

DENSO Robotics  
THIRD PARTY PRODUCTS



## PROVIDER MANUAL

Maker

**KEYENCE**

Products/Series

**Image Processing System**

**MODEL: XG Series**



# Vision

# Introduction

This document is a user's manual for the provider to use "KEYENCE Image Processing System XG Series" connected to the DENSO robot controller RC8 series. Note that some functions may be unavailable on old XG models. For details and handling of the connected device, refer to the user's manual of "KEYENCE Image Processing System XG 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**

**KEYENCE XG-7000/8000 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.

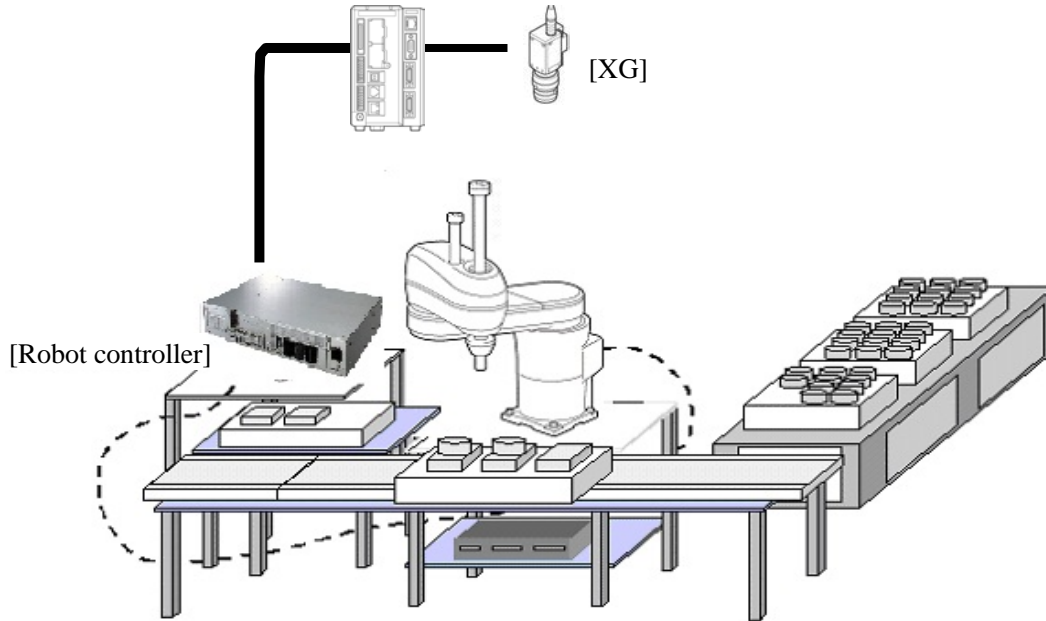
# Contents

Introduction.....	2
Important.....	2
Notice to Customers.....	2
1. Outline of This Product (Provider) .....	4
2. How to Connect .....	6
3. Communication Settings for Robot Controller and Device Used .....	7
4. Provider Execution Procedure .....	9
5. Command Description .....	10
6. Error Code of XG provider .....	25
7. Operation Panel Screen.....	26
8. Sample Program.....	27
Revision History .....	28

# 1. Outline of This Product (Provider)

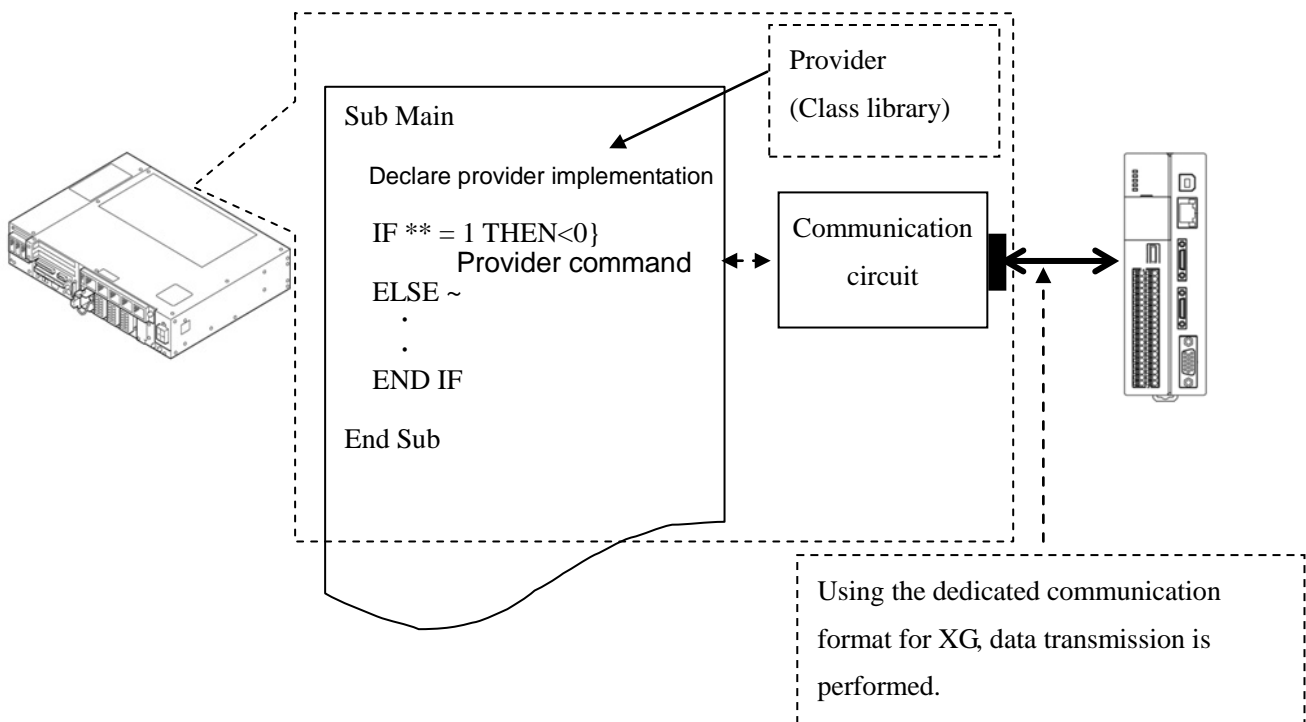
## 1.1 Target device of provider

This provider can be used only when a DENSO robot controller (RC8 series) is connected to the XG-7000/8000 series (hereinafter referred to as XG).



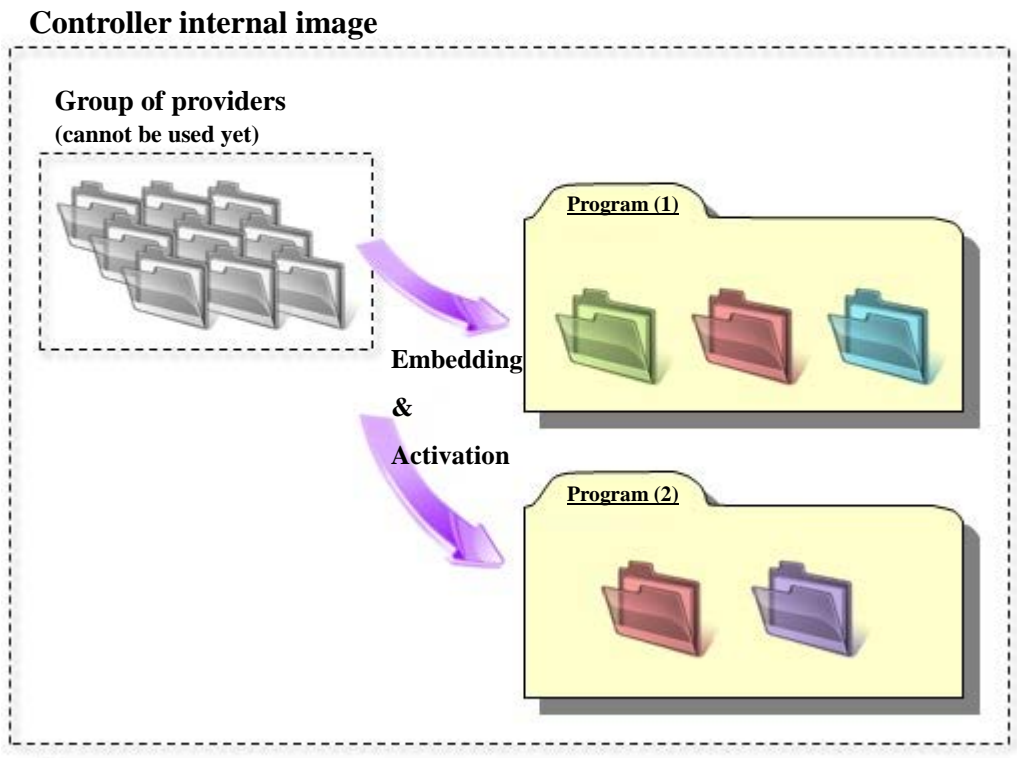
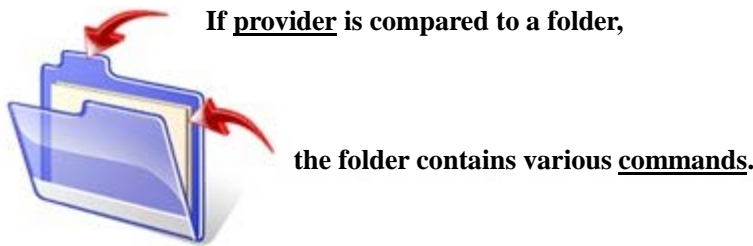
## 1.2 Features of provider


This provider is provided to use the XG native commands required to access XG 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 series. The following shows a diagram of provider embedding.





### 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.



 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).

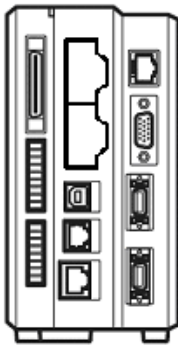
\* The provider is provided as a dynamic link library (abbreviated as DLL) which can be used from PacScript.

## 2. How to Connect

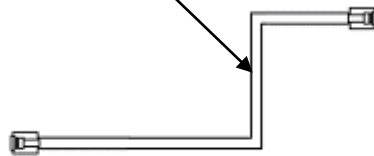
### 2.1 Ethernet (TCP/IP) connection example

To connect to the robot controller via Ethernet, use the optional dedicated cable (KEYENCE PN: OP-66843) or a crossover LAN cable. Also, when a switching hub/router is used, use the cable suitable for the switching hub/router specifications.

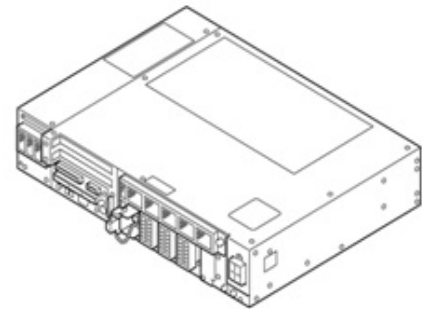
[XG series]



Crossover cable (3 m)  
OP-66843

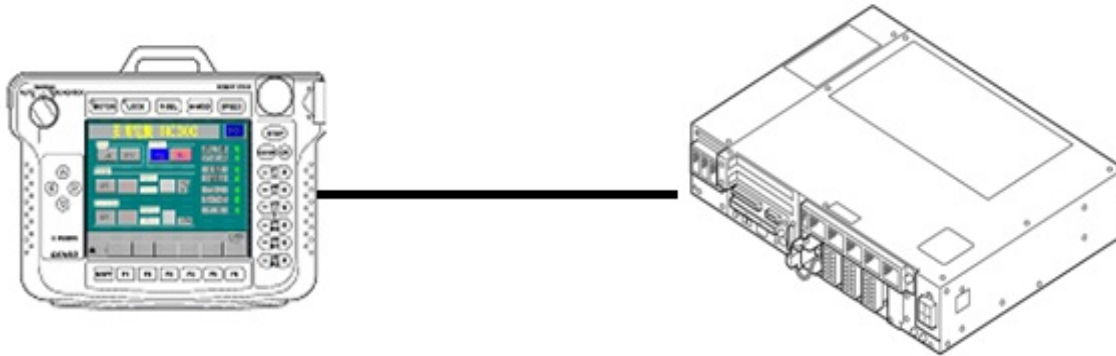


[Robot controller]



### 3. Communication Settings for Robot Controller and Device Used

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

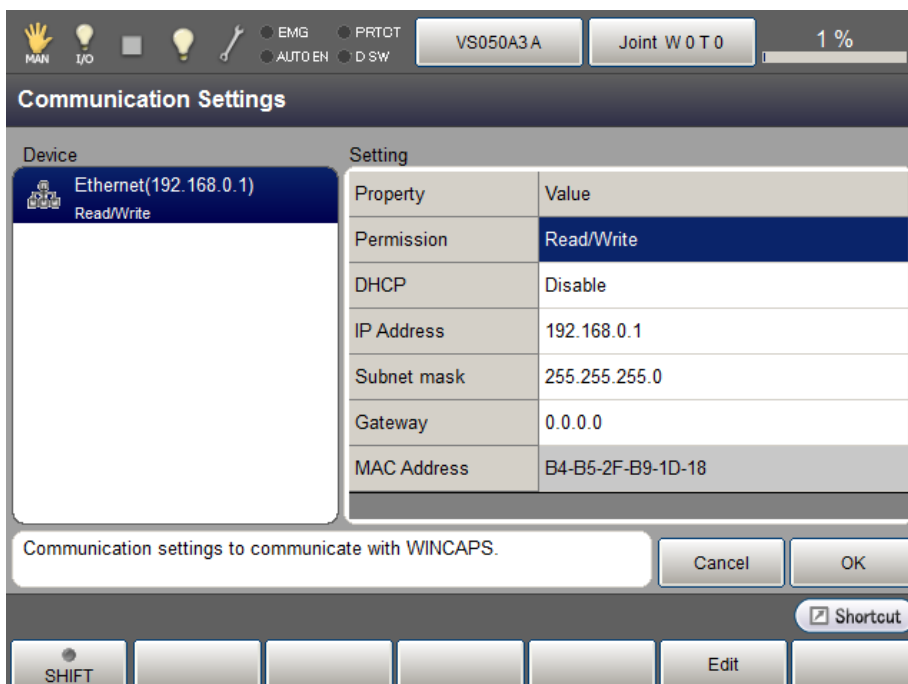


#### 3.1 Communication via Ethernet (TCP/IP)

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

Set the robot controller's IP address.

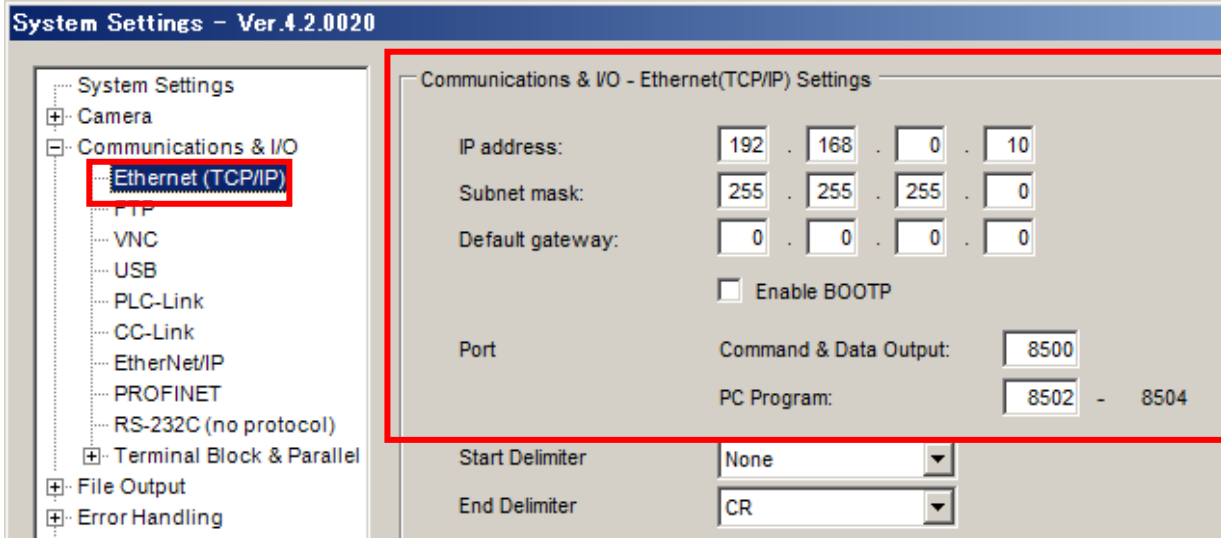
(1) Press [F6 Setting] - [F5 Communication and Token] - [F2 Network and Permission] to display the [Communication Settings] window. Set the IP address and subnet mask so that the robot controller and XG series t are within the same subnet mask.



### 3.1.2 Ethernet (TCP/IP) communication settings for XG series

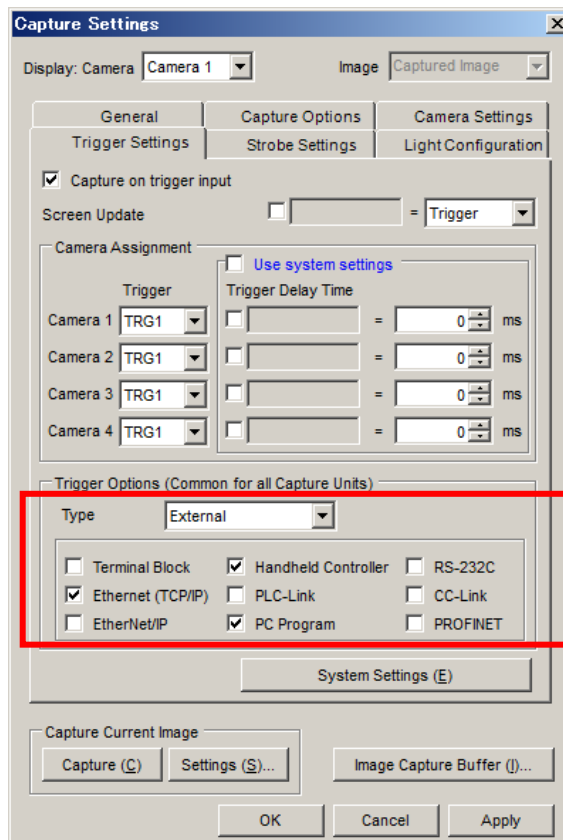
Select [Setting Menu] - [System Settings] on XG Vision Editor to display the [Communications & I/O - Ethernet(TCP/IP) Settings] window. Set the IP address and subnet mask so that the robot controller and XG series are within the same subnet mask. Set CR for delimiter.

\*For "Command & Data Output", set "8500" at all time. (Fixed value)



### 3.1.3 Other settings for XG series

(1) After creating a test flow with XG Vision Editor, select [Capture Settings] - [Trigger Settings] tab on [Flow View] to display the [Trigger Settings] window. Select "External" for the trigger type and check the "Ethernet (TCP/IP)" check box.





## 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("XG", "caoProv.KEYENCE.VWXG", "", "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. The commands are classified into connection commands, XG commands, and proprietary extension commands. For the detailed operation of XG commands, refer to the API list in the reference manual for V-Works for XG ActiveX Control for the KEYENCE XG-7000/8000 series.

**Table 5-1 List of commands**

command	Commands supported by KEYENCE	Usage	Refer to
Connection commands			
cao.AddController	—	Implements the provider to a variable and makes a connection to XG.	11
XG commands			
ChangeMode	ChangeMode	Changes the run/stop mode.	12
ReadMode	ReadMode	Reads the run/stop mode.	13
Reset	Reset	Causes a reset.	14
Restart	Restart	Jumps to the next unit of the start unit.	15
Trigger	Trigger	Issues a trigger.	16
EnableTrigger	EnableTrigger	Enables/disables trigger input.	17
ReadTriggerEnable	ReadTriggerEnable	Reads the trigger input enabled/disabled state.	18
WriteVariable	WriteVariable	Writes a variable value.	19
ReadVariable	ReadVariable	Reads a variable value.	20
ChangeInspectSetting	ChangeInspectSetting	Changes the test setting No.	21
ClearError	ClearError	Clears a system error.	22
Proprietary extension commands			
ExecuteCommand	—	Executes a non-procedural command.	23
TriggerAndGetResult	—	Issues a trigger and acquires the processing result of images.	24

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 XG.

**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.KEYENCE.VWXG"

<Provider running machine name> Omit this parameter.

<Option> [Connection parameter]

[Connection parameter] "conn=eth:<IP address>[:Port number]"

Default port number is 8502.

(The port number is optional.)

**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("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")
```

\* Specify a port number as follows:

```
caoCtrl=cao.AddController("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10:8503")
```

## <ImplVar>.ChangeMode

**Usage** Changes the operation mode to run or stop modes.

**Usage** **<ImplVar>.ChangeMode** <Mode>

Argument: <Mode> Switching between run and stop modes (integer)  
0: Stop mode  
1: Run mode

**Usage** The operation mode is changed to run or stop.

### Example

```
Dim caoCtrl as Object
```

```
caoCtrl=cao.AddController("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")  
caoCtrl.ChangeMode 1 'Enter run mode
```

## <ImplVar>.ReadMode

**Usage** Acquires the current operation mode (run, stop, or remote capture mode).

**Syntax** **<ImplVar>.ReadMode**

Return value: The current operation mode is stored. If acquisition fails, -1 is stored.

(Variant type)

0: Stop mode

1: Run mode

2: Remote capture mode

**Description** The current operation mode (run, stop, or remote capture mode) is acquired.

### Example

```
Dim caoCtrl as Object
```

```
Dim vntResult as Variant
```

```
caoCtrl=cao.AddController("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")
```

```
vntResult = caoCtrl.ReadMode
```

## <ImplVar>.Reset

**Usage**            Resets the controller.

**Syntax**            <ImplVar>.Reset

**Description**    The controller is reset.

### Example

```
Dim caoCtrl as Object
```

```
caoCtrl=cao.AddController("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")  
caoCtrl.Reset
```

## <ImplVar>.Restart

**Usage** Jumps to the next unit of the start unit.

**Syntax** <ImplVar>.Restart

**Description** The command makes a jump to the next unit of the start unit.

### Example

```
Dim caoCtrl as Object
```

```
caoCtrl=cao.AddController("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")  
caoCtrl.Restart
```

## <ImplVar>.Trigger

**Usage** Issues a trigger.

**Syntax** **<ImplVar>.Trigger** <Trigger No.>

Argument: <Trigger No.> Specify the number of trigger to issue (integer).

1 – 8: Trigger 1 to 8

-1: All triggers

However, Triggers 5 to 8 are supported in Version 5.0.0000 or later.

**Description** A trigger is issued.

### Example

```
Dim caoCtrl as Object
```

```
caoCtrl=cao.AddController("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")  
caoCtrl.Trigger 2
```



## <ImplVar>.EnableTrigger

**Usage** Enables/disables trigger input.

**Syntax** **<ImplVar>.EnableTrigger** <Enable mode>

Argument: <Enable mode> Specify whether to enable/disable trigger (integer).  
0: Trigger disabled  
1: Trigger enabled

**Description** Trigger is enabled/disabled.

### Example

```
Dim caoCtrl as Object
```

```
caoCtrl=cao.AddController("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")  
caoCtrl.EnableTrigger 1
```

## <ImplVar>.ReadTriggerEnable

**Usage** Acquires the current trigger status (enabled/disabled).

**Syntax** **<ImplVar>.ReadTriggerEnable**

Return value: The trigger status is stored. If acquisition fails, -1 is stored. (Variant type)  
0: Trigger disabled  
1: Trigger enabled

**Description** The trigger status (enabled/disabled) is acquired.

### Example

```
Dim caoCtrl as Object
Dim vntResult as Variant

caoCtrl=cao.AddController("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")
vntResult = caoCtrl.ReadTriggerEnable
```

## <ImplVar>.WriteVariable

**Usage** Writes a value to a specified scalar variable (global or local variable).

**Syntax** **<ImplVar>.WriteVariable** <Variable name>, <Value>  
, <Synchronization mode>

Argument: <Variable name> Specify a scalar variable name with one-byte characters (character string).

<Value> Specify a value to write to the variable (Double type).

<Synchronization mode> Specify whether or not to reflect in synchronization with a flow (integer).  
0: Reflected immediately without synchronization with flow (MW command)  
1: Reflected at the end unit of flow (MS command)

**Description** A value is written to a specified scalar variable.  
(This variable name is a name configured in the XG series.)

### Example

```
Dim caoCtrl as Object
```

```
caoCtrl=cao.AddController("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")
caoCtrl.WriteVariable "#MyVar", 2, 0
```

## <ImplVar>.ReadVariable

**Usage** Acquires the value of specified scalar variable.

**Syntax** **<ImplVar>.ReadVariable( <Variable name> )**

Argument: <Variable name> Specify a scalar variable name with one-byte characters (character string).

Return value: The variable value is stored. If acquisition fails, -1.0 is stored. (Variant type)

**Description** The value of specified scalar variable is acquired.  
(This variable name is a name configured in the XG series.)

### Example

```
Dim caoCtrl as Object
Dim vntResult as Variant

caoCtrl=cao.AddController("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")
vntResult = caoCtrl.ReadVariable("#MyVar")
```

## <ImplVar>.ChangeInspectSetting

**Usage** Changes the setting to the test setting No. of the specified SD card.

**Syntax** **<ImplVar>.ChangeInspectSetting** <SD card No.>, <Test setting No.>

Argument: <SD card No.> Specify the SD card No. (Integer 1, 2).

<Test setting No.> Specify the test setting No. (Integer 0 to 999)

**Description** The setting is changed to the test setting No. of the specified SD card.

### Example

```
Dim caoCtrl as Object
```

```
caoCtrl=cao.AddController("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")  
caoCtrl.ChangeInspectSetting 1, 2 'Change SD card 1 test setting to No.2
```

## <ImplVar>.ClearError

**Usage** Clears the error status and error code of the specified type.

**Syntax** **<ImplVar>.ClearError** <Error type>

Argument: <Error type> Specify the error status (Integer 0, 1).  
0: %Error0 and %Error0Code cleared  
1: %Error1 and %Error1Code cleared

**Description** The error status and error code of the specified type are cleared.

### Example

```
Dim caoCtrl as Object
```

```
caoCtrl=cao.AddController("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")  
caoCtrl.ClearError 0
```

## <ImplVar>.ExecuteCommand

**Usage** Executes a specified non-procedural command.  
A command response is acquired regardless of whether the command execution is successful or not.

**Syntax** **<ImplVar>.ExecuteCommand** (<Non-procedural command>)

Argument: <Non-procedural command> Specify a command with a character string.

Return value: Command response is returned with a character string. If acquisition fails, the character string is not stored. (Variant type)

**Description** For the supported non-procedural commands, refer to the reference manual of V-Works for XG ActiveX control for XG series.

### Example

Non-procedural command R0: The following shows an example of entering run mode.

```
Dim caoCtrl as Object
Dim vntResult as Variant

caoCtrl=cao.AddController("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")
vntResult = caoCtrl.ExecuteCommand("R0")           'Enter run mode
```

## <ImplVar>.TriggerAndGetResult

**Usage** Issues a specified trigger and acquires the processing result of images.

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

Argument: <Trigger No.> Specify the number of trigger to issue (integer).

1 – 8: Trigger 1 to 8

-1: All triggers

However, Triggers. 5 to 8 are supported in Version 5.0.0000 or later.

Return value: Output data specified for the result output unit is stored. If acquisition fails, -1.0 is stored. (Variant type)

**Description** Issues a specified trigger and acquires the processing result specified for the output unit of the XG series-side.

### Example

```
Dim caoCtrl as Object
```

```
Dim vntResult as Variant
```

```
caoCtrl=cao.AddController("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")
```

```
vntResult = caoCtrl.TriggerAndGetResult( -1 )
```



## 6. Error Code of XG provider

The specific error code of XG provider is created as shown below, based on the return value of XG.

0x80100000 + Return value

For the error code of each command, refer to ActiveX control reference manual of Keyence.

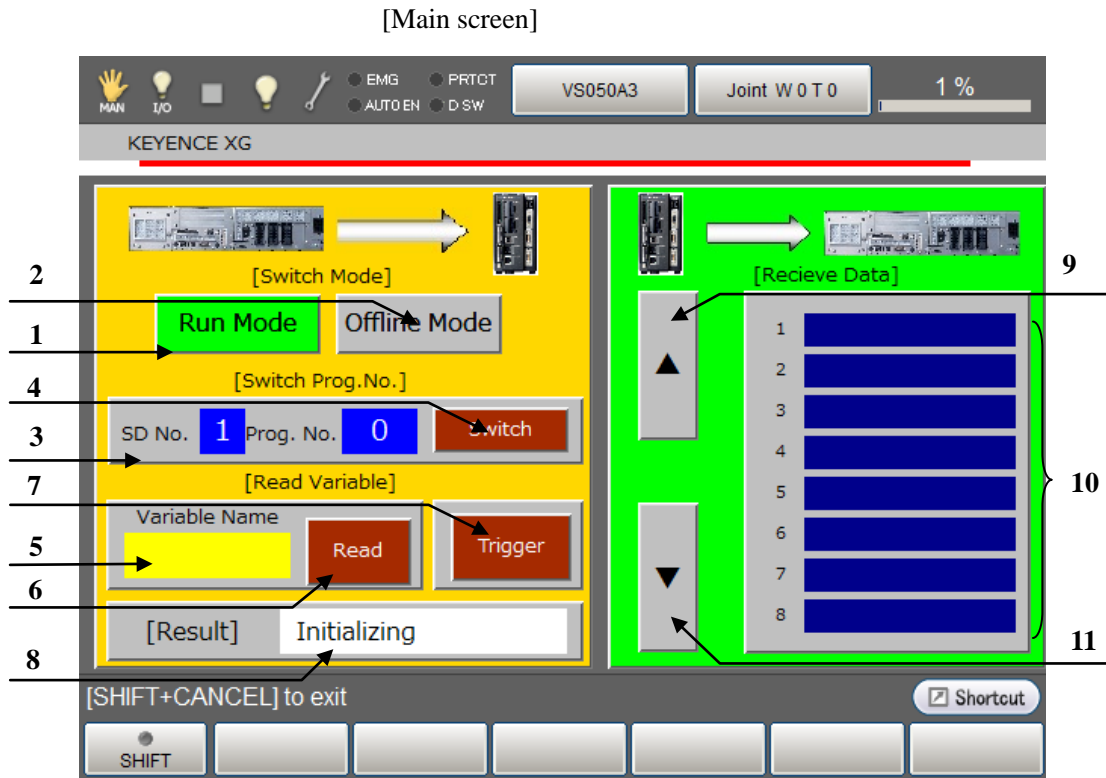
Example: When executing ChangeMode.

0x801003EA: Parameter value is out of range.

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 XG (implements the provider). The communication settings need to be configured beforehand. Closing the operation panel terminates the connection (releases the provider).



**Description** Each button functions as follows.

1. Changes to the "run mode". (ChangeMode)
2. Changes to the "stop mode". (ChangeMode)
3. Configures the test setting No. SD card No.: 1 to 2, setting No.: 0 to 999
4. Changes to the setting No. set in the step 3. (ChangeInspectSetting)
5. Configures the variable name to read. (This variable name is a name configured in the XG series.)
6. Reads out the value of the variable set in (5). Received data appears in the data display section (10). (ReadVariable)
7. Executes all triggers. (Trigger)
8. Displays the processing result.
9. Moves up the page displayed for received data.
10. Displays the received data.
11. Moves down the page displayed for received data.

Note 1: When a provider implementation (initialization) is done properly, "Connected" is displayed in the field 8.

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

## 8. Sample Program

Sub Main

On Error Goto ErrProc	'Declare error process routine
Dim caoXG as Object	'Declare provider variable
Dim vntResult as Variant	'Declare character-string variable
Dim pTargetPos as Position	'Declare P-type variable
takearm keep = 0	
pTargetPos = P11	
caoXG = cao.AddController("XG", "CaoProv.KEYENCE.VWXG", "", "Conn=eth:192.168.0.10")	
	'Provider implementation
caoXG.ChangeInspectSetting 1, 2	'Change to SD1 setting 2
caoXG.Trigger -1	'Trigger
delay 200	
vntResult = caoXG.ReadVariable("#XPos")	'Receive data
letx pTargetPos = posx(P11) + val(vntResult)	'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"	
Goto EndProc	

End Sub

## Revision History

---

DENSO Robot  
Provider  
User's Manual  
KEYENCE Image Processing System XG 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 command "TriggerAndGetResult"

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