

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

メーカー

シャープマニファクチャリングシステム株式会社

製品／シリーズ

画像センサカメラ

MODEL:IV シリーズ



Vision

はじめに

本書は、「シャープマニファクチャリングシステム株式会社製・画像センサカメラ・IV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IV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、「シャープマニファクチャリングシステム株式会社製・画像センサカメラ・IV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のご使用された場合は、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

シャープマニファクチャリングシステム株式会社製 IV シリーズ
対象となるIVシリーズは、IV-S150シリーズ / IV-S200シリーズ / IV-C250X
IV-S300シリーズ / IV-S400シリーズです。
※全ての機種での動作確認は行っていません

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。本製品を使用する場合は、弊社ホームページの本製品ダウンロードサイトに記載の「ソフトウェア使用許諾契約」を必ずご参照ください。

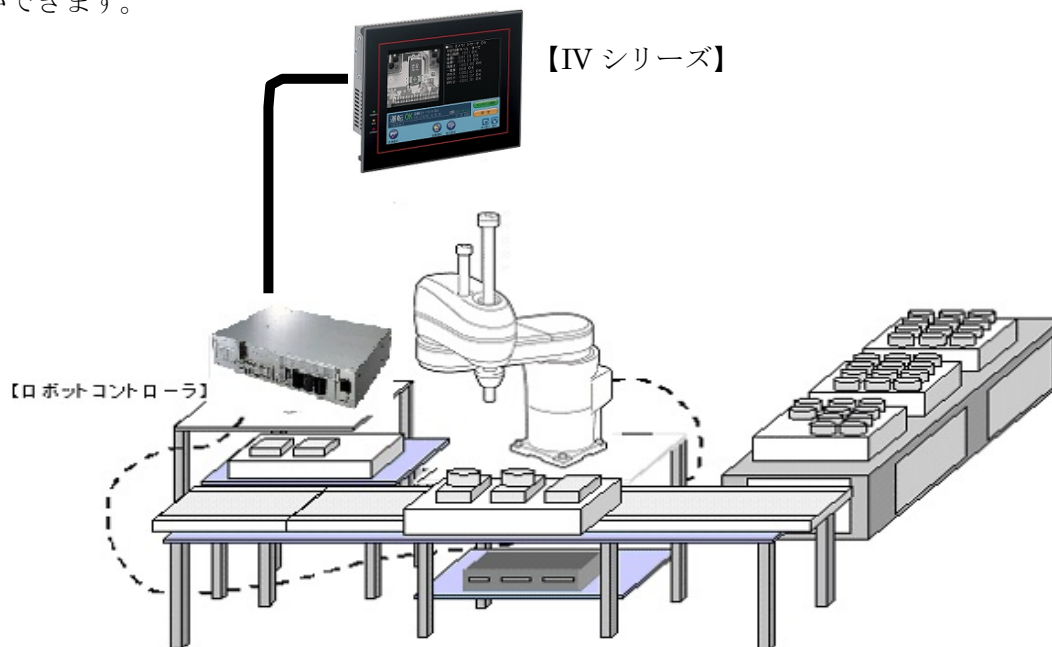
目次

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
2. 接続方法	6
3. ロボットコントローラと使用機器の通信設定	8
4. プロバイダ実行手順	13
5. コマンドの説明	14
6. エラーコード	74
7. 操作盤画面	75
8. サンプルプログラム	76

1. 本製品（プロバイダ）の概要

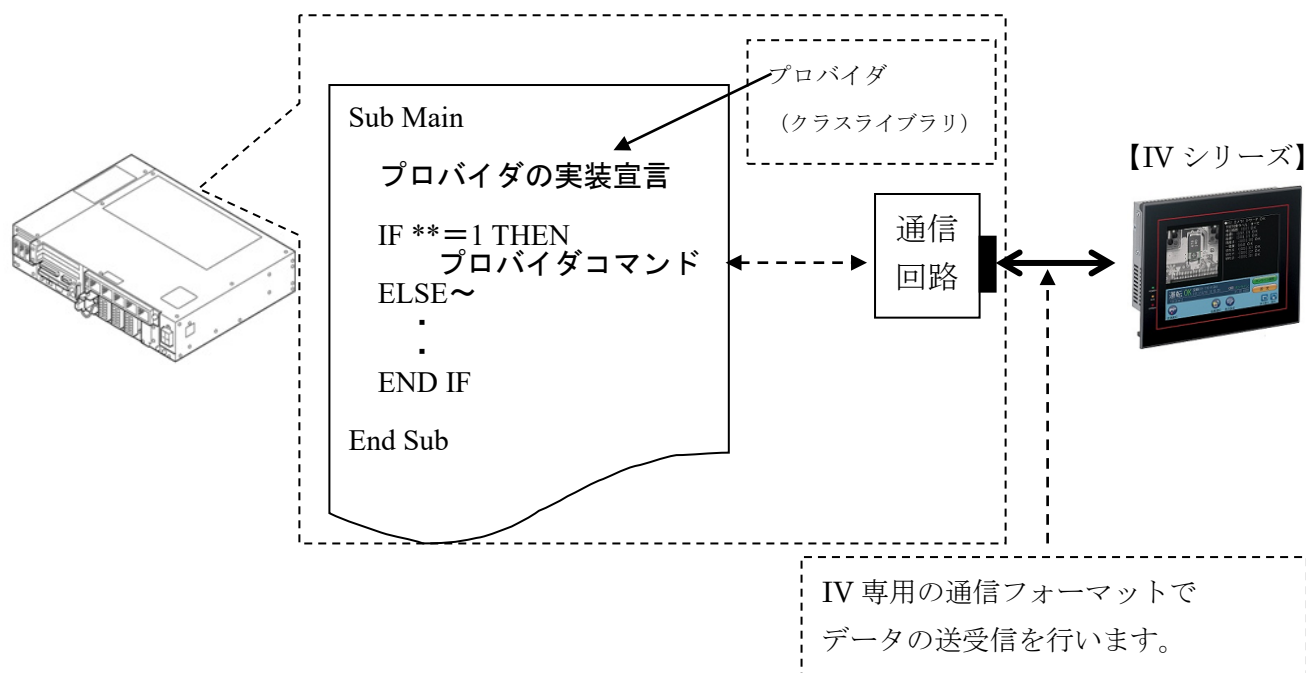
1.1 プロバイダの対象機器

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IVシリーズに接続した時にのみ使用することができます。



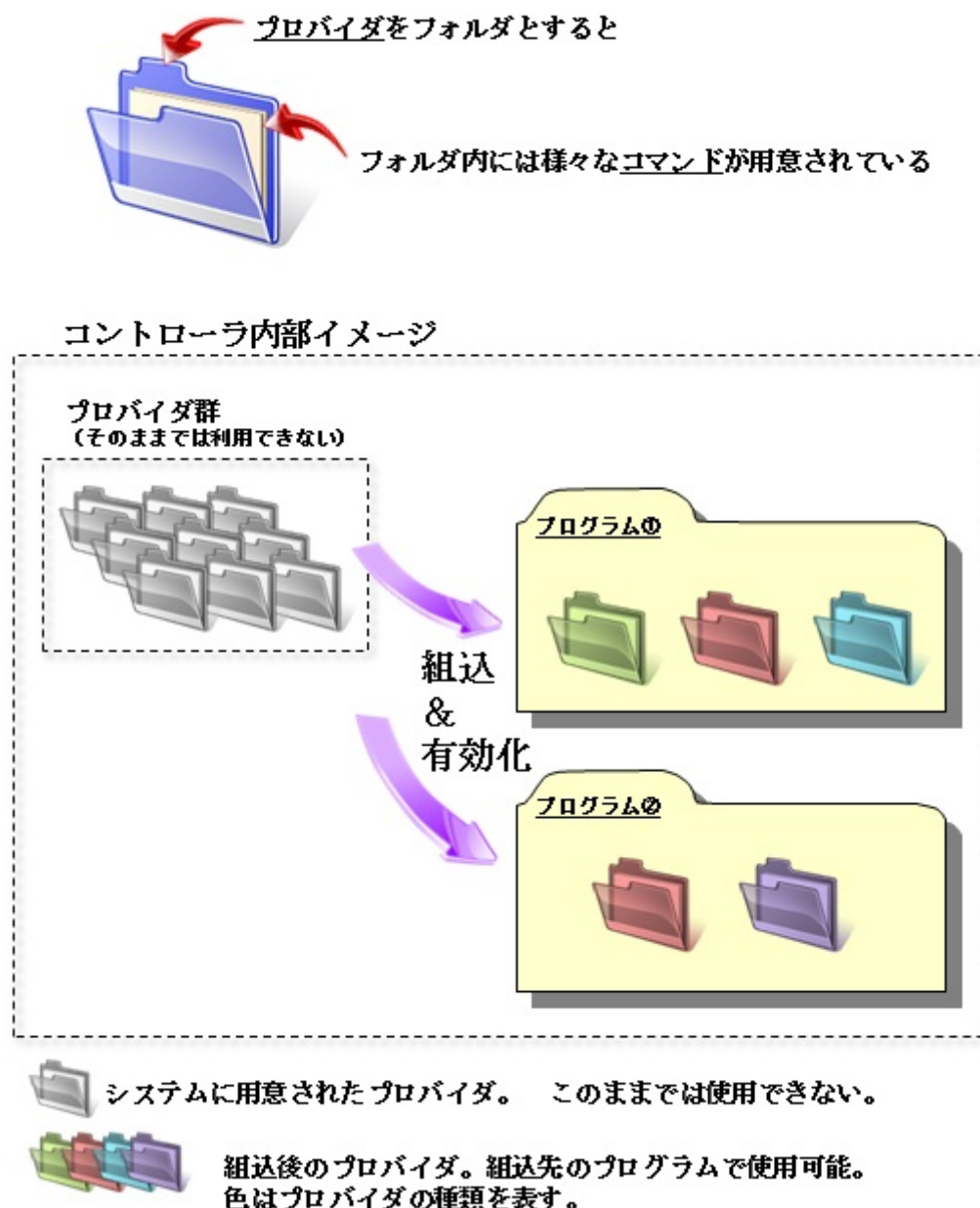
1.2 プロバイダの特長


IVシリーズにアクセスするために必要なIVのネイティブコマンドを、ロボットプログラムで利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、IV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ことが可能です。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 RS-232C 接続例

IVシリーズとロボットコントローラの通信のために、通信ケーブルで接続する必要があります。
ロボットコントローラとRS-232C接続する際には、下結線図を参考にRS232Cクロスケーブルで接続して下さい。

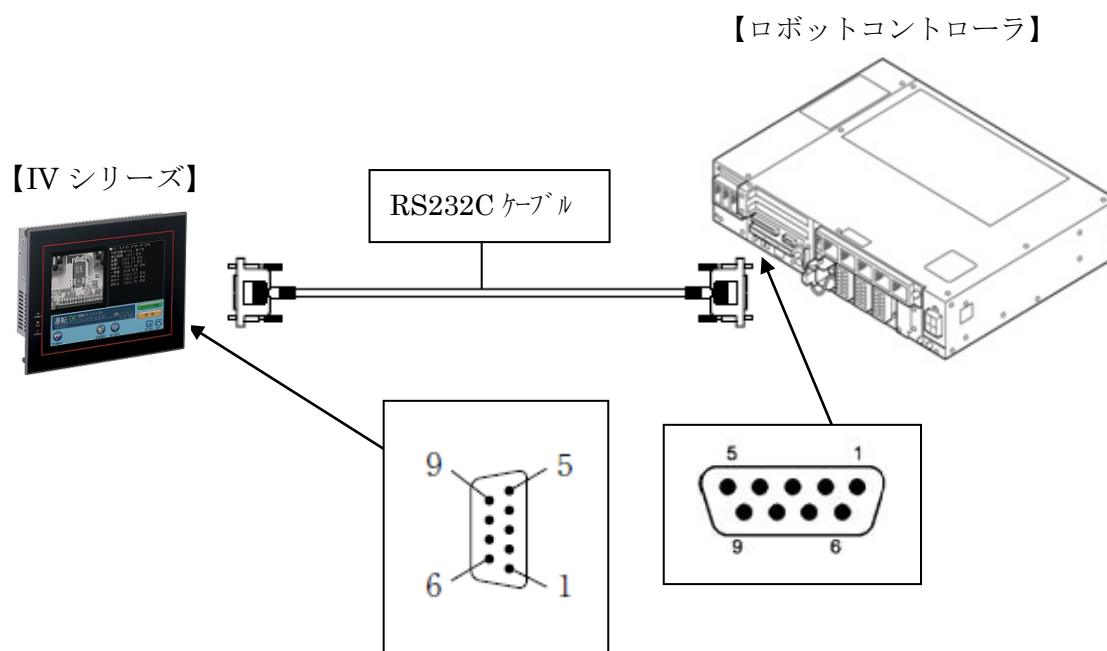


図 2-1 RS232C 結線図

IV 側 RS232C コネクタ (D-Sub9 ピンメス)

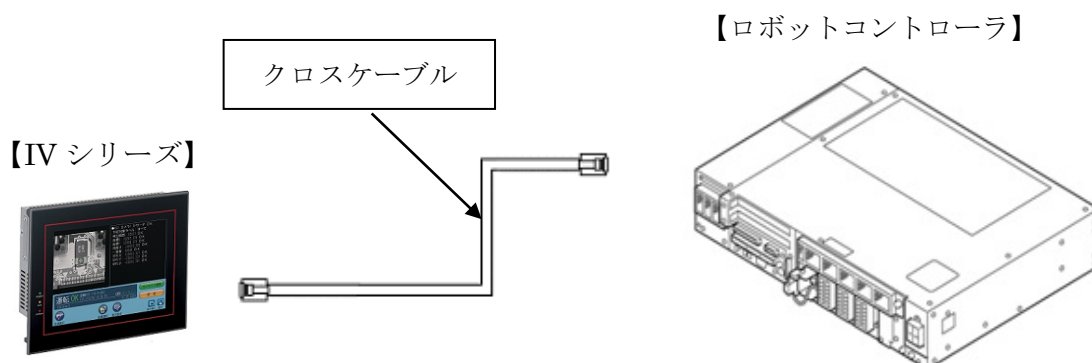
ピン番号	信号名
コネクタケース	FG
2	RD
3	SD
5	SG

RC8 側 RS232C コネクタ (D-Sub9 ピンオス)

ピン番号	信号名
コネクタケース	FG
2	RXD
3	TXD
5	SG

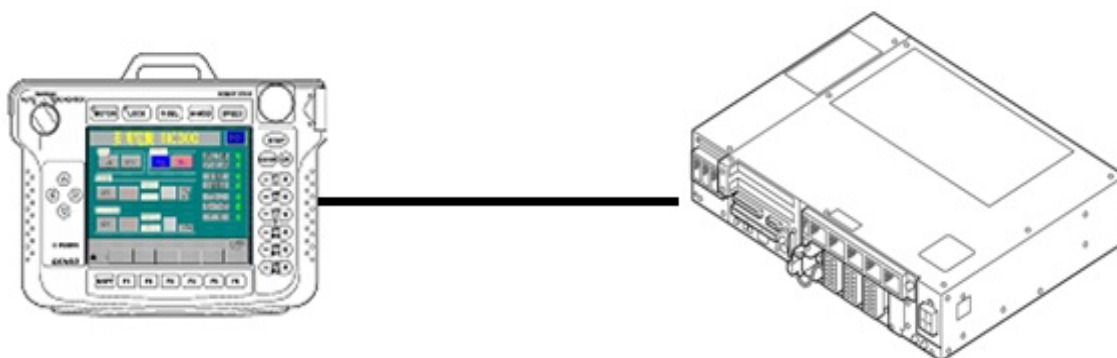
2.2 Ethernet (TCP/IP) 接続例

ロボットコントローラとEthernet接続するには、クロス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。



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 RS-232C による通信

3.1.1 ロボットコントローラの RS-232 通信設定

RS-232Cシリアルインターフェイスの各ポートの通信速度などを設定します。

〔F6設定メニュー〕〔F5通信と起動権〕〔F3データ通信〕で〔デバイス〕から対象のRS-232Cを選択し、〔F6編集〕を押すと、〔データ通信編集〕ウィンドウが表示されます。目的の設定項目を選択し、設定値を変更してください。又、デリミタはCRを設定して下さい。

標準装備のRS232Cポートは、回線番号1（COM2）です。

MAN I/O EMG PRTGT AUTO EN D SW

VS050A3 A

各軸 W O T O

1 %

データ通信編集[シリアル 回線番号1]

ボーレート	9600	19200	38400	57600	115200
パリティ	偶数	なし	奇数	0固定	1固定
データ長	4	5	6	7	8
ストップビット	1bit		1.5bit		2bit
フロー制御	なし		Xon/Xoff		ハードウェア制御
タイムアウト	-1 msec -1は無限待ちします。				
データタイプ	テキスト		バイナリ		
デリミタ	CR		CR+LF		LF
ヘッダ	なし		ENC		

Cancel OK

Shortcut

SHIFT

3.1.2 IVシリーズの RS-232C 通信設定

RS-232Cシリアルインターフェイスの各ポートの通信速度などを設定します。
IVの[システム] → [通信]を開き、[シリアル]を選択すると、設定画面が表示されます。
目的の設定項目を選択し、ロボットコントローラと同じ設定になるように設定値を変更して下さい。
又、通信種別は「RS-232C」、通信モードは「汎用」、を設定して下さい。

【IV の画面】

シリアル	通信種別	RS232C	方式	2線式
イーサネット	通信モード	汎用		
外部端子	ボーレート	115200bps		
PLCリンク	データ長	7ビット		
	パリティ	偶数		
	ストップビット	2ビット	自局番	000

設定 品種003 2013/03/25 13:15:26 USB オフライン
メイン/システム/通信

メイン 戻る

3.2 Ethernet (TCP/IP) による通信

3.2.1 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。

〔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IV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。

通信設定

デバイス: イーサネット(192.168.0.1)
読込/書込可

プロパティ	値
通信権	読込/書込可
DHCP	無効
IPアドレス	192.168.0.1
サブネットマスク	255.255.255.0
ゲートウェイ	0.0.0.0
MACアドレス	B4-B5-2F-B9-1D-18

WINCAPSとロボットコントローラの間で通信を行うための設定です。

Cancel OK

Shortcut

SHIFT 編集

3.2.2 IV の Ethernet (TCP/IP) 通信設定

IVのIPアドレスを設定します。

[システム] → [通信] を開き、[イーサネット]を選択すると、設定画面が表示されます。

ロボットコントローラとIV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、通信モードは「汎用」、を設定して下さい。

【IV の画面】

シリアル

イーサネット

外部端子

PLCリンク

アドレス設定

IPアドレス 192 168 000 002

サブネットマスク 255 255 255 000

デフォルトゲートウェイ 000 000 000 000

局番

自局番 000

通信モード

モード 汎用

ポート番号

コマンド 02001

データコレクター 02002

設定 品種001 2013/03/11 09:35:51 USB オフライン

メイン/システム/通信

メイン 戻る

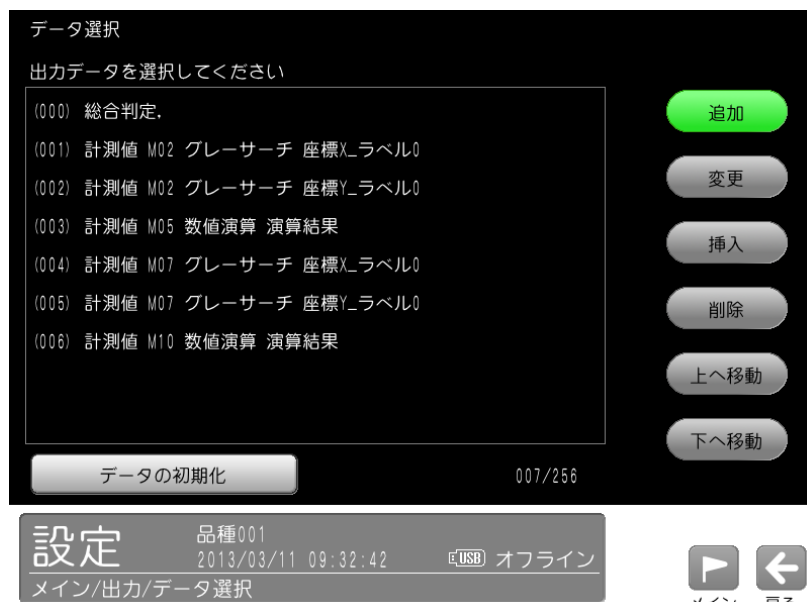
3.3 IV の出力データ設定

〔処理フロー〕の中の〔出力〕を開き、〔数値データ〕を選択すると、設定画面が表示されます。

〔出力データ〕の〔データ選択〕を選択すると、出力データ選択画面が表示されます。

RC8 コントローラに出力したいデータを選択します。

【IV の画面】



データ選択

出力データを選択してください

(000) 総合判定,

(001) 計測値 M02 グレーサーチ 座標X_ラベル0

(002) 計測値 M02 グレーサーチ 座標Y_ラベル0

(003) 計測値 M05 数値演算 演算結果

(004) 計測値 M07 グレーサーチ 座標X_ラベル0

(005) 計測値 M07 グレーサーチ 座標Y_ラベル0

(006) 計測値 M10 数値演算 演算結果

追加

変更

挿入

削除

上へ移動

下へ移動

データの初期化

007/256

設定 品種001 2013/03/11 09:32:42 E[USB] オフライン

メイン/出力/データ選択

メイン 戻る

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 strResult as String ③	‘結果取得用変数宣言

caoCtrl = cao.AddController(“TV_S150X”, “caoProv.SHARP.IV”, “”, “conn=eth:192.168.0.2:2001”) 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、IV コマンド、独自拡張コマンドに分類されます。尚、IV コマンドの詳細動作についてはシャープマニファクチャリングシステム社、各 IV のユーザーズマニュアルを参照してください。

表 5-2 コマンド一覧

コマンド	IV コマンド	機能	対応機種					
			IV-S150 シリーズ		IV-S200 シリーズ IV-C250X		IV-S300 シリーズ IV-S400 シリーズ	
							com	eth
接続コマンド								
cao.AddController	—	プロバイダを変数に実装して、接続処理を行います	○	○	○	○	○	○
IV コマンド								
Trigger And wait	T00	トリガを入力し結果を取得します	○	○	○	○	○	○
Trigger	T01	トリガを入力します	○	○	○	○	○	○
GetData	T01	画像処理の結果を取得します	○	○	○	○	○	○
RobotCalibration	T10	ロボットのキャリブレーションを実行します					○	○
SerialEnable	A00,A01	シリアル通信の許可/禁止を設定します			○			
SAlignmentTrigger	A00	各軸の現在値を設定します					○	○
SAlignmentCalibration	A01	キャリブレーション実行時の各種設定を行います					○	○
RemoteEnable	A10,A11	リモート設定キー入力の許可/禁止を設定します			○			
ViewLockEnable	A20,A21	運転画面ロックの許可/禁止を設定します			○			
GetKind	C00	品種番号を読み出します	○	○	○	○	○	○
PutKind	C01	品種番号を書き込みます	○	○	○	○	○	○
GetModule	C10	モジュール番号を読み出します			○	○		
PutModule	C11	モジュール番号を書き込みます			○	○		
GetViewMode	C20	画像更新モードを読み出します	○	○	○		○	○
PutViewMode	C21	画像更新モードを書き込みます	○	○	○		○	○
GetDispMode	C30	表示画像モードを読み出します	○	○	○		○	○
PutDispMode	C31	表示画像モードを書き込みます	○	○	○		○	○
GetMeasureData	C40	手動計測座標を読み出します			○			

ClearFigures	C40	計測回数をリセットします (全トリガ分)					○	○
PutMeasureData	C41	手動計測座標を書き込みます			○			
PutPassword	C60	運転画面ロックパスワードを書き込みます			○			
GetVal	C80	変数値を読み出します			○	○	○	○
PutVal	C81	変数値を書き込みます			○	○	○	○
RegStdImage	R00	IV-S2*0X の場合: 基準画像を上書き登録します IV-S3*0X の場合: 最後に取り込まれたカメラ画像を基準画像として保存 (不揮発メモリへ) します			○		○	○
GetShutterSp	R10	シャッター速度を読み出します			○			
PutShutterSp	R11	シャッター速度を書き込みます			○			
GetThreshold	R30	しきい値設定を読み出します			○			
PutThreshold	R31	しきい値設定を書き込みます			○			
GetGain	R40	ゲイン・オフセット設定を読み出します			○			
PutGain	R41	ゲイン・オフセット設定を書き込みます			○			
GetDate	R50	日時設定を読み出します	○	○	○		○	○
PutDate	R51	日時設定を書き込みます	○	○	○		○	○
GetRegData	R80	コードリーダー登録データを読み出します					○	○
PutRegData	R81	コードリーダー登録データを書込みます					○	○
PutDateString	R89	設定文字列を書き込みます (日付ブロッカー括 8 個, カメラ指定あり)					○	○
GetModuleString	R90	設定文字列を読み出します (モジュール指定)					○	○
GetBlockString	R92	設定文字列を読み出します (ブロック指定)					○	○
PutBlockString	R93,R94	設定文字列を書き込みます (ブロック指定)					○	○
Put20BlockString	R96	設定文字列を書き込みます (可変ブロッカー括 20 個, 末尾の空白除去あり)					○	○
Put10BlockString	R98	設定文字列を書き込みます (可変ブロッカー括 10 個, カメラ指定あり, 末尾の空白除去あり)					○	○
SnapShot	I01	スナップショット画像を外部メモリに保存します	○	○	○	○	○	○
ImageClear	I20	画像メモリを消去します			○	○		
GetKindState	P10	品種設定番号情報を読み出します				○		
GetStdImageState	P20	基準画像番号情報を読み出します				○		

GetVersion	D00	バージョン情報を読み出します			○	○		
AllReset	D10	システム設定・品種設定を初期化します			○	○		
SettingSave	D11	システム設定・品種設定を保存します	○	○	○	○	○	○
Reset	D12	本機をリセットします			○			
GetBrightness	D20	平均濃度を読み出します	○	○	○		○	○
GetParallel	D21	パラレルの入出力情報を読み出します	○	○	○		○	○
SelfCheck	D40	自己診断を行います					○	○
独自拡張コマンド								
Raw	—	生データ送受信	○	○	○	○		
ChangeTimeout	—	通信タイムアウト時間変更						

cao.AddController

機能

プロバイダを変数に実装して、IV 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. SHARP. IV”

<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], [タイムアウト時間], [機種タイプ],
[局番], [チェックサム]

[接続パラメータ] イーサネット又は、RS232C 接続パラメータを設定

・イーサネット接続の場合

“conn=eth:<<IP アドレス>>:<<ポート番号>>”

<< IP アドレス>> IV の IP アドレス

<<ポート番号>> IV のポート番号（省略可）

デフォルト：2001

・RS232C 接続の場合

“conn=com:<<com ポート番号>>:<<通信速度>>:<<パリティ>>:<<データビット>>:<<ストップビット>>”

<<com ポート番号>> RS232C のポート番号を指定

‘1’ -COM1, ‘2’ -COM2

<<通信速度>>RS232C の通信速度を指定（省略可）

2400, 4800, 9600, 19200, 38400,

57600, 115200

デフォルト：115200

<<パリティ>>RS232C の通信速度を指定（省略可）

‘N’ -NONE, ‘E’ -EVEN, ‘O’ -ODD

デフォルト：‘E’ -EVEN

<<データビット数>>RS232C のデータビット数を指定

（省略可）

‘7’ -7bit, ‘8’ -8bit

デフォルト：‘7’ -7bit

<<ストップビット数>>RS232C のストップビット数を指定

（省略可）

‘1’ -1bit, ‘2’ -2bit

デフォルト：‘2’ -2bit

[タイムアウト時間] 送受信時間タイムアウト時間(msec)を指定します。

“Timeout=<<タイムアウト時間>>”（省略可）

デフォルト：500

- [機種タイプ] 機種タイプを指定します。
 "Type=<<機種タイプ>>" (省略可)
 0 : IV-S150X/M (デフォルト)
 1 : IV-S200X, IV-S210X, IV-C250X
 2 : IV-S300X, IV-S310X
- [局番] 通信時の局番を指定します。(0～255)
 "AreaNo=<<局番>>" (省略可)
 デフォルト : 0
- [チェックサム] 通信時のチェックサムの有無を指定します。
 "Chksum=<<チェックサム>>" (省略可)
 0:無し (デフォルト)
 1:有り

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。)

有効と同時に接続を行います。接続の詳細は<オプション>で指定します。

用例

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")
```

※機種タイプなどを指定したい場合は次のように記述します

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", _  
    "Conn=eth:192.168.0.2:2001, Type=0, AreaNo=0, Chksum=0")
```

※RS232C 通信の場合は次のように記述します

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", _  
    "Conn=com:2, Type=0, AreaNo=0, Chksum=0")
```

※RS232C 通信で、通信パラメータがデフォルト値以外の場合は、次のように記述します

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", _  
    "Conn=com:2:38400:N:8:2, Type=0, AreaNo=0, Chksum=0")
```

実装変数.TriggerAndWait

機能 トリガを入力した後に、出力データを取得します。

書式 **実装変数.TriggerAndWait**(<トリガ番号>)

引数 : <トリガ番号> トリガ番号を指定します。(0～1)

戻り値 : IV から出力された、出力データ。

説明 トリガを入力し、出力データで設定した数値データを受信します。
出力データを設定していない場合は、タイムアウト時間まで待機します。
受信データが複数個有る場合は、配列として格納されます。

用例

```
Dim caoCtrl as Object  
Dim Res As Variant
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
Res = caoCtrl.TriggerAndWait(0)
```

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> トリガ番号を指定します。(0～1)

戻り値：なし。

説明 トリガを入力します。

用例

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
caoCtrl.Trigger 0
```

実装変数.GetData

機能 出力データを取得します。

書式 **実装変数.GetData(<トリガ番号>)**

引数 : <トリガ番号> トリガ番号を指定します。(0～1)

戻り値 : IV から出力された、出力データ。

説明 出力データで設定した数値データを受信します。
出力データを設定していない場合は、タイムアウト時間まで待機します。
受信データが複数個有る場合は、配列として格納されます。

用例

```
Dim caoCtrl as Object  
Dim Res As Variant
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
caoCtrl.Trigger 0  
Res = caoCtrl.GetData(0)
```

実装変数.RobotCalibration

機能

ロボットのキャリブレーションを実行します。

書式

実装変数.RobotCalibration(<トリガ番号>, <X 座標>,
<Y 座標>, <R 座標>)

引数 : <トリガ番号>	トリガ番号を指定します。(0～1)
<X 座標>	現在の、ロボットの X 座標絶対座標 (mm)
<Y 座標>	現在の、ロボットの Y 座標絶対座標 (mm)
<R 座標>	現在の、ロボットの R 座標絶対座標 (deg)

戻り値 : <完了フラグ>	終了／継続の状態を返します。
	True: 終了
	False: 継続
<X 座標>	次の、ロボットの X 座標絶対座標 (mm)
<Y 座標>	次の、ロボットの Y 座標絶対座標 (mm)
<R 座標>	次の、ロボットの R 座標絶対座標 (deg)

説明

ロボットのキャリブレーションを実行します。
各軸の絶対座標は小数点第 4 位で四捨五入され、小数部は 3 桁になります。

用例

「[8.2 キャリブレーション](#)」のサンプルプログラム参照

実装変数.SerialEnable

機能 シリアル通信の許可／禁止を設定します。

書式 **実装変数.SerialEnable** <許可／禁止>

引数：<許可／禁止> 許可／禁止を指定します。
True: 許可
False: 禁止

戻り値：なし。

説明 シリアル通信の許可／禁止を設定します。

用例

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
caoCtrl.SerialEnable True
```

実装変数.SAlignmentTrigger

機能 各軸の現在値を設定します。

書式 **実装変数.SAlignmentTrigger** (<トリガ番号>, <X 座標>, <Y 座標>, < θ 座標>)

引数 : <トリガ番号>	トリガ番号を指定します。(0～1)
<X 座標>	X 軸現在値
<Y 座標>	Y (Y1) 軸現在値
< θ 座標>	θ (Y2) 軸現在値

戻り値 : IV から出力されたデータ

説明 各軸の現在値を設定します。

用例

```
Dim caoCtrl as Object  
Dim Res As Variant
```

```
caoCtrl = cao.AddController("IV_S300", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
Res = caoCtrl.SAlignmentTrigger(0, 134.58, 25.48, 105.6)
```


実装変数.SAlignmentCalibration

機能

キャリブレーション実行時の各種設定を行います。

書式

実装変数.SAlignmentCalibration (<トリガ番号>, <X 座標>, <Y 座標>, <θ 座標>, <開始フラグ>)

引数 : <トリガ番号>	トリガ番号を指定します。(0～1)
<X 座標>	X 軸現在値
<Y 座標>	Y (Y1) 軸現在値
<θ 座標>	θ (Y2) 軸現在値
<開始フラグ>	開始フラグ
	True: キャリブレーション開始時, またはキャリブレーション再スタート時
	False: 上記以外

戻り値 : <X 座標>	X 軸移動量
<Y 座標>	Y (Y1) 軸移動量
<θ 座標>	θ (Y2) 軸移動量
<終了フラグ>	終了フラグ
	True: キャリブレーション完了
	False: キャリブレーション未完了

説明

キャリブレーション実行時の各種設定を行います。

用例

```
Dim caoCtrl as Object
Dim Res As Variant
```

```
caoCtrl = cao.AddController("IV_S300", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")
Res = caoCtrl.SAlignmentCalibration(0, 134.58, 25.48, 105.6, True)
```

実装変数.RemoteEnable

機能 リモート設定キー入力の許可／禁止を設定します。

書式 **実装変数.RemoteEnable** <許可／禁止>

引数：<許可／禁止> 許可／禁止を指定します。
True: 許可
False: 禁止

戻り値：なし。

説明 リモート設定キー入力の許可／禁止を設定します。

用例

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
caoCtrl.RemoteEnable True
```

実装変数.ViewLockEnable

機能 運転画面ロックの許可／禁止を設定します。

書式 **実装変数.ViewLockEnable** 〈許可／禁止〉

引数：〈許可／禁止〉 許可／禁止を指定します。
True: 許可
False: 禁止

戻り値：なし。

説明 運転画面ロックの許可／禁止を設定します。

用例

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
caoCtrl.ViewLockEnable True
```

実装変数.GetKind

機能 品種番号を読み出します。

書式 実装変数.GetKind()

引数：なし

戻り値：品種番号。

説明 品種番号を読み出します。
トリガ番号 0 品種番号、トリガ番号 1 品種番号の順番に配列として戻り値が格納されます。

用例

```
Dim caoCtrl as Object  
Dim Number As Variant
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
Number = caoCtrl.GetKind()
```

実装変数.PutKind

機能 品種番号を書き込みます。

書式 実装変数.PutKind <品種番号>

引数：<品種番号> 品種番号を指定します。
IV-S150X/M:0～99 IV-S2*0X, IV-C250X:0～2047
IV-S3*0X:0～199

戻り値：なし。

説明 品種番号を書き込みます。
品種の内容によっては、品種切替え時に時間がかかる場合があります。
その場合は、”ChangeTimeout”コマンドでタイムアウト時間を変更して下さい。

用例

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
caoCtrl.PutKind 1
```

実装変数.GetModule

機能 モジュール番号を読み出します。

書式 実装変数.GetModule()

引数：なし

戻り値：モジュール番号。

説明 モジュール番号を読み出します。

用例

```
Dim caoCtrl as Object
Dim Number As Integer

caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")
Number = caoCtrl.GetModule()
```

実装変数.PutModule

機能 モジュール番号を書き込みます。

書式 **実装変数.PutModule** <モジュール番号>

引数：<モジュール番号> モジュール番号を指定します。(0～127)

戻り値： なし。

説明 モジュール番号を書き込みます。

用例

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
caoCtrl.PutModule 1
```

実装変数.GetViewMode

機能 画像更新モードを読み出します。

書式 実装変数.GetViewMode()

引数：なし

戻り値：描画更新モード。

説明 画像更新モードを読み出します。

IV-S150* IV-S3*0X	LV	動画
	SC	カメラ画像
	RC	処理画像
IV-S2*0X IV-C250X	LV	動画
	ST	静止画（トリガ毎）
	SN	静止画（NG 毎）
	SO	静止画（OK 毎）

用例

```
Dim caoCtrl as Object
Dim strViewMode As String
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")
strViewMode = caoCtrl.GetViewMode()
```


実装変数.PutViewMode

機能 画面更新モードを書き込みます。

書式 **実装変数.PutViewMode** <画面更新モード>

引数：<画面更新モード> 画面更新モードを指定します。

戻り値： なし。

説明 画面更新モード書き込みます。

IV-S150* IV-S3*0X	LV	動画
	SC	カメラ画像
	RC	処理画像
IV-S2*0X IV-C250X	LV	動画
	ST	静止画（トリガ毎）
	SN	静止画（NG 毎）
	SO	静止画（OK 毎）

用例

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")
caoCtrl.PutViewMode "LV"
```

実装変数.GetDispMode

機能 表示画像モードを読み出します。

書式 実装変数.GetDispMode()

引数：なし

戻り値：表示画像モード。

説明 表示画像モードを読み出します。

IV-S150*	C1	カメラ 1 表示
	C2	カメラ 2 表示
	DV	分割表示
IV-S2*0X IV-C250X	MI	モジュール指定カメラ
	MO	モジュール出力画像
	C1	カメラ 1 表示
	C2	カメラ 2 表示
	C3	カメラ 3 表示
	C4	カメラ 4 表示
	DV	分割表示
IV-S3*0X	C1	カメラ 1 表示
	C2	カメラ 2 表示
	C3	カメラ 3 表示
	C4	カメラ 4 表示
	DV	カメラ 1+2 表示
	DW	カメラ 3+4 表示
	DX	カメラ 1+2+3+4 表示

用例

```
Dim caoCtrl as Object
Dim strDispMode As String
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")
strDispMode = caoCtrl.GetDispMode()
```

実装変数.PutDispMode

機能 表示画像モードを書き込みます。

書式 **実装変数.PutDispMode** <表示モード>

引数：<表示画像モード> 表示画像モードを指定します。

戻り値： なし。

説明 表示画像モード書き込みます。

IV-S150*	C1	カメラ 1 表示
	C2	カメラ 2 表示
	DV	分割表示
IV-S2*0X IV-C250X	MI	モジュール指定カメラ
	MO	モジュール出力画像
	C1	カメラ 1 表示
	C2	カメラ 2 表示
	C3	カメラ 3 表示
	C4	カメラ 4 表示
	DV	分割表示
IV-S3*0X	C1	カメラ 1 表示
	C2	カメラ 2 表示
	C3	カメラ 3 表示
	C4	カメラ 4 表示
	DV	カメラ 1+2 表示
	DW	カメラ 3+4 表示
	DX	カメラ 1+2+3+4 表示

用例

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")
caoCtrl.PutDispMode "DV"
```

実装変数.GetMeasureData

機能 手動計測座標を読み出します。

書式 実装変数.GetMeasureData(<カメラ番号>)

引数：<カメラ番号> カメラ番号を指定します。(1～4)

戻り値：座標値。

説明 手動計測座標を読み出します。
座標値は第 1 点の X 座標、1 点の Y 座標、2 点の X 座標、2 点の Y 座標の順番に配列として格納されます。

用例

```
Dim caoCtrl as Object  
Dim Res As Variant
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
Res = caoCtrl.GetMeasureData(1)
```

実装変数.ClearFigures

機能 計測回数を全トリガ分リセットします。

書式 実装変数.ClearFigures()

引数：なし。

戻り値：なし。

説明 計測回数を全トリガ分リセットします。

実装変数.PutMeasureData

機能 手動計測座標を書き込みます。

書式 **実装変数.PutMeasureData** <カメラ番号>, <第 1 点の X 座標>, <第 1 点の Y 座標>, <第 2 点の X 座標>, <第 1 点の Y 座標>

引数 : <カメラ番号>	カメラ番号を指定します。(1～4)
<第 1 点の X 座標>	第 1 点の X 座標を指定します。(0～511)
<第 1 点の Y 座標>	第 1 点の Y 座標を指定します。(0～479)
<第 2 点の X 座標>	第 2 点の X 座標を指定します。(0～511)
<第 2 点の Y 座標>	第 2 点の Y 座標を指定します。(0～479)

戻り値 : なし。

説明 手動計測座標を書き込みます。

用例

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")
caoCtrl.PutMeasureData 1,100,100,200,200
```

実装変数.PutPassword

機能 運転画面ロックパスワードを書き込みます。

書式 **実装変数.PutPassword** <パスワード>

引数 : <パスワード> パスワード (0～9999)

戻り値 : なし。

説明 運転画面ロックパスワードを書き込みます。

用例

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
caoCtrl.PutPassword 1234
```

実装変数.GetVal

機能 変数値を読み出します。

書式 **実装変数.Getval**(<変数種類>, <変数番号>)

引数 : <変数種類>

IV-S2*0X の場合
変数種類を設定

0	システム変数
1	モジュール変数 (トリガ 1)
2	モジュール変数 (トリガ 2)

IV-S3*0X の場合
トリガ番号を設定 (0~1)

<変数番号> 変数番号を設定 (0~31)

戻り値 : 変数値。

説明 変数値を読み出します。
変数値の有効桁数は、画像センサ本体の設定による指定となります。

用例

```
Dim caoCtrl as Object
DIM fGetVal As Single

caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")
fGetVal = caoCtrl.GetVal(0,1)
```


実装変数.PutVal

機能 変数値を書き込みます。

書式 **実装変数.Putval** <変数種類>, <変数番号>, <変数値>

引数 : <変数種類>

IV-S2*0X の場合
変数種類を設定

0	システム変数
1	モジュール変数 (トリガ 1)
2	モジュール変数 (トリガ 2)

IV-S3*0X の場合
トリガ番号を設定 (0～1)

<変数番号>

変数番号を設定 (0～31)

<変数値>

変数値を設定

戻り値 : なし。

説明 変数値を書き込みます。

変数値の有効桁数は、画像センサ本体の設定による指定となります。

用例

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")
caoCtrl.PutVal 0,1,1
```

実装変数.RegStdImage

機能

IV-S2*0X の場合：基準画像を上書き登録します。

IV-S3*0X の場合：最後に取り込まれたカメラ画像を基準画像として保存（不揮発メモリーへ）します。

書式

実装変数.RegStdImage <カメラ番号>,<基準画像番号>

引数：<カメラ番号>

IV-S2*0X の場合：カメラ番号を設定（1～4）

IV-S3*0X の場合：トリガ番号を設定（0～1）

<基準画像番号>

IV-S2*0X の場合：基準画像番号を設定（0～8191）

IV-S3*0X の場合：カメラ組み合わせ番号

カメラ組み合わせ	カメラ組み合わせ No
なし	0
カメラ 1	1
カメラ 2	2
カメラ 3	4
カメラ 4	8
カメラ 1+2	3
カメラ 1+3	5
カメラ 1+4	9
カメラ 2+3	6
カメラ 2+4	10
カメラ 3+4	12
カメラ 1+2+3	7
カメラ 1+2+4	11
カメラ 1+3+4	13
カメラ 2+3+4	14
カメラ 1+2+3+4	15

戻り値：なし。

説明

IV-S2*0X の場合：基準画像を上書き登録します。

IV-S3*0X の場合：最後に取り込まれたカメラ画像を基準画像として保存（不揮発メモリーへ）します。

【注意】 品種毎に各カメラの基準画像を保存する必要があります。

本コマンド実行前に、同じ品種で 1 回以上の検査・計測を行う必要があります。

用例

Dim caoCtrl as Object

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")
caoCtrl.RegStdImage 1,1
```

実装変数.GetShutterSp

機能 シャッター速度を読み出します。

書式 **実装変数.GetShutterSp**(<品 種 番 号>, <モ ジ ュ ー ル 番 号>,
<カメラ番号>)

引数：<品種番号> 品種種類を設定（0～2047）
 <モジュール番号> モジュール番号を設定（0～127）
 <カメラ番号> カメラ番号を設定（1～4）

戻り値：シャッター速度。

説明 シャッター速度を読み出します。

用例

```
Dim caoCtrl as Object  
DIM iShutterSp As Integer
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
iShutterSp = caoCtrl.GetShutterSp(0,1,1)
```

実装変数.PutShutterSp

機能 シャッター速度を書き込みます。

書式 **実装変数.PutShutterSp** <品種番号>, <モジュール番号>,
<カメラ番号>, <シャッター速度>

引数 : <品種番号> 品種種類を設定 (0～2047)
<モジュール番号> モジュール番号を設定 (0～127)
<カメラ番号> カメラ番号を設定 (1～4)
<シャッター速度> シャッター速度を設定 (0～38000)

戻り値 : なし。

説明 シャッター速度を書き込みます。

用例

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
caoCtrl.PutShutterSp 0,1,1,5000
```

実装変数.GetThreshold

機能 しきい値を読み出します。

書式 **実装変数.GetThreshold(** <品種番号>, <モジュール番号> **)**

引数 : <品種番号> 品種種類を設定 (0～2047)
 <モジュール番号> モジュール番号を設定 (0～127)

戻り値 : しきい値。

説明 しきい値を読み出します。
しきい値は上限値、下限値の順番に配列として格納されます。

用例

```
Dim caoCtrl as Object  
Dim vntThreshold As Variant
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV","", "Conn=eth:192.168.0.2:2001")  
vntThreshold = caoCtrl.GetThreshold(0,2)
```

実装変数.PutThreshold

機能 しきい値設定を書き込みます。

書式 **実装変数.PutThreshold** <品種番号>, <モジュール番号>, <しきい値上限値>, <しきい値下限値>

引数 : <品種番号>	品種種類を設定 (0～2047)
<モジュール番号>	モジュール番号を設定 (0～127)
<しきい値上限値>	しきい値上限値を設定 (0～255)
<しきい値下限値>	しきい値下限値を設定 (0～255)

戻り値 : なし。

説明 しきい値設定を書き込みます。

用例

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
caoCtrl.PutThreshold 0,2,200,100
```

実装変数.GetGain

機能 ゲイン・オフセット設定を読み出します。

書式 **実装変数.GetGain**(<品種番号>, <モジュール番号>, <カメラ番号>)

引数 : <品種番号> 品種種類を設定 (0～2047)
 <モジュール番号> モジュール番号を設定 (0～127)
 <カメラ番号> カメラ番号を設定 (1～4)

戻り値 : ゲイン値・オフセット値。

説明 ゲイン・オフセット設定を読み出します。
 ゲイン値、オフセット値の順番に配列として格納されます。

用例

```
Dim caoCtrl as Object  
Dim vntGain As Variant
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV","", "Conn=eth:192.168.0.2:2001")  
vntGain = caoCtrl.GetGain(0,2,0)
```


実装変数.PutGain

機能 ゲイン・オフセット設定を書き込みます。

書式 **実装変数.PutGain** <品種番号>, <モジュール番号>, <カメラ番号>, <ゲイン値>, <オフセット値>

引数 : <品種番号>	品種種類を設定 (0～2047)
<モジュール番号>	モジュール番号を設定 (0～127)
<カメラ番号>	カメラ番号を設定 (1～4)
<ゲイン値>	ゲイン値を設定 (0～1023)
<オフセット値>	オフセット値を設定 (0～1023)

戻り値 : なし。

説明 ゲイン・オフセット設定を書き込みます。

用例

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
caoCtrl.Putgain 0,2,100,100
```

実装変数.GetDate

機能 日時設定を読み出します。

書式 実装変数.GetDate()

引数：なし。

戻り値：日時設定

説明 日時設定を読み出します。
年、月、日、時、分、秒の順番に配列として格納されます。

用例

```
Dim caoCtrl as Object  
Dim vntDate As Variant
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV","", "Conn=eth:192.168.0.2:2001")  
vntDate = caoCtrl.GetDate()
```

実装変数.PutDate

機能 日時設定を書き込みます。

書式 **実装変数.PutDate** <年>, <月>, <日>, <時>, <分>, <秒>

引数 : <年> 年を設定 (2000～)
<月> 月を設定 (1～12)
<日> 日を設定 (1～31)
<時> 時を設定 (0～23)
<分> 分を設定 (0～59)
<秒> 秒を設定 (0～59)

戻り値 : なし。

説明 日時設定を書き込みます。

用例

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV","", "Conn=eth:192.168.0.2:2001")  
caoCtrl.PutDate 2099,1,1,0,0,0
```

実装変数.GetRegData

機能

コードリーダーモジュールの登録データを取得します。

書式

実装変数.GetRegData(<トリガ番号>, <モジュール番号>)

引数 : <トリガ番号> トリガ番号を指定 (0~1)
 <モジュール番号> モジュール番号を指定 (0~127)

戻り値 : 登録データ

説明

コードリーダーモジュールの登録データを取得します。

COM 通信を使用する場合は、機器側とプロバイダ側でデータビット数を 8bit に設定してください。

データビット数の設定がデフォルトで 7bit となっているため、使用する文字列によっては、8bit 目が欠落し、意図した動作となりません。

実装変数.PutRegData

機能

コードリーダーモジュールの登録データを書き込みます。

書式

実装変数. PutRegData <トリガ番号>, <モジュール番号>,
<登録データ>

引数 : <トリガ番号> トリガ番号を指定 (0～1)
 <モジュール番号> モジュール番号を指定 (0～127)
 <登録データ> 登録データを指定

戻り値 : なし。

説明

コードリーダーモジュールの登録データを書き込みます。

COM 通信を使用する場合は、機器側とプロバイダ側でデータビット数を 8bit に設定してください。

データビット数の設定がデフォルトで 7bit となっているため、使用する文字列によっては、8bit 目が欠落し、意図した動作となりません。

実装変数.PutDataString

機能 文字検査モジュールの設定文字列を書き込みます。

書式

実装変数. PutDataString <トリガ番号>, <カメラ番号>
 [, <年 1>[, <月 1>[, <日 1>[, <年 2>[, <月 2>[, <日 2>
 [, <年 3>[, <月 3>[, <日 3>[, <年 4>[, <月 4>[, <日 4>
 [, <年 5>[, <月 5>[, <日 5>[, <年 6>[, <月 6>[, <日 6>
 [, <年 7>[, <月 7>[, <日 7>
 [, <年 8>[, <月 8>[, <日 8>]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]

引数：〈トリガ番号〉	トリガ番号を指定 (0～1)
〈カメラ番号〉	カメラ番号を指定 (1～4)
〈年 1〉	年 1 を指定
〈月 1〉	月 1 を指定
〈日 1〉	日 1 を指定
〈年 2〉	年 2 を指定
〈月 2〉	月 2 を指定
〈日 2〉	日 2 を指定
〈年 3〉	年 3 を指定
〈月 3〉	月 3 を指定
〈日 3〉	日 3 を指定
〈年 4〉	年 4 を指定
〈月 4〉	月 4 を指定
〈日 4〉	日 4 を指定
〈年 5〉	年 5 を指定
〈月 5〉	月 5 を指定
〈日 5〉	日 5 を指定
〈年 6〉	年 6 を指定
〈月 6〉	月 6 を指定
〈日 6〉	日 6 を指定
〈年 7〉	年 7 を指定
〈月 7〉	月 7 を指定
〈日 7〉	日 7 を指定
〈年 8〉	年 8 を指定
〈月 8〉	月 8 を指定
〈日 8〉	日 8 を指定

戻り値：なし。

説明

文字検査モジュールの設定文字列を書き込みます。

COM 通信を使用する場合は、機器側とプロバイダ側でデータビット数を 8bit に設定してください。

データビット数の設定がデフォルトで 7bit となっているため、使用する文字列によっては、8bit 目が欠落し、意図した動作となりません。

実装変数.GetModuleString

機能 文字検査モジュールの設定文字列を読み出します。

書式 **実装変数. GetModuleString(<トリガ番号>, <モジュール番号>)**

引数 : <トリガ番号> トリガ番号を指定 (0～1)
 <モジュール番号> モジュール番号を指定 (0～127)

戻り値 : 登録データ

説明 文字検査モジュールの設定文字列を読み出します。
COM 通信を使用する場合は、機器側とプロバイダ側でデータビット数を 8bit に設定してください。
データビット数の設定がデフォルトで 7bit となっているため、使用する文字列によっては、8bit 目が欠落し、意図した動作となりません。

実装変数.GetBlockString

機能

文字検査モジュールの設定文字列を読み出します。

書式

実装変数.GetBlockString(<トリガ番号>, <モジュール番号>,
<ブロック番号>)

引数 : <トリガ番号> トリガ番号を指定 (0～1)
 <モジュール番号> モジュール番号を指定 (0～127)
 <ブロック番号> ブロック番号を指定 (0～7)

戻り値 : 登録データ

説明

文字検査モジュールの設定文字列を読み出します。

COM 通信を使用する場合は、機器側とプロバイダ側でデータビット数を 8bit に設定してください。

データビット数の設定がデフォルトで 7bit となっているため、使用する文字列によっては、8bit 目が欠落し、意図した動作となりません。

実装変数.PutBlockString

機能

文字検査モジュールの設定文字列を書き込みます。

書式

実装変数. PutBlockString <トリガ番号>, <モジュール番号>, <ブロック番号>, <文字列>, <空白除去あり/なし>

引数 : <トリガ番号> トリガ番号を指定 (0～1)
 <モジュール番号> モジュール番号を指定 (0～127)
 <ブロック番号> ブロック番号を指定 (0～7)
 <文字列> 文字列を指定
 <空白除去あり/なし> 空白除去あり/なしを指定

TRUE	空白除去あり
FALSE	空白除去なし

戻り値 : なし。

説明

文字検査モジュールの設定文字列を書き込みます。

空白除去は、文字列末尾の空白の除去を行います。

COM 通信を使用する場合は、機器側とプロバイダ側でデータビット数を 8bit に設定してください。

データビット数の設定がデフォルトで 7bit となっているため、使用する文字列によっては、8bit 目が欠落し、意図した動作となりません。

実装変数.Put20BlockString

機能 文字検査モジュールの設定文字列を書き込みます。

書式

実装変数.Put20BlockString <トリガ番号>[,<文字列 1>
[,<文字列 2>[,<文字列 3>[,<文字列 4>[,<文字列 5>[,<文字列 6>
[,<文字列 7>[,<文字列 8>[,<文字列 9>[,<文字列 10>
[,<文字列 11>[,<文字列 12>[,<文字列 13>[,<文字列 14>
[,<文字列 15>[,<文字列 16>[,<文字列 17>[,<文字列 18>
[,<文字列 19>[,<文字列 20>]]]]]]]]]]]]]]]]]]]]]

引数：〈トリガ番号〉	トリガ番号を指定 (0～1)
〈文字列 1〉	文字列 1 を指定
〈文字列 2〉	文字列 2 を指定
〈文字列 3〉	文字列 3 を指定
〈文字列 4〉	文字列 4 を指定
〈文字列 5〉	文字列 5 を指定
〈文字列 6〉	文字列 6 を指定
〈文字列 7〉	文字列 7 を指定
〈文字列 8〉	文字列 8 を指定
〈文字列 9〉	文字列 9 を指定
〈文字列 10〉	文字列 10 を指定
〈文字列 11〉	文字列 11 を指定
〈文字列 12〉	文字列 12 を指定
〈文字列 13〉	文字列 13 を指定
〈文字列 14〉	文字列 14 を指定
〈文字列 15〉	文字列 15 を指定
〈文字列 16〉	文字列 16 を指定
〈文字列 17〉	文字列 17 を指定
〈文字列 18〉	文字列 18 を指定
〈文字列 19〉	文字列 19 を指定
〈文字列 20〉	文字列 20 を指定

戻り値：なし。

説明

文字検査モジュールの設定文字列を書き込みます。

COM 通信を使用する場合は、機器側とプロバイダ側でデータビット数を 8bit に設定してください。

データビット数の設定がデフォルトで 7bit となっているため、使用する文字列によっては、8bit 目が欠落し、意図した動作となりません。

実装変数.Put10BlockString

機能

文字検査モジュールの設定文字列を書き込みます。

書式

実装変数. Put10BlockString <トリガ番号>, <カメラ番号>
 [, <文字列 1>[, <文字列 2>[, <文字列 3>[, <文字列 4>
 [, <文字列 5>[, <文字列 6>[, <文字列 7>[, <文字列 8>
 [, <文字列 9>[, <文字列 10>]]]]]]]]]

引数 : <トリガ番号>	トリガ番号を指定 (0～1)
<カメラ番号>	カメラ番号を指定 (1～4)
<文字列 1>	文字列 1 を指定
<文字列 2>	文字列 2 を指定
<文字列 3>	文字列 3 を指定
<文字列 4>	文字列 4 を指定
<文字列 5>	文字列 5 を指定
<文字列 6>	文字列 6 を指定
<文字列 7>	文字列 7 を指定
<文字列 8>	文字列 8 を指定
<文字列 9>	文字列 9 を指定
<文字列 10>	文字列 10 を指定

戻り値 : なし。

説明

文字検査モジュールの設定文字列を書き込みます。

COM 通信を使用する場合は、機器側とプロバイダ側でデータビット数を 8bit に設定してください。

データビット数の設定がデフォルトで 7bit となっているため、使用する文字列によっては、8bit 目が欠落し、意図した動作となりません。

実装変数.SnapShot

機能 スナップショット画像を外部メモリに保存します。

書式 **実装変数.SnapShot**

引数：なし。

戻り値：なし。

説明 スナップショット画像を外部メモリに保存します。

用例

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV","", "Conn=eth:192.168.0.2:2001")  
caoCtrl.SnapShot
```

実装変数.ImageClear

機能 本体内蔵の画像メモリを消去します。

書式 実装変数.ImageClear

引数：なし。

戻り値：なし。

説明 本体内蔵の画像メモリを消去します。

用例

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV","", "Conn=eth:192.168.0.2:2001")  
caoCtrl.ImageClear
```

実装変数.GetKindState

機能 品種設定番号情報を読み出します。

書式 実装変数.GetKindState()

引数：なし。

戻り値：品種設定番号情報

説明 品種設定番号情報を読み出します。
品種設定データ数、品種番号 1、品種番号 2、・・・品種番号 n の順番に配列として格納されます。

用例

```
Dim caoCtrl as Object  
Dim vntKindState As Variant
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
vntKindState = caoCtrl.GetKindState()
```

実装変数.GetStdImageState

機能 基準画像番号情報を読み出します。

書式 実装変数.GetStdImageState()

引数：なし。

戻り値：基準画像番号情報

説明 基準画像番号情報を読み出します。
基準画像番号数、基準画像番号 1、基準画像番号 2、・・・基準画像番号 n の
順番に配列として格納されます。

用例

```
Dim caoCtrl as Object
Dim vntStdImageState As Variant

caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")
vntStdImageState = caoCtrl.GetStdImageState()
```


実装変数.GetVersion

機能 バージョン情報を読み出します。

書式 **実装変数.GetVersion()**

引数：なし。

戻り値：バージョン情報

説明 バージョン情報を読み出します。
機種コード、バージョン情報の順番に配列として格納されます。

用例

```
Dim caoCtrl as Object  
Dim vntVersion As Variant
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
vntVersion = caoCtrl.GetVersion()
```

実装変数.AllReset

機能 システム設定、品番設定を初期化します。

書式 **実装変数.AllReset**

引数：なし。

戻り値：なし。

説明 システム設定、品番設定を初期化します。

用例

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
caoCtrl.AllReset
```

実装変数.SettingSave

機能 システム設定、品番設定を保存します。

書式 実装変数.SettingSave

引数：なし。

戻り値：なし。

説明 システム設定、品番設定を保存します。

用例

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
caoCtrl.SettingSave
```

実装変数.Reset

機能 本機をリセットします。

書式 実装変数.Reset

引数：なし。

戻り値：なし。

説明 本機をリセットします。

用例

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
caoCtrl.Reset
```

実装変数.GetBrightness

機能 平均濃度を読み出します。

書式 **実装変数.GetBrightness**(<カメラ番号>, <左上 X 座標値>, <左上 Y 座標値>, <右下 X 座標値>, <右下 Y 座標値>)

引数 : <カメラ番号>	カメラ番号を設定 (1～4)
<左上 X 座標値>	左上 X 座標値を設定
<左上 Y 座標値>	左上 Y 座標値を設定
<右下 X 座標値>	右下 X 座標値を設定
<右下 Y 座標値>	右下 Y 座標値を設定

戻り値 : 平均濃度。

説明 平均濃度を読み出します。

用例

```
Dim caoCtrl as Object
DIM iBrightness As Integer
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")
iBrightness = caoCtrl.GetBrightness(0,100,100,200,200)
```

実装変数.GetParallel

機能 パラレル入出力情報を読み出します。

書式 **実装変数.GetParallel**(〈入出力指定〉)

引数：〈入出力指定〉 入出力指定を設定

0	入出力
1	入力のみ
2	出力のみ

戻り値：入出力状態。

説明 変数値を読み出します。
 入力 1 バイト目～4 バイト目、出力 1 バイト目～5 バイト目の順番で、1 バイト毎に配列として格納されます。
 戻り値のフォーマットが IV のシリーズ毎で異なりますので、過去の機種から置き換えを行う場合はユーザー側で対応が必要です。

用例

```
Dim caoCtrl as Object
Dim vntParallel As Variant
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")
vntParallel = caoCtrl.GetParallel(0)
```

実装変数.SelfCheck

機能 コントローラの自己診断(5種類のテスト)を実行します。

書式 実装変数. SelfCheck ()

引数 : なし。

戻り値 : システムメモリーテスト結果 ※1

 : RAM テスト結果 ※1

 : FPGA アクセステスト結果 ※1

 : カメラ 1 接続テスト結果 ※2

 : カメラ 2 接続テスト結果 ※2

 : カメラ 3 接続テスト結果 ※2

 : カメラ 4 接続テスト結果 ※2

※1 テスト結果の値は次表のとおりです

値	結果
0	正常
1	異常

※2 カメラテスト結果の値は次表のとおりです。

値	結果
0	正常
1	カメラ接続テスト失敗
2	カメラ種別テスト失敗
3	カメラ視野テスト失敗
4	カメラ取込テスト失敗
5	カメラ取込ラインテスト失敗

説明 コントローラの自己診断(5種類のテスト)を実行します。

実装変数.Raw

機能 生データ送受信を行います。

書式 **実装変数.Raw**(〈送信文字列〉)

引数：〈送信文字列〉 送信文字列を設定

戻り値：受信文字列。

説明 引数で指定した文字列を送信します。受信したレスポンスを文字列で返します。
このとき送受信でコマンド及び、レスポンスの内容は一切加工されません。

用例

```
Dim caoCtrl as Object  
Dim Res As Variant
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
Res = caoCtrl.Raw(":00000000"&"00"&"T00"&"0,"&"@@")
```


実装変数.ChangeTimeout

機能 通信タイムアウト時間を設定します。

書式 **実装変数.ChangeTimeout** <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設定

戻り値 : なし。

説明 通信タイムアウト時間を設定します。
タイムアウト時間に-1 を指定すると **AddController** 時のタイムアウト時間を設定します。

用例

```
Dim caoCtrl as Object
```

```
caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")  
caoCtrl.ChangeTimeout 1000
```

6. エラーコード

IVプロバイダでは、以下の固有エラーコードが定義されています。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。

エラー名	エラー番号	説明
E_RESPONSE_FAILED	0x80100001	無効なレスポンスを受信しました
E_RESPONSE_CHECKSUM_FAILED	0x80100002	レスポンスデータチェックサムエラー
E_RESPONSE_COMMAND_FAILED	0x80100003	レスポンスデータコマンドエラー
E_RESPONSE_LENGTH_FAILED	0x80100004	レスポンスデータ長エラー
E_RESPONSE_AREA_FAILED	0x80100005	レスポンス局番エラー
E_DEVICE_ERROR	0x80100100～	デバイスエラー

※ E_DEVICE_ERRORについて

デバイスからのエラーレスポンスは”0x80100100”でマスクした値が出力されます。

例：コマンド実行不可：40(H) → CAOAPIのエラー：0x80100140

エラー内容についてはシャープマニファクチャリングシステム社の画像センサユーザーズマニュアルを参照下さい。

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IV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。



説明 各ボタンの動作内容について説明します。

1. 品種番号を設定します。
 2. 1. で設定した品種番号に切替えます。(PutKind)
 3. トリガ番号を設定します。
 4. 3. で設定したトリガ番号でトリガを入力します。(Trigger)
 5. データを受信します。(GetData)
 6. 3. で設定したトリガ番号でトリガを入力し、データを受信します。(TriggerAndWait)
 7. 処理結果の状態を表示します。
 8. 受信データの表示ページを Up します。
 9. 受信データの表示ページを Down します。
 10. 受信データを表示します。
- 注：プロバイダの実装（初期化）が正常に行われた場合は、7. に“接続完了”と表示されます。

8. サンプルプログラム

8.1 トリガを入力し結果を取得

Sub Main

```

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

Dim caoCtrl as Object          'プロバイダ用変数宣言
Dim Res As Variant             'Variant 変数型変数宣言
Dim pTargetPos as Position     'P 変数型変数宣言

takearm keep = 0
pTargetPos = P11

caoCtrl = cao.AddController("IV_S150X", "CaoProv.SHARP.IV", "", "Conn=eth:192.168.0.2:2001")
                                                    'プロバイダの実装

caoCtrl.PutKind 1              '品種 1 に切替え
Res = caoCtrl.TriggerAndWait(0) 'トリガーを入力し、結果データを受信

letx pTargetPos = posx(P11) + Res(1) '受信データの X 成分を位置データへ展開
lety pTargetPos = posy(P11) + Res(2) '受信データの Y 成分を位置データへ展開

approach p, pTargetPos, @p 20, s = 100 '補正後の位置上空へ
move l, @e pTargetPos, s = 10          '補正後の位置へ
Hand[0].Chuck 0                        'チャック
depart l, @p 50, s = 100               '上昇

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」

End Sub

```

※IV にトリガを入力し結果を取得する方法は、前述の様に **TriggerAndWait** コマンドを使って、トリガ入力から結果受信まで一度に行うことも出来ますし、下記のように **Trigger** コマンドを使いトリガを入力し、その後 **GetData** コマンドで結果を取得することも出来ます。この場合、IV 側の画像処理終了前に **GetData** コマンドを実行すると、画像処理終了前の結果を取得してしまうため、IV の RDY 信号などで画像処理終了を確認してから結果を取得する必要があります。

```
caoCtrl.Trigger 0
```

```
Res = caoCtrl.GetData(0)
```

8.2 キャリブレーション

'TV 機器の IP アドレス

```
#Define IP_ADDRESS "192.168.1.20"
```

'ロボットのホームポジションを格納した P 型変数の Index (最初と最後にこの座標へ移動します)

'事前にホームポジションの座標を P[HOME_POSITION_INDEX]に設定してください

```
#Define HOME_POSITION_INDEX 0
```

'RobotCalibration 開始位置(1 点目)の座標を格納した P 型変数の Index

'事前に開始位置の座標を P[START_POSITION_INDEX]に設定してください

```
#Define START_POSITION_INDEX 1
```

Sub Main

```
On Error GoTo ErrHandler
```

```
Dim objCtrl As Object
```

```
objCtrl = cao.AddController("iv", "CaoProv.SHARP.IV", "", "Conn=eth:" & IP_ADDRESS _  
    & ",Type=2,Checksum=1")
```

```
Dim CurrentPosition As Position
```

```
Dim CurrentX as Double
```

```
Dim CurrentY as Double
```

```
Dim CurrentZ as Double
```

```
Dim CurrentRx as Double
```

```
Dim CurrentRy as Double
```

```
Dim CurrentRz as Double
```

```
TakeArm Keep = 1
```

```
ExtSpeed 10
```

'ホームポジションへ移動

Hold "ホームポジションへ移動させます。周囲の安全を確認した後、再実行してください。"

```
Move P,P[HOME_POSITION_INDEX]
```

'RobotCalibration 開始位置へ移動

Hold "開始位置へ移動させます。周囲の安全を確認した後、再実行してください。"

Move P,P[START_POSITION_INDEX]

RestartProc:

CurrentPosition = CurPos

CurrentX = PosX(CurrentPosition)

CurrentY = PosY(CurrentPosition)

CurrentZ = PosZ(CurrentPosition)

CurrentRx = PosRx(CurrentPosition)

CurrentRy = PosRy(CurrentPosition)

CurrentRz = PosRz(CurrentPosition)

Dim varRet as Variant

' RobotCalibration 実行

varRet = objCtrl.RobotCalibration(0, CurrentX, CurrentY, CurrentRz)

IF(varRet(0) = False) Then

Hold "ロボットが動作します。周囲の安全を確認した後、再実行してください。"

'ロボット移動

Move P,P(varRet(1), varRet(2), CurrentZ, CurrentRx, CurrentRy, varRet(3), -2)

Goto RestartProc

End If

'ホームポジションへ移動

Hold "ホームポジションへ移動させます。周囲の安全を確認した後、再実行してください。"

Move P,P[HOME_POSITION_INDEX]

PostProc:

IF(IsNothing(objCtrl) = False)Then

cao.Controllers.Remove objCtrl.Index

objCtrl = Nothing

End If

Exit Sub

ErrorHandler:

MsgBox (Err.Description) , 0+16, ErrMsg(Err.Number) & " : " & (Hex(Err.Number))

Goto PostProc

End Sub

改訂履歴

デンソーロボット プロバイダ 取扱説明書

シャープマニファクチャリングシステム株式会社製 画像センサカメラ IV シリーズ

バージョン	対応RC8	改訂内容
Ver.1.0.0	Ver.1.4.5	初版
Ver.1.0.1	Ver.2.2.*	IV-S300X, IV-S310X対応 コマンド追加 (GetRegData, PutRegData, PutDateString, GetModuleString, GetBlockString, PutBlockString, Put20BlockString, Put10BlockString, SysSettingSave, SelfCheck, ClearFigures) コマンド修正 (PutKind, PutViewMode, PutDispMode, RegStdImage)
Ver.1.0.2	Ver.2.5.*	IV-S301M対応 SAlignmentTrigger, SAlignmentCalibration, RobotCalibration追加
Ver.1.0.3	Ver.2.8.*	IV-S402M, IV-S412M対応

株式会社デンソーウェブ

- この取扱説明書の一部または全部を無断で複製・転載することはお断りします。
- この説明書の内容は将来予告なしに変更することがあります。
- 本書の内容については、万全を期して作成いたしました。が、万一ご不審の点や誤り、記載もれなど、お気づきの点がありましたら、ご連絡ください。
- 運用した結果の影響については、上項にかかわらず責任を負いかねますのでご了承ください。

DENSO Robotics
THIRD PARTY PRODUCTS

株式会社 デンソーウェーブ