GPC-6502

レゾルバ D/R インタフェースモジュール用 Windows ドライバソフトウェア

Help for Windows

www.interface.co.jp

目　次

第1章　はじめに

1.1　概要

GPC-6502 は、Windows 上のアプリケーションから、弊社レゾルバ D/R 製品を制御する為のソフトウェアです。
弊社レゾルバ D/R 製品を Windows 上のアプリケーションから DLL をダイナミックリンクし、API をコールすることにより制御します。
本ドキュメントは、Windows 上で GPC-6502 を使用するための情報を掲載しています。

弊社レゾルバ D/R 製品は、1 相励磁／2 相出力レゾルバのエミュレータとして使用します。
ソフトウェアで指定した角度に応じて、入力波形をレゾルバの 2 相出力に変換する機能を持っています。

1.2　特長

●制御体系
レゾルバ D/R 製品を制御するシンプルなAPI体系となっており、簡単な手順でプログラムを作成できます。
また、それぞれの製品には下記の特長があります。

・一定角度出力と一定速度出力の二つの出力モードに対応しています。
・パターン出力を行うことができます。

●イベントコールバック
各割り込みなどの要因で発生するイベントコールバックに対応し、非同期イベントによる処理が設定できます。

●サンプルプログラム
各種機能毎にシンプルなサンプルプログラムを用意しています。
ソースを公開していますので、自由にプログラムを拡張することができます。

●ドキュメント
機能や使い方を説明するヘルプ（Help.pdf）をサポートしていますので、開発中に関数の詳細説明など、簡単に参照することができます。

第2章　製品仕様

2.1　基本仕様

最大デバイス数	16 枚（識別可能枚数）：256 枚（制御可能枚数）
基本機能	・レゾルバ信号出力（一定角度／一定速度） ・パターン出力（FIFO/FLASH ROM） ・異常信号出力 ・汎用出力 ・汎用入力
割り込み機能	割り込みが発生した場合に、アプリケーションにイベント（又はコールバック関数）にて通知することができます。 本 DLL は、以下の割り込み要因が使用可能です。 ・順方向角度割り込み 1 発生 ・逆方向角度割り込み 1 発生 ・順方向角度割り込み 2 発生 ・逆方向角度割り込み 2 発生 ・順方向角度割り込み 3 発生 ・逆方向角度割り込み 3 発生 ・アンダーラン時に割り込み発生 ・パターン出力停止検出 ※割り込み時にシグナル状態となるイベントハンドル（又はコールバック関数）は本 DLL の関数にて設定します。

2.2 製品構成

製品構成	ファイル名	説明
弊社管理用ファイル	gpc6502.ver	弊社ソフト管理用ファイル
最新情報ドキュメント	readme.htm	最新ドキュメント
インストールプログラム	setup.exe	インストールプログラム
サンプルプログラム Visual C# .NET Visual C++ Visual Basic .NET Visual Basic Delphi	Angle Velocity Output Pattern Noise InputDi OutputDo Event Callback	各種言語用サンプルプログラム
DLL	ifdr.dll	ダイナミックリンクライブラリ
	ifdr.lib	インポートライブラリ
デバイスドライバ	cp6502.sys	Windows 2000 以降用ドライバ
	gpc6502.inf	Windows 2000 以降用 ドライバ インストールファイル
	gpc6502.sld	Windows XP Embeddedn 用 ドライバ SLD ファイル
ヘッダファイル	ifdr.h	Visual C++用ヘッダファイル
	ifdr.bas	Visual Basic 用ヘッダファイル
	ifdr.pas	Delphi 用ヘッダファイル
ヘルプ	help.pdf	ヘルプ (PDF 形式)

※Visual C# .NET, Visual Basic.NET 用サンプルプログラムは、それぞれ Viausl C# .NET 2003, Visual Basic .NET 2003 を使用して作成しています。

第3章　導入方法

3.1　インストール手順

readme.htm のインストール方法を参照してください。

3.2　実行手順

同一型式を複数枚使用する場合は、インタフェースモジュール上のロータリースイッチ RSW1 の設定値が同一型式同士で重複しないように設定してからスロットに実装し、システムを起動して下さい。
同一型式が複数存在する場合、制御対象を一意に識別するための番号となります。重複していた場合、本ソフトウェアは正常に動作いたしません。

基本的な制御の手順は以下の通りです（記述例は C 言語です)。

GPC-6502 の API を C 言語から使用する場合、ソースコードの先頭部分で関数定義が記載されたヘッダファイル（ifdr.h）をインクルードする必要があります。

```
#include "ifdr.h"
```

各種 API 関数を使用する為には、レゾルバ D/R デバイスを DrOpen 関数で初期化します。
初期化が正常終了後、取得したデバイスハンドルを各種 API に指定することで、レゾルバ D/R デバイスの制御が可能となります。
オープンに失敗した場合は戻り値として INVALID_HANDLE_VALUE（-1）が返されますので、戻り値を確認してエラー処理を行います。

処理が完了後、DrClose 関数でデバイスの終了処理を行います。アプリケーションでは、必ずクローズ処理を行ってから終了してください。

```
HANDLE DeviceHandle;
INT Ret;

DeviceHandle = DrOpen( “IFDR1” );
if (DeviceHandle == INVALID_HANDLE_VALUE)
{
    // オープンエラー時の処理
    return -1;
}
  :
Ret = DrClose(DeviceHandle);
if (Ret != IFDR_ERROR_SUCCESS)
{
    return Ret;
}
```

※以降の記載例では説明の為にエラー処理を外して手順を簡略化しています。
実際のプログラムを作成する際は、戻り値を確認してエラー処理を行ってください。

1. 一定角度出力

出力角度を指定してレゾルバ D/R から出力を開始し、出力リレーを ON にすることで出力が有効になります。

出力角度の指定は 0000h～FFFFh の値が指定できます。
出力角度に指定する 16bit の設定値は下記の式で算出できます。

出力角度：
16bit 設定値 ＝ 出力角度（°） ÷ 360 × 10000h

下記の例では 180° を指定しています。
180 ÷ 360 × 10000h ＝ 8000h

```
HANDLE DeviceHandle;
INT Ret;
DRCONFIG Config;

// IFDR1 をオープン
DeviceHandle = DrOpen( “IFDR1” );

// DR 情報を取得
Ret = DrGetConfig (DeviceHandle, 1,Config);

// DR 情報の設定
Ret = DrSetConfig(DeviceHandle, 1,Config);

// 一定角度出力設定（位相ズレ補正なし、180° ）
Ret = DrOutputAngle(DeviceHandle, 1, 0x8000);

// 10 秒間のスリープ
Sleep(10000);

// デバイスをクローズ
Ret = DrClose(DeviceHandle);
```

2. 一定速度出力

更新レートと更新レートに対する速度を指定すると一定速度出力で出力されます。
一定速度出力が停止する場合には、更新レートに対する速度を 0 に設定します。

・速度（単位：LSB/更新周期）※角度更新周期1us時：LSB/us

ビット	15	14～4	3～0
	符号	小数点より上	小数点以下

※負の値は2の補数表記とする。
例　－123.25LSB/us
$-123.25 \times 2^4 = -1972 = 1111100001001100b$

```
HANDLE DeviceHandle;
INT Ret;

// IFDR1 をオープン
DeviceHandle = DrOpen(“IFDR1”);

// DR 情報を取得
Ret = DrGetConfig (DeviceHandle, 1,Config);

// DR 情報の設定
Ret = DrSetConfig(DeviceHandle, 1,Config);

// 一定速度出力開始
Ret = DrStartVelocity(DeviceHandle, 1, 100, 0x100);

// 10 秒間のスリープ
Sleep(10000);

// デバイスをクローズ
Ret = DrClose(DeviceHandle);
```

3.パターン出力(FIFO)

DrSetConfig 関数で DR の条件設定/パターン出力条件を設定します。

DrSetOutputData 関数でパターン出力データの設定を行います。

DrStartOutput 関数で開始位置を 0 に設定し、更新レート/件数/繰り返し回数を指定し実行するとパターン出力が開始されます。

DrStopOutput 関数を実行するとパターン出力が強制停止されます。

```
HANDLE DeviceHandle;
INT Ret;
DWORD Data[360];

// IFDR1 をオープン
DeviceHandle = DrOpen( “IFDR1” );

// DR 情報を取得
Ret = DrGetConfig(DeviceHandle, 1,Config);

// DR 情報の設定
Ret = DrSetConfig(DeviceHandle, 1,Config);

// パターン出力データの設定
Ret = DrSetOutputData(hDeviceHandle, 1, 360, Data);

// パターン出力開始
Ret = DrStartOutput(DeviceHandle, 1, 1, 1, 1024, 10);


// パターン出力強制停止
Ret = DrStopOutput(DeviceHandle, 1);

// デバイスをクローズ
Ret = DrClose(DeviceHandle);
```

4.パターン出力(FLASH ROM)

DrSetConfig 関数で DR の条件設定/パターン出力条件を設定します。

DrSetFlashRomData 関数でパターン出力データの設定を行います。
イベントもしくはコールバック関数を用いて FLASH ROM への書き込みが完了するまで待ちます。
DrGetStatus 関数をポーリングして確認することもできます。

DrStartOutput 関数で開始位置/更新レート/件数/繰り返し回数を指定し実行するとパターン出力が開始されます。

DrStopOutput 関数を実行するとパターン出力が強制停止されます。

```
int FlashWriteFlag;
DWORD User;
void CALLBACK EventProc(INT Channel, DWORD Event, PVOID UserData)
{
   // 割り込み発生時の処理
   FlashWriteFlag = 1;
}
```

```
HANDLE DeviceHandle;
INT Ret;
DWORD Data[360];
DRSTATUS Status;

// IFDR1 をオープン
DeviceHandle = DrOpen( “IFDR1” );

// DR 情報を取得
Ret = DrGetConfig(DeviceHandle, 1,Config);

// DR 情報の設定
Ret = DrSetConfig(DeviceHandle, 1,Config);

FlashWriteFlag = 0;
User = 0x01;

// パターン出力データの設定
Ret = DrSetFlashRomData(hDeviceHandle, 1, 360, Data,
                            NULL, (LPDRCALLBACK)EventProc, (PVOID) User);

// 書き込み完了まで待ちます。
while (FlashWriteFlag == 0) {
    Sleep(100);
}
```

```
/*
// DrGetStatus 関数でポーリングする場合。
do{
    Ret = DrGetStatus(hDeviceHandle, 1, &Status);
} while(Status. FlashStatus == 0x01);
*/

// パターン出力開始
Ret = DrStartOutput(DeviceHandle, 1, 1, 1, 1024, 10);


// パターン出力強制停止
Ret = DrStopOutput(DeviceHandle, 1);

// デバイスをクローズ
Ret = DrClose(DeviceHandle);
```

5.異常信号出力

DrSetNoiseData 関数で異常信号出力データの設定を行います。
単位は LSB で指定し最上位ビットが符号として設定できます。
1LSB あたり　20/65536=0.305mV となります。

DrStartNoise 関数で開始位置を 0 に設定し、更新レート/件数/繰り返しを指定し実行すると異常信号出力が開始されます。

DrStopNoise 関数を実行するとパターン出力が強制停止されます。

```
HANDLE DeviceHandle;
INT Ret;
DWORD Data;

// IFDR1 をオープン
DeviceHandle = DrOpen( “IFDR1” );

// 異常信号データの設定
Ret = DrSetNoiseData(DeviceHandle, 1, 256, Data);

// 異常信号出力開始
Ret = DrStartNoise(DeviceHandle, 1, 10, 10, IFDR_ENABLE);


// 異常信号出力強制停止
Ret = DrStopNoise(DeviceHandle, 1);

// デバイスをクローズ
Ret = DrClose(DeviceHandle);
```

6. 汎用入力

DrInputDI 関数を実行することで、指定した変数に汎用入力端子状態が格納されます。

```
HANDLE DeviceHandle;
INT Ret;
DWORD Data;

// IFDR1 をオープン
DeviceHandle = DrOpen( “IFDR1” );

// 汎用入力値を取得
Ret = DrInputDI(DeviceHandle, &Data);

// デバイスをクローズ
Ret = DrClose(DeviceHandle);
```

7. 汎用出力

DrOutputDO 関数を実行することで、指定した値で汎用出力端子状態を制御が可能です。
出力を開始後、DrClose 関数でリセットを有効にすると、汎用出力の状態は初期状態に戻ります。

```
HANDLE DeviceHandle;
INT Ret;

// IFDR1 をオープン
DeviceHandle = DrOpen( “IFDR1” );

// 汎用出力設定 (AAh)
Ret = DrOutputDO(DeviceHandle, 0xAA);

// 10 秒間のスリープ
Sleep(10000);

// デバイスをクローズ
Ret = DrClose(DeviceHandle);
```

8. 割り込みコールバック

DrSetEventMask 関数にて発生させる割り込み要因を設定います。

DrSetEvent 関数で割り込みが発生した際に呼び出されるコールバック関数を登録します。
発生させる要因が発生するとコールバック関数が呼び出される状態となります。

このコールバック関数内に処理を記述することで、パターン出力停止を通知することや、順方向角度割り込み/逆方向角度割り込みを通知することができます。

コールバック関数を解除する際は、DrKillEvent 関数を実行します。

DrSetEventMask 関数で要因を全て無効にすると要因は発生しなくなります。

```
void CALLBACK EventProc(INT Channel, DWORD Event, PVOID UserData)
{
   // 割り込み発生時の処理
}
```

```
HANDLE DeviceHandle;
INT Ret;

// IFDR1 をオープン
DeviceHandle = DrOpen(“IFDR1”);

// 割り込みマスクの設定
Ret = DrSetEventMask(DeviceHandle, 1, 0x1000);

// コールバック関数の登録
Ret = DrSetEvent(DeviceHandle, NULL, (LPDRCALLBACK)EventProc, 0);



// コールバック関数の解除
Ret = DrKillEvent(DeviceHandle);

// デバイスをクローズ
Ret = DrClose(DeviceHandle);
```

9. 割り込みイベント

DrSetEventMask 関数にて発生させる割り込み要因を設定います。

CreateEvent でイベントハンドルを作成し DrSetEvent 関数で登録することで設定した要因が発生した際にイベントが有効になります。

登録したイベントハンドルは WaitForSingleObject などを使用して、ハンドルが有効になるのを待つことが可能です。

イベントハンドルの解除を行う場合は DrKillEvent 関数を実行します。

```
HANDLE DeviceHandle;
HANDLE EventHandle = CreateEvent(NULL, FALSE, FALSE, NULL);
INT Ret;

// IFDR1 をオープン
DeviceHandle = DrOpen( “IFDR1” );

// 割り込みマスクの設定
Ret = DrSetEventMask(DeviceHandle, 1, 0x1000);

// イベントハンドルの登録
Ret = DrSetEvent(DeviceHandle, EventHandle, NULL, 0);

// イベントハンドルが有効になるのを待つ（10 秒間）
if (WaitForSingleObject(EventHandle, 10000) == WAIT_OBJECT_0)
{
    // 割り込み発生時の処理
}

// コールバック関数の解除
Ret = DrKillEvent(DeviceHandle);

// デバイスをクローズ
Ret = DrClose(DeviceHandle);
```

3.3 .NET 用クラスライブラリの参照方法

本製品では、クラスライブラリのソースファイルを用意しています。
ソースコードをビルドしてクラスライブライリを生成し、参照することで、DLL 関数の定義を容易にすることができます。
（DLL 関数の呼び出しをカスタマイズしたい場合は、クラスライブラリのソースを参照してください）

1.クラスライブラリの作成方法

.NET において DLL 関数を呼び出すには、まずクラスライブラリを用意する必要があります。

Visual C# .NET の場合

Visual Studio を起動し、以下のプロジェクトファイルを開きます。
＜インストール先＞¥interface¥GPC6502¥samples¥CS_NET¥IFCDR¥IFCDR.csproj
このプロジェクトをビルドすると、bin フォルダにクラスライブラリ ifcdr.dll が作成されます。

Visual Basic .NET の場合

Visual Studio を起動し、以下のプロジェクトファイルを開きます。
＜インストール先＞¥interface¥GPC6502¥samples¥VB_NET¥IFCDR¥IFCDR.vbproj
このプロジェクトをビルドすると、bin フォルダにクラスライブラリ ifcdr.dll が作成されます。

2.クラスライブラリの参照

Visual Studio のメニューの「プロジェクト」の「参照の追加」を選択してください。
「参照」ボタンをクリックして参照したいクラスライブラリ DLL を指定します。
例）
＜インストール先＞¥interface¥GPC6502¥samples¥CS_NET¥IFCDR¥bin¥Release¥ifcdr.dll
＜インストール先＞¥interface¥GPC6502¥samples¥VB_NET¥IFCDR¥bin¥Relaase¥ifcdr.dll

「選択されたコンポーネント」に指定した DLL が表示されます。
「OK」ボタンをクリックすると設定は完了です。

次にソースのヘッダで各言語毎に下記のように InterfaceCorpDllWrap の名前空間を追加すれば DLL 関数を次章の「使用例」の方法で呼び出すことができるようになります。

Visual C# .NET の場合

```
using     InterfaceCorpDllWrap;
```

Visual Basic .NET の場合

```
Imports   InterfaceCorpDllWrap
```

第4章　リファレンス

4.1　関数一覧

No	関数名	機能
●初期化/終了処理		
1	DrOpen	D/R デバイスのオープン、初期化を行います。
2	DrClose	D/R デバイスのクローズ、終了処理を行います。
●D/R 制御		
3	DrGetDeviceInfo	デバイス情報の取得を行います。
4	DrGetStatus	ステータスの取得を行います
5	DrGetConfig	D/R の情報を取得します。
6	DrSetConfig	D/R の情報を設定します。
●角度/速度制御関数		
7	DrOutputAngle	一定角度出力の設定を行います。
8	DrOutputVelocity	一定速度出力の設定を行います。
●パターン出力関数		
9	DrSetOutputData	パターン出力データの設定を行います。
10	DrSetFlashRomData	パターン出力データを FlashRom に書き込みます。
11	DrSetOutputRate	パターン出力の更新レートを変更します。
12	DrStartOutput	パターン出力の開始を発行します。
13	DrStopOutput	パターン出力の停止を発行します。
●異常信号出力関数		
14	DrSetNoiseData	異常信号出力のデータの設定を行います。
15	DrStartNoise	異常信号出力の開始を発行します。
16	DrStopNoise	異常信号出力の停止を発行します。
●汎用入出力関数		
17	DrInputDI	汎用入力の取得を行います。
18	DrOutputDO	汎用出力の制御を行います。
●コールバック/イベント関連関数		
19	DrSetEventMask	割り込みイベントマスクの設定を行います。
20	DrGetEventMask	割り込みイベントマスクの取得を行います。
21	DrSetEvent	割り込みイベント要因、割り込みコールバック・イベントの登録を行います。
22	DrKillEvent	割り込みコールバック・イベントの解除を行います。
23	DrGetEventStatus	割り込みイベント要因を取得します。

4.2　関数個別説明

1. DrOpen

【機能】

デバイスオープンを行い、以後のアクセスを行えるようにします。

【書式】

●C 言語

```
HANDLE DrOpen(
  LPCSTR     DeviceName
);
```

●Visual Basic

```
Declare Function DrOpen Lib "ifdr.dll"( _
  ByVal   DeviceName     As String
)As Long
```

●Delphi

```
function DrOpen(
  DeviceName:    String;
):THandle; stdcall; external 'ifdr.dll';
```

●Visual C#.NET

```
[DllImport("ifdr.dll")]
public static extern IntPtr DrOpen(
  String     DeviceName
);
```

●Visual Basic.NET

```
Declare Function DrOpen Lib "ifdr.dll"( _
  ByVal   DeviceName     As String
)As Integer
```

【パラメータ】

DeviceName

オープンするデバイス名を指定してください。

【戻り値】

戻り値	条件
有効なハンドル	正常終了
INVALID_HANDLE_VALUE (-1)	デバイス名が異常 すでにオープンされている

【備考】

●複数プロセスからの共有について
同じデバイスを 2 つ以上のアプリケーションで共有することはできません。

●デバイス名について
オープンするにはデバイス名を指定します。
レゾルバ D/R 製品のデバイス名は「IFDRx」(x は 1～255) となります。デバイス名は、ドライバが認識した順番に割り当てられます。
複数枚ご使用になる場合は、制御するインタフェースモジュールのデバイス名とボード ID の対応を事前にご確認ください。
各デバイス名とボード ID の確認は、デバイスマネージャにて確認できます。

【使用例】

●C 言語

```
HANDLE DeviceHandle;

DeviceHandle = DrOpen( “IFDR1” );
```

●Visual Basic

```
Dim DeviceName As String
Dim DeviceHandle As Long

DeviceName = "IFDR" & Chr( 0 )
DeviceHandle = DrOpen(DeviceName)
```

●Delphi

```
var
DeviceName: String;
DeviceHandle: THandle;

DeviceName := 'IFDR1';
DeviceHandle := DrOpen(DeviceName);
```

●Visual C#.NET

```
IntPtr DeviceHandle;
DeviceHandle = IFCDR.DrOpen( “IFDR1” );
```

●Visual Basic.NET

```
Dim DeviceName As String
Dim DeviceHandle As IntPtr

DeviceName = "IFDR"
DeviceHandle = IFCDR.DrOpen(DeviceName)
```

デバイス名「IFDR1」のデバイスをオープンします。

2. DrClose

【機能】

デバイスをクローズします。
デバイスアクセスのために使用されていた各種リソースの解放を行い、以後のデバイスへのアクセスを禁止します。

【書式】

●C 言語

```
INT DrClose(
   HANDLE         DeviceHandle
);
```

●Visual Basic

```
Declare Function DrClose Lib "ifdr.dll"( _
   ByVal    DeviceHandle     As Long _
)As Long
```

●Delphi

```
function DrClose (
   DeviceHandle:   Thandle
): Integer; stdcall; external 'ifdr.dll';
```

●Visual C#.NET

```
[DllImport("ifdr.dll")]
public static extern uint DrClose(
   IntPtr    DeviceHandle
);
```

●Visual Basic.NET

```
Declare Function DrClose Lib "ifdr.dll"( _
   ByVal    DeviceHandle     As IntPtr _
)As Integer
```

【パラメータ】

DeviceHandle

DrOpen 関数で取得したデバイスハンドルを指定してください。

【戻り値】

DrClose 関数は正常に終了すると IFDR_ERROR_SUCCESS を返します。
それ以外の場合は IFDR_ERROR_SUCCESS 以外の値を返します。
IFDR_ERROR_SUCCESS 以外の値が返された場合については、戻り値一覧を参照してください。

【備考】

●DrClose 未実行時の終了処理

DrClose 関数を実行せずにアプリケーションが終了した場合は、デバイスハンドルの解放と共にデフォルト動作（リレーOFF、設定情報の初期化）が実行されます。
再度、デバイスへのアクセスを行う場合にはオープン処理(DrOpen 関数)を呼び出してください。

【使用例】

●C 言語

```
INT Ret;
HANDLE DeviceHandle;

DeviceHandle = DrOpen( “IFDR1” );
   :
   :
Ret = DrClose(DeviceHandle);
```

●Visual Basic

```
Dim Ret As Long
Dim DeviceName As String
Dim DeviceHandle As Long

DeviceName = "IFDR1" & Chr( 0 )
DeviceHandle = DrOpen(DeviceName)
   :
   :
Ret = DrClose(DeviceHandle)
```

●Delphi

```
var
DeviceName: String;
DeviceHandle: THandle;

DeviceName := 'IFDR1';
DeviceHandle := DrOpen(DeviceName);
   :
   :
Ret := DrClose(DeviceHandle);
```

●Visual C#.NET

```
uint Ret;
IntPtr DeviceHandle;

DeviceHandle = IFCDR.DrOpen( “IFDR1” );
   :
   :
```

```
Ret = IFCDR.DrClose(DeviceHandle);
```

●Visual Basic.NET

```
Dim Ret As Integer
Dim DeviceName As String
Dim DeviceHandle As IntPtr

DeviceName = "IFDR1"
DeviceHandle = IFCDR.DrOpen(DeviceName)
   :
   :
Ret = IFCDR.DrClose(DeviceHandle)
```

デバイス名「IFDR1」のデバイスのクローズ処理を行います。

3. DrGetDeviceInfo

【機能】

デバイス情報を取得します。

【書式】

●C 言語

```
INT DrGetDeviceInfo (
   HANDLE       DeviceHandle,
   PDWORD       DeviceID,
   PDWORD       SubsystemID,
   PDWORD       BoardID,
   PDWORD       DeviceConfig
);
```

●Visual Basic

```
Declare Function DrGetDeviceInfo Lib "ifdr.dll"( _
   ByVal   DeviceHandle    As Long, _
   ByRef   DeviceID        As Long, _
   ByRef   SubsystemID     As Long, _
   ByRef   BoardID         As Long, _
   ByRef   DeviceConfig    As Long _
)As Long
```

●Delphi

```
function DrGetDeviceInfo (
   DeviceHandle:         Thandle;
   var DeviceID:         Dword;
   var SubsystemID:      Dword;
   var BoardID:          Dword;
   var DeviceConfig:     Dword
): Integer; stdcall; external 'ifdr.dll';
```

●Visual C#.NET

```
[DllImport("ifdr.dll")]
public static extern uint DrGetDeviceInfo(
   IntPtr       DeviceHandle,
   out uint     DeviceID,
   out uint     SubsystemID,
   out uint     BoardID,
   out uint     DeviceConfig
);
```

●Visual Basic.NET

```
Declare Function DrGetDeviceInfo Lib "ifdr.dll"( _
   ByVal   DeviceHandle    As IntPtr, _
   ByRef   DeviceID        As Integer, _
   ByRef   SubsystemID     As Integer, _
```

```
    ByRef    BoardID          As Integer, _
    ByRef    DeviceConfig     As Integer _
)As Integer
```

【パラメータ】

DeviceHandle

DrOpen 関数で取得したデバイスハンドルを指定してください。

DeviceID

デバイス ID を取得する変数への参照を指定します。

SubsystemID

サブシステム ID を取得する変数への参照を指定します。

BoardID

ボード ID を取得する変数への参照を指定します。

DeviceConfig

デバイスの機能情報を取得する変数への参照を指定します。

bit0～bit3：D/R 最大チャンネル数（0～15）

bit4 ：予約
bit5 ：予約
bit6 ：16bit 分解能（0：無し、1：有り）
bit7 ：予約

bit8 ：一定角度動作（0：無し、1：有り）
bit9 ：一定速度動作（0：無し、1：有り）
bit10：予約
bit11：予約

bit12～bit15：予約

bit16：DI（8bit）（0：無し、1：有り）
bit17～bit19：予約

bit20：DO（8bit）（0：無し、1：有り）
bit21～bit23：予約

bit24：インターバルタイマ（0：無し、1：有り）
bit25～bit31：予約

【戻り値】

DrGetDeviceInfo 関数は正常に終了すると IFDR_ERROR_SUCCESS を返します。
それ以外の場合は IFDR_ERROR_SUCCESS 以外の値を返します。
IFDR_ERROR_SUCCESS 以外の値が返された場合については、戻り値一覧を参照してください。

【使用例】

●C 言語

```
INT Ret;
HANDLE DeviceHandle;
DWORD DeviceID;
DWORD SubsystemID;
DWORD BoardID;
DWORD DeviceConfig;

Ret = DrGetDeviceInfo(DeviceHandle, &DeviceID, &SubsystemID,
                              &BoardID, &DeviceConfig);
```

●Visual Basic

```
Dim Ret As Long
Dim DeviceHandle As Long
Dim DeviceID As Long
Dim SubsystemID As Long
Dim BoardID As Long
Dim DeviceConfig As Long

Ret = DrGetDeviceInfo(DeviceHandle, DeviceID, SubsystemID, BoardID, DeviceConfig )
```

●Delphi

```
Ret : Integer;
DeviceHandle : THandle;
DeviceID : Dword;
SubsystemID : Dword;
BoardID : Dword;
DeviceConfig : Dword;

Ret := DrGetDeviceInfo(DeviceHandle, DeviceID, SubsystemID, BoardID, DeviceConfig);
```

●Visual C#.NET

```
uint Ret;
IntPtr DeviceHandle;
uint DeviceID;
uint SubsystemID;
uint BoardID;
uint DeviceConfig;

Ret = IFCDR.DrGetDeviceInfo(DeviceHandle, out DeviceID, out SubsystemID,
                              out BoardID, out DeviceConfig);
```

●Visual Basic.NET

```
Dim Ret As Integer
Dim DeviceHandle As IntPtr
Dim DeviceID As Integer
Dim SubsystemID As Integer
Dim BoardID As Integer
Dim DeviceConfig As Integer

Ret = IFCDR.DrGetDeviceInfo(DeviceHandle, DeviceID, SubsystemID,_
                            BoardID, DeviceConfig )
```

デバイスハンドル DeviceHandle のデバイス情報を取得します。

4. DrGetStatus

【機能】

ステータス・出力角度を取得します。

【書式】

●C 言語

```
INT DrGetStatus(
  HANDLE          DeviceHandle,
  INT             Channel,
  PDRSTATUS       Status
);
```

●Visual Basic

```
Declare Function DrGetStatus Lib "ifdr.dll"( _
   ByVal   DeviceHandle    As Long, _
   ByVal   Channel         As Long, _
   ByRef   Status          As DRSTATUS _
)As Long
```

●Delphi

```
function DrGetStatus (
   DeviceHandle:         Thandle;
   Channel:              Dword;
   var Status:           DRSTATUS
): Integer; stdcall; external 'ifdr.dll';
```

●Visual C#.NET

```
[DllImport("ifdr.dll")]
public static extern uint DrGetStatus (
   IntPtr          DeviceHandle,
   int             Channnel,
   out DRSTATUS    Status
);
```

```
[DllImport("ifdr.dll")]
public static extern uint DrGetStatus (
   IntPtr          DeviceHandle,
   int             Channnel,
   IntPtr          Status
);
```

●Visual Basic.NET

```
Declare Function DrGetStatus Lib "ifdr.dll"( _
    ByVal   DeviceHandle    As IntPtr, _
    ByVal   Channel         As Integer, _
    ByRef   Status          As DRSTATUS_
)As Integer
```

```
Declare Function DrGetStatus Lib "ifdr.dll"( _
    ByVal   DeviceHandle    As IntPtr, _
    ByVal   Channel         As Integer, _
    ByVal   Status          As IntPtr_
)As Integer
```

【パラメータ】

DeviceHandle

DrOpen 関数で取得したデバイスハンドルを指定してください。

Channel

チャンネルを指定します。

Status

DR 状態を取得する構造体 (DRSTATUS) のポインタを指定します

【戻り値】

DrGetStatus 関数は正常に終了すると IFDR_ERROR_SUCCESS を返します。
それ以外の場合は IFDR_ERROR_SUCCESS 以外の値を返します。
IFDR_ERROR_SUCCESS 以外の値が返された場合については、戻り値一覧を参照してください。

【使用例】

●C 言語

```
INT Ret;
DRSTATUS Status;

Ret = DrGetStatus(DeviceHandle, 1, &Status);
```

●Visual Basic

```
Dim Ret As Long
Dim DeviceHandle As Long
Dim Status As DRSTATUS

Ret = DrGetStatus(DeviceHandle, 1, Status)
```

●Delphi

```
Ret : Integer;
DeviceHandle : THandle;
Status : DRSTATUS;
```

```
Ret := DrGetStatus(DeviceHandle, 1, Status);
```

●Visual C#.NET

```
uint Ret;
IntPtr DeviceHandle;
IFCDR.DRSTATUS Status;

Ret = IFCDR.DrGetStatus(DeviceHandle, 1, ref Status);
```

●Visual Basic.NET

```
Dim Ret As Integer
Dim DeviceHandle As IntPtr
Dim Status As IFCDR.DRSTATUS

Ret = IFCDR.DrGetStatus(DeviceHandle, 1, Status)
```

デバイスハンドル DeviceHandle の DR 状態ステータスを取得します。

5. DrGetConfig

【機能】

D/R の情報を取得します。

【書式】

●C 言語

```
INT DrGetConfig (
  HANDLE          DeviceHandle,
  INT             Channel,
  PDRCONFIG       Config
);
```

●Visual Basic

```
Declare Function DrGetConfig Lib " ifdr.dll "( _
   ByVal   DeviceHandle    As Long _
   ByVal   Channel         As Long, _
   ByRef   Config          As DRCONFIG_
)As Long
```

●Delphi

```
function DrGetConfig (
   DeviceHandle:     THandle;
   Channel:          Integer;
   var Config:       DRCONFIG
): Integer; stdcall; external 'ifdr.dll ';
```

●Visual C# .NET

```
[DllImport("ifdr.dll ")]
public static extern uint DrGetConfig (
   IntPtr            DeviceHandle,
   uint              Channel,
   out DRCONFIG      Config
);
```

●Visual Basic .NET

```
Declare Function DrGetConfig Lib "ifdr.dll "( _
   ByVal   DeviceHandle    As IntPtr, _
   ByVal   Channel         As Integer, _
   ByRef   Config          As DRCONFIG
)As Integer
```

【パラメータ】

DeviceHandle

DrOpen 関数で取得したデバイスハンドルを指定してください。

Channel

チャンネルを指定します。

Config
D/R 設定情報の構造体（DRCONFIG）の変数へのポインタを指定します。

【戻り値】
DrGetConfig 関数は正常に終了すると IFDR_ERROR_SUCCESS を返します。
それ以外の場合は IFDR_ERROR_SUCCESS 以外の値を返します。
IFDR_ERROR_SUCCESS 以外の値が返された場合については、戻り値一覧を参照してください。

【使用例】
●C 言語

```
INT Ret;
DRCONFIG Config;

Ret = DrGetConfig(DeviceHandle, 1,Config);
```

●Visual Basic

```
Dim Ret As Long
Dim DeviceHandle As Long
Dim Config As DRCONFIG

Ret = DrGetConfig (DeviceHandle, Config)
```

●Delphi

```
var
Ret: Integer;
DeviceHandle: THandle;
Config: DRCONFIG;

begin
Ret := DrGetConfig (DeviceHandle, Config);
end;
```

●Visual C# .NET

```
int Ret;
uint DeviceHandle;
IFCDR.DRCONFIG Config;

Ret = IFCDRDrGetConfig (DeviceHandle, ref Config);
```

●Visual Basic .NET

```
Dim Ret As Integer
Dim DeviceHandle As Integer
Dim Config As IFCDR. DRCONFIG

Ret = IFCDR.DrGetConfig(DeviceHandle, Config)
```

デバイスハンドル DeviceHandle の DR 設定情報を取得します。

6. DrSetConfig

【機能】

D/R の情報を設定します。

【書式】

●C 言語

```
INT DrSetConfig (
  HANDLE          DeviceHandle,
  INT             Channel,
  PDRCONFIG       Config
);
```

●Visual Basic

```
Declare Function DrSetConfig Lib "ifdr.dll "( _
   ByVal   DeviceHandle        As Long _
   ByVal   Channel             As Long, _
   ByRef   Config              As DRCONFIG_
)As Long
```

●Delphi

```
function DrSetConfig (
   DeviceHandle:     THandle;
   Channel:          Integer;
   var Config:       DRCONFIG
): Integer; stdcall; external 'ifdr.dll ';
```

●Visual C# .NET

```
[DllImport(" IFDR.DLL ")]
public static extern uint DrSetConfig (
   IntPtr          DeviceHandle,
   uint            Channel,
   ref DRCONFIG    Config
);
```

●Visual Basic .NET

```
Declare Function DrSetConfig Lib "ifdr.dll "( _
   ByVal   DeviceHandle        As IntPtr, _
   ByVal   Channel             As Integer, _
   ByRef   Config              As DRCONFIG
)As Integer
```

【パラメータ】

DeviceHandle

DrOpen 関数で取得したデバイスハンドルを指定してください。

Channel

チャンネルを指定します。

Config
D/R 設定情報の構造体（DRCONFIG）の変数へのポインタを指定します。

【戻り値】
DrSetConfig 関数は正常に終了すると IFDR_ERROR_SUCCESS を返します。
それ以外の場合は IFDR_ERROR_SUCCESS 以外の値を返します。
IFDR_ERROR_SUCCESS 以外の値が返された場合については、戻り値一覧を参照してください。

【備考】
・本関数はパターン出力中には実行することはできません。
・位相ズレ補正有りを指定した場合、DrGetStatus を実行して位相ロック状態になるのを待つ必要があります。

【使用例】
●C 言語

```
INT Ret;
DRCONFIG Config;

Ret = DrGetConfig (DeviceHandle, 1,Config);

Config.Ratio = 0x8000;
Config.PhaseGapMode = IFDR_ENABLE;
Config.MisJoin = IFDR_JOIN;
Config.Origin = 0x0000;
Config.Hysteresis[0].HighRotation = 0x0000;
Config.Hysteresis[0].HighAngle= 0x0000;
Config.Hysteresis[0].LowRotation= 0x0000;
Config.Hysteresis[0].LowAngle= 0x0000;
Config.Hysteresis[1].HighRotation = 0x0000;
Config.Hysteresis[1].HighAngle= 0x0000;
Config.Hysteresis[1].LowRotation= 0x0000;
Config.Hysteresis[1].LowAngle= 0x0000;
Config.Hysteresis[2].HighRotation = 0x0000;
Config.Hysteresis[2].HighAngle= 0x0000;
Config.Hysteresis[2].LowRotation= 0x0000;
Config.Hysteresis[2].LowAngle= 0x0000;
Config.TrgOutMode = 0;

Ret = DrSetConfig(DeviceHandle, 1,Config);
```

●Visual Basic

```
Dim Ret As Long
Dim DeviceHandle As Long
Dim Config As DRCONFIG

Ret = DrGetConfig (DeviceHandle, Config)

Config.Ratio = &H8000
Config.PhaseGapMode = IFDR_ENABLE
Config.MisJoin = IFDR_JOIN
Config.Origin = &H0000
Config.Hysteresis(0).HighRotation = &H0000
Config.Hysteresis(0).HighAngle= &H0000
Config.Hysteresis(0).LowRotation= &H0000
Config.Hysteresis(0).LowAngle= &H0000
Config.Hysteresis(1).HighRotation = &H0000
Config.Hysteresis(1).HighAngle= &H0000
Config.Hysteresis(1).LowRotation= &H0000
Config.Hysteresis(1).LowAngle= &H0000
Config.Hysteresis(2).HighRotation = &H0000
Config.Hysteresis[2].HighAngle= &H0000
Config.Hysteresis[2].LowRotation= &H0000
Config.Hysteresis[2].LowAngle= &H0000
Config.TrgOutMode = &H0000

Ret = DrSetConfig (DeviceHandle, Config)
```

●Delphi

```
var
Ret: Integer;
DeviceHandle: THandle;
Config: DRCONFIG;

begin

Ret := DrGetConfig (DeviceHandle, Config);

Config.Ratio := $8000;
Config.PhaseGapMode := IFDR_ENABLE;
Config.MisJoin := IFDR_JOIN;
Config.Origin := $0000;
Config.Hysteresis[0].HighRotation := $0000;
Config.Hysteresis[0].HighAngle := $0000;
Config.Hysteresis[0].LowRotation := $0000;
Config.Hysteresis[0].LowAngle := $0000;
Config.Hysteresis[1].HighRotation := $0000;
```

```
Config.Hysteresis[1].HighAngle := $0000;
Config.Hysteresis[1].LowRotation := $0000;
Config.Hysteresis[1].LowAngle := $0000;
Config.Hysteresis[2].HighRotation := $0000;
Config.Hysteresis[2].HighAngle := $0000;
Config.Hysteresis[2].LowRotation := $0000;
Config.Hysteresis[2].LowAngle := $0000;
Config.TrgOutMode := $0;

Ret := DrSetConfig (DeviceHandle, Config);
end;
```

●Visual C# .NET

```
int Ret;
uint DeviceHandle;
IFCRD.DRCONFIG Config;

IFCDR.Config.Ratio = 0x8000;
IFCDR.Config.PhaseGapMode = IFCDR.IFDR_ENABLE;
IFCDR.Config.MisJoin = IFCDR.IFDR_JOIN;
IFCDR.Config.Origin = 0x0000;
IFCDR.Config.Hysteresis[0].HighRotation = 0x0000;
IFCDR.Config.Hysteresis[0].HighAngle= 0x0000;
IFCDR.Config.Hysteresis[0].LowRotation= 0x0000;
IFCDR.Config.Hysteresis[0].LowAngle= 0x0000;
IFCDR.Config.Hysteresis[1].HighRotation = 0x0000;
IFCDR.Config.Hysteresis[1].HighAngle= 0x0000;
IFCDR.Config.Hysteresis[1].LowRotation= 0x0000;
IFCDR.Config.Hysteresis[1].LowAngle= 0x0000;
IFCDR.Config.Hysteresis[2].HighRotation = 0x0000;
IFCDR.Config.Hysteresis[2].HighAngle= 0x0000;
IFCDR.Config.Hysteresis[2].LowRotation= 0x0000;
IFCDR.Config.Hysteresis[2].LowAngle= 0x0000;
IFCDR.Config.TrgOutMode = 0;

Ret = IFCDR.DrSetConfig (DeviceHandle, ref Config);
```

●Visual Basic .NET

```
Dim Ret As Integer
Dim DeviceHandle As Integer
Dim Config As IFCRD.DRCONFIG

Ret = IFCDR.DrGetConfig(DeviceHandle, Config)

IFCDR.Config.Ratio = &H8000
IFCDR.Config.PhaseGapMode = IFCDR.IFDR_ENABLE
IFCDR.Config.MisJoin = IFCDR.IFDR_JOIN
```

```
IFCDR.Config.Origin = &H0000
IFCDR.Config.Hysteresis(0).HighRotation = &H0000
IFCDR.Config.Hysteresis(0).HighAngle= &H0000
IFCDR.Config.Hysteresis(0).LowRotation= &H0000
IFCDR.Config.Hysteresis(0).LowAngle= &H0000
IFCDR.Config.Hysteresis(1).HighRotation = &H0000
IFCDR.Config.Hysteresis(1).HighAngle= &H0000
IFCDR.Config.Hysteresis(1).LowRotation= &H0000
IFCDR.Config.Hysteresis(1).LowAngle= &H0000
IFCDR.Config.Hysteresis(2).HighRotation = &H0000
IFCDR.Config.Hysteresis[2].HighAngle= &H0000
IFCDR.Config.Hysteresis[2].LowRotation= &H0000
IFCDR.Config.Hysteresis[2].LowAngle= &H0000
IFCDR.Config.TrgOutMode = &H0000

Ret = IFCDR.DrSetConfig (DeviceHandle, Config)
```

デバイスハンドル DeviceHandle の DR 設定情報を設定します。

7. DrOutputAngle

【機能】

一定角度出力を開始します。

【書式】

●C 言語

```
INT DrOutputAngle (
   HANDLE      DeviceHandle,
   INT         Channel,
   DWORD       Angle
);
```

●Visual Basic

```
Declare Function DrOutputAngle Lib "ifdr.dll"( _
   ByVal   DeviceHandle        As Long, _
   ByVal   Channel             As Long, _
   ByVal   Angle               As Long _
)As Long
```

●Delphi

```
function DrOutputAngle (
   DeviceHandle:            THandle;
   Channel:                 Integer;
   Angle:                   Dword
): Integer; stdcall; external 'ifdr.dll';
```

●Visual C#.NET

```
[DllImport("ifdr.dll")]
public static extern uint DrOutputAngle (
   IntPtr    DeviceHandle,
   uint      Channel,
   uint      Angle
);
```

●Visual Basic.NET

```
Declare Function DrOutputAngle Lib "ifdr.dll"( _
   ByVal   DeviceHandle        As IntPtr, _
   ByVal   Channel             As Integer, _
   ByVal   Angle               As Integer _
)As Integer
```

【パラメータ】

DeviceHandle

DrOpen 関数で取得したデバイスハンドルを指定してください。

Channel

チャンネルを指定します。

Angle

出力角度を指定します。
設定範囲：0x0000～0xFFFF

※360° ÷ 10000h （＝ 0.0054931640625°）毎に指定可能です。
0000h：0°
0001h：0.0054931640625°
0002h：0.010986328125°
0003h：0.0164794921875°
：
FFFFh：359.9945068359375°

【戻り値】

DrOutputAngle 関数は正常に終了すると IFDR_ERROR_SUCCESS を返します。
それ以外の場合は IFDR_ERROR_SUCCESS 以外の値を返します。
IFDR_ERROR_SUCCESS 以外の値が返された場合については、戻り値一覧を参照してください。

【備考】

・本関数はパターン出力中には実行することはできません。

【使用例】

●C 言語

```
INT Ret;
HANDLE DeviceHandle;

Ret = DrOutputAngle(DeviceHandle, 1, 0x8000);
```

●Visual Basic

```
Dim Ret As Long
Dim DeviceHandle As Long

Ret = DrOutputAngle(DeviceHandle, 1, &H8000)
```

●Delphi

```
Ret : Integer;
DeviceHandle : THandle;

Ret := DrOutputAngle(DeviceHandle, 1, $8000);
```

●Visual C#.NET

```
uint Ret;
IntPtr DeviceHandle;

Ret = IFCDR.DrOutputAngle(DeviceHandle, 1,.0x8000);
```

●Visual Basic.NET

```
Dim Ret As Integer
Dim DeviceHandle As IntPtr

Ret = IFCDR.DrOutputAngle(DeviceHandle, 1, &H8000)
```

デバイスハンドル DeviceHandle のデバイスに一定角度出力の設定を行います。

8. DrOutputVelocity

【機能】

一定速度出力の設定を行います。

【書式】

●C 言語

```
INT DrOutputVelocity (
    HANDLE          DeviceHandle,
    INT             Channel,
    DWORD           OutputRate,
    DWORD           Velocity
);
```

●Visual Basic

```
Declare Function DrOutputVelocity Lib "ifdr.dll"( _
    ByVal    DeviceHandle     As Long, _
    ByVal    Channel          As Long, _
    ByVal    OutputRate       As Long, _
    ByVal    Velocity         As Long _
)As Long
```

●Delphi

```
function DrOutputVelocity (
    DeviceHandle:             THandle;
    Channel:                  Integer;
    OutputRate:               Dword;
    Velocity:                 Dword
): Integer; stdcall; external 'ifdr.dll';
```

●Visual C#.NET

```
[DllImport("ifdr.dll")]
public static extern uint DrOutputVelocity (
    IntPtr    DeviceHandle,
    int       Channel,
    uint      OutputRate,
    uint      Velocity
);
```

●Visual Basic.NET

```
Declare Function DrOutputVelocity Lib "ifdr.dll"( _
    ByVal    DeviceHandle     As IntPtr, _
    ByVal    Channel          As Integer, _
    ByVal    OutputRate       As Integer, _
    ByVal    Velocity         As Integer _
)As Integer
```

【パラメータ】

DeviceHandle

DrOpen 関数で取得したデバイスハンドルを指定してください。

Channel

チャンネルを指定します。

OutpuRate

更新レート設定を指定します。
1μs の分解能で指定できます。

値	更新レート
0x00000000	設定禁止
0x00000001	1μs
0x00000002	2μs
:	:
0x00FFFFFF	16.777215s

Velocity

出力速度を指定します。
設定範囲：0x0000～0xFFFF

【戻り値】

DrOutputVelocity 関数は正常に終了すると IFDR_ERROR_SUCCESS を返します。
それ以外の場合は IFDR_ERROR_SUCCESS 以外の値を返します。
IFDR_ERROR_SUCCESS 以外の値が返された場合については、戻り値一覧を参照してください。

【備考】

・本関数はパターン出力中には実行することはできません。
・データフォーマットは下記にようになります。
速度（単位：LSB/更新周期）※角度更新周期1μs時：LSB/μs

ビット	15	14～4	3～0
	符号	小数点より上	小数点以下

※負の値は2の補数表記とする。
例　－123.25LSB/μs
$-123.25 \times 2^4 = -1972 = 1111100001001100b$

・速度設定方法例
3000 rpm を設定する場合
① 3000 rpmは1 分間に3000 回転の為、1 秒間に換算すると、
3000 rpm ÷ 60 s = 50 回転/s

② 1 回転は65536 LSBの為、
50 回転/s × 65536 = 3276800 LSB/s

③ 更新レートを1 μsとすると、
3276800 LSB/s ÷ 1000000 = 3.2768 LSB/μs

Velocityには、小数点以下4桁までしか設定できない為、
3.25 LSB/μs(0x34)を設定することでおおおよそ3000 rpm(2975 rpm程度)の回転速度を設定することができます。

また、回転速度に対する精度を向上させる場合、
③’ 更新レートを10μsとすると、
3276800 LSB/s ÷ 100000 = 32.768 LSB/(10μs)
Velocityに32.75 LSB/μs(0x20C)を設定することでほぼ3000 rpm(2998 rpm程度)の回転速度を設定することができます。

※更新レートの値を大きくすると回転速度に対する精度は向上しますが、角度変化が粗く刻まれるため、角度の精度は低下します。

【使用例】

●C 言語

```
INT Ret;
HANDLE DeviceHandle;

Ret = DrOutputVelocity(DeviceHandle, 1, 0x0001, 0x1000);
```

●Visual Basic

```
Dim Ret As Long
Dim DeviceHandle As Long

Ret = DrOutputVelocity(DeviceHandle, 1, &H0001, &H1000)
```

●Delphi

```
Ret : Integer;
DeviceHandle : THandle;

Ret := DrOutputVelocity(DeviceHandle, 1, $0001, $1000);
```

●Visual C#.NET

```
uint Ret;
IntPtr DeviceHandle;

Ret = IFCDR.DrOutputVelocity(DeviceHandle, 1, 0x0000, 0x1000);
```

●Visual Basic.NET

```
Dim Ret As Integer
Dim DeviceHandle As IntPtr

Ret = IFCDR.DrOutputVelocity(DeviceHandle, 1, &H0000, &H1000)
```

デバイスハンドル DeviceHandle のデバイスに一定速度出力の設定を行います。

9. DrSetOutputData

【機能】

パターン出力データの設定を行います。

【書式】

●C 言語

```
INT DrSetOutputData (
  HANDLE       DeviceHandle,
  INT          Channel,
  DWORD        SetNum,
  PDWORD       Data
);
```

●Visual Basic

```
Declare Function DrSetOutputData Lib "ifdr.dll"( _
   ByVal   DeviceHandle          As Long, _
   ByVal   Channel               As Long, _
   ByVal   SetNum                As Long, _
   ByRef   Data                  As Long _
)As Long
```

●Delphi

```
function DrSetOutputData (
   DeviceHandle:             THandle;
   Channel:                  Integer;
   SetNum:                   Dword;
   var Data:                 Dword
): Integer; stdcall; external 'ifdr.dll';
```

●Visual C#.NET

```
[DllImport("ifdr.dll")]
public static extern uint DrSetOutputData (
   IntPtr      DeviceHandle,
   int         Channel,
   uint        SetNum,
   ref uint[]  Data
);
```

```
[DllImport("ifdr.dll")]
public static extern uint DrSetOutputData (
   IntPtr      DeviceHandle,
   int         Channel,
   uint        SetNum,
   IntPtr      Data
);
```

●Visual Basic.NET

```
Declare Function DrSetOutputData Lib "ifdr.dll"( _
    ByVal   DeviceHandle        As IntPtr, _
    ByVal   Channel             As Integer, _
    ByVal   SetNum              As Integer, _
    ByRef   Data()              As Integer _
)As Integer
```

```
Declare Function DrSetOutputData Lib "ifdr.dll"( _
    ByVal   DeviceHandle        As IntPtr, _
    ByVal   Channel             As Integer, _
    ByVal   SetNum              As Integer, _
    ByVal   Data                As IntPtr _
)As Integer
```

【パラメータ】

DeviceHandle

DrOpen 関数で取得したデバイスハンドルを指定してください。

Channel

チャンネルを指定します。

SetNum

設定する件数を指定します。

Data

パターン出力データを格納する変数へのポインタを指定します。
パターン出力データは角度で指定します。
設定範囲：0x0000～0xFFFF

※360° ÷ 10000h （＝ 0.0054931640625°）毎に指定可能です。
0000h：0°
0001h：0.0054931640625°
0002h：0.010986328125°
0003h：0.0164794921875°
：
FFFFh：359.9945068359375°

【戻り値】

DrSetOutputData 関数は正常に終了すると IFDR_ERROR_SUCCESS を返します。
それ以外の場合は IFDR_ERROR_SUCCESS 以外の値を返します。
IFDR_ERROR_SUCCESS 以外の値が返された場合については、戻り値一覧を参照してください。

【備考】

・本関数はパターン出力中には実行することはできません。

・書き込むデータフォーマットは下記のようになります。

31 16	15 0
0000000000000000	1 件
0000000000000000	2 件
0000000000000000	:
0000000000000000	N-1 件
0000000000000000	N 件

【使用例】

●C 言語

```
INT Ret;
DWORD Data[360];

for (i = 0; I < 360; i++) {
    Data[i] = (DWORD)(i / 360.0 * 0x10000);
}

Ret = DrSetOutputData(hDeviceHandle, 1, 360, Data);
```

●Visual Basic

```
Dim Ret As Long
Dim DeviceHandle As Long
Dim Data(360) As Long

For i = 0 To 360 - 1
    Data(i) = i / 360# * &H10000
Next

Ret = DrSetOutputData(hDeviceHandle, 1, 360, Data)
```

●Delphi

```
Ret : Integer;
DeviceHandle : THandle;
Data: array[0..359] of Dword;

for i := 0 to 359 do
begin
    Data[i] := Trunc(i / 360 * $10000);
end;

Ret := DrSetOutputData(hDeviceHandle, 1, 360, Data[0]);
```

●Visual C#.NET

```
uint Ret;
IntPtr DeviceHandle;
uint[] Data = new uint[360];

for (int i = 0; i < 360; i++){
    Data[i] = (uint)(i/360.0 * 0x10000);
}

GCHandle gcData = GCHandle.Alloc(Data, GCHandleType.Pinned);
Ret = IFCDR.DrSetOutputData(hDeviceHandle, 1, 360, gcData.AddrOfPinnedObject());
```

●Visual Basic.NET

```
Dim Ret As Integer
Dim DeviceHandle As IntPtr
Dim Data(360) As Integer
Dim i As Integer
Dim gcData As GCHandle

For i = 0 To 360 -  1
   Data(i) = i / 360.0 * &H10000
Next

gcData = GCHandle.Alloc(Data, GCHandleType.Pinned)
Ret = IFCDR.DrSetOutputData(hDeviceHandle, 1, 360, gcData.AddrOfPinnedObject())
```

デバイスハンドル DeviceHandle のパターン出力データ（360 件）を設定します。

10. DrSetFlashRomData

【機能】

パターン出力データを FlashRom に書き込みます。

【書式】

●C 言語

```
INT DrSetFlashRomData (
  HANDLE          DeviceHandle,
  INT             Channel,
  DWORD           SetNum,
  PDWORD          Data,
  HANDLE          EventHandle,
  PDRCALLBACK     CallBackProc,
  PVOID           UserData
);
```

●Visual Basic

```
Declare Function DrSetFlashRomData Lib "ifdr.dll"( _
   ByVal   DeviceHandle        As Long, _
   ByVal   Channel             As Long, _
   ByVal   SetNum              As Long, _
   ByRef   Data                As Long, _
   ByVal   EventHandle         As Long, _
   ByVal   CallBackProc        As Long, _
   ByVal   UserData            As Any _
)As Long
```

●Delphi

```
function DrSetFlashRomData (
   DeviceHandle:           THandle;
   Channel:                Integer;
   SetNum:                 Dword;
   var Data:               Dword;
   EventHandle:            THandle;
   CallBackProc:           FARPROC;
   UserData:               Pointer
): Integer; stdcall; external 'ifdr.dll';
```

●Visual C#.NET

```
[DllImport("ifdr.dll")]
public static extern uint DrSetFlashRomData (
   IntPtr              DeviceHandle,
   uint                Channel,
   uint                SetNum,
   ref uint[]          Data,
   IntPtr              EventHandle,
   LPDRCALLBACK        CallBackProc,
   IntPtr              UserData
);
```

```
[DllImport("ifdr.dll")]
public static extern uint DrSetFlashRomData (
   IntPtr              DeviceHandle,
   uint                Channel,
   uint                SetNum,
   IntPtr              Data,
   IntPtr              EventHandle,
   LPDRCALLBACK        CallBackProc,
   IntPtr              UserData
);
```

```
[DllImport("ifdr.dll")]
public static extern uint DrSetFlashRomData (
   IntPtr              DeviceHandle,
   uint                Channel,
   uint                SetNum,
   ref uint[]          Data,
   IntPtr              EventHandle,
   LPDRCALLBACK        CallBackProc,
   ref uint            UserData
);
```

```
[DllImport("ifdr.dll")]
public static extern uint DrSetFlashRomData (
   IntPtr              DeviceHandle,
   uint                Channel,
   uint                SetNum,
   IntPtr              Data,
   IntPtr              EventHandle,
   LPDRCALLBACK        CallBackProc,
   ref uint            UserData
);
```

●Visual Basic.NET

```
Declare Function DrSetFlashRomData Lib "ifdr.dll"( _
   ByVal   DeviceHandle        As IntPtr, _
   ByVal   Channel             As Integer, _
   ByVal   SetNum              As Integer,_
   ByRef   Data()              As Integer,_
   ByVal   EventHandle         As IntPtr,_
   ByVal   CallBackProc        As LPDRCALLBACK,_
   ByVal   UserData            As IntPtr_
)As Integer
```

```
Declare Function DrSetFlashRomData Lib "ifdr.dll"( _
   ByVal   DeviceHandle        As IntPtr, _
   ByVal   Channel             As Integer, _
   ByVal   SetNum              As Integer,_
   ByVal   Data                As IntPtr,_
   ByVal   EventHandle         As IntPtr,_
   ByVal   CallBackProc        As LPDRCALLBACK,_
   ByVal   UserData            As IntPtr_
)As Integer
```

```
Declare Function DrSetFlashRomData Lib "ifdr.dll"( _
   ByVal   DeviceHandle        As IntPtr, _
   ByVal   Channel             As Integer, _
   ByVal   SetNum              As Integer, _
   ByRef   Data()              As Integer, _
   ByVal   EventHandle         As IntPtr,_
   ByVal   CallBackProc        As LPDRCALLBACK, _
   ByRef   UserData            As Integer _
)As Integer
```

```
Declare Function DrSetFlashRomData Lib "ifdr.dll"( _
   ByVal   DeviceHandle        As IntPtr, _
   ByVal   Channel             As Integer, _
   ByVal   SetNum              As Integer, _
   ByVal   Data                As IntPtr, _
   ByVal   EventHandle         As IntPtr,_
   ByVal   CallBackProc        As LPDRCALLBACK, _
   ByRef   UserData            As Integer _
)As Integer
```

【パラメータ】

DeviceHandle

DrOpen 関数で取得したデバイスハンドルを指定してください。

Channel

チャンネルを指定します。

SetNum

設定する件数を指定します。

Data

パターン出力データを格納する変数へのポインタを指定します。
パターン出力データは角度で指定します。
設定範囲：0x0000～0xFFFF

※360° ÷ 10000h （＝ 0.0054931640625°）毎に指定可能です。
0000h：0°
0001h：0.0054931640625°
0002h：0.010986328125°
0003h：0.0164794921875°
：
FFFFh：359.9945068359375°

EventHandle

割り込み発生時に有効になるイベントハンドルを指定します。

CallBackProc

書き込み完了時に呼ばれるコールバック関数を指定します。

UserData

コールバック関数に渡されるユーザデータを指定します。

【戻り値】

DrSetFlashRomData 関数は正常に終了すると IFDR_ERROR_SUCCESS を返します。
それ以外の場合は IFDR_ERROR_SUCCESS 以外の値を返します。
IFDR_ERROR_SUCCESS 以外の値が返された場合については、戻り値一覧を参照してください。

【備考】

・本関数はパターン出力中には実行することはできません。
・本関数を実行するとすぐに処理が返ります。
FLASH ROM への書き込みが完了を確認するには下記で確認してください。
① イベント処理
② コールバック処理
③ DrGetStatus 関数で Flash ROM 書き込み状態を確認
・設定するデータ件数が多い場合には、書き込みに時間がかかります。
・書き込むデータフォーマットは下記のようになります。

31 ～ 16	15 ～ 0
0000000000000000	1 件
0000000000000000	2 件
0000000000000000	:
0000000000000000	N-1 件
0000000000000000	N 件

【使用例】

●C 言語

```
void CALLBACK EventProc(INT Channel, DWORD Event, PVOID UserData)
{
    :
}

INT Ret;
DWORD Data[360];

for (i = 0; I < 360; i++) {
    Data[i] = (DWORD)(i / 360.0 * 0x10000);
}

Ret = DrSetFlashRomData (hDeviceHandle, 1, 360, Data,
                         NULL, (LPDRCALLBACK)EventProc, (PVOID)UserData);
```

●Visual Basic
コールバックルーチンはDrSetEvent関数の呼び出しを行うプロジェクト内の
標準モジュールの中に記述しなければなりません。
DrSetFlashRomData 関数の引数パラメータでプロシージャのアドレスを渡す為に AddressOf
演算子を使用します。

```
Sub EventProc(Channel As Long, Event As Long, UserData As Any)
   :
End Sub

Dim Ret As Long
Dim DeviceHandle As Long
Dim Data(360) As Long

For i = 0 To 360 - 1
    Data(i) = i / 360# * &H10000
Next

Ret = DrSetFlashRomData(hDeviceHandle, 1, 360, Data, 0, AddressOf EventProc, 0)
```

●Delphi

```
procedure EventProc(Channel : Integer; Event : Dword; UserData : Pointer); stdcall;
begin
   :
end;
Ret : Integer;
DeviceHandle : THandle;
Data: array[0..359] of Dword;

for i := 0 to 359 do
begin
    Data[i] := Trunc(i / 360 * $10000);
end;

Ret := DrSetFlashRomData(hDeviceHandle, 1, 360, Data[0] , Null, EventProc, 0);
```

●Visual C#.NET

```
void EventProc(int Channel, uint Event, IntPtr UserData)
{
    :
}

uint Ret;
IntPtr DeviceHandle;
uint[] Data = new uint[360];

for (int i = 0; i < 360; i++){
    Data[i] = (uint)(i/360.0 * 0x10000);
}

GCHandle gcData = GCHandle.Alloc(Data, GCHandleType.Pinned);
Ret = IFCDR.DrSetFlashRomData (hDeviceHandle, 1, 360, gcData.AddrOfPinnedObject()
             IntPtr.Zero, new IFCDR.DRCALLBACK(EventProc), IntPtr.Zero);
```

●Visual Basic.NET

```
Private Sub EventProc(Channel As Integer, Event As Integer, UserData As IntPtr)
   :
End Sub

Dim Ret As Integer
Dim DeviceHandle As IntPtr
Dim Data(360) As Integer
Dim i As Integer
Dim gcData As GCHandle

For i = 0 To 360 -  1
   Data(i) = i / 360.0 * &H10000
Next

gcData = GCHandle.Alloc(Data, GCHandleType.Pinned)
Ret = IFCDR.DrSetFlashRomData(hDeviceHandle, 1, 360, gcData.AddrOfPinnedObject(), _
              IntPtr.Zero, New IFCDR.IFDRCALLBACK(EventProc), IntPtr.Zero)
```

デバイスハンドル DeviceHandle のパターン出力データ（360 件）を設定します。

11. DrSetOutputRate

【機能】

パターン出力の更新レートを変更します。

【書式】

●C 言語

```
INT DrSetOutputRate (
  HANDLE       DeviceHandle,
  INT          Channel,
  DWORD        OutputRate
);
```

●Visual Basic

```
Declare Function DrSetOutputRate Lib "ifdr.dll"( _
   ByVal   DeviceHandle          As Long, _
   ByVal   Channel               As Long, _
   ByVal   OutputRate            As Long _
)As Long
```

●Delphi

```
function DrSetOutputRate (
   DeviceHandle:             THandle;
   Channel:                  Integer;
   OutputRate                Dword
): Integer; stdcall; external 'ifdr.dll';
```

●Visual C#.NET

```
[DllImport("ifdr.dll")]
public static extern uint DrSetOutputRate (
   IntPtr    DeviceHandle,
   uint      Channel,
   uint      OutputRate
);
```

●Visual Basic.NET

```
Declare Function DrSetOutputRate Lib "ifdr.dll"( _
   ByVal   DeviceHandle          As IntPtr, _
   ByVal   Channel               As Integer, _
   ByVal   OutputRate            As Integer _
)As Integer
```

【パラメータ】

DeviceHandle

DrOpen 関数で取得したデバイスハンドルを指定してください。

Channel

チャンネルを指定します。

OutputRate

更新レート設定を指定します。
1μs の分解能で指定できます。

値	タイマ設定値
0x00000000	設定禁止
0x00000001	1μs
0x00000002	2μs
:	:
0x00FFFFFF	16.777215s

【戻り値】

DrSetOutputRate 関数は正常に終了すると IFDR_ERROR_SUCCESS を返します。
それ以外の場合は IFDR_ERROR_SUCCESS 以外の値を返します。
IFDR_ERROR_SUCCESS 以外の値が返された場合については、戻り値一覧を参照してください。

【備考】

・本関数はパターン出力中に実行するとパターン出力の更新レートが関数実行時に変更されます。

【使用例】

●C 言語

```
INT Ret;

Ret = DrSetOutputRate (DeviceHandle, 1, 10);
```

●Visual Basic

```
Dim Ret As Long
Dim DeviceHandle As Long

Ret = DrSetOutputRate (DeviceHandle, 1, 10)
```

●Delphi

```
Ret : Integer;
DeviceHandle : THandle;

Ret := DrSetOutputRate (DeviceHandle, 1, 10);
```

●Visual C#.NET

```
uint Ret;
IntPtr DeviceHandle;

Ret = IFCDR. DrSetOutputRate (DeviceHandle, 1, 10);
```

●Visual Basic.NET

```
Dim Ret As Integer
Dim DeviceHandle As IntPtr

Ret = IFCDR. DrSetOutputRate (DeviceHandle, 1, 10)
```

デバイスハンドル DeviceHandle のパターン出力の更新レートを変更します。
チャンネル：1 チャンネル
更新レート：10μs

12. DrStartOutput

【機能】

パターン出力の開始を発行します。

【書式】

●C 言語

```
INT DrStartOutput (
  HANDLE      DeviceHandle,
  INT         Channel,
  DWORD       OutputRate,
  DWORD       StartPoint,
  DWORD       OutputNum,
  DWORD       OutputRepeat
);
```

●Visual Basic

```
Declare Function DrStartOutput Lib "ifdr.dll"( _
   ByVal   DeviceHandle        As Long, _
   ByVal   Channel             As Long, _
   ByVal   OutputRate          As Long, _
   ByVal   StartPoint          As Long, _
   ByVal   OutputNum           As Long, _
   ByVal   OutputRepeat        As Long _
)As Long
```

●Delphi

```
function DrStartOutput (
   DeviceHandle:           THandle;
   Channel:                Integer;
   OutputRate              Dword;
   StartPoint:             Dword;
   OutputNum:              Dword;
   OutputRepeat:           Dword
): Integer; stdcall; external 'ifdr.dll';
```

●Visual C#.NET

```
[DllImport("ifdr.dll")]
public static extern uint DrStartOutput (
   IntPtr    DeviceHandle,
   uint      Channel,
   uint      OutputRate,
   uint      StartPoint,
   uint      OutputNum,
   uint      OutputRepeat
);
```

●Visual Basic.NET

```
Declare Function DrStartOutput Lib "ifdr.dll"( _
    ByVal   DeviceHandle        As IntPtr, _
    ByVal   Channel             As Integer, _
    ByVal   OutputRate          As Integer, _
    ByVal   StartPoint          As Integer, _
    ByVal   OutputNum           As Integer, _
    ByVal   OutputRepeat        As Integer _
)As Integer
```

【パラメータ】

DeviceHandle

DrOpen 関数で取得したデバイスハンドルを指定してください。

Channel

チャンネルを指定します。

OutputRate

更新レート設定を指定します。
1μs の分解能で指定できます。

値	タイマ設定値
0x00000000	設定禁止
0x00000001	1μs
0x00000002	2μs
:	:
0x00FFFFFF	16.777215s

StartPoint

パターンデータの開始位置を指定します。
0 を指定すると DrSetOutputData 関数で指定されたデータが出力されます。
0 以外が設定された場合は、DrSetFlashRomData 関数で指定されたデータが出力されます。

SmplNum

出力件数を指定します。

OutputRepeat

繰り返し回数を指定します。
0 を指定すると DrStopOutput 関数が実行されるまで繰り返します。

【戻り値】

DrStartOutput 関数は正常に終了すると IFDR_ERROR_SUCCESS を返します。
それ以外の場合は IFDR_ERROR_SUCCESS 以外の値を返します。
IFDR_ERROR_SUCCESS 以外の値が返された場合については、戻り値一覧を参照してください。

【備考】
・本関数はパターン出力中には実行することはできません。
・本関数はデータ設定中には実行することはできません。
・StartPoint に 0 を指定した時、繰り返し 0 を使用する場合件数は 2048 以上の値を設定する必要があります。

【使用例】
●C 言語

```
INT Ret;

Ret = DrStartOutput(DeviceHandle, 1, 10, 1, 1024, 10);
```

●Visual Basic

```
Dim Ret As Long
Dim DeviceHandle As Long

Ret = DrStartOutput (DeviceHandle, 1, 10, 1, 1024, 10)
```

●Delphi

```
Ret : Integer;
DeviceHandle : THandle;

Ret := DrStartOutput (DeviceHandle, 1, 10, 1, 1024, 10);
```

●Visual C#.NET

```
uint Ret;
IntPtr DeviceHandle;

Ret = IFCDR. DrStartOutput (DeviceHandle, 1, 10, 1, 1024, 10);
```

●Visual Basic.NET

```
Dim Ret As Integer
Dim DeviceHandle As IntPtr

Ret = IFCDR. DrStartOutput (DeviceHandle, 1, 10, 1, 1024, 10)
```

デバイスハンドル DeviceHandle のパターン出力を開始します。
チャンネル：1 チャンネル
更新レート：10μs
開始位置：1 件
件数：1024 件
繰り返し回数：10 回

13. DrStopOutput

【機能】

パターン出力の停止を発行します。

【書式】

●C 言語

```
INT DrStopOutput(
  HANDLE       DeviceHandle,
  INT          Channel
);
```

●Visual Basic

```
Declare Function DrStopOutput Lib "ifdr.dll"( _
   ByVal   DeviceHandle         As Long, _
   ByVal   Channel              As Long _
)As Long
```

●Delphi

```
function DrStopOutput (
   DeviceHandle:              THandle;
   Channel:                   Integer
): Integer; stdcall; external 'ifdr.dll';
```

●Visual C#.NET

```
[DllImport("ifdr.dll")]
public static extern uint DrStopOutput (
   IntPtr    DeviceHandle,
   uint      Channel
);
```

●Visual Basic.NET

```
Declare Function DrStopOutput Lib "ifdr.dll"( _
   ByVal   DeviceHandle         As IntPtr, _
   ByVal   Channel              As Integer
)As Integer
```

【パラメータ】

DeviceHandle

DrOpen 関数で取得したデバイスハンドルを指定してください。

Channel

チャンネルを指定します。

【戻り値】

DrStopOutput 関数は正常に終了すると IFDR_ERROR_SUCCESS を返します。
それ以外の場合は IFDR_ERROR_SUCCESS 以外の値を返します。
IFDR_ERROR_SUCCESS 以外の値が返された場合については、戻り値一覧を参照してください。

【使用例】

●C 言語

```
INT Ret;

Ret = DrStopOutput DeviceHandle, 1);
```

●Visual Basic

```
Dim Ret As Long
Dim DeviceHandle As Long

Ret = DrStopOutput(DeviceHandle, 1)
```

●Delphi

```
Ret : Integer;
DeviceHandle : THandle;

Ret := DrStopOutput(DeviceHandle, 1);
```

●Visual C#.NET

```
uint Ret;
IntPtr DeviceHandle;

Ret = IFCDR.DrStopOutput (DeviceHandle, 1);
```

●Visual Basic.NET

```
Dim Ret As Integer
Dim DeviceHandle As IntPtr

Ret = IFCDR.DrStopOutput (DeviceHandle, 1)
```

デバイスハンドル DeviceHandle のパターン出力を強制停止させます。

14. DrSetNoiseData

【機能】

異常信号出力のデータの設定を行います。

【書式】

●C 言語

```
INT DrSetNoiseData(
  HANDLE      DeviceHandle,
  INT         Channel,
  DWORD       SetNum,
  PDWORD      Data
);
```

●Visual Basic

```
Declare Function DrSetNoiseData Lib "ifdr.dll"( _
   ByVal   DeviceHandle        As Long, _
   ByVal   Channel             As Long, _
   ByVal   SetNum              As Long, _
   ByRef   Data                As Long _
)As Long
```

●Delphi

```
function DrSetNoiseData (
   DeviceHandle:           THandle;
   Channel:                Integer;
   SetNum:                 Dword;
   :var Data:              Dword
): Integer; stdcall; external 'ifdr.dll';
```

●Visual C#.NET

```
[DllImport("ifdr.dll")]
public static extern uint DrSetNoiseData (
   IntPtr      DeviceHandle,
   uint        Channel,
   uint        SetNum,
   ref uint[]  Data
);
```

```
[DllImport("ifdr.dll")]
public static extern uint DrSetNoiseData (
   IntPtr      DeviceHandle,
   uint        Channel,
   uint        SetNum,
   IntPtr      Data
);
```

●Visual Basic.NET

```
Declare Function DrSetNoiseData Lib "ifdr.dll"( _
    ByVal   DeviceHandle        As IntPtr, _
    ByVal   Channel             As Integer, _
    ByVal   SetNum              As Integer, _
    ByRef   Data()              As Integer _
)As Integer
```

```
Declare Function DrSetNoiseData Lib "ifdr.dll"( _
    ByVal   DeviceHandle        As IntPtr, _
    ByVal   Channel             As Integer, _
    ByVal   SetNum              As Integer, _
    ByVal   Data                As IntPtr _
)As Integer
```

【パラメータ】

DeviceHandle

DrOpen 関数で取得したデバイスハンドルを指定してください。

Channel

チャンネルを指定します。

SetNum

設定する件数を指定します。
1～256 で指定できます。

Data

ノイズデータを格納する変数へのポインタを指定します。
設定範囲：0x0000～0xFFFF
単位は LSB で指定し最上位ビットが符号になります。
値は 2 の補数で設定する必要があります。
1LSB あたり　20/65536=0.305mV

【戻り値】

DrSetNoiseData 関数は正常に終了すると IFDR_ERROR_SUCCESS を返します。
それ以外の場合は IFDR_ERROR_SUCCESS 以外の値を返します。
IFDR_ERROR_SUCCESS 以外の値が返された場合については、戻り値一覧を参照してください。

【備考】

・書き込むデータフォーマットは下記のようになります。

31 ～ 16	15 ～ 0
0000000000000000	1 件
0000000000000000	2 件
0000000000000000	:
0000000000000000	N-1 件
0000000000000000	N 件

【使用例】

●C 言語

```
INT Ret;
DWORD Data[256];

for (i = 0; I < 256; i++) {
    Data[i] = 0x1000+i;
}

Ret = DrSetNoiseData(hDeviceHandle, 1, 256, Data);
```

●Visual Basic

```
Dim Ret As Long
Dim DeviceHandle As Long
Dim Data(256) As Long

For i = 0 To 256 - 1
    Data(i) = i + &H10000
Next

Ret = DrSetNoiseData(hDeviceHandle, 1, 256, Data)
```

●Delphi

```
Ret : Integer;
DeviceHandle : THandle;
Data: array[0..255] of Dword;

for i := 0 to 255 do
begin
    Data[i] := Trunc(i + $1000);
end;

Ret := DrSetNoiseData(hDeviceHandle, 1, 256, Data[0]);
```

●Visual C#.NET

```
uint Ret;
IntPtr DeviceHandle;
uint[] Data = new uint[256];

for (int i = 0; i < 256; i++){
    Data[i] = (uint)(I + 0x1000);
}

GCHandle gcData = GCHandle.Alloc(Data, GCHandleType.Pinned);
Ret = IFCDR.DrSetNoiseData(hDeviceHandle, 1, 256, gcData.AddrOfPinnedObject());
```

●Visual Basic.NET

```
Dim Ret As Integer
Dim DeviceHandle As IntPtr
Dim Data(256) As Integer
Dim i As Integer
Dim gcData As GCHandle

For i = 0 To 256 -  1
   Data(i) = i + &H1000
Next

gcData = GCHandle.Alloc(Data, GCHandleType.Pinned)
Ret = IFCDR.DrSetNoiseData(hDeviceHandle, 1, 256, gcData.AddrOfPinnedObject())
```

デバイスハンドル DeviceHandle の異常信号出力データ（256 件）を設定します。

15. DrStartNoise

【機能】

異常信号出力の開始を発行します。

【書式】

●C 言語

```
INT DrStartNoise (
  HANDLE        DeviceHandle,
  INT           Channel,
  DWORD         OutputRate,
  DWORD         OutputNum,
  DWORD         OutputRepeat
);
```

●Visual Basic

```
Declare Function DrStartNoise Lib "ifdr.dll"( _
   ByVal   DeviceHandle         As Long, _
   ByVal   Channel              As Long, _
   ByVal   OutputRate           As Long, _
   ByVal   OutputNum            As Long, _
   ByVal   OutputRepeat         As Long _
)As Long
```

●Delphi

```
function DrStartNoise (
   DeviceHandle:            THandle;
   Channel:                 Integer;
   OutputRate:              Dword;
   OutputNum:               Dword;
   OutputRepeat:            Dword
): Integer; stdcall; external 'ifdr.dll';
```

●Visual C#.NET

```
[DllImport("ifdr.dll")]
public static extern uint DrStartNoise (
   IntPtr    DeviceHandle,
   uint      Channel,
   uint      OutputRate,
   uint      OutputNum,
   uint      OutputRepeat
);
```

●Visual Basic.NET

```
Declare Function DrStartNoise Lib "ifdr.dll"( _
    ByVal    DeviceHandle        As IntPtr, _
    ByVal    Channel             As Integer, _
    ByVal    OutputRate          As Integer, _
    ByVal    OutputNum           As Integer, _
    ByVal    OutputRepeat        As Integer _
)As Integer
```

【パラメータ】

DeviceHandle

DrOpen 関数で取得したデバイスハンドルを指定してください。

Channel

チャンネルを指定します。

OutpuRate

更新レート設定を指定します。
1μs の分解能で指定できます。

値	タイマ設定値
0x00000000	設定禁止
0x00000001	1μs
0x00000002	2μs
:	:
0x000000FF	255μs

OutputNum

出力件数を指定します。
1～256 で指定できます。

OutputRepeat

繰り返しの有効/無効を指定します。

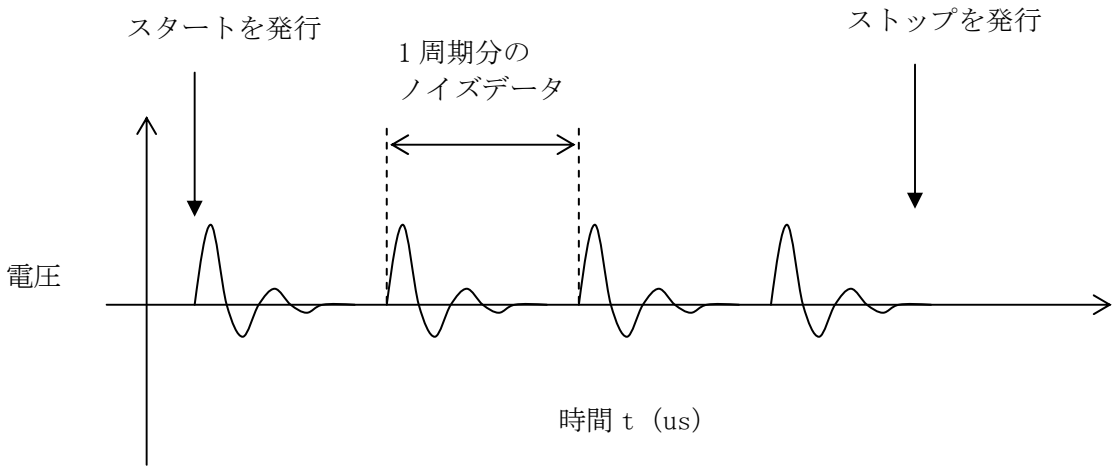
識別子	値	内容
IFDR_DISABLE	0x00	繰り返し無効
IFDR_ENABLE	0x01	繰り返し有効

【戻り値】

DrStartNoise 関数は正常に終了すると IFDR_ERROR_SUCCESS を返します。
それ以外の場合は IFDR_ERROR_SUCCESS 以外の値を返します。
IFDR_ERROR_SUCCESS 以外の値が返された場合については、戻り値一覧を参照してください。

【備考】

・異常信号出力は下記のように動作します。



【使用例】

●C 言語

```
INT Ret;

Ret = DrStartNoise(DeviceHandle, 1, 10, 10, IFDR_ENABLE);
```

●Visual Basic

```
Dim Ret As Long
Dim DeviceHandle As Long

Ret = DrStartNoise (DeviceHandle, 1, 10, 10, IFDR_ENABLE)
```

●Delphi

```
Ret : Integer;
DeviceHandle : THandle;

Ret := DrStartNoise(DeviceHandle, 1, 10, 10, IFDR_ENABLE);
```

●Visual C#.NET

```
uint Ret;
IntPtr DeviceHandle;

Ret = IFCDR.DrStartNoise(DeviceHandle, 1, 10, 10, IFCDR.IFDR_ENABLE);
```

●Visual Basic.NET

```
Dim Ret As Integer
Dim DeviceHandle As IntPtr

Ret = IFCDR.DrStartNoise(DeviceHandle, 1, 10, 10, IFCDR.IFDR_ENABLE)
```

デバイスハンドル DeviceHandle の異常信号出力を開始させます。

16. DrStopNoise

【機能】

異常信号出力の停止を発行します。

【書式】

●C 言語

```
INT DrStopNoise (
  HANDLE        DeviceHandle,
  INT           Channel
);
```

●Visual Basic

```
Declare Function DrStopNoise Lib "ifdr.dll"( _
   ByVal   DeviceHandle         As Long, _
   ByVal   Channel              As Long _
)As Long
```

●Delphi

```
function DrStopNoise (
   DeviceHandle:              THandle;
   Channel:                   Integer
): Integer; stdcall; external 'ifdr.dll';
```

●Visual C#.NET

```
[DllImport("ifdr.dll")]
public static extern uint DrStopNoise (
   IntPtr    DeviceHandle,
   uint      Channel
);
```

●Visual Basic.NET

```
Declare Function DrStopNoise Lib "ifdr.dll"( _
   ByVal   DeviceHandle         As IntPtr, _
   ByVal   Channel              As Integer _
)As Integer
```

【パラメータ】

DeviceHandle

DrOpen 関数で取得したデバイスハンドルを指定してください。

Channel

チャンネルを指定します。

【戻り値】

DrStopNoise 関数は正常に終了すると IFDR_ERROR_SUCCESS を返します。
それ以外の場合は IFDR_ERROR_SUCCESS 以外の値を返します。
IFDR_ERROR_SUCCESS 以外の値が返された場合については、戻り値一覧を参照してください。

【使用例】

●C 言語

```
INT Ret;

Ret = DrStopNoise(DeviceHandle, 1);
```

●Visual Basic

```
Dim Ret As Long
Dim DeviceHandle As Long

Ret = DrStopNoise (DeviceHandle, 1)
```

●Delphi

```
Ret : Integer;
DeviceHandle : THandle;

Ret := DrStopNoise (DeviceHandle, 1);
```

●Visual C#.NET

```
uint Ret;
IntPtr DeviceHandle;

Ret = IFCDR. DrStopNoise (DeviceHandle, 1);
```

●Visual Basic.NET

```
Dim Ret As Integer
Dim DeviceHandle As IntPtr

Ret = IFCDR. DrStopNoise (DeviceHandle, 1)
```

デバイスハンドル DeviceHandle の異常信号出力を強制停止させます。

17. DrInputDI

【機能】

汎用入力端子の状態を取得します。

【書式】

●C 言語

```
INT DrInputDI (
   HANDLE          DeviceHandle,
   PDWORD          Data
);
```

●Visual Basic

```
Declare Function DrInputDI Lib "ifdr.dll"( _
   ByVal   DeviceHandle    As Long, _
   ByRef   Data            As Long _
)As Long
```

●Delphi

```
function DrInputDI (
   DeviceHandle:         Thandle;
   var Data:             Dword
): Integer; stdcall; external 'ifdr.dll';
```

●Visual C#.NET

```
[DllImport("ifdr.dll")]
public static extern uint DrInputDI (
   IntPtr       DeviceHandle,
   out uint     Data
);
```

●Visual Basic.NET

```
Declare Function DrInputDI Lib "ifdr.dll"( _
   ByVal   DeviceHandle    As IntPtr, _
   ByRef   Data            As Integer _
)As Integer
```

【パラメータ】

DeviceHandle

DrOpen 関数で取得したデバイスハンドルを指定してください。

Data

汎用入力の端子状態を取得する変数への参照を指定します。
(0：High、1：Low)

【戻り値】

DrInputDI 関数は正常に終了すると IFDR_ERROR_SUCCESS を返します。
それ以外の場合は IFDR_ERROR_SUCCESS 以外の値を返します。
IFDR_ERROR_SUCCESS 以外の値が返された場合については、戻り値一覧を参照してください。

【使用例】

●C 言語

```
INT Ret;
HANDLE DeviceHandle;
DWORD Data;

Ret = DrInputDI(DeviceHandle, &Data);
```

●Visual Basic

```
Dim Ret As Long
Dim DeviceHandle As Long
Dim Data As Long

Ret = DrInputDI(DeviceHandle, Data)
```

●Delphi

```
Ret : Integer;
DeviceHandle : THandle;
Data : Dword;

Ret := DrInputDI(DeviceHandle, Data);
```

●Visual C#.NET

```
uint Ret;
IntPtr DeviceHandle;
uint Data;

Ret = IFCDR.DrInputDI(DeviceHandle, out Data);
```

●Visual Basic.NET

```
Dim Ret As Integer
Dim DeviceHandle As IntPtr
Dim Data As Integer

Ret = IFCDR.DrInputDI(DeviceHandle, Data)
```

デバイスハンドル DeviceHandle の汎用入力の端子状態を取得します。

18. DrOutputDO

【機能】

汎用出力端子の制御を行います。

【書式】

●C 言語

```
INT DrOutputDO (
   HANDLE          DeviceHandle,
   DWORD           Data
);
```

●Visual Basic

```
Declare Function DrOutputDO Lib "ifdr.dll"( _
   ByVal   DeviceHandle     As Long, _
   ByVal   Data             As Long _
)As Long
```

●Delphi

```
function DrOutputDO (
   DeviceHandle:          Thandle;
   Data:                  Dword
): Integer; stdcall; external 'ifdr.dll';
```

●Visual C#.NET

```
[DllImport("ifdr.dll")]
public static extern uint DrOutputDO (
   IntPtr    DeviceHandle,
   uint      Data
);
```

●Visual Basic.NET

```
Declare Function DrOutputDO Lib "ifdr.dll"( _
   ByVal   DeviceHandle     As IntPtr, _
   ByVal   Data             As Integer _
)As Integer
```

【パラメータ】

DeviceHandle

DrOpen 関数で取得したデバイスハンドルを指定してください。

Data

汎用出力端子の出力状態を指定します。
(0：High、1：Low)

【戻り値】

DrOutputDO 関数は正常に終了すると IFDR_ERROR_SUCCESS を返します。
それ以外の場合は IFDR_ERROR_SUCCESS 以外の値を返します。
IFDR_ERROR_SUCCESS 以外の値が返された場合については、戻り値一覧を参照してください。

【使用例】

●C 言語

```
INT Ret;
HANDLE DeviceHandle;

Ret = DrOutputDO(DeviceHandle, 0xAA);
```

●Visual Basic

```
Dim Ret As Long
Dim DeviceHandle As Long

Ret = DrOutputDO(DeviceHandle, &HAA)
```

●Delphi

```
Ret : Integer;
DeviceHandle : THandle;

Ret := DrOutputDO(DeviceHandle, $AA);
```

●Visual C#.NET

```
uint Ret;
IntPtr DeviceHandle;

Ret = IFCDR.DrOutputDO(DeviceHandle, 0xAA);
```

●Visual Basic.NET

```
Dim Ret As Integer
Dim DeviceHandle As IntPtr

Ret = IFCDR.DrOutputDO(DeviceHandle, &HAA)
```

デバイスハンドル DeviceHandle の汎用出力端子の出力状態を変更します。

19. DrSetEventMask

【機能】

割り込みイベントマスクの設定を行います。

【書式】

●C 言語

```
INT DrSetEventMask(
  HANDLE          DeviceHandle,
  INT             Channel,
  DWORD           EventMask
);
```

●Visual Basic

```
Declare Function DrSetEventMask Lib "ifdr.dll"( _
   ByVal   DeviceHandle        As Long, _
   ByVal   Channel             As Long, _
   ByVal   EventMask           As Long _
)As Long
```

●Delphi

```
function DrSetEventMask (
   DeviceHandle:           THandle;
   Channel:                Integer;
   EventMask:              Dword
): Integer; stdcall; external 'ifdr.dll';
```

●Visual C#.NET

```
[DllImport("ifdr.dll")]
public static extern uint DrSetEventMask (
   IntPtr    DeviceHandle,
   uint      Channel,
   uint      EventMask
);
```

●Visual Basic.NET

```
Declare Function DrSetEventMask Lib "ifdr.dll"( _
   ByVal   DeviceHandle        As IntPtr, _
   ByVal   Channel             As Integer, _
   ByVal   EventMask           As Integer _
)As Integer
```

【パラメータ】

DeviceHandle

DrOpen 関数で取得したデバイスハンドルを指定してください。

Channel

チャンネルを指定します。

EventMask

割り込みマスクを指定します。

ビット	31～16
	予約

ビット	15～13	12	11～9	8
	予約	SPE	予約	UN RUN

ビット	7,6	5	4	3	2	1	0
	予約	AGL3 DOWN	AGL3 UP	AGL2 DOWN	AGL2 UP	AGL1 DOWN	AGL1 UP

	内容
AGL1 UP	順方向角度割り込み 1 発生
AGL1 DOWN	逆方向角度割り込み 1 発生
AGL2 UP	順方向角度割り込み 2 発生
AGL2 DOWN	逆方向角度割り込み 2 発生
AGL3 UP	順方向角度割り込み 3 発生
AGL3 DOWN	逆方向角度割り込み 3 発生
UNRUN	アンダーラン時に割り込み発生
SPE	パターン出力停止検出

【戻り値】

DrSetEventMask 関数は正常に終了すると IFDR_ERROR_SUCCESS を返します。
それ以外の場合は IFDR_ERROR_SUCCESS 以外の値を返します。
IFDR_ERROR_SUCCESS 以外の値が返された場合については、戻り値一覧を参照してください。

【備考】

・本関数はパターン出力中には実行することはできません。

【使用例】

●C 言語

```
INT Ret;

Ret = DrSetEventMask(DeviceHandle, 1, 0x1000);
```

●Visual Basic

```
Dim Ret As Long
Dim DeviceHandle As Long

Ret = DrSetEventMask (DeviceHandle, 1, &H1000)
```

●Delphi

```
Ret : Integer;
DeviceHandle : THandle;

Ret := DrSetEventMask (DeviceHandle, 1, $1000);
```

●Visual C#.NET

```
uint Ret;
IntPtr DeviceHandle;

Ret = IFCDR.DrSetEventMask (DeviceHandle, 1, 0x1000);
```

●Visual Basic.NET

```
Dim Ret As Integer
Dim DeviceHandle As IntPtr

Ret = IFCDR.DrSetEventMask (DeviceHandle, 1, &H100)
```

デバイスハンドル DeviceHandle の割り込みマスク（パターン出力停止検出）を設定します。

20. DrGetEventMask

【機能】

割り込みイベントマスクの取得を行います。

【書式】

●C 言語

```
INT DrGetEventMask(
  HANDLE          DeviceHandle,
  INT             Channel,
  PDWORD          EventMask
);
```

●Visual Basic

```
Declare Function DrGetEventMask Lib "ifdr.dll"( _
   ByVal   DeviceHandle    As Long, _
   ByVal   Channel         As Long, _
   ByRef   EventMask       As Long _
)As Long
```

●Delphi

```
function DrGetEventMask (
   DeviceHandle:        Thandle;
   Channel:             Integer;
   var EventMask:       Dword
): Integer; stdcall; external 'ifdr.dll';
```

●Visual C#.NET

```
[DllImport("ifdr.dll")]
public static extern uint DrGetEventMask (
   IntPtr      DeviceHandle,
   uint        Channel,
   out uint    EventMask
);
```

●Visual Basic.NET

```
Declare Function DrGetEventMask Lib "ifdr.dll"( _
   ByVal   DeviceHandle    As IntPtr, _
   ByVal   Channel         As Integer, _
   ByRef   EventMask       As Integer _
)As Integer
```

【パラメータ】

DeviceHandle

DrOpen 関数で取得したデバイスハンドルを指定してください。

Channel

チャンネルを指定します。

EventMask

割り込みマスクを取得する変数へのポインタを指定します。

ビット	31～16
	予約

ビット	15～13	12	11～9	8
	予約	SPE	予約	UN RUN

ビット	7,6	5	4	3	2	1	0
	予約	AGL3 DOWN	AGL3 UP	AGL2 DOWN	AGL2 UP	AGL1 DOWN	AGL1 UP

	内容
AGL1 UP	順方向角度割り込み 1 発生
AGL1 DOWN	逆方向角度割り込み 1 発生
AGL2 UP	順方向角度割り込み 2 発生
AGL2 DOWN	逆方向角度割り込み 2 発生
AGL3 UP	順方向角度割り込み 3 発生
AGL3 DOWN	逆方向角度割り込み 3 発生
UNRUN	アンダーラン時に割り込み発生
SPE	パターン出力停止検出

【戻り値】

DrGetEventMask 関数は正常に終了すると IFDR_ERROR_SUCCESS を返します。
それ以外の場合は IFDR_ERROR_SUCCESS 以外の値を返します。
IFDR_ERROR_SUCCESS 以外の値が返された場合については、戻り値一覧を参照してください。

【使用例】

●C 言語

```
INT Ret;

Ret = DrGetEventMask (DeviceHandle, 1, &Mask);
```

●Visual Basic

```
Dim Ret As Long
Dim DeviceHandle As Long
Dim Mask As Long

Ret = DrGetEventMask(DeviceHandle, 1, Mask)
```

●Delphi

```
Ret : Integer;
DeviceHandle : THandle;
Mask: Dword;

Ret := DrGetEventMask(DeviceHandle, 1, Mask);
```

●Visual C#.NET

```
uint Ret;
IntPtr DeviceHandle;
uint Mask;

Ret = IFCDR.DrGetEventMask(DeviceHandle, 1, out Mask);
```

●Visual Basic.NET

```
Dim Ret As Integer
Dim DeviceHandle As IntPtr
Dim Mask As Integer

Ret = IFCDR. DrGetEventMask(DeviceHandle, Mask)
```

デバイスハンドル DeviceHandle の設定されている割り込みマスクを取得します。

21. DrSetEvent

【機能】

割り込み発生時に呼ばれる、コールバック関数・イベントハンドルを登録します。

【書式】

●C 言語

```
INT DrSetEvent (
    HANDLE              DeviceHandle,
    HANDLE              EventHandle,
    LPDRCALLBACK        EventProc,
    PVOID               UserData
);
```

●Visual Basic

```
Declare Function DrSetEvent Lib "ifdr.dll"( _
    ByVal   DeviceHandle    As Long, _
    ByVal   EventHandle     As Long, _
    ByVal   EventProc       As Long, _
    ByVal   UserData        As Any _
)As Long
```

●Delphi

```
function DrSetEvent (
    DeviceHandle:       THandle;
    EventHandle:        THandle;
    EventProc:          FARPROC;
    UserData:           Pointer
): Integer; stdcall; external 'ifdr.dll';
```

●Visual C#.NET

```
[DllImport("ifdr.dll")]
public static extern uint DrSetEvent (
    IntPtr              DeviceHandle,
    IntPtr              EventHandle,
    LPDRCALLBACK        EventProc,
    IntPtr              UserData
);
```

●Visual Basic.NET

```
Declare Function DrSetEvent Lib "ifdr.dll"( _
    ByVal   DeviceHandle    As IntPtr, _
    ByVal   EventHandle     As IntPtr, _
    ByVal   EventProc       As LPDRCALLBACK, _
    ByVal   UserData        As IntPtr _
)As Integer
```

【パラメータ】

DeviceHandle

DrOpen 関数で取得したデバイスハンドルを指定してください。

EventHandle

割り込み発生時に有効となるイベントハンドルを指定します。

EventProc

割り込み発生時に呼ばれるコールバック関数の参照渡しを指定します。

UserData

コールバック関数に渡す任意のデータを指定します。

【戻り値】

DrSetEvent 関数は正常に終了すると IFDR_ERROR_SUCCESS を返します。
それ以外の場合は IFDR_ERROR_SUCCESS 以外の値を返します。
IFDR_ERROR_SUCCESS 以外の値が返された場合については、戻り値一覧を参照してください。

【備考】

・本関数はパターン出力中には実行することはできません。
・登録したコールバック関数は、DrKillEvent 関数にて、削除できます。

【使用例】

●C 言語

```
void CALLBACK EventProc(INT Channel, DWORD Event, PVOID UserData)
{
    :
}

INT Ret;
HANDLE DeviceHandle;

Ret = DrSetEvent(DeviceHandle, NULL, (LPDRCALLBACK)EventProc, 0);
```

●Visual Basic
コールバックルーチンはDrSetEvent関数の呼び出しを行うプロジェクト内の
標準モジュールの中に記述しなければなりません。
DrSetEvent関数の引数パラメータでプロシージャのアドレスを渡す為にAddressOf演算子を使用します。

```
Sub EventProc(Channel As Long, Event As Long, UserData As Any)
   :
End Sub

Dim Ret As Long
Dim DeviceHandle As Long

Ret = DrSetEvent(DeviceHandle, 0, AddressOf EventProc, 0)
```

●Delphi

```
procedure EventProc(Channel : Integer; Event : Dword; UserData : Pointer); stdcall;
begin
   :
end;

Ret : Integer;
DeviceHandle : THandle;

Ret := DrSetEvent(DeviceHandle, Null, EventProc, 0);
```

●Visual C#.NET

```
void EventProc(int Channel, uint Event, IntPtr UserData)
{
    :
}

uint Ret;
IntPtr DeviceHandle;

Ret = IFCDR.DrSetEvent(DeviceHandle, IntPtr.Zero,
                       new IFCDR.DRCALLBACK(EventProc), IntPtr.Zero);
```

●Visual Basic.NET

```
Private Sub EventProc(Channel As Integer, Event As Integer, UserData As IntPtr)
   :
End Sub

Dim Ret As Intger
Dim DeviceHandle As IntPtr

Ret = IFCDR.DrSetEvent(DeviceHandle, IntPtr.Zero,_
```

```
New IFCDR.IFDRCALLBACK(EventProc), IntPtr.Zero)
```

デバイスハンドル DeviceHandle のデバイスにコールバック関数を登録します。

22. DrKillEvent

【機能】

コールバック関数・イベントハンドルの登録を解除します。

【書式】

●C 言語

```
INT DrKillEvent (
   HANDLE       DeviceHandle
);
```

●Visual Basic

```
Declare Function DrKillEvent Lib "ifdr.dll"( _
   ByVal   DeviceHandle    As Long _
)As Long
```

●Delphi

```
function DrKillEvent (
   DeviceHandle:   THandle
): Integer; stdcall; external 'ifdr.dll';
```

●Visual C#.NET

```
[DllImport("ifdr.dll")]
public static extern uint DrKillEvent (
   IntPtr    DeviceHandle
);
```

●Visual Basic.NET

```
Declare Function DrKillEvent Lib "ifdr.dll"( _
   ByVal   DeviceHandle    As IntPtr _
)As Integer
```

【パラメータ】

DeviceHandle

DrOpen 関数で取得したデバイスハンドルを指定してください。

【戻り値】

DrKillEvent 関数は正常に終了すると IFDR_ERROR_SUCCESS を返します。
それ以外の場合は IFDR_ERROR_SUCCESS 以外の値を返します。
IFDR_ERROR_SUCCESS 以外の値が返された場合については、戻り値一覧を参照してください。

【使用例】

●C 言語

```
INT Ret;
HANDLE DeviceHandle;

Ret = DrKillEvent(DeviceHandle);
```

●Visual Basic

```
Dim Ret As Long
Dim DeviceHandle As Long

Ret = DrKillEvent(DeviceHandle)
```

●Delphi

```
Ret : Integer;
DeviceHandle : THandle;

Ret := DrKillEvent(DeviceHandle);
```

●Visual C#.NET

```
uint Ret;
IntPtr DeviceHandle;

Ret = IFCDR.DrKillEvent(DeviceHandle);
```

●Visual Basic.NET

```
Dim Ret As Integer
Dim DeviceHandle As IntPtr

Ret = IFCDR.DrKillEvent(DeviceHandle)
```

デバイスハンドル DeviceHandle のデバイスに登録されたコールバック関数とイベントハンドルを解除します。

23. DrGetEventStatus

【機能】

イベント要因を取得します。

【書式】

●C 言語

```
INT DrGetEventStatus(
  HANDLE        DeviceHandle,
  INT           Channel,
  PDWORD        EventStatus
);
```

●Visual Basic

```
Declare Function DrGetEventStatus Lib "ifdr.dll"( _
   ByVal   DeviceHandle     As Long, _
   ByVal   Channel          As Long, _
   ByRef   EventStatus      As Long _
)As Long
```

●Delphi

```
function DrGetEventStatus (
   DeviceHandle:         Thandle;
   Channel:              Integer;
   var EventStatus:      Dword
): Integer; stdcall; external 'ifdr.dll';
```

●Visual C#.NET

```
[DllImport("ifdr.dll")]
public static extern uint DrGetEventStatus (
   IntPtr      DeviceHandle,
   uint        Channel,
   out uint    EventStatus
);
```

●Visual Basic.NET

```
Declare Function DrGetEventStatus Lib "ifdr.dll"( _
   ByVal   DeviceHandle     As IntPtr, _
   ByVal   Channel          As Integer, _
   ByRef   EventStatus      As Integer _
)As Integer
```

【パラメータ】

DeviceHandle

DrOpen 関数で取得したデバイスハンドルを指定してください。
DrOpen で取得したデバイス番号を指定します。

Channel

チャンネルを指定します。

EventStatus

イベント要因を取得する変数のポインタを指定します。
（0：イベント未発生、1：イベント発生）

ビット 31～16

予約

ビット 15～13 12 11～9 8

予約	SPE	予約	UN RUN

ビット 7,6 5 4 3 2 1 0

予約	AGL3 DOWN	AGL3 UP	AGL2 DOWN	AGL2 UP	AGL1 DOWN	AGL1 UP

	内容
AGL1 UP	順方向角度割り込み 1 発生
AGL1 DOWN	逆方向角度割り込み 1 発生
AGL2 UP	順方向角度割り込み 2 発生
AGL2 DOWN	逆方向角度割り込み 2 発生
AGL3 UP	順方向角度割り込み 3 発生
AGL3 DOWN	逆方向角度割り込み 3 発生
UNRUN	アンダーラン時に割り込み発生
SPE	パターン出力停止検出

【戻り値】

DrGetEventStatus 関数は正常に終了すると IFDR_ERROR_SUCCESS を返します。
それ以外の場合は IFDR_ERROR_SUCCESS 以外の値を返します。
IFDR_ERROR_SUCCESS 以外の値が返された場合については、戻り値一覧を参照してください。

【使用例】

●C 言語

```
INT Ret;
HANDLE DeviceHandle;
DWORD Status;

Ret = DrGetEventStatus (DeviceHandle, 1, &Status);
```

●Visual Basic

```
Dim Ret As Long
Dim DeviceHandle As Long
Dim Status As Long

Ret = DrGetEventStatus (DeviceHandle, 1, Status)
```

●Delphi

```
Ret : Integer;
DeviceHandle : THandle;
Status: Dword;

Ret := DrGetEventStatus (DeviceHandle, 1, Status);
```

●Visual C#.NET

```
uint Ret;
IntPtr DeviceHandle;
uint Status;

Ret = IFCDR.DrGetEventStatus (DeviceHandle, 1, out Status);
```

●Visual Basic.NET

```
Dim Ret As Integer
Dim DeviceHandle As IntPtr
Dim Status As Integer

Ret = IFCDR.DrGetEventStatus (DeviceHandle, 1, Status)
```

デバイスハンドル DeviceHandle のデバイスのイベント要因を取得します。

4.3　コールバック関数

4.3.1 DrSetFlashRomData 関数により設定されるコールバック関数

DrSetFlashRomData 関数にて登録し、割り込みイベントが発生した時にコールされるコールバックルーチンです。

【記述】

コールバックルーチンを使用する場合、下記のように記述します。
(下記はコールバックルーチンを EventProc とする場合の例です。)

●C 言語

```
void CALLBACK EventProc(INT Channel, DWORD Event, PVOID UserData)
{
   // 割り込みイベントに対応する処理を記述します
}
```

コールバックルーチンの関数型 LPDRCALLBACK は下記のように定義されます。

```
typedef void (CALLBACK DRCALLBACK)(INT Channel, DWORD Event, PVOID UserData);
typedef DRCALLBACK FAR *LPDRCALLBACK;
```

●Visual Basic

```
Function EventProc(Channel As Long, Event As Long, UserData As Long)
     ‘ 割り込みイベントに対応する処理を記述します
End Function
```

コールバックルーチンは DrSetFlashRomData 関数の呼び出しを行うプロジェクト内の標準モジュールの中に記述しなければなりません。
DrSetFlashRomData 関数の引数パラメータでプロシージャのアドレスを渡す為に AddressOf 演算子を使用します。(DrSetEvent 関数の使用例を参照してください。) AddressOf 演算子を使うと、プロシージャからの戻り値ではなく、プロシージャ自体のアドレスが、ダイナミック リンク ライブラリ (DLL)の DrSetFlashRomData 関数に渡されます。

●Delphi

```
procedure EventProc(Channel:Integer; Event:Dword; UserData:Pointer); stdcall;
begin
    // 割り込みイベントに対応する処理を記述します
end;
```

●Visual C#.NET

```
void EventProc(int Channel, uint Event, IntPtr UserData)
{
   // 割り込みイベントに対応する処理を記述します
}
```

●Visual Basic.NET

```
Private Sub EventProc(Channel As Integer, Event As Integer, UserData As IntPtr)
    ‘ 割り込みイベントに対応する処理を記述します
End Sub
```

【パラメータ】

Channel

割り込みが発生したチャンネルが渡されます。

Event

発生した割り込みイベント要因が渡されます。

0：FLASH ROM への書き込み成功

1：FLASH ROM への書き込みに失敗

UserData

コールバック登録時に設定したユーザデータが渡されます。

【戻り値】

コールバックルーチンに戻り値はありません。

4.3.2 DrSetEvent 関数により設定されるコールバック関数

DrSetEvent 関数にて登録し、割り込みイベントが発生した時にコールされるコールバックルーチンです。

【記述】

コールバックルーチンを使用する場合、下記のように記述します。
(下記はコールバックルーチンを EventProc とする場合の例です。)

●C 言語

```
void CALLBACK EventProc(INT Channel, DWORD Event, PVOID UserData)
{
   // 割り込みイベントに対応する処理を記述します
}
```

コールバックルーチンの関数型 LPDRCALLBACK は下記のように定義されます。

```
typedef void (CALLBACK DRCALLBACK)(INT Channel, DWORD Event, PVOID UserData);
typedef DRCALLBACK FAR *LPDRCALLBACK;
```

●Visual Basic

```
Function EventProc(Channel As Long, Event As Long, UserData As Long)
     ‘ 割り込みイベントに対応する処理を記述します
End Function
```

コールバックルーチンは DrSetEvent 関数の呼び出しを行うプロジェクト内の標準モジュールの中に記述しなければなりません。
DrSetEvent 関数の引数パラメータでプロシージャのアドレスを渡す為に AddressOf 演算子を使用します。(DrSetEvent 関数の使用例を参照してください。) AddressOf 演算子を使うと、プロシージャからの戻り値ではなく、プロシージャ自体のアドレスが、ダイナミック リンク ライブラリ (DLL)の DrSetEvent 関数に渡されます。

●Delphi

```
procedure EventProc(Channel:Integer; Event:Dword; UserData:Pointer); stdcall;
begin
    // 割り込みイベントに対応する処理を記述します
end;
```

●Visual C#.NET

```
void EventProc(int Channel, uint Event, IntPtr UserData)
{
   // 割り込みイベントに対応する処理を記述します
}
```

●Visual Basic.NET

```
Private Sub EventProc(Channel As Integer, Event As Integer, UserData As IntPtr)
    ‘ 割り込みイベントに対応する処理を記述します
End Sub
```

【パラメータ】

Channel

割り込みが発生したチャンネルが渡されます。

Event

発生した割り込みイベント要因が渡されます。
(0：イベント未発生、1：イベント発生)

ビット 31～16

予約

ビット 15～13 12 11～9 8

予約	SPE	予約	UN RUN

ビット 7,6 5 4 3 2 1 0

予約	AGL3 DOWN	AGL3 UP	AGL2 DOWN	AGL2 UP	AGL1 DOWN	AGL1 UP

	内容
AGL1 UP	順方向角度割り込み 1 発生
AGL1 DOWN	逆方向角度割り込み 1 発生
AGL2 UP	順方向角度割り込み 2 発生
AGL2 DOWN	逆方向角度割り込み 2 発生
AGL3 UP	順方向角度割り込み 3 発生
AGL3 DOWN	逆方向角度割り込み 3 発生
UNRUN	アンダーラン時に割り込み発生
SPE	パターン出力停止検出

UserData

コールバック登録時に設定したユーザデータが渡されます。

【戻り値】

コールバックルーチンに戻り値はありません。

4.3.3 Visual Basic 使用時の制約事項

●Microsoft Visual Basic 6.0 上で弊社ソフトウェアライブラリが提供する関数コールバック機能を使用した場合、下記のアプリケーションエラーが発生する場合があります。

「“0x660d64d0”の命令が“0x0000009c”のメモリを参照しました。メモリが“written”になることはできませんでした。」
※”0x660d64d0”は異なる場合があります

アプリケーションエラーは下記の条件で発生します。
○登録したコールバック関数内で、下記の関数、ステートメントをコールする。
・関数コール（弊社ソフトウェアライブラリが提供する関数含む）
・Visual Basic のステートメント（Str()など）
・スタティックテキストへの文字列代入
など

また、この問題は、Visual Basic 6.0 の Learning、Professional、Enterprise Edition のすべてに当てはまり、サービスパックの適用有無にかかわらず発生します。

コールバック関数内で、アプリケーションエラーを発生させる処理を行わずに別の機能を利用することで目的の処理が実行できるように設計を変更してください。

Microsoft Visual Basic 6.0 は、スレッディング モデルとして、アパートメント モデルを採用しています。
Microsoft Visual Basic 6.0 が作成、起動したスレッド以外からコールバック関数が実行された場合にアプリケーションエラーが発生する場合があります。

弊社ソフトウェアライブラリでは、ライブラリ内で起動した別のスレッドから登録されたコールバック関数の実行を行いますので、この問題が発生します。

4.4　構造体説明

構造体名	説明
DRSTATUS	DrGetStatus関数で使用する構造体です。
DRCONFIG	DrGetConfig / DrSetConfig関数で使用する構造体です。
DRHYSTERESIS	DRCONFIG構造体で使用する構造体です。

4.4.1　DRSTATUS 構造体

●C 言語

```
typedef struct {
  DWORD             Status;
  DWORD             FlashStatus;
  DWORD             OutputStatus;
  DWORD             NoiseStatus;
  DWORD             OutputCount;
  DWORD             OutputRepeat;
  DWORD             Angle;
  DWORD             LastAngle;
  DWORD             Rotation;
} DRSTATUS, *PDRSTATUS;
```

●Visual Basic

```
Type DRSTATUS
  Status            As Long
  FlashStatus       As Long
  OutputStatus      As Long
  NoiseStatus       As Long
  OutputCount       As Long
  OutputRepeat      As Long
  Angle             As Long
  LastAngle         As Long
  Rotation          As Long
End Type
```

●Delphi

```
PDRSTATUS = ^ DRSTATUS;
TDRSTATUS = record
  Status:           Dword;
  FlashStatus:      Dword;
  OutputStatus:     Dword;
  NoiseStatus:      Dword;
  OutputCount:      Dword;
  OutputRepeat:     Dword;
  Angle:            Dword;
  LastAngle:        Dword;
  Rotation:         Dword;
end;
```

● Visual C# .NET

```
[StructLayout(LayoutKind.Sequential)]
public struct DRSTATUS
    uint            Status;
    uint            FlashStatus;
    uint            OutputStatus;
    uint            NoiseStatus;
    uint            OutputCount;
    uint            OutputRepeat;
    uint            Angle;
    uint            LastAngle;
    uint            Rotation;
}
```

● Visual Basic .NET

```
<StructLayout(LayoutKind.Sequential)>
Structure DRSTATUS
    Status          As Integer
    FlashStatus     As Integer
    OutputStatus    As Integer
    NoiseStatus     As Integer
    OutputCount     As Integer
    OutputRepeat    As Integer
    Angle           As Integer
    LastAngle       As Integer
    Rotation        As Integer
End Structure
```

メンバ	説明
Status	位相ロック状態ステータスを取得します 1：位相ロック 0：位相未ロック
FlashStatus	Flash ROM 書き込み状態を取得します 0：書き込み中でない 1：書き込み中
OutputStatus	パターン出力状態を取得します

識別子	値	内容
IFDR_STATUS_NOW_OUTPUT	0x00	パターン出力中
IFDR_STATUS_STOP_OUTPUT	0x01	パターン出力停止

NoiseStatus 異常信号出力状態を取得します

識別子	値	内容
IFDR_STATUS_NOW_OUTPUT	0x00	異常信号出力中

IFDR_STATUS_STOP_OUTPUT	0x01	異常信号出力停止

OutputCount　　パターン出力済みカウンタを取得します。
設定した出力件数でデータは折り返されます。
パターン出力時のみ有効です。
それ以外では、0 が返ります。

OutputRepeat　　パターン出力繰り返しカウンタを取得します。
パターン出力時のみ有効です。
それ以外では、0 が返ります。

Angle　　現在角度を取得します　※1
範囲：0x0000～0xFFFF
0000h：0°
0001h：0.0054931640625°
0002h：0.010986328125°
0003h：0.0164794921875°
：
FFFFh：359.9945068359375°

LastAngle　　前回角度を取得します。　※1
範囲：0x0000～0xFFFF
0000h：0°
0001h：0.0054931640625°
0002h：0.010986328125°
0003h：0.0164794921875°
：
FFFFh：359.9945068359375°

Rotation　　現在の回転数を取得します。
範囲：0x0000～0xFFFF
最上位ビットが符号の 2 の補数となります。

※1
現在角度、前回角度は新たな角度が設定された際（DrOutputAngle 関数実行時、更新レート更新時）に更新されます。
DrOutputAngle 関数を使用する場合でも更新レートが加算されるため、更新レート毎に前回角度は更新されます。

4.4.2 DRCONFIG 構造体

●C 言語

```
typedef struct {
  DWORD                    Ratio;
  DWORD                    PhaseGapMode;
  DWORD                    MisJoin;
  DWORD                    Origin;
  DWORD                    Hysteresis[3];
  DWORD                    TrgOutMode;
} DRCONFIG, *PDRCONFIG;
```

●Visual Basic

```
Type DRCONFIG
  Ratio                    As Long
  PhaseGapMode             As Long
  MisJoin                  As Long
  Origin                   As Long
  Hysteresis(0 To 2)       As DRHYSTERESIS
  TrgOutMode               As Long
End Type
```

●Delphi

```
PDRCONFIG = ^ DRCONFIG;
TDRCONFIG = record
  Ratio:                   Dword;
  PhaseGapMode:            Dword;
  MisJoin:                 Dword;
  Origin:                  Dword;
  Hysteresis :             array[0..2] of DRHYSTERESIS;
  TrgOutMode:              Dword;
end;
```

● Visual C# .NET

```
[StructLayout(LayoutKind.Sequential)]
public struct DRCONFIG
{
   public uint             Ratio;
   public uint             PhaseGapMode;
   public uint             MisJoin;
   public uint             Origin;
   public DRHYSTERESIS     Hysteresis1;
   public DRHYSTERESIS     Hysteresis2;
   public DRHYSTERESIS     Hysteresis3;
   public uint             TrgOutMode;
}
```

● Visual Basic .NET

```
<StructLayout(LayoutKind.Sequential)>
Structure DRCONFIG
    Public Ratio            As Integer
    Public PhaseGapMode     As Integer
    Public MisJoin          As Integer
    Public Origin           As Integer
    Public Hysteresis1      As DRHYSTERESIS
    Public Hysteresis2      As DRHYSTERESIS
    Public Hysteresis3      As DRHYSTERESIS
    Public TrgOutMode       As Integer
End Structure
```

メンバ	説明
Ratio	入力電圧に対する出力電圧の変圧比を指定します。 変圧比を変更することで、励磁信号に対する出力信号の振幅を変更する事が出来ます。 （デフォルト：0x8000） 設定範囲：0x3333～0x8000 3333h：0.199996948 ： 4000h：0.25 ： 8000h：0.5 [計算式] 変圧比 = 設定値 × 2^(-16) < 例 > 13107 × 2^(-16) = 0.199996948 (3333h)
PhaseGapMode	位相ズレ補正モードを指定します。※1 （デフォルト：IFDR_DISABLE）

識別子	値	内容
IFDR_DISABLE	0x00	位相ズレ補正無効（遅延小）
IFDR_ENABLE	0x01	位相ズレ補正有効（遅延大）

※ 位相ズレ補正有りを指定した場合、DrGetStatus 関数を実行して位相ロック状態になるのを待つ必要があります。

MisJoin 接続状態を指定します。
（デフォルト：IFDR_JOIN）

識別子	値	内容
IFDR_JOIN	0x00	正常接続
IFDR_MIS_JOIN1	0x01	sin_p1 と sin_n1 の入れ替え
IFDR_MIS_JOIN2	0x02	cos_p1 と cos _n1 の入れ替え
IFDR_MIS_JOIN3	0x03	sin_p1 と cos_p1 の入れ替え

IFDR_MIS_JOIN4	0x04	断線状態

Origin　　原点補正値（原点となる角度）を指定します。※2
（デフォルト：0x0000）

Hysteresis　　ヒステリシスを設定する構造体（DRHYSTERESIS）です。※3
Hysteresis［0］：角度割り込み 1
Hysteresis［1］：角度割り込み 2
Hysteresis［2］：角度割り込み 3

TrgOutMode　　角度トリガ出力条件を指定します。※3
（デフォルト：0）
角度割り込み要因が発生した場合、デジタル出力が反転してます。
DrOutputDO 関数でのデジタル出力の設定は反映されません。

ビット　31～8

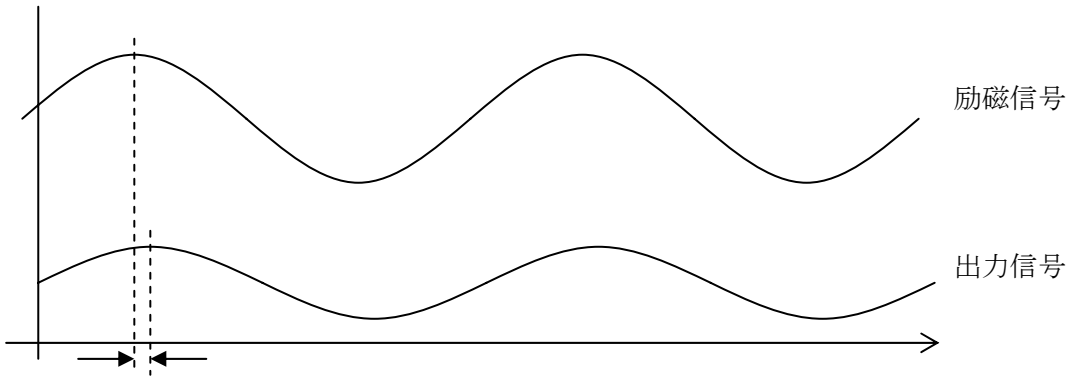
予約

ビット

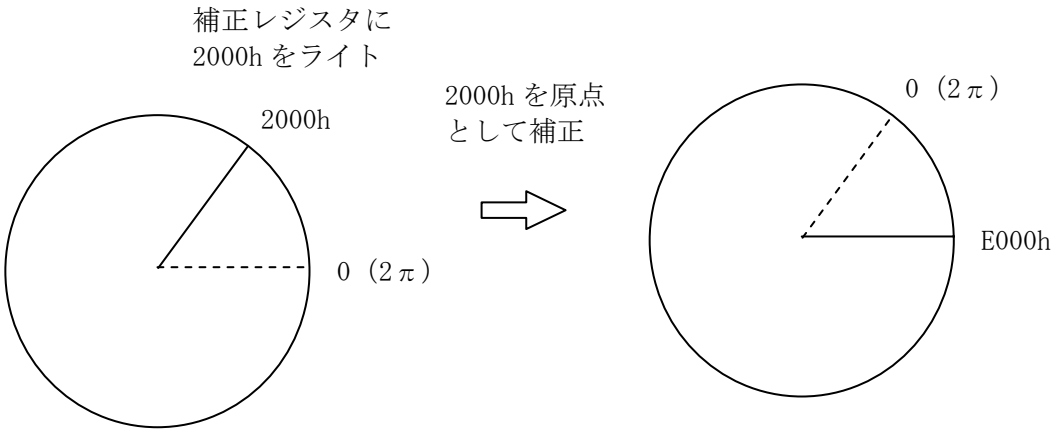
7,6	5	4	3	2	1	0
予約	OUT6	OUT5	OUT4	OUT3	OUT2	OUT1

	内容
OUT1	順方向角度割り込み 1 発生
OUT2	逆方向角度割り込み 1 発生
OUT3	順方向角度割り込み 2 発生
OUT4	逆方向角度割り込み 2 発生
OUT5	順方向角度割り込み 3 発生
OUT6	逆方向角度割り込み 3 発生

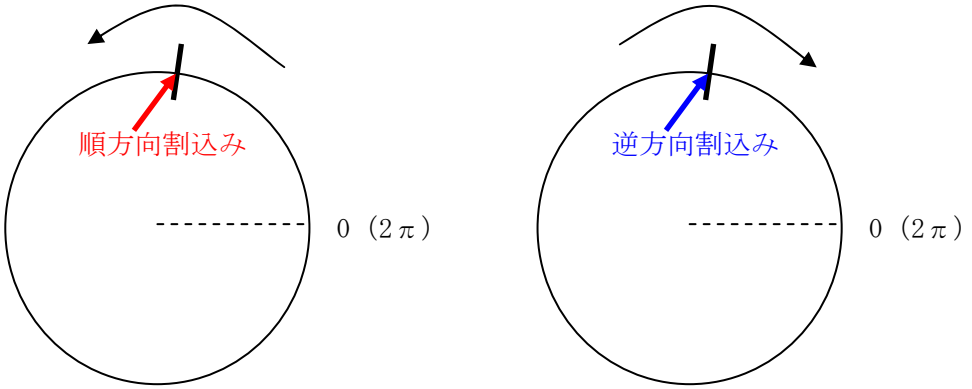
※1 位相ズレ補正モードについて
励磁信号に対する出力の信号位相差は、ソフトウェアにて調整を行う事ができます。



※2 原点補正について
レジスタにライトした値分原点を補正する。
モータとレゾルバ原点がずれた状態を再現することができる。



※3 角度トリガ出力/ヒステリシスについて
レゾルバの角度が指定角度になった場合、割り込みによる通知を行う。
また、レジスタの設定によって信号を汎用出力ピンから出力することができる。
指定角度になった場合、信号が反転する。
割り込みは順方向、逆方向の割り込みを持つ。
指定角度は回転数＋角度の組合せで指定でき、ヒステリシスは上限/下限で指定する。



ヒステリシス機能
下限の順方向通過 → 内部の順方向フラグを立てる → 上限の順方向通過　で順方向割込み
下限の順方向通過 → 内部の順方向フラグを立てる → 下限の逆方向通過　でフラグを下ろす
上限の逆方向通過 → 内部の逆方向フラグを立てる → 下限の逆方向通過　で逆方向割込み
上限の逆方向通過 → 内部の逆方向フラグを立てる → 上限の順方向通過　でフラグを下ろす

4.4.3 DRHYSTERESIS 構造体

●C 言語

```
typedef struct {
  DWORD              HighRotation;
  DWORD              HighAngle;
  DWORD              LowRotation;
  DWORD              LowAngle;
} DRHYSTERESIS, * PDRHYSTERESIS;
```

●Visual Basic

```
Type DRHYSTERESIS
  HighRotation       As Long
  HighAngle          As Long
  LowRotation        As Long
  LowAngle           As Long
End Type
```

●Delphi

```
PDRHYSTERESIS = ^ DRHYSTERESIS;
TDRHYSTERESIS = record
  HighRotation:      Dword;
  HighAngle:         Dword;
  LowRotation:       Dword;
  LowAngle:          Dword;
end;
```

● Visual C# .NET

```
[StructLayout(LayoutKind.Sequential)]
public struct DRHYSTERESIS
   uint              HighRotation;
   uint              HighAngle;
   uint              LowRotation;
   uint              LowAngle;
}
```

● Visual Basic .NET

```
<StructLayout(LayoutKind.Sequential)>
Structure DRHYSTERESIS
   HighRotation      As Integer,
   HighAngle         As Integer,
   LowRotation       As Integer,
   LowAngle          As Integer
End Structure
```

メンバ	説明
HighRotation	割り込み、トリガを発生させる回転数のヒステリシス上限値を指定します。 （デフォルト：0x8000） 設定できる範囲は、0x0000～0xFFFF の範囲で指定できます。 最上位ビットが符号の 2 の補数となります。 0x8000 を指定した場合、回転数は無視されます。 ヒステリシス上限を 0x8000 とした場合は下限も 0x8000 とします。
HighAngle	割り込み、トリガを発生させるヒステリシス上限値の角度を指定します。 （デフォルト：0x0000） 設定できる範囲は、0x0000～0xFFFF の範囲で指定できます。 ※360° ÷ 10000h （＝ 0.0054931640625°）毎に指定可能です。 0000h：0° 0001h：0.0054931640625° 0002h：0.010986328125° 0003h：0.0164794921875° ： FFFFh：359.9945068359375°
LowRotation	割り込み、トリガを発生させる回転数のヒステリシス下限値を指定します。 （デフォルト：0x8000） 設定できる範囲は、0x0000～0xFFFF の範囲で指定できます。 最上位ビットが符号の 2 の補数となります。 0x8000 を指定した場合、回転数は無視されます。 ヒステリシス上限を 0x8000 とした場合は下限も 0x8000 とします。
LowAngle	割り込み、トリガを発生させるヒステリシス下限値の角度を指定します。 （デフォルト：0x0000） 設定できる範囲は、0x0000～0xFFFF の範囲で指定できます。 ※360° ÷ 10000h （＝ 0.0054931640625°）毎に指定可能です。 0000h：0° 0001h：0.0054931640625° 0002h：0.010986328125° 0003h：0.0164794921875° ： FFFFh：359.9945068359375°

4.5 戻り値一覧

エラー識別子	値	意味	対処方法
IFDR_ERROR_SUCCESS	0	正常終了	
IFDR_ERROR_NOT_DEVICE	0xC0000001	ドライバを呼び出せません。	指定したデバイスが見つかりませんでした。 指定したデバイス番号が存在するかどうかを確認してください。
IFDR_ERROR_NOT_OPEN	0xC0000002	ドライバをOPENできません。	デバイスのオープン時、何らかのエラーが発生しました。 ドライバ内部のメモリ確保に失敗したなど。
IFDR_ERROR_INVALID_HANDLE	0xC0000003	デバイスハンドルが正しくありません。	不正なデバイスハンドルで呼び出しを行おうとしました。 OPEN 関数で返されたデバイスハンドルを使用してください。
IFDR_ERROR_ALREADY_OPEN	0xC0000004	既にオープンしているデバイスです。	既にオープンされているデバイスをオープンしようとしました。
IFDR_ERROR_INSUFFICIENT_BUFFER	0xC0000005	システムコールに渡されたデータ領域が小さすぎます。	ドライバの内部エラーです。どのような状況でエラーが発生したかをご連絡ください。
IFDR_ERROR_IO_PENDING	0xC0000006	非同期 I/O 操作が進行中です。	Win32API の WaitForSingleObject 関数等でイベントの完了を待つことができます。
IFDR_ERROR_NOT_SUPPORTED	0xC0000007	サポートされていない機能です。	ご使用になるインタフェースモジュールがサポートしていない機能を制御する関数をコールした場合にエラーコード IFDR_ERROR_NOT_SUPPORTED が返されます。
IFDR_ERROR_INVALID_PARAMETER	0xC0000008	パラメータが正しくありません。	関数説明を確認し、正しいパラメータ設定を指定してください。
IFDR_ERROR_NOT_ALLOCATE_MEMORY	0xC0000009	メモリの確保に失敗しました。	利用可能な空きメモリが不足しています。
IFDR_ERROR_NULL_POINTER	0xC000000A	NULL ポインタを指定しました	パターン出力データ領域の指定に NULL を

エラー識別子	値	意味	対処方法
			指定しました。
IFDR_ERROR_ALREADY_REGISTRATION	0xC000000B	既にイベントが登録されています。	イベントの登録を解除後に、再度登録を行ってください。
IFDR_ERROR_ALREADY_DELETE	0xC000000C	既にイベント登録が解除されています。	イベントを登録後に解除を実行してください。
IFDR_ERROR_NOW_OUTPUT	0xC0001001	既にパターン出力中です。	パターン出力中に実行できない関数を呼び出そうとしました。
IFDR_ERROR_STOP_OUTPUT	0xC0001002	既にパターン出力停止中です。	パターン出力停止中に実行できない関数を呼び出そうとしました。
IFDR_ERROR_INVALID_CHANNEL	0xC0001003	チャンネル設定が正しくありません。	関数説明を確認し、正しいチャンネル設定を指定してください。
IFDR_ERROR_INVALID_NUM	0xC0001004	件数設定が正しくありません。	関数説明を確認し、正しい件数設定を指定してください。
IFDR_ERROR_INVALID_RATE	0xC0001005	タイマ設定が正しくありません。	関数説明を確認し、正しいタイマ設定を指定してください。
IFDR_ERROR_INVALID_START_NUM	0xC0001006	開始位置設定が正しくありません。	関数説明を確認し、正しい開始位置設定を指定してください。
IFDR_ERROR_INVALID_REP	0xC0001007	繰り返し設定が正しくありません。	関数説明を確認し、正しい繰り返し設定を指定してください。
IFDR_ERROR_INVALID_RATIO	0xC0001008	変圧比設定が正しくありません。	関数説明を確認し、正しい変圧比設定を指定してください。
IFDR_ERROR_INVALID_PHASE	0xC0001009	位相ズレ補正モード設定が正しくありません。	関数説明を確認し、正しい位相ズレ補正モード設定を指定してください。
IFDR_ERROR_INVALID_MISJOIN	0xC000100A	接続状態設定が正しくありません。	関数説明を確認し、正しい接続状態設定を指定してください。
IFDR_ERROR_SET_DATA	0xC0001010	設定値が正しくありません。	関数説明を確認し、正しい設定値を指定してください。
IFDR_ERROR_NOW_WRITE	0xC0001011	FLASH ROM 書き込み中です。	FLASH ROM 書き込み中に実行できない関数を呼び出そうとしました。

第5章　サンプルプログラム

以下、各サンプルプログラムの概要を説明します。

5.1　Angle

【概要】

IFDR1 のデバイスをオープンし、一定角度出力を開始します。

【実行手順】

サンプルプログラムには実行形式のファイルが付属していません。
ソースコードをコンパイルして実行ファイルを生成してから、動作させてください。

●Visual C++の場合

・ Visual C++ の場合(Microsoft Visual C++ .NET 2003～2008)
プロジェクトファイル「Angle.vcproj」を開き、ビルドしてください。

・ Visual C++ の場合(Microsoft Visual C++ .NET 2010 以降)
プロジェクトファイル「Angle.vcxproj」を開き、ビルドしてください。
(Visual Studio 2010 より新しいVersionで実行する場合には後述の 注意事項を参照下さい)

●Visual Basic の場合

Visual Basic を起動し、プロジェクトファイル「Angle.vbp」を開き、ビルドしてください。

●Delphi の場合

Delphi を起動し、プロジェクトファイル「Angle.dpr」を開き、ビルドしてください。

●Visual C#.NET の場合

Visual Studio.NET を起動し、ソリューションファイル「Angle.sln」を開き、ビルドしてください。

●Visual Basic.NET の場合

Visual Studio.NET を起動し、ソリューションファイル「Angle.sln」を開き、ビルドしてください。

作成後、「Angle」を起動してください。

【内容】

●Visual C++ / Visual C#.NET / Visual Basic.NET

コンソールアプリケーションです。
デバイス名「IFDR1」のデバイスをオープン後、チャンネル 1 から一定角度出力（0x8000）を行います。

●Visual Basic / Delphi

ボタンをクリックするとデバイス名「IFDR1」のデバイスをオープン後、チャンネル 1 から一定角度出力（0x8000）を行います。

5.2 Velocity

【概要】

IFDR1 のデバイスをオープンし、一定速度出力を開始します。

【実行手順】

サンプルプログラムには実行形式のファイルが付属していません。
ソースコードをコンパイルして実行ファイルを生成してから、動作させてください。

●Visual C++の場合

・ Visual C++ の場合(Microsoft Visual C++ .NET 2003～2008)
プロジェクトファイル「Velocity.vcproj」を開き、ビルドしてください。

・ Visual C++ の場合(Microsoft Visual C++ .NET 2010 以降)
プロジェクトファイル「Velocity.vcxproj」を開き、ビルドしてください。
(Visual Studio 2010 より新しいVersionで実行する場合には後述の 注意事項を参照下さい)

●Visual Basic の場合

Visual Basic を起動し、プロジェクトファイル「Velocity.vbp」を開き、ビルドしてください。

●Delphi の場合

Delphi を起動し、プロジェクトファイル「Velocity.dpr」を開き、ビルドしてください。

●Visual C#.NET の場合

Visual Studio.NET を起動し、ソリューションファイル「Velocity.sln」を開き、ビルドしてください。

●Visual Basic.NET の場合

Visual Studio.NET を起動し、ソリューションファイル「Velocity.sln」を開き、ビルドしてください。

作成後、「Velocity」を起動してください。

【内容】

●Visual C++ / Visual C#.NET / Visual Basic.NET

コンソールアプリケーションです。
デバイス名「IFDR1」のデバイスをオープン後、チャンネル 1 から一定速度出力を行います。

●Visual Basic / Delphi

ボタンをクリックするとデバイス名「IFDR1」のデバイスをオープン後、チャンネル 1 か一定速度出力を行います。

5.3 Output

【概要】

IFDR1 のデバイスをオープンし、一定速度出力を開始します。

【実行手順】

サンプルプログラムには実行形式のファイルが付属していません。
ソースコードをコンパイルして実行ファイルを生成してから、動作させてください。

●Visual C++の場合

・ Visual C++ の場合(Microsoft Visual C++ .NET 2003～2008)
プロジェクトファイル「Output.vcproj」を開き、ビルドしてください。

・ Visual C++ の場合(Microsoft Visual C++ .NET 2010 以降)
プロジェクトファイル「Output.vcxproj」を開き、ビルドしてください。
(Visual Studio 2010 より新しいVersionで実行する場合には後述の 注意事項を参照下さい)

●Visual Basic の場合

Visual Basic を起動し、プロジェクトファイル「Output.vbp」を開き、ビルドしてください。

●Delphi の場合

Delphi を起動し、プロジェクトファイル「Output.dpr」を開き、ビルドしてください。

●Visual C#.NET の場合

Visual Studio.NET を起動し、ソリューションファイル「Output.sln」を開き、ビルドしてください。

●Visual Basic.NET の場合

Visual Studio.NET を起動し、ソリューションファイル「Output.sln」を開き、ビルドしてください。

作成後、「Output」を起動してください。

【内容】

●Visual C++ / Visual C#.NET / Visual Basic.NET

コンソールアプリケーションです。
デバイス名「IFDR1」のデバイスをオープン後、チャンネル 1 からパターン出力（FIFO データ）を行います。

●Visual Basic / Delphi

ボタンをクリックするとデバイス名「IFDR1」のデバイスをオープン後、チャンネル 1 からパターン出力（FIFO データ）を行います。

5.4 Pattern

【概要】

IFDR1 のデバイスをオープンし、一定角度出力を開始します。

【実行手順】

サンプルプログラムには実行形式のファイルが付属していません。
ソースコードをコンパイルして実行ファイルを生成してから、動作させてください。

●Visual C++の場合

・ Visual C++ の場合(Microsoft Visual C++ .NET 2003～2008)
プロジェクトファイル「Pattern.vcproj」を開き、ビルドしてください。

・ Visual C++ の場合(Microsoft Visual C++ .NET 2010 以降)
プロジェクトファイル「Pattern.vcxproj」を開き、ビルドしてください。
(Visual Studio 2010 より新しいVersionで実行する場合には後述の 注意事項を参照下さい)

●Visual Basic の場合

Visual Basic を起動し、プロジェクトファイル「Pattern.vbp」を開き、ビルドしてください。

●Delphi の場合

Delphi を起動し、プロジェクトファイル「Pattern.dpr」を開き、ビルドしてください。

●Visual C#.NET の場合

Visual Studio.NET を起動し、ソリューションファイル「Pattern.sln」を開き、ビルドしてください。

●Visual Basic.NET の場合

Visual Studio.NET を起動し、ソリューションファイル「Pattern.sln」を開き、ビルドしてください。

作成後、「Pattern」を起動してください。

【内容】

●Visual C++ / Visual C#.NET / Visual Basic.NET

コンソールアプリケーションです。
デバイス名「IFDR1」のデバイスをオープン後、チャンネル 1 からパターン出力 (FLASH ROM データ) を行います。

●Visual Basic / Delphi

ボタンをクリックするとデバイス名「IFDR1」のデバイスをオープン後、チャンネル 1 からパターン出力 (FLASH ROM データ) を行います。

5.5 Noise

【概要】

IFDR1 のデバイスをオープンし、汎用入力状態を取得します。

【実行手順】

サンプルプログラムには実行形式のファイルが付属していません。
ソースコードをコンパイルして実行ファイルを生成してから、動作させてください。

●Visual C++の場合

・ Visual C++ の場合(Microsoft Visual C++ .NET 2003～2008)
プロジェクトファイル「Noise.vcproj」を開き、ビルドしてください。

・ Visual C++ の場合(Microsoft Visual C++ .NET 2010 以降)
プロジェクトファイル「Noise.vcxproj」を開き、ビルドしてください。
(Visual Studio 2010 より新しいVersionで実行する場合には後述の 注意事項を参照下さい)

●Visual Basic の場合

Visual Basic を起動し、プロジェクトファイル「Noise.vbp」を開き、ビルドしてください。

●Delphi の場合

Delphi を起動し、プロジェクトファイル「Noise.dpr」を開き、ビルドしてください。

●Visual C#.NET の場合

Visual Studio.NET を起動し、ソリューションファイル「Noise.sln」を開き、ビルドしてください。

●Visual Basic.NET の場合

Visual Studio.NET を起動し、ソリューションファイル「Noise.sln」を開き、ビルドしてください。

作成後、「Noise」を起動してください。

【内容】

●Visual C++ / Visual C#.NET / Visual Basic.NET

コンソールアプリケーションです。
デバイス名「IFDR1」のデバイスをオープン後、チャンネル 1 から異常信号出力を行います。

●Visual Basic / Delphi

ボタンをクリックするとデバイス名「IFDR1」のデバイスをオープン後、チャンネル 1 か異常信号出力を行います。

5.6 InputDi

【概要】

IFDR1 のデバイスをオープンし、汎用出力端子の制御を行います。

【実行手順】

サンプルプログラムには実行形式のファイルが付属していません。
ソースコードをコンパイルして実行ファイルを生成してから、動作させてください。

●Visual C++の場合

・ Visual C++ の場合(Microsoft Visual C++ .NET 2003～2008)
プロジェクトファイル「InputDi.vcproj」を開き、ビルドしてください。

・ Visual C++ の場合(Microsoft Visual C++ .NET 2010 以降)
プロジェクトファイル「InputDi.vcxproj」を開き、ビルドしてください。
(Visual Studio 2010 より新しいVersionで実行する場合には後述の 注意事項を参照下さい)

●Visual Basic の場合

Visual Basic を起動し、プロジェクトファイル「InputDi.vbp」を開き、ビルドしてください。

●Delphi の場合

Delphi を起動し、プロジェクトファイル「InputDi.dpr」を開き、ビルドしてください。

●Visual C#.NET の場合

Visual Studio.NET を起動し、ソリューションファイル「InputDi.sln」を開き、ビルドしてください。

●Visual Basic.NET の場合

Visual Studio.NET を起動し、ソリューションファイル「InputDi.sln」を開き、ビルドしてください。

作成後、「InputDi」を起動してください。

【内容】

●Visual C++ / Visual C#.NET / Visual Basic.NET

コンソールアプリケーションです。
デバイス名「IFDR1」のデバイスをオープン後、汎用入力端子状態を取得し、取得した情報を表示します。

●Visual Basic / Delphi

ボタンをクリックするとデバイス名「IFDR1」のデバイスをオープン後、汎用入力端子状態を取得し、取得した情報をメッセージダイアログで表示します。
メッセージダイアログの OK ボタンを押すと、デバイスをクローズします。

5.7 OutputDo

【概要】

IFDR1 のデバイスをオープンし、イベントハンドルが有効になるのを確認します。

【実行手順】

サンプルプログラムには実行形式のファイルが付属していません。
ソースコードをコンパイルして実行ファイルを生成してから、動作させてください。

●Visual C++の場合

・ Visual C++ の場合(Microsoft Visual C++ .NET 2003～2008)
プロジェクトファイル「OutputDo.vcproj」を開き、ビルドしてください。

・ Visual C++ の場合(Microsoft Visual C++ .NET 2010 以降)
プロジェクトファイル「OutputDo.vcxproj」を開き、ビルドしてください。
(Visual Studio 2010 より新しいVersionで実行する場合には後述の 注意事項を参照下さい)

●Visual Basic の場合

Visual Basic を起動し、プロジェクトファイル「OutputDo.vbp」を開き、ビルドしてください。

●Delphi の場合

Delphi を起動し、プロジェクトファイル「OutputDo.dpr」を開き、ビルドしてください。

●Visual C#.NET の場合

Visual Studio.NET を起動し、ソリューションファイル「OutputDo.sln」を開き、ビルドしてください。

●Visual Basic.NET の場合

Visual Studio.NET を起動し、ソリューションファイル「OutputDo.sln」を開き、ビルドしてください。

作成後、「OutputDo」を起動してください。

【内容】

●Visual C++ / Visual C#.NET / Visual Basic.NET

コンソールアプリケーションです。
デバイス名「IFDR1」のデバイスをオープン後、指定した汎用出力を行います。

●Visual Basic / Delphi

ボタンをクリックするとデバイス名「IFDR1」のデバイスをオープン後、指定した汎用出力を行います。

5.8 Event

【概要】

IFDR1 のデバイスをオープンし、イベントが呼び出されるのを確認します。

【実行手順】

サンプルプログラムには実行形式のファイルが付属していません。
ソースコードをコンパイルして実行ファイルを生成してから、動作させてください。

●Visual C++の場合

・ Visual C++ の場合(Microsoft Visual C++ .NET 2003～2008)
プロジェクトファイル「Event.vcproj」を開き、ビルドしてください。

・ Visual C++ の場合(Microsoft Visual C++ .NET 2010 以降)
プロジェクトファイル「Event.vcxproj」を開き、ビルドしてください。
(Visual Studio 2010 より新しいVersionで実行する場合には後述の 注意事項を参照下さい)

●Visual Basic の場合

Visual Basic を起動し、プロジェクトファイル「Event.vbp」を開き、ビルドしてください。

●Delphi の場合

Delphi を起動し、プロジェクトファイル「Event.dpr」を開き、ビルドしてください。

●Visual C#.NET の場合

Visual Studio.NET を起動し、ソリューションファイル「Event.sln」を開き、ビルドしてください。

●Visual Basic.NET の場合

Visual Studio.NET を起動し、ソリューションファイル「Event.sln」を開き、ビルドしてください。

作成後、「Event」を起動してください。

【内容】

●Visual C++ / Visual C#.NET / Visual Basic.NET

コンソールアプリケーションです。
デバイス名「IFDR1」のデバイスをオープン後、チャンネル 1 からパターン出力を行い、パターン出力停止検出を行いイベントがシグナルになったら終了します。

●Visual Basic / Delphi

デバイス名「IFDR1」のデバイスをオープン後、チャンネル 1 からパターン出力を行い、パターン出力停止検出を行いイベントがシグナルになったら終了します。

5.9 Callback

【概要】

IFDR1 のデバイスをオープンし、コールバック関数が呼び出されるのを確認します。

【実行手順】

サンプルプログラムには実行形式のファイルが付属していません。
ソースコードをコンパイルして実行ファイルを生成してから、動作させてください。

●Visual C++の場合

・ Visual C++ の場合(Microsoft Visual C++ .NET 2003～2008)
プロジェクトファイル「Callback.vcproj」を開き、ビルドしてください。

・ Visual C++ の場合(Microsoft Visual C++ .NET 2010 以降)
プロジェクトファイル「Callback.vcxproj」を開き、ビルドしてください。
(Visual Studio 2010 より新しいVersionで実行する場合には後述の 注意事項を参照下さい)

●Delphi の場合

Delphi を起動し、プロジェクトファイル「Callback.dpr」を開き、ビルドしてください。

●Visual C#.NET の場合

Visual Studio.NET を起動し、ソリューションファイル「Callback.sln」を開き、ビルドしてください。

●Visual Basic.NET の場合

Visual Studio.NET を起動し、ソリューションファイル「Callback.sln」を開き、ビルドしてください。

作成後、「Callback」を起動してください。

【内容】

●Visual C++ / Visual C#.NET / Visual Basic.NET

コンソールアプリケーションです。
デバイス名「IFDR1」のデバイスをオープン後、チャンネル 1 からパターン出力を行い、パターン出力停止検出を行いコールバック関数が呼び出されると終了します。

●Delphi

ボタンをクリックするとデバイス名「IFDR1」のデバイスをオープン後、チャンネル 1 からパターン出力を行い、パターン出力停止検出を行いコールバック関数が呼び出されると終了します。

5.10 注意事項

・ Visual C# .NET, Visual Basic .NET のサンプルプログラムは「.NET Framework 2.0」を使用したサンプルプログラムとなっております。
「.NET Framework 2.0」がインストールされていない環境では、インストール、または別 Version の.NET Framework への変換が必要になります。ご注意下さい。

・ 「*.vcxproj」ファイルは Visual Studio 2010 で作成しております。
Visual Studio 2010 より新しい Version で実行する場合には、下記設定を行ってください。
Visual Studio のメニューから、「プロジェクト」- 「プロパティ」-「構成プロパティ」-「全般」を開き、「プラットフォームツールセット」を使用している環境に合わせて変更。
例) Visual Studio 2012 を使用している場合
Visual Studio 2012(v110)を選択(プルダウンメニューより選択できます)

第6章　重要な情報

保証の内容と制限

弊社は本ドキュメントに含まれるソースプログラムの実行が中断しないこと、またはその実行に誤りが無いことを保証していません。
本製品の品質や使用に起因する、性能に起因するいかなるリスクも使用者が負うものとします。
弊社はドキュメント内の情報の正確さに万全を期しています。万一、誤記または誤植などがあった場合、弊社は予告無く改訂する場合があります。ドキュメントまたはドキュメント内の情報に起因するいかなる損害に対しても弊社は責任を負いません。
ドキュメント内の図や表は説明のためであり、ユーザ個別の応用事例により変化する場合があります。

著作権、知的所有権

弊社は本製品に含まれるおよび本製品に対する権利や知的所有権を保持しています。
本製品はコンピュータ ソフトウェア、映像/音声(例えば図、文章、写真など)を含んでいます。

医療機器/器具への適用における注意

弊社の製品は人命に関わるような状況下で使用される機器に用いられる事を目的として設計、製造された物では有りません。
弊社の製品は人体の検査などに使用するに適する信頼性を確保する事を意図された部品や検査機器と共に設計された物では有りません。
医療機器、治療器具などの本製品の適用により、製品の故障、ユーザ、設計者の過失などにより、損傷/損害を引き起こす場合が有ります。

複製の禁止

弊社の許可なく、本ドキュメントの全て、または一部に関わらず、複製、改変などを行うことはできません。

責任の制限

弊社は、弊社または再販売者の予見の有無にかかわらず発生したいかなる特別損害、偶発的損害、間接的な損害、重大な損害について、責任を負いません。

本製品(ハードウェア，ソフトウェア)のシステム組み込み、使用、ならびに本製品から得られる結果に関する一切のリスクについては、本製品の使用者に帰属するものとします。

本製品に含まれるバグ、あるいは本製品の供給（納期遅延)、性能もしくは使用に起因する付帯的損害もしくは間接的損害に対して、弊社に全面的に責がある場合でも、弊社はその製品に対する改良(正常に動作する)、代品交換までとし、金銭面での賠償の責任は負わないものとしますので、予めご了承ください。

本製品（ソフトウェアを含む）は、日本国内仕様です。本製品を日本国外で使用された場合、弊社は一切責任を負いかねます。また、弊社は本製品に関し、海外での保守サービスおよび技術サポート等は行っておりません。

商標/登録商標

本書に掲載されている会社名、製品名は、それぞれ各社の商標または登録商標です。