

A blue-tinted image of a circuit board, likely the ADIOX-MK II, serving as a background for the top section of the document. The board is densely packed with components and traces.

ADIOX2-API

HardwareAccess
Storage/Logger
SignalCondition
SignalAnalysis
Graphics/Trend
WaveGenerate

ADIOX-MK II

MULTIFUNCTION-I/O-X2 SERIES
ADIOX2-API
REFERENCE ADX2-85/DX2-64
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	9
6. ポーリング	10
7. 校正	11
8. ヘルパ	11
9. 波形生成	13
10. 構造体	14

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 85-1M-PCIEX/ ADX II 85-1M-PCI/ DX II 64-1M-PCI** 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。

DX II -64-1M-PCI

ソフトウェア・レジスタでは、**ADX II 85-1M-PCIEX/ ADX II 85-1M-PCI** 互換ですが、ハードウェアにアナログ機能が搭載されていないので、ソフトウェア的にアナログ機能进行操作しても何も起こりません。また取得したアナログ値も無意味です。

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;”という名前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=0~3 が **ADX II 85-1M-PCIEX/ ADX II 85-1M-PCI/ DX II 64-1M-PCI** に割り当てられます。

1.初期化・再初期化

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

引数	hWnd bHwintr bCardID	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。 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) 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 *IpsTBUFSETUP,
    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
);
```


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 *sTXBUFSETUP,
    IOGEOSSETUP2 *sIOGEOSSETUP,
    TDIO_MISC *sTDIO_MISC,
    TADIO2 *sTADIO,
    TConfigPWM *sTConfigPWM,
    TSetupPWM *sTSetupPWM,
    SCP_SETUP_AIALL *sScpSetupAich,
    ADIOX_EXTENTION2 * IpsEXTENTION,
    BYTE CARD_ID,
    CharPayloadC *IpsCharPayload
);
```

C# `int bADioxStore_EX3`

```
(
    int bOpenDialog,
    ref TXBUFSETUP2 sTXBUFSETUP,
    ref IOGEOSSETUP2 sIOGEOSSETUP,
    ref TDIO_MISC sTDIO_MISC,
    ref TADIO2 sTADIO,
    ref TConfigPWM sTConfigPWM,
    ref TSetupPWM sTSetupPWM,
    ref SCP_SETUP_AIALL IpsScpSetupAich,
    ref ADIOX_EXTENTION IpsEXTENTION,
    byte bCARD_ID,
    ref CharPayloadC IpsCharPayload
);
```

VB `Declare Function bADioxStore_EX3B Lib "ADiox2.dll" _`

```
( _
    ByVal bOpenDialog As Long, _
    ByRef IpsTXBUFSETUP As TXBUFSETUP2, _
    ByRef IpsIOGEOSSETUP As IOGEOSSETUP2, _
    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_vb As ADIOX_EXTENTION2B, _
    ByVal bCardID As Byte, _
    ByRef IpsCharPayload As CharPayloadB_
) As Long
```

引数	<p><code>bOpenDialog</code> 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 <code>TRUE(=1)</code>とします。 <code>FALSE(=0)</code>とした場合、ファイル名を <code>IpsCharPayload.lpcConfigFileName</code> 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 <code>IpsCharPayload.lpcInitialDir</code> で指定します。</p> <p><code>IpsSAYA_DEVICE_INFO</code> デバイス情報構造体 <code>SAYA_DEVICE_INFO</code> へのポインタ。</p> <p><code>IpsTXBUFSETUP</code> コンフィグレーションファイルに保存したい <code>TXBUFSETUP2</code> 構造体へのポインタ。</p> <p><code>IpsIOGEOSSETUP</code> コンフィグレーションファイルに保存したい <code>IOGEOSSETUP2</code> 構造体へのポインタ。</p> <p><code>IpsTDIO_MISC</code> コンフィグレーションファイルに保存したい <code>TDIO_MISC</code> 構造体へのポインタ。</p> <p><code>IpsTADIO</code> コンフィグレーションファイルに保存したい <code>TADIO2</code> 構造体へのポインタ。</p> <p><code>IpsTConfigPWM</code> コンフィグレーションファイルに保存したい <code>TConfigPWM</code> 構造体へのポインタ。</p> <p><code>IpsTSetupPWM</code> コンフィグレーションファイルに保存したい <code>TSetupPWM</code> 構造体へのポインタ。</p> <p><code>IpsScpSetupAich</code> コンフィグレーションファイルに保存したい <code>SCP_SETUP_AIALL</code> 構造体へのポインタ。</p> <p><code>IpsEXTENTION</code> 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。コンフィグレーションファイルに保存したい <code>ADIOX_EXTENTION2</code> (VB 用は <code>ADIOX_EXTENTION2B</code>)構造体へのポインタ。</p> <p><code>bCardID</code> ターゲットデバイスのカード ID。</p> <p><code>IpsCharPayload</code> 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 <code>CharPayloadC</code>(VB 用は <code>CharPayloadB</code>)へのポインタ。</p>
戻り値	成功すると <code>1(TRUE)</code> 、失敗すると <code>0(FALSE)</code> 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);

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 *IpsIrqbuf, BYTE bCardID);

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

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

引数 IpsIrqbuf 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);

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(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。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。

C/C++ BOOL bADioxDmaReadEX3 (LPDWORD lpdwBuffer, double * lpdAiBuffer, TStatusPack2 *lpsTStatus , BYTE bCardID);

C# int bADioxDmaReadEX3 (ref uint lpdwBuffer, ref double lpdAiBuffer, ref TStatusPack2 lpsTStatus , byte bCardID);

VB Declare Function bADioxDmaReadEX3 Lib "ADiox2.dll" (ByRef lpdwBuffer As Long, _ ByRef lpdAiBuffer As Double, ByRef lpsTStatus As TStatusPack2, ByVal bCardID As Byte) As Long

引数

lpdwBuffer	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。
lpdAiBuffer	信号調節機能により、物理定数(電圧)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。
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)で稼働中です。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。

注意 旧 bADioxDmaReadEX2 は新しいデザインには推奨されません。本関数は、高速の [ADX II 14-80M-PCIEX](#) から、信号調節機能付きの [ADX II 42-1K-Ethernet](#) 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(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。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。

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

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

VB Declare Function bADioxWriteMemoryEX Lib "ADiox2.dll"(_ ByVal bAccessMode As Byte, ByRef lpdwBuffer As Long, ByVal bCardID As Byte) As Long

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。
lpdwBuffer	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。
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

<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>
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、

$$\text{dwPreWriteSize} = (\text{sSAYA_DEVICE_INFO.dwDeviceType} == \text{ADX14_PCI}) ? \text{dwBufferSize} * 3$$

$$: (\text{sSAYA_DEVICE_INFO.dwDeviceType} == \text{ADX42_LAN}) ? \text{dwBufferSize} * 3$$

$$: \text{dwBufferSize} * 2 + \text{ADX85_PCI_BUFFER_SIZE};$$

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。

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);

VB Declare Function bADioxStopPoint2 Lib "ADiox2.dll" (_
ByRef lpdwStopAddress As Long, _
ByRef lpsTStatus As TStatusPack2, ByVal bCardID As Byte) As Long

引数 lpdwStopAddress リングバッファが停止した場合 (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 lpsTXBUFSETUP As TXBUFSETUP2, _
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);

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);
VB Declare Function bADioxConfigPWM2 Lib "ADiox2.dll" (_
 ByRef sTConfigPWM As TConfigPWM, ByVal bCardID As Byte) As Long
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);
VB Declare Function bADioxSetupPWM Lib "ADiox2.dll" (_
 ByRef IpsTSetupPWM As TSetupPWM, ByVal bCardID As Byte) As Long
引数 sTSetupPWM PWM のデューティ比を設定します。
 bCardID 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。
 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

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(バッファオーバーラン・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);
VB Declare Function bADioxRingBufferMode Lib "ADiox2.dll" (_
 ByVal dwRingBufferMode As Long, ByVal bCardID As Byte) As Long
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 程度にししないとバッファオーバーラン/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。

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。

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 * 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 (struct IOGEOSUP2 *IpsIoGeoSetup,
LPDWORD lpdwData, int iInterval, BYTE bCardID);

C# int bADioxChannelAIw2 (ref IOGEOSUP2 IpsIoGeoSetup,
ref uint lpdwData, int iInterval, byte bCardID);

VB Declare Function bADioxChannelAIw2 Lib "ADiox2.dll" (ByRef IpsIoGeoSetup As IOGEOSUP2, _
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 IOGEOSUP2 構造体を指定します。取得したいアナログチャンネルもここで指定します。
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 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 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。

7.校正

bADioxAutoCal2

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);
VB Declare Function bADioxAutoCal2 Lib " ADiox2.dll " (ByRef lpsIOGEOSETUP As IOGEOSETUP2, _
ByRef lpsTBUFSETUP As TXBUFSETUP, ByVal bCardID As Byte) As Long
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

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);
VB Declare Function dwADioxAutoCal_Status1 "ADiox2.dll" (ByRef lpbInAutoCal As Long) As Long
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。

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 II 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 _`

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。

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`

引数

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`

引数

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。

bADioxClockMode

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);`

VB `Declare Function bADioxClockMode Lib "ADiox2.dll" (ByVal bClockOutEnable As Long, _ ByVal bCardID As Byte) As Long`

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。

9. 波形生成

bADioxWaveInit

波形ジェネレータを初期化します。

C/C++ `BOOL bADioxWaveInit (struct ADIOX_EXTENTION2 *IpsADIOX_EXTENTION2, double *dFrequency0, double *dFrequency1, BYTE bCardID);`

C# `int bADioxWaveInit (ref ADIOX_EXTENTION2 IpsADIOX_EXTENTION2 , ref double dFrequency0 , ref double dFrequency1 , byte bCardID);`

VB `Declare Function bADioxWaveInit Lib "ADiox2.dll" (_ ByRef IpsADIOX_EXTENTION2 As ADIOX_EXTENTION2 , _ ByRef dFrequency0 As Double , ByRef dFrequency1 As Double , _ ByVal bCardID As Byte) As Long`

引数 IpsADIOX_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 A00 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。

戻り値 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。

C/C++ `BOOL bADioxWaveGenerate (LPDWORD dwWriteBuffa, DWORD dwBufferSize, BYTE bCardID);`

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

VB `Declare Function bADioxWaveGenerate Lib "ADiox2.dll" (ByVal dwWriteBuffa As Long , ByVal dwBufferSize As Long , ByVal bCardID As Byte) As Long`

引数 IpdwBuffa 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 IpdwBuffa 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。

戻り値 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_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（以下で示されていないメンバ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。 OVERRUN_BUFFER : 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。

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 のエッジ割り込み
 Bit17 DI17 のエッジ割り込み
 Bit18 DI18 のエッジ割り込み
 Bit19 DI19 のエッジ割り込み
 Bit20 DI20 のエッジ割り込み
 Bit21 DI21 のエッジ割り込み
 Bit22 DI22 のエッジ割り込み
 Bit23 DI23 のエッジ割り込み
 Bit24 DI24 のエッジ割り込み
 Bit25 DI25 のエッジ割り込み
 Bit26 DI26 のエッジ割り込み
 Bit27 DI27 のエッジ割り込み
 Bit28 DI28 のエッジ割り込み
 Bit29 DI29 のエッジ割り込み
 Bit30 DI30 のエッジ割り込み
 Bit31 DI31 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;
};
```

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 サンプル周波数を指定します。サンプリング周波数=(40MHz÷dwClockScall)÷シーケンシャル取り込みチャンネル数となります。

【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。

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。

0: AI 1: AO0 2: AO1 3: AO2 4: AO3 5: AO4
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 **32 サンプル先**に、データ収集を開始します。

【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】

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】

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
この値が 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;
};
```

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。
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。

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。 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
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。
dwMux	入力チャンネルを切り替えます。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。
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。
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。
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;
};
```

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）

dwStrobeInternal	ストロブ入出力をデジタル入出力チャンネル 31 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。16ch の PWM を有します。
dwDinIntMode16	DI16 のエッジ割り込み要因を指定します。
dwDinIntMode17	DI17 のエッジ割り込み要因を指定します。
dwDinIntMode18	DI18 のエッジ割り込み要因を指定します。
dwDinIntMode19	DI19 のエッジ割り込み要因を指定します。
dwDinIntMode20	DI20 のエッジ割り込み要因を指定します。
dwDinIntMode21	DI21 のエッジ割り込み要因を指定します。
dwDinIntMode22	DI22 のエッジ割り込み要因を指定します。
dwDinIntMode23	DI23 のエッジ割り込み要因を指定します。
dwDinIntMode24	DI24 のエッジ割り込み要因を指定します。
dwDinIntMode25	DI25 のエッジ割り込み要因を指定します。
dwDinIntMode26	DI26 のエッジ割り込み要因を指定します。
dwDinIntMode27	DI27 のエッジ割り込み要因を指定します。
dwDinIntMode28	DI28 のエッジ割り込み要因を指定します。
dwDinIntMode29	DI29 のエッジ割り込み要因を指定します。
dwDinIntMode30	DI30 のエッジ割り込み要因を指定します。
dwDinIntMode31	DI31 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	エンコーダカウンター0 の動作モードを以下の 0-7 で指定します。
dwCounterMode_B	エンコーダカウンター1 の動作モードを以下の 0-7 で指定します。
dwCounterMode_C	エンコーダカウンター2 の動作モードを以下の 0-7 で指定します。
dwCounterMode_D	エンコーダカウンター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
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(若い方)のカウンタ値をカウンタコンペア値 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 カウンタコンペア出力と 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	DO16
カウンタ 0 コンペア値 LOW 以上	DO17
カウンタ 0 コンペア値 LOW 以下	DO18
カウンタ 0 コンペア値 LOW~HIGH の範囲内	DO19
カウンタ 1 コンペア値 LOW と一致	DO20
カウンタ 1 コンペア値 LOW 以上	DO21
カウンタ 1 コンペア値 LOW 以下	DO22
カウンタ 1 コンペア値 LOW~HIGH の範囲内	DO23
カウンタ 2 コンペア値 LOW と一致	DO24
カウンタ 2 コンペア値 LOW 以上	DO25
カウンタ 2 コンペア値 LOW 以下	DO26
カウンタ 2 コンペア値 LOW~HIGH の範囲内	DO27
カウンタ 3 コンペア値 LOW と一致	DO28
カウンタ 3 コンペア値 LOW 以上	DO29
カウンタ 3 コンペア値 LOW 以下	DO30
カウンタ 3 コンペア値 LOW~HIGH の範囲内	DO31

TADIO2

アナログデジタル入力・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）

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	アナログ入力値が格納されます。値は単位 mV の電圧値が入ります。
dwAo0-4	ログ出力チャンネル 0-4 出力値（最小~最大が、0~0xFFFF に対応します） dwAo0-4 がチャンネル 0~4 に相当します。
dAo0-4	前記 dwAo0-4 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。 dwAo0-4 が dAo0-4 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)
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)
dwDI_Latch	デジタル入力チャンネル 0~31 のストローラッチ値 (Bit0~Bit31 がデジタル入力チャンネル 0~31 に対応します)
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	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;
};

```

VB

```

Type TConfigPWM
    dwCycleMax0s As Long
    dwCycleMax1s As Long
    dwCycleMax2s As Long
    dwCycleMax3s As Long
    dwCycleMax4s As Long
    dwCycleMax5s As Long
    dwCycleMax6s As Long
    dwCycleMax7s As Long
    dwCycleMax8s As Long
    dwCycleMax9s As Long
    dwCycleMax10s As Long
    dwCycleMax11s As Long
    dwCycleMax12s As Long
    dwCycleMax13s As Long
    dwCycleMax14s As Long
    dwCycleMax15s As Long
    dwPwmPhase0s As Long
    dwPwmPhase1s As Long
    dwPwmPhase2s As Long
    dwPwmPhase3s As Long
    dwPwmPhase4s As Long
    dwPwmPhase5s As Long
    dwPwmPhase6s As Long
    dwPwmPhase7s As Long
    dwPwmPhase8s As Long
    dwPwmPhase9s As Long
    dwPwmPhase10s As Long
    dwPwmPhase11s As Long
    dwPwmPhase12s As Long
    dwPwmPhase13s As Long
    dwPwmPhase14s As Long
    dwPwmPhase15s As Long
    dwStart 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）

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;
};
```

VB

```
Type TSetupPWM
    dwPwm0Value As Long
    dwPwm1Value As Long
    dwPwm2Value As Long
    dwPwm3Value As Long
    dwPwm4Value As Long
    dwPwm5Value As Long
    dwPwm6Value As Long
    dwPwm7Value As Long
    dwPwm8Value As Long
    dwPwm9Value As Long
    dwPwm10Value As Long
    dwPwm11Value As Long
    dwPwm12Value As Long
    dwPwm13Value As Long
    dwPwm14Value As Long
    dwPwm15Value 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）

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;
};
```

VB

```
Type TStatusPack2
    dwBurstMax      As Long
    dwADO_underrun As Long
    dwADI_overrun  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）

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。 UNDERRUN_BUFFER : 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_overrun	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。 OVERRUN_BUFFER : 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(0)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          dwBufferSizeOfWORDsr;
    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) <table border="0"> <tr> <td>ADX II 85-1M-PCI(EX)</td> <td>0</td> <td>ADX II 42-1K-ETHERNET</td> <td>4</td> </tr> <tr> <td>ADX II 14-80M-PCIEX</td> <td>5</td> <td>ADX II 52-1K-ETHERNET</td> <td>6</td> </tr> <tr> <td>DX II 64-1M-PCI</td> <td>0</td> <td></td> <td></td> </tr> </table>	ADX II 85-1M-PCI(EX)	0	ADX II 42-1K-ETHERNET	4	ADX II 14-80M-PCIEX	5	ADX II 52-1K-ETHERNET	6	DX II 64-1M-PCI	0		
ADX II 85-1M-PCI(EX)	0	ADX II 42-1K-ETHERNET	4										
ADX II 14-80M-PCIEX	5	ADX II 52-1K-ETHERNET	6										
DX II 64-1M-PCI	0												
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 II 85-1M-PCI(EX)、DX II 64-1M-PCI 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 II 85-1M-PCI(EX)、DX II 64-1M-PCI 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、IOGEOS SETUP.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 II 14-80M-PCIE X 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 ADX II 52-1K-ETHERNET の拡張ボード形式の信号変換アンプや ADX II 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、IOGEOS SETUP.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)
dAoLsbOffsetEx	アナログ出力値は (D/A 設定値 × dAoLsbLevel + dAoLsbOffset) です。 アナログ出力 CH4~(可変電圧出力)のオフセット電圧(mV)
dwAdoBufMode	アナログ出力値は (D/A 設定値 × dAoLsbLevelEx + dAoLsbOffsetEx) です。 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 II 14-80M-PCIE X 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];
};
```

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）

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(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
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。
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)

※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。

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;
};
```

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	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 ADX II 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	計測開始のミリ秒
dwInrange	入力レンジ(IOGEOSSETUP2. dwAI_Range)