AIO プロバイダ

CONTEC AIO ボード

Version 1.0.6

ユーザーズ ガイド

December 21, 2021

【備考】

【改版履歴】

バージョン	日付	内容
1.0.0.0	2011-7-12	初版.
1.0.1.0	2012-5-29	メタモード追加.
1.0.1	2012-7-17	ドキュメントのバージョンルールを変更.
1.0.2	2013-2-7	アナログ出力レンジの設定/取得変数追加. AIO API が返すエラーコード追加.
1.0.3	2014-4-10	Addcontroller のオプション追加.
1.0.4	2017-6-7	アナログ入力サンプリング機能追加.
1.0.5	2017-9-5	アナログ入力サンプリング機能マルチプロセス時の OnMessage 対応追加.
	2017-10-17	誤記修正
1.0.6	2021-12-21	エラー誤検知を修正しました.

【対応機器】

機種	バージョン	注意事項
AIO-160802L-LPE		
ADI16-4(FIT)		USB 搭載 I/O コントロールモジュール(CPU-CA10(USB))と接続する必要がある.
DAI16-4(FIT)		USB 搭載 I/O コントロールモジュール(CPU-CA10(USB))と接続する必要がある.
ADI12-16(PCI)		
AIO121601E3-PE		
AIO121601M-PCI		
ADA-16-32/2(PCI)F		
ADI16-4(USB)		
AIO-121602AH-PCI		
AIO-160802AY-USB		
AIO-163202F-PE		

目次

1. はじめに

本書は，CONTEC 製 AIO ボードにアクセスするためのプロバイダである，AIO 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-AIO（WDM）Help（PCI 時） 又は API-USBP（W32）Help（USB 時）を参照して下さい.

> 注意：AIO プロバイダを使用するには，AIO ボードの AIO 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. 対象機器が PCI ボードであれば API-PAC（W32），USB であれば API-USBP（WDM）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

AIO 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 2-6 を参照してください.

表 2-1 AIO プロバイダ

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

2.2. メソッド・プロパティ

2.2.1. CaoWorkspace::AddController メソッド

AIO プロバイダでは Controller オブジェクトの生成時に AIO 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.AIO".
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※1 デフォルト:""(指定なし) ""指定なし場合, 初めに検出した使用可能なデバイスと接続. 注: AIO ボード ID に対応したデバイス名を指定. ※1

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

ScanCount=[<リトライ回数>]	リトライ回数（範囲：0～32767） デフォルト: 4 回 DeviceName オプション：“”(指定なし)の場合，検出可能デバイスとの接続が失敗した場合のリトライ回数.
Interval=[<デジタル入力状態変化サンプリング周期>]※2	デジタル入力状態変化サンプリング周期（範囲:0～65535） デフォルト: 0（オフ） デジタル入力※1 バイト値が変化した場合に OnMessage イベントを取得したい場合にそのサンプリング周期(ms)を指定します.
Mask=[<マスク値>]※2	マスク値（範囲：0～255） デフォルト: 255（マスクなし） Interval オプションが有効な時に，入力バイト値をマスクして不要なイベント発生を抑制します.
Coexistence=[<共存>]	共存設定 無効に設定した場合，他のプロセスが既に同一デバイスに接続している場合に，稀(デバイス機種，ドライババージョンの組み合わせ等により)に失敗(エラー)するケースがあります．そのため，本プロバイダでは省略時(デフォルト)“有効”としています. False：無効 True：有効(デフォルト) 注：デバイス動作中は無視(無効)されます. ※1 マルチプロセス環境でご利用の際は，上記エラー等による実害がない限り，“無効” を推奨します.
ResetDevice=[<デバイスリセット>]	デバイスのリセット，ドライバの初期化 False：無効 True：有効(デフォルト) 注：ドライバ内のパラメータはすべて初期値に戻ります．デバイスが動作中の場合強制停止されます. ※1 マルチプロセス環境でご利用の際は，“無効” を推奨します.
AiInputMethod=[<AI 入力方式>]	アナログ入力方式※3 0：シングルエンド入力 1：差動入力 省略時(デバイス初期値)： 　デバイスの種類やジャンパ（JP）の設定により異なります※1
AiChannels=[<AI 使用チャネル数>]	アナログ入力使用チャネル数※3 範囲：1 ～ 最大チャネル数 省略時(デバイス初期値)：1※1
AiRangeAll=[< AI レンジ全チャネル設定>]	アナログ入力レンジ全チャネル設定※4 設定範囲：デバイスにより設定できる範囲は異なります※1 省略時(デバイス初期値)： 　デバイスの種類によって異なります. ※1 参考：このオプションは，コントローラクラスシステム変数“@RANGE_AI” でも設定可能です.
AiMemoryType=[<AI メモリ形式>]	アナログ入力データ格納用メモリ形式設定※3 0：FIFO 1：RING 省略時(デバイス初期値)：FIFO※1 注：本プロバイダでは変換データの転送方式は，デバイスの起動初期値である “デバイスバッファモード”のみサポートしているため，“ユーザバッファ

	モード”時のメモリ形式は設定できません.
AiClockType=[<AI クロック種類>]	アナログ入力データクロックの種類を設定※1※3 0: 内部クロック 1: 外部クロック 10: イベントコントローラ出力 20: CH0 比較カウント一致 0 (AIO-121601M-PCI のみ設定可) 21: CH1 比較カウント一致 0 (AIO-121601M-PCI のみ設定可) 22: CH0 比較カウント一致 1 (AIO-121601M-PCI のみ設定可) 23: CH1 比較カウント一致 1 (AIO-121601M-PCI のみ設定可) 24: CH0 カウントアップ (AIO-121601M-PCI のみ設定可) 25: CH1 カウントアップ (AIO-121601M-PCI のみ設定可) 26: CH0 カウントダウン (AIO-121601M-PCI のみ設定可) 27: CH1 カウントダウン (AIO-121601M-PCI のみ設定可) 28: CH0 カウンタクリア (AIO-121601M-PCI のみ設定可) 29: CH1 カウンタクリア (AIO-121601M-PCI のみ設定可) 30: キャリーボロー (AIO-121601M-PCI のみ設定可) 31: タイマー (AIO-121601M-PCI のみ設定可) 省略時(デバイス初期値):内部クロック※1 [説明] ・クロックとして内部クロックを使用する場合, ”AiSamplingClock”オプションでサンプリング速度の指定が可能です. ※1 ・クロックとしてイベントコントローラ出力を使用する場合”SetEcuSignal”コマンドを実行する必要があります. ※1 -マルチファンクションデバイス AIO-121601M-PCI ご利用の場合- ・クロックを比較カウント一致に設定する場合, ”SetCntmNotifyCountUp”コマンドを実行する必要があります. ※1 ・クロックをカウントアップ・カウントダウンに設定する場合, ”StartCntmCount”を実行する必要があります. ※1 ・クロックをカウントクリアに設定する場合, ”SetCntmZeroClearCount”コマンドを実行する必要があります. ※1 ・クロックをキャリーボローに設定する場合, ”SetCntmNotifyCarryBorrow”コマンドを実行する必要があります. ※1 ・クロックをタイマーに設定する場合, ”SetCntmNotifyTimer”コマンドを実行する必要があります. ※1
AiSamplingClock=[<AI 変換速度>]	アナログ入力 内部クロック使用時のデータ変換速度設定※3 単位:μsec 設定範囲:デバイスにより設定できる範囲は異なります※1 省略時(デバイス初期値):デバイスによって異なります※1 ※”AiClockType”にて, 内部クロックの設定時に, 変換速度の指定が可能です. 内部クロックを使用しない場合には指定する必要はありません.
AiClockEdge=[<AI 入力タイミング>]	アナログ入力 外部クロック使用時の入力タイミング設定 0:立ち下がりエッジ 1:立ち上がりエッジ 省略時: 0(デフォルト) ※”AiClockType”にて, 外部クロックの設定時に, 立ち下がりエッジ/立ち上がりエッジのどちらを入力タイミングとするかの指定が可能です. 外部クロックを使用しない場合には指定する必要はありません.

AiAutoSampling= <Start>[:<SamplingTimes>][:<Event>[:<DataType>]]	アナログ入力自動サンプリング変換開始設定 AddController による接続直後からのサンプリングデータの AD 変換を各パラメータの条件で自動で開始するかどうかを設定します. 角括弧(“[ ]”)内は省略可能を示します. <Start> 1: 自動変換開始する 0: 自動変換開始しない 省略時: 0(デフォルト) 以降のパラメータは, <Start>値が ”1” の時のみ有効となります. <SamplingTimes> 指定サンプリング回数値を設定します. 内部的には, “SetAiEventSamplingTimes”コマンドを実行します. 省略時: 実行しない(デフォルト) <Event> CaoController::OnMesseage のイベント要因を 10 進数または 16 進数で設定します. 例) 指定サンプリング回数格納イベント 16 進数表記の場合: “0x80” または “0X80” 10 進数表記の場合:”128” 内部的には, “SetAiEventConditions”コマンド(P21 参照)を実行します. 省略時: 実行しない(デフォルト) 注:イベント要因:0 は実行しません. <DataType> <Event>値が 0 以外の時有効となります. CaoController::OnMesseage のイベント発生時の変換データタイプを設定します. 内部的には, “SetAiEventConditions”コマンド(P21 参照)を実行します. 省略時:0(電圧または電流値) ※本オプションによるサンプリング変換開始／停止条件は以下の様に固定となります. ・変換開始条件: ソフトウエア ・変換停止条件: コマンド

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.
※2:デジタル入力を搭載した機種でのみ有効です. 詳細は, CONTEC 製品マニュアルを参照して下さい.
※3:アナログ入力を搭載した機種でのみ使用可能です. 詳細は, CONTEC 製品マニュアルを参照して下さい.
※4:アナログ入力を搭載した機種で 且つ 関数実行によるレンジ設定が可能な機種でのみ利用可能です.
詳細は, CONTEC 製品マニュアルを参照して下さい.
※5:アナログ出力を搭載した機種でのみ使用可能です. 詳細は, CONTEC 製品マニュアルを参照して下さい.
※6:アナログ出力を搭載した機種で 且つ 関数実行によるレンジ設定が可能な機種でのみ利用可能です.
詳細は, CONTEC 製品マニュアルを参照して下さい.

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 2-3 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::OnMessage イベント

CaoController クラスの OnMessage イベントとしてクライアントにデータを受け渡します. このとき, Message::Value プロパティに受信データをそのまま格納します.

イベント番号	意味	値
1	Interval オプションをオフ(0)以外に設定した場合でデジタル入力※1※2 バイト値が変化した時に, Mask オプションでマスクしてないビットが変化した場合に発生.	デジタル入力バイト値 型:VT_I2
100	アナログ入力 AD 変換開始条件成立※1※3※4	なし
101	アナログ入力リピート終了※1※3※4	リピート回数 型:VT_I4
102	アナログ入力デバイス動作終了※1※3※4	アナログ入力サンプリング変換データ値 データ型は“SetAiEventConditions”コマンド実行時の<DataTpe>が ・電圧または電流値の時: 型 VT_ARRAY | VT_R4 ・バイナリ値の時 : 型 VT_ARRAY | VT_I4 ・取得データなし(イベント通知のみ)の時: VT_EMPTY ※サンプリング回数 0 時は VT_EMPTY となります. 例:使用チャネル数:8, サンプリング回数:n とした場合 ---------- 要素 No | サンプリングデータ ---------- 0 | VT_R4: CH0 変換データ 1 | VT_R4: CH1 変換データ 2 | VT_R4: CH2 変換データ 3 | VT_R4: CH3 変換データ 4 | VT_R4: CH4 変換データ 5 | VT_R4: CH5 変換データ 6 | VT_R4: CH6 変換データ 7 | VT_R4: CH7 変換データ 8 | VT_R4: CH0 変換データ : | : : | : n×8-1 | VT_R4: CH7 変換データ ---------- 注:変換データ値の取得に失敗した場合 エラーコード格納 型:VT_I4
103	アナログ入力指定サンプリング回数格納※1※3※4 使用チャネル数分の指定サンプリングデータをデバイスメモリから取得します. 変換データは,”SetAiEventConditions”コマンドで設定した変換データタイプ(“バイナリ値” か “電圧または電流値”)で格納されます.	
104	アナログ入力オーバーフロー※1※3※4	
105	アナログ入力サンプリングクロックエラー※1※3※4	
106	アナログ入力 AD 変換エラー※1※3※4	

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.

※2:デジタル入力を搭載した機種でのみ有効です. 詳細は, CONTEC 製品マニュアルを参照して下さい.
※3:アナログ入力を搭載した機種でのみ有効です. 詳細は, CONTEC 製品マニュアルを参照して下さい.
※4:イベントを有効にするには, 事前に“SetAiEventCondition”コマンドにて条件設定しておく必要があります.

2.2.4. CaoController::AddVariable メソッド

このメソッドでは, AIO 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)

2.2.5. CaoController::get_VariableNames プロパティ

2.4.1の変数名リストを取得します.

2.2.6. 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, 2.4.1 を参照して下さい.

2.2.7. 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, 2.4.1 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 2-3 CaoController::Execute コマンド一覧

コマンド名	機能	頁
SetAiStartTrigger	アナログ入力サンプリング 開始条件の設定	P.13
SetAiStartLevel	アナログ入力サンプリング レベル比較開始時のレベル設定(バイナリ)	P.14
SetAiStartLevelEx	アナログ入力サンプリング レベル比較開始時のレベル設定(電圧または電流)	P.14
SetAiStartInRange	アナログ入力サンプリング インレンジ比較開始時のレベル設定(バイナリ)	P.15
SetAiStartInRangeEx	アナログ入力サンプリング インレンジ比較開始時のレベル設定(電圧または電流)	P.15
SetAiStartOutRange	アナログ入力サンプリング アウトレンジ比較開始時のレベル設定(バイナリ)	P.15
SetAiStartOutRangeEx	アナログ入力サンプリング アウトレンジ比較開始時のレベル設定(電圧または電流)	P.16
SetAiStopTrigger	アナログ入力サンプリング 停止条件の設定	P.17
SetAiStopTimes	アナログ入力サンプリング 回数の設定	P.18
SetAiStopLevel	アナログ入力サンプリング レベル比較開始時のレベル設定(バイナリ)	P.18
SetAiStopLevelEx	アナログ入力サンプリング レベル比較開始時のレベル設定(電圧または電流)	P.18
SetAiStopInRange	アナログ入力サンプリング インレンジ比較開始時のレベル設定(バイナリ)	P.19
SetAiStopInRangeEx	アナログ入力サンプリング インレンジ比較開始時のレベル設定(電圧または電流)	P.19
SetAiStopOutRange	アナログ入力サンプリング アウトレンジ比較開始時のレベル設定(バイナリ)	P.20
SetAiStopOutRangeEx	アナログ入力サンプリング アウトレンジ比較開始時のレベル設定(電圧または電流)	P.20
SetAiStopDelayTimes	アナログ入力サンプリング 停止遅延回数の設定	P.21
SetAiRepeatTimes	アナログ入力サンプリング リピート回数の設定	P.21
SetAiEventConditions	アナログ入力 イベント発生条件の設定	P.21
SetAiEventSamplingTimes	アナログ入力 指定サンプリング回数格納イベント利用時のサンプリング回数の設定	P.22
StartAiSamplingAsync	アナログ入力サンプリング AD 変換開始(非同期動作)	P.22
StartAiSamplingSync	アナログ入力サンプリング AD 変換開始(同期動作)	P.22
StopAiSampling	アナログ入力サンプリング AD 変換停止	P.23
GetAiSamplingCount	アナログ入力 サンプリング回数の取得	P.23
GetAiStopTriggerCount	アナログ入力 停止トリガ入力時のサンプリング回数の取得	P.23
GetAiSamplingData	アナログ入力 指定サンプリング分のデータを読み込み(バイナリ)	P.23
GetAiSamplingDataEx	アナログ入力 指定サンプリング分のデータを読み込み(電圧または電流)	P.24
GetAiStatus	アナログ入力ステータスを取得	P.24
GetAiResolution	アナログ入力の分解能を取得	P.25
ResetAiMemory	アナログ入力デバイスメモリのリセット	P.25
ResetAiStatus	アナログ入力ステータスのリセット	P.25
SetEcuSignal	イベントコントローラの信号設定	P.26
SetCntmNotifyCountUp	カウンタのカウント一致による通知の指定と比較レジスタの設定	P.26
SetCntmZeroClearCount	カウンタのカウント値のゼロクリア	P.26
SetCntmNotifyCarryBorrow	カウンタのキャリー／ボロー通知の設定	P.26
SetCntmNotifyTimer	タイマー通知の設定	P.27
StartCntmCount	カウンタ動作の開始	P.27
StopCntmCount	カウンタ動作の停止	P.27
PresetCntm	カウンタのプリセット	P.27
ReadCntmCount	カウンタ値の読み込み	P.28
ReadCntmStatusEx	カウンタステータスの読み込み	P.28
SetCntmZMode	カウンタ Z 相使用方法の設定	P.28

2.3.2. CaoController::Execute コマンド詳細

SetAiStartTrigger

構文

object.SetAiStartTrigger(<Trigger>)

引数

<Trigger> = VT_I2: 変換開始条件※1

以下の範囲から設定します. デバイスにより設定できる値は異なります.

0: ソフトウェア
1: 外部トリガ立ち上がり
2: 外部トリガ立ち下がり
3: レベル比較
4: インレンジ比較
5: アウトレンジ比較
10: イベントコントローラ出力
20: CH0 比較カウント一致 0 (AIO-121601M-PCI のみ設定可)
21: CH1 比較カウント一致 0 (AIO-121601M-PCI のみ設定可)
22: CH0 比較カウント一致 1 (AIO-121601M-PCI のみ設定可)
23: CH1 比較カウント一致 1 (AIO-121601M-PCI のみ設定可)
24: CH0 カウントクリア (AIO-121601M-PCI のみ設定可)
25: CH1 カウントクリア (AIO-121601M-PCI のみ設定可)
26: キャリー ボロー (AIO-121601M-PCI のみ設定可)
27: タイマー (AIO-121601M-PCI のみ設定可)

戻り値

なし

説明

アナログ入力 AD 変換サンプリング開始条件を設定します. ※1

・変換開始条件をレベル比較に設定する場合, "SetAiStartLevel"または"SetAiStartLevelEx" コマンドでレベル比較開始の設定を行ってください.
・変換開始条件をインレンジ比較に設定する場合, "SetAiStartInRange"または"SetAiStartInRangeEx"コマンドでインレンジ比較開始の設定を行ってください.
・変換開始条件をアウトレンジ比較に設定する場合, "SetAiStartOutRange"または"SetAiStartOutRangeEx"コマンドでアウトレンジ比較開始の設定を行ってください.
・クロックとしてイベントコントローラ出力を使用する場合, "SetEcuSignal"コマンドでイベントコントローラの接続を行ってください.

[マルチファンクションデバイス AIO-121601M-PCI をご利用の方へ] ※1

・変換開始条件を比較カウント一致に設定する場合
"CntmNotifyCountUp"コマンドで比較カウント一致発生条件の設定を行ってください.
・変換開始条件をカウントクリアに設定する場合
"CntmZeroClearCount"コマンドでカウントクリアを行ってください.
・変換開始条件をキャリーボローに設定する場合
"CntmNotifyCarryBorrow"コマンドでキャリーボロー発生条件の設定を行ってください.
・変換開始条件をタイマーに設定する場合
"CntmNotifyTimer"コマンドでタイマーイベント発生条件の設定を行ってください.

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.

SetAiStartLevel

構文 *object*.SetAiStartLevel(<Channel>, <Level>, <Direction>)

引数 **<Channel>** = VT_I2: レベル比較を行う AI チャネル番号※1

<Level> = VT_I4: レベル比較を行うデータ※1
以下の範囲からバイナリ値で設定します. デバイスにより設定できる値は異なります.
12 ビット分解能のデバイス: 0～4095
16 ビット分解能のデバイス: 0～65535

<Direction> = VT_I2:レベル比較の方向※1
以下の範囲から設定します.
0: 両方
1: 立ち上がり
2: 立ち下がり

戻り値 なし

説明 アナログ入力 AD 変換サンプリング時のレベル比較開始時のレベルをバイナリ値で設定します. ※1
この設定は, "SetAiStartTrigger"コマンドで変換開始条件をレベル比較に設定した場合に必要になります.
※変換開始条件がレベル比較以外の場合には実行する必要はありません.

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.

SetAiStartLevelEx

構文 *object*.SetAiStartLevelEx(<Channel>, <Level>, <Direction>)

引数 **<Channel>** = VT_I2: レベル比較を行う AI チャネル番号※1

<Level> = VT_R4: レベル比較を行うデータ※1
電圧または電流で設定します.

<Direction> = VT_I2:レベル比較の方向※1
以下の範囲から設定します.
0: 両方
1: 立ち上がり
2: 立ち下がり

戻り値 なし

説明 アナログ入力 AD 変換サンプリング時のレベル比較開始時のレベルを電圧または電流で設定します. ※1
この設定は, "SetAiStartTrigger"コマンドで変換開始条件をレベル比較に設定した場合に必要になります.
※変換開始条件がレベル比較以外の場合には実行する必要はありません.

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい. .

SetAiStartInRange

構文 *object*.SetAiStartInRange(<Channel>, <Level1>, <Level2>, <StateTimes>)

引数

<Channel> = VT_I2: インレンジ比較を行う AI チャネル番号※1

<Level1>, <Level2> = VT_I4: インレンジ比較を行う範囲※1
以下の範囲からバイナリ値で設定します. デバイスにより設定できる値は異なります.
12 ビット分解能のデバイス: 0～4095
16 ビット分解能のデバイス: 0～65535

<StateTimes> = VT_I4: 状態保持回数※1
インレンジ比較成立後, 状態を保持する期間をサンプリング回数で設定します.
実際の変換は, StateTimes 分のサンプリング時間が経過した後に開始します.

戻り値 なし

説明 アナログ入力 AD 変換サンプリング時のインレンジ比較開始時のレベルをバイナリ値で設定します. ※1
この設定は, "SetAiStartTrigger"コマンドで変換開始条件をインレンジ比較に設定した場合に必要になります.
※変換開始条件がインレンジ比較以外の場合には実行する必要はありません.

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.

SetAiStartInRangeEx

構文 *object*.SetAiStartInRangeEx(<Channel>, <Level1>, <Level2>, <StateTimes>)

引数

<Channel> = VT_I2: インレンジ比較を行う AI チャネル番号※1

<Level1>, <Level2> = VT_R4: インレンジ比較を行う範囲※1
電圧または電流で設定します.

<StateTimes> = VT_I4: 状態保持回数※1
インレンジ比較成立後, 状態を保持する期間をサンプリング回数で設定します.
実際の変換は, StateTimes 分のサンプリング時間が経過した後に開始します.

戻り値 なし

説明 アナログ入力 AD 変換サンプリング時のインレンジ比較開始時のレベルを電圧または電流で設定します. ※1
この設定は, "SetAiStartTrigger"コマンドで変換開始条件をインレンジ比較に設定した場合に必要になります.
※変換開始条件がインレンジ比較以外の場合には実行する必要はありません.

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.

SetAiStartOutRange

構文 *object*.SetAiStartOutRange(<Channel>, <Level1>, <Level2>, <StateTimes>)

引数

<Channel> = VT_I4: アウトレンジ比較を行う AI チャネル番号※1

<Level1>, <Level2> = VT_I4:アウトレンジ比較を行う範囲※1
以下の範囲からバイナリ値で設定します. デバイスにより設定できる値は異なります.
12 ビット分解能のデバイス: 0～4095
16 ビット分解能のデバイス: 0～65535

<StateTimes> = VT_I4: 状態保持回数※1
アウトレンジ比較成立後, 状態を保持する期間をサンプリング回数で設定します.
実際の変換は, StateTimes 分のサンプリング時間が経過した後に開始します.

戻り値
なし

説明
アナログ入力 AD 変換サンプリング時のアウトレンジ比較開始時のレベルをバイナリ値で設定します. ※1
この設定は, "SetAiStartTrigger"コマンドで変換開始条件をアウトレンジ比較に設定した場合に必要になります.
※変換開始条件がアウトレンジ比較以外の場合には実行する必要はありません.

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.

SetAiStartOutRangeEx

構文
object.SetAiStartOutRangeEx(<Channel>, <Level1>, <Level2>, <StateTimes>)

引数
<Channel> = VT_I4: アウトレンジ比較を行う AI チャネル番号※1

<Level1>, <Level2> = VT_R4:アウトレンジ比較を行う範囲※1
電圧または電流で設定します.

<StateTimes> = VT_I4: 状態保持回数※1
アウトレンジ比較成立後, 状態を保持する期間をサンプリング回数で設定します.
実際の変換は, StateTimes 分のサンプリング時間が経過した後に開始します.

戻り値
なし

説明
アナログ入力 AD 変換サンプリング時のアウトレンジ比較開始時のレベルを電圧または電流で設定します. ※1
この設定は, "SetAiStartTrigger"コマンドで変換開始条件をアウトレンジ比較に設定した場合に必要になります.
※変換開始条件がアウトレンジ比較以外の場合には実行する必要はありません.

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.

SetAiStopTrigger

構文

object.SetAiStopTrigger(<Trigger>)

引数

<Trigger> = VT_I2: 変換停止条件※1

以下の範囲から設定します. デバイスにより設定できる値は異なります.

0: 設定回数変換終了
1: 外部トリガ立ち上がり
2: 外部トリガ立ち下がり
3: レベル比較
4: コマンド(“StopAiSampling”)
5: インレンジ比較
6: アウトレンジ比較
10: イベントコントローラ出力
20: CH0 比較カウント一致 0 (AIO-121601M-PCI のみ設定可)
21: CH1 比較カウント一致 0 (AIO-121601M-PCI のみ設定可)
22: CH0 比較カウント一致 1 (AIO-121601M-PCI のみ設定可)
23: CH1 比較カウント一致 1 (AIO-121601M-PCI のみ設定可)
24: CH0 カウントクリア (AIO-121601M-PCI のみ設定可)
25: CH1 カウントクリア (AIO-121601M-PCI のみ設定可)
26: デジタルフィルタエラー (AIO-121601M-PCI のみ設定可)
27: 異常入力エラー (AIO-121601M-PCI のみ設定可)
28: キャリーボロー (AIO-121601M-PCI のみ設定可)
29: タイマー (AIO-121601M-PCI のみ設定可)

戻り値

なし

説明

アナログ入力 AD 変換サンプリング停止条件を設定します. ※1

・変換停止条件を設定回数変換終了に設定する場合, ”SetAiStopTimes”コマンドで サンプリング回数の設定を行ってください.
・変換停止条件をレベル比較に設定する場合, ”SetAiStopLevel”または”SetAiStopLevelEx” コマンドでレベル比較停止の設定を行ってください.
・変換停止条件をインレンジ比較に設定する場合, ”SetAiStopInRange”または”SetAiStopInRangeEx”コマンドでインレンジ比較停止の設定を行ってください.
・変換停止条件をアウトレンジ比較に設定する場合, ”SetAiStopOutRange”または”SetAiStopOutRangeEx”コマンドでアウトレンジ比較停止の設定を行ってください.
・クロックとしてイベントコントローラ出力を使用する場合, ”SetEcuSignal”コマンドでイベントコントローラの接続を行ってください.

[マルチファンクションデバイス AIO-121601M-PCI をご利用の方へ] ※1

・変換停止条件を比較カウント一致に設定する場合
”CntmNotifyCountUp”コマンドで比較カウント一致発生条件の設定を行ってください.
・変換停止条件をカウントクリアに設定する場合
“CntmZeroClearCount”コマンドでカウントクリアを行ってください.
・変換停止条件をキャリーボローに設定する場合
“CntmNotifyCarryBorrow”コマンドでキャリーボロー発生条件の設定を行ってください.
・変換停止条件をタイマーに設定する場合
“CntmNotifyTimer”コマンドでタイマーイベント発生条件の設定を行ってください.

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.

SetAiStopTimes

構文 *object*.SetAiStopTimes(<Times>)

引数 **<Times>** = VT_I4: 変換停止サンプリング回数※1

戻り値 なし

説明 アナログ入力 AD 変換サンプリング停止回数を設定します. ※1
この設定は, ”SetAiStopTrigger”コマンドで変換停止条件を設定回数変換終了に設定した場合に必要になります.
※変換停止条件が設定回数変換終了以外の場合には実行する必要はありません.

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.

SetAiStopLevel

構文 *object*.SetAiStopLevel(<Channel>, <Level>, <Direction>)

引数 **<Channel>** = VT_I2: レベル比較を行う AI チャネル番号※1

<Level> = VT_I4: レベル比較を行うデータ※1
以下の範囲からバイナリ値で設定します.
デバイスにより設定できる値は異なります.
12 ビット分解能のデバイス: 0～4095
16 ビット分解能のデバイス: 0～65535

<Direction> = VT_I2:レベル比較の方向※1
以下の範囲から設定します.
0: 両方
1: 立ち上がり
2: 立ち下がり

戻り値 なし

説明 アナログ入力 AD 変換サンプリング時のレベル比較停止時のレベルをバイナリ値で設定します. ※1
この設定は, ”SetAiStopTrigger”コマンドで変換停止条件をレベル比較に設定した場合に必要になります.
※変換停止条件がレベル比較以外の場合には実行する必要はありません.

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.

SetAiStopLevelEx

構文 *object*.SetAiStopLevelEx(<Channel>, <Level>, <Direction>)

引数 **<Channel>** = VT_I2: レベル比較を行う AI チャネル番号※1

<Level> = VT_R4: レベル比較を行うデータ※1
電圧または電流で設定します.

<Direction> = VT_I2:レベル比較の方向※1
以下の範囲から設定します.

0：両方
1：立ち上がり
2：立ち下がり

戻り値 なし

説明 アナログ入力 AD 変換サンプリング時のレベル比較停止時のレベルを電圧または電流で設定します. ※1
この設定は，”SetAiStopTrigger”コマンドで変換停止条件をレベル比較に設定した場合に必要になります.
※変換停止条件がレベル比較以外の場合には実行する必要はありません.

※1:詳細については，API-AIO（WDM）Help（PCI 時） 又は API-USBP（W32）Help（USB 時）を参照して下さい.

SetAiStopInRange

構文 *object*.SetAiStopInRange(<Channel>, <Level1>, <Level2>, <StateTimes>)

引数

<Channel> = VT_I2：インレンジ比較を行う AI チャネル番号※1

<Level1>，<Level2> = VT_I4：インレンジ比較を行う範囲※1
以下の範囲からバイナリ値で設定します. デバイスにより設定できる値は異なります.
12 ビット分解能のデバイス：0～4095
16 ビット分解能のデバイス：0～65535

<StateTimes> = VT_I4：状態保持回数※1
インレンジ比較成立後，状態を保持する期間をサンプリング回数で設定します.
実際の変換は，StateTimes 分のサンプリング時間が経過した後に停止します.

戻り値 なし

説明 アナログ入力 AD 変換サンプリング時のインレンジ比較停止時のレベルをバイナリ値で設定します. ※1
この設定は，”SetAiStopTrigger”コマンドで変換停止条件をインレンジ比較に設定した場合に必要になります.
※変換停止条件がインレンジ比較以外の場合には実行する必要はありません.

※1:詳細については，API-AIO（WDM）Help（PCI 時） 又は API-USBP（W32）Help（USB 時）を参照して下さい.

SetAiStopInRangeEx

構文 *object*.SetAiStopInRangeEx(<Channel>, <Level1>, <Level2>, <StateTimes>)

引数

<Channel> = VT_I2：インレンジ比較を行う AI チャネル番号※1

<Level1>，<Level2> = VT_R4：インレンジ比較を行う範囲※1
電圧または電流で設定します.

<StateTimes> = VT_I4：状態保持回数※1
インレンジ比較成立後，状態を保持する期間をサンプリング回数で設定します.
実際の変換は，StateTimes 分のサンプリング時間が経過した後に停止します.

戻り値 なし

説明 アナログ入力 AD 変換サンプリング時のインレンジ比較停止時のレベルを電圧または電流で設定します. ※1

この設定は，”SetAiStopTrigger”コマンドで変換停止条件をインレンジ比較に設定した場合に必要になります．
※変換停止条件がインレンジ比較以外の場合には実行する必要はありません．

※1:詳細については，API-AIO（WDM）Help（PCI 時） 又は API-USBP（W32）Help（USB 時）を参照して下さい．

SetAiStopOutRange

構文 *object*.SetAiStopOutRange(<Channel>, <Level1>, <Level2>, <StateTimes>)

引数

<Channel> = VT_I2： アウトレンジ比較を行う AI チャネル番号※1

<Level1>，<Level2> = VT_I4：アウトレンジ比較を行う範囲※1
以下の範囲からバイナリ値で設定します．デバイスにより設定できる値は異なります．
12 ビット分解能のデバイス： 0～4095
16 ビット分解能のデバイス： 0～65535

<StateTimes> = VT_I4： 状態保持回数※1
アウトレンジ比較成立後，状態を保持する期間をサンプリング回数で設定します．
実際の変換は，StateTimes 分のサンプリング時間が経過した後に停止します．

戻り値 なし

説明 アナログ入力 AD 変換サンプリング時のアウトレンジ比較停止時のレベルをバイナリ値で設定します．※1
この設定は，”SetAiStopTrigger”コマンドで変換停止条件をアウトレンジ比較に設定した場合に必要になります．
※変換停止条件がアウトレンジ比較以外の場合には実行する必要はありません．

※1:詳細については，API-AIO（WDM）Help（PCI 時） 又は API-USBP（W32）Help（USB 時）を参照して下さい．

SetAiStopOutRangeEx

構文 *object*.SetAiStopOutRangeEx(<Channel>, <Level1>, <Level2>, <StateTimes>)

引数

<Channel> = VT_I2： アウトレンジ比較を行う AI チャネル番号※1

<Level1>，<Level2> = VT_R4：アウトレンジ比較を行う範囲※1
電圧または電流で設定します．

<StateTimes> = VT_I4： 状態保持回数※1
アウトレンジ比較成立後，状態を保持する期間をサンプリング回数で設定します．
実際の変換は，StateTimes 分のサンプリング時間が経過した後に停止します．

戻り値 なし

説明 アナログ入力 AD 変換サンプリング時のアウトレンジ比較停止時のレベルを電圧または電流で設定します．※1
この設定は，”SetAiStopTrigger”コマンドで変換停止条件をアウトレンジ比較に設定した場合に必要になります．
※変換停止条件がアウトレンジ比較以外の場合には実行する必要はありません．

※1:詳細については，API-AIO（WDM）Help（PCI 時） 又は API-USBP（W32）Help（USB 時）を参照して下さい．

SetAiStopDelayTimes

構文 *object*.SetAiStopDelayTimes(<Times>)

引数 **<Times>** = VT_I4: 変換停止遅延回数※1
変換停止遅延回数を設定

戻り値 なし

説明 アナログ入力 AD 変換サンプリング時の停止遅延回数を設定します. ※1

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.

SetAiRepeatTimes

構文 *object*.SetAiRepeatTimes(<Times>)

引数 **<Times>** = VT_I4: リピート回数※1
0: リピート動作を無限に繰り返します.
1～: 指定回数分のリピート動作を行います.

戻り値 なし

説明 アナログ入力 AD 変換サンプリング時のリピート回数を設定します. ※1
リピート動作とは, サンプリング開始から停止までの一連の動作の繰り返し回数です.

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.

SetAiEventConditions

構文 *object*.SetAiEventConditions(<Event>, <DataType>)

引数 **<Event>** = VT_I4: イベント要因※1
ビット単位で以下のような意味を持ち, これらを組み合わせて複数のイベント発生要因を設定可能です.
00000002H: AD 変換開始条件成立イベント
00000010H: リピート終了イベント
00000020H: デバイス動作終了イベント
00000080H: 指定サンプリング回数格納イベント
00010000H: オーバーフローイベント
00020000H: サンプリングクロックエラーイベント
00040000H: AD 変換エラーイベント

<DataType> = VT_I4: 変換データタイプ
CaoController::OnMesseage イベント発生時に取得されるデータの変換データタイプを指定
0: 電圧または電流値
1: バイナリ値
2: 取得データなし(イベント通知のみ)

戻り値 なし

説明 アナログ入力に関する CaoController::OnMesseage のイベント要因を設定します.
指定サンプリング回数格納イベント要因を使用する場合, "SetAiEventSamplingTimes"コマンドでイベントを発生させるサンプリング回数を設定してください.

注:本プロバイダは，変換データの転送方式はデバイスの起動初期値である，“デバイスバッファモード”のみサポートしています(“ユーザバッファモード”は未サポートです).

※1:詳細については，API-AIO（WDM）Help（PCI 時） 又は API-USBP（W32）Help（USB 時）を参照して下さい.

SetAiEventSamplingTimes

構文 *object*.SetAiEventSamplingTimes(<SamplingTimes>)

引数 **<SamplingTimes>** = VT_I4: イベントを発生させるサンプリング数※1

戻り値 なし

説明 アナログ入力指定サンプリング回数格納イベント利用時のサンプリング回数を設定します. ※1

“SetAiEventConditions”コマンドで，指定サンプリング回数格納イベント要因を使用する場合のサンプリング数の設定を行います.

※1:詳細については，API-AIO（WDM）Help（PCI 時） 又は API-USBP（W32）Help（USB 時）を参照して下さい.

StartAiSamplingAsync

構文 *object*.StartAiSamplingAsync()

引数 なし

戻り値 なし

説明 アナログ入力 AD 変換サンプリングを開始し直ちに処理を返します. (非同期動作)※1
設定された条件に基づいて AD 変換を開始します.
サンプリングクロックエラー，AD 変換エラーは自動的にリセットされます.
デバイスメモリはリセットされません. 過去の変換データがメモリに存在する場合，新たな変換データは過去のデータに続いて格納されます.

注:本プロバイダでは，事前に”SetAiEventCondition”コマンドによりイベント要因が設定されている(Event <> 0)場合は，AD 変換開始前に再度イベント要因の設定を実行します.
これはマルチプロセス動作環境の際に，別のプロセスから同一デバイスに対してイベント要因の設定がされていたとしても，本コマンドにてデバイスの動作を開始するプロセスで設定されたイベント要因<Event>と変換データタイプ<DataType>を優先させるためです.

※1:詳細については，API-AIO（WDM）Help（PCI 時） 又は API-USBP（W32）Help（USB 時）を参照して下さい.

StartAiSamplingSync

構文 *object*.StartAiSamplingSync(<Timeout>)

引数 **<Timeout>** = VT_I4: タイムアウト時間※1
関数が戻るまでのタイムアウト時間をミリ秒単位で指定します.
0 を指定すると，動作が停止するまで待ち続けます.

戻り値 なし

説明 アナログ入力 AD 変換サンプリングを開始しタイムアウト時間経過後に処理を返します. (同期動作)※1
設定された条件に基づいて AD 変換を開始します.

サンプリングクロックエラー，AD 変換エラーは自動的にリセットされます．
デバイスメモリはリセットされません．過去の変換データがメモリに存在する場合，新たな変換データは過去のデータに続いて格納されます．

注：本プロバイダでは，事前に”SetAiEventCondition”コマンドによりイベント要因が設定されている(Event <> 0)場合は，AD 変換開始前に再度イベント要因の設定を実行します．
これはマルチプロセス動作環境の際に，別のプロセスから同一デバイスに対してイベント要因の設定がされていたとしても，本コマンドにてデバイスの動作を開始するプロセスで設定されたイベント要因<Event>と変換データタイプ<DataType>を優先させるためです．

※1:詳細については，API-AIO（WDM）Help（PCI 時） 又は API-USBP（W32）Help（USB 時）を参照して下さい．

StopAiSampling

構文 *object*.StopAiSampling()

引数 なし

戻り値 なし

説明 アナログ入力 AD 変換サンプリングを停止します．※1
複数チャネルの変換を行っている場合，コマンドを実行した時点のサンプリングですべてのチャネルの変換が終了してからサンプリングが停止します．

※1:詳細については，API-AIO（WDM）Help（PCI 時） 又は API-USBP（W32）Help（USB 時）を参照して下さい．

GetAiSamplingCount

構文 *object*.GetAiSamplingCount()

引数 なし

戻り値 <Data> =VT_I4： サンプリング回数

説明 アナログ入力 デバイスメモリ(ドライバメモリ)中のサンプリング回数（位置）を取得．※1

※1:詳細については，API-AIO（WDM）Help（PCI 時） 又は API-USBP（W32）Help（USB 時）を参照して下さい．

GetAiSopTriggerCount

構文 *object*.GetAiStopTriggerCount()

引数 なし

戻り値 <Data> =VT_I4： サンプリング回数

説明 アナログ入力 停止トリガが入力された時のサンプリング回数を取得します．※1

※1:詳細については，API-AIO（WDM）Help（PCI 時） 又は API-USBP（W32）Help（USB 時）を参照して下さい．

GetAiSamplingData

構文 *object*.GetAiSamplingData(<SamplingTimes>)

引数 **<SamplingTimes>** = VT_I4: 指定サンプリング回数※1

戻り値 <Data> = VT_ARRAY | VT_VARIANT
Array[0]: VT_I4: 実際に取得されたサンプリング数
Array[1]: VT_EMPTY:サンプリングデータなし(Array[0]=0)時
VT_ARRAY | VT_I4:(Array[0]×使用 AI チャネル数)要素の変換データ配列
変換データはバイナリ値で以下の範囲
・12 ビット分解能のデバイス: 0～4095
・16 ビット分解能のデバイス: 0～65535

説明 デバイスメモリ(ドライバメモリ)から指定サンプリング分の変換データをバイナリ値で取得します. ※1
使用 AI チャネル数分のデータを格納します.
・FIFO メモリの場合
常に古いデータから読み込みを行います.
一度読み込んだデータを再度読み込むことはできません.
・RING メモリの場合
最新のデータから, 指定したサンプリング数のデータを取得します.
同じデータを繰り返し取得することができます.

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.

GetAiSamplingDataEx

構文 *object*.GetAiSamplingDataEx(<SamplingTimes>)

引数 **<SamplingTimes>** = VT_I4: 指定サンプリング回数※1

戻り値 <Data> = VT_ARRAY | VT_VARIANT
Array[0]: VT_I4:実際に取得されたサンプリング数
Array[1]: VT_EMPTY:サンプリングデータなし(Array[0]=0)時
VT_ARRAY | VT_R4:(Array[0]×使用 AI チャネル数)要素の変換データ配列
変換データは電圧または電流で格納されます.

説明 デバイスメモリ(ドライバメモリ)から指定サンプリング分の変換データを電圧または電流値で取得します. ※1
使用 AI チャネル数分のデータを格納します.
・FIFO メモリの場合
常に古いデータから読み込みを行います.
一度読み込んだデータを再度読み込むことはできません.
・RING メモリの場合
最新のデータから, 指定したサンプリング数のデータを取得します.
同じデータを繰り返し取得することができます.

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.

GetAiStatus

構文 *object*.GetAiStatus()

引数 なし

戻り値 <Status> = VT_I4: ステータス※1
00000001H: デバイス動作中
00000002H: 開始トリガ待ち

00000010H: 指定個数以上データ格納
00010000H: オーバーフロー
00020000H: サンプリングクロックエラー
00040000H: AD 変換エラー
00080000H: ドライバスペックエラー

説明 アナログ入力ステータスを取得します. ※1

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.

GetAiResolution

構文 *object*.GetAiResolution()

引数 なし

戻り値 <**Resolution**> = VT_I2: 分解能※1
12: 12 ビット分解能
16: 16 ビット分解能
0: アナログ入力機能なし

説明 アナログ入力の分解能を取得します. ※1

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.

ResetAiMemory

構文 *object*.ResetAiMemory()

引数 なし

戻り値 なし

説明 アナログ入力デバイスメモリをリセットします. ※1
このコマンドを実行すると次の状態がリセットされます.
・メモリを管理するポインタ(リードポインタ, ライトポインタ)の値を 0 にリセットします.
・リピート回数が 0 にリセットされます.
・停止トリガ入力時のサンプリング回数が 0 にリセットされます.
・バッファオーバーフローステータスがリセットされます.
・指定個数データ格納ステータスがリセットされます.

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.

ResetAiStatus

構文 *object*.ResetAiStatus()

引数 なし

戻り値 なし

説明 アナログ入力ステータスをリセットします. ※1
このコマンドを実行すると次の状態がリセットされます.
・サンプリングクロックエラーがリセットされます.
・AD 変換エラーステータスがリセットされます.

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.

SetEcuSignal

構文 *object*.SetEcuSignal(<Destination>, <Source>)

引数 **<Destination>** = VT_I2: 接続先信号
<Source> = VT_I2: 接続元信号
※詳細については, CONTEC 製品マニュアル および API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.

戻り値 なし

説明 イベントコントローラの信号設定を行います. ※1
各機能(Ai, Ao, Cnt)の制御信号を切り替えることで, 多機能な動作を行うことができます.
同期バスに関する制御もこの関数で設定します.
組み合わせには使用できるものとそうでないものが存在します. ※1

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.

SetCntmNotifyCountUp

AIO-121601M-PCI 使用時のみ※1

構文 *object*.SetCntmNotifyCountUp(<ChNo>, <RegNo>, <Count>)

引数 **<ChNo>** = VT_I2: カウンタ CH 番号
<RegNo> = VT_I2: 比較レジスタ番号
<Count> = VT_UI4:比較レジスタに設定する比較値
指定可能範囲:
24 ビットカウンタデバイスの場合 : 0h <= Count <= FFFFFFh
32 ビットカウンタデバイスの場合 : 0h <= Count <= FFFFFFFFh

戻り値 なし

説明 カウント一致による通知の指定と比較レジスタの設定を行ないます. ※1
注:本プロバイダでは, カウント一致によるメッセージ通知機能は使用していません.

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.

SetCntmZeroClearCount

AIO-121601M-PCI 使用時のみ※1

構文 *object*.SetCntmZeroClearCount(<Channels>)

引数 **<Channels>** = VT_ARRAY | VT_I2: カウンタ CH 番号を格納した配列

戻り値 なし

説明 カウンタのカウント値のゼロクリアを行ないます. ※1
指定したカウンタチャネルのカウント値をゼロクリアします. チャネル番号は配列に入れて渡します. この関数は, カウンタ動作スタート前でも, スタート後も有効です.

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.

SetCntmNotifyCarryBorrow

AIO-121601M-PCI 使用時のみ※1

構文 *object*.SetCntmNotifyCarryBorrow()

引数 なし
戻り値 なし
説明 カウンタのカウンタのキャリー／ボロー通知の設定を行います. ※1
この関数は, カウンタスタート前に実行してください.
注:本プロバイダでは, キャリー／ボローによるメッセージ通知機能は使用していません.

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.

SetCntmNotifyTimer

AIO-121601M-PCI 使用時のみ※1

構文 *object*.SetCntmNotifyTimer(<TimeValue>)
引数 **<TimeValue>** =VT_UI4: タイマ値[msec]
指定可能範囲: 0<=TimeValue<=6553
戻り値 なし
説明 タイマーの通知を設定します. ※1
注:本プロバイダでは, キャリー／ボローによるメッセージ通知機能は使用していません.

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.

StartCntmCount

AIO-121601M-PCI 使用時のみ※1

構文 *object*.StartCntmCount(<Channels>)
引数 **<Channels>** = VT_ARRAY | VT_I2: カウンタ CH 番号を格納した配列
戻り値 なし
説明 指定したチャネルのカウント動作をスタートします. ※1
チャネル番号は配列に入れて渡します. カウンタスタート前に設定されたモードは, この関数を呼び出した時点で設定されます. カウンタスタート前に設定されたプリセット値は, この関数を呼び出した時点で設定されます.

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.

StopCntmCount

AIO-121601M-PCI 使用時のみ※1

構文 *object*.StopCntmCount(<Channels>)
引数 **<Channels>** = VT_ARRAY | VT_I2: カウンタ CH 番号を格納した配列
戻り値 なし
説明 指定したチャネルのカウント動作をストップします. ※1
チャネル番号は配列に入れて渡します.

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.

PresetCntm

AIO-121601M-PCI 使用時のみ※1

構文 *object*.PresetCntm (<ChNo>, <PresetData>)

引数 **<ChNo>** = VT_I2: カウンタ CH 番号
<PresetData> = VT_UI4:プリセット値
戻り値 なし
説明 指定したチャネルのカウント値をプリセットします.

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.

ReadCntmCount

AIO-121601M-PCI 使用時のみ※1

構文 *object*.ReadCntmCount(<Channels>)
引数 **<Channels>** = VT_ARRAY | VT_I2: カウンタ CH 番号を格納した配列
戻り値 <**CntData**> = VT_ARRAY|VT_UI4 : 取得カウント値を格納する配列
説明 指定したチャネルのカウント値を読み込みます. ※1
チャネル番号は配列に入れて渡します. チャネル番号を配列に格納した順に, 対応するカウント値が配列に格納されます.
指定されたチャネルのカウント値をラッチしてから読み込むため, 複数チャネルを指定した場合でも同じタイミングのカウント値を取得することができます.

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.

ReadCntmStatusEx

AIO-121601M-PCI 使用時のみ※1

構文 *object*.ReadCntmStatusEx (<ChNo>)
引数 **<ChNo>** = VT_I2: カウンタ CH 番号
戻り値 **<Status>** = VT_UI4: 取得ステータス値※1
各ビットは, 次のように定義されています.
bit0: 汎用入力状態(1:LOW, 0:HIGH)
bit1: カウント一致レジスタ 0 (1:不一致, 0:一致)
bit2: カウント一致レジスタ 1 (1:不一致, 0:一致)
bit3: UP/DOWN(1:ダウンカウント中, 0:アップカウント中)
bit4: B 相(1:HIGH, 0:LOW)
bit5: A 相(1:HIGH, 0:LOW)
bit6: Z 相(1:正論理=HIGH/負論理=LOW, 0:正論理=LOW /負論理=HIGH)
bit7: Borrow(1:Borrow 検出, 0:Borrow 未検出)
bit8: Carry(1: Carry 検出, 0: Carry 未検出)
bit9: フィルタエラー(1:フィルタエラー検出, 0: フィルタエラー未検出)
bit10: 異常入力エラー(1:異常入力エラー検出, 0:異常入力エラー未検出)
bit11: 断線アラームエラー(1:断線アラームエラー検出, 0:断線アラームエラー未検出)
説明 指定したカウンタチャネルのステータスを取得します. ※1

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.

SetCntmZMode

AIO-121601M-PCI 使用時のみ※1

構文 *object*.SetCntmZMode (<ChNo>, <Mode>)
引数 **<ChNo>** = VT_I2: カウンタ CH 番号

<Mode> = VT_I2: 使用方法※1
指定可能範囲:
1: 未使用
2: 次の一回
3: 毎回

戻り値 なし

説明 指定したカウンタチャネルの Z 相使用方法を設定します(未使用／1 回／毎回). ※1

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.

SetCntmZLogic

AIO-121601M-PCI 使用時のみ※1

構文 *object*.SetCntmZLogic (<ChNo>, <Logic>)

引数
<ChNo> = VT_I2: カウンタ CH 番号
<Logic> = VT_I2: 論理※1
指定可能範囲:
0: 正論理
1: 負論理

戻り値 なし

説明 指定したカウンタチャネルの Z 相論理を設定します(正論理／負論理). ※1

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.

SetCntmCountDirection

AIO-121601M-PCI 使用時のみ※1

構文 *object*.SetCntmCountDirection (<ChNo>, <Dir>)

引数
<ChNo> = VT_I2: カウンタ CH 番号
<Dir> = VT_I2: カウント方向※1
指定可能範囲:
0: ダウンカウント
1: アップカウント

戻り値 なし

説明 指定したカウンタチャネルのカウント方向を設定します(アップカウント／ダウンカウント). ※1

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.

SetCntmOperationMode

AIO-121601M-PCI 使用時のみ※1

構文 *object*.SetCntmOperationMode (<ChNo>, <Phase>, <Mul>, <SyncClr>)

引数
<ChNo> = VT_I2: カウンタ CH 番号
<Phase > = VT_I2: 相数※1
指定可能範囲:
0: 単相
1: 2 相
2: ゲートコントロール
<Mul > = VT_I2: 逓倍※1

指定可能範囲:
0: 1 逓倍
1: 2 逓倍
2: 4 逓倍

<SyncClr > = VT_I2: 同期クリア/非同期クリア※1
指定可能範囲:
0: 非同期クリア
1: 同期クリア

戻り値 なし

説明 指定したカウンタチャネルの動作モードを設定します(相数, クリア, 逓倍). ※1

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.

SetCntmDigitalFilter

AIO-121601M-PCI 使用時のみ※1

構文 *object*.SetCntmDigitalFilter (<ChNo>, <FilterValue>)

引数 **<ChNo>** = VT_I2: カウンタ CH 番号
<FilterValue> = VT_I2:デジタルフィルタ係数値※1

戻り値 なし

説明 指定したカウンタチャネルのデジタルフィルタ値を設定します. ※1

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)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 2-4 コントローラクラス ユーザ変数一覧

変数名	データ型	説明	属性	
			get	put
AI? ※2	VT_R4	アナログ入力 CH?の電圧または電流値※1 を取得します. 変数名の後ろに論理番号※6を指定します. 例）“AI1”	○	―
AIS※2	VT_ARRAY | VT_R4 または VT_EMPTY	(使用 AI チャネル数×サンプリング回数)分の AD 変換データを電圧または電流値※1 で取得します. AD 変換データがない時は VT_EMPTY となります.	○	―
AO? ※3	VT_R4	アナログ出力 CH?へ指定電圧または電流値※1 を出力します. 変数名の後ろに論理番号※6を指定します. 例）“AO1”	―	○
DI? ※4	VT_I2	デジタル入力ビット?のビット値（0 or 1）※1 を取得します. 変数名の後ろに論理番号※6を指定します. 例）“DI1”	○	―
DO? ※5	VT_I2	デジタル出力ビット?へビット値（0 or 1）※1 を出力します. 変数名の後ろに論理番号※6を指定します. 例）“DO1”	―	○
DIB? ※4	VT_I2	デジタル入力バイト?のバイト値（0～255）※1 を取得します. 変数名の後ろに論理番号※6を指定します. 例）“DIB1”	○	―
DOB? ※5	VT_I2	デジタル出力バイト?へバイト値（0～255）※1 を出力します. 変数名の後ろに論理番号※6を指定します. 例）“DOB1”	―	○

※1:詳細については，API-AIO（WDM）Help（PCI 時） 又は API-USBP（W32）Help（USB 時）を参照して下さい.
※2:アナログ入力を搭載した機種でのみ使用可能です. 詳細は，CONTEC 製品マニュアルを参照して下さい.
※3:アナログ出力を搭載した機種でのみ使用可能です. 詳細は，CONTEC 製品マニュアルを参照して下さい.
※4:デジタル入力を搭載した機種でのみ使用可能です. 詳細は，CONTEC 製品マニュアルを参照して下さい.
※5:デジタル出力を搭載した機種でのみ使用可能です. 詳細は，CONTEC 製品マニュアルを参照して下さい.
※6:論理番号は 0～99 までの範囲で変数オブジェクトの生成は可能ですが，実際にデータの取得/設定が可能な範囲は搭載機種の CH 実装数等となります. 詳細は，CONTEC 製品マニュアルを参照して下さい.

表 2-5 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
@MAX_AI※2	VT_I2	アナログ入力チャンネルの最大数※1を取得します.	○	－
@MAX_AO※3	VT_I2	アナログ出力チャンネルの最大数※1を取得します.	○	－
@ProcessId	VT_I4	プロセス ID を取得します.	○	－
@DeviceName	VT_BSTR	接続されているボードのデバイス名※1を取得します.	○	－
@RANGE_AO※4	VT_I2	アナログ出力レンジ(全 CH 共通)※1 の設定/取得をします.	○	○
@RANGE_AI※5	VT_I2	アナログ入力レンジ(全 CH 共通)※1 の設定/取得をします.	○	○
@STS_AI※2	VT_I4	アナログ入力ステータス※1を取得します. 取得値: 00000001H:デバイス動作中 00000002H:開始トリガ待ち 00000010H:指定個数以上データ格納 00010000H:オーバーフロー 00020000H:サンプリングクロックエラー 00040000H:AD 変換エラー 00080000H:ドライバスペックエラー 注: 複数のステータスが同時に発生している場合は総和値が取得されます. ※1	○	－

※1:詳細については, API-AIO(WDM)Help(PCI 時) 又は API-USBP(W32)Help(USB 時)を参照して下さい.
※2:アナログ入力を搭載した機種でのみ使用可能です. 詳細は, CONTEC 製品マニュアルを参照して下さい.
※3:アナログ出力を搭載した機種でのみ使用可能です. 詳細は, CONTEC 製品マニュアルを参照して下さい.
※4:アナログ出力を搭載した機種で 且つ 関数実行によるレンジ設定が可能な機種でのみ利用可能です.
詳細は, CONTEC 製品マニュアルを参照して下さい.
※5:アナログ入力を搭載した機種で 且つ 関数実行によるレンジ設定が可能な機種でのみ利用可能です.
詳細は, CONTEC 製品マニュアルを参照して下さい.

2.5. エラーコード

AIO 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) AIO API が返すエラー

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

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

AIO API の詳細については，CONTEC 社 API-AIO(WDM) Help（PCI 時） 又は API-USBP（W32）Help（USB 時）を参照してください．

2) 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-AIO API 対応表

AIO プロバイダは，CaoVariable により値の設定/取得を行う API 関数を実行します．

表 2-6 コントローラクラス，変数クラスと AIO API 対応表

CAO API		AIO 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	AioQueryDeviceName() AioInit()
	Coexistence	AioGetAiStatus() AioGetAoStatus() AioResetProcess() ... デバイス停止中のみ※
	ResetDevice	AioResetDevice() < USB 機器のみ(ケーブル挿抜時の誤動作対策)> AioGetDeviceType() AioExit() AioInit()
	AiInputMethod	AioSetAiInputMethod()
	AiChannels	AioSetAiChannels()
	AiRangeAll	AioSetAiRangeAll()
	AiMemoryType	AioSetAiMemoryType()
	AiClockType	AioSetAiClockType()
	AiSamplingClock	AioSetAiSamplingClock()
	AiClockEdge	AioSetAiClockEdge
	AiAutoSampling	AioSetAiEventSamplingTimes() AioSetAiCallBackProc() AioSetAiStartTrigger() AioSetAiStopTrigger() AioStartAi()
CaoWorkspaces::Remove()	－	AioExit()
CaoController::Execute()	SetAiStartTrigger	AioSetAiStartTrigger()
	SetAiStartLevel	AioSetAiStartLevel()
	SetAiStartLevelEx	AioSetAiStartLevelEx()
	SetAiStartInRange	AioSetAiStartInRange()
	SetAiStartInRangeEx	AioSetAiStartInRangeEx()

	SetAiStartOutRange	AioSetAiStartOutRange()
	SetAiStartOutRangeEx	AioSetAiStartOutRangeEx()
	SetAiStopTrigger	AioSetAiStopTrigger()
	SetAiStopTimes	AioSetAiStopTimes()
	SetAiStopLevel	AioSetAiStopLevel()
	SetAiStopLevelEx	AioSetAiStopLevelEx()
	SetAiStopInRange	AioSetAiStopInRange()
	SetAiStopInRangeEx	AioSetAiStopInRangeEx()
	SetAiStopOutRange	AioSetAiStopOutRange()
	SetAiStopOutRangeEx	AioSetAiStopOutRangeEx()
	SetAiStopDelayTimes	AioSetAiStopDelayTimes()
	SetAiRepeatTimes	AioSetAiRepeatTimes()
	SetAiEventConditions	AioSetAiCallBackProc()
	SetAiEventSamplingTimes	AioSetAiEventSamplingTimes()
	StartAiSamplingAsync	AioSetAiCallBackProc() ... イベント要因: 0 以外時 AioGetAiStatus() AioStartAi()
	StartAiSamplingSync	AioSetAiCallBackProc() ... イベント要因: 0 以外時 AioStartAiSync()
	StopAiSampling	AioStopAi()
	GetAiSamplingCount	AioGetAiSamplingCount()
	GetAiStopTriggerCount	AioGetAiStopTriggerCount()
	GetAiSamplingData	AioGetAiSamplingData()
	GetAiSamplingDataEx	AioGetAiSamplingDataEx()
	GetAiResolution	AioGetAiResolution()
	ResetAiMemory	AioResetAiMemory()
	ResetAiStatus	AioResetAiStatus()
	SetEcuSignal	AioSetEcuSignal()
	SetCntmNotifyCountUp	AioCntmNotifyCountUp()
	SetCntmZeroClearCount	AioCntmZeroClearCount()
	SetCntmNotifyCarryBorrow	AioCntmNotifyCarryBorrow()
	SetCntmNotifyTimer	AioCntmNotifyTimer()
	StartCntmCount	AioCntmStartCount()
	StopCntmCount	AioCntmStopCount()
	PresetCntm	AioCntmPreset()
	ReadCntmCount	AioCntmReadCount()
	ReadCntmStatusEx	AioCntmReadStatusEx()
	SetCntmZMode	AioSetCntmZMode()
	SetCntmZLogic	AioSetCntmZLogic()
	SetCntmCountDirection	AioSetCntmCountDirection()
	SetCntmOperationMode	AioSetCntmOperationMode()
	SetCntmDigitalFilter	AioSetCntmDigitalFilter()
CaoController::OnMessage()	DI 関連	AioInputDiByte()
	AI 関連	AioGetAiChannels() AioGetAiSamplingData() AioGetAiSamplingDataEx()
CaoVariable::get_Value()	AI?	AioSingleAiEx()
	AIS	AioGetAiSamplingCount() AioGetAiChannels() AioGetAiSamplingDataEx()
	DI?	AioInputDiBit()
	DIB?	AioInputDiByte()

	@MAX_AI	AioGetAiMaxChannels()
	@MAX_AO	AioGetAoMaxChannels()
	@ProcessId	―
	@DeviceName	―
	@RANGE_AO	AioGetAoRange()
	@RANGE_AI	AioSetAiRangeAll()
	@STS_AI	AioGetAiStatus()
CaoVariable::put_Value()	AO?	AioSingleAoEx()
	DO?	AioOutputDoBit()
	DOB?	AioOutputDoByte()
	@RANGE_AO	AioSetAoRangeAll()

※ AIO API の詳細については，CONTEC 社 API-AIO（WDM）Help（PCI 時） 又は API-USBP（W32）Help（USB 時）を参照して下さい.

3. サンプルプログラム

3.1. 指定アナログ入力チャネル AD 変換データ取得サンプル(簡易 AI 入力機能)

以下に変数“AI1”で, AI CH1 の電圧または電流値を取得するサンプルを示します.

List 3-1 SampleAi.frm

```
Private caoEng As CaoEngine
Private caoAIOl As CaoController
Private caoVar As CaoVariable

Private Sub Form_Load()

    Set caoEng = New CaoEngine
    Set caoAIOl = caoEng.Workspaces(0).AddController("SampleAi", "CaoProv.CONTEC.AIO", "", _
                                                     "DeviceName=AIO001")
    Set caoVar = caoAIOl.AddVariable("AI1",  "")

End Sub


Private Sub cmdGet_Click()

    Dim sngRet As Single

    sngRet = caoVar.Value

    Text1.Text = CStr(sngRet)

End Sub
```

3.2. デジタル入力状態変化イベント受信サンプル

以下にデジタル入力状態変化サンプリング周期 1 秒で, デジタル入力バイト値が変化した時にイベントを受信するサンプルを示します.

List 3-2 SampleEventDi.frm

```
Private caoEng As CaoEngine
Private WithEvents caoAIOl As CaoController

Private Sub Form_Load()

    Set caoEng = New CaoEngine
    Set caoAIOl = caoEng.Workspaces(0).AddController("SampleEventDi","CaoProv.CONTEC.AIO", _
                                                     "","DeviceName=AIO001,Interval=1000")
End Sub


' 受信イベント
Private Sub ctrl_OnMessage(ppCaoMess As CAOLib.ICaoMessage)

    ' 受信デジタル入力バイト値
    text2.Text = ppCaoMess.Value

End Sub
```

3.3. アナログ入力サンプリング回数格納イベント受信サンプル(高機能 AI 入力機能)

以下に AD 指定サンプリング回数格納イベント受信時に, 2 チャネル分の AI サンプリングデータを電圧または電流値で取得するサンプルを示します.

List 3-3 SampleEventAi.frm

```
Private m_caoEng As CaoEngine
Private WithEvents m_caoCtrl As CaoController

Private Sub Form_Load()

    Dim vntArg As Variant

    Set m_caoEng = New CaoEngine
    Set m_caoCtrl = m_caoEng.Workspaces(0).AddController("SampleEventAi", "CaoProv.CONTEC.AIO",
"", "DeviceName=AI0000, AiChannels=2, AiSamplingClock=10000 ")

    ' サンプリング 開始条件：ソフトウエア(0)
    vntArg = Array(CLng(0))
    m_caoCtrl.Execute "SetAiStartTrigger", vntArg

    ' サンプリング 停止条件：コマンド(4)
    vntArg = Array(CLng(4))
    m_caoCtrl.Execute "SetAiStopTrigger", vntArg

    ' イベント発生サンプリング回数：100
    vntArg = Array(CLng(100))
    m_caoCtrl.Execute "SetAiEventSamplingTimes", vntArg

End Sub

' サンプリング開始ボタン
Private Sub cmdStartSampling_Click()

    txtText1.Text = ""
    DoEvents

    ' イベント要因：<Event    > 指定サンプリング回数格納イベント   (00000080H)
    '                           オーバーフローイベント             (00010000H)
    '                           サンプリングクロックエラーイベント(00020000H)
    '                           AD 変換エラーイベント               (00040000H)
    '               <DataType> 電圧または電流値(0)
    Dim vntArg As Variant
    vntArg = Array(CLng(&H70080), CLng(0))
    m_caoCtrl.Execute "SetAiEventConditions", vntArg

    ' デバイスメモリリセット
    vntArg = Empty
    m_caoCtrl.Execute "ResetAiMemory", vntArg

    ' ステータスリセット
    m_caoCtrl.Execute "ResetAiStatus", vntArg

    ' サンプリング開始
    m_caoCtrl.Execute "StartAiSamplingAsync", vntArg

End Sub

' サンプリング停止ボタン
Private Sub cmdStopSampling_Click()
```

```
    Dim vntArg As Variant

    ' サンプリング停止
    vntArg = Empty
    m_caoCtrl.Execute "StopAiSampling", vntArg

End Sub

' 受信イベント
Private Sub m_caoCtrl_OnMessage(ByVal pICaoMess As CAOLib.ICaoMessage)

    txtText1.Text = ""
    DoEvents

    Select Case pICaoMess.Number

       Case 103    ' 指定サンプリング回数格納イベント
            If (VarType(pICaoMess.Value) And vbArray) = vbArray Then
                Dim i As Long
                txtText1.Text = "Msg=103 : "
                For i = 0 To UBound(pICaoMess.Value)
                    txtText1.Text = txtText1.Text & CStr(pICaoMess.Value(i)) & ","
                Next
            End If

        Case 104    ' オーバーフローイベント
            txtText1.Text = "Msg=104 : Over flow error."

        Case 105    ' サンプリングクロックエラーイベント
            txtText1.Text = "Msg=105 : Sampling clock error."

        Case 106    ' AD 変換エラーエラーイベント
            txtText1.Text = "Msg=106 : AD conversion error."

   End Select

End Sub
```

3.4. アナログ入力自動サンプリングサンプル(高機能 AI 入力機能)

以下に AddController 起動オプションの設定のみ(Execute コマンドを使用しない)で, 2 チャネル分の AI サンプリングデータを電圧または電流値で取得するサンプルを示します.

List 3-4 SampleAiAutoSampling.frm

```
Private m_caoEng As CaoEngine
Private m_caoCtrls As CaoControllers
Private WithEvents m_caoCtrl As CaoController

Private Sub Form_Load()

    Dim vntArg As Variant

    Set m_caoEng = New CaoEngine
    Set m_caoCtrls = m_caoEng.Workspaces(0).Controllers
    Set m_caoCtrl = m_caoCtrls.Add("SampleAiAutoSampling", _
                                   "CaoProv.CONTEC.AIO", "", _
                                   "DeviceName=AIO000, AiChannels=2, AiSamplingClock=10000,
AiAutoSampling=1:100:0x00070080")

End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)

    If Not m_caoCtrl Is Nothing Then
        m_caoCtrls.Remove m_caoCtrl.Index
        Set m_caoCtrl = Nothing
    End If

End Sub

' 受信イベント
Private Sub m_caoCtrl_OnMessage(ByVal pICaoMess As CAOLib.ICaoMessage)

    txtText1.Text = ""
    DoEvents

    Select Case pICaoMess.Number

       Case 103    ' 指定サンプリング回数格納イベント
            If (VarType(pICaoMess.Value) And vbArray) = vbArray Then
                Dim i As Long
                txtText1.Text = "Msg=103 : "
                For i = 0 To UBound(pICaoMess.Value)
                    txtText1.Text = txtText1.Text & CStr(pICaoMess.Value(i)) & ","
                Next
            End If

        Case 104    ' オーバーフローイベント
            txtText1.Text = "Msg=104 : Over flow error."

        Case 105    ' サンプリングクロックエラーイベント
            txtText1.Text = "Msg=105 : Sampling clock error."

        Case 106    ' AD 変換エラーエラーイベント
            txtText1.Text = "Msg=106 : AD conversion error."

   End Select

End Sub
```