

# DENSO

DENSO Robotics

## THIRD PARTY PRODUCTS



# PROVIDER MANUAL

---

Maker

**Sharp Manufacturing Systems Corporation**

---

Products/Series **Image Sensor Cameras**

**MODEL: IV Series**

(IV-S150X / IV-S150M / IV-S200X / IV-S210X /  
IV-C250X/IV-S300X/IV-S310X/IV-S301M)

---



# Vision

# Introduction

This document is a user's manual for the provider to use "Sharp Manufacturing System Corporation: Image Sensor Camera IV Series" connected to the DENSO robot controller RC8 series. Note that some functions may be unavailable on the previous version of IV series. For details and handling of the connected device, refer to the user's manual of "Sharp Manufacturing System Corporation: Image Sensor Camera IV 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**

**Sharp Manufacturing Systems Corporation      IV Series**  
**Target models: IV-S150X / IV-S150M / IV-S200X / IV-S210X /**  
**IV-C250X / IV-S300X / IV-S310X / IV-S301M**

---

## 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. Before using this product, be sure to visit our website and read "Software License Agreement" on the product download page.

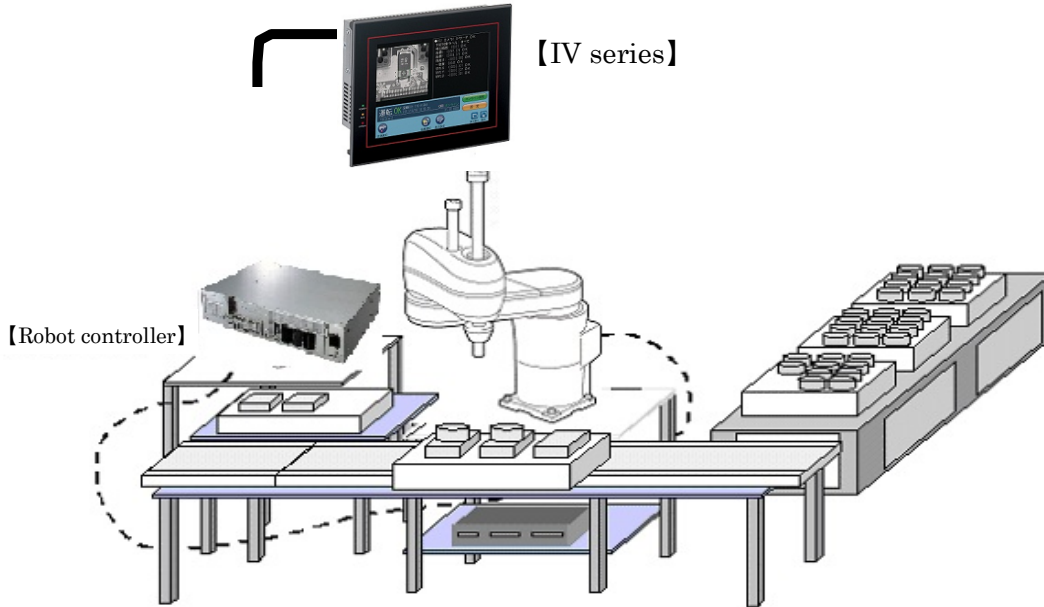
# Contents

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 .....	8
4. Provider Execution Procedure .....	13
5. Command Description .....	14
6. Error codes.....	74
7. Operation Panel Screen .....	75
8. Sample Program.....	76

# 1. Outline of This Product (Provider)

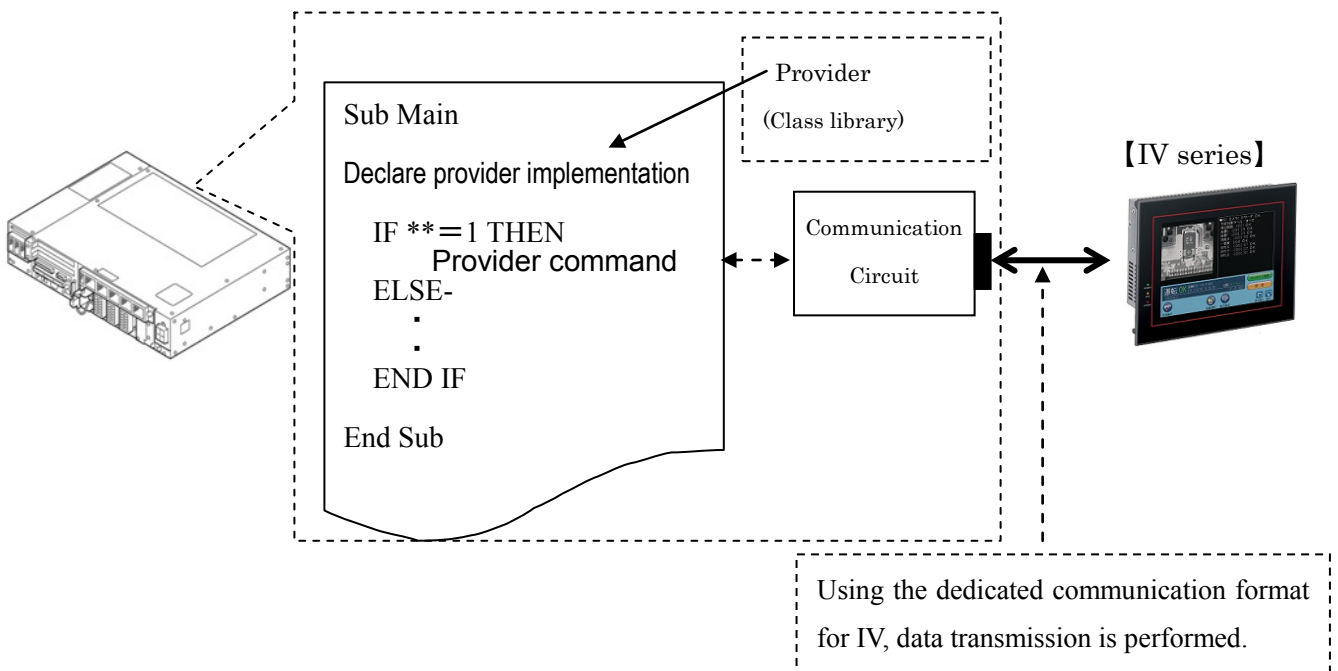
## 1.1 Target device of provider

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



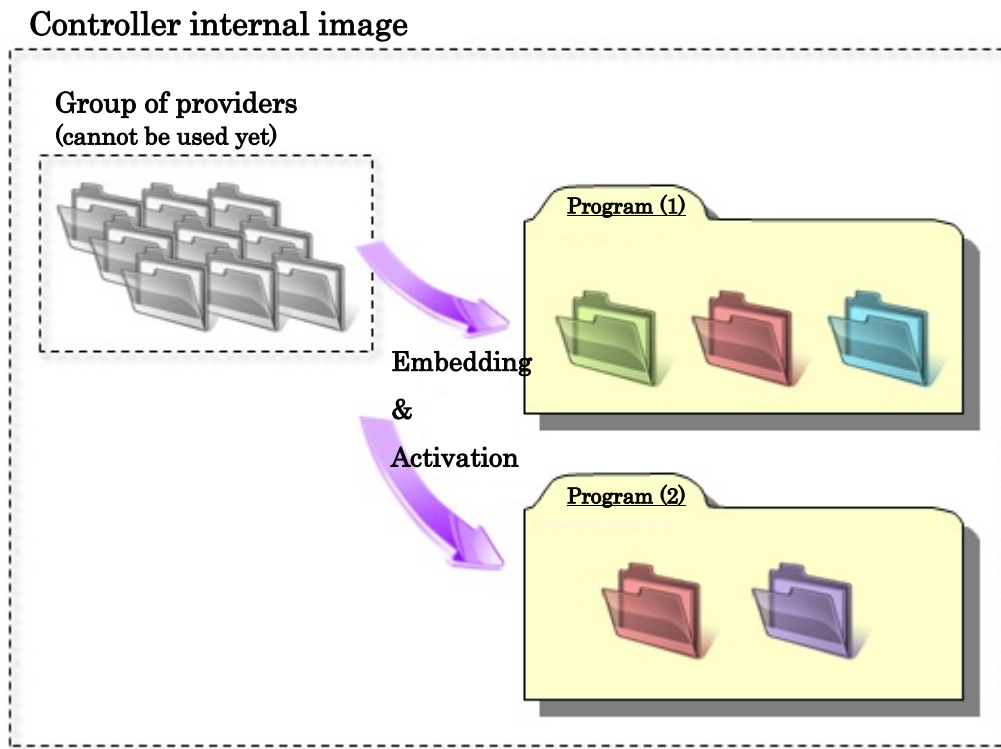
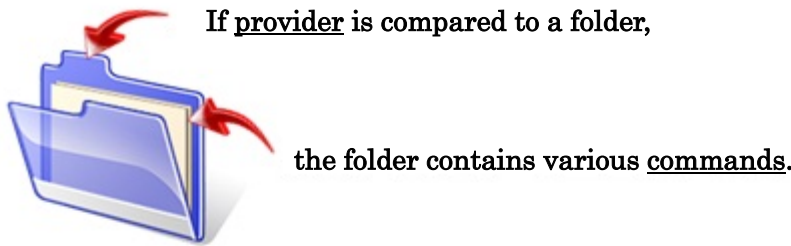
## 1.2 Features of provider

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

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 RS-232C connection example

The IV series and the robot controller need to be connected with a communication cable.  
 To connect the IV series to the robot controller through RS232-C, use an RS232C cross cable as shown below.

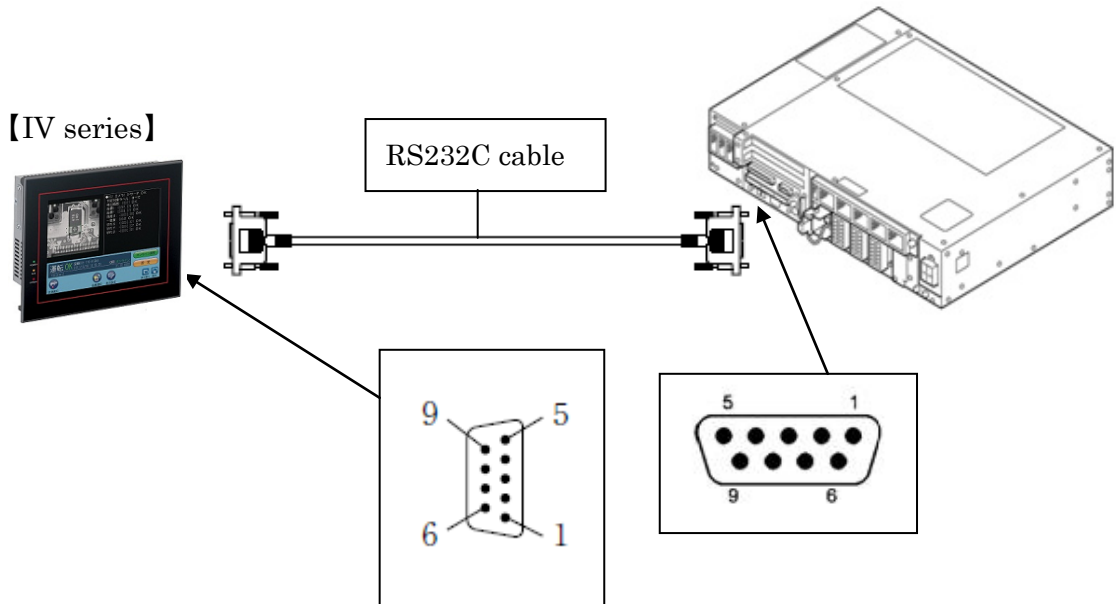


Figure 2-1 RS232C connection diagram

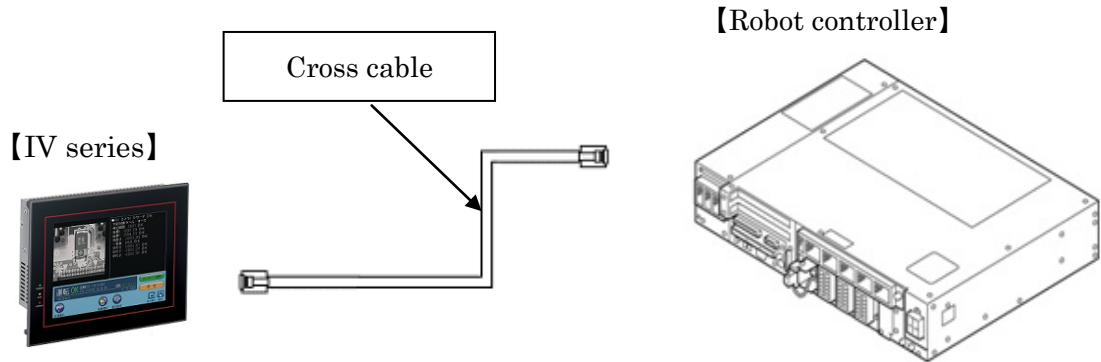
IV-side RS232C connector  
 (D-Sub 9 pin Female)

RC8-side RS232C connector  
 (D-Sub 9 pin Male)

Pin number	Signal name		Pin number	Signal name
Connector case	FG		Connector case	FG
2	RD		2	RXD
3	SD		3	TXD
5	SG		5	SG

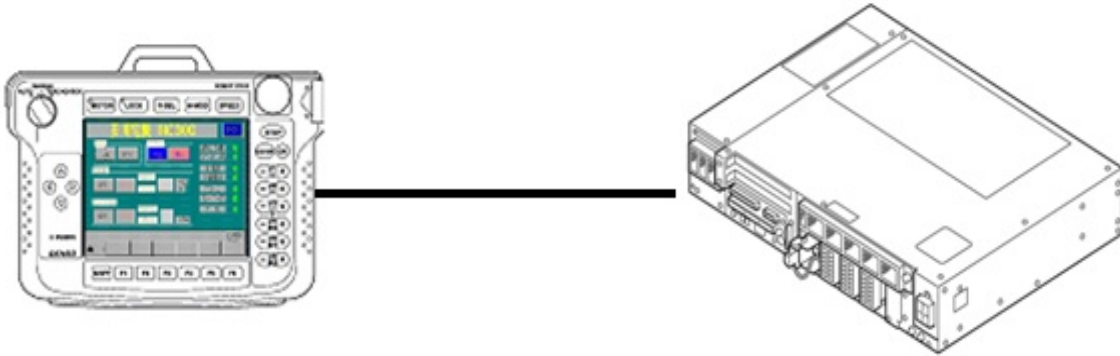
## 2.2 Ethernet (TCP/IP) connection example

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



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

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



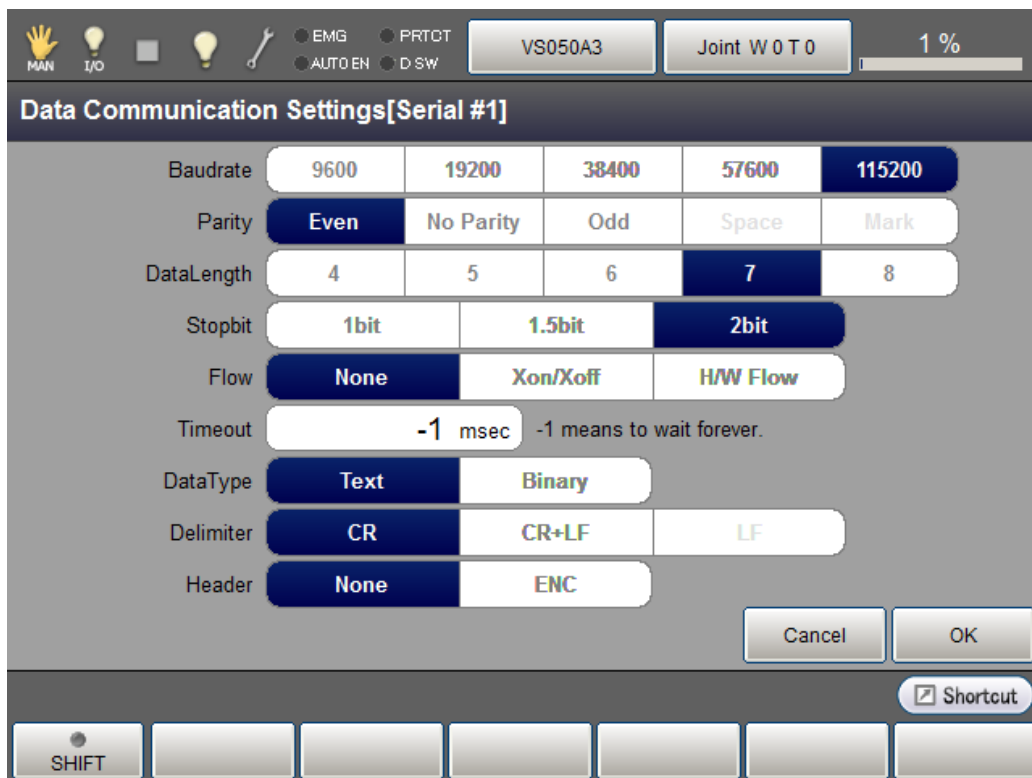
#### 3.1 Communication through RS-232C

##### 3.1.1 RS-232C communication settings on robot controller

Configure communication speed and other settings of each port on RS-232C serial interface.

Press [F6 Setting]-[F5 Communication and Token]-[F3 Data Communication]. Select desired RS-232C from [Device], and then press [F6 Edit] to display [Data Communication Settings] window. Select an item to set, and then change the value. Set “CR” for the delimiter.

In the default setting of the robot controller, the line number 1 (COM2) is set for RS232C port.





### 3.1.2 RS-232C communication settings on the IV series

Configure communication speed and other settings of each port on RS-232C serial interface.

In the IV series, open [System settings], and then select [Serial settings] to display the setting window.

Set [RS-232C settings] for Serial settings, and set [General] for communication mode.

#### 【Display of the IV】

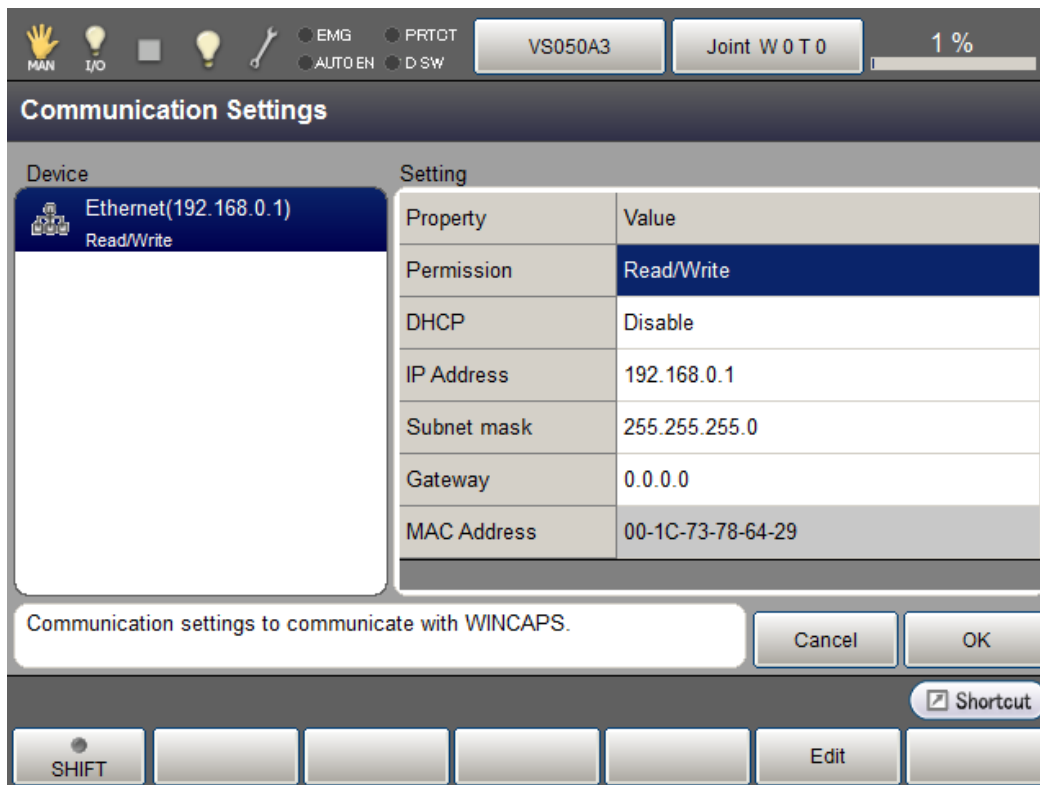
RS-232C settings	
Communication mode	General ▼
Baud rate	115200bps ▼
Data length	7 bits ▼
Parity	Even ▼
Stop bit	2 bits ▼
Terminal number	000

### 3.2 Communication through Ethernet (TCP/IP)

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

Set the robot controller's IP address.

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



### 3.2.2 Ethernet (TCP/IP) communication settings for IV

Set an IP address of the IV series

Open [System settings], and then select [Ethernet settings] to display the setting window.

Set the IP address and subnet mask so that the robot controller and IV series are within the same subnet mask.

Set [General] for communication mode.

#### 【Display of IV】

Ethernet settings							
Communication mode	General ▼						
/ IP / Port / Other \							
IP auto acquisition	Invalid ▼						
IP address	192	.	168	.	000	.	002
Subnet mask	255	.	255	.	255	.	000
Default gateway	000	.	000	.	000	.	000
PLC link address	192	.	168	.	001	.	021

Ethernet settings	
Communication mode	General ▼
/ IP / Port / Other \	
Online port	02000
Command port	02001
Data collector port	02002
Broadcast port	02010
PLC link port	05000

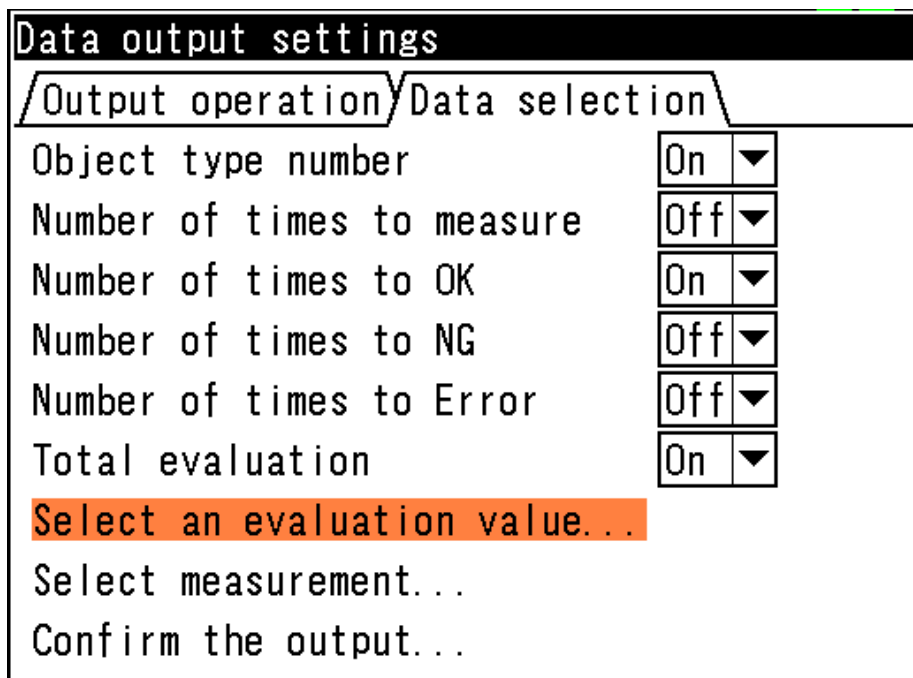
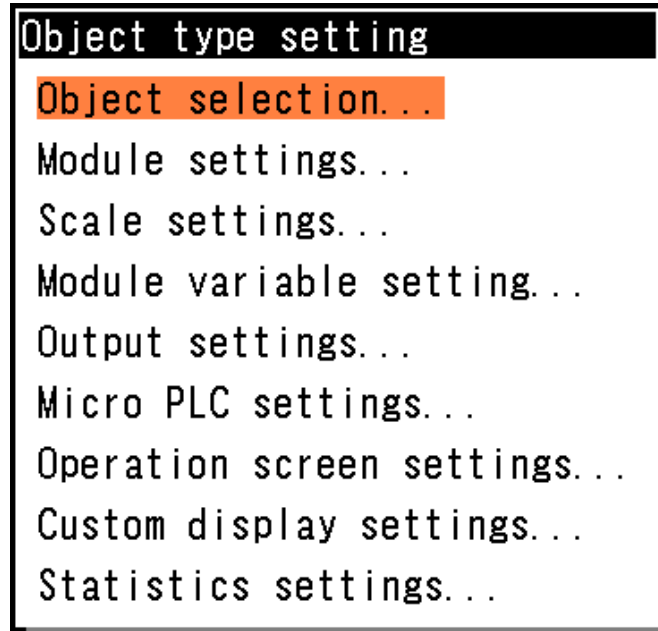
### 3.3 Output settings for IV

In [Process flow], open [Output], and then select [Numeric Data] to display the setting window.

In [Data output], select [Data selection] to display Data output settings.

Select the data that you intend to output to the RC8 controller.

#### 【Display of the IV】



## 4. Provider Execution Procedure

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

Sub Main

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

```
caoCtrl = cao.AddController("IV_S150X", "caoProv.SHARP.IV", "", "conn=eth:192.168.0.2:2001") 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 number. 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) The program can be stated using the provider command hereafter.

## 5. Command Description

This section contains descriptions of commands. The commands are classified into connection commands, IV commands, and proprietary extension commands. For the detailed operation of IV commands, refer to the reference manuals for IV series issued by Sharp Manufacturing System Corporation.

**Table 5-1 Command list**

Command	IV command	Usage	Target models					
			IV-S150X		IV-S2*0X		IV-S3*0X	
			IV-S150M		IV-C250X			
			com	eth	com	eth	com	eth
Connection command								
<a href="#">cao.AddController</a>	—	Implements the provider to a variable and establish a connection	○	○	○	○	○	○
IV commands								
<a href="#">Trigger And wait</a>	T00	Inputs the trigger and obtain the result	○	○	○	○	○	○
<a href="#">Trigger</a>	T01	Inputs the trigger	○	○	○	○	○	○
<a href="#">GetData</a>	T01	Obtains the image processing result	○	○	○	○	○	○
<a href="#">RobotCalibration</a>	T10	Performs robot calibration					○	○
<a href="#">SerialEnable</a>	A00,A01	Sets the serial communication to Enable/Disable			○			
<a href="#">SAlignmentTrigger</a>	A00	Specifies the current values of each axis					○	○
<a href="#">SAlignmentCalibration</a>	A01	Provides various settings about calibration execution					○	○
<a href="#">RemoteEnable</a>	A10,A11	Sets the remote key input setting to Enable/Disable			○			
<a href="#">ViewLockEnable</a>	A20,A21	Switches the operation screen lock setting to Enable/Disable			○			
<a href="#">GetKind</a>	C00	Reads the object type number.	○	○	○	○	○	○
<a href="#">PutKind</a>	C01	Writes the object type number.	○	○	○	○	○	○
<a href="#">GetModule</a>	C10	Reads the module number			○	○		
<a href="#">PutModule</a>	C11	Writes the module number			○	○		
<a href="#">GetViewMode</a>	C20	Reads the image mode	○	○	○		○	○
<a href="#">PutViewMode</a>	C21	Writes the image mode	○	○	○		○	○
<a href="#">GetDispMode</a>	C30	Reads the camera display mode	○	○	○		○	○
<a href="#">PutDispMode</a>	C31	Writes the camera display mode	○	○	○		○	○
<a href="#">GetMeasureData</a>	C40	Reads the coordinates for the manual measurement			○			
<a href="#">ClearFigures</a>	C40	Clears the statistics					○	○

<a href="#">PutMeasureData</a>	C41	Writes the coordinates for the manual measurement			○			
<a href="#">PutPassword</a>	C60	Writes the operation window lock password			○			
<a href="#">GetVal</a>	C80	Reads the variable value			○	○	○	○
<a href="#">PutVal</a>	C81	Writes the variable value			○	○	○	○
<a href="#">RegStdImage</a>	R00	For IV-S2*0X: Overwrites the reference image For IV-S3*0X: Saves (to a non-volatile memory) the last imported camera image as a reference image			○		○	○
<a href="#">GetShutterSp</a>	R10	Reads the shutter speed			○			
<a href="#">PutShutterSp</a>	R11	Writes the shutter speed			○			
<a href="#">GetThreshold</a>	R30	Reads the threshold value setting			○			
<a href="#">PutThreshold</a>	R31	Writes the threshold value setting			○			
<a href="#">GetGain</a>	R40	Reads the gain and offset settings			○			
<a href="#">PutGain</a>	R41	Writes the gain and offset settings			○			
<a href="#">GetDate</a>	R50	Reads the date and time setting	○	○	○		○	○
<a href="#">PutDate</a>	R51	Writes the date and time setting	○	○	○		○	○
<a href="#">GetRegData</a>	R80	Reads the code reader register data					○	○
<a href="#">PutRegData</a>	R81	Writes the code reader register data					○	○
<a href="#">PutDateString</a>	R89	Writes the set character string (8 date blocks (batch), with camera specification)					○	○
<a href="#">GetModuleString</a>	R90	Reads the set character string (module specification)					○	○
<a href="#">GetBlockString</a>	R92	Reads the set character string (block specification)					○	○
<a href="#">PutBlockString</a>	R93,R94	Writes the set character string (block specification)					○	○
<a href="#">Put20BlockString</a>	R96	Writes the set character string (20 variable blocks (batch), with delete space at the end)					○	○
<a href="#">Put10BlockString</a>	R98	Writes the set character string (10 variable blocks (batch), with camera specification, with delete space at the end)					○	○
<a href="#">SnapShot</a>	I01	Stores snapshot image on an external memory	○	○	○	○	○	○
<a href="#">ImageClear</a>	I20	Clears the image memory			○	○		
<a href="#">GetKindState</a>	P10	Reads the object type number information				○		
<a href="#">GetStdImageState</a>	P20	Reads the reference image number information				○		
<a href="#">GetVersion</a>	D00	Reads the version information			○	○		
<a href="#">AllReset</a>	D10	Initializes the system settings, and object type settings			○	○		
<a href="#">SettingSave</a>	D11	Saves the system settings, and object type settings	○	○	○	○	○	○
<a href="#">Reset</a>	D12	Resets the provider			○			

## THIRD PARTY PRODUCTS

<a href="#">GetBrightness</a>	D20	Reads the average density	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>		<input type="radio"/>	<input type="radio"/>
<a href="#">GetParallel</a>	D21	Reads parallel input/output	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>		<input type="radio"/>	<input type="radio"/>
<a href="#">SelfCheck</a>	D40	Performs self-check tests					<input type="radio"/>	<input type="radio"/>
Property extension commands								
<a href="#">Raw</a>	—	Sends or receives raw data.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>		
<a href="#">ChangeTimeout</a>	—	Changes communication timeout period						



# cao.AddController

**Usage** Implements the provider to a variable and then to connect to the IV.

**Syntax** `cao.AddController(<Controller name>,<Provider name>,  
<Provider running machine name>, <Option> )`

Argument :

<Controller name> Assign an arbitrary name (The name is used for control)

<Provider name> "CaoProv.SHARP.IV"

<Provider running machine name> Omit this parameter

<Option> [Connection parameter], [Timeout period], [Model name], [Area number],[Check Sum]

[Connection parameter] Specify communication parameters for Ethernet or RS232C

• For Ethernet

"conn=eth:<<IP address>>:<<port number>>"

<< IP address>> IP address of the IV

<<Port number>> Port number of the IV (This can be omitted.)

Default: 2001

• For RS232C

"conn=com:<<com port number>> : <<communication speed>>:<<parity>>:<<data bit>>:<<stop bit>>"

<<Com port number>> Specify a port number of RS232C

'1'-COM1, '2'-COM2

<<Communication speed>> Set communication speed of RS232C

(This can be omitted.)

2400,4800,9600,19200,38400,

57600,115200

Default: 115200

<<Parity>> Set communication speed of RS232C

(This can be omitted.)

'N'-NONE, 'E'-EVEN, 'O'-ODD

Default: 'E'-EVEN

<<Number of data bits>> Specify number of data bits used for RS232C

(This can be omitted.)

7'-7bit, '8'-8bit

Default: '7'-7bit

<<Number of stop bit>> Specify number of stop bits used for RS232C

(This can be omitted.)

'1'-1bit, '2'-2bit

Default: '2'-2bit

[Timeout period] Specify timeout period (msec) for data transmission and reception.

"Timeout=<<timeout period>> (This can be omitted)"

Default: 500

[Model name] Specify a model name.

"Type=<<Model name>>" (This can be omitted.)

0 : IV-S150X/M(default)  
 1 : IV-S200X, IV-S210X, IV-C250X  
 2 : IV-S300X, IV-S310X

[Area number] Specify an area number at communication (0-255)  
 "AreaNo=<<Area number>>" (This can be omitted.)  
 Default: 0

[Check Sum] Specify the CheckSum availability at communication  
 "Chksum=<<Check Sum>>" (This can be omitted.)  
 0: Disable (default)  
 1: Enable

**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**".)  
 The communication is established at the same time when this goes into effective. The details of communication are set by <Option>.

#### Example

Dim caoCtrl as Object

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")
```

\* Specify a model name and other items.

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", _  

  "Conn=eth:192.168.0.2:2001, Type=0, AreaNo=0, Chksum=0")
```

\* For RS232C communication

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", _  

  "Conn=com:2, Type=0, AreaNo=0, Chksum=0")
```

\* When communication parameter is set to other than the default value.

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", _  

  "Conn=com:2:38400:N:8:2, Type=0, AreaNo=0, Chksum=0")
```

## <ImplVar>.TriggerAndWait

**Usage** Sets a trigger and then obtains the output result.

**Syntax** **<ImplVar>.TriggerAndWait( <Trigger number> )**

Argument : <Trigger number> Specify a trigger number (0-1)

Return value : Data output by the IV

**Description** A trigger is set on the IV, and then numerical data that has been specified by the output settings is obtained. (See 3.3 Output settings for IV.) If output data is not specified, the command waits until the timeout period passes. If multiple items of data are received, the result is stored as in an array.

### Example

```
Dim caoCtrl as Object  
Dim Res As Variant
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
Res = caoCtrl.TriggerAndWait(0)
```

## <ImplVar>.Trigger

**Usage** Sets a trigger.

**Syntax** **<ImplVar>.Trigger** <Trigger number>

Argument : <Trigger number> Specify a trigger number (0-1)

Return value : None

**Description** This command is used to set a trigger.

### Example

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
caoCtrl.Trigger 0
```

## <ImplVar>.GetData

**Usage**      Obtains output data.

**Syntax**      **<ImplVar>.GetData( <Trigger number> )**

Argument :    <Trigger number> Specify a trigger number (0-1)

Return value: Output data from the IV.

**Description**      This command is used to obtain numerical data that is specified by the output settings. (See 3.3 Output settings for IV.) If output data is not specified, the command waits until the timeout period passes. If multiple items of data are received, the result is stored as in an array.

### Example

```
Dim caoCtrl as Object
Dim Res As Variant

caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")
caoCtrl.Trigger 0
Res = caoCtrl.GetData(0)
```

## <ImplVar>.RobotCalibration

**Usage**      Performs the robot calibration.

**Syntax**      **<ImplVar>.RobotCalibration** ( <Trigger number>, <X-coordinate>, <Y-coordinate>,<R-coordinate> )

Argument :

- <Trigger number> Specify a trigger number (0 or 1)
- <X-coordinate> Absolute coordinate value on X-coordinate system of the current robot position (mm).
- <Y-coordinate> Absolute coordinate value on Y-coordinate system of the current robot position (mm).
- <R-coordinate> Absolute coordinate value on R-coordinate system of the current robot position (deg).

Return value:

- <Complete flag> This flag returns the execution status of this command.  
True: Finish  
False: Continue
- <X-coordinate> Absolute coordinate value on X-coordinate system of the next robot position (mm).
- <Y-coordinate> Absolute coordinate value on Y-coordinate system of the next robot position (mm).
- <R-coordinate> Absolute coordinate value on R-coordinate system of the next robot position (deg).

**Description**      This command performs robot calibration.  
Absolute coordinates of each axis are rounded to thousandths.  
The decimal part will be three digits.

Example

Refer to "[8.2 Calibration](#)".

## <ImplVar>.SerialEnable

**Usage** Switches the serial communication setting to Enable/Disable.

**Syntax** **<ImplVar>.SerialEnable** <Enable/Disable>

Argument : <Enable/Disable> Specify Enable/Disable

True: Enable

False: Disable

Return value : None

**Description** This command switches the serial communication setting to enable/disable.

### Example

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
caoCtrl.SerialEnable True
```

## <ImplVar>.SAlignmentTrigger

**Usage** Specifies the current values of each axis.

**Syntax** **<ImplVar>.SAlignmentTrigger** ( <Trigger number>, <X-coordinate>, <Y-coordinate>, < $\theta$ -coordinate> )

Argument : <Trigger number> Specify a trigger number (0 or 1)  
 <X-coordinate> Current value of X-axis  
 <Y-coordinate> Current value of Y(Y1)-axis  
 < $\theta$ -coordinate> Current value of  $\theta$ (Y2)-axis

Return value: Output data from IV

**Description** This command sets the current values of each axis.

### Example

```
Dim caoCtrl as Object
Dim Res As Variant
```

```
caoCtrl = cao.AddController("IV_S300", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")
Res = caoCtrl.SAlignmentTrigger(0, 134.58, 25.48, 105.6)
```



## <ImplVar>.SAlignmentCalibration

**Usage** Performs calibration execution-related settings.

**Syntax** **<ImplVar>.SAlignmentCalibration** ( <Trigger number>,  
 <X-coordinate>,<Y-coordinate>  
 <θ-coordinate> , <Start flag>)

Argument : <Trigger number> Specify a trigger number (0 or 1)  
 <X-coordinate> Current value of X-axis  
 <Y-coordinate> Current value of Y(Y1)-axis  
 <θ-coordinate> Current value of θ(Y2)-axis  
 <Start flag> Start flag  
                   True: At the calibration start timing,  
                                   or at the calibration restart timing.  
                   False: Other than above

Return value: <X-coordinate> X-axis travel distance  
 <Y-coordinate> Y(Y1)-axis travel distance  
 <θ-coordinate> θ(Y2)-axis travel distance  
 < End flag> End flag  
                   True: Calibration completed  
                   False: Calibration not completed

**Description** This command provides various settings about calibration execution.

### Example

```
Dim caoCtrl as Object
Dim Res As Variant
```

```
caoCtrl = cao.AddController("IV_S300", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")
Res = caoCtrl.SAlignmentCalibration(0, 134.58, 25.48, 105.6, True)
```

## <ImplVar>.RemoteEnable

**Usage** Switches the remote key input settings to Enable/Disable

**Syntax** **<ImplVar>.RemoteEnable** <Enable/Disable>

Argument : <Enable/Disable> Specify Enable/Disable  
True: Enable  
False: Disable

Return value : None

**Description** This command is used to switch the remote key input setting to Enable/Disable.

### Example

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
caoCtrl.RemoteEnable True
```

## <ImplVar>.ViewLockEnable

**Usage** Switches the operation screen lock setting to Enable/Disable.

**Syntax** **<ImplVar>.ViewLockEnable** <Enable/Disable>

Argument : <Enable/Disable> Specify Enable/Disable  
True:Enable  
False:Disable

Return value : None

**Description** This command switches the operation screen lock setting to Enable/Disable.

### Example

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
caoCtrl.ViewLockEnable True
```

## <ImplVar>.GetKind

**Usage** Reads the object type number.

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

Argument : None

Return value : object type number

**Description** This command reads the object type number.  
The return value is stored in an array in order of "object type number of a trigger number 0", "object type number of a trigger number 1", and so on.

**Example**

```
Dim caoCtrl as Object  
Dim Number As Variant
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
Number = caoCtrl.GetKind()
```

## <ImplVar>.PutKind

**Usage** This command is used to write the object type number.

**Syntax** **<ImplVar>.PutKind** <Object type number>

Argument : <Object type number> Specify the object type number  
IV-S150X/M: 0-99  
IV-S2\*0X, IV-C250X:0-2047  
IV-S3\*0X: 0-199

Return value : None

**Description** This command is used to write the object type number.  
Some object type numbers may take a time to change the object type number. In that case, change the timeout period by using "ChangeTimeout" command.

**Example**

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV","", "Conn=eth:192.168.0.2:2001")  
caoCtrl.PutKind 1
```

## <ImplVar>.GetModule

**Usage** Reads the module number.

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

Argument : None

Return value : Module number

**Description** This command is used to read the module number.

### Example

```
Dim caoCtrl as Object  
Dim Number As Integer
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
Number = caoCtrl.GetModule()
```

## <ImplVar>.PutModule

**Usage**      Writes the module number.

**Syntax**      **<ImplVar>.PutModule** < Module number>

Argument : <Module number> Specify module number(0-127)

Return value : None

**Description**      This command is used to write the module number.

### Example

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
caoCtrl.PutModule 1
```

## <ImplVar>.GetViewMode

**Usage** Reads the image mode

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

Argument : None

Return value : Image mode

**Description** This command is used to read the image mode.

IV-S150*	LV	Live image
	SC	Camera image
IV-S3*0X	RC	Processed image
IV-S2*0X	LV	Live image
	ST	Still image (per trigger)
IV-C250X	SN	Still image (per fail)
	SO	Still image (per pass)

**Example**

```
Dim caoCtrl as Object
Dim strViewMode As String
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")
strViewMode = caoCtrl.GetViewMode()
```



## <ImplVar>.PutViewMode

**Usage**      Writes the image mode.

**Syntax**      **<ImplVar>.PutViewMode** <Image mode>

Argument : <Image mode> Specify an image mode

Return value : None

**Description**      This command is used to write the image mode.

IV-S150* IV-S3*0X	LV	Live image
	SC	Camera image
	RC	Processed image
IV-S2*0X IV-C250X	LV	Live image
	ST	Still image (per trigger)
	SN	Still image (per fail)
	SO	Still image (per pass)

**Example**

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")
caoCtrl.PutViewMode "LV"
```

## <ImplVar>.GetDispMode

**Usage** Reads the camera display mode.

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

Argument : None

Return value : Camera display mode

**Description** This command is used to read the camera display mode.

IV-S150*	C1	Display camera 1
	C2	Display camera 2
	DV	Divided view
IV-S2*0X IV-C250X	MI	Module specification camera
	MO	Module output image
	C1	Display camera 1
	C2	Display camera 2
	C3	Display camera 3
	C4	Display camera 4
	DV	Divided view
IV-S3*0X	C1	Display camera 1
	C2	Display camera 2
	C3	Display camera 3
	C4	Display camera 4
	DV	Display camera 1 + 2
	DW	Display camera 3 + 4
	DX	Display camera 1 + 2 + 3 + 4

**Example**

```
Dim caoCtrl as Object
```

```
Dim strDispMode As String
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")
strDispMode = caoCtrl.GetDispMode()
```

## <ImplVar>.PutDispMode

**Usage**      Writes the camera display mode.

**Syntax**      **<ImplVar>.PutDispMode** <Camera display mode>

Argument : <Camera display mode> Specify the camera display mode

Return value : None

**Description**      This command is used to write the camera display mode.

IV-S150*	C1	Display camera 1
	C2	Display camera 2
	DV	Divided view
IV-S2*0X IV-C250X	MI	Module specification camera
	MO	Module output image
	C1	Display camera 1
	C2	Display camera 2
	C3	Display camera 3
	C4	Display camera 4
	DV	Divided view
IV-S3*0X	C1	Display camera 1
	C2	Display camera 2
	C3	Display camera 3
	C4	Display camera 4
	DV	Display camera 1 + 2
	DW	Display camera 3 + 4
	DX	Display camera 1 + 2 + 3 + 4

Example

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV","", "Conn=eth:192.168.0.2:2001")
caoCtrl.PutDispMode "DV"
```

## <ImplVar>.GetMeasureData

**Usage** Reads coordinates for the manual measurement.

**Syntax** `<ImplVar>.GetMeasureData( <Camera number> )`

Argument : <Camera number> Specify a camera number (1-4)

Return value : coordinates value

**Description** This command is used to read coordinates for the manual measurement. Coordinate values are stored in an array in order of X coordinates of the first point, Y coordinates of the first point, X coordinates of the second point, and Y coordinates of the second point.

### Example

```
Dim caoCtrl as Object  
Dim Res As Variant
```

```
caoCtrl = cao.AddController("IV_S150X" ,"CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
Res = caoCtrl.GetMeasureData(1)
```

## <ImplVar>.ClearFigures

**Usage**      Clears the statistics.

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

Argument : None

Return value : None

**Description**    This command is used to clear the statistics.

## <ImplVar>.PutMeasureData

**Usage**      Writes the coordinates for the manual measurement.

**Syntax**      **<ImplVar>.PutMeasureData** <Camera number>,  
                   <X coordinates of the first point>, <Y coordinates of the first point>,  
                   <X coordinates of the second point>,<Y coordinates of the second point>

Argument :

<Camera number>	Specify the camera number (1-4)
<X coordinates of the first point>	Specify X coordinates of the first point (0-511)
<Y coordinates of the first point>	Specify Y coordinates of the first point (0-479)
<X coordinates of the second point>	Specify X coordinates of the second point (0-511)
<Y coordinates of the second point>	Specify Y coordinates of the second point (0-479)

Return value : None

**Description**      This command is used to write coordinates for the manual measurement.

Example

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")
caoCtrl.PutMeasureData 1,100,100,200,200
```

## <ImplVar>.PutPassword

**Usage**      Writes the operation window lock password.

**Syntax**     **<ImplVar>.PutPassword** <password>

Argument : <password>    Password(0-9999)

Return value : None

**Description**    This command is used to the write operation window lock password.

### Example

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
caoCtrl.PutPassword 1234
```

## <ImplVar>.GetVal

**Usage** Reads the variable value.

**Syntax** **<ImplVar>.Getval**( <Type of variable>,<Number of variable> )

Argument : <Type of variable> For IV-S2\*0X  
Specify a type of variable

0	System variable
1	Module variable (trigger1)
2	Module variable (trigger2)

For IV-S3\*0X  
Specify a number of trigger (0-1)

<Number of variable> Specify a number of variable (0-31)

Return value : Variable value

**Description** This command is used to read a value of variable. The number of significant figures of the variable depends on the setting of the image sensor device.

### Example

```
Dim caoCtrl as Object
DIM fGetVal As Single
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")
fGetVal = caoCtrl.GetVal(0,1)
```



## <ImplVar>.PutVal

**Usage** Writes the variable value.

**Syntax** **<ImplVar>.Putval** <Type of variable>, <Number of variable>, <Variable value>

Argument : <Type of variable> For IV-S2\*0X

Specify a type of variable

0	System variable
1	Module variable (trigger1)
2	Module variable (trigger2)

For IV-S3\*0X

Specify a number of trigger (0-1)

<Number of variable> Specify a number of variables (0-31)

<Variable value> Specify a variable value

Return value : None

**Description** This command is used to write a variable value.  
The number of significant figures of the variable depends on the setting of the image sensor device.

### Example

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")
caoCtrl.PutVal 0,1,1
```

## <ImplVar>.RegStdImage

### Usage

For IV-S2\*0X: Overwrites the reference image

For IV-S3\*0X: Saves (to a non-volatile memory) the last imported camera image as a reference image

### Syntax

**<ImplVar>.RegStdImage** <Camera number>, <Reference image number>

Argument : <Camera number> For IV-S2\*0X : Specify a camera number (1-4)

For IV-S3\*0X : Specify a number of trigger (0-1)

<Reference image number>

For IV-S2\*0X : Specify a reference image number (0-8191)

For IV-S3\*0X : Camera combination number

Camera combination	Camera combination No.
None	0
Camera 1	1
Camera 2	2
Camera 3	4
Camera 4	8
Camera 1+2	3
Camera 1+3	5
Camera 1+4	9
Camera 2+3	6
Camera 2+4	10
Camera 3+4	12
Camera 1+2+3	7
Camera 1+2+4	11
Camera 1+3+4	13
Camera 2+3+4	14
Camera 1+2+3+4	15

Return value : None

**Description** For IV-S2\*0X: This command is used to overwrite the reference image.  
For IV-S3\*0X: This command is used to save (to a non-volatile memory) the last imported camera image as a reference image.  
[Note] The reference image for each camera must be saved for each type.  
Before this command is executed, inspection and measurement must be performed for the same type at least once.

**Example**

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
caoCtrl.RegStdImage 1,1
```

## <ImplVar>.GetShutterSp

**Usage** This command is used to read the shutter speed.

**Syntax** <ImplVar>.GetShutterSp(<Object type number>,<Module number>,  
<Camera number> )

Argument : <Object type number> Specify the object type number (0-2047)

<Module number> Specify the module number (0-127)

<Camera number> Specify the camera number (1-4)

Return value : Shutter speed

**Description** This command is used to read the shutter speed.

Example

```
Dim caoCtrl as Object
```

```
DIM iShutterSp As Integer
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")
```

```
iShutterSp = caoCtrl.GetShutterSp(0,1,1)
```

## <ImplVar>.PutShutterSp

**Usage** Writes the shutter speed.

**Syntax** <ImplVar>.PutShutterSp <Object type number>,<Module number>,  
<Camera number>,<Shutter speed>

Argument :	<Object type number>	Specify the object type number (0-2047)
	<Module number>	Specify the module number (0-127)
	<Camera number>	Specify the camera number (1-4)
	<Shutter speed>	Specify the shutter speed (0-38000)

Return value : None

**Description** This command is used to write the shutter speed.

**Example**

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
caoCtrl.PutShutterSp 0,1,1,5000
```

## <ImplVar>.GetThreshold

**Usage** Reads the threshold value settings.

**Syntax** **<ImplVar>.GetThreshold**( <Object type number>,<Module number> )

Argument : <Object type number> Specify the object type number (0-2047)  
<Module number> Specify the module number (0-127)

Return value : Threshold value

**Description** This command is used to read the threshold value settings. Threshold value is stored in an array in order of the upper limit value, lower limit value.

### Example

```
Dim caoCtrl as Object  
Dim vntThreshold As Variant
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
vntThreshold = caoCtrl.GetThreshold(0,2)
```

# <ImplVar>.PutThreshold

**Usage** Writes threshold value settings.

**Syntax**    **<ImplVar>.PutThreshold** <Object type number>,<Module number>,  
                    <Upper limit of threshold value>,<Lower limit of threshold value>

Argument :

<Object type number>	Specify the object type number (0-2047)
<Module number>	Specify the module number (0-127)
<Upper limit of threshold value>	Specify an upper limit of threshold value (0-255)
<Lower limit of threshold value>	Specify a lower limit of threshold value (0-255)

Return value : None

**Description** This command is used to write threshold value settings.

Example

Dim caoCtrl as Object

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
caoCtrl.PutThreshold 0,2,200,100
```

## <ImplVar>.GetGain

**Usage** Reads the gain and offset settings.

**Syntax** **<ImplVar>.GetGain**( <Object type number>,<Module number>,  
<Camera number> )

Argument : <Object type number> Specify the object type number (0-2047)  
<Module number> Specify the module number (0-127)  
<Camera number> Specify the camera number (1-4)

Return value : Gain value, offset value

**Description** This command is used to read the gain and offset settings. The data is stored in an array in order of gain value, offset value.

### Example

```
Dim caoCtrl as Object  
Dim vntGain As Variant
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
vntGain = caoCtrl.GetGain(0,2,0)
```



# <ImplVar>.PutGain

**Usage** Writes the gain and offset settings.

**Syntax** **<ImplVar>.PutGain** <Object type number>,<Module number>,  
<Camera number>,<gain value>,<offset value>

Argument :	<Object type number>	Specify the object type number (0-2047)
	<Module number>	Specify the module number (0-127)
	<Camera number>	Specify the camera number (1-4)
	<Gain value>	Specify the gain value (0-1023)
	<Offset value>	Specify the offset value (0-1023)

Return value : None

**Description** This command is used to write the gain and offset settings.

## Example

Dim caoCtrl as Object

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")
caoCtrl.Putgain 0,2,100,100
```

## <ImplVar>.GetDate

**Usage** Reads date and time setting.

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

Argument : None

Return value: Date and time setting

**Description** This command is used to read date and time setting.  
The data is stored in an array in order of year, month, day, hour, minute, and second.

### Example

```
Dim caoCtrl as Object  
Dim vntDate As Variant
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
vntDate = caoCtrl.GetDate()
```

## <ImplVar>.PutDate

**Usage**      Writes date and time setting.

**Syntax**      **<ImplVar>.PutDate** <Year>,<Month>,<Day>,<Hour>,<Minute>,<Second>

Argument :	<Year>	Specify a year (2000-)
	<Month>	Specify a month (1-12)
	<Date>	Specify a date (1-31)
	<Hour>	Specify an hour (0-23)
	<Minute>	Specify a minute (0-59)
	<Second>	Specify a second (0-59)

Return value : None

**Description**      This command is used to write date and time setting.

### Example

Dim caoCtrl as Object

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")
caoCtrl.PutDate 2099,1,1,0,0,0
```

## <ImplVar>.GetRegData

**Usage**      Obtains the code reader module register data.

**Syntax**      **<ImplVar>.GetRegData** ( <Trigger number>,<Module number> )

Argument : < Trigger number > Specify a trigger number (0-1)  
            < Module number > Specify the module number (0-127)

Return value : Register data

### **Description**

This command is used to obtain the register data for the code reader module. When COM communication is used, set the number of data bits to 8 bits on the device side and provider side. The default setting for the number of data bits is 7 bits; depending on the character string used, the 8th bit may be dropped, resulting in unintended behavior.

## <ImplVar>.PutRegData

**Usage**      Writes the code reader module register data.

**Syntax**      **<ImplVar>.PutRegData** <Trigger number>,<Module number>,  
   < Register data >

Argument : < Trigger number >    Specify a trigger number (0-1)  
                 < Module number >    Specify the module number (0-127)  
                 < Register data >      Specify the register data

Return value : None

**Description**      This command is used to write the register data for the code reader module. When COM communication is used, set the number of data bits to 8 bits on the device side and provider side. The default setting for the number of data bits is 7 bits; depending on the character string used, the 8th bit may be dropped, resulting in unintended behavior.



**Description**

This command is used to write the set character string for the character inspection module.

When COM communication is used, set the number of data bits to 8 bits on the device side and provider side.

The default setting for the number of data bits is 7 bits; depending on the character string used, the 8th bit may be dropped, resulting in unintended behavior.

## <ImplVar>.GetModuleString

**Usage** Reads the set character string for the character inspection module.

**Syntax** **<ImplVar>.GetModuleString** ( <Trigger number>,<Module number> )

Argument : < Trigger number > Specify a trigger number (0-1)  
< Module number > Specify the module number (0-127)

Return value : Register data

**Description** This command is used to read the set character string for the character inspection module.  
When COM communication is used, set the number of data bits to 8 bits on the device side and provider side.  
The default setting for the number of data bits is 7 bits; depending on the character string used, the 8th bit may be dropped, resulting in unintended behavior.



## <ImplVar>.GetBlockString

**Usage** Reads the set character string for the character inspection module.

**Syntax** **<ImplVar>.GetBlockString** ( <Trigger number>,<Module number>,  
<Block number> )

Argument : < Trigger number > Specify a trigger number (0-1)  
< Module number > Specify the module number (0-127)  
< Block number > Specify the block number (0-7)

Return value : Register data

**Description** This command is used to read the set character string for the character inspection module.  
When COM communication is used, set the number of data bits to 8 bits on the device side and provider side.  
The default setting for the number of data bits is 7 bits; depending on the character string used, the 8th bit may be dropped, resulting in unintended behavior.

# <ImplVar>.PutBlockString

**Usage** Writes the set character string for the character inspection module.

**Syntax** **<ImplVar>.PutBlockString** <Trigger number>,<Module number>,  
<Block number>,<Character string>,  
< Delete space enable/disable >

Argument : < Trigger number >	Specify a trigger number (0-1)
< Module number >	Specify the module number (0-127)
< Block number >	Specify the block number (0-7)
< String character >	Specify the character string
<Delete space enable/disable>	Specify whether deletion of space is enabled or disabled

TRUE	Delete space enabled
FALSE	Delete space disabled

Return value : None

**Description** This command is used to write the set character string for the character inspection module.

When delete space is enabled, the space at the end of the character string will be removed.

When COM communication is used, set the number of data bits to 8 bits on the device side and provider side.

The default setting for the number of data bits is 7 bits; depending on the character string used, the 8th bit may be dropped, resulting in unintended behavior.



## <ImplVar>.Put10BlockString

**Usage**      Writes the set character string for the character inspection module.

**Syntax**      **<ImplVar>.Put10BlockString** <Trigger number>,<Camera number>  
 [,<Character string 1>[,<Character string 2>[,<Character string 3>  
 [,<Character string 4>[,<Character string 5>[,<Character string 6>  
 [,<Character string 7>[,<Character string 8>[,<Character string 9>  
 [,<Character string 10>]]]]]]]]]]]

Argument :	< Trigger number >	Specify a trigger number (0-1)
	< Camera number >	Specify the camera number (1-4)
	< Character string 1 >	Specify the character string 1
	< Character string 2 >	Specify the character string 2
	< Character string 3 >	Specify the character string 3
	< Character string 4 >	Specify the character string 4
	< Character string 5 >	Specify the character string 5
	< Character string 6 >	Specify the character string 6
	< Character string 7 >	Specify the character string 7
	< Character string 8 >	Specify the character string 8
	< Character string 9 >	Specify the character string 9
	< Character string 10 >	Specify the character string 10

Return value : None

**Description**      This command is used to write the set character string for the character inspection module.

When COM communication is used, set the number of data bits to 8 bits on the device side and provider side.

The default setting for the number of data bits is 7 bits; depending on the character string used, the 8th bit may be dropped, resulting in unintended behavior.

## <ImplVar>.SnapShot

**Usage** Stores snapshot image on an external memory.

**Syntax** <ImplVar>.SnapShot

Argument : None

Return value : None

**Description** This command is used to store snapshot image on an external memory.

### Example

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
caoCtrl.SnapShot
```

## <ImplVar>.ImageClear

**Usage**      Clears the image memory stored in the provider.

**Syntax**      <ImplVar>.ImageClear

Argument : None

Return value : None

**Description**      This command is used to clear the image memory stored in the provider.

**Example**

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
caoCtrl.ImageClear
```

## <ImplVar>.GetKindState

**Usage** Reads the object type setting number information.

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

Argument : None

Return value : Object type setting number information

**Description** This command is used to read the object type setting number information. Return value is stored in an array in order of the number of object types, object type number1, and object type number2 to the object type numberN.

### Example

```
Dim caoCtrl as Object
Dim vntKindState As Variant
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")
vntKindState = caoCtrl.GetKindState()
```

## <ImplVar>.GetStdImageState

**Usage** Reads the reference image number information.

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

Argument : None

Return value : Reference image number information

**Description** This command is used to read the reference image number information. Return value is stored in an array in order of the number of reference images, reference image number1, reference image number2 to the reference image numberN.

### Example

```
Dim caoCtrl as Object
Dim vntStdImageState As Variant

caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")
vntStdImageState = caoCtrl.GetStdImageState()
```



## <ImplVar>.GetVersion

**Usage** Reads version information

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

Argument: None

Return value: Version information

**Description** This command is used to read version information. The return value is stored in an array in order of the model code, version information.

### Example

```
Dim caoCtrl as Object  
Dim vntVersion As Variant
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
vntVersion = caoCtrl.GetVersion()
```

## <ImplVar>.AllReset

**Usage**      Initializes the system settings and object type settings

**Syntax**      <ImplVar>.AllReset

Argument : None

Return value : None

**Description**      This command is used to initialize the system settings and object type settings.

**Example**

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
caoCtrl.AllReset
```

## <ImplVar>.SettingSave

**Usage**      The system settings and object type settings are saved.

**Syntax**      <ImplVar>.SettingSave

Argument : None

Return value : None

**Description**      This command is used to save the system settings and object type settings.

### Example

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
caoCtrl.SettingSave
```

## <ImplVar>.Reset

**Usage**      Resets the system

**Syntax**      <ImplVar>.Reset

Argument : None

Return value : None

**Description**      This command is used to reset the system.

**Example**

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
caoCtrl.Reset
```

## <ImplVar>.GetBrightness

**Usage** Reads the average density.

**Syntax** **<ImplVar>.GetBrightness** (<Camera number>,  
<Upper left X coordinates>,<Upper left Y coordinates>,  
<Lower right X coordinates>,<Lower right Y coordinates>)

Argument :	<Camera number>	Specify a camera number (1-4)
	<Upper left X coordinates>	Specify the upper left X coordinates
	<Upper left Y coordinates>	Specify the upper left Y coordinates
	<Lower right X coordinates>	Specify the lower right X coordinates
	<Lower right Y coordinates>	Specify the lower right Y coordinates

Return value : Average density

**Description** This command is used to read the average density.

### Example

```
Dim caoCtrl as Object
DIM iBrightness As Integer

caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")
iBrightness = caoCtrl.GetBrightness(0,100,100,200,200)
```

## <ImplVar>.GetParallel

**Usage** Reads the I/O status of the parallel communication.

**Syntax** <ImplVar>.GetParallel( <I/O type> )

Argument : <I/O type> Specify input or output

0	Input/Output
1	Input only
2	Output only

Return value: I/O status

**Description** This command is used to read variable values. Return values are stored in an array in order of the 1st through 4th bytes of the input, the 1st through 5th bytes of the output.  
The format of the return value will vary for each IV series. Therefore, when replacing from older models, you must take appropriate measures to make sure the return values will be supported.

### Example

```
Dim caoCtrl as Object
Dim vntParallel As Variant
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")
vntParallel = caoCtrl.GetParallel(0)
```

## <ImplVar>.SelfCheck

**Usage**      Performs self-check tests (five types) for the controller.

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

Argument : None

Return value    : System memory test results \*1  
                   : RAM test results \*1  
                   : FPGA access test results \*1  
                   : Camera 1 connection test results \*2  
                   : Camera 2 connection test results \*2  
                   : Camera 3 connection test results \*2  
                   : Camera 4 connection test results \*2

\*1 The following table shows the values used in test results.

Value	Result
0	Pass
1	Fail

\*2 The following table shows the values used in camera test results.

Value	Result
0	Pass
1	Camera connection test failed
2	Camera type test failed
3	Camera field of view test failed
4	Camera import test failed
5	Camera import line test failed

**Description**      This command is used to perform self-check tests (five types) for the controller.

## <ImplVar>.Raw

**Usage**      Sends and receives raw data

**Syntax**      **<ImplVar>.Raw**( <String to send> )

Argument : <String to send> Specify a string to send

Return value : A received response

**Description**      A string specified by argument is sent. The received response is returned as a string. Command and response to be sent or received are not processed by the transmission and reception process.

### Example

```
Dim caoCtrl as Object  
Dim Res As Variant
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
Res = caoCtrl.Raw(":00000000"&"00"&"T00"&"0,"&"@@")
```



## <ImplVar>.ChangeTimeout

**Usage** Specifies communication timeout period

**Syntax** **<ImplVar>.ChangeTimeout** <timeout period>

Argument : <Timeout period> Specify timeout period

Return value : None

**Description** This command is used to specify the communication timeout period. To set the timeout period at AddCotroller, enter "-1" to the timeout period.

### Example

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
caoCtrl.ChangeTimeout 1000
```

## 6. Error codes

In the IV provider, specific error codes shown below are designated. About the commonness error, please refer to the chapter of the error code of "USER MANUALS".

Error name	Error code	Explanation
E_RESPONSE_FAILED	0x80100001	Received invalid response
E_RESPONSE_CHECKSUM_FAILED	0x80100002	Response data checksum error
E_RESPONSE_COMMAND_FAILED	0x80100003	Response data command error
E_RESPONSE_LENGTH_FAILED	0x80100004	Response data length error
E_RESPONSE_AREA_FAILED	0x80100005	Response area code error
E_DEVICE_ERROR	0x80100100-	Device error

For about E\_DEVICE\_ERROR

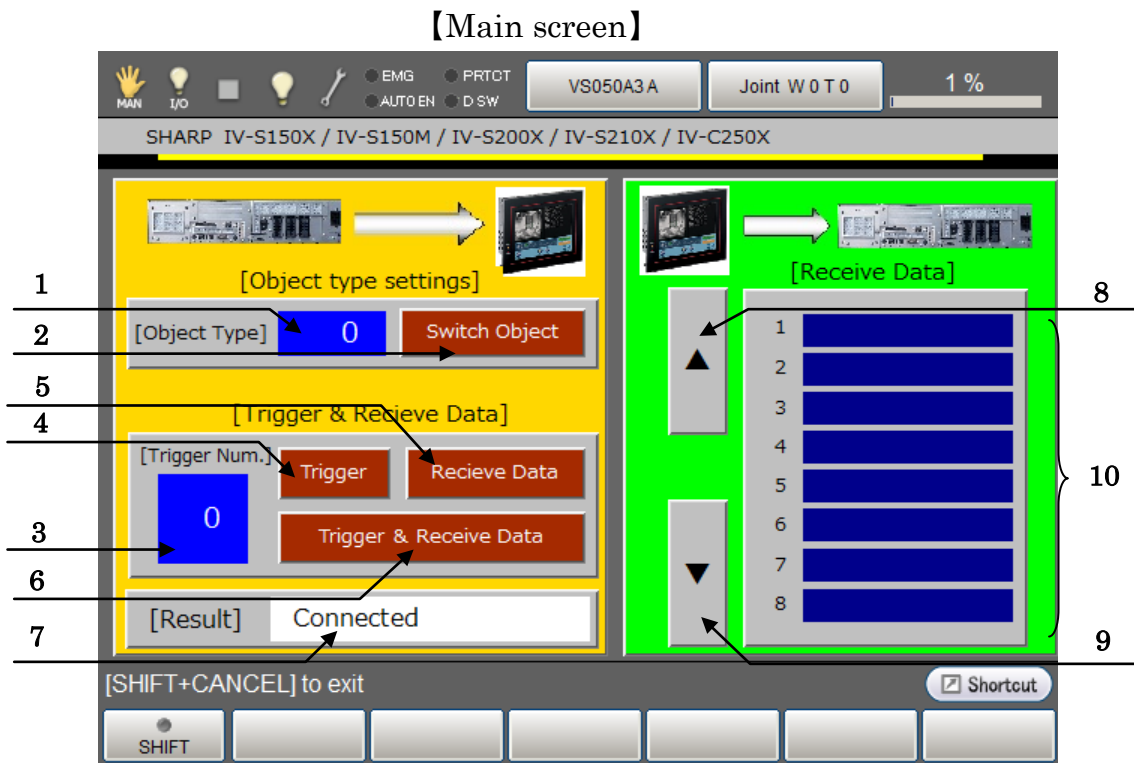
As an error response from the device, the value masked "0x80100100" is output.

Example: Impossible to execute command: 40(H) > CAOAPI error: 0x80100140

For details about errors, refer to the image sensor camera user's manual of Sharp manufacturing systems corporation.

## 7. Operation Panel Screen

This provider equips the operation panel screen as shown below. This operation panel uses the provider to check operations, and other purposes after connecting to the device. See the following as an application example of the operation panel. Displaying the operation panel establishes connection to the IV (implements the provider), so make sure to perform communication settings before displaying the operation panel. Closing the operation panel terminates the connection (release the provider).



**Description** Each button functions as follows.

1. Specifies the object type number
2. Switches to the object type number specified by (1). (PutKind)
3. Specifies a trigger number
4. Enters a trigger by the trigger number specified by (3). (Trigger)
5. Receives data (GetData)
6. Enters a trigger by the trigger number specified by (3), and then receive data (TriggerAndWait)
7. Displays the processing result
8. Moves up the page displayed for received data
9. Moves down the page displayed for received data
10. Displays received data

Note: When provider implementation (initialization) is done correctly, "Connected" will be shown in the processing result field (7).

## 8. Sample Program

### 8.1 Inputting a trigger to obtain a result

Sub Main

```

On Error Goto ErrProc          'Declare error process routine

Dim caoCtrl as Object          'Declare provider variable
Dim Res As Variant            'Declare Variant variable
Dim pTargetPos as Position     'Declare P-type variable

takearm keep = 0
pTargetPos = P11

caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")
                                                'Provider implementation

caoCtrl.PutKind 1              'Switch to object type number1
Res = caoCtrl.TriggerAndWait(0) 'Input a trigger and receive the result

letx pTargetPos = posx(P11) + Res(1) 'Expand X component of received data to position data
lety pTargetPos = posy(P11) + Res(2) 'Expand Y component of received data to position data

approach p, pTargetPos, @p 20, s = 100 'Go to above the after correction position
move l, @e pTargetPos, s = 10         'Go to position after correction
Hand[0].Chuck 0                      'Chuck
depart l, @p 50, s = 100              'Move upward

EndProc:                             'Normal end routine
    'State necessary end procedure
exit sub

ErrProc:                              'Abnormal end routine
    'State necessary error procedure

End Sub

```

\*There are several ways to obtain the result by inputting trigger on the IV. In the example shown above, only one command, `TriggerAndWait`, is used to input a trigger and receive the result. On the other hand, the following example employs two commands: `Trigger` command is for inputting trigger and `GetData` command is for obtaining the result. In the later example, you need check that the image processing has terminated by the `RDY` signal or other methods before obtaining the result. This is because if the `GetData` command is executed before the image processing termination of the IV, the command obtains the result where the image processing has not completed.

```
caoCtrl.Trigger 0
```

```
Res = caoCtrl.GetData(0)
```

## 8.2 Calibration

'IP address of IV device

```
#Define IP_ADDRESS "192.168.1.20"
```

'Index of the P-type variable that stores the home position of robot (Robot moves to this position at the start and end of motion.)

'Set the coordinate of the home position in P[HOME\_POSITION\_INDEX] beforehand.

```
#Define HOME_POSITION_INDEX 0
```

'Index of P-type variable that stores RobotCalibration start position (the first point)

'Set the coordinate of the start position in P[START\_POSITION\_INDEX] beforehand.

```
#Define START_POSITION_INDEX 1
```

Sub Main

```
On Error GoTo ErrHandler
```

```
Dim objCtrl As Object
```

```
objCtrl = cao.AddController("iv", "CaoProv.SHARP.IV", "", "Conn=eth:" & IP_ADDRESS _  
    & ",Type=2,Checksum=1")
```

```
Dim CurrentPosition As Position
```

```
Dim CurrentX as Double
```

```
Dim CurrentY as Double
```

```
Dim CurrentZ as Double
```

```
Dim CurrentRx as Double
```

```
Dim CurrentRy as Double
```

```
Dim CurrentRz as Double
```

```
TakeArm Keep = 1
```

```
ExtSpeed 10
```

```
'Move to the home position
```

Hold "Move the robot to the home position. Ensure the safety of surrounding environment, and then retry."

```
Move P,P[HOME_POSITION_INDEX]
```

'Move the robot to the start position of RobotCalibration

Hold "Move the robot to the start position. Ensure the safety of surrounding environment, and then retry."

Move P,P[START\_POSITION\_INDEX]

RestartProc:

CurrentPosition = CurPos

CurrentX = PosX(CurrentPosition)

CurrentY = PosY(CurrentPosition)

CurrentZ = PosZ(CurrentPosition)

CurrentRx = PosRx(CurrentPosition)

CurrentRy = PosRy(CurrentPosition)

CurrentRz = PosRz(CurrentPosition)

Dim varRet as Variant

'Execute RobotCalibration

varRet = objCtrl.RobotCalibration(0, CurrentX, CurrentY, CurrentRz)

IF(varRet(0) = False) Then

Hold "Robot starts moving. Ensure the safety of surrounding environment, and then retry."

'Move the robot

Move P,P(varRet(1), varRet(2), CurrentZ, CurrentRx, CurrentRy, varRet(3), -2)

Goto RestartProc

End If

'Move to the home position

Hold "Move the robot to the home position. Ensure the safety of surrounding environment, and then retry."

Move P,P[HOME\_POSITION\_INDEX]

PostProc:

IF(IsNothing(objCtrl) = False)Then

cao.Controllers.Remove objCtrl.Index

objCtrl = Nothing

End If

Exit Sub

ErrorHandler:

MsgBox (Err.Description) , 0+16, ErrMsg( Err.Number ) & " : " & (Hex( Err.Number ))

Goto PostProc

End Sub



## Revision history

Denso Robot  
Provider  
User's Manual

Sharp Manufacturing Systems Corporation      Image sensor camera      IV series

Version	Supported RC8	Content
Ver.1.0.0	Ver.1.4.5	First edition
Ver.1.0.1	Ver.2.2.*	IV-S300X, IV-S310X is supported. Addition of commands: "GetRegData", "PutRegData", "PutDateString", "GetModuleString", "GetBlockString", "PutBlockString", "Put20BlockString", "Put10BlockString", "SysSettingSave", "SelfCheck", "ClearFigures" Modified of commands: "PutKind", "PutViewMode", "PutDispMode", "RegStdImage"
Ver.1.0.2	Ver.2.5.*	IV-S301M is supported. Added SAlignmentTrigger, SAlignmentCalibration, RobotCalibration.

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