DENSO

DENSO Robotics

THIRD PARTY PRODUCTS

PROVIDER MANUAL

Maker

COGNEX

Products/Series

Vision Sensor

MODEL: In-Sight Series

Vision

Introduction

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

COGNEX In-Sight 5000/Micro 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

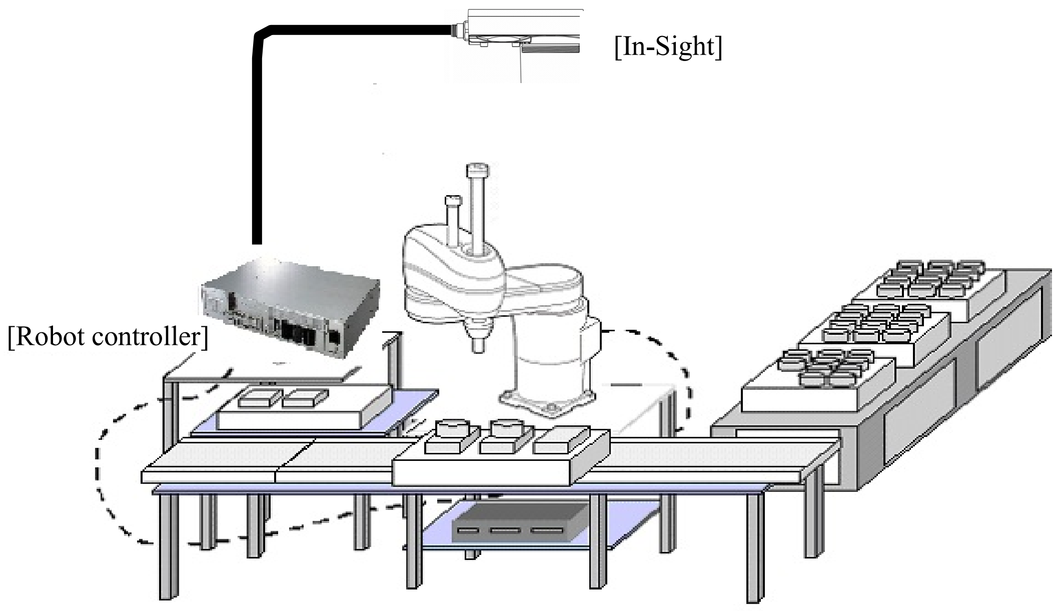
The user of this product shall be responsible for embedding and using the product (software) on a system and any result from using it. Before using this product, be sure to visit our website and read "Software License Agreement" on the product download page.

Contents

1. Outline of This Product (Provider)

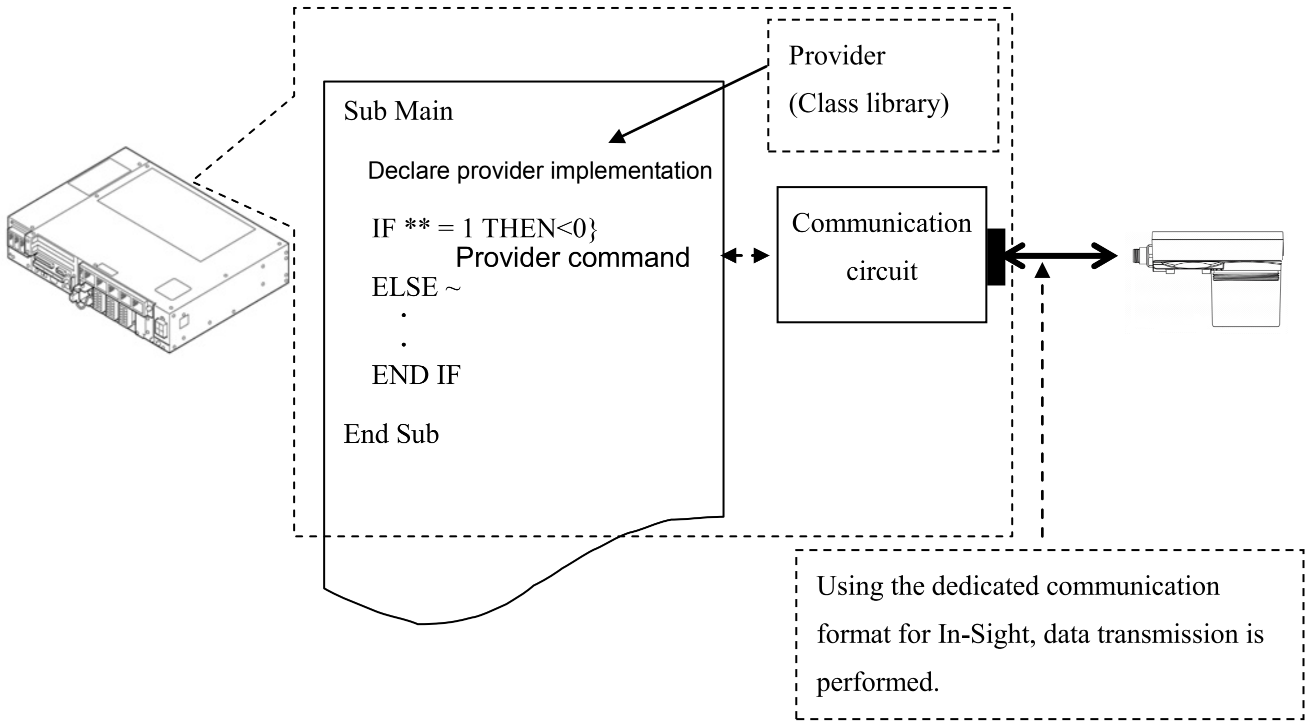
1.1 Target device of provider

This provider can be used only when a DENSO robot controller (RC8 series) is connected to the In-Sight series. (In-Sight2000 is not supported.)



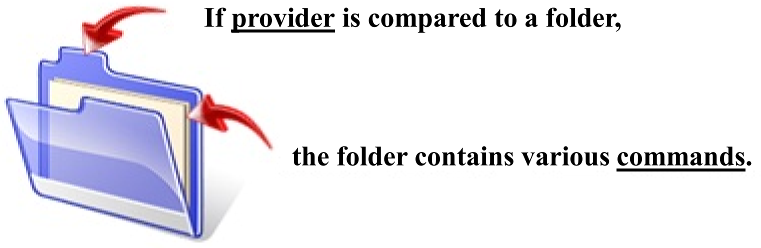
1.2 Features of provider

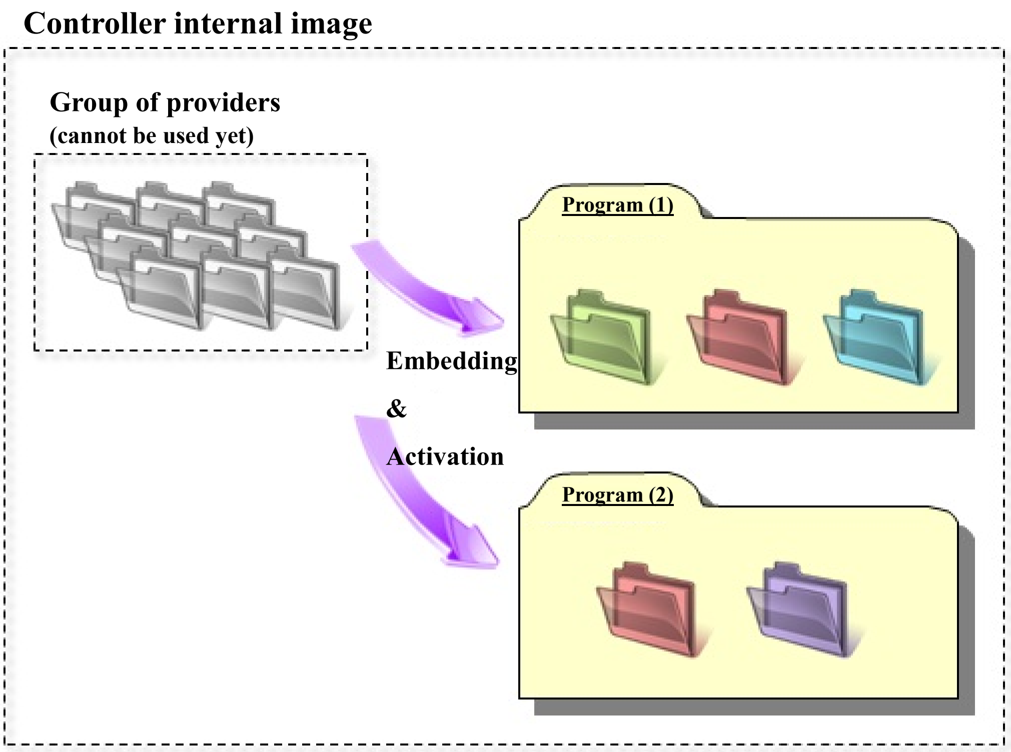
This provider is provided to use the In-Sight native commands required to access In-Sight series in the robot program. Use of this provider allows customers to establish communication with a robot easily without creating a communication program for In-Sight series. The following shows a diagram of provider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


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

1.3 Mechanism of provider

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


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



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

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

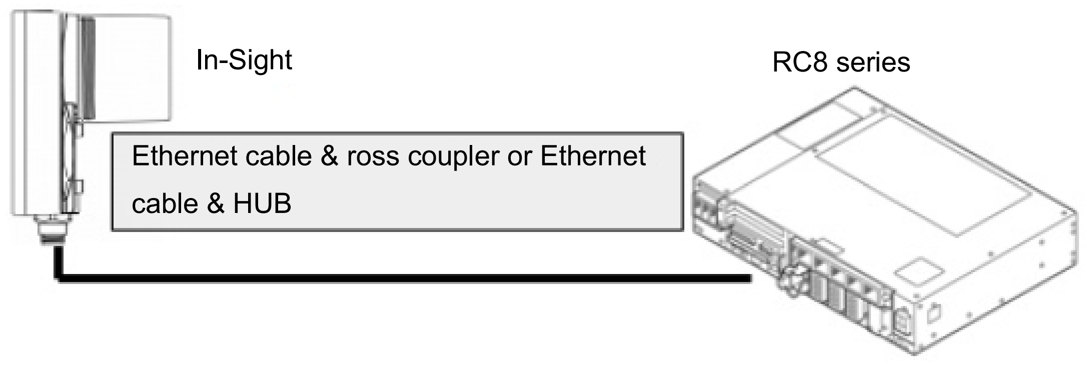
Note: When the same provider exists in different programs like in the above figure, exclusion process is required between the programs (tasks).

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

2. How to Connect

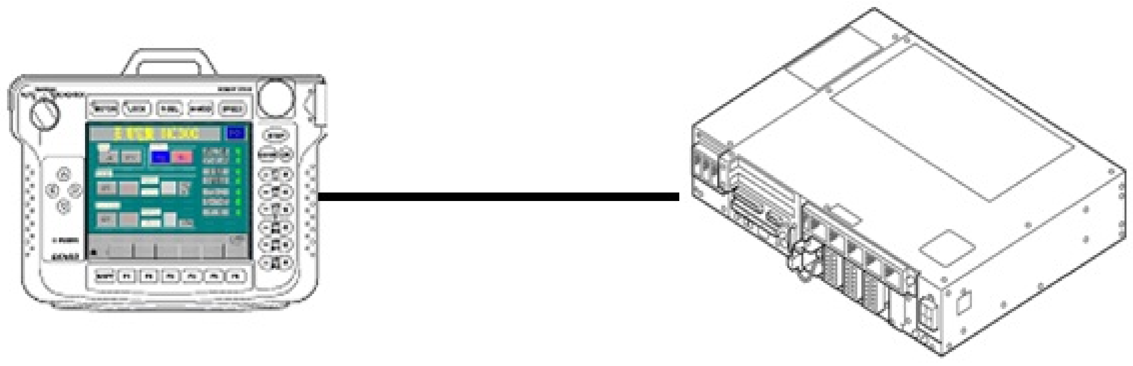
2.1 Ethernet (TCP/IP) connection example

To connect the robot controller to In-Sight via Ethernet (TCP/IP), use the network cable and the cross coupler provided with In-Sight. Also, when a switching hub/router is used, use the cable suitable for the switching hub/router specifications.



3. Communication Settings for Robot Controller and Device Used

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

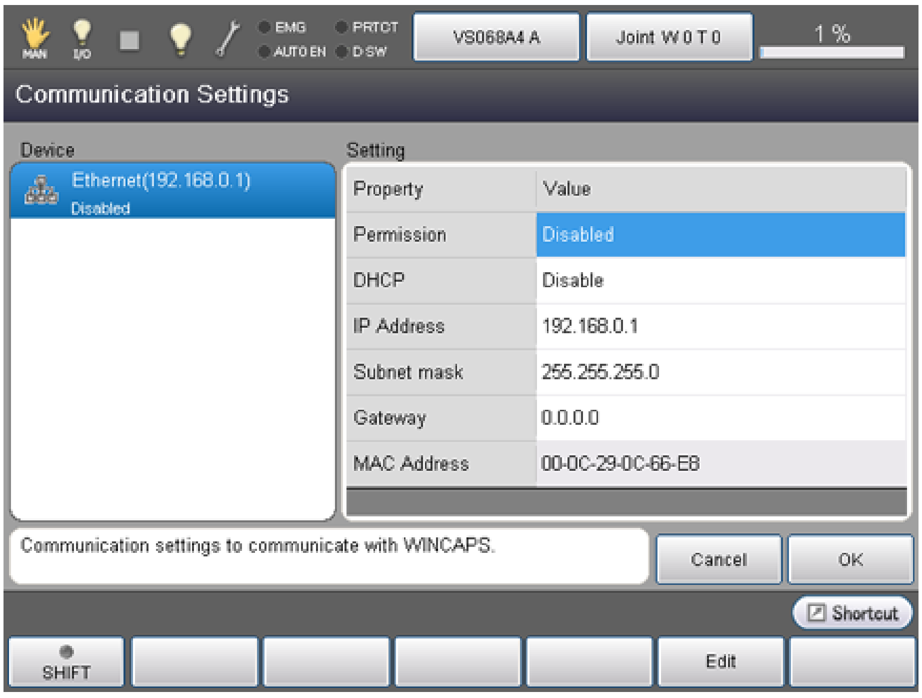


3.1 Communication via Ethernet (TCP/IP)

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

Set the robot controller's IP address.

(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 In-Sight are within the same subnet mask.



Communication Settings

Device

Ethernet(192.168.0.1)
Disabled

Setting

Property	Value
Permission	Disabled
DHCP	Disable
IP Address	192.168.0.1
Subnet mask	255.255.255.0
Gateway	0.0.0.0
MAC Address	00-0C-29-0C-66-E8

Communication settings to communicate with WINCAPS.

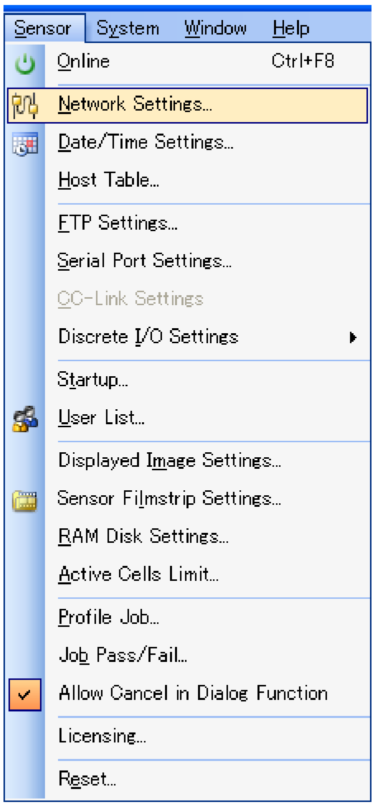
Cancel OK

3.1.2 Ethernet (TCP/IP) communication settings for In-Sight

Set the IP address and user list for In-Sight. with In-Sight Explorer.

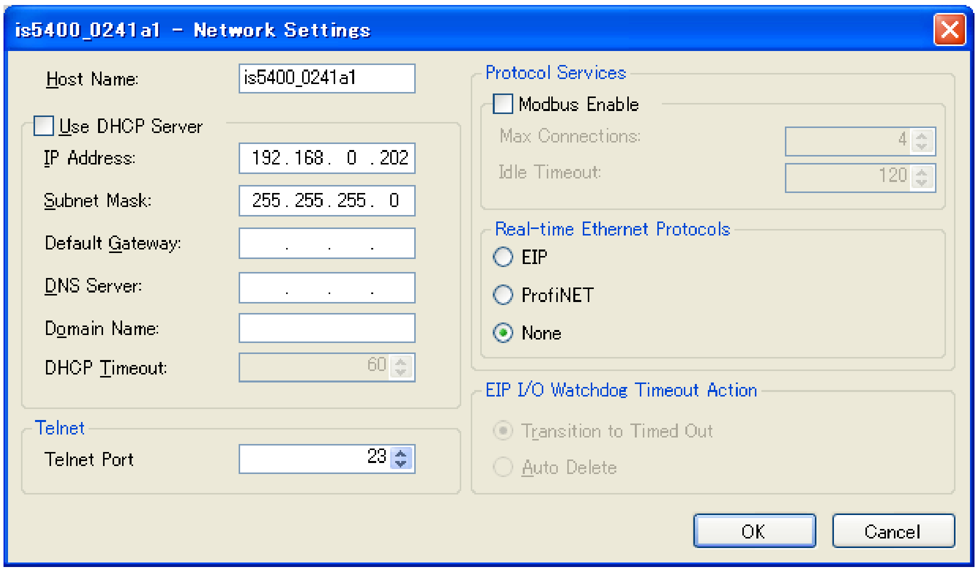
(1) Select [Network Settings] from [Sensor] menu.

[Screen of In-Sight Explorer]



(2) Set the IP address and subnet mask on [Network Settings] screen.

Set the IP address and subnet mask so that the robot controller and In-Sight are within the same subnet mask. .


is5400_0241a1 - Network Settings

Host Name: is5400_0241a1

☐ Use DHCP Server

IP Address: 192 . 168 . 0 . 202

Subnet Mask: 255 . 255 . 255 . 0

Default Gateway: . . .

DNS Server: . . .

Domain Name:

DHCP Timeout: 60

Telnet

Telnet Port 23

Protocol Services

☐ Modbus Enable

Max Connections: 4

Idle Timeout: 120

Real-time Ethernet Protocols

○ EIP

○ ProfiNET

● None

EIP I/O Watchdog Timeout Action

● Transition to Timed Out

○ Auto Delete

3.2 Output result setting

There are two major ways of acquiring the result from In-Sight:

1. The way of acquiring the data of spreadsheet
2. The way of acquiring the result by EasyBuilder

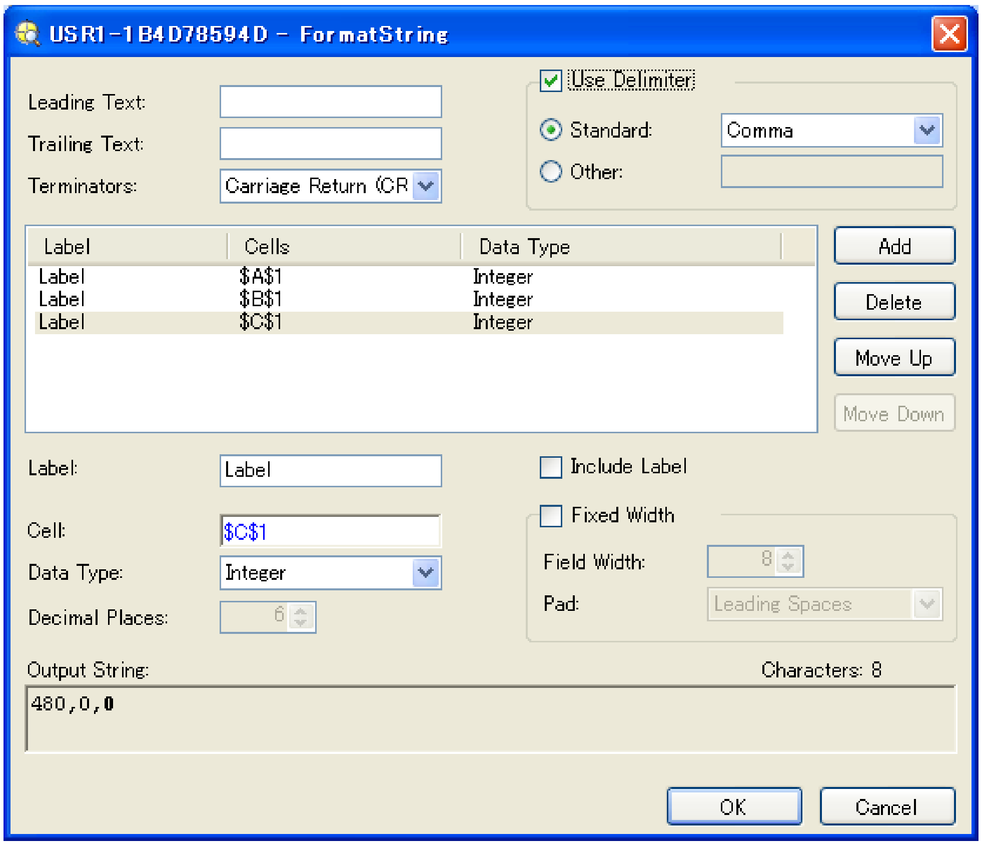
3.2.1 Acquiring the data of spreadsheet

Use the following spreadsheet program example as a guide to program for In-Sight with In-Sight Explorer. Inappropriate programming results in incorrect data input/output.
Refer to the COGNEX In-Sight user's manual for details about programming.

* The following shows an example that uses FormatString and WriteMessage functions for acquiring data collectively. This is not necessary when cell values are acquired individually.

(1) The result of image recognition which you want to output can be transformed to the string by using FormatString function.
To store the formula (function) and its value into the cell on the spreadsheet is possible. Specify the cell and define FormatStrin function on the following screen as the formula (function) of corresponding cell. That string transformed is entered as the value of corresponding cell.

Note: Be sure to set the terminal symbol because it will be used when receiving the result with SendMessageAndWait command.



USR1-1B4D78594D - FormatString

Leading Text:
Trailing Text:
Terminators: Carriage Return (CR

[✓] Use Delimiter
(●) Standard: Comma
( ) Other:

Label	Cells	Data Type
Label	A1	Integer
Label	B1	Integer
Label	C1	Integer

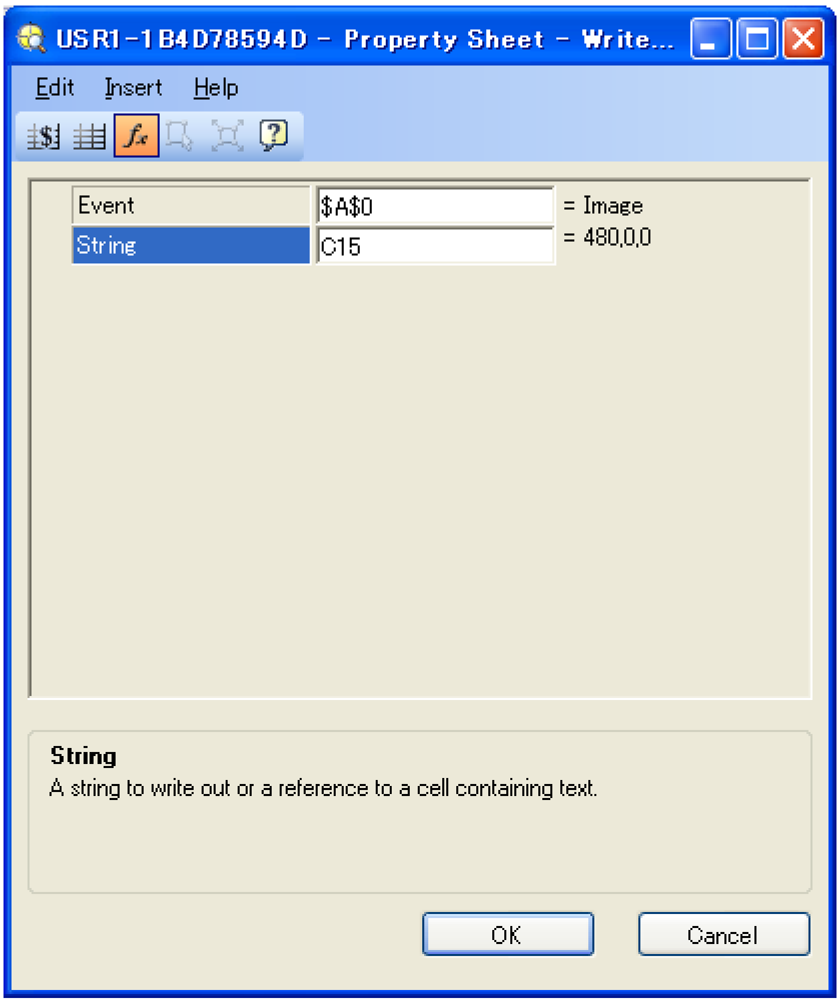
Label: Label
Cell: C1
Data Type: Integer
Decimal Places: 6

[ ] Include Label
[ ] Fixed Width
Field Width: 8
Pad: Leading Spaces

Output String: Characters: 8

```
480,0,0
```

(2) Output the string that was created by FormatString function, by using WriteMessage function.
WriteMessage function is able to acquire the string that was transformed by FormatString function, by specifying the cells to be defined FormatString function.



Event	A0	= Image
String	C15	= 480,0,0

String
A string to write out or a reference to a cell containing text.

Example of use:

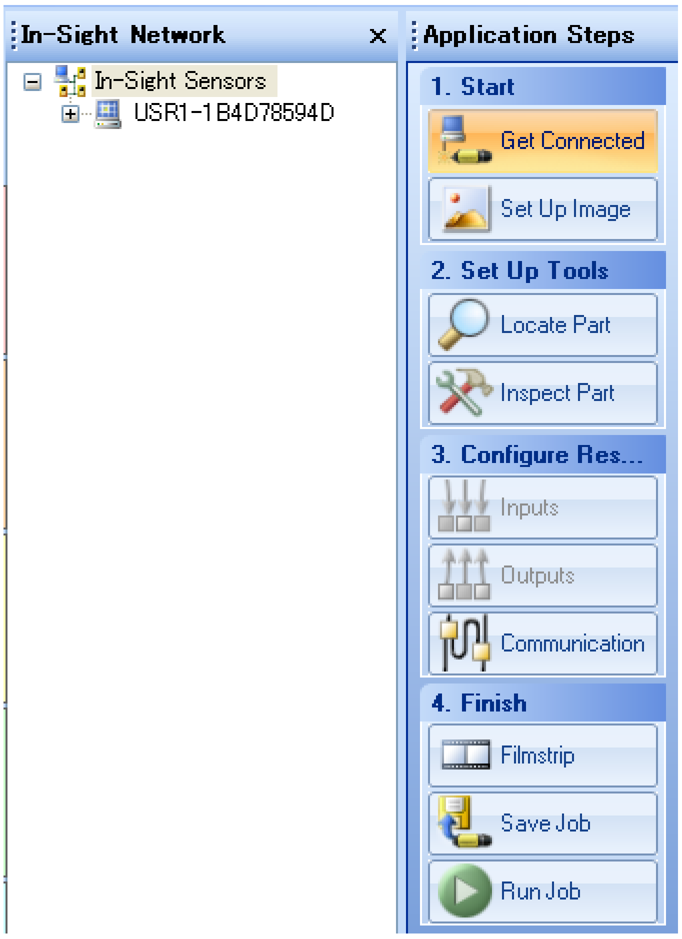
```
Sub Main

        Dim caoCtrl as Object
        Dim strResult as String

        caoCtrl = cao.AddController("InSight", _
        "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202,Timeout=1000)
        strResult = caoCtrl.SendMessageAndAndWait( "", 8, 1)

End Sub
```

3.2.2 EasyBuilder result acquisition

Use the communication function of EasyBuilder to acquire the processing result of images created with EasyBuilder. Specify desired output results in the communication settings to acquire the results using the SendMassageAndGetEZ command or receive the results as event output of CaoController. The following describes how to specify communication settings of EasyBuilder.

3.2.2.1 EasyBuilder setting

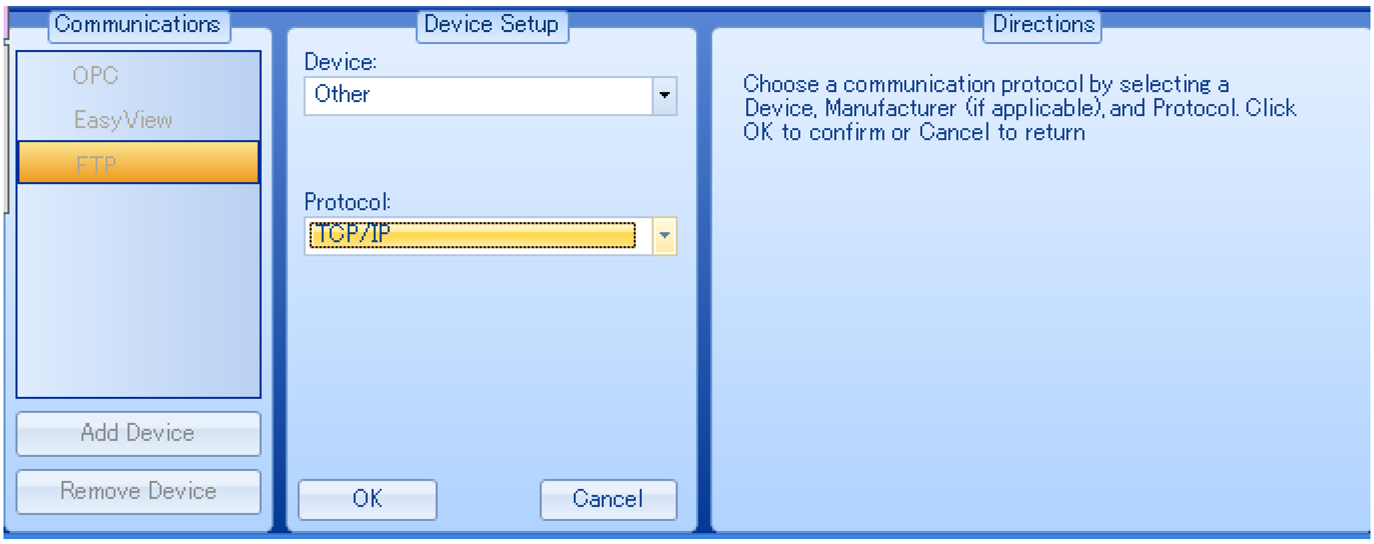
First, press [Result Setting] > [Communications] button to open the communication device setting screen.



Next, press the [Add Device] button to add the device shown below.

Device: Other

Protocol: TCP/IP



Device: Other

Protocol: TCP/IP

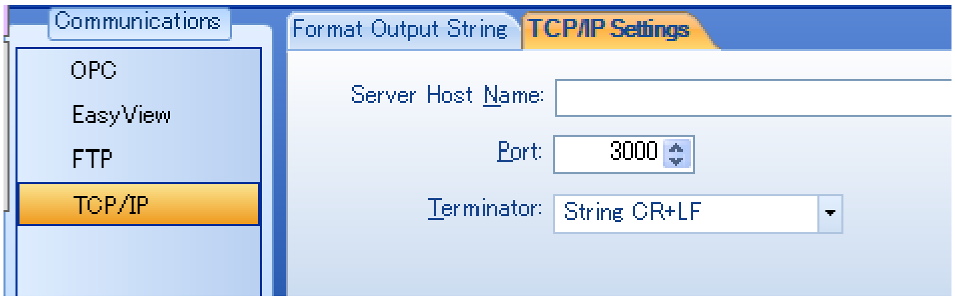
Choose a communication protocol by selecting a Device, Manufacturer (if applicable), and Protocol. Click OK to confirm or Cancel to return

Next, specify the TCP/IP settings.

Server host name: IP address of RC8 that uses the provider

Port: Port number specified in EZPort option of AddController

Terminal symbol: CRLF (fixed)

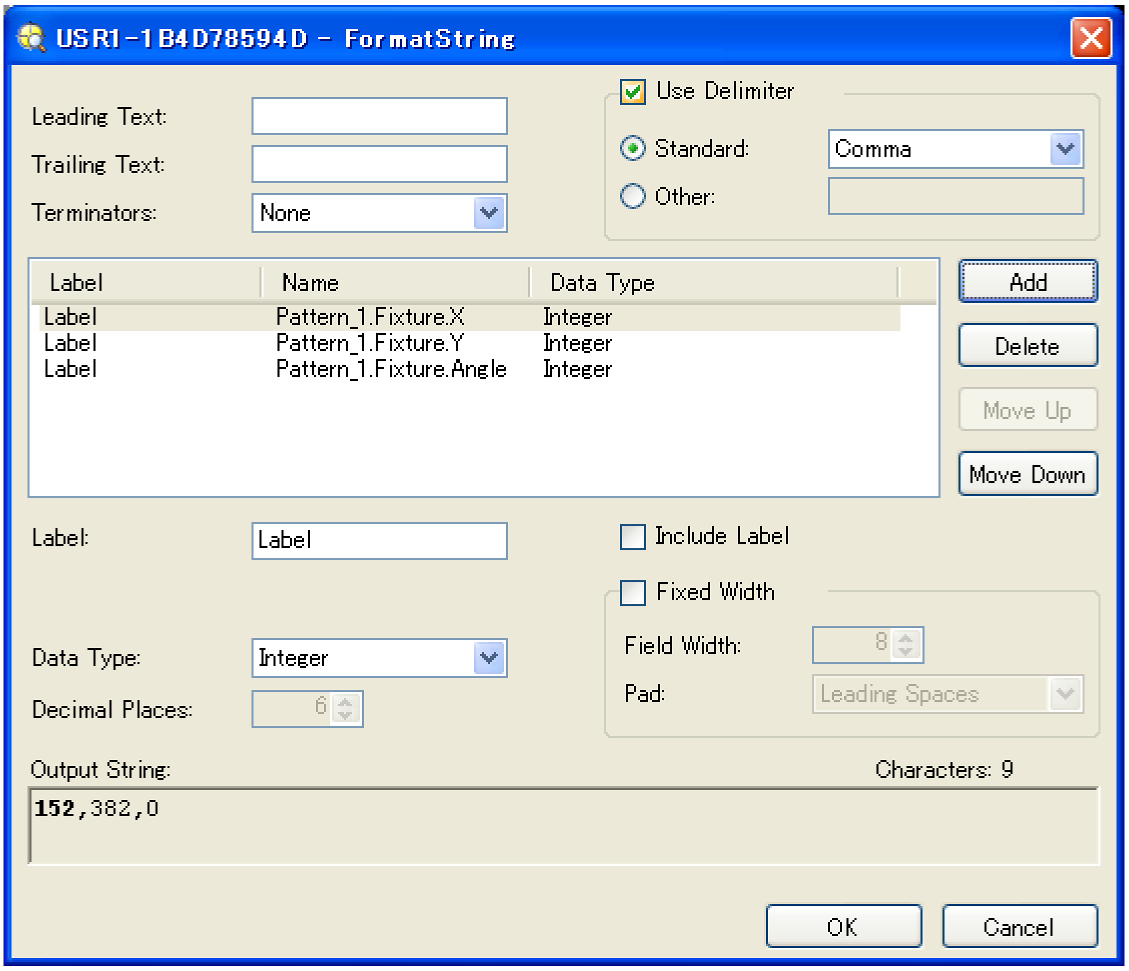


Server Host Name:

Port: 3000

Terminator: String CR+LF

Next, click the [Format Output Character String] tab and then the [Custom Format] button to register the desired output result. Set the terminal symbol to "None."


USR1-1B4D78594D - FormatString

Leading Text:

Trailing Text:

Terminators: None

[✓] Use Delimiter

(●) Standard: Comma

( ) Other:

Label	Name	Data Type
Label	Pattern_1.Fixture.X	Integer
Label	Pattern_1.Fixture.Y	Integer
Label	Pattern_1.Fixture.Angle	Integer

Label: Label

[ ] Include Label

[ ] Fixed Width

Field Width: 8

Pad: Leading Spaces

Data Type: Integer

Decimal Places: 6

Output String: Characters: 9

```
152,382,0
```

In online mode with the settings described above, the results are sent to the specified controller after a trigger is set. The following section describes how to receive results.

3.2.2.2 Result acquisition

To receive the specified results on EasyBuilder, specify the output port specified for EasyBuilder as "EZPort=<Port number>" in the AddController option.

To receive results, use either the "SendMessageAndGetEZ" command or the "SendMessageEZ" and "GetEZ" commands. The "SendMessageAndGetEZ" command sets a software trigger and waits until results are received. Since the "SendMessageEZ" only sets a software trigger, use the "GetEZ" command to receive results.

Sample program for SendMessageAndGetEZ

```
Dim caoCtrl as Object
Dim objCell as Object
Dim strResult as String

    caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202,
                    Timeout=1000, EZPort=3000")

    strResult = caoCtrl.SendMessageAndGetEZ( "Pattern2", 8, 1000 )
```

Sample program for SendMessageEZ and GetEZ

```
Dim caoCtrl as Object
Dim objCell as Object
Dim strResult as String

    caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202,
                    Timeout=1000, EZPort=3000")

    caoCtrl.SendMessageEZ "Pattern2", 8

    'Other processing

    strResult = caoCtrl.GetEZ(1000)
```

3.3 Other settings

In-Sight will reject a switch to online in the native mode (signal from the robot controller) when any of the following conditions is met.

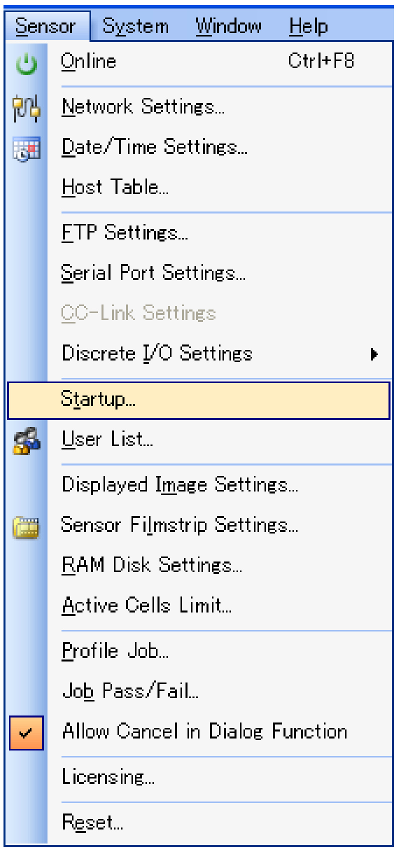
- Offline state is entered from In-Sight Explorer, etc.
- Offline state is entered by discrete input.
- In-Sight is in the offline state when system startup is completed after power ON.

In any of above cases, a switch to online in the native mode will be accepted when the online state is brought by In-Sight Explorer or discrete input.

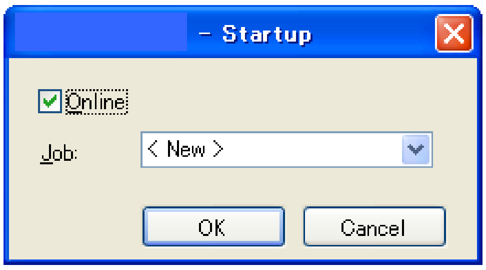
In-Sight can be started in the online mode by setting the In-Sight startup to online using the following procedure. Refer to the COGNEX In-Sight user's manual for details about startup.

(1) Select [Startup] from [Sensor] menu.

[Screen of In-Sight Explorer]



(2) Check the [Online] checkbox on [Startup] setting screen.



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("InSight", "caoProv.Cognex.In-Sight", "",
      "conn=eth:192.168.0.202") (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, In-Sight commands, and proprietary extension commands. For the detailed operation of In-Sight commands, refer to the communications reference in In-Sight Explorer Reference of Cognex.

Table 5-1 List of commands

Command	In-Sight command	Usage	Refer to
Connection commands			
cao.AddController	—	Implements the provider to a variable and makes a connection.	20
Addvariable	—	Creates a variable used exclusively for acquiring image or cell value.	21
Value	—	Acquires data through the variable created by AddVariable.	22
In-Sight commands			
LoadFile	Load File	Loads the specified job from the memory and makes it active.	23
StoreFile	Store File	Saves the active job to the memory.	24
DeleteFile	Delete File	Deletes a job from the memory.	25
GetFile	Get File	Acquires the name of active job file.	26
SetJob	Set Job	Loads a job from the job slot and makes it active.	27
StoreJob	Store Job	Saves the current job to the specified job slot.	28
DeleteJob	Delete Job	Deletes a job from the specified job slot within the memory.	29
GetJob	Get Job	Returns the job slot to which an active job has been loaded.	30
GetValue	Get Value	Returns a value contained in the specified cell.	31
SetInteger	Set Integer	Sets the edit box control stored in a cell to the specified integer.	32
SetFloat	Set Float	Sets the edit box control stored in a cell to the specified floating point.	33
SetString	Set String	Sets the edit box control stored in a cell to the specified character string.	34
GetInfo	Get Info	Returns the system information from the In-Sight device.	35
StoreSettings	Store Settings	Saves the settings in the proc.set file in the memory.	36
SetIPLock	Set IP Lock	Prevents the IP address from being changed without permission.	37
GetIPLock	Get IP Lock	Returns whether access to IP address access is disabled or enabled.	38
SetOnline	Set Online	Sets the In-Sight device to online or offline.	39

GetOnline	Get Online	Returns the online state of In-Sight processor.	40
SetEvent	Set Event	Sets a trigger on the specified event (such as image capture).	41
SetEventAndWait	Set Event & Wait	Sets a trigger on a specified event (such as image capture) and returns a response after a command is completed.	42
SendMessage	Send Message	Sends an ASCII character string to the In-Sight device spreadsheet through connection to the native mode.	43
GetFilelist	Get Filelist	Returns the number of files and file names stored in the memory.	44
Proprietary extension commands			
NativeMode	—	Transfers native mode commands.	45
SendMessageAndWait	—	Causes a software trigger and receives WriteMessage.	46
GetMessageEX	—	Receives WriteMessage.	47
SetTimeoutNM	—	Sets the timeout period for communications with In-Sight.	48
GetTimeoutNM	—	Acquires the timeout period for communications with In-Sight.	49
EZ Builder commands			
SendMessageAndGetEZ	—	Causes a software trigger and receives the result output of TCP communications specified on EasyBuilder.	50
SendMessageEZ	—	Causes a software trigger and prepares for receiving the results by GetEZ command.	51
GetEZ	—	Receives the specified results on EasyBuilder.	52

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 In-Sight.

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.Cognex.In-Sight"
<Provider running machine name> Omit this parameter.
<Option> [Connection parameter], [User name], [Password]
[Timeout period], [Port number]

[Connection parameter] "conn=eth:<IP address>[:Port number]"
Default port number: 23 (The port number is optional).
[User name] Specify the user name to log on to In-Sight.
"Usr[=<User name>]". Default: admin (optional).
[Password] Specify the password used to log on to In-Sight.
"Password[=<Password name>]. Default: None (optional)
[Timeout period] Specify the timeout period (msec) for transmission.
"Timeout[=<Time>]" Default: 500 (optional).
[<Port number>] Specify the port number for acquiring results when EasyBuilder is used.
"EZPort=<Port number>" (When this is omitted, EasyBuilder cannot be used).

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".)
A telnet communication is established when the provider becomes effective. Set the connection detail with <Option>.

Example

```
    Dim caoCtrl as Object

    caoCtrl =    cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth: 192.168.0.202")
```

* Specify a port number, etc. as follows:

```
   caoCtrl =    cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth: 192.168.0.202:23, Usr = admin,
    Password = pass, Timeout = 1000")
```

<ImplVar>.AddVariable

Usage

Creates a variable (property variable) used exclusively for acquiring image or cell value.

Syntax

<ImplVar>.AddVariable (<Variable name>, [<Option>] **)**

Argument:	<Variable name>	For a cell value, specify the variable name with an arbitrary character string and the location of the cell with an optional character string. Acquiring image via native mode: @BITMAP Acquiring image via Datachannel: @BITMAP_DC Optional character string can be used in acquiring image via Datachannel.

	<Option>	Cell = cell number
		Screen = 1 (default), 2, 4
		Port = 5000 (default)
		Timeout = 500 (default)
		SM8 = False (default)

Description

A variable is created to acquire image or cell value from In-Sight.
An image is acquired with an array. A cell value is acquired with a character string.
The Screen option is either set value or screen size.
Specify the Datachannel port number for Port.
Specify the Datachannel timeout for Timeout.
SM8 sends SendMessage 8 to trigger when the data is acquired.
AddVariable cannot be executed unless In-Sight camera captures an image once.
A property variable created with AddVariable is dedicated for acquiring In-Sight image or data from a fixed cell. To acquire value from a different cell using the same variable, the variable needs to be redirected using AddVariable.

Example

```
Dim caoCtrl as Object
Dim objCell as Object
Dim strResult as String

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202")
caoCtrl.SetEventAndWait 8

objCell = caoCtrl.AddVariable("C40", "Cell = C40")
strResult = objCell.Value
objCell = caoCtrl.AddVariable("C41", "Cell = C41")
strResult = objCell.Value
```

<PropVar>.Value

Usage Acquires data from the provider through the variable created by AddVariable.

Syntax **<PropVar>.Value**

Return value: Depends on the type created by AddVariable.

Description Data is acquired from the provider (implementation variable) through the variable created by AddVariable.

Example

```
                "If acquiring a cell value"
        Dim caoCtrl as Object
        Dim objCell as Object
        Dim strResult as String

        caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202")
        caoCtrl.SetEventAndWait 8


        objCell = caoCtrl.AddVariable("C40", "Cell = C40")
        strResult = objCell.Value



        "If acquiring an image"
        Dim caoCtrl as Object
        Dim objBmp as Object
        Dim strResult as String

        caoCtrl =    cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202")
        caoCtrl.SetEventAndWait 8

        objBmp = caoCtrl.AddVariable("@BITMAP")
        Label.Picture = objBmp.Value
```

<ImplVar>.LoadFile

Usage

Loads the specified job from the memory and makes it active.

Syntax

<ImplVar>.LoadFile <Job name>

Argument: <Job name> Specify the job name with a character string.

Description

The specified job is loaded from the In-Sight flash memory and made active.
In-Sight needs to be in the offline state.

Example

```
Dim caoCtrl as Object

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202")
caoCtrl.SetOnline 0
caoCtrl.LoadFile "PRO1.job"
caoCtrl.SetOnline 1
```

<ImplVar>.StoreFile

Usage Saves the current job in the In-Sight flash memory.

Syntax **<ImplVar>.StoreFile** <Job name>

Argument: <Job name> Specify the job name with a character string.

Description The current job is stored in the In-Sight flash memory. Specify the file name with an argument.
In-Sight needs to be in the offline state.

Example

```
Dim caoCtrl as Object

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202")
caoCtrl.SetOnline 0
caoCtrl.StoreFile "PRO1.job"
caoCtrl.SetOnline 1
```

<ImplVar>.DeleteFile

Usage

Deletes the specified job from the In-Sight flash memory.

Syntax

<ImplVar>.DeleteFile <Job name>

Argument: <Job name> Specify the job name with a character string.

Description

The job file specified by an argument is deleted from the In-Sight flash memory.
In-Sight needs to be in the offline state.

Example

```
Dim caoCtrl as Object

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202")
caoCtrl.SetOnline 0
caoCtrl.DeleteFile "PRO1.job"
caoCtrl.SetOnline 1
```

<ImplVar>.GetFile

Usage

Returns the name of In-Sight active job file.

Syntax

<ImplVar>.GetFile

Return value: Job name is returned as a character string.

Description

The file name is acquired as a character string.
Active jobs need to be saved.

Example

```
    Dim caoCtrl as Object
    Dim strFileName as String

    caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202")
    strFileName = caoCtrl.GetFile
```

<ImplVar>.SetJob

Usage

Loads a job from one of the job slots in the In-Sight flash memory and makes it active.

Syntax

<ImplVar>.SetJob <Job ID>

Argument: <Job ID> Specify the job ID number with an integer. (Integer 0 to 999)

Description

A job ID number is used to load a job from one of the job slots in the In-Sight flash memory and make the job active.
The job to be loaded needs to be saved with a prefix number between 0 and 999.
In-Sight needs to be in the offline state.

Example

```
Dim caoCtrl as Object

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202")
caoCtrl.SetOnline 0
caoCtrl.SetJob 12
caoCtrl.SetOnline 1
```

<ImplVar>.StoreJob

Usage Saves the current job in the specified job slot within the In-Sight flash memory.

Syntax **<ImplVar>.StoreJob** <Job ID>, <Job name>

Argument: <Job ID> Job ID number (Integer 0 to 999)
<Job name> Job name (character string)

Description The current job is saved in the specified job slot within the In-Sight flash memory. In-Sight needs to be in the offline state.

Example

```
Dim caoCtrl as Object

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202")
caoCtrl.SetOnline 0
caoCtrl.StoreJob 3, "Test.job"
caoCtrl.SetOnline 1
```

<ImplVar>.DeleteJob

Usage

Deletes a job from the specified job slot within the In-Sight flash memory.

Syntax

<ImplVar>.DeleteJob <Job ID>

Argument: <Job ID> Job ID number (Integer 0 to 999)

Description

A job is deleted from the job slot specified by a job ID number.
In-Sight needs to be in the offline state.

Example

```
        Dim caoCtrl as Object

        caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202")
        caoCtrl.SetOnline 0
        caoCtrl.DeleteJob 3
        caoCtrl.SetOnline 1
```

<ImplVar>.GetJob

Usage

Returns the active job ID of the In-Sight.

Syntax

<ImplVar>.GetJob

Return value: Job ID is returned as an integer.

Description

To use the function of job ID number, the job to be loaded needs to be saved with a prefix number between 0 and 999.
An active job needs to be saved with a prefix number to execute this command correctly.

Example

```
        Dim caoCtrl as Object
        Dim iResult as Integer

        caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202")
        iResult = caoCtrl.GetJob
```

<ImplVar>.GetValue

Usage

Returns a value contained in the specified cell.

Syntax

<ImplVar>.GetValue(<Column name>, <Row number>**)**

Argument: <Column name> Specify with a character string (A to Z).
<Row number> Specify with an integer (0 to 399).
Return value: Cell value is returned as a character string.

Description

When In-Sight cell contains a numeric value, a floating point formatted to the third decimal place is returned regardless of the value type (either an integer or a floating point). If a cell contains characters like a structure which cannot be printed, a sharp (#) symbol will be sent instead. A null character will be sent for a blank cell.

Example

```
Dim caoCtrl as Object
Dim strResult as String

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202")
strResult = caoCtrl.GetValue("B", 14)
```

<ImplVar>.SetInteger

Usage

Sets the control contained in a cell to the specified integer. The control should be EditInt, CheckBox or ListBox type.

Syntax

<ImplVar>.SetInteger <Column name>, <Row number>, <Set value>

Argument: <Column name> Specify with a character string (A to Z).
<Row number> Specify with an integer (0 to 399).
<Set value> Specify with an integer.

Description

The value of the cell specified by <Column name> and <Row number> is set as <Set value>.
In-Sight needs to be in the offline state.

Example

```
Dim caoCtrl as Object

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202")
caoCtrl.SetInteger "B", 14, 200
```

<ImplVar>.SetFloat

Usage

Sets the edit box control stored in a cell to the specified floating point. The edit box control must be EditFloat type.

Syntax

<ImplVar>.SetFloat <Column name>, <Row number>, <Set value>

Argument: <Column name> Specify with a character string (A to Z).
<Row number> Specify with an integer (0 to 399).
<Set value> Specify with a floating point.

Description

The value of the cell specified by <Column name> and <Row number> is set as <Set value>.
In-Sight needs to be in the offline state.

Example

```
Dim caoCtrl as Object

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202")
caoCtrl.SetInteger "B", 14, 12.56
```

<ImplVar>.SetString

Usage

Sets the edit box control stored in a cell to the specified character string. The edit box control must be EditString type.

Syntax

<ImplVar>.SetString <Column name>, <Row number>, <Set value>

Argument: <Column name> Specify with a character string (A to Z).
<Row number> Specify with an integer (0 to 399).
<Set value> Specify with a character string.

Description

The value of the cell specified by <Column name> and <Row number> is set as <Set value>.
In-Sight needs to be in the offline state.

Example

```
Dim caoCtrl as Object

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202")
caoCtrl.SetString "B", 14, "Code"
```

<ImplVar>.GetInfo

Usage Returns In-Sight sensor information.

Syntax **<ImplVar>.GetInfo**

Return value: System information

Description In-Sight system information is acquired. The result will be stored in the order of the serial number, application version, monitor version, MAC address and build date.

Example

```
Dim caoCtrl as Object
Dim vntResult as Variant

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202")
vntResult = caoCtrl.GetInfo
```

<ImplVar>.StoreSetteings

Usage Saves the In-Sight system settings in the proc.set file.

Syntax **<ImplVar>.StoreSettings**

Description The system settings are saved.

Example

```
Dim caoCtrl as Object

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202")
caoCtrl.StoreSetteings
```

<ImplVar>.SetIPLock

Usage Prevents the IP address of In-Sight from being changed without permission.

Syntax **<ImplVar>.SetIPLock** <Access settings>

Argument: <Access settings>

0	Access to IP address enabled
1	Access to IP address disabled

Description When "Access to IP address disabled" is set, a change made in In-Sight Explorer to IP address by a user with access right of "Protected" or "Locked" will not be saved. When access to IP address is disabled, a user with access right of "Protected" or "Locked" can log on to In-Sight, but cannot change the IP address.

Example

```
Dim caoCtrl as Object

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202")
caoCtrl.SetIPLock 1
```

<ImplVar>.GetIPLock

Usage Returns the IP address security status on the In-Sight sensor.

Syntax **<ImplVar>.GetIPLock**

Return value:	0	Access to IP address available
	1	Access to IP address unavailable

Description IP address status is returned as an integer.。

Example

```
Dim caoCtrl as Object
Dim iResult as Integer

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202")
iResult = caoCtrl.GetIPLock
```

<ImplVar>.SetOnline

Usage

Sets the In-Sight sensor to online or offline.

Syntax

<ImplVar>.SetOnline <Online state>

Argument: <Online state>

0	In-Sight set to offline
1	In-Sight set to online

Description

In-Sight online/offline state is set with an integer.

Example

```
Dim caoCtrl as Object

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202")
caoCtrl.SetOnline 0
```

<ImplVar>.GetOnline

Usage Returns the online state of In-Sight sensor.

Syntax **<ImplVar>.GetOnline**

Return value:	0	In-Sight currently in offline mode
	1	In-Sight currently in online mode

Description In-Sight online/offline state is returned with an integer.

Example

```
Dim caoCtrl as Object
Dim iResult as Integer

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202")
iResult = caoCtrl.GetOnline
```

<ImplVar>.SetEvent

Usage Sets a trigger on the specified event on In-Sight.

Syntax **<ImplVar>.SetEvent** <Event code>

Argument: <Event code>

0 to 7	Sets software trigger.
8	Updates image acquisition and spreadsheet. To use this option, the argument that triggers the AcquireImage function needs to be set on a camera, external trigger or manual trigger.

Description Software trigger is set on In-Sight. Specify the event code with an integer.
In-Sight needs to be in the online state.

Example

```
Dim caoCtrl as Object
Dim strResult as String

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202")
caoCtrl.SetEvent 8
delay 1000
strResult = caoCtrl.GetValue("C", 13)
```

<ImplVar>.SetEventAndWait

Usage

Executes the specified event to the In-Sight sensor and returns a response after spreadsheet update is completed.

Syntax

<ImplVar>.SetEventAndWait <Event code>

Argument: <Event code>

0 to 7	Sets software trigger.
8	Updates image acquisition and spreadsheet. To use this option, the argument that triggers the AcquireImage function needs to be set on a camera, external trigger or manual trigger.

Return value: None

Description

Software trigger is set on In-Sight. Specify the event code with an integer.
In-Sight needs to be in the online state.

Example

```
Dim caoCtrl as Object
Dim strResult as String

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202")
caoCtrl.SetEventAndWait 8
strResult = caoCtrl.GetValue("C", 13)
```

<ImplVar>.SendMessage

Usage Sends an ASCII character string to the In-Sight spreadsheet through connection to the native mode. Trigger can be set on a spreadsheet event.

Syntax **<ImplVar>.SendMessage** <Set character string>, [<Event code>]

Argument: <Set character string> Specify the character string to be set.
<Event code> Same as SetEvent and SetEventAndWait.
This can be omitted.

Description A character string is sent to the ReadMessage function on the spreadsheet. Event trigger can be set by specifying the event code.
In-Sight needs to be in the online state.

Example

```
Dim caoCtrl as Object
Dim strResult as String

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202")
caoCtrl.SendMessage "Pattern2", 8
delay 1000
strResult = caoCtrl.GetValue("C", 13)
```

<ImplVar>.GetFilelist

Usage Returns the list of file names stored in the In-Sight memory.

Syntax **<ImplVar>.GetFilelist**

Return value: A file list consisting of character strings

Description Returns the list of file names stored in the In-Sight memory. In-Sight needs to be in the online state.

Example

```
	Dim caoCtrl as Object
	Dim vntResult as Variant

	caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202")
	vntResult = caoCtrl.GetFilelist
```

<ImplVar>.NativeMode

Usage

Executes a specified native mode command. A command response is acquired regardless of whether the command execution is successful or not.

Syntax

<ImplVar>.NativeMode (<Native mode command>, [<Option>] **)**

Argument: <Native mode command> Specify a command with a character string.
[<Option>] Specify a type of return value.
BSTR=False : A byte array is returned. (default setting)
BSTR=True : A character string is returned.

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

Description

A specified native mode command is executed. For supported native mode commands, refer to the reference manuals for Cognex In-Sight Explorer.

Example

```
Dim caoCtrl as Object
Dim vntResult as Variant

caoCtrl=cao.AddController("InSight","CaoProv. Cognex.In-Sight", "", "conn=eth:192.168.0.202")
vntResult = caoCtrl.NativeMode("GF", "BSTR=True")
```

<ImplVar>.SendMessageAndWait

Usage

Receives a character string output from the WriteMessage function of In-Sight after issuing the SendMessage command.

Syntax

<ImplVar>.SendMessageAndWait ([<Set character string>]
, [<Event code>]
, [<Terminal symbol number>] **)**

Argument:	<Set character string>	When omitted, a blank character is sent.
	<Event code>	Same as SetEvent and SetEventAndWait. When omitted, this will be set to 8.
	<Terminal symbol number>	0: None, 1: Carriage return (CR), 2: Line feed (LF), 3: CR+LF When omitted, this will be set to 1.

Return value: A character string output from In-Sight

Description

The system waits to receive a character string output by the WriteMessage function after executing the SendMessage command. If the character string sent by the WriteMessage function does not contain a terminal symbol, the command waits until the timeout period passes and then returns as much of character string that could be received. When the WriteMessage function is not set, a timeout will occur.
In-Sight needs to be in the online state.

Example

```
        Dim caoCtrl as Object
        Dim strResult as String

        caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202")
        strResult = caoCtrl.SendMessageAndWait( "Pattern2", 8, 1 )
```

<ImplVar>.Execute("GetMessageEX")

Usage

Receives a character string output from the WriteMessage function of In-Sight.

Syntax

<ImplVar>.Execute("GetMessageEX ", [<Terminal symbol number>] **)**

Argument: <Terminal symbol number>	0: None, 1: Carriage return (CR), 2: Line feed (LF), 3: CR+LF When omitted, this will be set to 1.

Return value: A character string output from In-Sight

Description

A character string output from the WriteMessage function of In-Sight is received.
If the character string sent by the WriteMessage function does not contain a terminal symbol, the command waits until the timeout period passes and then returns as much of character string that could be received. When the WriteMessage function is not set, a timeout will occur.
In-Sight needs to be in the online state.

Example

```
        Dim caoCtrl as Object
        Dim strResult as String

        caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202")
        strResult = caoCtrl.Execute("GetMessageEX")
```

<ImplVar>.SetTimeoutNM

Usage Changes the timeout period for communications in Native Mode.

Syntax **<ImplVar>.SetTimeoutNM** [<Timeout period>]

Argument: <Timeout period> Timeout period to set (msec)

Return value: None

Description The timeout period for communications in Native Mode is changed.
The initial value is set in the AddController option, Timeout.

In-Sight needs to be in the online state.

Example

```
Dim caoCtrl as Object

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202")
caoCtrl.SetTimeoutNM 1000
```

<ImplVar>.GetTimeoutNM

Usage Acquires the timeout period for communications in Native Mode.

Syntax **<ImplVar>.GetTimeoutNM**

Argument: None

Return value: Timeout period (msec)

Description The timeout period for communications in Native Mode is acquired.

In-Sight needs to be in the online state.

Example
```
        Dim caoCtrl as Object
        Dim iTimeout as Integer

        caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202")
        iTimeout = caoCtrl.GetTimeoutNM
```

<ImplVar>.SendMessageAndGetEZ

Usage

Receives a character string output via communications of EasyBuilder (TCP/IP) after issuing the SendMessage command. For the EasyBuilder settings, refer to Sections 3.2.4 and 3.2.5.

Syntax

<ImplVar>.SendMessageAndGetEZ ([<Set character string>]
, [<Event code>]
, [<Timeout period>] **)**

Argument:	<Set character string>	When omitted, a blank character is sent.
	<Event code>	Same as SetEvent and SetEventAndWait. When omitted, this will be set to 8.
	<Timeout period>	Maximum period in which the system waits to receive data (msec) If omitted, the system waits for 500 msec.

Return value: A character string output from In-Sight

Description

A character string specified in the EasyBuilder communication settings is received after executing SendMessage. If the communication settings are not done or the reception wait time is exceeded, a timeout will occur.
In-Sight needs to be in the online state.
If EZPort option is not specified by AddController, an error of E_EZERROR will occur.

Example

```
Dim caoCtrl as Object
Dim strResult as String

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202,
                    Timeout=1000, EZPort=3000")
strResult = caoCtrl.SendMessageAndGetEZ( "Pattern2", 8, 1000 )
```

<ImplVar>.SendMessageEZ

Usage

Prepares to receive a character string output via communications of EasyBuilder (TCP/IP) after issuing the SendMessage command.

Syntax

<ImplVar>.SendMessageEZ [<Set character string>], [<Event code>]

Argument:	<Set character string>	When omitted, a blank character is sent.
	<Event code>	Same as SetEvent and SetEventAndWait. When omitted, this will be set to 8.

Return value: None

Description

The system prepares to receive a character string output via communications of EasyBuilder (TCP/IP) after issuing the SendMessage command. Use the GetEZ command to receive a character string.
In-Sight needs to be in the online state.

Example

```
Dim caoCtrl as Object
Dim strResult as String

caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202,
                    Timeout=1000, EZPort=3000")
caoCtrl.SendMessageEZ "Pattern2", 8
```

<ImplVar>.GetEZ

Usage

Receives a character string output via communications of EasyBuilder (TCP/IP).

Syntax

<ImplVar>.GetEZ([<Timeout period>] **)**

Argument: <Timeout period> Maximum period in which the system waits to receive data (msec)
If omitted, the system waits for 500 msec.

Return value: A character string output from In-Sight

Description

A character string output via communications of EasyBuilder (TCP/IP) is received.
To receive the result using the GetEZ command, set a trigger using the SendMessageEZ command.
In-Sight needs to be in the online state.
If EZPort option is not specified by AddController, an error of E_EZERROR will occur.

Example

```
    Dim caoCtrl as Object
    Dim strResult as String

    caoCtrl = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "conn=eth:192.168.0.202,
                    Timeout=1000, EZPort=3000")
    strResult = caoCtrl.GetEZ(1000)
```

6. Error code of In-Sight provider

In the In-Sight provider, specific error codes shown below are designated. About the ORiN2 commonness error, please refer to the chapter of the error code of "ORiN2 Programming guide".

Error name	Error code	Explanation
E_INSIGHTERROR	0x80100010 ～ 0x80100019	In-Sight Error
E_INVALIDPACKET	0x80100020	Received invalid packet
E_CONNECTION	0x80100021	Communication disconnected
E_INVALIDPASSWORD	0x80100022	Invalid user name / password
E_UNPREPAREDFORIMAGE	0x80100023	Preparation for image acquisition is not finished.
E_EZERROR	0x80100024	A port for result acquisition of EZBuilder is not specified.

For the status code of the specific error code, an error code is created based on the return value as shown below. The return value of the status code from In-Sight is returned as HRESULT.
0x80100010 + Absolute value of the returned value (returned value: 0 to -9)

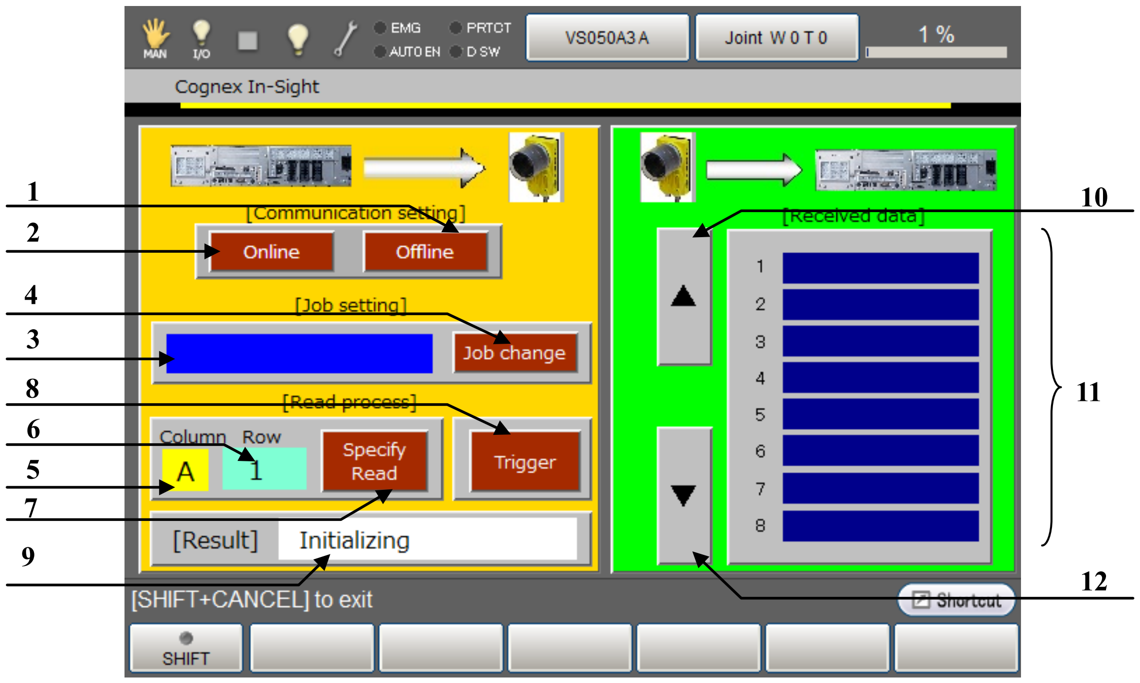
For respective error code of each command, refer to [Native mode communications] of the In-Sight Explorer Reference of Cognex.

Example: Error code "0x80100012" is issued when executing SetEvent.
Status code : -2
Description : Fail to command execution or system is off-line.

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

[Main screen]



Description Each button functions as follows.

1. Switches to the offline state. (SetOnline 0)
2. Switches to the online state. (SetOnline 1)
3. A field for setting a job file name for change. The currently set job name is displayed by default.
4. Changes to the job file name specified in (3). (LoadFile)
5. A field for setting the cell column for reading the specified cell data from In-Sight. Range: A to Z
6. A field for setting the cell row for reading the specified cell data from In-Sight Range: 0 to 399
7. Sends the column and row data set in (5) and (6) to In-Sight and receives the data of the specified cell, and then displays it in the data display section (11). (GetValue)
8. Sends a trigger for image acquisition (SetEvent 8) to In-Sight.
9. Displays the processing result.
10. Moves up the page displayed for received data.
11. Moves down the page displayed for received data.

Note: When provider implementation (initialization) is done correctly, "Connected" will be shown in the processing result field (9). An active job file name on In-Sight will be shown in the job field (3).

8. Sample Program

```
Sub Main
  On Error Goto ErrProc                                'Declare error process routine

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

  takearm keep = 0

  pTargetPos = P11

  caoCognex = cao.AddController("InSight", "CaoProv.Cognex.In-Sight", "", "Conn=eth:127.0.0.1,Timeout =
  1000")                         'Provider implementation
      caoCognex.SetOnline 0                            'Go to offline state
      caoCognex.LoadFile "Check1.job"                  'Switch to Check1.job file
      caoCognex.SetOnline 1                            'Go to online state
      caoCognex.SetEventAndWait 8                      'Trigger -> wait for process
      strResult = caoCognex.GetValue("C", 31)          'Acquire C31 value on spreadsheet
  letx pTargetPos = posx(P11) + val(strResult)         'Expand X component of received data to position data
      strResult = caoCognex.GetValue("D", 31)          'Acquire D31 value on spreadsheet
      lety pTargetPos = posy(P11) + val(strResult)     'Expand Y 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
      hand[0].Chuck 0
      depart l, @p 50, s = 100

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

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

End Sub
```

* As well as receiving the data from In-Sight by specifying the cell to read the data as previously described, it is also possible to make In-Sight execute the trigger, image processing and data transmission in bulk. Locating the ReadMessage function on the spreadsheet allows the use of SendMessageAndWait command which can simplify the process. Refer to the SendMessage function described in the COGNEX In-Sight user’s manual for details about In-Sight settings.

```
strResult = caoCognex.SendMessageAndWait 8                  ‘Receipt from trigger executed in one line
 (ReadMessage and WriteMessage functions are required on the spreadsheet.)
```

```
caoCognex.SendMessage "Trig1", 8                            ‘Trigger only
delay 1000
strResult = caoCognex.GetValue("D", 33)             ‘Read D33 cell
 (ReadMessage function is required on the spreadsheet.)
```

Revision History

DENSO Robot
Provider
User's Manual
COGNEX Vision Sensor In-Sight 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 "GetMessage" Addition of argument "SendMessageAndWait"
Ver.1.0.2	Ver.1.4.* and later	Support of EasyBuilder, addition of commands SendMessageAndGetEZ, SendMessageEZ, GetEZ, SetTimeoutNM, GetTimeoutNM Correction of command "AddController"
Ver.1.0.3	Ver.1.5.* and later	Modify the command name from "GetMessage" to "GetMessageEX".
Ver.1.0.4	Ver.1.12.* and later	Add an error code
Ver.1.0.5		Modify the output result setting

DENSO WAVE INCORPORATED

DENSO Robotics

THIRD PARTY PRODUCTS

DENSO WAVE INCORPORATED