

The background of the top section is a blue-tinted image of a complex printed circuit board (PCB) with various components and traces. The text is overlaid on this image.

ADIOX2-API

HardwareAccess  
Storage/Logger  
SignalCondition  
SignalAnalysis  
Graphics/Trend  
WaveGenerate

# ADIOX-MK II

MULTIFUNCTION-I/O-X2 SERIES

ADIOX2-API

REFERENCE

ADX2-42 GROUPE

UPDATE 2011-10-30

SAYA INC.

## 目次

はじめに	2
1. 初期化・再初期化・終了処理	3
2. 設定	5
3. ステータス	6
4. ポーリング	6
5. 校正	8
6. ヘルパ	9
7. 構造体	10

## はじめに

## このマニュアルについて

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

## 実装されていないボードへのアクセス

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

## 開発用ファイル

**VisualC++, C++, C 用**

64bit 系の Windows は CDROM¥MFIO\_X2¥sdk¥VisuaCPP\_X64

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

をお使いください。

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

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

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

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

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]

レ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;”という名前スペースを追加記述してください。  
.netFramework2.0 以降に対応します。

**VisualBasic 用**

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

ADIOX-API\_VB.bas/[ADiox2.dll+ ADioxScp.dll ]

メインライブラリの宣言をまとめたファイルです。(VisualBASIC プロジェクトに追加してください、.NET 用ではありません)

## CardId について

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

# 1.初期化・再初期化

## bADioxScpLoad2

複数の [ADX 42-1K-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ク確保を行います。ハードウェアの初期化は、TXBUFSETUP2、IOGEOSETUP2、TDIO\_MISC、TConfigPWM、TSetupPWM、TADIO2、SCP\_SETUP\_AIALL (センサーモード、ゼロスキャン校正位置、ゼロスキャン校正係数、スケールング、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。

接続先は、付属ソフトのMultiCardIDConf.EXE で登録し、この情報は起動フォルダのConfId.cid にファイル保存されます。複数を手動リンクするアプリケーションでは割り込みを使わず、ポーリング方式を推奨します。

**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

**引数**

hWnd bHwintr	ドライバからのメッセージを受け取るアプリケーションのウィンドウハンドルを設定します。 前記メッセージ番号を指定します。本引数 0 ~ 15 に対し、メッセージ 3028 ~ 3043 番が割り当てられます。この値は ADiox2.h、ADiox2Library.cs、ADIOX-API_VB.bas にて 3028 ~ 3043 番を、それぞれ、WM_HWINTR0 ~ WM_HWINTRF として定義しています。(末尾の 0 ~ F は本引数 0 ~ 15 の 16 進数に相当する)
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.lpcConfigFileName で指定します。“ファイルを開くダイアログボックス”の初期フォルダは IpsCharPayload.lpcInitialDir 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。

## vADioxIrqEmuration

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isable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 42-1K-ETHERNET](#) が電源遮断や通信途絶によってロスした場合、リトライをかけます。(=再接続と初期化を試みる)これは複数のポート I/O でデータ収集中に、一部のポート I/O がロスしても、残りのポート I/O で運用を続行した場合に有効です。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 bHwintr	ドライバからのメッセージを受け取るアプリケーションのウィンドウハンドルを設定します。 前記メッセージ番号を指定します。本引数 0 ~ 15 に対し、メッセージ 3028 ~ 3043 番が割り当てられます。この値は ADiox2.h、ADiox2Library.cs、ADIOX-API_VB.bas にて 3028 ~ 3043 番を、それぞれ、WM_HWINTR0 ~ WM_HWINTRF として定義しています。(末尾の 0 ~ F は本引数 0 ~ 15 の 16 進数に相当する)
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 42-1K-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してしまいます。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 42-1K-ETHERNET が起動できたかどうか確認します。複数の ADX 42-1K-ETHERNET のうち生存中の ADX 42-1K-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 42-1K-ETHERNET を確認できます。ロストしている ADX 42-1K-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

<b>引数</b>	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。

## bADioxScpStore2

複数の ADX 42-1K-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、ゼロスピン校正位置、ゼロスピン校正係数、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

<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、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。

## 2.設定

### bADioxScpSetup2

信号調節設定 (SCP\_SETUP\_AICH 構造体の内容) をハードウェアに反映します。この関数をコールすると、ゼロ スピン校正が解除されますので、ご注意ください。もしゼロ・スピン校正を解除したくない場合には、ADioxScpSetupEX2 を使用してください。

**C/C++** BOOL bADioxScpSetup2(struct SCP\_SETUP\_AICH sScpSetupAich, BYTE bCardID);

**C#** int bADioxScpSetup2(SCP\_SETUP\_AICH sScpSetupAich, byte bCardID);

**VB** Declare Function bADioxScpSetup2 Lib "ADiox2.dll" ( \_  
ByRef sScpSetupAich As SCP\_SETUP\_AICH, ByVal bCardID As Byte) As Long

**引数** sScpSetupAich シグナルコンディション設定を格納したSCP\_SETUP\_AICH 構造体を指定します。  
bCardID ターゲットデバイスのカードID。

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

### bADioxScpSetupEX2

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

**C/C++** BOOL bADioxScpSetupEX2(struct SCP\_SETUP\_AIALL sScpSetupAich, BYTE bCardID);

**C#** int bADioxScpSetupEX2(SCP\_SETUP\_AIALL sScpSetupAich, byte bCardID);

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

**引数** sScpSetupAich シグナルコンディション設定を格納したSCP\_SETUP\_AICH 構造体を指定します。  
bCardID ターゲットデバイスのカードID。

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

### bADioxScpDefault2

引数で指定したカードID、チャンネル番号、センサー種別に対する、適切な信号調節設定のデフォルト値を生成します。このデフォルト値は、DLL 内部のSCP\_SETUP2 構造体に設定されるとともに、第4引数以降から取得することができます。本関数は値の取得だけで、ハードウェアへの反映は行われないので注意してください。

**C/C++** BOOL bADioxScpDefault2(DWORD dwSensorMode, DWORD dwDeviceNumber,  
DWORD dwCH, struct SCP\_SETUP \* lpsSCP\_SETUP);

**C#** int bADioxScpDefaultCS2(uint dwSensorMode, uint dwDeviceNumber,  
uint dwCH, ref SCP\_SETUP lpsSCP\_SETUP);

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

**引数** dwSensorMode センサー種別  
dwDeviceNumber ターゲットデバイスのカードID (bCardID と同じです)  
dwCH アナログ入力チャンネル番号  
lpsSCP\_SETUP ドライバ内部 SCP\_SETUP2 構造体のポインタ。  
VB ではSCP\_SETUP2 構造体を3分割したポインタとなります。( )

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

## 3. ステータス

### 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用することができます。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

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。
```

## 4. ポーリング

### bADioxADIO2

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

```
C/C++ BOOL bADioxADIO2 ( struct TADIO2 * 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。変数により入力と
                                出力に分かれます。
    bCardID             ターゲットデバイスのカードID。
戻り値 成功すると1(TRUE)、失敗すると0(FALSE)が返ります。
```

## bADioxChannelAIw2

チャンネルを指定してアナログ入力値を取得します。サンプリング周期とは非同期にデータを読み出せるので、読み出し周期より先、サンプリング速度が遅いと、同じ値を何度も読みつづけることとなります。そのため、ある程度早いサンプリング速度に設定することを推奨します。

**C/C++** BOOL bADioxChannelAIw2 ( struct IOGEOSETUP2 \*IpsIoGeoSetup,  
LPDWORD lpdwData, int iInterval, BYTE bCardID);

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

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

**引数**

IpsIoGeoSetup	アナログデジタル入出力インターフェース設定値が入ったIOGEOSETUP2 構造体を指定します。
lpdwData	取得したいアナログチャンネルもここで指定します。
iInterval	取得されたアナログ入力値を格納したポインタ。下位 16Bit のみ有効で上位は 0 です。アナログチャンネルを指定してから、アナログ入力値を読み込むまでの時間を指定してください。(msec)値が小さいと、チャンネルの切り替えが完了せず、正確なデータが得られません。アナログ入力の信号源がバッファされていても35msec 以上にしないと干渉する恐れがあります。
bCardID	ターゲットデバイスのカードID。

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

## bADioxDI

デジタル入力値を取得します。

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

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

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

**引数**

lpdwDI	デジタル入力値を保持したポインタ
bCardID	ターゲットデバイスのカードID。

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

## bADioxDO

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

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

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

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

**引数**

dwDO	デジタル出力値。
bCardID	ターゲットデバイスのカードID。

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

## dADioxLinCoef

bADioxChannelAIw2 で取得したDWORD 型のアナログ入力値を本関数に引き渡すと、ゼロ校正、スリット校正、リアライザ、スケーリング、アラーム、バーンアウト検出などの処理を行い、結果を返します。設定内容はドライバ内部の SCP\_SETUP 構造体から bCardID, bChannel に相当する部分を取り出して適用します。

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

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

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

**引数**

dwANALOG	bADioxADIO2 や、bADioxChannelAIw2 で取得したDWORD 型のアナログ入力値をここにセットしてください。
lpbBurnOut	バーンアウト検出フラグで BOOL 型のポインタです。TRUE(=1)でバーンアウト発生。
lpdwAlarm	アラーム検出フラグで BOOL 型のポインタです。TRUE(=1)でアラーム発生。
bCardID	ターゲットデバイスのカードID。
bChannel	アナログ入力チャンネル番号

**戻り値** 変換されたアナログ値



## 6. ヘルパ

### bADioxDefaultInit

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

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

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

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

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

<b>引数</b>	*lpsTBUFSETUP	デフォルト値を格納したTBUFSETUP2 構造体へのポインタ
	*lpsIOGEOSETUP	デフォルト値を格納したIOGEOSETUP2 構造体へのポインタ
	*lpsTDIO_MISC	デフォルト値を格納したTDIO_MISC 構造体へのポインタ
	*lpsTConfigPWM	デフォルト値を格納したTConfigPWM 構造体へのポインタ
	*lpsTSetupPWM	デフォルト値を格納したTSetupPWM 構造体へのポインタ
	*lpsTADIO	デフォルト値を格納したTADIO2 構造体へのポインタ
	bCardID	ターゲットデバイスのカードID。

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

### dADioxCalcFreq

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

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

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

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

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

**戻り値** 周波数(KHz)が返ります。

### dwADioxSrcStatus

ドライバ、ハードウェアで発生したエラー回数を返します。エラー回数には回復に成功したものも含まれます。この数値が増大している場合、ネットワーク障害が発生する恐れがあります。

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

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

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

<b>引数</b>	dwCommand	0 を指定してください。
-----------	-----------	--------------

**戻り値** エラー発生回数が返ります。

## 7. 構造体

### 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となります。従来のADIOx シリーズに比べ大幅に強化されました。

#### C/C++

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

#### C#

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

## VB

```

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

```

End Type

## メンバ変数 (以下で示されていないメンバ変数は使用されません)

## 【サンプリング周波数】

dwClockScall サンプル時間を設定します。シーケンシャル取り込み数を  $n$  とした場合のサンプリング時間は  
 デジタルフィルタオフ  $16.672224\text{ns} \times \text{dwClockScall} \times n$   
 デジタルフィルタオン  $66.688896\text{ns} \times \text{dwClockScall} \times n$   
 となります。

## 【トリガモード】

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

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

## 【トリガソース】

dwAnalogTrigSourceStart アナログスタートトリガソースを指定します。以下の中から選択できます。  
 dwAnalogTrigSourceStop アナログストップトリガソースを指定します。以下の中から選択できます。  
 0: AI 1: AO0 2: AO1  
 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: カウンタエペア( )  
 IRQ\_BUFFER.dwDI\_CT\_interrupt の Bit0-15 に相当する値を DI に見立ててトリガソースとします。

## 【アナログトリガ】

dwStartTrigLevel1 アナログレベルトリガ・アナログエリアトリガ用のスタートトリガレベル1 の指定。  
 (アナログレベル最小 ~ 最大が 0 ~ 0xFFFF に対応します)  
 dwStartTrigLevel2 アナログレベルトリガ・アナログエリアトリガ用のスタートトリガレベル2 の指定。  
 (アナログレベル最小 ~ 最大が 0 ~ 0xFFFF に対応します)  
 dwStopTrigLevel1 アナログレベルトリガ・アナログエリアトリガ用のストップトリガレベル1 の指定。  
 (アナログレベル最小 ~ 最大が 0 ~ 0xFFFF に対応します)  
 dwStopTrigLevel2 アナログレベルトリガ・アナログエリアトリガ用のストップトリガレベル2 の指定。  
 (アナログレベル最小 ~ 最大が 0 ~ 0xFFFF に対応します)

**【デジタルエッジトリガ】**

dwStartTrigSourceDI\_ch デジタルエッジスタートトリガ用のチャンネル指定。何チャンネル目のデジタル入出力でエッジトリガをかけるか設定します。0-31 のいずれかを指定してください。

dwStopTrigSourceDI\_ch デジタルエッジストップトリガ用のチャンネル指定。何チャンネル目のデジタル入出力でエッジトリガをかけるか設定します。0-31 のいずれかを指定してください。

**【デジタルパターントリガ】**

dwStartMask デジタルパターンスタートトリガ用のマスク。この変数のビットフィールドが、デジタル入出力のチャンネルに相当します。例えばBit5(0x20)は、デジタル入出力チャンネル5に相当します。該当ビットが1でマスク、0でアンマスクです。

dwStopMask デジタルパターンストップトリガ用のマスク。この変数のビットフィールドが、デジタル入出力のチャンネルに相当します。例えばBit5(0x20)は、デジタル入出力チャンネル5に相当します。該当ビットが1でマスク、0でアンマスクです。

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

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

**【トリガ遅延・プリトリガ】**

dwStartTrigDelay スタートトリガ遅延の指定。(プリトリガ)この値がナトリガより先送れて、データの取り込みを開始します。最大 65535 まで指定できます。

dwStopTrigDelay ストップトリガ遅延の指定。(プリトリガ)この値がナトリガより先送れて、データの取り込みを終了します。最大 65535 まで指定できます。

dwPreTrigger スタートポストトリガの on/off を指定します。1 で on、0 で off です。トリガより先に、データの取り込みを開始します。何サンプル先になるかは、ハードウェアの仕様を参照願います。

**【同期運転】**

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

**【トリガ・リングバッファ開始停止・自動停止】**

dwInterruptMode DMA\_INT、NOT\_INT で開始、NOT\_INT で停止となります。

dwStopCounterValue この値で指定した分のバンクチェンジが発生すると自動停止します。プライマリオンチップリングバッファまたは、リングバッファの容量に本変数を乗算したサンプル数で自動停止します。0 を指定すると本自動停止機能は無効となり、停止トリガもしくは停止コマンドが実施されるまで無制限にデータ収集を行います。

dwDeadTime スタートトリガ有効後、ストップトリガ検出が有効になるまでの時間を指定します。いかなるストップトリガがかわってしまうのを防ぐためです。値はサンプル数で、0-0xFFFFFFFF まで有効です。

**【その他様々な機能】**

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

dwAoHspBufferd AO をリングバッファ経由にするか否か。0 でポーリング、1 でリングバッファ経由。  
リングバッファ経由にできる AO チャンネルはハードウェアにより固定されています。

dwDoHspBufferd DO を経由にするか否か。0 でポーリング、1 でリングバッファ経由。  
リングバッファ経由にできる DO チャンネルはハードウェアにより固定されています。

bConnectBuffer エンコーダカウンタをリングバッファ経由にするか否かを指定します。0 でしない、1 でカウンタCH0、2 でカウンタCH0+CH1、3 でカウンタCH0+CH1+CH2+CH3 をリングバッファ経由にします。エンコーダカウンタのリングバッファ経由をしようしている間はDIはリングバッファ経由になりません。

bTrigEnable TRUE(=1)でトリガイネーブルが有効になります。FALSE(=0)で無効になります。

## IOGEOSETUP2

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

### C/C++

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

### C#

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

### VB

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

### メンバ変数 (以下で示されていないメンバ変数は使用されません)

dwAo0Mode	アナログ出力チャンネル0の動作モードを指定します。動作モードは以下の3つから選択してください。
dwAo1Mode	アナログ出力チャンネル1の動作モードを指定します。動作モードは以下の3つから選択してください。
dwAo2Mode	アナログ出力チャンネル2の動作モードを指定します。動作モードは以下の3つから選択してください。
dwAo3Mode	アナログ出力チャンネル3の動作モードを指定します。動作モードは以下の3つから選択してください。
<b>NO_LINK</b>	エンコーダカウンターとは連動しない、通常のアナログ出力モード。関数 bADioxADIO で指定したアナログ出力値が採用されます。
<b>LIVE CTC</b>	エンコーダカウンターの現在値をアナログ出力値にします。同一チャンネル数のカウンタの値とリンクします。
<b>LATCH CTC</b>	エンコーダカウンターのラッチ値をアナログ出力値にします。同一チャンネル数のカウンタの値とリンクします。
dwCOB	この値が1の場合、アナログ入力値はコンPLEMENTバイナリ(2の補数=short型相当)となります。この値が0の場合、アナログ入力値はストレートバイナリ(WORD型相当)となります。両者の変換はハードウェアで行われるので、負荷がかかりません。

dwFilterEnable	アナログ入力信号へのデジタルフィルタの切り替えを行います。本変数を0とすることで、フィルタをオフ1または3とすることで4次移動平均フィルタをオンにします。
dwNoiseShaper	この値が0の場合、アナログ入力信号へのデジタルフィルタのノイズシェーパーをOFFにします。この値が1の場合、アナログ入力信号へのデジタルフィルタのノイズシェーパーをONにします。ノイズシェーパーは5次16次ローパスフィルタでのみ有効です。ノイズシェーパーはフィルターによるビット落ちを積算して最下位ビットに加算します。
dwMux	入力チャンネルを切り替えます。0-7が有効な値です。
dwChatCan	この値が0の場合、デジタル入力のチャタリングキャンセラー(フィルタ)をOFFにします。この値が1の場合、デジタル入力のチャタリングキャンセラー(フィルタ)をONにします。

## TDIO\_MISC

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

C/C++

```

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

```

C#

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

**VB**

```

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

```

## メモリ変数 (以下で示されていないメモリ変数は使用されません)

dwStrobeInternal	ストロブ出力をデジタル出力チャンネル15に埋め込むか否かを設定します。1で埋め込み、0で埋め込まない(ストロブ専用ヘッダネクタまたはピンを使用)
dwDinIntMode16	DI0のエッジ割込み要因を指定します。
dwDinIntMode17	DI1のエッジ割込み要因を指定します。
dwDinIntMode18	DI2のエッジ割込み要因を指定します。
dwDinIntMode19	DI3のエッジ割込み要因を指定します。
dwDinIntMode20	DI4のエッジ割込み要因を指定します。
dwDinIntMode21	DI5のエッジ割込み要因を指定します。
dwDinIntMode22	DI6のエッジ割込み要因を指定します。
dwDinIntMode23	DI7のエッジ割込み要因を指定します。
dwDinIntMode24	DI8のエッジ割込み要因を指定します。
dwDinIntMode25	DI9のエッジ割込み要因を指定します。
dwDinIntMode26	DI10のエッジ割込み要因を指定します。
dwDinIntMode27	DI11のエッジ割込み要因を指定します。
dwDinIntMode28	DI12のエッジ割込み要因を指定します。
dwDinIntMode29	DI13のエッジ割込み要因を指定します。
dwDinIntMode30	DI14のエッジ割込み要因を指定します。
dwDinIntMode31	DI15のエッジ割込み要因を指定します。 < dwDinIntMode16~31で指定する割込み要因の種類 > <b>NO_INT</b> 割込み要因なし <b>POSEDGE_INT</b> 立ち上がりエッジ <b>NEGEDGE_INT</b> 立下りエッジ <b>DUALEDGE_INT</b> デュアルエッジ
dwCounterMode_A	エンコーダカウンタ0の動作モードを以下の0-7で指定します。
dwCounterMode_B	エンコーダカウンタ1の動作モードを以下の0-7で指定します。
dwCounterMode_C	エンコーダカウンタ2の動作モードを以下の0-7で指定します。
dwCounterMode_D	エンコーダカウンタ3の動作モードを以下の0-7で指定します。 < dwCounterMode_A~Dで指定するエンコーダカウンタモード > 0:4倍速エンコーダカウンタ、Z相未使用 1:4倍速エンコーダカウンタ、Z相使用 2:2倍速エンコーダカウンタ、Z相未使用 3:2倍速エンコーダカウンタ、Z相使用 4:1倍速エンコーダカウンタ、Z相未使用 5:1倍速エンコーダカウンタ、Z相使用 6:アップダウンカウンタ(パルスカウンタ)、Z相未使用 7:アップダウンカウンタ(パルスカウンタ)、Z相使用
dwLatchMode_A	カウンタ0ラッチモードを以下の3つの中から指定します。
dwLatchMode_B	カウンタ1ラッチモードを以下の3つの中から指定します。
dwLatchMode_C	カウンタ2ラッチモードを以下の3つの中から指定します。
dwLatchMode_D	カウンタ3ラッチモードを以下の3つの中から指定します。 < dwLatchMode_A~Dで指定するラッチモード > <b>SOFT</b> ソフトウェアラッチ <b>Z_PHASE</b> Z相条件成立でラッチ <b>DI_SEL</b> Y相立ち上がりエッジでラッチ
dwZ_CENTER_A	カウンタ0、Z相条件成立モード(カウンタリセット)を以下の0-1のいずれかで指定してください。
dwZ_CENTER_B	カウンタ1、Z相条件成立モード(カウンタリセット)を以下の0-1のいずれかで指定してください。
dwZ_CENTER_C	カウンタ2、Z相条件成立モード(カウンタリセット)を以下の0-1のいずれかで指定してください。
dwZ_CENTER_D	カウンタ3、Z相条件成立モード(カウンタリセット)を以下の0-1のいずれかで指定してください。 < dwZ_CENTER_A ~ Dで指定するカウンタリセットモード > 1:Z相が1、B相が1、A相の立ち上がりエッジの3条件で、カウンタリセット 0:Z相立ち上がりエッジ条件で、カウンタリセット
dwSoftwareClear_A	dwLatchMode_AをSOFTとした場合、この変数が1でカウンタリセット、0で非リセット。
dwSoftwareClear_B	dwLatchMode_BをSOFTとした場合、この変数が1でカウンタリセット、0で非リセット。
dwSoftwareClear_C	dwLatchMode_CをSOFTとした場合、この変数が1でカウンタリセット、0で非リセット。
dwSoftwareClear_D	dwLatchMode_DをSOFTとした場合、この変数が1でカウンタリセット、0で非リセット。
dwCompareMode	0を指定するとカウンタ比較値LOWは、dwReferenceLow_A、dwReferenceLow_B、dwReferenceLow_C、dwReferenceLow_Dを使用する。1を指定すると隣接カウンタ(若し方)のカウンタ値をカウンタ比較値LOWとして利用する。カウンタ0のみカウンタ3の値を使う

dwLinkCounterToDO_A	カウンタ0 のコペア結果を DO にリンクするか否かを指定します。0x0-0xF が有効な値で、各ビットフィールドは以下の意味を持ちます。
dwLinkCounterToDO_B	カウンタ1 のコペア結果を DO にリンクするか否かを指定します。0x0-0xF が有効な値で、各ビットフィールドは以下の意味を持ちます。
dwLinkCounterToDO_C	カウンタ2 のコペア結果を DO にリンクするか否かを指定します。0x0-0xF が有効な値で、各ビットフィールドは以下の意味を持ちます。
dwLinkCounterToDO_D	カウンタ3 のコペア結果を DO にリンクするか否かを指定します。0x0-0xF が有効な値で、各ビットフィールドは以下の意味を持ちます。 < dwLinkCounterToDO_A ~ D のビットフィールド > bit0= コペア値 LOW と一致 bit1= コペア値 LOW 以上 bit2= コペア値 LOW 以下 bit3= コペア値 LOW ~ コペア値 HIGH の範囲内 カウンタコペア出力と DO の割り付けはハードウェア仕様書(カウンタの章)に記載。
dwCounterIntMode_A	カウンタ0 のコペア結果を割り込みにリンクするか否かを指定します。0x0-0xF が有効な値で、各ビットフィールドは以下の意味を持ちます。
dwCounterIntMode_B	カウンタ1 のコペア結果を割り込みにリンクするか否かを指定します。0x0-0xF が有効な値で、各ビットフィールドは以下の意味を持ちます。
dwCounterIntMode_C	カウンタ2 のコペア結果を割り込みにリンクするか否かを指定します。0x0-0xF が有効な値で、各ビットフィールドは以下の意味を持ちます。
dwCounterIntMode_D	カウンタ3 のコペア結果を割り込みにリンクするか否かを指定します。0x0-0xF が有効な値で、各ビットフィールドは以下の意味を持ちます。 < dwCounterIntMode_A ~ D のビットフィールド > bit0= コペア値 LOW と一致 bit1= コペア値 LOW 以上 bit2= コペア値 LOW 以下 bit3= コペア値 LOW ~ コペア値 HIGH の範囲内
dwReferenceLow_A	カウンタ0 コペア値 LOW (32Bit カウンタなので 0 ~ 0xFFFFFFFF を指定して下さい)
dwReferenceLow_B	カウンタ1 コペア値 LOW (32Bit カウンタなので 0 ~ 0xFFFFFFFF を指定して下さい)
dwReferenceLow_C	カウンタ2 コペア値 LOW (32Bit カウンタなので 0 ~ 0xFFFFFFFF を指定して下さい)
dwReferenceLow_D	カウンタ3 コペア値 LOW (32Bit カウンタなので 0 ~ 0xFFFFFFFF を指定して下さい)
dwReferenceHigh_A	カウンタ0 コペア値 HIGH (32Bit カウンタなので 0 ~ 0xFFFFFFFF を指定して下さい)
dwReferenceHigh_B	カウンタ1 コペア値 HIGH (32Bit カウンタなので 0 ~ 0xFFFFFFFF を指定して下さい)
dwReferenceHigh_C	カウンタ2 コペア値 HIGH (32Bit カウンタなので 0 ~ 0xFFFFFFFF を指定して下さい)
dwReferenceHigh_D	カウンタ3 コペア値 HIGH (32Bit カウンタなので 0 ~ 0xFFFFFFFF を指定して下さい)
dwSetGate	周波数カウンタのゲートを指定します。ゲートとは、パルスをカウントする時間です。1 秒の場合、カウントした値はそのまま Hz(ヘルツ)になります。以下の 0-3 のいずれかを指定して下さい。 0:1sec、1:100msec、2:10msec、3:1msec

[カウンタコペア出力の DO 出力の割り付け]

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

## TADIO2

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

### C/C++

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

**C#**

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

**VB**

```

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

```

**メモリ変数 (以下で示されていないメモリ変数は使用されません)**

dwAi0	アナログ入力値が格納されます。値は0~0xFFFFの16Bitコードが入ります。
dAi0	アナログ入力値が格納されます。値は単位 mV の電圧値が入ります。
dwAo0-1	ログ出力チャンネル0-1 出力値 (最小~最大が 0~0xFFFF に対応します ) dwAo0-1 がチャンネル0~1 に相当します。
dAo0-1	前記 dwAo0-1 アナログ出力値を電圧に変換した値が格納されます。単位は mV になります。 dwAo0-1 が dAo0-1 に相当します。
dwDOS	デジタル出力チャンネル0~15 の出力値 (Bit0~Bit15 がデジタル出力チャンネル0~15 に対応します )
dwDI	デジタル入力チャンネル0~15 の入力値 (Bit0~Bit15 がデジタル入力チャンネル0~15 に対応します )
dwDI_Latch	デジタル入力チャンネル0~15 のストロブラッチ値 (Bit0~Bit15 がデジタル入力チャンネル0~15 に対応します )
dwCounterA	エンコーダカウンタ0 のライブ値 (現在の値 1)
dwCounterB	エンコーダカウンタ1 のライブ値 (現在の値 1)
dwCounterC	エンコーダカウンタ2 のライブ値 (現在の値 1)
dwCounterD	エンコーダカウンタ3 のライブ値 (現在の値 1)
dwLatchA	エンコーダカウンタ0 のラッチ値 1
dwLatchB	エンコーダカウンタ1 のラッチ値 1
dwLatchC	エンコーダカウンタ2 のラッチ値 1
dwLatchD	エンコーダカウンタ3 のラッチ値 1
dwFreqLatch_A	周波数カウンタ0 のライブ値 (現在の値 2)
dwFreqLatch_B	周波数カウンタ1 のライブ値 (現在の値 2)
dwFreqLatch_C	周波数カウンタ2 のライブ値 (現在の値 2)
dwFreqLatch_D	周波数カウンタ3 のライブ値 (現在の値 2)
dwMode	必ず 0 をセットしてください。

- 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。
- ゲート周期に何サイクルあったかを表します。

## TStatusPack2

システムの稼動状態を格納します。

**C/C++**

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

**C#**

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

**VB**

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

**メモリ変数 (以下で示されていないメモリ変数は使用されません)**

dwADO_underrun	AO/DO バッファアンダーフローステータス。アンダーフローが発生した場合、バッファへの書き込みが無かつたため、送るべきデータが無かつたことを意味します。この場合前の値が保持されています。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_overrun	AI/DI バッファオーバーフローステータス。オーバーフローが発生した場合、バッファからの読み出しが無かつたため、バッファにデータを収容しきれなくなつたことを意味します。(データが失われた)以下の定義された数値が格納されます。 <b>OVERRUN_BUFFER</b> : バッファオーバーフローの発生 (エラー) 上記以外 : 問題なし
dwTrigSeq	トリガの状態を以下の0-4で表します 0: アイドル状態 1: 稼動状態 2: 停止状態に遷移中 3: 最終バンクの転送待ち 4: ストップトリガのデッドタイム
上記以外の変数	全て予約

**SAYA\_DEVICE\_INFO**

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
```

**メモリ変数 (以下で示されていないメモリ変数は使用されません)**

dwDeviceType	デバイスの種類(4)が入ります。
dwBufferSizeOfByte	1ングバッファ(プライマリオンチップ1ングバッファ)のByte(8Bit)単位サイズが入ります。
dwBufferSizeOfDWORD	1ングバッファ(プライマリオンチップ1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

## 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

```

### メモリ変数 (以下で示されていないメモリ変数は使用されません)

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

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

## SCP\_SETUP2

複数の ADX 42-1K-ETHERNET の各種構造体を内包する大規模構造体です。1 台の ADX 42-1K-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・スケール・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\_ID0 ~ 31 までの 32 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 64 k 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** (以下で示されていないメンバ変数は使用されません)

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	シグナルエディクションコントロールレジスタ設定値 1 を MAX_MFIO 個格納します。 (アプリケーションでは参照はできても変更してはなりません)
dwLDO2	シグナルエディクションコントロールレジスタ設定値 2 を MAX_MFIO 個格納します。 (アプリケーションでは参照はできても変更してはなりません)
doZeroPos	ゼロ校正位置を 'MAX_MFIO x MAX_AI_CH' 個格納します。
doSpanPos	スパン校正位置を 'MAX_MFIO x MAX_AI_CH' 個格納します。
doZero_Coefficient	ゼロ校正係数を 'MAX_MFIO x MAX_AI_CH' 個格納します。
doSpan_Coefficient	スパン校正係数を 'MAX_MFIO x MAX_AI_CH' 個格納します。
bScalling	スケーリングする場合、TRUE(=1) しない場合 FALSE(=0) をセットします。 これを 'MAX_MFIO x MAX_AI_CH' 個格納します。
dScallingRatio	スケーリング係数を 'MAX_MFIO x 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) を 'MAX_MFIO x MAX_AI_CH' 個格納します。
dOutBottomScall	変換後のスケーリング基準値 (下) を 'MAX_MFIO x MAX_AI_CH' 個格納します。
dInTopScall	変換前のスケーリング基準値 (上) を 'MAX_MFIO x MAX_AI_CH' 個格納します。
dInBottomScall	変換前のスケーリング基準値 (下) を 'MAX_MFIO x MAX_AI_CH' 個格納します。
bAlarmModeA	アラームモードを指定します。0 を指定するとオス 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 x MAX_AI_CH" 個格納します。
dAlarmUpper	アラーム設定値 (上) を 'MAX_MFIO x MAX_AI_CH' 個格納します。
dAlarmLower	アラーム設定値 (下) を 'MAX_MFIO x 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。		
	NOT_USE	0	シグナルコンディショニング未使用
	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/500 終端
	I_4_20EX	23	電流 4-20mA/350 終端
	I_4_20EX2	26	電流 4-20mA/47 終端 ADX 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
	< 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_5V	20	電圧 +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 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ルリングを使うことができます。

## SCP\_SETUP\_AICH

信号調節関連の設定を格納します。SCP\_SETUP2 構造体ではなく、体構造体を使うことによりターゲットデバイス1個分のメモリ変数を変更できるのでセキュリティが高まります。構造体定義の配列番号 MAX\_MFIO はターゲットデバイスのカードIDを示します。

### C/C++

```
struct SCP_SETUP_AICH
{
    DWORD          dwSensorMode[MAX_AI_CH];
    DWORD          dwLDO[MAX_AI_CH];
    double         doZeroPos[MAX_AI_CH];
    double         doSpanPos[MAX_AI_CH];
    BOOL           bScalling[MAX_AI_CH];
    double         dOutTopScall[MAX_AI_CH];
    double         dOutBottomScall[MAX_AI_CH];
    double         dInTopScall[MAX_AI_CH];
    double         dInBottomScall[MAX_AI_CH];
    DWORD          bAlarmMode[MAX_AI_CH];
    double         dAlarmUpper[MAX_AI_CH];
    double         dAlarmLower[MAX_AI_CH];
};
```

### C#

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

### VB

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

### メモリ変数 (以下で示されていないメモリ変数は使用されません)

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

## SCP\_SETUP\_AIALL

SCP\_SETUP\_AICH に、校正係数の doZero\_Coefficient と doSpan\_Coefficient を加えたもの。ドライバ内部の SCP\_SETUP に対して強制的に校正係数を与えることができる。

### C/C++

```
struct SCP_SETUP_AICH
{
    DWORD          dwSensorMode[MAX_AI_CH];
    DWORD          dwLDO[MAX_AI_CH];
    double         doZeroPos[MAX_AI_CH];
    double         doSpanPos[MAX_AI_CH];
    double         doZero_Coefficient[MAX_AI_CH];
    double         doSpan_Coefficient[MAX_AI_CH];
    BOOL           bScalling[MAX_AI_CH];
    double         dOutTopScall[MAX_AI_CH];
    double         dOutBottomScall[MAX_AI_CH];
    double         dInTopScall[MAX_AI_CH];
    double         dInBottomScall[MAX_AI_CH];
    DWORD          bAlarmMode[MAX_AI_CH];
    double         dAlarmUpper[MAX_AI_CH];
    double         dAlarmLower[MAX_AI_CH];
};
```

### C#

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

### VB

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

### メンバ変数

doZeroPos[MAX\_AI\_CH]                      ゼロ校正係数。vADioxScpCopy2 関数などから取得する必要があります。

doSpanPos[MAX\_AI\_CH]                      スパン校正係数。vADioxScpCopy2 関数などから取得する必要があります。

上記以外のメンバ変数は SCP\_SETUP\_AICH を参照願います。

## CharPayloadC

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

### C/C++

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

### C#

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

### VB

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

### メンバ変数

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

## LOG\_FRONTEND

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

### C/C++

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

**C#**

```

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

```

**VB**

```

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

```

**メンバ変数 (以下で示されていないメンバ変数は使用されません)**

dwHeaderCode	ファイルヘッダ識別コード、必ず 0x41594154 をセットしてください。
dwDeviceName	機種コード(SAYA_DEVICE_INFO.dwDeviceType)
dwBufaSize	リングバッファサイズ
dClockScall	サンプリング周波数(Hz)
bBitScall	量子化ビット数
bAI_ChannelScall AI	チャンネル数
bDI_ChannelScall DI	チャンネル数
DataType	保存形式(DWORD=0,double=1)double 保存はADX 42-1K-ETHERNETのみ
dwGetYear	計測開始の年
dwGetMonth	計測開始の月
dwGetDay	計測開始の日
dwGetHour	計測開始の時
dwGetMinute	計測開始の分
dwGetSecond	計測開始の秒
dwGetMilliseconds	計測開始のミ秒