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

画像処理システム

MODEL:XG シリーズ

Vision

はじめに

本書は、「(株)キーエンス製・画像処理システム・ＸＧ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。古いＸＧに関しては一部使用できない機能があります。接続機器の詳細および取扱いは、「(株)キーエンス製・画像処理システム・ＸＧシリーズ」の取扱説明書をご参照ください。

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-7000/8000 シリーズ

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

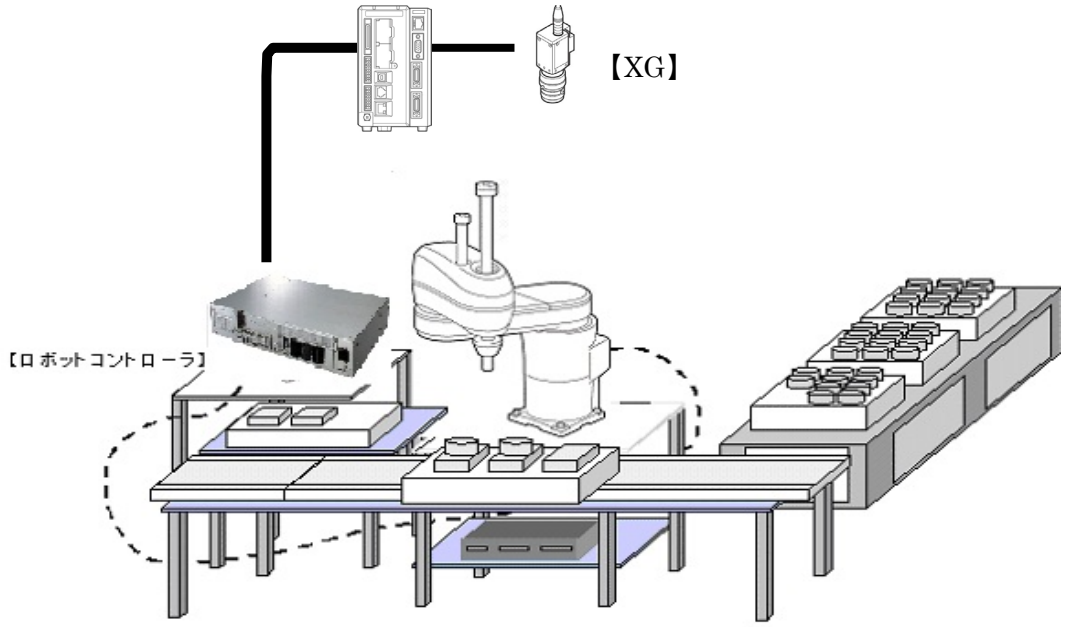
本製品（ソフトウェア）のシステム組み込み・使用ならびに本製品の使用結果に関する一切のリスクは、本製品の使用者に帰属するものとします。

目次

1. 本製品（プロバイダ）の概要

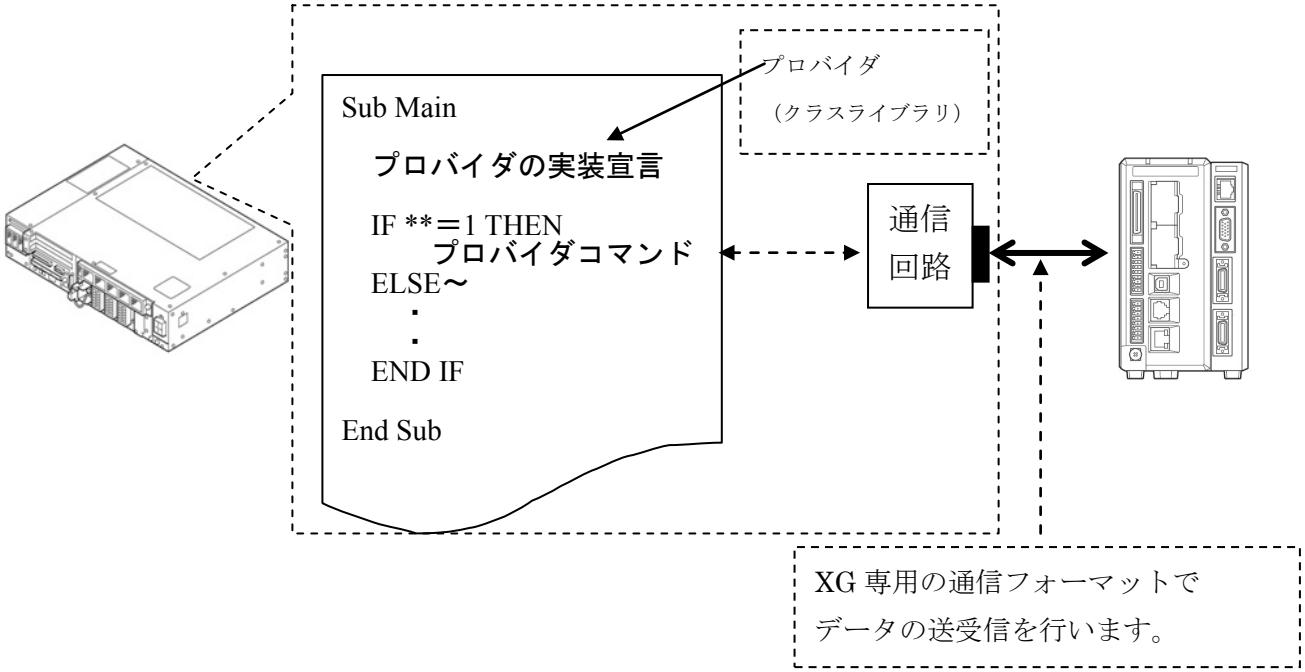
1.1 プロバイダの対象機器

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-7000/8000シリーズ（以降はＸＧと記載）を接続する時に使用することができます。



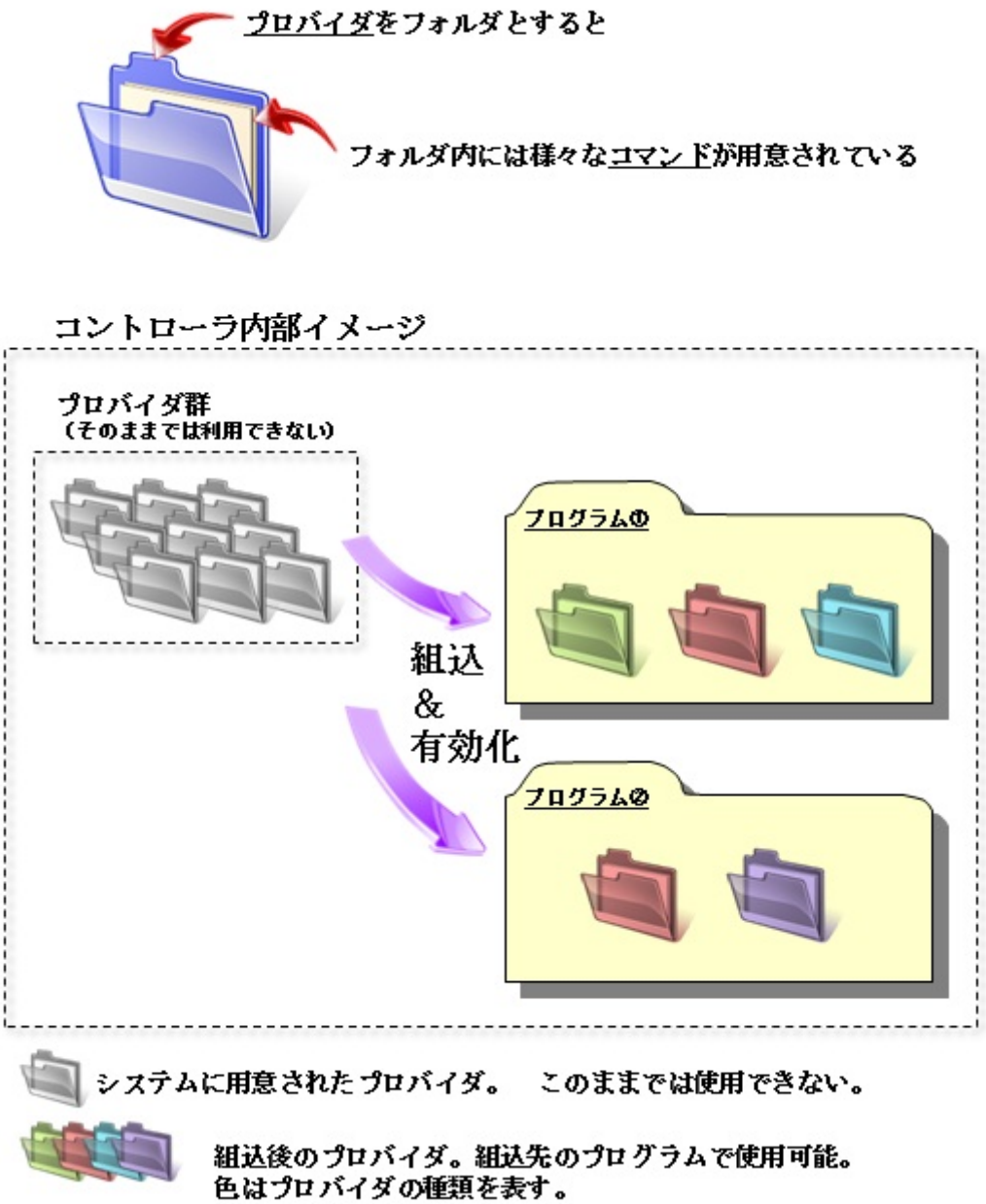
1.2 プロバイダの特長

ＸＧシリーズにアクセスするために必要なＸＧ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事が出来ます。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事が可能です。本プロバイダはコントローラ内に予め用意されていますので、インストール作業は不要です。又、違う種類のプロバイダであれば複数個実装する事も可能です。但し、同じ種類のプロバイダは同じプログラム（プロシージャ）内で存在する事は出来ません。



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。

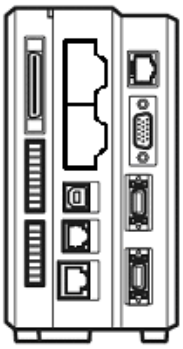
※プロバイダは PacScript から使用できるダイナミックリンクライブラリ（以下 DLL）として用意されています。

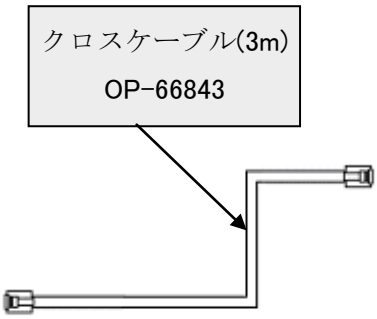
2. 接続方法

2.1 Ethernet（TCP/IP）接続例

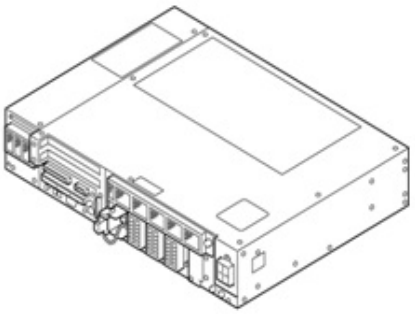
ロボットコントローラとEthernet接続する際には、オプション品の専用ケーブル（㈱キーエンス品番：OP-66843）又は、クロスのＬＡＮケーブルを使用して下さい。又、スイッチングハブ/ルータを使用する場合は、スイッチングハブ/ルータの仕様に合ったケーブルをご使用下さい。

【XG シリーズ】



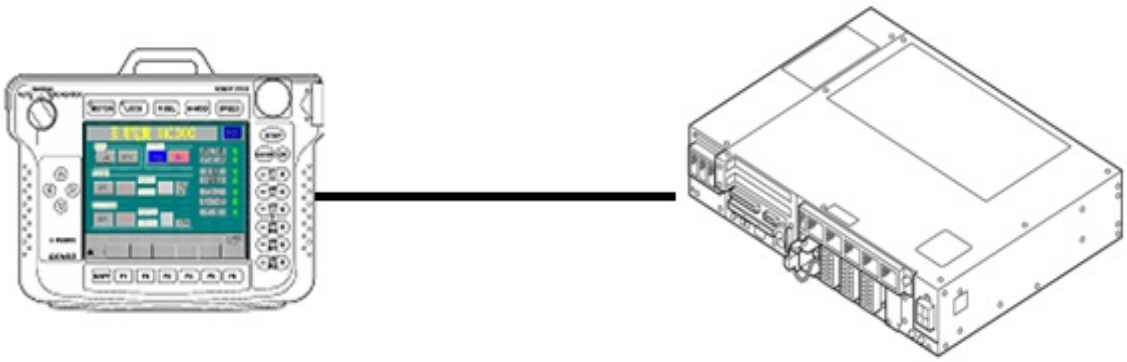


【ロボットコントローラ】



3. ロボットコントローラと使用機器の通信設定

ティーチングペンダントを使用して、各通信設定項目を使用機器に合わせて下さい。

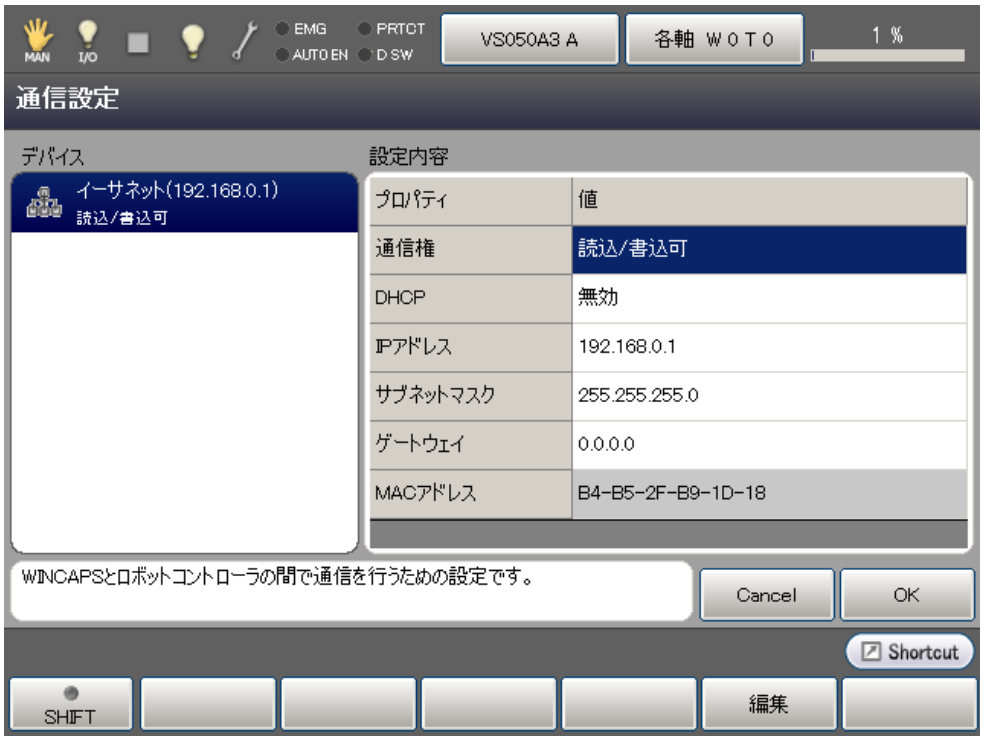


3.1 Ethernet（TCP/IP）による通信

3.1.1 ロボットコントローラの Ethernet（TCP/IP）通信設定

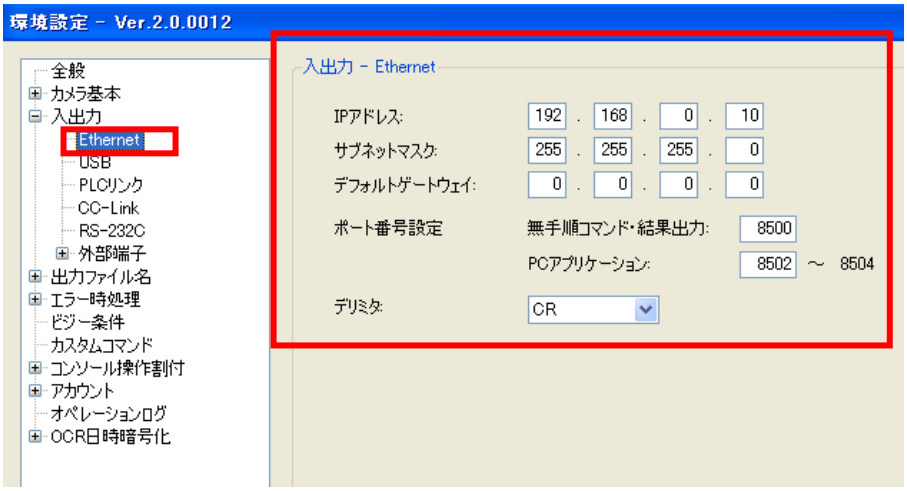
ロボットコントローラのIPアドレスを設定します。

①［F6設定］［F5通信と起動権］［F2ネットワークと通信権］で、通信設定ウィンドウが表示されます。ロボットコントローラとXGシリーズt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。



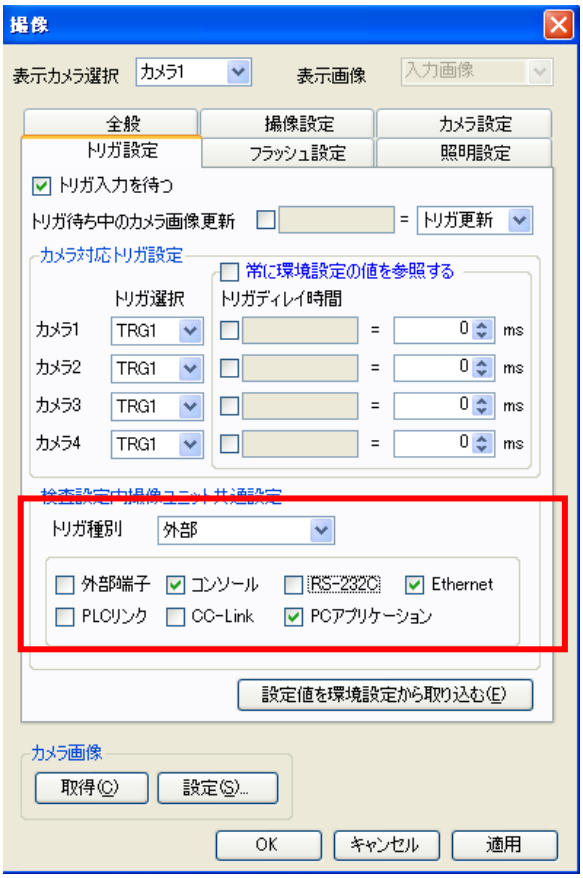
3.1.2　XG シリーズの Ethernet（TCP/IP）通信設定

XG Vision Editorの[設定メニュー]　→［環境設定］を開くと[入出力、Ethenet設定]、ウィンドウが表示されます。ロボットコントローラとXG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。又、デリミタはＣＲを設定して下さい。
※「無手順コマンド・結果出力」は8500を指定してください。（固定）



3.1.3　XG シリーズその他の設定

① XG Vision Editorで検査フローを作成後、［フロービュー］にて［撮像］→［トリガ設定］タブを選択して、［トリガ設定］を表示させます。トリガ種別は「外部」を選定し、「Ethernet」のチェックボックスにチェックを入れて下さい。



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("XG", "caoProv.KEYENCE.VWXG", "", "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 コマンド、独自拡張コマンドに分類されます。尚、XG コマンドの詳細動作については KEYENCE 社の XG-7000/8000 シリーズ用 V-Works for XG ActiveX コントロールのリファレンスマニュアルの API 一覧を参照してください。

表 5-1 コマンド一覧

コマンド	キーエンス対応コマンド	機能	参照
接続コマンド			
cao.AddController	―	プロバイダを変数に実装して、XG に接続処理を行います。	11
XG コマンド			
ChangeMode	ChangeMode	運転／停止モードを変更します。	12
ChangeModeAsync		運転／停止モードを変更します。(非同期)	13
ReadMode	ReadMode	運転／停止モードを読み出します。	14
Reset	Reset	リセットします。	15
Restart	Restart	スタートユニットの次のユニットにジャンプします。	16
RestartAsync		スタートユニットの次のユニットにジャンプします。(非同期)	17
Trigger	Trigger	トリガを発行します。	18
EnableTrigger	EnableTrigger	トリガ入力を許可／禁止します。	19
ReadTriggerEnable	ReadTriggerEnable	トリガ入力許可状態を読み出します。	20
WriteVariable	WriteVariable	変数値を書き込みます。	21
ReadVariable	ReadVariable	変数値を読み出します。	22
ChangeInspectSetting	ChangeInspectSetting	検査設定 No.を切り換えます。	23
ChangeInspectSettingAsync		検査設定 No.を切り換えます。(非同期)	24
ReadInspectSetting	ReadInspectSetting	検査設定 No.を取得します。	25
ClearError	ClearError	システムエラーをクリアします。	26
独自拡張コマンド			
ExecuteCommand	ExecuteCommand	無手順コマンドを実行します。	27
ExecuteCommandAsync		無手順コマンドを実行します。(非同期)	28
TriggerAndGetResult	―	トリガを発行し、画像処理結果を取得します。	29
GetCommandResult	―	非同期コマンドの戻り値を取得します。	30
RecievePacket	―	パケットの受信を行います。	31

cao.AddController

機能

プロバイダを変数に実装して、XG に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）
〈プロバイダ名〉 "CaoProv.KEYENCE.VWXG"
〈プロバイダの実行マシン名〉 省略して下さい
〈オプション〉［接続パラメータ］

［接続パラメータ］ "conn=eth:〈IP アドレス〉[:ポート番号]"
ポート番号のデフォルトは 8502 です。
（ポート番号は省略可）
［タイムアウト時間］ 送受信時のタイムアウト時間(msec)を指定します。
"Timeout[=時間]"。デフォルトは 500 です。
（タイムアウト時間は省略可）

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

caoCtrl=cao.AddController("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")
```

※ポート番号を指定したい場合は次のように記述します

```
caoCtrl=cao.AddController("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10:8503")
```

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＞

引数：〈モード〉 運転モード・停止モードの切り替え（整数）。
0：停止モード。
1：運転モード。

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("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")
caoCtrl.ChangeMode 1                    '運転モードへ
```

実装変数.ChangeModeAsync

機能

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。

書式

実装変数.ChangeModeAsync ＜モード＞

引数：〈モード〉 運転モード・停止モードの切り替え（整数）。
0：停止モード。
1：運転モード。

説明

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。

用例

```
Dim caoCtrl as Object
Dim vntResult as Variant

caoCtrl=cao.AddController("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")

caoCtrl.ChangeModeAsync 1   '運転モードへ

'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（運転モード、停止モード、リモートキャプチャモード）を取得します。

書式

実装変数.ReadMode

戻り値：現在の動作モードが格納されます。取得に失敗した場合は-1 が格納されます。(Variant 型)
0：停止モード。
1：運転モード。
2：リモートキャプチャモード。

説明

現在の動作モード（運転モード、停止モード、リモートキャプチャモード）を取得します。

用例

```
Dim caoCtrl as Object
Dim vntResult as Variant

caoCtrl=cao.AddController("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")
vntResult = caoCtrl.ReadMode
```

実装変数.Reset

機能

コントローラをリセットします。

書式

実装変数.Reset

説明

コントローラをリセットします。

用例

```
Dim caoCtrl as Object

caoCtrl=cao.AddController("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")
caoCtrl.Reset
```

実装変数.Restart

機能

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

実装変数.Restart

説明

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("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")
caoCtrl.Restart
```

実装変数.RestartAsync

機能

非同期で、スタートユニットの次のユニットにジャンプします。
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。

書式

実装変数.RestartAsync

説明

非同期で、スタートユニットの次のユニットにジャンプします。
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("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")
caoCtrl.Restart

caoCtrl.RestartAsync

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

実装変数.Trigger

機能

トリガを発行します。

書式

実装変数.Trigger ＜トリガ番号＞

引数：〈トリガ番号〉発行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。

説明

トリガを発行します。

用例

```
Dim caoCtrl as Object

caoCtrl=cao.AddController("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")
caoCtrl.Trigger 2
```

実装変数.EnableTrigger

機能

トリガ入力の許可／禁止を設定します。

書式

実装変数.EnableTrigger ＜許可モード＞

引数：〈許可モード〉トリガの許可／不許可を指定します。（整数）
0：トリガ禁止。
1：トリガ許可。

説明

トリガの許可／禁止を設定します。

用例

```
Dim caoCtrl as Object

caoCtrl=cao.AddController("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")
caoCtrl.EnableTrigger 1
```

実装変数.ReadTriggerEnable

機能

現在のトリガ許可／禁止を取得します。

書式

実装変数.ReadTriggerEnable

戻り値：トリガの許可、禁止状態が格納されます。取得に失敗した場合は-1が格納されます。(Variant 型)
0：トリガ禁止状態。
1：トリガ許可状態。

説明

トリガの許可、禁止状態を取得します。

用例

```
Dim caoCtrl as Object
Dim vntResult as Variant

caoCtrl=cao.AddController("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")
vntResult = caoCtrl.ReadTriggerEnbale
```

実装変数.WriteVariable

機能

指定したスカラ型変数(グローバル変数、ローカル変数)へ値を書き込みます。

書式

実装変数.WriteVariable ＜変数名＞
, ＜値＞
, ＜同期モード＞

引数：〈変数名〉スカラ型の変数名を半角文字で指定します。(文字列)
〈値〉 変数に書き込む値を指定します。(Double 型)
〈同期モード〉 フロート同期して反映するか指定します。(整数)
0:フローと同期せず即時反映する。(MW コマンド)
1:フローの終了ユニットで反映する。(MS コマンド)

説明

指定したスカラ型変数へ値を書き込みます。
(この変数名は XG シリーズで設定した変数名です。)

用例

```
Dim caoCtrl as Object

caoCtrl=cao.AddController("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")
caoCtrl.WriteVariable "#MyVar", 2, 0
```

実装変数.ReadVariable

機能

指定されたスカラ型変数の値を取得します。

書式

実装変数.ReadVariable(＜変数名＞ **)**

引数：〈変数名〉スカラ型の変数名を半角文字で指定します。(文字列)
戻り値：変数値が格納されます。取得に失敗した場合は-1.0 が格納されます。
(Variant 型)

説明

指定されたスカラ型変数の値を取得します。
(この変数名は XG シリーズで設定した変数名です。)

用例

```
Dim caoCtrl as Object
Dim vntResult as Variant

caoCtrl=cao.AddController("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")
vntResult = caoCtrl.ReadVariable("#MyVar")
```

実装変数.ChangeInspectSetting

機能

指定された SD カードの検査設定№に設定を切り替えます。

書式

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

引数：〈SD カード番号〉 SD カード番号を指定します。（整数 1, 2）
〈検査設定№〉 検査設定№を指定します。（整数 0～999）

説明

指定された SD カードの検査設定№に設定を切り替えます。

用例

```
Dim caoCtrl as Object

caoCtrl=cao.AddController("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")
caoCtrl.ChangeInspectSetting 1, 2              'SD カード 1 の検査設定 No. 2 へ切り替え
```

実装変数.ChangeInspectSettingAsync

機能

非同期で指定された SD カードの検査設定No.に設定を切り替えます。
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.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, 2）
〈検査設定No.〉 検査設定No.を指定します。（整数 0～999）

説明

非同期で指定された SD カードの検査設定No.に設定を切り替えます。
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("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")
caoCtrl.ChangeInspectSettingAsync 1, 2                'SD カード 1 の検査設定 No. 2 へ切り替え

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

実装変数.ReadInspectSetting

機能

現在使用している検査設定No.を取得します。

書式

実装変数.ReadInspectSetting

引数　：なし
戻り値：設定値が配列型に格納されます。
　　要素番号 0：〈SD カード番号〉（整数 1, 2）
　　要素番号 1：〈検査設定No.〉（整数 0～999）

説明

現在使用している検査設定No.を取得します。

用例

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

caoCtrl=Cao.AddController("XG","CaoProv.KEYENCE.VWXG","","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.ClearError

機能

指定した種別のエラー状態とエラーコードをクリアします。

書式

実装変数.ClearError ＜エラー種別＞

引数：〈エラー種別〉エラー状態種別を指定します。（整数 0, 1）
0：%Error0、%Error0Code をクリアします。
1：%Error1、%Error1Code をクリアします。

説明

指定した種別のエラー状態とエラーコードをクリアします。

用例

```
Dim caoCtrl as Object

caoCtrl=cao.AddController("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")
caoCtrl.ClearError 0
```

実装変数.ExecuteCommand

機能

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関わらずコマンドの応答を取得します。

書式

実装変数.ExecuteCommand(〈無手順コマンド〉**)**

引数： 〈無手順コマンド〉 コマンドを文字列で指定します。
戻り値： コマンドの応答を文字列で返します。取得に失敗した場合は、文字列が格納されません。(Variant 型)

説明

サポートする無手順コマンドは、XG シリーズ用 V-Works for XG ActiveX コントロールのリファレンスマニュアルを参照して下さい。

用例

無手順コマンド R0：運転モード移行を実施する例を示します。

```
Dim caoCtrl as Object
Dim vntResult as Variant

caoCtrl=cao.AddController("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")
vntResult = caoCtrl.ExecuteCommand("R0")                    '運転モードへ移行
```

実装変数.ExecuteCommandAsync

機能

指定した無手順コマンドを非同期で実行します。
コマンド実行の成否に関わらず戻り値は、GetCommandResult コマンドで取得します。

書式

実装変数.ExecuteCommandAsync(〈無手順コマンド〉**)**

引数：　〈無手順コマンド〉　コマンドを文字列で指定します。
戻り値：　なし

説明

サポートする無手順コマンドは、XG シリーズ用 V-Works for XG ActiveX コントロールのリファレンスマニュアルを参照して下さい。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

無手順コマンド　R0：運転モード移行を実施する例を示します。

```
Dim caoCtrl as Object
Dim vntResult as Variant

caoCtrl=cao.AddController("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")
vntResult = caoCtrl.ExecuteCommandAsync("R0")                '運転モードへ移行

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

実装変数.TriggerAndGetResult

機能

指定したトリガを発行し、画像処理結果を取得します。

書式

実装変数.TriggerAndGetResult(＜トリガ番号＞ **)**

引数：<トリガ番号>発行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：結果出力ユニットで設定した出力データを格納します。取得に失敗した場合は-1.0 が格納されます。(Variant 型)

説明

指定したトリガを発行し、XG シリーズ側の出力ユニットで設定した処理結果を取得します。

用例

```
Dim caoCtrl as Object
Dim vntResult as Variant

caoCtrl=cao.AddController("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")
vntResult = caoCtrl.TriggerAndGetResult( -1 )
```

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（Variant 型）

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(0x80103000)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("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")

caoCtrl.Execute "ReStartAsync"
vntResult = caoCtrl.GetCommandResult
```

実装変数.ReceivePacket

機能

パケットの受信を行います。

書式

実装変数.ReceivePacket

引数：なし

戻り値：受信パケット

説明

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。

用例

```
Dim caoCtrl as Object
Dim strRet as String

caoCtrl=cao.AddController("XG","CaoProv.KEYENCE.VWXG", "", "conn=eth:192.168.0.10")

caoCtrl.Trigger 1
strRet = caoCtrl.RecievePacket
```

6. エラーコード

XGプロバイダの独自のエラーコードは、XGの戻り値をもとに下記のように生成されます。

0x80100000 + 戻り値

各コマンドにおけるエラーコードは、キーエンス社のActiveXコントロールリファレンスマニュアルの取説を参照してください。

例：ChangeModeを実行したとき.
0x801003EA： 引数の値が範囲外です.

また、独自エラーコードとして以下のエラーコードが定義されています。

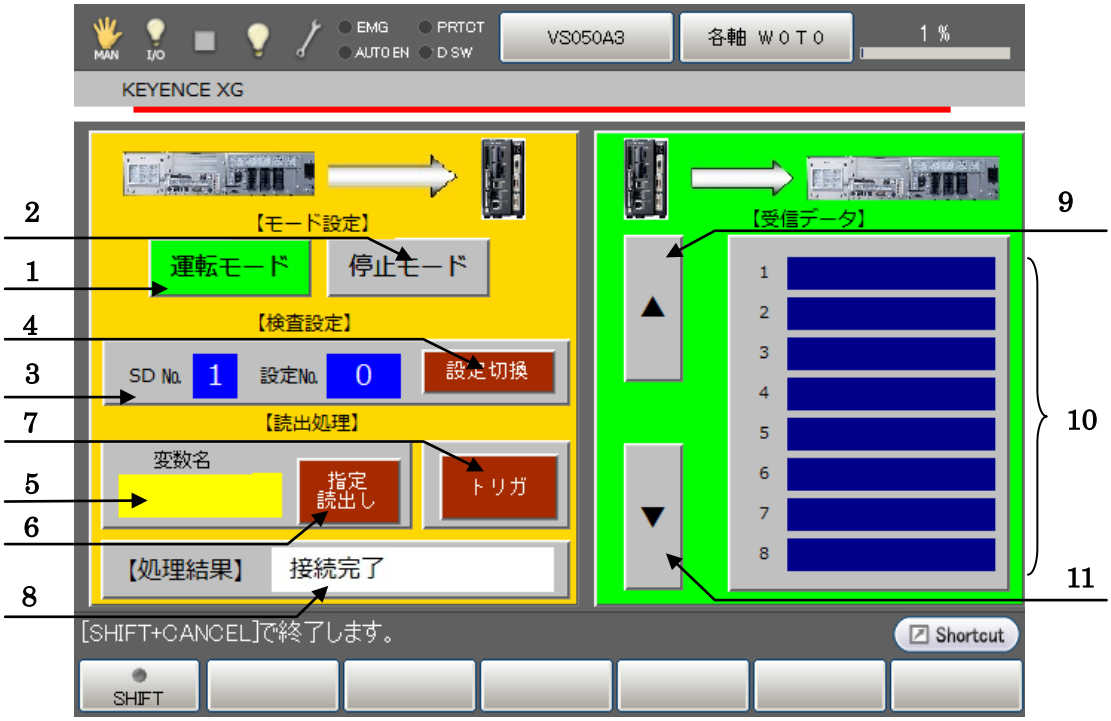
エラー名	エラー番号	説明
E_BASED_VWXG	0x80100000	VWXGシリーズ固有エラー
E_ENABLED_CANCEL	0x80101000	キャンセル処理に失敗しました。
E_EXECUTING	0x80102000	コマンド実行中に別コマンドを実行しました。
E_GET_COMMAND_RESULT	0x80103000	同期コマンド実行後にGetCommandResultコマンドを実行しました。

共通エラーコードについてはUSER MANUALSのプロバイダガイドのエラーコードについてを参照してください。

7. 操作盤画面

本プロバイダには下記の操作盤画面を準備しています。この操作盤はプロバイダを使用したもので、機器接続後の動作確認等に使用することができます。操作盤のアプリケーション例の参考にして下さい。操作盤を表示するとXGへ接続（プロバイダの実装）をしますので予め通信設定を行って下さい。操作盤を閉じると切断（プロバイダの解放）されます。

【メイン面】



説明　各ボタンの動作内容について説明します。

1. 「運転モード」へ切り替えます。(ChangeMode)
2. 「停止モード」へ切り替えます。(ChangeMode)
3. 検査設定No.を設定します。SD カード番号：1～2　設定番号：0～999
4. 3. で設定した設定番号に切り替えます。(ChangeInspectSetting)
5. 読み出しを行う変数名を設定します。(この変数名は XG シリーズで設定した変数名です。)
6. 5. で設定した変数名の値を読み出します。受信したデータは 10. のデータ表示部に表示します。(ReadVariable)
7. 全トリガを実行します。(Trigger)
8. 処理結果を表示します。
9. 受信データの表示ページを Up します。
10. 受信データを表示します。
11. 受信データの表示ページを Down します。

注１：プロバイダの実装（初期化）が正常に行われた場合は、8. に"接続完了"と表示されます。

注２：PacScript プログラムにて XG プロバイダを使用している場合は、操作盤操作をしないで下さい。

8. サンプルプログラム

```
Sub Main

    On Error Goto ErrProc                       '異常処理ルーチン宣言

    Dim caoXG as Object                         'プロバイダ用変数宣言
    Dim vntResult as Variant                    '文字列変数宣言
    Dim pTargetPos as Position                  'P 変数型変数宣言

    takearm keep = 0

    pTargetPos = P11


    caoXG = cao.AddController("XG", "CaoProv.KEYENCE.VWXG", "","Conn=eth:192.168.0.10")
                                                'プロバイダの実装
    caoXG.ChangeInspectSetting 1, 2             'SD1 の設定 2 に切換
    caoXG.Trigger -1                            'トリガ
    delay 200
    vntResult = caoXG.ReadVariable("#XPos")     'データ受信
    letx pTargetPos = posx(P11) + val(vntResult)  '受信データの X 成分を位置データへ展開

    approach p, pTargetPos, @p 20, s = 100      '補正後の位置へ
    move l, @e pTargetPos, s = 10
    call Hand.Close
    depart l, @p 50, s = 100

EndProc:                                        '正常終了ルーチン
    「必要な終了処理を記述」
    exit sub

ErrProc:                                        '異常終了ルーチン
    「必要な異常処理を記述」
    Goto EndProc

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　画像処理システム　ＸＧシリーズ

バージョン	対応RC8	改訂内容
Ver.1.0.0	Ver.1.1.2	初版
Ver.1.0.1	Ver.1.3.6～	コマンド追加"TriggerAndGetResult"
Ver.1.0.2	Ver.1.12.*	コマンド追加" ReadInspectSetting "
Ver.1.0.3	Ver.1.13.0	非同期処理のコマンドの追加 GetCommandResultコマンドの追加 RecievePacketコマンドの追加 TriggerAndGetResultコマンド修正
Ver.1.0.4	Ver.1.13.0	サンプルプログラムの修正
Ver.1.0.5	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