DENSO

DENSO Robotics

THIRD PARTY PRODUCTS

【サードパーティ プロダクト】

PROVIDER MANUAL

プロバイダ取扱説明書

メーカ

（株）キーエンス 製

製品／シリーズ

画像センサ

MODEL: XG-X シリーズ

Vision

はじめに

本書は、「(株)キーエンス製・画像センサ・XG-Xシリーズ」をデンソーロボットコントローラRC8シリーズと接続して使用するためのプロバイダの取扱説明書です。

ご注意：(1)取扱説明書に記載された内容以外でご使用された場合、機能・性能の保証はできませんのでご注意ください。
(2) 本書に掲載されている会社名や製品は、一般に各社の商標または登録商標です。

本書が扱う対象製品

(株)キーエンス製　　XG-X2000 シリーズ

以下、上記製品をXG-Xシリーズと表記します。

お願い

ご使用前に「マニュアル・安全上のご注意」をお読みいただき正しく安全にプロバイダをご使用下さい。

お客さまへ

1. ご使用に係わるリスクについて

本製品（ソフトウェア）のシステム組み込み・使用ならびに本製品の使用結果に関する一切のリスクは、本製品の使用者に帰属するものとします。

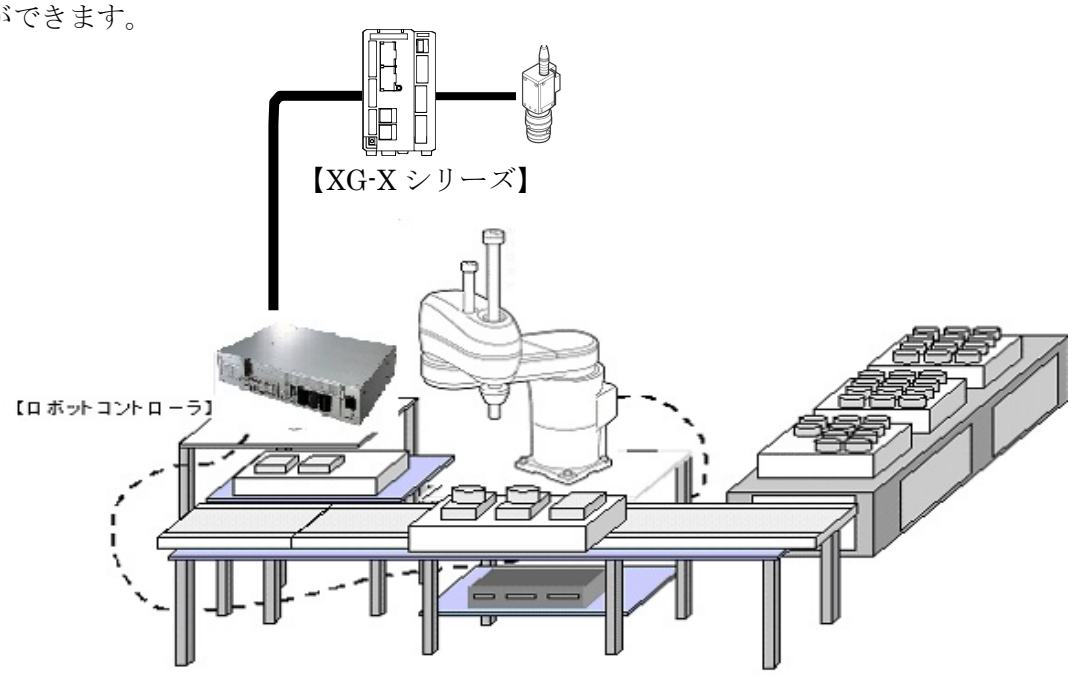
目次

はじめに 2
お願い 2
お客さまへ 2
1. 本製品（プロバイダ）の概要 4
1.1. プロバイダの対象機器 4
1.2. プロバイダの特長 4
1.3. プロバイダの仕組み 5
2. 接続方法 6
2.1. Ethernet 接続例 6
2.2. RS232C 接続例 6
3. 通信設定 7
3.1. Ethernet の場合 7
3.1.1. XG-X シリーズの通信設定 7
3.1.2. ロボットコントローラの通信設定 9
3.2. RS232C の場合 10
3.2.1. XG-X シリーズの通信設定 10
3.2.2. ロボットコントローラの通信設定 10
4. プロバイダ実行手順 11
5. コマンドの説明 12
表 5-1 コマンド一覧 13
6. エラーコード 58
7. サンプルプログラム 58
改訂履歴 59

1. 本製品（プロバイダ）の概要

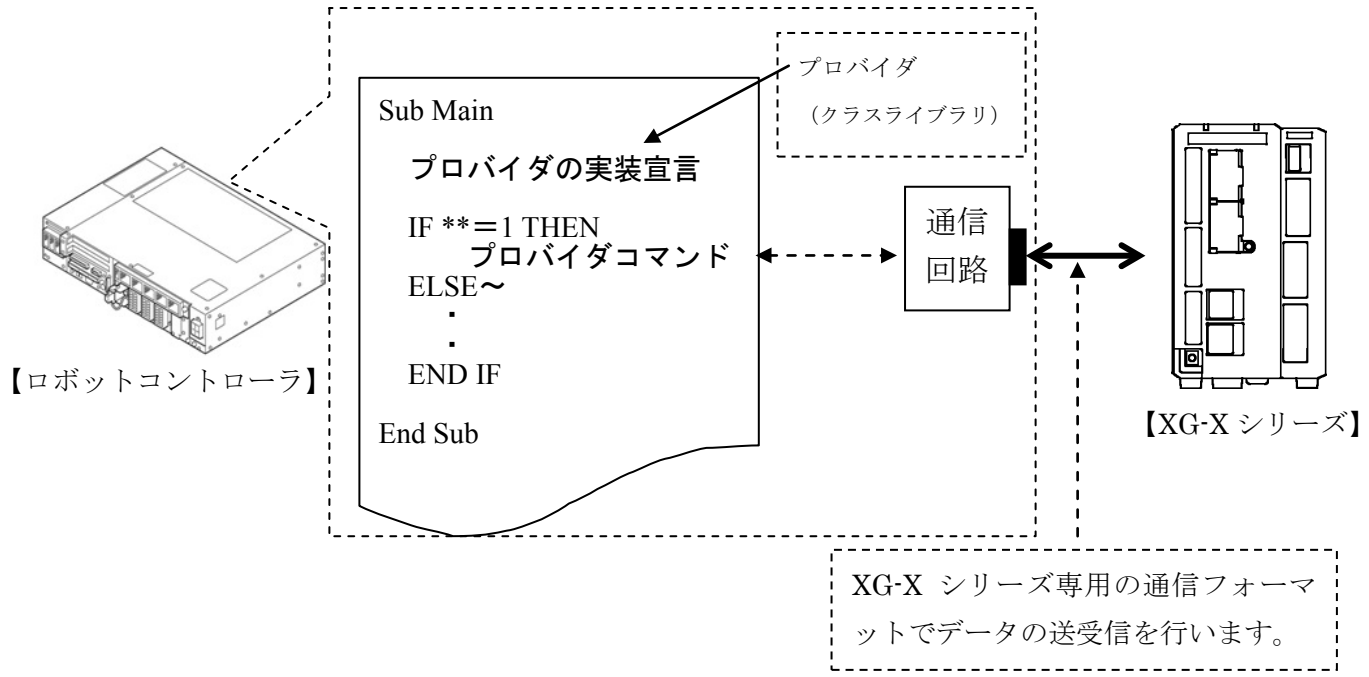
1.1. プロバイダの対象機器

本プロバイダは、デンソーロボットコントローラ（RC8シリーズ）とXG-Xシリーズを接続する時に使用することができます。



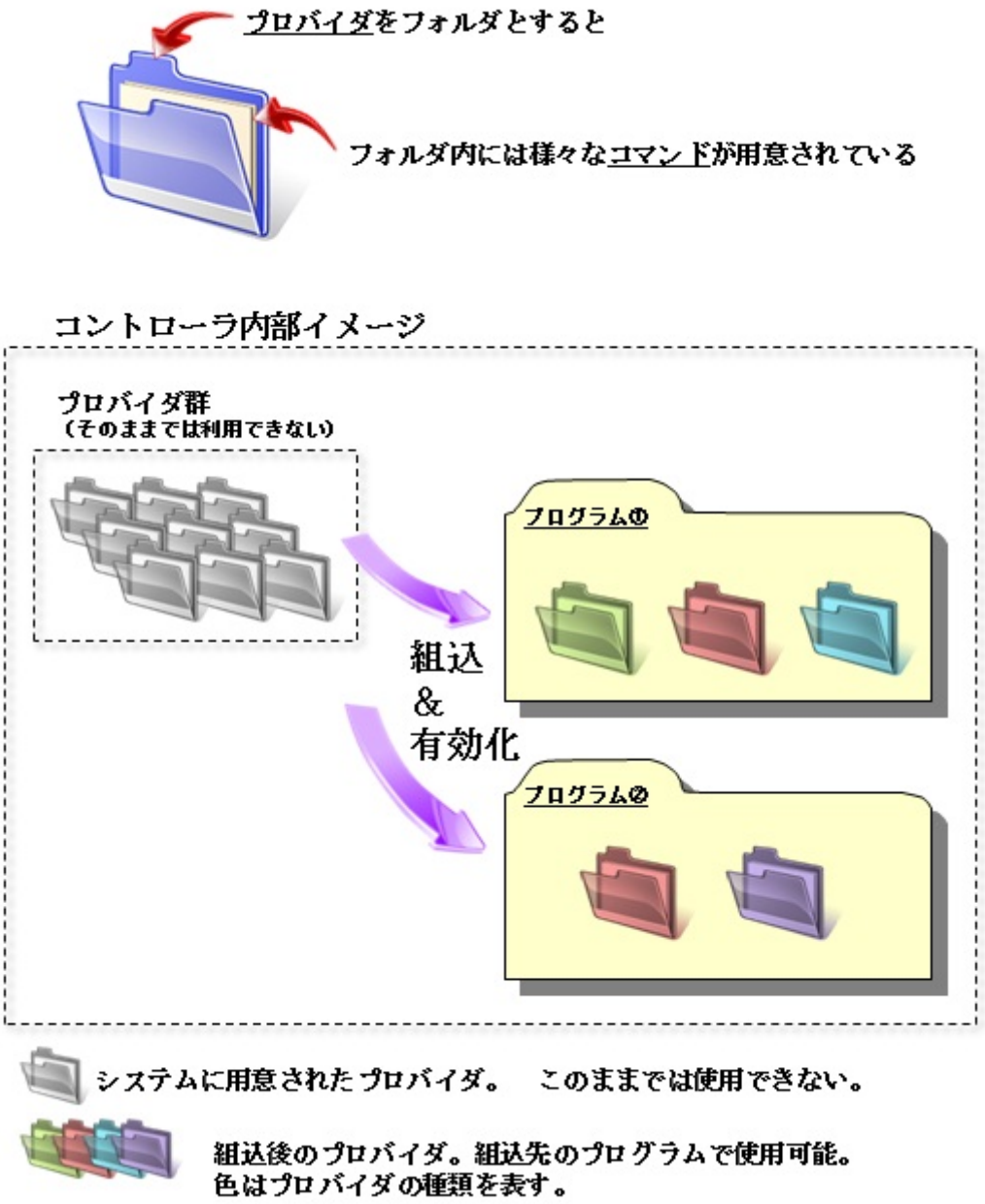
1.2. プロバイダの特長

XG-Xシリーズにアクセスするために必要なXG-Xシリーズのネイティブコマンドを、ロボットプログラムで使用できるプロバイダとして準備しています。本プロバイダを使用することで、XG-Xシリーズの通信部分のプログラムを組むことなく、容易にロボットからの通信を行うことができます。下記にプロバイダの組込関係を示します。



1.3. プロバイダの仕組み

本プロバイダは対象製品の制御を行うための各種プログラムをひとつのプロバイダとして提供しています。使用にあたってはライセンスを有効化するだけで使用することができます。使用したいプログラムファイルで実装の宣言をすれば、それ以降はプロバイダが用意する関数をコマンドとしてユーザープログラム内で使用することが可能です。本プロバイダはコントローラ内にあらかじめ用意されていますので、インストール作業は不要です。また、違う種類のプロバイダであれば複数個実装することも可能です。ただし、同じ種類のプロバイダは同じプログラム（プロシージャ）内で存在することはできません。



注：上図のプロバイダ  の様に、同一のプロバイダがプログラム別に存在する場合は、プログラム（タスク）間で排他処理を行う必要があります。

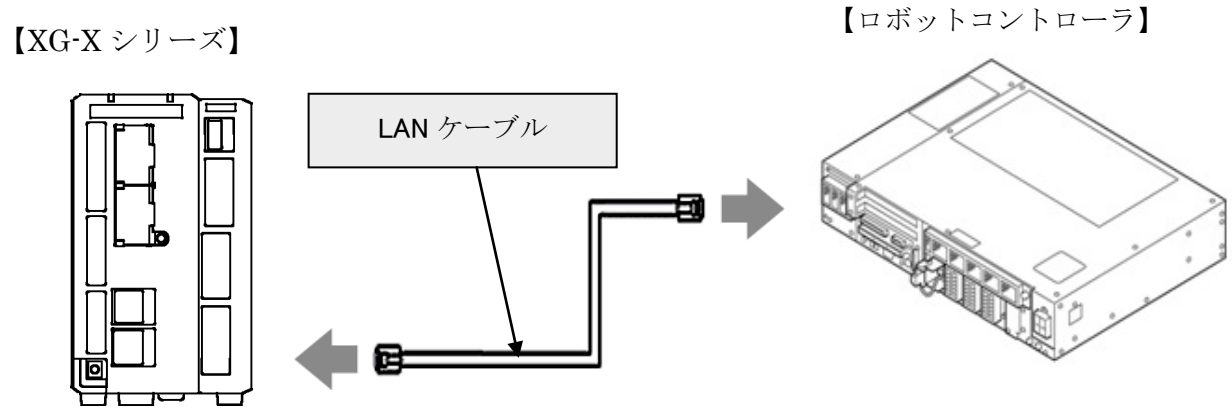
※プロバイダは PacScript から使用できるダイナミックリンクライブラリ（以下 DLL）として用意されています。

2. 接続方法

ロボットコントローラとXG-Xシリーズは、EthernetかRS232Cによって通信することができます。接続する際は各種通信用のケーブルで接続します。各種通信用ケーブルの詳細は、キーエンス社のXG-Xシリーズセットアップマニュアルを参照してください。

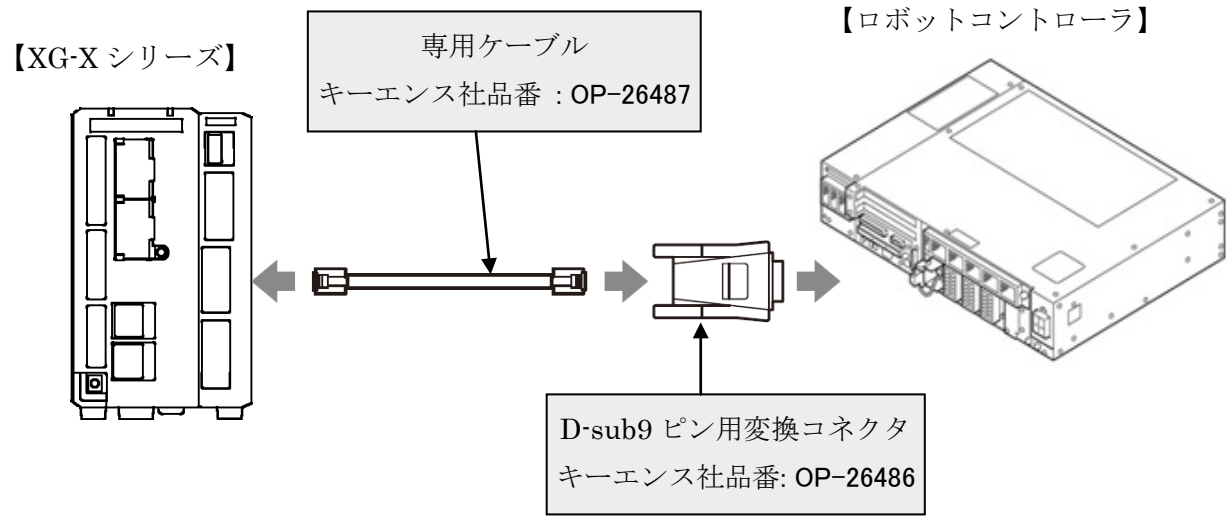
2.1. Ethernet 接続例

Ethernet通信する際には、クロスのＬＡＮケーブルを使用して下さい。また、スイッチングハブ/ルータを使用する場合は、スイッチングハブ/ルータの仕様に合ったケーブルをご使用下さい。



2.2. RS232C 接続例

RS232C通信する際には、別売りの専用ケーブルと変換コネクタ（ それぞれキーエンス社より発売 ）を使用します。変換コネクタは2種類ありますが、ロボットコントローラのRS232Cコネクタは D-sub9ピンなので、D-sub9ピン用のものを使用します。



3. 通信設定

3.1.Ethernet の場合

3.1.1. XG-X シリーズの通信設定

XG-Xシリーズの通信設定は、XG-Xシリーズ本体に接続したモニタ（ 別売り ）に表示される設定画面を、XG-Xシリーズに同梱されているマウスで操作して行います。詳細はキーエンス社のXG-Xシリーズ　ユーザーズマニュアルを参照してください。
なお、設定項目の中で、一部の項目は、次の設定以外に設定しないでください。

設定項目	設定値
IPアドレスを自動的に取得する(BOOTP)	チェックしない
デリミタ	CR
トリガーコマンド応答をCV互換モードにする	チェックしない

ここでは、XG-X2000シリーズの設定例を説明します。
XG-X2000シリーズの設定画面を、[環境設定] - [外部入出力設定] - [ネットワーク]と操作すると、ネットワーク設定ウィンドウが表示されます。

- 「IPアドレスを自動的に取得する(BOOTP)」のチェックボックスはチェックしません。
- IPアドレス及び、サブネットマスクはロボットコントローラとXG-Xシリーズが、同一のサブネットマスク内になるように設定して下さい。ここでは、IPアドレスに192.168.0.43を、サブネットマスクに255.255.255.0を設定してします。
- ゲートウェイは必要に応じて設定してください。ここでは0.0.0.0を設定しています。
- ポート番号には任意のポート番号を設定してください。ここで設定したポート番号は、ロボットコントローラのCao.AddControllerコマンドを実行する時にオプションで指定するポート番号となります。ここでは08500を設定しています。
- デリミタは「CR」に設定します。
- 「トリガーコマンド応答をCV互換モードにする」チェックボックスはチェックしません。
- 「PCアプリケーション接続ポート番号」は本プロバイダには関係ありません。

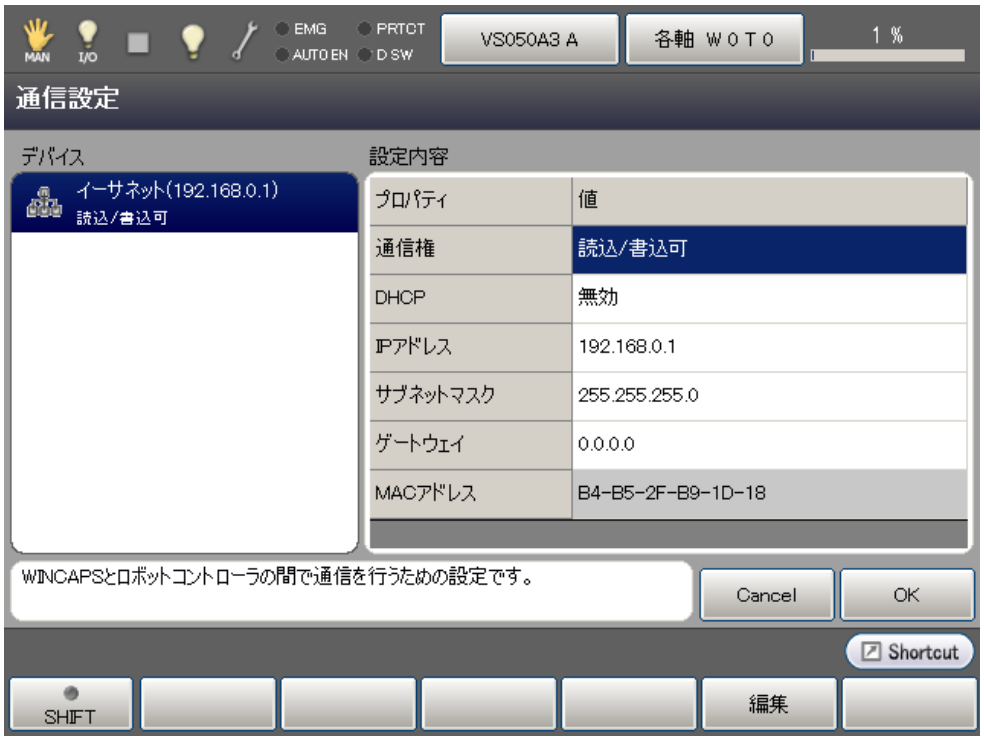
3.1.2. ロボットコントローラの通信設定

ロボットコントローラのEthernetの通信設定は、ティーチングペンダントかミニペンダントのどちらかで行えます。各設定方法はDENSO ROBOT USER MANUALSの下記内容を参照してください。

設定機器	参照箇所
ティーチングペンダント	ティーチングペンダント操作ガイドの「通信設定画面の表示・変更」
ミニペンダント	ミニペンダント操作ガイドの「DHCPの設定」、「IPアドレスの設定」

ここではティーチングペンダントによる設定例を説明します。
［F6設定］-［F5通信と起動権］-［F2ネットワークと通信権］と操作すると、通信設定ウィンドウが表示されます。

- 通信権はXG-Xシリーズとの通信には関係ありません。
- DHCPを有効にすると、IPアドレスが自動的に設定されます（ ただし、同じネットワークにDHCPサーバーがつながっている必要があります ）。ここでは無効に設定しています。
- DHCPを無効にした場合、ロボットコントローラとXG-Xシリーズが、同一のサブネットマスク内になるようにIPアドレス及び、サブネットマスクを設定して下さい。ここでは、IPアドレスに192.168.0.1を、サブネットマスクに255.255.255.0を設定してします。
- ゲートウェイは必要に応じて設定してください。ここでは0.0.0.0を設定しています。



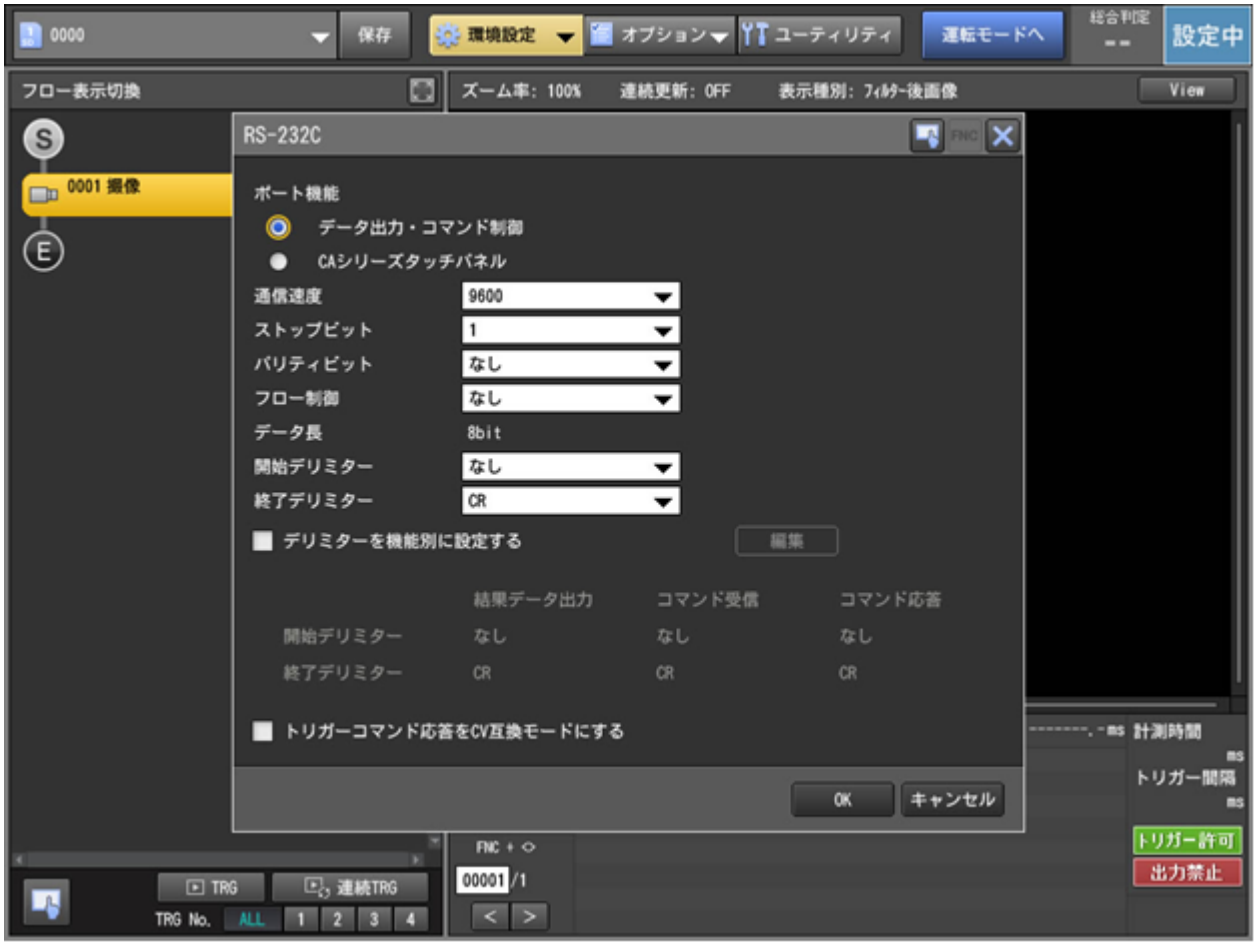
3.2. RS232C の場合

3.2.1. XG-X シリーズの通信設定

XG-XシリーズのRS232Cの通信設定は、XG-Xシリーズ本体に接続したモニタ（ 別売り ）に表示される設定画面を、XG-Xシリーズに同梱されているマウスで操作して行います。詳細はキーエンス社のXG-Xシリーズ　ユーザマニュアルを参照してください。
なお、設定項目の中で、一部の項目は、次の設定以外に設定しないでください。

設定項目	設定値
フロー制御	なし
デリミタ	CR
トリガーコマンド応答をCV互換モードにする	チェックしない

ここでは、XG-X2000シリーズの設定例を説明します。
XG-X2000シリーズの設定画面を、[環境設定] - [外部入出力設定] - [RS-232C]と操作すると、RS-232C（無手順）の設定ウィンドウが表示されます。上記設定項目以外は任意に設定してください。



3.2.2. ロボットコントローラの通信設定

ロボットコントローラのRS232Cの通信設定は、Cao.AddControllerコマンド実行時のオプションで指定します。XG-XシリーズのRS232Cの通信設定にあわせてオプションを指定してください。
なお、ティーチングペンダントまたはミニペンダントでもRS232Cの通信設定を設定できますが、この設定は、Comm.Openコマンドを実行する時に使用される設定で、本プロバイダでは使用されません。

4. プロバイダ実行手順

プロバイダは実装(宣言)→実行が基本の手順になります。本プロバイダは実装時に接続処理を行います。操作は必要な分だけ繰り返す事が出来ます。プログラム例を下記に示します。

```
Sub Main

    On Error Goto ErrorProc     ①                   '異常処理ルーチンの宣言
    Dim caoCtrl as Object       ②                   'プロバイダ用変数宣言
    Dim vntResult as Variant    ③                   '結果取得用変数宣言

    caoCtrl = Cao.AddController("XGX", "CaoProv.KEYENCE.XGX", "", "conn=eth:192.168.0.10")    ④

    「トリガ　～　データ受信処理を記述」          ⑤

EndProc:
    '終了処理
    Exit Sub

ErrorProc:
    '異常処理

End Sub
```

① 必要があればプロバイダ異常時の処理ルーチンを宣言します。(宣言時の接続異常検出)
② プロバイダを実装させる変数を Object 型で宣言します。変数名は任意に指定できます。
③ 結果を取得する変数を宣言します。データ型はコマンドにより違います。
④ プロバイダ宣言コマンド Cao.AddController で実装します。設定に必要なパラメータはプロバイダで違います。これ以降は実装変数 caoCtrl を利用してプロバイダコマンドを利用できます。
⑤ これ以降は、プロバイダコマンドを使用したプログラムが記述可能です。

5. コマンドの説明

本章では各コマンドについて説明します。コマンドは
- 接続コマンド
- XG-Xシリーズ対応コマンド
- 独自拡張コマンド

に分類されます。XG-Xシリーズ対応コマンドは、XG-Xシリーズに用意されているコマンド（XG-Xシリーズコマンド ）と対になっているコマンドです。XG-Xシリーズ対応コマンドに対する、XG-Xシリーズコマンドについては次のコマンド一覧を参照してください。また、XG-Xシリーズコマンドの詳細動作については、キーエンス社のXG-Xシリーズ ユーザーズマニュアルを参照してください。

表 5-1 コマンド一覧

XG-X シリーズプロバイダ コマンド	XG-X シリーズ コマンド	機能	参照頁
接続コマンド			
Cao.AddController	―	プロバイダを変数に実装して、XG-X シリーズへの接続処理を行います。	15
XG-X2000 シリーズ対応コマンド			
トリガー			
Trigger	T1 、T2 、T3 、T4、TA	トリガを入力します。	17
モード切替			
ChangeMode	R0、S0	運転モードまたは、停止モードに移行します。	18
ChangeModeAsync		運転モードまたは、停止モードに非同期で移行します。	18
ReadMode	RM	現在の動作モードを取得します。	19
検査設定 No.切替			
ChangeInspectSetting	PW	指定した SD カードの検査設定 No.に、設定を切り換えます。	19
ChangeInspectSettingAsync		非同期で指定した SD カードの検査設定 No.に、設定を切り換えます。	20
ChangeInspectSettingString		指定された検査設定名称の検査に設定を切り換えます。	21
ChangeInspectSettingStringAsync		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。	21
ReadInspectSetting	PR	現在設定されている検査設定 No.とその SD カード番号を取得します。	22
ReadInspectSettingString		検査設定名称を取得します。	23
コントローラー制御			
ClearError	CE	エラー状態をクリアします。	23
ReadRegisterImageNo	NR	参照先登録画像 No の読み出しを行います。	24
UpdateRegisterImageNo	NU	参照先登録画像 No の変数参照値への更新を行います。	25
WriteRegisterImageNo	NW	参照先登録画像 No の切り換えを行います。	26
RenameInspectionSetting	PN	検査設定名称書き換えを行います。	27
ReturnFlowTop	RE	フロー先頭復帰を行います。	28
UpdatePosAdjustment	RR	位置補正基準値更新を行います。	29
Reset	RS	リセットを行います。	30
ReCalcImageInfo	RU	画像基準情報の再計算を行います。	31
SaveSetting	SS	設定保存を行います。	31
ExecuteTeaching	TG	ティーチング実行を行います。	32
CancelWaitStatus	WG	待ち状態解除を行います。	32
OCR・2D コードリーダー・1D コードリーダー関連			
WriteCharReg	CW	判定文字列の書き換えを行います。	33
ReadCharReg	CR	判定文字列の読み出しを行います。	34
AutoTuning	AT	自動チューニングを行います。	35
RegisterCharLibrary	CA	辞書 1 文字登録を行います。	36
DeleteCharLibrary	CD	辞書 1 文字削除を行います。	37

データ入出力関連			
GetIntVariable	IR	整数値での変数読み出しを行います。	37
PutIntVariableEx	IS	整数値での変数同期書き込みを行います。	38
PutIntVariable	IW	整数値での変数書き込みを行います。	39
GetVariable	MR	変数読み出しを行います。	40
PutVariableEx	MS	変数同期書き込みを行います。	41
PutVariable	MW	変数書き込みを行います。	42
GetTerminalVariable	RP	端子変数読み出しを行います。	43
PutTerminalOffset	WO	端子オフセット書き込みを行います。	43
PutTerminalVariable	WP	端子変数書き込みを行います。	44
レシピ機能関連			
CopyRecipe	RPC	レシピ設定コピーを行います。	44
MoveRecipe	RPM	レシピ設定移動を行います。	45
RenameRecipe	RPN	レシピ設定名称書き換えを行います。	46
ReadRecipe	RPR	レシピ設定 No の読み出しを行います。	47
ReadRecipeString		レシピ設定名称読み出しを行います。	47
ChangeRecipe	RPW	レシピ設定 No を指定してレシピ設定 No の切り換えを行います。	48
ChangeRecipeString	RPT	レシピ設定名称を指定してレシピ設定 No を切り換えを行います。	49
その他			
InputSimConsole	KY	コンソール擬似入力を行います。	50
SearchUnitNo	UQ	ユニット番号検索を行います。	51
ReadVersionInfo	VI	コントローラーのシステム情報を読み出します。	51
独自拡張コマンド			
ExecuteCommand	―	XG-X シリーズコマンドを XG-X シリーズコマンドの書式で実行します。	52
ExecuteCommandAsync	―	XG-X シリーズコマンドを非同期で XG-X シリーズコマンドの書式で実行します。	52
TriggerAndGetResult	―	トリガを入力し、結果データを取得します。	53
RecievePacket	―	トリガの出力に対する結果を取得します。	54
ClearPacket	―	ロボットコントローラに蓄積された結果データを消去します。	55
SetTimeout	―	タイムアウト時間を設定します。	55
GetTimeout	―	設定されているタイムアウト時間を取得します。	56
GetCommandResult	―	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。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。	57

Cao.AddController

機能　プロバイダを変数に実装して、XG-Xシリーズへの接続処理を行います。

書式　Cao.AddController(<コントローラ名>,<プロバイダ名>,
<プロバイダの実行マシン名>,<オプション>)

引数　<コントローラ名>

任意名を付けて下さい（名前で管理しています）（文字列）。

<プロバイダ名>

"CaoProv.KEYENCE.XGX"を指定してください（文字列）。

<プロバイダの実行マシン名>

""を指定してください（文字列）。

<オプション>

次の項目を指定します（文字列）。

書式　"Conn=<接続パラメータ>,Timeout=<タイムアウト時間>"

引数　<接続パラメータ>

通信方法によって異なります。次の「接続パラメータの説明」を参照してください。

<タイムアウト時間>

本プロバイダのコマンド実行時に、XG-Xシリーズからの応答を待つ時間（ ミリ秒 ）を指定します。省略可能で、省略すると500（ ミリ秒 ）が指定されます。

接続パラメータの説明

Ethernetの場合

書式　"eth:<IPアドレス>:<ポート番号>"

引数　<IPアドレス>

接続するXG-XシリーズのIPアドレスを指定します。

<ポート番号>

接続するXG-Xシリーズのポート番号を指定します。省略可能で、省力すると8500が設定されます。

RS232Cの場合

書式　com:<COMポート番号>:<通信速度>:<パリティ>
:<データビット数>:<ストップビット数>:<フロー制御>

引数　<COMポート番号>
接続するXG-XシリーズにつながっているロボットコントローラのCOMポート番号を指定します。
指定した番号がそのままCOMポートの番号となります。たとえば1を指定した場合、COM1を指定したこととなります。
ロボットコントローラのフロント側のシリアル通信用コネクタを使用する場合で増設RS232Cを使用していない場合は、2を指定してください。

<通信速度>
接続するXG-Xシリーズの通信速度に合わせて指定してください。
4800、9600、19200、38400、57600、115200（bps）のいずれかを指定することができます。
省略可能で、省略した場合、9600が指定されます。

<パリティ>
接続するXG-Xシリーズのパリティに合わせて指定してください。指定方法は次の通りです。
N： なし
E： 偶数パリティ
O： 奇数パリティ
省略可能で、省略するとNが指定されます。

<データビット数>
接続するXG-Xシリーズのデータビット数に合わせて指定してください。指定方法は次の通りです。
7： 7ビット
8： 8ビット
省略可能で、省略すると8が指定されます。

<ストップビット数>
接続するXG-Xシリーズのストップビット数に合わせて指定してください。指定方法は次の通りです。
1： 1ビット
2： 2ビット
省略可能で、省略すると1が指定されます。

<フロー制御>
フロー制御の方法を指定することができますが、XG-Xシリーズと通信する場合は、フロー制御なしを指定してください。指定方法は次の通りです。
0： フロー制御なし
1： Xon / Xoff
2： ハードウェア制御
省略可能で、省略するとフロー制御なしが指定されます。

戻り値　実装したオブジェクトです（Object）。

説明　プロバイダを変数に実装すると同時に有効にします。これ以降は実装したObject型変数を使用してプロバイダにアクセスします。（実装された変数を"実装変数"と呼びます。）

用例

```
Dim caoCtrl as Object

'====================== Ethernet通信の場合 ======================
'ポートを省略する場合
caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX", _
                                        "", "conn=eth:192.168.0.10")

'ポートも指定する場合
caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX", _
                                        "", "conn=eth:192.168.0.10:8503")

'====================== RS232C通信の場合 ======================
'通信速度以降を省略する場合
caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX", _
                                        "", "conn= com:2")

'通信速度以降も指定する場合
caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX", _
                                        "", "conn= com:2:115200:E:8:1:0")

'通信速度以降も指定する場合（ パリティのみ指定して他は省略する場合 ）
caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX", _
                                        "", "conn= com:2::E:::")
```

実装変数.Trigger

機能 トリガを入力します。

書式 実装変数.Trigger <トリガ番号>

引数 <トリガ番号>

入力対象のトリガ番号を指定します（整数）。

1～4 ： トリガ1～4

-1 ： 全トリガ

戻り値 なし。

説明 トリガを入力します。結果を出力設定している場合はRecievePacketコマンドで受信してください。トリガの入力と結果を取得する処理を一つのコマンドで行いたい場合は、TriggerAndGetResultコマンドを使用してください。

用例

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'トリガ1にトリガを入力する。
caoCtrl.Trigger 1
```

実装変数.ChangeMode

機能 運転モードまたは、停止モードに移行します。

書式 実装変数.ChangeMode <モード番号>

引数 <モード番号>

変更先のモードを指定します（整数）。

0： 停止モード

1： 運転モード

戻り値 なし

説明 運転モードまたは、停止モードに移行します。

用例

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'運転モードに切り替える。
caoCtrl.ChangeMode 1
```

実装変数.ChangeModeAsync

機能 運転モードまたは、停止モードに非同期で移行します。

書式 実装変数.ChangeModeAsync <モード番号>

引数 <モード番号>

変更先のモードを指定します（整数）。

0： 停止モード

1： 運転モード

戻り値 なし

説明 運転モードまたは、停止モードに非同期で移行します。
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。

用例

```
Dim caoCtrl as Object
Dim vntResult as Variant

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'運転モードに切り替える。
caoCtrl.ChangeModeAsync 1

'ChangeModeAsyncコマンドの戻り値取得
vntResult = caoCtrl.GetCommandResult
```

実装変数.ReadMode

機能 現在の動作モードを取得します。

書式 実装変数.ReadMode ( )

引数 なし

戻り値 動作モード（整数）。

0 ： 設定モード

1 ： 運転モード

説明 現在の動作モードを取得します。

用例

```
Dim caoCtrl as Object
Dim lMode as Long

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'現在の動作モードを取得します。
lMode=caoCtrl.ReadMode
```

実装変数.ChangeInspectSetting

機能 指定したSDカードの検査設定No.に、設定を切り換えます。

書式 実装変数.ChangeInspectSetting <SDカード番号>,<検査設定No.>

引数 <SDカード番号>

SDカード番号を指定します。（整数）

1 ： SD1

2 ： SD2

<検査設定No.>

検査設定No.を指定します。（整数）0～999の間で指定してください。

戻り値 なし。

説明 指定したSDカードの検査設定No.に、設定を切り換えます。

用例

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'SD1の検査設定No.1に設定を切り換える。
caoCtrl.ChangeInspectSetting 1,1
```

実装変数.ChangeInspectSettingAsync

機能　非同期で指定したSDカードの検査設定No.に、設定を切り換えます。

書式　実装変数.ChangeInspectSettingAsync <SDカード番号>,<検査設定No.>

引数　<SDカード番号>

SDカード番号を指定します。（整数）

1　SD1

2　SD2

<検査設定No.>

検査設定No.を指定します。（整数）0～999の間で指定してください。

戻り値　なし。

説明　非同期で指定した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。

用例

```
Dim caoCtrl as Object
Dim vntResult as Variant

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                    "", "conn=eth:192.168.0.10")

'SD1の検査設定No.1に設定を切り換える。
caoCtrl.ChangeInspectSettingAsync 1,1

'ChangeInspectionSettingAsyncコマンドの戻り値取得
vntResult = caoCtrl.GetCommandResult
```

実装変数.ChangeInspectSettingString

機能 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 実装変数. ChangeInspectSettingString <検査設定名称>

引数 <検査設定名称>

検査設定名称を指定します（文字列）。

戻り値 なし。

説明 検査設定名称testに設定を切り換えます。

用例

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'検査設定名称testに設定を切り換える
caoCtrl.ChangeInspectSettingString "test"
```

実装変数.ChangeInspectSettingStringAsync

機能 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。

書式 実装変数. ChangeInspectSettingStringAsync <検査設定名称>

引数 <検査設定名称>

検査設定名称を指定します（文字列）。

戻り値 なし。

説明 検査設定名称testに設定を切り換えます。

用例

```
Dim caoCtrl as Object
Dim vntResult as Variant

'検査設定名称をtestに切り換える。
caoCtrl.ChangeInspectSettingStringAsync "test"

'ChangeInspectionSettingAsyncコマンドの戻り値取得
vntResult = caoCtrl.GetCommandResult
```

実装変数.ReadInspectSetting

機能 現在設定されている検査設定No.とそのSDカード番号を取得します。

書式 実装変数.ReadInspectSetting

引数 なし

戻り値 次の2つの値が整数型の配列に格納されます。

<SDカード番号>

現在設定されているSDカード番号です。

1 ： SD1

2 ： SD2

<検査設定No. >

現在設定されている検査設定No.です。

説明 現在設定されている検査設定No.とそのSDカード番号を取得します。

用例

```
Dim caoCtrl as Object
Dim vntRet as Variant
Dim iaryData(1) as Integer

caoCtrl=Cao.AddController("XGX","CaoProv.KEYENCE.XGX","", _
                                                "conn=eth:192.168.0.10")

'現在設定されている検査設定No.とそのSDカード番号を取得する。
'iaryData(0)にはSDカード番号が
'iaryData(1)には検査設定No.が格納される。
vntRet = caoCtrl.ReadInspectSetting
iaryData(0) = vntRet(0)
iaryData(1) = vntRet(1)
```

実装変数.ReadInspectSettingString

機能 検査設定名称を取得します。

書式 実装変数. ReadInspectSettingString

引数 なし

戻り値 <検査設定名称>

現在設定されている検査設定名称です（文字列）。

説明 現在設定されている検査設定名称を取得します。

用例

```
Dim caoCtrl as Object
Dim strRet as String

caoCtrl=Cao.AddController("XGX","CaoProv.KEYENCE.XGX","", _
                                                "conn=eth:192.168.0.10")

strRet = caoCtrl.ReadInspectSettingString
```

実装変数.ClearError

機能 エラー状態をクリアします。

書式 実装変数.ClearError

引数 なし。

戻り値 なし。

説明 エラー状態をクリアします。エラー状態でないときも、正常終了します。

用例

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                                "", "conn=eth:192.168.0.10")

'エラー状態をクリアする。
caoCtrl.ClearError
```

実装変数.ReadRegisterImageNo

機能 参照先登録画像Noの読み出しを行います。

書式 1 画像番号を指定しない場合

実装変数.ReadRegisterImageNo (<ユニットID>)

引数 <ユニットID>

ユニットIDを指定します（整数）。0～999の間で指定してください。

戻り値 <登録画像No>

登録画像No （整数）範囲は0～999です。

書式 2 画像番号を指定する場合

実装変数.ReadRegisterImageNo (<ユニットID>, < 画像番号 >)

<ユニットID>

ユニットIDを指定します（整数）。0～999の間で指定してください。

< 画像番号 >

画像演算ユニットまたはC言語ユニットのユニットID指定時は元画像番号を指定します（整数）。1～2の間で指定してください。
キャリブレーションユニットID 指定時はティーチング画像番号を指定します（整数）。

戻り値 <登録画像番号>

登録画像番号 （整数）範囲は0～999です。

説明 現在計測に使用している登録画像番号を読み出します。

用例

```
Dim caoCtrl as Object
Dim lRegisterImageNo as Long

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'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。
lRegisterImageNo = caoCtrl.ReadRegisterImageNo(101)
```

実装変数.UpdateRegisterImageNo

機能 参照先登録画像Noの変数参照値への更新を行います。

書式 実装変数.UpdateRegisterImageNo <ユニットID>

引数 ユニットID

ユニットIDを指定します（整数）。

0～999	指定するユニットID
-1	すべてのユニット

戻り値 なし。

説明 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。

用例

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'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。
caoCtrl.UpdateRegisterImageNo 101
```

実装変数.WriteRegisterImageNo

機能 参照先登録画像Noの切り換えを行います。

書式 1 画像番号を指定しない場合

実装変数.WriteRegisterImageNo <ユニットID>, <登録画像番号>

引数 <ユニットID>

ユニットIDを指定します（整数）

0～999	指定するユニットID
-1	すべてのユニット

<登録画像番号>

登録画像番号を指定します（整数）0～999の間で指定してください。

戻り値 なし。

書式 2 画像番号を指定する場合

実装変数.WriteRegisterImageNo (<ユニットID>, <登録画像番号>, <画像番号>)

引数 <ユニットID>

0～999	指定するユニットID
-1	すべてのユニット

<登録画像番号>

登録画像番号を指定します（整数）0～999の間で指定してください。

<画像番号>

画像演算ユニットまたはC言語ユニットのユニットIDの場合は元画像番号を指定します（整数）。1～2の間で指定してください。
キャリブレーションユニットIDの場合はティーチング画像番号を1指定します（整数）。1～16の間で指定してください。

戻り値 なし。

説明 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。

用例

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                    "", "conn=eth:192.168.0.10")

'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。
caoCtrl.WriteRegisterImageNo 101, 202
```

実装変数.RenameInspectionSetting

機能 検査設定名称書き換えを行います。

書式 実装変数. RenameInspectionSetting <SDカード番号>, <検査設定番号>, <検査設定名称> , [<名称タイプ>]

引数

<SDカード番号>

SDカード番号を指定します（整数）。1～2の間で指定してください。

<検査設定番号>

検査設定番号を指定します（整数）。0～999の間で指定してください。

<検査設定名称>

検査設定名称を書き換えるを文字列または，スカラ型配列変数で指定します。（文字列）

<検査設定名称タイプ>

<検査設定名称>のタイプを指定します。（ブール型）省略可能です。

FALSE	<検査設定名称>を文字列として処理します。（省略時）
TRUE	<検査設定名称>をスカラ型配列変数として処理します。

戻り値 なし。

説明 検査設定名称書き換えを行います。

用例

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

' SDカード番号1,検査設定番号101の検査設定名称をTestNameに書き換える
caoCtrl.RenameInspectionSetting 1, 101, "TestName", FALSE
```

実装変数.ReturnFlowTop

機能 フロー先頭復帰を行います。

書式 実装変数. ReturnFlowTop

引数 なし。

戻り値 なし。

説明 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。

用例

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'スタートユニットの次のユニットにジャンプする
caoCtrl.ReturnFlowTo
```

実装変数.UpdatePosAdjustment

機能 位置補正基準値更新を行います。

書式 1 基準値として取り込む値を指定しない場合

実装変数.UpdatePosAdjustment <ユニットID >

引数 <ユニットID >

ユニットIDを指定します（整数）

0～999	指定するユニットID
-1	すべてのユニット

戻り値 なし。

書式 2 基準値として取り込む値を指定する場合

実装変数.UpdatePosAdjustment <ユニットID >, <基準値>

引数 <ユニットID >

ユニットIDを指定します（整数）

0～999	指定するユニットID
-1	すべてのユニット

<基準値>

基準値として取り込む値を指定します。

0	最新結果
1	登録画像による再計算結果

戻り値 なし。

説明 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。

用例

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'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
caoCtrl.UpdatePosAdjustment 101
```

実装変数.Reset

機能 リセットを行います。

書式 実装変数.Reset

引数 なし。

戻り値 なし。

説明 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。

用例

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'リセットを実行する
caoCtrl.Reset
```

実装変数.ReCalcImageInfo

機能 画像基準情報の再計算を行います。

書式 実装変数. ReCalcImageInfo <ユニットID >

引数 <ユニットID >

ユニットIDを指定します（整数）

0～999	指定するユニットID
-1	すべてのユニット

戻り値 なし。

説明 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。

用例

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'ユニットID101の画像基準情報を再計算する
caoCtrl.ReCalcImageInfo 101
```

実装変数.SaveSetting

機能 設定保存を行います。

書式 実装変数.SaveSetting

引数 なし。

戻り値 なし。

説明 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。

用例

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'ユニットID101の画像基準情報を再計算する
caoCtrl.SaveSetting
```

実装変数.ExecuteTeaching

機能 ティーチング実行を行います。

書式 実装変数.ExecuteTeaching <ユニットID >

引数 <ユニットID >

ユニットIDを指定します（整数）。0～999の間で指定してください。

戻り値 なし。

説明 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。

用例

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'ユニットID101のティーチングを実行する
caoCtrl. ExecuteTeaching 101
```

実装変数.CancelWaitStatus

機能 待ち状態解除を行います。

書式 実装変数.CancelWaitStatus <合致条件>

引数 <合致条件>

合致条件に割り付けるビットを指定します。（整数）0～(2^{20}-1)の間で指定してください。

戻り値 なし。

説明 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。

用例

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'解除する待ちユニットの結果データを参照しない
caoCtrl.CancelWaitStatus 0
```

実装変数.WriteCharReg

機能

判定文字列の書き換えを行います。
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

実装変数.WriteCharReg <ユニットID >[,<行番号>,<判定文字列>[, <判定タイプ>]]

引数

<ユニットID >

ユニットIDを指定します（整数）0～999の間で指定してください。

<行番号>

行番号を指定します。（整数）省略可能です。
OCR ユニットの場合は 1～2 の間で指定してください。
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は 1～16 の間で指定してください。

<判定文字列>

判定文字列を文字列または、スカラ型配列変数で指定します。（文字列）

<判定タイプ>

<判定文字列>のタイプを指定します。（ブール型）省略可能です。

FALSE	<判定文字列>を文字列として処理します。（省略時）
TRUE	<判定文字列>をスカラ型配列変数として処理します。

戻り値

なし。

説明

書式による挙動は下記の通りとなります。

1．引数が<ユニットID >・<行番号>・<判定文字列>で<判定タイプ>が「文字列」の場合
ユニットIDの行番号の照合条件のREGの内容を文字列とします。

2．引数が<ユニットID >・<行番号>・<判定文字列>で<判定タイプ>が「スカラ型配列変数」の場合
ユニットIDの行番号の照合条件のREGの内容にスカラ型配列変数の値を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）

用例

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                    "", "conn=eth:192.168.0.10")

'ツール番号101のOCRツールの判定文字列を"DEF"に設定する。
caoCtrl.WriteCharReg 101,1,"DEF"
```

実装変数.ReadCharReg

機能

判定文字列の読み出しを行います。

CR 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

実装変数.ReadCharReg (<ユニットID >, <行番号 >[, <判定文字列>])

引数

<ユニットID >

ツール番号を指定します（整数）。0～999の間で指定してください。

<行番号>

行番号を指定します（整数）。

CR ユニットの場合は 1～2 の間で指定してください。

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は1～16の間で指定してください。

<判定文字列>

判定文字列をスカラ型配列変数で指定します。省略可能です。

戻り値

REGの内容（文字列）またはなし。

<判定文字列>を指定しなかった場合は、取得したREGの内容を返します。（文字列）

<判定文字列>を指定した場合は、戻り値はありません。

説明

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の行番号の照合条件のREGの内容を戻り値で返します。

２．<ユニットID >・<行番号>・<判定文字列>の場合

ユニットIDの行番号の照合条件のREGの内容をスカラ型配列変数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。

用例

```
Dim caoCtrl as Object
Dim bstrParam as String

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'ユニットID101のOCRユニット行番号1の文字列を取得する
bstrParam = caoCtrl.ReadCharReg(101,1)
```

実装変数.AutoTuning

機能 自動チューニングを行います。

書式 実装変数.AutoTuning <ユニットID >, <対象画像>

引数 <ユニットID >

ユニットIDを指定します。（整数）0～999の間で指定してください。

<対象画像>

チューニングする対象画像を指定します。

0	入力画像を対象にする。
1	登録画像を対象にする。

戻り値 なし。

説明 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。

用例

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'ユニットID101の登録画像を対象にして自動チューニングをする
caoCtrl.AutoTuning 101, 1
```

実装変数.RegisterCharLibrary

機能 辞書1文字登録を行います。

書式 1 最新の検出結果を辞書に登録する場合

実装変数.RegisterCharLibrary <ユニットID>, <行番号>, <文字番号>, <文字種>

引数 <ユニットID >

ユニットIDを指定します。（整数）0～999の間で指定してください。

<行番号>

検出結果の行番号を指定します。（整数）1～2の間で指定してください。

<文字番号>

検出結果の文字番号を指定します。（整数）0～20の間で指定してください。

<文字種>

登録先の文字種を指定します。（整数）-1～59の間で指定してください。

書式 2 履歴結果を辞書に登録する場合

実装変数.RegisterCharLibrary <ユニットID>, <条件>, <履歴蓄積条件>, <過去j回目>, <行番号>, <文字番号>, <文字種>

引数 <ユニットID >

ユニットIDを指定します。（整数）0～999の間で指定してください。

<履歴蓄積条件>

履歴蓄積条件を指定します。（整数）0～7の間で指定してください。

<過去j回目>

過去j回目を指定します。（整数）0～(蓄積条件回数-1)の間で指定してください。

<検出結果行番号>

検出結果行番号を指定します。（整数）1～2の間で指定してください。

<検出結果文字番号>

検出結果文字番号を指定します。（整数）1～20の間で指定してください。

<文字種>

登録先文字種を指定します。（整数）-1～59の間で指定してください。

戻り値 なし。

説明 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。

用例

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'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
caoCtrl.RegisterCharLibrary 101,1,2,3
```

実装変数.DeleteCharLibrary

機能 辞書1文字削除を行います。

書式 実装変数.DeleteCharLibrary <ユニットID >, <文字種>

引数 <ユニットID >

ユニットIDを指定します。（整数）0～999の間で指定してください。

<文字種>

文字種を指定します。（整数）-1～59の間で指定してください。

戻り値 なし。

説明 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。

用例

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'ユニットID101の文字種1を削除する
caoCtrl.DeleteCharLibrary 101, 1
```

実装変数.GetIntVariable

機能 整数値での変数読み出しを行います。

書式 実装変数.GetIntVariable (<変数名1> [,<変数名2> [,<変数名3> [,<変数名4> [,<変数名5> [,<変数名6> [,<変数名7> [,<変数名8> [,<変数名9> [,<変数名10> [,<変数名11> [,<変数名12> [,<変数名13> [,<変数名14> [,<変数名15> [,<変数名16>]]]]]]]]]]]]]]])

引数 <変数名1～16>

呼び出し元の変数名を指定します。（文字列）

戻り値 読み出し元の数値。（倍精度実数|配列）

説明 整数の値を読み出し,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。

用例

```
Dim caoCtrl as Object
Dim vntValList as VARIANT

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'変数名:#Data1,#Data2.#Data3の数値を読み出す
vntValList = caoCtrl.GetIntVariable("#Data1","#Data2","#Data3")
```

実装変数.PutIntVariableEx

機能 整数値での変数同期書き込みを行います。

書式 実装変数.PutIntVariableEx <変数名のリスト>, <指定値>

引数 <変数名>

書き込み先の変数名のリストを指定します。（文字列|配列）

<指定値>

書き込み元の数値または変数名のリストを指定します。（文字列|配列）

戻り値 なし。

説明 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。

用例

```
Dim caoCtrl as Object
Dim vntNameList as VARIANT
Dim vntValList as VARIANT

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

vntNameList = Array("#Data1", "#Data2", "#Data3")
vntValList = Array(1, 2, 3)

'変数名:#Data1,#Data2,#Data3に値:1,2,3を書き込む
caoCtrl.PutIntVariableEx vntNameList, vntValList
```

実装変数.PutIntVariable

機能 整数値での変数書き込みを行います。

書式 実装変数.PutIntVariable <変数名のリスト>, <指定値>

引数 <変数名のリスト>

書き込み先の変数名のリストを指定します。（文字列|配列）

<指定値>

書き込み元の数値または変数名のリストを指定します。（文字列|配列）

戻り値 なし。

説明 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。

用例

```
Dim caoCtrl as Object
Dim vntNameList as VARIANT
Dim vntValList as VARIANT

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

vntNameList = Array("#Data1", "#Data2", "#Data3")
vntValList = Array(1, 2, 3)

'変数名:#Data1,#Data2,#Data3に値:1,2,3を書き込む
caoCtrl.PutIntVariable vntNameList, vntValList
```

実装変数.GetVariable

機能 変数読み出しを行います。

書式 実装変数.GetVariable (<変数名1> [,<変数名2> [,<変数名3> [,<変数名4> [,<変数名5> [,<変数名6> [,<変数名7> [,<変数名8> [,<変数名9> [,<変数名10> [,<変数名11> [,<変数名12> [,<変数名13> [,<変数名14> [,<変数名15> [,<変数名16>]]]]]]]]]]]]]]])

引数 <変数名1～16>

呼び出し元の変数名を指定します。（文字列）

戻り値 読み出し元の数値。（倍精度実数|配列）

説明 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。

用例

```
Dim caoCtrl as Object
Dim vntValList as VARIANT

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'変数名:#Data1,#Data2.#Data3の数値を読み出す
vntValList = caoCtrl.GetVariable("#Data1","#Data2","#Data3")
```

実装変数.PutVariableEx

機能 変数同期書き込みを行います。

書式 実装変数.PutVariableEx <変数名のリスト>, <指定値>

引数 <変数名のリスト>

書き込み先の変数名のリストを指定します。（文字列|配列）

<指定値>

書き込み元の数値または変数名のリストを指定します。（文字列|配列）

戻り値 なし。

説明 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。

用例

```
Dim caoCtrl as Object
Dim vntNameList as VARIANT
Dim vntValList as VARIANT

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

vntNameList = Array("#Data1", "#Data2", "#Data3")
vntValList = Array(1, 2, 3)

'変数名:#Data1,#Data2,#Data3に値:1,2,3を書き込む
caoCtrl.PutVariableEx vntNameList, vntValList
```

実装変数.PutVariable

機能 変数書き込みを行います。

書式 実装変数. PutVariable <変数名のリスト>, <指定値>

引数 <変数名のリスト>

書き込み先の変数名のリストを指定します。（文字列|配列）

<指定値>

書き込み元の数値または変数名のリストを指定します。（文字列|配列）

戻り値 なし。

説明 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。

用例

```
Dim caoCtrl as Object
Dim vntNameList as VARIANT
Dim vntValList as VARIANT

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

vntNameList = Array("#Data1", "#Data2", "#Data3")
vntValList = Array(1, 2, 3)

'変数名:#Data1,#Data2,#Data3に値:1,2,3を書き込む
caoCtrl.PutVariable vntNameList, vntValList
```

実装変数.GetTerminalVariable

機能 端子変数読み出しを行います。

書式 実装変数.GetTerminalVariable (<システム変数名>)

引数 <システム変数名>

読み出すシステム変数名。（文字列）（%OutDataAsyncA～%OutDataAsyncH）

戻り値 読み出した値。（整数）

説明 指定された端子の状態を読み出します。

用例

```
Dim caoCtrl as Object
Dim lTerminalCond as Long

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'%OutDataAsyncAの端子状態を読み出す
lTerminalCond = caoCtrl.GetTerminalVariable("%OutDataAsyncA")
```

実装変数.PutTerminalOffset

機能 端子オフセット書き込みを行います。

書式 実装変数.PutTerminalOffset <オフセットの倍率>, <オフセット値>

引数 <オフセットの倍率>

オフセットの倍率を指定します。（整数）

<オフセット値>

オフセット値を指定します。（整数）

戻り値 なし。

説明 %CmdParamOffsetにオフセットの倍率×オフセット値を書き込みます。

用例

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'倍率2でオフセット値1を書き込む
caoCtrl.PutTerminalOffset 2, 1
```

実装変数.PutTerminalVariable

機能 端子変数書き込みを行います。

書式 実装変数. PutTerminalVariable <システム変数名>, <指定値>

引数 <システム変数名>

書き込むシステム変数名を指定します。（文字列）（%OutDataAsyncA ～%OutDataAsyncH）

<指定値>

書き込み元の数値または変数名を指定します。

戻り値 なし。

説明 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。

用例

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'%OutDataAsyncAに1を書き込む
caoCtrl. PutTerminalVariable "%OutDataAsyncA", 1
```

実装変数.CopyRecipe

機能 レシピ設定コピーを行います。

書式 実装変数.CopyRecipe <コピー元レシピNo >, <コピー先レシピNo>

引数 <コピー元レシピNo>

コピー元レシピNoを指定します。（整数）0～999の間で指定してください。

<コピー先レシピNo >

コピー先レシピNoを指定します。（整数）0～999の間で指定してください。

戻り値 なし。

説明 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。

用例

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'レシピNo1のレシピをレシピNo2にコピーする
caoCtrl.CopyRecipe 1, 2
```

実装変数.MoveRecipe

機能 レシピ設定移動を行います。

書式 実装変数. MoveRecipe <移動元レシピNo>, <移動先レシピNo>

引数 <移動元レシピNo>

移動元レシピNoを指定します。（整数）0～999の間で指定してください。

<移動先レシピNo>

移動先レシピNoを指定します。（整数）0～999の間で指定してください。

戻り値 なし。

説明 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。

用例

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'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
caoCtrl. MoveRecipe 1, 2
```

実装変数.RenameRecipe

機能 レシピ設定名称書き換えを行います。

書式 実装変数.RenameRecipe <レシピ設定No>, <レシピ設定名称>[, <設定タイプ>]

引数 <レシピ設定No>

レシピ設定Noを指定します。（整数）0～999の間で指定してください。

<レシピ設定名称>

レシピ設定名称を指定します。（文字列）

<設定タイプ>

<レシピ設定名称>のタイプを指定します。（ブール型）省略可能です。

FALSE	<レシピ設定名称>を文字列として処理します。（省略時）
TRUE	<レシピ設定名称>をスカラ型配列変数として処理します。

戻り値 なし。

説明 指定したレシピ設定No.の名称を変更します。

用例

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'レシピNo1の設定名称を「Recipe1」にする
caoCtrl.RenameRecipe 1, "Recipe1", FALSE
```

実装変数.ReadRecipe

機能 レシピ設定Noの読み出しを行います。

書式 実装変数.ReadRecipe ( )

引数 なし。

戻り値 レシピ設定Noを返します。（整数）

0～999	現在のレシピ設定No
-1	設定を使用していない場合

説明 使用中のレシピ設定Noの.読み出しを行います。

用例

```
Dim caoCtrl as Object
Dim lRecipeInfoNo as long

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'使用中のレシピ設定No.を読み出す
lRecipeInfoNo = caoCtrl.ReadRecipe
```

実装変数.ReadRecipeString

機能 レシピ設定名称読み出しを行います。

書式 実装変数.ReadRecipeString ( )

引数 なし。

戻り値 レシピ設定名称を返します。（文字列）

説明 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。

用例

```
Dim caoCtrl as Object
Dim strRecipeInfoName as string

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'使用中のレシピ設定No.を読み出す
strRecipeInfoName = caoCtrl.ReadRecipeString
```

実装変数.ChangeRecipe

機能 レシピ設定Noを指定してレシピ設定Noの切り換えを行います。

書式 実装変数.ChangeRecipe <レシピ設定No>[, <保存指定>]

引数 <レシピ設定No>

切り換えるレシピ設定Noを指定します。（整数）

-1	レシピ設定を使用しない。
0～999	レシピ設定No.を指定します。

<保存指定>

保存をする/しないを指定します。（整数）省略可能です。

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 なし。

説明 開いているダイアログをすべて閉じて、指定された番号のレシピ設定に切り換えます。

用例

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                    "", "conn=eth:192.168.0.10")

'切換後のレシピ設定No.を保存しないで、レシピNo.を1に変更する
caoCtrl.ChangeRecipe 1, 0
```

実装変数.ChangeRecipeString

機能 レシピ設定名称を指定してレシピ設定Noを切り換えを行います。

書式 実装変数. ChangeRecipeString <レシピ設定名称>[, <保存指定>]

引数 <レシピ設定名称>

切り換えるレシピ設定名称を指定します。（整数）

<保存指定>

保存をする/しないを指定します。（整数）省略可能です。

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 なし。

説明 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。

用例

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'切換後のレシピ設定No.を保存しないで、レシピ名称を「Recipe1」に変更する
caoCtrl.ChangeRecipeString "Recipe1", 0
```

実装変数.InputSimConsole

機能　コンソール擬似入力を行います。
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　実装変数.InputSimConsole <キーコード1>[, <キーコード2>]

引数　<キーコード1>

キーコードを指定します。（整数）キーコードの値については下記の説明を参照してください。

<キーコード2>

キーコードを指定します。（整数）省略可能です。キーコードの値については下記の説明を参照してください。

戻り値　なし。

説明　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。
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した値になります。

用例

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'0キーと1キーを同時入力する
caoCtrl. InputSimConsole 0, 1
```

実装変数.SearchUnitNo

機能 ユニット番号検索を行います。

書式 実装変数.SearchUnitNo (<ユニット名称>[, <ユニット名称タイプ>])

引数 <ユニット名称>

ユニット名称を文字列または、スカラ型配列変数で指定します。（文字列）

<ユニット名称タイプ>

<ユニット名称>のタイプを指定します。（ブール型）

FALSE	<ユニット名称>を文字列として処理します。（省略時）
TRUE	<ユニット名称>をスカラ型配列変数として処理します。

戻り値 ユニット番号を返します。（整数）

説明 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。

用例

```
Dim caoCtrl as Object
Dim lUnitNo as long

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'ユニット名称"Capture"を指定し、そのユニット番号を返す
lUnitNo = caoCtrl.SearchUnitNo("Capture", FALSE)
```

実装変数.ReadVersionInfo

機能 コントローラーのシステム情報を読み出します。

書式 実装変数.ReadVersionInfo ( )

引数 なし。

戻り値 システム情報を返します。（文字列|配列）

説明 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。

用例

```
Dim caoCtrl as Object
Dim vntVerInfo as Variant

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'システム情報を取得する
vntVerInfo = caoCtrl.ReadVersionInfo
```

実装変数.ExecuteCommand

機能 XG-Xシリーズコマンドを XG-Xシリーズコマンドの書式で実行します。

書式 実装変数.ExecuteCommand (<XG-Xシリーズコマンド書式>)

引数 <XG-Xシリーズコマンド書式>

XG-Xシリーズコマンド書式を指定します（文字列）。

戻り値 < XG-Xシリーズコマンド実行後の受信データ>

XG-Xシリーズコマンド実行後の受信データです（文字列）。

説明 XG-Xシリーズコマンドを XG-Xシリーズコマンドの書式で実行します。XG-Xシリーズコマンドについては、キーエンス社の XG-Xシリーズ ユーザーズマニュアルを参照してください。

用例

```
Dim caoCtrl as Object
Dim strRet as String

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

' XG-Xシリーズコマンドを指定し、XGXを運転モードに切替える
strRet = caoCtrl. ExecuteCommand("R0")
```

実装変数.ExecuteCommandAsync

機能 XG-Xシリーズコマンドを非同期で XG-Xシリーズコマンドの書式で実行します。

書式 実装変数.ExecuteCommandAsync <XG-Xシリーズコマンド書式>

引数 <XG-Xシリーズコマンド書式>

XG-Xシリーズコマンド書式を指定します（文字列）。

戻り値 なし

説明 XG-Xシリーズコマンドを非同期で XG-Xシリーズコマンドの書式で実行します。XG-Xシリーズコマンドについては、キーエンス社の XG-Xシリーズ 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。

用例

```
Dim caoCtrl as Object
Dim vntResult as Variant

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")
' XG-Xシリーズコマンドを指定し、XGXを運転モードに切替える
caoCtrl.ExecuteCommandAsync "R0"

' ExecuteCommandAsyncコマンドの戻り値取得
vntResult = caoCtrl.GetCommandResult
```

実装変数.TriggerAndGetResult

機能 トリガを入力し、結果データを取得します。

書式 実装変数.TriggerAndGetResult (<トリガ番号>)

引数 <トリガ番号>

トリガ番号を指定します（整数）。

1~4 ： トリガ1~4

戻り値 <結果データ>

取得した結果データです（文字列）。

説明 トリガを出力した後、結果データを取得します。結果データがXG-Xシリーズから送信されていない場合、タイムアウト時間（ Cao.AddControllerコマンドのオプションまたは、SetTimeoutコマンドで設定 ）まで待ち、それでも送信されない場合はエラーとなります。
トリガを出力した後、結果データが送信されるまでの時間に、他の処理を行いたい場合は、Triggerコマンドでトリガを出力した後、任意の処理を実行し、その後、RecievePacketコマンドで結果データを取得してください。

用例

```
Dim caoCtrl as Object
Dim strRet as String

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'トリガ1にトリガを入力し、結果を取得する。
strRet = caoCtrl.TriggerAndGetResult(1)
```

実装変数.RecievePacket

機能 トリガの出力に対する結果を取得します。

書式 実装変数.RecievePacket

引数 なし。

戻り値 <結果データ>

取得した結果データです（文字列）。

説明 トリガの入力に対する結果を取得します。
XG-Xシリーズの設定が、トリガ入力に対し結果を出力しない設定なっている場合、結果データがXG-Xシリーズから送信されないため、タイムアウト時間（ Cao.AddControllerコマンドのオプションまたは、SetTimeoutコマンドで設定 ）経過後にエラーとなります。

注意 トリガの出力を行った後、RecievePacketコマンドを実行しないで、再びトリガの出力を行うと、2回分の結果データがロボットコントローラに蓄積されます。その後RecievePacketコマンドを実行すると、最初のトリガの出力に対する結果データを取得することになります。したがって、トリガの出力回数に対し、RecievePacketコマンドの実行回数が伴わなくなってしまった場合は、ClearPacketコマンドでロボットコントローラに蓄積された結果データを消去し、再度トリガを出力した後、RecievePacketコマンドで結果データを取得してください。

用例

```
Dim caoCtrl as Object
Dim strRet as String

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'トリガ1にトリガを入力する。
caoCtrl.Trigger 1

'結果データを取得する。
strRet = caoCtrl.RecievePacket
```

実装変数.ClearPacket

機能 ロボットコントローラに蓄積された結果データを消去します。

書式 実装変数.ClearPacket

引数 なし。

戻り値 なし。

説明 ロボットコントローラに蓄積された結果データを消去します。

用例

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'結果データを消去する。
caoCtrl.ClearPacket
```

実装変数.SetTimeout

機能 タイムアウト時間を設定します。

書式 実装変数. SetTimeout <タイムアウト時間>

引数 <タイムアウト時間>

タイムアウト時間を指定します（整数）。単位はミリ秒です。

戻り値 なし。

説明 タイムアウト時間は、Cao.AddControllerコマンド実行時に指定しますが、Cao.AddControllerコマンド実行後にタイムアウト時間を設定したい場合に使用します。

用例

```
Dim caoCtrl as Object

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'タイムアウト時間を1000ミリ秒に設定する。
caoCtrl.SetTimeout 1000
```

実装変数.GetTimeout

機能 設定されているタイムアウト時間を取得します。

書式 実装変数.GetTimeout

引数 なし。

戻り値 <タイムアウト時間>

設定されているタイムアウト時間です（整数）。単位はミリ秒です。

説明 設定されているタイムアウト時間を取得します。

用例

```
Dim caoCtrl as Object
Dim iTimeout as Integer

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'タイムアウト時間を取得する。
iTimeout = caoCtrl.GetTimeout
```

実装変数.GetCommandResult

機能 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。

書式 実装変数.GetCommandResult

引数 なし。

戻り値 <非同期コマンドの戻り値>

非同期コマンドの戻り値が格納されます。

説明 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)に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。

用例

```
Dim caoCtrl as Object
Dim vntResult as variant

caoCtrl=Cao.AddController("XGX"," CaoProv.KEYENCE.XGX ", _
                                        "", "conn=eth:192.168.0.10")

'ツール番号100のエッジツールの下限値を123.456に設定する。
caoCtrl.ExecuteCommandAsync "DW,100,82,1,123.456"

'コマンドの戻り値をGetCommandResultコマンドで取得する。
vntResult = caoCtrl.GetCommandResult
```

6. エラーコード

プロバイダのエラーの見方に関しては、DENSO ROBOT USER MANUALSの「プロバイダガイド」の「プロバイダエラーの見方」を参照してください。
プロバイダのエラーの中で、XG-Xシリーズから送信されたエラーに関しては、オリジナルナンバが80108000（16進数）～ 80108063（16進数） の範囲の番号となり、下2桁がXG-Xシリーズから送信されたエラーコードを表します。例えば、ChangeInspectSettingコマンドを実行する時に、SDカード番号に5を指定して実行すると、ロボットコントローラのエラーのオリジナルナンバは80108016（ 16進数)となります。下2桁の16（16進数 ）は10進数に変換すると22であり、キーエンス社のXG-Xシリーズ　ユーザマニュアルのPWコマンドの説明を参照すると、エラーコード22は"パラメータ数、パラメータ範囲が違うとき"ということがわかります。

エラー名	エラー番号	説明
E_CVERROR_CVERR	0x80108000～0x80108063	XG-Xシリーズ固有エラー
E_CVERROR_LENGTH	0x80100000	パケット長エラー
E_CVERROR_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。

7. サンプルプログラム

```
Sub Main
        Dim caoCtrl As Object
        Dim strRet As String

        'XG-X シリーズプロバイダの実装
        caoCtrl = Cao.AddController("XGX", "CaoProv.KEYENCE.XGX", "", _
                                    "conn=eth:192.168.0.3, timeout=1000")

        'トリガ 1 にトリガを入力し結果データを取得する。
        strRet = caoCtrl.TriggerAndGetResult(1)

        '結果データをティーチングペンダントのメッセージ出力ウインドウに出力する。
        PrintDbg strRet

        ' XG-X シリーズプロバイダを切断し削除する。
        cao.Cotrollers.Remove caoCtrl.Index
        caoCtrl = Nothing
End Sub
```

改訂履歴

デンソーロボット
プロバイダ
取扱説明書
株式会社キーエンス製 画像センサ XG-X シリーズ

バージョン	対応RC8	改訂内容
Ver.1.0.0	Ver.2.0.*	初版
Ver.1.0.1	Ver.2.2.*	34コマンド追加
Ver.1.0.2	Ver.2.2.*	サンプルプログラムの修正
Ver.1.0.3	Ver.2.3.*	誤記訂正

株式会社デンソーウェーブ

DENSO Robotics

THIRD PARTY PRODUCTS

株式会社デンソーウェーブ