XGX プロバイダ
KEYENCE XG-X2000 シリーズ用プロバイダ

Version 1.0.2

ユーザーズ ガイド

August 29, 2019

【備考】

【改版履歴】

バージョン	日付	内容
1.0.0	2016-06-17	初版.
1.0.1	2016-07-12	WriteCharReg コマンドの引数（bIsScalar）を追加
1.0.2	2017-02-10	下記 34 コマンドの追加 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	非同期コマンド使用時の注意点追加

目次

1. はじめに

本書は KEYENCE 社製ビジョンシステムである XG-X2000 シリーズ用の CAO プロバイダである，XGX プロバイダのユーザーズガイドです．XGX プロバイダは RS-232C または Ethernet で接続された XG-X2000 シリーズコントローラと無手順方式でコマンドの実行や処理結果の通知を行います．

本書は，このXGXプロバイダの機能と実装されているメソッドについて説明します．

1.1. 機器の設定

環境設定から使用する通信手段の設定を行ってください．本プロバイダでは RS-232C かネットワークが使用できます．入出力設定のメニューより，接続する手段を選択して設定してください．

図 1-1 環境設定

1.1.1. RS-232C 設定

RS-232C の通信設定では，デリミター以外が設定変更可能になっています．デフォルト設定から変更した場合は，AddControllerの接続パラメータを変更してください(2.2.1.1 参照)．開始デリミターは"なし"，終了デリミターは"CR"を設定してください．

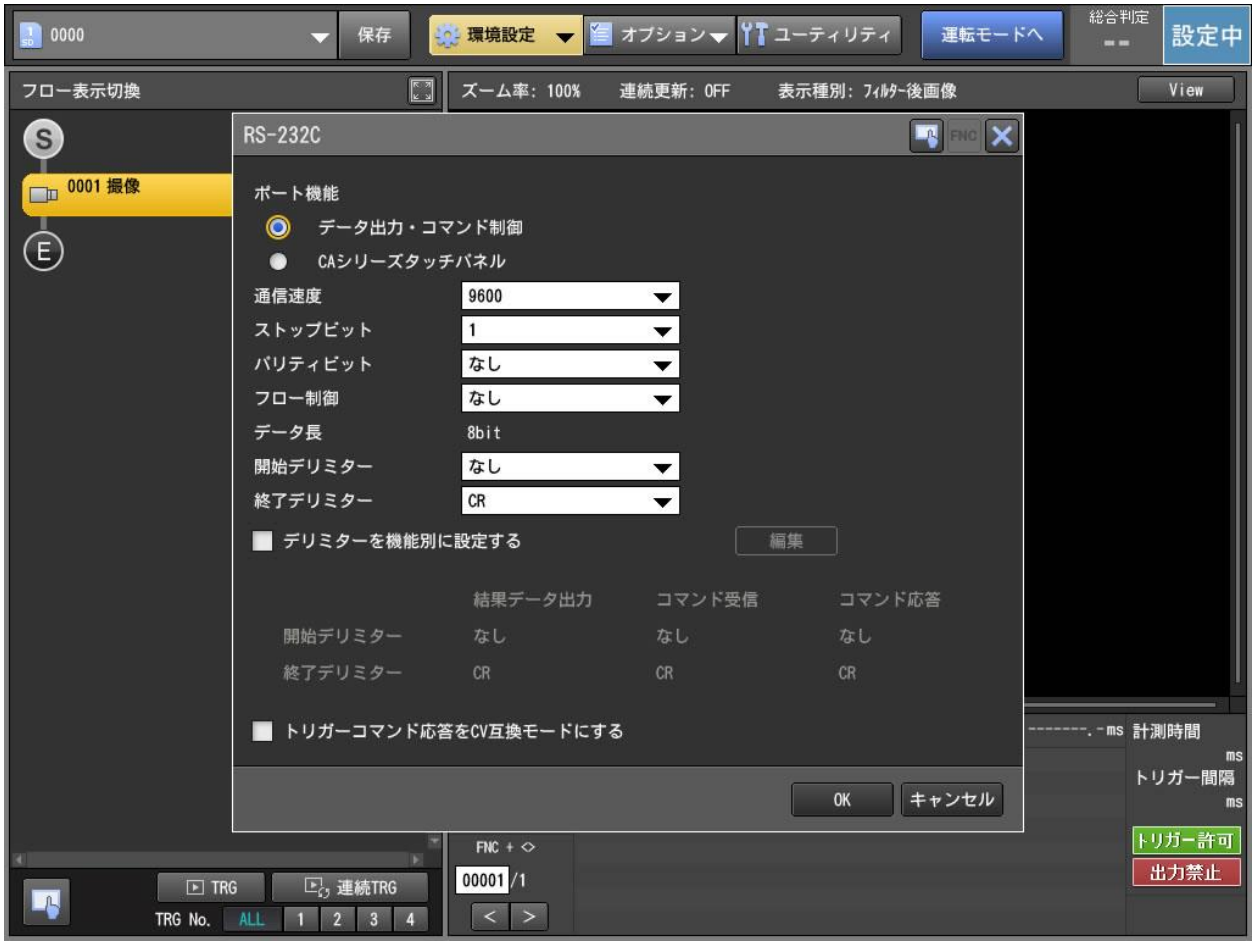

図 1-2 RS-232C 設定

1.1.2. Ethernet 設定

Ethernet の通信設定では，デリミター以外が設定変更可能になっています．デフォルト設定から変更した場合は，AddControllerの接続パラメータを変更してください(2.2.1.1 参照)．開始デリミターは"なし"，終了デリミターは"CR"を設定してください．

図 1-3 Ethernet 設定

2. プロバイダの概要

2.1. 概要

XGX プロバイダは, コマンドの実行方法として CaoController::Execute による方法を提供しています.
CaoController::Execute では, シリアルインタフェースを利用しコマンドの送受信を行います.

表 2-1 XGX プロバイダ

ファイル名	CaoProvKEYENCEXGX.dll
ProgID	CaoProv.KEYENCE.XGX
レジストリ登録[1]	regsvr32 CaoProvKEYENCEXGX.dll
レジストリ登録の抹消	regsvr32 /u CaoProvKEYENCEXGX.dll

2.2. メソッド・プロパティ

2.2.1. CaoWorkspace::AddController メソッド

XGX プロバイダでは, AddController 時に, 通信用の接続パラメータを参照し, 通信の接続を行います. この時, オプションで通信形態を指定します.

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

bstrCtrlName : [in] コントローラ名
bstrProvName : [in] プロバイダ名 固定値 = ” CaoProv.KEYENCE.XGX”
bstrPcName : [in] プロバイダの実行マシン名（任意）
bstrOption : [in] オプション文字列

以下にオプション文字列に指定するリストを示します.

表 2-2 CaoWorkspace::AddController のオプション文字列

オプション	意味
Conn =<接続パラメータ>	必須. 通信形態とその接続パラメータを設定します. 詳細は 2.2.1.1 を参照してください.
Timeout[=<タイムアウト時間>]	送受信時のタイムアウト時間(ミリ秒)を指定します.（デフォルト: 500）

[1] ORiN SDK でインストールした場合は手動で登録／抹消する必要はありません.

2.2.1.1. Conn オプション

以下に Conn オプションの接続パラメータ文字列を示します. ここで角括弧(“[ ]”)内は省略可能を示します. また, 各パラメータの解説中の下線部はオプションを指定しなかったときのデフォルト値を示します.

- Ethernet デバイス

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

  <IP Address> : 接続する XGX シリーズの IP アドレス
  例:”192.168.0.10”

  <Port No> : 接続ポート番号
  8500, 8501, . . . 任意指定可能

使用例

```
  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 デバイス

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

  <COM Port> : COM ポート番号
  ‘1’-COM1, ‘2’-COM2, …

  <BaudRate> : 通信速度
  4800, 9600, 19200, 38400, 57600, 115200.

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

  <DataBits> : データビット数
  ‘7’-7bit, ‘8’-8bit.

  <StopBits> : ストップビット数
  ‘1’-1bit, ‘2’-2bit.

  <Flow> : フロー制御
  ‘0’-フロー制御なし, ‘1’-Xon/Xoff, ‘2’-ハードウェア制御.
  OR をとって指定できます.

使用例

```
  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 メソッド

RS-232C, または, イーサネットを介して無手順方式でコマンドの送受信を行います. 第 1 引数にコマンド名, 第2引数にコマンドのパラメータを指定します. 各コマンドの詳細は3章はじめにコマンドリファレンスを参照してください.

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

bstrCommandName : [in] コマンド名

vntParam : [in] パラメータ

2.2.3. エラーコード

メソッドを実行した際の XGX シリーズからの処理結果は HRESULT として返されます. XGX シリーズ固有のエラーは 0x80108000 でマスクされて返されます. エラー内容については XGX シリーズのユーザーズマニュアルを参照してください.

正常に処理された場合(OK) : S_OK (0)

正常に処理されなかった場合(ER) : 0x80108000 + 戻り値

例:ChangeMode を実行したとき.

hr = 0x80108016 : 不正なパラメータが付与されています.

その他プロバイダ共通のエラー内容については ORiN2SDK プログラマーズユーザーズガイドを参照してください.

表 2-3 エラーコード一覧

エラー名	エラー番号	説明
E_XGXERROR_XGXERR	0x80108000 | XGXエラー	XGX シリーズ固有エラー(2.2.3 参照)
E_XGXERROR_LENGTH	0x80100000	パケット長エラー
E_XGXERROR_PACKET	0x80100001	パケット異常エラー
E_COMMAND_EXECUTING	0x80100002	コマンド実行中に別コマンドを実行しました
E_GET_COMMAND_RESULT	0x80100003	同期コマンド実行後に GetCommandResult コマンドを実行しました

3. コマンドリファレンス

本章では CaoController::Execute メソッドの各コマンドについて解説します. 各コマンドの詳細動作については KEYENCE 社の XG-X2000 シリーズ通信制御マニュアルの制御用通信コマンドリファレンスを参照してください.

表 3-1 CaoController::Execute コマンド一覧

XGX シリーズ 無手順コマンド	コマンド	機能	
トリガー			
T1, T2, T3, T4, TA	Trigger	トリガを発行します	P. 15
モード切替			
R0, S0	ChangeMode	運転／停止モードを変更します	P. 15
	ChangeModeAsync	非同期で運転／停止モードを変更します (非推奨)	P. 15
RM	ReadMode	運転/設定モード読み出しを行います	P. 16
検査設定 No.切替			
PW	ChangeInspectSetting	指定した設定 No に切り替えます	P. 16
	ChangeInspectSettingAsync	非同期で指定した設定 No に切り替えます (非推奨)	P. 17
	ChangeInspectSettingString	指定した設定 No に切り換えます（名前指定）	P. 18
	ChangeInspectSettingStringAsync	非同期で指定した設定 No に切り換えます（名前指定）(非推奨)	P. 18
PR	ReadInspectSetting	現在読み込んでいる検査設定 No を取得します	P. 18
	ReadInspectSettingString	現在読み込んでいる検査設定 No の名前を取得します	P. 19
コントローラー制御			
CE	ClearError	エラークリアを行います	P. 19
NR	ReadRegisterImageNo	参照先登録画像 No の読み出しを行います	P. 20
NU	UpdateRegisterImageNo	参照先登録画像 No の変数参照値への更新を行います	P. 20
NW	WriteRegisterImageNo	参照先登録画像 No の切り換えを行います	P. 21
PN	RenameInspectionSetting	検査設定名称書き換えを行います	P. 22
RE	ReturnFlowTop	フロー先頭復帰を行います	P. 22
RR	UpdatePosAdjustment	位置補正基準値更新を行います	P. 22
RS	Reset	リセットを行います	P. 23

-	GetTimeout	タイムアウト時間の取得を行います	P. 41
-	GetCommandResult	非同期コマンドの戻り値を取得します	P. 41

3.1. トリガー

3.1.1. CaoController::Execute ("Trigger") コマンド

トリガを発行します. 結果を出力設定している場合は RecievePacket コマンド等で受信してください.

書式　Trigger(< iTriggerNo >)

iTriggerNo　:　[in] 発行対象のトリガ番号を指定します(VT_I2)
1～4 ： トリガ1～4
-1 ： 全トリガ

戻り値　:　[out] なし

トリガ番号1を発行する場合の例を以下に示します.

使用例

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

3.2. モード切替

3.2.1. CaoController::Execute ("ChangeMode") コマンド

運転モードまたは, 停止モードに移行します.

書式　ChangeMode(< iMode >)

iMode　:　[in] 変更先のモードを指定します(VT_UI4)
0 ： 停止モード
1 ： 運転モード

戻り値　:　[out] なし

運転モードに切替える場合の例を以下に示します.

使用例

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

3.2.2. CaoController::Execute ("ChangeModeAsync") コマンド (非推奨)

非同期で運転モードまたは, 停止モードに移行します.

コマンドの戻り値は GetCommandResult コマンドで取得し, 確認してください. GetCommandResult コマンドの詳細については, 3.9.8.CaoController::Execute ("GetCommandResult") コマンドを参照ください.

コマンドの使用時は，必ず GetCommandResult コマンドを実行後にプロバイダを切断してください．
非同期処理で異常が発生する場合があります．

書式　ChangeModeAsync(< iMode >)

iMode	:	[in] 変更先のモードを指定します(VT_UI4)
		0 ： 停止モード
		1 ： 運転モード
戻り値	:	[out] なし

運転モードに切替える場合の例を以下に示します．

使用例

```
Dim vntResult as Variant

caoCtrl.Execute "ChangeModeAsync", 1

‘ ChangeModeAsync コマンドの戻り値取得
vntResult = caoCtrl.Execute("GetCommandResult")

‘ vntResult : 戻り値 (Empty)
```

3.2.3. CaoController::Execute ("ReadMode") コマンド

現在の運転モードを取得します．

書式　ReadMode ()

引数	:	[in] なし
戻り値	:	[out]モード(VT_I4)
		0 ： 設定モード
		1 ： 運転モード

現在の運転モードを取得する例を以下に示します．

使用例

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

3.3. 検査設定 No.切替

3.3.1. CaoController::Execute ("ChangeInspectSetting") コマンド

指定された SD カードの検査設定 No.に設定を切り換えます．

書式　ChangeInspectSetting(< iDriveNo >, < iSettingNo >)

iDriveNo	:	[in] SD カード番号を指定します（VT_I4）
		1 ： SD1
		2 ： SD2
iSettingNo	:	[in] 検査設定 No.を指定します（0～999）（VT_I4）
戻り値	:	[out] なし

SD カード（SD1）の検査設定 No.1 に設定を切り換える場合の例を以下に示します.

使用例

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

3.3.2. CaoController::Execute（"ChangeInspectSettingAsync"）コマンド（非推奨）

非同期で指定された SD カードの検査設定 No.に設定を切り換えます.
コマンドの戻り値は GetCommandResult コマンドで取得し，確認してください. GetCommandResult コマンドの詳細については，3.9.8.CaoController::Execute ("GetCommandResult") コマンドを参照ください.
コマンド使用時は，必ず GetCommandResult コマンドを実行後にプロバイダを切断してください.
非同期処理で異常が発生する場合があります.

書式　ChangeInspectSettingAsync(< iDriveNo >, < iSettingNo >)

iDriveNo	:	[in] SD カード番号を指定します（VT_I4）
		1 ： SD1
		2 ： SD2
iSettingNo	:	[in] 検査設定 No.を指定します（0～999）（VT_I4）
戻り値	:	[out] なし

SD カード（SD1）の検査設定 No.1 に設定を切り換える場合の例を以下に示します.

使用例

```
Dim vntResult as Variant

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

‘ ChangeInspectSettingAsync コマンドの戻り値取得
vntResult = caoCtrl.Execute("GetCommandResult")

vntResult : 戻り値 (Empty)
```

3.3.3. CaoController::Execute ("ChangeInspectSettingString") コマンド

指定された検査設定名称の検査に設定を切り換えます.

書式　ChangeInspectSettingString(< strName >)

strName　:　[in] 検査設定名称(VT_BSTR)

戻り値　:　[out] なし

検査設定名称 test に設定を切り換える場合の例を以下に示します.

使用例

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

3.3.4. CaoController::Execute ("ChangeInspectSettingStringAsync") コマンド (非推奨)

非同期で指定された検査設定名称の検査に設定を切り換えます.

コマンドの戻り値は GetCommandResult コマンドで取得し, 確認してください. GetCommandResult コマンドの詳細については, 3.9.8.CaoController::Execute ("GetCommandResult") コマンドを参照ください.

コマンド使用時は, 必ず GetCommandResult コマンドを実行後にプロバイダを切断してください.

非同期処理で異常が発生する場合があります.

書式　ChangeInspectSettingStringAsync(< strName >)

iDriveNo　:　[in] 検査設定名称(VT_BSTR)

戻り値　:　[out] なし

検査設定名称 test に設定を切り換える場合の例を以下に示します.

使用例

```
Dim vntResult as Variant

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

'ChangeInspectSettingStringAsync コマンドの戻り値取得
vntResult = caoCtrl.Execute("GetCommandResult")

vntResult : 戻り値 (Empty)
```

3.3.5. CaoController::Execute ("ReadInspectSetting") コマンド

検査設定 No.を取得します.

書式　ReadInspectSetting( )

引数　:　[in] なし

戻り値 : [out] < iDriveNo >, < iSettingNo > （VT_I4 | VT_ARRAY）
iDriveNo ： SD カード番号
iSettingNo ： 検査設定 No

現在設定されている検査設定 No.の取得方法を下記に示します.

使用例

```
Dim vntRet as Variant

vntRet = caoCtrl.Execute("ReadInspectSetting")

‘ vntRet(0) : SD カード番号
‘ vntRet(1) : 検査設定 No
```

3.3.6. CaoController::Execute（"ReadInspectSettingString"）コマンド

検査設定名称を取得します.

書式 ReadInspectSettingString( )

引数 : [in] なし
戻り値 : [out] < strName > (VT_BSTR)
strName ： 検査設定名称

現在設定されている検査設定名称の取得方法を下記に示します.

使用例

```
Dim strRet as String

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

3.4. コントローラー制御

3.4.1. CaoController::Execute（"ClearError"）コマンド

種別(0 または 1)を指定して, エラー状態とエラーコードをクリアします. エラー状態でないときも, 正常終了します.

書式 ClearError(< iErrorType >)

iErrorType : [in] エラー状態種別を指定します（VT_UI4）
0 ： %Error0 と%Error0Code をクリアする
1 ： %Error1 と%Error1Code をクリアする
戻り値 : [out] なし

例を以下に示します.

使用例

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

3.4.2. CaoController::Execute ("ReadRegisterImageNo") コマンド

計測に現在使用している登録画像 No.を返信します.

以下の書式で実行できます.

1. 画像番号を指定しない場合

書式　ReadRegisterImageNo (< lUnitNo >)

lUnitNo	:	[in] ユニット ID(0～999)(VT_I4)
戻り値	:	[out] 登録画像 No(0～999)(VT_I4)

2. 画像番号を指定する場合

書式　ReadRegisterImageNo (< lUnitNo >, < lImgNo >)

lUnitNo	:	[in] ユニット ID(0～999)(VT_I4)
lImgNo	:	[in] 画像番号(VT_I4) 画像演算ユニットまたはC 言語ユニットのユニットID 指定時:元画像 No.の指定 1～2 キャリブレーションユニット ID 指定時:ティーチング画像番号 1～16
戻り値	:	[out] 登録画像 No(0～999)(VT_I4)

ユニット ID101 が計測に現在使用している登録画像 No を取得する例を以下に示します.

使用例

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

3.4.3. CaoController::Execute ("UpdateRegisterImageNo") コマンド

指定されたユニットの登録画像 No.の切り替え先を変数参照している場合に,変数の現在値を取り込んで登録画像 No.を切り替えます. 必要が有れば,その番号の登録画像で画像基準情報を更新します.

書式　UpdateRegisterImageNo (< lUnitNo >)

lUnitNo	:	[in] ユニット ID(VT_I4) 0～999 : 指定するユニット ID -1 : すべてのユニット

戻り値 : [out] なし

ユニット ID101 の登録画像の変数参照を更新し, 登録画像 No を切り替える例を以下に示します.

使用例

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

3.4.4. CaoController::Execute ("WriteRegisterImageNo") コマンド

指定されたユニットの登録画像 No.を切り換えます. 必要があれば,その番号の登録画像で画像基準情報を更新します.

以下の書式で実行できます.

1. 画像番号を指定しない場合

書式 WriteRegisterImageNo (< lUnitNo >, < lRegImgNo >)

lUnitNo : [in] ユニット ID (VT_I4)
0～999 : 指定するユニット ID
-1 : すべてのユニット

lRegImgNo : [in] 登録画像 No(0～999)(VT_I4)

戻り値 : [out] なし

2. 画像番号を指定する場合

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

lUnitNo : [in] ユニット ID (VT_I4)
0～999 : 指定するユニット ID
-1 : すべてのユニット

lRegImgNo : [in] 登録画像 No(0～999)(VT_I4)

lImgNo : [in] 画像番号(VT_I4)
画像演算ユニットまたはC言語ユニットのユニットID指定時:元画像 No.の指定 1～2
キャリブレーションユニット ID 指定時:ティーチング画像番号 1～16

戻り値 : [out] なし

ユニット ID101 の登録画像 No を 202 に変更する例を以下に示します.

使用例

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

3.4.5. CaoController::Execute ("RenameInspectionSetting") コマンド

指定した設定 No の名称を変更します.

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

lSdNo	:	[in] SD カード番号(1～2)(VT_I4)
lTestNo	:	[in] 検査設定 No(0～999) (VT_I4)
bstrParam	:	[in] 文字列または, スカラ型配列変数(VT_BSTR)
bIsScalar	:	[in] <bstrParam>の引数がスカラ型配列変数かを指定します. (VT_BOOL) FALSE : <bstrParam>を文字列として処理します. (デフォルト) TRUE : <bstrParam>をスカラ型配列変数として処理します.
戻り値	:	[out] なし

SD カード番号 1,検査設定 No101 の検査設定名称を TestName に変える例を以下に示します.

使用例

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

3.4.6. CaoController::Execute ("ReturnFlowTop") コマンド

ダイアログ条件待ちユニット,タイマー条件待ちユニット以外の待ちユニットおよび撮像ユニットによる待ち状態から,スタートユニットの次のユニットにジャンプします.

書式　ReturnFlowTop ()

引数	:	[in] なし
戻り値	:	[out] なし

スタートユニットの次のユニットにジャンプする例を以下に示します.

使用例

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

3.4.7. CaoController::Execute ("UpdatePosAdjustment") コマンド

指定された位置補正ユニットが現在参照している最新の値,または登録画像による再計算結果の値を基準値として取り込みます.

以下の書式で実行できます.

1. 基準値として取り込む値を指定しない場合

書式　UpdatePosAdjustment (< lUnitNo >)

lUnitNo	:	[in] ユニット ID (VT_I4)
		0～999 ： 指定するユニット ID
		-1 ： すべてのユニット
戻り値	:	[out] なし

2. 基準値として取り込む値を指定する場合

書式　UpdatePosAdjustment (< lUnitNo >, < lStandardValue >)

lUnitNo	:	[in] ユニット ID (VT_I4)
		0～999 ： 指定するユニット ID
		-1 ： すべてのユニット
lStandardValue	:	[in] 基準値として取り込む値 (VT_I4)
		0:最新結果
		1:登録画像による再計算結果
戻り値	:	[out] なし

ユニット ID101 が現在参照している最新の値を基準値として取り込む例を以下に示します.

使用例

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

3.4.8. CaoController::Execute ("Reset") コマンド

以下のすべての項目を実施します.

・ システム変数をすべて初期化,画像を含む各種バッファをすべてクリアします.

・ ユニットのトリガ待ち,イベント待ちを解除します.

・ データを保存するファイルのファイル名を新規作成します.

・ ユーザー定義のローカル変数で「リセット時に初期化する」の設定が ON になっているものは初期化します.

・ ユーザー定義のグローバル変数で「リセット時に初期化する」の設定が ON になっているものは初期化します.

・ %JAHold を初期化します.

・ フローの先頭に戻ります.

・ 履歴データはすべてクリアします.

・ 統計データはすべてクリアします.

・ 欠陥分類の結果はすべてクリアします.

書式　Reset ()

引数　：[in] なし
戻り値　：[out] なし

リセットを実行する例を以下に示します.

使用例

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

3.4.9. CaoController::Execute ("ReCalcImageInfo") コマンド

指定されたユニット ID の画像基準情報を,現在の登録画像と設定パラメータによって再計算した結果で更新します.

書式　ReCalcImageInfo (< lUnitNo >)

lUnitNo　：[in] ユニット ID(VT_I4)
0～999　：指定するユニット ID
-1　：すべてのユニット
戻り値　：[out] なし

ユニット ID101 の画像基準情報を再計算する例を以下に示します.

使用例

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

3.4.10. CaoController::Execute ("SaveSetting") コマンド

現在の検査設定,グローバル変数,ローカル変数,環境設定を保存します.

書式　SaveSetting ()

引数　：[in] なし
戻り値　：[out] なし

現在の設定を保存する例を以下に示します.

使用例

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

3.4.11. CaoController::Execute (″ExecuteTeaching″) コマンド

指定したキャリブレーションユニットに対して,現在設定されている登録画像を用いてティーチングを実行します.

書式　ExecuteTeaching (< lUnitNo >)

lUnitNo	:	[in] ユニット ID(0～999) (VT_I4)
戻り値	:	[out] なし

ユニット ID101 のティーチングを実行する例を以下に示します.

使用例

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

3.4.12. CaoController::Execute (″CancelWaitStatus″) コマンド

端子条件待ちおよび変数条件待ちユニットの待ち状態を解除します. 入力パラメータにより,解除する待ちユニットの結果データ(判定 No.および合致条件ビット論理和)を任意の状態にすることもできます.

・ 解除する待ちユニットの結果データを参照しない場合は,入力パラメータは 0 を指定します. この場合,判定 No.および合致条件ビット論理和は 0 のまま待ち状態が解除されます.

・ 入力パラメータは二進数として合致条件に割り付けられ,1 となっているビットの中で最下位のビットが割り付けられている条件を判定 No.とします.

書式　CancelWaitStatus (< lUnitNo >)

lUnitNo	:	[in] 合致条件に割り付けるビット(0～(2^{20}-1)) (VT_I4)
戻り値	:	[out] なし

解除する待ちユニットの結果データを参照しない例を以下に示します.

使用例

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

3.5. OCR・2D コードリーダー・1D コードリーダー関連

3.5.1. CaoController::Execute (″WriteCharReg″) コマンド

OCR または 2D コードリーダー, 1D コードリーダーの REG を書き換えます. 書式によって挙動が変化します. OCR で指定可能な ASCII コードについて詳しくは, XG-X2000 シリーズマニュアル「OCR ユニット用文字コ

ード表」(2-91 ページ)をご覧ください.

書式　WriteCharReg(< lUnitNo >[, < lRowNo >, < bstrParam > [, <bIsScalar>]])

lUnitNo	:	[in] ユニット ID(0～999)(VT_I4)
lRowNo	:	[in] 行番号(VT_I4) OCR ユニットの場合 ： 1～2 1D コードリーダーツール, 2D コードリーダーツールの場合 ： 1～16
bstrParam	:	[in] 文字列または, スカラ型配列変数(VT_BSTR)
bIsScalar	:	[in] <bstrParam>の引数がスカラ型配列変数かを指定します. (VT_BOOL) FALSE ： <bstrParam>を文字列として処理します. (デフォルト) TRUE ： <bstrParam>をスカラ型配列変数として処理します.
戻り値	:	[out] なし

書式による挙動は下記の通りとなります.

1. ユニット ID・行番号・文字列の場合
   ユニット ID(lUnitNo)の行番号(lRowNo)の照合条件の REG の内容を文字列(bstrParam)とします.
2. ユニット ID・行番号・スカラ型配列変数の場合
   ユニット ID(lUnitNo)の行番号(lRowNo)の照合条件の REG の内容にスカラ型配列変数(bstrParam)の値を ASCII コードとして設定します.
3. ユニット ID の場合
   ユニット ID(lUnitNo)の REG に, そのユニットの最新読み取り結果を設定します. 未計測の場合は, クリアされます (OCR ユニットの場合はスペースが入ります).

ユニット ID101 の行番号 1(OCR ユニットの場合)の文字列を"DEF"に設定する例を以下に示します.

使用例

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

3.5.2. CaoController::Execute ("ReadCharReg") コマンド

OCR または 2D コードリーダー, 1D コードリーダーの REG を読み出します.
書式によって挙動が変化します.

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

lUnitNo	:	[in] ユニット ID(0～999)(VT_I4)

lRowNo	:	[in] 行番号(VT_I4) OCR ユニットの場合 ： 1～2 1D コードリーダーツール, 2D コードリーダーツールの場合 ： 1～16
bstrParam	:	[in] スカラ型配列変数(VT_BSTR)
戻り値	:	[out] 文字列(VT_BSTR)または, なし(※) ※bstrParam を指定した場合, 戻り値なし

書式による挙動は下記の通りとなります.

1. ユニット ID・行番号の場合
   ユニット ID(lUnitNo)の行番号(lRowNo)の照合条件の REG の内容を戻り値で返します.
2. ユニット ID・行番号・スカラ型配列変数の場合
   ユニット ID(lUnitNo)の行番号(lRowNo)の照合条件の REG の内容をスカラ型配列変数(bstrParam)のインデックスで指定された要素から順に ASCII コードとして格納します.

ユニット ID101 の行番号 1(OCR ユニットの場合)の文字列を取得する例を以下に示します.

使用例

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

3.5.3. CaoController::Execute (″AutoTuning″) コマンド

指定した 2D コードリーダーユニットおよび 1D コードリーダーユニットで入力画像,または登録画像を用いた自動チューニングを行います.

書式 AutoTuning (< lUnitNo >, < lImg >)

lUnitNo	:	[in] ユニット ID(0～999)(VT_I4)
lImg	:	[in] チューニング対象画像(VT_I4) 0 ： 入力画像を対象とする 1 ： 登録画像を対象とする
戻り値	:	[out] なし

ユニット ID101 の登録画像を対象とする自動チューニングの例を以下に示します.

使用例

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

3.5.4. CaoController::Execute ("RegisterCharLibrary") コマンド

最新の検出結果または履歴画像,履歴結果の指定された文字を,指定文字種の文字として辞書に登録します.
以下の書式で実行できます.

1. 最新の検出結果を辞書に登録する場合

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

lUnitNo	:	[in] ユニット ID(0～999)(VT_I4)
lLineNo	:	[in] 検出結果行番号(1～2)(VT_I4)
lStringNo	:	[in] 検出結果文字番号(1～20)(VT_I4)
lStringKind	:	[in] 登録先文字種(-1～59)(VT_I4)
戻り値	:	[out] なし

2. 履歴結果を辞書に登録する場合

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

lUnitNo	:	[in] ユニット ID(0～999)(VT_I4)
lHistory	:	[in] 履歴蓄積条件(0～7)(VT_I4)
lNumber	:	[in] 過去 j 回目(0～(蓄積条件回数-1))(VT_I4)
lLineNo	:	[in] 検出結果行番号(1～2)(VT_I4)
lStringNo	:	[in] 検出結果文字番号(1～20)(VT_I4)
lStringKind	:	[in] 登録先文字種(-1～59)(VT_I4)
戻り値	:	[out] なし

ユニット ID101 の最新の検出結果(行番号 1,文字番号 2,登録先文字種 3)を辞書に登録する例を以下に示します.

使用例

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

3.5.5. CaoController::Execute ("DeleteCharLibrary") コマンド

指定された文字種の最後の登録番号の文字を辞書から削除します.

書式　DeleteCharLibrary (< lUnitNo >, < lStringKind >)

lUnitNo	:	[in] ユニット ID(0～999)(VT_I4)
lStringKind	:	[in] 文字種(-1～59)(VT_I4)
戻り値	:	[out] なし

ユニット ID101 の文字種 1 を削除する例を以下に示します.

使用例

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

3.6. データ入出力関連

3.6.1. CaoController::Execute (“GetIntVariable”) コマンド

変数の値を読み出し,整数に丸めて(四捨五入)出力します.

変数は最大で 16 個まで同時に読み出しができます.

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

bstrVar1	:	[in] 呼び出し元の変数名(VT_BSTR)
bstrVar2	:	[in] 呼び出し元の変数名(VT_BSTR)
bstrVar3	:	[in] 呼び出し元の変数名(VT_BSTR)
bstrVar4	:	[in] 呼び出し元の変数名(VT_BSTR)
bstrVar5	:	[in] 呼び出し元の変数名(VT_BSTR)
bstrVar6	:	[in] 呼び出し元の変数名(VT_BSTR)
bstrVar7	:	[in] 呼び出し元の変数名(VT_BSTR)
bstrVar8	:	[in] 呼び出し元の変数名(VT_BSTR)
bstrVar9	:	[in] 呼び出し元の変数名(VT_BSTR)
bstrVar10	:	[in] 呼び出し元の変数名(VT_BSTR)
bstrVar11	:	[in] 呼び出し元の変数名(VT_BSTR)
bstrVar12	:	[in] 呼び出し元の変数名(VT_BSTR)
bstrVar13	:	[in] 呼び出し元の変数名(VT_BSTR)
bstrVar14	:	[in] 呼び出し元の変数名(VT_BSTR)
bstrVar15	:	[in] 呼び出し元の変数名(VT_BSTR)
bstrVar16	:	[in] 呼び出し元の変数名(VT_BSTR)
戻り値	:	[out] 読み出し元の数値(V_I4 | VT_ARRAY)

変数名:#Data1,#Data2.#Data3 の数値を読み出す例を以下に示します.

使用例

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

3.6.2. CaoController::Execute ("PutIntVariableEx") コマンド

値を整数として変数に書き込みを実行して,フローの最初の撮像ユニットまたはフロー最後のエンドユニットに到達した時点で,指定した変数値に書き換えます.
書き込み先の変数名のリストと書き込み元の数値または変数名のリストは, 最大で 16 個まで同時に書き込みができます.
書き込み先の変数名のリストと, 書き込み元の数値または変数名のリストの要素数が一致しない場合, エラーとなります.

書式　PutIntVariableEx(< bstrDestVarNameList > , < bstrDestVarValList >)

bstrDestVarNameList	:	[in]書き込み先の変数名のリスト(VT_BSTR | VT_ARRAY)
bstrDestVarValList	:	[in] 書き込み元の数値または変数名のリスト(VT_BSTR | VT_ARRAY)
戻り値	:	[out] なし

変数名:#Data1,#Data2,#Data3 に値:1,2,3 を書き込む例を以下に示します.

使用例

```
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") コマンド

値を整数として変数に書き込みを実行して,フローの最初の撮像ユニットまたはフロー最後のエンドユニットに到達した時点で,指定した変数値に書き換えます.
書き込み先の変数名のリストと書き込み元の数値または変数名のリストは, 最大で 16 個まで同時に書き込みができます.
書き込み先の変数名のリストの要素数と, 書き込み元の数値または変数名のリストの要素数が一致しない場合, エラーとなります.

書式　PutIntVariable (< bstrDestVarNameList > , < bstrDestVarValList >)

bstrDestVarNameList	:	[in]書き込み先の変数名のリスト(VT_BSTR | VT_ARRAY)

bstrDestVarValList : [in] 書き込み元の数値または変数名のリスト（VT_BSTR | VT_ARRAY）

戻り値 : [out] なし

変数名:#Data1,#Data2,#Data3 に値:1,2,3 を書き込む例を以下に示します.

使用例

```
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"）コマンド

指定されたスカラ型変数の値を読み出します.

変数は最大で 16 個まで同時に読み出しができます.

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

bstrDestVar1 : [in]呼び出し元の変数名（VT_BSTR）

bstrDestVar2 : [in]呼び出し元の変数名（VT_BSTR）

bstrDestVar3 : [in]呼び出し元の変数名（VT_BSTR）

bstrDestVar4 : [in]呼び出し元の変数名（VT_BSTR）

bstrDestVar5 : [in]呼び出し元の変数名（VT_BSTR）

bstrDestVar6 : [in]呼び出し元の変数名（VT_BSTR）

bstrDestVar7 : [in]呼び出し元の変数名（VT_BSTR）

bstrDestVar8 : [in]呼び出し元の変数名（VT_BSTR）

bstrDestVar9 : [in]呼び出し元の変数名（VT_BSTR）

bstrDestVar10 : [in]呼び出し元の変数名（VT_BSTR）

bstrDestVar11 : [in]呼び出し元の変数名（VT_BSTR）

bstrDestVar12 : [in]呼び出し元の変数名（VT_BSTR）

bstrDestVar13 : [in]呼び出し元の変数名（VT_BSTR）

bstrDestVar14 : [in]呼び出し元の変数名（VT_BSTR）

bstrDestVar15 : [in]呼び出し元の変数名（VT_BSTR）

bstrDestVar16 : [in]呼び出し元の変数名（VT_BSTR）
戻り値 : [out] 読み出し元の数値(VT_R8 | VT_ARRAY)

変数名:#Data1,#Data2,#Data3 の数値を読み出す例を以下に示します.

使用例

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

3.6.5. CaoController::Execute（"PutVariableEx"）コマンド

変数書き込みを実行して,フローの最初の撮像ユニットまたはフロー最後のエンドユニットに到達した時点で,指定した変数値に書き換えます.
書き込み先の変数名のリストと書き込み元の数値または変数名のリストは，最大で 16 個まで同時に書き込みができます.
書き込み先の変数名のリストの要素数と，書き込み元の数値または変数名のリストの要素数が一致しない場合，エラーとなります.

書式 PutVariableEx (< bstrDestVarNameList > , < bstrDestVarValList >)

bstrDestVarNameList : [in]書き込み先の変数名のリスト（VT_BSTR | VT_ARRAY）
bstrDestVarValList : [in] 書き込み元の数値または変数名のリスト（VT_BSTR | VT_ARRAY）
戻り値 : [out] なし

変数名:#Data1,#Data2,#Data3 に値:1,2,3 を書き込む例を以下に示します.

使用例

```
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"）コマンド

指定されたスカラ型変数（グローバル変数,ローカル変数）を指定値で書き換えます. 反映タイミングは実行時です.
書き込み先の変数名のリストと書き込み元の数値または変数名のリストは，最大で 16 個まで同時に書き込

みができます.
書き込み先の変数名のリストの要素数と, 書き込み元の数値または変数名のリストの要素数が一致しない場合, エラーとなります.

書式　PutVariable (< bstrDestVarNameList > , < bstrDestVarValList >)

bstrDestVarNameList : [in]書き込み先の変数名のリスト(VT_BSTR | VT_ARRAY)
bstrDestVarValList : [in] 書き込み元の数値または変数名のリスト(VT_BSTR | VT_ARRAY)
戻り値 : [out] なし

変数名:#Data1,#Data2,#Data3 に値:1,2,3 を書き込む例を以下に示します.

使用例

```
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") コマンド

指定された端子の状態を読み出します.

書式　GetTerminalVariable (< bstrSysVarName >)

bstrSysVarName : [in]読み出すシステム変数名(VT_BSTR)
(%OutDataAsyncA～%OutDataAsyncH)
戻り値 : [out]読み出した値(VT_I4)

%OutDataAsyncA の端子状態を読み出す例を以下に示します.

使用例

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

3.6.8. CaoController::Execute ("PutTerminalOffset") コマンド

%CmdParamOffset に lMagOffset * lOffsetValue を書き込みます.

書式　PutTerminalOffset (< lMagOffset > , < lOffsetValue >)

lMagOffset : [in] オフセットの倍率(VT_I4)
lOffsetValue : [in] オフセット値(VT_I4)
戻り値 : [out] なし

倍率 2 でオフセット値 1 を書き込む例を以下に示します.

使用例

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

3.6.9. CaoController::Execute ("PutTerminalVariable") コマンド

端子割り付け可能でコマンドから書込可能なシステム変数の値を書き換えます. 反映タイミングは実行時です.

書式 PutTerminalVariable (< bstrSysVarName > , < bstrOriginVar >)

bstrSysVarName : [in]書き込むシステム変数名(VT_BSTR)
(%OutDataAsyncA～%OutDataAsyncH)
bstrOriginVar : [in]書き込み元の数値または変数名(VT_ BSTR)
戻り値 : [out] なし

%OutDataAsyncA に 1 を書き込む例を以下に示します.

使用例

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

3.7. レシピ機能関連

3.7.1. CaoController::Execute ("CopyRecipe") コマンド

コピー先として指定したレシピ設定を,コピー元として指定したレシピ設定の内容ですべて上書きします.

書式 CopyRecipe (< lOriginRecipeNo > , < lDestRecipeNo >)

lOriginRecipeNo : コピー元レシピ No (0～999) (VT_I4)
lDestRecipeNo : コピー先レシピ No (0～999) (VT_I4)
戻り値 : [out] なし

レシピ No1 のレシピをレシピ No2 にコピーする例を以下に示します.

使用例

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

3.7.2. CaoController::Execute ("MoveRecipe") コマンド

移動先として指定したレシピ設定を,移動元として指定したレシピ設定の内容ですべて上書きします. 移動に成功すると,移動元のレシピ設定内容をすべて消去します.

書式　MoveRecipe (< lOriginRecipeNo > , < lDestRecipeNo >)

lOriginRecipeNo	:	[in] 移動元レシピ No (0～999) (VT_I4)
lDestRecipeNo	:	[in] 移動先レシピ No (0～999) (VT_I4)
戻り値	:	[out] なし

レシピ No1 のレシピをレシピ No2 に移動する例を以下に示します.

使用例

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

3.7.3. CaoController::Execute ("RenameRecipe") コマンド

指定したレシピ設定 No.の名称を変更します.

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

lRecipeNo	:	[in] レシピ設定 No(VT_I4)(0～999)
bstrParam	:	[in] レシピ設定名称(VT_BSTR)
bIsScalar	:	[in] <bstrParam>の引数がスカラ型配列変数かを指定します. (VT_BOOL) FALSE : <bstrParam>を文字列として処理します. (デフォルト) TRUE : <bstrParam>をスカラ型配列変数として処理します.
戻り値	:	[out] なし

レシピ No1 の設定名称を Recipe1 する例を以下に示します.

使用例

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

3.7.4. CaoController::Execute ("ReadRecipe") コマンド

使用中のレシピ設定 No.を返します.

書式　ReadRecipe ()

引数 : [in] なし

戻り値 : [out]レシピ設定 No(VT_I4)

0～999 ： 現在のレシピ設定 No

-1 ： レシピ設定を使用していない場合

例を以下に示します.

使用例

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

3.7.5. CaoController::Execute ("ReadRecipeString") コマンド

使用中のレシピ設定名称を返します.

レシピ設定を使用していない場合は空文字が返ります.

書式 ReadRecipeString ()

引数 : [in] なし

戻り値 : [out]レシピ設定名称(VT_BSTR)

例を以下に示します.

使用例

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

3.7.6. CaoController::Execute ("ChangeRecipe") コマンド

レシピ設定 No を指定して,開いているダイアログをすべて閉じて,指定された名称のレシピ設定に切り換えます.

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

lRecipeNo : [in] :レシピ設定 No(VT_I4)

-1 ： レシピ設定を使用しない

0～999 ： 指定したレシピ設定 No.に切り換え

lSave : [in] 切換後のレシピ設定保存(VT_I4)

0 :切換後のレシピ設定 No.を保存しない(省略時)

1 :切換後のレシピ設定 No.を保存する

戻り値 : [out] なし

切換後のレシピ設定 No.を保存しないで,レシピ No:1 に変更する例を以下に示します.

使用例

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

3.7.7. CaoController::Execute ("ChangeRecipeString") コマンド

レシピ設定 No を指定して,開いているダイアログをすべて閉じて,指定された名称のレシピ設定に切り換えます.

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

bstrRecipeName	:	[in] :レシピ設定名称(VT_BSTR)
lSave	:	[in] 切換後のレシピ設定保存(VT_I4)
		0 :切換後のレシピ設定 No.を保存しない(省略時)
		1 :切換後のレシピ設定 No.を保存する
戻り値	:	[out] なし

切換後のレシピ設定 No.を保存しないで,レシピ名称:Recipe1 に変更する例を以下に示します.

使用例

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

3.8. その他

3.8.1. CaoController::Execute ("InputSimConsole") コマンド

通信コマンドで,コンソールのボタン操作と同様の入力をできます.

キーコードを 2 つ入力した場合は同時押しの扱いになります.

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

lKeyCode1	:	[in]キーコード(VT_I4)
lKeyCode2	:	[in]キーコード(VT_I4)
戻り値	:	[out] なし

キーコードの値については, 以下の通りです.

0: 0 キー, 1: 1 キー, 2: 2 キー, 3: 3 キー, 4: 4 キー, 5: 5 キー, 6: 6 キー, 7: 7 キー, 8: 8 キー

11: 左下キー, 12: 下キー, 13: 右下キー, 14: 左キー, 16: 右キー, 17: 左上キー, 18: 上キー, 19: 右上キー

※パソコンのテンキーに対応しています. 11～19 は 5 キーを中心として,上下左右の方向にあるキーの数

字に+10 した値.

0 キーと 1 キーを同時入力する例を以下に示します.

使用例

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

3.8.2. CaoController::Execute ("SearchUnitNo") コマンド

フロー内から指定した名称を持つユニットを検索し,ユニット番号を返します. 同じ名称を持つユニットが複数存在する場合は,最も小さいユニット番号を返します.

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

bstrParam	:	[in] 文字列または, スカラ型配列変数(VT_BSTR)
bIsScalar	:	[in] <bstrParam>の引数がスカラ型配列変数かを指定します. (VT_BOOL) FALSE :<bstrParam>を文字列として処理します. (デフォルト) TRUE : <bstrParam>をスカラ型配列変数として処理します.
戻り値	:	[out] なし

名称:Capture を指定し,そのユニット番号を返す例を以下に示します.

使用例

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

3.8.3. CaoController::Execute ("ReadVersionInfo") コマンド

コントローラーのシステム情報を読み出します.

戻り値のシステム情報は,型式,ファームウェアバージョンの順で格納されます.

書式 ReadVersionInfo ()

引数	:	[in] なし
戻り値	:	[out] システム情報(VT_BSTR | VT_ARRAY)

システム情報を取得する例を以下に示します.

使用例

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

3.9. 独自拡張コマンド

3.9.1. CaoController::Execute ("ExecuteCommand") コマンド

指定した無手順コマンドを実行します. コマンド実行の成否にかかわらずコマンドの応答を取得します.

サポートする無手順コマンドについては, XGX シリーズのユーザーズマニュアルを参照してください.

書式　ExecuteCommand(< bstrCommand >)

bstrCommand　:　[in] コマンドの文字列を指定します(VT_BSTR)

戻り値　:　[out] コマンドの応答を返します(VT_BSTR)

無手順コマンドを指定し, XGX を運転モードに切替える場合の例を以下に示します.

使用例

```
Dim strRet as String

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

3.9.2. CaoController::Execute ("ExecuteCommandAsync") コマンド (非推奨)

指定した無手順コマンドを非同期で実行します.

サポートする無手順コマンドについては, XGX シリーズのユーザーズマニュアルを参照してください.

コマンドの戻り値は GetCommandResult コマンドで取得し, 確認してください. GetCommandResult コマンドの詳細については, 3.9.8.CaoController::Execute ("GetCommandResult") コマンドを参照ください.

コマンド使用時は, 必ず GetCommandResult コマンドを実行後にプロバイダを切断してください.

非同期処理で異常が発生する場合があります.

書式　ExecuteCommandAsync(< bstrCommand >)

bstrCommand　:　[in] コマンドの文字列を指定します(VT_BSTR)

戻り値　:　[out] なし

無手順コマンドを指定し, XGX を運転モードに切替える場合の例を以下に示します.

使用例

```
Dim vntResult as Variant

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

' ExecuteCommandAsync コマンドの戻り値取得
vntResult = caoCtrl.Execute("GetCommandResult")

' vntResult : 無手順コマンドの戻り値 (BSTR)
```

3.9.3. CaoController::Execute ("TriggerAndGetResult") コマンド

トリガをかけた後に結果の受信を行います. 出力結果が受信できない場合は, Timeout 時間まで待ちます.

書式　TriggerAndGetResult(< iTriggerNo >)

iTriggerNo	:	[in] トリガ番号 1～4 (VT_I2)
戻り値	:	[out] 出力結果(VT_BSTR)

例を以下に示します.

使用例

```
Dim strRet as String

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

3.9.4. CaoController::Execute ("RecievePacket") コマンド

パケットの受信を行います. 受信バッファにパケットがある場合は, 受信バッファからパケットの取得を行います.

書式　RecievePacket( )

引数	:	[in] なし
戻り値	:	[out] 受信パケット(VT_BSTR)

Trigger コマンドとの組み合わせ例を下記に示します.

使用例

```
Dim strRet as String

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

3.9.5. CaoController::Execute ("ClearPacket") コマンド

受信バッファにあるパケットを消去します.

書式　ClearPacket( )

引数	:	[in] なし
戻り値	:	[out] なし

例を以下に示します.

使用例

```
caoCtrl.Execute "ClearPacket"
```

3.9.6. CaoController::Execute ("SetTimeout") コマンド

コマンド送受信のタイムアウト時間を設定します

書式　SetTimeout(< iTimeout >)

iTimeout	:	[in] タイムアウト時間(msec) (VT_UI4)
戻り値	:	[out] なし

使用例を示します.

使用例

```
caoCtrl.Execute "SetTimeout", 1000
```

3.9.7. CaoController::Execute ("GetTimeout") コマンド

設定されているコマンド送受信のタイムアウト時間を取得します.

書式　GetTimeout( )

引数	:	[in] なし
戻り値	:	[out] タイムアウト値(msec) (VT_UI4)

例を以下に示します.

使用例

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

3.9.8. CaoController::Execute ("GetCommandResult") コマンド

非同期コマンドの完了待ちを行い, 非同期コマンドの戻り値を取得します.

戻り値がない非同期コマンドを実行した場合, 戻り値はありません.

また, 同期コマンドの後で使用した場合は, GetCommandResult コマンド実行時に結果取得エラー(0x80100003)となり, 戻り値はありません.

非同期コマンドの実行でエラーが発生した場合, 非同期コマンドの実行時にはエラーは発生せず,

GetCommandResult コマンド実行時にエラーとなります.

非同期コマンドの完了待ちの際, 設定されているタイムアウト時間以内に応答がない場合,

タイムアウトエラー(0x80000900)が発生します.

非同期コマンド実行後に別のコマンドを実行した場合は，先に実行した非同期コマンドの結果は削除されますのでご注意ください．

書式　[< vntRet > =]GetCommandResult( )

引数　：　[in] なし

vntRet　：　[out] 非同期コマンドの戻り値（VT_VARIANT）

非同期で検査を実行した場合の，戻り値を取得する例を以下に示します．

使用例

```
Dim vntResult as Variant

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