MULTIFUNCTION-I/O-X2 SERIES
ADIOX2-API
REFERENCE

UPDATE 2016-2-17

SAYA Inc.

目次

はじめに

ADiox2-API

ADiox2-API は、MultifunctionI/O-X2 シリーズおよびインフラサウンドマルチセンサーを効率よく使うために開発された API です。ADiox2-API は、ハードウェアの制御にとどまらず、計測アプリケーションを構築するのに必要な、FFT、信号調節、波形生成、高速画面描画アシスト、計測ファイル保存、CSV データロガー、トレンドグラフなどを統合しており、効率的な開発が可能です。

① ハードウェアドライバの設定と計測制御などの中枢機能
② 機能設定の保存・読出し・管理
③ 信号調節・信号解析（FFT、センサインターフェース、校正、スケーリング、アラーム）
④ 計測ファイルのバイナリ高速保存、および計測ファイル CSV 保存（ロガー機能）
⑤ 波形生成
⑥ 複数 ADXⅡ42C***、ADXⅡ-INF01 のサバイバル機能（生存中のハードウェアを探し動的に計測グループを変えていく）

スケーラビリティ

ADiox2-API は、ADXⅡ14-125M-PCIEX、ADXⅡ85-1M-PCIEX、ADXⅡ42C-2K-Ethernet、ADXⅡ42C-WiFi、ADXⅡ42C-CORE、ADXⅡ42FE-250K-Ethernet、ADXⅡ42FE-MPU、ADXⅡ42FE-CORE、ADXⅡ-INF01、ADXⅡINF-02 など MultifunctionI/O-XⅡ シリーズをサポートします。対応言語も、C,C++,C#,Basic と広範囲です。**以降の解説では、**
ADXⅡ42C-2K-Ethernet、ADXⅡ42C-WiFi、ADXⅡ42C-CORE、ADXⅡ42FE-250K-Ethernet、ADXⅡ42FE-MPU、ADXⅡ42FE-CORE **をまとめて ADXⅡ42***と呼ぶ場合があります。また、**ADXⅡ-INF01、ADXⅡINF-02 をまとめて **ADXⅡINF*** と呼ぶ場合があります。

実装されていないボードへのアクセス

実装されていないボードへアクセスした場合には、関数は何も実行されずに、FALSE(=0)を返します。

開発用ファイル

VisualC++、C++、C 用

64bit 系の Windows は　CDROM¥MFIO_X2¥sdk¥VisuaCPP_X64
32bit 系の Windows は　CDROM¥MFIO_X2¥sdk¥VisuaCPP_X86
をお使いください。

ADiox2.h／ADiox2.LIB　[ADiox2.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。

ADioxCsvm.h／ADioxLogm.LIB／[ADioxLogm.dll]
　CSV ロガー用ヘッダファイル、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。

ADioxTrlogMl.h／ADioxTrlogMl.LIB／[ADioxTrlogMl.dll]
　トレンドグラフ用ヘッダファイル、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。

ADioxReportMl.h／ADioxReportMl.lib／[ADioxReportMl.dll]
　リポート用ヘッダファイル、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。

VisualC#用

“CDROM¥MFIO_X2¥sdk¥VisualC#”にあるファイルをお使いください。

AdioxLibrary2.cs／[ADiox2.dll+ ADioxScp.dll]
　メインライブラリのクラスライブラリです。プロジェクトのフォルダ内にコピーして、プロジェクトに「既存項目の追加」で追加してください。そして各フォーム等のコードの最上部に、“using Saya.AdioxLibrary;”というネームスペースを追加記述してください。
　.netFramework2.0 以降に対応します。

シンプルなサンプルソース

ADiox2-API の使用方法を理解しやすいよう、ポーリングとバッファの 2 カテゴリで、言語別のサンプルソースを提供しています。

CardId について

CardID=0～3　　が、ADXⅡ85-1M-PCIEX に、
CardId=4～27　　が、ADXⅡ42***、ADXⅡ-INF** に、
CardId=28　　が、ADXⅡ14-125M-PCIEX に割り当てられます。

非公開の API も問い合わせください

ヘッダファイルには本マニュアル以外の API も記載されています。これらについては、お問い合わせください。

1.初期化・再初期化 [ADiox2.dll]

vSetupTcpIp

ADXⅡ42***、ADXⅡINF***専用の関数です。IPアドレスおよびポート番号を、CARD_IDに割り付けます。本関数は、初期化関数をコールする前に実行してください。

C/C++ void vSetupTcpIp(int IP1, int IP2, int IP3,int IP4, int Port, BYTE bCardID);

C# void vSetupTcpIp(int IP1, int IP2, int IP3, int IP4, int Port, byte bCardID);

引数	IP1	IP アドレス設定（例えば IP アドレスが 192.168.0.2 なら 192 を指定してください）
	IP2	IP アドレス設定（例えば IP アドレスが 192.168.0.2 なら 168 を指定してください）
	IP3	IP アドレス設定（例えば IP アドレスが 192.168.0.2 なら 0 を指定してください）
	IP4	IP アドレス設定（例えば IP アドレスが 192.168.0.2 なら 2 を指定してください）
	Port	ポート番号を指定してください。
	bCardID	ターゲットデバイスのカード ID。

bADioxOpen2

ドライバのオープン、ハードウェアの初期化、システムメモリへのバッファ確保等を行います。

C/C++ BOOL bADioxOpen2 (HWND hWnd, BYTE bHwintr, BYTE bCardID);

C# int bADioxOpen2 (int hWnd, byte bHwintr, byte bCardID);

引数	hWnd	ドライバからのメッセージを受け取るアプリケーションのウィンドウハンドルを設定します。
	bHwintr	前記メッセージ番号を指定します。本引数 0～15 に対し、メッセージ 3028～3043 番が割り当てられます。この値は Adiox2.h、ADiox2Library.cs、ADIOX-API_VB.bas にて 3028～3043 番を、それぞれ、WM_HWINTR0～WM_HWINTRF として定義しています。（末尾の 0～F は本引数 0～15 の 16 進数に相当する）
	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。

bADioxScpLoad2

複数の ADXⅡ42***、ADXⅡINF***に対し、ドライバのオープン、ハードウェアの初期化、システムメモリへのバッファ確保を行います。**ポーリング向けです。**ハードウェアの初期化は、TXBUFSETUP2、IOGEOSETUP2、TDIO_MISC、TConfigPWM、TSetupPWM、TADIO2、SCP_SETUP_AIALL（センサーモード、ゼロスパン校正位置、ゼロスパン校正係数、スケーリング、アラーム）など全機能に及びます。これらの情報は、SCP_SETUP2 構造体を格納したコンフィグレーションファイル(拡張子 scp)から取得します。

C/C++ BOOL bADioxScpLoad2(HWND hWnd,BYTE bWintr,CharPayloadC *lpsCharPayload,BOOL bOpenDialog);

C# int bADioxScpLoad2 (int hWnd, byte bWintr , CharPayloadC *lpsCharPayload , int bOpenDialog);

引数	hWnd	ドライバからのメッセージを受け取るアプリケーションのウィンドウハンドルを設定します。
	bHwintr	前記メッセージ番号を指定します。本引数 0～15 に対し、メッセージ 3028～3043 番が割り当てられます。この値は ADiox2.h、ADiox2Library.cs、ADIOX-API_VB.bas にて 3028～3043 番を、それぞれ、WM_HWINTR0～WM_HWINTRF として定義しています。（末尾の 0～F は本引数 0～15 の 16 進数に相当する）
	bOpenDialog	コンフィグレーションファイルを ADiox2.dll 内蔵の“ファイルを開くダイアログボックス”で指定する場合、本引数を TRUE(=1)とします。本引数を FALSE(=0)とした場合、lpsCharPayload.lpcConfigFileName で指定します。“ファイルを開くダイアログボックス”の初期フォルダは lpsCharPayload.lpcInitialDir で指定します。
	lpsCharPayload	コンフィグレーションファイル名、もしくは、“ファイル保存ダイアログボックス”のベースフォルダ名を指定する構造体 CharPayloadC(VB 用は CharPayloadB)へのポインタ。

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

bADioxLoad_EX3

単一の MultifunctionI/O に対し、ドライバのオープン、ハードウェアの初期化、メモリ確保、割り込みの使用許可を行います。**リングバッファおよびポーリングの双方の運用に適します。**引数または当関数内蔵の“ファイルを開くダイアログボックス”にて指定した、コンフィグレーションファイル（拡張子 adx2 の設定ファイル）から設定値を取得、ハードウェアと引数に反映します。コンフィグレーションファイルを指定しなかった場合には、デフォルト設定を作成し、これを引数とハードウェアに反映します。

C/C++
```
BOOL bADioxLoad_EX3
        (
        HWND    hWnd,
        BYTE    bWintr,
        BOOL    bOpenDialog,
        SAYA_DEVICE_INFO    * lpsSAYA_DEVICE_INFO,
        TXBUFSETUP2    *lpsTBUFSETUP,
        IOGEOSETUP2    *lpsIOGEOSETUP,
        TDIO_MISC    *lpsTDIO_MISC,
        TADIO2    *lpsTADIO,
        TConfigPWM    *lpsTConfigPWM,
        TSetupPWM    *lpsTSetupPWM,
        SCP_SETUP_AIALL    *lpsScpSetupAich,
        ADIOX_EXTENTION2    * lpsEXTENTION,
        DWORD    dwRingbufferSize,
        BYTE    CARD_ID,
        CharPayloadC    *lpsCharPayload
        );
```

C#
```
int bADioxLoad_EX3
        (
        int    hWnd,
        byte    bWintr,
        int    bOpenDialog,
        ref    SAYA_DEVICE_INFO    lpsSAYA_DEVICE_INFO,
        ref    TXBUFSETUP2    lpsTBUFSETUP,
        ref    IOGEOSETUP2    lpsIOGEOSETUP,
        ref    TDIO_MISC    lpsTDIO_MISC,
        ref    TADIO2    lpsTADIO,
        ref    TConfigPWM    lpsTConfigPWM,
        ref    TSetupPWM    lpsTSetupPWM,
        ref    SCP_SETUP_AIALL    lpsScpSetupAich,
        ref    ADIOX_EXTENTION2    lpsEXTENTION,
        uint    dwRingbufferSize,
        byte    bCARD_ID,
        ref    CharPayloadC    lpsCharPayload
        );
```

引数

hWnd	ドライバからのメッセージを受け取るアプリケーションのウィンドウハンドルを設定します。
bHwintr	前記メッセージ番号を指定します。本引数 0～15 に対し、メッセージ 3028～3043 番が割り当てられます。この値は Adiox2.h、ADiox2Library.cs、ADIOX-API_VB.bas にて 3028～3043 番を、それぞれ、WM_HWINTR0～WM_HWINTRF として定義しています。（末尾の 0～F は本引数 0～15 の 16 進数に相当する）
bOpenDialog	コンフィグレーションファイルを当関数内蔵の“ファイルを開くダイアログボックス”で指定する場合、本引数を TRUE(＝1)とします。FALSE(＝0)とした場合、lpsCharPayload. lpcConfigFileName でファイルを直接指定します。“ファイルを開くダイアログボックス”の初期フォルダは lpsCharPayload.lpcInitialDir で指定します。
lpsSAYA_DEVICE_INFO	デバイス情報構造体 SAYA_DEVICE_INFO へのポインタ。
lpsTBUFSETUP	コンフィグレーションファイルにより反映された TXBUFSETUP2 構造体へのポインタ。
lpsIOGEOSETUP	コンフィグレーションファイルにより反映された IOGEOSETUP2 構造体へのポインタ。
lpsTDIO_MISC	コンフィグレーションファイルにより反映された TDIO_MISC 構造体へのポインタ。
lpsTADIO	コンフィグレーションファイルにより反映された TADIO2 構造体へのポインタ。
lpsTConfigPWM	コンフィグレーションファイルにより反映された TConfigPWM 構造体へのポインタ。
lpsTSetupPWM	コンフィグレーションファイルにより反映された TSetupPWM 構造体へのポインタ。
lpsScpSetupAich	コンフィグレーションファイルにより反映された SCP_SETUP_AIALL 構造体へのポインタ。この引数は信号調節機能付き MultifunctionI/O でのみ意味があります。
lpsEXTENTION	コンフィグレーションファイルにより反映された ADIOX_EXTENTION2(VB 用は ADIOX_EXTENTION2B)構造体へのポインタ。
dwRingbufferSize	セカンダリリングバッファの倍率。この引数は ADXⅡ85-1M-PCIEX のみ意味があります。
bCardID	ターゲットデバイスのカード ID。
lpsCharPayload	コンフィグレーションファイル名、もしくは、“ファイル保存ダイアログボックス”のベースフォルダ名を指定する構造体 CharPayloadC(VB 用は CharPayloadB)へのポインタ。

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

vADioxIrqEmuration

ADXⅡ42C-2K-Ethernet、ADXⅡ42C-WiFi、ADXⅡ42C-CORE、ADXⅡ-INF01 専用の関数です。イーサネットでは PCI バスのように割り込みがありませんが、ADiox2-API では、ドライバで割り込みエミュレーションを行います。本関数は割り込みエミュレーション機能をディセーブルにして負荷を下げます。割り込みの必要ないポーリング場合に、本関数をご使用ください。必ず初期化前に実行してください。本関数を呼び出さない場合、割り込みエミュレーションが有効になります。

C/C++ void vADioxIrqEmuration (BOOL bEnableIrqEmuration);

C# void vADioxIrqEmuration (int bEnableIrqEmuration);

引数	bEnableIrqEmuration	TRUE(=1)で割り込みエミュレーションを有効にします。FALSE(=0)で割り込みエミュレーションを無効にします。

bADioxScpRetry

ADXⅡ42***、ADXⅡINF***が電源遮断や通信途絶によってロストした場合、リトライをかけます。(＝再接続と初期化を試みる)これは複数のリモート I/O でデータ収集中に、一部のリモート I/O がロストしても、残りのリモート I/O で運用を続行したい場合に有効です。リトライに失敗すると、TCP/IP プロトコルスタックがタイムアウト待ちの為に無応答になるので、頻繁にリトライを行うのは避けてください。

C/C++ BOOL bADioxScpRetry (HWND hWnd, BYTE bWintr, BYTE bCardID);

C# int bADioxScpRetry (int hWnd, byte bWintr, byte bCardID);

引数	hWnd	ドライバからのメッセージを受け取るアプリケーションのウィンドウハンドルを設定します。
	bHwintr	前記メッセージ番号を指定します。本引数 0～15 に対し、メッセージ 3028～3043 番が割り当てられます。この値は Adiox2.h、ADiox2Library.cs、ADIOX-API_VB.bas にて 3028～3043 番を、それぞれ、WM_HWINTR0～WM_HWINTRF として定義しています。(末尾の 0～F は本引数 0～15 の 16 進数に相当する)
	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。

vADioxErrorMessageStop

ADXⅡ42***、ADXⅡINF***の通信エラーに関わるエラーメッセージを全て出さないようにします。例えば bADioxScpRetry を使って複数のリモート I/O のうち生き残っているものだけでデータ収集を行い、リトライで復旧したリモート I/O をデータ収集に参加させるような場合、ロストするときにも、リトライ時失敗時にもエラーメッセージが出てしまいます。本関数はこうしたエラーメッセージを抑制します。

C/C++ void vADioxErrorMessageStop(BOOL bStop);

C# void vADioxErrorMessageStop(int 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。

bADioxGetBootStatus

ADXⅡ42***、ADXⅡINF***が起動できたか確認します。複数の中で生存中の機器だけでデータ収集を行いたい場合、途中でロスト・リトライなどが考えられる場合、本関数で起動時にロストしているか否かを確認できます。ロストしている場合は、bADioxScpRetry で復活させられる可能性があります。

C/C++ BOOL bADioxGetBootStatus(BYTE bCardID, LPBYTE lpbErrorType);

C# int bADioxGetBootStatus(byte bCardID, ref byte 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 0 :エラー要因がない。 1 :設定ファイルに何らかのエラーがある。 2 :接続に失敗した。或いは接続できてもエラーとなった。

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

2. 終了処理 [ADiox2.dll]

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。

C/C++ BOOL bADioxClose(BYTE bCardID);

C# int 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。

bADioxScpStore2

複数の ADXⅡ42***、ADXⅡINF***に対し、ドライバのクローズ、ハードウェアの終了処理、メモリ開放を行います。さらに TXBUFSETUP2、IOGEOSETUP2、TDIO_MISC、TConfigPWM、TSetupPWM、TADIO2、SCP_SETUP_AIALL（センサーモード、ゼロスパン校正位置、ゼロスパン校正係数、スケーリング、アラーム）を SCP_SETUP2 構造体に格納し、これをコンフィグレーションファイル(拡張子 scp)に保存します。信号調節機能がない場合でも本関数を使うことができます。

C/C++ BOOL bADioxScpStore2(CharPayloadC *lpsCharPayload, BOOL bOpenDialog);

C# int bADioxScpStore2 (ref CharPayloadC lpsCharPayload, int bOpenDialog);

引数 bOpenDialog コンフィグレーションファイルの保存先を当関数内蔵の“ファイル保存ダイアログボックス”で指定する場合、TRUE(＝1)とします。FALSE(＝0)とした場合、lpsCharPayload. LpcConfigFileName で直接ファイルを指定します。“ファイル保存ダイアログボックス”の初期フォルダは lpsCharPayload.lpcInitialDir で指定します。

lpsCharPayload コンフィグレーションファイル名、もしくは、“ファイル保存ダイアログボックス”のベースフォルダ名を指定する構造体 CharPayloadC(VB 用は CharPayloadB)へのポインタ。

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

bADioxStore_EX3

単一の MultifunctionI/O に対し、ドライバのクローズ、ハードウェアの終了処理、メモリ開放、割り込みの使用禁止処理を行います。引数または当関数内蔵の“ファイルを開くダイアログボックス”にて指定した、コンフィグレーションファイル（拡張子 adx2 の設定ファイル）に、引数の各構造体の内容を保存させることが可能です。

C/C++
```
BOOL bADioxStore_EX3
        (
        BOOL   bOpenDialog,
        TXBUFSETUP2   *sTBUFSETUP,
        IOGEOSETUP2   *sIOGEOSETUP,
        TDIO_MISC   *sTDIO_MISC,
        TADIO2   *sTADIO,
        TConfigPWM   *sTConfigPWM,
        TSetupPWM   *sTSetupPWM,
        SCP_SETUP_AIALL   *sScpSetupAich,
        ADIOX_EXTENTION2   * lpsEXTENTION,
        BYTE   CARD_ID,
        CharPayloadC   *lpsCharPayload
        );
```

C#
```
int bADioxStore_EX3
        (
        int   bOpenDialog,
        ref   TXBUFSETUP2   sTBUFSETUP,
        ref   IOGEOSETUP2   sIOGEOSETUP,
        ref   TDIO_MISC   sTDIO_MISC,
        ref   TADIO2   sTADIO,
        ref   TConfigPWM   sTConfigPWM,
        ref   TSetupPWM   sTSetupPWM,
        ref   SCP_SETUP_AIALL   lpsScpSetupAich,
        ref   ADIOX_EXTENTION   lpsEXTENTION,
        byte   bCARD_ID,
        ref   CharPayloadC   lpsCharPayload
        );
```

引数	bOpenDialog	コンフィグレーションファイルの保存先を当関数内蔵の“ファイル保存ダイアログボックス”で指定する場合、本引数を TRUE(=1)とします。FALSE(=0)とした場合、ファイル名を lpsCharPayload.lpcConfigFileName で直接指定します。“ファイル保存ダイアログボックス”の初期フォルダは lpsCharPayload.lpcInitialDir で指定します。
	lpsTBUFSETUP	コンフィグレーションファイルに保存したい TXBUFSETUP2 構造体へのポインタ。
	lpsIOGEOSETUP	コンフィグレーションファイルに保存したい IOGEOSETUP2 構造体へのポインタ。
	lpsTDIO_MISC	コンフィグレーションファイルに保存したい TDIO_MISC 構造体へのポインタ。
	lpsTADIO	コンフィグレーションファイルに保存したい TADIO2 構造体へのポインタ。
	lpsTConfigPWM	コンフィグレーションファイルに保存したい TConfigPWM 構造体へのポインタ。
	lpsTSetupPWM	コンフィグレーションファイルに保存したい TSetupPWM 構造体へのポインタ。
	lpsScpSetupAich	コンフィグレーションファイルに保存したい SCP_SETUP_AIALL 構造体へのポインタ。 この引数は信号調節機能付き MultifunctionI/O でのみ意味があります。
	lpsEXTENTION	コンフィグレーションファイルに保存したい ADIOX_EXTENTION2 (VB 用は ADIOX_EXTENTION2B)構造体へのポインタ。
	bCardID	ターゲットデバイスのカード ID。
	lpsCharPayload	コンフィグレーションファイル名、もしくは、“ファイル保存ダイアログボックス”のベースフォルダ名を指定する構造体 CharPayloadC(VB 用は CharPayloadB)へのポインタ。

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

3.割り込み [ADiox2.dll]

bADioxInterruptStart

割り込みメッセージの有効・無効を設定します。リングバッファを使う場合、カウンター割り込みを使う場合、DI 割り込みを使う場合は、有効にしてください。割り込みを使用しない場合、スレッド等で割り込みステータスを監視することになり、負荷が重くなります。

C/C++ BOOL bADioxInterruptStart(BOOL bStart, BYTE bCardID);

C# int bADioxInterruptStart(int bStart, byte bCardID);

引数	bStart	TRUE を指定すると、割り込みメッセージが有効になります。終了時には FALSE を指定して割り込みメッセージをディセーブルにしてください。割り込みメッセージを有効にすると、初期化関数の bHwintr で指定したメッセージが、初期化関数で師弟した hWnd のウィンドウハンドルを持つアプリケーションに送信されます。
	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。

bADioxInterruptStatus

割り込みステータスを取得します。割り込み要因は、リングバッファー割り込み、DI エッジ割り込み、カウンター割り込みの 3 つあり本関数で特定できます。更に DI エッジ割り込みと、カウンター割り込みの場合は、どのチャンネルが割り込み要因なのかなど詳細を特定できます。

C/C++ BOOL bADioxInterruptStatus(struct IRQ_BUFFER *lpsIrqbuf, BYTE bCardID);

C# int ADioxInterruptStatus(ref IRQ_BUFFER lpsIrqbuf, byte bCardID);

VB Declare Function bADioxInterruptStatus Lib "ADiox2.dll" (_
ByRef lpsIrqbuf As IRQ_BUFFER, ByVal bCardID As Byte) As Long

引数	lpsIrqbuf	割り込み要因を格納した構造体 IRQ_BUFFER 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。

bADioxMessageCount

デバイスドライバが、アプリケーションに送った割り込みメッセージの回数を取得します。この値よりも、アプリケーションで受け取ったメッセージが少ない場合、メッセージの取りこぼしが発生しており、①コンピュータの能力に対してサンプリング速度が早すぎる、②DI やカウンタから大量の割り込みが発生して処理しきれない、③アプリケーション割り込みメッセージの処理の内容が重過ぎるなどの課題が発生していることを示します。このメッセージ回数は、bADioxSetupSymmetryEngine2 でリングバッファを開始させるとリセットされます。

C/C++ BOOL bADioxMessageCount (LPDWORD lpdwMessageCount, BYTE bCardID);

C# int bADioxMessageCount (ref uint lpdwMessageCount, byte bCardID);

引数	lpdwMessageCount	割り込みメッセージの発生回数を可能したポインタ。
	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。

bADioxPauseIrqThread

ADXⅡ42FE-250K-Ethernet、ADXⅡ42FE-MPU、ADXⅡ42FE-CORE、ADXⅡ-INF02 専用の関数です。この機種では高速で割り込みをエミュレーションするスレッドがドライバ内部で実行されていますが、本関数は、このスレッドを一時停止させます。設定などでデータ収集から抜ける場合など、本関数を使用してください。再び、割り込みエミュレーションを使うには bADioxRestartIrqThread を実施してください。割り込みエミュレーション用のスレッドが停止したままだと、割り込み機能が使えません。

C/C++ BOOL bADioxPauseIrqThread ( );

C# int bADioxPauseIrqThread ();

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

bADioxRestartIrqThread

ADXⅡ42FE-250K-Ethernet、ADXⅡ42FE-MPU、ADXⅡ42FE-CORE、ADXⅡ-INF02 専用の関数です。この機種では高速で割り込みをエミュレーションするスレッドがドライバ内部で実行されていますが、本関数は、bADioxPauseIrqThread で一時停止させたスレッドを再スタートさせます。

C/C++ BOOL bADioxRestartIrqThread ( );

C# int bADioxRestartIrqThread ();

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

4.リングバッファ　[ADiox2.dll]

bADioxDmaReadEX3

AI/DI のリングバッファ(2 ステージリングバッファではセカンダリリングバッファ)からのデータの読み出しを行います。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。本関数は ADIOX_EXTENTION2 構造体でファイル保存を指定した場合、ファイル保存も実施します。またステータスの取得、終了確認(ストップトリガ)などの周辺処理も一括して行います。引数 lpdAiBuffa に物理定数への変換やスケーリングされた値も格納されます。

C/C++
```
BOOL bADioxDmaReadEX3
        (
        LPDWORD lpdwBuffa,
        double   * lpdAiBuffa,
        TStatusPack2 *lpsTStatus,
        BYTE bCardID
        );
```

C#
```
int bADioxDmaReadEX3
        (
        ref uint          lpdwBuffa,
        ref double        lpdAiBuffa,
        ref TStatusPack2  lpsTStatus,
        byte              bCardID
        );
```

引数

lpdwBuffa	リングバッファのデータのコピー先バッファへのポインタ。ADXⅡ14-125M-PCIEX を除く機種では、データは Bit31-16 が DI15ch-0ch に、Bit15-0 が AI の 16Bit データに割り付けられます。 ADXⅡ14-125M-PCIEX では、Bit31-18 が AI1ch、Bit17-16 が DI3-2ch、Bit15-2 が AI0ch、Bit1-0 が DI1-0ch に割り当てられます。(尚 DI の割付をなくすことで、16bit の A/D 変換データの取得も可能です)
lpdAiBuffa	信号調節機能により、物理定数(電圧や温度などの値)に変換されたアナログ値 ×リングバッファサイズのバッファへのポインタ。詳細は下記内容です。 ADXⅡ42***、ADXⅡINF***ではゼロ校正・スパン校正・スケーリング・リニアライズされた値。 ADXⅡ85-1M-PCIEX では単に電圧値に変換された値。 ADXⅡ14-125M-PCIEX では無意味
lpsTStatus	システムステータスを格納した、TStatusPack 構造体へのポインタ。
bCardID	ターゲットデバイスのカード ID。

戻り値 バッファトリガエンジンの状態を示します。TRUE(=1)で停止(停止トリガか、停止カウンタが機能した)、FALSE(=0)で稼動中です。オーバラン(高負荷時)や停止トリガ検出時に自動的に停止します。

備考 **＜ADXⅡ85-1M-PCIEX 以外の機種の*バッファサイズの算出方法*＞**
SAYA_DEVICE_INFO.dwBufferSizeOfDWORD そのものです。

＜ADXⅡ85-1M-PCIEX の場合の*バッファサイズの算出方法*＞
SAYA_DEVICE_INFO.dwBufferSizeOfDWORD に、セカンダリリングバッファ倍率を乗算したサイズです。セカンダリリングバッファ倍率は、bADioxRingBufferMode や bADioxLoad_EX3 の dwRingBufferMode に相当します。

注意 旧 bADioxDmaReadEX2、旧 bADioxReadScpBuf2 は新しいデザインには推奨されません。

bADioxWriteMemoryEX

AO/DO のリングバッファ(2 ステージリングバッファではセカンダリリングバッファ)へデータの書き込みを行います。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、バッファへデータを書き込みます。本関数は ADIOX_EXTENTION2 構造体でファイル読み出し指定した場合、関数内部で、ファイル読み出しを実施し、読み出した値を AO/DO に出力します。またバッファトリガエンジンを駆動する前の、バッファへの書込み(プリライト)を行う場合も本関数にアクセスしてください。

C/C++ BOOL bADioxWriteMemoryEX(BYTE bAccessMode, LPDWORD lpdwBuffa, BYTE bCardID);

C# int bADioxWriteMemoryEX(byte bAccessMode, ref uint lpdwBuffa, byte bCardID);

引数

bAccessMode	0 で通常のリングバッファ割り込み時の書き込み。14 でプリライト書き込み。
lpdwBuffa	リングバッファ書込みデータへのポインタ。ADX Ⅱ 14-125M-PCIEX を除く機種では、データは Bit31-16 が DO15ch-0ch に、Bit15-0 が AO の 16Bit データに割り付けられます。 ADX Ⅱ 14-125M-PCIEX で DO をバッファ経由とする場合、Bit31-18 が AO1ch、Bit17-16 が DO3-2ch、Bit15-2 が AO0ch、Bit1-0 が DO1-0ch に割り当てられます。 ADX Ⅱ 14-125M-PCIEX で DO をバッファ経由としない場合(ポーリング)、Bit31-16 が AO1ch、Bit15-0 が AO0ch に割り当てられます。
bCardID	ターゲットデバイスのカード ID。

戻り値 ファイル読み出し指定した場合 TRUE(=1)で終了、FALSE(=0)で読み出し中となります。ファイル読み出しを指定しない場合 FALSE(=0)が返りますが特に意味がありません。

備考 **＜ADX Ⅱ 85-1M-PCIEX 以外の バッファサイズの算出方法＞**
SAYA_DEVICE_INFO.dwBufferSizeOfDWORD そのものです。

＜ADX Ⅱ 85-1M-PCIEX の場合の バッファサイズの算出方法＞
SAYA_DEVICE_INFO.dwBufferSizeOfDWORD に、セカンダリリングバッファ倍率を乗算したサイズです。セカンダリリングバッファ倍率は、bADioxRingBufferMode や bADioxLoad_EX3 の dwRingBufferMode に相当します。

＜プリライトについて＞
ADX Ⅱ 14-125M-PCIEX と ADX Ⅱ 42***、ADX Ⅱ INF***ではリングバッファ稼動前に 2 バンクのリングバッファと、転送元のソフトウェアバッファを予め書き込んでおく必要があります。すなわち前述したバッファサイズの 3 倍を予め書き込んでおく必要があります。
ADX Ⅱ 85-1M-PCIEX ではセカンダリリングバッファ 2 バンクと、プライマリリングバッファ 1 バンク分を予め書き込んでおく必要があります。
前述したバッファサイズを dwBufferSize とした場合、

```
dwPreWriteSize   = (sSAYA_DEVICE_INFO.dwDeviceType==ADX14_PCI )  ? dwBufferSize * 3
                 : (sSAYA_DEVICE_INFO.dwDeviceType==ADX42_LAN) ? dwBufferSize * 3
                 :   dwBufferSize * 2 + ADX85_PCI_BUFFER_SIZE;
```

がプリライトサイズとなります。
実際には、このような計算をしなくても、SAYA_DEVICE_INFO_EX. dwPreWriteSizeOfDWORD から同値を取得できます。

bADioxReadWriteEX

ADX Ⅱ 42FE-250K-Ethernet、ADX Ⅱ 42FE-MPU、ADX Ⅱ 42FE-CORE、ADX Ⅱ -INF02 専用の関数で、bADioxDmaReadEX3 と、bADioxWriteMemoryEX を併合した関数です。すなわち、AO/DO のリングバッファ(2 ステージリングバッファではセカンダリリングバッファ)へデータの書き込みを行い、同時に AI/DI のリングバッファ(2 ステージリングバッファではセカンダリリングバッファ)からのデータの読み出しを行います。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、バッファへのデータを書き込みとデータ取得を行ってください。本関数は ADIOX_EXTENTION2 構造体でファイル保存や、ファイル読み出しを指定した場合、これらも関数内部で実施されます。またステータスの取得、終了確認(ストップトリガ)などの周辺処理も一括して行います。**本関数を活用することで、bADioxDmaReadEX3 と、bADioxWriteMemoryEX をそれぞれコールするよりも、約半分の負荷ですみます。**

C/C++

```
BOOL bADioxReadWriteEX
        (
        BYTE            bAccessMode,
        LPDWORD         lpdwBuffa,
        LPDWORD         lpdwReadBuf,
        double          * lpdAiBuffa,
        BYTE            bCardID,
        LPBOOL          lpbStopNow,
        TStatusPack2    *lpsTStatus
        );
```

C#

```
int bADioxReadWriteEX
        (
        byte            bAccessMode,
        ref uint        lpdwBuffa,
        ref uint        lpdwReadBuffa,
        ref double      lpdAiBuffa,
        byte            bCardID,
        ref int         lpbStopNow,
        ref TStatusPack2 lpsTStatus
        );
```

引数	bAccessMode	0 で通常のリングバッファ割り込み時の書き込み。14 でプリライト書き込み。
	lpdwBuffa	リングバッファ書込みデータへのポインタ。データは Bit31-16 が DO15ch-0ch に、Bit15-0 が AO の 16Bit データに割り付けられます。
	lpdwReadBuffa	リングバッファのデータのコピー先バッファへのポインタ。データは Bit31-16 が DI15ch-0ch に、Bit15-0 が AI の 16Bit データに割り付けられます。
	lpdAiBuffa	信号調節機能により、物理定数（電圧や温度などの値）に変換されたアナログ値 ×リングバッファサイズのバッファへのポインタ。このデータ配列には、ゼロ校正・スパン校正・スケーリング・リニアライズされた値が格納されます。
	lpbStopNow	バッファトリガエンジンの状態を示します。TRUE(＝1)で停止（停止トリガか、停止カウンタが機能した）、FALSE(＝0)で稼動中です。オーバラン・アンダーラン（高負荷時）や停止トリガ検出時に自動的に停止します。
	lpsTStatus	システムステータスを格納した、TStatusPack 構造体へのポインタ。
	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。

備考 **＜バッファサイズの算出方法＞**
SAYA_DEVICE_INFO.dwBufferSizeOfDWORD そのものです。

＜プリライトについて＞
リングバッファ稼動前に 2 バンクのリングバッファと、転送元のソフトウェアバッファを予め書き込んでおく必要があります。すなわち前述したバッファサイズの 3 倍を予め書き込んでおく必要があります。

bADioxStopPoint2

リングバッファが停止トリガや停止コマンドなどにより停止した否かを戻り値として返します。戻り値が TRUE なら終了で、この時点における、残留データサイズを第一引数のポインタで返します。bADioxDmaReadEX2 や bADioxReadScpBuf2 にも本機能が内蔵されており、ファイル保存も自動的に残留サイズを保存するようになっています。

C/C++ BOOL bADioxStopPoint2(LPDWORD lpdwStopAddress, TStatusPack2 *lpsTStatus, BYTE bCardID);

C# int bADioxStopPoint2(ref uint lpdwStopAddress, ref TStatusPack2 lpsTStatus byte bCardID);

引数	lpdwStopAddress	リングバッファが停止したの場合 (1 単位＝4byte)を表します。
	lpsTStatus	システムステータスを格納した、TStatusPack 構造体へのポインタ。
	bCardID	ターゲットデバイスのカード ID。

戻り値 バッファトリガエンジンの状態を示します。TRUE(＝1)で停止（停止トリガか、停止カウンタが機能した）、FALSE(＝0)で稼動中です。オーバラン（高負荷時）や停止トリガ検出時に自動的に停止します。

bADioxCyclicTrigger

本関数はサイクリックトリガ機能が搭載機種のみ有効です。（ADX Ⅱ 85-1M-PCIEX 以外）本関数では、サイクリックトリガ機能の設定を行います。サイクリックトリガは、オシロスコープをエミュレートするもので、トリガがかかると、本関数の第 2 引数“dwSetupBufferSize”で指定されたサイズを取り込んだ段階でバンクチェンジを行って 1 次停止、ユーザアプリケーションにデータの取得を促すために、割り込みを発生させます。1 次停止したトリガ及びリングバッファは、関数“bADioxCyclicRestart”で再開します。“bADioxCyclicRestart”を割り込み処理の最後に追加することで、あたかもオシロスコープのような動作が可能です。連続データ収集ではないので、より速度のデータ収集を実現できます。（注：本関数を使用する場合にはリングバッファ出力を併用しないでください）

C/C++ BOOL bADioxCyclicTrigger (BOOL bCyclicTrigger,DWORD dwSetupBufferSize,BYTE bCardID);

C# int bADioxCyclicTrigger (int bCyclicTrigger , uint dwSetupBufferSize , byte bCardID);

引数	bCyclicTrigge	サイクリックトリガを on にする場合 TRUE、off にする場合 false
	dwSetupBufferSize	サイクリックトリガで有効とするバッファサイズ。64～1048576 の値が有効です。 ADX Ⅱ 14-125M-PCIEX 以外ではこの引数では無意味で常に 1024 相当になります。
	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。

bADioxCyclicRestart

本関数はサイクリックトリガ機能が搭載機種のみ有効です。（ADXⅡ85-1M-PCIEX 以外）サイクリックトリガは、オシロスコープをエミュレートするもので、トリガがかかると、関数“bADioxCyclicTrigger”の第 2 引数“dwSetupBufferSize”で指定されたサイズを取り込んだ段階でバンクチェンジを行って 1 次停止、ユーザアプリケーションにデータの取得を促すために、割り込みを発生させます。本関数は 1 次停止したトリガ及びリングバッファを再開させるものです。超高速で稼動させる場合、本関数を割り込み処理の最後に入れるとPC の負荷が高くなりすぎて、ユーザアプリケーションのマウスイベントの取得などを阻害します。従って、多少のウェイトを入れてください。（例えば Console2.exe では Sleep(50)を入れている）

C/C++ BOOL bADioxCyclicRestart (BYTE bCardID;

C# int bADioxCyclicRestart (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。

5. 設定 [ADiox2.dll]

bADioxSetupSymmetryEngine2

高速連続データ収集システムの心臓部でもある、リングバッファ(2 ステージリングバッファ)およびトリガコントローラ、ファイル処理、サンプリング速度の設定を行います。

C/C++ BOOL bADioxSetupSymmetryEngine2 (struct TXBUFSETUP2 *sTBufSetup,
ADIOX_EXTENTION2 *lpsADIOX_EXTENTION2 ,BYTE bCardID);

C# int bADioxSetupSymmetryEngine2 (ref TXBUFSETUP sTBufSetup,
ref ADIOX_EXTENTION2 lpsADIOX_EXTENTION2 , byte bCardID);

引数	sTBufSetup	リングバッファ(2 ステージリングバッファ)およびトリガコントローラ、サンプリング速度の設定値が入った TXBUFSETUP2 構造体を指定します。
	lpsADIOX_EXTENTION2	ファイル保存、ファイル読み出し情報の入った ADIOX_EXTENTION2 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。

bADioxAnalogConfigration2

アナログデジタル入出力インターフェース部の機能設定(入出力レンジ、チャンネル、取り込み方法、チャタリング除去など)を行います。

C/C++ BOOL bADioxAnalogConfigration2 (IOGEOSETUP2 *sIoGeoSetup, BYTE bCardID);

C# int bADioxSetupSymmetryEngine2 (ewf IOGEOSETUP2 sIoGeoSetup, byte bCardID);

引数	sIoGeoSetup	アナログデジタル入出力インターフェース設定値が入った IOGEOSETUP2 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。

bADioxDioMisc2

エンコーダカウンター、周波数カウンター、PWM(一部)、DI 割り込み、ストローブラッチの設定を行います。

C/C++ BOOL bADioxDioMisc2 (struct TDIO_MISC *sDIO_MISC, BYTE bCardID);

C# int bADioxDioMisc2 (ref TDIO_MISC sDIO_MISC, byte bCardID);

引数	sDIO_MISC	エンコーダカウンター、周波数カウンター、PWM(一部)、DI 割り込み、ストローブラッチの各種設定値が入った TDIO_MISC 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。

bADioxConfigPWM2

PWM チャンネル毎に、位相と発振サイクル数、PWM 開始/停止の設定を行います。

C/C++ BOOL bADioxConfigPWM2 (struct TConfigPWM *sTConfigPWM, BYTE bCardID);

C# int bADioxConfigPWM2 (ref TConfigPWM sTConfigPWM, byte bCardID);

引数	sTConfigPWM	位相と発振サイクル数、開始フラグを格納した TConfigPWM 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。

bADioxSetupPWM2

PWM チャンネル毎に、デューティー比を設定します。稼動状態での連続変更が可能です。

C/C++ BOOL bADioxSetupPWM2 (struct TSetupPWM *sTSetupPWM, BYTE bCardID);

C# int bADioxSetupPWM2 (ref TSetupPWM sTSetupPWM, byte bCardID);

引数		
	sTSetupPWM	PWM のデューティ比を設定します。 PWM でアナログ相当の制御を行う場合のインターフェースはここになります。
	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。

bADioxRingBufferMode

ADXⅡ85-1M-PCIEX 専用の関数です。2 ステージリングバッファにおける、セカンダリリングバッファサイズを変更します。本機能を使うことで、レスポンスと負荷のバランスを調整します。レスポンスを重視であればバッファサイズを小さく、速度重視（低負荷）にするには（バッファオーバラン・アンダーランを防止するには）バッファサイズを大きくします。（2 ステージリングバッファの詳細は、ハードウェア仕様書を参照願います） bADioxStore_EX3 には本関数に相当する機能が組み込まれているので本関数を呼び出す必要はありません。**本関数（もしくは** bADioxStore_EX3）**はリングバッファの稼動までに必ず実行されなければなりません。**

C/C++ BOOL bADioxRingBufferMode (DWORD dwRingBufferMode, BYTE bCardID);

C# int bADioxRingBufferMode (uint dwRingBufferMode, byte bCardID);

引数		
	bCardID	ターゲットデバイスのカード ID
	dwRingBufferMode	セカンダリ PC リングバッファサイズをプライマリオンチップリングバッファの何倍にするかを設定します。値は 1～512 の範囲としてください。WindowsVista 以降の OS では場合最高サンプリング速度で 200 程度にしないとバッファオーバラン/アンダーランが回避できない場合があります。

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

bADioxScpSetup2

ADXⅡ42***、ADXⅡINF***の信号調節設定（SCP_SETUP_AICH 構造体の内容）をハードウェアに反映します。この関数をコールすると、ゼロ、スパン校正が解除されるので、ご注意ください。もしゼロ・スポン校正を解除したくない場合には、ADioxScpSetupEX2 を使用してください。

C/C++ BOOL bADioxScpSetup2(struct SCP_SETUP_AICH sScpSetupAich,BYTE bCardID);

C# int bADioxScpSetup2(SCP_SETUP_AICH sScpSetupAich, byte bCardID);

引数		
	sScpSetupAich	シグナルコンディション設定を格納した SCP_SETUP_AICH 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。

bADioxScpSetupEX2

ADXⅡ42***、ADXⅡINF***の信号調節設定（SCP_SETUP_AIALL 構造体の内容）をハードウェアに反映します。この関数をコールしても、ゼロ、スパン校正係数は解除されません。よってセンサ種別を変更した場合には、値が正確ではなくなる可能性がありますのでご注意ください。ゼロ・スポン校正を解除してデフォルト値をロードする場合には、ADioxScpSetup2 を使用してください。

C/C++ BOOL bADioxScpSetupEX2(struct SCP_SETUP_AIALL sScpSetupAich,BYTE bCardID);

C# int bADioxScpSetupEX2(SCP_SETUP_AIALL sScpSetupAich,byte bCardID);

引数		
	sScpSetupAich	シグナルコンディション設定を格納した SCP_SETUP_AICH 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。

bADioxScpDefault2

ADXⅡ42***、ADXⅡINF***専用の関数です。引数で指定したカード ID、チャンネル番号、センサー種別に対する、適切な信号調節設定のデフォルト値を生成します。このデフォルト値は、DLL 内部の SCP_SETUP2 構造体に設定されるとともに、第 4 引数以降から取得することができます。本関数は値の取得だけで、ハードウェアへの反映は行われないので注意してください。

C/C++ BOOL bADioxScpDefault2(DWORD dwSensorMode,DWORD dwDeviceNumber,
DWORD dwCH,struct SCP_SETUP * lpsSCP_SETUP);

C# int bADioxScpDefaultCS2(uint dwSensorMode, uint dwDeviceNumber,
uint dwCH, ref SCP_SETUP lpsSCP_SETUP);

引数		
	dwSensorMode	センサー種別
	dwDeviceNumber	ターゲットデバイスのカード ID（bCardID と同じです）。
	dwCH	アナログ入力チャンネル番号
	lpsSCP_SETUP	ドライバ内部 SCP_SETUP2 構造体のポインタ。 VB では SCP_SETUP2 構造体を 3 分割したポインタとなります。（※）

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

bADioxEEPROM_WRITE

ADXⅡ42FE-250K-Ethernet、ADXⅡ42FE-MPU、ADXⅡ42FE-CORE、ADXⅡ-INF02 専用の関数です。ネットワークアドレス構造体 sTCP_IP_SETTING で示される、ハードウェアに対し、ネットワークアドレス構造体 sNS の内容を書き込みます。

C/C++ BOOL bADioxEEPROM_WRITE (TCP_IP_SETTING sTCP_IP_SETTING , NETWORK_SETUP sNS);

C# int bADioxEEPROM_WRITE (TCP_IP_SETTING sTCP_IP_SETTING , NETWORK_SETUP sNS);

引数		
	sTCP_IP_SETTING	現在のハードウェアの IP アドレス、ポート番号設定を格納した TCP_IP_SETTING への構造体。
	sNS	更新したい IP アドレス、ゲートウェイ、サブネットマスク、ポート番号の設定を格納した NETWORK_SETUP 構造体。

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

bADioxClockMode

ADXⅡ85-1M-PCIEX 専用関数です。DO30 にサンプリングクロック（A/D スタートパルス）を割り当てるか否かを決定します。

C/C++ BOOL bADioxClockMode (BOOL bClockOutEnable,BYTE bCardID);

C# int bADioxClockMode(int bClockOutEnable,byte bCardID);

引数		
	bClockOutEnable	クロック出力を行う場合 TRUE、通常オペレーション FALSE。
	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。

bADioxDefaultInit

ADIOX2-API は膨大な構造体の設定が必要で、大変な作業です。本関数はデフォルト値を生成、ハードウェアに反映し、この構造体のポインタを引数とします。アプリケーションは、引数より取得した構造体より必要なメンバ変数のみ変更すればよく、コーディング作業が効率的になります。
初期化は以下の通りです、ここに示されていない全てのメンバ変数はゼロになります。

C/C++ BOOL bADioxDefaultInit (TXBUFSETUP2 * lpsTBUFSETUP , IOGEOSETUP2 * lpsIOGEOSETUP,
TDIO_MISC * lpsTDIO_MISC , TConfigPWM * lpsTConfigPWM ,
TSetupPWM * lpsTSetupPWM , TADIO2 * lpsTADIO , BYTE bCardID);

C# bool bADioxDefaultInit (ref TXBUFSETUP2 lpsTBUFSETUP , ref IOGEOSETUP2 lpsIOGEOSETUP,
ref TDIO_MISC lpsTDIO_MISC , ref TConfigPWM lpsTConfigPWM,
ref TSetupPWM lpsTSetupPWM , ref TADIO2 lpsTADIO , byte bCARD_ID);

引数		
	*lpsTBUFSETUP	デフォルト値を格納した TBUFSETUP2 構造体へのポインタ。
	*lpsIOGEOSETUP	デフォルト値を格納した IOGEOSETUP2 構造体へのポインタ。
	*lpsTDIO_MISC	デフォルト値を格納した TDIO_MISC 構造体へのポインタ。
	*lpsTConfigPWM	デフォルト値を格納した TConfigPWM 構造体へのポインタ。
	*lpsTSetupPWM	デフォルト値を格納した TSetupPWM 構造体へのポインタ。
	*lpsTADIO	デフォルト値を格納した TADIO2 構造体へのポインタ。
	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. ステータス [ADiox2.dll]

bADioxStatus2

システムの稼動状態を取得します。

C/C++ BOOL bADioxStatus2 (struct TStatusPack2 * sTStatus, BYTE bCardID);

C# int bADioxStatus2 (ref TStatusPack2 sTStatus, byte bCardID);

引数

* sTStatus	システムステータスを格納した、TStatusPack 構造体へのポインタ。
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。

vADioxDeviceInfoEX

デバイスの各種情報を取得します。この関数により、ボード毎に個別のソフトウェアを構築せず、共通化することができます。

C/C++ void vADioxDeviceInfoEx (struct SAYA_DEVICE_INFO_EX * lpsSAYA_DEVICE_INFO,BYTE bCardID);

C# void vADioxDeviceInfoEx (ref SAYA_DEVICE_INFO_EX lpsSAYA_DEVICE_INFO, byte bCardID);

引数

* lpsSAYA_DEVICE_INFO	デバイス情報構造体 SAYA_DEVICE_INFO へのポインタ。
bCardID	ターゲットデバイスのカード ID。

注意：旧 vADioxDeviceInfo も使用可能ですが、新規のデザインには推奨されません。新しい vADioxDeviceInfoEx の使用により機種依存コードを大幅に削減することが可能です。

vADioxScpCopy2

ドライバ内部の SCP_SETUP2 構造体の内容を取得します。bADioxScpLoad2 が成功した場合、ドライバ内部の SCP_SETUP2 構造体にコンフィグレーションファイルの情報が展開されます。失敗した場合にはドライバ内部の SCP_SETUP 構造体にフォルトのコンフィグレーション情報が展開されます。本関数はこれら内部状態を取得するものです。信号調節機能がない場合でも本関数を使うことができます。更に、ADiox2-API では、構造体 TXBUFSETUP2、IOGEOSETUP2、TDIO_MISC、TConfigPWM、TSetupPWM、TADIO2、SCP_SETUP_AICH、SCP_SETUP_AIALL を引数とする関数にアクセスすると、ドライバ内部の SCP_SETUP2 構造体が更新されますが、本関数はこうした運用中の内部状態も取得できます。SCP_SETUP 構造体の内容は bADioxScpStore2 で保存することができます。

C/C++ void vADioxScpCopy2 (SCP_SETUP2 * lpsSCP_SETUP);

C# void vADioxScpCopy_CS2 (ref SCP_SETUP_CS2 lpsSCP_SETUP);

引数

lpsSCP_SETUP	ドライバ内部 SCP_SETUP2 構造体のポインタ。 VB では SCP_SETUP2 構造体を 3 分割したポインタとなります。

dwADioxNetError

ADXⅡ42***, ADXⅡINF***の通信エラーを報告します。

C/C++ DWORD dwADioxNetError(LPDWORD lpdwWriteRetry,LPDWORD lpdwReadRetry,
LPDWORD lpdwReadFlameError,LPDWORD lpdwReadAddressError);

C# uint dwADioxNetError(ref uint lpdwWriteRetry, ref uint lpdwReadRetry,
ref uint lpdwReadFlameError, ref uint lpdwReadAddressError);

引数

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。

bADioxGetIrqThreadErr (new)

内部のスレッドによる I/O アクセスで ADXⅡ42***, ADXⅡINF***が通信途絶状態になったことを示します。

C/C++ BOOL bADioxGetIrqThreadErr (BYTE bCardID);

引数

bCardID	ターゲットデバイスのカード ID。

戻り値 問題なければ 1(TRUE)、通信途絶の場合 0(FALSE)が返ります。

dwADioxSrcStatus

ドライバ、ハードウェアで発生したエラー回数を返します。エラー回数には回復に成功したものも含みます。

C/C++ DWORD dwADioxSrcStatus (DWORD dwCommand);

C# uint dwADioxSrcStatus (uint dwCommand);

引数 dwCommand 0 を指定してください。

戻り値 エラー発生回数が返ります。

7. ポーリング [ADiox2.dll]

bADioxADIO2

アナログデジタル入出力・エンコーダカウンタ・周波数カウンタ・温度の一斉ポーリングを行います。アナログ入出力を電圧値に変換した値を取り出すこともできます。アナログデジタル出力をリングバッファ経由とした場合、AO チャンネル、DO チャンネルはリングバッファデータが優先されます。

C/C++ BOOL bADioxADIO2 (struct TADIO2 * lpsTADio, BYTE bCardID);

C# int bADioxADIO2 (ref TADIO2 lpsTADio ,byte bCardID);

引数

*lpsTADio	アナログ入出力・デジタル入出力・エンコーダカウンタ・周波数カウンタ・温度の値を格納した TADIO2 構造体へのポインタ。アナログ入出力を電圧値に変換した値も格納されます。メンバ変数により入力と出力に分かれます。
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。

bADioxChannelAIw2_fx

アナログ入力値を取得し、次に取得するチャンネルを指定します。取得したアナログ入力データは前回本関数を呼び出したときに指定したチャンネルとなります。(①ch 指定→②A/D 値取得ではなく、①A/D 値取得→②ch 指定)
サンプリング周期とは非同期にデータを読み出せるので、読み出し周期よりも、サンプリング速度が遅いと、同じ値を何度も読みつづけることになります。そのため、ある程度早いサンプリング速度に設定することを推奨します。

C/C++ BOOL bADioxChannelAIw2_fx (struct IOGEOSETUP2 *lpsIoGeoSetup,
LPDWORD lpdwData, int iInterval, BYTE bCardID);

C# int bADioxChannelAIw2_fx (ref IOGEOSETUP2 lpsIoGeoSetup,
ref uint lpdwData, int iInterval, byte bCardID);

引数

lpsIoGeoSetup	アナログデジタル入出力インターフェース設定値が入った IOGEOSETUP2 構造体を指定します。“次回”取得したいアナログチャンネルをここで指定します。
lpdwData	取得されたアナログ入力値を格納したポインタ。下位 16Bit のみ有効で上位は 0 です。
iInterval	使用されません。
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。

bADioxDI

デジタル入力値を取得します。

C/C++ BOOL bADioxDI(LPDWORD lpdwDI,BYTE bCardID);

C# int bADioxDI (ref uint lpdwDI, byte bCardID);

引数

lpdwDI	デジタル入力値を保持したポインタ。
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。

bADioxDO

デジタル出力値を設定します。デジタル出力をリングバッファ経由とした場合、DO チャンネルはリングバッファデータが優先されます。

C/C++ BOOL bADioxDO (DWORD dwDO, BYTE bCardID);

C# int bADioxDO (uint dwDO, byte bCardID);

引数		
	dwDO	デジタル出力値。
	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。

dADioxLinCoef

bADioxChannelAIw2 や bADioxChannelAIw2_fx、bADioxFastDaq で取得した DWORD 型のアナログ入力値・カウンタ値を本関数に引き渡すと、ゼロ校正、スパン校正、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。設定内容はドライバ内部の SCP_SETUP 構造体から、bCardID,bChannel に相当する部分を取り出して適用します。

C/C++ double dADioxLinCoef(DWORD dwANALOG,LPBOOL lpbBurnOut,
LPDWORD lpdwAlarm,BYTE bCardID,BYTE bChannel);

C# double dADioxLinCoef(uint dwANALOG, ref int lpbBurnOut,
ref uint lpdwAlarm, byte bCardID, byte bChannel);

引数		
	dwANALOG	bADioxADIO2 や、bADioxChannelAIw2 で取得した DWORD 型のアナログ入力値をここにセットしてください。
	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。
	bChannel	アナログ入力チャンネル番号

戻り値 変換されたアナログ値

bADioxPeakHold (new)

ADXⅡ42***、ADXⅡINF***でポーリング期間中のピークホールドを行うか否かを設定します。

C/C++ BOOL bADioxSetPeakHold (DWORD dwPeakholdCh,BYTE bCardID)

引数		
	dwPeakholdCh	この引数はピークホールドを行うチャンネルをバイナリで指定します。各ビットフィールドがアナログチャンネルに相当し該当ビット 1 でピークホールド ON、0 でピークホールド OFF です。例えば 0-3ch をピークホールドしたい場合、0xF を指定します。リングバッファ使用時は必ず OFF してください。
	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。

bADioxFastPolling

ADXⅡ42***、ADXⅡINF***で高速ポーリングの設定を行います。高速ポーリング使用中は、通常のポーリングでは正しい値を取得できない点に注意ください。また高速ポーリング中はオートゼロ関数(**bADioxAutoZero**)を使用しないでください。

C/C++ BOOL bADioxFastPolling (DWORD dwClock,BYTE bCardID)

引数		
	dwClock	この引数は高速ポーリングのオンオフと、サンプリング時間の双方を設定します。 高速ポーリングでは自動的にチャンネルを切り替えながらサンプリングを行いますが、この引数は そのサンプリング時間を指定します。内容は構造体 TXBUFSETUP2 のメンバ変数 dwClockScall と同様の意味を持ちます。また値 0 をセットすると高速ポーリングがオフになります。
	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。

bADioxFastDaq

ADXⅡ42***、ADXⅡINF***で高速ポーリングの値の取得を行います。通常のポーリングはチャンネル指定を行うコマンドを送信し、その後データを受信しますが、高速ポーリングではハードウェアで全チャンネルのデータを順次収集してチャンネル毎のレジスタを設けているので、チャンネル指定コマンド送信や、チャンネル切替～データ安定までのディレイがなく低負荷＝高速です。またアナログなら 1 回の読み出しで 2ch を取得できます。取得した値のスケーリングなどは dADioxLinCoef で行います。

C/C++ BOOL bADioxFastDaq (BYTE ch, LPDWORD lpdwAdcLow, LPDWORD lpdwAdcHigh,
double *lpdAdData, BYTE bCardID);

引数	ch	取得する入力チャンネルを指定します。0-7 がアナログ 8-11 がカウンタ/インフラサウンドデータです。 0 又 1 を指定すると lpdwAdcLow に AI0、lpdwAdcHigh に AI1 を格納します。 2 又 3 を指定すると lpdwAdcLow に AI2、lpdwAdcHigh に AI3 を格納します。 4 又 5 を指定すると lpdwAdcLow に AI4、lpdwAdcHigh に AI5 を格納します。 6 又 7 を指定すると lpdwAdcLow に AI6、lpdwAdcHigh に AI7 を格納します。 8～11 を指定すると lpdwAdcLow にカウンタ（エンコーダ、パルス、周波数、風速）または インフラサウンドデータが入ります。
	lpdwAdcLow	データ格納用のポインタ変数です（32bit 符号なし）
	lpdwAdcHigh	データ格納用のポインタ変数です（32bit 符号なし）
	lpdAdData	データ格納用のポインタ変数です（double 型） この変数は風速にのみ使用されます。風速は m/sec で 3 秒平均の値（瞬間風速）です。
	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。

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 ( );

bADioxWindSpeed

風速を取得します。風速は m/sec で 3 秒平均の値（瞬間風速）です。**（対応機種のみ）**

C/C++ BOOL bADioxWindSpeed (double * lpdCounter,BYTE bCardID)

	lpdCounter	風速を格納する double 型のポインタ
	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。

bADioxBattsens

バッテリーレベルを取得します。**（対応機種のみ）**

C/C++ BOOL bADioxBattsens (int * lpiBatteryLevel, BYTE bCardID)

引数	lpiBatteryLevel	バッテリ残量を格納する int 型のポインタ。値は百分率で%表記です。 バッテリ機能がない場合には 500 を格納します。
	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。

8.校正 [ADiox2.dll]

bADioxAutoCal2

ADXⅡ85-1M-PCIEX のダイナミック校正（アナログ入力の校正）を開始させます。本関数により校正スレッドがドライバ内部で起動し、本関数は処理の終了を待たずに直ちにリターンします。校正スレッドの終了までは dwADioxAutoCal_Status1 を除くADiox2API 関数群は一切呼び出してはいけません。校正の終了は dwADioxAutoCal_Status1 で知ることができます。

C/C++ BOOL bADioxAutoCal2(struct IOGEOSETUP2 *lpsIOGEOSETUP,
struct TXBUFSETUP2 *lpsTBUFSETUP,BYTE bCardID);

C# int bADioxAutoCal2(ref IOGEOSETUP2 lpsIOGEOSETUP,
ref TXBUFSETUP lpsTBUFSETUP, byte bCardID);

引数

sTBufSetup	TXBUFSETUP2 構造体へのポインタを指定します。
sIoGeoSetup	IOGEOSETUP2 構造体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。

dwADioxAutoCal_Status1

ADXⅡ85-1M-PCIEX 専用の関数です。bADioxAutoCal2 関数で起動した校正スレッドの終了を検出する関数です。

C/C++ DWORD dwADioxAutoCal_Status1 (LPBOOL lpbInAutoCal);

C# uint dwADioxAutoCal_Status1 (ref int lpbInAutoCal);

引数

lpbInAutoCal	本引数が TRUE(＝1)の場合、校正実施中です。

戻り値 内部の処理状況を表す数値が返ります。処理が進むと数値は増大しつづけます。

bADioxZeroAdj

ADXⅡ42***、ADXⅡINF***専用の関数です。指定したカード ID、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 または bADioxScpSetupEX 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）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、ゼロ校正とスポン校正を繰り返してください。

C/C++ BOOL bADioxZeroAdj (BYTE bCardID, BYTE bChannel);

C# int bADioxZeroAdj (byte bCardID, byte bChannel);

引数

bCardID	ターゲットデバイスのカード 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

ADXⅡ42***、ADXⅡINF***専用の関数です。指定したカード 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 または bADioxScpSetupEX 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。

C/C++ BOOL bADioxSpanAdj (BYTE bCardID,BYTE bChannel);

C# int bADioxSpanAdj (byte bCardID, byte bChannel);

引数

bCardID	ターゲットデバイスのカード 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

ADXⅡ42***、ADXⅡINF***専用の関数で、指定したカードID、チャンネル番号におけるゼロ校正量を0に、スパン校正量を1にします。すなわちこれは、校正をかけない状態と同じです。

C/C++ void vADioxFreeAdj (BYTE bCardID,BYTE bChannel);

C# void vADioxFreeAdj (byte bCardID, byte bChannel);

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

bADioxAutoZero

ADXⅡ42***、ADXⅡINF***専用の関数です。bADioxZeroAdj や bADioxSpanAdj はユーザにより指定された、ゼロ校正位置、およびスパン校正位置への校正を行うもので、データ収集前の初期校正を前提にしています。これらの関数は高精度ですが負荷が高いので頻繁に呼び出すのは向いていません。これに対し、本関数（自動ゼロ校正）は、基板上のアンプのオフセットをゼロ校正します。校正位置の概念はないので低負荷で、運用直前や運用中の一時校正への使用を前提にしています。長時間の運用で、計測精度が低下することを防ぐことが出来ます。本関数は頻繁には使わないでください。本関数使用後にはオフセットが校正されるのでゼロ点に段差ができる可能性があることも考慮ください。また高速ポーリング、リングバッファ使用時には使用しないでください。高速ポーリング、リングバッファ使用時に使用したい場合には、高速ポーリング、リングバッファの使用を一度中断してから本関数を呼び出してください。

C/C++ BOOL bADioxAutoZero(BYTE bCardID);

C# int 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。

9.FFT [ADiox2.dll]

bADioxFft_Config

FFT の点数、窓関数の設定を行います。FFT を実行する前に設定してください。

C/C++ BOOL bADioxFft_Config(int iPoint, int iWindow);

C# Int bADioxFft_Config(int iPoint, int iWindow);

引数

iPoint	2 の iPoint 乗が FFT 点数になります。例えば 256 ポイントの場合 8、512 ポイントの場合には 9 を設定します。必ず 2～12 の値が有効です。すなわち 4～4096 点の FFT が可能です。
iWindow	以下の数値を指定して窓関数を設定します。 0：ハニング窓　1：ハミング窓　2:ブラックマン窓 3:ガウス窓　4:方形窓

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

dADioxFft_PreScall

ADI リングバッファの DWORD 型アナログ信号データを double 型に変換します。bADioxFft_Analyze の入力が double 型で±32768.0 の入力レンジなので本関数を通す必要があります。

C/C++ double dADioxFft_PreScall (WORD wGetSample,BOOL bCOB);

C# double dADioxFft_PreScall(ushort wGetSample, int bCOB);

引数

dwGetSample	元データ（リングバッファデータの DWORD 型の下位 16Bit）を設定してください。
bCOB	0 の場合ストレートバイナリ、1 の場合コンプリメントバイナリ（2 の補数） ADiox2-API のデフォルト運用状態は 0 です。

戻り値 変換された結果（アナログ値）が返ります。

bADioxFft_Analyze

*dDataIn データ配列を dFreqOut 周波数配列に高速フーリエ変換します。

C/C++ BOOL bADioxFft_Analyze(double * dFreqOut,double * dDataIn);

C# int bADioxFft_Analyze(ref double dFreqOut, ref double dDataIn);

引数

*dFreqOut	FFT 点数に相当するスペクトルデータが格納されます。結果は 16Bit フルスケール値を 0dB とする対数形式で、符号は取り除かれますので 0(最大)～98.08(最小)が格納されます。FFT 点数の 1/2 までの値を使ってください。残りは鏡像反転された周波数スペクトルが格納されます。
*dDataIn	計測データ配列。FFT ポイント数に相当するデータ数（配列）が必要です。

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

10.波形生成 [ADiox2.dll]

任意の振幅・オフセット・周波数で sin,soc,exp(2π),sqrt(2π),dc,step,file_load 波形を連続生成します。

vADioxWaveInit

波形ジェネレータを初期化します。

C/C++ **void** bADioxWaveInit (struct ADIOX_EXTENTION2 *lpsADIOX_EXTENTION2,
double *dFrequency0,double *dFrequency1,BYTE bCardID);

C# **void** bADioxWaveInit (ref ADIOX_EXTENTION2 lpsADIOX_EXTENTION2 ,
ref double dFrequency0 , ref double dFrequency1 , byte bCardID);

引数		
	lpsADIOX_EXTENTION2	ファイル保存、ファイル読み出し情報の入った ADIOX_EXTENTION2 構造体を指定します。
	dFrequency0	AO0 の周波数
	dFrequency1	AO1 の周波数（ADXⅡ14-125M-PCIEX のみ有効）
	bCardID	ターゲットデバイスのカード ID。

bADioxWaveGenerate

波形を生成します。sin,cos,exp,sqrt,triangle,Lamp,Square,dc,step,file などの様々な波形を任意の周波数、振幅、オフセットで生成します。リングバッファを前提にしているので、ADXⅡ14-125M-PCIEX 以外では 1CH、ADXⅡ14-125M-PCIEX は 2CH の波形を生成します。DI は 1Bit シフト波形(DI0 が 1 になり、DI1⇒DI2 と 1 である DIch が移動する)となります。

C/C++ BOOL bADioxWaveGenerate (LPDWORD dwWriteBuffa,DWORD dwBufferSize,BYTE bCardID,
struct ADIOX_EXTENTION2 *lpsADIOX_EXTENTION2);

C# int bADioxWaveGenerate (ref uint dwWriteBuffa , uint dwBufferSize , byte bCardID,
ref ADIOX_EXTENTION2 lpsADIOX_EXTENTION2);

引数		
	lpdwBuffa	波形データを格納しバッファへのポインタ。このデータをリングバッファ出力の変数に使用します。 （この関数は波形データを作成するだけで、リングバッファへのデータ転送は、リングバッファ関連の関数を仕様してください。）
	dwBufferSize	前記バッファサイズ(DWORD 単位)
	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。

11. マルチリモート I/O データロガー[ADioxLogm.dll]

本関数群は複数の ADXⅡ42***、ADXⅡ-INF 専用関数です。(複数台対応)これらの関数郡では計測デー(アナログ 0-7ch、カウンタ 0-3ch、DI16ch)を CSV ファイルに保存します。保存形式は MultiLogger3 と同様です。

bADioxLoggerInit

データロガー(CSV ファイル書き出し機能)の初期化を行います。

C/C++ BOOL bADioxLoggerInit (struct ADIOX_CSV_FORMAT_EX sADIOX_CSV_FORMAT_EX);

引数 sADIOX_CSV_FORMAT_EX 設定構造体 ADIOX_CSV_FORMAT_EX 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。

vADioxAddProjet

ロガーファイル先頭のプロジェクト名等を指定します。

C/C++ void vADioxAddProjyect (char *lpcProjyect,char *lpcOwner,char *lpcComment);

引数

lpcProjyect	プロジェクト名
lpcOwner	オーナー名(測定者)
lpcComment	コメント

bADioxLoggerWrite

データロガー(CSV ファイル書き出し機能)の実行関数です。引数の計測データ 1 サンプル分を、CSV ファイル 1 行に変換します。

C/C++ BOOL bADioxLoggerWrite (double *dAI_Buffa, WORD *wDI_Buffa,
BOOL *bBO_Buffa, BYTE bCardId,BOOL bRefreshFile);

引数

dAI_Buffa	アナログ値を格納したバッファへのポインタ。この変数には ADIOX_CSV_FORMAT_EX.bAI_ChannelScall で指定した記録チャンネル数分のデータを若いチャンネル順に並べてください。(1～12ch 分)
wDI_Buffa	デジタル値を格納したバッファへのポインタ。
bBO_Buffa	バーンアウト値を格納したバッファへのポインタ。バーンアウトしたアナログ入力チャンネルの表示は--になります。この変数には ADIOX_CSV_FORMAT_EX.bAI_ChannelScall で指定した記録チャンネル数分のデータを若いチャンネル順に並べてください。(1～12ch 分)
bCardID	保存するデータののカード ID。
bRefreshFile	ファイル更新する場合 TRUE(これまでのファイルを閉じ、新しいファイルを作成します)

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

12.構造体 1 [ADiox2.dll]

IRQ_BUFFER

割り込みステータスを格納します。割り込み要因はリングバッファ、カウンタコンペア、DI エッジ割り込みなど複数あります。本構造体は割り込み要因を特定するために使用されます。

C/C++

```
struct IRQ_BUFFER
        {
        DWORD dwADO_underrun;
        DWORD dwADI_overrun;
        DWORD dwADO_wr_addr_over;
        DWORD dwADI_rd_addr_over;
        DWORD dwPIO_mode_not_support;
        DWORD dwTimming_error;
        DWORD dwRequestPostmessage;
        DWORD dwDI_CT_interrupt;
        DWORD dwIrqst;
        DWORD dwTheCallCount;
        };
```

C#

```
public struct IRQ_BUFFER
        {
        public uint dwADO_underrun;
        public uint dwADI_overrun;
        public uint dwADO_wr_addr_over;
        public uint dwADI_rd_addr_over;
        public uint dwPIO_mode_not_support;
        public uint dwTimming_error;
        public uint dwRequestPostmessage;
        public uint dwDI_CT_interrupt;
        public uint dwIrqst;
        public uint dwTheCallCount;
        };
```

メンバ変数

dwADO_underrun
AO/DO バッファーアンダーフローステータス。アンダーフローが発生した場合、バッファへの書き込みが無かったため、送るべきデータが無くなったことを意味します。この場合前の値が保持されています。以下の定義された数値が格納されます。

UNDERRUN_BUFFER	:バッファーアンダーフローの発生（エラー）
上記以外	:問題なし

dwADI_overrun
AI/DI バッファーオーバーフローステータス。オーバーフローが発生した場合、バッファからの読み出しが無かったため、バッファにデータを収容しきれなくなったことを意味します。（データが失われた）以下の定義された数値が格納されます。

OVERRUN_BUFFER	バッファーオーバーフローの発生（エラー）
上記以外	:問題なし

dwADO_wr_addr_over　使われていません。
dwADI_rd_addr_over　使われていません。
dwPIO_mode_not_support　使われていません。
dwTimming_error　使われていません。
dwIrqst　予約
dwTheCallCount　カーネルモードデバイスドライバで受信したハードウェア割り込みの回数
dwRequestPostmessage　割り込みの要因が格納されます。以下の 4 つの要因があり、同時に複数の割り込み要因が格納される場合があります。複数の割り込みから一つの要因を抽出する場合には以下のように、抽出したい割り込み要因との論理積をとって、それを抽出したい割り込み要因と一致しているか確認してください。

```
if ((dwRequestPostmessage & DI_EVENT )== DI_EVENT)
        {
        割り込み処理
        }
```

COUNTER_EVENT	エンコーダカウンター割り込み。
DI_EVENT	DI エッジ割り込み。
DMA_SIGNAL	リングバッファ割り込み。本メッセージを受信したら、速やかにバッファ読み出し、書き込みを行って下さい。

dwDI_CT_interrupt	DI 割り込み・エンコーダカウンター割り込み要因の詳細。
	Bit00　カウンター0 コンペア・比較対象と一致
	Bit01　カウンター0 コンペア・比較対象を上回った
	Bit02　カウンター0 コンペア・比較対象を下回った
	Bit03　カウンター0 コンペア・比較対象範囲に入った
	Bit04　カウンター1 コンペア・比較対象と一致
	Bit05　カウンター1 コンペア・比較対象を上回った
	Bit06　カウンター1 コンペア・比較対象を下回った
	Bit07　カウンター1 コンペア・比較対象範囲に入った
	Bit08　カウンター2 コンペア・比較対象と一致
	Bit09　カウンター2 コンペア・比較対象を上回った
	Bit10　カウンター2 コンペア・比較対象を下回った
	Bit11　カウンター2 コンペア・比較対象範囲に入った
	Bit12　カウンター3 コンペア・比較対象と一致
	Bit13　カウンター3 コンペア・比較対象を上回った
	Bit14　カウンター3 コンペア・比較対象を下回った
	Bit15　カウンター3 コンペア・比較対象範囲に入った
	Bit16　DI16(ADXⅡ85-1M-PCIEX)／DI0(前記以外の機種)のエッジ割り込み
	Bit17　DI17(ADXⅡ85-1M-PCIEX)／DI1(前記以外の機種)のエッジ割り込み
	Bit18　DI18(ADXⅡ85-1M-PCIEX)／DI2(前記以外の機種)のエッジ割り込み
	Bit19　DI19(ADXⅡ85-1M-PCIEX)／DI3(前記以外の機種)のエッジ割り込み
	Bit20　DI20(ADXⅡ85-1M-PCIEX)／DI4(前記以外の機種)のエッジ割り込み
	Bit21　DI21(ADXⅡ85-1M-PCIEX)／DI5(前記以外の機種)のエッジ割り込み
	Bit22　DI22(ADXⅡ85-1M-PCIEX)／DI6(前記以外の機種)のエッジ割り込み
	Bit23　DI23(ADXⅡ85-1M-PCIEX)／DI7(前記以外の機種)のエッジ割り込み
	Bit24　DI24(ADXⅡ85-1M-PCIEX)／DI8(前記以外の機種)のエッジ割り込み
	Bit25　DI25(ADXⅡ85-1M-PCIEX)／DI9(前記以外の機種)のエッジ割り込み
	Bit26　DI26(ADXⅡ85-1M-PCIEX)／DI10(前記以外の機種)のエッジ割り込み
	Bit27　DI27(ADXⅡ85-1M-PCIEX)／DI11(前記以外の機種)のエッジ割り込み
	Bit28　DI28(ADXⅡ85-1M-PCIEX)／DI12(前記以外の機種)のエッジ割り込み
	Bit29　DI29(ADXⅡ85-1M-PCIEX)／DI13(前記以外の機種)のエッジ割り込み
	Bit30　DI30(ADXⅡ85-1M-PCIEX)／DI14(前記以外の機種)のエッジ割り込み
	Bit31　DI31(ADXⅡ85-1M-PCIEX)／DI15(前記以外の機種)のエッジ割り込み

TXBUFSETUP2

リングバッファ、トリガコントローラ、トリガソース、サンプリングタイマー、シーケンシャル取り込みモード等の設定項目を格納します。この構造体で定義された機能は、バッファトリガエンジンへのコマンドになります。従来の ADiox シリーズに比べ大幅に強化されました。

C/C++

```
struct TXBUFSETUP2
        {
        DWORD dwClockScall;
        DWORD dwStartTrigDelay;
        DWORD dwStartTrigLevel1;
        DWORD dwStartTrigLevel2;
        DWORD dwStopTrigDelay;
        DWORD dwStopTrigLevel1;
        DWORD dwStopTrigLevel2;
        DWORD dwStartMask;
        DWORD dwStartDiPattern;
        DWORD dwStartTrigSourceDI_ch;
        DWORD dwStopMask;
        DWORD dwStopDiPattern;
        DWORD dwStopTrigSourceDI_ch;
        DWORD dwTrigStopMode;
        DWORD dwTrigStartMode;
        DWORD dwPreTrigger;
        DWORD dwIamSlave;
        DWORD dwAnalogTrigSourceStart;
        DWORD dwDigitalTrigSourceStart;
        DWORD dwAnalogTrigSourceStop;
        DWORD dwDigitalTrigSourceStop;
        DWORD dwIntrruptMode;
        DWORD dwDeadTime;
        DWORD dwStopCounterValue;
        DWORD dwMuxSeqenceAuto;
        DWORD dwAoHspBufferd;
        DWORD dwDoHspBufferd;
        DWORD dwAdiBufferOn;
        BYTE    bConnectBuffer;
        BOOL    bTrigEnable;
        };
```

C#

```
public struct TXBUFSETUP2
        {
        public uint         dwClockScall;
        public uint         dwStartTrigDelay;
        public uint         dwStartTrigLevel1;
        public uint         dwStartTrigLevel2;
        public uint         dwStopTrigDelay;
        public uint         dwStopTrigLevel1;
        public uint         dwStopTrigLevel2;
        public uint         dwStartMask;
        public uint         dwStartDiPattern;
        public uint         dwStartTrigSourceDI_ch;
        public uint         dwStopMask;
        public uint         dwStopDiPattern;
        public uint         dwStopTrigSourceDI_ch;
        public uint         dwTrigStopMode;
        public uint         dwTrigStartMode;
        public uint         dwPreTrigger;
        public uint         dwIamSlave;
        public uint         dwAnalogTrigSourceStart;
        public uint         dwDigitalTrigSourceStart;
        public uint         dwAnalogTrigSourceStop;
        public uint         dwDigitalTrigSourceStop;
        public uint         dwIntrruptMode;
        public uint         dwDeadTime;
        public uint         dwStopCounterValue;
        public uint         dwMuxSeqenceAuto;
        public uint         dwAoHspBufferd;
        public uint         dwDoHspBufferd;
        public uint         dwAdiBufferOn;
        public byte         bConnectBuffer;
        public int          bTrigEnable;
        };
```

メンバ変数

[サンプリング周波数]

dwClockScall　サンプリング周波数を指定します。シーケンシャル取り込みチャンネル数を n とした場合の、サンプリング周波数の設定方法は以下の通り。（機種毎にことなります）

ADXII85-1M-PCIEX
サンプリング周波数＝(40MHz÷dwClockScall)÷n
ADXII42C-2K-Ethernet, ADXII42C-WiFi, ADXII42C-CORE, ADXII-INF01
サンプリング時間＝16.95421ns×dwClockScall×n
ADXII42FE-250K-Ethernet, ADXII42FE-MPU, ADXII42FE-CORE, ADXII-INF02
サンプリング時間＝20.34505208333ns×dwClockScall×n
ADXII14-125M-PCIEX
0x0=90MHz,0x1=45MHz,0x3=22.5MHz,0x7=11.25MHz,0xF=5.625MHz
0x10=125MHz,0x11=62.5MHz,0x13=31.25MHz,0x17=15.625MHz,0x1F=7.8125MHz,
0x20=ExCLK,0x21=ExCLK/2,0x23=ExCLK/4,0x21=ExCLK/8,0x2F=ExCLK/16
※ExCLK は外部クロックの意味

[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	**トリガ条件は成立しない**
BURST	**無条件にトリガを成立させる。**
DI_POSEDGE	**デジタル入出力の立ち上がりエッジでトリガを成立させる**
DI_NEGEDGE	**デジタル入出力の立ち下がりエッジでトリガを成立させる**
DI_PATTERN	**デジタル入出力が指定したパターンとなったときにトリガを成立させる**
AI_LEVEL	**アナログ入出力のレベル（エッジ）トリガ**
AI_AREA	**アナログ入出力のエリアトリガ**

[トリガソース]

dwAnalogTrigSourceStart　アナログスタートトリガソースを指定します。以下の中から選択できます。
dwAnalogTrigSourceStop　アナログストップトリガソースを指定します。以下の中から選択できます。

ADXⅡ14-125M-PCIEX	0:AI0	1:AI1	2:AO0	3:AO1		
ADXⅡ42***	0:AI	1:AO0	2:AO1			
ADXⅡ85-1M-PCIEX	0:AI	1:AO0	2:AO1	3:AO2	4:AO3	5:AO4
ADXⅡINF***	0:AI					

dwDigitalTrigSourceStart　デジタルスタートトリガソースを指定します。以下の中から選択できます。
dwDigitalTrigSourceStop　デジタルストップトリガソースを指定します。以下の中から選択できます。
0: DI　1:DO　2:カウンタコンペア（※）

※　IRQ_BUFFER.dwDI_CT_interrupt の Bit0-15 に相当する値を DI に見立ててトリガソースとします。

[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。
（アナログレベル最小～最大が、0～0xFFFF に対応します）
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。
（アナログレベル最小～最大が、0～0xFFFF に対応します）
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。
（アナログレベル最小～最大が、0～0xFFFF に対応します）
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。
（アナログレベル最小～最大が、0～0xFFFF に対応します）

[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。何チャンネル目のデジタル入出力でエッジトリガをかけるか設定します。0-31 のいずれかを指定してください。
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。何チャンネル目のデジタル入出力でエッジトリガをかけるか設定します。0-31 のいずれかを指定してください。

[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、デジタル入出力のチャンネルに相当します。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。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、デジタル入出力のチャンネルに相当します。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。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出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出力値が一致した場合に、トリガが成立します。

[トリガディレイ・プリトリガ]

dwStartTrigDelay	スタートトリガディレイの指定。(プリトリガ)この値だけトリガよりも送れて、データの取り込みを開始します。最大 65535 まで指定できます。
dwStopTrigDelay	ストップトリガディレイの指定。(プリトリガ)この値だけトリガよりも送れて、データの取り込みを終了します。最大 65535 まで指定できます。
dwPreTrigger	プリトリガの on/off を指定します。1 で on、0 で off です。トリガよりも n サンプル先に、データ収集を開始します。ADXⅡ14-125M-PCIEX(n=256)、ADXⅡ85-1M-PCIEX(n=32)のみ有効です。

[同期運転]

dwIamSlave	複数ボードの同期運転を行う場合、自分がマスタになるかスレーブになるかを指定します。1 でスレーブ、0 でマスタです。単独使用の場合には必ずマスタ(0)にしてください。

[トリガ・リングバッファ開始停止・自動停止]

dwIntrruptMode	DMA_INT、NOT_INT で開始、NOT_INT で停止となります。
dwStopCounterValue	この値で指定した分のバンクチェンジが発生すると自動停止します。プライマリオンチップリングバッファまたは、リングバッファの容量に本変数を乗算したサンプル数で自動停止します。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。値はサンプル数で、0-0xFFFFFFFF まで有効です。

[その他様々な機能]

dwMuxSeqenceAuto	シーケンシャル取り込みの設定を行います。ADXⅡ42***、ADXⅡINF***では本数値は 0-3 までとしてください。ADXⅡ85-1M-PCIEX は 0-4 までの数値としてください。本変数は同時サンプルの ADXⅡ14-125M-PCIEX では無意味です。値の意味は以下の通りです。 値が 0 の場合、シーケンシャル取り込みはディセーブル 値が 1 の場合、2ch 自動切換え 値が 2 の場合、4ch 自動切換え 値が 3 の場合、8ch 自動切換え 値が 4 の場合、16ch 自動切換え
dwAoHspBufferd	AO をリングバッファ経由にするか否か。0 でポーリング、1 でリングバッファー経由。 リングバッファ経由にできる AO チャンネルはハードウェアにより固定されています。
dwDoHspBufferd	DO を経由にするか否か。0 でポーリング、1 でリングバッファー経由。 リングバッファ経由にできる DO チャンネルはハードウェアにより固定されています。
dwAdiBufferOn	ADXⅡ14-125M-PCIEX で AI/DI をリングバッファ経由にするか否か？1 でリングバッファ経由にします。本変数を 0 、dwAoHspBufferd=1、dwDoHspBufferd=1 にして AO/DO のみをリングバッファ経由にすることができます。
bConnectBuffer	ADXⅡ85-1M-PCIEX, ADXⅡ42***にて、エンコーダカウンタをリングバッファ経由にするか否かを指定します。0 でしない、1 でカウンタ CH0、2 でカウンタ CH0+CH1、3 でカウンタ CH0+CH1+CH2+CH3 をリングバッファ経由にします。エンコーダーカウンタのリングバッファ経由をしようしている間は DI はリングバッファ経由になりません。
bTrigEnable	TRUE(=1)でトリガイネーブルが有効になります。FALSE(=0)で無効になります。

IOGEOSETUP2

アナログ入出力・デジタル入出力の設定項目を格納します。アナログ出力モード、アナログ入力モード、アナログ入力チャンネル、差動/シングルエンド切り替え、アナログ入力レンジ、アナログ出力レンジ、デジタルフィルタ、チャタリングキャンセラ等の設定をまとめてあります。

C/C++

```
struct IOGEOSETUP2
        {
        DWORD dwAo3Mode;
        DWORD dwAo2Mode;
        DWORD dwAo1Mode;
        DWORD dwAo0Mode;
        DWORD dwInputShort;
        DWORD dwCOB;
        DWORD dwFilterEnable;
        DWORD dwDifferential;
        DWORD dwMux;
        DWORD dwAI_Range;
        DWORD dwAO_Range;
        DWORD dwChatCan;
        DWORD dwNoiseShaper;
        };
```

C#

```
struct IOGEOSETUP2
        {
        public uint dwAo3Mode;
        public uint dwAo2Mode;
        public uint dwAo1Mode;
        public uint dwAo0Mode;
        public uint dwInputShort;
        public uint dwCOB;
        public uint dwFilterEnable;
        public uint dwDifferential;
        public uint dwMux;
        public uint dwAI_Range;
        public uint dwAO_Range;
        public uint dwChatCan;
        public uint dwNoiseShaper;
        };
```

メンバ変数

dwAo0Mode　アナログ出力チャンネル 0 の動作モードを指定します。動作モードは以下の 3 つから選択してください。

dwAo1Mode　アナログ出力チャンネル 1 の動作モードを指定します。動作モードは以下の 3 つから選択してください。

dwAo2Mode　アナログ出力チャンネル 2 の動作モードを指定します。動作モードは以下の 3 つから選択してください。

dwAo3Mode　アナログ出力チャンネル 3 の動作モードを指定します。動作モードは以下の 3 つから選択してください。

NO_LINK　エンコーダカウンターとは連動しない、通常のアナログ出力モード。関数 bADioxADIO で指定したアナログ出力値が採用されます。

LIVE_CTC　エンコーダカウンターの現在値をアナログ出力値にします。同一チャンネル数のカウンタの値とリンクします。

LATCH_CTC　エンコーダカウンターのラッチ値をアナログ出力値にします。同一チャンネル数のカウンタの値とリンクします。

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、3 でループバック、2 で+5V のオンボードリファレンス電圧（高精度）に接続されます。ADXⅡ85-1M-PCIEX 専用の変数です。

dwCOB　この値が 1 の場合、アナログ入力値はコンプリメントバイナリ(2 の補数=short 型相当)となります。
この値が 0 の場合、アナログ入力値はストレートバイナリ(WORD 型相当)となります。
両者の変換はハードウェアで行われるので、負荷がかかりません。

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。以下のように、本変数を 0,1,3 とすることで、フィルタの次数が変わります。4 次は移動平均、5 次と 16 次はローパスフィルタです。

ADXⅡ85-1M-PCIEX			
	シーケンシャル取り込み有効	0:OFF	1:ON4 次　3:ON4 次
	シーケンシャル取り込み無効	0:OFF	1:ON5 次　3:ON16 次
ADXⅡ42***		0:OFF	1:ON8 次　3:ON8 次
ADXⅡINF***		0:OFF	1:ON8 次　3:ON8 次
ADXⅡ14-125M-PCIEX		0:OFF	1:ON5 次　3:ON16 次

dwNoiseShaper　この値が 0 の場合、アナログ入力信号へのデジタルフィルタのノイズシェーパーを OFF にします。
この値が 1 の場合、アナログ入力信号へのデジタルフィルタのノイズシェーパーを ON にします。
ノイズシェーパーは 5 次 16 次ローパスフィルタでのみ有効です。
※　ノイズシェーパーはフィルターによるビット落ちを積算して最下位ビットに加算します。

dwDifferential	この値が 0 の場合、シングルエンド 16 チャンネルアナログ入力となります。 この値が 1 の場合、差動 8 チャンネルアナログ入力となります。 ADXⅡ85-1M-PCIEX 専用の変数です。
dwMux	入力チャンネルを切り替えます。ADXⅡ85-1M-PCIEX ではシングルエンド 16 チャンネルの場合 0-15、差動 8 チャンネルの場合 0-7 が有効な値です。ADXⅡ42***、ADXⅡINF***では 0-7 が有効です。ADXⅡ14-125M-PCIEX では同時サンプルなので無意味です。
dwAI_Range	アナログ入力レンジ(フルスケール)を設定します。0-7 まで有効で、0 から順番に入力レンジを並べると "±10.0V","±5.0V","±2.5V","±1.25V","+10V","+5V","+2.5V","+1.25V"となります。ADXⅡ85-1M-PCIEX 専用の変数です。
dwAO_Range	アナログ出力レンジ(フルスケール)を設定します。0-7 まで有効で、0 から順番に入力レンジを並べると "±5.0V","±2.5V","±1.25V","±625mV","+10V","+5V","+2.5V","+1.25V"となります。ADXⅡ85-1M-PCIEX 専用の変数です。
dwChatCan	この値が 0 の場合、デジタル入力のチャタリングキャンセラー(フィルタ)を OFF にします。 この値が 1 の場合、デジタル入力のチャタリングキャンセラー(フィルタ)を ON にします。

TDIO_MISC

エンコーダーカウンター、周波数カウンター、PWM(パルスジェネレーター)、ストローブラッチ、DI エッジ割り込みの設定を格納します。

C/C++

```
struct  TDIO_MISC
        {
        DWORD dwStrobeInternal;
        DWORD dwSetFreq;
        DWORD dwLinkPwmToDO;
        DWORD dwDinIntMode16;
        DWORD dwDinIntMode17;
        DWORD dwDinIntMode18;
        DWORD dwDinIntMode19;
        DWORD dwDinIntMode20;
        DWORD dwDinIntMode21;
        DWORD dwDinIntMode22;
        DWORD dwDinIntMode23;
        DWORD dwDinIntMode24;
        DWORD dwDinIntMode25;
        DWORD dwDinIntMode26;
        DWORD dwDinIntMode27;
        DWORD dwDinIntMode28;
        DWORD dwDinIntMode29;
        DWORD dwDinIntMode30;
        DWORD dwDinIntMode31;
        DWORD dwCounterMode_A;
        DWORD dwCounterMode_B;
        DWORD dwCounterMode_C;
        DWORD dwCounterMode_D;
        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;
        DWORD dwCompareMode;
        DWORD dwLinkCounterToDO_A;
        DWORD dwLinkCounterToDO_B;
        DWORD dwLinkCounterToDO_C;
        DWORD dwLinkCounterToDO_D;
        DWORD dwCounterIntMode_A;
        DWORD dwCounterIntMode_B;
        DWORD dwCounterIntMode_C;
        DWORD dwCounterIntMode_D;
        DWORD dwReferenceLow_A;
        DWORD dwReferenceLow_B;
        DWORD dwReferenceLow_C;
        DWORD dwReferenceLow_D;
        DWORD dwReferenceHigh_A;
        DWORD dwReferenceHigh_B;
        DWORD dwReferenceHigh_C;
        DWORD dwReferenceHigh_D;
        DWORD dwSetGate;
        };
```

C#

```
public struct TDIO_MISC
{
public uint dwStrobeInternal;
public uint dwSetFreq;
public uint dwLinkPwmToDO;
public uint dwDinIntMode16;
public uint dwDinIntMode17;
public uint dwDinIntMode18;
public uint dwDinIntMode19;
public uint dwDinIntMode20;
public uint dwDinIntMode21;
public uint dwDinIntMode22;
public uint dwDinIntMode23;
public uint dwDinIntMode24;
public uint dwDinIntMode25;
public uint dwDinIntMode26;
public uint dwDinIntMode27;
public uint dwDinIntMode28;
public uint dwDinIntMode29;
public uint dwDinIntMode30;
public uint dwDinIntMode31;
public uint dwCounterMode_A;
public uint dwCounterMode_B;
public uint dwCounterMode_C;
public uint dwCounterMode_D;
public uint dwLatchMode_A;
public uint dwLatchMode_B;
public uint dwLatchMode_C;
public uint dwLatchMode_D;
public uint dwZ_CENTER_A;
public uint dwZ_CENTER_B;
public uint dwZ_CENTER_C;
public uint dwZ_CENTER_D;
public uint dwSoftwareClear_A;
public uint dwSoftwareClear_B;
public uint dwSoftwareClear_C;
public uint dwSoftwareClear_D;
public uint dwCompareMode;
public uint dwLinkCounterToDO_A;
public uint dwLinkCounterToDO_B;
public uint dwLinkCounterToDO_C;
public uint dwLinkCounterToDO_D;
public uint dwCounterIntMode_A;
public uint dwCounterIntMode_B;
public uint dwCounterIntMode_C;
public uint dwCounterIntMode_D;
public uint dwReferenceLow_A;
public uint dwReferenceLow_B;
public uint dwReferenceLow_C;
public uint dwReferenceLow_D;
public uint dwReferenceHigh_A;
public uint dwReferenceHigh_B;
public uint dwReferenceHigh_C;
public uint dwReferenceHigh_D;
public uint dwSetGate;
};
```

メンバ変数

dwStrobeInternal	ストローブ入出力をデジタル入出力チャンネル 31（ADXⅡ85-1M-PCIEX）、デジタル出力 15(ADXⅡ42***)、デジタル出力 3(ADXⅡ14-125M-PCIEX)に埋め込むか否かを設定します。1 で埋め込み、0 で埋め込まない（ストローブ専用ヘッダコネクタまたはピンを使用）
dwSetFreq	PWM 周期を設定します。この値と PWM サイクル周波数の関係は以下の通りです。 0=1.96608msec　1=3.93216msec 2=7.86432msec　3=15.72864msec 4=31.45728msec　5=62.91456msec 6=125.82912msec　7=251.65824msec
dwLinkPwmToDO	PWM 出力をデジタル出力に出すか否かを設定します。この変数のビットフィールドが、PWM チャンネルに相当します。ビットフィールド 1 で PWM 有効になります。ADXⅡ85-1M-PCIEX では 16ch、ADXⅡ42***では 8ch、ADXⅡ14-125M-PCIEX では 4ch の PWM を有します。
dwDinIntMode16	DI16(ADXⅡ85-1M-PCIEX)／DI0(前記以外の機種)のエッジ割り込み要因を指定します。
dwDinIntMode17	DI17(ADXⅡ85-1M-PCIEX)／DI1(前記以外の機種)のエッジ割り込み要因を指定します。
dwDinIntMode18	DI18(ADXⅡ85-1M-PCIEX)／DI2(前記以外の機種)のエッジ割り込み要因を指定します。
dwDinIntMode19	DI19(ADXⅡ85-1M-PCIEX)／DI3(前記以外の機種)のエッジ割り込み要因を指定します。
dwDinIntMode20	DI20(ADXⅡ85-1M-PCIEX)／DI4(前記以外の機種)のエッジ割り込み要因を指定します。
dwDinIntMode21	DI21(ADXⅡ85-1M-PCIEX)／DI5(前記以外の機種)のエッジ割り込み要因を指定します。
dwDinIntMode22	DI22(ADXⅡ85-1M-PCIEX)／DI6(前記以外の機種)のエッジ割り込み要因を指定します。
dwDinIntMode23	DI23(ADXⅡ85-1M-PCIEX)／DI7(前記以外の機種)のエッジ割り込み要因を指定します。
dwDinIntMode24	DI24(ADXⅡ85-1M-PCIEX)／DI8(前記以外の機種)のエッジ割り込み要因を指定します。
dwDinIntMode25	DI25(ADXⅡ85-1M-PCIEX)／DI9(前記以外の機種)のエッジ割り込み要因を指定します。
dwDinIntMode26	DI26(ADXⅡ85-1M-PCIEX)／DI10(前記以外の機種)のエッジ割り込み要因を指定します。
dwDinIntMode27	DI27(ADXⅡ85-1M-PCIEX)／DI11(前記以外の機種)のエッジ割り込み要因を指定します。
dwDinIntMode28	DI28(ADXⅡ85-1M-PCIEX)／DI12(前記以外の機種)のエッジ割り込み要因を指定します。
dwDinIntMode29	DI29(ADXⅡ85-1M-PCIEX)／DI13(前記以外の機種)のエッジ割り込み要因を指定します。
dwDinIntMode30	DI30(ADXⅡ85-1M-PCIEX)／DI14(前記以外の機種)のエッジ割り込み要因を指定します。
dwDinIntMode31	DI31(ADXⅡ85-1M-PCIEX)／DI15(前記以外の機種)のエッジ割り込み要因を指定します。 ＜↑dwDinIntMode16～31 で指定する割り込み要因の種類＞ **NO_INT**　割り込み要因なし **POSEDGE_INT**　立ち上がりエッジ **NEGEDGE_INT**　立下りエッジ **DUALEDGE_INT**デュアルエッジ
dwCounterMode_A	カウンター0 の動作モードを以下の 0-7 で指定します。
dwCounterMode_B	カウンター1 の動作モードを以下の 0-7 で指定します。
dwCounterMode_C	カウンター2 の動作モードを以下の 0-7 で指定します。
dwCounterMode_D	ダカウンター3 の動作モードを以下の 0-7 で指定します。 ＜↑dwCounterMode_A～D で指定するエンコーダカウンタモード＞ 0:4 倍速エンコーダカウンター、Z 相未使用 1:4 倍速エンコーダカウンター、Z 相使用 2:2 倍速エンコーダカウンター、Z 相未使用 3:2 倍速エンコーダカウンター、Z 相使用 4:1 倍速エンコーダカウンター、Z 相未使用 5:1 倍速エンコーダカウンター、Z 相使用 6:アップダウンカウンター（パルスカウンター）、Z 相未使用 7:アップダウンカウンター（パルスカウンター）、Z 相使用 (ADXⅡ14-125M-PCIEX ではカウンター0 のみ)
(※以降 ADXⅡ14-125M-PCIEX ではカウンター0 のみ)	
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。 ＜↑dwLatchMode_A～D で指定するラッチモード＞ **SOFT**　ソフトウェアラッチ **Z_PHASE**　Z 相条件成立でラッチ **DI_SEL**　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。 ＜dwZ_CENTER_A ～D で指定する Z 相成立条件＝カウンタリセット＝原点復帰＞ 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。
dwCompareMode	0 を指定すると、カウンターコンペア値 LOW は、dwReferenceLow_A、dwReferenceLow_B、dwReferenceLow_C、dwReferenceLow_D を使用する。1 を指定すると、隣接カウンター（若い方）のカウンター値をカウンターコンペア値 LOW として利用する。カウンター0 のみカウンター3 の値を使う。
dwLinkCounterToDO_A	カウンター0 のコンペア結果を DO にリンクするか否かを指定します。0x0-0xF が有効な値で、各ビットフィールドは以下の意味を持ちます。
dwLinkCounterToDO_B	カウンター1 のコンペア結果を DO にリンクするか否かを指定します。0x0-0xF が有効な値で、各ビットフィールドは以下の意味を持ちます。
dwLinkCounterToDO_C	カウンター2 のコンペア結果を DO にリンクするか否かを指定します。0x0-0xF が有効な値で、各ビットフィールドは以下の意味を持ちます。
dwLinkCounterToDO_D	カウンター3 のコンペア結果を DO にリンクするか否かを指定します。0x0-0xF が有効な値で、各ビットフィールドは以下の意味を持ちます。 ＜↑dwLinkCounterToDO_A～D のビットフィールド＞ bit0=コンペア値 LOW と一致 bit1=コンペア値 LOW 以上 bit2=コンペア値 LOW 以下 bit3=コンペア値 LOW～コンペア値 HIGH の範囲内 カンウターコンペア出力と DO の割り付けはハードウェア仕様書(カウンタの章)に記載。
dwCounterIntMode_A	カウンター0 のコンペア結果を割り込みにリンクするか否かを指定します。0x0-0xF が有効な値で、各ビットフィールドは以下の意味を持ちます。
dwCounterIntMode_B	カウンター1 のコンペア結果を割り込みにリンクするか否かを指定します。0x0-0xF が有効な値で、各ビットフィールドは以下の意味を持ちます。
dwCounterIntMode_C	カウンター2 のコンペア結果を割り込みにリンクするか否かを指定します。0x0-0xF が有効な値で、各ビットフィールドは以下の意味を持ちます。
dwCounterIntMode_D	カウンター3 のコンペア結果を割り込みにリンクするか否かを指定します。0x0-0xF が有効な値で、各ビットフィールドは以下の意味を持ちます。 ＜dwCounterIntMode_A ～D のビットフィールド＞ bit0=コンペア値 LOW と一致 bit1=コンペア値 LOW 以上 bit2=コンペア値 LOW 以下 bit3=コンペア値 LOW～コンペア値 HIGH の範囲内
dwReferenceLow_A	カウンター0 コンペア値 LOW （32Bit カウンタなので 0～0xFFFFFFFF を指定して下さい）
dwReferenceLow_B	カウンター1 コンペア値 LOW （32Bit カウンタなので 0～0xFFFFFFFF を指定して下さい）
dwReferenceLow_C	カウンター2 コンペア値 LOW （32Bit カウンタなので 0～0xFFFFFFFF を指定して下さい）
dwReferenceLow_D	カウンター3 コンペア値 LOW （32Bit カウンタなので 0～0xFFFFFFFF を指定して下さい）
dwReferenceHigh_A	カウンター0 コンペア値 HIGH （32Bit カウンタなので 0～0xFFFFFFFF を指定して下さい）
dwReferenceHigh_B	カウンター1 コンペア値 HIGH （32Bit カウンタなので 0～0xFFFFFFFF を指定して下さい）
dwReferenceHigh_C	カウンター2 コンペア値 HIGH （32Bit カウンタなので 0～0xFFFFFFFF を指定して下さい）
dwReferenceHigh_D	カウンター3 コンペア値 HIGH （32Bit カウンタなので 0～0xFFFFFFFF を指定して下さい）
dwSetGate	周波数カウンターのゲートを指定します。ゲートとは、パルスをカウントする時間です。1 秒の場合、カウントした値はそのまま Hz(ヘルツ)になります。以下の 0-3 のいずれかを指定してください。 0:1sec、1:100msec、2:10msec、3:1msec

[カウンタコンペア出力の DO 出力の割付]

カウンタ 0 コンペア値 LOW と一致	ADXⅡ42***	DO00
	ADXⅡ85-1M-PCIEX	DO16
	ADXⅡ14-125M-PCIEX	DO00
カウンタ 0 コンペア値 LOW 以上	ADXⅡ42***	DO01
	ADXⅡ85-1M-PCIEX	DO17
	ADXⅡ14-125M-PCIEX	DO01
カウンタ 0 コンペア値 LOW 以下	ADXⅡ42***	DO02
	ADXⅡ85-1M-PCIEX	DO18
	ADXⅡ14-125M-PCIEX	DO02
カウンタ 0 コンペア値 LOW～HIGH の範囲内	ADXⅡ42***	DO03
	ADXⅡ85-1M-PCIEX	DO19
	ADXⅡ14-125M-PCIEX	DO03
カウンタ 1 コンペア値 LOW と一致	ADXⅡ42***	DO04
	ADXⅡ85-1M-PCIEX	DO20
カウンタ 1 コンペア値 LOW 以上	ADXⅡ42***	DO05
	ADXⅡ85-1M-PCIEX	DO21
カウンタ 1 コンペア値 LOW 以下	ADXⅡ42***	DO06
	ADXⅡ85-1M-PCIEX	DO22
カウンタ 1 コンペア値 LOW～HIGH の範囲内	ADXⅡ42***	DO07
	ADXⅡ85-1M-PCIEX	DO23
カウンタ 2 コンペア値 LOW と一致	ADXⅡ42***	DO08
	ADXⅡ85-1M-PCIEX	DO24
カウンタ 2 コンペア値 LOW 以上	ADXⅡ42***	DO09
	ADXⅡ85-1M-PCIEX	DO25
カウンタ 2 コンペア値 LOW 以下	ADXⅡ42***	DO10
	ADXⅡ85-1M-PCIEX	DO26
カウンタ 2 コンペア値 LOW～HIGH の範囲内	ADXⅡ42***	DO11
	ADXⅡ85-1M-PCIEX	DO27
カウンタ 3 コンペア値 LOW と一致	ADXⅡ42***	DO12
	ADXⅡ85-1M-PCIEX	DO28
カウンタ 3 コンペア値 LOW 以上	ADXⅡ42***	DO13
	ADXⅡ85-1M-PCIEX	DO29
カウンタ 3 コンペア値 LOW 以下	ADXⅡ42***	DO14
	ADXⅡ85-1M-PCIEX	DO30
カウンタ 3 コンペア値 LOW～HIGH の範囲内	ADXⅡ42***	DO15
	ADXⅡ85-1M-PCIEX	DO31

TADIO2

アナログデジタル入出力・エンコーダカウンタ・周波数カウンタ・温度の一斉ポーリングのための構造体です。アナログ入出力を電圧値に変換した値を格納しています。

C/C++

```
struct   TADIO2
         {
         DWORD dwAi0;
         DWORD dwAi1;
         DWORD dwAi2;
         DWORD dwAi3;
         DWORD dwAi4;
         DWORD dwAi5;
         DWORD dwAi6;
         DWORD dwAi7;
         DWORD dwAo0;
         DWORD dwAo1;
         DWORD dwAo2;
         DWORD dwAo3;
         DWORD dwAo4;
         DWORD dwAo5;
         DWORD dwAo6;
         DWORD dwAo7;
         DWORD dwDOS;
         DWORD dwDI;
         DWORD dwDI_Latch;
         double  dAi0;
         double  dAi1;
         double  dAi2;
         double  dAi3;
         double  dAi4;
         double  dAi5;
         double  dAi6;
         double  dAi7;
         double  dAo0;
         double  dAo1;
         double  dAo2;
         double  dAo3;
         double  dAo4;
         double  dAo5;
         double  dAo6;
         double  dAo7;
         DWORD dwCounterA;
         DWORD dwCounterB;
         DWORD dwCounterC;
         DWORD dwCounterD;
         DWORD dwLatchA;
         DWORD dwLatchB;
         DWORD dwLatchC;
         DWORD dwLatchD;
         DWORD dwFreqLatch_A;
         DWORD dwFreqLatch_B;
         DWORD dwFreqLatch_C;
         DWORD dwFreqLatch_D;
         double  dTemp;
         DWORD dwMode;
         };
```

C#

```
public struct TADIO2
        {
        public uint         dwAi0;
        public uint         dwAi1;
        public uint         dwAi2;
        public uint         dwAi3;
        public uint         dwAi4;
        public uint         dwAi5;
        public uint         dwAi6;
        public uint         dwAi7;
        public uint         dwAo0;
        public uint         dwAo1;
        public uint         dwAo2;
        public uint         dwAo3;
        public uint         dwAo4;
        public uint         dwAo5;
        public uint         dwAo6;
        public uint         dwAo7;
        public uint         dwDOS;
        public uint         dwDI;
        public uint         dwDI_Latch;
        public double       dAi0;
        public double       dAi1;
        public double       dAi2;
        public double       dAi3;
        public double       dAi4;
        public double       dAi5;
        public double       dAi6;
        public double       dAi7;
        public double       dAo0;
        public double       dAo1;
        public double       dAo2;
        public double       dAo3;
        public double       dAo4;
        public double       dAo5;
        public double       dAo6;
        public double       dAo7;
        public uint         dwCounterA;
        public uint         dwCounterB;
        public uint         dwCounterC;
        public uint         dwCounterD;
        public uint         dwLatchA;
        public uint         dwLatchB;
        public uint         dwLatchC;
        public uint         dwLatchD;
        public uint         dwFreqLatch_A;
        public uint         dwFreqLatch_B;
        public uint         dwFreqLatch_C;
        public uint         dwFreqLatch_D;
        public double       dTemp;
        public uint         dwMode;
        };
```

メンバ変数

dwAi0-7	アナログ入力値。同時サンプルではない ADXⅡ14-125M-PCIEX 以外の機種は dwAi0 のみ有効で他の変数は使われていません。同時サンプルの ADXⅡ14-125M-PCIEX は dwAi0 に AI0、dwAi1 に AI1 の値が格納されます。dwAi2-7 は現在使われていません。
dAi0-7	前記 dwAi0-7 アナログ入力値を電圧に変換した値が格納されます。単位は mV になります。dwAi0-7 が dAi0-7 に相当します。
dwAo0-7	アナログ出力チャンネル 0-7 の出力値（最小～最大が、0～0xFFFF に対応します） ADXⅡ85-1M-PCIEX では dwAo0-5、ADXⅡ14-125M-PCIEX、ADXⅡ42*** では dwAo0-1 が有効です。
dAo0-7	前記 dwAo0-7 アナログ出力値を電圧に変換した値が格納されます。単位は mV になります。dwAo0-7 が dAo0-7 に相当します。
dwDOS	デジタル出力チャンネル 0～31 の出力値 （Bit0～Bit31 がデジタル出力チャンネル 0～31 に対応します ） （ハードウェアに実装されていないデジタル出力チャンネルのビットフィールド値は無視されます）
dwDI	デジタル入力チャンネル 0～31 の入力値 （Bit0～Bit31 がデジタル入力チャンネル 0～31 に対応します ） （ハードウェアに実装されていないデジタル出力チャンネルのビットフィールド値は 0 になります）
dwDI_Latch	デジタル入力チャンネル 0～31 のストローブラッチ値 （Bit0～Bit31 がデジタル入力チャンネル 0～31 に対応します ） （ハードウェアに実装されていないデジタル出力チャンネルのビットフィールド値は 0 になります）
dwCounterA	エンコーダーカウンター0 のライブ値 （現在の値※1）
dwCounterB	エンコーダーカウンター1 のライブ値 （現在の値※1）
dwCounterC	エンコーダーカウンター2 のライブ値 （現在の値※1）
dwCounterD	エンコーダーカウンター3 のライブ値 （現在の値※1）
dwLatchA	エンコーダーカウンター0 のラッチ値 ※1
dwLatchB	エンコーダーカウンター1 のラッチ値 ※1
dwLatchC	エンコーダーカウンター2 のラッチ値 ※1
dwLatchD	エンコーダーカウンター3 のラッチ値 ※1
dwFreqLatch_A	周波数カウンター0 のライブ値 （現在の値 ※2）
dwFreqLatch_B	周波数カウンター1 のライブ値 （現在の値 ※2）
dwFreqLatch_C	周波数カウンター2 のライブ値 （現在の値 ※2）
dwFreqLatch_D	周波数カウンター3 のライブ値 （現在の値 ※2）
dTemp	ADXⅡ85-1M-PCIEX、ADXⅡ42***、ADXⅡINF***専用でボードの温度を格納します。
dwMode	必ず 0 をセットしてください。

※1 32Bit のカウンタなので、0～0xFFFFFFFF の値になります。0 でデクリメントすると 0xFFFFFFFF になります。
※2 ゲート周期に何サイクルあったかを表します。

TConfigPWM

PWM 各チャンネルの個別設定（位相、パルス発生回数）を格納します。

C/C++ struct TConfigPWM

```
{
DWORD dwCycleMax0s;
DWORD dwCycleMax1s;
DWORD dwCycleMax2s;
DWORD dwCycleMax3s;
DWORD dwCycleMax4s;
DWORD dwCycleMax5s;
DWORD dwCycleMax6s;
DWORD dwCycleMax7s;
DWORD dwCycleMax8s;
DWORD dwCycleMax9s;
DWORD dwCycleMax10s;
DWORD dwCycleMax11s;
DWORD dwCycleMax12s;
DWORD dwCycleMax13s;
DWORD dwCycleMax14s;
DWORD dwCycleMax15s;
DWORD dwPwmPhase0s;
DWORD dwPwmPhase1s;
DWORD dwPwmPhase2s;
DWORD dwPwmPhase3s;
DWORD dwPwmPhase4s;
DWORD dwPwmPhase5s;
DWORD dwPwmPhase6s;
DWORD dwPwmPhase7s;
DWORD dwPwmPhase8s;
DWORD dwPwmPhase9s;
DWORD dwPwmPhase10s;
DWORD dwPwmPhase11s;
DWORD dwPwmPhase12s;
DWORD dwPwmPhase13s;
DWORD dwPwmPhase14s;
DWORD dwPwmPhase15s;
DWORD dwStart;
};
```

C# public struct TConfigPWM

```
{
public uint dwCycleMax0s;
public uint dwCycleMax1s;
public uint dwCycleMax2s;
public uint dwCycleMax3s;
public uint dwCycleMax4s;
public uint dwCycleMax5s;
public uint dwCycleMax6s;
public uint dwCycleMax7s;
public uint dwCycleMax8s;
public uint dwCycleMax9s;
public uint dwCycleMax10s;
public uint dwCycleMax11s;
public uint dwCycleMax12s;
public uint dwCycleMax13s;
public uint dwCycleMax14s;
public uint dwCycleMax15s;
public uint dwPwmPhase0s;
public uint dwPwmPhase1s;
public uint dwPwmPhase2s;
public uint dwPwmPhase3s;
public uint dwPwmPhase4s;
public uint dwPwmPhase5s;
public uint dwPwmPhase6s;
public uint dwPwmPhase7s;
public uint dwPwmPhase8s;
public uint dwPwmPhase9s;
public uint dwPwmPhase10s;
public uint dwPwmPhase11s;
public uint dwPwmPhase12s;
public uint dwPwmPhase13s;
public uint dwPwmPhase14s;
public uint dwPwmPhase15s;
public uint dwStart;
};
```

メンバ変数

dwCycleMax0s	PWM チャンネル 0 において、PWM サイクルを何回実行するか指定します。(※1)
dwCycleMax1s	PWM チャンネル 1 において、PWM サイクルを何回実行するか指定します。(※1)
dwCycleMax2s	PWM チャンネル 2 において、PWM サイクルを何回実行するか指定します。(※1)
dwCycleMax3s	PWM チャンネル 3 において、PWM サイクルを何回実行するか指定します。(※1)
dwCycleMax4s	PWM チャンネル 4 において、PWM サイクルを何回実行するか指定します。(※1)
dwCycleMax5s	PWM チャンネル 5 において、PWM サイクルを何回実行するか指定します。(※1)
dwCycleMax6s	PWM チャンネル 6 において、PWM サイクルを何回実行するか指定します。(※1)
dwCycleMax7s	PWM チャンネル 7 において、PWM サイクルを何回実行するか指定します。(※1)
dwCycleMax8s	PWM チャンネル 8 において、PWM サイクルを何回実行するか指定します。(※1)
dwCycleMax9s	PWM チャンネル 9 において、PWM サイクルを何回実行するか指定します。(※1)
dwCycleMax10s	PWM チャンネル 10 において、PWM サイクルを何回実行するか指定します。(※1)
dwCycleMax11s	PWM チャンネル 11 において、PWM サイクルを何回実行するか指定します。(※1)
dwCycleMax12s	PWM チャンネル 12 において、PWM サイクルを何回実行するか指定します。(※1)
dwCycleMax13s	PWM チャンネル 13 において、PWM サイクルを何回実行するか指定します。(※1)
dwCycleMax14s	PWM チャンネル 14 において、PWM サイクルを何回実行するか指定します。(※1)
dwCycleMax15s	PWM チャンネル 15 において、PWM サイクルを何回実行するか指定します。(※1)
dwPwmPhase0s	PWM チャンネル 0 の開始位相を指定します。(※2)
dwPwmPhase1s	PWM チャンネル 1 の開始位相を指定します。(※2)
dwPwmPhase2s	PWM チャンネル 2 の開始位相を指定します。(※2)
dwPwmPhase3s	PWM チャンネル 3 の開始位相を指定します。(※2)
dwPwmPhase4s	PWM チャンネル 4 の開始位相を指定します。(※2)
dwPwmPhase5s	PWM チャンネル 5 の開始位相を指定します。(※2)
dwPwmPhase6s	PWM チャンネル 6 の開始位相を指定します。(※2)
dwPwmPhase7s	PWM チャンネル 7 の開始位相を指定します。(※2)
dwPwmPhase8s	PWM チャンネル 8 の開始位相を指定します。(※2)
dwPwmPhase9s	PWM チャンネル 9 の開始位相を指定します。(※2)
dwPwmPhase10s	PWM チャンネル 10 の開始位相を指定します。(※2)
dwPwmPhase11s	PWM チャンネル 11 の開始位相を指定します。(※2)
dwPwmPhase12s	PWM チャンネル 12 の開始位相を指定します。(※2)
dwPwmPhase13s	PWM チャンネル 13 の開始位相を指定します。(※2)
dwPwmPhase14s	PWM チャンネル 14 の開始位相を指定します。(※2)
dwPwmPhase15s	PWM チャンネル 15 の開始位相を指定します。(※2)
dwStart	PWM 開始・終了のコマンドです。この変数のビットフィールドが、PWM チャンネルに相当します。例えば Bit5(0x20)は、PWM チャンネル 5 に相当します。

(※1　0-255 が有効で、0 を指定した場合のみ、PWM サイクル実行回数制限なしになります)
(※2　0-255 が有効で、360/256 度=1.40625 度単位で開始位相を指定できます)

TSetupPWM

PWM デューティー比を設定します。

C/C++

```
struct TSetupPWM
        {
        DWORD dwPwm0Value;
        DWORD dwPwm1Value;
        DWORD dwPwm2Value;
        DWORD dwPwm3Value;
        DWORD dwPwm4Value;
        DWORD dwPwm5Value;
        DWORD dwPwm6Value;
        DWORD dwPwm7Value;
        DWORD dwPwm8Value;
        DWORD dwPwm9Value;
        DWORD dwPwm10Value;
        DWORD dwPwm11Value;
        DWORD dwPwm12Value;
        DWORD dwPwm13Value;
        DWORD dwPwm14Value;
        DWORD dwPwm15Value;
        };
```

C#

```
public struct TSetupPWM
        {
        public uint     dwPwm0Value;
        public uint     dwPwm1Value;
        public uint     dwPwm2Value;
        public uint     dwPwm3Value;
        public uint     dwPwm4Value;
        public uint     dwPwm5Value;
        public uint     dwPwm6Value;
        public uint     dwPwm7Value;
        public uint     dwPwm8Value;
        public uint     dwPwm9Value;
        public uint     dwPwm10Value;
        public uint     dwPwm11Value;
        public uint     dwPwm12Value;
        public uint     dwPwm13Value;
        public uint     dwPwm14Value;
        public uint     dwPwm15Value;
        };
```

メンバ変数

dwPwm0Value	PWM チャンネル 0 のデューティー比を指定します。0-4096 が有効な値です。
dwPwm1Value	PWM チャンネル 1 のデューティー比を指定します。0-4096 が有効な値です。
dwPwm2Value	PWM チャンネル 2 のデューティー比を指定します。0-4096 が有効な値です。
dwPwm3Value	PWM チャンネル 3 のデューティー比を指定します。0-4096 が有効な値です。
dwPwm4Value	PWM チャンネル 4 のデューティー比を指定します。0-4096 が有効な値です。
dwPwm5Value	PWM チャンネル 5 のデューティー比を指定します。0-4096 が有効な値です。
dwPwm6Value	PWM チャンネル 6 のデューティー比を指定します。0-4096 が有効な値です。
dwPwm7Value	PWM チャンネル 7 のデューティー比を指定します。0-4096 が有効な値です。
dwPwm8Value	PWM チャンネル 8 のデューティー比を指定します。0-4096 が有効な値です。
dwPwm9Value	PWM チャンネル 9 のデューティー比を指定します。0-4096 が有効な値です。
dwPwm10Value	PWM チャンネル 10 のデューティー比を指定します。0-4096 が有効な値です。
dwPwm11Value	PWM チャンネル 11 のデューティー比を指定します。0-4096 が有効な値です。
dwPwm12Value	PWM チャンネル 12 のデューティー比を指定します。0-4096 が有効な値です。
dwPwm13Value	PWM チャンネル 13 のデューティー比を指定します。0-4096 が有効な値です。
dwPwm14Value	PWM チャンネル 14 のデューティー比を指定します。0-4096 が有効な値です。
dwPwm15Value	PWM チャンネル 15 のデューティー比を指定します。0-4096 が有効な値です。

TStatusPack2

システムの稼動状態を格納します。

C/C++

```
struct TStatusPack2
        {
        DWORD dwBurstMax;
        DWORD dwADO_underrun;
        DWORD dwADI_overrun;
        DWORD dwWriteAddress;
        DWORD dwBusmasterOn;
        DWORD dwDaqInit;
        DWORD dwDaqEnable;
        DWORD dwTrigSens2;
        DWORD dwTrigSens1;
        DWORD dwTrigSens0;
        DWORD dwTrigSeq;
        };
```

C#

```
public struct TStatusPack2
        {
        public uint     dwBurstMax;
        public uint     dwADO_underrun;
        public uint     dwADI_overrun;
        public uint     dwWriteAddress;
        public uint     dwBusmasterOn;
        public uint     dwDaqInit;
        public uint     dwDaqEnable;
        public uint     dwTrigSens2;
        public uint     dwTrigSens1;
        public uint     dwTrigSens0;
        public uint     dwTrigSeq;
        };
```

メンバ変数

dwADO_underrun	AO/DO バッファーアンダーフローステータス。アンダーフローは、リングバッファへの書き込みが、次のバンクチェンジに間に合わず、出力データ欠損を意味します。以下の定義された数値が格納されます。 **UNDERRUN_BUFFER** :バッファーアンダーフローの発生（エラー） 上記以外 :問題なし
dwADI_overrun	AI/DI バッファーオーバーフローステータス。オーバーフローは、リングバッファからの読み出しが、次のバンクチェンジに間に合わず、入力データ欠損を意味します。以下の定義された数値が格納されます。 **OVERRUN_BUFFER** バッファーオーバーフローの発生（エラー） 上記以外 :問題なし
dwTrigSeq	トリガの状態を以下の 0-4 で表します 0:アイドル状態 1:稼動状態 2:停止状態に遷移中 3:最終バンクの転送待ち 4:ストップトリガのデッドタイム
上記以外の変数	全て予約

SAYA_DEVICE_INFO

デバイス情報を格納します。

C/C++

```
struct  SAYA_DEVICE_INFO
        {
        DWORD           dwDeviceType;
        DWORD           dwBufferSizeOfByte;
        DWORD           dwBufferSizeOfDWORD;
        int             iDIO_TRIG_SOURCE_MAX;
        int             iAIO_TRIG_SOURCE_MAX;
        int             iTRIG_MODE_MAX;
        DWORD           dwSAMPLE_FAST;
        DWORD           dwSAMPLE_SLOW;
        int             iPRE_TRIG_MAX;
        };
```

C#

```
public struct SAYA_DEVICE_INFO
        {
        public uint     dwDeviceType;
        public uint     dwBufferSizeOfByte;
        public uint     dwBufferSizeOfDWORD;
        public int      iDIO_TRIG_SOURCE_MAX;
        public int      iAIO_TRIG_SOURCE_MAX;
        public int      iTRIG_MODE_MAX;
        public uint     dwSAMPLE_FAST;
        public uint     dwSAMPLE_SLOW;
        public int      iPRE_TRIG_MAX;
        };
```

メンバ変数

dwDeviceType	デバイスの種類が入ります（以下参照） ADXⅡ85-1M-PCIEX = 0 ADXⅡ42***、ADXⅡINF*** = 4 ADXⅡ14-125M-PCIEX = 5
dwBufferSizeOfByte	リングバッファ（プライマリオンチップリングバッファ）の Byte(8Bit)単位サイズが入ります。
dwBufferSizeOfDWORD	リングバッファ（プライマリオンチップリングバッファ）の DWORD(32Bit)単位サイズが入ります。
iDIO_TRIG_SOURCE_MAX	デジタルトリガソースの種類
iAIO_TRIG_SOURCE_MAX	アナログトリガソースの種類
iTRIG_MODE_MAX	トリガモードの種類
dwSAMPLE_FAST	構造体 TXBUFSETUP2 の dwClockScall に設定できる、最高サンプリング周波数
dwSAMPLE_SLOW	構造体 TXBUFSETUP2 の dwClockScall に設定できる、最低サンプリング周波数
iPRE_TRIG_MAX	プリトリガの種類

SAYA_DEVICE_INFO_EX

デバイス情報を格納します。SAYA_DEVICE_INFO よりも情報が多いので、デバイス依存のコードを大幅に減らす事が出来ます。

C/C++

```
struct  SAYA_DEVICE_INFO_EX
        {
        DWORD           dwDeviceType;
        DWORD           dwBufferSizeOfByte;
        DWORD           dwBufferSizeOfDWORD;
        int             iDIO_TRIG_SOURCE_MAX;
        int             iAIO_TRIG_SOURCE_MAX;
        int             iTRIG_MODE_MAX;
        DWORD           dwSAMPLE_FAST;
        DWORD           dwSAMPLE_SLOW;
        int             iPRE_TRIG_MAX;

        DWORD           dwSAMPLE_SLOW2;
        DWORD           dwTempSensor;
        DWORD           dwOnboardCAL;
        DWORD           dwCounterToRingbuf;
        DWORD           dwClockOut;
        DWORD           dwPreTrigSize;
        DWORD           dwBankSize;
        DWORD           dwBankSizeUnit;
        DWORD           dwClockIn;
        DWORD           dwChseqMax;
        DWORD           dwChseqMin;
        DWORD           dwCyclicTrig;
        DWORD           dwChseqDfType;
        DWORD           dwCoreAirange;
        DWORD           dwDoMap;
        DWORD           dwAiMap;
        DWORD           dwDiMap;
        DWORD           dwAoMap;
        DWORD           dwCounterMap;
        DWORD           dwFreqCounterMap;
        DWORD           dwPwmMap;
        DWORD           dwProgramableScp;
        DWORD           dwAdaptiveFreqUnit;
        double          dAoLsbLevel;
        double          dAoLsbLevelEx;
        double          dAoLsbOffset;
        double          dAoLsbOffsetEx;
        DWORD           dwDI[16];
        DWORD           dwDO[16];
        DWORD           dwBufferSizeOfBytesr;
        DWORD           dwBufferSizeOfDWORDsr;
        DWORD           dwPreWriteSizeOfByte;
        DWORD           dwPreWriteSizeOfDWORD;
        DWORD           dwAdcStyle;
        DWORD           dwAdoBufMode;
        };
```

C#

```
public struct SAYA_DEVICE_INFO_EX
        {
        public uint         dwDeviceType;
        public uint         dwBufferSizeOfByte;
        public uint         dwBufferSizeOfDWORD;
        public int          iDIO_TRIG_SOURCE_MAX;
        public int          iAIO_TRIG_SOURCE_MAX;
        public int          iTRIG_MODE_MAX;
        public uint         dwSAMPLE_FAST;
        public uint         dwSAMPLE_SLOW;
        public int          iPRE_TRIG_MAX;
        public uint         dwSAMPLE_SLOW2;
        public uint         dwTempSensor;
        public uint         dwOnboardCAL;
        public uint         dwCounterToRingbuf;
        public uint         dwClockOut;
        public uint         dwPreTrigSize;
        public uint         dwBankSize;
        public uint         dwBankSizeUnit;
        public uint         dwClockIn;
        public uint         dwChseqMax;
        public uint         dwChseqMin;
        public uint         dwCyclicTrig;
        public uint         dwChseqDfType;
        public uint         dwCoreAirange;
        public uint         dwDoMap;
        public uint         dwAiMap;
        public uint         dwDiMap;
        public uint         dwAoMap;
        public uint         dwCounterMap;
        public uint         dwFreqCounterMap;
        public uint         dwPwmMap;
        public uint         dwProgramableScp;
        public uint         dwAdaptiveFreqUnit;
        public double       dAoLsbLevel;
        public double       dAoLsbLevelEx;
        public double       dAoLsbOffset;
        public double       dAoLsbOffsetEx;
        public uint         dwDI[16];
        public uint         dwDO[16];
        public uint         dwBufferSizeOfBytesr;
        public uint         dwBufferSizeOfDWORDsr;
        public uint         dwPreWriteSizeOfByte;
        public uint         dwPreWriteSizeOfDWORD;
        public uint         dwAdcStyle;
        public uint         dwAdoBufMode;
        };
```

メンバ変数

dwDeviceType	デバイスの種類が入ります（以下参照） ADXⅡ85-1M-PCIEX　= 0 ADXⅡ42***、 ADXⅡINF***　= 4 ADXⅡ14-125M-PCIEX　= 5
dwBufferSizeOfByte	リングバッファ（プライマリオンチップリングバッファ）の Byte(8Bit)単位サイズが入ります。本変数の値は ADXⅡ85-1M-PCIEX においてソフトウェアで処理すべきデータ量メモリ量ではないので、使用をお勧めできません。dwBufferSizeOfBytesr を使ってください。
dwBufferSizeOfDWORD	リングバッファ（プライマリオンチップリングバッファ）の DWORD(32Bit)単位サイズが入ります。本変数の値は ADXⅡ85-1M-PCIEX においてソフトウェアで処理すべきデータ量メモリ量ではないので、使用をお勧めできません。dwBufferSizeOfDWORDsr を使ってください。
dwBufferSizeOfBytesr	リングバッファ（セカンダリリングバッファ）の Byte(8Bit)単位サイズが入ります。 本変数の値は確保すべきメモリ量、データ量を表しています。
dwBufferSizeOfDWORDsr	リングバッファ（セカンダリリングバッファ）の DWORD(32Bit)単位サイズが入ります。 本変数の値は確保すべきメモリ量、データ量を表しています。
dwPreWriteSizeOfByte	AO/DO 側リングバッファへのプリライトサイズを Byte(8Bit)単位で返します。
dwPreWriteSizeOfDWORD	AO/DO 側リングバッファへのプリライトサイズを DWORD(32Bit)単位で返します。
iDIO_TRIG_SOURCE_MAX	デジタルトリガソースの種類
iAIO_TRIG_SOURCE_MAX	アナログトリガソースの種類
iTRIG_MODE_MAX	トリガモードの種類
dwSAMPLE_FAST	構造体 TXBUFSETUP2 の dwClockScall に設定できる、最高サンプリング周波数
dwSAMPLE_SLOW	構造体 TXBUFSETUP2 の dwClockScall に設定できる、最低サンプリング周波数
iPRE_TRIG_MAX	プリトリガの種類
dwSAMPLE_SLOW2	dwSAMPLE_SLOW ではサンプリング周波数の調整範囲が広くなりすぎるので 現実的な最低サンプリング周波数を定義
dwTempSensor	基板上の温度センサの数。
dwOnboardCAL	オンボードキャリブレーターを実装している場合 1、実装していない場合 0 です。 この値が 1 であれば、IOGEOSETUP.dwInputShort により校正電圧を、アナログ入力の信号源とすることが可能です。
dwCounterToRingbuf	カウンタのリングバッファ接続が可能であれば 1、不可能なら 0。
dwClockOut	クロックアウトを装備していれば 1、していなければ 0。
dwPreTrigSize	プリトリガのサイズをサンプル数で返します。（Byte ではないので注意）
dwBankSize	ハードウェアリングバッファの 1 バンクあたりのサイズを返します。 (ADXⅡ14-125M-PCIEX のみメガバイト単位、他機種はサンプル数です)
dwBankSizeUnit	上記単位(0 ならサンプル数,1 ならメガバイト)
dwClockIn	クロックインプットを装備していれば 1、していなければ 0。
dwChseqMax	TBUFSETUP.dwMuxSeqenceAuto(チャンネルシーケンス)の最大値
dwChseqMin	TBUFSETUP.dwMuxSeqenceAuto(チャンネルシーケンス)の最小値
dwCyclicTrig	サイクリックトリガを装備していれば 1、していなければ 0。
dwChseqDfType	シーケンシャル取り込み off 時のデジタルフィルタの構成。 (0=移動平均，1=デジタルフィルタオフと同じ FIR 型，2=未実装)
dwCoreAirange	信号調節を考慮しない場合のコアアンプの入力レンジ。ADXⅡ42***ではプログラマブル信号調節アンプを取り去った状態の入力レンジを、IOGEOSETUP.dwAI_Range と同等の規格で返します。0xFF を返す場合には信号調節未対応です。
dwDoMap	デジタル出力の実装数。
dwAiMap	アナログ入力の実装数。（シングルエンドの場合）
dwDiMap	デジタル入力の実装数。
dwAoMap	アナログ出力の実装数。
dwCounterMap	カウンタの実装数。
dwFreqCounterMap	周波数カウンタの実装数。
dwPwmMap	PWM の実装数。
dwProgramableScp	プログラマブル信号調節を装備していれば 1、していなければ 0。
dwAdaptiveFreqUnit	最適なサンプリング周波数表示単位 （0:Hz ， 1:KHz ， 2:MHz）
dwAdcStyle	A/D コンバータの実装方法 （0:マルチプレクス方式 ，1:同時サンプリング方式）
dAoLsbLevel	アナログ出力 CH0～3（固定電圧出力）の 1LSB あたりの電圧(mV)
dAoLsbLevelEx	アナログ出力 CH4～（可変電圧出力）の 1LSB あたりの電圧(mV)
dAoLsbOffset	アナログ出力 CH0～3（固定電圧出力）のオフセット電圧(mV) アナログ出力値は （D/A 設定値×dAoLsbLevel ＋ dAoLsbOffset） です。
dAoLsbOffsetEx	アナログ出力 CH4～（可変電圧出力）のオフセット電圧(mV) アナログ出力値は （D/A 設定値×dAoLsbLevelEx ＋ dAoLsbOffsetEx） です。
dwAdoBufMode	リングバッファを出力専用、DI をカットオフすることができるか(1:yes,0:no) 現状は、ADXⅡ14-125M-PCIEX のみ 1 になります。
dwDI[16]	インテリジェント DI0-15（カウンタや DI 割り込み）と、実際の DI の割付(0xFF なら未搭載)。 DI[0]=16 なら、DI16 に DI 割り込み 0 やカウンタの A 相が実装されていることになります。
dwDO[16]	インテリジェント DO0-15（PWM やコンペア出力）と、実際の DO の割付(0xFF なら未搭載)。 DO[0]=16 なら、DO16 に PWM0 が実装されていることになります。

ADIOX_EXTENTION2

AI/DI データ⇒ファイル保存、ファイル読み出し⇒AO/DO 出力、波形ジェネレータの主要機能を格納します。

C/C++

```
struct ADIOX_EXTENTION2
        {
        char            lpcAdiFileName[256];
        char            lpcDmyFileName[256];
        BOOL            bDoubleSave;
        DWORD           dwADI_style;
        int             iAO_Gain0;
        int             iAO_Gain1;
        int             iAO_Offset0;
        int             iAO_Offset1;
        int             iAO_SamplePerCycle0;
        int             iAO_SamplePerCycle1;
        DWORD           dwADO_style0;
        DWORD           dwADO_style1;
        char            lpcAdoFileName[256];
        DWORD           dwReserverd[16];
        };
```

C#

```
public struct ADIOX_EXTENTION2
        {
        [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 256)]
        public string lpcAdiFileName;
        [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 256)]
        public string lpcDmyFileName;
        public int              bDoubleSave;
        public uint             dwADI_style;
        public int              iAO_Gain0;
        public int              iAO_Gain1;
        public int              iAO_Offset0;
        public int              iAO_Offset1;
        public int              iAO_SamplePerCycle0;
        public int              iAO_SamplePerCycle1;
        public uint             dwADO_style0;
        public uint             dwADO_style1;
        [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 256)]
        public string lpcAdoFileName;
        public fixed uint dwReserverd[16];
        };
```

メンバ変数

bDoubleSave	ADXⅡ42***, ADXⅡINF***でアナログ入力を double 型で保存する場合 TRUE(=1)、DWORD で保存する場合 FALSE(=0)を指定します。
iAO_Gain0	波形ジェネレータ AO0 ゲイン。0～100 を指定します。
iAO_Gain1	波形ジェネレータ AO1 ゲイン。0～100 を指定します。(ADXⅡ14-125M-PCIEX のみ有効)
iAO_Offset0	波形ジェネレータ AO0 オフセット。±10000 を指定します。
iAO_Offset1	波形ジェネレータ AO1 オフセット。±10000 を指定します。(ADXⅡ14-125M-PCIEX のみ有効)
dwADO_style0	波形ジェネレータ AO0 波形。以下のいずれかを指定します。
dwADO_style1	波形ジェネレータ AO0 波形。以下のいずれかを指定します。 (ADXⅡ14-125M-PCIEX のみ有効) ADX_Sin　サイン ADX_Cos　コサイン ADX_Exp　Exp(2n) ADX_Sqrt　Sqrt(2π) Sqrt は平方根 ADX_Triangle　三角波 ADX_Lamp　ランプ波形（のこぎり波形） ADX_Square　方形波 ADX_DC0　0x8000 連続 DC 出力(ゼロオフセット校正用) ADX_DC1　0xE000 連続 DC 出力(ゲイン校正用) ADX_File　ファイル出力 (AO0,AO1 のいずれかがファイル出力なら両方ファイル出力になります) ADX_LargeStep　ステップ（1 リングバッファ単位で 0x1000 ずつインクリメントする）
iAO_SamplePerCycle0	1 サイクルあたりのサンプル数。サンプリング周波数÷iAO_SamplePerCycle0 が波形生成周波数になります。
iAO_SamplePerCycle1	1 サイクルあたりのサンプル数。サンプリング周波数÷iAO_SamplePerCycle1 が波形生成周波数になります。(ADXⅡ14-125M-PCIEX のみ有効)
dwADI_style	アナログ入力形式 NO_SAVE　ファイル非保存 DIRECT_FILE　ファイル保存
lpcAdiFileName	AI/DI 値保存ファイル名(dwADI_style で DIRECT_FILE を指定した場合必須)
lpcDmyFileName	ダミーファイル保存ファイル名(dwADI_style で DIRECT_FILE を指定した場合必須※)
lpcAdoFileName	AO/DO 出力ファイル名 (dwADO_style0,dwADO_style1 で ADX_File を指定した場合必須)
dwReserverd	予備

※ダミーファイルは高速データ収集では必須です。計測データをリアルタイムに書き込もうとしてもハードディスクは回転速度を上げるには時間がかかり、その待ち時間でバッファオーバランが生じてしまいます。（ちなみに非同期書き込みでも書き込み＝負荷に偏りがあり、負荷の高い時期にやはりオーバランする） ダミーファイルはデータ収集開始前に、lpcAdiFileName のドライブにダミーファイルを書き込むことで、回転速度を上げ、その直後からデータ収集⇒ファイル書き込むことでバッファオーバランを回避します。ゆえにダミーファイルは lpcAdiFileName と同じドライブにしてください。またスタートトリガが有効になるまで時間がかかるとこのダミーファイルの効果は薄れますので注意が必要です。

SCP_SETUP2

複数の、ADX Ⅱ 42***, ADX Ⅱ INF***の各種設定用構造体を格納する構造体です。この構造体のメンバには TXBUFSETUP2、IOGEOSETUP2、TDIO_MISC、TConfigPWM、TSetupPWM、信号調節変数としてセンサーモード・ゼロスパン校正位置・ゼロスパン校正係数・スケーリング係数・アラーム設定をチャンネル数×最大 24 台分（CARD_ID4～27）保持します。SCP_SETUP2 構造体はドライバ内部のデータベースに相当し、運用中 TXBUFSETUP2、IOGEOSETUP2、TDIO_MISC、TConfigPWM、TSetupPWM などの構造体を使用する関数をアクセスした場合、たとえ SCP_SETUP2 をアクセスしていなくとも、ドライバ内部で SCP_SETUP2 構造体に変更内容が反映されます。

C/C++

```
struct SCP_SETUP2
        {
        BOOL            bMultifunctionIO_Enable[MAX_MFIO];
        TXBUFSETUP2     sTXBUFSETUP[MAX_MFIO];
        IOGEOSETUP2     sIOGEOSETUP[MAX_MFIO];
        TDIO_MISC       sTDIO_MISC[MAX_MFIO];
        TConfigPWM      sTConfigPWM[MAX_MFIO];
        TSetupPWM       sTSetupPWM[MAX_MFIO];
        DWORD           dwLDO1[MAX_MFIO];
        DWORD           dwLDO2[MAX_MFIO];
        DWORD           dwSensorMode[MAX_MFIO][MAX_AI_CH];
        double          doZeroPos[MAX_MFIO][MAX_AI_CH];
        double          doSpanPos[MAX_MFIO][MAX_AI_CH];
        double          doZero_Coefficient[MAX_MFIO][MAX_AI_CH];
        double          doSpan_Coefficient[MAX_MFIO][MAX_AI_CH];
        BOOL            bScalling[MAX_MFIO][MAX_AI_CH];
        double          dScallingRatio[MAX_MFIO][MAX_AI_CH];
        double          dOutTopScall[MAX_MFIO][MAX_AI_CH];
        double          dOutBottomScall[MAX_MFIO][MAX_AI_CH];
        double          dInTopScall[MAX_MFIO][MAX_AI_CH];
        double          dInBottomScall[MAX_MFIO][MAX_AI_CH];
        DWORD           bAlarmMode[MAX_MFIO][MAX_AI_CH];
        double          dAlarmUpper[MAX_MFIO][MAX_AI_CH];
        double          dAlarmLower[MAX_MFIO][MAX_AI_CH];
        };
```

C#

```
public struct SCP_SETUP2
        {
        public int              [] bMultifunctionIO_Enable;
        public TXBUFSETUP2      [] sTXBUFSETUP;
        public IOGEOSETUP2      [] sIOGEOSETUP;
        public TDIO_MISC        [] sTDIO_MISC;
        public TConfigPWM       [] sTConfigPWM;
        public TSetupPWM        [] sTSetupPWM;
        public uint             [] dwLDO1;
        public uint             [] dwLDO2;
        public uint             [,] dwSensorMode;
        public double           [,] doZeroPos;
        public double           [,] doSpanPos;
        public double           [,] doZero_Coefficient;
        public double           [,] doSpan_Coefficient;
        public int              [,] bScalling;
        public double           [,] dScallingRatio;
        public double           [,] dOutTopScall;
        public double           [,] dOutBottomScall;
        public double           [,] dInTopScall;
        public double           [,] dInBottomScall;
        public uint             [,] bAlarmMode;
        public double           [,] dAlarmUpper;
        public double           [,] dAlarmLower;
        };
```

メンバ変数

bMultifunctionIO_Enable	MultifunctionIO を有効にする場合 TRUE(=1)、無効にするには FALSE(=0)にしてください。
sTXBUFSETUP	TXBUFSETUP2 構造体を MAX_MFIO 個格納します。
sIOGEOSETUP	IOGEOSETUP2 構造体を MAX_MFIO 個格納します。
sTDIO_MISC	TDIO_MISC 構造体を MAX_MFIO 個格納します。
sTConfigPWM	TConfigPWM 構造体を MAX_MFIO 個格納します。
sTSetupPWM	TSetupPWM 構造体を MAX_MFIO 個格納します。
dwLDO1	シグナルコンディションコントロールレジスタ設定値 1 を MAX_MFIO 個格納します。 (アプリケーションでは参照はできても変更してはなりません)
dwLDO2	シグナルコンディションコントロールレジスタ設定値 2 を MAX_MFIO 個格納します。 (アプリケーションでは参照はできても変更してはなりません)
doZeroPos	ゼロ校正位置を”MAX_MFIO×MAX_AI_CH”個格納します。
doSpanPos	スパン校正位置を”MAX_MFIO×MAX_AI_CH”個格納します。
doZero_Coefficient	ゼロ校正係数を”MAX_MFIO×MAX_AI_CH”個格納します。
doSpan_Coefficient	スパン校正係数を”MAX_MFIO×MAX_AI_CH”個格納します。
bScalling	スケーリングする場合、TRUE(=1)しない場合 FALSE(=0)をセットします。 これを”MAX_MFIO×MAX_AI_CH”個格納します。
dScallingRatio	スケーリング係数を”MAX_MFIO×MAX_AI_CH”個格納します。 (アプリケーションでは参照はできても変更してはなりません)
dOutTopScall	変換後のスケーリング基準値(上)。これを”MAX_MFIO×MAX_AI_CH”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_MFIO×MAX_AI_CH”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_MFIO×MAX_AI_CH”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_MFIO×MAX_AI_CH”個格納します。
bAlarmModeA	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_MFIO×MAX_AI_CH”個格納します。
dIarmUpper	アラーム設定値(上)を”MAX_MFIO×MAX_AI_CH”個格納します。
dAlarmLower	アラーム設定値(下)を”MAX_MFIO×MAX_AI_CH”個格納します。

dwSensorMode　　カード ID、AI チャンネル毎にターゲットのセンサー番号を指定します。センサーモードは以下のように定義されています。

設定可能な種類は以下の通りです。

AI0-7 に設定可能な信号源

定義	値	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
CA_J	2	熱電対 J
CA_E	3	熱電対 E
CA_T	4	熱電対 T
CA_R	5	熱電対 R
CA_S	6	熱電対 S
CA_N	7	熱電対 N
CA_B	8	熱電対 B
PT100	9	白金測温抵抗体 Pt100
JPT100	10	白金測温抵抗体 JPt100
VBP_10mV	11	電圧±10mV レンジ
VBP_100mV	12	電圧±100mV レンジ
VBP_1V	13	電圧±1V レンジ
VBP_10V	17	電圧±10V レンジ
I_4_20	22	電流 4-20mA/500Ω終端
I_4_20EX	23	電流 4-20mA/350Ω終端
I_4_20EX2	26	電流 4-20mA/47Ω終端 ↑ オンボードの終端
D16BIT	27	0-65535 のデジタル値のこと
RION_DC80	28	リオン騒音計振動計 80dB レンジ
RION_DC90	29	リオン騒音計振動計 90dB レンジ
RION_DC100	30	リオン騒音計振動計 100dB レンジ
RION_DC110	31	リオン騒音計振動計 110dB レンジ
RION_DC120	32	リオン騒音計振動計 120dB レンジ
RION_DC130	33	リオン騒音計振動計 130dB レンジ

※ リオン騒音計振動計はいずれも DC2.5V 出力用

DUST10000	34	柴田粉塵計 10000cpm (1V レンジ)
DUST1000	35	柴田粉塵計 10000cpm (1V レンジ)
MT321	36	マザーツール騒音計 (DC 接続
AKW4802C_100OHM	37	松下カレントトランス AKW4802C 100Ω
AKW4803_4804C_10OHM	38	松下カレントトランス AKW4804C 100Ω
AKW4808C_10OHM	39	松下カレントトランス AKW4808C 10Ω

※　松下カレントトランスは予約、現在のプラットフォームでは対応していません

以下はインフラサウンドセンサーADXⅡ-INF01 内蔵機能

VIB	100	ADXⅡ-INF01 内蔵加速度計 XYZ（地震）
SPL	101	ADXⅡ-INF01 内蔵騒音計(Z 特性)
APL	102	ADXⅡ-INF01 内蔵気圧計

AI8-11(実際にはカウンタ)に設定可能な信号源

定義	値	内訳
EC_4X	40	4 倍速カウンタ Z 未使用
EC_4XZ	41	4 倍速カウンタ Z 使用
EC_2X	42	2 倍速カウンタ Z 未使用
EC_2XZ	43	2 倍速カウンタ Z 使用
EC_1X	44	1 倍速カウンタ Z 未使用
EC_1XZ	45	1 倍速カウンタ Z 使用
UPC	46	アップダウンカウンタ Z 未使用
UPC_Z	47	アップダウンカウンタ Z 使用
WIND	48	風速計

以下はインフラサウンドセンサーADXⅡ-INF01 内蔵機能

INFRS_TA	52	ADXⅡ-INF01 内蔵温度計
INFRS_FB	50	ADXⅡ-INF01 インフラサウンド DC
INFRS_DIF	55	ADXⅡ-INF01 インフラサウンド AC

※　配列番号 MAX_MFIO はターゲットデバイスのカード ID を示します。

※　配列番号 MAX_AI_CH はアナログ入力チャンネルを表します。ADXⅡ42***ではアナログ入力は AI0-11 が有効で 8-11 はカウンタに相当します。カウンタでもアラームやスケーリングを使うことが出来ます。

SCP_SETUP_AICH

ADXⅡ42***, ADXⅡINF***の信号調節関連の設定を格納します。SCP_SETUP2 構造体ではグループ全体の設定が格納できますが、本構造体はターゲットデバイス 1 個分のメンバ変数のみを変更できるのでセキュリティが高まります。構造体定義の配列番号 MAX_AI_CH はターゲットの入力チャンネル(AI0-11 8-11 はカウンタ)を示します。

C/C++

```
struct SCP_SETUP_AICH
        {
        DWORD           dwSensorMode[MAX_AI_CH];
        DWORD           dwLDO[MAX_AI_CH];
        double          doZeroPos[MAX_AI_CH];
        double          doSpanPos[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];
        };
```

C#

```
public struct SCP_SETUP_AICH
        {
        public fixed uint           dwSensorMode[MAX_AI_CH];
        public fixed uint           dwLDO[MAX_AI_CH];
        public fixed double         doZeroPos[MAX_AI_CH];
        public fixed double         doSpanPos[MAX_AI_CH];
        public fixed int            bScalling[MAX_AI_CH];
        public fixed double         dOutTopScall[MAX_AI_CH];
        public fixed double         dOutBottomScall[MAX_AI_CH];
        public fixed double         dInTopScall[MAX_AI_CH];
        public fixed double         dInBottomScall[MAX_AI_CH];
        public fixed uint           bAlarmMode[MAX_AI_CH];
        public fixed double         dAlarmUpper[MAX_AI_CH];
        public fixed double         dAlarmLower[MAX_AI_CH];
        };
```

メンバ変数

dwSensorMode	AI チャンネル毎にターゲットのセンサー番号を指定します。
dwLDO	センサーモード NOT_USE と組み合わせてシグナルコンディションコントロールレジスタ設定値を強制ロードさせるときに使います。通常は使われません。
doZeroPos	ゼロ校正位置を MAX_AI_CH 個格納します。
doSpanPos	スパン校正位置を MAX_AI_CH 個格納します。
bScalling	スケーリングする場合 TRUE(=1)、しない場合 FALSE(=0)をセットします。これを MAX_AI_CH 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 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 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 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 個格納します。
bAlarmMod	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 個格納します。
dAlarmUpper	アラーム設定値(上)。これを MAX_AI_CH 個格納します。
dAlarmLower	アラーム設定値(下)。これを MAX_AI_CH 個格納します。

SCP_SETUP_AIALL

ADXⅡ42***, ADXⅡINF***用です。SCP_SETUP_AICH に、校正係数の doZero_Coefficient と、doSpan_Coefficient を加えたもの。ドライバ内部の SCP_SETUP に対して強制的に校正係数を与えることができる。

C/C++

```
struct SCP_SETUP_AICH
        {
        DWORD       dwSensorMode[MAX_AI_CH];
        DWORD       dwLDO[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];
        };
```

C#

```
public struct SCP_SETUP_AIALL
        {
        public fixed uint       dwSensorMode[MAX_AI_CH];
        public fixed uint       dwLDO[MAX_AI_CH];
        public fixed double     doZeroPos[MAX_AI_CH];
        public fixed double     doSpanPos[MAX_AI_CH];
        public fixed double     doZero_Coefficient[MAX_AI_CH];
        public fixed double     doSpan_Coefficient[MAX_AI_CH];
        public fixed int        bScalling[MAX_AI_CH];
        public fixed double     dOutTopScall[MAX_AI_CH];
        public fixed double     dOutBottomScall[MAX_AI_CH];
        public fixed double     dInTopScall[MAX_AI_CH];
        public fixed double     dInBottomScall[MAX_AI_CH];
        public fixed uint       bAlarmMode[MAX_AI_CH];
        public fixed double     dAlarmUpper[MAX_AI_CH];
        public fixed double     dAlarmLower[MAX_AI_CH];
        };
```

メンバ変数

doZeroPos[MAX_AI_CH]	ゼロ校正係数。vADioxScpCopy2 関数などから取得する必要があります。
doSpanPos[MAX_AI_CH]	スパン校正係数。vADioxScpCopy2 関数などから取得する必要があります。

上記以外のメンバ変数は SCP_SETUP_AICH を参照願います。

CharPayloadC

設定ファイル名文字列、ファイルを開くダイアログボックスのデフォルトフォルダ名文字列を格納しています。

C/C++

```
struct CharPayloadC
        {
        char    lpcInitialDir[256];
        char    lpcConfigFileName[256];
        };
```

C#

```
public struct CharPayloadC
        {
        [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 256)]
        public string lpcInitialDir;
        [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 256)]
        public string lpcConfigFileName;
        };
```

メンバ変数

lpcInitialDir	ファイルを開くダイアログボックスのデフォルトフォルダ名
lpcConfigFileName	設定ファイル名

LOG_FRONTEND

ADiox2.dll の保存データ(adi ファイル)は、先頭にこのヘッダが付加され、その後リングバッファイメージをダイレクトに書き込んでいきます。double 型保存では double 型の AI チャンネルデータ(バッファサイズ分)、その後にリングバッファデータ(バッファサイズ分)が繰り返し保存されます。

C/C++

```
struct LOG_FRONTEND
        {
        DWORD  dwHeaderCode;
        DWORD  dwDeviceName;
        DWORD  dwBuffaSize;
        double   dClockScall;
        BYTE     bBitScall;
        BYTE     bAI_ChannelScall;
        BYTE     bDI_ChannelScall;
        BYTE     DataType;
        DWORD  dwGetYear;
        DWORD  dwGetMonth;
        DWORD  dwGetDay;
        DWORD  dwGetHour;
        DWORD  dwGetMinute;
        DWORD  dwGetSecond;
        DWORD  dwGetMilliseconds;
        DWORD  dwInrange;
        DWORD  dwReserved;
        DWORD  dwReserved;
        };
```

C#

```
public struct LOG_FRONTEND
        {
        public uint         dwHeaderCode;
        public uint         dwDeviceName;
        public uint         dwBuffaSize;
        public double       dClockScall;
        public byte         bBitScall;
        public byte         bAI_ChannelScall;
        public byte         bDI_ChannelScall;
        public byte         DataType;
        public uint         dwGetYear;
        public uint         dwGetMonth;
        public uint         dwGetDay;
        public uint         dwGetHour;
        public uint         dwGetMinute;
        public uint         dwGetSecond;
        public uint         dwGetMilliseconds;
        public uint         dwInrange;
        public uint         dwReserved1;
        public uint         dwReserved2;
        };
```

メンバ変数

dwHeaderCode	ファイルヘッダ識別コード、必ず 0x41594154 をセットしてください。
dwDeviceName	機種コード(SAYA_DEVICE_INFO.dwDeviceType)
dwBuffaSize	リングバッファサイズ (ADXⅡ85-1M-PCIEX ではセカンダリリングバッファサイズ)
dClockScall	サンプリング周波数(Hz)
bBitScall	量子化ビット数
bAI_ChannelScall AI	チャンネル数
bDI_ChannelScall DI	チャンネル数
DataType	保存形式(DWORD=0,double=1)　double 保存は ADXⅡ42***, ADXⅡINF***のみ
dwGetYear	計測開始の年
dwGetMonth	計測開始の月
dwGetDay	計測開始の日
dwGetHour	計測開始の時
dwGetMinute	計測開始の分
dwGetSecond	計測開始の秒
dwGetMilliseconds	計測開始のミリ秒
dwInrange	入力レンジ (ADXⅡ85-1M-PCIEX は AI レンジ= IOGEOSETUP2. dwAI_Range) (ADXⅡ14-125M-PCIEX は 1 でユニポーラ／0 でバイポーラ)
dwReserved1;	予備
dwReserved2;	予備

13.構造体 2 [ADioxLogm.dllM]

これらの構造体は全て ADXⅡ42***, ADXⅡ-INF 用です。

ADIOX_IO_NAME

Multi device data logger 向け。CSV ログの名称を 1 チャンネル分格納します。本構造体は上位の構造体のメンバになります。本構造体を単独で使用することはありません。ADXⅡ42***, ADXⅡINF***用です。

C/C++

```
struct ADIOX_IO_NAME
        {
        char    csStr[50];
        };
```

メンバ変数 csStr[50]　　CSV ログの名称を 1 チャンネル分格納します。

ADIOX_CSV_NAME

Multi device data logger 向け。CSV ログの名称を MAX_AI_CH(32)チャンネル分格納します。本構造体は上位の構造体のメンバになります。本構造体を単独で使用することはありません。ADXⅡ42***, ADXⅡINF***用です。

C/C++

```
struct ADIOX_CSV_NAME
        {
        ADIOX_IO_NAME           sCH[MAX_AI_CH];
        };
```

メンバ変数 sCH[MAX_AI_CH]　　CSV ログの名称を MAX_AI_CH(32)チャンネル分格納します。

ADIOX_CSV_FORMAT_EX

CSV ロガーの設定を行う構造体です。ADXⅡ42***, ADXⅡINF***用です。

C/C++

```
struct  ADIOX_CSV_FORMAT_EX
        {
        BOOL    bMultifunctionIO_Enable[MAX_MFIO];
        double  dClockScall;
        BYTE    bAI_ChannelScall;
        BYTE    bDI_ChannelScall;
        char    *FolderName;
        char    *unit_Title;
        BOOL    bCsvLogEnable[MAX_AI_CH][MAX_MFIO];
        ADIOX_CSV_NAME          sDIOX_CSV_NAME[MAX_MFIO];
        };
```

メンバ変数

bMultifunctionIO_Enable	複数の ADXⅡ42***, ADXⅡINF***のログを取る場合、どの CARD_ID の ADXⅡ42***, ADXⅡINF***を使うか本変数で指定します。本変数の配列番号は CARD_ID を表し、TRUE なら、その CARD_ID はログの対象です。FALSE であれば CARD_ID はログの対象になりません。
dClockScall	サンプリング時間(sec)を指定します。
bAI_ChannelScall	記録するチャンネル数を指定します。8ch のアナログ入力信号と 4ch のカウンタ信号をあわせて 12ch までを記録できるようになっており、これを 1～12 で任意の数値を設定します。 bADioxLoggerWrite で書き込むバッファは、先頭から本変数で指定した AI チャンネル数分のデータが順列に配置してください。AI チャンネルという名称ですが、カウンタなども、ここに入れることが出来、カウンタは AI8-11 に相当します。ます。
bDI_ChannelScall	DI チャンネル数を指定します。
FolderName	CSV ログファイルの保存場所をフルパスで指定して下さい。(例)"C:¥¥MFIO_Files"
unit_Title	見出し、不要な場合は NULL を指定してください。(例)"SAYA MFIO 計測 DATA"
bCsvLogEnable	CSV ログの対象になるアナログ入力チャンネル（含：カウンタ）を指定します。本変数の配列番号はチャンネル番号及び、CARD_ID を表します。TRUE であれば、その CARD_ID におけるチャンネル番号は CSV ログの対象になります。FALSE であれば、その CARD_ID におけるチャンネル番号 CARD_ID はログの対象になりません。
sDIOX_CSV_NAME	CSV ログの各チャンネルの名称を、各 CARD_ID×チャンネル数分格納します。名称自体は最下層の構造体 ADIOX_IO_NAME に格納され、これが MAX_AI_CH チャンネル分集合する事で、構造体 ADIOX_CSV_NAME になり、更にそれが MAX_MFIO だけ集合して本変数になります。