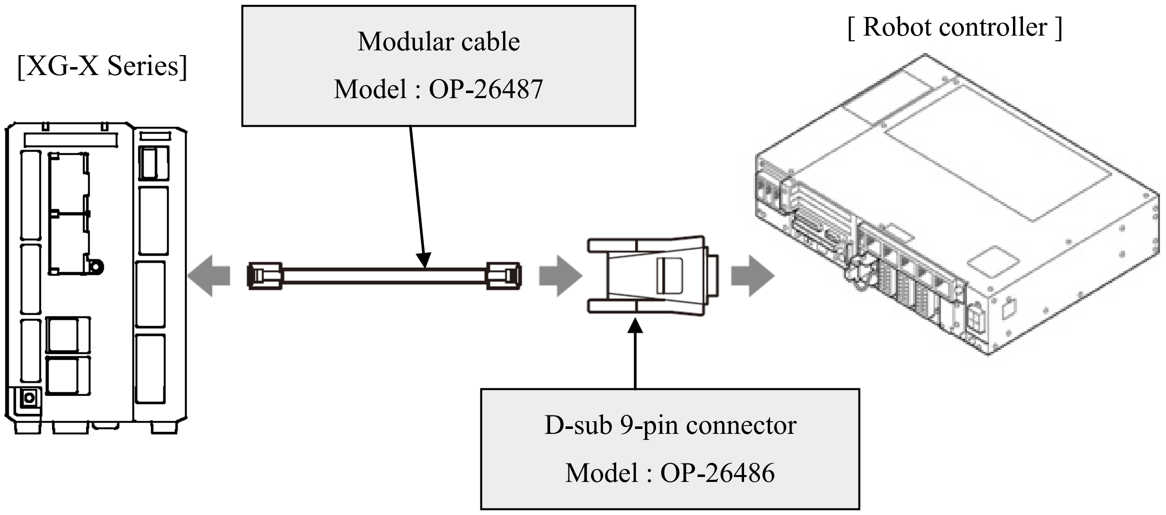
2.1. Ethernet Connection Example

To connect to the robot controller via Ethernet, use a crossover LAN cable. Also, when a switching hub/router is used, use the cable suitable for the switching hub/router specifications.



2.2. RS-232C Connection Example

When you establish a connection with RS-232C, use a Modular cable and D-sub 9-pin connector. Both of them are sold by KEYENCE as optional parts. There are two types of connectors though, use a D-sub 9-pin connector since the RS-232C connector mounted in the robot controller is D-sub 9-pin.



3. Communication settings

3.1. Setup for Ethernet connection

3.1.1. Communication setting for XG-X series

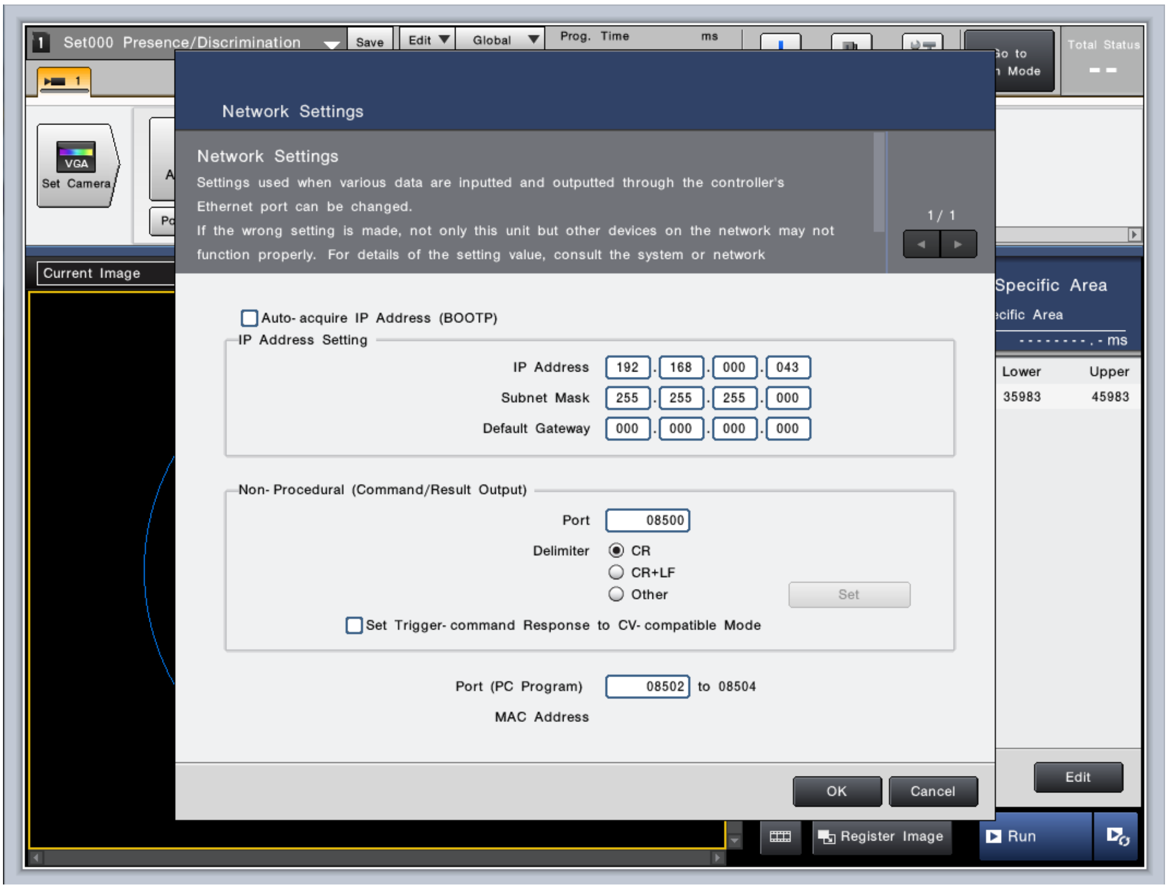
Communication settings for XG-X series is carried out by manipulating a setting window displayed in the monitor (sold separately) plugged in the XG-X series main unit by means of a mouse that comes with XG-X series. For details, refer to the XG-X series User's Manual of KEYENCE.
Note that the following items must be the same settings always.

Item	Setting
Auto-acquire IP address (BOOTP)	Not select
Delimiter	CR
Set Trigger-command Response to CV-compatible Mode	Not select

This setting example shows when XG-X2000 series is used.

From the XG-X2000 setting window, click [Global] - [Communications & I/O] - [Network] to display the Network Settings window as shown below.

- Do not select the "Auto-acquire IP address (BOOTP)" checkbox.
- When you set "IP address" and "Subnet mask", make sure that these of the robot controller and XG-X series are in the same subnet mask. In this example, IP address and Subnet mask are 192.168.0.43 and 255.255.255.0., respectively.
- Set a "Default Gateway", if necessary. In this example, 0.0.0.0 is set.
- Set desired port number to "Port". The port number specified here will be the port number that is specified at the robot controller's Cao.AddController command execution as an option. In this example, the port number is set to 08500.
- "Delimiter" must be set to "CR" always.
- Do not select the "Set Trigger-command Response to CV-compatible Mode" checkbox.
- "Port (PC Program)"has no relation to this provider



Network Settings

Network Settings

Settings used when various data are inputted and outputted through the controller's Ethernet port can be changed.
If the wrong setting is made, not only this unit but other devices on the network may not function properly. For details of the setting value, consult the system or network

□ Auto-acquire IP Address (BOOTP)

IP Address Setting

IP Address 192 . 168 . 000 . 043

Subnet Mask 255 . 255 . 255 . 000

Default Gateway 000 . 000 . 000 . 000

Non-Procedural (Command/Result Output)

Port 08500

Delimiter ● CR ○ CR+LF ○ Other

Set

□ Set Trigger-command Response to CV-compatible Mode

Port (PC Program) 08502 to 08504

MAC Address

OK Cancel

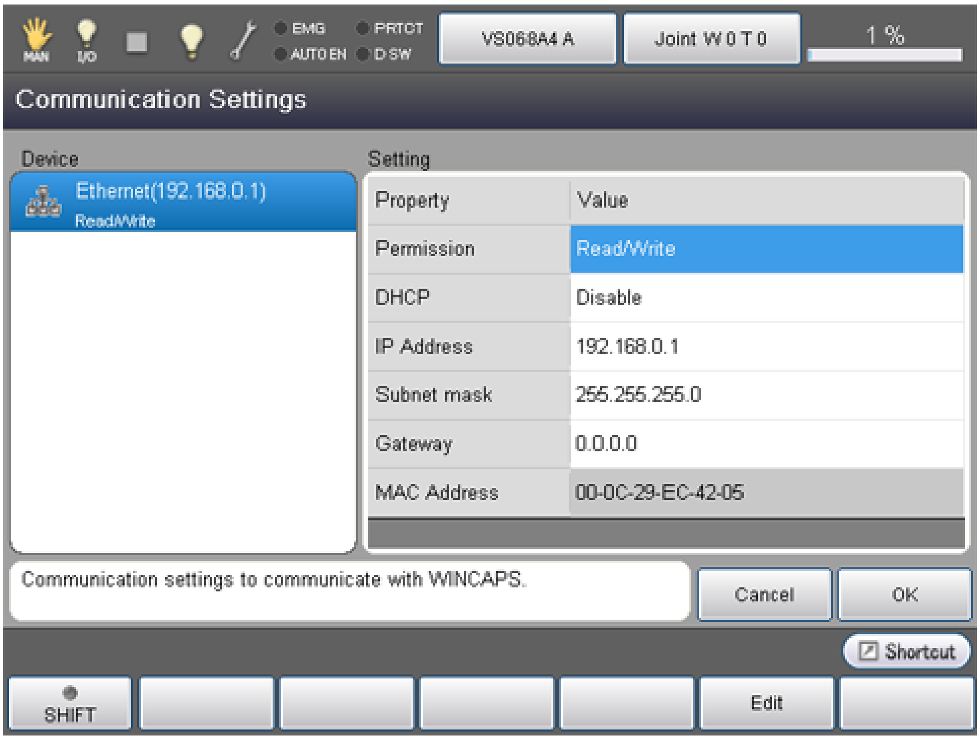
3.1.2. Communication setting for Robot controller

To set Ethernet communication setting for a robot controller, you can use a teach pendant or a mini-pendant. For detailed information about setting, refer to the followings on the DENSO ROBOT USER MANUALS.

Device	Referenced
Teach pendant	"Displaying and Changing Communication Settings Screen" of the TEACH PENDANT OPERATION GUIDE
Mini-pendant	"Setting DHCP" and "Setting IP Address" of the MINI PENDANT OPERATION GUIDE

This example shows the way of communication setting with a teach pendant.
From the top screen of a teach pendant, press [F6 Setting] - [F5 Communication and Token] - [F2 Network and Permission] to display the [Communication Settings] window.

- [Permission] has no relation to XG-X series communication.
- Once DHCP is enabled, IP address will automatically set. (Note that DHCP server may connect to the same network.) This example select "Disable".
- If you set DHCP to "Disable", make sure that IP addresses and subnet masks of the robot controller and XG-X series are the same. In this example, IP address and the subnet mask are 192.168.0.1 and 255.255.255.0, respectively.
- Set a gateway, if necessary. In this example, 0.0.0.0 is set.



Communication Settings

Device

Ethernet(192.168.0.1)
Read/Write

Setting

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	00-0C-29-EC-42-05

Communication settings to communicate with WINCAPS.

3.2. Setup for RS-232C connection

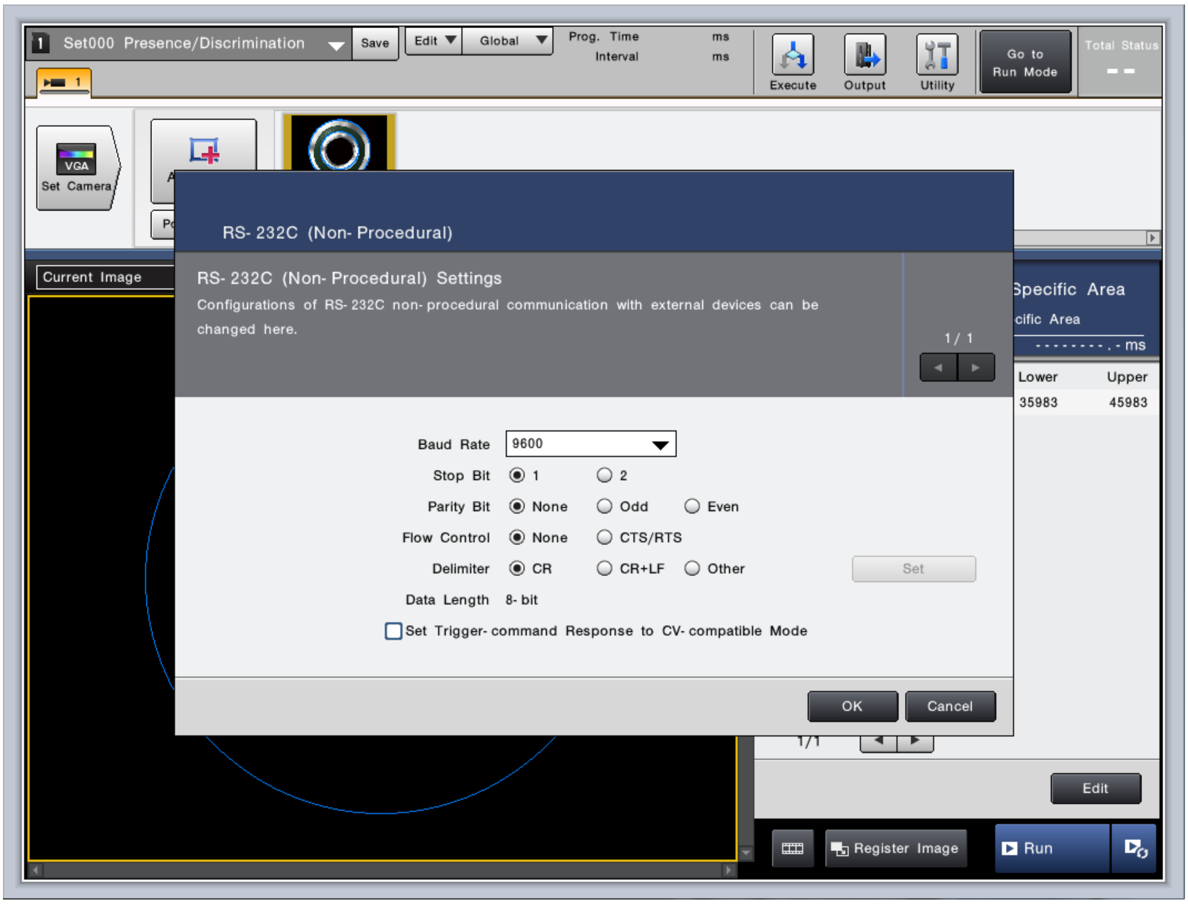
3.2.1. Communication setting for XG-X Series

RS-232C communication setting for XG-X series is carried out by manipulating a setting window displayed in the monitor (sold separately) plugged in the XG-X series main unit by means of a mouse that comes with XG-X series. For details, refer to the XG-X series User's Manual of KEYENCE.
Note that the following items must be the same settings always.

Items	Setting
Flow Control	None
Delimiter	CR
Set Trigger-command Response to CV-compatible Mode	Not select

This setting example shows when XG-X2000 series is used.

From the XG-X2000 setting window, click [Global] - [Communications & I/O] - [RS-232C] to display the following window. You can set arbitrary values, except for items on the table above.



RS-232C (Non-Procedural)

RS-232C (Non-Procedural) Settings

Configurations of RS-232C non-procedural communication with external devices can be changed here.

1/1

Baud Rate 9600

Stop Bit ● 1 ○ 2

Parity Bit ● None ○ Odd ○ Even

Flow Control ● None ○ CTS/RTS

Delimiter ● CR ○ CR+LF ○ Other Set

Data Length 8-bit

□ Set Trigger-command Response to CV-compatible Mode

OK Cancel

3.2.2. Communication setting for Robot controller

RS-232C communication setting for the robot controller is carried out at the Cao.AddController command execution, by specifying an option parameter. Set an appropriate option according to the communication setting of RS-232C on the XG-X series.
You can carry out the RS-232C communication setup with a teach pendant or a mini-pendant; however, these are for Comm.Open command execution, so not applicable to this provider.

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 vntResult as Variant      (3)                     'Declare result acquisition variable

    caoCtrl = Cao.AddController("XGX", "CaoProv.KEYENCE.CVX", "", _
                                    "conn=eth:192.168.0.10")      (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.
Commands are classified the following three types.

- Connection commands
- XG-X series-supported command
- Proprietary extension commands

A XG-X series-supported command is the command that is paired with a XG-X series command. The correspondence between the XG-X series commands and the XG-X series-supported commands is shown in the command list on the next page.
For the detailed operation of XG-X Series commands, refer to the XG-X Series User's manual of KEYENCE.

Table 5-1 Command list

XG-X series provider command	XG-X series command	Usage	Page
Connection commands			
Cao.AddController	—	Implement the provider to a variable and makes a connection to XG-X series.	15
XG-X2000 series-supported command			
Trigger			
Trigger	T1 、T2 、T3 、T4、TA	Issue a specified trigger.	17
Change mode			
ChangeMode	R0、S0	Switch the controller between Run mode and Setup mode.	18
ChangeModeAsync		Change the operation mode (Run or Setup mode) asynchronously	18
ReadMode	RM	Obtain the current operation mode.	19
Change inspection setting number			
ChangeInspectSetting	PW	Load the specified program from the specified SD card.	19
ChangeInspectSettingAsync		Loads the specified program from the specified SD card asynchronously	20
ChangeInspectSettingString		Switch to the specified name's program.	20
ChangeInspectSettingStringAsync		Switch to the specified name's program asynchronously.	21
ReadInspectSetting	PR	Read the currently loaded program number.	22
ReadInspectSettingString		Read the currently loaded program name.	23
Control of controller			
ClearError	CE	Clear the error status.	23
ReadRegisterImageNo	NR	Read out the referenced registered image number.	24
UpdateRegisterImageNo	NU	Change the referenced registered image number to a variable reference value.	25
WriteRegisterImageNo	NW	Change the referenced registered image number.	26
RenameInspectionSetting	PN	Change the inspection setting name.	27
ReturnFlowTop	RE	Return to the top of flow.	28
UpdatePosAdjustment	RR	Update the position adjustment reference value.	29
Reset	RS	Perform reset.	30
ReCalcImageInfo	RU	Re-calculate image reference information.	31
SaveSetting	SS	Save the settings.	31
ExecuteTeaching	TG	Perform teaching.	32
CancelWaitStatus	WG	Cancel the wait status.	33
OCR, 2D code reader, and 1D code reader related			
WriteCharReg	CW	Re-write the registered string.	34
ReadCharReg	CR	Read out the registered string..	35
AutoTuning	AT	Perform auto tuning.	36
RegisterCharLibrary	CA	Register characters into the library.	37
DeleteCharLibrary	CD	Delete a character from the library.	38
Data input/output related			
GetIntVariable	IR	Read out variables with integer values.	39
PutIntVariableEx	IS	Synchronously write variables with integer values.	40
PutIntVariable	IW	Write variables with integer values.	41
GetVariable	MR	Read out variables.	42

PutVariableEx	MS	Synchronously write variables.	43
PutVariable	MW	Write the variables.	44
GetTerminalVariable	RP	Read out terminal variables.	45
PutTerminalOffset	WO	Write the terminal offset.	45
PutTerminalVariable	WP	Write the terminal variables.	46
Recipe function related			
CopyRecipe	RPC	Copy the recipe setting.	46
MoveRecipe	RPM	Move the recipe setting.	47
RenameRecipe	RPN	Rename the recipe setting.	48
ReadRecipe	RPR	Read out the recipe setting number.	49
ReadRecipeString		Read out the recipe setting name.	49
ChangeRecipe	RPW	Change the number of the recipe setting by specifying the recipe setting number.	50
ChangeRecipeString	RPT	Change the recipe setting number by specifying the recipe setting name.	51
Others			
InputSimConsole	KY	Perform console simulated input.	52
SearchUnitNo	UQ	Search the unit number.	53
ReadVersionInfo	VI	Read out the system information of the controller.	53
Original expansion commands			
ExecuteCommand	—	Execute a XG-X series command with a syntax of XG-X series command.	54
ExecuteCommandAsync	—	Execute a XG-X series command with a syntax of XG-X series command asynchronously.	55
TriggerAndGetResult	—	Obtain a result after trigger execution.	56
RecievePacket	—	Obtain the result of trigger input.	57
ClearPacket	—	Delete result data stored in a robot controller.	58
SetTimeout	—	Set a time-out period.	58
GetTimeout	—	Obtain a currently assigned timeout period.	59
GetCommandResult	—	Wait for the completion of the asynchronous command to get the return value of it.	60

Cao.AddController

Usage Implement the provider to a variable and makes a connection to XG-X series.

Syntax Cao.AddController(<Controller name>,<Provider name>,
< Provider execution machine name>,<Option>)

Argument <Controller name>

Assign a name (The name is used for control) (character string).

<Provider name>

Specify "CaoProv.KEYENCE.XGX" with character string type data.

< Provider execution machine name>

Specify "" with character string type data.

<Option>

Specify following items with character string type data.

Syntax "Conn=<Connection parameter>,Timeout=<Timeout>"

Argument <Connection parameter>

This differ from communication methods. Refer to "Description for parameters of each connection".

<Timeout>

Set an allowable waiting time given to the response from XG-X series at this provider's command execution by millisecond-unit. This is optional. This should be 500 milliseconds if it is omitted.

Description for parameters of each connection

For Ethernet

Syntax "eth:<IP address>:<Port number>"

Argument <IP address>

Specify IP address of XG-X series to connect.

<Port number>

Specify port number of XG-X series to connect. This is optional. This should be 8500 if it is omitted.

For RS-232C

Syntax com:<COM Port>:<BaudRate>:<Parity>
:<DataBits>:<StopBits>:<Flow>

Argument <COM Port>

Specify a COM port number of a robot controller plugged in the XG-X series. Entered number will be the COM port number. For example, if you enter 1, it indicates COM1 is specified. If you use a serial communication connector on the front side of the controller while expansion RS-232C communication module is not used, enter 2 in this parameter.

<BaudRate>

According to the communication speed of XG-X series to connect, select suitable baud rate from 4800, 9600, 19200, 38400, 57600, 115200 (bps).

This is optional. This should be "9600" if it is omitted.

<Parity>

According to the XG-X series to connect, select suitable parity from the followings.

N : None
E : Even parity
O : Odd parity

This is optional. This should be "N" if it is omitted.

<DataBits>

According to the data bit count of XG-X series to connect, select suitable number from the followings.

7 : 7 bits
8 : 8 bits

This is optional. This should be "8" if it is omitted.

<StopBits>

According to the stop bit count of XG-X series to connect, select suitable number from the followings.

1 : 1 bit
2 : 2 bits

This is optional. This should be "1" if it is omitted.

<Flow>

The flow control selection is prepared as shown below. However, to communicate with XG-X series, set this parameter to "0: Without flow control".

0 : Without flow control
1 : Xon / Xoff
2 : Hardware control

This is optional. This should be "0 " if it is omitted.

Return value Implemented objects are returned (Object).

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

'=================   For Ethernet    ===================
'To specify a time-out period, but not specify COM port
caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE. XGX", _
                                  "", "conn=eth:192.168.0.10, timeout=1000")

''To specify a time-out period and COM port
caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCEX. XGX", _
                                  "", "conn=eth:192.168.0.10:8503")

'=================   For RS-232C    ====================
'To omit a baud rate and the following.
caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX", _
                                  "", "conn= com:2")

'To specify a baud rate and the followings.
caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX", _
                                  "", "conn= com:2:115200:E:8:1:0, timeout=1000")

'To specify a baud rate and the following.( Specify parity but omit others )
caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX", _
                                  "", "conn= com:2::E:::")
```

<ImplVar>.Trigger

Usage Issue a specified trigger.

Syntax <ImplVar>.Trigger <Trigger No.>

Argument <Trigger No.>

Specify a trigger number (integer).

1 to 4	:	Trigger 1 to 4
-1	:	All triggers

Return value None

Description This command issue a specified trigger. To receive result data generated by trigger, use RecievePacket command. To issue trigger and receive result at one processing, use TriggerAndGetResult command.

Example

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                  "", "conn=eth:192.168.0.10")

'Issue a Trigger1.
caoCtrl.Trigger 1
```

<ImplVar>.ChangeMode

Usage Switch the controller between Run mode and Setup mode.

Syntax <ImplVar>.ChangeMode <Mode>

Argument <Mode>

Specify a desired mode (integer).

0 : Setup mode

1 : Run mode

Return value None

Description This command switches the controller between Run mode and Setup mode.

Example

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'Switch the mode to Run mode.
caoCtrl.ChangeMode 1
```

<ImplVar>.ChangeModeAsync

Usage Change the operation mode (Run or Setup mode) asynchronously

Syntax <ImplVar>.ChangeModeAsync <Mode>

Argument <Mode>

Specify a desired mode (integer).

0 : Setup mode

1 : Run mode

Return value None

Description Change the operation mode (Run or Setup mode) asynchronously
To obtain and check the return value of the command, use GetCommandResult command.

Example

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'Switch the mode to Run mode.
caoCtrl.ChangeModeAsync 1

'Obtain the return value of ChangeModeAsync command
vntResult = caoCtrl.GetCommandResult
```

<ImplVar>.ReadMode

Usage Obtain the current operation mode.

Syntax <ImplVar>. ReadMode ( )

Argument None

Operation mode (integer).

0 ： Setup mode

1 ： Run mode

Return value None

Description This command obtains the current operation mode.

Example

```
Dim caoCtrl as Object
Dim lMode as Long

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'Obtain the current operation mode.
lMode=caoCtrl.ReadMode
```

<ImplVar>.ChangeInspectSetting

Usage Load the specified program from the specified SD card.

Syntax <ImplVar>.ChangeInspectSetting <SD card number>,<Inspection setting number>

Argument <SD card number>

Specify an SD card number with integer type data .

1 ： SD1

2 ： SD2

<Inspection setting number>

Specify an inspection setting number with integer type data ranging from 0 to 999.

Return value None

Description This command loads the specified program from the specified SD card.

Example

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'Change the setting to the inspection setting number 1 of the SD1.
caoCtrl.ChangeInspectSetting 1,1
```

<ImplVar>.ChangeInspectSettingAsync

Usage Loads the specified program from the specified SD card asynchronously

Syntax <ImplVar>.ChangeInspectSettingAsync <SD card number>,<Inspection setting number>

Argument <SD card number>

Specify an SD card number with integer type data .

1 ： SD1

2 ： SD2

<Inspection setting number>

Specify an inspection setting number with integer type data ranging from 0 to 999.

Return value None

Description This command loads the specified program from the specified SD card asynchronously.
To obtain and check the return value of the command, use GetCommandResult command.

Example

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'Change the setting to the inspection setting number 1 of the SD1.
caoCtrl.ChangeInspectSettingAsync 1,1

'Obtain the return value of ChangeInspectionSettingAsync command
vntResult = caoCtrl.GetCommandResult
```

<ImplVar>.ChangeInspectSettingString

Usage Switch to the specified name's program.

Syntax <ImplVar>.ChangeInspectSettingString <strName >

Argument <strName >

Specify the program name with character string type data .

Return value None

Description This command changes the program to the specified name's program.

Example

```
caoCtrl.ChangeInspectSettingString "test"
```

<ImplVar>.ChangeInspectSettingStringAsync

Usage Switch to the specified name's program asynchronously.
To obtain and check the return value of the command, use GetCommandResult command.

Syntax <ImplVar>.ChangeInspectSettingStringAsync <strName >

Argument <strName >

Specify the program name with character string type data .

Return value None

Description This command changes the program to the specified name's program.

Example

```
Dim caoCtrl as Object

'Select the inspection program which name is "test".
caoCtrl.ChangeInspectSettingStringAsync "test"

'Obtain the return value of ChangeInspectionSettingAsync command.
vntResult = caoCtrl.GetCommandResult
```

<ImplVar>.ReadInspectSetting

Usage Read the currently loaded program number.

Syntax <ImplVar>.ReadInspectSetting

Argument None

Return value The following two items are stored in an array of integer.

<SD card number>

Currently selected SD card number

1 ： SD1

2 ： SD2

<Inspection setting number>

Currently selected program number.

Description This command reads the currently loaded program number.

Example

```
Dim caoCtrl as Object
Dim vntRet as Variant
Dim iaryData(1) as Integer

caoCtrl=Cao.AddController("XGX","CaoProv.KEYENCE.XGX","", _
                                        "conn=eth:192.168.0.10")

'Obtain currently selected inspection setting number and
'its SD card number.
'iaryData(0) stores an SD card number.
'iaryData(1) stores an inspection setting number.
vntRet = caoCtrl.ReadInspectSetting
iaryData(0) = vntRet(0)
iaryData(1) = vntRet(1)
```

<ImplVar>.ReadInspectSettingString

Usage Read the currently loaded program name.

Syntax <ImplVar>.ReadInspectSettingString

Argument None

Return value <Inspection setting name>

Currently selected program name.

Description This command reads the currently loaded program name.

Example

```
Dim caoCtrl as Object
Dim strRet as String

caoCtrl=Cao.AddController("XGX","CaoProv.KEYENCE.XGX","", _
                                        "conn=eth:192.168.0.10")

strRet = caoCtrl.ReadInspectSettingString
```

<ImplVar>.ClearError

Usage Clear the error status.

Syntax <ImplVar>.ClearError

Argument None

Return value None

Description This command clears the error status. Even when an error status does not exist, the command execution finishes normally.

Example

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                  "", "conn=eth:192.168.0.10")

'Clear the error status.
caoCtrl.ClearError
```

<ImplVar>.ReadRegisterImageNo

Usage Read out the referenced registered image number.

Syntax 1 Without specifying the image number

<ImplVar>.ReadRegisterImageNo (<UnitNo>)

Argument <UnitNo>

Specify the unit ID (integer). Specify a value within the range of 0 to 999.

Return value <Registered image No.>

The range of registered image numbers (integer) is 0 to 999.

Syntax 2 By specifying the image number

<ImplVar>.ReadRegisterImageNo (< UnitNo>, <ImageNo>)

Argument <UnitNo>

Specify the unit ID (integer). Specify a value within the range of 0 to 999.

<ImageNo>

Specify the original image number (integer) when specifying the unit ID of the image computing unit or C language unit. Specify a value within the range of 1 to 2.
Specify the teaching image number (integer) when specifying the calibration unit ID. Specify a value within the range of 1 to 16.

Return value <Registered image No.>

The range of registered image numbers (integer) is 0 to 999.

Description This command reads out the registered image number currently used for measurement.

Example

```
Dim caoCtrl as Object
Dim lRegisterImageNo as Long

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                   "", "conn=eth:192.168.0.10")

'Obtain the registered image number currently used for measurement by unit ID
101.
lRegisterImageNo = caoCtrl.ReadRegisterImageNo(101)
```

<ImplVar>.UpdateRegisterImageNo

Usage Change the referenced registered image number to a variable reference value.

Syntax <ImplVar>.UpdateRegisterImageNo <UnitNo>

Argument <UnitNo>

Specify the unit ID (integer). Specify a value within the range of 0 to 999.

0 to 999	Unit ID to specify
-1	All units

Return value None

Description When the registered image number (image to be changed to) of the specified unit is referenced with a variable, this command obtains the current value of the variable and changes the registered image number. If necessary, image reference information will be updated with the registered image of that number.

Example

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'Update the variable reference of the registered image in unit ID 101 and change the
registered image number.
caoCtrl.UpdateRegisterImageNo 101
```

<ImplVar>.WriteRegisterImageNo

Usage Change the referenced registered image number.

Syntax 1 Without specifying the image number

<ImplVar>. WriteRegisterImageNo <UnitNo>, <Registered image No.>

Argument <UnitNo>

Specify the unit ID (integer). Specify a value within the range of 0 to 999.

0 to 999	Unit ID to specify
-1	All units

<Registered image No.>

Specify the registered image number (integer) within the range of 0 to 999.

Return value None

Syntax 2 By specifying the image number

<ImplVar>. WriteRegisterImageNo <UnitNo>, <Registered image No.>, < ImageNo >

Argument <UnitNo>

Specify the unit ID (integer). Specify a value within the range of 0 to 999.

0 to 999	Unit ID to specify
-1	All units

<Registered image No.>

Specify the registered image number (integer) within the range of 0 to 999.

<ImageNo>

Specify the original image number (integer) when specifying the unit ID of the image computing unit or C language unit. Specify a value within the range of 1 to 2.
Specify the teaching image number (integer) when specifying the calibration unit ID. Specify a value within the range of 1 to 16.

Return value None

Description This command changes the registered image number of the specified unit. If necessary, image reference information will be updated with the registered image of that number.

Example

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'Change the registered image number in unit ID 101 to "202".
caoCtrl.WriteRegisterImageNo 101, 202
```

<ImplVar>.RenameInspectionSetting

Usage Change the inspection setting name.

Syntax <ImplVar>.RenameInspectionSetting <SD card number>, <Inspection setting number>, <strName>, [<Inspection setting name type>]

Argument <SD card number>

Specify a desired mode (integer). Specify a value within the range of 1 to 2.

<SD card number>

Specify the inspection setting number (integer). Specify a value within the range of 0 to 999.

<strName>

Specify the name with a character string or scalar type array variable (character string).

<Inspection setting name type>

Specify the type for <strName> (boolean type). This is optional.

FALSE	<strName> is treated as a character string (when omitted).
TRUE	<strName> is treated as a scalar type array variable.

Return value None

Description This command changes the inspection setting name of the specified number.

Example

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                    "", "conn=eth:192.168.0.10")

'Re-write the inspection setting name of SD card number 1, inspection setting
number 101 to "TestName".
caoCtrl.RenameInspectionSetting 1, 101, "TestName", FALSE
```

<ImplVar>.ReturnFlowTop

Usage Return to the top of flow.

Syntax <ImplVar>. ReturnFlowTop

Argument None

Return value None

Description This command performs a jump to the next unit that follows the start unit; applies to any waiting unit and imaging unit in wait status, except for units in dialog condition wait status or timer condition wait status.

Example

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'Jump to the next unit that follows the start unit.
caoCtrl.ReturnFlowTo
```

<ImplVar>.UpdatePosAdjustment

Usage Update the position adjustment reference value.

Syntax 1 Without specifying the value to obtain as the reference value

<ImplVar>.UpdatePosAdjustment <UnitNo>

Argument <UnitNo>

Specify the unit ID (integer).

0 to 999	Unit ID to specify
-1	All units

Return value None

Syntax 2 By specifying the value to obtain as the reference value

<ImplVar>.UpdatePosAdjustment <UnitNo>, < Reference value >

Argument <UnitNo>

Specify the unit ID (integer). Specify a value within the range of 0 to 999.

0 to 999	Unit ID to specify
-1	All units

< Reference value >

Specify the value (integer) to obtain as the reference value.

0	Latest results
1	Results of re-calculation by registered image

Return value None

Description This command obtains the latest value currently referenced by the specified position adjustment unit or the results (value) of re-calculating the registered image, as the reference value.

Example

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                    "", "conn=eth:192.168.0.10")

'Obtain the latest value currently referenced by unit ID 101 as the reference value.
caoCtrl.UpdatePosAdjustment 101
```

<ImplVar>.Reset

Usage Perform reset.

Syntax <ImplVar>.Reset

Argument None

Return value None

Description This command performs all of the followings:

- Initializes all system variables and clears all types of buffers including images.
- Cancels trigger wait and event wait of units.
- Newly creates a file name for the file where data is saved.
- Initializes all user-defined local variables of which [Initialize upon reset] is set to ON.
- Initializes all user-defined global variables of which [Initialize upon reset] is set to ON.
- Initializes %JAHold.
- Returns to the top of flow.
- Clears all history data.
- Clears all statistic data.
- Clears all failure classification results.

Example

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                   "", "conn=eth:192.168.0.10")

'Perform reset.
caoCtrl.Reset
```

<ImplVar>.ReCalcImageInfo

Usage Re-calculate image reference information.

Syntax <ImplVar>. ReCalcImageInfo <UnitNo>

Argument <UnitNo>

Specify the unit ID (integer).

0 to 999	Unit ID to specify
-1	All units

Return value None

Description This command updates the image reference information of the specified unit ID with the results of re-calculating the current registered image and setting parameters.

Example

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'Re-calculate image reference information of unit ID 101.
caoCtrl.ReCalcImageInfo 101
```

<ImplVar>.SaveSetting

Usage Save the settings.

Syntax <ImplVar>.SaveSetting

Argument None

Return value None

Description This command saves the current inspection settings, global variables, local variables, and environment settings.

Example

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'Save the current settings.
caoCtrl.SaveSetting
```

<ImplVar>.ExecuteTeaching

Usage Perform teaching.

Syntax <ImplVar>.ExecuteTeaching <UnitNo>

Argument <UnitNo>

Specify the unit ID (integer). Specify a value within the range of 0 to 999.

Return value None

Description This command performs teaching using the currently set registered image for the specified calibration unit.

Example

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'Perform teaching for unit ID 101.
caoCtrl. ExecuteTeaching 101
```

<ImplVar>.CancelWaitStatus

Usage Cancel the wait status.

Syntax <ImplVar>. CancelWaitStatus <Match condition>

Argument <Match condition>

Specify the bit (integer) to allocate the match condition. Specify a value within the range of 0 to (2^{20}-1).

Return value None

Description This command cancels the wait status of units in terminal condition wait or variable condition wait. Using input parameters, the result data (bitwise OR of determination number and match condition) of units in wait status to be cancelled can be set to a given status.

- Specify "0" for the input parameter when the result data of the unit in wait status to be cancelled will not be referenced. In this case, the bitwise OR of determination number and match condition will remain 0, and the wait status will be cancelled.
- The input parameter is allocated to the match condition as a binary. The condition allocated to the least significant bit among all bits that are "1" will be the determination number.

Example

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                              "", "conn=eth:192.168.0.10")

'Do not reference result data of the unit in wait status to be cancelled.
caoCtrl.CancelWaitStatus 0
```

<ImplVar>.WriteCharReg

Usage Re-write the registered string.

Syntax <ImplVar>.WriteCharReg <UnitNo>[, <Line No.>,
<Registered string >, [, <Determination type>]]

Argument <UnitNo>

Specify the unit ID (integer). Specify a value within the range of 0 to 999.

<Line No.>

Specify the line number (integer). This is optional.
For OCR units, specify a value within the range of 1 to 2.
For 1D and 2D code reader tools, specify a value within the range of 1 to 16.

<Registered string >

Specify the registered string with a character string or scalar type array variable (character string). This is optional.

<Determination type>

Specify the type for <Registered string> (boolean type). This is optional.

FALSE	<Registered string> is treated as a character string (when omitted).
TRUE	<Registered string> is treated as a scalar type array variable.

Return value None

Description This command re-writes the REG for OCR, 2D code readers, or 1D code readers. Depending on the format, the behavior changes as follows: For details on ASCII codes that can be specified for OCRs, refer to "Character code table for OCR units" (page 2-91) in the XG-X2000 Series manual.

1. When the arguments are <UnitNo>, <Line No.>, and <Registered string>, and the <Determination type> is "character string"
   Sets the REG contents for the match condition of the specified line number of the specified unit ID as the character string.
2. When the arguments are <UnitNo>, <Line No.>, and <Registered string>, and the <Determination type> is "scalar type array variable"
   Set the value of a scalar array type variable in ASCII codes as the content of REG for the match condition of the specified line number of the specified unit ID.
3. When the argument is <UnitNo>
   Sets the latest obtained results of the unit to the REG of the specified unit ID. If measurement has not been performed, this will be cleared. (For OCR units, a space is inserted.)

Example

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                          "", "conn=eth:192.168.0.10")

'Set the registered string for the OCR tool with tool number 101 to "DEF".
caoCtrl.WriteCharReg 101,1,"DEF"
```

<ImplVar>.ReadCharReg

Usage Read out the registered string.

Syntax <ImplVar>.ReadCharReg (<UnitNo>, <Line No.>[, <Registered string >])

Argument <UnitNo>

Specify the unit ID (integer). Specify a value within the range of 0 to 999.

<Line No.>

Specify the line number (integer).

For OCR units, specify a value within the range of 1 to 2.

For 1D and 2D code reader tools, specify a value within the range of 1 to 16.

<Registered string >

Specify the registered string with a scalar type array variable. This is optional.

Return value REG contents (character string) or none.

When <Registered string> is not specified, the obtained REG contents (character string) will be returned.

When <Registered string> is specified, no value will be returned.

Description This command reads out the REG for OCR, 2D code readers, or 1D code readers.

Depending on the format, the behavior changes as follows:

1. When the arguments are <UnitNo> and <Line No.>
   Return value is the REG contents for the match condition of the specified line number of the specified unit ID.
2. When the arguments are <UnitNo>, <Line No.>, and <Registered string>
   Stores the REG contents for the match condition of the specified line number of the specified unit ID, in the order of the elements specified in the index of the scalar type array variable, as the ASCII code.

Example

```
Dim caoCtrl as Object
Dim bstrParam as String

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'Obtain the character string of OCR unit line number 1 of unit ID 101.
bstrParam = caoCtrl.ReadCharReg(101,1)
```

<ImplVar>.AutoTuning

Usage Perform auto tuning.

Syntax <ImplVar>. AutoTuning <UnitNo>, <Target image>

Argument <UnitNo>

Specify the unit ID (integer). Specify a value within the range of 0 to 999.

<Target image>

Specify the target image (integer) to perform tuning.

0	Specify the input image as the target.
1	Specify the registered image as the target.

Return value None

Description This command performs auto tuning using the input image or registered image for the specified 2D or 1D code reader unit.

Example

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                    "", "conn=eth:192.168.0.10")

'Perform auto tuning using the registered image as the target for unit ID 101.
caoCtrl.AutoTuning 101, 1
```

<ImplVar>.RegisterCharLibrary

Usage Register characters into the library.

Syntax 1 To register the latest search results into the library

<ImplVar>.RegisterCharLibrary <UnitNo>, <Line No. of the search result>, <Character No. of the search result>, <Character type>

Argument <UnitNo>

Specify the unit ID (integer). Specify a value within the range of 0 to 999.

<Line No. of the search result>

Specify the line number (integer) of the search results. Specify a value within the range of 1 to 2.

<Character No. of the search result>

Specify the character number (integer) of the search results. Specify a value within the range of 0 to 20.

<Character type>

Specify the character type (integer) to register. Specify a value within the range of -1 to 59.

Return value None

Syntax 2 To register the search history results into the library

<ImplVar>.RegisterCharLibrary <UnitNo>, <Image archive conditions>, <Past jth time>, <Line No. of the search result>, <Character No. of the search result>, <Character type>

Argument <UnitNo>

Specify the unit ID (integer). Specify a value within the range of 0 to 999.

<Image archive conditions>

Specify the history accumulation condition (integer). Specify a value within the range of 0 to 7.

<Past jth time>

Specify the past jth time (integer). Specify a value within the range of 0 to (accumulation condition count - 1).

<Line No. of the search result>

Specify the line number (integer) of the search results. Specify a value within the range of 1 to 2.

<Character No. of the search result>

Specify the character number (integer) of the search results. Specify a value within the range of 1 to 20.

<Character type>

Specify the character type (integer) to register. Specify a value within the range of -1 to 59.

Return value None

Description This command registers into the library the latest search results or the specified character in the history image or search history results as a character of the specified type.

Example

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'Register the latest search results (line number 1, character number 2, character type 3 to register) for unit ID 101 into the library.
caoCtrl.RegisterCharLibrary 101,1,2,3
```

<ImplVar>.DeleteCharLibrary

Usage Delete a character from the library.

Syntax <ImplVar>. DeleteCharLibrary <UnitNo>, <Character type>

Argument <UnitNo>

Specify the unit ID (integer). Specify a value within the range of 0 to 999.

<Character type>

Specify the character type (integer). Specify a value within the range of -1 to 59.

Return value None

Description This command deletes from the library the character with the last registration number of the specified character type.

Example

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'Delete character type 1 of unit ID 101.
caoCtrl.DeleteCharLibrary 101, 1
```

<ImplVar>.GetIntVariable

Usage Read out variables with integer values.

Syntax <ImplVar>.GetIntVariable (<VarName1> [,<VarName2> [,<VarName3> [,<VarName4> [,<VarName5> [,<VarName6> [,<VarName7> [,<VarName8> [,<VarName9> [,<VarName10> [,<VarName11> [,<VarName12> [,<VarName13> [,<VarName14> [,<VarName15> [,<VarName16>]]]]]]]]]]]]]]])

Argument <VarName1 to 16>

Specify the name (character string) of the caller variable.

Return value Numerical value (double-precision real | array) of the caller variable

Description This command reads out the values of variables, rounds them off to an integer, and outputs those integer values.

Up to 16 variables can be read out at once.

Example

```
Dim caoCtrl as Object
Dim vntValList as VARIANT

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'Specify the name (character string) of the caller variable.
vntValList = caoCtrl.GetIntVariable("#Data1","#Data2","#Data3")
```

<ImplVar>.PutIntVariableEx

Usage Synchronously write variables with integer values.

Syntax <ImplVar>. PutIntVariableEx <Variable name list>, <Specified value>

Argument <Variable name list>

Specify the variable name list (character string | array) to which the values are written.

<Specified value>

Specify the numerical value or variable name list (character string | array) from where the values are written.

Return value None

Description This command writes integer values for variables. At the first imaging unit in the flow or end unit at the end of the flow, the values are re-written with the specified variable values.

Up to 16 variable names in a list can be specified at once to be written to (destination) with the same number of numerical values or variable names in a list specified as the source.

If the number of elements specified as the source does not match with the number of elements in the destination, the operation will end with an error.

Example

```
Dim caoCtrl as Object
Dim vntNameList as VARIANT
Dim vntValList as VARIANT

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

vntNameList = Array("#Data1", "#Data2", "#Data3")
vntValList = Array(1, 2, 3)

'Write values "1", "2", "3" for variable names "#Data1", "#Data2", "#Data3".
caoCtrl.PutIntVariableEx vntNameList, vntValList
```

<ImplVar>.PutIntVariable

Usage Write variables with integer values.

Syntax <ImplVar>. PutIntVariable <Variable name list>, <Specified value>

Argument <Variable name list>

Specify the variable name list (character string | array) to which the values are written.

<Specified value>

Specify the numerical value or variable name list (character string | array) from where the values are written.

Return value None

Description This command rounds off the specified values to integers and writes those integer values to variables. Values will be applied upon execution.

Up to 16 variable names in a list can be specified at once to be written to (destination) with the same number of numerical values or variable names in a list specified as the source.

If the number of elements specified as the source does not match with the number of elements in the destination, the operation will end with an error.

Example

```
Dim caoCtrl as Object
Dim vntNameList as VARIANT
Dim vntValList as VARIANT

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

vntNameList = Array("#Data1", "#Data2", "#Data3")
vntValList = Array(1, 2, 3)

'Write values "1", "2", "3" for variable names "#Data1", "#Data2", "#Data3".
caoCtrl.PutIntVariable vntNameList, vntValList
```

<ImplVar>.GetVariable

Usage Read out variables.

Syntax <ImplVar>. GetVariable (<VarName1> [,<VarName2> [,<VarName3> [,<VarName4> [,<VarName5> [,<VarName6> [,<VarName7> [,<VarName8> [,<VarName9> [,<VarName10> [,<VarName11> [,<VarName12> [,<VarName13> [,<VarName14> [,<VarName15> [,<VarName16>]]]]]]]]]]]]]]])

Argument <VarName1 to 16>

Specify the name (character string) of the caller variable.

Return value Numerical value (double-precision real | array) of the caller variable

Description This command reads out the values for the specified scalar type variables.

Up to 16 variables can be read out at once.

Example

```
Dim caoCtrl as Object
Dim vntValList as VARIANT

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'Read out numerical values for variable names "#Data1", "#Data2", "#Data3".
vntValList = caoCtrl.GetVariable("#Data1","#Data2","#Data3")
```

<ImplVar>.PutVariableEx

Usage Synchronously write variables.

Syntax <ImplVar>. PutVariableEx <Variable name list>, <Specified value>

Argument <Variable name list>

Specify the variable name list (character string | array) to which the values are written.

<Specified value>

Specify the numerical value or variable name list (character string | array) from where the values are written.

Return value None

Description This command writes variables. At the first imaging unit in the flow or end unit at the end of the flow, the values are re-written with the specified variable values.

Up to 16 variable names in a list can be specified at once to be written to (destination) with the same number of numerical values or variable names in a list specified as the source.

If the number of elements specified as the source does not match with the number of elements in the destination, the operation will end with an error.

Example

```
Dim caoCtrl as Object
Dim vntNameList as VARIANT
Dim vntValList as VARIANT

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

vntNameList = Array("#Data1", "#Data2", "#Data3")
vntValList = Array(1, 2, 3)

'Write values "1", "2", "3" for variable names "#Data1", "#Data2", "#Data3".
caoCtrl.PutVariableEx vntNameList, vntValList
```

<ImplVar>.PutVariable

Usage Write the variables.

Syntax <ImplVar>. PutVariableEx <Variable name list>, <Specified value>

Argument <Variable name list>

Specify the variable name list (character string | array) to which the values are written.

<Specified value>

Specify the numerical value or variable name list (character string | array) from where the values are written.

Return value None

Description This command re-writes the specified scalar type variables (global or local variables) with the specified values. Values will be applied upon execution.

Up to 16 variable names in a list can be specified at once to be written to (destination) with the same number of numerical values or variable names in a list specified as the source.

If the number of elements specified as the source does not match with the number of elements in the destination, the operation will end with an error.

Example

```
Dim caoCtrl as Object
Dim vntNameList as VARIANT
Dim vntValList as VARIANT

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

vntNameList = Array("#Data1", "#Data2", "#Data3")
vntValList = Array(1, 2, 3)

'Write values "1", "2", "3" for variable names "#Data1", "#Data2", "#Data3".
caoCtrl.PutVariable vntNameList, vntValList
```

<ImplVar>.GetTerminalVariable

Usage Read out terminal variables.

Syntax <ImplVar>.GetTerminalVariable (<system variable>)

Argument <system variable>

The system variable name (character string) (%OutDataAsyncA to %OutDataAsyncH) to read out.

Return value Read out values (integer)

Description This command reads out the status of the specified terminals.

Example

```
Dim caoCtrl as Object
Dim lTerminalCond as Long

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'Read out the terminal status of %OutDataAsyncA.
lTerminalCond = caoCtrl. GetTerminalVariable("%OutDataAsyncA")
```

<ImplVar>.PutTerminalOffset

Usage Write the terminal offset.

Syntax <ImplVar>. PutTerminalOffset <Offset scale >, <Offset value >

Argument <Offset scale >

Specify the offset scale (integer).

<Offset value >

Specify the offset value (integer).

Return value None

Description This command writes to %CmdParamOffset the value obtained by multiplying the offset scale and offset value.

Example

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'Write offset scale of "2" with offset value of "1".
caoCtrl.PutTerminalOffset 2, 1
```

<ImplVar>.PutTerminalVariable

Usage Write the terminal variables.

Syntax <ImplVar>. PutTerminalVariable <system variable>, <Specified value>

Argument <system variable>

Specify the system variable names (character string | array) (%OutDataAsyncA to %OutDataAsyncH) to write.

<Specified value>

Specify the numerical values or variable names to be written.

Return value None

Description This command re-writes the system variable values that are writable from commands and can be allocated to terminals. Values will be applied upon execution.

Example

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                   "", "conn=eth:192.168.0.10")

'Write "1" to %OutDataAsyncA.
caoCtrl. PutTerminalVariable "%OutDataAsyncA", 1
```

<ImplVar>.CopyRecipe

Usage Copy the recipe setting.

Syntax <ImplVar>.CopyRecipe <Copy source recipe No>, <Copy destination recipe No>

Argument <Copy source recipe No>

Specify the copy source recipe number (integer). Specify a value within the range of 0 to 999.

<Copy destination recipe No>

Specify the copy destination recipe number (integer). Specify a value within the range of 0 to 999.

Return value None

Description This command overwrites all of the recipe setting specified as the copy destination with contents of the recipe setting specified as the copy source.

Example

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                   "", "conn=eth:192.168.0.10")

'Copy the recipe of recipe number 1 to recipe number 2.
caoCtrl.CopyRecipe 1, 2
```

<ImplVar>.MoveRecipe

Usage Move the recipe setting.

Syntax <ImplVar>. MoveRecipe <Move source recipe No>, <Move destination recipe No>

Argument <Move source recipe No>

Specify the move source recipe number (integer). Specify a value within the range of 0 to 999.

<Move destination recipe No>

Specify the move destination recipe number (integer). Specify a value within the range of 0 to 999.

Return value None

Description This command overwrites all of the recipe setting specified as the move destination with contents of the recipe setting specified as the move source. When move is successfully completed, all contents of the move source recipe setting will be deleted.

Example

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'Move the recipe of recipe number 1 to recipe number 2.
caoCtrl. MoveRecipe 1, 2
```

<ImplVar>.RenameRecipe

Usage Rename the recipe setting.

Syntax <ImplVar>.RenameRecipe <Recipe setting No>, <Recipe setting name>[, <Setting type>]

Argument <Recipe setting No>

Specify the recipe setting number (integer). Specify a value within the range of 0 to 999.

<Recipe setting name>

Specify the recipe setting name (character string).

<Setting type>

Specify the type for <Recipe setting name> (boolean type). This is optional.

FALSE	<Recipe setting name> is treated as a character string (when omitted).
TRUE	<Recipe setting name> is treated as a scalar type array variable.

Return value None

Description This command changes the name of the recipe setting of the specified number.

Example

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                    "", "conn=eth:192.168.0.10")

'Change the recipe setting name of recipe number 1 to "Recipe1"
caoCtrl.RenameRecipe 1, "Recipe1", FALSE
```

<ImplVar>.ReadRecipe

Usage Read out the recipe setting number.

Syntax <ImplVar>. ReadRecipe ( )

Argument None

Return value Returns the recipe setting number (integer).

0～999	Current recipe setting number
-1	When no setting is used

Description This command reads out the recipe setting number currently used.

Example

```
Dim caoCtrl as Object
Dim lRecipeInfoNo as long

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                              "", "conn=eth:192.168.0.10")

'Read out the recipe setting number currently used.
lRecipeInfoNo = caoCtrl.ReadRecipe ( )
```

<ImplVar>.ReadRecipeString

Usage Read out the recipe setting name.

Syntax <ImplVar>. ReadRecipeString ( )

Argument None

Return value Returns the recipe setting name (character string).

Description This command returns the name of the recipe setting currently used.
When no setting is used, an empty string is returned.

Example

```
Dim caoCtrl as Object
Dim strRecipeInfoName as string

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                              "", "conn=eth:192.168.0.10")

'Read out the number of the recipe setting currently used.
strRecipeInfoName = caoCtrl.ReadRecipeString ( )
```

<ImplVar>.ChangeRecipe

Usage Change the number of the recipe setting by specifying the recipe setting number.

Syntax <ImplVar>.ChangeRecipe <Recipe setting No>[, <Save specification>]

Argument <Recipe setting No>

Specify the recipe settings number (integer) to change.

-1	Do not use recipe setting.
0 to 999	Specify the recipe setting number.

<Save specification>

Specify whether to save or not (integer). This is optional.

0	Do not save the recipe setting number after change (when omitted).
1	Save the recipe settings number after change.

Return value None

Description This command closes all open dialogs and changes to the recipe setting of the specified number.

Example

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'Change to recipe setting number 1 without saving the recipe setting number after
the change.
caoCtrl.ChangeRecipe 1, 0
```

<ImplVar>.ChangeRecipeString

Usage Change the recipe setting number by specifying the recipe setting name.

Syntax <ImplVar>.ChangeRecipeString <Recipe setting name>[, <Save specification>]

Argument <Recipe setting name>

Specify the recipe setting name (character string) to change.

<Save specification>

Specify whether to save or not (integer). This is optional.

0	Do not save the recipe setting number after change (when omitted).
1	Save the recipe settings number after change.

Return value None

Description This command closes all open dialogs and changes to the recipe setting of the specified name.

Example

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'Change the recipe name to "Recipe1" without saving the recipe setting number after the change.
caoCtrl.ChangeRecipeString "Recipe1", 0
```

<ImplVar>.InputSimConsole

Usage Perform console simulated input.

Syntax <ImplVar>.InputSimConsole <Key code 1>[, <Key code 2>]

Argument <Key code 1>

Specify the key code (integer). For details on key code values, refer to the descriptions below.

<Key code 2>

Specify the key code (integer). This is optional. For details on key code values, refer to the descriptions below.

Return value None

Description This command performs input that simulates console button operations for communication commands.

Specifying two key codes will be treated as an operation in which those two keys are pressed simultaneously.

Key code values are as follows:

0: [0] key, 1: [1] key, 2: [2] key, 3: [3] key, 4: [4] key, 5: [5] key, 6: [6] key,
7: [7] key, 8: [8] key
11: "Down and to the left" key, 12: "Down" key, 13: "Down and to the right" key,
14: "Left" key, 16: "Right" key, 17: "Up and to the left" key, 18: "Up" key,
19: "Up and to the right" key

*These codes correspond to the numeric keypad of the PC. The value of each code is obtained by adding 10 to the number assigned to the key in each direction from key 5 at the center.

Example

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'Simultaneously input the [0] key and [1] key.
caoCtrl. InputSimConsole 0, 1
```

<ImplVar>.SearchUnitNo

Usage Search the unit number.

Syntax <ImplVar>.SearchUnitNo (<Unit name>[, <Unit name type>])

Argument <Unit name>

Specify the unit name with a character string or scalar type array variable (character string).

<Unit name type>

Specify the type for <Unit name> (boolean type).

FALSE	Specify the type for <Unit name> (boolean type).
TRUE	Specify the type for <Unit name> (boolean type).

Return value Returns the unit number (integer).

Description This command searches within the flow the unit of the specified name and returns the unit number. If multiple units of the same name exist, the smallest unit number will be returned.

Example

```
Dim caoCtrl as Object
Dim lUnitNo as long

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'Specify the unit name "Capture" and return its unit number.
lUnitNo = caoCtrl.SearchUnitNo("Capture", FALSE)
```

<ImplVar>.ReadVersionInfo

Usage Read out the system information of the controller.

Syntax <ImplVar>.ReadVersionInfo ( )

Argument None

Return value Returns the system information (character string | array).

Description This command reads out the system information of the controller.

The return value (system information) is stored in the following order: type, firmware version.

Example

```
Dim caoCtrl as Object
Dim vntVerInfo as Variant

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'Obtain the system information.
vntVerInfo = caoCtrl.ReadVersionInfo ( )
```

<ImplVar>.ExecuteCommand

Usage Execute a XG-X series command with a syntax of XG-X series command.

Syntax <ImplVar>.ExecuteCommand (<XG-X series command syntax>)

Argument <XG-X series command syntax>

Specify XG-X series command syntax with character string type data.

Return value <Execution result data of XG-X series command>

The return value is the execution result data of XG-X series command. The data is returned with character string type data.

Description Execute a XG-X series command with a syntax of XG-X series command. For detailed operation of XG-X Series commands, refer to the XG-X Series User's manual of KEYENCE.

Example

```
Dim caoCtrl as Object
Dim strRet as String

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'Specify the XG-X series command and switch the XG-X to run mode.
strRet = caoCtrl. ExecuteCommand("R0")
```

<ImplVar>.ExecuteCommandAsync

Usage Execute a XG-X series command with a syntax of XG-X series command asynchronously.

Syntax <ImplVar>.ExecuteCommandAsync <XG-X series command syntax>

Argument <XG-X series command syntax>

Specify XG-X series command syntax with character string type data.

Return value None

Description Execute a XG-X series command with a syntax of XG-X series command asynchronously. For detailed operation of XG-X Series commands, refer to the XG-X Series User's manual of KEYENCE.

To obtain and check the return value of the command, use GetCommandResult command.

Example

```
Dim caoCtrl as Object
Dim vntResult as Variant

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")
'Specify the XG-X series command and switch the XG-X to run mode.
caoCtrl.ExecuteCommandAsync "R0"

'Obtain the return value for the ExecuteCommandAsync command.
vntResult = caoCtrl.GetCommandResult ( )
```

<ImplVar>.TriggerAndGetResult

Usage Obtain a result after trigger execution.

Syntax <ImplVar>.TriggerAndGetResult (<Trigger No.>)

Argument <Trigger No.>

Specify a trigger number with integer type data.

1 to 4 : Trigger 1 to 4

Return value <Result data>

Result of a trigger execution is returned with character string type data.

Description Obtain the result after trigger execution. If no result data returns from XG-X series, wait until time-out period passes. (To set time-out period, use Cao.AddController command option, or SetTimeout command). If still no result data returns, an error is issued.

If you want to execute other operations while waiting for the result, after inputting a trigger, do the following steps; 1) Input trigger with Trigger command. 2) Execute desired operations. 3) Obtain the result data with RecievePacket command.

Example

```
Dim caoCtrl as Object
Dim strRet as String

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                                  "", "conn=eth:192.168.0.10")

'Input trigger in Trigger 1 and then obtain the result.
strRet = caoCtrl.TriggerAndGetResult(1)
```

<ImplVar>.RecievePacket

Usage Obtain the result of trigger input.

Syntax <ImplVar>.RecievePacket

Argument None

Return value <Result data>

Result data generated by trigger input is received with character string type data.

Description Obtain result data generated by trigger input.

If the XG-X series is set so as to generate no result output against trigger input, no result data returns from XG-X series. As a result, an error is issued when a time-out period passes. (Timeout period is set by Cao.AddController command option, or SetTimeout command).

Also, after trigger input, if you input trigger one more time without executing ReceivePacket command, the result data for two of trigger inputs are stored in a robot controller. Under the condition if you execute the ReceivePacket command, the first trigger's result data will be returned.

Therefore, if the situation where the number of trigger input does not match with the number of ReceivePacket command execution occurs, delete the result data stored in the robot controller by executing ClearPacket command first. Then, input trigger again, and then execute ReceivePacket command to obtain result data.

Example

```
Dim caoCtrl as Object
Dim strRet as String

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'Input trigger in Trigger 1.
caoCtrl.Trigger 1

'Obtain the result data.
strRet = caoCtrl.RecievePacket
```

<ImplVar>.ClearPacket

Usage Delete result data stored in a robot controller.

Syntax <ImplVar>.ClearPacket

Argument None

Return value None

Description Delete result data stored in a robot controller

Example

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                    "", "conn=eth:192.168.0.10")

'Delete result data
caoCtrl.ClearPacket
```

<ImplVar>.SetTimeout

Usage Set a time-out period.

Syntax <ImplVar>.SetTimeout <Time>

Argument <Time>

Set a time-out period with integer type data. Unit is millisecond.

Return value None

Description Basically, a timeout period is set at the Cao.AddController command execution. Use this command if you want to set a time-out period after Cao.AddController command execution.

Example

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                    "", "conn=eth:192.168.0.10")

'Set a time-out period to 1000 milliseconds.
caoCtrl.SetTimeout 1000
```

<ImplVar>.GetTimeout

Usage Obtain a currently assigned timeout period.

Syntax <ImplVar>.GetTimeout

Argument None

Return value <Time>

Currently assigned timeout period is returned with integer type data. Unit is millisecond.

Description Obtain a currently assigned time-out period.

Example

```
Dim caoCtrl as Object
Dim iTimeout as Integer

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                                "", "conn=eth:192.168.0.10")

'Obtain a timeout period.
iTimeout = caoCtrl.GetTimeout
```

<ImplVar>.GetCommandResult

Usage Wait for the completion of the asynchronous command to get the return value of it.

Syntax <ImplVar>.GetCommandResult

Argument None

Return value <Execution result of asynchronous command>

The return value of asynchronous command is stored.

Description Wait for the completion of the asynchronous command to get the return value of it.
If the executed asynchronous command which has not return value (e.g,; ChangeModeAsync) is executed, it returns nothing.
If any synchronous command is used before this command, "Get result error" (0x80100003) occurs and no value will be returned.
If an asynchronous command, which is the target of GetCommandResult command, ends with an error, the error will be ignored within the process of asynchronous command, and the error occurs at GetCommandResult command execution.
If there is no response within the specified timeout-period during the waiting time of the asynchronous command completion, a time-out error (0x80000900) will occur.
Note that if another command is executed after an asynchronous command, the execution result of the asynchronous command that you've just get will be deleted.

Example

```
Dim caoCtrl as Object
Dim vntResult as variant

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

' Set the lower limit on the edge tool of the tool No.100 to 123.456.
caoCtrl.ExecuteCommandAsync "DW,100,82,1,123.456"

'To obtain the return value of command, use GetCommandResult.
vntResult = caoCtrl.GetCommandResult
```

6. Error Code

As for how to check the provider errors, refer to Provider Errors in PROVIDER GUIDE on the DENSO ROBOT USER MANUALS.

In provider errors, an error issued by XG-X series will have original number ranging from 80108000 (hexadecimal) to 80108063 (hexadecimal), which lower two digits represents an error code sending from XG-X series. For example, when ChangeInspectSetting command execution, if you enter 5 in the SD card number, the original number of the robot controller's error will be 80108016 (hexadecimal). The lower two digits "16" (hexadecimal) is equal to "22" in decimal number. According to the explanation of UW command written in the XG-X Series User's Manual, the error code 22 stands for "The number or the range of the parameter is incorrect."

Error	Error number	Description
E_CVERROR_CVERR	0x80108000 to 0x80108063	XGX series original error
E_CVERROR_LENGTH	0x80100000	Packet length error
E_CVERROR_PACKET	0x80100001	Packet error
E_COMMAND_EXECUTING	0x80100002	Another command was executed during a command execution
E_GET_COMMAND_RESULT	0x80100003	GetCommandResult command was executed after a Synchronous command

7. Sample Program

```
Sub Main
        Dim caoCtrl As Object
        Dim strRet As String

        'XG-X series provider implementation
        caoCtrl = Cao.AddController("XGX", "CaoProv.KEYENCE.CVX", "", _
                                        "conn=eth:192.168.0.3, timeout=1000")

        'Input trigger in Trigger 1 and then obtain the result data.
        strRet = caoCtrl.TriggerAndGetResult(1)

        'Output the result data to the message output window on the teach pendant.
        PrintDbg strRet

        'Disconnect XG-X series provider and delete it.
        cao.Cotrollers.Remove caoCtrl.Index
        caoCtrl = Nothing
End Sub
```

Revision History

DENSO Robot
Provider
User’s Manual

KEYENCE Machine Vision System XG-X Series

Version	Supported RC8	Content
Ver.1.0.0	Ver.2.0.*	First edition
Ver.1.0.1	Ver.2.2.*	Addition of 34 commands
Ver.1.0.2	Ver.2.2.*	Modified the sample program.
Ver.1.0.3	Ver.2.3.*	Modified version.

DENSO WAVE INCORPORATED

- No part of this manual may be duplicated or reproduced without permission.
- The contents of this manual are subject to change without notice.
- Every effort has been made to ensure that the information in this manual is accurate. However, should any unclear point, error or omission be found, please contact us.
- Please note that we will not be responsible for any effects resulted from the use of this manual regardless of the above clauses.

DENSO Robotics

THIRD PARTY PRODUCTS

DENSO WAVE INCORPORATED