



ADIOX2-API

HardwareAccess

Storage/Logger

SignalCondition

SignalAnalysis

Graphics/Trend

WaveGenerate

# ADIOX-MK II

MULTIFUNCTION-I/O-X2 SERIES

ADIOX2-API

REFERENCE ADX2-42

UPDATE 2011-10-30

SAYA INC.

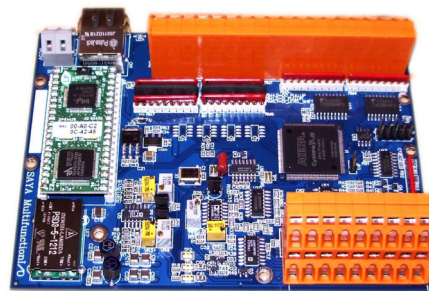
# 目次

はじめに	2
1. 初期化・再初期化・終了処理	3
2. 割り込み	6
3. リングバッファ	7
4. 設定	8
5. ステータス	10
6. ポーリング	10
7. 校正	12
8. ヘルパ	13
9. 波形生成	14
10. 構造体 1	15

## はじめに

### このマニュアルについて

このマニュアルは ADiox2-API の中で **ADX II 42-1K-Ethernet** 用のソフトウェアを作成するのに必要なものを最低限集めたものです。解説を容易にするため、信号解析 (FFT)、高速画面描画アシストなどのプレゼンス機能は、このマニュアルには記載されていません。これらの機能をはじめ、全ての API の解説は、ADIOX2-API REFERENCE (adiox2api\_all.pdf) を参照してください。



### 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。

#### 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。

#### VisualBasic 用

“CDROM¥MFIO\_X2¥sdk¥VisualBasic”にあるファイルをお使いください。

ADIOX-API\_VB.bas/[ADiox2.dll+ ADioxScp.dll ]  
メインライブラリの宣言をまとめたファイルです。(VisualBASIC プロジェクトに追加してください、.NET 用ではありません)

### CardId について

CardID=4~27 が **ADX II 42-1K-Ethernet** に割り当てられます。

# 1.初期化・再初期化

## vSetupTcpIp

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 );

**VB** Declare Sub vSetupTcpIp Lib "ADiox2.dll" ( \_  
ByVal IP1 As Integer, ByVal IP2 As Integer, ByVal IP3 As Integer, ByVal IP4 As Integer, \_  
ByVal Port As Integer, ByVal bCardID As Byte )

<b>引数</b>	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 );

**VB** Declare Function bADioxOpen2 Lib "ADiox2.dll" ( ByVal hWnd As Long, \_  
ByVal bHwintr As Byte, ByVal bCardID As Byte ) As Long

<b>引数</b>	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。

## 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。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 \*IpsSAYA\_DEVICE\_INFO,  
    TXBUFSETUP2 \*IpsTXBUFSETUP,  
    IOGEOSETUP2 \*IpsIOGEOSETUP,  
    TDIO\_MISC \*IpsTDIO\_MISC,  
    TADIO2 \*IpsTADIO,  
    TConfigPWM \*IpsTConfigPWM,  
    TSetupPWM \*IpsTSetupPWM,  
    SCP\_SETUP\_AIALL \*IpsScpSetupAich,  
    ADIOX\_EXTENTION2 \*IpsEXTENTION,  
    DWORD dwRingbufferSize,  
    BYTE CARD\_ID,  
    CharPayloadC \*IpsCharPayload  
);

```

C#    int bADioxLoad_EX3
        (
            int hWnd,
            byte bWintr,
            int bOpenDialog,
            ref SAYA_DEVICE_INFO IpsSAYA_DEVICE_INFO,
            ref TXBUFSETUP2 IpsTBUFSETUP,
            ref IOGEOSETUP2 IpsIOGEOSETUP,
            ref TDIO_MISC IpsTDIO_MISC,
            ref TADIO2 IpsTADIO,
            ref TConfigPWM IpsTConfigPWM,
            ref TSetupPWM IpsTSetupPWM,
            ref SCP_SETUP_AIALL IpsScpSetupAich,
            ref ADIOX_EXTENTION2 IpsEXTENTION,
            uint dwRingbufferSize,
            byte bCARD_ID,
            ref CharPayloadC IpsCharPayload
        );
    
```

```

VB    Declare Function bADioxLoad_EX3B Lib "ADiox2.dll" _
        ( _
            ByVal hWnd As Long, _
            ByVal bWintr As Byte, _
            ByVal bOpenDialog As Long, _
            ByRef IpsSAYA_DEVICE_INFO As SAYA_DEVICE_INFO, _
            ByRef IpsTBUFSETUP As TXBUFSETUP2, _
            ByRef IpsIOGEOSETUP As IOGEOSETUP2, _
            ByRef IpsTDIO_MISC As TDIO_MISC, _
            ByRef IpsTADIO As TADIO2, _
            ByRef IpsTConfigPWM As TConfigPWM, _
            ByRef IpsTSetupPWMM As TSetupPWM, _
            ByRef IpsScpSetupAich As SCP_SETUP_AIALL, _
            ByRef IpsEXTENTION As ADIOX_EXTENTION2B, _
            ByVal dwRingbufferSize As Long, _
            ByVal bCardID As Byte, _
            ByRef IpsCharPayload As CharPayloadB_
        ) As Long
    
```

<b>引数</b>	hWnd bHwintr  bOpenDialog  IpsSAYA_DEVICE_INFO IpsTBUFSETUP IpsIOGEOSETUP IpsTDIO_MISC IpsTADIO IpsTConfigPWM IpsTSetupPWM IpsScpSetupAich  IpsEXTENTION  dwRingbufferSize bCardID IpsCharPayload	<p>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。</p> <p>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_HWINTRO ~ 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)</p> <p>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、IpsCharPayload.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 IpsCharPayload.lpcInitialDir で指定します。</p> <p>デバイス情報構造体 SAYA_DEVICE_INFO へのポインタ</p> <p>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</p> <p>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</p> <p>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</p> <p>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</p> <p>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</p> <p>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</p> <p>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</p> <p>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。</p> <p>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</p> <p>意味がありません。</p> <p>ターゲットデバイスのカードID。</p> <p>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。</p>
<b>戻り値</b>	成功すると1(TRUE)、失敗すると0(FALSE)が返ります。	



lpsScpSetupAich	コンフィグレーションファイルに保存された 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保存された 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。

## 2. 割り込み

### 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、スレット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 );

**VB** Declare Function bADioxInterruptStart Lib "ADiox2.dll" ( \_  
ByVal bStart As Long, ByVal bCardID As Byte) As Long

**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isable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、リングバッファ割り込み、DI エッジ割り込み、カウンタ割り込みの3 つあり、関数で特定できます。更にDI エッジ割り込みと、カウンタ割り込みの場合は、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 bADioxInterruptStatus( 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。この値より先、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、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、DI やカウンタが大量の割り込みが発生して処理しきれない、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セットされます。

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

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

**VB** Declare Function bADioxMessageCount Lib "ADiox2.dll" ( \_  
ByRef lpdwMessageCount As Long, ByVal bCardID As Byte ) As Long

**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。

## 3. リングバッファ

### bADioxDmaReadEX3 [New, Update]

AI/DI のリングバッファ( 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。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 構造体でファイル読み出し指定した場合、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。

<b>C/C++</b>	BOOL bADioxDmaReadEX3 ( LPDWORD lpdwBufa, double * lpdAiBufa, TStatusPack2 *lpsTStatus , BYTE bCardID );
<b>C#</b>	int bADioxDmaReadEX3 ( ref uint lpdwBufa, ref double lpdAiBufa, ref TStatusPack2 lpsTStatus , byte bCardID );
<b>VB</b>	Declare Function bADioxDmaReadEX3 Lib "ADiox2.dll" ( ByRef lpdwBufa As Long, _ ByRef lpdAiBufa As Double, ByRef lpsTStatus As TStatusPack2, ByVal bCardID As Byte) As Long
<b>引数</b>	<p>lpdwBufa                      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。</p> <p>lpdAiBufa                      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・スリム校正・スケールング・ニアライズされた値が格納されます。</p> <p>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。</p> <p>bCardID                        ターゲットデバイスのカードID。</p>
<b>戻り値</b>	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。
<b>注意</b>	旧 bADioxDmaReadEX2、旧 bADioxReadScpBuf2 は新しいデザインには推奨されません。本関数は、高速の <a href="#">ADX 14-80M-PCIEX</a> から、信号調節機能付きの <a href="#">ADX 42-1K-Ethernet</a> までカバーできるので機種依存のコードを減らすことができます。

### bADioxWriteMemoryEX

AO/DO のリングバッファ( 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。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、バッファへの書き込み(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。

<b>C/C++</b>	BOOL bADioxWriteMemoryEX( BYTE bAccessMode, LPDWORD lpdwBufa, BYTE bCardID );
<b>C#</b>	int bADioxWriteMemoryEX( byte bAccessMode, ref uint lpdwBufa, byte bCardID );
<b>VB</b>	Declare Function bADioxWriteMemoryEX Lib "ADiox2.dll" ( _ ByVal bAccessMode As Byte, ByRef lpdwBufa As Long, ByVal bCardID As Byte ) As Long
<b>引数</b>	<p>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。</p> <p>lpdwBufa                        リングバッファ書き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。</p> <p>bCardID                        ターゲットデバイスのカードID。</p>
<b>戻り値</b>	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。

#### < プリライトについて >

リングバッファ稼動前にリングバッファ2 バンクと転送元のソフトバッファに出力データを書き込んでおく必要があります。即ちバッファサイズの3倍を予め書いておく必要があります。このサイズはSAYA\_DEVICE\_INFO\_EX.dwPreWriteSizeOfDWORD から取得できます。

### 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。

<b>C/C++</b>	BOOL bADioxStopPoint2( LPDWORD lpdwStopAddress, TStatusPack2 *lpsTStatus, BYTE bCardID );
<b>C#</b>	int bADioxStopPoint2( ref uint lpdwStopAddress, ref TStatusPack2 lpsTStatus byte bCardID );
<b>VB</b>	Declare Function bADioxStopPoint2 Lib "ADiox2.dll" ( ByRef lpdwStopAddress As Long, _ ByRef lpsTStatus As TStatusPack2, ByVal bCardID As Byte ) As Long
<b>引数</b>	<p>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。</p> <p>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。</p> <p>bCardID                        ターゲットデバイスのカードID。</p>
<b>戻り値</b>	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。

## 4.設定

### 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 );`

**VB** `Declare Function bADioxSetupSymmetryEngine_VB2 Lib "ADiox2.dll" (_  
ByRef lpsTBufSetup As TXBUFSETUP, _  
ByRef ADIOX_EXTENTION2B lpsADIOX_EXTENTION2 , ByVal bCardID As Byte ) As Long`

**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。

### bADioxAnalogConfiguration2

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 bADioxAnalogConfiguration2 ( IOGEOSETUP2 *sIoGeoSetup, BYTE bCardID );`

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

**VB** `Declare Function bADioxAnalogConfiguration2 "ADiox2.dll" (_  
ByRef lpsIOGEOSETUP2 As IOGEOSETUP2, ByVal bCardID As Byte ) As Long`

**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、ストローブリッジの設定を行います。

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

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

**VB** `Declare Function bADioxDioMisc2 Lib "ADiox2.dll" (_  
ByRef lpsDIO_MISC As TDIO_MISC, ByVal bCardID As Byte ) As Long`

**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、ストローブリッジ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。

### bADioxScpSetup2

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ますので、ご注意ください。もしゼロ・スパ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);`

**VB** `Declare Function bADioxScpSetup2 Lib "ADiox2.dll" (_  
ByRef sScpSetupAich As SCP_SETUP_AICH, ByVal bCardID As Byte) As Long`

**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

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 スピン校正係数は解除されません。よってセンサー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・スピ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);

**VB** Declare Function bADioxScpSetupEX2 Lib "ADiox2.dll" ( \_  
ByRef sScpSetupAich As SCP\_SETUP\_AIALL, ByVal bCardID As Byte) As Long

**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

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);

**VB** Declare Function bADioxScpDefault\_VB2 Lib "ADiox2.dll" ( ByVal dwSensorMode As Long, \_  
ByVal dwDeviceNumber As Long, ByVal dwCH As Long, \_  
ByRef lpsSCP\_SETUP1 As SCP\_SETUP2\_PART1, ByRef lpsSCP\_SETUP2 As SCP\_SETUP2\_PART2, \_  
ByRef lpsSCP\_SETUP3 As SCP\_SETUP2\_PART3 ) As Long

**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。

## 5. ステータス

### bADioxStatus2

システムの稼動状態を取得します。

**C/C++** BOOL bADioxStatus ( struct TStatusPack2 \* sTStatus, BYTE bCardID );  
**C#** int bADioxStatus ( ref TStatusPack2 sTStatus, byte bCardID );  
**VB** Declare Function bADioxStatus Lib "ADiox2.dll" ( \_  
 ByRef sTStatus As TStatusPack2, ByVal bCardID As Byte ) As Long  
**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 **[New, Update]**

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 \* IpsSAYA\_DEVICE\_INFO, BYTE bCardID);  
**C#** void vADioxDeviceInfoEx ( ref SAYA\_DEVICE\_INFO\_EX IpsSAYA\_DEVICE\_INFO, byte bCardID);  
**VB** Declare Sub vADioxDeviceInfoEx Lib "ADiox2.dll" ( \_  
 ByRef IpsSAYA\_DEVICE\_INFO\_EX As SAYA\_DEVICE\_INFO, ByVal bCardID As Byte)  
**引数** \* IpsSAYA\_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。

### dwADioxNetError

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);  
**VB** Declare Function dwADioxNetError Lib "ADiox2.dll"(ByVal lpdwWriteRetry As Long, \_  
 ByVal lpdwReadRetry As Long, ByVal lpdwReadFlameError As Long, \_  
 ByVal lpdwReadAddressError As Long) As Long  
**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バリエーションエラー。  
**戻り値** 0 が返ります。

## 6. ポーリング

### 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ク経由とした場合、AO チャンネル、DO チャンネルはリングバックデータが優先されます。

**C/C++** BOOL bADioxADIO2 ( struct TADIO2 \* IpsTADio, BYTE bCardID );  
**C#** int bADioxADIO2 ( ref TADIO2 IpsTADio ,byte bCardID );  
**VB** Declare Function bADioxADIO2 Lib "ADiox2.dll" ( \_  
 ByRef IpsTADio As TADIO2, ByVal bCardID As Byte ) As Long  
**引数** \*IpsTADio 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

チャンネルを指定してアナログ入力値を取得します。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、読み出し周期より先、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となります。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 ( struct IOGEOSETUP2 \*IpsIoGeoSetup,  
LPDWORD lpdwData, int iInterval, BYTE bCardID);

**C#** int bADioxChannelAIw2 ( ref IOGEOSETUP2 IpsIoGeoSetup,  
ref uint lpdwData, int iInterval, byte bCardID);

**VB** Declare Function bADioxChannelAIw2 Lib "ADiox2.dll" (ByRef IpsIoGeoSetup As IOGEOSETUP2, \_  
ByRef lpdwData As Long, ByVal iInterval As Integer, ByVal bCardID As Byte ) As Long

**引数**

IpsIoGeoSetup	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。
lpdwData	取得したいアナログチャンネルもここで指定します。
iInterval	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。アナログチャンネルを指定してから、アナログ入力値を読み込むまでの時間を指定してください。(msec)値が小さいと、チャンネルの切り替えが完了せず、正確なデータが得られません。アナログ入力の信号源がバッファされていても 35msec 以上にしないと干渉する恐れがあります。
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。

## 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);

**VB** Declare Function bADioxDI Lib "ADiox2.dll" (ByRef lpdwDI As Long, ByVal bCardID As Byte) As Long

**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 );

**VB** Declare Function bADioxDO "ADiox2.dll" ( ByVal DWORD As Long, ByVal bCardID As Byte ) As Long

**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 で取得したDWORD 型のアナログ入力値を本関数に引き渡すと、ゼロ校正、スリム校正、リアライザ、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);

**VB** Declare Function dADioxLinCoef Lib "ADiox2.dll"(ByVal dwANALOG As Long, \_  
ByRef lpbBurnOut As Long, ByRef lpdwAlarm As Long, ByVal bCardID As Byte, \_  
ByVal bChannel As Byte) As Double

**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



## 8. ヘルパ

### 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。

```
sTBUFSETUP.dwClockScall           = 1000;
sTBUFSETUP.dwInterruptMode        = NOT_INT;
sTBUFSETUP.dwTrigStopMode         = RESET;
sTBUFSETUP.dwTrigStartMode        = BURST;
sTBUFSETUP.dwDeadTime             = 10;
sTBUFSETUP.dwMuxSequenceAuto      = ADX 14 の場合には 1、それ以外 0;
sTBUFSETUP.bTrigEnable            = FALSE;
sTBUFSETUP.dwAdiBufferOn          = 1;
sIOGEOSETUP.dwAo3Mode ~ sIOGEOSETUP.dwAo0Mode = NO_LINK;
sTDIO_MISC.dwSetFreq              = 3;
sTDIO_MISC.dwDinIntMode16 ~ sTDIO_MISC.dwDinIntMode31 = NO_INT;
sTDIO_MISC.dwCounterMode_A ~ sTDIO_MISC.dwCounterMode_D = 1;
sTDIO_MISC.dwLatchMode_A ~ sTDIO_MISC.dwLatchMode_D = Z_PHASE;
```

```
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 );
```

```
VB     Declare Function bADioxDefaultInit_Simple_vb Lib "ADiox2.dll" ( _
ByRef lpsTBUFSETUP As TXBUFSETUP2 , ByRef lpsIOGEOSETUP As IOGEOSETUP2, _
ByRef lpsTDIO_MISC As TDIO_MISC , ByRef lpsTConfigPWM As TConfigPWM, _
ByRef lpsTSetupPWM As TSetupPWM , ByRef lpsTADIO As TADIO2 , ByVal bCardID As Byte _
) As Long _
```

<b>引数</b>	*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。

### dADioxCalcFreq

構造体 TXBUFSETUP2 の dwClockScall に相当する周波数を KHz にて算出します。機種間の違いも自動的に吸収されます。チャンネルシーケンスを使用した場合の速度は、本関数の戻り値をチャンネル数で除算してください。

```
C/C++  double dADioxCalcFreq( DWORD dwSampling, BYTE bCardID );
```

```
C#      double dADioxCalcFreq( uint dwSampling, byte bCardID);
```

```
VB     Declare Function dADioxCalcFreq Lib "ADiox2.dll" ( _
ByRef dwSampling As Long, ByVal bCardID As Byte ) As Double
```

<b>引数</b>	dwSampling	TXBUFSETUP2.dwClockScall を指定します。
	bCardID	ターゲットデバイスのカードID。

**戻り値** 周波数(KHz)が返ります。

### dADiox\_Measurement\_Fs\_detection

本関数は割り込み間隔からサンプリング時間を計算します。チャンネルシーケンスは考慮されません。ソフトウェアによる時間計測なので精度はよくありませんが目安になります。

```
C/C++  double dADiox_Measurement_Fs_detection ( BYTE bCardID );
```

```
C#      double dADiox_Measurement_Fs_detection ( byte bCardID );
```

```
VB     Declare Function dADiox_Measurement_Fs_detection Lib "ADiox2.dll"(ByVal bCardID As Byte)As Double
```

<b>引数</b>	bCardID	ターゲットデバイスのカードID。
-----------	---------	------------------

**戻り値** サンプリング周波数を KHz で返します。

## 9. 波形生成

### bADioxWaveInit

波形ジェネレータを初期化します。

<b>C/C++</b>	BOOL bADioxWaveInit ( struct ADIOX_EXTENSION2 *lpsADIOX_EXTENSION2, double *dFrequency0,double *dFrequency1, BYTE bCardID );
<b>C#</b>	int bADioxWaveInit ( ref ADIOX_EXTENSION2 lpsADIOX_EXTENSION2 , ref double dFrequency0 , ref double dFrequency1 , byte bCardID );
<b>VB</b>	Declare Function bADioxWaveInit Lib "ADiox2.dll" ( _ ByRef lpsADIOX_EXTENSION2 As ADIOX_EXTENSION2 , _ ByRef dFrequency0 As Double , ByRef dFrequency1 As Double , _ ByVal bCardID As Byte) As Long
<b>引数</b>	lpsADIOX_EXTENSION2 ファイル保存、ファイル読み出し情報の入ったADIOX_EXTENSION2 構造体を指定します。 dFrequency0 AO0 の周波数 dFrequency1 意味がありません。 bCardID ターゲットデバイスのカードID。
<b>戻り値</b>	成功すると1(TRUE)、失敗すると0(FALSE)が返ります。

### 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、オフセットで生成できます。リングバッファ対象のAO、DI を前提にしているので AO は 1CH 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。

<b>C/C++</b>	BOOL bADioxWaveGenerate ( LPDWORD dwWriteBuffera,DWORD dwBufferSize, BYTE bCardID );
<b>C#</b>	int bADioxWaveGenerate ( ref uint dwWriteBuffera , uint dwBufferSize , byte bCardID );
<b>VB</b>	Declare Function bADioxWaveGenerate Lib "ADiox2.dll" ( ByRef dwWriteBuffera As Long , ByVal dwBufferSize As Long , ByVal bCardID As Byte ) As Long
<b>引数</b>	lpdwBuffera 波形を格納したリングバッファのデータのコピー元バッファへのポインタ。bADioxWriteMemoryEX を呼び lpdwBuffera にアタッチします。 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。
<b>戻り値</b>	成功すると1(TRUE)、失敗すると0(FALSE)が返ります。

# 10. 構造体

## 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_overnun;
    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_overnun;
    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;
};
```

### VB

```
Type IRQ_BUFFER
    dwADO_underrun           As Long
    dwADI_overnun           As Long
    dwADO_wr_addr_over      As Long
    dwADI_rd_addr_over      As Long
    dwPIO_mode_not_support  As Long
    dwTimming_error         As Long
    dwRequestPostmessage    As Long
    dwDI_CT_interrupt       As Long
    dwIrqst                  As Long
    dwTheCallCount          As Long
End Type
```

### メンバ変数

dwADO_underrun	AO/DO バッファ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。 <b>UNDERRUN_BUFFER</b> : バッファアンダーフローの発生 (エラー) 上記以外 : 問題なし
dwADI_overnun	AI/DI バッファ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。 <b>OVERRUN_BUFFER</b> : バッファ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3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には以下のように、抽出した割り込み要因との論理積をとって、それを抽出した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。

dwDI\_CT\_interrupt DI 割り込み・エンコーダカウンタ割り込み要因の詳細。

Bit00	カウンタ0 コペア 比較対象と一致
Bit01	カウンタ0 コペア 比較対象を上回った
Bit02	カウンタ0 コペア 比較対象を下回った
Bit03	カウンタ0 コペア 比較対象範囲に入った
Bit04	カウンタ1 コペア 比較対象と一致
Bit05	カウンタ1 コペア 比較対象を上回った
Bit06	カウンタ1 コペア 比較対象を下回った
Bit07	カウンタ1 コペア 比較対象範囲に入った
Bit08	カウンタ2 コペア 比較対象と一致
Bit09	カウンタ2 コペア 比較対象を上回った
Bit10	カウンタ2 コペア 比較対象を下回った
Bit11	カウンタ2 コペア 比較対象範囲に入った
Bit12	カウンタ3 コペア 比較対象と一致
Bit13	カウンタ3 コペア 比較対象を上回った
Bit14	カウンタ3 コペア 比較対象を下回った
Bit15	カウンタ3 コペア 比較対象範囲に入った
Bit16	DI0 のエッジ割り込み
Bit17	DI1 のエッジ割り込み
Bit18	DI2 のエッジ割り込み
Bit19	DI3 のエッジ割り込み
Bit20	DI4 のエッジ割り込み
Bit21	DI5 のエッジ割り込み
Bit22	DI6 のエッジ割り込み
Bit23	DI7 のエッジ割り込み
Bit24	DI8 のエッジ割り込み
Bit25	DI9 のエッジ割り込み
Bit26	DI10 のエッジ割り込み
Bit27	DI11 のエッジ割り込み
Bit28	DI12 のエッジ割り込み
Bit29	DI13 のエッジ割り込み
Bit30	DI14 のエッジ割り込み
Bit31	DI15 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、バックア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;
};
```

## VB

```

Type TXBUFSETUP2
    dwClockScall           As Long
    dwStartTrigDelay       As Long
    dwStartTrigLevel1     As Long
    dwStartTrigLevel2     As Long
    dwStopTrigDelay       As Long
    dwStopTrigLevel1     As Long
    dwStopTrigLevel2     As Long
    dwStartMask           As Long
    dwStartDiPattern      As Long
    dwStartTrigSourceDI_ch As Long
    dwStopMask            As Long
    dwStopDiPattern       As Long
    dwStopTrigSourceDI_ch As Long
    dwTrigStopMode        As Long
    dwTrigStartMode       As Long
    dwPreTrigger          As Long
    dwIamSlave            As Long
    dwAnalogTrigSourceStart As Long
    dwDigitalTrigSourceStart As Long
    dwAnalogTrigSourceStop As Long
    dwDigitalTrigSourceStop As Long
    dwIntrruptMode        As Long
    dwDeadTime            As Long
    dwStopCounterValue    As Long
    dwMuxSeqenceAuto      As Long
    dwAoHspBufferd        As Long
    dwDoHspBufferd        As Long
    dwAdiBufferOn         As Long
    bConnectBuffer        As Byte
    bTrigEnable           As Long

```

End Type

### メンバ変数 (以下で示されていないメンバ変数は使用されません)

#### 【サンプリング周波数】

dwClockScall                      サンプル時間を設定します。シーケンシャル取り込み数を  $n$  とした場合のサンプリング時間は  
 デジタルフィルタオフ     $16.95421\text{ns} \times \text{dwClockScall} \times n$   
 デジタルフィルタオン     $67.81684\text{ns} \times \text{dwClockScall} \times n$   
 となります。

#### 【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。

<b>RESET</b>	トリガ条件は成立しない
<b>BURST</b>	無条件にトリガを成立させる。
<b>DI_POSEDGE</b>	デジタル入出力の立ち上がりがエッジでトリガを成立させる
<b>DI_NEGEDGE</b>	デジタル入出力の立ち下がりがエッジでトリガを成立させる
<b>DI_PATTERN</b>	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
<b>AI_LEVEL</b>	アナログ入出力のレベル(エッジ)トリガ
<b>AI_AREA</b>	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。  
 0:AI    1:A00    2:A01

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。

**【トリガ遅延・プリトリガ】**

dwStartTrigDelay スタートトリガ遅延の指定。(プリトリガ)この値がナトリガより先送れて、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 ストップトリガ遅延の指定。(プリトリガ)この値がナトリガより先送れて、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。ADX 42-1K-Ethernet ではこの機能は実装されないため、意味がありません。

**【同期運転】**

dwIamSlave 複数ポートの同期運転を行う場合、自分がマスター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でマスターです。単独使用の場合には必ずマスター(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】**

dwInterruptMode 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。いかなるストップトリガがかわ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】**

dwMuxSequenceAuto シーケンシャル取り込みの設定を行います。  
この値が0の場合、シーケンシャル取り込みはディセーブル  
この値が1の場合、2ch自動切換え  
この値が2の場合、4ch自動切換え  
この値が3の場合、8ch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。

bConnectBuffer 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/シングルエンド切替、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;
};
```

### VB

```
Type IOGEOSETUP2
    dwAo3Mode           As Long
    dwAo2Mode           As Long
    dwAo1Mode           As Long
    dwAo0Mode           As Long
    dwInputShort        As Long
    dwCOB               As Long
    dwFilterEnable      As Long
    dwDifferential      As Long
    dwMux               As Long
    dwAI_Range          As Long
    dwAO_Range          As Long
    dwChatCan           As Long
    dwNoiseShaper       As Long
End Type
```

### メンバ変数 (以下で示されていないメンバ変数は使用されません)

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。
<b>NO_LINK</b>	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。
<b>LIVE_CTC</b>	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。
<b>LATCH_CTC</b>	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。
dwCOB	この値が1の場合、アナログ入力値はコンPLEMENT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。本変数を0 とすることで、フィルタをオフ、1 または3 とすることで4 次移動平均フィルタをオンにします。
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をON にします。ノイズシェーパは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を加算します。
dwMux	入力チャンネルを切り替えます。0-7 が有効な値です。
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

エンコーダカウンタ、周波数カウンタ、PWM(パルスジェネレータ)、ストロブラッチ、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;
};
```

**VB**

```

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

```

## メモリ変数 (以下で示されていないメモリ変数は使用されません)

dwStrobeInternal	ストロブ出力をデジタル出力チャンネル15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(ストロブ専用ヘッダネクタまたはピンを使用)
dwDinIntMode16	DI0のエッジ割り込み要因を指定します。
dwDinIntMode17	DI1のエッジ割り込み要因を指定します。
dwDinIntMode18	DI2のエッジ割り込み要因を指定します。
dwDinIntMode19	DI3のエッジ割り込み要因を指定します。
dwDinIntMode20	DI4のエッジ割り込み要因を指定します。
dwDinIntMode21	DI5のエッジ割り込み要因を指定します。
dwDinIntMode22	DI6のエッジ割り込み要因を指定します。
dwDinIntMode23	DI7のエッジ割り込み要因を指定します。
dwDinIntMode24	DI8のエッジ割り込み要因を指定します。
dwDinIntMode25	DI9のエッジ割り込み要因を指定します。
dwDinIntMode26	DI10のエッジ割り込み要因を指定します。
dwDinIntMode27	DI11のエッジ割り込み要因を指定します。
dwDinIntMode28	DI12のエッジ割り込み要因を指定します。
dwDinIntMode29	DI13のエッジ割り込み要因を指定します。
dwDinIntMode30	DI14のエッジ割り込み要因を指定します。
dwDinIntMode31	DI15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 >
	<b>NO_INT</b> 割り込み要因なし
	<b>POSEDGE_INT</b> 立ち上がりエッジ
	<b>NEGEDGE_INT</b> 立下りエッジ
	<b>DUALEDGE_INT</b> デュアルエッジ
dwCounterMode_A	エンコーダカウンタ0の動作モードを以下の0-7で指定します。
dwCounterMode_B	エンコーダカウンタ1の動作モードを以下の0-7で指定します。
dwCounterMode_C	エンコーダカウンタ2の動作モードを以下の0-7で指定します。
dwCounterMode_D	エンコーダカウンタ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、Z相未使用
	1:4 倍速エンコーダカウンタ、Z相使用
	2:2 倍速エンコーダカウンタ、Z相未使用
	3:2 倍速エンコーダカウンタ、Z相使用
	4:1 倍速エンコーダカウンタ、Z相未使用
	5:1 倍速エンコーダカウンタ、Z相使用
	6:アップダウンカウンタ (リレスカウンタ)、Z相未使用
	7:アップダウンカウンタ (リレスカウンタ)、Z相使用
dwLatchMode_A	カウンタ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 >
	<b>SOFT</b> ソフトウェアラッチ
	<b>Z_PHASE</b> Z相条件成立でラッチ
	<b>DI_SEL</b> Y相立ち上がりエッジでラッチ
dwZ_CENTER_A	カウンタ0、Z相条件成立モード(カウンタリセット)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1、Z相条件成立モード(カウンタリセット)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2、Z相条件成立モード(カウンタリセット)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3、Z相条件成立モード(カウンタリセット)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でカウンタリセット、0で非リセット。
dwCompareMode	0を指定すると、カウンタ比較値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(若い方)のカウンタ値をカウンタ比較値LOWとして利用する。カウンタ0のみカウンタ3の値を使う

dwLinkCounterToDO_A	カウンタ0 のコペア結果を 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1 のコペア結果を 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2 のコペア結果を 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3 のコペア結果を 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= コペア値 LOW と一致 bit1= コペア値 LOW 以上 bit2= コペア値 LOW 以下 bit3= コペア値 LOW ~ コペア値 HIGH の範囲内 カウンタコペア出力と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0 のコ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1 のコ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2 のコ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3 のコ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= コペア値 LOW と一致 bit1= コペア値 LOW 以上 bit2= コペア値 LOW 以下 bit3= コペア値 LOW ~ コペア値 HIGH の範囲内
dwReferenceLow_A	カウンタ0 コペア値 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1 コペア値 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2 コペア値 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3 コペア値 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0 コペア値 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1 コペア値 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2 コペア値 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3 コペア値 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下さい。 0:1sec、1:100msec、2:10msec、3:1msec

[カウンタコペア出力の DO 出力の割り付け]

カウンタ0 コペア値 LOW と一致	DO0
カウンタ0 コペア値 LOW 以上	DO1
カウンタ0 コペア値 LOW 以下	DO2
カウンタ0 コペア値 LOW ~ HIGH の範囲内	DO3
カウンタ1 コペア値 LOW と一致	DO4
カウンタ1 コペア値 LOW 以上	DO5
カウンタ1 コペア値 LOW 以下	DO6
カウンタ1 コペア値 LOW ~ HIGH の範囲内	DO7
カウンタ2 コペア値 LOW と一致	DO8
カウンタ2 コペア値 LOW 以上	DO9
カウンタ2 コペア値 LOW 以下	DO10
カウンタ2 コペア値 LOW ~ HIGH の範囲内	DO11
カウンタ3 コペア値 LOW と一致	DO12
カウンタ3 コペア値 LOW 以上	DO13
カウンタ3 コペア値 LOW 以下	DO14
カウンタ3 コペア値 LOW ~ HIGH の範囲内	DO15

## 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;
};
```

## VB

```

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

```

End Type

**メモリ変数 (以下で示されていないメモリ変数は使用されません)**

dwAi0	アナログ入力値が格納されます。値は0~0xFFFFの16Bitコードが入ります。
dAi0	アナログ入力値(信号調節で変換された物理値)が格納されます。
dwAo0-1	アナログ出力チャンネル0-1 出力値 (最小~最大が、0~0xFFFFに対応します ) dwAo0-1 がチャンネル0~1に相当します。
dAo0-1	前記 dwAo0-1 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。 dwAo0-1 がdAo0-1に相当します。
dwDOS	デジタル出力チャンネル0~15の出力値 (Bit0~Bit15がデジタル出力チャンネル0~15に対応します )
dwDI	デジタル入力チャンネル0~15の入力値 (Bit0~Bit15がデジタル入力チャンネル0~15に対応します )
dwDI_Latch	デジタル入力チャンネル0~15のストロープラッチ値 (Bit0~Bit15がデジタル入力チャンネル0~15に対応します )
dwCounterA	エンコーダーカウンタ0のライブ値 (現在の値 1)
dwCounterB	エンコーダーカウンタ1のライブ値 (現在の値 1)
dwCounterC	エンコーダーカウンタ2のライブ値 (現在の値 1)
dwCounterD	エンコーダーカウンタ3のライブ値 (現在の値 1)
dwLatchA	エンコーダーカウンタ0のラッチ値 1
dwLatchB	エンコーダーカウンタ1のラッチ値 1
dwLatchC	エンコーダーカウンタ2のラッチ値 1

dwLatchD	エンコーダカウンタ3 のラッチ値 1
dwFreqLatch_A	周波数カウンタ0 のライブ値 (ゲート周期に何サイクルあつたかを表します)
dwFreqLatch_B	周波数カウンタ1 のライブ値 (ゲート周期に何サイクルあつたかを表します)
dwFreqLatch_C	周波数カウンタ2 のライブ値 (ゲート周期に何サイクルあつたかを表します)
dwFreqLatch_D	周波数カウンタ3 のライブ値 (ゲート周期に何サイクルあつたかを表します)
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。

## 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_overnun;
    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_overnun;
    public uint dwWriteAddress;
    public uint dwBusmasterOn;
    public uint dwDaqInit;
    public uint dwDaqEnable;
    public uint dwTrigSens2;
    public uint dwTrigSens1;
    public uint dwTrigSens0;
    public uint dwTrigSeq;
};
```

### VB

```
Type TStatusPack2
    dwBurstMax As Long
    dwADO_underrun As Long
    dwADI_overnun As Long
    dwWriteAddress As Long
    dwBusmasterOn As Long
    dwDaqInit As Long
    dwDaqEnable As Long
    dwTrigSens2 As Long
    dwTrigSens1 As Long
    dwTrigSens0 As Long
    dwTrigSeq As Long
End Type
```

### メモリ変数 (以下で示されていないメモリ変数は使用されません)

dwADO_underrun	AO/DO バッファ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つたため、送るべきデータが無かつたことを意味します。この場合前の値が保持されます。以下の定義された数値が格納されます。 <b>UNDERRUN_BUFFER</b> : バッファアンダーフローの発生 (エラー) 上記以外 : 問題なし
dwADI_overnun	AI/DI バッファ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つたため、バッファにデータを収容しきれなかつ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。 <b>OVERRUN_BUFFER</b> : バッファ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

簡単なデバイス情報を格納します。詳細なデバイス情報はSAYA\_DEVICE\_INFO\_EX を参照願います。

### 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;
};
```

### VB

```
Type SAYA_DEVICE_INFO
    dwDeviceType          As Long
    dwBufferSizeOfByte    As Long
    dwBufferSizeOfDWORD  As Long
    iDIO_TRIG_SOURCE_MAX As Integer
    iAIO_TRIG_SOURCE_MAX As Integer
    iTRIG_MODE_MAX       As Integer
    dwSAMPLE_FAST        As Long
    dwSAMPLE_SLOW        As Long
    iPRE_TRIG_MAX        As Integer
End Type
```

### メンバ変数 (以下で示されていないメンバ変数は使用されません)

dwDeviceType	デバイスの種類(4)が入ります。
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 [New]**

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;
};

```

## VB

```

Type SAYA_DEVICE_INFO
    dwDeviceType           As Long
    dwBufferSizeOfByte     As Long
    dwBufferSizeOfDWORD   As Long
    iDIO_TRIG_SOURCE_MAX  As Integer
    iAIO_TRIG_SOURCE_MAX  As Integer
    iTRIG_MODE_MAX        As Integer
    dwSAMPLE_FAST         As Long
    dwSAMPLE_SLOW        As Long
    iPRE_TRIG_MAX         As Integer
    dwSAMPLE_SLOW2       As Long
    dwTempSensor          As Long
    dwOnboardCAL          As Long
    dwCounterToRingbuf    As Long
    dwClockOut            As Long
    dwPreTrigSize         As Long
    dwBankSize            As Long
    dwBankSizeUnit        As Long
    dwClockIn            As Long
    dwChseqMax            As Long
    dwChseqMin            As Long
    dwCyclicTrig          As Long
    dwChseqDfType         As Long
    dwCoreAirange         As Long
    dwDoMap               As Long
    dwAiMap               As Long
    dwDiMap               As Long
    dwAoMap               As Long
    dwCounterMap          As Long
    dwFreqCounterMap      As Long
    dwPwmMap              As Long
    dwProgramableScp      As Long
    dwAdaptiveFreqUnit    As Long
    dAoLsbLevel           As Double
    dAoLsbLevelEx         As Double
    dAoLsbOffset          As Double
    dAoLsbOffsetEx        As Double
    dwDI[16]              As Long
    dwDO[16]              As Long
    dwBufferSizeOfBytesr  As Long
    dwBufferSizeOfWORDsr  As Long
    dwPreWriteSizeOfByte  As Long
    dwPreWriteSizeOfDWORD As Long
    dwAdcStyle            As Long
    dwAdoBufMode          As Long
End Type

```

## メンバ変数

dwDeviceType	デバイスの種類が入ります (以下参照) ADX 85-1M-PCI(EX) 0 ADX 42-1K-ETHERNET 4 ADX 14-80M-PCIEX 5 ADX 52-1K-ETHERNET 6
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 ADX 85-1M-PCI(EX)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 ADX 85-1M-PCI(EX)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。dwBufferSizeOfWORDsr 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。
dwBufferSizeOfWORDsr	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ション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、IOGEOSSETUP.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。 (ADX 14-80M-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.dwMuxSequenceAuto (チャンネルシーケンス)の最大値
dwChseqMin	TBUFSETUP.dwMuxSequenceAuto (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	シーケンス取り込み 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 ADX 52-1K-ETHERNET の拡張ポート形式の信号変換アンプや ADX 42-1K-ETHERNET の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、IOGEOSSETUP.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。 (ADX 52-1K-ETHERNET はプログラマブルではないので 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) (ADX 52-1K-ETHERNET は拡張ポート式なので無意味)
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 ADX 14-80M-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 やエベア出力)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];
};
```

**VB** Type ADIOX\_EXTENTION2B  
 lpcAdiFileName As String  
 lpcDmyFileName As String  
 bDoubleSave As Long  
 dwADI\_style As Long  
 iAO\_Gain0 As Integer  
 iAO\_Gain1 As Integer  
 iAO\_Offset0 As Integer  
 iAO\_Offset1 As Integer  
 iAO\_SamplePerCycle0 As Integer  
 iAO\_SamplePerCycle1 As Integer  
 dwADO\_style0 As Long  
 dwADO\_style1 As Long  
 lpcAdoFileName As String  
 dwReserverd(16) As Long  
 End Type

**メンバ変数 (以下で示されていないメンバ変数は使用されません)**

bDoubleSave	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_Offset0	波形ジェネレータAO0 オフセット。±10000 を指定します。
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。 ADX_Sin サイン ADX_Cos コサイン ADX_Exp Exp(2p) ADX_Sqrt Sqrt(2 ) Sqrt は平方根 ADX_Triangle 三角波 ADX_Lamp ランプ波形 (のこぎり波) 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)
iAO_SamplePerCycle0	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) 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。
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ー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ーランする) 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ーランを回避します。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わ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\_SETUP\_AICH

信号調節関連の設定を格納します。SCP\_SETUP2 構造体ではなく、体構造体を使うことによりターゲットデバイス1個分のメモリ変数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\_MFIO はターゲットデバイスのカードID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];
};
```

### VB

```
Type SCP_SETUP_AICH
    dwSensorMode(MAX_AI_CH)    As Long
    dwLDO(MAX_AI_CH)          As Long
    doZeroPos(MAX_AI_CH)      As Double
    doSpanPos(MAX_AI_CH)      As Double
    bScalling(MAX_AI_CH)      As Long
    dOutTopScall(MAX_AI_CH)   As Double
    dOutBottomScall(MAX_AI_CH) As Double
    dInTopScall(MAX_AI_CH)    As Double
    dInBottomScall(MAX_AI_CH) As Double
    bAlarmMode(MAX_AI_CH)     As Long
    dAlarmUpper(MAX_AI_CH)    As Double
    dAlarmLower(MAX_AI_CH)    As Double
End Type
```

### メモリ変数 (以下で示されていないメモリ変数は使用されません)

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エンデ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用します。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) 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) 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) 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) 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) を MAX_AI_CH 個格納します。
dAlarmLower	アラーム設定値 (下) を MAX_AI_CH 個格納します。

## SCP\_SETUP\_AIALL

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];
};
```

### VB

```
Type SCP_SETUP_AIALL
    dwSensorMode(MAX_AI_CH)    As Long
    dwLDO(MAX_AI_CH)          As Long
    doZeroPos(MAX_AI_CH)      As Double
    doSpanPos(MAX_AI_CH)      As Double
    doZero_Coefficient(MAX_AI_CH) As Double
    doSpan_Coefficient(MAX_AI_CH) As Double
    bScalling(MAX_AI_CH)      As Long
    dOutTopScall(MAX_AI_CH)   As Double
    dOutBottomScall(MAX_AI_CH) As Double
    dInTopScall(MAX_AI_CH)    As Double
    dInBottomScall(MAX_AI_CH) As Double
    bAlarmMode(MAX_AI_CH)     As Long
    dAlarmUpper(MAX_AI_CH)    As Double
    dAlarmLower(MAX_AI_CH)    As Double
End Type
```

### 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;
};
```

### VB

```
Type CharPayloadB
    lpcInitialDir           As String
    lpcConfigFileName      As String
End Type
```

### 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

計測ファイル保存ヘッダです。ADiox2.dll 内保存計測データは、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    dwBufaSize;
    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;
};

```

**VB**

```

Type LOG_FRONTEND
    dwHeaderCode           As Long
    dwDeviceName          As Long
    dwBufaSize            As Long
    dClockScall           As Double
    bBitScall             As Byte
    bAI_ChannelScall      As Byte
    bDI_ChannelScall      As Byte
    DataType              As Byte
    dwGetYear             As Long
    dwGetMonth            As Long
    dwGetDay              As Long
    dwGetHour             As Long
    dwGetMinute           As Long
    dwGetSecond           As Long
    dwGetMilliseconds     As Long
    dwInrange             As Long
    dwReserved1           As Long
    dwReserved2           As Long
End Type

```

**メンバ変数 (以下で示されていないメンバ変数は使用されません)**

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)
dwBufaSize	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ADX 42-1K-ETHERNET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	計測開始のミ秒