XGX provider
KEYENCE XG-X2000 series provider

Version 1.0.2

User’s guide

August 29, 2019

[Remarks]

[Revision history]

Version	Date	Content
1.0.0	2016-06-17	First edition
1.0.1	2016-07-12	Added an argument (bIsScalar) to WriteCharReg command.
1.0.2	2017-02-10	Added the following 34 commands. AutoTuning RegisterCharLibrary DeleteCharLibrary ReadRegisterImageNo UpdateRegisterImageNo WriteRegisterImageNo RenameInspectionSetting ReturnFlowTop UpdatePosAdjustment Reset ReCalcImageInfo SaveSetting ExecuteTeaching CancelWaitStatus InputSimConsole SearchUnitNo ReadVersionInfo GetIntVariable PutIntVariableEx PutIntVariable GetVariable PutVariableEx PutVariable GetTerminalVariable PutTerminalOffset PutTerminalVariable ReadMode CopyRecipe MoveRecipe RenameRecipe ReadRecipe

		ReadRecipeString ChangeRecipeString ChangeRecipe
1.0.2	2019-08-29	Added precautions for the use of asynchronous command.

Contents

1. Introduction

This is a user’s guide of XGX provider that is a CAO provider for the KEYENCE Vision system XG-X2000 series. XGX provider enables to execute commands to Ethernet-connected or RS-232C-connected XG-X2000 series controller with non-procedure method and then notifies the execution results.

This document describes the functions and the implemented methods of XGX provider.

1.1. Device Settings

From the [環境設定], point to [入出力設定] and then select a communication method for setup. This provider supports RS-232C and Ethernet..

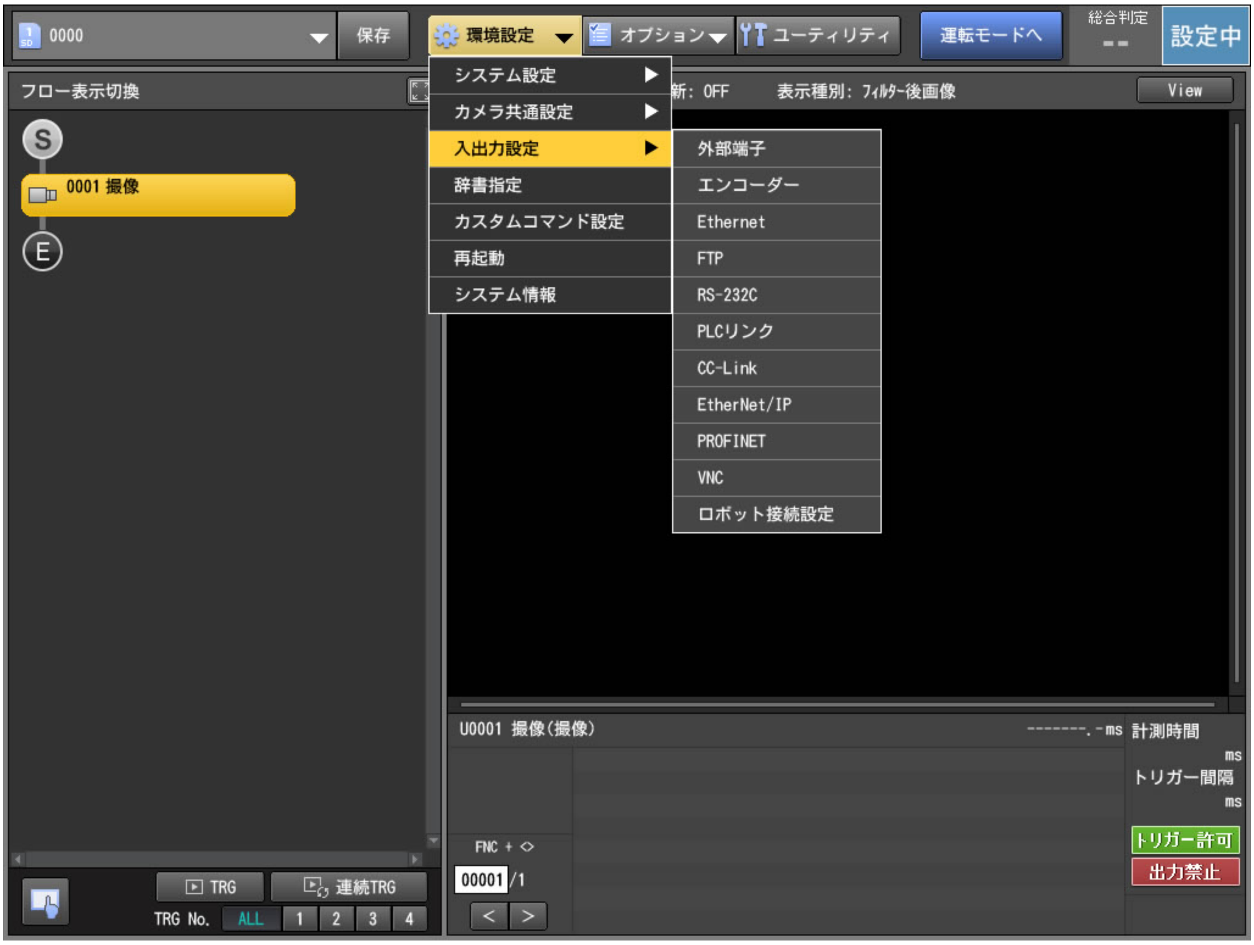
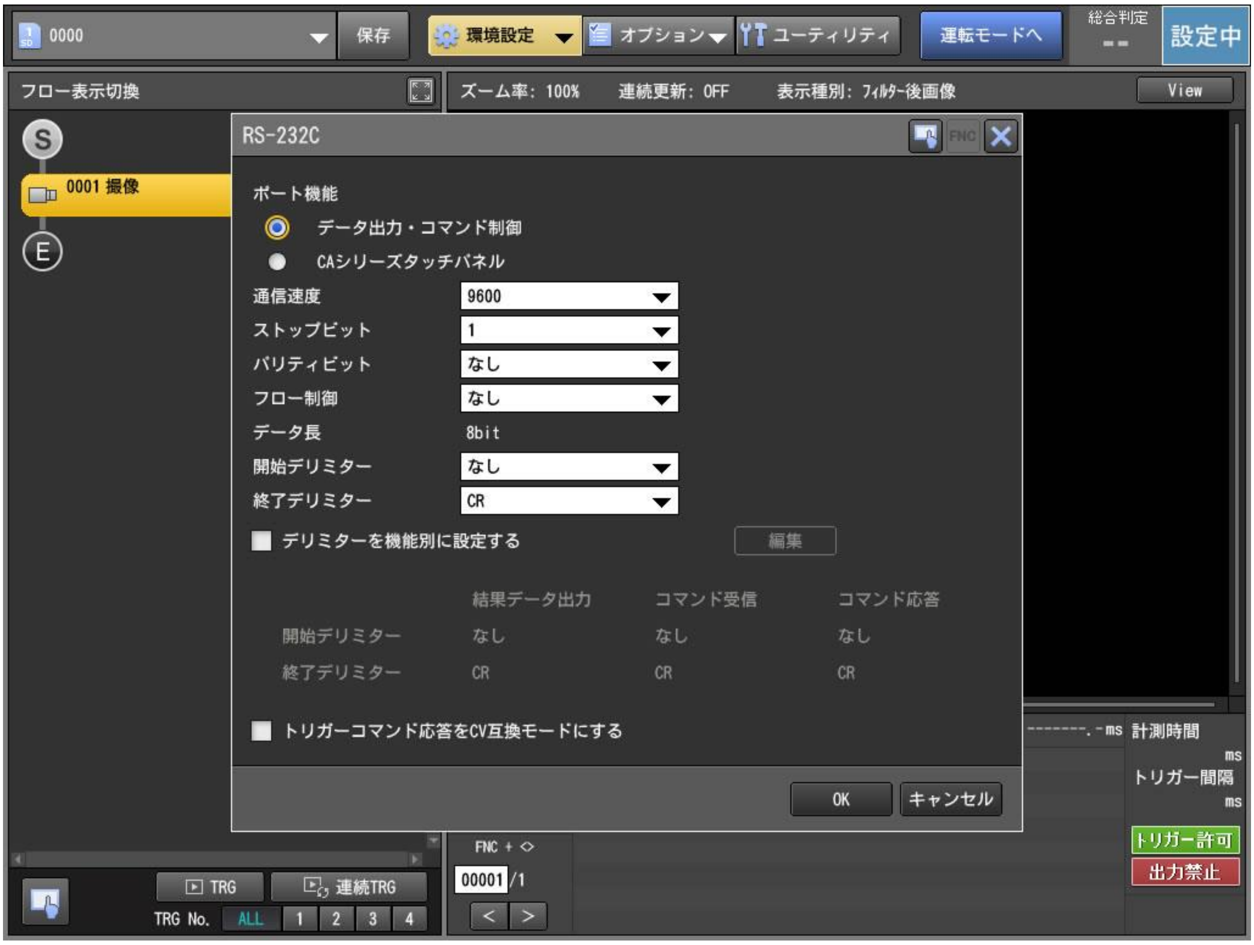


Figure 1-1 Environment setting

1.1.1. Communication setting for RS-232C

In the communication settings for RS-232C, you can change items other than delimiter. If you change any items from the default settings, please change the connection parameters of AddController as well (see 2.2.1.1). For “Start delimiter”, select “None" and for “End delimiter” select "CR".



RS-232C

ポート機能

● データ出力・コマンド制御

○ CAシリーズタッチパネル

通信速度	9600
ストップビット	1
パリティビット	なし
フロー制御	なし
データ長	8bit
開始デリミター	なし
終了デリミター	CR

□ デリミターを機能別に設定する 編集

	結果データ出力	コマンド受信	コマンド応答
開始デリミター	なし	なし	なし
終了デリミター	CR	CR	CR

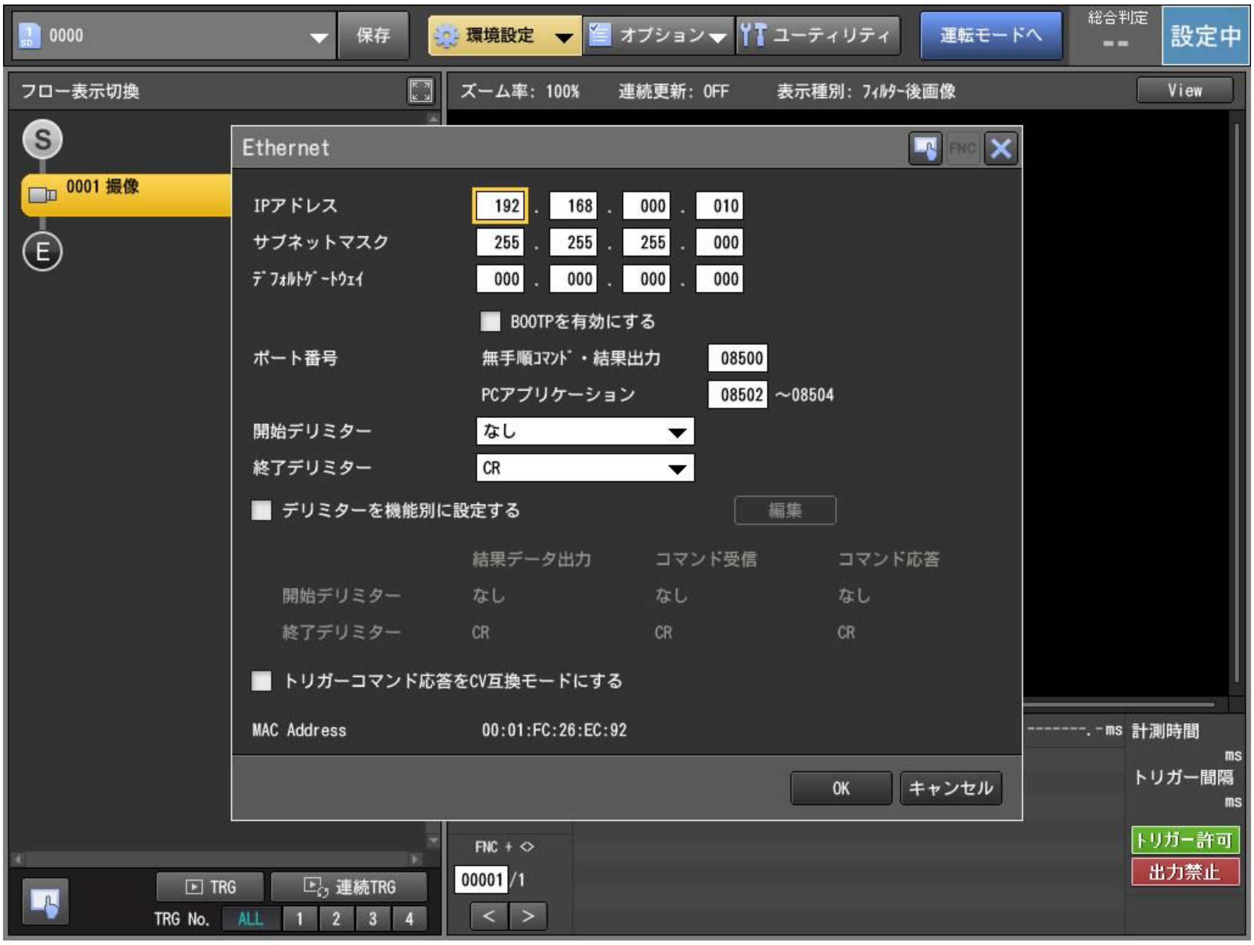
□ トリガーコマンド応答をCV互換モードにする

OK キャンセル

Figure 1-2 Settings for RS-232C

1.1.2. Communication setting for Ethernet

In the communication settings for Ethernet, you can change items other than delimiter. If you change any items from the default settings, please change the connection parameters of AddController as well (see 2.2.1.1). For “Start Delimiter”, select "None", and for “End Delimiter” select "CR".



Ethernet

IPアドレス 192 . 168 . 000 . 010

サブネットマスク 255 . 255 . 255 . 000

ﾃﾞﾌｫﾙﾄｹﾞｰﾄｳｪｲ 000 . 000 . 000 . 000

□ BOOTPを有効にする

ポート番号 無手順ｺﾏﾝﾄﾞ・結果出力 08500

PCアプリケーション 08502 ～08504

開始デリミター なし

終了デリミター CR

□ デリミターを機能別に設定する 編集

	結果データ出力	コマンド受信	コマンド応答
開始デリミター	なし	なし	なし
終了デリミター	CR	CR	CR

□ トリガーコマンド応答をCV互換モードにする

MAC Address 00:01:FC:26:EC:92

OK キャンセル

Figure 1-3 Settings for Ethernet

2. Outline of provider

2.1. Outline

XGX provider provides CaoController::Execute method for a command execution.

CaoController::Execute uses serial interface to send and receive commands.

Table 2-1 XGX provider

File name	CaoProvKEYENCEXGX.dll
ProgID	CaoProv.KEYENCE.XGX
Registration[1]	regsvr32 CaoProvKEYENCEXGX.dll
Deregistration	regsvr32 /u CaoProvKEYENCEXGX.dll

2.2. Method and Properties

2.2.1. CaoWorkspace::AddController method

XGX provider establishes communication by referring to the connection parameters for communication when AddController is executed. Also, communication method is specified as option.

Syntax AddController(<bstrCtrlName:VT_BSTR>, <bstrProvName:VT_BSTR>,
<bstrPcName:VT_BSTR> [, <bstrOption:VT_BSTR>])

bstrCtrlName : [in] Controller name

bstrProvName : [in] Provider name. Fixed to = ” CaoProv.KEYENCE.XGX”

bstrPcName : [in] Computer name where provider runs (any)

bstrOption : [in] Option character strings

The following shows a list of option character string items.

Table 2-2 Option character string of CaoWorkspace::AddController

Option	Description
Conn =<Connection parameter>	Required. Specify the communication configuration and connection parameters. For details, refer to 2.2.1.1.
Timeout[=<Timeout period>]	Specify a timeout period (millisecond) at the receiving and sending. (Default: 500)

[1] If the registry is installed by ORiN2 SDK, you do not need to register /delete it manually.

2.2.1.1. Conn option

The following shows connection parameter strings for Conn option. Items enclosed with square brackets (“[]”) are omittable. Underlined part shows the default value when the option is not specified.

- Ethernet device

"eth:<IP Address> [:<Port No>]"

<IP Address> : IP address of XGX series to connect
Example: ”192.168.0.10”

<Port No> : Connection port number
8500, 8501, . . . any number

Example

```
        Dim caoEng as CaoEngine
        Dim caoCtrl as CaoController

        Set caoEng  = New caoEngine
        Set caoCtrl = caoEng.Workspaces(0).AddController("XGX",  "CaoProv.KEYENCE.XGX",  "",
      "conn=eth:192.168.0.1,  timeout=800")
```

- RS232C device

"Conn=com:<COM Port>[:<BaudRate>[:<Parity>:<DataBits>:<StopBits>[:<Flow>]]]"

<COM Port> : COM port number
‘1’-COM1, ‘2’-COM2, …

<BaudRate> : Baud rate
4800, 9600, 19200, 38400, 57600, 115200.

<Parity> : Parity
‘N’-NONE, ‘E’-EVEN, ‘O’-ODD.

<DataBits> : Data bit
‘7’-7bit, ‘8’-8bit.

<StopBits> : Stop bit
‘1’-1bit, ‘2’-2bit.

<Flow> : Flow control
‘0’- without flow control, ‘1’-Xon/Xoff, ‘2’-Harware control.
You can specify it by taking OR.

Example

```
          Dim caoEng as CaoEngine
          Dim caoCtrl as CaoController

          Set caoEng  = New caoEngine
          Set caoCtrl = caoEng.Workspaces(0).AddController(″XGX″,  ″CaoProv.KEYENCE.XGX″,  ″″,
        ″conn=com:1″)
```

2.2.2. CaoController::Execute method

XG-X provider sends and receives commands with non-procedure methods through RS-232C or Ethernet. Enter a command name in the first argument, a command parameter in the second argument. For details about each command, refer to Command reference.

Syntax Execute (<bstrCommandName:VT_BSTR>, [<vntParam : VT_VARIANT>])

bstrCommandName : [in] Command name

vntParam : [in] Parameter

2.2.3. Error code

The processing result from the XGX series at the method execution is returned as HRESULT. XGX series-original errors are masked with 0x80108000 and returned. For details about XGS series-original errors, refer to XGX series user’s manual.

Process completes normally (OK) : S_OK (0)

Process completes with errors (ER) : 0x80108000 + Return value

Example: Executing ChangeMode.

hr = 0x80108016 : Incorrect parameter is assigned.

For details about other provider-common errors, please refer to ORiN2 SDK Programmers User’s Guide.

Table 2-3 Error code list

Error name	Error number	Description
E_XGXERROR_XGXERR	0x80108000 | XGX error	XGX series original error (see 2.2.3)
E_XGXERROR_LENGTH	0x80100000	Packet length error
E_XGXERROR_PACKET	0x80100001	Abnormal packet error
E_COMMAND_EXECUTING	0x80100002	Another command run during command execution.
E_GET_COMMAND_RESULT	0x80100003	GetCommandResult command was executed after any synchronous command execution.

3. Command reference

This chapter describes each command of CaoController::Execute method. For operation details of each command, refer to Communication command reference in XG-X2000 series communication control manual from KEYENCE.

Table 3-1 CaoController::Execute command list

XGX series Non-procedure command	Command	Function	
Trigger			
T1, T2, T3, T4, TA	Trigger	Input a trigger.	P. 15
Mode control			
R0, S0	ChangeMode	Change operation mode.	P. 15
	ChangeModeAsync	Change operation mode asynchronously. (not recommended)	P. 16
RM	ReadMode	Read the current operation mode (Run mode or Setup mode)	P. 16
Program number control			
PW	ChangeInspectSetting	Load the specified program from the specified SD.	P. 17
	ChangeInspectSettingAsync	Load the specified program from the specified SD asynchronously. (not recommended)	P. 17
	ChangeInspectSettingString	Switch to the specified name’s program.	P. 18
	ChangeInspectSettingStringAsync	Switch to the specified name’s program asynchronously. (not recommended)	P. 18
PR	ReadInspectSetting	Read the currently loaded program number.	P. 19
	ReadInspectSettingString	Read the currently loaded program name.	P. 19
System control			
CE	ClearError	Clear the error.	P. 20
NR	ReadRegisterImageNo	Return the registered image number currently used for measurement.	P. 20
NU	UpdateRegisterImageNo	Update the registered image number for a specified unit that references a variable to change the registered image.	P. 21
NW	WriteRegisterImageNo	Switch the registered image number for the specified unit.	P. 21
PN	RenameInspectionSetting	Change the name of the specified program number.	P. 22

RE	ReturnFlowTop	Return to the top of the flowchart process	P. 22
RR	UpdatePosAdjustment	Load the latest value currently referenced by a specified position adjustment unit.	P. 23
RS	Reset	Reset	P. 24
RU	ReCalcImageInfo	Recalculate the reference image information.	P. 24
SS	SaveSetting	Save current inspection settings and others.	P. 25
TG	ExecuteTeaching	Perform teaching for a specified calibration unit	P. 25
WG	CancelWaitStatus	Cancel the waiting status.	P. 25
OCR・2Dcode reader・1Dcode reader-related command			
CW	WriteCharReg	Change the REG.	P. 26
CR	ReadCharReg	Read the REG.	P. 27
AT	AutoTuning	Automatic tuning	P. 27
CA	RegisterCharLibrary	Register character to library	P. 28
CD	DeleteCharLibrary	Delete character from library	P. 29
Data I/O-related function			
IR	GetIntVariable	Integer read from variable	P. 29
IS	PutIntVariableEx	Simultaneous integer write to variable	P. 30
IW	PutIntVariable	Integer write to variable	P. 31
MR	GetVariable	Read from variable	P. 31
MS	PutVariableEx	Simultaneous write to variable	P. 32
MW	PutVariable	Write to variable	P. 33
RP	GetTerminalVariable	Read from terminal variable	P. 33
WO	PutTerminalOffset	Write terminal offset	P. 34
WP	PutTerminalVariable	Write to terminal variable	P. 34
Recipe-related function			
RPC	CopyRecipe	Copy the source recipe settings.	P. 34
RPM	MoveRecipe	Move the source recipe settings.	P. 35
RPN	RenameRecipe	Change the name of the specified recipe number.	P. 35
RPR	ReadRecipe	Read the recipe number or recipe name.	P. 36
	ReadRecipeString	Read the recipe settings	P. 36
RPW	ChangeRecipe	Change the recipe setting number by specifying a recipe setting number.	P. 36
RPT	ChangeRecipeString	Change the recipe setting number by specifying a recipe setting name	P. 37
Others			
KY	InputSimConsole	Execute console pseudo input.	P. 37
UQ	SearchUnitNo	Search the unit number	P. 38
VI	ReadVersionInfo	Read the version information	P. 38
Original expansion command			
-	ExecuteCommand	Execute a non-procedure command.	P. 39
-	ExecuteCommandAsync	Execute a non-procedure command	P. 39

		asynchronously. (not recommended)	
-	TriggerAndGetResult	Issue a trigger and receive the output result.	P. 40
-	RecievePacket	Receive a packet.	P. 40
-	ClearPacket	Clear the received packet in the reception buffer.	P. 41
-	SetTimeout	Set the timeout period.	P. 41
-	GetTimeout	Obtain the timeout period.	P. 41
-	GetCommandResult	Obtain the return value of the asynchronous command.	P. 41

3.1. Trigger

3.1.1. CaoController::Execute ("Trigger") command

This command issue a specified trigger. If the execution result is output, use ReceivePacket command to receive the result.

Syntax Trigger(< iTriggerNo >)

iTriggerNo : [in] Specify the trigger number to issue.(VT_I2)
1 to 4 : Trigger 1 to 4
-1 : All triggers

Return value : [out] none

The following example shows how to issue a trigger number 1.

Example

```
        caoCtrl.Execute "Trigger",  1
```

3.2. Mode control

3.2.1. CaoController::Execute ("ChangeMode") command

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

Syntax ChangeMode(< iMode >)

iMode : [in] Specify the operation mode (VT_UI4)
0 : Setup mode
1 : Run mode

Return value : [out] none

The following example shows how to switch to the Run mode.

Example

```
caoCtrl.Execute "ChangeMode", 1
```

3.2.2. CaoController::Execute ("ChangeModeAsync") command (not recommended)

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

To receive and confirm the return value, use GetCommandResult command. For details about GetCommandResult command, refer to 3.9.8.CaoController::Execute ("GetCommandResult") .

When you use this command, be sure to disconnect the provider after the execution of GetCommandResult command.

Otherwise, an error may occur at asynchronous operation.

Syntax ChangeModeAsync(< iMode >)

iMode	:	[in] Specify the operation mode to set (VT_UI4)
		0 : Setup mode
		1 : Run mode
Return value	:	[out] none

The following shows how to switch the operation mode to Run mode.

Example

```
Dim vntResult as Variant

caoCtrl.Execute "ChangeModeAsync", 1

‘ Obtain the return value of ChangeModeAsync command
vntResult = caoCtrl.Execute("GetCommandResult")

‘ vntResult : Return value(Empty)
```

3.2.3. CaoController::Execute ("ReadMode") command

This command reads the currently selected operation mode.

Syntax ReadMode ()

argument	:	[in] None
Return value	:	[out] Operation mode(VT_I4)
		0 : Setup mode
		1 : Run mode

The following sample shows how to obtain the currently selected operation mode.

Example

```
        Dim lMode as Long
        lMode =  caoCtrl.Execute(″ReadMode″)
```

3.3. Program number control

3.3.1. CaoController::Execute ("ChangeInspectSetting") command

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

Syntax ChangeInspectSetting(< iDriveNo >, < iSettingNo >)

iDriveNo : [in] Specify the SD card number (VT_I4)
1 : SD1
2 : SD2
iSettingNo : [in] Specify the program number (0 to 999) (VT_I4)
Return value : [out] none

The following sample shows how to load the program number 1 from the SD card (SD1).

Example

```
        Call caoCtrl.Execute(″ChangeInspectSetting″,  Array(1,  1))
```

3.3.2. CaoController::Execute ("ChangeInspectSettingAsync") command (not recommended)

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

To receive and confirm the return value, use GetCommandResult command. For details about GetCommandResult command, refer to 3.9.8.CaoController::Execute ("GetCommandResult") .

When you use this command, be sure to disconnect the provider after the execution of GetCommandResult command.

Otherwise, an error may occur at asynchronous operation.

Syntax ChangeInspectSettingAsync(< iDriveNo >, < iSettingNo >)

iDriveNo : [in] Specify the SD card number (VT_I4)
1 : SD1
2 : SD2
iSettingNo : [in] Specify the program number (0 to 999) (VT_I4)
Return value : [out] none

The following sample shows how to load the program number 1 from the SD card (SD1).

Example

```
Dim vntResult as Variant

Call caoCtrl.Execute("ChangeInspectSettingAsync",  Array(1,  1))

‘ Obtain the return value of ChangeInspectSettingAsync command.
vntResult = caoCtrl.Execute("GetCommandResult")

vntResult : Return value(Empty)
```

3.3.3. CaoController::Execute ("ChangeInspectSettingString") command

This command changes the program to the specified name’s program.

Syntax ChangeInspectSettingString(< strName >)

strName	:	[in] Program name (VT_BSTR)
Return value	:	[out] none

The following sample shows how to select the program name “test”.

Example

```
Call caoCtrl.Execute("ChangeInspectSettingString",  "test")
```

3.3.4. CaoController::Execute ("ChangeInspectSettingStringAsync") command (not recommended)

This command changes the program to the specified name’s program asynchronously.

To receive and confirm the return value, use GetCommandResult command. For details about GetCommandResult command, refer to 3.9.8.CaoController::Execute ("GetCommandResult”) .

When you use this command, be sure to disconnect the provider after the execution of GetCommandResult command.

Otherwise, an error may occur at asynchronous operation.

Syntax ChangeInspectSettingStringAsync(< strName >)

iDriveNo	:	[in] Program name(VT_BSTR)
Return value	:	[out] none

The following sample shows how to select the program name “test”.

Example

```
Dim vntResult as Variant

Call caoCtrl.Execute(″ChangeInspectSettingStringAsync″,  ″test″)

‘Obtain the return value of ChangeInspectSettingStringAsync command.
vntResult = caoCtrl.Execute(″GetCommandResult″)

vntResult ： Return value(Empty)
```

3.3.5. CaoController::Execute ("ReadInspectSetting") command

This command reads the currently loaded program number.

Syntax ReadInspectSetting( )

Argument : [in] none

Return value : [out] < iDriveNo >, < iSettingNo > (VT_I4 | VT_ARRAY)

iDriveNo : SD card number

iSettingNo : Program number

The following sample shows how to read the currently loaded program number.

Example

```
Dim vntRet as Variant

vntRet = caoCtrl.Execute(″ReadInspectSetting″)

‘ vntRet(0) ： SD card number
‘ vntRet(1) ： Program number
```

3.3.6. CaoController::Execute ("ReadInspectSettingString") command

This command reads the currently loaded program name.

Syntax ReadInspectSettingString( )

Argument : [in] none

Return value : [out] < strName > (VT_BSTR)

strName : Program name

The following sample shows how to read the currently loaded program name.

Example

```
Dim strRet as String

strRet = caoCtrl.Execute(″ReadInspectSettingString″)
```

3.4. System control

3.4.1. CaoController::Execute ("ClearError") command

This command clears the error status and error code of a specified type (0 or 1). If this command is executed while there is no errors, it completes normally.

Syntax ClearError(< iErrorType >)

iErrorType : [in] Specify the type of error. (VT_UI4)
0 : Clear "%Error0" and "%Error0Code"
1 : Clear "%Error1" and "%Error1Code".

Return value : [out] none

Example

```
        caoCtrl.Execute "ClearError", 1
```

3.4.2. CaoController::Execute ("ReadRegisterImageNo") command

This command returns the registered image number that is currently used for measurement.

To execute this command, follow the syntaxes written below.

1. To execute without specifying the image number.

Syntax ReadRegisterImageNo (< lUnitNo >)

lUnitNo : [in] Unit ID (0 to 999)(VT_I4)

Return value : [out] Registered image No. (0 to 999)(VT_I4)

2. To execute with specifying the image number

Syntax ReadRegisterImageNo (< lUnitNo >, < lImgNo >)

lUnitNo : [in] Unit ID(0 to 999)(VT_I4)

lImgNo : [in] Image number (VT_I4)
If the image operation unit ID or C-language unit ID is specified: source image number 1 or 2.
If the calibration unit ID is specified: teaching image number from "1" to "16".

Return value : [out] Registered image No. (0 to 999)(VT_I4)

The following example shows how to obtain the registered image number which is currently used by Unit ID101 for measurement.

Example

```
Dim lRegisterImageNo as Long
lRegisterImageNo = caoCtrl.Execute("ReadRegisterImageNo", 101)
```

3.4.3. CaoController::Execute ("UpdateRegisterImageNo") command

This command changes the registered image number for a specified unit that uses variable reference to change the registered image. The current value in the variable is retrieved and used to change the registered image number. If necessary, the reference image information is updated with the registered image of that number.

Syntax UpdateRegisterImageNo (< lUnitNo >)

lUnitNo	:	[in] Unit ID(VT_I4)
		0 to 999 : Unit ID
		-1 : All units
Return value	:	[out] None

The following sample shows how to refresh the variable reference of the registered image of Unit ID101 and change the registered image number.

Example

```
caoCtrl.Execute "UpdateRegisterImageNo", 101
```

3.4.4. CaoController::Execute ("WriteRegisterImageNo") command

This command switches the registered image number for the specified unit. If necessary, the reference image information is updated with the registered image of that number.

To execute this command, follow the syntaxes written below.

1. To execute without specifying the image number.

Syntax WriteRegisterImageNo (< lUnitNo >, < lRegImgNo >)

lUnitNo	:	[in] Unit ID (VT_I4)
		0 to 999 : Unit ID
		-1 : All units
lRegImgNo	:	[in] Registered image No. (0 to 999)(VT_I4)
Return value	:	[out] None

2. To execute with specifying the image number

Syntax WriteRegisterImageNo (< lUnitNo >, < lRegImgNo >, < lImgNo >)

lUnitNo	:	[in] Unit ID (VT_I4)

0 to 999 : Unit ID

-1 : All units

lRegImgNo : [in] Registered image No(0 to 999)(VT_I4)

lImgNo : [in] Image number (VT_I4)

If the image operation unit ID or C-language unit ID is specified: source image number 1 or 2.

If the calibration unit ID is specified: teaching image number from “1” to “16”.

Return value : [out] None

The following sample shows how to change the registered image number of Unit ID101 to “202”.

Example

```
Call caoCtrl.Execute(″WriteRegisterImageNo″, Array(101, 202))
```

3.4.5. CaoController::Execute ("RenameInspectionSetting") command

This command changes the name of the specified program number.

Syntax RenameInspectionSetting (< lSdNo >, < lTestNo >, < bstrParam > [, < bIsScalar >])

lSdNo : [in] SD card number(1 or 2)(VT_I4)

lTestNo : [in] inspection setting No(0 to 999) (VT_I4)

bstrParam : [in] String or Scalar array variable (VT_BSTR)

bIsScalar : [in] Specify the data type of argument of <bstrParam> (VT_BOOL)

FALSE : Treat <bstrParam> as a string. (default)

TRUE : Treat <bstrParam> as a scalar array variable.

Return value : [out] None

The following sample shows how to change the inspection setting name of the SD card number 1, inspection setting No101 into TestName.

Example

```
caoCtrl.Execute ″RenameInspectionSetting″, Array(1, 101, ″TestName″, FALSE)
```

3.4.6. CaoController::Execute ("ReturnFlowTop") command

This command makes the flowchart process jump to the unit after the start unit when the process is in a waiting status caused by a capture unit or a waiting unit other than a user menu or timer setup unit.

Syntax ReturnFlowTop ()

Argument	:	[in] None
Return value	:	[out] None

The following sample shows how to jump to the next of the start unit.

Example

```
        caoCtrl.Execute "ReturnFlowTop"
```

3.4.7. CaoController::Execute ("UpdatePosAdjustment") command

This command loads the latest value currently referenced by a specified position adjustment unit, or value of the recalculation result obtained with the registered image.

To execute this command, follow the syntaxes written below.

1. To execute the command without specifying the value to be loaded as a reference value

Syntax UpdatePosAdjustment (< lUnitNo >)

lUnitNo : [in] Unit ID (VT_I4)
0 to 999 : Unit ID
-1 : All units
Return value : [out] None

2. To execute the command with specifying the value to be loaded as a reference value

Syntax UpdatePosAdjustment (< lUnitNo >, < lStandardValue >)

lUnitNo : [in] Unit ID (VT_I4)
0 to 999 : Unit ID
-1 : All units
lStandardValue : [in] Value to be loaded as a reference value (VT_I4)
0: The latest result
1: Recalculation result of the registered image
Return value : [out] None

The following sample shows how to obtain the latest value that is currently referred by Unit ID 101 as the reference value.

Example

```
        Call caoCtrl.Execute("UpdatePosAdjustment",101)
```

3.4.8. CaoController::Execute ("Reset") command

This command performs all of the following:

- Initialize all system variables. Clear all buffers including the image buffer.
- Cancel the trigger/event waiting status of the unit.
- Create a new file name for the file used to store data.
- Initialize the user-defined local variables for which [Initialize on reset] is selected.
- Initialize the user-defined global variables for which [Initialize on reset] is selected.
- Initialize %JAHold.
- Return to the beginning of the flowchart.
- Clear all archived data.
- Clear all statistical data.
- All results of the target classification are cleared.

Syntax Reset ()

argument	:	[in] None
Return value	:	[out] None

Example

```
        caoCtrl.Execute "Reset"
```

3.4.9. CaoController::Execute ("ReCalcImageInfo") command

This command updates the reference image information of the specified unit ID with the result of the recalculation using the current registered image and setting parameters.

Syntax ReCalcImageInfo (< lUnitNo >)

lUnitNo	:	[in] Unit ID(VT_I4)
		0 to 999 : Unit ID
		-1 : All units
Return value	:	[out] None

The following example shows how to recalculate the image reference information of Unit ID101.

Example

```
        caoCtrl.Execute "ReCalcImageInfo", 101
```

3.4.10. CaoController::Execute ("SaveSetting") command

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

Syntax SaveSetting ()

argument	:	[in] None
Return value	:	[out] None

The following sample shows how to save the current settings.

Example

```
caoCtrl.Execute "SaveSetting"
```

3.4.11. CaoController::Execute ("ExecuteTeaching") command

This command conducts teaching for a specified calibration unit using the currently set registered image.

Syntax ExecuteTeaching (< lUnitNo >)

lUnitNo	:	[in] Unit ID(0 to 999) (VT_I4)
Return value	:	[out] None

The following example shows how to conduct teaching of Unit ID101.

Example

```
caoCtrl.Execute "ExecuteTeaching", 101
```

3.4.12. CaoController::Execute ("CancelWaitStatus") command

This command cancels the waiting status caused by a terminal I/O delay or variable delay unit. This command can also be issued with an input parameter to specify the desired status of result data (Judge No. and logical OR of satisfied conditions) of the unit for which the waiting status will be canceled.

· When you do not refer the result data for the unit, specify 0 as the input parameter. In this case, the waiting status will be canceled while the Judge No. and logical OR of satisfied conditions for the unit remain 0.

· The input parameter is converted into a binary number and assigned to satisfied conditions. The condition assigned to the lowest order bit among the bits containing 1 is assumed to be the Judge No.

Syntax CancelWaitStatus (< lUnitNo >)

lUnitNo : [in] Bits assigned to satisfied conditions(0 to (2^{20}-1)) (VT_I4)

Return value : [out] None

The following sample shows how to cancel the waiting status of a unit without referring to the result data.

Example

```
        caoCtrl.Execute "CancelWaitStatus", 0
```

3.5. OCR・2Dcode reader・1Dcode reader-related command

3.5.1. CaoController::Execute ("WriteCharReg") command

This command changes the REG for an OCR unit, 2D code reader, or 1D code reader. The operation varies depending on the format. See "Character Code Table for OCR Unit" (Page 2-72) for more details on the ASCII codes that can be specified on the OCR.

Syntax WriteCharReg(< lUnitNo >[, < lRowNo >, < bstrParam >])

lUnitNo : [in] Unit ID (0 to 999) (VT_I4)

lRowNo : [in] Line number (VT_I4)
For OCR unit : 1 to 2
For 1D or 2D code reader tool : 1to16

bstrParam : [in] Text or SCALAR array type variable (VT_BSTR)

Return value : [out] none

The following shows the difference of operations depending on the format.

1. Unit ID ・Line number ・Text
   Set the text (bstrParam) as the content of REG for the match condition of the Line number (lRowNo) of Unit ID (1UnitNo).
2. Unit ID ・Line number ・SCALAR array type variable
   Set the value of a SCALAR array type variable (bstrParam) with ASCII code as the content of REG for the match condition of Line number (1RowNo) in Unit ID (1UnitNo).
3. Unit ID
   Set the latest reading result of the unit with unit ID as REG for the unit. If the unit has not been measured, REG is cleared(replaced with a space in an OCR unit).

The following example show how to set the text of Line number 1 (for OCR unit) in Unit ID101 to “DEF”.

Example

```
caoCtrl.Execute "WriteCharReg",  Array(101,  1,  "DEF")
```

3.5.2. CaoController::Execute ("ReadCharReg") command

This command reads the REG for an OCR unit, 2D code reader, or 1D code reader. The operation varies depending on the format.

Syntax ReadCharReg(< lUnitNo >, < lRowNo >[, < bstrParam >])

lUnitNo	:	[in] Unit ID (0 to 999) (VT_I4)
lRowNo	:	[in] Line number (VT_I4) For OCR unit : 1 to 2 For 1D or 2D code reader tool : 1 to 16
bstrParam	:	[in] SCALAR array type variable (VT_BSTR)
Return value	:	[out] Text (VT_BSTR) or no return value (※) ※There will be no return value if “bstrParam” is specified.

The following shows the difference of operations depending on the format.

1. Unit ID・Line number

   The content of REG for the Line number (1RowNo) in Unit ID (1UnitNo) is returned as a command response.

2. Unit ID・Line number ・SCALAR array type variable

   The content of REG for the Line number (1RowNo) in Unit ID(1UnitNo) is stored as ASCII codes into the elements of a SCALAR array type variable (bstrParam) from the element specified by the index.

The following example show how to obtain the text of Line number 1 (for OCR unit) in Unit ID101.

Example

```
Dim bstrParam as String
bstrParam = caoCtrl.Execute("ReadCharReg",  Array(101,  1))
```

3.5.3. CaoController::Execute ("AutoTuning") command

This command performs automatic tuning on the specified 2D and 1D code reader units using the input image or registered image.

Syntax AutoTuning (< lUnitNo >, < lImg >)

lUnitNo	:	[in] Unit ID(0 to 999)(VT_I4)
lImg	:	[in] Tuning target image (VT_I4)

0 : Captured image

1 : Registered image

Return value : [out] None

The following shows how to perform auto tuning for the registered image of Unit ID101.

Example

```
Call caoCtrl.Execute("AutoTuning", Array(101, 1))
```

3.5.4. CaoController::Execute ("RegisterCharLibrary") command

This command registers a character specified in the latest search result, archived image, or archived results with the library as a specified registration character. To execute this command, follow the syntaxes written below.

1. To execute the command with registering the latest search result into library

Syntax RegisterCharLibrary (< lUnitNo >, < lLineNo >, < lStringNo >, < lStringKind >)

lUnitNo : [in] Unit ID(0 to 999)(VT_I4)

lLineNo : [in] Line number of the search result (1 or 2)(VT_I4)

lStringNo : [in] Character number of the search result (1 to 20)(VT_I4)

lStringKind : [in] Intended registration character type (-1 to 59)(VT_I4)

Return value : [out] None

2. To execute the command with registering the archived result into library.

Syntax RegisterCharLibrary (< lUnitNo >, < lHistory >, < lNumber >, < lLineNo >, < lStringNo >, < lStringKind >)

lUnitNo : [in] Unit ID(0 to 999)(VT_I4)

lHistory : [in] Image archive conditions (0 to 7)(VT_I4)

lNumber : [in] Go back to j-th archive from the current one (0 to (archive condition count -1))(VT_I4)

lLineNo : [in] Line number of the search result (1 or 2)(VT_I4)

lStringNo : [in] Character number of the search result (1 to 20)(VT_I4)

lStringKind : [in] Intended registration character type (-1 to 59)(VT_I4)

Return value : [out] None

The following sample shows how to register the latest search result (line number 1, character number 2, registration character type 3) of Unit ID101 into library.

Example

```
          Call caoCtrl.Execute(″RegisterCharLibrary″, Array(2,1,2,3))
```

3.5.5. CaoController::Execute ("DeleteCharLibrary") command

This command deletes the character of the last registration number for a specified registration character from the library.

Syntax DeleteCharLibrary (< lUnitNo >, < lStringKind >)

lUnitNo : [in] Unit ID(0 to 999)(VT_I4)

lStringKind : [in] Registration character type (-1 to 59)(VT_I4)

Return value : [out] None

The following sample shows how to delete the registration character type 1 of Unit ID101.

Example

```
          Call caoCtrl.Execute(″DeleteCharLibrary″, Array(101, 1))
```

3.6. Data I/O-related function

3.6.1. CaoController::Execute ("GetIntVariable") command

This command reads the value of a specified variable, rounds it off to the nearest integer, and outputs the result.

Values up to 16 variables can be read.

Syntax GetIntVariable (< bstrVar1 > [, < bstrVar2 > [, < bstrVar3 > [, < bstrVar4 > [, < bstrVar4 > [, < bstrVar5> [, < bstrVar6> [, < bstrVar7 > [, < bstrVar8> [, < bstrVar9 > [, < bstrVar10 > [, < bstrVar11 > [, < bstrVar12 > [, < bstrVar13 > [, < bstrVar14 > [, < bstrVar15 > [, < bstrVar16 >]]]]]]]]]]]]]]])

bstrVar1 : [in] Name of target variable (VT_BSTR)

bstrVar2 : [in] Name of target variable (VT_BSTR)

bstrVar3 : [in] Name of target variable (VT_BSTR)

bstrVar4 : [in] Name of target variable (VT_BSTR)

bstrVar5 : [in] Name of target variable (VT_BSTR)

bstrVar6 : [in] Name of target variable (VT_BSTR)

bstrVar7 : [in] Name of target variable (VT_BSTR)

bstrVar8 : [in] Name of target variable (VT_BSTR)

bstrVar9 : [in] Name of target variable (VT_BSTR) bstrVar10 : [in] Name of target variable (VT_BSTR)

bstrVar11 : [in] Name of target variable (VT_BSTR)

bstrVar12 : [in] Name of target variable (VT_BSTR)

bstrVar13 : [in] Name of target variable (VT_BSTR)

bstrVar14 : [in] Name of target variable (VT_BSTR)

bstrVar15 : [in] Name of target variable (VT_BSTR)

bstrVar16 : [in] Name of target variable (VT_BSTR)

Return value : [out] Read value (V_I4 | VT_ARRAY)

The following sample shows how to read values of the variable names #Data1, #Data2, and #Data3.

Example

```
Dim vntValList as VARIANT
vntValList = caoCtrl.Execute("GetIntVariable",Array("#Data1","#Data2","#Data3"))
```

3.6.2. CaoController::Execute ("PutIntVariableEx") command

This command writes a specified value for a variable and changes it when the process reaches the first capture unit of the flowchart or the end unit at the end of the flowchart.

Values of up to 16 variables can be changed simultaneously.

If the element count of the writing target variable name list and of the list of values or variable name to be written is different, an error occurs.

Syntax PutIntVariableEx(< bstrDestVarNameList > , < bstrDestVarValList >)

bstrDestVarNameList : [in] Target variable name list (VT_BSTR | VT_ARRAY)

bstrDestVarValList : [in] List of value or variable name to be written (VT_BSTR | VT_ARRAY)

Return value : [out] None

The following program shows how to write values (1, 2, 3) into variables (#Data1, #Data2, #Data3).

Example

```
Dim vntNameList as VARIANT
Dim vntValList as VARIANT
     vntNameList = Array("#Data1", "#Data2", "#Data3")
     vntValList = Array(1, 2, 3)
    Call caoCtrl.Execute("PutIntVariableEx", Array(vntNameList, vntValList))
```

3.6.3. CaoController::Execute ("PutIntVariable") command

This command rounds off a specified value to the nearest integer and writes it as the value of a specified variable. The variable is updated immediately when this command is executed.

Values of up to 16 variables can be changed simultaneously.

If the element count of the writing target variable name and the number of value or variable name to be written is different, an error occurs.

Syntax PutIntVariable (< bstrDestVarNameList > , < bstrDestVarValList >)

bstrDestVarNameList : [in] Target variable name list (VT_BSTR | VT_ARRAY)

bstrDestVarValList : [in] List of value or variable name to be written (VT_BSTR | VT_ARRAY)

Return value : [out] None

The following program shows how to write values (1, 2, 3) into variables (#Data1, #Data2, #Data3).

Example

```
Dim vntNameList as VARIANT
Dim vntValList as VARIANT
  vntNameList = Array(″#Data1″, ″#Data2″, ″#Data3″)
  vntValList = Array(1, 2, 3)
  Call caoCtrl.Execute(″PutIntVariable″, Array(vntNameList, vntValList))
```

3.6.4. CaoController::Execute ("GetVariable") command

This command reads the value of a specified scalar type variable.

Values up to 16 variables can be read simultaneously.

Syntax GetVariable (< bstrDestVar1 > [, < bstrDestVar2 > [, < bstrDestVar3 > [, < bstrDestVar4 > [, < bstrDestVar4 > [, < bstrDestVar5> [, < bstrDestVar6> [, < bstrDestVar7 > [, < bstrDestVar8> [, < bstrDestVar9 > [, < bstrDestVar10 > [, < bstrDestVar11 > [, < bstrDestVar12 > [, < bstrDestVar13 > [, < bstrDestVar14 > [, < bstrDestVar15 > [, < bstrDestVar16 >]]]]]]]]]]]]]]])

bstrDestVar1 : [in]Name of the target variable(VT_BSTR)

bstrDestVar2 : [in]Name of the target variable(VT_BSTR)

bstrDestVar3 : [in]Name of the target variable(VT_BSTR)

bstrDestVar4 : [in]Name of the target variable(VT_BSTR)

bstrDestVar5 : [in]Name of the target variable(VT_BSTR)

bstrDestVar6 : [in]Name of the target variable(VT_BSTR) bstrDestVar7 : [in]Name of the target variable(VT_BSTR)

bstrDestVar8 : [in]Name of the target variable(VT_BSTR)

bstrDestVar9 : [in]Name of the target variable(VT_BSTR)

bstrDestVar10 : [in]Name of the target variable(VT_BSTR)

bstrDestVar11 : [in]Name of the target variable(VT_BSTR)

bstrDestVar12 : [in]Name of the target variable(VT_BSTR)

bstrDestVar13 : [in]Name of the target variable(VT_BSTR)

bstrDestVar14 : [in]Name of the target variable(VT_BSTR)

bstrDestVar15 : [in]Name of the target variable(VT_BSTR)

bstrDestVar16 : [in]Name of the target variable(VT_BSTR)

Return value : [out] Value of the target variable(VT_R8 | VT_ARRAY)

The following shows how to read the numerical values of variables (#Data1, #Data2, #Data3).

Example

```
        Dim vntValList as VARIANT
        vntValList = caoCtrl.Execute("GetVariable",Array("#Data1","#Data2","#Data3"))
```

3.6.5. CaoController::Execute ("PutVariableEx") command

This command writes a specified value for a variable and changes it when the process reaches the first capture unit of the flowchart or the end unit at the end of the flowchart.

Values up to 16 variables can be changed.

If the element count of the writing target variable name list and of the list of values or variable name to be written is different, an error occurs.

Syntax PutVariableEx (< bstrDestVarNameList > , < bstrDestVarValList >)

bstrDestVarNameList : [in] Target variable name list (VT_BSTR | VT_ARRAY)

bstrDestVarValList : [in] List of value or variable name to be written (VT_BSTR | VT_ARRAY)

Return value : [out] None

The following shows how to write values (1, 2, 3) into variables (#Data1, #Data2, #Data3).

Example

```
    Dim vntNameList as VARIANT
    Dim vntValList as VARIANT
        vntNameList = Array("#Data1", "#Data2", "#Data3")
```

```
     vntValList = Array(1, 2, 3)
     Call caoCtrl.Execute("PutVariableEx", Array(vntNameList, vntValList))
```

3.6.6. CaoController::Execute ("PutVariable") command

This command changes the value of a specified scalar type variable (global or local) to a specified value. The variable is updated immediately when this command is executed.

Values up to 16 variables can be changed.

If the element count of the writing target variable name list and of the list of values or variable name to be written is different, an error occurs.

Syntax PutVariable (< bstrDestVarNameList > , < bstrDestVarValList >)

bstrDestVarNameList : [in] Target variable name list (VT_BSTR | VT_ARRAY)

bstrDestVarValList : [in] List of value or variable name to be written (VT_BSTR | VT_ARRAY)

Return value : [out] None

The following shows how to write values (1, 2, 3) into variables (#Data1, #Data2, #Data3).

Example

```
Dim vntNameList as VARIANT
Dim vntValList as VARIANT
     vntNameList = Array("#Data1", "#Data2", "#Data3")
     vntValList = Array(1, 2, 3)
     Call caoCtrl.Execute("PutVariable", Array(vntNameList, vntValList))
```

3.6.7. CaoController::Execute ("GetTerminalVariable") command

This command reads the status of a specified terminal.

Syntax GetTerminalVariable (< bstrSysVarName >)

bstrSysVarName : [in] Target system variable (VT_BSTR)
(%OutDataAsyncA to %OutDataAsyncH)

Return value : [out] Read value (VT_I4)

The following sample shows how to read the terminal status of %OutDataAsyncA.

Example

```
Dim lTerminalCond as long
    lTerminalCond = caoCtrl.Execute("GetTerminalVariable", "%OutDataAsyncA")
```

3.6.8. CaoController::Execute ("PutTerminalOffset") command

This command writes lMagOffset * lOffsetValue into %CmdParamOffset.

Syntax PutTerminalOffset (< lMagOffset > , < lOffsetValue >)

lMagOffset	:	[in] Offset scale (VT_I4)
lOffsetValue	:	[in] Offset value (VT_I4)
Return value	:	[out] None

The following sample shows how to write [offset value: 1] * [offset scale: 2] into %CmdParamOffset.

Example

```
    Call caoCtrl.Execute("PutTerminalOffset", Array(2, 1))
```

3.6.9. CaoController::Execute ("PutTerminalVariable") command

This command changes the value of the system variable which allows terminal assignment and can be written via a command. The variable is updated immediately when this command is executed.

Syntax PutTerminalVariable (< bstrSysVarName > , < bstrOriginVar >)

bstrSysVarName	:	[in] Target system variable (VT_BSTR) (%OutDataAsyncA to %OutDataAsyncH)
bstrOriginVar	:	[in] Value or variable name to be written(VT_ BSTR)
Return value	:	[out] None

The following sample shows how to write “1” into “%OutDataAsyncA”.

Example

```
    Call caoCtrl.Execute("PutTerminalVariable", Array("%OutDataAsyncA", 1))
```

3.7. Recipe-related function

3.7.1. CaoController::Execute ("CopyRecipe") command

Destination recipe settings are all overwritten by the source recipe settings.

Syntax CopyRecipe (< lOriginRecipeNo > , < lDestRecipeNo >)

lOriginRecipeNo : Copy source recipe No. (0 to 999) (VT_I4)

lDestRecipeNo : Copy destination recipe No. (0 to 999) (VT_I4)

Return value : [out] None

The following sample shows how to copy Recipe number 1 to Recipe number 2.

Example

```
Call caoCtrl.Execute("CopyRecipe", Array(1, 2))
```

3.7.2. CaoController::Execute ("MoveRecipe") command

Destination recipe settings are all overwritten by the source recipe settings. If a recipe is moved successfully, the source recipe settings are completely deleted.

Syntax MoveRecipe (< lOriginRecipeNo > , < lDestRecipeNo >)

lOriginRecipeNo : [in] Source recipe No. (0 to 999) (VT_I4)

lDestRecipeNo : [in] Destination recipe No. (0 to 999) (VT_I4)

Return value : [out] None

The following sample shows how to move Recipe number 1 to Recipe number 2.

Example

```
Call caoCtrl.Execute("MoveRecipe", Array(1, 2))
```

3.7.3. CaoController::Execute ("RenameRecipe") command

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

Syntax RenameRecipe (< lRecipeNo >, < bstrParam> [, < bIsScalar >])

lRecipeNo : [in] Recipe setting number (VT_I4)(0 to 999)

bstrParam : [in] Recipe setting name (VT_BSTR)

bIsScalar : [in] Specify the data type of argument of <bstrParam>. (VT_BOOL)
FALSE : Treat <bstrParam> as a string. (default)
TRUE : Treat <bstrParam> as a scalar array variable.

Return value : [out] None

The following sample shows how to set the recipe name of recipe number 1 as “Recipe1”.

Example

```
            caoCtrl.Execute ″RenameRecipe″, Array(1, ″Recipe1″, FALSE)
```

3.7.4. CaoController::Execute ("ReadRecipe") command

This command returns the recipe setting number currently used.

Syntax ReadRecipe ()

argument : [in] None

Return value : [out] Recipe setting number (VT_I4)

0 to 999 : Recipe setting number currently used

If no recipe setting is used, “-1” is returned.

Example

```
            Dim lRecipeInfoNo as long
            lRecipeInfoNo =  caoCtrl.Execute(″ReadRecipe″)
```

3.7.5. CaoController::Execute ("ReadRecipeString") command

This command returns the recipe setting namd currently used.

If no recipe settings are used, the return value will be blank.

Syntax ReadRecipeString ()

argument : [in] None

Return value : [out] Recipe setting name (VT_BSTR)

Example

```
            Dim strRecipeInfoName as string
            strRecipeInfoName =  caoCtrl.Execute(″ReadRecipeString″)
```

3.7.6. CaoController::Execute ("ChangeRecipe") command

This command closes all open dialog boxes and changes to the recipe setting of the specified recipe number.

Syntax ChangeRecipe (< lRecipeNo > [, < lSave >])

lRecipeNo : [in] : Recipe setting number(VT_I4)

-1 : Do not use recipe settings.

0 to 999 : Change to the recipe setting of the specified Recipe number.

lSave : [in] Specify whether to save the recipe setting number (VT_I4)

0 : Do not save the recipe setting number after the change(when omitted)

1 : Save the recipe number after the change

Return value : [out] None

The following sample shows how to change the recipe to the Recipe number 1 without saving the recipe setting number after the change.

Example

```
caoCtrl.Execute "ChangeRecipe", Array(1, 0)
```

3.7.7. CaoController::Execute ("ChangeRecipeString") command

This command closes all open dialog boxes and changes to the recipe setting of the specified name.

Syntax ChangeRecipeString (< bstrRecipeName > [, < lSave >])

bstrRecipeName : [in] : Recipe setting name (VT_BSTR)

lSave : [in] Specify whether to save the recipe setting number (VT_I4)

0 : Does not save the recipe setting number after the change (when omitted)

1 : Save the recipe setting number after the change.

Return value : [out] None

The following sample shows how to change the recipe to the Recipe name “Recipe1” without saving the recipe setting number after the change.

Example

```
caoCtrl.Execute "ChangeRecipeString", Array("Recipe1", 0)
```

3.8. Others

3.8.1. CaoController::Execute ("InputSimConsole") command

This command allows a communication command to serve as the input from the button of the handheld controller.

If two key codes are entered, these keys are deemed to be pressed simultaneously.

Syntax InputSimConsole (< lKeyCode1 > [, < lKeyCode2 >])

lKeyCode1 : [in] key code (VT_I4)

lKeyCode2 : [in] key code (VT_I4)

Return value : [out] None

The following shows the correspondence between values and the button names.

0: No. 0 button, 1: No. 1 button, 2: No. 2 button, 3: No. 3 button, 4: No. 4 button, 5: No. 5 button, 6: No. 6 button, 7: No. 7 button, 8: No. 8 button

11: Lower left, 12: Down, 13: Lower right, 14: Left, 16: Right, 17: Upper left, 18: Up, 19: Upper right

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

The following example shows how to enter 0 and 1 keys simultaneously.

Example

```
        Call caoCtrl.Execute("InputSimConsole", Array(0,1))
```

3.8.2. CaoController::Execute ("SearchUnitNo") command

This command searches the unit corresponding to the specified name in the flowchart and then returns the unit number. When multiple units with the same name exist, it returns the lowest unit number.

Syntax SearchUnitNo (< bstrParam > [, < bIsScalar >])

bstrParam : [in] String or Scalar array variable (VT_BSTR)

bIsScalar : [in] Specify the data type of argument of <bstrParam>.(VT_BOOL)
FALSE : Treat <bstrParam> as string. (default)
TRUE : Treat <bstrParam> as Scalar array variable/

Return value : [out] None

The following sample shows how to specify “Capture” as a name and then receive the unit number.

Example

```
        Dim lUnitNo as long
        lUnitNo = caoCtrl.Execute("SearchUnitNo", Array("Capture", FALSE))
```

3.8.3. CaoController::Execute ("ReadVersionInfo") command

This command reads the system information from the controller.

The system information is stored in the order of model, firmware version.

Syntax ReadVersionInfo ()

Argument : [in] None

Return value : [out] System information (VT_BSTR | VT_ARRAY)

The following sample shows how to obtain the system information.

Example

```
Dim vntVerInfo as Variant
vntVerInfo = caoCtrl.Execute("ReadVersionInfo")
```

3.9. Original expansion command

3.9.1. CaoController::Execute ("ExecuteCommand”) command

This command executes a specified non-procedure command and receives command response regardless of the command execution result (success or failure).

For information about supported non-procedure commands, please refer to XGX series user’s manual.

Syntax ExecuteCommand(< bstrCommand >)

bstrCommand : [in] Specify the character string of a command (VT_BSTR)

Return value : [out] Command response (VT_BSTR)

The following sample shows how to specify a non-procedure command and switches the XGX to Run mode.

Example

```
Dim strRet as String

strRet = caoCtrl.Execute("ExecuteCommand",  "R0")
```

3.9.2. CaoController::Execute ("ExecuteCommandAsync ”) command (not recommended)

This command executes a specified non-procedure command asynchronously.

For information about supported non-procedure commands, please refer to XGX series user’s manual.

To receive and confirm the return value, use GetCommandResult command. For details about GetCommandResult command, refer to 3.9.8.CaoController::Execute ("GetCommandResult”) .

When you use this command, be sure to disconnect the provider after the execution of GetCommandResult command.

Otherwise, an error may occur at asynchronous operation.

Syntax ExecuteCommandAsync(< bstrCommand >)

bstrCommand : [in] Specify the character string of a command (VT_BSTR)

Return value : [out] none

The following sample shows how to specify a non-procedure command and switches the XGX to Run mode.

Example

```
Dim vntResult as Variant

Call caoCtrl.Execute(″ExecuteCommandAsync″,  ″R0″)

‘ Obtain the return value of ExecuteCommandAsync command.
vntResult = caoCtrl.Execute(″GetCommandResult″)

‘ vntResult : Return value(BSTR)of a non-procedure command
```

3.9.3. CaoController::Execute ("TriggerAndGetResult”) command

This command receives the execution result after trigger issuance. If there is not output result, it waits until the timeout period passes.

Syntax TriggerAndGetResult(< iTriggerNo >)

iTriggerNo	:	[in] Trigger number. 1 to 4 (VT_I2)
Return value	:	[out] Output result (VT_BSTR)

Example

```
Dim strRet as String

strRet = caoCtrl.Execute(″TriggerAndGetResult″,  1)
```

3.9.4. CaoController::Execute ("RecievePacket") command

This command receives a packet. If a packet has been stored in the reception buffer, the command retrieves the packet.

Syntax RecievePacket( )

Argument	:	[in] none
Return value	:	[out] Received packet (VT_BSTR)

The following sample shows the combination of this command and Trigger command.

Example

```
Dim strRet as String

caoCtrl.Execute ″Trigger″,  1
strRet = caoCtrl.Execute(″RecievePacket″)
```

3.9.5. CaoController::Execute ("ClearPacket") command

This command clears the packet in the reception buffer.

Syntax ClearPacket( )

Argument	:	[in] none
Return value	:	[out] none

Example

```
caoCtrl.Execute ″ClearPacket″
```

3.9.6. CaoController::Execute ("SetTimeout") command

This command sets the timeout period for the command sending and receiving.

Syntax SetTimeout(< iTimeout >)

iTimeout	:	[in] Timeout period (msec) (VT_UI4)
Return value	:	[out] none

Example

```
caoCtrl.Execute ″SetTimeout″,  1000
```

3.9.7. CaoController::Execute ("GetTimeout") command

This command obtains the currently specified timeout period for the command sending and receiving.

Syntax GetTimeout( )

Argument	:	[in] none
Return value	:	[out] timeout value (msec) (VT_UI4)

Example

```
Dim timeout as long
timeout = caoCtrl.Execute(″GetTimeout″)
```

3.9.8. CaoController::Execute ("GetCommandResult”) command

This command waits the completion of an asynchronous command and then obtains the return value of the asynchronous command.

If the executed asynchronous command has no return value, nothing is returned.

If this command is executed after a synchronous command, an error (0x80100003) occurs, and nothing is returned.

If an error occurs in an asynchronous command execution, the error is not issued at the asynchronous command execution timing but issued at the GetCommandResult command execution.

While waiting the asynchronous command completion, if there is no response within the predetermined timeout period, a timeout error (0x80000900) occurs.

If any other asynchronous command is executed after the asynchronous command, which you intend to obtain the command result, the previously executed command result will be deleted.

Syntax [< vntRet > =]GetCommandResult( )

Argument	:	[in] none
vntRet	:	[out] Return value of asynchronous command (VT_VARIANT)

The following sample shows how to obtain the return value when a test is executed asynchronously.

Example

```
Dim vntResult as Variant

Call caoCtrl.Execute("ExecuteCommandAsync",  "R0")
vntResult = caoCtrl.Execute("GetCommandResult")
```