SMC プロバイダ

CONTEC SMC ボード

Version 1.0.4

ユーザーズ ガイド

May 15, 2017

【備考】

【改版履歴】

バージョン	日付	内容
1.0.0.0	2011-7-27	初版.
1.0.1.0	2011-9-27	マニュアル修正.
1.0.2.0	2012-5-29	メタモード追加.
1.0.2	2012-7-17	ドキュメントのバージョンルールを変更.
1.0.3	2012-10-25	“ProviderCancel”,” ProviderClear” コマンドを追加
1.0.4	2013-2-7	SMC API が返すエラーコードマスク値修正
	2017-5-15	マニュアル修正

【対応機器】

機種	バージョン	注意事項
SMC-4DL-PE		
SMC-4DF-PCI		

目次

1. はじめに 4

2. プロバイダの概要 5

2.1. 概要 5
2.2. メソッド・プロパティ 6
2.2.1. CaoWorkspace::AddController メソッド 6
2.2.2. CaoController::Execute メソッド 6
2.2.3. CaoController::AddVariable メソッド 6
2.2.4. CaoController::GetVariableNames プロパティ 7
2.2.5. CaoController::AddExtension メソッド 7
2.2.6. CaoController::GetExtensionNames プロパティ 7
2.2.7. CaoExtension::Execute メソッド 8
2.2.8. CaoExtension::AddVariable メソッド 8
2.2.9. CaoExtension::GetVariableNames プロパティ 8
2.2.10. CaoVariable::get_Value プロパティ 8
2.2.11. CaoVariable::put_Value プロパティ 8
2.3. コマンド一覧 9
2.3.1. コントローラクラス 9
2.3.2. 拡張ボードクラス 9
2.4. 変数一覧 11
2.4.1. コントローラクラス 11
2.4.2. 拡張ボードクラス 11
2.5. エラーコード 19
2.6. CAO-SMC API 対応表 19

3. サンプルプログラム 22

1. はじめに

本書は, CONTEC 製 SMC ボードにアクセスするためのプロバイダである, SMC プロバイダのユーザーズガイドです.

詳細については, CONTEC 社 API-SMC（WDM）Help を参照して下さい.

注意: SMC プロバイダを使用するには、SMC ボードの SMC デバイスドライバをインストールしなければなりません. CONTEC API-PAC（W32）よりインストールして下さい. ドライバインストール後にプロバイダをレジストリ登録する必要があります. レジストリ登録の方法は表 2-1 を参照してください.

2. プロバイダの概要

2.1. 概要

SMC プロバイダは，CAO API を実行するときに対応する CONTEC 社 API を実行します．

CAO API と CONTEC 社 API の対応については表 2-10 を参照してください．

表 2-1 SMC プロバイダ

ファイル名	CaoProvSMC.dll
ProgID	CaoProv.CONTEC.SMC
レジストリ登録 [1]	regsvr32 CaoProvSMC.dll
レジストリ登録の抹消	regsvr32 /u CaoProvSMC.dll

[1] SMCボードのドライバをインストールしていないと，SMCプロバイダの登録はできません．

2.2. メソッド・プロパティ

2.2.1. CaoWorkspace::AddController メソッド

SMC プロバイダでは Controller オブジェクトの生成時に SMC ボードとの接続（オープン）処理を行います.

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

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

マシン名は空文字列で構いません.
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
DeviceName=<デバイス名>	接続先ボードのデバイス名 注: SMC ボード ID に対応したデバイス名を指定. ※1

※1:詳細については, API-SMC（WDM）Help を参照して下さい.

2.2.2. CaoController::Execute メソッド

使用できるコマンド名とパラメータの詳細は表 2-5 を参照してください.

書式 Execute(< bstrCommand:BSTR > [,<vntParam:VARIANT>[,< pVal:VARIANT>]])

bstrCommand : [in] コマンド名
vntParam : [in] パラメータ
pVal : [out] 取得データ

2.2.3. CaoController::AddVariable メソッド

SMC ボードにアクセスする変数オブジェクトを生成します. 変数名には, 2.4.1 の変数のみ使用することができます. これら以外の変数名を指定したときは, このメソッドはエラーを返します.

書式 AddVariable(<bstrName:BSTR > [, <bstrOption:BSTR>])

bstrName : [in] 任意の名前
bstrOption : [in] オプション文字列（未使用）

2.2.4. CaoController::GetVariableNames プロパティ

2.4.1の変数名リストを取得します.

2.2.5. CaoController::AddExtension メソッド

SMC 拡張ボードの動作を行う CaoExtension を生成します.

書式 AddExtension (<bstrName:BSTR > [,<bstrOption:BSTR>])

bstrName : [in] 拡張ボード名

bstrOption : [in] オプション文字列

使用可能な "拡張ボード名" を下表に記します.

表 2-3 拡張ボード名一覧

拡張ボード名	データ型	説明
Axis?	VT_ BSTR	?は、SMC ボードの制御対象モータ軸番号(1～)※1を指定します. 変数名の後ろに論理番号を指定します. 例) "Axis1"

※1:SMC ボード機種により軸の搭載数が異なります. 詳細は, API-SMC(WDM)Help を参照して下さい.

使用可能な "オプション文字列"を下表に記します.

表 2-4 CaoController: AddExtension のオプション文字列

オプション	意味
MotorOut[=<モータ出力>]	モータ出力先となる汎用出力信号※を割り当てます. 0:割当てしない(デフォルト) 1:汎用出力信号 1(OUT1) 2:汎用出力信号 2(OUT2) 3:汎用出力信号 3(OUT3)

※ 詳細については, API-SMC(WDM)Help を参照して下さい.

2.2.6. CaoController::GetExtensionNames プロパティ

2.2.5の拡張ボード名リストを取得します.

2.2.7. CaoExtension::Execute メソッド

書式 Execute(< bstrCommand:BSTR > [,<vntParam:VARIANT>[,< pVal:VARIANT>]])

bstrCommand : [in] コマンド名
vntParam : [in] パラメータ
pVal : [out] 取得データ

使用可能な“コマンド名”は表 2-6 を参照してください.

2.2.8. CaoExtension::AddVariable メソッド

SMC ボードの指定軸にアクセスする変数オブジェクトを生成します. 変数名には, 2.4.2 の変数のみ使用することができます. これら以外の変数名を指定したときは, このメソッドはエラーを返します.

書式 AddVariable(<bstrName:BSTR > [, <bstrOption:BSTR>])

bstrName : [in] 任意の名前
bstrOption : [in] オプション文字列（未使用）

2.2.9. CaoExtension::GetVariableNames プロパティ

2.4.2の変数名リストを取得します.

2.2.10. CaoVariable::get_Value プロパティ

変数に対応する情報を取得します. 各変数の実装状況および取得データについては, 2.4.2 を参照して下さい.

2.2.11. CaoVariable::put_Value プロパティ

変数に対応する情報を設定します. 各変数の実装状況および設定データについては, 2.4.2 を参照して下さい.

2.3. コマンド一覧

2.3.1. コントローラクラス

表 2-5 CaoController::Execute コマンド一覧

コマンド名	データ型	パラメータ	取得データ	説明
ProviderCancel	―	―	―	キャンセル状態設定 [全オブジェクト(Axis1～)が MotorOut=0 時] 何もしません. 実行結果は正常(S_OK)が返ります. [MotorOut=0 以外時] オブジェクト(Axis1～)のモータ出力信号を強制OFF 後、”STOP“コマンドを実行します. また、実行結果は、”E_PROV_CANCEL“エラーを返します.
ProviderClear	―	―	―	キャンセル状態解除 何もしません. 実行結果は正常(S_OK)が返ります.

2.3.2. 拡張ボードクラス

表 2-6 CaoExtension::Execute コマンド一覧

コマンド名	データ型	パラメータ※1	取得データ	説明
STOP	―	―	―	停止
DSTP	―	―	―	減速停止
ALMCLR	―	―	―	アラームクリア信号パルスの出力
ERCOUT	VT_I2	0: ERC 信号を出力. 1: ERC 信号をリセット.	―	偏差カウンタクリア(ERC)信号の出力
ORG	VT_I2	動作方向（0:CW, 1:CCW）	―	原点復帰
MOVP	VT_ARRAY|VT_I4	<要素 1: Coodinate> 位置の座標タイプを設定 0:絶対座標, 1:相対座標 <要素 2: StopPosition> 停止位置を設定 - 設定可能範囲 - 絶対座標:-134,217,728～+134,217,728 相対座標:-134,217,728～+134,217,72(0 以外)	―	指定位置移動
MOVJ	VT_I2	動作方向（0:CW, 1:CCW）	―	JOG 移動

MCHG	VT_ARRAY| VT_I4	動作中のモータ動作の変更 <要素 1: ChangeType> モータ動作変更タイプを設定 0:瞬時に開始速度(FL 速度)へ変更 1:瞬時に目標速度(FH 速度)へ変更 2:減速して開始速度(FL 速度)へ変更 3:加速して目標速度(FH 速度)へ変更 4:動作速度と加減速度を変更 5:モータ停止位置変更 6:PCS 信号によるモータ停止位置変更設定 <要素 2: Coodinate >要素 1 が 5 の時のみ必要 位置の座標タイプを設定 0:絶対座標, 1:相対座標 <要素 3: StopPosition >要素 1 が 5 の時のみ必要 停止位置を設定 -設定可能範囲- 絶対座標:-134,217,728～+134,217,728 相対座標:-134,217,728～+134,217,72(0 以外)	ー	動作中の速度変更, 停止位置変更

※1:詳細については, API-SMC（WDM）Help を参照して下さい.

2.4. 変数一覧

2.4.1. コントローラクラス

表 2-7 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	属性	
			get	put
@ERROR	VT_I4	最後に発生した SMC ドライバ関数のエラーコード※1を読込みます.	○	―

※1:詳細については, API-SMC（WDM）Help を参照して下さい.

2.4.2. 拡張ボードクラス

表 2-8 拡張ボードクラス システム変数一覧

変数名	データ型	説明	属性	
			get	put
@INI_PLS	VT_ARRAY| VT_I2	パルス出力関連の初期設定用パラメータを設定/取得. ※1 <要素 1: PulseMode> パルスの出力モードを設定. 0:共通パルス方式 OUT:負論理, DIR+:High, DIR-:Low 1:共通パルス方式 OUT:正論理, DIR+:High, DIR-:Low 2:共通パルス方式 OUT:負論理, DIR+:Low, DIR-:High 3:共通パルス方式 OUT:正論理, DIR+:Low, DIR-:High 4:2 パルス方式:負論理（デフォルト） 5:2 パルス方式:正論理 6:90 度位相差モード OUT:進み信号. DIR:遅れ信号 7:90 度位相差モード OUT:遅れ信号. DIR:進み信号 [<要素 2: DirTimer>] 方向変化時のウェイト挿入. 0:OFF, 1:ON (デフォルト) 注:共通パルス方式の場合のみ有効 [<要素 3: Duty>] デューティ比を設定. 0:パルス出力速度により変化 1:デューティ比 50%固定 デフォルト:DL シリーズ ”1”, DF シリーズ “0” 注:[省略可能要素] 又は “-1“指定時は, デフォルト値.	○	○

@INI_CNT	VT_ARRAY| VT_I2	カウンタの動作の初期設定用パラメータを設定/取得. ※1 <要素 1: ClearCntLtc> LTC 信号が OFF→ON へ変化した時にクリアする カウンタの種類を設定. 0:カウンタをクリアしない（デフォルト） 1:出力パルスカウンタをクリア 2:エンコーダカウンタをクリア 3:出力パルスカウンタおよびエンコーダカウンタをクリア [<要素 2: LtcMode>] LTC 信号入力時にラッチするカウンタの種類を設定. 0:ラッチ機能を使用しない（デフォルト） 1:出力パルスカウンタをラッチ 2:エンコーダカウンタをラッチ 3:出力パルスカウンタおよびエンコーダカウンタをラッチ [<要素 3: ClearCntClr>] CLR 信号が OFF→ON へ変化した時にクリアする カウンタの種類を設定. 0:カウンタをクリアしない（デフォルト） 1:出力パルスカウンタをクリア 2:エンコーダカウンタをクリア 3:出力パルスカウンタおよびエンコーダカウンタをクリア 注:[省略可能要素] 又は “-1“指定時は, デフォルト値.	○	○
@INI_DIO	VT_ARRAY| VT_I2	制御入出力信号関連の初期設定用パラメータを設定/取得. ※1 <要素 1: IoLog> 制御入出力信号論理を設定. [0 | 0 | 0 | 0 | 0 | OUT3 | OUT2 | OUT1| LIM | IN7 | IN6 | IN5 | IN4 | IN3 | IN2 | IN1] 設定範囲:0～7FF(Hex) 各 bit の設定値: 0 負論理, 1 正論理 デフォルト:0（すべて負論理） [<要素 2:InType>] 制御入力信号形式（汎用入力/ALM, INP, SD, LTC, PCS, CLR）を設定. [0 | 0 | IN6/CLR | IN5/PCS | IN4/LTC | IN3/SD | IN2/INP | IN1/ALM] 設定範囲:0～3F(Hex) デフォルト:1（IN1 をアラーム(ALM)信号入力として使用） [<要素 3:Out1>] [<要素 4:Out2>] [<要素 5:Out3>] 制御出力 OUT1～3 の信号形式を設定. 0:汎用出力（デフォルト） 1:アラームクリア信号 2:偏差カウンタクリア信号（ERC） 3:出力パルスカウンタカウント一致信号（CP1） 4:エンコーダカウンタカウント一致信号（CP2） 5:ホールドオフ信号 注:[省略可能要素] 又は “-1“指定時は, デフォルト値.	○	○

@INI_ENC	VT_ARRAY| VT_I2	エンコーダ関連の初期設定用パラメータを設定/取得. ※1 <要素 1:EncType> エンコーダ入力形式を設定. 0:A/B 1 逓倍(デフォルト), 1:A/B 2 逓倍, 2:A/B 4 逓倍, 3:U/D, 4:使用しない [<要素 2: ErcTime>] 偏差カウンタクリア信号幅を設定. 0:12[μsec](デフォルト), 1:102[μsec], 2:408[μsec], 3:1.6[msec], 4:13[msec], 5:52[msec], 6:104[msec], 7:レベル出力 [<要素 3:ErcOffTimer>] 偏差カウンタクリア信号 OFF タイマ時間を設定. 0:0[μsec](デフォルト), 1:12[μsec], 2:1.6[msec], 3:104[msec] [<要素 4: AlmTime>] アラームクリア信号幅を設定. 0:12[μsec](デフォルト), 1:102[μsec], 2:408[μsec], 3:1.6[msec], 4:13[msec], 5:52[msec], 6:104[msec] [<要素 5:ErcMode>] ERC 信号自動出力の設定を指定. [0 | 0 | 0 | 0 | 0 | 0 | bit1 | bit0] 設定範囲:0～3(Hex) bit0 0:LIM、ALM 信号入力による停止時に ERC 信号を出力しない 1:LIM、ALM 信号入力による停止時に ERC 信号を自動出力 bit1 0:原点復帰動作完了時に ERC 信号を出力しない 1:原点復帰動作完了時に ERC 信号を自動出力 デフォルト: 0 注:[省略可能要素] 又は “-1“指定時は, デフォルト値.	○	○

@INI_ORG	VT_ARRAY| VT_I2	原点復帰関連の初期設定用パラメータを設定/取得. ※1 <要素 1: OrgLog> 原点入力論理とエッジを設定 0:立下りエッジ, 負論理(デフォルト) 1:立下りエッジ, 正論理 2:立上りエッジ, 負論理 3:立上りエッジ, 正論理 [<要素 2: LimitTurn>] 原点復帰動作中のリミット反転の有無を設定. 0:リミット反転しない 1:リミット反転する(デフォルト) [<要素 3: OrgType>] Z 相の使用有無を設定します. 0:使用しない(ORG のみ)(デフォルト) 1:使用する(ORG+Z 相) 注:"0"設定時、ZCount 設定は無効. [<要素 4: EndDir>] 原点復帰時の原点突入方向(原点復帰終了方向). 0:未指定(デフォルト) 1:正方向(CW) 2:負方向(CCW) [<要素 5: ZCount>] 原点復帰時の Z 相の数を設定. 設定範囲: 1～16 デフォルト: 1 注: "0"設定時にはこの設定は無効です. OrgType は"0"[使用しない]に自動的に変更される. 注:[省略可能要素] 又は "-1"指定時は, デフォルト値.	○	○
@INI_EXT	VT_ARRAY| VT_I2	その他の初期設定用パラメータを設定/取得. ※1 <要素 1: SAccelType> S 字加減速の使用有無を設定. 0:使用しない(デフォルト), 1:使用する [<要素 2: SdMode>] SD 信号入力時の動作を設定. 0:減速停止(デフォルト) 1:減速のみ(開始速度で定速動作します) 2:減速停止 + SD 信号ラッチ 3:減速 + SD 信号ラッチ [<要素 3: FilterType >] 入力フィルタ特性を設定. 0:フィルタを挿入しない(デフォルト) 1:3.2[μsec], 2:25[μsec], 3:200[μsec], 4:1.6[msec] 注:[省略可能要素] 又は "-1"指定時は, デフォルト値.	○	○

@MV_RDY	VT_ARRAY| VT_I2	基本動作の開始準備と モータ動作タイプ/動作開始方向を取得. ※1 <要素 1: MotionType> モータ動作タイプの設定 0:動作なし, 1: PTP 動作, 2 :JOG 動作, 3:原点復帰動作, 6: Z 相カウント動作 <要素 2: StartDir> モータ動作の開始方向の取得 0:正方向(CW), 1:負方向(CCW) 注: 本変数は取得専用です. 設定は MOV 系コマンド実行時に自動的に行われます.	○	―
@MV_CHGRDY	VT_I2	モータ動作変更タイプを取得※1 -モータ動作変更タイプ- 0: 瞬時に開始速度(FL 速度)へ変更 1: 瞬時に目標速度(FH 速度)へ変更 2: 減速して開始速度(FL 速度)へ変更 3: 加速して目標速度(FH 速度)へ変更 4: 動作速度と加減速度を変更 5: モータ停止位置を変更 6 : PCS 信号によるモータ停止位置変設定 注: 本変数は取得専用です. 設定はMCHGコマンド実行時に自動的に行われます.	○	―
@MV_STSPD	VT_R8	パルス出力開始速度を設定/取得※1 単位:PPS デフォルト(100) 動作速度設定範囲: SMC-4/8DL シリーズ の場合: 0.2929687～9829800 SMC-4/8DF シリーズ の場合: 0.073242187～6553500 注: “-1“指定時は, デフォルト値.	○	○
@MV_TGSPD	VT_R8	パルス出力目標速度を設定/取得※1 単位:PPS デフォルト(1000) 動作速度設定範囲: SMC-4/8DL シリーズ の場合: 0.2929687～9829800 SMC-4/8DF シリーズ の場合: 0.073242187～6553500 注: “-1“指定時は, デフォルト値.	○	○
@MV_ACCTM	VT_R8	加速時間を設定/取得※1 単位:ms デフォルト(50) 0:加速を行わず、瞬時に目標速度に設定 注: “-1“指定時は, デフォルト値.	○	○
@MV_DECTM	VT_R8	減速時間を設定/取得※1 単位:ms デフォルト(50) 0: @MV_ACCTM で設定した値を減速時間として使用設定 注: “-1“指定時は, デフォルト値.	○	○

@MV_RESPD	VT_R8	速度分解能を設定/取得※1 単位:PPS デフォルト(1) 速度分解能設定範囲: SMC-4/8DL シリーズの場合: -1, 0.2929687～600 SMC-4/8DF シリーズの場合: -1, 0.073242187～100 注: “-1“指定時は, デフォルト値.	○	○
@MV_SFSPD	VT_R8	S 字区間を設定/取得※1 単位:PPS デフォルト(400) 0:直線区間のない S 字動作を行います(最大の S 字動作). 注: “-1“指定時は, デフォルト値. 本機能は@INI_EXT で, S 字加減速を「1: 使用する」と設定している場合に有効.	○	○
@MV_STPPOS	VT_ARRAY| VT_I4	モータの停止位置(総出力パルス数)を取得 <要素 1: StopPositionAbs > 絶対座標の停止位置 絶対座標範囲: -134,217,728 ～ +134,217,727 <要素 2: StopPositionRel> 相対座標の停止位置 相対座標範囲: -134,217,728 ～ +134,217,727 (0 を除く) 注: 本変数は取得専用です. 設定は“MOVP”又は“MCHG”コマンド実行時に自動的に設定されます.	○	―
@MV_ZCNT	VT_ARRAY| VT_I2	Z 相カウント動作の動作設定/取得. ※1 <要素 1: ZMoveCount> Z 相カウント動作での Z 相カウント数を設定 デフォルト(1) 設定範囲:1～16 [<要素 2: ZLog>] Z 相信号の入力信号論理設定 0:立下りエッジ(デフォルト), 1:立上りエッジ 注:[省略可能要素] 又は “-1“指定時は, デフォルト値. 本機能は SMC-4/8DL シリーズ では使用できません.	○	○
@CNT_PLS	VT_I4	フィードバック出力パルス数※1を取得/設定します. 設定可能範囲: -134,217,728 ～+134,217,727	○	○
@CNT_ENC	VT_I4	エンコーダのカウント値※1を取得/設定します. 設定可能範囲: -134,217,728 ～+134,217,727	○	○
@CNT_ENZ	VT_I2	原点復帰動作に伴う Z 相カウント動作時の Z 相カウント数, もしくは Z 相カウント動作時の Z 相カウント数※1を取得.	○	―
@LTC_PLS	VT_I4	LTC 信号入力によりラッチされたフィードバック出力パルス数※1 を取得します.	○	―
@LTC_ENC	VT_I4	LTC 信号入力によりラッチされたエンコーダカウンタ値※1 を取得します.	○	―

@STS_PLS	VT_I2	パルス出力状態※1を取得します. 0:パルス出力停止中 1:開始速度(FL 速度)で定速動作中 2:目標速度(FH 速度)で定速動作中 3:同期スタート待ち 4:ERC タイマ完了待ち 5:方向変化タイマ完了待ち 6:加速動作中 7:減速動作中 8:INP 入力待ち状態	○	―
@STS_MOV	VT_I2	モータの動作状態※1を取得します. 0:停止中 1:PTP 動作中 2:JOG 動作中 3:原点復帰動作中 4:バンク動作中(シングル) 5:バンク動作中(ループ)	○	―
@STS_STP	VT_I2	モータの停止要因※1を取得します. 0: 動作中 1: 停止コマンド 2: 減速停止コマンド 3: 他軸の停止 4: アラーム/緊急停止信号 5: 正方向リミット停止信号 6: 負方向リミット停止信号 7: 減速停止信号 255:動作関数の終了	○	―
@STS_LMT	VT_I2	リミット状態※1を取得します. bit7 [0 | 0 | 0 | SD | ORG | -LIM | +LIM | ALM] bit0 ALM アラーム/緊急停止リミット +LIM 正方向リミット -LIM 負方向リミット ORG 原点リミット SD 減速停止リミット 設定値 0 無効, 1 有効	○	―
@STS_SPD	VT_R8	動作中のパルス出力速度を取得※1を取得します.	○	―

@LMT_MASK	-put 時- VT_ARRAY| VT_I2 -get 時- VT_I2	リミット信号の有無を設定/取得. ※1 <要素 1: LimitMask> ・・・ put/get 共使用 リミット信号の有無を設定/取得 [0 | 0 | 0 | 0 | 0 | 0 | ALM | SD] -信号の意味- SD 減速停止リミット ALM アラーム/緊急停止リミット -設定/取得値- 0:リミット有効, 1:リミット無効 <要素 2: LimitMaskEnable> ・・・ put 時のみ使用 [0 | 0 | 0 | 0 | 0 | 0 | ALM | SD] -信号の意味- SD 減速停止リミット ALM アラーム/緊急停止リミット -設定値- 0 変更しない, 1 変更する	○	○
@HOLD_OFF	VT_I2	ホールドオフ信号を設定/取得. ※1 0 軸をホールド 1 軸のホールドを解除 注: @INI_DIO Out1～3 のどれかの設定に「ホールドオフ信号」が設定されている必要があります.	○	○
@ALM_CODE	VT_I2	アラームコードを取得. ※1 取得する値は、IN5～IN7の状態を示しています. ドライバユニットからのアラームコードアウト出力が存在すれば ドライバユニットとボードの IN5～IN7を接続してください. アラームコードの詳細については各モータドライバユニットの解説書を参照してください.	○	―
@SIG_DO	-put 時- VT_ARRAY| VT_I2 -get 時- VT_I2	汎用出力信号データ※1を取得/設定します. <要素 1: OutData > ・・・ put/get 共使用 汎用出力信号データを設定. [0 | 0 | 0 | 0 | 0 | OUT3 | OUT2 | OUT1] 1:OUT 1 ビット, 2:OUT 2 ビット, 4:OUT 3 ビット <要素 2: OutDataEnable > ・・・ put 時のみ使用 汎用出力信号において, データを変更するビットを設定. @INI_DIO で OUT1～3 の信号形式を汎用出力に設定したビットのみを 1 に設定可能です. [0 | 0 | 0 | 0 | 0 | OUT3 | OUT2 | OUT1] 0:無効, 1:有効	○	○
@SIG_DI	VT_I2	汎用入力信号データ※1を取得します. [0 | IN7 | IN6 | IN5 | IN4 | IN3 | IN2 | IN1] 0:信号 OFF, 1:信号 ON	○	―

※1:詳細については, API-SMC（WDM）Help を参照して下さい.

2.5. エラーコード

SMC プロバイダでは，固有のエラーコードとして以下の2種類があります。

1) SMC API が返すエラー

SMC API が返すエラー番号を“0x8010****”でマスクした値を返します．

例）SMC API のエラー:0xFFFF → SMC API のエラー:0x8010FFFF

SMC API の詳細については，CONTEC 社 API-SMC(WDM) Help を参照してください．

2) SMC プロバイダ独自が返すエラー

表 2-9 SMC プロバイダ独自エラーコード一覧

エラー名	エラー番号	説明
E_PROV_CANCEL	0x80180001	プロバイダキャンセルによる動作停止．

3) ORiN2 共通エラー

「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．

2.6. CAO-SMC API 対応表

SMC プロバイダは，コマンドの実行方法として CaoExtension::Execute，CaoVariable による 2 通りの方法を提供しています．CaoExtension::Execute メソッドは，動作を行う API 関数を実行します．

CaoVariable は，値の設定/取得を行う API 関数を実行します．

表 2-10 コントローラクラス，拡張ボードクラス，変数クラスと SMC API 対応表

CAO API		SMC API※
クラス::メッソド名	パラメータ名／コマンド名／変数名	
CaoWorkspace::AddController()	DeviceName	SmcWInit()
CaoWorkspaces::Remove()	-	SmcWExit()
CaoController::AddExtension ()	Axis?	SmcWGet 系の全関数 SmcWSetInitParam()
CaoExtension::Execute()	STOP	SmcWMotionStop()
	DSTP	SmcWMotionDecStop()
	ALMCLR	SmcWSetAlarmClear()
	ERCOUT	SmcWSetErcOut()
	ORG	SmcWSetReady() SmcWMotionStart()
	MOVP	SmcWSetStopPosition() SmcWSetReady() SmcWMotionStart()
	MOVJ	SmcWSetReady() SmcWMotionStart()

	MCHG	SmcWSetMotionChangeReady() SmcWMotionChange()
CaoVariable::get_Value()	@INI_PLS	SmcWGetPulseType() SmcWGetPulseDuty()
	@INI_CNT	SmcWGetCounterMode()
	@INI_DIO	SmcWGetCtrlInOutLog() SmcWGetCtrlTypeIn() SmcWGetCtrlTypeOut()
	@INI_ENC	SmcWGetEncType() SmcWGetErcAlmClearTime() SmcWGetErcMode()
	@INI_ORG	SmcWGetOrgLog() SmcWGetOrgMode()
	@INI_EXT	SmcWGetSAccelType() SmcWGetSDMode() SmcWGetInFilterType()
	@MV_RDY	SmcWGetReady()
	@MV_CHGRDY	SmcWGetMotionChangeReady()
	@MV_STSPD	SmcWGetStartSpeed()
	@MV_TGSPD	SmcWGetTargetSpeed()
	@MV_ACCTM	SmcWGetAccelTime()
	@MV_DECTM	SmcWGetDecelTime()
	@MV_RESPD	SmcWGetResolveSpeed()
	@MV_SFSPD	SmcWGetSSpeed()
	@MV_STPPOS	SmcWGetStopPosition()
	@MV_ZCNT	SmcWGetZCountMotion()
	@CNT_PLS	SmcWGetOutPulse()
	@CNT_ENC	SmcWGetCountPulse()
	@CNT_ENZ	SmcWGetZCount()
	@LTC_PLS	SmcWGetLatchOutPulse()
	@LTC_ENC	SmcWGetLatchCountPulse()
	@STS_PLS	SmcWGetPulseStatus()
	@STS_MOV	SmcWGetMoveStatus()
	@STS_STP	SmcWGetStopStatus()
	@STS_LMT	SmcWGetLimitStatus()
	@STS_SPD	SmcWGetMoveSpeed()
	@LMT_MASK	SmcWGetLimitMask()
	@HOLD_OFF	SmcWGetHoldOff()
	@ALM_CODE	SmcWGetAlarmCode()
	@SIG_DO	SmcWGetDigitalOut()
	@SIG_DI	SmcWGetDigitalIn()
	@ERROR	―
CaoVariable::put_Value()	@INI_PLS	SmcWSetPulseType() SmcWSetPulseDuty()
	@INI_CNT	SmcWSetCounterMode()
	@INI_DIO	SmcWSetCtrlInOutLog() SmcWSetCtrlTypeIn() SmcWSetCtrlTypeOut()
	@INI_ENC	SmcWSetEncType() SmcWSetErcAlmClearTime() SmcWSetErcMode()

	@INI_ORG	SmcWSetOrgLog() SmcWSetOrgMode()
	@INI_EXT	SmcWSetSAccelType() SmcWSetSDMode() SmcWSetInFilterType()
	@MV_RDY	―
	@MV_CHGRDY	―
	@MV_STSPD	SmcWSetStartSpeed()
	@MV_TGSPD	SmcWSetTargetSpeed()
	@MV_ACCTM	SmcWSetAccelTime()
	@MV_DECTM	SmcWSetDecelTime()
	@MV_RESPD	SmcWSetResolveSpeed()
	@MV_SFSPD	SmcWSetSSpeed()
	@MV_STPPOS	SmcWSetStopPosition()
	@MV_ZCNT	SmcWSetZCountMotion()
	@CNT_PLS	SmcWSetOutPulse()
	@CNT_ENC	SmcWSetCountPulse()
	@CNT_ENZ	―
	@LTC_PLS	―
	@LTC_ENC	―
	@STS_PLS	―
	＠STS_MOV	―
	＠STS_STP	―
	＠STS_LMT	―
	＠STS_SPD	―
	@LMT_MASK	SmcWSetLimitMask()
	@HOLD_OFF	SmcWSetHoldOff()
	@ALM_CODE	―
	＠SIG_DO	SmcWSetDigitalOut
	＠SIG_DI	―
	＠ERROR	―

※SMC API の詳細については，CONTEC 社 API-SMC（WDM）Help を参照して下さい.

3. サンプルプログラム

SMC ボード 軸番号 1 の原点復帰を行い，原点復帰が完了しているか確認後，ポイントの 1 番に移動させるコードを示します．

List 3-1 Sample.frm

```
Dim Eng As CaoEngine
Dim Ctrl As CaoController
Dim ExtAxis As CaoExtension
Dim VarIniPls As CaoVariable
Dim VarMov As CaoVariable
Dim VarSpd As CaoVariable

Private Sub Form_Load()

        ‘ CAO エンジンの生成
        Set Eng = New CaoEngine

        ‘ SMC への接続
        Set Ctrl = Eng.Workspaces(0).AddController(“Sample”, _
                                    ”CaoProv.CONTEC.SMC”, ””, ”DeviceName=SMC000”)

        ‘ 軸番号 1 の拡張ボードクラス生成
        Set ExtAxis = Ctrl.AddExtension(“Axis1”)

        ‘ 各変数の生成
        Set VarIniPls = ExtAxis.AddVariable(“@INI_PLS“)‘パルス出力設定関連のシステム変数生成
        Set VarMov = ExtAxis.AddVariable(“@STS_MOV“)    ‘ 動作状態確認用のシステム変数生成
        Set VarSpd = ExtAxis.AddVariable(“@MV_TGSPD“)   ‘ 目標速度設定用システム変数生成

        ‘モータ(ドライバ)関連の初期設定
        VarIniPls = Array(5)

        ‘ CCW 方向へ原点復帰を行います
        ExtAxis.Execute “ORG”, 1

End Sub

Private Sub Command1_Click()

        ‘ 原点復帰が未了の間待ち
        Do While ( VarMov <> 0 )
            DoEvents
        Loop

        VarSpd = 500#                                   ‘目標速度を 500PPS に設定
        ExtAxis.Execute “MOVP”, Array(1, 5000)          ‘相対座標で+(CW)方向へ 5000 パルス分移動

        ‘ 動作中は待ち
        Do While ( VarMov <> 0 )
            DoEvents
        Loop

End Sub
```