PV プロバイダ
Panasonic PV シリーズ用プロバイダ

Version 1.0.6

ユーザーズ ガイド

February 3, 2021

【備考】

【改版履歴】

バージョン	日付	内容
1.0.0.0	2011-12-8	初版.
1.0.0.1	2012-1-16	コマンド名修正
1.0.0	2012-07-17	ドキュメントのバージョンルールを変更
1.0.1	2012-09-24	@ResultDisable 変数追加
1.0.2	2012-11-7	RunManual コマンド バグ修正
1.0.3	2013-01-25	Quit コマンドの誤記を修正 MemoryRead コマンド, MemoryWrite コマンド PV500 対応
1.0.4	2015-07-28	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
1.0.5	2017-04-27	CaoWorkspace::AddController のオプション文字列に MyIP オプション追加
	2020-11-13	PV プロバイダに改名
1.0.6	2021-02-03	CalibrationStart, CalibrationStartAsync コマンド PV260 Ver1.10 対応 （カメラ No の引数を追加）

目次

1. はじめに

本書は Panasonic 社製 IMAGECHECKER PV シリーズ用の CAO プロバイダである，PV プロバイダのユーザーズガイドです．

PV プロバイダは Ethernet 接続でコマンドの送受信や画像の取得を行います．通信は Ethernet(TCP/IP)接続をサポートしています（現在，RS-232C の接続に関しては未対応です）．

PV プロバイダは，Panasonic 社製 IMAGECHECKER PV200，PV260 と PV500 に対応しています．

PV プロバイダは，マルチスレッドでは使用出来ません．

2. プロバイダの概要

2.1. 概要

PV プロバイダは, コマンドの実行方法として CaoController::Execute による方法を提供しています.

CaoController::Execute は Ethernet インタフェースで汎用通信により, PV シリーズを制御するための, 各ポート共通のコマンドの送受信を行うことができます.

表 2-1 PV プロバイダ

ファイル名	CaoProvPV.dll
ProgID	CaoProv.Panasonic.PV
レジストリ登録[1]	regsvr32 CaoProvPV.dll
レジストリ登録の抹消	regsvr32 /u CaoProvPV.dll

2.2. メソッド・プロパティ

2.2.1. CaoWorkspace::AddController メソッド

PV プロバイダでは AddController 時に, 通信用の接続パラメータを参照し, 通信の接続を行います. このときオプションで通信形態, タイムアウトを指定します.

PV プロバイダでは Ethernet 接続をサポートしています.

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

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

以下にオプション文字列に指定するリストを示します.

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

オプション	意味
Conn = <接続パラメータ>	必須. 通信形態とその接続パラメータを指定します.
Timeout[=<タイムアウト時間>]	送受信時のタイムアウト時間を指定します. (デフォルト:500msec)
`PV260[=<接続パラメータ>]	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

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

	合に指定します. 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.
MyIP=[<ローカル IP アドレス>[:ローカルポート番号]]	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.

2.2.1.1. Conn オプション

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

- Ethernet デバイス

  “eth:<IP Address>”

  <IP Address> : 接続する PV シリーズの IP アドレス.

  例:”127.0.0.1”, “10.5.5.100”

2.2.2. CaoController::Execute メソッド

PV シリーズを制御するための, 各ポート共通のコマンドの送受信を行います. 第 1 引数にコマンド名, 第 2 引数にコマンドのパラメータを指定します. 各コマンドの詳細はコマンドリファレンスを参照してください.

書式 [<vntRet:VARIANT> =]Execute (<bstrCmd:VT_BSTR>,[<vntParam : VT_VARIANT>])

bstrCommandName: [in] コマンド名

vntParam : [in] パラメータ

vntRet : [out] 戻り値

Ececute メソッド実行した際の PV シリーズからのステータスコードの戻り値は HRESULT として返されます.

正常に処理された場合 : S_OK

正常に処理されなかった場合 : 0x80100000 + 戻り値(16 進)

例: Start を実行したとき.

hr = 0x801000C8 : 運転停止(STOP)中のため, 実行不可.

エラー内容については Panasonic 社の PV シリーズのマニュアルを参照してください.

2.2.3. CaoController::AddVariable メソッド

画像の取得をするための変数を作成します. 画像の取得には表 2-3 コントローラクラス システム変数一覧を参照してください.

書式 AddVariable(<bstrVariableName:VT_BSTR>,[< bstrOption: VT_BSTR >])

bstrVariableName : [in] 変数名

bstrOption : [in] オプション文字列

使用例

```
Dim objBITMAP as object
Dim vntBITMAP as variant
objBITMAP = caoCtrl.AddVariable("@BITMAP", "")
vntBITMAP = objBITMAP.Value

vntBITMAP : カメラ画像のバイナリデータ
```

2.2.4. CaoVariable::put_value プロパティ

現在変数クラスでは put_value プロパティをサポートしていません.

2.2.5. CaoVariable::get_value プロパティ

画像は表 2-3 コントローラクラス システム変数一覧のフォーマットで取得できます.

2.3. 変数一覧

2.3.1. コントローラクラス

表 2-3 コントローラクラス システム変数一覧

変数名	データ型	説明
@BITMAP	VT_UI1|VT_ARRAY	カメラ画像を BITMAP フォーマットで取得します. 運用画面でのみ使用できます.
@BITMAP_MONITOR	VT_UI1|VT_ARRAY	PV シリーズのモニタ画像を BITMAP フォーマットで取得します.
@ResultDisable	VT_BOOL	True : 画像処理結果を取得しません False : 画像処理結果を取得します

※ カラーカメラの画像取得に関しては，現状未対応

2.4. エラーコード

PV 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．ORiN2 共通エラーについては，「ORiN2 プログラミングガイド」のエラーコードの章を参照してください．

表 2-4 独自エラーコード一覧

エラー名	エラー番号	説明
E_COMMAND_EXECUTING	0x80F00000	コマンド実行中に，別のコマンドを実行しました．
E_COMMAND_CONNECTED	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．

3. コマンドリファレンス

本章では CaoController::Execute メソッドの各コマンドについて解説します. 各コマンドの詳細動作については Panasonic 社の PV シリーズのマニュアルの汎用通信コマンド詳細を参照してください.

各コマンドと PV シリーズの対応一覧は,

また, 各コマンドの実行可能モードについては, Panasonic 社の PV シリーズのマニュアルの汎用通信コマンド一覧を参照してください.

表 3-1 CaoController::Execute コマンド一覧

PV シリーズコマンド	コマンド	概要	参照
汎用通信コマンド (同期)			
%S	Start	検査実行(全実行/分岐実行)	P15
		検査実行(指定実行)	P15
%R	Restart	再検査実行 (撮像せずに現在のメモリ画像で検査する)	P16
%X	Xtype	品種切り替え	P17
%MW	MemoryWrite	設定データ保存 本体保存用メモリ	P17
%CW	CFWrite	設定データ保存 SD カードメモリ	P17
%MR	MemoryRead	設定データ読み出し 本体保存用メモリ	P18
%CR	CFRead	設定データ読み出し SD カードメモリ	P18
%CD	CancelData	設定データ保存/読み出しの中断(キャンセル)	P19
%SS	SDSave	保存画像メモリ 保存(SD メモリーカード)	P19
%SR	SDReset	保存画像メモリ 消去	P19
%PS	PrintScreen	プリントスクリーン	P20
%Q	Quit	統計リセット	P20
%RM	RunManual	運転/停止(RUN/STOP)の切り替え	P20
%E	ErrorReset	エラー信号のリセット	P21
%CC	Cancel	検査/処理の中断(各種動作キャンセル)	P21
%K	KeyEmulator	キーエミュレート	P21
%BS	Bstop	キーパッド操作 受付禁止/受付許可	P23
%BC	Bconfirm	キーパッド操作 受付状態の確認	P23
%I	LayOutChange	レイアウト切り替え	P23
%A	AgainTemplate	テンプレートの再登録	P24
%PR	ParameterRead	パラメータ 読み出し	P24
%PRP	ParameterReadPair	パラメータ ペア読み出し(各種上下限値など)	P25

		ンセル)	
%SS	SDSaveAsync	非同期で保存画像メモリ 保存(SD メモリーカード)	P39
%SR	SDResetAsync	非同期で保存画像メモリ 消去	P40
%PS	PrintScreenAsync	非同期でプリントスクリーン	P40
%Q	QuitAsync	非同期で統計リセット	P41
%RM	RunManualAsync	非同期で運転/停止(RUN/STOP)の切り替え	P41
%E	ErrorResetAsync	非同期でエラー信号のリセット	P42
%CC	CancelAsync	非同期で検査/処理の中断(各種動作キャンセル)	P42
%K	KeyEmulatorAsync	非同期でキーエミュレート	P43
%BS	BstopAsync	非同期でキーパッド操作 受付禁止/受付許可	P43
%BC	BconfirmAsync	非同期でキーパッド操作 受付状態の確認	P44
%I	LayOutChangeAsync	非同期でレイアウト切り替え	P44
%A	AgainTemplateAsync	非同期でテンプレートの再登録	P45
%PR	ParameterReadAsync	非同期でパラメータ 読み出し	P46
%PRP	ParameterReadPairAsync	非同期でパラメータ ペア読み出し(各種上下限値など)	P46
%PW	ParameterWriteAsync	非同期でパラメータ 変更	P47
%PWP	ParameterWritePairAsync	非同期でパラメータ ペア変更(各種上下限値など)	P47
PV260 ロボットキャリブレーション用コマンド (非同期)			
%P=	SetPointAsync	非同期でロボット座標通知	P48
%CA	CalibrateAsync	非同期で計測開始指示(全実行)	P49
		非同期で計測開始指示(指定実行)	P49
%RCA	ReCalibrateAsync	非同期で再計測開始指示(全実行)	P50
		非同期で再計測開始指示(指定実行)	P51
%CAS	CalibrationStartAsync	非同期でキャリブレーション自動設定開始	P51
―	CalibrationEndAsync	非同期で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)	P52
%WCS	WorkSetAsync	非同期でワーク検出基準位置再登録	P52
%WRS	WorkResetAsync	非同期でワーク検出基準位置再登録開始	P53
―	WorkResetEndAsync	非同期で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)	P54
%MVE	MoveEndAsync	非同期で移動完了	P54
%TCD	GetTeachPointAsync	非同期でティーチング座標要求	P55

－	GetMovePointAsync	非同期でロボット移動座標受信 （CalibrationStart, WorkReset 関連コマンド）	P55
独自コマンド（非同期）			
－	RawAsync	非同期でコマンドメッセージの送信	P56
－	GetResult	非同期コマンドの戻り値を取得	P56

3.1. 汎用通信コマンド（同期）

3.1.1. CaoController::Execute(“Start”) コマンド

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.

3.1.1.1. 一括トリガで, 実行モードが「全実行」または「分岐実行」の場合

書式　Start ( )

引数　：　なし
戻り値　：　[out] 画像処理結果（VT_BSTR）

検査を実行する場合の例を以下に示します.

使用例

```
Dim bstrResult as string
bstrResult = caoCtrl.Execute("Start")

‘ bstrResult : 画像処理結果
```

3.1.1.2. 一括トリガで, 実行モードが「指定実行」の場合

書式　Start (<lBlockNum>)

< lBlockNum >　：　[in] 実行するブロックナンバー（VT_I4）（0～9）
戻り値　：　[out] 画像処理結果（VT_BSTR）

ブロックナンバー1 を指定し, 検査を実行する場合の例を以下に示します.

使用例

```
Dim lBlockNum as long
Dim bstrResult as string

lBlockNum = 1
```

```
bstrResult = caoCtrl.Execute("Start", lBlockNum)
' bstrResult : 画像処理結果
```

3.1.2. CaoController::Execute(“Restart”) コマンド

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.

3.1.2.1. 実行モードが「全実行」または「分岐実行」の場合

書式 Restart ( )

引数 : なし
戻り値 : [out] 画像処理結果(VT_BSTR)

再検査を実行する場合の例を以下に示します.

使用例

```
Dim bstrResult as string
bstrResult = caoCtrl.Execute "Restart"
' bstrResult : 画像処理結果
```

3.1.2.2. 実行モードが「指定実行」の場合

書式 Restart (< lBlockNum >)

< lBlockNum > : [in] 実行するブロックナンバー (VT_I4) (0～9)
戻り値 : [out] 画像処理結果(VT_BSTR)

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.

使用例

```
Dim lBlockNum as long
Dim bstrResult as string

lBlockNum = 1

bstrResult = caoCtrl.Execute "Restart", lBlockNum
```

```
‘ bstrResult : 画像処理結果
```

3.1.3. CaoController::Execute (“Xtype”) コマンド

品種を切り替えます.

書式 Xtype (<lProductNum>)

< lProductNum > : [in] 品種ナンバー (VT_I4) (0～255)
戻り値 : なし

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.

使用例

```
Dim lProductNum as long
lProductNum = 100
 caoCtrl.Execute "Xtype", lProductNum
```

3.1.4. CaoController::Execute (“MemoryWrite”) コマンド

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 MemoryWrite ([< lWriteNum >])

< lWriteNum > : このパラメータは接続している PV シリーズによって以下のように異なります.
PV200 なし
PV500 [in]本体の保存エリアナンバー(VT_I4) 0～99)
戻り値 : なし

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.

使用例

```
caoCtrl.Execute "MemoryWrite"
```

3.1.5. CaoController::Execute (“CFWrite”) コマンド

設定データを SD メモリーカードへ保存します.

書式 CFWrite (< lWriteNum >)

< lWriteNum > : [in] SD メモリーカードの保存エリアナンバー(VT_I4) (0～99)
戻り値 : なし

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.

使用例

```
Dim lWriteNum as long
 lWriteNum = 10
caoCtrl.Execute "CFWrite", lWriteNum
```

3.1.6. CaoController::Execute (“MemoryRead”) コマンド

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.

書式 MemoryRead ([<lReadNum>])

<lReadNum> : このパラメータは接続している PV シリーズによって以下のように異なります.

PV200 なし

PV500 [in]本体の保存エリアナンバー(VT_I4) (0～99)

戻り値 : なし

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.

使用例

```
caoCtrl.Execute "MemoryRead"
```

3.1.7. CaoController::Execute (“CFRead”) コマンド

SD メモリーカードから設定データを読み出します.

書式 CFRead (<lReadNum >)

< lReadNum > : [in] SD メモリーカードの読み出しエリアナンバー (VT_I4) (0～99)

戻り値 : なし

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.

使用例

```
Dim lReadNum as long
lReadNum = 10
caoCtrl.Execute "CFRead", lReadNum
```

3.1.8. CaoController::Execute (“CancelData”) コマンド

設定データの保存・読み出しを中断します.

書式 CancelData ( )

引数 : なし
戻り値 : なし

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.

使用例

```
caoCtrl.Execute "CancelData"
```

3.1.9. CaoController::Execute (“SDSave”) コマンド

本体に保存されている画像メモリを SD メモリーカードに保存します.

書式 SDSave ( )

引数 : なし
戻り値 : なし

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.

使用例

```
caoCtrl.Execute "SDSave"
```

3.1.10. CaoController::Execute (“SDReset”) コマンド

本体に保存されている画像メモリを消去します.

書式 SDReset ( )

引数 : なし
戻り値 : なし

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.

使用例

```
caoCtrl.Execute "SDReset"
```

3.1.11. CaoController::Execute (“PrintScreen”) コマンド

現在の表示画面(表示されているものすべて)をキャプチャし, SD メモリーカード, または Ethernet インタフェース経由でパソコンに保存します.

書式 PrintScreen ( )

引数 : なし
戻り値 : なし

現在の表示画面を保存する場合の例を以下に示します.

使用例

```
caoCtrl.Execute "PrintScreen"
```

3.1.12. CaoController::Execute (“Quit”) コマンド

統計データ, 走査回数をクリアします.

書式 Quit ( )

引数 : なし
戻り値 : なし

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.

使用例

```
caoCtrl.Execute "Quit"
```

3.1.13. CaoController::Execute (“RunManual”) コマンド

PV シリーズの運転・停止を切り替えます.

書式 [< lResult > =] RunManual (< lMode >)

< lMode > : [in] 運転・停止の切り替え (VT_I4) (0～1)
0 : 運転へ切り替え.
1 : 停止へ切り替え.

< lResult > : [out] 切り替えたモードの値 (VT_I4) (0～1)
0 : 運転.
1 : 停止.

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.

使用例

```
Dim lMode as long
Dim lResult as long

lMode = 1

lResult = caoCtrl.Execute("RunManual", lMode)

' lResult : 1
```

3.1.14. CaoController::Execute (“ErrorReset”) コマンド

Error 信号をリセットします.

書式 ErrorReset ( )

引数 : なし
戻り値 : なし

エラーをクリアする場合の例を以下に示します.

使用例

```
caoCtrl.Execute "ErrorReset"
```

3.1.15. CaoController::Execute (“Cancel”) コマンド

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.

書式 Cancel ( )

引数 : なし
戻り値 : なし

実行途中をキャンセルする場合の例を以下に示します.

使用例

```
caoCtrl.Execute "Cancel"
```

3.1.16. CaoController::Execute (“KeyEmulator”) コマンド

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.

書式 KeyEmulator (< lShift > < lKey >)

< lShift > : [in] シフトキーの ON・OFF (VT_I4) (0～1)

0 : OFF

1 : ON

< lKey > : [in] 各種キーに割り当てられた値（VT_I4）（1～16）

※各種キーに割り当てられた値は別表参照

戻り値 : なし

表 3-2 各種キーへの値割り当て一覧

No	割り当てキー	No	割り当てキー
1	ENTER キー左下	9	ENTER キー右上
2	ENTER キー下	10	TRIG キー
3	ENTER キー右下	11	CANCEL キー
4	ENTER キー左	12	FUNC キー
5	ENTER キー中心	13	F1 キー
6	ENTER キー右	14	F2 キー
7	ENTER キー左上	15	F3 キー
8	ENTER キー上	16	OPE/SET スイッチ

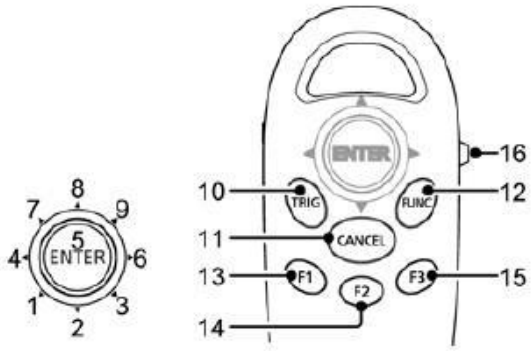

図 3-1 値割当てに対するキーパッドの対応図

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．

使用例

```
Dim lShift as long
Dim lKey as long

lShift = 0
lKey = 16

caoCtrl.Execute "KeyEmulator", Array(lShift, lKey)
```

3.1.17. CaoController::Execute (“Bstop”) コマンド

運用画面において, キーパッドによる操作の受け付けを禁止・許可します.

書式 Bstop (< lEnabled >)

< lEnabled > : [in] キーパッド操作受け付け可否 (VT_I4) (0～1)
0 : 受付許可
1 : 受付拒否

戻り値 : なし

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.

使用例

```
Dim lEnabled as long
lEnabled = 1
caoCtrl.Execute "Bstop", lEnabled
```

3.1.18. CaoController::Execute (“Bconfirm”) コマンド

キーパッドによる操作の受付状態を取得します.

書式 Bconfirm ()

引数 : なし

戻り値 : [out] キーパッド操作受付状態 (VT_I4) (0～1)
0 : 受付許可
1 : 受付拒否

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.

使用例

```
Dim lResult as long
lResult = caoCtrl.Execute("Bconfirm")
' lResult : 0
```

3.1.19. CaoController::Execute (“LayoutChange”) コマンド

運用画面で, 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.

書式 LayoutChange (< lLayOut >)

< lLayOut > : [in] レイアウトナンバー (VT_I4) (0～15)
戻り値 : なし

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.

使用例

```
Dim lLayOut as long

lLayOut = 1

caoCtrl.Execute "LayoutChange", lLayOut
```

3.1.20. CaoController::Execute ("AgainTemplate") コマンド

3.1.20.1. 領域表示をしない場合

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.

書式 AgainTemplate (< lChecker > , < lTemplate >)

< lChecker > : [in] チェッカナンバー (VT_I4) (0～999)
< lTemplate > : [in] テンプレートナンバー (VT_I4) (0～63)
戻り値 : なし

注意: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できません.

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.

使用例

```
Dim lChecker as long
Dim lTemplate as long

lChecker = 1
lTemplate = 10

caoCtrl.Execute "AgainTemplate", Array(lChecker, lTemplate)
```

3.1.21. CaoController::Execute ("ParameterRead") コマンド

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.

書式 ParameterRead (< bstrParam >)

< bstrParam > : [in] 読み出す対象のコマンドパラメータ (VT_BSTR)

戻り値 : [out] 指定したパラメータの値（VT_BSTR）

現在時刻を読み出す場合の例を以下に示します.

使用例

```
Dim bstrParam as string
Dim bstrResult as string

bstrParam = “SYS_TIME”

bstrResult = caoCtrl.Execute("ParameterRead", bstrParam)

‘ bstrResult : 12:29:30
```

3.1.22. CaoController::Execute（“ParameterReadPair”）コマンド

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.

書式 ParameterReadPair (< bstrParam >)

< bstrParam > : [in] 読み出し対象のコマンドパラメータ (VT_BSTR)

戻り値 : [out] 指定したパラメータの値（VT_BSTR | VT_ARRAY）

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.

使用例

```
Dim bstrParam as string
Dim vntResult as variant

bstrParam = “BLV:PAIRA”

vntResult = caoCtrl.Execute("ParameterReadPair", bstrParam)

‘ vntResult : 80,255（下限値,上限値）
```

3.1.23. CaoController::Execute（“ParameterWrite”）コマンド

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.

書式 ParameterWrite (< bstrParam >, < vntData >)

< bstrParam > : [in] 変更対象のコマンドパラメータ (VT_BSTR)

< vntData > : [in] 変更する値（VT_VARIANT）

戻り値 : なし

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.

使用例

```
Dim bstrParam as string
Dim vntData as variant

bstrParam = "SYS:REG0"
vntData = 3.14

caoCtrl.Execute "ParameterWrite", Array(bstrParam, vntData)
```

3.1.24. CaoController::Execute ("ParameterWritePair") コマンド

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.

書式 ParameterWritePair (< bstrParam >, < vntData1 >, < vntData2 >)

< bstrParam > : [in] 変更対象のコマンドパラメータ (VT_BSTR)
< vntData1 > : [in] 変更する値1(VT_VARIANT)
< vntData2 > : [in] 変更する値2(VT_VARIANT)
戻り値 : なし

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.

使用例

```
Dim bstrParam as string
Dim vntData1 as variant
Dim vntData2 as variant

bstrParam = "CAC010:LPAIR"
vntData1 = 50
vntData2 = 100

caoCtrl.Execute "ParameterWritePair", Array(bstrParam, vntData1, vntData2)
```

3.2. PV260 ロボットキャリブレーション用コマンド (同期)

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

PV にロボット座標を通知します.

書式 SetPoint (< dblX >, < dblY >, < dblZ >, < dblRx >, < dblRy >, < dblRz >, < lFig >)

1.

< dblX >	:	[in] ロボット座標(X)(VT_R8)
< dblY >	:	[in] ロボット座標(Y)(VT_R8)
< dblZ >	:	[in] ロボット座標(Z)(VT_R8)
< dblRx >	:	[in] ロボット座標(Rx)(VT_R8)
< dblRy >	:	[in] ロボット座標(Ry)(VT_R8)
< dblRz >	:	[in] ロボット座標(Rz)(VT_R8)
< lFig >	:	[in] ロボット座標(Fig)(VT_I4)
戻り値	:	なし

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.

使用例

```
caoCtrl.Execute "SetPoint", Array(POSX(CURPOS), POSY(CURPOS), POSZ(CURPOS),
                                  POSRX(CURPOS), POSRY(CURPOS), POSRZ(CURPOS), FIG(CURPOS))
```

3.2.2. CaoController::Execute("Calibrate") コマンド

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.

計測を行うためには,「4.2Calibrate, Recalibrate コマンドシーケンス」に沿って, ロボットと連携したプログラムを用意する必要があります.

3.2.2.1. 実行モードが「全実行」の場合

書式 Calibrate (< lCalibrationNum >)

< lCalibrationNum >	:	[in] キャリブレーション No(VT_I4)(0～5)
戻り値	:	[out] ロボット座標(X, Y, Rz, Fig)の配列(VT_VARIANT)

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.

使用例

```
Dim lCalibrationNum as long
Dim vntResult as variant

lCalibrationNum = 0

vntResult = caoCtrl.Execute("Calibrate", lCalibrationNum)

' vntResult : ロボット座標情報 (X, Y, Rz, Fig)
```

3.2.2.2. 実行モードが「指定実行」の場合

書式 Calibrate (< lCalibrationNum >, <lBlockNum>)

< lCalibrationNum > : [in] キャリブレーション No（VT_I4）（0～5）
< lBlockNum > : [in] 実行するブロックナンバー（VT_I4）（0～9）
戻り値 : [out] ロボット座標（X, Y, Rz, Fig）の配列（VT_VARIANT）

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.

使用例

```
Dim lCalibrationNum as long
Dim lBlockNum as long
Dim vntResult as variant

lCalibrationNum = 0
lBlockNum = 1

vntResult = caoCtrl.Execute("Calibrate", Array(lCalibrationNum, lBlockNum))

‘ vntResult : ロボット座標情報 (X, Y, Rz, Fig)
```

3.2.3. CaoController::Execute(“ReCalibrate”) コマンド

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.

再計測を行うためには,「4.2Calibrate, Recalibrate コマンドシーケンス」に沿って, ロボットと連携したプログラムを用意する必要があります.

3.2.3.1. 実行モードが「全実行」の場合

書式 ReCalibrate (< lCalibrationNum >)

< lCalibrationNum > : [in] キャリブレーション No（VT_I4）（0～5）
戻り値 : [out] ロボット座標（X, Y, Rz, Fig）の配列（VT_VARIANT）

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.

使用例

```
Dim lCalibrationNum as long
Dim vntResult as variant

lCalibrationNum = 0

vntResult = caoCtrl.Execute("ReCalibrate", lCalibrationNum)
```

```
' vntResult : ロボット座標情報 (X, Y, Rz, Fig)
```

3.2.3.2. 実行モードが「指定実行」の場合

書式 ReCalibrate (< lCalibrationNum >, <lBlockNum>)

< lCalibrationNum > : [in] キャリブレーション No (VT_I4) (0～5)
< lBlockNum > : [in] 実行するブロックナンバー (VT_I4) (0～9)
戻り値 : [out] ロボット座標(X, Y, Rz, Fig)の配列(VT_VARIANT)

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.

使用例

```
Dim lCalibrationNum as long
Dim lBlockNum as long
Dim vntResult as variant

lCalibrationNum = 0
lBlockNum = 1

vntResult = caoCtrl.Execute("ReCalibrate", Array(lCalibrationNum, lBlockNum))

' vntResult : ロボット座標情報 (X, Y, Rz, Fig)
```

3.2.4. CaoController::Execute("CalibrationStart") コマンド

自動キャリブレーションを開始します. 自動キャリブレーションを行うためには,「4.3CalibrationStart コマンドシーケンス」に沿ってロボットと連携したプログラムを用意する必要があります.

カメラ No の引数を省略した場合は, カメラ No の指定なしとなります. (Ver1.1.0 以前のフォーマット)

書式 CalibrationStart (< lCalibrationNum > [, < lCameraNum >])

< lCalibrationNum > : [in] キャリブレーション No (VT_I4)(0～5)
< lCameraNum > : [in] カメラ No (VT_I4) (0～1)
戻り値 : なし

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.

使用例

```
Dim lCalibrationNum as long
Dim lCameraNum as long
```

```
lCalibrationNum = 0
lCameraNum = 1

‘ カメラ No 指定なし(%CAS0) を PV に通知
caoCtrl.Execute "CalibrationStart", lCalibrationNum

‘ カメラ No 指定あり(%CAS0,1)を PV に通知
caoCtrl.Execute "CalibrationStart", Array(lCalibrationNum, lCameraNum)
```

3.2.5. CaoController::Execute(“CalibrationEnd”) コマンド

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．自動キャリブレーションを行うためには，「4.3CalibrationStartコマンドシーケンス」に沿ってロボットと連携したプログラムを用意する必要があります．

書式　CalibrationEnd ( )

引数　：　なし

戻り値　：　なし

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．

使用例

```
caoCtrl.Execute "CalibrationEnd"
```

3.2.6. CaoController::Execute(“WorkSet”) コマンド

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　WorkSet ( )

引数　：　なし

戻り値　：　なし

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．

使用例

```
caoCtrl.Execute "WorkSet"
```

3.2.7. CaoController::Execute(“WorkReset”) コマンド

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．

撮像ありの基準位置の再登録を行うためには,「4.4WorkReset コマンドシーケンス」に沿ってロボットと連携したプログラムを用意する必要があります.

書式 WorkReset (< lWorkNum >)

< lWorkNum > : [in] ワーク検出 No (VT_I4)(0～15)
戻り値 : なし

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.

使用例

```
Dim lWorkNum as long
lWorkNum = 0
caoCtrl.Execute "WorkReset", lWorkNum
```

3.2.8. CaoController::Execute("WorkResetEnd") コマンド

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. ワーク検出基準位置の再登録を行うためには,「4.4WorkReset コマンドシーケンス」に沿ってロボットと連携したプログラムを用意する必要があります.

書式 WorkResetEnd ( )

引数 : なし
戻り値 : なし

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.

使用例

```
caoCtrl.Execute "WorkResetEnd"
```

3.2.9. CaoController::Execute("MoveEnd") コマンド

ロボットの移動が完了したことを PV に通知します.

書式 MoveEnd ( )

引数 : なし
戻り値 : なし

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.

使用例

```
caoCtrl.Execute "MoveEnd"
```

3.2.10. CaoController::Execute(“GetTeachPoint”) コマンド

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 GetTeachPoint ( )

引数 : なし

戻り値 : ロボット移動座標(X, Y, Rz, Fig)配列
(VT_VARIANT | VT_ARRAY)

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.

使用例

```
Dim vntResult as variant
vntResult = caoCtrl.Execute("GetTeachPoint")
‘ vntResult : ロボット座標情報 (X, Y, Rz, Fig) 配列
```

3.2.11. CaoController::Execute(“GetMovePoint”) コマンド

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. 自動キャリブレーション, ワーク検出基準位置の再登録を行うためには,「4.3CalibrationStart コマンドシーケンス」,「4.4WorkReset コマンドシーケンス」に沿ってロボットと連携したプログラムを用意する必要があります.

書式 GetMovePoint ( )

引数 : なし

戻り値 : ロボット移動座標(X, Y, Rz, Fig)(VT_VARIANT)

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.

使用例

```
Dim vntMovePos as variant
vntMovePos = caoCtrl.Execute("GetMovePoint")
‘ vntMovePos : ロボット移動座標
```

3.3. 独自コマンド (同期)

3.3.1. CaoController::Execute(“Raw”) コマンド

コマンドメッセージを送受信します. BCC については, 内部で自動計算します.

書式 Raw (< bstrCmdMessage >)

< bstrCmdMessage > : [in] 送信するコマンドメッセージ (VT_BSTR)

戻り値 : [out] 受信したコマンドメッセージ(VT_BSTR)

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．

使用例

```
Dim bstrParam as string
Dim bstrResult as string

bstrCmdMessage = "%S"

vntResult = caoCtrl.Execute "Raw", bstrCmdMessage
```

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

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 SetTimeout (< lTimeout >)

< lTimeout > : [in] タイムアウト時間 msec (VT_I4)

戻り値 : なし

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．

使用例

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

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

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 GetTimeout ( )

引数 : なし

戻り値 : [out] タイムアウト時間 msec(VT_I4)

タイムアウト時間(1000msec)を取得する例を以下に示します．

使用例

```
Dim lResult as long

lResult = caoCtrl.Execute("GetTimeout")

' lResult:1000
```

3.4. 汎用通信コマンド （非同期）

3.4.1. CaoController::Execute(“StartAsync”) コマンド

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. GetResult コマンドの詳細については, 3.6.2.CaoController::Execute(“GetResult”) コマンドを参照ください.

3.4.1.1. 一括トリガで, 実行モードが「全実行」または「分岐実行」の場合

書式　StartAsync ( )

引数　：　なし

戻り値　：　なし

非同期で検査を実行する場合の例を以下に示します.

使用例

```
Dim vntResult as variant

caoCtrl.Execute "StartAsync"

‘ StartAsync コマンドの戻り値取得
vntResult = caoCtrl.Execute(“GetResult”)

‘ vntResult : 検査結果
```

3.4.1.2. 一括トリガで, 実行モードが「指定実行」の場合

書式　StartAsync (<lBlockNum>)

< lBlockNum >　：　[in] 実行するブロックナンバー （VT_I4） （0～9）

戻り値　：　なし

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

使用例

```
Dim lBlockNum as long
Dim bstrResult as string

lBlockNum = 1

caoCtrl.Execute "StartAsync", lBlockNum

‘ StartAsync コマンドの戻り値取得
bstrResult = caoCtrl.Execute(“GetResult”)

‘ bstrResult : 検査結果
```

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

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. GetResult コマンドの詳細については, 3.6.2.CaoController::Execute("GetResult") コマンドを参照ください.

3.4.2.1. 実行モードが「全実行」または「分岐実行」の場合

書式 RestartAsync ( )

引数 : なし
戻り値 : なし

非同期で再検査を実行する場合の例を以下に示します.

使用例

```
Dim vntResult as variant

caoCtrl.Execute "RestartAsync"

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

' vntResult : 検査結果
```

3.4.2.2. 実行モードが「指定実行」の場合

書式 RestartAsync (< lBlockNum >)

< lBlockNum > : [in] 実行するブロックナンバー (VT_I4) (0～9)
戻り値 : なし

非同期で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.

使用例

```
Dim lBlockNum as long
Dim vntResult as variant

lBlockNum = 1

caoCtrl.Execute "RestartAsync", lBlockNum
```

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

' vntResult : 画像処理結果
```

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

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. GetResult コマンドの詳細については, 3.6.2.CaoController::Execute("GetResult") コマンドを参照ください.

書式 XtypeAsync (<lProductNum>)

< lProductNum > : [in] 品種ナンバー (VT_I4) (0～255)

戻り値 : なし

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.

使用例

```
Dim lProductNum as long
Dim vntResult as variant

lProductNum = 100

caoCtrl.Execute "XtypeAsync", lProductNum

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

' vntResult : 戻り値 (Empty)
```

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

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. GetResult コマンドの詳細については, 3.6.2.CaoController::Execute("GetResult") コマンドを参照ください.

書式 MemoryWriteAsync ([< lWriteNum >])

< lWriteNum > : このパラメータは接続している PV シリーズによって以下のように異なります.

PV200 なし

PV500 [in]本体の保存エリアナンバー(VT_I4) 0～99)

戻り値 : なし

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.

使用例

```
Dim vntResult as variant

caoCtrl.Execute "MemoryWriteAsync"

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

‘ vntResult : 戻り値 (Empty)
```

3.4.5. CaoController::Execute (“CFWriteAsync”) コマンド

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. GetResult コマンドの詳細については, 3.6.2.CaoController::Execute(“GetResult”) コマンドを参照ください.

書式 CFWriteAsync (< lWriteNum >)

< lWriteNum > : [in] SD メモリーカードの保存エリアナンバー(VT_I4) (0～99)

戻り値 : なし

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.

使用例

```
Dim lWriteNum as long
Dim vntResult as variant

 lWriteNum = 10

caoCtrl.Execute "CFWriteAsync", lWriteNum

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

‘ vntResult : 戻り値 (Empty)
```

3.4.6. CaoController::Execute (“MemoryReadAsync”) コマンド

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. GetResult コマンドの詳細については, 3.6.2.CaoController::Execute(“GetResult”) コマンドを参照ください.

書式 MemoryReadAsync ([<lReadNum>])

<lReadNum> : このパラメータは接続している PV シリーズによって以下のように異なります.

PV200 なし

PV500 [in]本体の保存エリアナンバー（VT_I4）（0～99）

戻り値 : なし

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.

使用例

```
Dim vntResult as variant

caoCtrl.Execute "MemoryReadAsync"

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

‘ vntResult : 戻り値（Empty）
```

3.4.7. CaoController::Execute（“CFReadAsync”）コマンド

非同期で SD メモリー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. GetResult コマンドの詳細については，3.6.2.CaoController::Execute(“GetResult”) コマンドを参照ください.

書式 CFReadAsync (<lReadNum >)

< lReadNum > : [in] SD メモリーカードの読み出しエリアナンバー（VT_I4）（0～99）

戻り値 : なし

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.

使用例

```
Dim lReadNum as long
Dim vntResult as variant

lReadNum = 10

caoCtrl.Execute "CFReadAsync", lReadNum

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

‘ vntResult : 戻り値（Empty）
```

3.4.8. CaoController::Execute (“CancelDataAsync”) コマンド

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. GetResult コマンドの詳細については, 3.6.2.CaoController::Execute(“GetResult”) コマンドを参照ください.

書式 CancelDataAsync ( )

引数 : なし

戻り値 : なし

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.

使用例

```
Dim vntResult as variant

caoCtrl.Execute "CancelDataAysnc"

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

‘ vntResult : 戻り値 (Empty)
```

3.4.9. CaoController::Execute (“SDSaveAsync”) コマンド

非同期で本体に保存されている画像メモリを SD メモリー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. GetResult コマンドの詳細については, 3.6.2.CaoController::Execute(“GetResult”) コマンドを参照ください.

書式 SDSaveAsync ( )

引数 : なし

戻り値 : なし

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.

使用例

```
Dim vntResult as variant

caoCtrl.Execute "SDSaveAsync"

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

‘ vntResult : 戻り値 (Empty)
```

3.4.10. CaoController::Execute (“SDResetAsync”) コマンド

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. GetResult コマンドの詳細については, 3.6.2.CaoController::Execute(“GetResult”) コマンドを参照ください.

書式 SDResetAsync ( )

引数 : なし

戻り値 : なし

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.

使用例

```
Dim vntResult as variant

caoCtrl.Execute "SDResetAsync"

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

‘ vntResult : 戻り値 (Empty)
```

3.4.11. CaoController::Execute (“PrintScreenAsync”) コマンド

非同期で現在の表示画面(表示されているものすべて)をキャプチャし, SD メモリーカード, または Ethernet インタフェー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. GetResult コマンドの詳細については, 3.6.2.CaoController::Execute(“GetResult”) コマンドを参照ください.

書式 PrintScreenAsync ( )

引数 : なし

戻り値 : なし

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.

使用例

```
Dim vntResult as variant

caoCtrl.Execute "PrintScreenAsync"

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

‘ vntResult : 戻り値 (Empty)
```

3.4.12. CaoController::Execute (“QuitAsync”) コマンド

非同期で統計データ, 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. GetResult コマンドの詳細については, 3.6.2.CaoController::Execute(“GetResult”) コマンドを参照ください.

書式 QuitAsync ( )

引数 : なし

戻り値 : なし

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.

使用例

```
Dim vntResult as variant

caoCtrl.Execute "QuitAsync"

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

‘ vntResult : 戻り値 (Empty)
```

3.4.13. CaoController::Execute (“RunManualAsync”) コマンド

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. GetResult コマンドの詳細については, 3.6.2.CaoController::Execute(“GetResult”) コマンドを参照ください.

書式 RunManualAsync (< lMode >)

< lMode > : [in] 運転・停止の切り替え (VT_I4) (0～1)

0 : 運転へ切り替え.

1 : 停止へ切り替え.

戻り値 : なし

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.

使用例

```
Dim lMode as long
Dim vntResult as variant

lMode = 1

caoCtrl.Execute "RunManualAsync", lMode

‘ RunManualAsync コマンドの戻り値取得
vntResult = caoCtrl.Execute("GetResult")
```

```
' vntResult : 1
```

3.4.14. CaoController::Execute (“ErrorResetAsync”) コマンド

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. GetResult コマンドの詳細については, 3.6.2.CaoController::Execute(“GetResult”) コマンドを参照ください.

書式 ErrorResetAsync ( )

引数 : なし

戻り値 : なし

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.

使用例

```
Dim vntResult as variant

caoCtrl.Execute "ErrorResetAsync"

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

' vntResult : 戻り値 (Empty)
```

3.4.15. CaoController::Execute (“CancelAsync”) コマンド

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. GetResult コマンドの詳細については, 3.6.2.CaoController::Execute(“GetResult”) コマンドを参照ください.

書式 CancelAsync ( )

引数 : なし

戻り値 : なし

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.

使用例

```
Dim vntResult as variant

caoCtrl.Execute "CancelAsync"

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

' vntResult : 戻り値 (Empty)
```

3.4.16. CaoController::Execute (“KeyEmulatorAsync”) コマンド

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. GetResult コマンドの詳細については, 3.6.2.CaoController::Execute(“GetResult”) コマンドを参照ください.

書式　KeyEmulator (< lShift > < lKey >)

< lShift >	:	[in] シフトキーの ON・OFF (VT_I4) (0～1) 0 : OFF 1 : ON
< lKey >	:	[in] 各種キーに割り当てられた値 (VT_I4) (1～16) ※各種キーに割り当てられた値は KeyEmulator コマンドを参照
戻り値	:	なし

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.

使用例

```
Dim lShift as long
Dim lKey as long
Dim vntResult as variant

lShift = 0
lKey = 16

caoCtrl.Execute "KeyEmulatorAsync", Array(lShift, lKey)

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

‘ vntResult : 戻り値 (Empty)
```

3.4.17. CaoController::Execute (“BstopAsync”) コマンド

非同期で運用画面において, 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. GetResult コマンドの詳細については, 3.6.2.CaoController::Execute(“GetResult”) コマンドを参照ください.

書式　BstopAsync (< lEnabled >)

< lEnabled >	:	[in] キーパッド操作受け付け可否 (VT_I4) (0～1) 0 : 受付許可 1 : 受付拒否
戻り値	:	なし

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.

使用例

```
Dim lEnabled as long
Dim vntResult as variant

lEnabled = 1

caoCtrl.Execute "BstopAsync", lEnabled

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

‘ vntResult : 戻り値 (Empty)
```

3.4.18. CaoController::Execute (“BconfirmAsync”) コマンド

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. GetResult コマンドの詳細については, 3.6.2.CaoController::Execute(“GetResult”) コマンドを参照ください.

書式 BconfirmAsync ()

引数 : なし

戻り値 : なし

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.

使用例

```
Dim vntResult as long

caoCtrl.Execute "BconfirmAsync"

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

‘ vntResult : 0
```

3.4.19. CaoController::Execute (“LayOutChangeAsync”) コマンド

非同期で運用画面で, 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. GetResult コマンドの詳細については, 3.6.2.CaoController::Execute(“GetResult”) コマンドを参照ください.

書式 LayoutChangeAsync (< lLayOut >)

< lLayOut > : [in] レイアウトナンバー (VT_I4) (0～15)

戻り値 : なし

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.

使用例

```
Dim lLayOut as long
Dim vntResult as variant

lLayOut = 1

caoCtrl.Execute "LayoutChangeAsync", lLayOut

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

‘ vntResult : 戻り値 (Empty)
```

3.4.20. CaoController::Execute (“AgainTemplateAsync”) コマンド

3.4.20.1. 領域表示をしない場合

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. GetResult コマンドの詳細については, 3.6.2.CaoController::Execute(“GetResult”) コマンドを参照ください.

書式 AgainTemplateAsync (< lChecker > , < lTemplate >)

< lChecker >	:	[in] チェッカナンバー (VT_I4) (0～999)
< lTemplate >	:	[in] テンプレートナンバー (VT_I4) (0～63)
戻り値	:	なし

注意: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できません.

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.

使用例

```
Dim lChecker as long
Dim lTemplate as long
Dim vntResult as variant

lChecker = 1
lTemplate = 10

call caoCtrl.Execute("AgainTemplateAsync", Array(lChecker, lTemplate))

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

‘ vntResult : 戻り値 (Empty)
```

3.4.21. CaoController::Execute (“ParameterReadAsync”) コマンド

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. GetResult コマンドの詳細については, 3.6.2.CaoController::Execute(“GetResult”) コマンドを参照ください.

書式 ParameterReadAsync (< bstrParam >)

< bstrParam >	:	[in] 読み出す対象のコマンドパラメータ (VT_BSTR)
戻り値	:	[out] 指定したパラメータの値 (VT_BSTR)

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.

使用例

```
Dim bstrParam as string
Dim vntResult as string

bstrParam = “SYS_TIME”

caoCtrl.Execute "ParameterReadAsync", bstrParam

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

‘ vntResult : 12:29:30
```

3.4.22. CaoController::Execute (“ParameterReadPairAsync”) コマンド

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. GetResult コマンドの詳細については, 3.6.2.CaoController::Execute(“GetResult”) コマンドを参照ください.

書式 ParameterReadPairAsync (< bstrParam >)

< bstrParam >	:	[in] 読み出し対象のコマンドパラメータ (VT_BSTR)
戻り値	:	[out] 指定したパラメータの値 (VT_BSTR | VT_ARRAY)

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.

使用例

```
Dim bstrParam as string
Dim vntResult as variant
```

```
bstrParam = “BLV:PAIRA”

caoCtrl.Execute "ParameterReadPairAsync", bstrParam

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

‘ vntResult : 80,255 （下限値,上限値）
```

3.4.23. CaoController::Execute (“ParameterWriteAsync”) コマンド

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. GetResult コマンドの詳細については, 3.6.2.CaoController::Execute(“GetResult”) コマンドを参照ください.

書式 ParameterWriteAsync (< bstrParam >, < vntData >)

< bstrParam >	:	[in] 変更対象のコマンドパラメータ (VT_BSTR)
< vntData >	:	[in] 変更する値（VT_VARIANT）
戻り値	:	なし

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.

使用例

```
Dim bstrParam as string
Dim vntData as variant
Dim vntResult as variant

bstrParam = “SYS:REG0”
vntData = 3.14

caoCtrl.Execute "ParameterWriteAsync", Array(bstrParam, vntData)

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

‘ vntResult : 戻り値 (Empty)
```

3.4.24. CaoController::Execute (“ParameterWritePairAsync”) コマンド

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. GetResult コマンドの詳細については, 3.6.2.CaoController::Execute(“GetResult”) コマンドを参照ください.

書式 ParameterWritePairAsync (< bstrParam >, < vntData1 >, < vntData2 >)

< bstrParam >	:	[in] 変更対象のコマンドパラメータ (VT_BSTR)
< vntData1 >	:	[in] 変更する値1(VT_VARIANT)
< vntData2 >	:	[in] 変更する値2(VT_VARIANT)
戻り値	:	なし

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.

使用例

```
Dim bstrParam as string
Dim vntData1 as variant
Dim vntData2 as variant
Dim vntResult as variant

bstrParam = "CAC010:LPAIR"
vntData1 = 50
vntData2 = 100

caoCtrl.Execute "ParameterWritePairAsync", Array(bstrParam, vntData1, vntData2)

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

' vntResult : 戻り値 (Empty)
```

3.5. PV260 ロボットキャリブレーション用コマンド (非同期)

3.5.1. CaoController::Execute ("SetPointAsync") コマンド

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. GetResult コマンドの詳細については, 3.6.2.CaoController::Execute("GetResult") コマンドを参照ください.

書式 SetPointAsync (< dblX >, < dblY >, < dblZ >, < dblRx >, < dblRy >, < dblRz >, < lFig >)

< dblX >	:	[in] ロボット座標(X)(VT_R8)
< dblY >	:	[in] ロボット座標(Y)(VT_R8)
< dblZ >	:	[in] ロボット座標(Z)(VT_R8)
< dblRx >	:	[in] ロボット座標(Rx)(VT_R8)
< dblRy >	:	[in] ロボット座標(Ry)(VT_R8)
< dblRz >	:	[in] ロボット座標(Rz)(VT_R8)
< lFig >	:	[in] ロボット座標(Fig)(VT_I4)
戻り値	:	なし

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.

使用例

```
Dim vntResult as variant

caoCtrl.Execute "SetPointAsync", Array(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.Execute("GetResult")

' vntResult : Empty
```

3.5.2. CaoController::Execute (“CalibrateAsync”) コマンド

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.

計測を行うためには,「4.2Calibrate, Recalibrate コマンドシーケンス」に沿って, ロボットと連携したプログラムを用意する必要があります.

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. GetResult コマンドの詳細については, 3.6.2.CaoController::Execute(“GetResult”) コマンドを参照ください.

3.5.2.1. 実行モードが「全実行」の場合

書式 CalibrateAsync (< lCalibrationNum >)

< lCalibrationNum > : [in] キャリブレーション No(VT_I4)(0～5)

戻り値 : なし

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.

使用例

```
Dim lCalibrationNum as long
Dim vntResult as variant

lCalibrationNum = 0

caoCtrl.Execute "CalibrateAsync", lCalibrationNum

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

' vntResult : ロボット座標情報 (X, Y, Rz, Fig)
```

3.5.2.2. 実行モードが「指定実行」の場合

書式 CalibrateAsync (< lCalibrationNum >, <lBlockNum>)

< lCalibrationNum > : [in] キャリブレーション No (VT_I4) (0～9)

< lBlockNum > : [in] 実行するブロックナンバー (VT_I4) (0～9)
戻り値 : なし

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.

使用例

```
Dim lCalibrationNum as long
Dim lBlockNum as long
Dim vntResult as variant

lCalibrationNum = 0
lBlockNum = 1

call caoCtrl.Execute("CalibrateAsync", Array(lCalibrationNum, lBlockNum))

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

‘ vntResult : ロボット座標情報 (X, Y, Rz, Fig)
```

3.5.3. CaoController::Execute (“ReCalibrateAsync”) コマンド

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.

再計測を行うためには,「4.2Calibrate, Recalibrate コマンドシーケンス」に沿って, ロボットと連携したプログラムを用意する必要があります.

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. GetResult コマンドの詳細については, 3.6.2.CaoController::Execute(“GetResult”) コマンドを参照ください.

3.5.3.1. 実行モードが「全実行」の場合

書式 ReCalibrateAsync (< lCalibrationNum >)

< lCalibrationNum > : [in] キャリブレーション No(VT_I4)(0～5)
戻り値 : なし

非同期で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.

使用例

```
Dim lCalibrationNum as long
Dim vntResult as variant

lCalibrationNum = 0

caoCtrl.Execute "ReCalibrateAsync", lCalibrationNum
```

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

'vntResult : ロボット座標情報 (X, Y, Rz, Fig)
```

3.5.3.2. 実行モードが「指定実行」の場合

書式　ReCalibrateAsync (< lCalibrationNum >, <lBlockNum>)

< lCalibrationNum >	:	[in] キャリブレーション No（VT_I4）（0～5）
< lBlockNum >	:	[in] 実行するブロックナンバー（VT_I4）（0～9）
戻り値	:	なし

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.

使用例

```
Dim lCalibrationNum as long
Dim lBlockNum as long
Dim vntResult as variant

lCalibrationNum = 0
lBlockNum = 1

call caoCtrl.Execute("ReCalibrateAsync", Array(lCalibrationNum, lBlockNum))

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

' vntResult : ロボット座標情報 (X, Y, Rz, Fig)
```

3.5.4. CaoController::Execute (“CalibrationStartAsync”) コマンド

非同期で自動キャリブレーションを開始します. 自動キャリブレーションを行うためには, 「4.3CalibrationStartコマンドシーケンス」に沿ってロボットと連携したプログラムを用意する必要があります.

カメラ No の引数を省略した場合は, カメラ No の指定なしとなります. (Ver1.1.0 以前のフォーマット)

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. GetResult コマンドの詳細については, 3.6.2.CaoController::Execute(“GetResult”) コマンドを参照ください.

書式　CalibrationStartAsync (< lCalibrationNum > [, < lCameraNum >])

< lCalibrationNum >	:	[in] キャリブレーション No (VT_I4)(0～5)
< lCameraNum >	:	[in] カメラ No (VT_I4) (0～1)
戻り値	:	なし

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．

使用例

```
Dim lCalibrationNum as long
Dim lCameraNum as long
Dim vntResult as variant

lCalibrationNum = 0
lCameraNum = 1

‘ カメラ No 指定なし(%CAS0) を PV に通知
caoCtrl.Execute "CalibrationStartAsync", lCalibrationNum

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


‘ カメラ No 指定あり(%CAS0,1)を PV に通知
caoCtrl.Execute "CalibrationStartAsync", Array(lCalibrationNum, lCameraNum)

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


‘ vntResult : 戻り値なし (Empty)
```

3.5.5. CaoController::Execute (“CalibrationEndAsync”) コマンド

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．自動キャリブレーションを行うためには，「4.3CalibrationStart コマンドシーケンス」に沿ってロボットと連携したプログラムを用意する必要があります．

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．GetResult コマンドの詳細については，3.6.2.CaoController::Execute(“GetResult”) コマンドを参照ください．

書式 CalibrationEndAsync ( )

引数 : なし

戻り値 : なし

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．

使用例

```
Dim vntResult as variant

caoCtrl.Execute "CalibrationEndAsync"

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

‘ vntResult : 戻り値 (Empty)
```

3.5.6. CaoController::Execute (“WorkSetAsync”) コマンド

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．GetResult コマンドの詳細については，3.6.2.CaoController::Execute(“GetResult”) コマンドを参照ください．

書式　WorkSetAsync ( )

引数　：　なし

戻り値　：　なし

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．

使用例

```
Dim vntResult as variant

caoCtrl.Execute "WorkSetAsync"

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

‘ vntResult : 戻り値 (Empty)
```

3.5.7. CaoController::Execute (“WorkResetAsync”) コマンド

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．

撮像ありの基準位置の再登録を行うためには，「4.4WorkReset コマンドシーケンス」に沿ってロボットと連携したプログラムを用意する必要があります．

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．GetResult コマンドの詳細については，3.6.2.CaoController::Execute(“GetResult”) コマンドを参照ください．

書式　WorkResetAsync (< lWorkNum >)

< lWorkNum >　：　[in] ワーク検出 No (VT_I4)(0～15)

戻り値　：　なし

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．

使用例

```
Dim lWorkNum as long
Dim vntResult as variant

lWorkNum = 0

caoCtrl.Execute "WorkResetAsync", lWorkNum

‘ WorkResetAsync コマンドの戻り値取得
```

```
vntResult = caoCtrl.Execute("GetResult")

' vntResult : 戻り値 (Empty)
```

3.5.8. CaoController::Execute("WorkResetEndAsync") コマンド

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. ワーク検出基準位置の再登録を行うためには,「4.4WorkResetコマンドシーケンス」に沿ってロボットと連携したプログラムを用意する必要があります.

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. GetResult コマンドの詳細については, 3.6.2.CaoController::Execute("GetResult") コマンドを参照ください.

書式 WorkResetEndAsync ( )

引数 : なし

戻り値 : なし

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.

使用例

```
Dim vntResult as variant

caoCtrl.Execute "WorkResetEndAsync"

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

' vntResult : 戻り値 (Empty)
```

3.5.9. CaoController::Execute ("MoveEndAsync") コマンド

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. GetResult コマンドの詳細については, 3.6.2.CaoController::Execute("GetResult") コマンドを参照ください.

書式 MoveEndAsync ( )

引数 : なし

戻り値 : なし

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.

使用例

```
Dim vntResult as variant

caoCtrl.Execute "MoveEndAsync"

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

```
' vntResult : 戻り値 (Empty)
```

3.5.10. CaoController::Execute ("GetTeachPointAsync") コマンド

非同期で PV で設定してある, 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. GetResult コマンドの詳細については, 3.6.2.CaoController::Execute("GetResult") コマンドを参照ください.

書式 GetTeachPointAsync ( )

引数 : なし

戻り値 : なし

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.

使用例

```
Dim vntResult as variant

caoCtrl.Execute "GetTeachPointAsync"

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

' vntResult : ロボット座標情報 (X, Y, Rz, Fig) 配列
```

3.5.11. CaoController::Execute("GetMovePointAsync") コマンド

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. 自動キャリブレーション, ワーク検出基準位置の再登録を行うためには,「4.3CalibrationStart コマンドシーケンス」,「4.4WorkReset コマンドシーケンス」に沿ってロボットと連携したプログラムを用意する必要があります.

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. GetResult コマンドの詳細については, 3.6.2.CaoController::Execute("GetResult") コマンドを参照ください.

書式 GetMovePoint ( )

引数 : なし

戻り値 : ロボット移動座標(X, Y, Rz, Fig)(VT_VARIANT)

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.

使用例

```
Dim vntMovePos as variant

caoCtrl.Execute "GetMovePointAsync"

' GetMovePointAsync コマンドの戻り値取得
vntMovePos = caoCtrl.Execute("GetResult")
```

```
' vntMovePos : ロボット移動座標
```

3.6. 汎用通信コマンド（非同期）

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

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. GetResult コマンドの詳細については, 3.6.2.CaoController::Execute(“GetResult”) コマンドを参照ください.

書式 RawAsync (< bstrCmdMessage >)

< bstrCmdMessage >	:	[in] 送信するコマンドメッセージ (VT_BSTR)
戻り値	:	[out] なし

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.

使用例

```
Dim bstrParam as string
Dim vntResult as variant

bstrCmdMessage = "%S"

caoCtrl.Execute "RawAsync", bstrCmdMessage

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

' vntResult : 検査結果
```

3.6.2. CaoController::Execute(“GetResult”) コマンド

非同期コマンドの完了待ちを行い, 非同期コマンドの戻り値を取得します.

戻り値がない非同期コマンドを実行した場合, 戻り値はありません.

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 GetResult ( )

引数	:	なし
戻り値	:	[out] 非同期コマンドの戻り値(VT_VARIANT) 戻り値は実行したコマンドに依存します

非同期で検査を実行した場合の，戻り値を取得する例を以下に示します．

使用例

```
Dim vntResult as variant
caoCtrl.Execute "StartAsync"
vntResult = caoCtrl.Execute("GetResult")

‘ vntResult : 戻り値
```

4. PV260 ロボットキャリブレーション機能

4.1. PV260 の設定

PV プロバイダでは PV260 のロボットキャリブレーション機能と連携するためのコマンドを提供しています. このコマンドを使用するために, PV260 に以下の設定を行ってください.

1. 「2.ロボット設定」－「ロボット通信設定」－「メーカ」を「DENSO」に設定（図 2-1 参照）
2. 「2.ロボット設定」－「ロボット通信設定」－「座標フォーマット設定」－「手系 右手」を「0」に設定（図 2-2 参照）
3. 「2.ロボット設定」－「ロボット通信設定」－「座標フォーマット設定」－「手系 左手」を「1」に設定（図 2-2 参照）

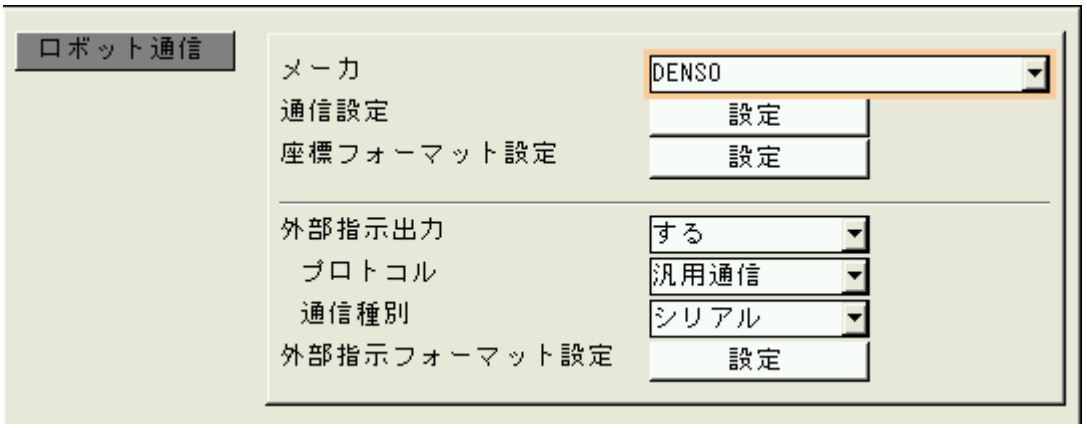

図 2-1. ロボット通信設定 – メーカ

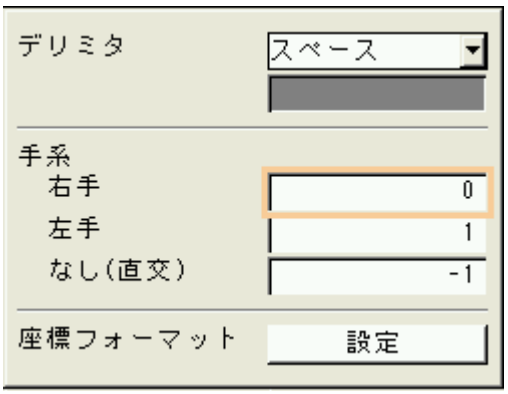

図 2-2. ロボット通信設定 ― 手系

4.2. Calibrate, Recalibrate コマンドシーケンス

Calibrate, Recalibrate コマンドを PV, ロボットと連携して実行するシーケンスを以下に示します.

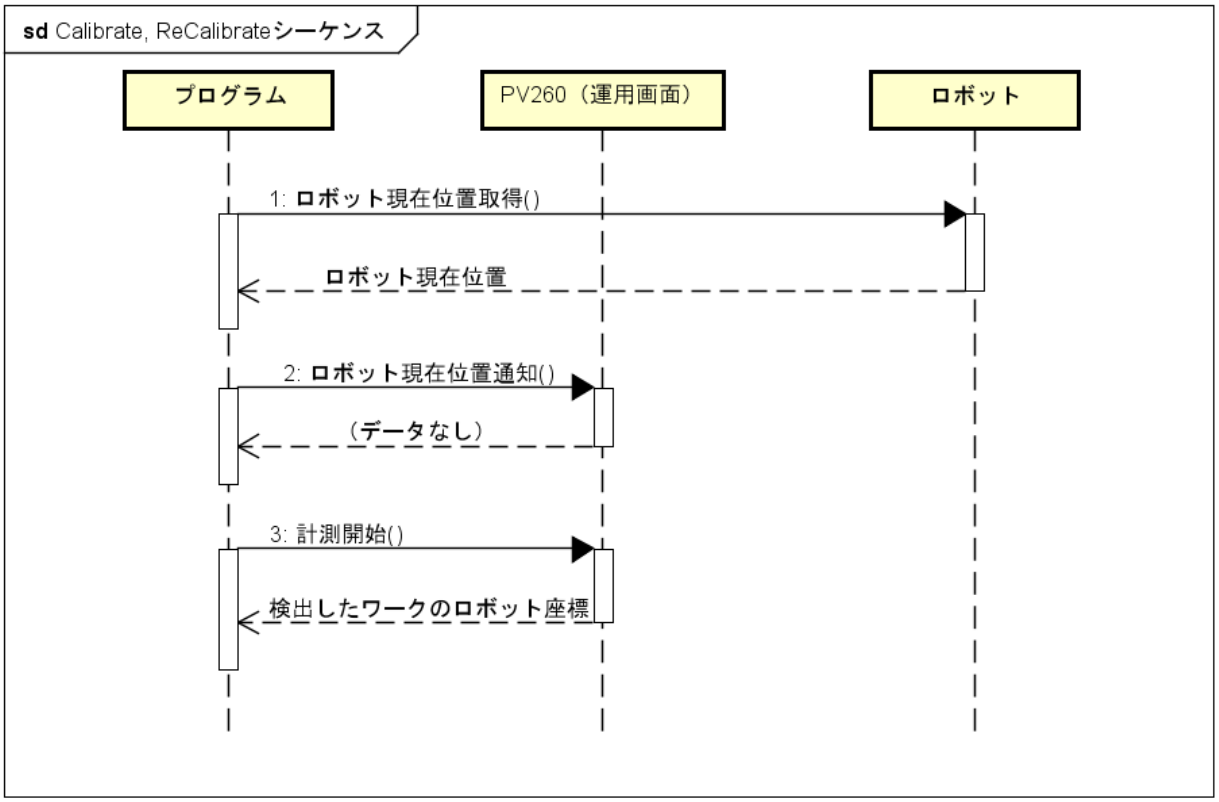

図 4-1 Calibrate, Recalibrate コマンドシーケンス

表 4-1 Calibrate, Recalibrate コマンドシーケンス処理概要

図中番号	処理概要	対応コマンド	備考
1	ロボット現在位置取得	CurPos	PAC コマンド
2	ロボット現在位置通知	SetPoint	－
3	計測開始	Calibrate / Recalibrate	－

以下に PacScript で作成したサンプルプログラムを示します． IP アドレスは PV で設定した値を用いてください. サンプルプログラムでは以下の設定値を用いて接続しています.

IP アドレス(PV) ： 192.168.0.62

Sample ProCA.pcs

```
'!TITLE "ProCA"
#Include "Variant.h"

#Define Address "192.168.0.62"
```

```
Sub Main
	TakeArm

	Dim objPV As Object
	Dim vntRet As Variant
	Dim vntPos As Variant
	Dim li As Long
	Dim lPosBase As Position
	Dim lPosMove As Position

	lPosBase = CurPos
	Set objPV = Cao.AddController( "pv", "CaoProv.Panasonic.PV", "", "PV260=1, Conn=eth:" & Address )

	' SetPoint
	Call objPV.SetPoint( VarChangeType( CurPos, VT_R4+VT_ARRAY ) )

	' Calibrate
	vntRet = objPV.Calibrate( 0 )

	If UBound( vntRet ) >= 0 Then
	    For li = 0 To UBound( vntRet )
	        vntPos = vntret( li )
	        If ( ( vntPos( 0 ) = 0 ) And ( vntPos( 1 ) = 0 ) And ( vntPos( 2 ) = 0 ) ) Then
	            PrintDbg "NotFound"
	            Exit Sub
	        End If

	        lPosMove = lPosBase
	        LetX lPosMove = vntPos( 0 )
	        LetY lPosMove = vntPos( 1 )
	        LetRz lPosMove = vntPos( 2 )
	        LetF lPosMove = vntPos( 3 )

	        Move P, @E lPosMove
	        Delay 500
	    Next
	End If

	GiveArm
End Sub
```

4.3. CalibrationStart コマンドシーケンス

CalibrationStart コマンドを PV, ロボットと連携して実行するシーケンスを以下に示します.

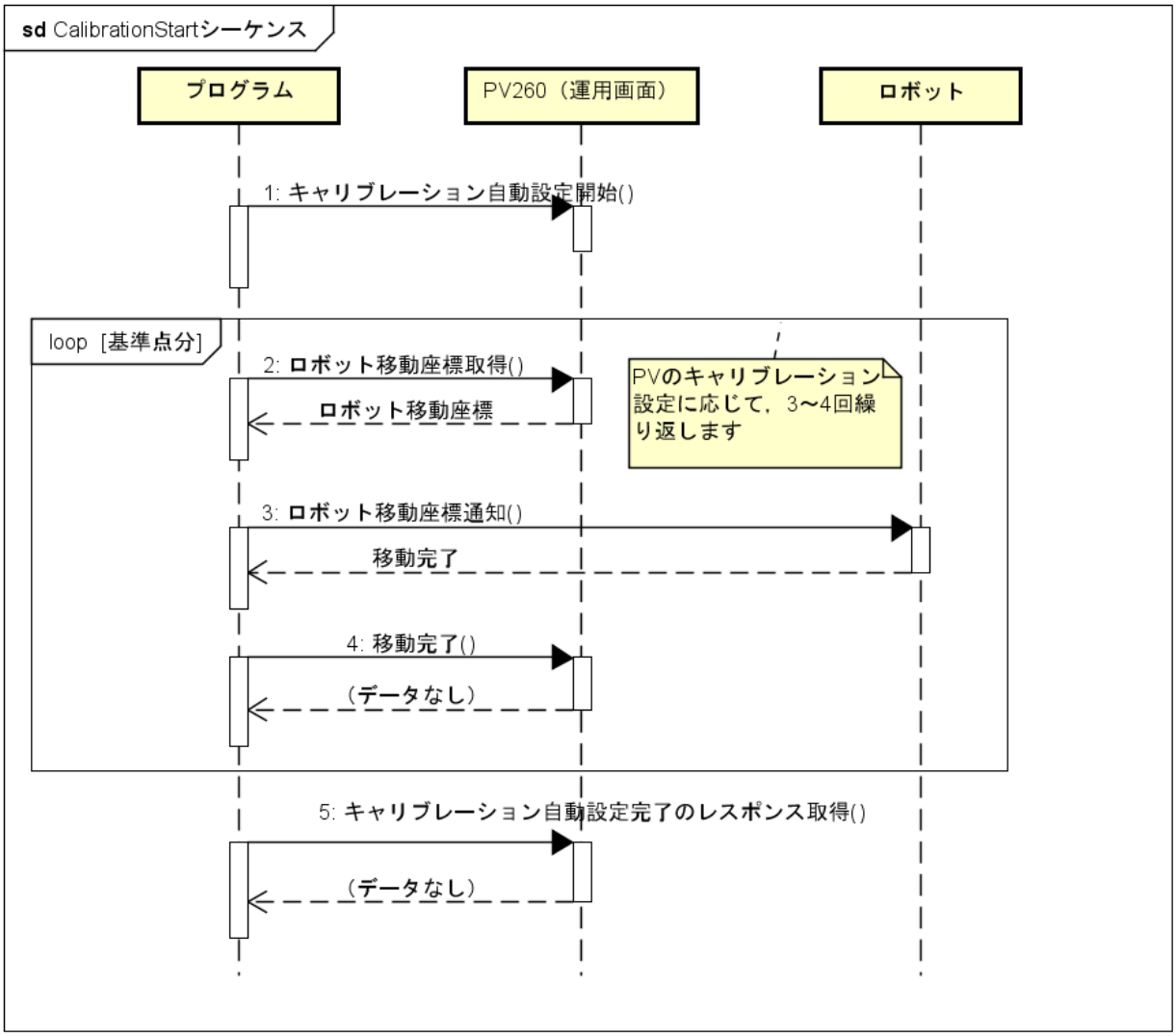

図 4-2 CalibrationStart コマンドシーケンス

表 4-2 CalibrationStart コマンドシーケンス処理概要

図中番号	処理概要	対応コマンド	備考
1	キャリブレーション自動設定開始	CalibrationStart	－
2	ロボット移動座標取得	GetMovePoint	－
3	ロボット移動座標通知	Move	PAC コマンド
4	移動完了	MoveEnd	－
5	キャリブレーション自動設定完了のレスポンス取得	CalibrationEnd	－

以下に PacScript で作成したサンプルプログラムを示します． IP アドレスは PV で設定した値を用いてください．サンプルプログラムでは以下の設定値を用いて接続しています．

IP アドレス(PV) : 192.168.0.62

Sample ProCAS.pcs

```
'!TITLE "ProCAS"
#include "Variant.h"

#define ADDRESS "192.168.0.62"
#define CAL_NO 0

Sub Main
	takearm

	Dim objPV as Object
	Dim vntVal as Variant
	Dim li as long
	Dim lpBase as Position
	Dim lpMove(3) as Position

	lpBase = CurPos

	set objPV = cao.AddController("pv", "CaoProv.Panasonic.PV", "", " PV260=1,Conn=eth:" & ADDRESS)

	' CalibrationStart
	call objPV.Execute("CalibrationStart", CAL_NO)

	' 3 点取得
	for li = 0 to 3
	    ' GetMovePoint
	    vntVal = objPV.Execute("GetMovePoint")

	    ' ベース位置をコピー
	    lpMove(li) = lpBase

	    ' 座標データの設定
	    if (vartype(vntVal) And VT_ARRAY) then
	        LETX lpMove(li) = vntVal(0)
	        LETY lpMove(li) = vntVal(1)
	        LETRZ lpMove(li) = vntVal(2)
	        LETF lpMove(li) = vntVal(3)

	        Move P, @E lpMove(li)
	        delay 500
	    end if

	    ' MoveEnd
	    call objPV.Execute("MoveEnd")
	    delay 1000
	next

	' CalibrationEnd
	call objPV.Execute("CalibrationEnd")

	givearm
End Sub
```

4.4. WorkReset コマンドシーケンス

WorkReset コマンドを PV, ロボットと連携して実行するシーケンスを以下に示します.

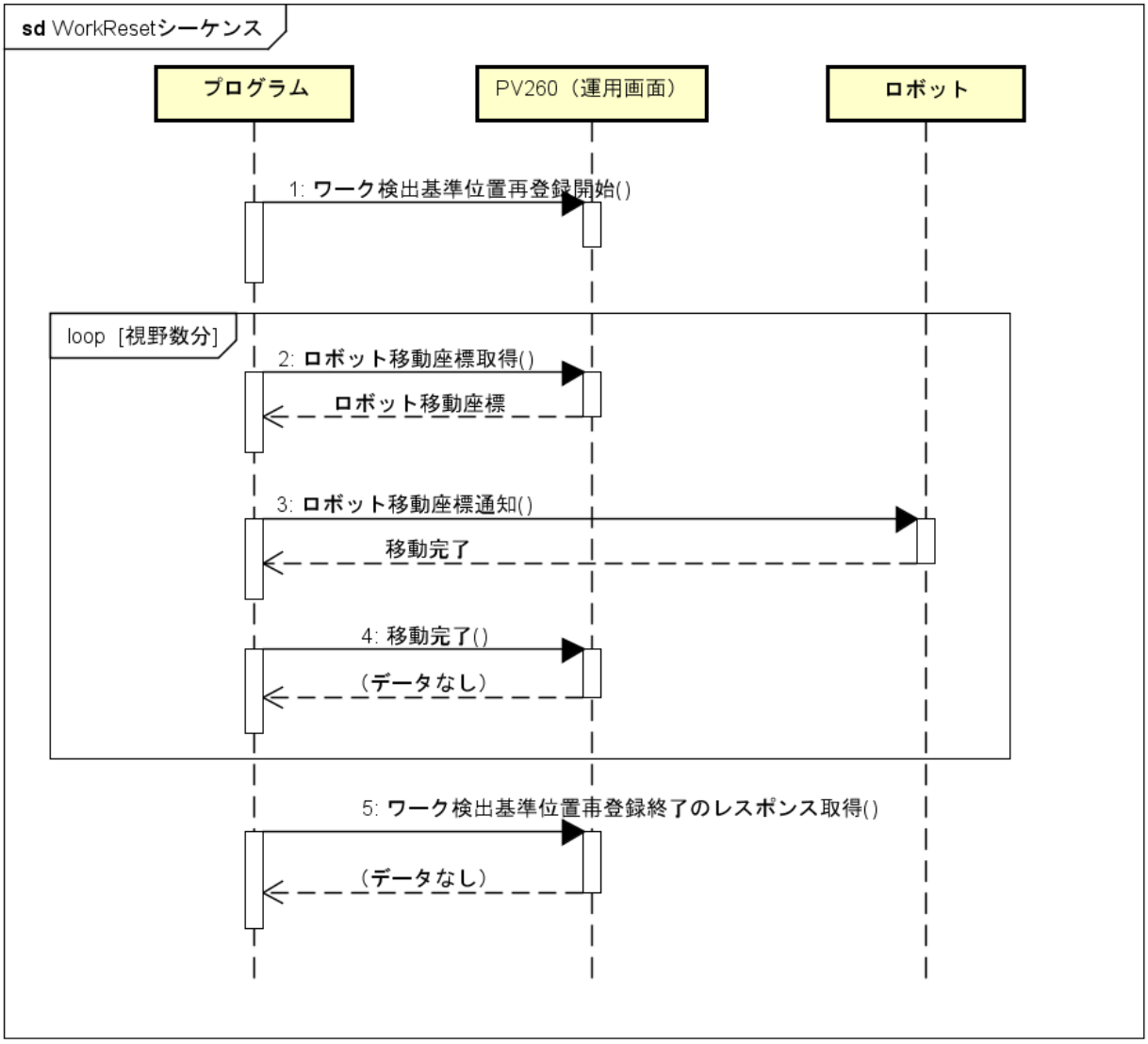

図 4-3 WorkReset コマンドシーケンス

表 4-3 WorkReset コマンドシーケンス処理概要

図中番号	処理概要	対応コマンド	備考
1	ワーク検出基準位置再登録開始	WorkReset	－
2	ロボット移動座標取得	GetMovePoint	－
3	ロボット移動座標通知	Move	PAC コマンド
4	移動完了	MoveEnd	－
5	ワーク検出基準位置再登録終了のレスポンス取得	WorkResetEnd	－

以下に PacScript で作成したサンプルプログラムを示します． IP アドレスは PV で設定した値を用いてくださ

い. サンプルプログラムでは以下の設定値を用いて接続しています.

IP アドレス(PV) : 192.168.0.62

Sample ProWRS.pcs

```
'!TITLE "ProWRS"

#Include "Variant.h"

#Define ADDRESS "192.168.0.62"
#Define WORK_NO 0

Sub Main
    TakeArm

    Dim objPV As Object
    Dim vntVal As Variant
    Dim li As Long
    Dim lpBase As Position
    Dim lpMove( 1 ) As Position

    lpBase = CurPos

    Set objPV = Cao.AddController( "pv", "CaoProv.Panasonic.PV", "", " PV260=1,Conn=eth:" & ADDRESS )

    ' WorkReset
    Call objPV.Execute( "WorkReset", WORK_NO )

    ' 2 点取得
    For li = 0 To 1
        ' GetMovePoint
        vntVal = objPV.Execute( "GetMovePoint" )

        ' ベース位置をコピー
        lpMove( li ) = lpBase

        ' 座標データの設定
        If ( VarType( vntVal ) And VT_ARRAY ) Then
            LetX lpMove( li ) = vntVal( 0 )
            LetY lpMove( li ) = vntVal( 1 )
            LetRz lpMove( li ) = vntVal( 2 )
            LetF lpMove( li ) = vntVal( 3 )

            Move P, @E lpMove( li )
            Delay 500
        End If

        ' MoveEnd
        Call objPV.Execute( "MoveEnd" )
        Delay 1000
    Next

    ' CalibrationEnd
    Call objPV.Execute( "WorkResetEnd" )

    GiveArm
End Sub
```

付録A. コマンド対応表

表 A-1 コマンド対応表

		画像処理装置		
PV コマンド名	プロバイダコマンド名	PV200	PV500	PV260
PV シリーズ対応コマンド				
%S	Start	○	○	○
%R	Restart	○	○	○
%X	Xtype	○	○	○
%MW	MemoryWrite	○	○	○
%CW	CFWrite	○	○	○
%MR	MemoryRead	○	○	○
%CR	CFRead	○	○	○
%CD	CancelData	○	○	○
%SS	SDSave	○	○	○
%SR	SDReset	○	○	○
%PS	PrintScreen	○	○	○
%Q	Quit	○	○	○
%RM	RunManual	○	○	○
%E	ErrorReset	○	○	○
%CC	Cancel	○	○	○
%K	KeyEmulator	○	○	○
%BS	Bstop	○	○	○
%BC	Bconfirm	○	○	○
%I	LayoutChange	○	○	○
%A	AgainTemplate	○	○	○
%PR	ParameterRead	○	○	○
%PRP	ParameterReadPair	○	○	○
%PW	ParameterWrite	○	○	○
%PWP	ParameterWritePair	○	○	○
%P=	SetPoint	×	×	○
%CA	Calibrate	×	×	○
%RCA	Recalibrate	×	×	○

%CAS	CalibrationStart	×	×	○
―	CalibrationEnd	×	×	○
%WCS	WorkSet	×	×	○
%WRS	WorkReset	×	×	○
―	WorkResetEnd	×	×	○
%MVE	MoveEnd	×	×	○
%TCD	GetTeachPoint	×	×	○
―	GetMovePoint	×	×	○
独自コマンド				
―	Raw	○	○	○
―	SetTimeout	○	○	○
―	GetTimeout	○	○	○
PV シリーズ対応コマンド（非同期）				
%S	StartAsync	○	○	○
%R	RestartAsync	○	○	○
%X	XtypeAsync	○	○	○
%MW	MemoryWriteAsync	○	○	○
%CW	CFWriteAsync	○	○	○
%MR	MemoryReadAsync	○	○	○
%CR	CFReadAsync	○	○	○
%CD	CancelDataAsync	○	○	○
%SS	SDSaveAsync	○	○	○
%SR	SDResetAsync	○	○	○
%PS	PrintScreenAsync	○	○	○
%Q	QuitAsync	○	○	○
%RM	RunManualAsync	○	○	○
%E	ErrorResetAsync	○	○	○
%CC	CancelAsync	○	○	○
%K	KeyEmulatorAsync	○	○	○
%BS	BstopAsync	○	○	○
%BC	BconfirmAsync	○	○	○
%I	LayoutChangeAsync	○	○	○
%A	AgainTemplateAsync	○	○	○
%PR	ParameterReadAsync	○	○	○

%PRP	ParameterReadPairAsync	○	○	○
%PW	ParameterWriteAsync	○	○	○
%PWP	ParameterWritePairAsync	○	○	○
%P=	SetPointAsync	×	×	○
%CA	CalibrateAsync	×	×	○
%RCA	RecalibrateAsync	×	×	○
%CAS	CalibrationStartAsync	×	×	○
－	CalibrationEndAsync	×	×	○
%WCS	WorkSetAsync	×	×	○
%WRS	WorkResetAsync	×	×	○
－	WorkResetEndAsync	×	×	○
%MVE	MoveEndAsync	×	×	○
%TCD	GetTeachPointAsync	×	×	○
－	GetMovePointAsync	×	×	○
独自コマンド（非同期）				
－	RawAsync	○	○	○
－	GetResult	○	○	○