



MULTIFUNCTION-I/O-X2 SERIES  
ADIOX2-API  
REFERENCE  
UPDATE 2011-10-30

SAYA INC.

## 目次

はじめに	3
1.初期化・再初期化	4
2.終了処理	7
3.割り込み	9
4.リングバッファ	10
5.設定	13
6.ステータス	15
7.ポーリング	16
8.校正	18
9. ヘルパ	19
10. FFT	22
11. 画面描画アシスト	23
12. 波形生成	27
13. マルチリモート I/O データロガー	27
14. マルチリモート I/O トレンドグラフ	28
15. マルチリモート I/O リポート	31
16. ハードウェア非依存関数	32
17. 構造体 1	35
18. 構造体 2	71

## はじめに

### ADiox2-API

ADiox2-API は、MultifunctionI/O-X2 シリーズを効率よく使うために開発された API です。これまでの MultifunctionI/O-X シリーズ用の ADiox-API に比べると、API や構造体が合理化、整理され、スマートなコードで設計することが可能です。ADiox2-API は、ハードウェアの制御にとどまらず、計測アプリケーションを構築するのに必要な、FFT、信号調節、波形生成、高速画面描画アシスト、計測ファイル保存、CSV データログ、トレンドグラフなどを統合しており、効率的な開発が可能です。

- ① ハードウェアドライバの設定と計測制御などの中枢機能
- ② 機能設定の保存・読出し・管理
- ③ 信号調節・信号解析(FFT、センサインターフェース、校正、スケーリング、アラーム)
- ④ 高速画面描画アシスト
- ⑤ 計測ファイルのバイナリ高速保存、および計測ファイル CSV 保存(ログ機能)
- ⑥ 波形生成
- ⑦ トrendグラフ
- ⑧ リポート
- ⑨ 複数 **ADX II 42-1K-ETHERNET** のサバイバル機能(生存中のハードウェアを探し動的に計測グループを変えていく)

### スケラビリティ

ADiox2-API は、機種間の差異を吸収できる構造を持っており、**ADX II 14-80M-PCIEX**、**ADX II 85-1M-PCI(EX)**、**ADX II 52-1K-ETHERNET**、**ADX II 42-1K-ETHERNET**、**DX II 64-1M-PCI**をサポートします。対応言語も、C,C++,C#,Basicと広範囲です。

### 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。

### 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 CDRM¥MFIO\_X2¥sdk¥VisuaCPP\_X64

32bit 系の Windows は CDRM¥MFIO\_X2¥sdk¥VisuaCPP\_X86

をお使いください。

ADiox2.h/ADiox2.LIB [ADiox2.dll + ADioxScp.dll ]

メインライブラリのヘッダファイル、インポートライブラリ、ダイナミックリンクライブラリです。

ADioxCsvm.h/ADioxLogm.LIB/[ADioxLogm.dll]

CSV ログ用ヘッダファイル、インポートライブラリ、ダイナミックリンクライブラリです。

ADioxTrlogMI.h/ADioxTrlogMI.LIB/[ADioxTrlogMI.dll]

トレンドグラフ用ヘッダファイル、インポートライブラリ、ダイナミックリンクライブラリです。

ADioxReportMI.h/ADioxReportMI.lib/[ADioxReportMI.dll]

リポート用ヘッダファイル、インポートライブラリ、ダイナミックリンクライブラリです。

#### VisualC#用

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

AdioxLibrary2.cs/[ADiox2.dll+ ADioxScp.dll ]

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

#### 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)

### シンプルなサンプルソース

ADiox2-API の使用方法を理解しやすいよう、ポーリングとバッファの 2 カテゴリで、言語別のサンプルソースを提供しています。コードは非常に短く理解がしやすいように努めています。また ADiox2.dll の機能を網羅した Console2.exe のコードも公開しています。

### CardId について

CardID=0~3 が **ADX II 85-1M-PCI(EX)**、**DX II 64-1M-PCI** に、4-27 が **ADX II 42-1K-ETHERNET** および **ADX II 52-1K-ETHERNET**、28 が **ADX II 14-80M-PCIEX** に割り当てられます。

# 1.初期化・再初期化 [ADiox2.dll]

## vSetupTcpIp

ADX II 42-1K-ETHERNET、ADX II 52-1K-ETHERNET、専用の関数です。IP アドレスおよびポート番号を、CARD\_ID に割り付けます。本関数は、初期化関数をコールする前に実行してください。

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

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

**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。

## bADioxScpLoad2

複数の ADX II 42-1K-ETHERNET、ADX II 52-1K-ETHERNET に対し、ドライバのオープン、ハードウェアの初期化、システムメモリへのバッファ確保を行います。ハードウェアの初期化は、TXBUFSETUP2、IOGEOSSETUP2、TDIO\_MISC、TConfigPWM、TSetupPWM、TADIO2、SCP\_SETUP\_AIALL(センサーモード、ゼロスパン校正位置、ゼロスパン校正係数、スケールリング、アラーム)など全機能に及びます。これらの情報は、SCP\_SETUP2 構造体を格納したコンフィグレーションファイル(拡張子 scp)から取得します。信号調節機能がない場合でも本関数を使うことができます。

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

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

**VB** Declare Function bADioxScpLoad2B Lib "ADiox2.dll" ( ByVal hWnd As Long, \_  
ByRef IpsCharPayload As CharPayloadB, ByVal bOpenDialog As Long ) 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)
	bOpenDialog	コンフィグレーションファイルを ADiox2.dll 内蔵の“ファイルを開くダイアログボックス”で指定する場合、本引数を TRUE(=1)とします。本引数を FALSE(=0)とした場合、IpsCharPayload.IpcConfigFileName で指定します。“ファイルを開くダイアログボックス”の初期フォルダは IpsCharPayload.IpcInitialDir で指定します。
	IpsCharPayload	コンフィグレーションファイル名、もしくは、“ファイル保存ダイアログボックス”のベースフォルダ名を指定する構造体 CharPayloadC(VB 用は CharPayloadB)へのポインタ。

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

## bADioxLoad\_EX3

単一のMultifunctionI/Oに対し、ドライバのオープン、ハードウェアの初期化、メモリ確保、割り込みの使用許可を行います。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
);
```

**VB** Declare Function bADioxLoad\_EX3B Lib "ADiox2.dll" \_

```
( _
    ByVal hWnd As Long, _
    ByVal bWintr As Byte, _
    ByVal bOpenDialog As Long, _
    ByVal IpsSAYA_DEVICE_INFO As SAYA_DEVICE_INFO, _
    ByVal IpsTBUFSETUP As TXBUFSETUP2, _
    ByVal IpsIOGEOSETUP As IOGEOSETUP2, _
    ByVal IpsTDIO_MISC As TDIO_MISC, _
    ByVal IpsTADIO As TADIO2, _
    ByVal IpsTConfigPWM As TConfigPWM, _
    ByVal IpsTSetupPWMM As TSetupPWM, _
    ByVal IpsScpSetupAich As SCP_SETUP_AIALL, _
    ByVal IpsEXTENTION As ADIOX_EXTENTION2B, _
    ByVal dwRingbufferSize As Long, _
    ByVal bCardID As Byte, _
    ByVal IpsCharPayload As CharPayloadB _
) As Long
```

<b>引数</b>	hWnd bHwintr  bOpenDialog  IpsSAYA_DEVICE_INFO IpsTBUFSETUP	<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_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)</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、IpcConfigFileName 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.IpcInitialDir 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。 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>
-----------	---	---

lpsIOGEOSETUP	コンフィグレーションファイルにより反映された IOGEOSETUP2 構造体へのポインタ。
lpsTDIO_MISC	コンフィグレーションファイルにより反映された TDIO_MISC 構造体へのポインタ。
lpsTADIO	コンフィグレーションファイルにより反映された TADIO2 構造体へのポインタ。
lpsTConfigPWM	コンフィグレーションファイルにより反映された TConfigPWM 構造体へのポインタ。
lpsTSetupPWM	コンフィグレーションファイルにより反映された TSetupPWM 構造体へのポインタ。
lpsScpSetupAich	コンフィグレーションファイルにより反映された SCP_SETUP_AIALL 構造体へのポインタ。
lpsEXTENTION	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より反映された ADIOX_EXTENTION2(VB 用は ADIOX_EXTENTION2B)構造体へのポインタ。
dwRingbufferSize	セカンダリリングバッファの倍率。当引数は <b>ADX II 85-1M-PCI(EX)</b> 、 <b>DX II 64-1M-PCI</b> のみ意味があります。
bCardID	ターゲットデバイスのカード ID。
lpsCharPayload	コンフィグレーションファイル名、もしくは、“ファイル保存ダイアログボックス”のベースフォルダ名を指定する構造体 CharPayloadC(VB 用は CharPayloadB)へのポインタ。

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

## vADioxIrqEmuration

**ADX II 42-1K-ETHERNET**、**ADX II 52-1K-ETHERNET** 専用の関数です。イーサネットでは PCI バスのように割り込みがありませんが、ADiox2-API では、ドライバで割り込みエミュレーションを行います。本関数は割り込みエミュレーション機能をディセーブルにして負荷を下げます。割り込みの必要ないポーリング場合に、本関数をご使用ください。必ず初期化前に実行してください。本関数を呼び出さない場合、割り込みエミュレーションが有効になります。

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

**C#** void vADioxIrqEmuration ( int bEnableIrqEmuration );

**VB** Declare Sub vADioxIrqEmuration Lib "ADiox2.dll" (ByVal bEnableIrqEmuration As Long)

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

## bADioxScpRetry

**ADX II 42-1K-ETHERNET**、**ADX II 52-1K-ETHERNET** が電源遮断や通信途絶によってロストした場合、リトライをかけます。(=再接続と初期化を試みる)これは複数のリモート I/O でデータ収集中に、一部のリモート I/O がロストしても、残りのリモート I/O で運用を続行したい場合に有効です。リトライに失敗すると、TCP/IP プロトコルスタックがタイムアウト待ちの為に無応答になるので、頻繁にリトライを行うのは避けてください。

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

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

**VB** Declare Function bADioxScpRetry Lib "ADiox2.dll"  
(ByVal hWnd As Long, ByVal bWintr As Byte, ByVal bCardID As Byte) As Long

**引数** hWnd ドライバからのメッセージを受け取るアプリケーションのウィンドウハンドルを設定します。  
bHwintr 前記メッセージ番号を指定します。本引数 0~15 に対し、メッセージ 3028~3043 番が割り当てられます。この値は Adiox2.h、ADiox2Library.cs、ADIOX-API\_VB.bas にて 3028~3043 番を、それぞれ、WM\_HWINTR0~WM\_HWINTRF として定義しています。(末尾の 0~F は本引数 0~15 の 16 進数に相当する)

bCardID ターゲットデバイスのカード ID

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

## vADioxErrorMessageStop

**ADX II 42-1K-ETHERNET**、**ADX II 52-1K-ETHERNET** の通信エラーに関わるエラーメッセージを全て出さないようにします。例えば bADioxScpRetry を使って複数のリモート I/O のうち生き残っているものだけでデータ収集を行い、リトライで復旧したリモート I/O をデータ収集に参加させるような場合、ロストするときにも、リトライ時失敗時にもエラーメッセージが出てしまいます。本関数はこうしたエラーメッセージを抑制します。

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

**C#** void vADioxErrorMessageStop( int bStop );

**VB** Declare Sub vADioxErrorMessageStop Lib "ADiox2.dll" (ByVal bStop As Long)

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

## bADioxGetBootStatus

ADX II 42-1K-ETHERNET、ADX II 52-1K-ETHERNET が起動できたかどうか確認します。複数の ADX II 42-1K-ETHERNET、ADX II 52-1K-ETHERNET のうち生存中の ADX II 42-1K-ETHERNET、ADX II 52-1K-ETHERNET だけでデータ収集を行いたい場合、途中でロスト・リトライなどが考えられる場合、本関数で起動時にロストしている ADX II 42-1K-ETHERNET、ADX II 52-1K-ETHERNET を確認できます。ロストしている ADX II 42-1K-ETHERNET、ADX II 52-1K-ETHERNET は、bADioxScpRetry で復活させられる可能性があります。

**C/C++** BOOL bADioxGetBootStatus( BYTE bCardID, LPBYTE lpbErrorType );  
**C#** int bADioxGetBootStatus( byte bCardID, ref byte lpbErrorType );  
**VB** Declare Function bADioxGetBootStatus Lib "ADiox2.dll" \_  
 (ByVal bCardID As Byte, ByRef lpbErrorType As Byte) As Long

**引数**

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

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

## 2. 終了処理 [ADiox2.dll]

### bADioxClose

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

**C/C++** BOOL bADioxClose( BYTE bCardID );  
**C#** int bADioxClose( byte bCardID );  
**VB** Declare Function bADioxClose Lib "ADiox2.dll" ( ByVal bCardID As Byte ) As Long

**引数** bCardID ターゲットデバイスのカード ID

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

### bADioxScpStore2

複数の ADX II 42-1K-ETHERNET、ADX II 52-1K-ETHERNET に対し、ドライバのクローズ、ハードウェアの終了処理、メモリ開放を行います。さらに TXBUFSETUP2、IOGEOSSETUP2、TDIO\_MISC、TConfigPWM、TSetupPWM、TADIO2、SCP\_SETUP\_AIALL(センサーモード、ゼロスパン校正位置、ゼロスパン校正係数、スケーリング、アラーム)を SCP\_SETUP2 構造体に格納し、これをコンフィグレーションファイル(拡張子 scp)に保存します。信号調節機能がない場合でも本関数を使うことができます。

**C/C++** BOOL bADioxScpStore2(CharPayloadC \*lpsCharPayload, BOOL bOpenDialog );  
**C#** int bADioxScpStore( ref CharPayloadC lpsCharPayload, int bOpenDialog );  
**VB** Declare Function bADioxScpStore2 Lib "ADiox2.dll"( \_  
 ByRef lpsCharPayload As CharPayloadB, ByVal bOpenDialog As Long ) As Long

**引数**

bOpenDialog	コンフィグレーションファイルの保存先を当関数内蔵の“ファイル保存ダイアログボックス”で指定する場合、TRUE(=1)とします。FALSE(=0)とした場合、lpsCharPayload.LpcConfigFileName で直接ファイルを指定します。“ファイル保存ダイアログボックス”の初期フォルダは lpsCharPayload.lpcInitialDir で指定します。
lpsCharPayload	コンフィグレーションファイル名、もしくは、“ファイル保存ダイアログボックス”のベースフォルダ名を指定する構造体 CharPayloadC(VB 用は CharPayloadB)へのポインタ。

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

## bADioxStore\_EX3

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

**C/C++** BOOL bADioxStore\_EX3

```
(
    BOOL bOpenDialog,
    TXBUFSETUP2 *sTBUFSETUP,
    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 sTBUFSETUP,
    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 IpsTBUFSETUP 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
```

<b>引数</b>	bOpenDialog	コンフィグレーションファイルの保存先を当関数内蔵の“ファイル保存ダイアログボックス”で指定する場 合、本引数を TRUE(=1)とします。FALSE(=0)とした場合、ファイル名を IpsCharPayload. IpcConfigFileName 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 IpsCharPayload.IpcInitialDir で指定します。
	IpsTBUFSETUP	コンフィグレーションファイルに保存したい TXBUFSETUP2 構造体へのポインタ。
	IpsIOGEOSSETUP	コンフィグレーションファイルに保存したい IOGEOSSETUP2 構造体へのポインタ。
	IpsTDIO_MISC	コンフィグレーションファイルに保存したい TDIO_MISC 構造体へのポインタ。
	IpsTADIO	コンフィグレーションファイルに保存したい TADIO2 構造体へのポインタ。
	IpsTConfigPWM	コンフィグレーションファイルに保存したい TConfigPWM 構造体へのポインタ。
	IpsTSetupPWM	コンフィグレーションファイルに保存したい TSetupPWM 構造体へのポインタ。
	IpsScpSetupAich	コンフィグレーションファイルに保存したい SCP_SETUP_AIALL 構造体へのポインタ。
	IpsEXTENTION	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 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。
	IpsCharPayload	コンフィグレーションファイル名、もしくは、“ファイル保存ダイアログボックス”のベースフォルダ名を指定する 構造体 CharPayloadC(VB 用は CharPayloadB)へのポインタ。

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

## 3. 割り込み [ADiox2.dll]

### bADioxInterruptStart

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

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

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

**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 \*lpsIrqbuf, BYTE bCardID );

**C#** int ADioxInterruptStatus( ref IRQ\_BUFFER lpsIrqbuf, byte bCardID );

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

**引数** lpsIrqbuf 割り込み要因を格納した構造体 IRQ\_BUFFER へのポインタを指定します。

bCardID ターゲットデバイスのカード ID。

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

### bADioxMessageCount

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

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

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

**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。

## 4.リングバッファ [\[ADiox2.dll\]](#)

### 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。引数 lpdAiBuffa に物理定数への変換やスケールリングされた値も格納されます。

**C/C++** BOOL bADioxDmaReadEX3

```
(
    LPDWORD lpdwBuffa,
    double * lpdAiBuffa,
    TStatusPack2 *lpsTStatus,
    BYTE bCardID
);
```

**C#** int bADioxDmaReadEX3

```
(
    ref uint lpdwBuffa,
    ref double lpdAiBuffa,
    ref TStatusPack2 lpsTStatus,
    byte bCardID
);
```

**VB** Declare Function bADioxDmaReadEX3 Lib "ADiox2.dll" \_

```
(_
    ByRef lpdwBuffa As Long,_
    ByRef lpdAiBuffa,_
    ByRef lpsTStatus As TStatusPack2,_
    ByVal bCardID As Byte_
) As Long
```

- |            |  |   |
|------------|--|---|
| <b>引数</b>  | lpdwBuffa  | リングバッファのデータのコピー先バッファへのポインタ。 <a href="#">ADX II 14-80M-PCIEX</a> 以外の各機種では、データは Bit31-16 が DI15ch-0ch に、Bit15-0 が AI の 16Bit データに割り付けられます。 <a href="#">ADX II 14-80M-PCIEX</a> では、Bit31-18 が AI1ch、Bit17-16 が DI3-2ch、Bit15-2 が AI0ch、Bit1-0 が DI1-0ch に割り当てられます。(尚 DI の割付をなくすことで、16bit の A/D 変換データの取得も可能です)                                    |
|            | lpdAiBuffa   | 信号調節機能により、物理定数(電圧や温度などの値)に変換されたアナログ値×リングバッファサイズのバッファへのポインタ。詳細は下記内容です。<br><a href="#">ADX II 42-1K-Ethernet</a> ではゼロ校正・スパン校正・スケールリング・リニアライズされた値。<br><a href="#">ADX II 52-1K-Ethernet</a> ではスケールリングされた値<br><a href="#">ADX II 85-1M-PCI(EX)</a> では単に電圧値に変換された値。<br><a href="#">ADX II 14-80M-PCIEX</a> , <a href="#">DX II 64-1M-PCI</a> では無意味 |
|            | lpsTStatus   | システムステータスを格納した、TStatusPack 構造体へのポインタ。   |
|            | bCardID  | ターゲットデバイスのカード ID。   |
| <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>備考</b>  | <p>&lt;<a href="#">ADX II 14-80M-PCIEX</a>,<a href="#">ADX II 42-1K-ETHERNET</a>, <a href="#">ADX II 52-1K-ETHERNET</a>,の場合のバッファサイズの算出方法&gt;<br/>SAYA_DEVICE_INFO.dwBufferSizeOfDWORD そのものです。</p> <p>&lt;<a href="#">ADX II 85-1M-PCI(EX)</a>, <a href="#">DX II 64-1M-PCI</a> の場合のバッファサイズの算出方法&gt;<br/>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。</p> |   |
| <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 II 14-80M-PCIEX</a> から、信号調節機能付きの <a href="#">ADX II 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(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)ヘ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、バッファヘ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, ByVal 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。ADX II 14-80M-PCIEX 以外では、データは Bit31-16 が DO15ch-0ch に、Bit15-0 が AO の 16Bit データに割り付けられます。ADX II 14-80M-PCIEX で DO をバッファ経由とする場合、Bit31-18 が AO1ch、Bit17-16 が DO3-2ch、Bit15-2 が AO0ch、Bit1-0 が DO1-0ch に割り当てられます。ADX II 14-80M-PCIEX で DO をバッファ経由としない場合(ポーリング)、Bit31-16 が AO1ch、Bit15-0 が AO0ch に割り当てられます。
bCardID	ターゲットデバイスのカード ID。

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

**備考** <ADX II 14-80M-PCIEX, ADX II 42-1K-ETHERNET, ADX II 52-1K-ETHERNET, の場合のバッファサイズの算出方法>  
SAYA\_DEVICE\_INFO.dwBufferSizeOfDWORD そのものです。

<ADX II 85-1M-PCI(EX), DX II 64-1M-PCI の場合のバッファサイズの算出方法>

SAYA\_DEVICE\_INFO.dwBufferSizeOfDWORD に、セカンダリリングバッファ倍率を乗算したサイズです。セカンダリリングバッファ倍率は、bADioxRingBufferMode や bADioxLoad\_EX3 の dwRingBufferMode に相当します。

<プリライトについて>

ADX II 14-80M-PCIEX と ADX II 42-1K-ETHERNET, ADX II 52-1K-ETHERNET ではリングバッファ稼動前に 2 バンクのリングバッファと、転送元のソフトウェアバッファを予め書き込んでおく必要があります。すなわち前述したバッファサイズの 3 倍を予め書き込んでおく必要があります。ADX II 85-1M-PCI(EX)、DX II 64-1M-PCI ではセカンダリリングバッファ 2 バンクと、プライマリリングバッファ 1 バンク分を予め書き込んでおく必要があります。前述したバッファサイズを dwBufferSize とした場合、

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

がプリライトサイズとなります。

実際には、このような計算をしなくても、SAYA\_DEVICE\_INFO\_EX.dwPreWriteSizeOfDWORD から同値を取得できます。

## 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" ( ByVal 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。

## bADioxCyclicTrigger

[ADX II 14-80M-PCIEX](#) ではサイクリックトリガ機能が搭載されていますが、本関数では、サイクリックトリガ機能の設定を行います。サイクリックトリガは、オシロスコープをエミュレートするもので、トリガがかかると、本関数の第 2 引数“dwSetupBufferSize”で指定されたサイズを取り込んだ段階でバンクチェンジを行って 1 次停止、ユーザアプリケーションにデータの取得を促すために、割り込みを発生させます。1 次停止したトリガ及びリングバッファは、関数“bADioxCyclicRestart”で再開します。“bADioxCyclicRestart”を割り込み処理の最後に追加することで、あたかもオシロスコープのような動作が可能です。連続データ収集ではないので、最高速度の 80MHz も容易に達成できます。(注: 本関数を使用する場合にはリングバッファ出力を併用しないでください)

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

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

**VB** Declare Function bADioxCyclicTrigger Lib "ADiox2.dll" ( ByVal bCyclicTrigger As Long, \_  
ByVal dwSetupBufferSize As Long, ByVal bCardID As Byte ) As Long

**引数** bCyclicTrigger サイクリックトリガを on にする場合 TRUE、off にする場合 false  
dwSetupBufferSize サイクリックトリガで有効とするバッファサイズ。64~1048576 の値が有効です。  
bCardID ターゲットデバイスのカード ID。

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

## bADioxCyclicRestart

[ADX II 14-80M-PCIEX](#) ではサイクリックトリガ機能が搭載されています。サイクリックトリガは、オシロスコープをエミュレートするもので、トリガがかかると、関数“bADioxCyclicTrigger”の第 2 引数“dwSetupBufferSize”で指定されたサイズを取り込んだ段階でバンクチェンジを行って 1 次停止、ユーザアプリケーションにデータの取得を促すために、割り込みを発生させます。本関数は 1 次停止したトリガ及びリングバッファを再開させるものです。80MHz や 66MHz など高速で稼働させる場合、本関数を割り込み処理の最後に入れると PC の負荷が高くなりすぎて、ユーザアプリケーションのマウスイベントの取得などを阻害します。従って、多少のウェイトを入れてください。(例えば Console2.exe では Sleep(50)を入れている)

**C/C++** BOOL bADioxCyclicRestart ( BYTE bCardID;

**C#** int bADioxCyclicRestart ( byte bCardID );

**VB** Declare Function bADioxCyclicRestart Lib "ADiox2.dll"(ByVal bCardID As Byte ) As Long

**引数** bCardID ターゲットデバイスのカード ID。

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

## 5. 設定 [ADiox2.dll]

### bADioxSetupSymmetryEngine2

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

**C/C++** `BOOL bADioxSetupSymmetryEngine2 ( struct TXBUFSETUP2 *sTBufSetup,  
ADIOX_EXTENTION2 *IpsADIOX_EXTENTION2 ,BYTE bCardID);`

**C#** `int bADioxSetupSymmetryEngine2 ( ref TXBUFSETUP sTBufSetup,  
ref ADIOX_EXTENTION2 IpsADIOX_EXTENTION2 , byte bCardID );`

**VB** `Declare Function bADioxSetupSymmetryEngine_VB2 Lib "ADiox2.dll" ( _  
ByRef IpsTBufSetup As TXBUFSETUP, _  
ByRef ADIOX_EXTENTION2B IpsADIOX_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。

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。

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 IpsIOGEOSETUP2 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 IpsDIO_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。

### 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。  
PWM でアナログ相当の制御を行う場合のインターフェースはここになります。  
bCardID ターゲットデバイスのカード ID。  
**戻り値** 成功すると 1(TRUE)、失敗すると 0(FALSE)が返ります。

## bADioxRingBufferMode

ADX II 85-1M-PCI(EX)、DX II 64-1M-PCI 専用の関数です。2 ステージリングバッファにおける、セカンダリリングバッファサイズを変更します。本機能を使うことで、レスポンスと負荷のバランスを調整します。レスポンスを重視であればバッファサイズを小さく、速度重視(低負荷)にするには(バッファオーバーラン・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。

## bADioxScpSetup2

ADX II 42-1K-ETHERNET, ADX II 52-1K-ETHERNET の信号調節設定(SCP\_SETUP\_AICH 構造体の内容)をハードウェアに反映します。この関数をコールすると、ゼロ、スパン校正が解除されますので、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

ADX II 42-1K-ETHERNET, ADX II 52-1K-ETHERNET の信号調節設定(SCP\_SETUP\_AIALL 構造体の内容)をハードウェアに反映します。この関数をコールしても、ゼロ、スパン校正係数は解除されません。よってセンサ種別を変更した場合には、値が正確ではなくなる可能性がありますのでご注意ください。ゼロ・スパ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

本関数は、[ADX II 42-1K-ETHERNET](#)、[ADX II 52-1K-ETHERNET](#) 用の関数で、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**

<code>dwSensorMode</code>	センサー種別
<code>dwDeviceNumber</code>	ターゲットデバイスのカード ID (bCardID と同じです)。
<code>dwCH</code>	アナログ入力チャンネル番号
<code>lpsSCP_SETUP</code>	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。

## 6. ステータス [ADiox2.dll]

### bADioxStatus2

システムの稼動状態を取得します。

**C/C++** `BOOL 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**

<code>* sTStatus</code>	システムステータスを格納した、TStatusPack 構造体へのポインタ。
<code>bCardID</code>	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 * lpsSAYA_DEVICE_INFO, BYTE bCardID);`

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

**VB** `Declare Sub vADioxDeviceInfoEx Lib "ADiox2.dll" (_  
ByRef lpsSAYA_DEVICE_INFO_EX As SAYA_DEVICE_INFO, ByVal bCardID As Byte)`

**引数**

<code>* lpsSAYA_DEVICE_INFO</code>	デバイス情報構造体 SAYA_DEVICE_INFO へのポインタ。
<code>bCardID</code>	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。

### bADX14input\_type

[ADX II 14-80M-PCIEX](#) におけるアナログ入力がユニポーラかバイポーラかを返します。他機種では本関数に意味がありません。

**C/C++** `BOOL bADX14input_type ( BYTE bCardID );`

**C#** `int bADX14input_type ( byte bCardID );`

**VB** `Declare Function bADX14input_type Lib "ADiox2.dll" (ByVal bCardID As byte ) As Long`

**引数**

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

**戻り値** ユニポーラだと 1(TRUE)、バイポーラだと 0(FALSE)が返ります。

## vADioxScpCopy2

ドライバ内部の SCP\_SETUP2 構造体の内容を取得します。bADioxScpLoad2 が成功した場合、ドライバ内部の SCP\_SETUP2 構造体にコンフィギュレーションファイルの情報が展開されます。失敗した場合にはドライバ内部の SCP\_SETUP 構造体にフォルトのコンフィギュレーション情報が展開されます。本関数はこれら内部状態を取得するものです。信号調節機能がない場合でも本関数を使うことができます。更に、ADiox2-API では、構造体 TXBUFSETUP2、IOGEOSETUP2、TDIO\_MISC、TConfigPWM、TSetupPWM、TADIO2、SCP\_SETUP\_AICH、SCP\_SETUP\_AIALL を引数とする関数にアクセスすると、ドライバ内部の SCP\_SETUP2 構造体が更新されますが、本関数はこうした運用中の内部状態も取得できます。SCP\_SETUP 構造体の内容は bADioxScpStore2 で保存することができます。

```
C/C++ void vADioxScpCopy2 ( SCP_SETUP2 * lpsSCP_SETUP );
C# void vADioxScpCopy_CS2 (ref SCP_SETUP_CS2 lpsSCP_SETUP );
VB Declare Sub vADioxScpCopy_VB2 Lib "ADiox2.dll" ( _
    ByRef lpsSCP_SETUP1 As SCP_SETUP2_PART1, _
    ByRef lpsSCP_SETUP2 As SCP_SETUP2_PART2, _
    ByRef lpsSCP_SETUP3 As SCP_SETUP2_PART3 )
引数 lpsSCP_SETUP          ドライバ内部 SCP_SETUP2 構造体のポインタ。
                                VB では SCP_SETUP2 構造体を 3 分割したポインタとなります。
```

## dwADioxNetError

[ADX II 42-1K-ETHERNET](#), [ADX II 52-1K-ETHERNET](#) の通信エラーを報告します。

```
C/C++ DWORD dwADioxNetError(LPDWORD lpdwWriteRetry,LPDWORD lpdwReadRetry,
    LPDWORD lpdwReadFlameError,LPDWORD lpdwReadAddressError);
C# uint dwADioxNetError(ref uint lpdwWriteRetry, ref uint lpdwReadRetry,
    ref uint lpdwReadFlameError, ref uint lpdwReadAddressError);
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。
```

## 7. ポーリング [ADiox2.dll]

### bADioxADIO2

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

```
C/C++ BOOL bADioxADIO2 ( struct TADIO2 * lpsTADio, BYTE bCardID );
C# int bADioxADIO2 ( ref TADIO2 lpsTADio ,byte bCardID );
VB Declare Function bADioxADIO2 Lib "ADiox2.dll" ( _
    ByRef lpsTADio As TADIO2, ByVal bCardID As Byte ) As Long
引数 *lpsTADio          アナログ入出力・デジタル入出力・エンコーダカウンタ・周波数カウンタ・温度の値を格納した TADIO2
                                構造体へのポインタ。アナログ入出力を電圧値に変換した値も格納されます。メンバ変数により入力と
                                出力に分かれます。
    bCardID              ターゲットデバイスのカード ID。
戻り値 成功すると 1(TRUE)、失敗すると 0(FALSE)が返ります。
```

### bADioxChannelAIw2

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 IOGEOSETUP2 *lpsIoGeoSetup,
    LPDWORD lpdwData, int iInterval, BYTE bCardID);
C# int bADioxChannelAIw2 ( ref IOGEOSETUP2 lpsIoGeoSetup,
    ref uint lpdwData, int iInterval, byte bCardID);
```

<b>VB</b>	Declare Function bADioxChannelAIw2 Lib "ADiox2.dll" (ByRef lpsIoGeoSetup As IOGEOSETUP2, _ ByRef lpdwData As Long, ByVal iInterval As Integer, ByVal bCardID As Byte ) As Long
<b>引数</b>	<p>lpsIoGeoSetup      アナログデジタル入出力インターフェース設定値が入った IOGEOSETUP2 構造体を指定します。 取得したいアナログチャンネルもここで指定します。</p> <p>lpdwData 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。</p> <p>bCardID            ターゲットデバイスのカード ID。</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。

## bADioxDI

デジタル入力値を取得します。

<b>C/C++</b>	BOOL bADioxDI(LPDWORD lpdwDI, BYTE bCardID);
<b>C#</b>	int bADioxDI (ref uint lpdwDI, byte bCardID);
<b>VB</b>	Declare Function bADioxDI Lib "ADiox2.dll" (ByRef lpdwDI As Long, ByVal bCardID As Byte) As Long
<b>引数</b>	<p>lpdwDI             デジタル入力値を保持したポインタ。</p> <p>bCardID            ターゲットデバイスのカード ID。</p>
<b>戻り値</b>	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。

<b>C/C++</b>	BOOL bADioxDO ( DWORD dwDO, BYTE bCardID );
<b>C#</b>	int bADioxDO ( uint dwDO, byte bCardID );
<b>VB</b>	Declare Function bADioxDO "ADiox2.dll" ( ByVal DWORD As Long, ByVal bCardID As Byte ) As Long
<b>引数</b>	<p>dwDO                デジタル出力値。</p> <p>bCardID            ターゲットデバイスのカード ID。</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。

## 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、スパン校正、リニアライザ、スケーリング、アラーム、バーンアウト検出などの処理を行い、結果を返します。設定内容はドライバ内部の SCP\_SETUP 構造体から、bCardID, bChannel に相当する部分を取り出して適用します。

<b>C/C++</b>	double dADioxLinCoef(DWORD dwANALOG, LPBOOL lpbBurnOut, LPDWORD lpdwAlarm, BYTE bCardID, BYTE bChannel);
<b>C#</b>	double dADioxLinCoef(uint dwANALOG, ref int lpbBurnOut, ref uint lpdwAlarm, byte bCardID, byte bChannel);
<b>VB</b>	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
<b>引数</b>	<p>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。</p> <p>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。</p> <p>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。</p> <p>bCardID            ターゲットデバイスのカード ID。</p> <p>bChannel           アナログ入力チャンネル番号</p>
<b>戻り値</b>	変換されたアナログ値

## 8.校正 [ADiox2.dll]

### bADioxAutoCal2

ADX II 85-1M-PCI(EX)のダイナミック校正(アナログ入力)の校正を開始させます。本関数により校正スレッドがドライバ内部で起動し、本関数は処理の終了を待たずに直ちにリターンします。校正スレッドの終了までは dwADioxAutoCal\_Status1 を除く ADiox2API 関数群は一切呼び出してはいけません。校正の終了は dwADioxAutoCal\_Status1 で知ることができます。

**C/C++** `BOOL bADioxAutoCal2(struct IOGEOSSETUP2 *IpsIOGEOSSETUP,  
struct TXBUFSETUP2 *IpsTXBUFSETUP, BYTE bCardID);`

**C#** `int bADioxAutoCal2( ref IOGEOSSETUP2 IpsIOGEOSSETUP,  
ref TXBUFSETUP IpsTXBUFSETUP, byte bCardID);`

**VB** `Declare Function bADioxAutoCal2 Lib "ADiox2.dll" (ByRef IpsIOGEOSSETUP As IOGEOSSETUP2, _  
ByRef IpsTXBUFSETUP 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 IOGEOSSETUP2 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

ADX II 85-1M-PCI(EX)用の関数で、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" ( ByVal 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。

### bADioxZeroAdj

本関数は、ADX II 42-1K-ETHERNET 用の関数で、指定したカード ID、チャンネル番号におけるゼロ校正(=オフセット校正)を行います。ゼロ校正位置は、bADioxScpSetup または bADioxScpSetupEX で設定します。本関数を実施する前に、対象となるアナログ入力に(ハードウェアに)ゼロ基準値を与えてください。より正確な校正を行うには、ゼロ校正とスポン校正を繰り返してください。

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

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

**VB** `Declare Function bADioxZeroAdj Lib "ADiox2.dll"(ByVal bCardID As Byte, _  
ByVal bChannel As Byte ) As Long`

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

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

### bADioxSpanAdj

本関数は、ADX II 42-1K-ETHERNET 用の関数で、指定したカード ID、チャンネル番号におけるスパン校正(=ゲイン校正)を行います。スパン校正位置は、bADioxScpSetup または bADioxScpSetupEX で設定します。本関数を実施する前に、対象となるアナログ入力に(ハードウェアに)スパン基準値を与えてください。より正確な校正を行うには、ゼロ校正とスポン校正を繰り返してください。

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

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

**VB** `Declare Function bADioxSpanAdj Lib "ADiox2.dll" ( ByVal bCardID As Byte, _  
ByVal bChannel As Byte) As Long`

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

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

## vADioxFreeAdj

本関数は、[ADX II 42-1K-ETHERNET](#) 用の関数で、指定したカードID、チャンネル番号におけるゼロ校正量を0に、スパン校正量を1にします。すなわちこれは、校正をかけない状態と同じです。

```
C/C++ void vADioxFreeAdj ( BYTE bCardID, BYTE bChannel );
C# void vADioxFreeAdj ( byte bCardID, byte bChannel );
VB Declare Sub vADioxFreeAdj Lib "ADiox2.dll" ( ByVal bCardID As Byte, ByVal bChannel As Byte)
引数 bCardID          ターゲットデバイスのカード ID。
      bChannel        アナログ入力チャンネル番号
```

## bADioxAutoZero

本関数は、[ADX II 42-1K-ETHERNET](#) 用の関数です。bADioxZeroAdj や bADioxSpanAdj はデータ収集前に校正をかけるものですが本関数によるオートゼロ(自動ゼロ校正)は、データ収集中でもゼロ校正を行い、オフセット(ゼロ誤差)をキャンセルします。本関数を定期的呼び出すことで、温度変動の大きな環境や、長時間の運用で、計測精度が低下することを防ぐことが出来ます。本関数を使わない場合、熱電対 K 型 1300℃レンジで約 6℃程度の誤差が生じることがあります。本関数は比較的負荷が重いので頻繁には使わないでください。

```
C/C++ BOOL bADioxAutoZero( BYTE bCardID );
C# int bADioxAutoZero(byte bCardID );
VB Declare Function bADioxAutoZero Lib "ADiox2.dll"( ByVal bCardID As Byte ) As Long
引数 bCardID          ターゲットデバイスのカード ID。
戻り値 成功すると 1(TRUE)、失敗すると 0(FALSE)が返ります。
```

## 9. ヘルパ [\[ADiox2.dll\]](#)

### 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" ( \_  
ByVal 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。

## iADioxCheckBits

引数 dwData の、bChannel 指定のビットフィールドが 1 か 0 かを返します。DI の各ビットフィールドが On か Off かを簡単にチェックできます。

**C/C++** int iADioxCheckBits( DWORD dwDATA, BYTE bChannel );

**C#** int iADioxCheckBits ( uint dwDATA, byte bChannel );

**VB** Declare Function iADioxCheckBits Lib "ADiox2.dll" ( ByVal dwDATA As Long, \_  
ByVal bChannel As Byte ) As Integer

**引数** dwDATA 調べたい 32Bit データ  
bChannel ビットフィールドを指定します。値は 0-31 まで有効です。

**戻り値** dwDATA の bChannel で示されるビットフィールドが 1 の場合には 1 を、0 の場合には 0 を返します。

## dwADioxSrcStatus

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

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

**C#** uint dwADioxSrcStatus (uint dwCommand );

**VB** Declare Function dwADioxSrcStatus Lib "ADiox2.dll" (ByVal dwCommand As Long ) As Long

**引数** dwCommand 0 を指定してください。

**戻り値** エラー発生回数が返ります。

## bADioxChangeScaler

本関数は、[ADX II 42-1K-ETHERNET](#)、[ADX II 52-1K-ETHERNET](#) 用の関数で、信号調節機能のスケーリングを計測中に変更します。騒音計や振動計など同じ入力値でも支指示値が変更されるような機器に有効です。dBeforeTop～dBeforeBottom(スケーリング前の値)を、dAfterTop～dAfterBottom にスケーリングします。

**C/C++** BOOL bADioxChangeScaler ( BYTE bCardID, BYTE bCH, double dAfterTop, double dAfterBottom,  
double dBeforeTop, double dBeforeBottom );

**C#** int bADioxChangeScaler ( byte bCardID, byte bCH, double dAfterTop, double dAfterBottom,  
double dBeforeTop, double dBeforeBottom );

**VB** Declare Function bADioxChangeScaler Lib "ADiox2.dll" (ByVal bCardID As Byte, ByVal bCH As Byte, \_  
ByVal dAfterTop As Double, ByVal dAfterBottom As Double, ByVal dBeforeTop As Double, \_  
ByVal dBeforeBottom As Double) As Long

**引数** bCardID ターゲットデバイスのカード ID。  
bCH アナログ入力チャンネル番号  
dAfterTop dBeforeTop の値(mV/°C)を変換後に幾つにするか  
dAfterBottom dBeforeBottom の値(mV/°C)を変換後に幾つにするか  
dBeforeTop dAfterTop に変換される元の値。  
dBeforeBottom dAfterBottom に変換される元の値。

**戻り値** 成功すると 1(TRUE)、失敗すると 0(FALSE)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([ADX II 14-80M-PCIEX](#) では MHz、[ADX II 85-1M-PCI\(EX\)](#)、[DX II 64-1M-PCI](#)、[ADX II 42-1K-ETHERNET](#)、[ADX II 52-1K-ETHERNET](#) では KHz)

## dADioxAdxFullScall

フルスケール電圧を返します。

**C/C++** double dADioxAdxFullScall ( BYTE bCardID,BOOL bFiles,  
DWORD dwFileRange,DWORD dwFileHardwareType );  
**C#** double dADioxAdxFullScall ( byte bCardID , int bFiles , uint dwFileRange , uint dwFileHardwareType );  
**VB** Declare Function dADioxAdxFullScall Lib "ADiox2.dll" (ByVal bCardID As Byte, ByVal bFiles As Long, \_  
ByVal dwFileRange As Long, ByVal dwFileHardwareType As Long) As Double  
**引数** bCardID ターゲットデバイスのカード ID。  
bFiles ファイルの場合 TRUE(=1)、ライブ計測の場合 FALSE(=0)  
dwFileRange ファイルの場合、ファイルヘッダの LOG\_FRONTEND.dwInrange を指定します、  
ライブ計測の場合本引数は使われません。  
dwFileHardwareType ファイルの場合、ファイルヘッダの sLOG\_FRONTEND.dwDeviceName を指定、  
ライブ計測の場合 SAYA\_DEVICE\_INFO.dwDeviceType を指定してください。  
**戻り値** フルスケール電圧が返ります。バイポーラの場合には上限-下限となり±10V レンジの場合、20 が返ります。

## bADioxAdxMaxCH

チャンネル数を返します。[ADX II 14-80M-PCIEX](#) では 1 を返します。

**C/C++** BYTE bADioxAdxMaxCH ( BYTE bCardID );  
**C#** int bADioxAdxMaxCH ( byte bCardID );  
**VB** Declare Function bADioxAdxMaxCH Lib "ADiox2.dll" (ByVal bCardID As Byte) As Byte  
**引数** bCardID ターゲットデバイスのカード ID。  
**戻り値** チャンネル数を返します。[ADX II 14-80M-PCIEX](#) では 1 を返します。

## bADioxAdxMaxCH2

チャンネル数を返します。[ADX II 14-80M-PCIEX](#) では 2 を返します。

**C/C++** BYTE bADioxAdxMaxCH2 ( BYTE bCardID );  
**C#** int bADioxAdxMaxCH2 ( byte bCardID );  
**VB** Declare Function bADioxAdxMaxCH2 Lib "ADiox2.dll" (ByVal bCardID As Byte) As Byte  
**引数** bCardID ターゲットデバイスのカード ID。  
**戻り値** チャンネル数を返します。[ADX II 14-80M-PCIEX](#) では 2 を返します。

## dADioxAdxTriggerPosition

[ADX II 85-1M-PCI\(EX\)](#)、[ADX II 14-80M-PCIEX](#) で入力電圧の dPercent(%)に相当する値を返します。[ADX II 42-1K-ETHERNET](#)、[ADX II 52-1K-ETHERNET](#) では dPercent(%)がそのまま返ります。

**C/C++** double dADioxAdxTriggerPosition ( double dPercent, BYTE bCardID );  
**C#** double dADioxAdxTriggerPosition ( double dPercent , byte bCardID );  
**VB** Declare Function dADioxAdxTriggerPosition Lib "ADiox2.dll" (ByVal dPercent As Double , \_  
ByVal bCardID As Byte) As Double  
**引数** dPercent 100 分率で割合を指定する(%)。  
bCardID ターゲットデバイスのカード ID。  
**戻り値** 入力電圧の dPercent(%)に相当する値を返します。[ADX II 42-1K-ETHERNET](#)、[ADX II 52-1K-ETHERNET](#) では dPercent(%)がそのまま返ります。

## bADioxClockMode

ADX II 85-1M-PCI(EX)、DX II 64-1M-PCI で 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。

## 10.FFT [ADiox2.dll]

### bADioxFft\_Config

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

**C/C++** `BOOL bADioxFft_Config( int iPoint, int iWindow );`  
**C#** `Int bADioxFft_Config( int iPoint, int iWindow );`  
**VB** `Declare Function bADioxFft_Config Lib "ADiox2.dll" ( ByVal iPoint As Integer, _  
ByVal iWindow As Integer) As Long`

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

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

### dADioxFft\_PreScall

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

**C/C++** `double dADioxFft_PreScall ( WORD wGetSample, BOOL bCOB);`  
**C#** `double dADioxFft_PreScall(ushort wGetSample, int bCOB);`  
**VB** `Declare Function dADioxFft_PreScall Lib "ADiox2.dll" ( _  
ByVal wGetSample As Long, ByVal bCOB As Long) As Double`

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

**戻り値** 変換された結果(アナログ値)が返ります。

### bADioxFft\_Analyze

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

**C/C++** `BOOL bADioxFft_Analyze(double * dFreqOut, double * dDataIn);`  
**C#** `int bADioxFft_Analyze(ref double dFreqOut, ref double dDataIn);`  
**VB** `Declare Function bADioxFft_Analyze Lib "ADiox2.dll" ( _  
ByRef dFreqOut As Double, ByRef dDataIn As Double) As Long`

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

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

## 11. 描画アシスト [ADiox2.dll]

### vADioxInitializeViewArea

波形描画,FFT 描画,DI 波形描画アシスト機能を初期化します。

**C/C++** BOOL vADioxInitializeViewArea(DWORD wDiViewerSize\_Y);

**C#** int vADioxInitializeViewArea ( uint dwDiViewerSize\_Y );

**VB** Declare Function vADioxInitializeViewArea Lib "ADiox2.dll" (ByVal dwDiViewerSize\_Y As Long) As Long

引数 wDiViewerSize\_Y DI 波形描画領域の垂直サイズ。  
DI 波形描画をしない場合には 1000 程度の値を指定してください。

### bADioxGenerateViewAreaAI

アナログ入力計測データ、各種パラメタを与え、波形描画イメージ、FFT 描画イメージを作成します。複雑な波形描画、FFT 描画を簡潔に記述することが可能です。本関数による画面描画を割り込みコールバック関数に記述するとバッファオーバーラン・バッファアンダーランのリスクが高まります。Console2.exe では割り込みコールバック関数からスレッドを作成し、このスレッド内部で本関数による画面描画を行ない、スレッドが終了しない間は次のスレッドを生成しないようにすることで負荷に応じて画面描画の回数を調整しています。これによりバッファオーバーラン・バッファアンダーランのリスクを低減します。

**C/C++** BOOL bADioxGenerateViewAreaAI(DWORD dwDeviceType,LPDWORD dwReadBuffa,  
DWORD dwBufferSize, BYTE bMaxAiCh,DWORD dwViewerSize\_X, DWORD dwViewerSize\_Y,  
double dY\_Offset,double dY\_Span,DWORD dwX\_Offset,DWORD dwX\_Span,  
DWORD dwFFTSize\_X,DWORD dwFFTSize\_Y,LPDWORD lpdwPlotStart,  
LPDWORD lpdwPlotStop,LPBOOL lpbFftAnalyzeComplete,LPDWORD dwCh\_00\_Data,  
LPDWORD dwCh\_01\_Data,LPDWORD dwCh\_02\_Data,LPDWORD dwCh\_03\_Data,  
LPDWORD dwCh\_04\_Data,LPDWORD dwCh\_05\_Data,LPDWORD dwCh\_06\_Data,  
LPDWORD dwCh\_07\_Data,LPDWORD dwCh\_08\_Data,LPDWORD dwCh\_09\_Data,  
LPDWORD dwCh\_10\_Data,LPDWORD dwCh\_11\_Data,LPDWORD dwCh\_12\_Data,  
LPDWORD dwCh\_13\_Data,LPDWORD dwCh\_14\_Data,LPDWORD dwCh\_15\_Data,  
int \*iCh\_00\_Fft,int \*iCh\_01\_Fft,int \*iCh\_02\_Fft,int \*iCh\_03\_Fft,int \*iCh\_04\_Fft,  
int \*iCh\_05\_Fft,int \*iCh\_06\_Fft,int \*iCh\_07\_Fft,int \*iCh\_08\_Fft,int \*iCh\_09\_Fft,  
int \*iCh\_10\_Fft,int \*iCh\_11\_Fft,int \*iCh\_12\_Fft,int \*iCh\_13\_Fft,int \*iCh\_14\_Fft,  
int \*iCh\_15\_Fft);

**C#** int bADioxGenerateViewAreaAI ( uint dwDeviceType,ref uint dwReadBuffa,uint dwBufferSize,  
byte bMaxAiCh,uint dwViewerSize\_X,uint dwViewerSize\_Y,double dY\_Offset,  
double dY\_Span,uint dwX\_Offset,uint dwX\_Span,uint dwFftPoint,uint dwFFTSize\_X,  
uint dwFFTSize\_Y,ref uint lpdwPlotStart,ref uint lpbFftAnalyzeComplete,  
ref uint dwCh\_00\_Data,ref uint dwCh\_01\_Data,ref uint dwCh\_02\_Data,  
ref uint dwCh\_03\_Data,ref uint dwCh\_04\_Data,ref uint dwCh\_05\_Data,  
ref uint dwCh\_06\_Data,ref uint dwCh\_07\_Data,ref uint dwCh\_08\_Data,  
ref uint dwCh\_09\_Data,ref uint dwCh\_10\_Data,ref uint dwCh\_11\_Data,  
ref uint dwCh\_12\_Data,ref uint dwCh\_13\_Data,ref uint dwCh\_14\_Data,  
ref uint dwCh\_15\_Data,ref int iCh\_00\_Fft,ref int iCh\_01\_Fft,ref int iCh\_02\_Fft,  
ref int iCh\_03\_Fft,ref int iCh\_04\_Fft,ref int iCh\_05\_Fft,ref int iCh\_06\_Fft,ref int iCh\_07\_Fft,  
ref int iCh\_08\_Fft,ref int iCh\_09\_Fft,ref int iCh\_10\_Fft,ref int iCh\_11\_Fft,ref int iCh\_12\_Fft,  
ref int iCh\_13\_Fft,ref int iCh\_14\_Fft,ref int iCh\_15\_Fft );

**VB** Declare Function bADioxGenerateViewAreaAI Lib "ADiox2.dll" ( ByVal dwDeviceType As Long, \_  
ByRef dwReadBuffa As Long, ByVal dwBufferSize As Long,ByVal bMaxAiCh As Byte, \_  
ByVal dwViewerSize\_X As Long, ByVal dwViewerSize\_Y As Long, ByVal dY\_Offset As Double, \_  
ByVal dY\_Span As Double, ByVal dwX\_Offset As Long, ByVal dwX\_Span As Long, \_  
ByVal dwFftPoint As Long, ByVal dwFFTSize\_X As Long, ByVal dwFFTSize\_Y As Long, \_  
ByRef lpdwPlotStart As Long, ByRef lpdwPlotStop As Long, \_  
ByRef lpbFftAnalyzeComplete As Long, ByRef dwCh\_00\_Data As Long, \_  
ByRef dwCh\_01\_Data As Long, ByRef dwCh\_02\_Data As Long, \_  
ByRef dwCh\_03\_Data As Long, ByRef dwCh\_04\_Data As Long, \_  
ByRef dwCh\_05\_Data As Long, ByRef dwCh\_06\_Data As Long, \_  
ByRef dwCh\_07\_Data As Long, ByRef dwCh\_08\_Data As Long, \_  
ByRef dwCh\_09\_Data As Long, ByRef dwCh\_10\_Data As Long, \_  
ByRef dwCh\_11\_Data As Long, ByRef dwCh\_12\_Data As Long, \_  
ByRef dwCh\_13\_Data As Long, ByRef dwCh\_14\_Data As Long, \_  
ByRef dwCh\_15\_Data As Long, ByRef iCh\_00\_Fft As Integer, \_  
ByRef iCh\_01\_Fft As Integer, ByRef iCh\_02\_Fft As Integer, ByRef iCh\_03\_Fft As Integer, \_  
ByRef iCh\_04\_Fft As Integer, ByRef iCh\_05\_Fft As Integer, ByRef iCh\_06\_Fft As Integer, \_  
ByRef iCh\_07\_Fft As Integer, ByRef iCh\_08\_Fft As Integer, ByRef iCh\_09\_Fft As Integer, \_  
ByRef iCh\_10\_Fft As Integer, ByRef iCh\_11\_Fft As Integer, ByRef iCh\_12\_Fft As Integer, \_  
ByRef iCh\_13\_Fft As Integer, ByRef iCh\_14\_Fft As Integer, ByRef iCh\_15\_Fft As Integer \_  
) As Long

<b>引数</b>	dwDeviceType	機種コード。ファイルの場合、ファイルヘッダの sLOG_FRONTEND.dwDeviceName を指定、ライブ計測の場合 SAYA_DEVICE_INFO.dwDeviceType を指定してください。
	dwReadBufFa	計測データを格納した入力データバッファ。bADioxDmaReadEX2 や bADioxReadScpBuf2 の dwReadBufFa、読み出したファイルのデータバッファを指定してください。
	dwBufferSize	前記入力データバッファサイズ
	bMaxAiCh	最大 AI チャンネル
	dwViewerSize_X	波形描画領域の水平サイズ
	dwViewerSize_Y	波形描画領域の垂直サイズ
	dY_Offset	波形描画位置の垂直オフセットを 1.0~0.0 で指定します。0.5 で波形はセンタに位置します。1.0 と 0.0 で波形描画領域の端に波形を配置します。
	dY_Span	波形描画の垂直スケールを指定します。1.0 で波形フルスケールが、波形描画領域の垂直サイズに収まります。値を大きくするとその分拡大していきます。拡大してもセンタ位置が維持されますので、dY_Offset は純粋に垂直移動分を与えることになります。
	dwX_Offset	波形描画位置の水平オフセットを 0 以上で指定します。本値はサンプル数ですので、10 であれば 10 サンプルが水平位置の端となります。チャンネルシーケンスは内部で自動計算されるので考慮する必要がありません。
	dwX_Span	波形描画の水平スケールを指定します。値は間引き量で 0 で間引きなしになります。チャンネルシーケンスは内部で自動計算されるので考慮する必要がありません。
	dwFftPoint	FFT ポイント数を指定します。
	dwFFTSize_X	FFT 描画領域の水平サイズ
	dwFFTSize_Y	FFT 描画領域の垂直サイズ
	lpdwPlotStart	波形描画開始位置を返します。入力数が画面数に足りず、波形が全描画できない場合、波形描画領域のどこからどこを描画するかが返ります。波形描画、波形の画面打点位置を格納したバッファの双方で本変数を参照してください。
	lpdwPlotStop	波形描画停止位置を返します。入力数が画面数に足りず、波形が全描画できない場合、波形描画領域のどこからどこを描画するかが返ります。波形描画、波形の画面打点位置を格納したバッファの双方で本変数を参照してください。
	lpbFftAnalyzeComplete	FFT 計算が成功した場合 TRUE(=1)、失敗もしくはデータ数が足りない場合など FALSE(=0)が返ります。
	dwCh_00_Data~dwCh_15_Data	波形の画面打点位置を格納したバッファ。00~15 はチャンネルを現します。本変数のアドレス(配列位置)が波形描画領域の X 軸本変数の値は波形描画領域の Y 軸に相当します。
	iCh_00_Fft~iCh_15_Fft	FFT の画面打点位置を格納したバッファ。00~15 はチャンネルを現します。本変数のアドレス(配列位置)が FFT 描画領域の X 軸本変数の値は FFT 描画領域の Y 軸に相当します。
<b>戻り値</b>	成功すると 1(TRUE)、失敗すると 0(FALSE)が返ります。	

## bADioxGenerateViewAreaDI

デジタル入力計測データ、各種パラメタを与え、DI 波形描画イメージを作成します。複雑な波形描画を簡潔に記述することが可能です。本関数による画面描画を割り込みコールバック関数に記述するとバッファオーバーラン・バッファアンダーランのリスクが高まります。Console2.exe では割り込みコールバック関数からスレッドを作成し、このスレッド内部で本関数による画面描画を行ない、スレッドが終了しない間は次のスレッドを生成しないようにすることで負荷に応じて画面描画の回数を調整しています。これによりバッファオーバーラン・バッファアンダーランのリスクを低減します。

<b>C/C++</b>	<pre> BOOL bADioxGenerateViewAreaDI(DWORD dwDeviceType,LPDWORD dwReadBuffa,     DWORD dwBufferSize,BYTE bMaxAiCh,DWORD dwViewerSize_X,DWORD dwX_Offset,     DWORD dwX_Span,LPBYTE lpbDiPaintMode,LPDWORD lpdwPlotStart,LPDWORD lpdwPlotStop,     LPDWORD dwCh_00_Data,LPDWORD dwCh_01_Data,LPDWORD dwCh_02_Data,     LPDWORD dwCh_03_Data,LPDWORD dwCh_04_Data,LPDWORD dwCh_05_Data,     LPDWORD dwCh_06_Data,LPDWORD dwCh_07_Data,LPDWORD dwCh_08_Data,     LPDWORD dwCh_09_Data,LPDWORD dwCh_10_Data,LPDWORD dwCh_11_Data,     LPDWORD dwCh_12_Data,LPDWORD dwCh_13_Data,LPDWORD dwCh_14_Data,     LPDWORD dwCh_15_Data); </pre>																						
<b>C#</b>	<pre> int bADioxGenerateViewAreaDI ( uint dwDeviceType,ref uint dwReadBuffa,uint dwBufferSize,     byte bMaxAiCh,uint dwViewerSize_X,uint dwX_Offset,uint dwX_Span,     ref byte lpbDiPaintMode,ref uint lpdwPlotStart,ref uint lpdwPlotStop,ref uint dwCh_00_Data,     ref uint dwCh_01_Data,ref uint dwCh_02_Data,ref uint dwCh_03_Data,     ref uint dwCh_04_Data,ref uint dwCh_05_Data,ref uint dwCh_06_Data,     ref uint dwCh_07_Data,ref uint dwCh_08_Data,ref uint dwCh_09_Data,     ref uint dwCh_10_Data,ref uint dwCh_11_Data,ref uint dwCh_12_Data,     ref uint dwCh_13_Data,ref uint dwCh_14_Data,ref uint dwCh_15_Data ); </pre>																						
<b>VB</b>	<pre> Declare Function bADioxGenerateViewAreaDI Lib "ADiox2.dll" ( ByVal dwDeviceType As Long, _     ByVal dwReadBuffa As Long, ByVal dwBufferSize As Long, ByVal bMaxAiCh As Byte, _     ByVal dwViewerSize_X As Long, ByVal dwX_Offset As Long, ByVal dwX_Span As Long, _     ByVal lpbDiPaintMode As Byte, ByVal lpdwPlotStart As Long, ByVal lpdwPlotStop As Long, _     ByVal dwCh_00_Data As Long, ByVal dwCh_01_Data As Long, _     ByVal dwCh_02_Data As Long, ByVal dwCh_03_Data As Long, _     ByVal dwCh_04_Data As Long, ByVal dwCh_05_Data As Long, _     ByVal dwCh_06_Data As Long, ByVal dwCh_07_Data As Long, _     ByVal dwCh_08_Data As Long, ByVal dwCh_09_Data As Long, _     ByVal dwCh_10_Data As Long, ByVal dwCh_11_Data As Long, _     ByVal dwCh_12_Data As Long, ByVal dwCh_13_Data As Long, _     ByVal dwCh_14_Data As Long, ByVal dwCh_15_Data As Long) As Long </pre>																						
<b>引数</b>	<table border="0"> <tr> <td>dwDeviceType</td> <td>機種コード。ファイルの場合、ファイルヘッダの sLOG_FRONTEND.dwDeviceName を指定、ライブ計測の場合 SAYA_DEVICE_INFO.dwDeviceType を指定してください。</td> </tr> <tr> <td>dwReadBuffa</td> <td>計測データを格納した入力データバッファ。bADioxDmaReadEX2やbADioxReadScpBuf2のdwReadBuffa、読み出したファイルのデータバッファを指定してください。</td> </tr> <tr> <td>dwBufferSize</td> <td>前記入力データバッファサイズ</td> </tr> <tr> <td>bMaxAiCh</td> <td>最大 AI チャンネル</td> </tr> <tr> <td>dwViewerSize_X</td> <td>DI 波形描画領域の水平サイズ(DI 波形描画領域の垂直サイズは vADioxInitializeViewArea で指定します)</td> </tr> <tr> <td>dwX_Offset</td> <td>DI 波形描画位置の水平オフセットを 0 以上で指定します。本値はサンプル数ですので、10 であれば 10 サンプルが水平位置の端となります。チャンネルシーケンスは内部で自動計算されるので考慮する必要がありません。</td> </tr> <tr> <td>dwX_Span</td> <td>DI 波形描画の水平スケールを指定します。値は間引き量で 0 で間引きなしになります。チャンネルシーケンスは内部で自動計算されるので考慮する必要がありません。</td> </tr> <tr> <td>lpbDiPaintMode,</td> <td>DI チャンネルの描画数、0 で無描画、1 で 0-3ch 描画、2 で 0-15ch 描画します。</td> </tr> <tr> <td>lpdwPlotStart</td> <td>DI 波形描画開始位置を返します。入力数が画面数に足りず、波形が全描画できない場合、DI 波形描画領域のどこからどこを描画するかが返ります。DI 波形描画、波形の画面打点位置を格納したバッファの双方で本変数を参照してください。</td> </tr> <tr> <td>lpdwPlotStop</td> <td>DI 波形描画停止位置を返します。入力数が画面数に足りず、波形が全描画できない場合、DI 波形描画領域のどこからどこを描画するかが返ります。DI 波形描画、波形の画面打点位置を格納したバッファの双方で本変数を参照してください。</td> </tr> <tr> <td>dwCh_00_Data~dwCh_15_Data</td> <td>DI 波形の画面打点位置を格納したバッファ。00~15 はチャンネルを現します。本変数のアドレス(配列位置)が波形描画領域の X 軸本変数の値は波形描画領域の Y 軸に相当します。</td> </tr> </table>	dwDeviceType	機種コード。ファイルの場合、ファイルヘッダの sLOG_FRONTEND.dwDeviceName を指定、ライブ計測の場合 SAYA_DEVICE_INFO.dwDeviceType を指定してください。	dwReadBuffa	計測データを格納した入力データバッファ。bADioxDmaReadEX2やbADioxReadScpBuf2のdwReadBuffa、読み出したファイルのデータバッファを指定してください。	dwBufferSize	前記入力データバッファサイズ	bMaxAiCh	最大 AI チャンネル	dwViewerSize_X	DI 波形描画領域の水平サイズ(DI 波形描画領域の垂直サイズは vADioxInitializeViewArea で指定します)	dwX_Offset	DI 波形描画位置の水平オフセットを 0 以上で指定します。本値はサンプル数ですので、10 であれば 10 サンプルが水平位置の端となります。チャンネルシーケンスは内部で自動計算されるので考慮する必要がありません。	dwX_Span	DI 波形描画の水平スケールを指定します。値は間引き量で 0 で間引きなしになります。チャンネルシーケンスは内部で自動計算されるので考慮する必要がありません。	lpbDiPaintMode,	DI チャンネルの描画数、0 で無描画、1 で 0-3ch 描画、2 で 0-15ch 描画します。	lpdwPlotStart	DI 波形描画開始位置を返します。入力数が画面数に足りず、波形が全描画できない場合、DI 波形描画領域のどこからどこを描画するかが返ります。DI 波形描画、波形の画面打点位置を格納したバッファの双方で本変数を参照してください。	lpdwPlotStop	DI 波形描画停止位置を返します。入力数が画面数に足りず、波形が全描画できない場合、DI 波形描画領域のどこからどこを描画するかが返ります。DI 波形描画、波形の画面打点位置を格納したバッファの双方で本変数を参照してください。	dwCh_00_Data~dwCh_15_Data	DI 波形の画面打点位置を格納したバッファ。00~15 はチャンネルを現します。本変数のアドレス(配列位置)が波形描画領域の X 軸本変数の値は波形描画領域の Y 軸に相当します。
dwDeviceType	機種コード。ファイルの場合、ファイルヘッダの sLOG_FRONTEND.dwDeviceName を指定、ライブ計測の場合 SAYA_DEVICE_INFO.dwDeviceType を指定してください。																						
dwReadBuffa	計測データを格納した入力データバッファ。bADioxDmaReadEX2やbADioxReadScpBuf2のdwReadBuffa、読み出したファイルのデータバッファを指定してください。																						
dwBufferSize	前記入力データバッファサイズ																						
bMaxAiCh	最大 AI チャンネル																						
dwViewerSize_X	DI 波形描画領域の水平サイズ(DI 波形描画領域の垂直サイズは vADioxInitializeViewArea で指定します)																						
dwX_Offset	DI 波形描画位置の水平オフセットを 0 以上で指定します。本値はサンプル数ですので、10 であれば 10 サンプルが水平位置の端となります。チャンネルシーケンスは内部で自動計算されるので考慮する必要がありません。																						
dwX_Span	DI 波形描画の水平スケールを指定します。値は間引き量で 0 で間引きなしになります。チャンネルシーケンスは内部で自動計算されるので考慮する必要がありません。																						
lpbDiPaintMode,	DI チャンネルの描画数、0 で無描画、1 で 0-3ch 描画、2 で 0-15ch 描画します。																						
lpdwPlotStart	DI 波形描画開始位置を返します。入力数が画面数に足りず、波形が全描画できない場合、DI 波形描画領域のどこからどこを描画するかが返ります。DI 波形描画、波形の画面打点位置を格納したバッファの双方で本変数を参照してください。																						
lpdwPlotStop	DI 波形描画停止位置を返します。入力数が画面数に足りず、波形が全描画できない場合、DI 波形描画領域のどこからどこを描画するかが返ります。DI 波形描画、波形の画面打点位置を格納したバッファの双方で本変数を参照してください。																						
dwCh_00_Data~dwCh_15_Data	DI 波形の画面打点位置を格納したバッファ。00~15 はチャンネルを現します。本変数のアドレス(配列位置)が波形描画領域の X 軸本変数の値は波形描画領域の Y 軸に相当します。																						
<b>戻り値</b>	成功すると 1(TRUE)、失敗すると 0(FALSE)が返ります。																						

## dwADioxGetBufferStartAddress

リングバッファ、もしくはファイル読み出し位置先頭を計算するヘルパ。VB 関数は 32Bit ファイルサイズ制限があります。

<b>C/C++</b>	DWORDLONG dwADioxGetBufferStartAddress(DWORD dwDeviceType, BYTE bMaxAiCh, DWORDLONG dwlDataSize, DWORD dwViewerSize_X, DWORD dwX_Span, DWORD dwFileOffset, DWORD dwHposCurrent, DWORD dwHposMax);	
<b>C#</b>	ulong dwADioxGetBufferStartAddress ( uint dwDeviceType, byte bMaxAiCh, ulong dwlDataSize, uint dwViewerSize_X, uint dwX_Span, uint dwFileOffset, uint dwHposCurrent, uint dwHposMax );	
<b>VB</b>	Declare Function dwADioxGetBufferStartAddress_VB Lib "ADiox2.dll" (ByVal dwDeviceType As Long, _ ByVal bMaxAiCh As Byte, ByVal dwDataSize As Long, ByVal dwViewerSize_X As Long, _ ByVal dwX_Span As Long, ByVal dwFileOffset As Long, ByVal dwHposCurrent As Long, _ ByVal dwHposMax As Long) As Long	
<b>引数</b>	dwDeviceType	機種コード。ファイルの場合、ファイルヘッダの LOG_FRONTEND.dwDeviceName を指定、ライブ計測の場合 SAYA_DEVICE_INFO.dwDeviceType を指定してください。
	bMaxAiCh	最大 AI チャンネル
	dwlDataSize	最大データサイズ(ファイルならファイルサイズ、ライブならリングバッファサイズ)
	dwViewerSize_X	波形描画領域の水平サイズ
	dwX_Span	波形描画の水平スケールを指定します。値は間引き量で 0 で間引きなしになります。チャンネルシーケンスは内部で自動計算されるので考慮する必要がありません。
	dwFileOffset	ファイルの場合の、データ先頭位置。即ちファイルヘッダを除く部分。Console2.exe では LOG_FRONTEND がヘッダになります。
	dwHposCurrent	水平位置(スクロールバーの値をそのまま入力します。MFC ならスクロールバーオブジェクトを GetPos したもの)
	dwHposMax	水平最大値(スクロールバーの値の最大値。MFC ならスクロールバーオブジェクトの SetRange の値)
<b>戻り値</b>	読み出し位置が返ります。ファイルの場合、この値までシークしてデータを取り出し bADioxGenerateViewAreaAI、bADioxGenerateViewAreaDI にデータを与えてください。ファイルの場合、既にシークしているので dwX_Offset は 0 でよい。(Console2.exe ではメモリを節約するため間引きも行つて bADioxGenerateViewAreaAI、bADioxGenerateViewAreaDI にデータを渡しますゆえに dwX_Span も 0 です)ライブ計測の場合、この戻り値を bADioxGenerateViewAreaAI、bADioxGenerateViewAreaDI の dwX_Offset に与えてください。	

## dADioxGetGridText

水平グリッド位置の指示値を返します。波形描画領域のグリッド指示値は、垂直スクロール・垂直スケーリング(拡大)などにより複雑に変動します。本関数は垂直スクロール、垂直スケーリング、入力レンジ、機種などによりグリッド指示値を自動的に計算します。

<b>C/C++</b>	double dADioxGetGridText (DWORD dwDeviceType, DWORD dwAiRange85, BOOL bAiRange14, double dY_Offset, double dY_Span, double dLinePosition );	
<b>C#</b>	double dADioxGetGridText(uint dwDeviceType, uint dwAiRange85, int bAiRange14, double dY_Offset, double dY_Span, double dLinePosition);	
<b>VB</b>	Declare Function dADioxGetGridText Lib "ADiox2.dll" (ByVal dwDeviceType As Long, _ ByVal dwAiRange85 As Long, ByVal bAiRange14 As Long, ByVal dY_Offset As Double, _ ByVal dY_Span As Double, ByVal dLinePosition As Double) As Double	
<b>引数</b>	dwDeviceType	機種コード。ファイルの場合、ファイルヘッダの LOG_FRONTEND.dwDeviceName を指定、ライブ計測の場合 SAYA_DEVICE_INFO.dwDeviceType を指定してください。
	dwAiRange85	<a href="#">ADX II 85-1M-PCI(EX)</a> における入力レンジ (=IOGEOSETUP2.dwAI_Range)
	bAiRange14	<a href="#">ADX II 14-80M-PCIEX</a> における入力レンジ。TRUE(=1) でユニポーラ、FALSE(=0) でバイポーラ。(bADX14input_type で取得できます)
	dY_Offset	波形描画位置の垂直オフセットを 1.0~0.0 で指定します。0.5 で波形はセンタに位置します。1.0 と 0.0 で波形描画領域の端に波形を配置します。
	dY_Span	波形描画の垂直スケールを指定します。1.0 で波形フルスケールが、波形描画領域の垂直サイズに取まります。値を大きくするとその分拡大していきます。拡大してもセンタ位置が維持されますので、dY_Offset は純粋に垂直移動分を与えることになります。
	dLinePosition	グリッドの位置を 1.0~0.0 で指定します。波形描画領域中央に描画されたグリッドであれば 0.5 を指定します。
<b>戻り値</b>	グリッドの指示値を返します。	

## 12. 波形生成 [ADiox2.dll]

任意の振幅・オフセット・周波数で sin, cos, exp, sqrt, triangle, Lamp, Square, dc, step, file\_load 波形を連続生成します。

### bADioxWaveInit

波形ジェネレータを初期化します。

```
C/C++  BOOL bADioxWaveInit ( struct ADIOX_EXTENTION2 *lpsADIOX_EXTENTION2,
                          double *dFrequency0, double *dFrequency1, BYTE bCardID );
C#      int bADioxWaveInit ( ref ADIOX_EXTENTION2 lpsADIOX_EXTENTION2 ,
                          ref double dFrequency0 , ref double dFrequency1 , byte bCardID );
VB      Declare Function bADioxWaveInit Lib "ADiox2.dll" ( _
                          ByRef lpsADIOX_EXTENTION2 As ADIOX_EXTENTION2 , _
                          ByRef dFrequency0 As Double , ByRef dFrequency1 As Double , _
                          ByVal bCardID As Byte) As Long
```

**引数**

lpsADIOX_EXTENTION2	ファイル保存、ファイル読み出し情報の入った ADIOX_EXTENTION2 構造体を指定します。
dFrequency0	AO0 の周波数
dFrequency1	AO1 の周波数 (ADX II 14-80M-PCIEX のみ有効)
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。リングバッファを前提にしているため、[ADX II 85-1M-PCI\(EX\)](#)、[ADX II 42-1K-ETHERNET](#)、[ADX II 52-1K-ETHERNET](#) では 1CH、[ADX II 14-80M-PCIEX](#) は 2CH の波形を生成します。DI は 1Bit シフト波形(DI0 が 1 になり、DI1⇒DI2 と 1 である DIch が移動する)となります。

```
C/C++  BOOL bADioxWaveGenerate ( LPDWORD dwWriteBuffa, DWORD dwBufferSize, BYTE bCardID );
C#      int bADioxWaveGenerate ( ref uint dwWriteBuffa , uint dwBufferSize , byte bCardID );
VB      Declare Function bADioxWaveGenerate Lib "ADiox2.dll" ( ByRef dwWriteBuffa As Long ,
                          ByVal dwBufferSize As Long , ByVal bCardID As Byte ) As Long
```

**引数**

lpdwBuffa	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 lpdwBuffa 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。

## 13. マルチリモート I/O データロガー [ADioxLogm.dll]

本関数群は複数の [ADX II 42-1K-ETHERNET](#)、[ADX II 52-1K-ETHERNET](#) の計測データを CSV ファイルに保存します。カウンタの計測データを AI8-11 としてアナログ信号のように保存することができます。

### bADioxLoggerInit

データロガー(CSV ファイル書き出し機能)の初期化を行います。

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

**引数** sADIOX\_CSV\_FORMAT\_EX 設定構造体 ADIOX\_CSV\_FORMAT\_EX を指定します。

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

### bADioxLoggerWrite

データロガー(CSV ファイル書き出し機能)の実行関数です。引数のバッファに格納された計測データを、CSV ファイルに変換します。

```
C/C++  BOOL bADioxLoggerWrite ( double *dAI_Buffa, WORD *wDI_Buffa,
                               SYSTEMTIME *lpsSysTime, BOOL *bBO_Buffa, BYTE bCardID );
```

**引数**

dAI_Buffa	アナログ値を格納したバッファへのポインタ。
wDI_Buffa	デジタル値を格納したバッファへのポインタ。
lpsSysTime	ファイル生成時刻を格納した SYSTEMTIME 構造体 (Win32) へのポインタ。
bBO_Buffa	バーンアウト値を格納したバッファへのポインタ。バーンアウトしたアナログ入力チャンネルの表示は -- になります。
bCardID	保存するデータのカード ID。

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

## bADioxLoggerLastSize

書き込みを行うサイズを変更します。これが CSV ファイルの 1 シート(1 ファイル)分になります。主に、データログ作業終了時に端数の残留データを書き出す時の書き出しサイズ設定として使います。

**C/C++** BOOL bADioxLoggerLastSize(WORD dwBuffaSize);

**引数** dwBuffaSize 1 回の書き込みを行うサイズを指定します。これが CSV ファイルの 1 シート(1 ファイル)分になります。

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

## vADioxAddBackup

任意のフォルダへバックアップを行います。ネットワークドライブなどにも対応しています

**C/C++** void vADioxAddBackup(BOOL bEnable,char \*cBackupDirectory);

**引数** BOOL bEnable バックアップを行う場合に TRUE(=1)にします。  
char \*cBackupDirectory バックアップを行うディレクトリ(フルパス)

# 14. マルチリモート I/O トレンドグラフ[ADioxTrlogMI.dll]

本関数群は複数の [ADX II 42-1K-ETHERNET](#), [ADX II 52-1K-ETHERNET](#) の計測データに対応したトレンドグラフ機能で、トレンドグラフ保存、ヒストリカルトレンド、バーンアウト処理、トレンドグラフおよびヒストリカルトレンドグラフからの測定値の逆引きなど主要な機能を装備します。カウンタの計測データを AI8-11 としてアナログ信号のように扱うことができます。

## bADioxInitializeGraphicBufferMI

トレンドグラフ機能の初期化を行います。

**C/C++** BOOL bADioxInitializeGraphicBufferMI(struct ADIOX\_TREND\_SETUP sADIOX\_TREND\_SETUP);

**引数** sADIOX\_TREND\_SETUP 設定構造体 ADIOX\_TREND\_SETUP を指定します。

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

## bADioxWriteGraphicBufferMI

トレンドグラフ用バッファへのデータの書き込みを行います。画面イメージのバッファに書き込まれ、X 方向が時間軸、Y 方向がスケールされた計測値になります。X 方向左端から右端へ向かって打点していき、画面右端まで行くと、全体が左にシフト(スクロール)しながら、最も右側だけが更新されます。Y 方向の打点位置は内部で自動的にスケールされるので、dADioxLinCoef や bADioxReadScpBuf2 の結果をそのまま本関数にセットすることができます。結果は bADioxReadGraphicDataMI から読み出せます。同時に本関数を呼び出す都度、ヒストリカルトレンド用のログ作成(tif ファイル)の追記を行います。

**C/C++** BOOL bADioxWriteGraphicBufferMI(  
struct ADIOX\_SET\_TREND\_DATA sADIOX\_SET\_TREND\_DATA,BOOL bSetpixelOfTIME );

**引数** sADIOX\_SET\_TREND\_DATA 計測データを保持する ADIOX\_SET\_TREND\_DATA 構造体を指定します。  
bSetpixelOfTIME 時間情報を表示するか否かを指定します。  
TRUE(=1)で表示、FALSE(=0)で非表示です。

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

## bADioxReadGraphicBufferMI

トレンドグラフ用バッファから 1 つのデータの読み出しを行います。WM\_PAINT イベントなどが発生して全ピクチャボックスの再描画が必要な場合のみ、本関数の X 座標をインクリメントしつつ、グラフのプロット位置を連続で読み出し、ピクチャボックスに描画してください。通常は全体を描画せずに、次の bADioxReadGraphicBufferAuto 関数で更新する場所のみを読み出したほうがシステムの負荷を抑えることが出来ます。

**C/C++** BOOL bADioxReadGraphicBufferMI (int iXlocation,int \*lpiSetpixel,LPBOOL lpbBurnOut,  
LPBOOL lpbSetpixelOfTIME,SYSTEMTIME \*lpsSysTime,BYTE bCardID );

**引数** iXlocation X 軸の位置を指定します。  
lpiSetpixel 上記 X 軸にプロットする Y 軸の位置を読み出せます。  
lpbBurnOut バーンアウトフラグを格納したポインタ。この値が TRUE(=1)ならプロットすべきではありません。  
lpbSetpixelOfTIME 時刻フラグを格納したポインタ。この値が TRUE なら時刻も描画すべきです。  
lpsSysTime 前記引数が TRUE であった場合の時刻を格納した SYSTEMTIME 構造体へのポインタ  
bCardID トレンドを表示したいのカード ID(内部では複数の CARD\_ID のトレンドデータが混在しているのでこの引数で選択する。)

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

## bADioxReadGraphicBufferAutoMI

bADioxReadGraphicBufferMI は X 軸の位置を指定してプロットしていましたが、本関数ではどの X 軸を更新すべきかは、ADIOX-API 内部で行われます。すなわちトレンドグラフを開始させると、画面右端に達するまでプロットして、右端に達すると全体を BitBlt 関数などでスクロールさせます。これら X 座標を取得することができます。但し座標の更新は 1X 座標だけなので、画面が重なったり、最小化から最大化したときなどは、bADioxReadGraphicBufferMI で全体を再描画しないと、トレンドグラフが消失してしまいます。

```
C/C++ BOOL bADioxReadGraphicBufferAutoMI(int *lpiSetpixel,LPBOOL bBurnOut,
LPBOOL lpbSetpixelOFTIME,SYSTEMTIME *lpsSysTime,int *iXlocation,BYTE bCardId );
```

<b>引数</b>	lpiSetpixel bBurnOut lpbSetpixelOFTIME lpsSysTime iXlocation  bCardID	プロットすべき Y 軸の位置を読み出せます。 バーンアウトフラグを格納したポインタ。この値が TRUE(=1)ならプロットすべきではありません。 時刻フラグを格納したポインタ。この値が TRUE(=1)なら時刻も描画すべきです。 前記引数が TRUE(=1)であった場合の時刻を格納した SYSTEMTIME 構造体へのポインタ プロットすべき X 軸の位置を読み出せます。この値が右端に達したら、画面を BitBlt 関数などで 1 ピクセルスクロールさせてください。*lpiSetpixel を画面右端に描画するのはその後行ってください。 トレンドを表示したいのカード ID (内部では複数の CARD_ID のトレンドデータが混在しているのでこの引数で選択する。)
-----------	---	--

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

## bADioxGetBufferDepth

bADioxReadGraphicBufferAutoMI 関数の引数を\*iXlocation のみにしたものです。画面が重なったり、最小化から最大化したときなど、bADioxReadGraphicBufferMI でどこまで描画すべきかを取得します。

```
C/C++ BOOL bADioxGetBufferDepthMI(int *iXlocation);
```

<b>引数</b>	iXlocation	プロットすべき X 軸の位置を読み出せます。
-----------	------------	------------------------

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

## bADioxReadGraphicDataMI

トレンドグラフ画面の X 軸に対する、計測値(Y 値)を呼び出します。マウスでトレンドをクリックした時など、座標をこの関数で与えて、その位置の計測値を読み出します。

```
C/C++ BOOL bADioxReadGraphicDataMI(int iXlocation,
double *lpdAnalogData,LPBOOL bBurnOut,BYTE bCardId);
```

<b>引数</b>	iXlocation lpdAnalogData bBurnOut bCardID	調べたいトレンドグラフの X 座標 アナログデータの計測値へのポインタ。チャンネル数分のデータが並びます。 アナログデータのバーンアウトフラグへのポインタ。チャンネル数分のデータが並びます。 トレンドを表示したいのカード ID (内部では複数の CARD_ID のトレンドデータが混在しているのでこの引数で選択する。)
-----------	--	--

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

## bADioxExitGraphicBufferMI

トレンドグラフ停止。画面処理は行わずヒストリカルトレンド用のログファイルフラッシュのみ行われます。

```
C/C++ BOOL bADioxExitGraphicBufferMI ( );
```

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

## bADioxOpenTrendLogMI

ヒストリカルトレンドのファイルオープン。画面サイズやヒストリカルトレンドログのチャンネル数などの基本構造は、同じになっていることが前提です。チャンネル数と画面サイズを固定パラメータにしたソフトウェアではないと、ファイルの授受は行えません。

```
C/C++ BOOL bADioxOpenTrendLogMI(char * szFile,BOOL bOpenDialog,
RECT pPpbRectSetup,int *lpiBlockSize);
```

<b>引数</b>	szFile bOpenDialog  pPpbRectSetup  lpiBlockSize	ヒストリカルトレンドのファイルのフルパス。 ヒストリカルトレンドのファイルをファイルを開くダイアログでオープンする場合 TRUE(=1)、前記引数でオープンする場合 FALSE(=0)を指定してください。 画面サイズを格納した、RECT 構造体(Win32/MFC の構造体)。この変数は bADioxInitializeGraphicBufferEX の pPpbRectSetup と同じである必要があります。 ファイルに含まれるチャンネルあたりのトータルサンプル数。すなわちチャンネル数(dwCHMAX) × 本引数(*lpiBlockSize)が全データ量になります。
-----------	--	---

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

## bADioxReadTrendLogMI

ヒストリカルトレンドの特定位置の読出し。

**C/C++** BOOL bADioxReadTrendLog(int iSeekPoint,LPBOOL lpbFinish, BYTE bCardId);

**引数** iSeekPoint 読み出したい場所を設定。すなわち先頭からのチャンネルあたりのサンプル数。  
lpbFinish データ最後の識別フラグへのポインタ。もうこれ以上データが無い場合に TRUE(=1)になります。  
bCardID トレンドを表示したいのカード ID (内部では複数の CARD\_ID のトレンドデータが混在しているのでこの引数で選択する。)

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

## bADioxCloseTrendLogMI

ヒストリカルトレンドのファイルクローズ。

**C/C++** BOOL bADioxCloseTrendLogMI ( );

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

## bADioxCheckValidChMI

ヒストリカルトレンドにおいて、引数で指定した CARD\_ID のチャンネルがトレンドに保存されているか確認する。この関数により I/O の使用条件などが異なる場合でもヒストリカルトレンドを運用できる。

**C/C++** BOOL bADioxCheckValidChMI ( BYTE bCH, BYTE bCardId );

**引数** bCH 確認したいチャンネル番号 (アナログ 0-7、カウンタ 8-11)  
bCardId 確認したい CARD\_ID 番号

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

## bADioxCheckValidIoMI

ヒストリカルトレンドにおいて、引数で指定した CARD\_ID がトレンドに保存されているか確認する。この関数により I/O の使用条件などが異なる場合でもヒストリカルトレンドを運用できる。

**C/C++** BOOL bADioxCheckValidIoMI ( BYTE bCardId );

**引数** bCardId 確認したい CARD\_ID 番号

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

## bADioxTrendLogModeMI

トレンドのログを残さない場合に本関数で引数を FALSE にします。本関数をコールしなければ基本的にトレンドログが残ります。

**C/C++** BOOL bADioxTrendLogModeMI(BOOL bLogMode);

**引数** bLogMode ログを残さない場合 FALSE(=0)、残す場合 TRUE(=1)

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

## bADioxHistDirMI

bADioxOpenTrendLogMI で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を指定します。

**C/C++** bADioxHistDirMI(DWORD dwSetInitialDir, char \*FolderName);

**引数** dwSetInitialDir 0 を指定するとパスは無効になります。有効にするには 9 か 12 より大きい数値を指定して下さい。  
FolderName ファイルを開くダイアログのデフォルトフォルダ。dwSetInitialDir が 9 又は 12 以上で有効です。

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

## 15. マルチリモート I/O レポート [ADioxReportMI.dll]

本関数群は積算レポート機能を提供します。アナログ値は過去データの平均、カウンタ値は過去データの積算を行い、これを 1 日、1 月、過去累計の 3 項目にてまとめてレポート(CSV ファイル)を出力します。カウンタの計測データを AI8-11 としてアナログ信号のように扱うことができます。

### bADioxInitializeReport

積算レポート機能を使用する際の初期化を行います。

**C/C++** `BOOL bADioxInitializeReport(struct ADIOX_SET_REPORT_IN sADIOX_SET_REPORT_IN, struct ADIOX_CSV_FORMAT_EX sADIOX_CSV_FORMAT_IN);`

**引数** `sADIOX_SET_REPORT_IN` 初期化設定を行います。  
`sADIOX_CSV_FORMAT_IN` 初期化設定を行います。

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

### bADioxWriteReport

積算レポート機能に新しいデータを積算して再計算します。本関数呼び出しの際、日にちが更新されていればレポート(CSV ファイル)を出力します。同時に `lpbResetRequest` が TRUE になりますので、カウンタをリセットして下さい。

**C/C++** `BOOL bADioxWriteReport(struct ADIOX_SET_TREND_DATA sADIOX_SET_TREND_DATA, LPBOOL lpbResetRequest, BOOL bDebugMonth, BOOL bDebugDaily);`

**引数** `sADIOX_SET_TREND_DATA` 計測データを保持する `ADIOX_SET_TREND_DATA` 構造体を指定します。通常トレンド関数 `bADioxWriteGraphicBufferMI` で使った値をそのまま流用します。レポート(CSV ファイル)を出力した場合 TRUE になります。この時は、日にちが変わった事を示すので、カウンタもリセットしてください。本引数が FALSE ならデータを積算しただけです。  
`lpbResetRequest` テスト用。TRUE にすると月更新が実施されます。通常は FALSE にして下さい。  
`bDebugMonth` テスト用。TRUE にすると月更新が実施されます。通常は FALSE にして下さい。  
`bDebugDaily` テスト用。TRUE にすると日更新が実施されます。通常は FALSE にして下さい。

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

### bADioxStoreReport

`bADioxInitializeReport` によって、積算データを読み込んで以降は、積算データはメモリ上にて更新されていきます。本関数は、メモリ上の積算情報をファイルに保存します。このファイルは専用のバイナリファイルで `bADioxInitializeReport` で再ロードすることになります。

**C/C++** `BOOL bADioxStoreReport();`

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

### vADioxReadReport

現在の積算データを読み出します。

**C/C++** `void vADioxReadReport(struct REPORT_PACK *lpsREPORT_PACK);`

**引数** `lpsREPORT_PACK` 積算データを保持する `REPORT_PACK` 構造体へのポインタ。

### vADioxAddBackup2

任意のフォルダへバックアップを行います。ネットワークドライブなどにも対応しています。CSV ログ機能の `vADioxAddBackup` と同じ作用です。

**C/C++** `void vADioxAddBackup2(BOOL bEnable, char *cBackupDirectory);`

**引数** `BOOL bEnable` バックアップを行う場合に TRUE(=1)にします。  
`cBackupDirectory` バックアップを行うディレクトリ(フルパス)

### bADioxResetReport

現在の積算データをリセットします。実際にバイナリファイルに反映するには `bADioxStoreReport` を実行する必要があります。

**C/C++** `BOOL bADioxResetReport();`

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

## 16. ハードウェア非依存関数 [ADiox2.dll]

ハードウェアなしでも動く関数です。設定ファイルやデバイス情報をハードウェアなしで読み出し、書き込み可能です。

### vADioxLoadFile\_EX3 [New]

設定ファイルを読み込んで引数の各構造体にアサインします。設定ファイルが不正もしくは無い場合には、デフォルト値が適用され、引数の構造体にアサインされます。

<b>C/C++</b>	void vADioxLoadFile_EX3 ( BOOL bOpenDialog , SAYA_DEVICE_INFO_EX * lpsSAYA_DEVICE_INFO , TXBUFSETUP2 *lpsTXBUFSETUP , IOGEOSETUP2 *lpsIOGEOSETUP , TDIO_MISC *lpsTDIO_MISC , TADIO2 *lpsTADIO , TConfigPWM *lpsTConfigPWM , TSetupPWM *lpsTSetupPWM , SCP_SETUP_AIALL * lpsScpSetupAich , ADIOX_EXTENTION2 * lpsEXTENTION , CharPayloadC *lpsCharPayload,DWORD dwDeviceType );																																				
<b>C#</b>	void vADioxLoadFile_EX3 ( int bOpenDialog, ref SAYA_DEVICE_INFO2 lpsSAYA_DEVICE_INFO, ref TXBUFSETUP2 lpsTXBUFSETUP, ref IOGEOSETUP2 lpsIOGEOSETUP, ref TDIO_MISC lpsTDIO_MISC, ref TADIO2 lpsTADIO, ref TConfigPWM lpsTConfigPWM, ref TSetupPWM lpsTSetupPWM, ref SCP_SETUP_AIALL lpsScpSetupAich, ref ADIOX_EXTENTION2 lpsEXTENTION, ref CharPayloadC lpsCharPayload, uint dwDeviceType );																																				
<b>VB</b>	Declare Function vADioxLoadFile_EX3B Lib "ADiox2.dll" _ ( _ ByVal bOpenDialog As Long, _ ByRef lpsSAYA_DEVICE_INFO2 As SAYA_DEVICE_INFO, _ ByRef lpsTXBUFSETUP As TXBUFSETUP2, _ ByRef lpsIOGEOSETUP As IOGEOSETUP2, _ ByRef lpsTDIO_MISC As TDIO_MISC, _ ByRef lpsTDIO As TADIO2, _ ByRef lpsTConfigPWM As TConfigPWM, _ ByRef lpsTSetupPMM As TSetupPWM, _ ByRef lpsScpSetupAich As SCP_SETUP_AIALL, _ ByRef lpsEXTENTION As ADIOX_EXTENTION2B, _ ByRef lpsCharPayload As CharPayloadB, _ ByVal dwDeviceType As Long, _ )																																				
<b>引数</b>	<table border="0"> <tbody> <tr> <td>bOpenDialog</td> <td>コンフィグレーションファイルを当関数内蔵の“ファイルを開くダイアログボックス”で指定する場合、本引数を TRUE(=1)とします。FALSE(=0)とした場合、lpsCharPayload.lpcConfigFileName でファイルを直接指定します。 “ファイルを開くダイアログボックス”の初期フォルダは lpsCharPayload.lpcInitialDir で指定します。</td> </tr> <tr> <td>lpsSAYA_DEVICE_INFO</td> <td>デバイス情報構造体 SAYA_DEVICE_INFO2 へのポインタ。bADioxLoad_EX3 では SAYA_DEVICE_INFO ですが、本関数では SAYA_DEVICE_INFO2 となります。</td> </tr> <tr> <td>lpsTXBUFSETUP</td> <td>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。</td> </tr> <tr> <td>lpsIOGEOSETUP</td> <td>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。</td> </tr> <tr> <td>lpsTDIO_MISC</td> <td>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。</td> </tr> <tr> <td>lpsTADIO</td> <td>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。</td> </tr> <tr> <td>lpsTConfigPWM</td> <td>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。</td> </tr> <tr> <td>lpsTSetupPWM</td> <td>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。</td> </tr> <tr> <td>lpsScpSetupAich</td> <td>コンフィグレーションファイルにより反映された SCP_SETUP_AIALL 構造体へのポインタ。 この引数は信号調節機能付き MultifunctionI/O でのみ意味があります。</td> </tr> <tr> <td>lpsEXTENTION</td> <td>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。</td> </tr> <tr> <td>dwRingbufferSize</td> <td>セカンダリリングバッファの倍率。当引数は <b>ADX II 85-1M-PCI(EX)</b>、<b>DX II 64-1M-PCI</b> でのみ意味があります。</td> </tr> <tr> <td>lpsCharPayload</td> <td>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。</td> </tr> <tr> <td>dwDeviceType</td> <td>デバイスの種類を設定してください。  <table border="0"> <tbody> <tr> <td><b>ADX II 85-1M-PCI(EX)</b></td> <td>0</td> </tr> <tr> <td><b>DX II 64-1M-PCI</b></td> <td>0</td> </tr> <tr> <td><b>ADX II 42-1K-ETHERNET</b></td> <td>4</td> </tr> <tr> <td><b>ADX II 14-80M-PCIE X</b></td> <td>5</td> </tr> <tr> <td><b>ADX II 52-1K-ETHERNET</b></td> <td>6</td> </tr> </tbody> </table> </td> </tr> </tbody> </table>	bOpenDialog	コンフィグレーションファイルを当関数内蔵の“ファイルを開くダイアログボックス”で指定する場合、本引数を TRUE(=1)とします。FALSE(=0)とした場合、lpsCharPayload.lpcConfigFileName でファイルを直接指定します。 “ファイルを開くダイアログボックス”の初期フォルダは lpsCharPayload.lpcInitialDir で指定します。	lpsSAYA_DEVICE_INFO	デバイス情報構造体 SAYA_DEVICE_INFO2 へのポインタ。bADioxLoad_EX3 では SAYA_DEVICE_INFO ですが、本関数では SAYA_DEVICE_INFO2 となります。	lpsTXBUFSETUP	コンフィグレーションファイルにより反映された TXBUFSETUP2 構造体へのポインタ。	lpsIOGEOSETUP	コンフィグレーションファイルにより反映された IOGEOSETUP2 構造体へのポインタ。	lpsTDIO_MISC	コンフィグレーションファイルにより反映された TDIO_MISC 構造体へのポインタ。	lpsTADIO	コンフィグレーションファイルにより反映された TADIO2 構造体へのポインタ。	lpsTConfigPWM	コンフィグレーションファイルにより反映された TConfigPWM 構造体へのポインタ。	lpsTSetupPWM	コンフィグレーションファイルにより反映された TSetupPWM 構造体へのポインタ。	lpsScpSetupAich	コンフィグレーションファイルにより反映された SCP_SETUP_AIALL 構造体へのポインタ。 この引数は信号調節機能付き MultifunctionI/O でのみ意味があります。	lpsEXTENTION	コンフィグレーションファイルにより反映された ADIOX_EXTENTION2(VB 用は ADIOX_EXTENTION2B)構造体へのポインタ。	dwRingbufferSize	セカンダリリングバッファの倍率。当引数は <b>ADX II 85-1M-PCI(EX)</b> 、 <b>DX II 64-1M-PCI</b> でのみ意味があります。	lpsCharPayload	コンフィグレーションファイル名、もしくは、“ファイル保存ダイアログボックス”のベースフォルダ名を指定する構造体 CharPayloadC(VB 用は CharPayloadB)へのポインタ。	dwDeviceType	デバイスの種類を設定してください。 <table border="0"> <tbody> <tr> <td><b>ADX II 85-1M-PCI(EX)</b></td> <td>0</td> </tr> <tr> <td><b>DX II 64-1M-PCI</b></td> <td>0</td> </tr> <tr> <td><b>ADX II 42-1K-ETHERNET</b></td> <td>4</td> </tr> <tr> <td><b>ADX II 14-80M-PCIE X</b></td> <td>5</td> </tr> <tr> <td><b>ADX II 52-1K-ETHERNET</b></td> <td>6</td> </tr> </tbody> </table>	<b>ADX II 85-1M-PCI(EX)</b>	0	<b>DX II 64-1M-PCI</b>	0	<b>ADX II 42-1K-ETHERNET</b>	4	<b>ADX II 14-80M-PCIE X</b>	5	<b>ADX II 52-1K-ETHERNET</b>	6
bOpenDialog	コンフィグレーションファイルを当関数内蔵の“ファイルを開くダイアログボックス”で指定する場合、本引数を TRUE(=1)とします。FALSE(=0)とした場合、lpsCharPayload.lpcConfigFileName でファイルを直接指定します。 “ファイルを開くダイアログボックス”の初期フォルダは lpsCharPayload.lpcInitialDir で指定します。																																				
lpsSAYA_DEVICE_INFO	デバイス情報構造体 SAYA_DEVICE_INFO2 へのポインタ。bADioxLoad_EX3 では SAYA_DEVICE_INFO ですが、本関数では SAYA_DEVICE_INFO2 となります。																																				
lpsTXBUFSETUP	コンフィグレーションファイルにより反映された TXBUFSETUP2 構造体へのポインタ。																																				
lpsIOGEOSETUP	コンフィグレーションファイルにより反映された IOGEOSETUP2 構造体へのポインタ。																																				
lpsTDIO_MISC	コンフィグレーションファイルにより反映された TDIO_MISC 構造体へのポインタ。																																				
lpsTADIO	コンフィグレーションファイルにより反映された TADIO2 構造体へのポインタ。																																				
lpsTConfigPWM	コンフィグレーションファイルにより反映された TConfigPWM 構造体へのポインタ。																																				
lpsTSetupPWM	コンフィグレーションファイルにより反映された TSetupPWM 構造体へのポインタ。																																				
lpsScpSetupAich	コンフィグレーションファイルにより反映された SCP_SETUP_AIALL 構造体へのポインタ。 この引数は信号調節機能付き MultifunctionI/O でのみ意味があります。																																				
lpsEXTENTION	コンフィグレーションファイルにより反映された ADIOX_EXTENTION2(VB 用は ADIOX_EXTENTION2B)構造体へのポインタ。																																				
dwRingbufferSize	セカンダリリングバッファの倍率。当引数は <b>ADX II 85-1M-PCI(EX)</b> 、 <b>DX II 64-1M-PCI</b> でのみ意味があります。																																				
lpsCharPayload	コンフィグレーションファイル名、もしくは、“ファイル保存ダイアログボックス”のベースフォルダ名を指定する構造体 CharPayloadC(VB 用は CharPayloadB)へのポインタ。																																				
dwDeviceType	デバイスの種類を設定してください。 <table border="0"> <tbody> <tr> <td><b>ADX II 85-1M-PCI(EX)</b></td> <td>0</td> </tr> <tr> <td><b>DX II 64-1M-PCI</b></td> <td>0</td> </tr> <tr> <td><b>ADX II 42-1K-ETHERNET</b></td> <td>4</td> </tr> <tr> <td><b>ADX II 14-80M-PCIE X</b></td> <td>5</td> </tr> <tr> <td><b>ADX II 52-1K-ETHERNET</b></td> <td>6</td> </tr> </tbody> </table>	<b>ADX II 85-1M-PCI(EX)</b>	0	<b>DX II 64-1M-PCI</b>	0	<b>ADX II 42-1K-ETHERNET</b>	4	<b>ADX II 14-80M-PCIE X</b>	5	<b>ADX II 52-1K-ETHERNET</b>	6																										
<b>ADX II 85-1M-PCI(EX)</b>	0																																				
<b>DX II 64-1M-PCI</b>	0																																				
<b>ADX II 42-1K-ETHERNET</b>	4																																				
<b>ADX II 14-80M-PCIE X</b>	5																																				
<b>ADX II 52-1K-ETHERNET</b>	6																																				

## vADioxStoreFile\_EX3 [New]

設定ファイルを書き出します。

**C/C++** void vADioxStoreFile\_EX3

```
(
    BOOL bOpenDialog,
    TXBUFSETUP2 *sTXBUFSETUP,
    IOGEOSETUP2 *sIOGEOSETUP,
    TDIO_MISC *sTDIO_MISC,
    TADIO2 *sTADIO,
    TConfigPWM *sTConfigPWM,
    TSetupPWM *sTSetupPWM,
    SCP_SETUP_AIALL *sScpSetupAich,
    ADIOX_EXTENTION2 * IpsEXTENTION,
    CharPayloadC *IpsCharPayload
);
```

**C#** vADioxStoreFile\_EX3

```
(
    int bOpenDialog,
    ref TXBUFSETUP2 sTXBUFSETUP,
    ref IOGEOSETUP2 sIOGEOSETUP,
    ref TDIO_MISC sTDIO_MISC,
    ref TADIO2 sTADIO,
    ref TConfigPWM sTConfigPWM,
    ref TSetupPWM sTSetupPWM,
    ref SCP_SETUP_AIALL IpsScpSetupAich,
    ref ADIOX_EXTENTION2 IpsEXTENTION,
    ref CharPayloadC IpsCharPayload
);
```

**VB** Declare Function vADioxStoreFile\_EX3 Lib "ADiox2.dll" \_

```
( _
    ByVal bOpenDialog As Long, _
    ByRef IpsTXBUFSETUP 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_vb As ADIOX_EXTENTION2B, _
    ByRef IpsCharPayload As CharPayloadB_
)
```

<b>引数</b>	bOpenDialog	<p>コンフィグレーションファイルの保存先を当関数内蔵の“ファイル保存ダイアログボックス”で指定する場合、本引数を TRUE(=1)とします。FALSE(=0)とした場合、ファイル名を IpsCharPayload.lpcConfigFileName で直接指定します。“ファイル保存ダイアログボックス”の初期フォルダは IpsCharPayload.lpcInitialDir で指定します。</p>
	IpsTXBUFSETUP	コンフィグレーションファイルに保存したい TXBUFSETUP2 構造体へのポインタ。
	IpsIOGEOSETUP	コンフィグレーションファイルに保存したい IOGEOSETUP2 構造体へのポインタ。
	IpsTDIO_MISC	コンフィグレーションファイルに保存したい TDIO_MISC 構造体へのポインタ。
	IpsTADIO	コンフィグレーションファイルに保存したい TADIO2 構造体へのポインタ。
	IpsTConfigPWM	コンフィグレーションファイルに保存したい TConfigPWM 構造体へのポインタ。
	IpsTSetupPWM	コンフィグレーションファイルに保存したい TSetupPWM 構造体へのポインタ。
	IpsScpSetupAich	コンフィグレーションファイルに保存したい SCP_SETUP_AIALL 構造体へのポインタ。
	IpsEXTENTION	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 ADIOX_EXTENTION2 (VB 用は ADIOX_EXTENTION2B)構造体へのポインタ。
	IpsCharPayload	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。

## vADioxDeviceInfoLockupEX [New]

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 vADioxDeviceInfoLockupEx ( struct SAYA\_DEVICE\_INFO\_EX \* lpsSAYA\_DEVICE\_INFO,  
DWORD dwDeviceType , DWORD dwRingbufferSize);

**C#** void vADioxDeviceInfoLockupEx ( ref SAYA\_DEVICE\_INFO\_EX lpsSAYA\_DEVICE\_INFO,  
uint dwDeviceType , uint dwRingbufferSize);

**VB** Declare Sub vADioxDeviceInfoLockupEx Lib "ADiox2.dll" ( \_  
ByRef lpsSAYA\_DEVICE\_INFO\_EX As SAYA\_DEVICE\_INFO, \_  
ByVal dwDeviceType As Long, ByVal dwRingbufferSize As Long )

**引数** \* lpsSAYA\_DEVICE\_INFO デバイス情報構造体 SAYA\_DEVICE\_INFO2 へのポインタ。  
dwDeviceType デバイスの種類を設定してください。

ADX II 85-1M-PCI(EX)	0
DX II 64-1M-PCI	0
ADX II 42-1K-ETHERNET	4
ADX II 14-80M-PCIEX	5
ADX II 52-1K-ETHERNET	6

dwRingbufferSize ADX II 85-1M-PCI(EX), DX II 64-1M-PCI におけるセカンダリリングバッファサイズ(倍率)を指定します。

## dADioxCalcFreqLu [New]

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 dADioxCalcFreqLu ( DWORD dwSampling, DWORD dwDevice,  
DWORD IOGEOSSETUP\_dwFilterEnable );

**C#** double dADioxCalcFreqLu ( uint dwSampling, uint dwDevice,  
uint IOGEOSSETUP\_dwFilterEnable);

**VB** Declare Function dADioxCalcFreqLu Lib "ADiox2.dll" ( ByVal dwSampling As Long,  
ByVal dwDevice As Long , ByVal IOGEOSSETUP\_dwFilterEnable As Long ) As Double

**引数** dwSampling TXBUFSETUP2.dwClockScall を指定します。  
dwDevice デバイスの種類を設定してください。

ADX II 85-1M-PCI(EX)	0
DX II 64-1M-PCI	0
ADX II 42-1K-ETHERNET	4
ADX II 14-80M-PCIEX	5
ADX II 52-1K-ETHERNET	6

IOGEOSSETUP\_dwFilterEnable

ADX II 42-1K-ETHERNET におけるデジタルフィルタの設定状態を表します。  
構造体 IOGEOSSETUP のメンバ変数 dwFilterEnable をセットしてください。  
(ADX II 42-1K-ETHERNET では、デジタルフィルタ ON 時に、サンプリング速度が 1/4 になる為)

**戻り値** 周波数(KHz)が返ります。

## 17. 構造体 1 [ADiox2.dll]

1 章～12 章の API で使用される構造体です。ADiox2.dll (下に ADioxScp.dll) に実装されています。

### 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

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4つの要因があり、同時に複数の割り込み要因が格納される場合があります。複数の割り込みから一つの要因を抽出する場合には以下のように、抽出したい割り込み要因との論理積をとって、それを抽出したい割り込み要因と一致しているか確認してください。

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

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

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

Bit00 カウンター0 コンペア・比較対象と一致  
 Bit01 カウンター0 コンペア・比較対象を上回った  
 Bit02 カウンター0 コンペア・比較対象を下回った  
 Bit03 カウンター0 コンペア・比較対象範囲に入った  
 Bit04 カウンター1 コンペア・比較対象と一致  
 Bit05 カウンター1 コンペア・比較対象を上回った  
 Bit06 カウンター1 コンペア・比較対象を下回った  
 Bit07 カウンター1 コンペア・比較対象範囲に入った  
 Bit08 カウンター2 コンペア・比較対象と一致  
 Bit09 カウンター2 コンペア・比較対象を上回った  
 Bit10 カウンター2 コンペア・比較対象を下回った  
 Bit11 カウンター2 コンペア・比較対象範囲に入った  
 Bit12 カウンター3 コンペア・比較対象と一致  
 Bit13 カウンター3 コンペア・比較対象を上回った  
 Bit14 カウンター3 コンペア・比較対象を下回った  
 Bit15 カウンター3 コンペア・比較対象範囲に入った  
 Bit16 DI16(ADX II 85-1M-PCI(EX) , DX II 64-1M-PCI)  
 /DI0(上記以外の機種)のエッジ割り込み  
 Bit17 DI17(ADX II 85-1M-PCI(EX) , DX II 64-1M-PCI)  
 /DI1(上記以外の機種)のエッジ割り込み  
 Bit18 DI18(ADX II 85-1M-PCI(EX) , DX II 64-1M-PCI)  
 /DI2(上記以外の機種)のエッジ割り込み  
 Bit19 DI19(ADX II 85-1M-PCI(EX) , DX II 64-1M-PCI)  
 /DI3(上記以外の機種)のエッジ割り込み  
 Bit20 DI20(ADX II 85-1M-PCI(EX) , DX II 64-1M-PCI)  
 /DI4(上記以外の機種)のエッジ割り込み  
 Bit21 DI21(ADX II 85-1M-PCI(EX) , DX II 64-1M-PCI)  
 /DI5(上記以外の機種)のエッジ割り込み  
 Bit22 DI22(ADX II 85-1M-PCI(EX) , DX II 64-1M-PCI)  
 /DI6(上記以外の機種)のエッジ割り込み  
 Bit23 DI23(ADX II 85-1M-PCI(EX) , DX II 64-1M-PCI)  
 /DI7(上記以外の機種)のエッジ割り込み  
 Bit24 DI24(ADX II 85-1M-PCI(EX) , DX II 64-1M-PCI)  
 /DI8(上記以外の機種)のエッジ割り込み  
 Bit25 DI25(ADX II 85-1M-PCI(EX) , DX II 64-1M-PCI)  
 /DI9(上記以外の機種)のエッジ割り込み  
 Bit26 DI26(ADX II 85-1M-PCI(EX) , DX II 64-1M-PCI)  
 /DI10(上記以外の機種)のエッジ割り込み  
 Bit27 DI27(ADX II 85-1M-PCI(EX) , DX II 64-1M-PCI)  
 /DI11(上記以外の機種)のエッジ割り込み  
 Bit28 DI28(ADX II 85-1M-PCI(EX) , DX II 64-1M-PCI)  
 /DI12(上記以外の機種)のエッジ割り込み  
 Bit29 DI29(ADX II 85-1M-PCI(EX) , DX II 64-1M-PCI)  
 /DI13(上記以外の機種)のエッジ割り込み  
 Bit30 DI30(ADX II 85-1M-PCI(EX) , DX II 64-1M-PCI)  
 /DI14(上記以外の機種)のエッジ割り込み  
 Bit31 DI31(ADX II 85-1M-PCI(EX) , DX II 64-1M-PCI)  
 /DI15(上記以外の機種)のエッジ割り込み

## TXBUFSETUP2

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

### C/C++

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

### C#

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

## 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
dwInterruptMode	As Long
dwDeadTime	As Long
dwStopCounterValue	As Long
dwMuxSequenceAuto	As Long
dwAoHspBufferd	As Long
dwDoHspBufferd	As Long
dwAdiBufferOn	As Long
bConnectBuffer	As Byte
bTrigEnable	As Long

End Type

## メンバ変数

## 【サンプリング周波数】

dwClockScall

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

[ADX II 85-1M-PCI\(EX\)](#), [DX II 64-1M-PCI](#)

サンプリング周波数=(40MHz÷dwClockScall)÷シーケンシャル取り込み数。

[ADX II 42-1K-ETHERNET](#)

サンプリング時間=16.95421ns×dwClockScall×n(デジタルフィルタオフ)

サンプリング時間=67.81684ns×dwClockScall×n(デジタルフィルタオン)

[ADX II 52-1K-ETHERNET](#)

サンプリング時間=3686.4KHz×dwClockScall×n

[ADX II 14-80M-PCIEX](#)

0x0=66MHz,0x1=33MHz,0x3=16.5MHz,0x7=8.25MHz,0xF=4.125MHz

0x10=80MHz,0x11=40MHz,0x13=20MHz,0x17=10MHz,0x1F=5MHz,

0x20=ExCLK,0x21=ExCLK/2,0x23=ExCLK/4,0x21=ExCLK/8,0x2F=ExCLK/16

※ExCLK は外部クロックの意味

## 【トリガモード】

dwTrigStopMode

ストップトリガを指定します。下記 7 つの中から選択してください。

dwTrigStartMode

スタートトリガを指定します。下記 7 つの中から選択してください。

**RESET**

トリガ条件は成立しない

**BURST**

無条件にトリガを成立させる。

**DI\_POSEDGE**

デジタル入出力の立ち上がりエッジでトリガを成立させる

**DI\_NEGEDGE**

デジタル入出力の立ち下がりエッジでトリガを成立させる

**DI\_PATTERN**

デジタル入出力が指定したパターンとなったときにトリガを成立させる

**AI\_LEVEL**

アナログ入出力のレベル(エッジ)トリガ

**AI\_AREA**

アナログ入出力のエリアトリガ

## 【トリガソース】

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

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

[ADX II 14-80M-PCIEX](#) 0:AI0 1:AI1 2:AO0 3:AO1

[ADX II 42-1K-ETHERNET](#) 0:AI 1:AO0 2:AO1

[ADX II 52-1K-ETHERNET](#) 0:AI 1:AO0 2:AO1 3:AO2 4:AO3

[ADX II 85-1M-PCI\(EX\)](#) 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 n サンプル先に、データ収集を開始します。 <a href="#">ADX II 14-80M-PCIEX(n=256)</a> 、 <a href="#">ADX II 85-1M-PCI(n=32)</a> のみ有効です。

**【同期運転】**

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

**【トリガ・リングバッファ開始停止・自動停止】**

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】**

dwMuxSeqenceAuto	シーケンシャル取り込みの設定を行います。本変数は同時サンプルの <a href="#">ADX II 14-80M-PCIEX</a> では無意味です。 <a href="#">ADX II 42-1K-ETHERNET</a> では本数値は 0-3 までとしてください。 <a href="#">ADX II 85-1M-PCI(EX)</a> 、 <a href="#">ADX II 52-1K-ETHERNET</a> は 0-4 までの数値としてください。意味は以下の通りです。 値が 0 の場合、シーケンシャル取り込みはディセーブル 値が 1 の場合、2ch 自動切換え 値が 2 の場合、4ch 自動切換え 値が 3 の場合、8ch 自動切換え 値が 4 の場合、16ch 自動切換え
dwAoHspBufferd	AO をリングバッファ経由にするか否か。0 でポーリング、1 でリングバッファ経由。 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。
dwAdiBufferOn	<a href="#">ADX II 14-80M-PCIEX</a> で AI/DI をリングバッファ経由にするか否か？1 でリングバッファ経由にします。本変数を 0、dwAoHspBufferd=1、dwDoHspBufferd=1 にして AO/DO のみをリングバッファ経由にすることができます。
bConnectBuffer	<a href="#">ADX II 85-1M-PCI(EX)</a> 、 <a href="#">ADX II 42-1K-ETHERNET</a> 、 <a href="#">ADX II 52-1K-ETHERNET</a> 、 <a href="#">DX II 64-1M-PCI</a> にて、エンコーダカウンタをリングバッファ経由にするか否かを指定します。0 でしない、1 でカウンタ CH0、2 でカウンタ CH0+CH1、3 でカウンタ CH0+CH1+CH2+CH3 をリングバッファ経由にします。エンコーダ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ル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

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。
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。ADX II 85-1M-PCI(EX) 専用の変数です。
dwCOB	この値が 1 の場合、アナログ入力値はコンプリメントバイナリ(2 の補数=short 型相当)となります。この値が 0 の場合、アナログ入力値はストレートバイナリ(WORD 型相当)となります。両者の変換はハードウェアで行われるので、負荷がかかりません。
dwFilterEnable	アナログ入力信号へのデジタルフィルタの切り替えを行います。以下のように、本変数を 0,1,3 とすることで、フィルタの次数が変わります。4 次は移動平均、5 次と 16 次はローパスフィルタです。 ADX II 85-1M-PCI(EX) 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 ADX II 42-1K-ETHERNET            0:OFF    1:ON4 次    3:ON4 次 ADX II 14-80M-PCIEX                0:OFF    1:ON5 次    3:ON16 次 ADX II 52-1K-ETHERNET            本変数は無意味
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に加算します。 ※ ADX II 52-1K-ETHERNET では無意味です。
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。ADX II 85-1M-PCI(EX) 専用の変数です。
dwMux	入力チャンネルを切り替えます。ADX II 85-1M-PCI(EX) ではシングルエンド 16 チャンネルの場合 0-15、差動 8 チャンネルの場合 0-7 が有効な値です。ADX II 52-1K-ETHERNET では 0-15、ADX II 42-1K-ETHERNET では 0-7 が有効です。ADX II 14-80M-PCIEX では無意味です。
dwAI_Range	アナログ入力レンジ(フルスケール)を設定します。0-7 まで有効で、0 から順番に入力レンジを並べると "±10.0V", "±5.0V", "±2.5V", "±1.25V", "+10V", "+5V", "+2.5V", "+1.25V" となります。ADX II 85-1M-PCI(EX) 専用の変数です。
dwAO_Range	アナログ出力レンジ(フルスケール)を設定します。0-7 まで有効で、0 から順番に入力レンジを並べると "±5.0V", "±2.5V", "±1.25V", "±625mV", "+10V", "+5V", "+2.5V", "+1.25V" となります。ADX II 85-1M-PCI(EX) 専用の変数です。
dwChatCan	この値が 0 の場合、デジタル入力のチャタリングキャンセラー(フィルタ)を OFF にします。この値が 1 の場合、デジタル入力のチャタリングキャンセラー(フィルタ)を ON にします。

## TDIO\_MISC

エンコーダーカウンター、周波数カウンター、PWM(パルスジェネレーター)、ストローラッチ、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

dwStrobeInternal	ストローブ入出力をデジタル入出力チャンネル 31 (ADX II 85-1M-PCI(EX), DX II 64-1M-PCI)、デジタル出力 15(ADX II 42-1K-ETHERNET, ADX II 52-1K-ETHERNET)、デジタル出力 3(ADX II 14-80M-PCIEX)に埋め込みか否かを設定します。1 で埋め込み、0 で埋め込まない(ストローブ専用ヘッダコネクタまたはピンを使用)
dwSetFreq	PWM 周期を設定します。この値と PWM サイクル周波数の関係は以下の通りです。 0=1.96608msec                    1=3.93216msec 2=7.86432msec                    3=15.72864msec 4=31.45728msec                   5=62.91456msec 6=125.82912msec                   7=251.65824msec
dwLinkPwmToDo	PWM 出力をデジタル出力に出すか否かを設定します。この変数のビットフィールドが、PWM チャンネルに相当します。ビットフィールド 1 で PWM 有効になります。ADX II 85-1M-PCI(EX)、DX II 64-1M-PCI では 16ch、ADX II 14-80M-PCIEX では 4ch の PWM を有します。
dwDinIntMode16	DI16(ADX II 85-1M-PCI(EX), DX II 64-1M-PCI) / DI0(上記以外の機種)のエッジ割り込み要因を指定します。
dwDinIntMode17	DI17(ADX II 85-1M-PCI(EX), DX II 64-1M-PCI) / DI1(上記以外の機種)のエッジ割り込み要因を指定します。
dwDinIntMode18	DI18(ADX II 85-1M-PCI(EX), DX II 64-1M-PCI) / DI2(上記以外の機種)のエッジ割り込み要因を指定します。
dwDinIntMode19	DI19(ADX II 85-1M-PCI(EX), DX II 64-1M-PCI) / DI3(上記以外の機種)のエッジ割り込み要因を指定します。
dwDinIntMode20	DI20(ADX II 85-1M-PCI(EX), DX II 64-1M-PCI) / DI4(上記以外の機種)のエッジ割り込み要因を指定します。
dwDinIntMode21	DI21(ADX II 85-1M-PCI(EX), DX II 64-1M-PCI) / DI5(上記以外の機種)のエッジ割り込み要因を指定します。
dwDinIntMode22	DI22(ADX II 85-1M-PCI(EX), DX II 64-1M-PCI) / DI6(上記以外の機種)のエッジ割り込み要因を指定します。
dwDinIntMode23	DI23(ADX II 85-1M-PCI(EX), DX II 64-1M-PCI) / DI7(上記以外の機種)のエッジ割り込み要因を指定します。
dwDinIntMode24	DI24(ADX II 85-1M-PCI(EX), DX II 64-1M-PCI) / DI8(上記以外の機種)のエッジ割り込み要因を指定します。
dwDinIntMode25	DI25(ADX II 85-1M-PCI(EX), DX II 64-1M-PCI) / DI9(上記以外の機種)のエッジ割り込み要因を指定します。
dwDinIntMode26	DI26(ADX II 85-1M-PCI(EX), DX II 64-1M-PCI) / DI10(上記以外の機種)のエッジ割り込み要因を指定します。
dwDinIntMode27	DI27(ADX II 85-1M-PCI(EX), DX II 64-1M-PCI) / DI11(上記以外の機種)のエッジ割り込み要因を指定します。
dwDinIntMode28	DI28(ADX II 85-1M-PCI(EX), DX II 64-1M-PCI) / DI12(上記以外の機種)のエッジ割り込み要因を指定します。
dwDinIntMode29	DI29(ADX II 85-1M-PCI(EX), DX II 64-1M-PCI) / DI13(上記以外の機種)のエッジ割り込み要因を指定します。
dwDinIntMode30	DI30(ADX II 85-1M-PCI(EX), DX II 64-1M-PCI) / DI14(上記以外の機種)のエッジ割り込み要因を指定します。
dwDinIntMode31	DI31(ADX II 85-1M-PCI(EX), DX II 64-1M-PCI) / DI15(上記以外の機種)のエッジ割り込み要因を指定します。 < ↑ dwDinIntMode16~31 で指定する割り込み要因の種類 > <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 dwCounterMode_B dwCounterMode_C dwCounterMode_D	カウンター0 の動作モードを以下の 0-7 で指定します。 カウンター1 の動作モードを以下の 0-7 で指定します。 カウンター2 の動作モードを以下の 0-7 で指定します。 ダウカウンター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 (ADX II 14-80M-PCIEX ではカウンター0 のみ)

(※以降 ADX II 14-80M-PCIEX ではカウンター0のみ)

dwLatchMode_A dwLatchMode_B dwLatchMode_C dwLatchMode_D	<p>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。 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。 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。 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。</p> <p>&lt; ↑ dwLatchMode_A~D で指定するラッチモード &gt;  <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</p>
dwZ_CENTER_A dwZ_CENTER_B dwZ_CENTER_C dwZ_CENTER_D	<p>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。 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。 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。 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。</p> <p>&lt; dwZ_CENTER_A ~D で指定するZ相成立条件=カウンターリセット=原点復帰 &gt;            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</p>
dwSoftwareClear_A dwSoftwareClear_B dwSoftwareClear_C dwSoftwareClear_D	<p>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。 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。 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。 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。</p>
dwCompareMode	<p>0を指定すると、カウンターコンペア値LOWは、dwReferenceLow_A、dwReferenceLow_B、dwReferenceLow_C、dwReferenceLow_Dを使用する。1を指定すると、隣接カウンター(若い方)のカウンター値をカウンターコンペア値LOWとして利用する。カウンター0のみカウンター3の値を使う。</p>
dwLinkCounterToDO_A dwLinkCounterToDO_B dwLinkCounterToDO_C dwLinkCounterToDO_D	<p>カウンター0のコンペア結果をDOにリンクするか否かを指定します。0x0-0xFが有効な値で、各ビットフィールドは以下の意味を持ちます。 カウンター1のコンペア結果をDOにリンクするか否かを指定します。0x0-0xFが有効な値で、各ビットフィールドは以下の意味を持ちます。 カウンター2のコンペア結果をDOにリンクするか否かを指定します。0x0-0xFが有効な値で、各ビットフィールドは以下の意味を持ちます。 カウンター3のコンペア結果をDOにリンクするか否かを指定します。0x0-0xFが有効な値で、各ビットフィールドは以下の意味を持ちます。</p> <p>&lt; ↑ dwLinkCounterToDO_A~Dのビットフィールド &gt;            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            カウンターコンペア出力とDOの割り付けはハードウェア仕様書(カウンタの章)に記載。</p>
dwCounterIntMode_A dwCounterIntMode_B dwCounterIntMode_C dwCounterIntMode_D	<p>カウンター0のコンペア結果を割り込みにリンクするか否かを指定します。0x0-0xFが有効な値で、各ビットフィールドは以下の意味を持ちます。 カウンター1のコンペア結果を割り込みにリンクするか否かを指定します。0x0-0xFが有効な値で、各ビットフィールドは以下の意味を持ちます。 カウンター2のコンペア結果を割り込みにリンクするか否かを指定します。0x0-0xFが有効な値で、各ビットフィールドは以下の意味を持ちます。 カウンター3のコンペア結果を割り込みにリンクするか否かを指定します。0x0-0xFが有効な値で、各ビットフィールドは以下の意味を持ちます。</p> <p>&lt; dwCounterIntMode_A ~Dのビットフィールド &gt;            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</p>
dwReferenceLow_A dwReferenceLow_B dwReferenceLow_C dwReferenceLow_D dwReferenceHigh_A dwReferenceHigh_B dwReferenceHigh_C dwReferenceHigh_D dwSetGate	<p>カウンター0 コンペア値LOW (32Bit カウンタなので0~0xFFFFFFFFを指定して下さい)            カウンター1 コンペア値LOW (32Bit カウンタなので0~0xFFFFFFFFを指定して下さい)            カウンター2 コンペア値LOW (32Bit カウンタなので0~0xFFFFFFFFを指定して下さい)            カウンター3 コンペア値LOW (32Bit カウンタなので0~0xFFFFFFFFを指定して下さい)            カウンター0 コンペア値HIGH (32Bit カウンタなので0~0xFFFFFFFFを指定して下さい)            カウンター1 コンペア値HIGH (32Bit カウンタなので0~0xFFFFFFFFを指定して下さい)            カウンター2 コンペア値HIGH (32Bit カウンタなので0~0xFFFFFFFFを指定して下さい)            カウンター3 コンペア値HIGH (32Bit カウンタなので0~0xFFFFFFFFを指定して下さい)</p> <p>周波数カウンターのゲートを指定します。ゲートとは、パルスをカウントする時間です。1秒の場合、カウントした値はそのままHz(ヘルツ)になります。以下の0-3のいずれかを指定してください。            0:1sec, 1:100msec, 2:10msec, 3:1msec</p>

[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	ADX II 42-1K-ETHERNET	DO00
	ADX II 52-1K-ETHERNET	DO00
	ADX II 85-1M-PCI(EX)	DO16
	DX II 64-1M-PCI	DO16
	<u>ADX II 14-80M-PCIEX</u>	DO00
カウンタ 0 コンペア値 LOW 以上	ADX II 42-1K-ETHERNET	DO01
	ADX II 52-1K-ETHERNET	DO01
	ADX II 85-1M-PCI(EX)	DO17
	DX II 64-1M-PCI	DO17
	<u>ADX II 14-80M-PCIEX</u>	DO01
カウンタ 0 コンペア値 LOW 以下	ADX II 42-1K-ETHERNET	DO02
	ADX II 52-1K-ETHERNET	DO02
	ADX II 85-1M-PCI(EX)	DO18
	DX II 64-1M-PCI	DO18
	<u>ADX II 14-80M-PCIEX</u>	DO02
カウンタ 0 コンペア値 LOW~HIGH の範囲内	ADX II 42-1K-ETHERNET	DO03
	ADX II 52-1K-ETHERNET	DO03
	ADX II 85-1M-PCI(EX)	DO19
	DX II 64-1M-PCI	DO19
	<u>ADX II 14-80M-PCIEX</u>	DO03
カウンタ 1 コンペア値 LOW と一致	ADX II 42-1K-ETHERNET	DO04
	ADX II 52-1K-ETHERNET	DO04
	ADX II 85-1M-PCI(EX)	DO20
	DX II 64-1M-PCI	DO20
カウンタ 1 コンペア値 LOW 以上	ADX II 42-1K-ETHERNET	DO05
	ADX II 52-1K-ETHERNET	DO05
	ADX II 85-1M-PCI(EX)	DO21
	DX II 64-1M-PCI	DO21
カウンタ 1 コンペア値 LOW 以下	ADX II 42-1K-ETHERNET	DO06
	ADX II 52-1K-ETHERNET	DO06
	ADX II 85-1M-PCI(EX)	DO22
	DX II 64-1M-PCI	DO22
カウンタ 1 コンペア値 LOW~HIGH の範囲内	ADX II 42-1K-ETHERNET	DO07
	ADX II 52-1K-ETHERNET	DO07
	ADX II 85-1M-PCI(EX)	DO23
	DX II 64-1M-PCI	DO23
カウンタ 2 コンペア値 LOW と一致	ADX II 42-1K-ETHERNET	DO08
	ADX II 52-1K-ETHERNET	DO08
	ADX II 85-1M-PCI(EX)	DO24
	DX II 64-1M-PCI	DO24
カウンタ 2 コンペア値 LOW 以上	ADX II 42-1K-ETHERNET	DO09
	ADX II 52-1K-ETHERNET	DO09
	ADX II 85-1M-PCI(EX)	DO25
	DX II 64-1M-PCI	DO25
カウンタ 2 コンペア値 LOW 以下	ADX II 42-1K-ETHERNET	DO10
	ADX II 52-1K-ETHERNET	DO10
	ADX II 85-1M-PCI(EX)	DO26
	DX II 64-1M-PCI	DO26
カウンタ 2 コンペア値 LOW~HIGH の範囲内	ADX II 42-1K-ETHERNET	DO11
	ADX II 52-1K-ETHERNET	DO11
	ADX II 85-1M-PCI(EX)	DO27
	DX II 64-1M-PCI	DO27
カウンタ 3 コンペア値 LOW と一致	ADX II 42-1K-ETHERNET	DO12
	ADX II 52-1K-ETHERNET	DO12
	ADX II 85-1M-PCI(EX)	DO28
	DX II 64-1M-PCI	DO28
カウンタ 3 コンペア値 LOW 以上	ADX II 42-1K-ETHERNET	DO13
	ADX II 52-1K-ETHERNET	DO13
	ADX II 85-1M-PCI(EX)	DO29
	DX II 64-1M-PCI	DO29
カウンタ 3 コンペア値 LOW 以下	ADX II 42-1K-ETHERNET	DO14
	ADX II 52-1K-ETHERNET	DO14
	ADX II 85-1M-PCI(EX)	DO30
	DX II 64-1M-PCI	DO30
カウンタ 3 コンペア値 LOW~HIGH の範囲内	ADX II 42-1K-ETHERNET	DO15
	ADX II 52-1K-ETHERNET	DO15
	ADX II 85-1M-PCI(EX)	DO31
	DX II 64-1M-PCI	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

dwAi0-7	アナログ入力値。同時サンプルではない <a href="#">ADX II 85-1M-PCI(EX)</a> , <a href="#">ADX II 42-1K-ETHERNET</a> , <a href="#">ADX II 52-1K-ETHERNET</a> では dwAi0 のみ有効で他の変数は使われていません。同時サンプルの <a href="#">ADX II 14-80M-PCIEX</a> は dwAi0 に AI0、dwAi1 に AI1 の値が格納されます。dwAi2-7 は現在使われていません。
dAi0-7	前記 dwAi0-7 アナログ入力値を電圧に変換した値が格納されます。単位は mV になります。dwAi0-7 が dAi0-7 に相当します。
dwAo0-7	アナログ出力チャンネル 0-7 の出力値 (最小~最大が、0~0xFFFF に対応します) <a href="#">ADX II 85-1M-PCI(EX)</a> では dwAo0-5、 <a href="#">ADX II 14-80M-PCIEX</a> と <a href="#">ADX II 42-1K-ETHERNET</a> は dwAo0-1 が、 <a href="#">ADX II 52-1K-ETHERNET</a> では dwAi0-3 が有効です。
dAo0-7	前記 dwAo0-7 アナログ出力値を電圧に変換した値が格納されます。単位は mV になります。dwAo0-7 が dAo0-7 に相当します。
dwDOS	デジタル出力チャンネル 0~31 の出力値 (Bit0~Bit31 がデジタル出力チャンネル 0~31 に対応します) (ハードウェアに実装されていないデジタル出力チャンネルのビットフィールド値は無視されます)
dwDI	デジタル入力チャンネル 0~31 の入力値 (Bit0~Bit31 がデジタル入力チャンネル 0~31 に対応します) (ハードウェアに実装されていないデジタル出力チャンネルのビットフィールド値は 0 になります)
dwDI_Latch	デジタル入力チャンネル 0~31 のストローラッチ値 (Bit0~Bit31 がデジタル入力チャンネル 0~31 に対応します) (ハードウェアに実装されていないデジタル出力チャンネルのビットフィールド値は 0 になります)
dwCounterA	エンコーダーカウンター 0 のライブ値 (現在の値※1)
dwCounterB	エンコーダーカウンター 1 のライブ値 (現在の値※1)
dwCounterC	エンコーダーカウンター 2 のライブ値 (現在の値※1)
dwCounterD	エンコーダーカウンター 3 のライブ値 (現在の値※1)
dwLatchA	エンコーダーカウンター 0 のラッチ値 ※1
dwLatchB	エンコーダーカウンター 1 のラッチ値 ※1
dwLatchC	エンコーダーカウンター 2 のラッチ値 ※1
dwLatchD	エンコーダーカウンター 3 のラッチ値 ※1
dwFreqLatch_A	周波数カウンター 0 のライブ値 (現在の値 ※2)
dwFreqLatch_B	周波数カウンター 1 のライブ値 (現在の値 ※2)
dwFreqLatch_C	周波数カウンター 2 のライブ値 (現在の値 ※2)
dwFreqLatch_D	周波数カウンター 3 のライブ値 (現在の値 ※2)
dTemp	<a href="#">ADX II 85-1M-PCI(EX)</a> 専用でボードの温度を格納します。
dwMode	必ず 0 をセットしてください。

※1 32Bit のカウンタなので、0~0xFFFFFFFF の値になります。0 でデクリメントすると 0xFFFFFFFF になります。

※2 ゲート周期に何サイクルあったかを表します。

## TConfigPWM

PWM 各チャンネルの個別設定(位相、パルス発生回数)を格納します。

### C/C++

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

**C#**

```

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

```

**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

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

(※1 0-255 が有効で、0 を指定した場合のみ、PWM サイクル実行回数制限なしになります)

(※2 0-255 が有効で、360/256 度 = 1.40625 度単位で開始位相を指定できます)

# TSetupPWM

PWM デューティー比を設定します。

## C/C++

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

## C#

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

## 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**

dwPwm0Value	PWM チャンネル 0 のデューティ比を指定します。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_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_overflow	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
dwTrigSeq	トリガの状態を以下の 0-4 で表します 0:アイドル状態 1:稼動状態 2:停止状態に遷移中 3:最終バンクの転送待ち 4:ストップトリガのデッドタイム
上記以外の変数	全て予約

## SAYA\_DEVICE\_INFO

デバイス情報を格納します。

## C/C++

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

## C#

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

## 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

dwDeviceType	デバイスの種類が入ります(以下参照) <a href="#">ADX II 85-1M-PCI(EX)/DX II 64-1M-PCI = 0</a> <a href="#">ADX II 42-1K-ETHERNET = 4</a> <a href="#">ADX II 14-80M-PCIEX = 5</a> <a href="#">ADX II 52-1K-ETHERNET = 6</a>
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) ADX II 85-1M-PCI(EX)/DX II 64-1M-PCI = 0    ADX II 42-1K-ETHERNET = 4 ADX II 14-80M-PCIEX = 5    ADX II 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 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、IOGEOSUP.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-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	クロックインポー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、IOGEOSUP.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 II 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 II 52-1K-ETHERNET は拡張ボード式なので無意味)
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)
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-PCIEX のみ 1 になります。
dwDI[16]	インテリジェント DI0-15(カウンタや DI 割り込み)と、実際の DI の割付(0xFF なら未搭載)。 DI[0]=16 なら、DI16 に DI 割り込み 0 やカウンタの A 相が実装されていることになります。
dwDO[16]	インテリジェント DO0-15(PWM やコンペア出力)と、実際の DO の割付(0xFF なら未搭載)。 DO[0]=16 なら、DO16 に PWM 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

bDoubleSave	ADX II 42-1K-ETHERNET, ADX II 52-1K-ETHERNET でアナログ入力を double 型で保存する場合 TRUE(=1)、DWORD で保存する場合 FALSE(=0)を指定します。
iAO_Gain0	波形ジェネレータ AO0 ゲイン。0~100 を指定します。
iAO_Gain1	波形ジェネレータ AO1 ゲイン。0~100 を指定します。(ADX II 14-80M-PCIEX のみ有効)
iAO_Offset0	波形ジェネレータ AO0 オフセット。±10000 を指定します。
iAO_Offset1	波形ジェネレータ AO1 オフセット。±10000 を指定します。(ADX II 14-80M-PCIEX のみ有効)
dwADO_style0	波形ジェネレータ AO0 波形。以下のいずれかを指定します。
dwADO_style1	波形ジェネレータ AO0 波形。以下のいずれかを指定します。(ADX II 14-80M-PCIEX のみ有効)
ADX_Sin	サイン
ADX_Cos	コサイン
ADX_Exp	Exp(2n)
ADX_Sqrt	Sqrt(2π) Sqrt は平方根
ADX_Triangle	三角波
ADX_Lamp	ランプ波形 (のこぎり波形)
ADX_Square	方形波
ADX_DC0	0x8000 連続 DC 出力(ゼロオフセット校正用)
ADX_DC1	0xE000 連続 DC 出力(ゲイン校正用)
ADX_File	ファイル出力 (AO0,AO1 のいずれかがファイル出力なら両方ファイル出力になります)
ADX_LargeStep	ステップ(1 リングバッファ単位で 0x1000 ずつインクリメントする)
iAO_SamplePerCycle0	1 サイクルあたりのサンプル数。サンプリング周波数 ÷ iAO_SamplePerCycle0 が波形生成周波数になります。
iAO_SamplePerCycle1	1 サイクルあたりのサンプル数。サンプリング周波数 ÷ iAO_SamplePerCycle1 が波形生成周波数になります。(ADX II 14-80M-PCIEX のみ有効)
dwADI_style	アナログ入力形式
	NO_SAVE      ファイル非保存
	DIRECT_FILE      ファイル保存
lpcAdiFileName	AI/DI 値保存ファイル名(dwADI_style で DIRECT_FILE を指定した場合必須)
lpcDmyFileName	ダミーファイル保存ファイル名(dwADI_style で DIRECT_FILE を指定した場合必須※)
lpcAdoFileName	AO/DO 出力ファイル名 (dwADO_style0,dwADO_style1 で ADX_File を指定した場合必須)
dwReserverd	予備

※ダミーファイルは高速データ収集では必須です。計測データをリアルタイムに書き込もうとしてもハードディスクは回転速度を上げるには時間がかかり、その待ち時間でバッファオーバ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、その直後からデータ収集⇒ファイル書き込むことでバッファオーバ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。

## SCP\_SETUP2

複数の [ADX II 42-1K-ETHERNET](#), [ADX II 52-1K-ETHERNET](#) の、各種構造体を内包する大規模構造体です。1 台の [ADX II 42-1K-ETHERNET](#), [ADX II 52-1K-ETHERNET](#) のメンバは、TXBUFSETUP2、IOGEOSETUP2、TDIO\_MISC、TConfigPWM、TSetupPWM、信号調節変数としてセンサーモード・ゼロスパン校正位置・ゼロスパン校正係数・スケーリング・アラームをチャンネル数保持します。SCP\_SETUP2 構造体は、更にこれらを CARD\_ID4~27 までの 24 台分格納します。SCP\_SETUP2 構造体はドライバ内部のデータベースに相当し、運用中、TXBUFSETUP2、IOGEOSETUP2、TDIO\_MISC、TConfigPWM、TSetupPWM などの構造体を使用する関数をアクセスした場合には、たとえ SCP\_SETUP2 をアクセスしていなくとも、ドライバ内部で SCP\_SETUP2 構造体に変更内容が反映されます。

### C/C++

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

### C#

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

**VB**(構造体に 64k制限があるために 3 分割されています)

```

Type SCP_SETUP2_PART1
    bMultifunctionIO_Enable(MAX_MFIO)           As Long
    sTXBUFSETUP(MAX_MFIO)                       As TXBUFSETUP2
    sIOGEOSETUP(MAX_MFIO)                      As IOGEOSETUP2
    sTDIO_MISC(MAX_MFIO)                       As TDIO_MISC
    sTConfigPWM(MAX_MFIO)                      As TConfigPWM
    sTSetupPWM(MAX_MFIO)                       As TSetupPWM
    dwLDO1(MAX_MFIO)                           As Long
    dwLDO2(MAX_MFIO)                           As Long
    dwSensorMode(MAX_MFIO, MAX_AI_CH)          As Long
    doZeroPos(MAX_MFIO, MAX_AI_CH)             As Double
    doSpanPos(MAX_MFIO, MAX_AI_CH)             As Double
    doZero_Coefficient(MAX_MFIO, MAX_AI_CH)    As Double
    doSpan_Coefficient(MAX_MFIO, MAX_AI_CH)    As Double
End Type

Type SCP_SETUP2_PART2
    bScalling(MAX_MFIO, MAX_AI_CH)             As Long
    dScallingRatio(MAX_MFIO, MAX_AI_CH)        As Double
    dOutTopScall(MAX_MFIO, MAX_AI_CH)          As Double
    dOutBottomScall(MAX_MFIO, MAX_AI_CH)       As Double
    dInTopScall(MAX_MFIO, MAX_AI_CH)           As Double
    dInBottomScall(MAX_MFIO, MAX_AI_CH)        As Double
End Type

Type SCP_SETUP2_PART3
    bAlarmMode(MAX_MFIO, MAX_AI_CH)            As Long
    dAlarmUpper(MAX_MFIO, MAX_AI_CH)           As Double
    dAlarmLower(MAX_MFIO, MAX_AI_CH)           As Double
End Type
    
```

**メンバ変数**

bMultifunctionIO_Enable	MultifunctionIO を有効にする場合 TRUE(=1)、無効にするには FALSE(=0)にしてください。
sTXBUFSETUP	TXBUFSETUP2 構造体を MAX_MFIO 個格納します。
sIOGEOSETUP	IOGEOSETUP2 構造体を MAX_MFIO 個格納します。
sTDIO_MISC	TDIO_MISC 構造体を MAX_MFIO 個格納します。
sTConfigPWM	TConfigPWM 構造体を MAX_MFIO 個格納します。
sTSetupPWM	TSetupPWM 構造体を MAX_MFIO 個格納します。
dwLDO1	シグナルコンディションコントロールレジスタ設定値 1 を MAX_MFIO 個格納します。 (アプリケーションでは参照はできても変更してはなりません) ADX II 52-1K-ETHERNET では無意味です。
dwLDO2	シグナルコンディションコントロールレジスタ設定値 2 を MAX_MFIO 個格納します。 (アプリケーションでは参照はできても変更してはなりません) ADX II 52-1K-ETHERNET では無意味です。
doZeroPos	ゼロ校正位置を"MAX_MFIO×MAX_AI_CH"個格納します。 ADX II 52-1K-ETHERNET では無意味です。
doSpanPos	スパン校正位置を"MAX_MFIO×MAX_AI_CH"個格納します。 ADX II 52-1K-ETHERNET では無意味です。
doZero_Coefficient	ゼロ校正係数を"MAX_MFIO×MAX_AI_CH"個格納します。 ADX II 52-1K-ETHERNET では必ず 0 になります。
doSpan_Coefficient	スパン校正係数を"MAX_MFIO×MAX_AI_CH"個格納します。 ADX II 52-1K-ETHERNET では必ず 1 になります。
bScalling	スケーリングする場合、TRUE(=1)しない場合 FALSE(=0)をセットします。 これを"MAX_MFIO×MAX_AI_CH"個格納します。
dScallingRatio	スケーリング係数を"MAX_MFIO×MAX_AI_CH"個格納します。 (アプリケーションでは参照はできても変更してはなりません)
dOutTopScall	変換後のスケーリング基準値(上)。これを"MAX_MFIO×MAX_AI_CH"個格納します。
dOutBottomScall	変換後のスケーリング基準値(下)。これを"MAX_MFIO×MAX_AI_CH"個格納します。
dInTopScall	変換前のスケーリング基準値(上)。これを"MAX_MFIO×MAX_AI_CH"個格納します。
dInBottomScall	変換前のスケーリング基準値(下)。これを"MAX_MFIO×MAX_AI_CH"個格納します。
bAlarmModeA	アラームモードを指定します。0 を指定すると:オフ、1 を指定すると:dAlarmUpper 以上でアラーム(オーバー)、2 を指定すると dAlarmLower 以下でアラーム(アンダー)、3 を指定すると dAlarmUpper ~ dAlarmLower の範囲内でアラーム(インレンジ)、4 を指定すると dAlarmUpper ~ dAlarmLower の範囲内でアラーム(アウトレンジ)になります。これを"MAX_MFIO×MAX_AI_CH"個格納します。
dAlarmUpper	アラーム設定値(上)を"MAX_MFIO×MAX_AI_CH"個格納します。
dAlarmLower	アラーム設定値(下)を"MAX_MFIO×MAX_AI_CH"個格納します。

dwSensorMode

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

ADX II 42-1K-ETHERNET で設定可能な種類は以下の通りです。

NOT_USE	0	シグナルコンディション未使用
CA_K	1	熱電対 K
CA_J	2	熱電対 J
CA_E	3	熱電対 E
CA_T	4	熱電対 T
CA_R	5	熱電対 R
CA_S	6	熱電対 S
CA_N	7	熱電対 N
CA_B	8	熱電対 B
PT100	9	白金測温抵抗体 Pt100
JPT100	10	白金測温抵抗体 JPt100
VBP_10mV	11	電圧±10mV レンジ
VBP_100mV	12	電圧±100mV レンジ
VBP_1V	13	電圧±1V レンジ
VBP_10V	17	電圧±10V レンジ
I_4_20	22	電流 4-20mA/500Ω 終端
I_4_20EX	23	電流 4-20mA/350Ω 終端
I_4_20EX2	26	電流 4-20mA/47Ω 終端
		↑ ADX II 42-1K-ETHERNET オンボードの終端
EC_4X	40	4 倍速カウンタ Z 未使用
EC_4XZ	41	4 倍速カウンタ Z 使用
EC_2X	42	2 倍速カウンタ Z 未使用
EC_2XZ	43	2 倍速カウンタ Z 使用
EC_1X	44	1 倍速カウンタ Z 未使用
EC_1XZ	45	1 倍速カウンタ Z 使用
UPC	46	アップダウンカウンタ Z 未使用
UPC_Z	47	アップダウンカウンタ Z 使用

ADX II 52-1K-ETHERNET で設定可能な種類は以下の通りで、信号調節ライブラリの意味は実質スケーリング・アラームに留まります。但し同機種はハードウェアの拡張ボードで様々な入力フォーマットに対応します。

VUP_5V	20	電圧+5V レンジ
--------	----	-----------

<以下は現在未使用>

VBP_1_25V	14	電圧±1.25V レンジ
VBP_2_5V	15	電圧±2.5V レンジ
VBP_5V	16	電圧±5V レンジ
VUP_1_25V	18	電圧+1.25V レンジ
VUP_2_5V	19	電圧+2.5V レンジ
VUP_10V	21	電圧+10V レンジ
VBP_30mV	24	電圧±33mV レンジ
VBP_3V	25	電圧±3.3V レンジ

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

※ 配列番号 MAX\_AI\_CH はアナログ入力チャンネルを表します。ADX II 42-1K-ETHERNET ではアナログ入力は AI0-11 が有効で 8-11 はカウンタに相当します。カウンタでもアラームやスケーリングを使うことが出来ます。ADX II 52-1K-ETHERNET ではアナログ入力は AI0-15 が有効で 16-19 はカウンタに相当します。カウンタでもアラームやスケーリングを使うことが出来ます。

# SCP\_SETUP\_AICH

ADX II 42-1K-ETHERNET、ADX II 52-1K-ETHERNET の信号調節関連の設定を格納します。SCP\_SETUP2 構造体ではなく本構造体を使うことによりターゲットデバイス 1 個分のメンバ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
```

## メンバ変数

dwSensorMode	AI チャンネル毎にターゲットのセンサー番号を指定します。
dwLDO	センサーモード NOT_USE と組み合わせでシグナルコンディションコントロールレジスタ設定値を強制モードさせるときに使います。通常は使われません。
doZeroPos	ゼロ校正位置を MAX_AI_CH 個格納します。
doSpanPos	スパン校正位置を MAX_AI_CH 個格納します。
bScalling	スケールリングする場合 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	変換後のスケールリング基準値(上)。これを MAX_AI_CH 個格納します。
dOutBottomScall	変換後のスケールリング基準値(下)。これを MAX_AI_CH 個格納します。
dInTopScall	変換前のスケールリング基準値(上)。これを MAX_AI_CH 個格納します。
dInBottomScall	変換前のスケールリング基準値(下)。これを MAX_AI_CH 個格納します。
bAlarmMod	アラームモードを指定します。0 を指定すると:オフ、1 を指定すると:dAlarmUpper 以上でアラーム(オーバー)、2 を指定すると dAlarmLower 以下でアラーム(アンダー)、3 を指定すると dAlarmUpper ~ dAlarmLower の範囲内でアラーム(インレンジ)、4 を指定すると dAlarmUpper ~ dAlarmLower の範囲内でアラーム(アウトレンジ)になります。これを MAX_AI_CH 個格納します。
dAlarmUpper	アラーム設定値(上)。これを MAX_AI_CH 個格納します。
dAlarmLower	アラーム設定値(下)。これを MAX_AI_CH 個格納します。

## SCP\_SETUP\_AIALL

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    dwBuffaSize;
    public double  dClockScall;
    public byte    bBitScall;
    public byte    bAI_ChannelScall;
    public byte    bDI_ChannelScall;
    public byte    DataType;
    public uint    dwGetYear;
    public uint    dwGetMonth;
    public uint    dwGetDay;
    public uint    dwGetHour;
    public uint    dwGetMinute;
    public uint    dwGetSecond;
    public uint    dwGetMilliseconds;
    public uint    dwInrange;
    public uint    dwReserved1;
    public uint    dwReserved2;
};

```

## VB

```

Type LOG_FRONTEND
    dwHeaderCode           As Long
    dwDeviceName          As Long
    dwBuffaSize           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

dwHeaderCode	ファイルヘッダ識別コード、必ず 0x41594154 をセットしてください。
dwDeviceName	機種コード(SAYA_DEVICE_INFO.dwDeviceType)
dwBuffaSize	リングバッファサイズ ( <a href="#">ADX II 85-1M-PCI(EX)</a> , <a href="#">DX II 64-1M-PCI</a> ではセカンダリリングバッファサイズ)
dClockScall	サンプリング周波数(Hz)
bBitScall	量子化ビット数
bAI_ChannelScall AI	チャンネル数
bDI_ChannelScall DI	チャンネル数
DataType	保存形式(DWORD=0,double=1)double 保存は <a href="#">ADX II 42-1K-ETHERNET</a> , <a href="#">ADX II 52-1K-ETHERNET</a> 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 ( <a href="#">ADX II 85-1M-PCI(EX)</a> は AI レンジ= IOGEOSSETUP2. dwAI_Range) ( <a href="#">ADX II 14-80M-PCIEX</a> は 1 でユニポーラ/0 でバイポーラ)
dwReserved1;	予備
dwReserved2;	予備

## 18.構造体 2

### [ADioxLogm.dll、ADioxReportMl.dll、ADioxTrlogMl.dll]

13章～15章のAPIで使用される構造体です。ADioxLogm.dll、ADioxReportMl.dll、ADioxTrlogMl.dllに実装されています。

### ADIOX\_IO\_NAME

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

```
C/C++
struct ADIOX_IO_NAME
{
    char    csStr[50];
};
```

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

### ADIOX\_CSV\_NAME

Multi device data logger 向け。CSVログの名称をMAX\_AI\_CH(32)チャンネル分格納します。本構造体は上位の構造体のメンバになります。本構造体を単独で使用するはありません。

```
C/C++
struct ADIOX_CSV_NAME
{
    ADIOX_IO_NAME    sCH[MAX_AI_CH];
};
```

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

### ADIOX\_CSV\_FORMAT\_EX

Multi device data logger 向け。CSVロガー各種設定を行う為のメンバ変数群で構成されています。

```
C/C++
struct ADIOX_CSV_FORMAT_EX
{
    BOOL    bMultifunctionIO_Enable[MAX_MFIO];
    DWORD  dwBuffaSize;
    double  dClockScall;
    BYTE    bAI_ChannelScall;
    BYTE    bDI_ChannelScall;
    char    *FolderName;
    char    *unit_Title;
    BOOL    bTdMode;
    BOOL    bCsvLogEnable[MAX_AI_CH][MAX_MFIO];
    ADIOX_CSV_NAME    sADIOX_CSV_NAME[MAX_MFIO];
    DWORD  dwReserve0;
    DWORD  dwReserve1;
    DWORD  dwReserve2;
    DWORD  dwReserve3;
    double  dReserve0;
    double  dReserve1;
    double  dReserve2;
    double  dReserve3;
    double  dTimeOfSample;
};
```

## メンバ変数

bMultifunctionIO_Enable	複数のADR42Cのログを取る場合、どのCARD_IDのADR42Cを使うか、本変数で指定します。本変数の配列番号はCARD_IDを表し、TRUEであれば、そのCARD_IDはCSVログの対象になります。FALSEであればCARD_IDはログの対象になりません。
dwBuffaSize	bADioxLoggerWriteで1度書き込むバッファサイズを指定します。これがCSVファイルの1シート(1ファイル)分に相当します。bADioxLoggerWriteにはアナログ入力信号 xAI チャンネル数、バーンアウト信号 xAI チャンネル数、デジタル入力信号 xDI チャンネル数がありますが、これらをdwBuffaSize 分用意する必要があります。
dClockScall	サンプリングレート(Hz)を指定します。
bAI_ChannelScallAI	チャンネル数を指定します。1~32 で任意の数値を設定できます。bADioxLoggerWrite で書き込むバッファは、先頭から本変数で指定した AI チャンネル数分のデータが順列に配置され、これをdwBuffaSize で指定した回数繰り返すようにして下さい。AI チャンネルという名称ですが、カウンタなども、ここに入れることが出来ます。MultiLogger.exeではADR42Cの8chのアナログ入力信号と4chのカウンタ信号をあわせて12chで記録できるようになっています。
bDI_ChannelScall	DIチャンネル数を指定します。
FolderName	CSVログファイルの保存場所をフルパスで指定して下さい。(例)"C:\¥¥MFIO_Files"
unit_Title	見出し、不要な場合はNULLを指定して下さい。(例)"SAYA MFIO 計測 DATA"
bTdMode	予約。必ずFALSEにして下さい。
bCsvLogEnable	CSVログの対象になるアナログ入力チャンネル(含:カウンタ)を指定します。本変数の配列番号はチャンネル番号及び、CARD_IDを表します。TRUEであれば、そのCARD_IDにおけるチャンネル番号はCSVログの対象になります。FALSEであれば、そのCARD_IDにおけるチャンネル番号CARD_IDはログの対象になりません。
sDIOX_CSV_NAME	CSVログの各チャンネルの名称を、各CARD_ID×チャンネル数分格納します。名称自体は最下層の構造体ADIOX_IO_NAMEに格納され、これがMAX_AI_CHチャンネル分集合する事で、構造体ADIOX_CSV_NAMEになり、更にそれがMAX_MFIOだけ集合して本変数になります。
dwReserve0	予約
dwReserve1	予約
dwReserve2	予約
dwReserve3	予約
dReserve0	予約
dReserve1	予約
dReserve2	予約
dReserve3	予約
dTimeOfSample	予約(何も設定しないで下さい)

## ADIOX\_TREND\_SETUP

Multi device trend graph 向け。トレンド機能各種設定を行う為のメンバ変数群で構成されています。

**C/C++**

```

struct ADIOX_TREND_SETUP
{
    RECT    pPpbRect;
    char    *FolderName;
    DWORD   dwCHMAX[MAX_MFIO];
    double  dDivScall[MAX_AI_CH][MAX_MFIO];
    double  dOffsetScall[MAX_AI_CH][MAX_MFIO];
    double  dLimiter[MAX_AI_CH][MAX_MFIO];
    BOOL    bMultifunctionIO_Enable[MAX_MFIO];
    BOOL    bCsvLogEnable[MAX_AI_CH][MAX_MFIO];
    DWORD   dwReserve0;
    DWORD   dwReserve1;
    DWORD   dwReserve2;
    DWORD   dwReserve3;
    double  dReserve0;
    double  dReserve1;
    double  dReserve2;
    double  dReserve3;
};

```

## メンバ変数

pPpbRect	トレンド用のピクチャボックスサイズを格納した、RECT 構造体 (Win32/MFC の構造体)
FolderName	トレンドログファイル (拡張子 tif) を保存する場所を フルパス名 で指定して下さい。 (例) "ADIOX_CSV_FORMAT_EX.FolderName"
dwCHMAX	最大チャンネル数を設定します。これが配列で MAX_MFIO 台分並んでいます。配列は CARD_ID を表します。
dDivScall	画面縦軸のスケーリング比率 (トレンド用ピクチャボックスサイズ/入力信号フルスケール) の設定値配列。これが MAX_AI_CH チャンネル × MAX_MFIO 台数分並んでいます。入力信号は、アナログ入力だけではなくカウンタも使用することが出来ます。配列 [MAX_MFIO] は CARD_ID を表し、配列 [MAX_AI_CH] はチャンネル番号を表します。
dOffsetScall	画面縦軸のオフセット (dDivScall * 信号スケール ミニマム) の設定値配列。例えば 1000 ~ -1000 をフルスケールにしている場合に、-1000 を画面最下部に位置させるために必要です。これが MAX_AI_CH チャンネル × MAX_MFIO 台数分並んでいます。入力信号は、アナログ入力だけではなくカウンタも使用することが出来ます。配列 [MAX_MFIO] は CARD_ID を表し、配列 [MAX_AI_CH] はチャンネル番号を表します。
dLimiter	本引数は 0 の場合に無視されます。0 以上を指定すると、トレンドはこの値 ~ 0 の間でスケーリングされます。例えば熱電対などは 1000°C を超える温度からマイナスまでの広いレンジを持っていますが、常温付近で使うと、殆どトレンドは振れなくなってしまう。そこで本引数を例えば 200 とすると 200°C ~ 0°C の間がトレンド画面のフルスケールになります。これが MAX_AI_CH チャンネル × MAX_MFIO 台数分並んでいます。入力信号は、アナログ入力だけではなくカウンタも使用することが出来ます。配列 [MAX_MFIO] は CARD_ID を表し、配列 [MAX_AI_CH] はチャンネル番号を表します。
bMultifunctionIO_Enable	複数の ADR42C のログを取る場合、どの CARD_ID の ADR42C を使うか、本変数で指定します。本変数の配列番号は CARD_ID を表し、TRUE であれば、その CARD_ID は CSV ログの対象になります。FALSE であれば CARD_ID はログの対象になりません。 ※ADIOX_CSV_FORMAT_EX の同一名称変数と同じ意味
bCsvLogEnable	CSV ログの対象になるアナログ入力チャンネル (含: カウンタ) を指定します。本変数の配列番号はチャンネル番号及び、CARD_ID を表します。TRUE であれば、その CARD_ID におけるチャンネル番号は CSV ログの対象になります。FALSE であれば、その CARD_ID におけるチャンネル番号は CSV ログの対象になりません。 ※ADIOX_CSV_FORMAT_EX の同一名称変数と同じ意味
dwReserve0	予約
dwReserve1	予約
dwReserve2	予約
dwReserve3	予約
dReserve0	予約
dReserve1	予約
dReserve2	予約
dReserve3	予約

## ADIOX\_SET\_TREND\_DATA

Multi device trend graph 向け。トレンドグラフ機能各種設定を行う為のメンバ変数群で構成されています。

## C/C++

```
struct ADIOX_SET_TREND_DATA
{
    double  dAnalogData[MAX_AI_CH][MAX_MFIO];
    BOOL    bBurnOut[MAX_AI_CH][MAX_MFIO];
};
```

## メンバ変数

dAnalogData	アナログデータ (カウンタでも良い) の配列。これが MAX_AI_CH チャンネル × MAX_MFIO の配列になっています。配列 [MAX_MFIO] は CARD_ID を表し、配列 [MAX_AI_CH] はチャンネル番号を表します。
bBurnOut	バーンアウト (ロストでも良い) の配列。これが MAX_AI_CH チャンネル × MAX_MFIO の配列になっています。配列 [MAX_MFIO] は CARD_ID を表し、配列 [MAX_AI_CH] はチャンネル番号を表します。

## ADIOX\_SET\_REPORT\_IN

Multi device reporting 向け。レポート機能各種設定を行う為のメンバ変数群で構成されています。

### C/C++

```
struct ADIOX_SET_REPORT_IN
{
    char    *FolderName;
    BYTE    bAccBoundary;
    BYTE    bChMax;
    BOOL    bIoEnable[MAX_MFIO];
    BOOL    bChEnable[MAX_AI_CH][MAX_MFIO];
    DWORD   dwReserve0;
    DWORD   dwReserve1;
    DWORD   dwReserve2;
    DWORD   dwReserve3;
    double  dReserve0;
    double  dReserve1;
    double  dReserve2;
    double  dReserve3;
};
```

### メンバ変数

FolderName	レポートログファイル(拡張子 br 及び csv)を保存する場所を フルパス名 で指定して下さい。 (例)“C:\Report”
bAccBoundary	過去平均処理(アナログ)と過去積算処理(カウンタ)の境界チャンネル。MultiLogger の場合 ADR42C の AI0-7 をチャンネル 0-7 に、カウンタ 0-3 をチャンネル 8-11 相当で、トレンドや CSV ロガーを動かしていますから、本変数は 8 になります。
bChMax	使用するリモート I/O の最大チャンネル数 (MultiLogger なら ADR42C の AI0-7 をチャンネル 0-7 に、カウンタ 0-3 をチャンネル 8-11 相当で 12)
bIoEnable	複数の ADR42C のログを取る場合、どの CARD_ID の ADR42C を使うか、本変数で指定します。本変数の配列番号は CARD_ID を表し、TRUE であれば、その CARD_ID は CSV ログの対象になります。FALSE であれば CARD_ID はログの対象になりません。ADIOX_CSV_FORMAT_EX.bMultifunctionIO_Enable[MAX_MFIO]と同じ意味
bChEnable	CSV ログの対象になるアナログ入力チャンネル(含:カウンタ)を指定します。本変数の配列番号はチャンネル番号及び、CARD_ID を表します。TRUE であれば、その CARD_ID におけるチャンネル番号は CSV ログの対象になります。FALSE であれば、その CARD_ID におけるチャンネル番号 CARD_ID はログの対象になりません。ADIOX_CSV_FORMAT_EX. bCsvLogEnable[MAX_AI_CH][MAX_MFIO]と同じ意味
dwReserve0	予約
dwReserve1	予約
dwReserve2	予約
dwReserve3	予約
dReserve0	予約
dReserve1	予約
dReserve2	予約
dReserve3	予約

# REPORT\_PACK

Multi device reporting 向け。積算情報の読み出し用の構造体です。

## C/C++

```

struct REPORT_PACK
{
    double          dDailyReport[MAX_AI_CH][MAX_MFIO];
    double          dMonthlyReport[MAX_AI_CH][MAX_MFIO];
    double          dTotal[MAX_AI_CH][MAX_MFIO];
    DWORD          dwDailyCount[MAX_MFIO];
    DWORD          dwMonthlyCount[MAX_MFIO];
    DWORD          dwTotalCount[MAX_MFIO];
    double          dDailyAverage[MAX_AI_CH][MAX_MFIO];
    double          dMonthlyAverage[MAX_AI_CH][MAX_MFIO];
    double          dTotalAverage[MAX_AI_CH][MAX_MFIO];
    SYSTEMTIME      sPastTime;
    DWORD          dwReserve0;
    DWORD          dwReserve1;
    DWORD          dwReserve2;
    double          dReserve0;
    double          dReserve1;
    double          dReserve3;
};

```

## メンバ変数

dDailyReport	1 日分のデータの合計 (アナログも積算される)これが MAX_AI_CH チャンネル×MAX_MFIO 台数分並んでいます。配列[MAX_MFIO]は CARD_ID を表し、配列[MAX_AI_CH]はチャンネル番号を表します。
dMonthlyReport	1 ヶ月分のデータの合計 (アナログも積算される)これが MAX_AI_CH チャンネル×MAX_MFIO 台数分並んでいます。配列[MAX_MFIO]は CARD_ID を表し、配列[MAX_AI_CH]はチャンネル番号を表します。
dTotal	過去データの総合計 (アナログも積算される)これが MAX_AI_CH チャンネル×MAX_MFIO 台数分並んでいます。配列[MAX_MFIO]は CARD_ID を表し、配列[MAX_AI_CH]はチャンネル番号を表します。
dwDailyCount	1 日分のデータの積算回数 (アナログの平均を求めるのに必要)これが MAX_MFIO 台数分並んでいます。配列[MAX_MFIO]は CARD_ID を表します。
dwMonthlyCount	1 ヶ月分のデータの積算回数 (アナログの平均を求めるのに必要)これが MAX_MFIO 台数分並んでいます。配列[MAX_MFIO]は CARD_ID を表します。
dwTotalCount	過去データの総積算回数 (アナログの平均を求めるのに必要)これが MAX_MFIO 台数分並んでいます。配列[MAX_MFIO]は CARD_ID を表します。
dDailyAverage	1 日分のデータの平均 (カウンタでは意味がない)これが MAX_AI_CH チャンネル×MAX_MFIO 台数分並んでいます。配列[MAX_MFIO]は CARD_ID を表し、配列[MAX_AI_CH]はチャンネル番号を表します。
dMonthlyAverage	1 ヶ月分のデータの平均 (カウンタでは意味がない)これが MAX_AI_CH チャンネル×MAX_MFIO 台数分並んでいます。配列[MAX_MFIO]は CARD_ID を表し、配列[MAX_AI_CH]はチャンネル番号を表します。
dTotalAverage	過去データの総平均 (カウンタでは意味がない)これが MAX_AI_CH チャンネル×MAX_MFIO 台数分並んでいます。配列[MAX_MFIO]は CARD_ID を表し、配列[MAX_AI_CH]はチャンネル番号を表します。
sPastTime	積算を行った最後の日時
dwReserve0	予約
dwReserve1	予約
dwReserve2	予約
dReserve0	予約
dReserve1	予約
dReserve3	予約