Multifunction-I/O-X3 series
ADIOX3-API
REFERENCE

Update 2019-3-23

SAYA Inc.

目次

1.概要	3
2.オープンクローズ	3
3.設定	4
4.データ収集～信号変換	5
5.校正	7
6.高速タイマ	7
7.構造体	8

1.概要

ADiox3-API

ADiox3-API は MultifunctionI/O-X3 シリーズ及びインフラサウンドセンサー用のアプリケーションを、素早く開発するためにデザインされた 27 個の関数と 4 個の構造体で構成されたシンプルな API です。**ADXⅢ42LE-***、ADXⅢ42FE-***、ADXⅢ-INF01LE、ADXⅢ-INF04LE** に対応します。いずれも Ethernet インターフェースに対応し、RS232C、LVTTL の UART 版には対応しません。

① ハードウェアにアクセスするドライバ機能
② 機能および信号調節の設定の保存・読出し
③ 複数機器に対するサバイバル機能（生存中のハードウェアを探して、計測を継続する）

存在しない機器へのアクセス

この場合、関数は実行されず FALSE(=0)を返します。

開発用ファイル

VC++系にはヘッダファイルとインポートライブラリが提供されます

64bit 系の Windows は CDROM¥MFIO_X3¥sdk¥VisuaCPP_X64
32bit 系の Windows は CDROM¥MFIO_X3¥sdk¥VisuaCPP_X86
ADiox3.h／ADiox3.LIB [ADiox3.dll + ADioxScp.dll]
メインライブラリのヘッダファイル、インポートライブラリ、ダイナミックリンクライブラリです。

他の開発環境

Windows の API 同様なので、他の開発環境にも対応します。但しヘルパファイルは提供されません。

CARD_ID について

最大 4 台の機器をハンドリングでき、IP アドレスに割り付けられた CARD_ID（0～3）で識別します。

2.オープンクローズ(7)

bADioxOpen

ドライバのオープンを行います。 bADioxSystemLoad を使って一気にオープン～設定を行う場合には、本関数をコールする必要はありません。

BOOL bADioxOpen (BYTE bCardID,ADR_IP_CONF sADR_IP_CONF);

引数 bCardID オープンしたい CARD_ID
sADR_IP_CONF ADR_IP_CONF 構造体を指定します。
この構造体で、IP アドレスおよびポート番号を、CARD_ID に割り付けます。

戻り値 成功すると 1(TRUE)、失敗すると 0(FALSE)が返ります。

bADioxClose

ドライバのクローズ、ハードウェアの終了処理、システムメモリへのバッファ開放等を行います。

BOOL bADioxClose(BYTE bCardID);

引数 bCardID ターゲットデバイスのカード ID

戻り値 成功すると 1(TRUE)、失敗すると 0(FALSE)が返ります。

bADioxSystemLoad

複数の機器に対し、ドライバのオープン、ハードウェアの初期化（設定反映）、メモリバッファ確保を行います。本関数は設定ファイルを開き、bADioxOpen（オープン）→bADioxConfigration（設定）を任意台数分行います。ここで設定された内容は、DLL 内部の**"設定構造体データベース"**に反映されます。

BOOL bADioxSystemLoad (char *szFile , ADR_IP_CONF sADR_IP_CONF);

引数 *szFile 設定ファイルの場所を指定します。
sADR_IP_CONF ADR_IP_CONF 構造体を指定します。
この構造体で、IP アドレスおよびポート番号を、CARD_ID に割り付けます。

設定ファイルの構造

サイズ	内容	値・注意事項
DWORD (4Byte)	ヘッダ	必ず 0x954EF06B
DWORD (4Byte)	バージョン	必ず 0x00000001
ADIOX_SYSTEM 構造体	CARD_ID =0 の設定	CardID=0 は必須
ADIOX_SYSTEM 構造体	CARD_ID =1 の設定	必要がなくてもダミーを入れる
ADIOX_SYSTEM 構造体	CARD_ID =2 の設定	必要がなくてもダミーを入れる
ADIOX_SYSTEM 構造体	CARD_ID =3 の設定	必要がなくてもダミーを入れる
DWORD (4Byte)	フッタ	必ず 0x954EF06B

面倒なら ファイル作成の為のコーディングが面倒ならば、付属ソフトウェアの MultiLoggerX3 で設定を行ってください。
MultiLoggerX3 終了後、同フォルダに作成されている ConfMlt.scp が、上のフォーマットのファイルです。

戻り値 成功すると 1(TRUE)、失敗すると 0(FALSE)が返ります。

bADioxSystemStore

複数の機器に対し、ドライバのクローズ、ハードウェアの終了処理、メモリバッファ解放を行い、設定ファイルに、機器設定（ADIOX_SYSTEM）を保存します。本関数で機器設定を保存するには、事前に vDbWrite で、DLL 内部の"設定構造体データベース"に反映しておく必要があります。

BOOL bADioxSystemStore（char *szFile , ADR_IP_CONF sADR_IP_CONF);

引数　*szFile　設定ファイルの場所を指定します。
sADR_IP_CONF　IP アドレスおよびポート番号を CARD_ID に割り付けた ADR_IP_CONF 構造体を指定します。

設定ファイルの構造　↑の bADioxSystemLoad と同じです。

ヒント　設定変更を行う必要がなければ、bADioxSystemLoad の時と同じファイルをコールするか、bADioxClose で終了してください。

戻り値　成功すると 1(TRUE)、失敗すると 0(FALSE)が返ります。

bADioxRetryOpen

運用中、エラーを返す機器は bADioxClose で切断すべきです。本関数は切断された機器にリトライ（再接続と初期化）をかけます。切断された機器に定期的にリトライすることで、自動回復を実現できます。リトライに失敗すると、応答待ちでロックするので、リトライ頻度は適切にします。

BOOL bADioxRetryOpen (ADR_IP_CONF sADR_IP_CONF, BYTE bCardID);

引数　bCardID　オープンしたい CARD_ID
sADR_IP_CONF　IP アドレスおよびポート番号を CARD_ID に割り付けた ADR_IP_CONF 構造体を指定します。

戻り値　成功すると 1(TRUE)、失敗すると 0(FALSE)が返ります。

bADioxBootStatus

指定した CARD_ID の機器が起動できたか確認します。複数機器でのデータ収集の場合、現在有効な CARD_ID と切断中の（リトライ対象の）CARD_ID の識別に使います。単一機器でも、状態を把握するのに使います。

BOOL bADioxBootStatus(BYTE bCardID, LPBYTE lpbErrorType);

引数　bCardID　起動確認したい CARD_ID
lpbErrorType　起動できたかどうかのステータスを格納したポインタ。値は、エラー要因がない（0）、設定ファイルに何らかのエラーがある（1）、接続に失敗した・・或いは接続できてもエラーとなった(2)です。

戻り値　成功すると 1(TRUE)、失敗すると 0(FALSE)が返ります。

vADioxErrorMessageStop

エラーメッセージを出さないようにします。エラーダイアログがポップアップして、データ収集を阻害するのを防ぎます。エラーで切断→リトライで自動復旧する仕組みを導入する場合は、エラーメッセージが出るのは好ましくないからです。

void vADioxErrorMessageStop(BOOL bStop);

引数　bStop　TRUE(＝1)でエラーメッセージを抑制します。FALSE(＝0)でエラーメッセージを有効にします。

3. 設定　(5)

bADioxConfigration

CARD_ID で指定した、機器設定（ADIOX_SYSTEM）をハードウェアに反映します。DLL 内部の"設定構造体データベース"には影響しません。

BOOL bADioxConfigration (struct ADIOX_SYSTEM *lpsTBufSetup,BYTE bCardID);

引数　*lpsTBufSetup　機器の設定値が入った ADIOX_SYSTEM 構造体を指定します。
bCardID　ターゲットデバイスのカード ID。

戻り値　成功すると 1(TRUE)、失敗すると 0(FALSE)が返ります。

vDbWrite

CARD_ID で指定した、機器設定（ADIOX_SYSTEM）を DLL 内部の"設定構造体データベース"に反映します。但しハードウェアにアクセスしません。bADioxSystemStore で設定を保存するには事前に本関数をコールしてください。

void　vDbWrite (struct ADIOX_SYSTEM sTBufSetup,BYTE bCardID);

引数　sTBufSetup　機器の設定値が入った ADIOX_SYSTEM 構造体を指定します。
bCardID　ターゲットデバイスのカード ID。

戻り値　成功すると 1(TRUE)、失敗すると 0(FALSE)が返ります。

vDbRead

DLL 内部の"設定構造体データベース"から、CARD_ID で指定した機器設定（ADIOX_SYSTEM）を読み出します。

void　vDbWrite (struct ADIOX_SYSTEM *lpsTBufSetup,BYTE bCardID);

引数　sTBufSetup　機器の設定値が入った ADIOX_SYSTEM 構造体を指定します。
bCardID　ターゲットデバイスのカード ID。

戻り値　成功すると 1(TRUE)、失敗すると 0(FALSE)が返ります。

bADioxScpSetup

機器設定(ADIOX_SYSTEM)の中から、信号調節に関する設定(センサーモード、校正位置、校正係数、スケーリング、アラーム)のみをハードウェアと DLL 内部の"設定構造体データベース"に反映します。

BOOL bADioxScpSetup (BOOL bExtentionAPI , struct ADIOX_SYSTEM * lpsTBufSetup , BYTE bCardID);

引数	bExtentionAPI	常に FALSE を指定してください。
	*lpsTBufSetup	機器の設定値(信号調節も含まれる)が入った ADIOX_SYSTEM 構造体を指定します。
	bCardID	ターゲットデバイスのカード ID。

戻り値 成功すると 1(TRUE)、失敗すると 0(FALSE)が返ります。

bADioxScpDefault

指定した CARD_ID、チャンネル番号、信号種別に対し、信号調節のデフォルト値を生成、DLL 内部の"設定構造体データベース"(機器設定(ADIOX_SYSTEM))に反映します。

BOOL bADioxScpDefault(DWORD dwSensorMode, BYTE bCardID, BYTE Bch);

引数	dwSensorMode	センサー種別
	bCardID	ターゲットデバイスのカード ID。
	dwCH	入力チャンネル番号(アナログなら 0-7、カウンタ・インフラサウンドなら 8-11 です)

戻り値 成功すると 1(TRUE)、失敗すると 0(FALSE)が返ります。

4.データ取得～信号変換(6)

bADioxIrqStatus

リングバッファデータ収集において、割り込みステータスを取得します。

BOOL bADioxIrqStatus (struct ADIOX_IRQ * lpsTIrqPack , BYTE bCardID);

引数	lpsTIrqPack	割り込み要因を格納した構造体 ADIOX_IRQ へのポインタを指定します。
	bCardID	ターゲットデバイスのカード ID。

戻り値 成功すると 1(TRUE)、失敗すると 0(FALSE)が返ります。

bADioxBufferRead

リングバッファから、128 サンプルのブロックデータを読み出します。リングバッファ割り込みメッセージを受信したら、本関数にアクセスして、バッファデータを取得してください。

```
BOOL bADioxBufferRead
      (
      LPDWORD lpdwAdData ,
      LPDWORD lpdwCtcData,
      LPDWORD lpdwDI,
      GPS_DATA_EX *lpsGPS_DATA,
      int * lpiBatteryLevel,
      double * lpdTemperature,
      BYTE bCardID,
      BOOL bInit );
```

引数	lpdwAdData	先頭から順番にアナログ入力データ・チャンネル 0～7(AI0～7)が格納され、これを 128 回繰り返します。アナログデータのサイズは 16Bit で、上位 16Bit は常に 0 です。16Bit のストレートバイナリデータは、プログラマがコード上でスケーリングするか、dADioxLinCoef でスケーリングしてください。8chX128 サンプル＝1024 の配列を確保してください。 各 ch の入力フォーマットは ADIOX_SYSTEM 構造体の dwSensorMode で指定します。
	lpdwCtcData	先頭から順番にカウンタ入力データ・チャンネル 0～3(CTC0～3)が格納され、これを 128 回繰り返します。一つのアナログデータのサイズは 32Bit です。32Bit のストレートバイナリデータは、プログラマがコード上でスケーリングするか、dADioxLinCoef でスケーリングしてください。4chX128 サンプル＝512 の配列を確保してください。 各 ch の入力フォーマットは ADIOX_SYSTEM 構造体の dwSensorMode で指定します。
	lpdwDI	デジタル入力を格納したポインタ。これは 1 サンプルだけです。下位 16Bit のみ有効で上位 16Bit は常に 0 です。インフラサウンドセンサでは、引数は無意味な数字が格納されます。
	*lpsGPS_DATA	GPS データを格納した GPS_DATA_EX 構造体へのポインタ
	*lpiBatteryLevel	バッテリレベルを格納したポインタ 0～100%指示です。インフラサウンドセンサでは無意味です。
	*lpdTemperature	機器温度を格納したポインタ。信号変換されて℃にスケーリングされた値が格納されます。
	bCardID	ターゲットデバイスのカード ID。
	bInit	TRUE を指定するとインフラサウンドセンサーで、このデータ列の先頭サンプル値をゼロに校正します。データ収集開始の最初だけ TRUE、以降は FALSE にしてください。でないと毎回ゼロ校正されてしまいます。

戻り値 成功すると 1(TRUE)、失敗すると 0(FALSE)が返ります。

※アナログ入力データ・チャンネル または カウンタ入力データ・チャンネルにインフラサウンドデータが割り付けられるケースがあります。

bADioxLastBankCtrl

リングバッファデータ収集における、最終バンクを読み出し可能にします。

BOOL bADioxLastBankCtrl (BOOL bBankLock , DWORD dwStopAddr , BYTE bCardID);

引数	bBankLock	TRUE で停止時のデータ収集側バンクを、読み出しに固定します。(そうしないとバンクが変化する) FALSE でバンク固定をフリーに戻します。つまりコードの流れ的には bADioxLastBankCtrl(TRUE ,stopaddr ,) bADioxBufferRead() bADioxLastBankCtrl(FALSE , ダミー,) ADIOX_IRQ **構造体の** dwStopAddr までをデータ有効範囲として処理
	dwStopAddr	停止したアドレス(0-127)を指定します。ADIOX_IRQ 構造体の dwStopAddr を指定します。
	bCardID	ターゲットデバイスのカード ID。

戻り値　成功すると 1(TRUE)、失敗すると 0(FALSE)が返ります。

bADioxBlockRead

各種データを一括して読み出します。本関数へのアクセスはソフトウェア側で判断してください。(つまりポーリングしてください)

```
BOOL bADioxBlockRead
        (
        LPDWORD lpdwTmp ,
        GPS_DATA_EX *lpsGPS_DATA,
        double *lpdTemp,
        int * lpiBatteryLevel,
        BYTE bCardID,
        BOOL bInit
        );
```

引数	lpdwTmp	先頭から順番にアナログ入力データ・チャンネル 0～7(AI0～7)、続いてカウンタ入力データ・チャンネル 0～3(CTC0～3)、デジタル入力データが格納されます。アナログデータサイズは 16Bit で上位 16Bit は常に 0 です。ストレートバイナリデータは、プログラマがコード上でスケーリングするか、dADioxLinCoef でスケーリングしてください。アナログ入力 8ch+カウンタ/インフラサウンド 4ch/デジタル入力で 13 配列を確保してください。各 ch の入力フォーマットは ADIOX_SYSTEM 構造体の dwSensorMode で指定します。
	*lpsGPS_DATA	GPS データを格納した GPS_DATA_EX 構造体へのポインタ
	*lpdTemp	機器温度を格納したポインタ。信号変換されて℃にスケーリングされた値が格納されます。
	*lpiBatteryLevel	バッテリレベルを格納したポインタ 0～100%指示です。インフラサウンドセンサでは無意味です。
	bCardID	ターゲットデバイスのカード ID。
	bInit	TRUE を指定するとインフラサウンドセンサーで、このサンプル値をゼロに校正します。データ収集開始の最初だけ TRUE、以降は FALSE にしてください。でないと毎回ゼロ校正されてしまいます。

戻り値　成功すると 1(TRUE)、失敗すると 0(FALSE)が返ります。

※アナログ入力データ・チャンネル　または　カウンタ入力データ・チャンネルにインフラサウンドデータが割り付けられるケースがあります。

dADioxLinCoef

bADioxBufferRead や bADioxBlockRead で取得した DWORD 型のアナログ・カウンタ・インフラサウンド入力値を本関数に引き渡すと、ゼロ&スパン校正の反映、リニアライザ、スケーリング、アラーム、バーンアウト検出などの処理を行い、結果を返します。

```
double dADioxLinCoef ( DWORD dwANALOG, double dTemp, LPBOOL lpbBurnOut,
                                    LPDWORD lpdwAlarm,BYTE bCardID,BYTE bCH );
```

引数	dwANALOG	アナログ・カウンタ・インフラサウンドなどの入力値をセットします
	dTemp	温度(℃)をセットします。熱電対などの零接点補償に使用します。
	lpbBurnOut	バーンアウト検出フラグで BOOL 型のポインタです。TRUE(=1)でバーンアウト発生。
	lpdwAlarm	アラーム検出フラグで BOOL 型のポインタです。TRUE(=1)でアラーム発生。
	bCardID	ターゲットデバイスのカード ID。
	bCH	入力チャンネル番号(アナログなら 0-7、カウンタ・インフラサウンドなら 8-11 です)

戻り値　変換された値

vADioxErrorReport

エラーを報告します。

```
void vADioxErrorReport ( LPDWORD lpdwSetError , LPDWORD lpdwWriteRetry,
        LPDWORD lpdwReadRetry,LPDWORD lpdwReadFlameError,LPDWORD lpdwReadAddressError );
```

引数	lpdwSetError	関数応答なしなどの最終エラーの回数を格納したポインタ
	lpdwWriteRetry	データ送信のリトライ回数を格納したポインタ
	lpdwReadRetry	データ受信のリトライ回数を格納したポインタ
	lpdwReadFlameError	データ送受信パケットのフレーム構造の崩壊回数を格納したポインタ。
	lpdwReadAddressError	データ送受信のアドレスベリファイエラー。

戻り値　0 が返ります。

5.校正(4)

bADioxZeroAdj

指定した CARD_ID、入力チャンネル番号におけるゼロ校正（＝オフセット校正）を行います。ゼロ校正位置は、bADioxScpSetup で設定します。本関数を実施する前に、対象となる入力に（ハードウェアに）ゼロ基準値を与えてください。より正確な校正を行うには、ゼロ校正とスパン校正を繰り返してください。

BOOL bADioxZeroAdj (BYTE bCardID, BYTE bChannel);

引数 bCardID ターゲットデバイスの CARD_ID。
bChannel アナログ入力チャンネル番号

戻り値 成功すると 1(TRUE)、失敗すると 0(FALSE)が返ります。

bADioxSpanAdj

指定した CARD_ID、チャンネル番号におけるスパン校正（＝ゲイン校正）を行います。スパン校正位置は、bADioxScpSetup で設定します。本関数を実施する前に、対象となるアナログ入力に（ハードウェアに）スパン基準値を与えてください。より正確な校正を行うには、ゼロ校正とスポン校正を繰り返してください。

BOOL bADioxSpanAdj (BYTE bCardID,BYTE bChannel);

引数 bCardID ターゲットデバイスの CARD_ID。
bChannel アナログ入力チャンネル番号

戻り値 成功すると 1(TRUE)、失敗すると 0(FALSE)が返ります。

vADioxFreeAdj

指定した CARD_ID、チャンネル番号を、校正をかけない状態、即ちゼロ校正量を 0 に、スパン校正量を 1 にします。

void vADioxFreeAdj (BYTE bCardID,BYTE bChannel);

引数 bCardID ターゲットデバイスの CARD_ID。
bChannel アナログ入力チャンネル番号

bADioxAutoZero

基板上のアンプや A/D コンバータのオフセットをゼロ校正します。本関数は頻繁には使わないでください。ポーリング、リングバッファ使用時にはゼロ点に段差ができるので使用しないでください。よほどの精度を求めなければ、本関数を呼び出す必要はありません。

BOOL bADioxAutoZero(BYTE bCardID);

引数 bCardID ターゲットデバイスのカード ID。

戻り値 成功すると 1(TRUE)、失敗すると 0(FALSE)が返ります。

6.高速タイマー(3)

vStartTimerIRQ

タイマー割り込みを開始します。このタイマーは、Windows の標準タイマーよりも高速、かつ精密です。

C/C++ void vStartTimerIRQ (UINT uiPacerDelay,HWND hmWnd);
引数 uiPacerDelay タイマー割り込み間隔を 1msec 単位で指定します。
hmWnd タイマー割り込み（メッセージ）伝達する対象のウィンドウハンドル。タイマー割り込みが発生すると、このウィンドウハンドルに対し WM_USER+2020 のメッセージが送られます。

vStopMtimerIRQ

vStartTimerIRQ で開始したタイマー割り込みを停止します。

C/C++ void vStopMtimerIRQ ( );

vReserMtimerIRQ

vStartTimerIRQ で開始したタイマー割り込みにおいて、割り込み処理が完了したことを通知します。この関数をコールするまでは、次のタイマー割り込みは再開されません。

C/C++ void vReserMtimerIRQ ( );

7.構造体(4)

ADIOX_SYSTEM

ハードウェア、信号調節の全設定を網羅した設定構造体。

```
struct ADIOX_SYSTEM
        {
        DWORD   dwClockScall;

        // AI TRIG
        DWORD   dwStartTrigDelay;
        DWORD   dwStartTrigLevel1;
        DWORD   dwStartTrigLevel2;
        DWORD   dwStopTrigDelay;
        DWORD   dwStopTrigLevel1;
        DWORD   dwStopTrigLevel2;

        // DI  TRIG
        DWORD   dwStartMask;
        DWORD   dwStartDiPattern;
        DWORD   dwStartTrigSourceDI_ch;
        DWORD   dwStopMask;
        DWORD   dwStopDiPattern;
        DWORD   dwStopTrigSourceDI_ch;

        // TRIG MODE
        DWORD   dwTrigStopMode;
        DWORD   dwTrigStartMode;

        // TRIG MISC
        DWORD   dwIntrruptMode;
        DWORD   dwDeadTime;

        // STOP COUNTER
        DWORD   dwStopCounterValue;

        // INFRA SOUND
        double    dSetCoef;
        DWORD   dwInfraSoundMode;

        // COUNTER
        DWORD   dwChatCan;
        DWORD   dwLatchMode_A;
        DWORD   dwLatchMode_B;
        DWORD   dwLatchMode_C;
        DWORD   dwLatchMode_D;
        DWORD   dwZ_CENTER_A;
        DWORD   dwZ_CENTER_B;
        DWORD   dwZ_CENTER_C;
        DWORD   dwZ_CENTER_D;
        DWORD   dwSoftwareClear_A;
        DWORD   dwSoftwareClear_B;
        DWORD   dwSoftwareClear_C;
        DWORD   dwSoftwareClear_D;

        // AO,DO
        DWORD   dwAo0;
        DWORD   dwAo1;
        DWORD   dwDOS;

        // AI MODE
        DWORD   dwInputShort;
        DWORD   dwFilterEnable;
        DWORD   dwPeakholdCh;

        // 信号調節
        DWORD   dwSensorMode[MAX_AI_CH];
        double    doZeroPos[MAX_AI_CH];
        double    doSpanPos[MAX_AI_CH];
        double    doZero_Coefficient[MAX_AI_CH];
        double    doSpan_Coefficient[MAX_AI_CH];
        BOOL      bScalling[MAX_AI_CH];
        double    dOutTopScall[MAX_AI_CH];
        double    dOutBottomScall[MAX_AI_CH];
        double    dInTopScall[MAX_AI_CH];
        double    dInBottomScall[MAX_AI_CH];
        DWORD   bAlarmMode[MAX_AI_CH];
        double    dAlarmUpper[MAX_AI_CH];
        double    dAlarmLower[MAX_AI_CH];
        double    dScallingRatio[MAX_AI_CH];
        };
```

メンバ変数

[サンプリング周波数]

dwClockScall　A/Dコンバーターのサンプリング周波数を指定します。常時 8ch のマルチプレクスでサンプリングします。サンプリング周波数＝ (460.8KHz÷dwClockScall)、有効範囲 0x17～0x1FFFFFF。

[トリガモード]

dwTrigStopMode　ストップトリガを指定します。下記 7 つの中から選択してください。

dwTrigStartMode　スタートトリガを指定します。下記 7 つの中から選択してください。

RESET	(0x0)	トリガ条件は成立しない
BURST	(0x1)	無条件トリガ
DI_POSEDGE	(0x2)	デジタル入力の立ち上がりエッジでトリガ
DI_NEGEDGE	(0x3)	デジタル入力の立ち下がりエッジでトリガ
DI_PATTERN	(0x4)	デジタル入力が指定パターンとなったらトリガ
AI_LEVEL	(0xB)	アナログ入力のレベル（エッジ）トリガ
AI_AREA	(0xC)	アナログ入力のエリアトリガ

トリガなしなら dwTrigStartMode＝BURST、dwTrigStopMode＝RESET にします。
これで他のトリガディレイを除きトリガ関係の引数は無意味になります。

[アナログトリガ]

dwStartTrigLevel1　アナログレベルトリガ・アナログエリアトリガ用のスタートトリガレベル 1 の指定。（※1）
dwStartTrigLevel2　アナログレベルトリガ・アナログエリアトリガ用のスタートトリガレベル 2 の指定。（※1）
dwStopTrigLevel1　アナログレベルトリガ・アナログエリアトリガ用のストップトリガレベル 1 の指定。（※1）
dwStopTrigLevel2　アナログレベルトリガ・アナログエリアトリガ用のストップトリガレベル 2 の指定。（※1）

[トリガディレイ]

dwStartTrigDelay　スタートトリガディレイ。値 1 で 1/8 サンプリング時間で、それだけトリガより遅れてデータ収集を開始します。0x0～0xFFFF まで指定できます。

dwStopTrigDelay　ストップトリガディレイ。値 1 で 1/8 サンプリング時間で、それだけトリガより遅れてデータ収集を停止します。0x0～0xFFFF まで指定できます。

[デジタルエッジトリガ]

dwStartTrigSourceDI_ch　デジタルエッジ・スタートトリガ用のチャンネル指定。チャンネル数即ち 0-15 のいずれかを指定します。
dwStopTrigSourceDI_ch　デジタルエッジ・ストップトリガ用のチャンネル指定。チャンネル数即ち 0-15 のいずれかを指定します。

[デジタルパターントリガ]

dwStartMask　デジタルパターンスタートトリガ用のマスク。この変数のビットフィールドが、デジタル入力のチャンネルに相当します。（※3）該当ビットが 1 でマスク、0 でアンマスクです。

dwStopMask　デジタルパターンストップトリガ用のマスク。この変数のビットフィールドが、デジタル入力のチャンネルに相当します。（※3）該当ビットが 1 でマスク、0 でアンマスクです。

dwStartDiPattern　デジタルパターンスタートトリガ用のトリガパターンを指定します。この値とデジタル入力値が一致した場合に、トリガが成立します。

dwStopDiPattern　デジタルパターンストップトリガ用のトリガパターンを指定します。この値とデジタル入力値が一致した場合に、トリガが成立します。

[トリガ・リングバッファ開始停止・自動停止]

dwIntrruptMode　リングバッファ開始・停止。DMA_INT(1)で開始、NOT_INT(0)で停止です。

dwStopCounterValue　この変数で指定した分のバンク数をデータ収集すると自動停止します。0 を指定すると、本自動停止機能は無効となり、停止トリガもしくは停止コマンドが実施されるまで無制限にデータ収集を行います。

dwDeadTime　スタートトリガ有効後、ストップトリガ検出が有効になるまでの時間を指定します。いきなりストップトリガがかかってしまうのを防ぐためです。値は 1/8 サンプリング時間で、0-0xFFFFFFFF まで有効です。

[インフラサウンド]

dSetCoef　インフラサウンド校正係数（感度を指定します）、例えば 2017 年式は 2.0 です。

dwInfraSoundMode　インフラサウンドセンサーの場合 1 を指定してください。それ以外は 0 です。

[エンコーダーカウンタ]

dwChatCan　1 でチャタリングキャンセラーを on、0 で off にします。

dwLatchMode_A　カウンター0 ラッチモードを以下の 3 つの中から指定します。
dwLatchMode_B　カウンター1 ラッチモードを以下の 3 つの中から指定します。
dwLatchMode_C　カウンター2 ラッチモードを以下の 3 つの中から指定します。
dwLatchMode_D　カウンター3 ラッチモードを以下の 3 つの中から指定します。

SOFT	(0x0)	ソフトウェアラッチ
Z_PHASE	(0x1)	Z 相条件成立でラッチ
DI_SEL	(0x2)	Y 相立ち上がりエッジでラッチ

dwZ_CENTER_A　カウンター0、Z 相条件成立モード（カウンターリセット）を以下の 0-1 のいずれかで指定してください。
dwZ_CENTER_B　カウンター1、Z 相条件成立モード（カウンターリセット）を以下の 0-1 のいずれかで指定してください。
dwZ_CENTER_C　カウンター2、Z 相条件成立モード（カウンターリセット）を以下の 0-1 のいずれかで指定してください。
dwZ_CENTER_D　カウンター3、Z 相条件成立モード（カウンターリセット）を以下の 0-1 のいずれかで指定してください。
1: CCW 方向：AZ 相が 1 の時 B 相立下り, CW 方向：BZ 相が 1 の時 A 相立下り
0:Z 相立ち上がり条件で、カウンターリセット

dwSoftwareClear_A　dwLatchMode_A を SOFT とした場合、この変数が 1 でカウンターリセット、0 で非リセット。
dwSoftwareClear_B　dwLatchMode_B を SOFT とした場合、この変数が 1 でカウンターリセット、0 で非リセット。
dwSoftwareClear_C　dwLatchMode_C を SOFT とした場合、この変数が 1 でカウンターリセット、0 で非リセット。
dwSoftwareClear_D　dwLatchMode_D を SOFT とした場合、この変数が 1 でカウンターリセット、0 で非リセット。

[出力]

dwAo0　アナログ出力チャンネル 0。0x0～0xFFFF を指定します。
dwAo1　アナログ出力チャンネル 1。0x0～0xFFFF を指定します。
dwDOS　デジタル出力チャンネルの出力値。この変数のビットフィールドが、デジタル入力のチャンネルに相当します。(※3)最大 16ch の DO が実装されるので、0～0xFFFF を指定します。

[アナログ・ハードウェア機能]

dwInputShort　0 で通常のアナログ入力、1 で入力をグラウンドにショートします。
dwFilterEnable　アナログ入力信号へのデジタルフィルタの切り替えを行います。0 で Off,1 で On です。
dwPeakholdCh　ピークホールドを指定します。この変数のビットフィールドが、アナログ入力のチャンネルに相当します。1 で on、0 で off です。リングバッファデータ収集の場合必ず off にしてください。
ポーリングではデータを取得～次の取得までの最高値を記録し、それをデータとします。

[信号調節]

dwSensorMode[*]　入力チャンネル毎に信号種類を指定します。アンプのゲインやスケーリング、リニアライザが自動設定されます。例えば CA_K にすると、bADioxBufferRead や bADioxBlockRead で取得したデータを dADioxLinCoef に通すだけで自動的に℃に変換されます。設定可能な定義を以下に示します。尚、末尾に赤字のあるものは、チャンネル割付が固定になっています。

[チャンネル 0～7]に設定可能な信号源

定義	値(10 進)	内訳
NOT_USE	0	シグナルコンディション未使用
CA_K	1	熱電対 K 1400～-240℃
CA_Kb	2	熱電対 K 250～0℃低雑音
CA_J	3	熱電対 J 1200～-200℃
CA_Jb	4	熱電対 J 190～0℃ 低雑音
CA_E	5	熱電対 E 1000～-240℃
CA_Eb	6	熱電対 E 155～0℃ 低雑音
CA_T	7	熱電対 T 400～-200℃
CA_Tb	8	熱電対 T 215～0℃ 低雑音
CA_R	9	熱電対 R 1760～-50℃
CA_Rb	10	熱電対 R 955～0℃ 低雑音
CA_S	11	熱電対 S 1300～-200℃
CA_Sb	12	熱電対 S 1040～0℃ 低雑音
CA_N	13	熱電対 N 1400～-240℃
CA_Nb	14	熱電対 N 320～0℃ 低雑音
CA_B	15	熱電対 B 1800～200℃
CA_Bb	16	熱電対 B 1495～0℃ 低雑音
PT100	17	白金測温抵抗体 Pt100 800～0℃
JPT100	18	白金測温抵抗体 JPt100 500～0℃
VBP_10mV	19	電圧±10mV レンジ
VBP_100mV	20	電圧±100mV レンジ
VBP_1V	21	電圧±1V レンジ
VBP_10V	22	電圧±10V レンジ
I_4_20	23	電流 4-20mA/500Ω 終端
I_4_20EX	24	電流 4-20mA/350Ω 終端
I_4_20EX2	25	電流 4-20mA/47Ω オンボード終端
D16BIT	26	0-65535 のデジタル値のこと
以下はインフラサウンドセンサー**ADXⅢ-INF01LE** 機能		
VIB	46	加速度計 XYZ(地震) 必ず AI0-2(Gal)
SPL	47	騒音計(Z 特性) 必ず AI3(dB/SPL)
APL	48	気圧計 必ず AI4(KPa)
以下はインフラサウンドセンサー**ADXⅢ-INF04LE** 機能		
VIB	46	加速度計 XYZ(地震) 必ず AI0-2(Gal)
INF04_HF	49	インフラサウンド HF (0.001Hz-1000Hz) 必ず AI3(mPa)

[チャンネル 8～11]に設定可能な信号源

定義	値	内訳
EC_4X	27	4 倍速エンコーダカウンタ Z 未使用
EC_4XZ	28	4 倍速エンコーダカウンタ Z 使用
EC_2X	29	2 倍速エンコーダカウンタ Z 未使用
EC_2XZ	30	2 倍速エンコーダカウンタ Z 使用
EC_1X	31	1 倍速エンコーダカウンタ Z 未使用
EC_1XZ	32	1 倍速エンコーダカウンタ Z 使用
UPC	33	アップダウンカウンタ Z 未使用
UPC_Z	34	アップダウンカウンタ Z 使用
以下はインフラサウンドセンサー**ADXⅢ-INF01LE** 機能		
INFRS_TA	37	温度センサー 必ず CTC2 (℃)
INFRS_FB	36	インフラサウンド DC 必ず CTC0 (mPa)
INFRS_DIF	41	インフラサウンド AC 必ず CTC1 (mPa)

以下はインフラサウンドセンサー**ADXⅢ-INF04LE** 機能

INF04_LF	50	インフラサウンド LF（気圧） 必ず CTC0(hPa)
INF04-TMP	51	温度センサー 必ず CTC1 （℃）

[信号調節校正]

doZeroPos[*]	ゼロ校正位置を"MAX_AI_CH"（※2）個格納します。
doSpanPos[*]	スパン校正位置を" MAX_AI_CH"（※2）個格納します。
doZero_Coefficient[*]	ゼロ校正係数を" MAX_AI_CH"（※2）個格納します。
doSpan_Coefficient[*]	スパン校正係数を" MAX_AI_CH"（※2）個格納します。

[信号調節スケーリング]

bScalling[*]	スケーリングする場合 TRUE、しない場合 FALSE をセットします。 "MAX_AI_CH" （※2）個格納します。
dOutTopScall[*]	変換後のスケーリング基準値（上）。これを"MAX_AI_CH"（※2）個格納します。
dOutBottomScall[*]	変換後のスケーリング基準値（下）。これを"MAX_AI_CH"（※2）個格納します。
dInTopScall[*]	変換前のスケーリング基準値（上）。これを"MAX_AI_CH"（※2）個格納します。
dInBottomScall[*]	変換前のスケーリング基準値（下）。これを"MAX_AI_CH"（※2）個格納します。
dScallingRatio[*]	内部で使用する変数です。操作しないでください。

例えば、4～20mA を 1000～10000rpm にする場合，
dOutTopScall=10000、dOutBottomScall＝1000、dInTopScall＝20、dInBottomScall＝4、bScalling=TRUE

[信号調節アラーム]

bAlarmMode[*]	アラームモードを指定します。0 を指定すると:オフ、1 を指定すると:dAlarmUpper 以上でアラーム（オーバー）、2 を指定すると dAlarmLower 以下でアラーム（アンダー）、3 を指定すると dAlarmUpper ～ dAlarmLower の範囲内でアラーム（インレンジ）、4 を指定すると dAlarmUpper～dAlarmLower の範囲内でアラーム（アウトレンジ）になります。 これを"MAX_AI_CH"（※2）個格納します。
dlarmUpper[*]	アラーム設定値（上）を"MAX_AI_CH"（※2）個格納します。
dAlarmLower[*]	アラーム設定値（下）を"MAX_AI_CH"（※2）個格納します。

※1　アナログレベル最小～最大が、0～0xFFFF に対応
※2　MAX_AI_CH は 12 で、0-7 がアナログ入力、8-11 がカウンタ及びインフラサウンドです。
※3　例えば Bit5(0x20)は、デジタル入出力チャンネル 5 に相当します。

ADIOX_IRQ

割り込みステータスを格納します。

```
struct ADIOX_IRQ
        {
        DWORD   dwBankCount;
        DWORD   dwTrigSeq;
        DWORD   dwBankAddr;
        DWORD   dwIrq;
        DWORD   dwStopAddr;
        };
```

メンバ変数

dwBankCount	リングバッファのバンクチェンジ数（割り込み発生数）を格納します。 この数と、割り込み発生数が異なれば、バッファオーバーランで、サンプリングが速すぎです。		
dwTrigSeg	トリガ・リングバッファエンジンの状態		
	TRIG_IDLE	0x0	停止中または開始トリガ待ち
	TRIG_RUN	0x1	データ収集中
	TRIG_TURN	0x2	停止トリガ待ち
	TRIG_HIST	0x3	デッドタイム中
dwBankAddr	リングバッファのバンクを示す		
dwIrq	割り込み発生状況、1 で発生→すぐにリングバッファのデータを取得してください。		
dwStopAddr	リングバッファの停止アドレス （どこで停止したのか）		

ADR_IP_CONF

CARD_ID に対し、IP アドレスとポート番号を割り付けます。

```
struct ADR_IP_CONF
        {
        int         iIP1[MAX_MFIO];
        int         iIP2[MAX_MFIO];
        int         iIP3[MAX_MFIO];
        int         iIP4[MAX_MFIO];
        int         iPORT[MAX_MFIO];
        BOOL        bEnable[MAX_MFIO];
        };
```

メンバ変数

iIP1[*]	[配列番号＝CARD_ID]に対する IP アドレス、1 桁目（192.168.1.50 なら 192）
iIP2[*]	[配列番号＝CARD_ID]に対する IP アドレス、2 桁目（192.168.1.50 なら 168）
iIP3[*]	[配列番号＝CARD_ID]に対する IP アドレス、3 桁目（192.168.1.50 なら 1）
iIP4[*]	[配列番号＝CARD_ID]に対する IP アドレス、4 桁目（192.168.1.50 なら 50）
iPORT[*]	[配列番号＝CARD_ID]に対するポート番号
bEnable[*]	[配列番号＝CARD_ID]の機器を有効にするか否か、TRUE で有効、FALSE で無効。

GPS_DATA_EX

インフラサウンドセンサにおける GPS データを格納します。

```
struct GPS_DATA_EX
        {
        DWORD   dwYear;
        DWORD   dwMonth;
        DWORD   dwDay;
        DWORD   dwHour;
        DWORD   dwMinute;
        double  dSecond;
        };
```

メンバ変数

dwYear	年
dwMonth	月
dwDay	日
dwHour	時
dwMinute	分
dSecond	秒（小数点 3 桁）