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

パナソニック デバイスSUNX(株) 製

製品／シリーズ

ビジョンセンサ

MODEL:PV シリーズ

Vision

はじめに

本書は、「パナソニック デバイスＳＵＮＸ(株)製・ビジョンセンサ・ＰＶ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ＰＶ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、「パナソニック デバイスＳＵＮＸ(株)製・ビジョンセンサ・ＰＶ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

パナソニック デバイスＳＵＮＸ(株)製　　PV200／PV500 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

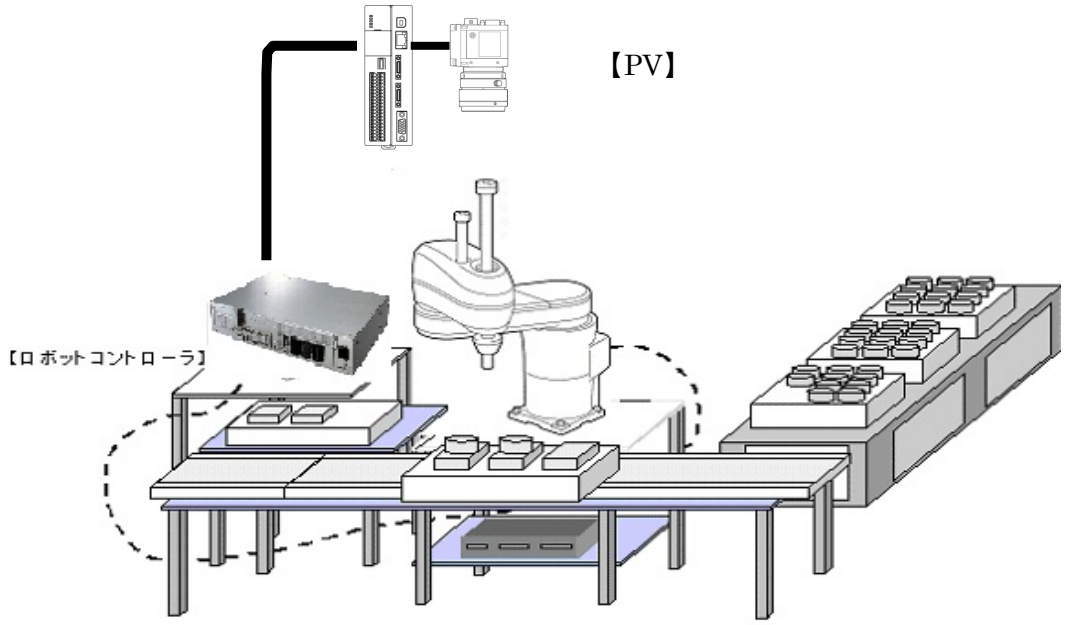
本製品（ソフトウェア）のシステム組み込み・使用ならびに本製品の使用結果に関する一切のリスクは、本製品の使用者に帰属するものとします。

目次

1. 本製品（プロバイダ）の概要

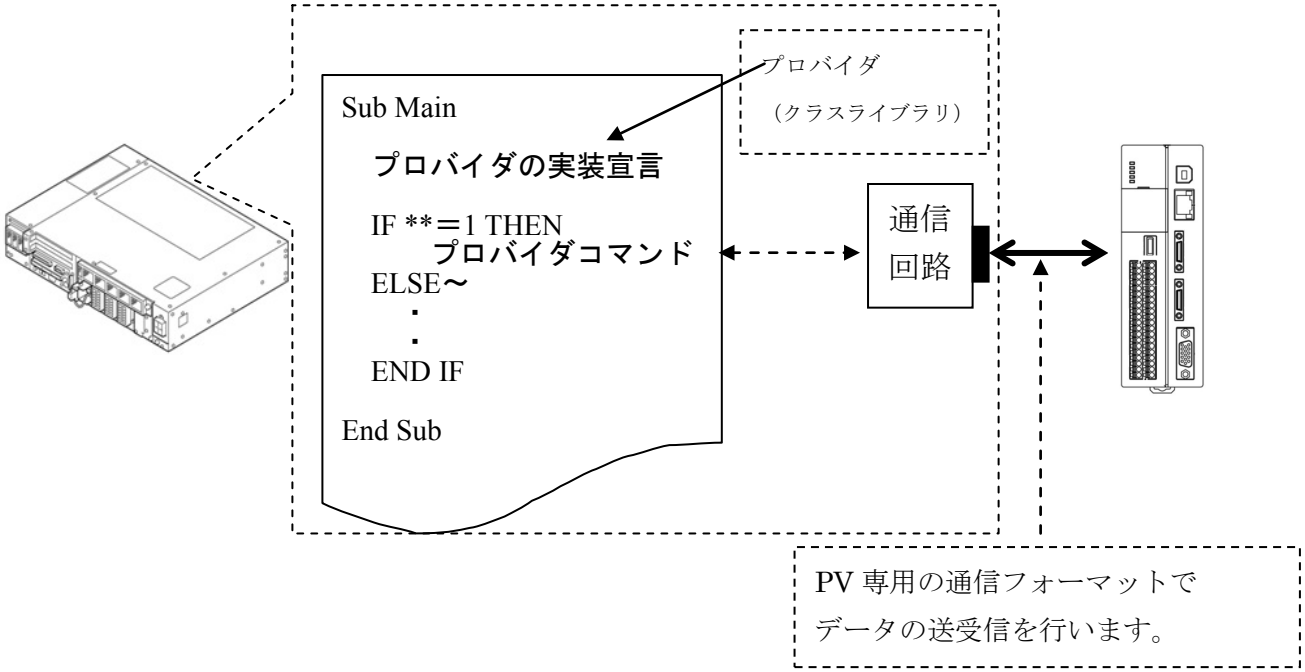
1.1 プロバイダの対象機器

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ＰＶシリーズを接続する時に使用することができます。



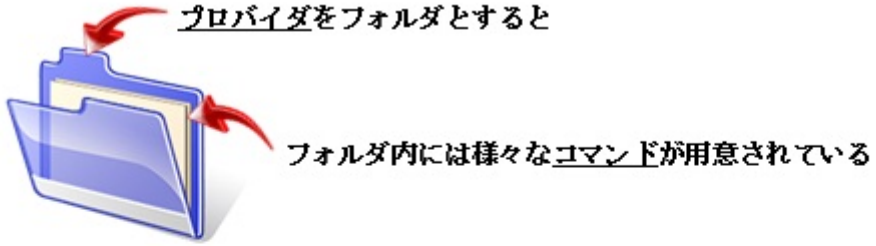
1.2 プロバイダの特長

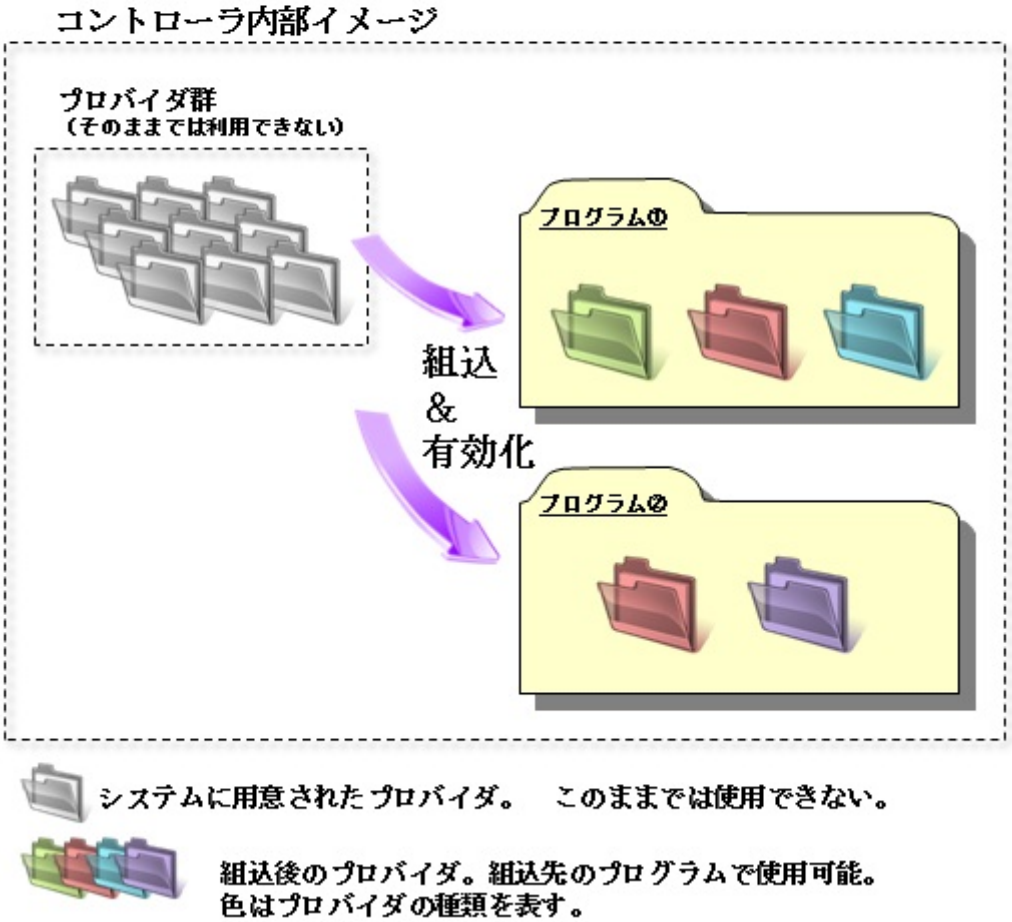
ＰＶシリーズにアクセスするために必要なＰＶのネイティブコマンドを、ロボットプログラムで使用できるプロバイダとして準備しています。本プロバイダを使用することで、ＰＶシリーズの通信部分のプログラムを組むことなく、容易にロボットからの通信を行うことができます。下記にプロバイダの組込関係を示します。



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。

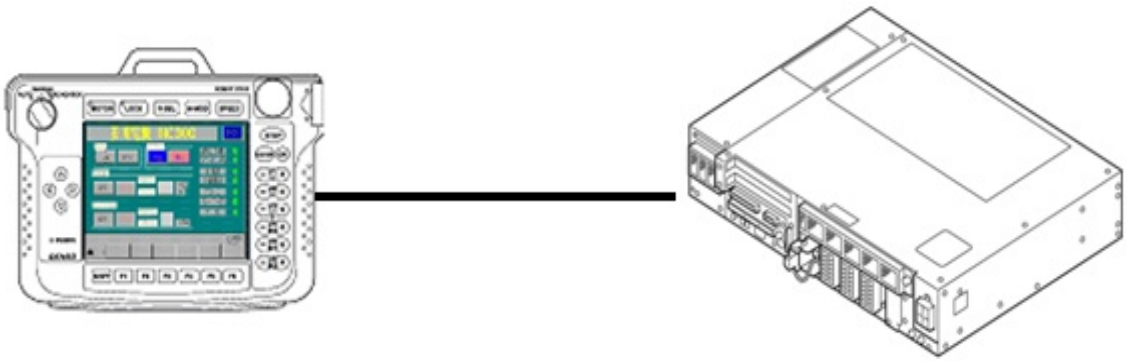
2. 接続方法

2.1 Ethernet（TCP/IP）接続例

ロボットコントローラとPVシリーズをEthernet(TCP/IP)接続する際には、クロスイーサネットケーブルをご使用下さい。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。

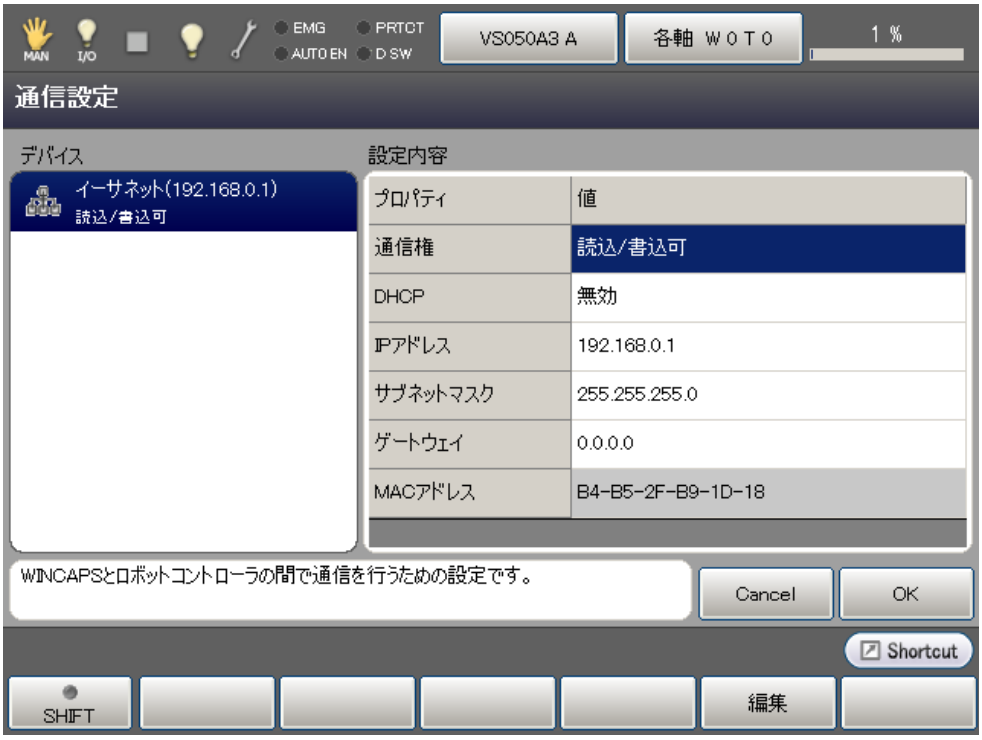


3.1 Ethernet（TCP/IP）による通信

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。

［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PV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。

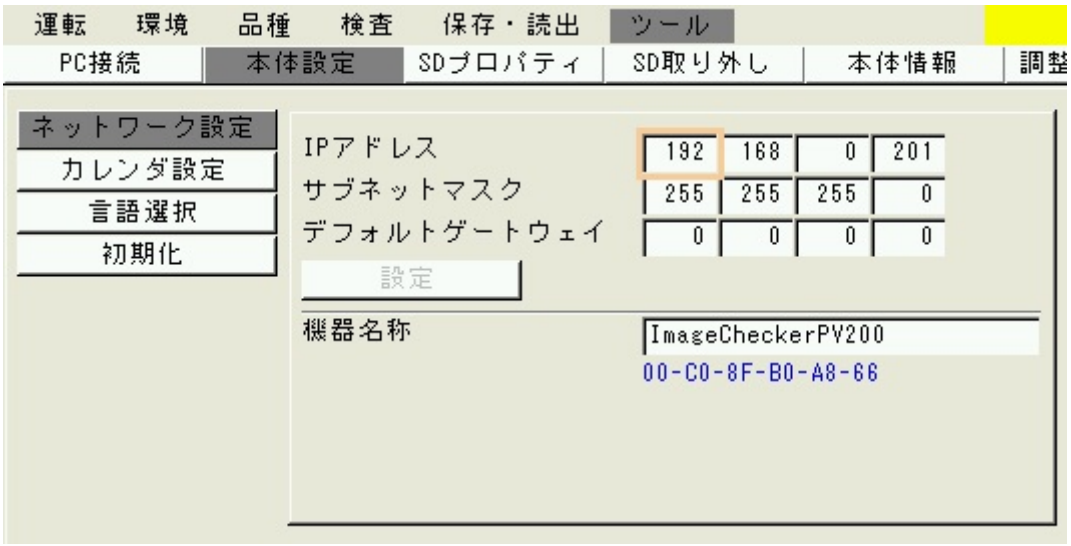


3.2.2 PV の Ethernet（TCP/IP）通信設定

PVのIPアドレスを設定します。

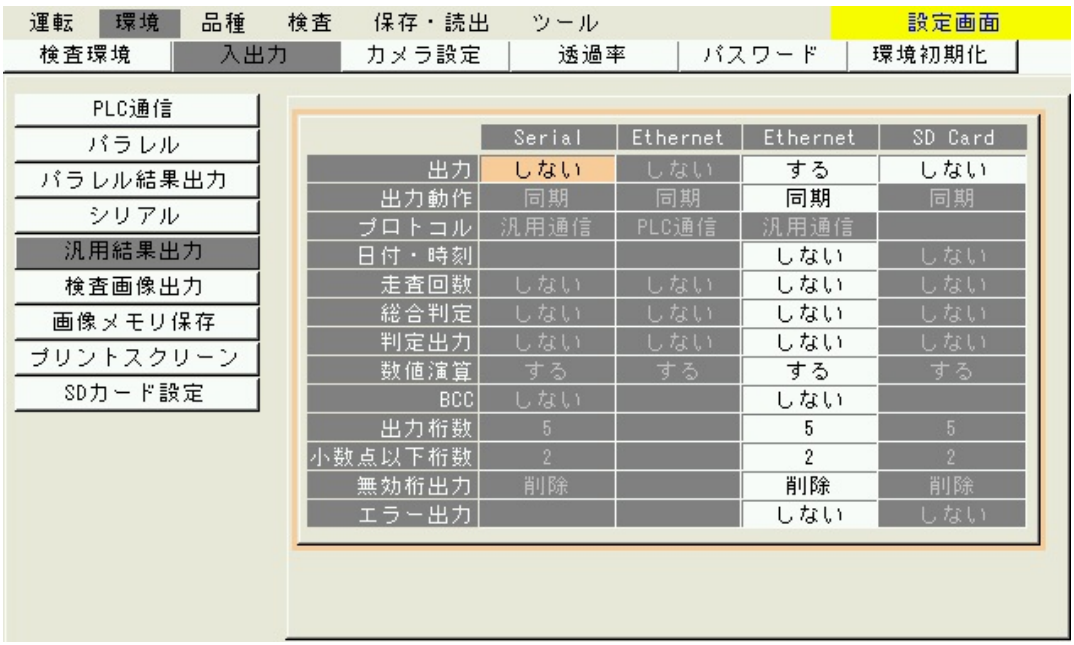
[ツール]－[ネットワーク設定]を選択します。ネットワーク設定画面で IP アドレスとサブネットマスクを設定して下さい。ロボットコントローラと PV が、同一のサブネットマスク内になるように IP アドレス及び、サブネットマスクを設定して下さい。

【PV の画面】



3.2.3 PV の結果出力設定

[環境]－[入出力]－[汎用結果出力]を選択します。Ethernetの汎用出力（プロトコル）の設定を行って下さい。なお、詳細な設定については、パナソニック社製PVユーザーズマニュアルを参照下さい。



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 strResult as String    ③                 '結果取得用変数宣言

caoCtrl = cao.AddController("PV", "caoProv.Panasonic.PV", "", "conn=eth:192.168.0.201") 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、PV コマンド、PV 汎用通信コマンド、独自拡張コマンドに分類されます。尚、PV コマンドの詳細動作については Panasonic 社の PV シリーズのマニュアル汎用通信コマンド詳細を参照してください。

表 5-1 コマンド一覧

コマンド	PVコマンド	機能	PV260専用	参照
接続コマンド				
cao.AddController	－	プロバイダを変数に実装して、PV に接続処理を行います		13
Addvariable	－	画像やセルの値を取得する為の専用変数を作成します		15
Value	－	AddValiable で作成した変数を経由して取得します		16
PV コマンド				
Start	%S	検査実行		17
Restart	%R	再検査実行（撮像せずに現在のメモリ画像で検査する）		18
Xtype	%X	品種切り替え		19
MemoryWrite	%MW	設定データ保存　本体保存用メモリ		20
CFWrite	%CW	設定データ保存　SD カードメモリ		21
MemoryRead	%MR	設定データ読み出し　本体保存用メモリ		22
CFRead	%CR	設定データ読み出し　SD カードメモリ		23
CancelData	%CD	設定データ保存/読み出しの中断（キャンセル）		24
SDSave	%SS	保存画像メモリ　保存(SD メモリーカード)		25
SDReset	%SR	保存画像メモリ　消去		26
PrintScreen	%PS	プリントスクリーン		27
Quit	%Q	統計リセット		28
RunManual	%RM	運転/停止（RUN/STOP）の切り替え		29
ErrorReset	%E	エラー信号のリセット		30
Cancel	%CC	検査/処理の中断（各種動作キャンセル）		31
KeyEmulator	%K	キーエミュレート		32
Bstop	%BS	キーパッド操作　受付禁止/受付許可		33
Bconfirm	%BC	キーパッド操作　受付状態の確認		34
LayOutChange	%I	レイアウト切り替え		35
AgainTemplate	%A	テンプレートの再登録		36
ParameterRead	%PR	パラメータ　読み出し		37
ParameterReadPair	%PRP	パラメータ　ペア読み出し（各種上下限値など）		38
ParameterWrite	%PW	パラメータ　変更		39
ParameterWritePair	%PWP	パラメータ　ペア変更（各種上下限値など）		40
PV 汎用通信コマンド(非同期)				
StartAsync	%S	検査実行		41
ReStartAsync	%R	再検査実行（撮像せずに現在のメモリ画像で検査する）		42
XTypeAsync	%X	品種切り替え		43

MemoryWriteAsync	%MW	設定データ保存　本体保存用メモリ		44
CFWriteAsync	%CW	設定データ保存　SD カードメモリ		45
MemoryReadAsync	%MR	設定データ読み出し　本体保存用メモリ		46
CFReadAsync	%CR	設定データ読み出し　SD カードメモリ		47
CancelDataAsync	%CD	設定データ保存/読み出しの中断（キャンセル）		48
SDSaveAsync	%SS	保存画像メモリ　保存(SD メモリーカード)		49
SDResetAsync	%SR	保存画像メモリ　消去		50
PrintScreenAsync	%PS	プリントスクリーン		51
QuitAsync	%Q	統計リセット		52
RunManualAsync	%RM	運転/停止（RUN/STOP）の切り替え		53
ErrorResetAsync	%E	エラー信号のリセット		54
CancelAsync	%CC	検査/処理の中断（各種動作キャンセル）		55
KeyEmulatorAsync	%K	キーエミュレート		56
BstopAsync	%BS	キーパッド操作　受付禁止/受付許可		57
BconfirmAsync	%BC	キーパッド操作　受付状態の確認		58
LayOutChangeAsync	%I	レイアウト切り替え		59
AgainTemplateAsync	%A	テンプレートの再登録		60
ParameterReadAsync	%PR	パラメータ　読み出し		61
Parameter ReadPairAsync	%PRP	パラメータ　ペア読み出し（各種上下限値など）		62
Parameter WriteAsync	%PW	パラメータ　変更		63
Parameter-WritePairAsync	%PWP	パラメータ　ペア変更（各種上下限値など）		64
独自コマンド				
Raw	－	コマンドメッセージの送受信		65
SetTimeout	－	通信のタイムアウト時間を設定		66
GetTimeout	－	通信のタイムアウト時間を取得		67
独自コマンド(非同期コマンド用)				
RawAsync	－	コマンドメッセージの送信		68
GetResult	－	非同期コマンドの戻り値を取得		69
PV260 ロボットキャリブレーション用コマンド (同期)				
SetPoint	%P=	ロボット座標通知	○	70
Calibrate	%CA	計測開始指示	○	71
ReCalibrate	%RCA	再計測開始指示	○	72
CalibrationStart	%CAS	キャリブレーション自動設定開始	○	73
CalibrationEnd	－	キャリブレーション自動設定完了の応答受信 （CalibrationStart 関連コマンド）	○	74
WorkSet	%WCS	ワーク検出基準位置再登録	○	75
WorkReset	%WRS	ワーク検出基準位置再登録開始	○	76
WorkResetEnd	－	ワーク検出基準位置再登録完了の応答受信 （WorkReset 関連コマンド）	○	77

MoveEnd	%MVE	移動完了	○	78
GetTeachPoint	%TCD	ティーチング座標要求	○	79
GetMovePoint	－	ロボット移動座標受信 （CalibrationStart, WorkReset 関連コマンド）	○	80
PV260 ロボットキャリブレーション用コマンド (非同期)				
SetPointAsync	%P=	ロボット座標通知	○	81
CalibrateAsync	%CA	計測開始指示	○	82
ReCalibrateAsync	%RCA	再計測開始指示	○	83
Calibration-StartAsync	%CAS	キャリブレーション自動設定開始	○	84
CalibrationEndAsync	－	キャリブレーション自動設定完了の応答受信 （CalibrationStart 関連コマンド）	○	85
WorkSetAsync	%WCS	ワーク検出基準位置再登録	○	86
WorkResetAsync	%WRS	ワーク検出基準位置再登録開始	○	87
WorkResetEndAsync	－	ワーク検出基準位置再登録完了の応答受信 （CalibrationStart, WorkReset 関連コマンド）	○	88
MoveEndAsync	%MVE	移動完了	○	89
GetTeachPointAsync	%TCD	ティーチング座標要求	○	90
GetMovePointAsync	－	ロボット移動座標受信 （CalibrationStart, WorkReset 関連コマンド）	○	91

cao.AddController

機能

プロバイダを変数に実装して、PV 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.Panasonic.PV"
〈プロバイダの実行マシン名〉 省略して下さい

〈オプション〉 [PV260 パラメータ],[接続パラメータ],[タイムアウト時間],
[IP アドレス:ポート]

[PV260 パラメータ] PV260 のロボットキャリブレーション関連のコマンドを使用する場合に指定します。このオプションは Ver.1.12.*より有効です。
0 ： ロボットキャリブレーションコマンドを使用しない。(デフォルト値)
1 ： ロボットキャリブレーションコマンドを使用します。
"PV260=0" or "PV260=1"
[接続パラメータ] "conn=eth:〈IP アドレス〉"
[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：500。
[IP アドレス:ポート] 複数の NIC を使う場合にこのオプションで IP アドレスを指定して NIC を選択することができます。
省略した場合は、自動的に選択されます。ローカルマシンに割り当てられていない IP アドレスを指定したときはエラーを返します。
省略時のローカルポート番号は 0 となります。
このオプションは Ver.2.3.*から有効です。
"MyIP=[〈ローカル IP アドレス〉[:ローカルポート番号]]"

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( "PV ", "CaoProv.Panasonic.PV", "", _
                                    "PV260=1,Conn = eth:192.168.0.201" )
```

タイムアウトを指定したい場合は次のように記述します

```
caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", _
                                    "PV260=1,Conn = eth:192.168.0.201, Timeout = 1000")
```

実装変数.AddVariable

機能

画像を取得する為のプロパティ変数を作成します。

書式

実装変数.AddVariable 〈画像指定〉, [〈オプション〉]

引数： 〈画像指定〉 取得画像の種類を指定します。
@BITMAP：カメラ画像
@BITMAP_MONITOR：モニタ画像
〈オプション〉 無し

説明

PV から画像を取得する為の変数を Object 型で作成します。

用例

画像を取得し，操作盤の画面に表示する例を示します。

```
Dim caoCtrl As Object
Dim objBmp As Object
Dim vntResult as Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )

objBmp = caoCtrl.AddVariable( "@BITMAP", "" )
vntResult = objBmp.Value
```

プロパティ変数.Value

機能

画像データを、AddValiable で作成した変数を経由して取得します。

書式

プロバイダ変数.Value( )

戻り値： BITMAP フォーマットデータ。

説明

プロバイダ（実装変数）から画像データを、AddVariable で作成した変数を経由して取得します。

用例

画像を取得し、操作盤に表示する例を示します。

```
Dim caoCtrl As Object
Dim objBmp As Object
Dim vntResult as Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )

objBmp = caoCtrl.AddVariable( "@BITMAP", "" )
vntResult = objBmp.Value
```

実装変数.Start

機能

検査を実行します。実行モードが「全実行」または「分岐実行」の場合と、「指定実行」の場合で書式が異なります。画像処理結果は PV シリーズの「汎用結果出力」で設定した値を文字列で返します。

書式

実装変数.Start([<ブロックナンバー>] **)**

引数：<ブロックナンバー>　実行するブロックナンバー（整数 0～9）

戻り値：画像処理結果（文字列）

説明

一括トリガで実行モードが［指定実行］の場合のみ引数のブロックナンバーが必要です。
［全実行］、「分岐実行」の場合、引数は不要です。

用例

ブロックナンバーを 1 に指定し、検査を実行する場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim strResult As String

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
strResult = caoCtrl.Start( 1 )                    '指定実行
```

検査を実行する場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim strResult As String

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )

strResult = caoCtrl.Start                  '全実行または分岐実行
```

実装変数.Restart

機能

画像取り込みをせずに検査を実行します。実行モードが［全実行］または［分岐実行］の場合と、「指定実行」の場合では書式が異なります。

書式

実装変数.Restart(［<ブロックナンバー>］ **)**

引数：<ブロックナンバー>　実行するブロックナンバー（整数 0～9）

戻り値：画像処理結果（文字列）

説明

一括トリガで実行モードが［指定実行］の場合のみ引数のブロックナンバーが必要です。
［全実行］、「分岐実行」の場合、引数は不要です。

用例

ブロックナンバーを 1 に指定し、再検査を実行する場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim strResult As String

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
strResult = caoCtrl.Restart( 1 )                    '指定実行
```

再検査を実行する場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim strResult As String

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )

strResult = caoCtrl.Restart                         '全実行または分岐実行
```

実装変数.Xtype

機能

品種を切り替えます。

書式

実装変数.Xtype 〈品種ナンバー〉

引数：〈品種番ナンバー〉 (整数 0～255)

説明

品種を切り替えます。

用例

品種ナンバーを 100 に切り替える場合の例を以下に示します。

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.Xtype 100
```

実装変数.MemoryWrite

機能

設定データを PV シリーズ本体のメモリへ保存します。

書式

実装変数.MemoryWrite [<エリアナンバー>]

引数： <エリアナンバー> SD メモリカードの保存エリアナンバーを指定します。

PV200 なし

PV500 <エリアナンバー> (整数 0～99)

説明

設定データを PV シリーズ本体のメモリへ保存します。

用例

本体のメモリに設定データを保存する例を以下に示します。

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.MemoryWrite
```

実装変数.CFWrite

機能

設定データを SD メモリカードへ保存します。

書式

実装変数.CFWrite 〈エリアナンバー〉

引数:〈エリアナンバー〉SD メモリカードの保存エリアナンバーを指定します。
（整数 0～99）

説明

設定データを SD メモリカードへエリアナンバーを指定して保存します。

用例

SD メモリーカードの保存エリアナンバー10 に設定データを保存する場合の例を以下に示します。

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.CFWrite 10
```

実装変数.MemoryRead

機能　本体メモリから設定データを読み出します。

書式　**実装変数.MemoryRead**　[<エリアナンバー>]

引数：<エリアナンバー> SD メモリカードの読み出しエリアナンバーを指定します。

PV200	なし
PV500	<エリアナンバー>（整数 0～99）

説明　本体メモリからエリアナンバーを指　定して設定データを読み出します。

用例

本体のメモリから設定データを読み出す場合の例を以下に示します。

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.MemoryRead
```

実装変数.CFRead

機能　SD メモリカードから設定データを読み出します。

書式　**実装変数.CFRead** 〈エリアナンバー〉

引数：〈エリアナンバー〉SD メモリカードの読み出しエリアナンバーを指定します。（整数 0～99）

説明　SD メモリカードからエリアナンバーを指定して読み出します。

用例

エリアナンバー10 を指定し，設定データを読み出す場合の例を以下に示します。

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.CFRead 10
```

実装変数.CancelData

機能

設定データの保存・読み出しを中断します。

書式

実装変数.CancelData

説明

設定データの保存・読み出しを中断します。

用例

設定データの保存・読み出しを中断する場合の例を以下に示します。

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.CancelData
```

実装変数.SDSave

機能

本体に保存されている画像メモリを SD メモリカードに保存します。

書式

実装変数.SDSave

説明

本体に保存されている画像メモリを SD メモリカードに保存します。
ＳＤメモリカードの保存先は，空き番号を探して保存します。（保存先の番号は指定できません）

用例

SD メモリーカードへ画像メモリを保存する場合の例を以下に示します。

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.SDSave
```

実装変数.SDReset

機能

本体に保存されている画像メモリを消去します。

書式

実装変数.SDReset

説明

本体に保存されている画像メモリを消去します。
設定画面で［保存・読出］→［画像メモリ消去］を実行したときと同じ動作です。

用例

本体の画像メモリを消去する場合の例を以下に示します。

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.SDReset
```

実装変数.PrintScreen

機能

現在の画面表示（表示されているものすべて）をキャプチャし、SD メモリカード、または Ethernet インターフェイス経由でパソコンに保存します。

書式

実装変数.PrintScreen

説明

現在の画面表示をキャプチャし、SD メモリカード、または Ethernet インターフェイス経由でパソコンへ保存します。
保存先は［環境］→［入出力］→［プリントスクリーン］の［出力先］で指定した場所です。このコマンドでは出力先を指定できません。

用例

現在の表示画面を保存する場合の例を以下に示します。

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.PrintScreen
```

実装変数.Quit

機能

統計データと走査回数をクリアします。

書式

実装変数.Quit

説明

統計データと走査回数をクリアします。

用例

統計データ，走査回数をクリアする場合の例を以下に示します。

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.Quit
```

実装変数.RunManual

機能

PV シリーズの運転・停止を切り替えます。

書式

実装変数.RunManual(〈モード〉 **)**

引数：〈モード〉　運転・停止の切り替え（整数）。
0：運転へ切り替え。
1：停止へ切り替え。
戻り値：切り替えたモードの値（整数）。
0：運転。
1：停止。

説明

PV シリーズの運転・停止を切り替えます。

用例

PV シリーズを運転から停止へ切り替える場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim iResult As Integer

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
iResult = caoCtrl.RunManual( 1 )
```

実装変数.ErrorReset

機能

Error 信号をリセットします。

書式

実装変数.ErrorReset

説明

Error 信号をリセットします。

用例

エラーをクリアする場合の例を以下に示します。

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.ErrorReset
```

実装変数.Cancel

機能

実行途中の動作をキャンセルします。

書式

実装変数.Cancel

説明

実行途中の動作をキャンセルし、その動作の開始前の状態にします。

用例

実行途中をキャンセルする場合の例を以下に示します。

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.Cancel
```

実装変数.KeyEmulator

機能

キーパッドと同じ操作を実行します。

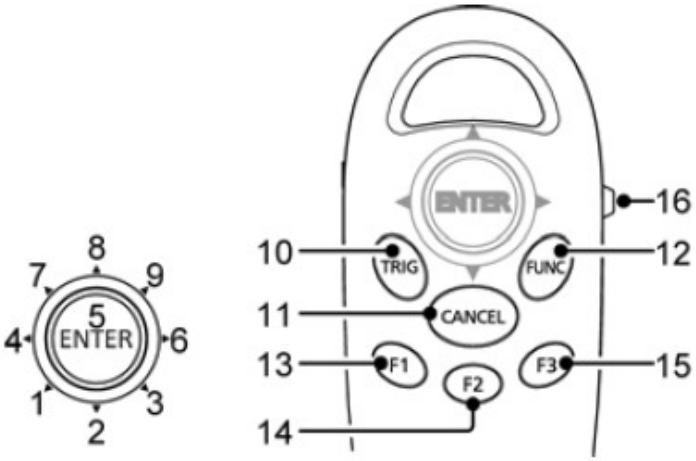
書式

実装変数.KeyEmulator 〈シフト〉, 〈キー〉

引数：〈シフト〉　シフトキーの ON・OFF（整数 0, 1）。
0：OFF。
1：ON。
〈キー〉　各種キーに割り当てられた値（整数 1～16）。
詳細は下図参照。

説明

キーパッドと同じ操作を実行します。
PV シリーズからのレスポンスはありません。



用例

キーパッドを操作し，運用画面/設定画面を切り替える場合の例を以下に示します。

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.KeyEmulator 0, 16
```

実装変数.Bstop

機能

運転画面において、キーパッドによる操作の受付を禁止・許可します。

書式

実装変数.Bstop 〈可否〉

引数：〈可否〉　キーパッド操作の受付けの可否（整数 0, 1）。
0：受付許可。
1：受付禁止。

説明

運転画面において、キーパッドによる操作の受付を禁止・許可します。

用例

キーパッド操作を受付拒否にする場合の例を以下に示します。

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.Bstop 1
```

実装変数.Bconfirm

機能

キーパッドによる操作の受付状態を取得します。

書式

実装変数.Bconfirm( )

戻り値： キーパッド操作受付状態（整数 0, 1）。
0：受付許可。
1：受付禁止。

説明

キーパッドによる操作の受付状態を取得します。

用例

キーパッドの操作の受付状態（受付許可）を取得する場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim iResult As Integer

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
iResult = caoCtrl.Bconfirm
```

実装変数.LayoutChange

機能　運転画面でモニタ表示するレイアウトを切り替えます。

書式　**実装変数.LayoutChange** 〈レイアウトナンバー〉

引数：〈レイアウトナンバー〉　整数で指定（0～15）。

説明　運転画面でモニタ表示するレイアウトを切り替えます。

用例

レイアウトを 1 に変更する場合の例を以下に表示します。

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.LayoutChange 1
```

実装変数.AgainTemplate

機能

スマートマッチングチェカーのテンプレートを再登録します。

書式

実装変数.AgainTemplate 〈チェッカーナンバー〉
，〈テンプレートナンバー〉

引数：〈チェッカーナンバー〉 整数で指定（0～999）。
〈テンプレートナンバー〉 整数で指定（0～63）。

説明

再登録できるのは［チェッカ］下のスマートマッチングです。位置補正、領域調整で使っているスマートマッチングは、テンプレートの再登録は出来ません。

用例

スマートマッチングチェッカのテンプレートを再登録する場合の例を以下に示します。

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.AgainTemplate 1, 10
```

実装変数.ParameterRead

機能

PV シリーズ本体の設定値やシステム値を読み出します。

書式

実装変数.ParameterRead(〈パラメータ〉 **)**

引数：〈パラメータ〉　文字列で指定。

戻り値：指定したパラメータ値。（文字列）

説明

PV シリーズ本体の設定値やシステム値を読み出します。運転中にのみ有効です。読み出し可能なデータ、各種コマンドパラメータについては、Panasonic 社の PV シリーズマニュアルを参照して下さい。

用例

現在時刻を読み出す場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim strResult As String

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
strResult = caoCtrl.ParameterRead( "SYS_TIME" )
```

実装変数.ParameterReadPair

機能

PV シリーズ本体の設定値やシステム値の 2 データを読み出します。

書式

実装変数.ParameterReadPair(〈パラメータ〉 **)**

引数：〈パラメータ〉　文字列で指定。

戻り値：指定したパラメータ値。(Variant 型)

説明

PV シリーズ本体の設定値やシステム値を 2 データ読み出します。上下限値データなどのセットデータを読み出します。運転中にのみ有効です。読み出し可能なデータ、各種コマンドパラメータについては、Panasonic 社の PV シリーズマニュアルを参照して下さい。

用例

カメラ 0 の 2 値化レベルグループ「A」の上下限値を読み出す場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
vntResult = caoCtrl.ParameterReadPair( "BLV:PAIRA" )
```

実装変数.ParameterWrite

機能

PV シリーズ本体の設定値やシステム値を変更します。

書式

実装変数.ParameterWrite 〈パラメータ〉, 〈データ〉

引数：〈パラメータ〉　文字列で指定。
　　〈データ〉　Variant 型で指定。

説明

PV シリーズ本体の設定値やシステム値を変更します。運転中にのみ有効です。
変更可能なデータ、各種コマンドパラメータについては、Panasonic 社の PV シリーズマニュアルを参照して下さい。

用例

汎用レジスタ 0 の値を「3.14」に変更する場合の例を以下に示します。

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.ParameterWrite "SYS:REG0", 3.14
```

実装変数.ParameterWritePair

機能

PV シリーズ本体の設定値やシステム値を 2 データ変更します。

書式

実装変数.ParameterWritePair 〈パラメータ〉
, 〈データ 1〉
, 〈データ 2〉

引数：〈パラメータ〉　文字列で指定。
〈データ 1〉　Variant 型で指定。
〈データ 2〉　Variant 型で指定。

説明

PV シリーズ本体の設定値やシステム値を 2 データ変更します。運転中にのみ有効です。変更可能なデータ、各種コマンドパラメータについては、Panasonic 社の PV シリーズマニュアルを参照して下さい。

用例

数値演算 No.10 の上下限値を上限値「100」，下限値「50」に変更する場合の例を以下に示します。

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.ParameterWritePair "BLV:PAIRA", 50, 100
```

実装変数.StartAsync

機能

非同期で検査を実行します。実行モードが「全実行」または「分岐実行」の場合と、「指定実行」の場合で書式が異なります。コマンドの戻り値は GetResult コマンドで取得し、確認してください。取得するデータは文字列型です。

書式

実装変数. StartAsync 〈ブロックナンバー〉

引数：〈ブロックナンバー〉 実行するブロックナンバー 整数で指定（0～9）

説明

一括トリガで実行モードが［指定実行］の場合のみ引数のブロックナンバーが必要です。
［全実行］、「分岐実行」の場合、引数は不要です。

用例

非同期でブロックナンバーを 1 に指定し，検査を実行する場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = cao.AddController("PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201")
caoCtrl.StartAsync 1                    '指定実行

'StartAsync コマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

非同期で検査を実行する場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = cao.AddController("PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201")
caoCtrl.StartAsync                      '全実行または分岐実行

'StartAsync コマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.ReStartAsync

機能

非同期で画像撮り込みをせずに検査を実行します（再検査）. 実行モードが「全実行」または「分岐実行」の場合と、「指定実行」の場合で書式が異なります。コマンドの戻り値は GetResult コマンドで取得し、確認してください。取得するデータは文字列型です。

書式

実装変数. ReStartAsync 〈ブロックナンバー〉

引数：〈ブロックナンバー〉 実行するブロックナンバー 整数で指定（0～9）

説明

一括トリガで実行モードが［指定実行］の場合のみ引数のブロックナンバーが必要です。
［全実行］、「分岐実行」の場合、引数は不要です。

用例

非同期でブロックナンバーを 1 に指定し，検査を実行する場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = cao.AddController("PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201")
caoCtrl.ReStartAsync 1                  '指定実行

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

非同期で検査を実行する場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = cao.AddController("PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201")
caoCtrl.ReStartAsync                    '全実行または分岐実行

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

実装変数.XTypeAsync

機能

非同期で品種を切り替えます。コマンドの戻り値は GetResult コマンドで取得し、確認してください。

書式

実装変数. XTypeAsync 〈品種ナンバー〉

引数：〈品種番ナンバー〉 整数で指定（0～255）

説明

非同期で品種を切り替えます。コマンドの戻り値は GetResult コマンドで取得し、確認してください。

用例

非同期で品種ナンバーを 100 に切り替える場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = cao.AddController("PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201")
caoCtrl.XtypeAsync 100

'XtypeAsync コマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.MemoryWriteAsync

機能

非同期で設定データを PV シリーズ本体のメモリへ保存します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。

書式

実装変数. MemoryWriteAsync [<エリアナンバー>]

引数:<エリアナンバー> SD メモリカードの保存エリアナンバーを指定します。
PV200 なし
PV500 <エリアナンバー> (整数 0～99)

説明

非同期で設定データを本体メモリにエリアナンバーを指定して保存します。
コマンドの戻り値は GetResult コマンドで取得し、確認してください。

用例

非同期で本体のメモリに設定データを保存する例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = cao.AddController("PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201")
caoCtrl.MemoryWriteAsync

'MemoryWriteAsync コマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.CFWriteAsync

機能

非同期で設定データを SD メモリーカードへ保存します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。

書式

実装変数. CFWriteAsync 〈エリアナンバー〉

引数:〈エリアナンバー〉SD メモリカードの保存エリアナンバーを指定します。
整数で指定（0～99）

説明

非同期で設定データを SD メモリカードへエリアナンバーを指定して保存します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。

用例

非同期で SD メモリーカードの保存エリアナンバー10 に設定データを保存する場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = cao.AddController("PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201")
caoCtrl.CFWriteAsync 10

'CFWriteAsync コマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.MemoryReadAsync

機能

非同期で PV シリーズ本体のメモリから設定データを読み出します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。

書式

実装変数. MemoryReadAsync [<エリアナンバー>]

引数:<エリアナンバー> SD メモリカードの保存エリアナンバーを指定します。
PV200　なし
PV500　<エリアナンバー>（整数 0～99）

説明

非同期で PV シリーズ本体のメモリから設定データを読み出します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。

用例

非同期で本体のメモリから設定データを読み出す場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.MemoryReadAsync

'MemoryReadAsync コマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.CFReadAsync

機能

非同期で SD メモリカードから設定データを読み出します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。

書式

実装変数. CFReadAsync 〈エリアナンバー〉

引数：〈エリアナンバー〉SD メモリカードの読み出しエリアナンバーを指定します。　整数で指定（0～99）

説明

非同期で SD メモリカードから設定データを読み出します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。

用例

非同期でエリアナンバー10 を指定し，設定データを読み出す場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.CFReadAsync 10

'CFReadAsync コマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.CancelDataAsync

機能

非同期で設定データの保存・読み出しを中断します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。

書式

実装変数. CancelDataAsync

説明

非同期で設定データの保存・読み出しを中断します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。

用例

非同期で設定データの保存・読み出しを中断する場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.CancelDataAsync

'CancelDataAsync コマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.SDSaveAsync

機能

非同期で本体に保存されている画像メモリを SD メモリカードに保存します。
コマンドの戻り値は GetResult コマンドで取得し、確認してください。

書式

実装変数.SDSaveAsync

説明

非同期で本体に保存されている画像メモリを SD メモリカードに保存します。
コマンドの戻り値は GetResult コマンドで取得し、確認してください。

用例

非同期で SD メモリーカードへ画像メモリを保存する場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.SDSaveAsync

'SDSaveAsyncコマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.SDResetAsync

機能

非同期で本体に保存されている画像メモリを消去します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。

書式

実装変数.SDResetAsync

説明

非同期で本体に保存されている画像メモリを消去します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。

用例

非同期で本体の画像メモリを消去する場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.SDResetAsync

'SDResetAsync コマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.PrintScreenAsync

機能

非同期で現在の画面表示（表示されているものすべて）をキャプチャし、SD メモリカード、または Ethernet インターフェイス経由でパソコンに保存します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。

書式

実装変数. PrintScreenAsync

説明

非同期で現在の画面表示（表示されているものすべて）をキャプチャし、SD メモリカード、または Ethernet インターフェイス経由でパソコンに保存します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。

用例

非同期で現在の表示画面を保存する場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.PrintScreenAsync

'PrintScreenAsync コマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.QuitAsync

機能

非同期で統計データと走査回数をクリアします。
コマンドの戻り値は GetResult コマンドで取得し、確認してください。

書式

実装変数.QuitAsync

説明

非同期で統計データと走査回数をクリアします。
コマンドの戻り値は GetResult コマンドで取得し、確認してください。

用例

非同期で統計データ，走査回数をクリアする場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.QuitAsync

'QuitAsync コマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.RunManualAsync

機能

非同期で PV シリーズの運転・停止を切り替えます。コマンドの戻り値は GetResult コマンドで取得し、確認してください。取得するデータは整数型です。

書式

実装変数.RunManualAsync 〈モード〉

引数：〈モード〉 運転・停止の切り替え 整数で指定（0, 1）
0：運転へ切り替え。
1：停止へ切り替え。

説明

非同期で PV シリーズの運転・停止を切り替えます。コマンドの戻り値は GetResult コマンドで取得し、確認してください。取得するデータは整数型です。

用例

非同期で PV シリーズを運転から停止へ切り替える場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.RunManualAsync 1

'RunManualAsync コマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.ErrorResetAsync

機能

非同期で Error 信号をリセットします。コマンドの戻り値は GetResult コマンドで取得し、確認してください。

書式

実装変数.ErrorResetAsync

説明

非同期で Error 信号をリセットします。コマンドの戻り値は GetResult コマンドで取得し、確認してください。

用例

非同期でエラーをクリアする場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.ErrorResetAsync

'ErrorResetAsync コマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.CancelAsync

機能

非同期で実行途中の動作をキャンセルし、その動作の開始前の状態にします。
コマンドの戻り値は GetResult コマンドで取得し、確認してください。

書式

実装変数.CancelAsync

説明

非同期で実行途中の動作をキャンセルし、その動作の開始前の状態にします。
コマンドの戻り値は GetResult コマンドで取得し、確認してください。

用例

非同期で実行途中をキャンセルする場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.CancelAsync

'CancelAsync コマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.KeyEmulatorAsync

機能

非同期でキーパッドと同じ操作を実行します。
PV シリーズからのレスポンスはありません。
コマンドの戻り値は GetResult コマンドで取得し、確認してください。

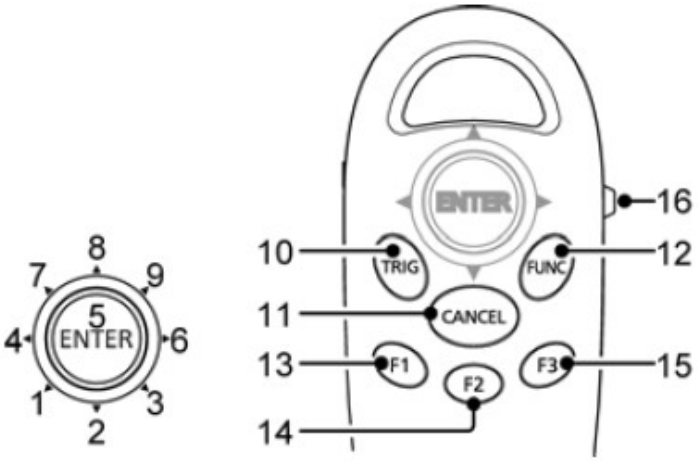
書式

実装変数.KeyEmulatorAsync 〈シフト〉, 〈キー〉

引数：〈シフト〉　シフトキーの ON・OFF　整数で指定（0, 1）。
0：OFF。
1：ON。
〈キー〉　各種キーに割り当てられた値　整数で指定（1～16）。
詳細は下図参照。

説明

非同期でキーパッドと同じ操作を実行します。
PV シリーズからのレスポンスはありません。
コマンドの戻り値は GetResult コマンドで取得し、確認してください。



用例

非同期でキーパッドを操作し，運用画面/設定画面を切り替える場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.KeyEmulatorAsync 0, 16

'KeyEmulatorAsync コマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.BstopAsync

機能

非同期で運転画面において、キーパッドによる操作の受け付けを禁止・許可します。
コマンドの戻り値は GetResult コマンドで取得し、確認してください。

書式

実装変数.BstopAsync 〈可否〉

引数：〈可否〉　キーパッド操作の受付けの可否　整数で指定（0, 1）。
0：受付許可。
1：受付禁止。

説明

非同期で運転画面において、キーパッドによる操作の受け付けを禁止・許可します。
コマンドの戻り値は GetResult コマンドで取得し、確認してください。

用例

非同期でキーパッド操作を受付拒否にする場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.BstopAsync 1

'BstopAsyncコマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.BconfirmAsync

機能

非同期でキーパッドによる操作の受付状態を取得します。
コマンドの戻り値は GetResult コマンドで取得し、確認してください。

書式

実装変数.BconfirmAsync

説明

非同期でキーパッドによる操作の受付状態を取得します。
コマンドの戻り値は GetResult コマンドで取得し、確認してください。

用例

非同期でキーパッドの操作の受付状態（受付許可）を取得する場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.BconfirmAsync

'BconfirmAsync コマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.LayoutChangeAsync

機能

非同期で運転画面において、モニタに表示するレイアウトを外部機器からの信号によって切り替えるときに使用します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。

書式

実装変数.LayoutChangeAsync 〈レイアウトナンバー〉

引数：〈レイアウトナンバー〉 整数で指定（0～15）。

説明

非同期で運転画面において、モニタに表示するレイアウトを外部機器からの信号によって切り替えるときに使用します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。

用例

非同期でレイアウトを 1 に変更する場合の例を以下に表示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.LayOutChangeAsync 1

'LayoutChangeAsync コマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.AgainTemplateAsync

機能

非同期でスマートマッチングチェッカのテンプレートを再登録します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。

書式

実装変数.AgainTemplateAsync 〈チェッカーナンバー〉
，〈テンプレートナンバー〉

引数：〈チェッカーナンバー〉 整数で指定（0～999）。
〈テンプレートナンバー〉 整数で指定（0～63）。

説明

再登録できるのは［チェッカ］下のスマートマッチングです。位置補正、領域調整で使っているスマートマッチングは、テンプレートの再登録は出来ません。

用例

非同期でスマートマッチングチェッカのテンプレートを再登録する場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.AgainTemplateAsync 1, 10

'AgainTemplateAsync コマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.ParameterReadAsync

機能

非同期で PV シリーズ本体の設定値やシステム値を読み出します。
読み出し可能なデータ、各種コマンドパラメータについては、Panasonic 社の PV シリ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 GetResult コマンドで取得し、確認してください。取得するデータは文字列です。

書式

実装変数.ParameterReadAsync 〈パラメータ〉

引数：〈パラメータ〉　文字列で指定。

説明

非同期で PV シリーズ本体の設定値やシステム値を読み出します。
読み出し可能なデータ、各種コマンドパラメータについては、Panasonic 社の PV シリ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 GetResult コマンドで取得し、確認してください。取得するデータは文字列です。

用例

非同期で現在時刻を読み出す場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.ParameterReadAsync "SYS_TIME"

'ParameterReadAsync コマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.ParameterReadPairAsync

機能

非同期で PV シリーズ本体の 2 データを読み出します。
読み出し可能なデータ、各種コマンドパラメータについては、Panasonic 社の PV シリ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 GetResult コマンドで取得し、確認してください。取得するデータは Variant 型です。

書式

実装変数.ParameterReadPairAsync 〈パラメータ〉

引数：〈パラメータ〉 文字列で指定。

説明

非同期で PV シリーズ本体の 2 データを読み出します。
読み出し可能なデータ、各種コマンドパラメータについては、Panasonic 社の PV シリ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 GetResult コマンドで取得し、確認してください。取得するデータは Variant 型です。

用例

非同期でカメラ 0 の 2 値化レベルグループ「A」の上下限値を読み出す場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.ParameterReadPairAsync "BLV:PAIRA"

'ParameterReadPairAsync コマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.ParameterWriteAsync

機能

非同期で PV シリーズ本体の設定値やシステム値を変更します。
変更可能なデータ、各種コマンドパラメータについては、Panasonic 社の PV シリ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 GetResult コマンドで取得し、確認してください。

書式

実装変数.ParameterWriteAsync 〈パラメータ〉, 〈データ〉

引数：〈パラメータ〉　文字列で指定。
　　〈データ〉　Variant 型で指定。

説明

非同期で PV シリーズ本体の設定値やシステム値を変更します。
変更可能なデータ、各種コマンドパラメータについては、Panasonic 社の PV シリ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 GetResult コマンドで取得し、確認してください。

用例

非同期で汎用レジスタ 0 の値を「3.14」に変更する場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.ParameterWriteAsync "SYS:REG0", 3.14

'ParameterWriteAsync コマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.ParameterWritePairAsync

機能

非同期で PV シリーズ本体の 2 データの値を変更します。
読み出し可能なデータ、各種コマンドパラメータについては、Panasonic 社の PV シリ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 GetResult コマンドで取得し、確認してください。

書式

実装変数.ParameterWritePairAsync 〈パラメータ〉
, 〈データ 1〉
, 〈データ 2〉

引数：〈パラメータ〉 文字列で指定。
〈データ 1〉 Variant 型で指定。
〈データ 2〉 Variant 型で指定。

説明

非同期で PV シリーズ本体の 2 データの値を変更します。
読み出し可能なデータ、各種コマンドパラメータについては、Panasonic 社の PV シリ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 GetResult コマンドで取得し、確認してください。

用例

非同期で数値演算 No.10 の上下限値を上限値「100」，下限値「50」に変更する場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.ParameterWritePairAsync "CAC010:LPAIR", 50, 100

'ParameterWritePairAsync コマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.Raw

機能

コマンドメッセージを送受信します。

書式

実装変数.Raw(〈送信コマンドメッセージ〉 **)**

引数：〈送信コマンドメッセージ〉　文字列で指定。

戻り値：受信したコマンドメッセージ。（文字列）

説明

PV シリーズのコマンドを直接送受信します。BCC（ブロックチェックコード）に付いては内部で自動計算します。
コマンドについては Panasonic 社の PV シリーズマニュアルを参照して下さい。

用例

一括トリガで，実行モードが「全実行」または「分岐実行」で検査を実行する場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim strResult As String

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
strResult = caoCtrl.Raw( "%S" )
```

実装変数.SetTimeout

機能

通信のタイムアウト時間を設定します。通常は AddController で設定された値になっています。

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〉 整数型で指定。

説明

通信のタイムアウト時間を設定します。通常は AddController で設定された値になっています。

用例

タイムアウト時間を 1 秒（1000msec）に設定する例を以下に示します。

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.SetTimeout 1000
```

実装変数.GetTimeout

機能

通信のタイムアウト時間を取得します。通常は AddController で設定された値になっています。

書式

実装変数. GetTimeout ( )

戻り値： タイムアウト時間。（整数型）

説明

通信のタイムアウト時間を取得します。通常は AddController で設定された値になっています。

用例

タイムアウト時間（1000msec）を取得する例を以下に示します。

```
Dim caoCtrl As Object
Dim iResult As Integer

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
iResult = caoCtrl.GetTimeout
```

実装変数.RawAsync

機能

非同期でコマンドメッセージを送信します。BCC については、内部で自動計算します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。取得するデータは文字列型です。

書式

実装変数. RawAsync 〈送信コマンドメッセージ〉

引数：〈送信コマンドメッセージ〉 文字列で指定。

説明

非同期でコマンドメッセージを送信します。BCC については、内部で自動計算します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。取得するデータは文字列型です。

用例

非同期で一括トリガで，実行モードが「全実行」または「分岐実行」で検査を実行する場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201" )
caoCtrl.RawAsync "%S"

'RawAsync コマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.GetResult

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。
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。
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、GetResult 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。
タイムアウトエラーが発生する場合は、SetTimeout コマンドまたは、AddController のオプションでタイムアウト時間を延ばしてください。

書式

実装変数. GetResult( )

戻り値：非同期コマンドの戻り値。(Variant 型)
戻り値は実行したコマンドに依存します

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。
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、GetResult 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。
タイムアウトエラーが発生する場合は、SetTimeout コマンドまたは、AddController のオプションでタイムアウト時間を延ばしてください。

用例

非同期で検査を実行した場合の，戻り値を取得する例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", "Conn=eth:192.168.0.201" )
caoCtrl.StartAsync

vntResult = caoCtrl.GetResult
```

実装変数.SetPoint

機能

PV にロボット座標を通知します。

書式

実装変数. SetPoint 〈ロボット座標（X）〉,〈ロボット座標（Y）〉
,〈ロボット座標（Z）〉,〈ロボット座標（Rx）〉
,〈ロボット座標（Ry）〉,〈ロボット座標（Rz）〉
,〈ロボット座標（Fig）〉

引数：〈ロボット座標（X）〉 倍精度実数型で指定。
〈ロボット座標（Y）〉 倍精度実数型で指定。
〈ロボット座標（Z）〉 倍精度実数型で指定。
〈ロボット座標（Rx）〉 倍精度実数型で指定。
〈ロボット座標（Ry）〉 倍精度実数型で指定。
〈ロボット座標（Rz）〉 倍精度実数型で指定。
〈ロボット座標（Fig）〉 整数型で指定。

説明

PV にロボット座標を通知します。

用例

ロボットの現在位置を PV に通知する例を以下に示します。

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", _
                                    "PV260=1, Conn=eth:192.168.0.201" )
caoCtrl.SetPoint POSX(CURPOS), POSY(CURPOS), POSZ(CURPOS), _
                    POSRX(CURPOS),POSRY(CURPOS), POSRZ(CURPOS), FIG(CURPOS)
```

実装変数.Calibrate

機能

計測を実行します。実行モードが「全実行」の場合と、「指定実行」の場合で書式が異なります。

書式

実装変数. Calibrate (〈キャリブレーションナンバー〉 ,
［〈ブロックナンバー〉］ **)**

引数：〈キャリブレーションナンバー〉 整数で指定（0～5）
〈ブロックナンバー〉整数で指定（0～9）

戻り値：ロボット座標（X, Y, Rz, Fig）の配列。(Variant 型)

説明

実行モードが［指定実行］の場合のみ引数のブロックナンバーが必要です。
［全実行］の場合、ブロックナンバーは不要です。

用例

キャリブレーションナンバー0 に対し，計測を実行する場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", _
                             "PV260=1, Conn=eth:192.168.0.201" )

vntResult = caoCtrl.Calibrate( 0 )          '全実行
```

キャリブレーションナンバー0，ブロックナンバー1 を指定し，計測を実行する場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", _
                             "PV260=1, Conn=eth:192.168.0.201" )

vntResult = caoCtrl.Calibrate( 0, 1 )          '指定実行
```

実装変数.ReCalibrate

機能

画像撮り込みをせずに計測を実行します（再計測）。実行モードが「全実行」の場合と、「指定実行」の場合で書式が異なります。

書式

実装変数. ReCalibrate(〈キャリブレーションナンバー〉,
［〈ブロックナンバー〉］**)**

引数：〈キャリブレーションナンバー〉 整数で指定（0～5）
〈ブロックナンバー〉整数で指定（0～9）

戻り値：ロボット座標（X, Y, Rz, Fig）の配列。(Variant 型)

説明

実行モードが［指定実行］の場合のみ引数のブロックナンバーが必要です。
［全実行］の場合、ブロックナンバーは不要です。

用例

キャリブレーションナンバー0 に対し，再計測を実行する場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", _
                             "PV260=1, Conn=eth:192.168.0.201" )

vntResult = caoCtrl.ReCalibrate( 0 )        '全実行
```

キャリブレーションナンバー0，ブロックナンバー1 を指定し，再計測を実行する場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", _
                             "PV260=1, Conn=eth:192.168.0.201" )

vntResult = caoCtrl.ReCalibrate( 0, 1 )        '指定実行
```

実装変数.CalibrationStart

機能　自動キャリブレーションを開始します。

書式　**実装変数. CalibrationStart**　〈キャリブレーションナンバー〉

引数：〈キャリブレーションナンバー〉　整数で指定（0～5）

説明　自動キャリブレーションを開始します。

用例

キャリブレーションナンバー0 に対し，自動キャリブレーションを開始する場合の例を以下に示します。

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", _
                                    "PV260=1, Conn=eth:192.168.0.201" )
caoCtrl.CalibrationStart 0
```

実装変数.CalibrationEnd

機能

自動キャリブレーション完了のレスポンスを取得します。

書式

実装変数. CalibrationEnd

説明

自動キャリブレーション完了のレスポンスを取得します。

用例

自動キャリブレーション完了のレスポンスを取得する例を以下に示します。

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", _
                                    "PV260=1, Conn=eth:192.168.0.201" )
caoCtrl.CalibrationEnd
```

実装変数.WorkSet

機能

ワーク検出基準位置の再登録を行います（撮像なし）。基準位置を登録後、キャリブレーションの設定を変更した場合は、再度基準位置を登録する必要があります。本コマンドを実行することにより、基準位置を自動的に再計算します。

書式

実装変数. WorkSet

説明

ワーク検出基準位置の再登録を行います（撮像なし）。基準位置を登録後、キャリブレーションの設定を変更した場合は、再度基準位置を登録する必要があります。本コマンドを実行することにより、基準位置を自動的に再計算します。

用例

ワーク検出基準位置の再登録を行う場合の例を以下に示します。

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", _
                                    "PV260=1, Conn=eth:192.168.0.201" )
caoCtrl.WorkSet
```

実装変数.WorkReset

機能

ワーク検出基準位置の再登録を行います（撮像あり）。
基準位置を登録後、基準ワークを変更した場合は、再度基準位置を登録する必要があります。
再登録の際に、基準位置登録時と設定（視野数、マーク数、基準登録時のロボット位置情報）が変わらない場合、本コマンドを実行することにより、基準位置を自動的に再計算します。

書式

実装変数. WorkReset 〈ワーク検出ナンバー〉

引数：〈ワーク検出ナンバー〉 整数で指定（0～15）

説明

ワーク検出基準位置の再登録を行います（撮像あり）。
基準位置を登録後、基準ワークを変更した場合は、再度基準位置を登録する必要があります。
再登録の際に、基準位置登録時と設定（視野数、マーク数、基準登録時のロボット位置情報）が変わらない場合、本コマンドを実行することにより、基準位置を自動的に再計算します。

用例

ワーク検出基準位置の再登録を開始する場合の例を以下に示します。

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", _
                                   "PV260=1, Conn=eth:192.168.0.201" )
caoCtrl.WorkReset 0
```

実装変数.WorkResetEnd

機能

ワーク検出基準位置の再登録完了のレスポンスを取得します。

書式

実装変数. WorkResetEnd

説明

ワーク検出基準位置の再登録完了のレスポンスを取得します。

用例

ワーク検出基準位置の再登録完了のレスポンスを取得する例を以下に示します。

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", _
                                    "PV260=1, Conn=eth:192.168.0.201" )
caoCtrl.WorkResetEnd
```

実装変数.MoveEnd

機能

ロボットの移動が完了したことを PV に通知します。

書式

実装変数. MoveEnd

説明

ロボットの移動が完了したことを PV に通知します。

用例

ロボットの移動が完了したことを PV に通知する場合の例を以下に示します。

```
Dim caoCtrl As Object

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", _
                                    "PV260=1, Conn=eth:192.168.0.201" )
caoCtrl.MoveEnd
```

実装変数.GetTeachPoint

機能

PV で設定してある、ティーチング座標を全て取得します。

書式

実装変数. GetTeachPoint( )

戻り値：ロボット座標移動（X, Y, Rz, Fig）の配列。(Variant 型)

説明

PV で設定してある、ティーチング座標を全て取得します。

用例

ティーチング座標を要求する場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", _
                                    "PV260=1, Conn=eth:192.168.0.201" )
vntResult = caoCtrl.GetTeachPoint
```

実装変数.GetMovePoint

機能

自動キャリブレーション（CalibrationStart）、ワーク検出基準位置の再登録（WorkReset）中に PV から通知されたロボット移動座標を取得します。

書式

実装変数. GetMovePoint( )

戻り値：ロボット移動座標（X, Y, Rz, Fig）の配列。(Variant 型)

説明

自動キャリブレーション（CalibrationStart）、ワーク検出基準位置の再登録（WorkReset）中に PV から通知されたロボット移動座標を取得します。

用例

PV から通知された，ロボット移動座標を取得する例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResultPos As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", _
                                   "PV260=1, Conn=eth:192.168.0.201" )
vntResultPos = caoCtrl.GetMovePoint
```

実装変数.SetPoint Async

機能

非同期で PV にロボット座標を通知します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。

書式

実装変数. SetPoint Async <ロボット座標（X）>,<ロボット座標（Y）>
,<ロボット座標（Z）>,<ロボット座標（Rx）>
,<ロボット座標（Ry）>,<ロボット座標（Rz）>
,<ロボット座標（Fig）>

引数：<ロボット座標（X）> 倍精度実数型で指定。
<ロボット座標（Y）> 倍精度実数型で指定。
<ロボット座標（Z）> 倍精度実数型で指定。
<ロボット座標（Rx）> 倍精度実数型で指定。
<ロボット座標（Ry）> 倍精度実数型で指定。
<ロボット座標（Rz）> 倍精度実数型で指定。
<ロボット座標（Fig）> 整数型で指定。

説明

非同期で PV にロボット座標を通知します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。

用例

非同期でロボットの現在位置を PV に通知する例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", _
                                    "PV260=1, Conn=eth:192.168.0.201" )
caoCtrl.SetPointAsync PosX( CurPos ), PosY( CurPos ), PosZ( CurPos ), PosRx( CurPos ), _
                            PosRy( CurPos ), PosRz( CurPos ), Fig( CurPos )

'SetPointAsyncコマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.CalibrateAsync

機能

非同期で計測を実行します。実行モードが「全実行」の場合と、「指定実行」の場合で書式が異なります。コマンドの戻り値は GetResult コマンドで取得し、確認してください。取得するデータは Variant 型です。

書式

実装変数. CalibrateAsync 〈キャリブレーションナンバー〉
,〈ブロックナンバー〉

引数：〈キャリブレーションナンバー〉 整数で指定（0～5）
〈ブロックナンバー〉整数で指定（0～9）

説明

実行モードが［指定実行］の場合のみ引数のブロックナンバーが必要です。
［全実行］の場合、ブロックナンバーは不要です。

用例

非同期でキャリブレーションナンバー0 に対し，計測を実行する場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", _
                             "PV260=1, Conn=eth:192.168.0.201" )
caoCtrl.CalibrateAsync 0                    '全実行

'CalibrateAsync コマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

非同期でキャリブレーションナンバー0，ブロックナンバー1 を指定し，計測を実行する場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", _
                             "PV260=1, Conn=eth:192.168.0.201" )

caoCtrl.CalibrateAsync 0, 1                 '指定実行

'CalibrateAsync コマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.ReCalibrateAsync

機能

非同期で画像撮り込みをせずに計測を実行します（再計測）。実行モードが「全実行」の場合と、「指定実行」の場合で書式が異なります。コマンドの戻り値は GetResult コマンドで取得し、確認してください。取得するデータは Variant 型です。

書式

実装変数. ReCalibrateAsync 〈キャリブレーションナンバー〉
,〈ブロックナンバー〉

引数：〈キャリブレーションナンバー〉 整数で指定（0～5）
〈ブロックナンバー〉整数で指定（0～9）

説明

実行モードが［指定実行］の場合のみ引数のブロックナンバーが必要です。
［全実行］の場合、ブロックナンバーは不要です。

用例

非同期でキャリブレーションナンバー0 に対し，計測を実行する場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", _
                             "PV260=1, Conn=eth:192.168.0.201" )
caoCtrl.ReCalibrateAsync 0                    '全実行

'ReCalibrateAsync コマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

非同期でキャリブレーションナンバー0，ブロックナンバー1 を指定し，再計測を実行する場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", _
                             "PV260=1, Conn=eth:192.168.0.201" )

caoCtrl.ReCalibrateAsync 0, 1                 '指定実行

'ReCalibrateAsync コマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.CalibrationStartAsync

機能

非同期で自動キャリブレーションを開始します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。

書式

実装変数. CalibrationStartAsync 〈キャリブレーションナンバー〉

引数：〈キャリブレーションナンバー〉 整数で指定（0～5）

説明

非同期で自動キャリブレーションを開始します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。

用例

キャリブレーションナンバー0 に対し，自動キャリブレーションを開始する場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", _
                                    "PV260=1, Conn=eth:192.168.0.201" )
caoCtrl.CalibrationStartAsync 0

'CalibrationStartAsync コマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.CalibrationEndAsync

機能

非同期で自動キャリブレーション完了のレスポンスを取得します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。

書式

実装変数. CalibrationEndAsync

説明

非同期で自動キャリブレーション完了のレスポンスを取得します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。

用例

非同期で自動キャリブレーション完了のレスポンスを取得する例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", _
                                   "PV260=1, Conn=eth:192.168.0.201" )
caoCtrl.CalibrationEndAsync

'CalibrationEndAsyncコマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.WorkSetAsync

機能

非同期でワーク検出基準位置の再登録を行います（撮像なし）。基準位置を登録後、キャリブレーションの設定を変更した場合は、再度基準位置を登録する必要があります。本コマンドを実行することにより、基準位置を自動的に再計算します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。

書式

実装変数. WorkSet Async

説明

非同期でワーク検出基準位置の再登録を行います（撮像なし）。基準位置を登録後、キャリブレーションの設定を変更した場合は、再度基準位置を登録する必要があります。本コマンドを実行することにより、基準位置を自動的に再計算します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。

用例

非同期でワーク検出基準位置の再登録を行う場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", _
                                   "PV260=1, Conn=eth:192.168.0.201" )
caoCtrl.WorkSetAsync

'WorkSetAsyncコマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.WorkResetAsync

機能

非同期でワーク検出基準位置の再登録を行います（撮像あり）。
基準位置を登録後、基準ワークを変更した場合は、再度基準位置を登録する必要があります。
再登録の際に、基準位置登録時と設定（視野数、マーク数、基準登録時のロボット位置情報）が変わらない場合、本コマンドを実行することにより、基準位置を自動的に再計算します。
コマンドの戻り値は GetResult コマンドで取得し、確認してください。

書式

実装変数. WorkResetAsync 〈ワーク検出ナンバー〉

引数：〈ワーク検出ナンバー〉 整数で指定（0～15）

説明

非同期でワーク検出基準位置の再登録を行います（撮像あり）。
基準位置を登録後、基準ワークを変更した場合は、再度基準位置を登録する必要があります。
再登録の際に、基準位置登録時と設定（視野数、マーク数、基準登録時のロボット位置情報）が変わらない場合、本コマンドを実行することにより、基準位置を自動的に再計算します。
コマンドの戻り値は GetResult コマンドで取得し、確認してください。

用例

非同期でワーク検出基準位置の再登録を開始する場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", _
                                   "PV260=1, Conn=eth:192.168.0.201" )
caoCtrl.WorkResetAsync 0

'WorkResetAsyncコマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.WorkResetEndAsync

機能

非同期でワーク検出基準位置の再登録完了のレスポンスを取得します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。

書式

実装変数. WorkResetEndAsync

説明

非同期でワーク検出基準位置の再登録完了のレスポンスを取得します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。

用例

非同期でワーク検出基準位置の再登録完了のレスポンスを取得する例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", _
                                    "PV260=1, Conn=eth:192.168.0.201" )
caoCtrl.WorkResetEndAsync

'WorkResetEndAsyncコマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.MoveEndAsync

機能

非同期でロボットの移動が完了したことを PV に通知します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。

書式

実装変数. MoveEndAsync

説明

非同期でロボットの移動が完了したことを PV に通知します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。

用例

非同期でロボットの移動が完了したことを PV に通知する場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", _
                                    "PV260=1, Conn=eth:192.168.0.201" )
caoCtrl.MoveEndAsync

'MoveEndAsyncコマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.GetTeachPointAsync

機能

非同期で PV に設定してある、ティーチング座標を全て取得します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。取得するデータは Variant 型です。

書式

実装変数. GetTeachPointAsync

説明

非同期で PV に設定してある、ティーチング座標を全て取得します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。取得するデータは Variant 型です。

用例

非同期でティーチング座標を要求する場合の例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", _
                                   "PV260=1, Conn=eth:192.168.0.201" )
caoCtrl.GetTeachPointAsync

'GetTeachPointAsyncコマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

実装変数.GetMovePointAsync

機能

非同期で自動キャリブレーション（CalibrationStart）、ワーク検出基準位置の再登録（WorkReset）中に PV から通知されたロボット移動座標を取得します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。取得するデータは Variant 型です。

書式

実装変数. GetMovePointAsync

説明

非同期で自動キャリブレーション（CalibrationStart）、ワーク検出基準位置の再登録（WorkReset）中に PV から通知されたロボット移動座標を取得します。コマンドの戻り値は GetResult コマンドで取得し、確認してください。取得するデータは Variant 型です。

用例

非同期で PV から通知された，ロボット移動座標を取得する例を以下に示します。

```
Dim caoCtrl As Object
Dim vntResult As Variant

caoCtrl = Cao.AddController( "PV", "CaoProv.Panasonic.PV", "", _
                                    "PV260=1, Conn=eth:192.168.0.201" )
caoCtrl.GetMovePointAsync

'GetMovePointAsync コマンドの戻り値取得
vntResult = caoCtrl.GetResult
```

6. エラーコード

PVプロバイダの独自のエラーコードは、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、パナソニックデバイスSUNX社のPVシリーズリファレンスマニュアルの取説を参照してください。

例:Start を実行したとき。

0x801000C8 ： 運転停止(STOP)中のため、実行不可。

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_COMMAND_EXECUTING	0x80F00000	非同期コマンド実行中に，別のコマンドを実行しました。
E_COMMAND_CONNECTED (Ver.1.12.*)	0x80F00001	接続していない通信ポートに対して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PV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. 「全実行」または「分岐実行」へ切り替えます。
2. 「指定実行」へ切り替えます。
3. 変更する品種を設定します。範囲：0～255
4. 3. で設定した品種に切り替えます。(Xtype)
5. 「指定実行」時のブロック No.を設定します。範囲：0～9
6. 3. 5. で設定した番号に従い検査を実行します。受信したデータは 9. のデータ表示部に表示します。(Start)
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. 受信データを表示します。
10. 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、7. に"接続完了"と表示されます。

注２：PacScript プログラムにて PV 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 caoPV as Object                         'プロバイダ用変数宣言
    Dim strResult as String                     '文字列変数宣言
    Dim pTargetPos as Position                  'P 変数型変数宣言

    takearm keep = 0

    pTargetPos = P11

    caoPV = cao.AddController("PV", "CaoProv.Panasonic.PV", "", _
                              "Conn=eth:192.168.0.110, Timeout = 1000")
                                                'プロバイダの実装
    caoPV.Xtype 2                               '品種 2 に切換
    strResult = caoPV.Start                     'トリガ→処理待ち
    letx pTargetPos = posx(P11) + val(strResult)    '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」

End Sub
```

改訂履歴

デンソーロボット
プロバイダ
取扱説明書
パナソニックデバイス SUNX(株)製　ビジョンセンサ　PV 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	変数追加　"@ResultDisable"
Ver.1.0.2	Ver.1.3.7	RunManualコマンド修正
Ver.1.0.4	Ver.1.12.*	独自エラー（E_COMMAND_EXECUTING）追加 非同期コマンドの追加 キャリブレーション用コマンドの追加（PV260対応） タイムアウト設定／取得用コマンドの追加
Ver.1.0.5	Ver.2.3.*	AddControllerのオプション文字列にMyIPオプション追加
Ver.1.0.6		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