株式会社ワイツー

DAQ-Xシリーズ デジタル入出力ユニット
ソフトウェアマニュアル

このマニュアルについて

このソフトウェアマニュアルにはソフトウェアに関する情報が記載されています。
取扱説明書（ハードウェアのマニュアル）も併せてお読み下さい。

ソフトウェアについて

本ソフトウェアは、DAQ-Xシリーズ デジタル入出力ユニットを制御する為のソフトウェアです。
入出力の制御は、提供されるDLLの関数をコールすることで実現できますので、開発者はUSB接続であるという事を意識せずに使用することができます。

簡易デジタル入力

指定したチャネルのデジタル入力を１回おこなえます。
複数チャネルの指定も可能です。
設定も少なく、簡単に使用する事ができます。

以下の関数を使用します。

- YdxDiInput関数
- YdxDiInputBit関数

一方、連続サンプリングなど色々な条件でデジタル入力をおこないたい場合は 高機能デジタル入力 を使用します。

参考

- 実行手順（簡易デジタル入力）
- サンプルプログラム（DioBit）

高機能デジタル入力

連続サンプリングなど、色々な条件でデジタル入力をおこなえます。
設定項目は多くなりますが、連続サンプリングや、開始・停止・繰り返しなどのシーケンス制御をユニット側に任せられる為、パソコン側にあまり負担をかけずに高速なシステムが実現できます。
外部クロック・外部トリガ機能を使用して、外部との同期や連携も可能です。

一方、デジタル入力を１回おこないたい場合は 簡易デジタル入力 を使用します。

参考

- 実行手順（高機能デジタル入力）
- サンプルプログラム（DiPolling）

  高機能デジタル入力のサンプルプログラムです。
  デジタル入力を1000回おこない、データを表示します。
  動作状態の監視をポーリングでおこなっています。

データバッファ

データバッファは、ユニット内部にあり、データを一時的に記憶します。
パソコンからはデータをまとめて読み出す事が可能になるため効率的で、パソコン側の負荷を大幅に軽減する事が可能になります。

用途に応じて、FIFOバッファ形式とリングバッファ形式が選択できます。

- 形式の選択には YdxDiSetBuffer関数 を使用します。
- データの読み出しには YdxDiGetData関数 を使用します。
- データをクリアするには YdxDiClearData関数 を使用します。

FIFOバッファ形式

読み出しは、古いデータから順におこなわれます。
読み出されたデータは、バッファから破棄されます。
動作中 にデータを読み出す事が可能です。
読み出されていないデータがバッファに満杯の状態でサンプリングがおこなわれると、オーバランエラーが発生します。

※ 入力動作中にデータを読み込む事が可能ですので、データバッファが満杯にならないように定期的にデータを読み出す事で容量以上の長時間のサンプリングが可能になります。

リングバッファ形式

読み出しは、新しいデータからおこなわれます。
読み出されたデータは、バッファから破棄されません。
（再度読み出す事が可能）
動作中 にデータを読み出す事はできません。
読み出されていないデータがバッファに満杯の状態でサンプリングがおこなわれると、古いデータに上書きして記憶されます。

※ 全てのデータの読み出しの必要はなく、入力動作停止直前のデータのみ必要な場合などに有効です。

サンプリングクロック

サンプリングクロックは、サンプリングのタイミングを決定します。
内部クロックと外部クロックが選択できます。
選択には以下の関数を使用します。

- YdxDiSetClock関数

内部クロック

ユニット内部でクロックを生成します。
クロック周期の設定には以下の関数を使用します。

- YdxDiSetClockInternal関数

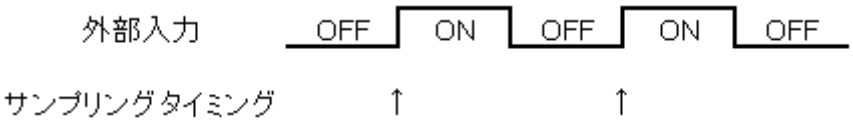
外部クロック

外部入力をクロックとして使用します。
外部クロックとして使用するデジタル入力チャネルと入力タイミングの設定には以下の関数を使用します。

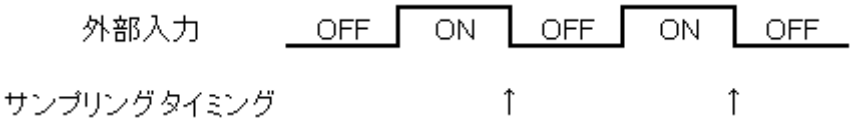
- YdxDiSetClockExternal関数

入力タイミングは、立ち上がりエッジセンス・立ち下がりエッジセンス・両エッジセンスが選択できます。

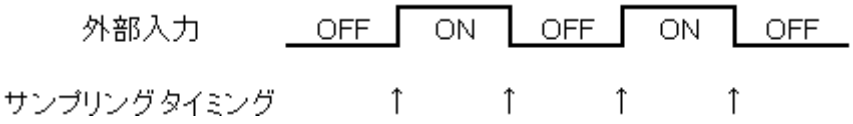
立ち上がりエッジセンス



立ち下がりエッジセンス



両エッジセンス



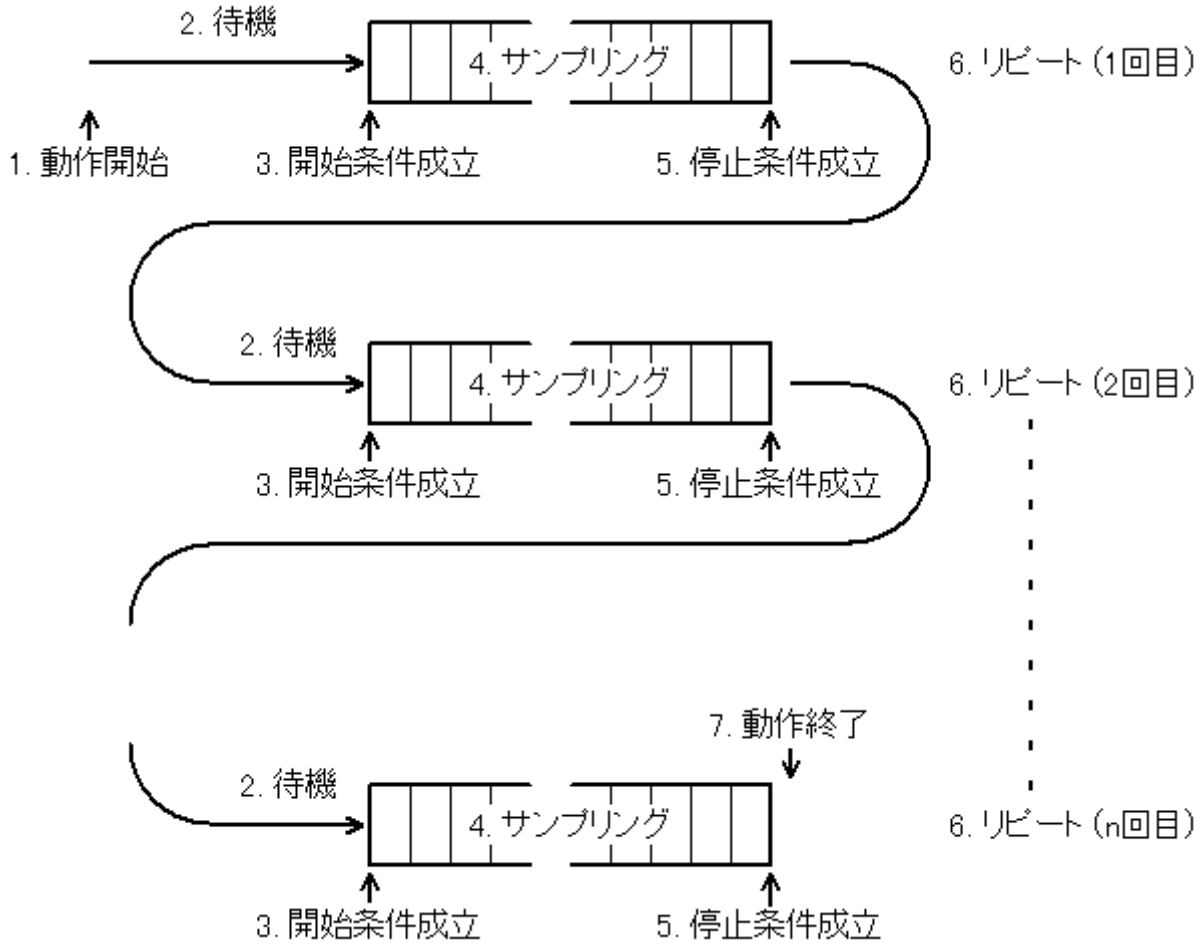
開始条件・停止条件・リピート

開始条件とは、サンプリングの開始タイミングを決定する条件です。
設定には YdxDiSetStartCondition関数 を使用します。

停止条件とは、サンプリングの停止タイミングを決定する条件です。
設定には YdxDiSetStopCondition関数 を使用します。

リピートとは、開始条件から停止条件までの動作を、繰り返しおこなう事です。
設定には YdxDiSetRepeat関数 を使用します。

動作の大まかな流れは以下のとおりです。



1. 以下の関数を使用して、動作を開始します。
   - YdxDiStart関数
2. 開始条件成立まで待機します。

   ただし、開始条件を「ソフトウェア（自動）」に設定した場合は、待機せずに4に進みます。
3. 開始条件の成立を検出します。
4. サンプリングをおこないます。
5. 停止条件の成立を検出します。
6. リピート設定回数分、2～5を繰り返します。

7. リピートが完了したら、動作を終了します。

外部トリガ

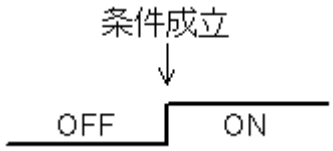
外部トリガとは、外部からのデジタル入力を サンプリング開始条件・停止条件 として使用する事です。
外部トリガとして使用するデジタル入力チャネルと動作モードの設定には、以下の関数を使用します。

- サンプリング開始条件
  - YdxDiSetStartExternal関数
- サンプリング停止条件
  - YdxDiSetStopExternal関数

動作モードは、立ち上がりエッジセンス・立ち下がりエッジセンス・両エッジセンス・ハイレベルセンス・ローレベルセンスから選択できます。

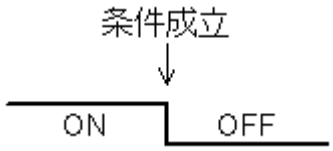
立ち上がりエッジセンス

OFF→ONに変化した時に、条件成立。



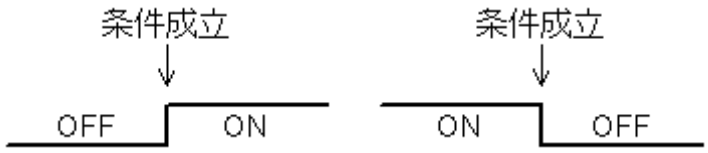
立ち下がりエッジセンス

ON→OFFに変化した時に、条件成立。



両エッジセンス

「ON→OFF」または「OFF→ON」に変化した時に、条件成立。



ハイレベルセンス

ONの時に、条件成立。
最初からONだった場合も、条件成立。
（最初、OFFだった場合は、立ち上がりエッジセンスと同じタイミング）

ローレベルセンス

OFFの時に、条件成立。
最初からOFFだった場合も、条件成立。
（最初、ONだった場合は、立ち下がりエッジセンスと同じタイミング）

サンプル数の監視

サンプル数を監視して、ステータスとして通知する事ができます。

データバッファのデータが、監視サンプル数以上になった場合、以下の動作となります。

- YdxDiGetStatus関数 で、ステータスを読み出した時、監視サンプル数ビットがオンになります。

監視サンプル数は、YdxDiSetCheckSampleNum関数 で設定します。

簡易デジタル出力

指定したチャネルのデジタル出力を１回おこなえます。
複数チャネルの指定も可能です。
設定も少なく、簡単に使用する事ができます。

以下の関数を使用します。

- YdxDoOutput関数
- YdxDoOutputBit関数

一方、連続サンプリングなど色々な条件でデジタル出力をおこないたい場合は 高機能デジタル出力 を使用します。

参考

- 実行手順（簡易デジタル出力）
- サンプルプログラム（DioBit）

高機能デジタル出力

連続サンプリングなど、色々な条件でデジタル出力がおこなえます。
設定項目は多くなりますが、連続サンプリングや、開始・停止・繰り返しなどのシーケンス制御をユニット側に任せられる為、パソコン側にあまり負担をかけずに高速なシステムが実現できます。
外部クロック・外部トリガ機能を使用して、外部との同期や連携も可能です。

一方、デジタル出力を１回おこないたい場合は 簡易デジタル出力 を使用します。

参考

- 実行手順（高機能デジタル出力）
- サンプルプログラム（DoPolling）

  高機能デジタル出力のサンプルプログラムです。
  デジタル出力を1000回おこないます。
  動作状態の監視をポーリングでおこなっています。

データバッファ

データバッファは、ユニット内部にあり、データを一時的に記憶します。
パソコンからはデータをまとめて設定する事が可能になるため効率的で、パソコン側の負荷を大幅に軽減する事が可能になります。

用途に応じて、FIFOバッファ形式とリングバッファ形式が選択できます。

- 形式の選択には、YdxDoSetBuffer関数 を使用します。
- データの設定には YdxDoSetData関数 を使用します。
- データをクリアするには YdxDoClearData関数 を使用します。

FIFOバッファ形式

出力は、先に設定したデータから順におこなわれます。
出力したデータは、バッファから破棄されます。
動作中 にデータを設定（追加）する事が可能です。

※ 動作中にデータを設定（追加）する事が可能ですので、データバッファが空にならないように定期的にデータを設定（追加）する事で容量以上の長時間のサンプリングが可能になります。

リングバッファ形式

データを最後まで出力すると、先頭に戻って繰り返し出力がおこなわれます。
出力したデータは、バッファから破棄されません。
動作中 にデータを設定する事はできません。

※ 波形を繰り返し出力する場合などに便利です。

サンプリングクロック

サンプリングクロックは、サンプリングのタイミングを決定します。
内部クロックと外部クロックが選択できます。
選択には以下の関数を使用します。

- YdxDoSetClock関数

内部クロック

ユニット内部でクロックを生成します。
クロック周期の設定には以下の関数を使用します。

- YdxDoSetClockInternal関数

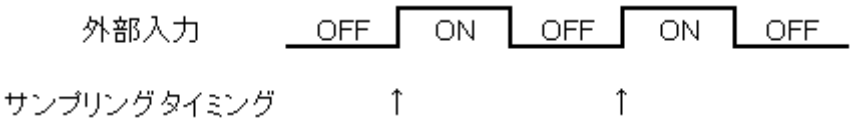
外部クロック

外部入力をクロックとして使用します。
外部クロックとして使用するデジタル入力チャネルと入力タイミングの設定には以下の関数を使用します。

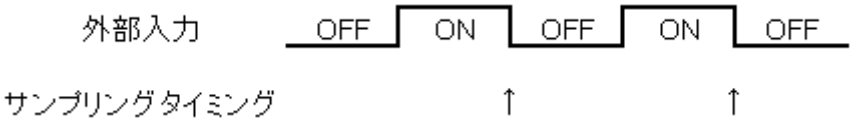
- YdxDoSetClockExternal関数

入力タイミングは、立ち上がりエッジセンス・立ち下がりエッジセンス・両エッジセンスが選択できます。

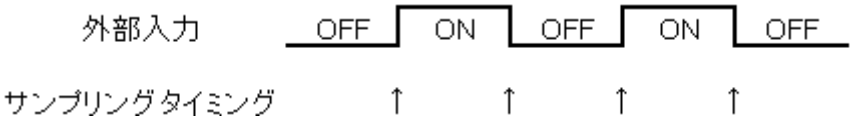
立ち上がりエッジセンス



立ち下がりエッジセンス



両エッジセンス



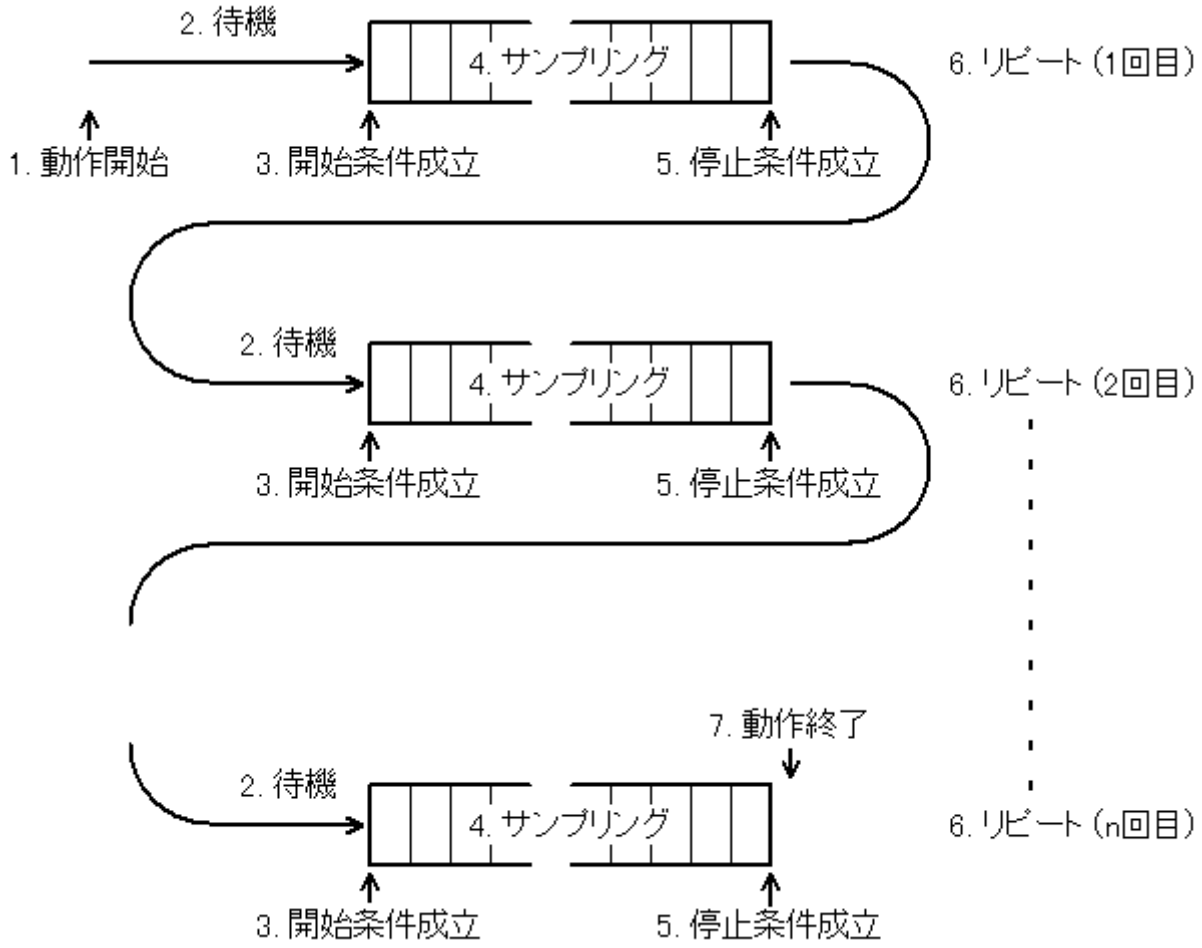
開始条件・停止条件・リピート

開始条件とは、サンプリングの開始タイミングを決定する条件です。
設定には YdxDoSetStartCondition関数 を使用します。

停止条件とは、サンプリングの停止タイミングを決定する条件です。
設定には YdxDoSetStopCondition関数 を使用します。

リピートとは、開始条件から停止条件までの動作を、繰り返しおこなう事です。
設定には YdxDoSetRepeat関数 を使用します。

動作の大まかな流れは以下のとおりです。



1. 以下の関数を使用して、動作を開始します。
   - YdxDoStart関数
2. 開始条件成立まで待機します。

   ただし、開始条件を「ソフトウェア（自動）」に設定した場合は、待機せずに4に進みます。
3. 開始条件の成立を検出します。
4. サンプリングをおこないます。
5. 停止条件の成立を検出します。
6. リピート設定回数分、2～5を繰り返します。

7. リピートが完了したら、動作を終了します。

外部トリガ

外部トリガとは、外部からのデジタル入力を サンプリング開始条件・停止条件 として使用する事です。

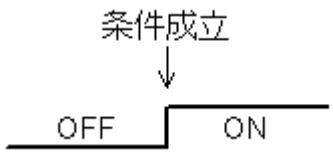
外部トリガとして使用するデジタル入力チャネルと動作モードの設定には、以下の関数を使用します。

- サンプリング開始条件
  - YdxDoSetStartExternal関数
- サンプリング停止条件
  - YdxDoSetStopExternal関数

動作モードは、立ち上がりエッジセンス・立ち下がりエッジセンス・両エッジセンス・ハイレベルセンス・ローレベルセンスから選択できます。

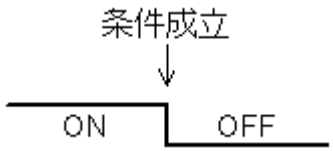
立ち上がりエッジセンス

OFF→ONに変化した時に、条件成立。



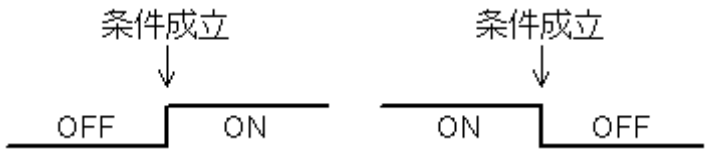
立ち下がりエッジセンス

ON→OFFに変化した時に、条件成立。



両エッジセンス

「ON→OFF」または「OFF→ON」に変化した時に、条件成立。



ハイレベルセンス

ONの時に、条件成立。
最初からONだった場合も、条件成立。
（最初、OFFだった場合は、立ち上がりエッジセンスと同じタイミング）

ローレベルセンス

OFFの時に、条件成立。
最初からOFFだった場合も、条件成立。
（最初、ONだった場合は、立ち下がりエッジセンスと同じタイミング）

サンプル数の監視

サンプル数を監視して、ステータスとして通知する事ができます。

未出力サンプル数（データバッファにデータが残っているサンプル数）が、監視サンプル数以下になった場合、以下の動作となります。

- YdxDoGetStatus関数 で、ステータスを読み出した時、監視サンプル数ビットがオンになります。

監視サンプル数は、YdxDoSetCheckSampleNum関数 で設定します。

基本仕様

接続台数

1台のパソコンから制御できるユニットの最大数は、各機種16台（DAQ-Xシリーズ全体で32台）です。

ハードウェア仕様

ハードウェア仕様については取扱説明書を参照してください。

注意事項

使用中にはパソコンがスリープ（スタンバイ）や休止状態とならないようにOSを設定してください。
スリープ（スタンバイ）や休止状態になると、パソコンとのUSB通信が出来なくなってしまう為、エラー停止します。

動作環境

パソコン

IBM PC/AT互換機（DOS/V機）

OS

Windows 11 x64
Windows 10 x86, x64
Windows 10 IoT Enterprise[1]
Windows 8.1 x86, x64
Windows 8 x86, x64
Windows 7 x86, x64
Windows Vista x86, x64[2]
~~Windows XP x64~~[3]
~~Windows XP~~[3]

対応言語

Microsoft Visual C++（6.0, .NET2002以降）
Microsoft Visual C#
Microsoft Visual Basic（6.0, .NET2002以降）
VBA[4]
Python3
その他、Win32API関数をサポートしているプログラミング言語

---

1. Windows 10 IoT Enterprise以外のWindows 10 IoTでは使用できません。 ↩
2. 2014年3月6日リリースの旧バージョンのドライバを使用します。 ↩
3. 使用できなくなりました。（使用するために必要なマイクロソフトの更新プログラムの配付が終了したため） ↩ ↩
4. VBAのサンプルはありません。VB6.0のサンプルコードを参考にしてください。 ↩

動作確認ユーティリティ

デジタル入力・デジタル出力の動作確認ができます。

使用手順

- オープン

  「識別スイッチ」「型名」を選択して「オープン」ボタンをクリックしてください。

- デジタル入力

  入力状態を100msec毎に表示します。
  入力がONの場合は緑、OFFの場合は灰色で表示します。

- デジタル出力

  出力ON/OFFを切り替えるには番号をクリックしてください。
  出力がONの場合は緑、OFFの場合は灰色で表示します。

備考

動作させる為には .NET Framework 4以降がインストールされている必要があります。

サンプルプログラム一覧

C#、Visual Basic（.NET2002以降）、Visual Basic 6.0、C++/CLI のサンプルプログラムが付属しています。

簡易デジタル入出力

サンプル名	動作概要
DioBit	簡易デジタル入出力をおこないます。

高機能デジタル入力

サンプル名	動作概要
DiPolling	高機能デジタル入力 のサンプルプログラムです。 デジタル入力を1000回おこない、データを表示します。 動作状態の監視をポーリングでおこなっています。
DiFile	高機能デジタル入力 のサンプルプログラムです。 デジタル入力を連続でおこない、ファイルに保存します。
DiChart	高機能デジタル入力 のサンプルプログラムです。 デジタル入力を連続でおこない、波形をグラフ表示します。

高機能デジタル出力

サンプル名	動作概要
DoPolling	高機能デジタル出力 のサンプルプログラムです。 デジタル出力を1000回おこないます。 動作状態の監視をポーリングでおこなっています。
DoFile	高機能デジタル出力 のサンプルプログラムです。 波形データをCSVファイルから読み出し、デジタル出力をおこないます。

備考

付属しているサンプルを、他のバージョンで使用する場合は、以下のようにしてください。

- Visual C# 2005以降

  Visual C# .NET2003のプロジェクトを変換して使用してください。

- Visual Basic 2005以降

Visual Basic .NET2003のプロジェクトを変換して使用してください。

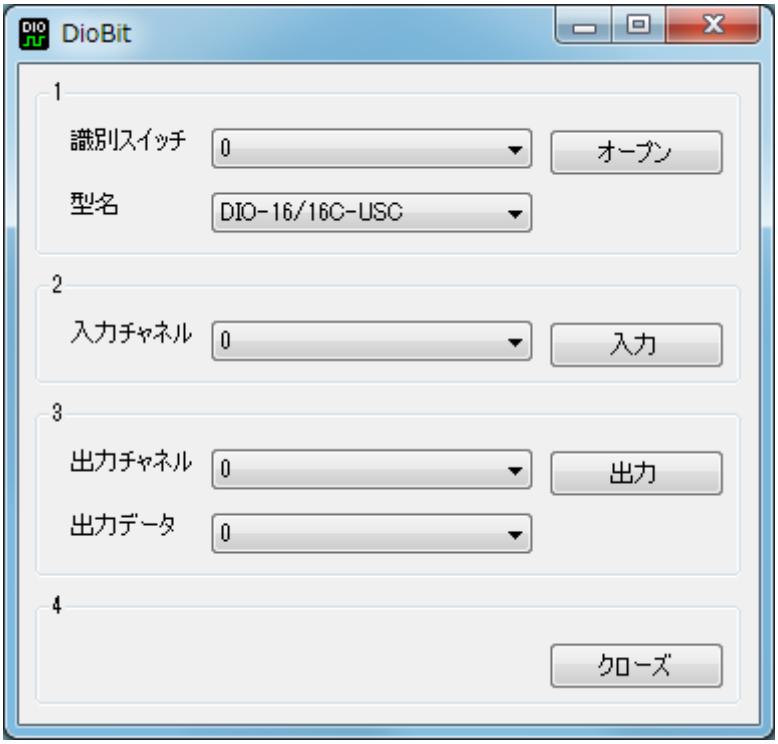
- Visual C++ 2008以降

  Visual C++ 2005のプロジェクトを変換して使用してください。

DioBit

デジタル入力・デジタル出力をビットごとにおこないます。

画面



1. オープン

   ユニットのオープンをします。

2. デジタル入力

   デジタル入力端子の状態を読み込み、表示します。

3. デジタル出力

   デジタル出力端子を制御します。

4. クローズ

   ユニットのクローズをします。
   オープンをした場合は、必ず実行する必要があります。

サンプルソース

- C#
- Visual Basic（.NET2002以降）
- Visual Basic 6.0
- C++/CLI

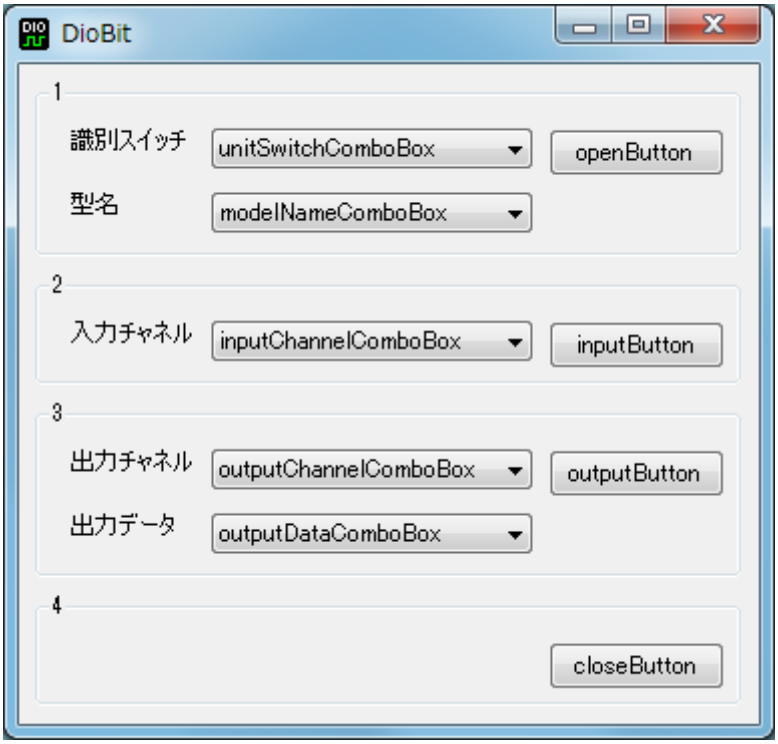
C#

開発環境の設定

1. Ydx.cs をプロジェクトフォルダにコピーします。
2. Ydx.cs をプロジェクトに追加します。
3. ソースファイルにusing ディレクティブを使ってYdxCsを宣言します。

```
using YdxCs;
```

コントロール



変数

```
private int id;
```

実行結果の表示

```
private void ResultShow(string title, int resultCode)
{
    string resultString;
    Ydx.CnvResultToString(resultCode, out resultString);
    switch (resultCode)
    {
        case 0:
        case Ydx.YDX_RESULT_DI_EXCEED_DATA_NUM:
        case Ydx.YDX_RESULT_DI_EXCEED_BUF_SIZ:
            MessageBox.Show(resultString, title, MessageBoxButtons.OK,
MessageBoxIcon.Asterisk);
```

```
            break;
        default:
            MessageBox.Show(resultString, title, MessageBoxButtons.OK, MessageBoxIcon.Hand);
            break;
    }
}
```

フォームロード

```
private void Form1_Load(object sender, EventArgs e)
{
    // ユニット識別スイッチ
    unitSwitchComboBox.ResetText();
    unitSwitchComboBox.Items.AddRange(new string[] { "0", "1", "2", "3", "4", "5", "6", "7",
"8", "9", "A", "B", "C", "D", "E", "F" });
    unitSwitchComboBox.SelectedIndex = 0;

    // 型名
    modelNameComboBox.ResetText();
    modelNameComboBox.Items.AddRange(new string[] { "DIO-16/16C-USC", "DIO-16/16D-UBC",
"DIO-16/16D-USC" });
    modelNameComboBox.SelectedIndex = 0;

    // 入力チャネル
    inputChannelComboBox.ResetText();
    inputChannelComboBox.Items.AddRange(new string[] { "0", "1", "2", "3", "4", "5", "6",
"7", "8", "9", "10", "11", "12", "13", "14", "15" });
    inputChannelComboBox.SelectedIndex = 0;

    // 出力チャネル
    outputChannelComboBox.ResetText();
    outputChannelComboBox.Items.AddRange(new string[] { "0", "1", "2", "3", "4", "5", "6",
"7", "8", "9", "10", "11", "12", "13", "14", "15" });
    outputChannelComboBox.SelectedIndex = 0;

    // 出力データ
    outputDataComboBox.ResetText();
    outputDataComboBox.Items.AddRange(new string[] { "0", "1" });
    outputDataComboBox.SelectedIndex = 0;
}
```

オープン

```
private void openButton_Click(object sender, EventArgs e)
{
    int result = Ydx.Open(unitSwitchComboBox.SelectedIndex, modelNameComboBox.Text, 0, out
id);
    if(result != 0)
        ResultShow("YdxOpen", result);
    else
    {
        unitSwitchComboBox.Enabled = false;
        modelNameComboBox.Enabled = false;
        ResultShow("オープン", result);
    }
}
```

入力

```
private void inputButton_Click(object sender, EventArgs e)
{
    int[] data = new int[1];
    int result = Ydx.DiInputBit(id, inputChannelComboBox.SelectedIndex, 1, 0, data);
    if(result != 0)
        ResultShow("YdxDiInputBit", result);
    else
        MessageBox.Show("データ : " + data[0].ToString(), "入力", MessageBoxButtons.OK,
MessageBoxIcon.Information);
}
```

出力

```
private void outputButton_Click(object sender, EventArgs e)
{
    int[] data = new int[1];
    data[0] = outputDataComboBox.SelectedIndex;
    int result = Ydx.DoOutputBit(id, outputChannelComboBox.SelectedIndex, 1, data);
    ResultShow("出力", result);
}
```

クローズ

```
private void closeButton_Click(object sender, EventArgs e)
{
    unitSwitchComboBox.Enabled = true;
    modelNameComboBox.Enabled = true;
    int result = Ydx.Close(id);
    if(result != 0)
        ResultShow("YdxClose", result);
    else
        ResultShow("クローズ", result);
}
```

フォームクローズ

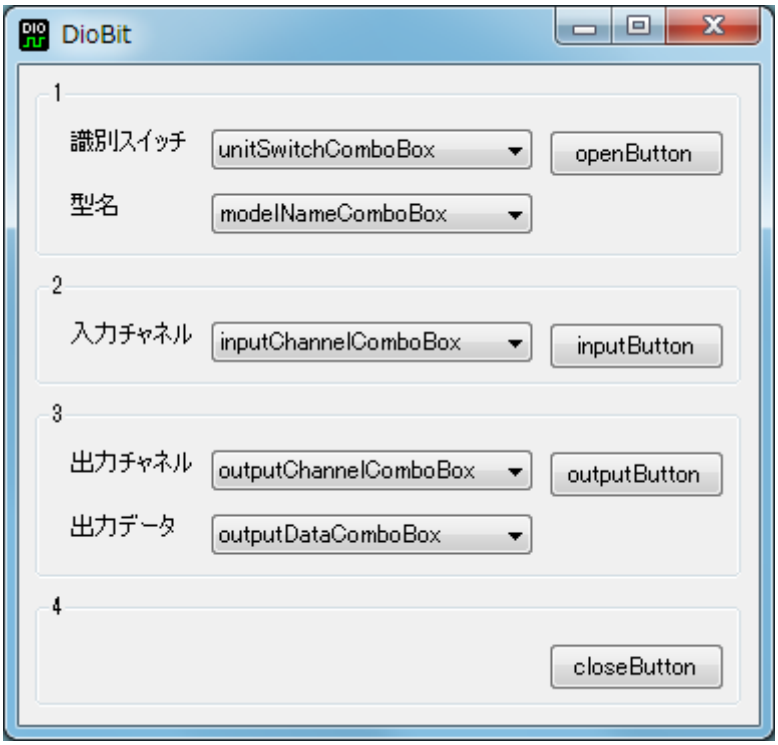
```
private void Form1_Closing(object sender, System.ComponentModel.CancelEventArgs e)
{
    int result = Ydx.Close(id);
    if((result != 0) && (result != Ydx.YDX_RESULT_NOT_OPEN))
    {
        ResultShow("YdxClose", result);
    }
}
```

Visual Basic（.NET2002以降）

開発環境の設定

1. Ydx.vb をプロジェクトフォルダにコピーします。
2. Ydx.vb をプロジェクトに追加します。

コントロール



変数

```
Dim id As Integer
Dim result As Integer
```

実行結果の表示

```
Private Sub ResultShow(ByVal title As String, ByVal resultCode As Integer)
    Dim resultString As New StringBuilder(256)
    YdxCnvResultToString(resultCode, resultString)
    Select Case resultCode
        Case 0, Ydx.YDX_RESULT_DI_EXCEED_DATA_NUM, Ydx.YDX_RESULT_DI_EXCEED_BUF_SIZ
            MessageBox.Show(resultString.ToString(), title, MessageBoxButtons.OK,
MessageBoxIcon.Asterisk)
        Case Else
            MessageBox.Show(resultString.ToString(), title, MessageBoxButtons.OK,
MessageBoxIcon.Hand)
    End Select
End Sub
```

フォームロード

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
    ' ユニット識別スイッチ
    unitSwitchComboBox.ResetText()
    unitSwitchComboBox.Items.AddRange(New String() { "0", "1", "2", "3", "4", "5", "6", "7",
"8", "9", "A", "B", "C", "D", "E", "F" })
    unitSwitchComboBox.SelectedIndex = 0

    ' 型名
    modelNameComboBox.ResetText()
    modelNameComboBox.Items.AddRange(New String() { "DIO-16/16C-USC", "DIO-16/16D-UBC",
"DIO-16/16D-USC" })
    modelNameComboBox.SelectedIndex = 0

    ' 入力チャネル
    inputChannelComboBox.ResetText()
    inputChannelComboBox.Items.AddRange(New String() { "0", "1", "2", "3", "4", "5", "6",
"7", "8", "9", "10", "11", "12", "13", "14", "15" })
    inputChannelComboBox.SelectedIndex = 0

    ' 出力チャネル
    outputChannelComboBox.ResetText()
    outputChannelComboBox.Items.AddRange(New String() { "0", "1", "2", "3", "4", "5", "6",
"7", "8", "9", "10", "11", "12", "13", "14", "15" })
    outputChannelComboBox.SelectedIndex = 0

    ' 出力データ
    outputDataComboBox.ResetText()
    outputDataComboBox.Items.AddRange(New String() { "0", "1" })
    outputDataComboBox.SelectedIndex = 0
End Sub
```

オープン

```
Private Sub openButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles openButton.Click
    result = YdxOpen(unitSwitchComboBox.SelectedIndex, modelNameComboBox.Text, 0, id)
    If result <> 0 Then
        ResultShow("YdxOpen", result)
    Else
        unitSwitchComboBox.Enabled = False
        modelNameComboBox.Enabled = False
        ResultShow("オープン", result)
    End If
End Sub
```

入力

```
Private Sub inputButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles inputButton.Click
    Dim data(0) As Integer
    result = YdxDiInputBit(id, inputChannelComboBox.SelectedIndex, 1, 0, data)
    If result <> 0 Then
        ResultShow("YdxDiInputBit", result)
    Else
        MessageBox.Show("データ : " & data(0).ToString(), "入力", MessageBoxButtons.OK,
MessageBoxIcon.Information)
```

```
    End If
End Sub
```

出力

```
Private Sub outputButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles outputButton.Click
    Dim data(0) As Integer
    data(0) = outputDataComboBox.SelectedIndex
    result = YdxDoOutputBit(id, outputChannelComboBox.SelectedIndex, 1, data)
    ResultShow("出力", result)
End Sub
```

クローズ

```
Private Sub closeButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles closeButton.Click
    unitSwitchComboBox.Enabled = True
    modelNameComboBox.Enabled = True
    result = YdxClose(id)
    If result <> 0 Then
        ResultShow("YdxClose", result)
    Else
        ResultShow("クローズ", result)
    End If
End Sub
```

フォームクローズ

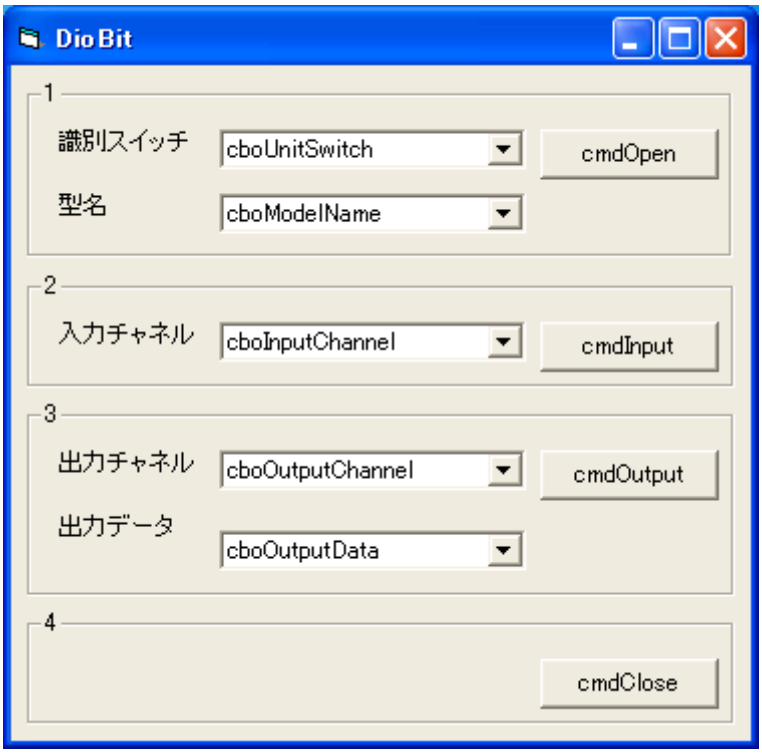
```
Private Sub Form1_Closing(ByVal sender As Object, ByVal e As
System.ComponentModel.CancelEventArgs) Handles MyBase.Closing
    result = YdxClose(id)
    If result <> 0 And result <> YDX_RESULT_NOT_OPEN Then
        ResultShow("YdxClose", result)
    End If
End Sub
```

Visual Basic 6.0

開発環境の設定

1. Ydx.bas をプロジェクトフォルダにコピーします。
2. Ydx.bas をプロジェクトに追加します。

コントロール



変数

```
Dim id As Long
Dim result As Long
```

実行結果の表示

```
Private Sub ResultShow(ByVal title As String, ByVal resultCode As Long)
    Dim resultString As String
    Call YdxCnvResultToString(resultCode, resultString)
    Select Case resultCode
        Case 0, Ydx.YDX_RESULT_DI_EXCEED_DATA_NUM, Ydx.YDX_RESULT_DI_EXCEED_BUF_SIZ
            MsgBox resultString, vbInformation, title
        Case Else
            MsgBox resultString, vbCritical, title
    End Select
End Sub
```

フォームロード

```
Private Sub Form_Load()
    ' ユニット識別スイッチ
    cboUnitSwitch.AddItem "0"
    cboUnitSwitch.AddItem "1"
    cboUnitSwitch.AddItem "2"
    cboUnitSwitch.AddItem "3"
    cboUnitSwitch.AddItem "4"
    cboUnitSwitch.AddItem "5"
    cboUnitSwitch.AddItem "6"
    cboUnitSwitch.AddItem "7"
    cboUnitSwitch.AddItem "8"
    cboUnitSwitch.AddItem "9"
    cboUnitSwitch.AddItem "A"
    cboUnitSwitch.AddItem "B"
    cboUnitSwitch.AddItem "C"
    cboUnitSwitch.AddItem "D"
    cboUnitSwitch.AddItem "E"
    cboUnitSwitch.AddItem "F"
    cboUnitSwitch.ListIndex = 0

    ' 型名
    cboModelName.AddItem "DIO-16/16C-USC"
    cboModelName.AddItem "DIO-16/16D-UBC"
    cboModelName.AddItem "DIO-16/16D-USC"
    cboModelName.ListIndex = 0

    ' 入力チャネル
    cboInputChannel.AddItem "0"
    cboInputChannel.AddItem "1"
    cboInputChannel.AddItem "2"
    cboInputChannel.AddItem "3"
    cboInputChannel.AddItem "4"
    cboInputChannel.AddItem "5"
    cboInputChannel.AddItem "6"
    cboInputChannel.AddItem "7"
    cboInputChannel.AddItem "8"
    cboInputChannel.AddItem "9"
    cboInputChannel.AddItem "10"
    cboInputChannel.AddItem "11"
    cboInputChannel.AddItem "12"
    cboInputChannel.AddItem "13"
    cboInputChannel.AddItem "14"
    cboInputChannel.AddItem "15"
    cboInputChannel.ListIndex = 0

    ' 出力チャネル
    cboOutputChannel.AddItem "0"
    cboOutputChannel.AddItem "1"
    cboOutputChannel.AddItem "2"
    cboOutputChannel.AddItem "3"
    cboOutputChannel.AddItem "4"
    cboOutputChannel.AddItem "5"
    cboOutputChannel.AddItem "6"
    cboOutputChannel.AddItem "7"
    cboOutputChannel.AddItem "8"
    cboOutputChannel.AddItem "9"
    cboOutputChannel.AddItem "10"
    cboOutputChannel.AddItem "11"
    cboOutputChannel.AddItem "12"
    cboOutputChannel.AddItem "13"
    cboOutputChannel.AddItem "14"
    cboOutputChannel.AddItem "15"
    cboOutputChannel.ListIndex = 0
```

```
    ' 出力データ
    cboOutputData.AddItem "0"
    cboOutputData.AddItem "1"
    cboOutputData.ListIndex = 0
End Sub
```

オープン

```
Private Sub cmdOpen_Click()
    result = YdxOpen(cboUnitSwitch.ListIndex, cboModelName.Text, 0, id)
    If result <> 0 Then
        Call ResultShow("YdxOpen", result)
    Else
        cboUnitSwitch.Enabled = False
        cboModelName.Enabled = False
        Call ResultShow("オープン", result)
    End If
End Sub
```

入力

```
Private Sub cmdInput_Click()
    Dim data(0) As Long
    result = YdxDiInputBit(id, cboInputChannel.ListIndex, 1, 0, data(0))
    If result <> 0 Then
        Call ResultShow("YdxDiInputBit", result)
    Else
        MsgBox "データ : " & data(0), vbInformation, "入力"
    End If
End Sub
```

出力

```
Private Sub cmdOutput_Click()
    Dim data(0) As Long
    data(0) = cboOutputData.ListIndex
    result = YdxDoOutputBit(id, cboOutputChannel.ListIndex, 1, data(0))
    Call ResultShow("出力", result)
End Sub
```

クローズ

```
Private Sub cmdClose_Click()
    cboUnitSwitch.Enabled = True
    cboModelName.Enabled = True
    result = YdxClose(id)
    If result <> 0 Then
        Call ResultShow("YdxClose", result)
    Else
        Call ResultShow("クローズ", result)
    End If
End Sub
```

フォームアンロード

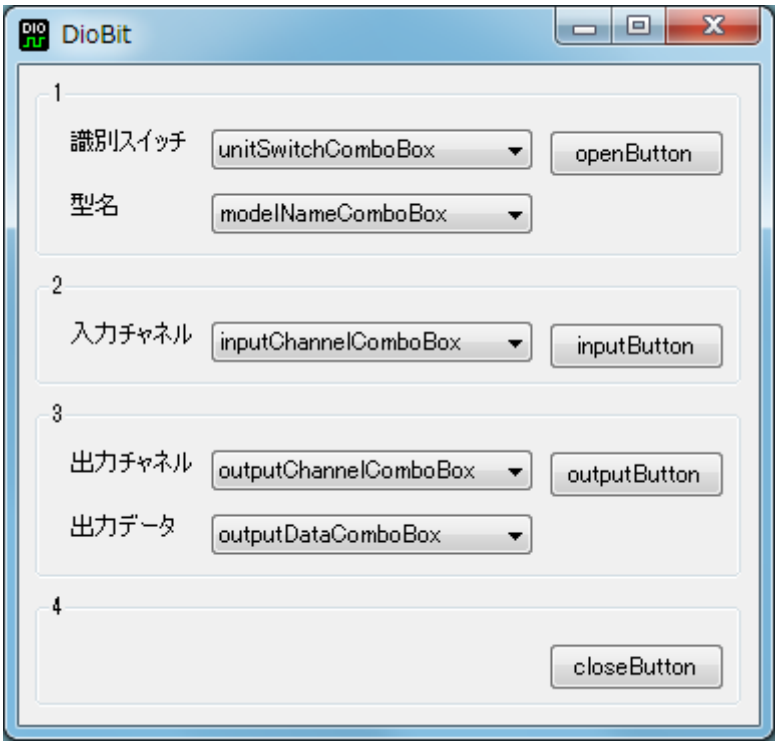
```
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    result = YdxClose(id)
    If result <> 0 And result <> YDX_RESULT_NOT_OPEN Then
        Call ResultShow("YdxClose", result)
    End If
End Sub
```

C++/CLI

開発環境の設定

1. YdxCLI.h をプロジェクトフォルダにコピーします。
2. YdxCLI.h をプロジェクトに追加します。
3. ソースファイルに YdxCLI.h をインクルードします。
4. usingディレクティブを使ってYdxCLIを宣言します。

コントロール



変数

```
int id;
```

実行結果の表示

```
private: System::Void ResultShow(String^ title, int resultCode)
{
    StringBuilder ^resultString = gcnew StringBuilder(256);
    YdxCnvResultToString(resultCode, resultString);
    switch (resultCode)
    {
    case 0:
    case YDX_RESULT_DI_EXCEED_DATA_NUM:
    case YDX_RESULT_DI_EXCEED_BUF_SIZ:
        MessageBox::Show(resultString->ToString(), title, MessageBoxButtons::OK,
MessageBoxIcon::Asterisk);
        break;
```

```
    default:
        MessageBox::Show(resultString->ToString(), title, MessageBoxButtons::OK,
MessageBoxIcon::Hand);
        break;
    }
}
```

フォームロード

```
private: System::Void Form1_Load(System::Object^ sender, System::EventArgs^ e)
{
    // ユニット識別スイッチ
    unitSwitchComboBox->ResetText();
    unitSwitchComboBox->Items->AddRange(gcnew array<String^> { "0", "1", "2", "3", "4", "5",
"6", "7", "8", "9", "A", "B", "C", "D", "E", "F" });
    unitSwitchComboBox->SelectedIndex = 0;

    // 型名
    modelNameComboBox->ResetText();
    modelNameComboBox->Items->AddRange(gcnew array<String^> { "DIO-16/16C-USC", "DIO-16/16D-
UBC", "DIO-16/16D-USC" });
    modelNameComboBox->SelectedIndex = 0;

    // 入力チャネル
    inputChannelComboBox->ResetText();
    inputChannelComboBox->Items->AddRange(gcnew array<String ^>{"0", "1", "2", "3", "4",
"5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15"});
    inputChannelComboBox->SelectedIndex = 0;

    // 出力チャネル
    outputChannelComboBox->ResetText();
    outputChannelComboBox->Items->AddRange(gcnew array<String ^>{"0", "1", "2", "3", "4",
"5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15"});
    outputChannelComboBox->SelectedIndex = 0;

    // 出力データ
    outputDataComboBox->ResetText();
    outputDataComboBox->Items->AddRange(gcnew array<String ^>{"0", "1"});
    outputDataComboBox->SelectedIndex = 0;
}
```

オープン

```
private: System::Void openButton_Click(System::Object^ sender, System::EventArgs^ e)
{
    int getId;
    int result = YdxOpen(unitSwitchComboBox->SelectedIndex, modelNameComboBox->Text, 0,
&getId);
    if(result != 0)
        ResultShow("YdxOpen", result);
    else
    {
        unitSwitchComboBox->Enabled = false;
        modelNameComboBox->Enabled = false;
        ResultShow("オープン", result);
        id = getId;
    }
}
```

デジタル入力

```
private: System::Void inputButton_Click(System::Object^ sender, System::EventArgs^ e)
{
    int data[1];
    int result = YdxDiInputBit(id, inputChannelComboBox->SelectedIndex, 1, 0, data);
    if(result != 0)
        ResultShow("YdxDiInputBit", result);
    else
        MessageBox::Show("データ : " + data[0].ToString(), "入力", MessageBoxButtons::OK,
MessageBoxIcon::Information);
}
```

デジタル出力

```
private: System::Void outputButton_Click(System::Object^ sender, System::EventArgs^ e)
{
    int data[1];
    data[0] = outputDataComboBox->SelectedIndex;
    int result = YdxDoOutputBit(id, outputChannelComboBox->SelectedIndex, 1, data);
    ResultShow("出力", result);
}
```

クローズ

```
private: System::Void closeButton_Click(System::Object^ sender, System::EventArgs^ e)
{
    unitSwitchComboBox->Enabled = true;
    modelNameComboBox->Enabled = true;
    int result = YdxClose(id);
    if(result != 0)
        ResultShow("YdxClose", result);
    else
        ResultShow("クローズ", result);
}
```

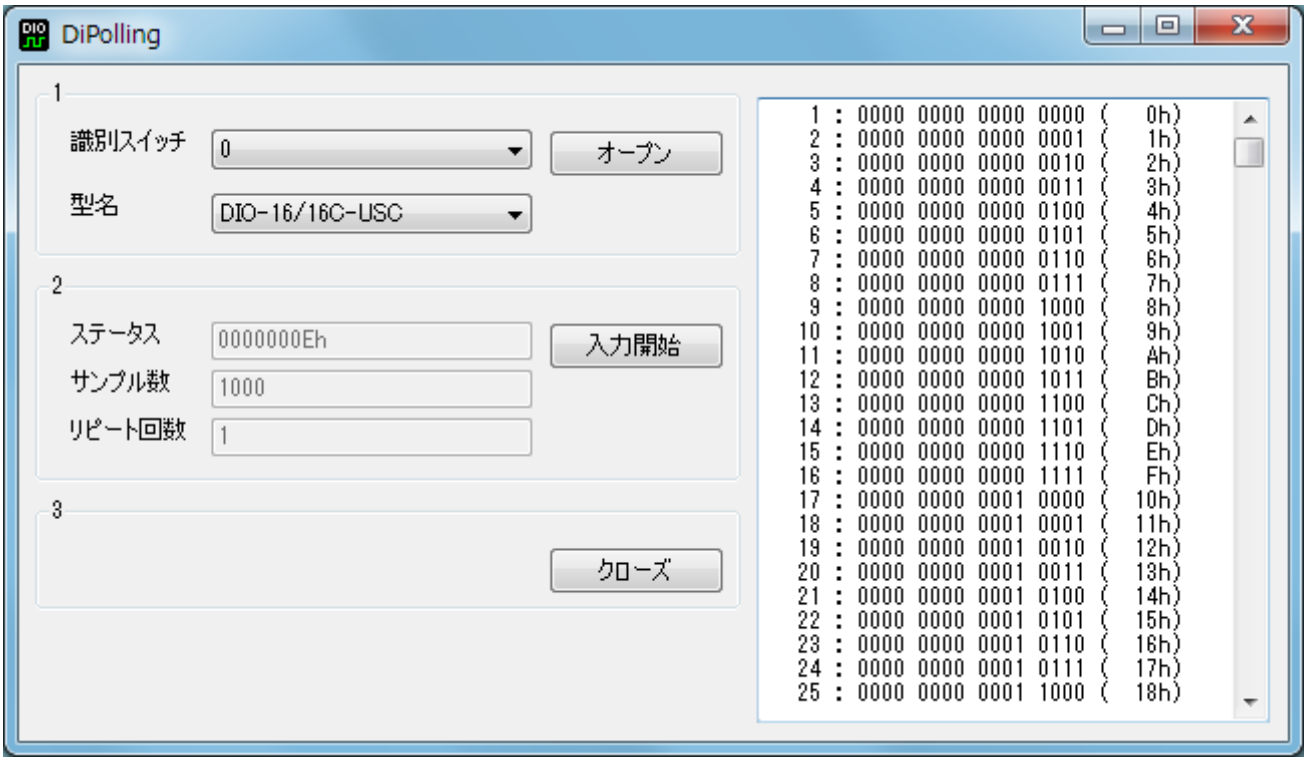
フォームクローズ

```
private: System::Void Form1_FormClosing(System::Object^ sender,
System::Windows::Forms::FormClosingEventArgs^ e)
{
    int result = YdxClose(id);
    if((result != 0) && (result != YDX_RESULT_NOT_OPEN))
    {
        ResultShow("YdxClose", result);
    }
}
```

DiPolling

高機能デジタル入力 のサンプルプログラムです。
デジタル入力を1000回おこない表示します。
動作状態の監視をポーリングでおこなっています。

画面



1. オープン

   ユニットのオープンをします。

2. 入力開始

   デジタル入力動作を開始します。
   「動作条件の設定」→「動作開始」→「状態監視（動作が停止するまで）」という手順が実行されます。

   周期 1msecで1000回サンプリングします。
   動作が停止すると、データを読み出して、表示します。

3. クローズ

   ユニットのクローズをします。
   オープンをした場合は、必ず実行する必要があります。

サンプルソース

- C#
- Visual Basic（.NET2002以降）
- Visual Basic 6.0
- C++/CLI

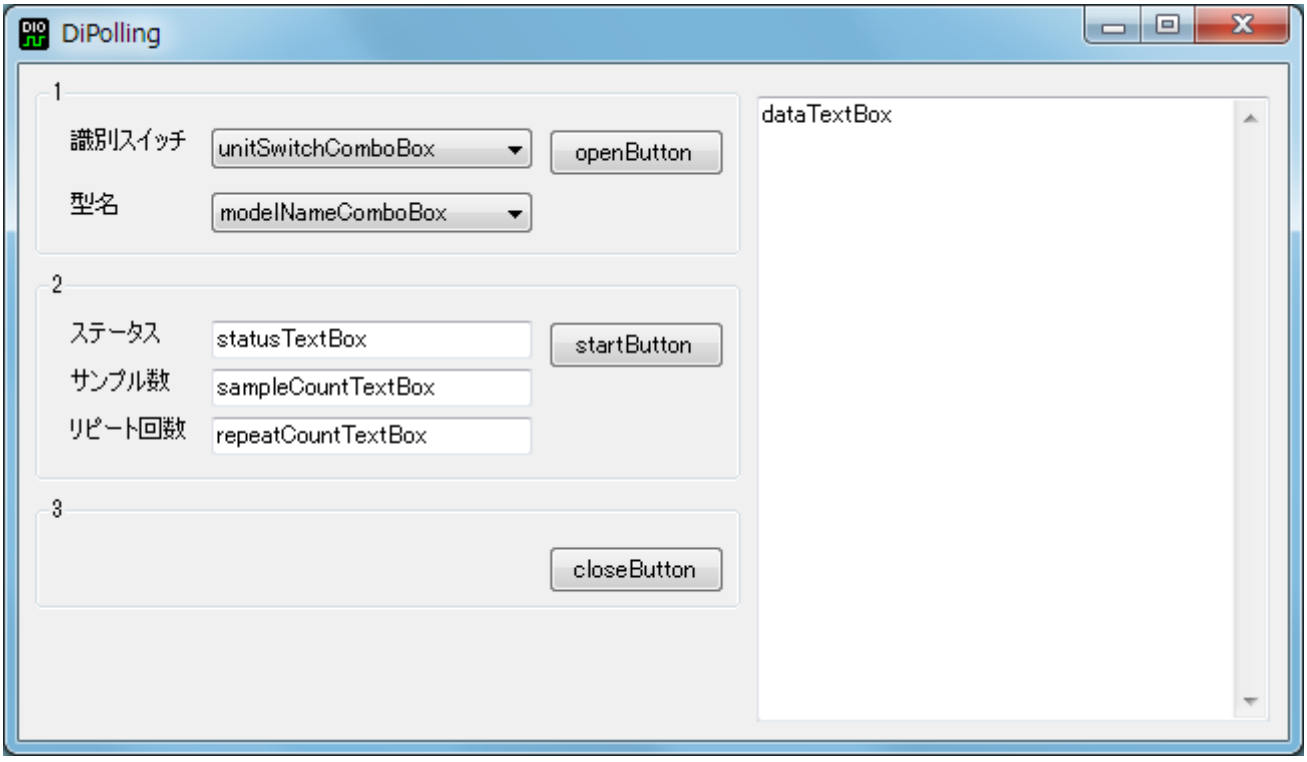
C#

開発環境の設定

1. Ydx.cs をプロジェクトフォルダにコピーします。
2. Ydx.cs をプロジェクトに追加します。
3. ソースファイルにusing ディレクティブを使ってYdxCsを宣言します。

```
using YdxCs;
```

コントロール



変数

```
private int id;
```

実行結果の表示

```
private void ResultShow(string title, int resultCode)
{
    string resultString;
    Ydx.CnvResultToString(resultCode, out resultString);
    switch (resultCode)
    {
        case 0:
        case Ydx.YDX_RESULT_DI_EXCEED_DATA_NUM:
        case Ydx.YDX_RESULT_DI_EXCEED_BUF_SIZ:
            MessageBox.Show(resultString, title, MessageBoxButtons.OK,
MessageBoxIcon.Asterisk);
```

```
            break;
        default:
            MessageBox.Show(resultString, title, MessageBoxButtons.OK, MessageBoxIcon.Hand);
            break;
    }
}
```

フォームロード

```
private void Form1_Load(object sender, EventArgs e)
{
    // ユニット識別スイッチ
    unitSwitchComboBox.ResetText();
    unitSwitchComboBox.Items.AddRange(new string[] { "0", "1", "2", "3", "4", "5", "6", "7",
"8", "9", "A", "B", "C", "D", "E", "F" });
    unitSwitchComboBox.SelectedIndex = 0;

    // 型名
    modelNameComboBox.ResetText();
    modelNameComboBox.Items.AddRange(new string[] { "DIO-16/16C-USC", "DIO-16/16D-UBC",
"DIO-16/16D-USC" });
    modelNameComboBox.SelectedIndex = 0;
}
```

オープン

```
private void openButton_Click(object sender, EventArgs e)
{
    int result = Ydx.Open(unitSwitchComboBox.SelectedIndex, modelNameComboBox.Text, 0, out
id);
    if(result != 0)
        ResultShow("YdxOpen", result);
    else
    {
        unitSwitchComboBox.Enabled = false;
        modelNameComboBox.Enabled = false;
        ResultShow("オープン", result);
    }
}
```

入力開始

```
private void startButton_Click(object sender, EventArgs e)
{
    dataTextBox.ResetText();
    Application.DoEvents();

    // データバッファの設定
    int result = Ydx.DiSetBuffer(id, 0);

    // FIFOバッファ
    if(result != 0)
    {
        ResultShow("YdxDiSetBuffer", result);
        return;
    }

    // サンプリングクロックの設定
```

```
    result = Ydx.DiSetClock(id, 0); // 内部クロック
    if(result != 0)
    {
        ResultShow("YdxDiSetClock", result);
        return;
    }

    // 内部クロック周期の設定
    result = Ydx.DiSetClockInternal(id, 1000); // 1000μsec
    if(result != 0)
    {
        ResultShow("YdxDiSetClockInternal", result);
        return;
    }

    // サンプリング開始条件の設定
    result = Ydx.DiSetStartCondition(id, 0, 0); // ソフトウェア  if(result != 0)
    {
        ResultShow("YdxDiSetStartCondition", result);
        return;
    }

    // サンプリング停止条件の設定
    result = Ydx.DiSetStopCondition(id, 0, 0); // サンプル数  if(result != 0)
    {
        ResultShow("YdxDiSetStopCondition", result);
        return;
    }

    // サンプリング停止条件(サンプル数)の設定
    result = Ydx.DiSetStopSampleNum(id, 1000);
    if(result != 0)
    {
        ResultShow("YdxDiSetStopSampleNum", result);
        return;
    }

    // データをクリア
    result = Ydx.DiClearData(id);
    if(result != 0)
    {
        ResultShow("YdxDiClearData", result);
        return;
    }

    // デジタル入力動作を開始
    result = Ydx.DiStart(id);
    if(result != 0)
    {
        ResultShow("YdxDiStart", result);
        return;
    }

    // 動作終了待ち
    int status, sampleCount, repeatCount;
    // 動作中ステータスがOFFになるまでポーリング
    do
    {
        // ステータスの取得
        result = Ydx.DiGetStatus(id, out status, out sampleCount, out repeatCount);
        if(result != 0)
        {
            ResultShow("YdxDiGetStatus", result);
            return;
```

```
        }
        statusTextBox.Text = status.ToString("X").PadLeft(8, '0') + "h";
        sampleCountTextBox.Text = sampleCount.ToString();
        repeatCountTextBox.Text = repeatCount.ToString();
        Application.DoEvents();

        if((status & Ydx.YDX_STATUS_COMMUNICATE_ERR) != 0)
        {
            MessageBox.Show("通信エラーが発生しました", "", MessageBoxButtons.OK,
MessageBoxIcon.Hand);
            return;
        }

        if((status & Ydx.YDX_STATUS_HARDWARE_ERR) != 0)
        {
            MessageBox.Show("ハードウェアエラーが発生しました", "", MessageBoxButtons.OK,
MessageBoxIcon.Hand);
            return;
        }

        if((status & Ydx.YDX_STATUS_OVERRUN_ERR) != 0)
        {
            MessageBox.Show("オーバランエラーが発生しました", "", MessageBoxButtons.OK,
MessageBoxIcon.Hand);
            return;
        }

        if((status & Ydx.YDX_STATUS_SAMPLE_CLOCK_ERR) != 0)
        {
            MessageBox.Show("サンプリングクロックエラーが発生しました", "", MessageBoxButtons.OK,
MessageBoxIcon.Hand);
            return;
        }
    } while ((status & Ydx.YDX_STATUS_BUSY) != 0);

    // データの読み出し
    int[] data = new int[sampleCount];
    result = Ydx.DiGetData(id, ref sampleCount, data);
    if(result != 0)
    {
        ResultShow("YdxDiGetData", result);
        if((result != Ydx.YDX_RESULT_DI_EXCEED_DATA_NUM) && (result !=
Ydx.YDX_RESULT_DI_EXCEED_BUF_SIZ))
            return;
    }

    // 表示
    string txt = "";
    for (int sampleIndex = 0; sampleIndex < sampleCount; sampleIndex++)
    {
        txt += (sampleIndex + 1).ToString().PadLeft(5) + " : ";

        // 2進数表記
        txt += Convert.ToString(data[sampleIndex] >> 16 & 0x0f, 2).PadLeft(4, '0') + " ";
        txt += Convert.ToString(data[sampleIndex] >> 8 & 0x0f, 2).PadLeft(4, '0') + " ";
        txt += Convert.ToString(data[sampleIndex] >> 4 & 0x0f, 2).PadLeft(4, '0') + " ";
        txt += Convert.ToString(data[sampleIndex] & 0x0f, 2).PadLeft(4, '0');

        // 16進数表記
        txt += " (" + data[sampleIndex].ToString("X").PadLeft(4) + "h)";
        txt += Environment.NewLine;
    }
```

```
    dataTextBox.Text = txt;
}
```

クローズ

```
private void closeButton_Click(object sender, EventArgs e)
{
    unitSwitchComboBox.Enabled = true;
    modelNameComboBox.Enabled = true;
    int result = Ydx.Close(id);
    if(result != 0)
        ResultShow("YdxClose", result);
    else
        ResultShow("クローズ", result);
}
```

フォームクローズ

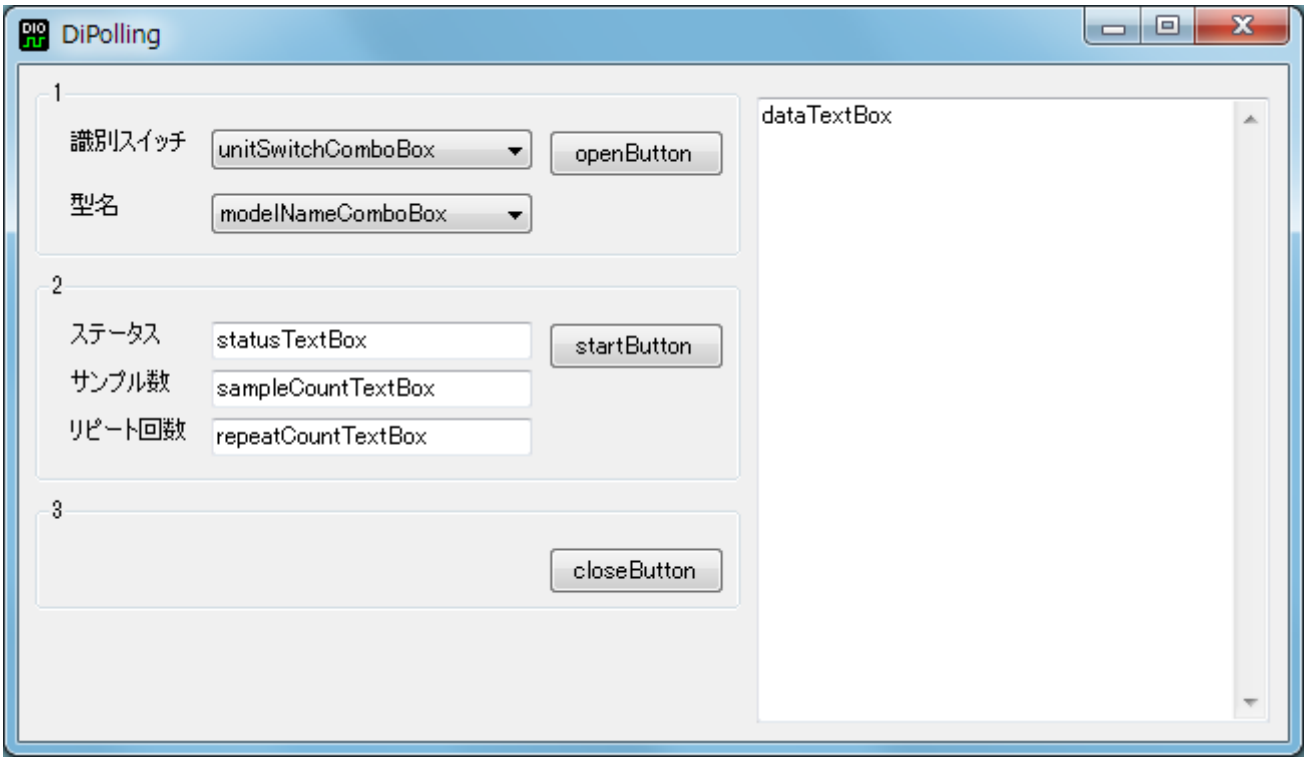
```
private void Form1_Closing(object sender, System.ComponentModel.CancelEventArgs e)
{
    int result = Ydx.Close(id);
    if((result != 0) && (result != Ydx.YDX_RESULT_NOT_OPEN))
        ResultShow("YdxClose", result);
}
```

Visual Basic（.NET2002以降）

開発環境の設定

1. Ydx.vb をプロジェクトフォルダにコピーします。
2. Ydx.vb をプロジェクトに追加します。

コントロール



変数

```
Dim id As Integer
Dim result As Integer
```

実行結果の表示

```
Private Sub ResultShow(ByVal title As String, ByVal resultCode As Integer)
    Dim resultString As New StringBuilder(256)
    YdxCnvResultToString(resultCode, resultString)
    Select Case resultCode
        Case 0, Ydx.YDX_RESULT_DI_EXCEED_DATA_NUM, Ydx.YDX_RESULT_DI_EXCEED_BUF_SIZ
            MessageBox.Show(resultString.ToString(), title, MessageBoxButtons.OK,
MessageBoxIcon.Asterisk)
        Case Else
            MessageBox.Show(resultString.ToString(), title, MessageBoxButtons.OK,
MessageBoxIcon.Hand)
    End Select
End Sub
```

フォームロード

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
    ' ユニット識別スイッチ
    unitSwitchComboBox.ResetText()
    unitSwitchComboBox.Items.AddRange(New String() { "0", "1", "2", "3", "4", "5", "6", "7",
"8", "9", "A", "B", "C", "D", "E", "F" })
    unitSwitchComboBox.SelectedIndex = 0

    ' 型名
    modelNameComboBox.ResetText()
    modelNameComboBox.Items.AddRange(New String() { "DIO-16/16C-USC", "DIO-16/16D-UBC",
"DIO-16/16D-USC" })
    modelNameComboBox.SelectedIndex = 0
End Sub
```

オープン

```
Private Sub openButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles openButton.Click
    result = YdxOpen(unitSwitchComboBox.SelectedIndex, modelNameComboBox.Text, 0, id)
    If result <> 0 Then
        ResultShow("YdxOpen", result)
    Else
        unitSwitchComboBox.Enabled = False
        modelNameComboBox.Enabled = False
        ResultShow("オープン", result)
    End If
End Sub
```

入力開始

```
Private Sub startButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles startButton.Click
    dataTextBox.ResetText()
    Application.DoEvents()

    ' データバッファの設定
    result = YdxDiSetBuffer(id, 0)    ' FIFOバッファ
    If result <> 0 Then
        ResultShow("YdxDiSetBuffer", result)
        Exit Sub
    End If

    ' サンプリングクロックの設定
    result = YdxDiSetClock(id, 0)     ' 内部クロック
    If result <> 0 Then
        ResultShow("YdxDiSetClock", result)
        Exit Sub
    End If

    ' 内部クロック周期の設定
    result = YdxDiSetClockInternal(id, 1000) ' 1000μsec
    If result <> 0 Then
        ResultShow("YdxDiSetClockInternal", result)
        Exit Sub
    End If
```

```
    ' サンプリング開始条件の設定
    result = YdxDiSetStartCondition(id, 0, 0)    ' ソフトウェア
    If result <> 0 Then
        ResultShow("YdxDiSetStartCondition", result)
        Exit Sub
    End If

    ' サンプリング停止条件の設定
    result = YdxDiSetStopCondition(id, 0, 0)     ' サンプル数
    If result <> 0 Then
        ResultShow("YdxDiSetStopCondition", result)
        Exit Sub
    End If

    ' サンプリング停止条件(サンプル数)の設定
    result = YdxDiSetStopSampleNum(id, 1000)
    If result <> 0 Then
        ResultShow("YdxDiSetStopSampleNum", result)
        Exit Sub
    End If

    ' データをクリア
    result = YdxDiClearData(id)
    If result <> 0 Then
        ResultShow("YdxDiClearData", result)
        Exit Sub
    End If

    ' デジタル入力動作を開始
    result = YdxDiStart(id)
    If result <> 0 Then
        ResultShow("YdxDiStart", result)
        Exit Sub
    End If

    ' 動作終了待ち
    Dim status, sampleCount, repeatCount As Integer
    ' 動作中ステータスがOFFになるまでポーリング
    Do
        ' ステータスの取得
        result = YdxDiGetStatus(id, status, sampleCount, repeatCount)
        If result <> 0 Then
            ResultShow("YdxDiGetStatus", result)
            Exit Sub
        End If

        statusTextBox.Text = status.ToString("X").PadLeft(8, "0"c) & "h"
        sampleCountTextBox.Text = sampleCount.ToString()
        repeatCountTextBox.Text = repeatCount.ToString()
        Application.DoEvents()

        If (status And YDX_STATUS_COMMUNICATE_ERR) <> 0 Then
            MessageBox.Show("通信エラーが発生しました", "", MessageBoxButtons.OK,
MessageBoxIcon.Hand)
            Exit Sub
        End If

        If (status And YDX_STATUS_HARDWARE_ERR) <> 0 Then
            MessageBox.Show("ハードウェアエラーが発生しました", "", MessageBoxButtons.OK,
MessageBoxIcon.Hand)
            Exit Sub
        End If

        If (status And YDX_STATUS_OVERRUN_ERR) <> 0 Then
```

```
            MessageBox.Show("オーバランエラーが発生しました", "", MessageBoxButtons.OK,
MessageBoxIcon.Hand)
            Exit Sub
        End If

        If (status And YDX_STATUS_SAMPLE_CLOCK_ERR) <> 0 Then
            MessageBox.Show("サンプリングクロックエラーが発生しました", "", MessageBoxButtons.OK,
MessageBoxIcon.Hand)
            Exit Sub
        End If

    Loop While(status And YDX_STATUS_BUSY) <> 0

    ' データの読み出し
    Dim data(sampleCount) As Integer
    result = YdxDiGetData(id, sampleCount, data)
    If result <> 0 Then
        ResultShow("YdxDiGetData", result)
        If result <> YDX_RESULT_DI_EXCEED_DATA_NUM And result <>
YDX_RESULT_DI_EXCEED_BUF_SIZ Then
            Exit Sub
        End If
    End If

    ' 表示
    Dim txt As String = ""
    For sampleIndex As Integer = 0 To sampleCount - 1
        txt &= (sampleIndex + 1).ToString().PadLeft(5) & " : "
        ' 2進数表記
        txt &= Convert.ToString(data(sampleIndex) >> 16 And &HF, 2).PadLeft(4, "0"c) & " "
        txt &= Convert.ToString(data(sampleIndex) >> 8 And &HF, 2).PadLeft(4, "0"c) & " "
        txt &= Convert.ToString(data(sampleIndex) >> 4 And &HF, 2).PadLeft(4, "0"c) & " "
        txt &= Convert.ToString(data(sampleIndex) And &HF, 2).PadLeft(4, "0"c)
        ' 16進数表記
        txt &= " (" & data(sampleIndex).ToString("X").PadLeft(4) & "h)"
        txt &= Environment.NewLine
    Next

    dataTextBox.Text = txt
End Sub
```

クローズ

```
Private Sub closeButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles closeButton.Click
    unitSwitchComboBox.Enabled = True
    modelNameComboBox.Enabled = True
    result = YdxClose(id)
    If result <> 0 Then
        ResultShow("YdxClose", result)
    Else
        ResultShow("クローズ", result)
    End If
End Sub
```

フォームクローズ

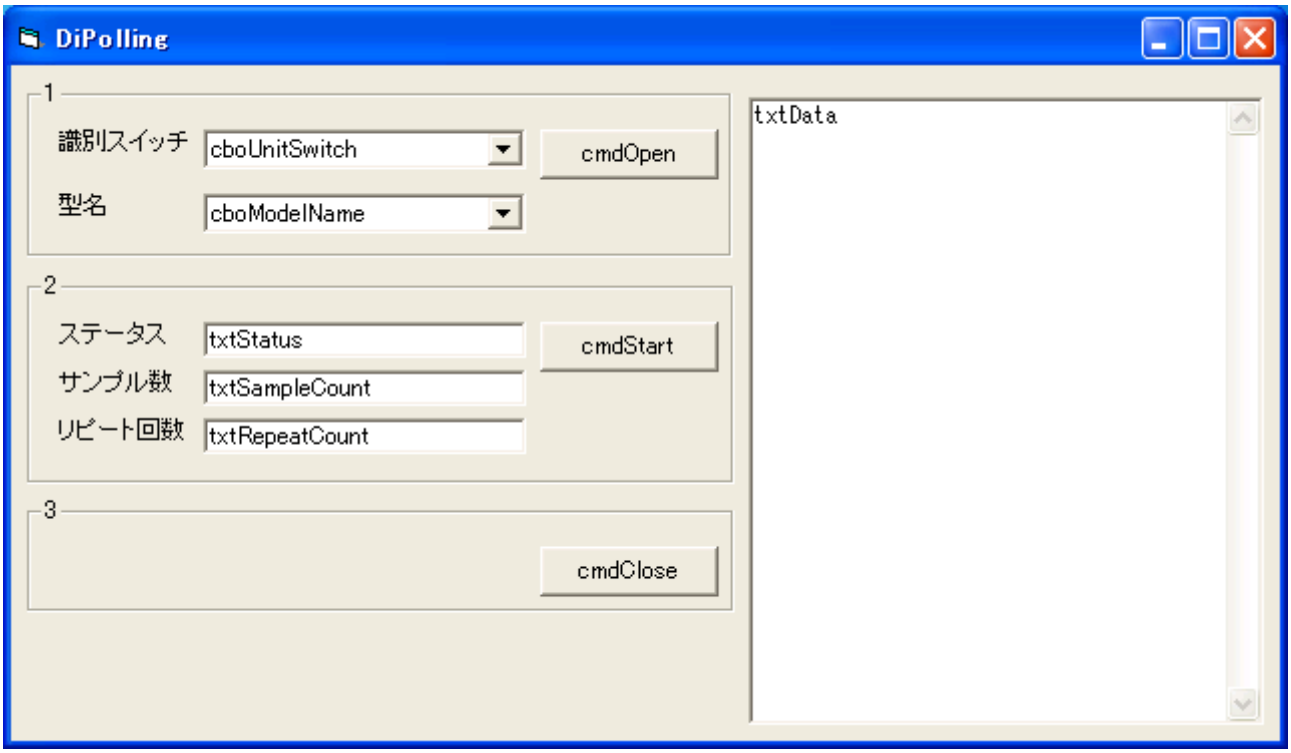
```
Private Sub Form1_Closing(ByVal sender As Object, ByVal e As
System.ComponentModel.CancelEventArgs) Handles MyBase.Closing
    result = YdxClose(id)
```

```
    If result <> 0 And result <> YDX_RESULT_NOT_OPEN Then
        ResultShow("YdxClose", result)
    End If
End Sub
```

開発環境の設定

1. Ydx.bas をプロジェクトフォルダにコピーします。
2. Ydx.bas をプロジェクトに追加します。

コントロール



変数

```
Dim id As Long
Dim result As Long
```

実行結果の表示

```
Private Sub ResultShow(ByVal title As String, ByVal resultCode As Long)
    Dim resultString As String
    Call YdxCnvResultToString(resultCode, resultString)
    Select Case resultCode
        Case 0, Ydx.YDX_RESULT_DI_EXCEED_DATA_NUM, Ydx.YDX_RESULT_DI_EXCEED_BUF_SIZ
            MsgBox resultString, vbInformation, title
        Case Else
            MsgBox resultString, vbCritical, title
    End Select
End Sub
```

フォームロード

```
Private Sub Form_Load()
    ' ユニット識別スイッチ
    cboUnitSwitch.AddItem "0"
    cboUnitSwitch.AddItem "1"
    cboUnitSwitch.AddItem "2"
    cboUnitSwitch.AddItem "3"
    cboUnitSwitch.AddItem "4"
    cboUnitSwitch.AddItem "5"
    cboUnitSwitch.AddItem "6"
    cboUnitSwitch.AddItem "7"
    cboUnitSwitch.AddItem "8"
    cboUnitSwitch.AddItem "9"
    cboUnitSwitch.AddItem "A"
    cboUnitSwitch.AddItem "B"
    cboUnitSwitch.AddItem "C"
    cboUnitSwitch.AddItem "D"
    cboUnitSwitch.AddItem "E"
    cboUnitSwitch.AddItem "F"
    cboUnitSwitch.ListIndex = 0

    ' 型名
    cboModelName.AddItem "DIO-16/16C-USC"
    cboModelName.AddItem "DIO-16/16D-UBC"
    cboModelName.AddItem "DIO-16/16D-USC"
    cboModelName.ListIndex = 0
End Sub
```

オープン

```
Private Sub cmdOpen_Click()
    result = YdxOpen(cboUnitSwitch.ListIndex, cboModelName.Text, 0, id)
    If result <> 0 Then
        Call ResultShow("YdxOpen", result)
    Else
        cboUnitSwitch.Enabled = False
        cboModelName.Enabled = False
        Call ResultShow("オープン", result)
    End If
End Sub
```

入力開始

```
Private Sub cmdStart_Click()
    txtData.Text = ""
    DoEvents

    ' データバッファの設定
    result = YdxDiSetBuffer(id, 0)    ' FIFOバッファ
    If result <> 0 Then
        Call ResultShow("YdxDiSetBuffer", result)
        Exit Sub
    End If

    ' サンプリングクロックの設定
    result = YdxDiSetClock(id, 0)     ' 内部クロック
    If result <> 0 Then
        Call ResultShow("YdxDiSetClock", result)
        Exit Sub
    End If
```

```
' 内部クロック周期の設定
result = YdxDiSetClockInternal(id, 1000) ' 1000μsec
If result <> 0 Then
    Call ResultShow("YdxDiSetClockInternal", result)
    Exit Sub
End If

' サンプリング開始条件の設定
result = YdxDiSetStartCondition(id, 0, 0)   ' ソフトウェア
If result <> 0 Then
    Call ResultShow("YdxDiSetStartCondition", result)
    Exit Sub
End If

' サンプリング停止条件の設定
result = YdxDiSetStopCondition(id, 0, 0)    ' サンプル数
If result <> 0 Then
    Call ResultShow("YdxDiSetStopCondition", result)
    Exit Sub
End If

' サンプリング停止条件（サンプル数）の設定
result = YdxDiSetStopSampleNum(id, 1000)
If result <> 0 Then
    Call ResultShow("YdxDiSetStopSampleNum", result)
    Exit Sub
End If

' データをクリア
result = YdxDiClearData(id)
If result <> 0 Then
    Call ResultShow("YdxDiClearData", result)
    Exit Sub
End If

' デジタル入力動作を開始
result = YdxDiStart(id)
If result <> 0 Then
    Call ResultShow("YdxDiStart", result)
    Exit Sub
End If

' 動作終了待ち
Dim status, sampleCount, repeatCount As Long
' 動作中ステータスがOFFになるまでポーリング
Do
    ' ステータスの取得
    result = YdxDiGetStatus(id, status, sampleCount, repeatCount)
    If result <> 0 Then
        Call ResultShow("YdxDiGetStatus", result)
        Exit Sub
    End If
    txtStatus.Text = Right("0000000" & Hex(status), 8) & "h"
    txtSampleCount.Text = Format(sampleCount)
    txtRepeatCount.Text = Format(repeatCount)
    DoEvents

    If (status And YDX_STATUS_COMMUNICATE_ERR) <> 0 Then
        MsgBox "通信エラーが発生しました", vbCritical
        Exit Sub
    End If

    If (status And YDX_STATUS_HARDWARE_ERR) <> 0 Then
```

```
            MsgBox "ハードウェアエラーが発生しました", vbCritical
            Exit Sub
        End If

        If(status And YDX_STATUS_OVERRUN_ERR) <> 0 Then
            MsgBox "オーバランエラーが発生しました", vbCritical
            Exit Sub
        End If

        If(status And YDX_STATUS_SAMPLE_CLOCK_ERR) <> 0 Then
            MsgBox "サンプリングクロックエラーが発生しました", vbCritical
            Exit Sub
        End If
    Loop While(status And YDX_STATUS_BUSY) <> 0

    ' データの読み出し
    Dim data() As Long
    ReDim data(sampleCount)
    result = YdxDiGetData(id, sampleCount, data(0))
    If result <> 0 Then
        Call ResultShow("YdxDiGetData", result)
        If result <> YDX_RESULT_DI_EXCEED_DATA_NUM And result <>
YDX_RESULT_DI_EXCEED_BUF_SIZ Then
            Exit Sub
        End If
    End If

    ' 表示
    Dim txt As String
    txt = ""
    Dim sampleIndex As Long
    For sampleIndex = 0 To sampleCount - 1
        txt = txt & Right("    " & Str(sampleIndex + 1), 5) & " : "

        ' 2進数表記
        Dim dt As Long
        dt = data(sampleIndex)
        Dim bitMask As Long
        bitMask = 32768
        Dim place As Integer
        For place = 0 To 15
            if(dt And bitMask) <> 0 Then
                txt = txt & "1"
            Else
                txt = txt & "0"
            End If
            If place = 3 Or place = 7 Or place = 11 Then
                txt = txt & " "
            End If
            bitMask = bitMask / 2
        Next

        ' 16進数表記
        txt = txt & " (" & Right("   " & Hex(data(sampleIndex)), 4) & "h)" & vbCrLf
    Next

    txtData.Text = txt
End Sub
```

クローズ

```
Private Sub cmdClose_Click()
    cboUnitSwitch.Enabled = True
    cboModelName.Enabled = True
    result = YdxClose(id)
    If result <> 0 Then
        Call ResultShow("YdxClose", result)
    Else
        Call ResultShow("クローズ", result)
    End If
End Sub
```

フォームアンロード

```
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    result = YdxClose(id)
    If result <> 0 And result <> YDX_RESULT_NOT_OPEN Then
        Call ResultShow("YdxClose", result)
    End If
End Sub
```

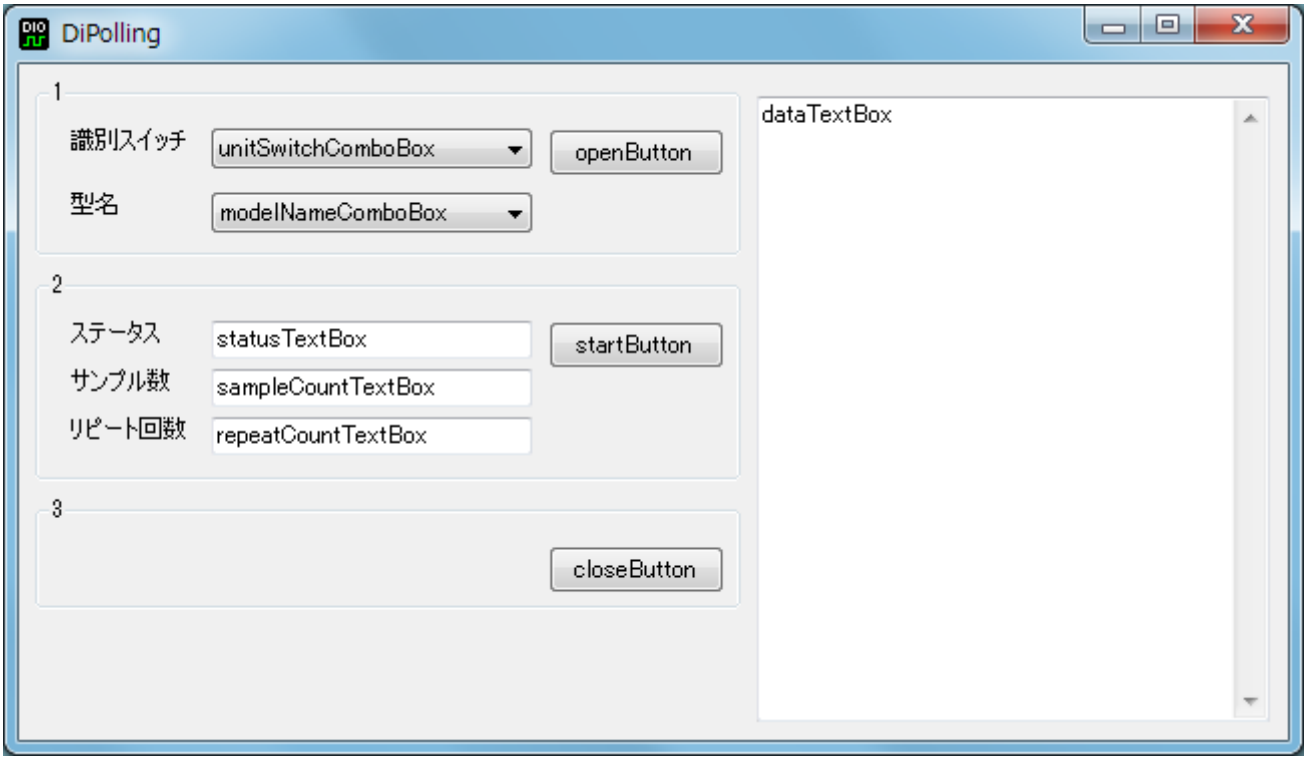
C++/CLI

開発環境の設定

1. YdxCLI.h をプロジェクトフォルダにコピーします。
2. YdxCLI.h をプロジェクトに追加します。
3. ソースファイルに YdxCLI.h をインクルードします。
4. usingディレクティブを使ってYdxCLIを宣言します。

```
using namespace YdxCLI;
```

コントロール



変数

```
int id;
```

実行結果の表示

```
private: System::Void ResultShow(String^ title, int resultCode)
{
    StringBuilder^ resultString = gcnew StringBuilder(256);
    YdxCnvResultToString(resultCode, resultString);
    switch (resultCode)
    {
    case 0:
    case YDX_RESULT_DI_EXCEED_DATA_NUM:
    case YDX_RESULT_DI_EXCEED_BUF_SIZ:
```

```
        MessageBox::Show(resultString->ToString(), title, MessageBoxButtons::OK,
MessageBoxIcon::Asterisk);
        break;
    default:
        MessageBox::Show(resultString->ToString(), title, MessageBoxButtons::OK,
MessageBoxIcon::Hand);
        break;
    }
}
```

フォームロード

```
private: System::Void Form1_Load(System::Object^ sender, System::EventArgs^ e)
{
    // ユニット識別スイッチ
    unitSwitchComboBox->ResetText();
    unitSwitchComboBox->Items->AddRange(gcnew array<String^> { "0", "1", "2", "3", "4", "5",
"6", "7", "8", "9", "A", "B", "C", "D", "E", "F" });
    unitSwitchComboBox->SelectedIndex = 0;
    // 型名
    modelNameComboBox->ResetText();
    modelNameComboBox->Items->AddRange(gcnew array<String^> { "DIO-16/16C-USC", "DIO-16/16D-
UBC", "DIO-16/16D-USC" });
    modelNameComboBox->SelectedIndex = 0;
}
```

オープン

```
private: System::Void openButton_Click(System::Object^ sender, System::EventArgs^ e)
{
    int getId;
    int result = YdxOpen(unitSwitchComboBox->SelectedIndex, modelNameComboBox->Text, 0,
&getId);
    if(result != 0)
        ResultShow("YdxOpen", result);
    else
    {
        unitSwitchComboBox->Enabled = false;
        modelNameComboBox->Enabled = false;
        ResultShow("オープン", result);
        id = getId;
    }
}
```

入力開始

```
private: System::Void startButton_Click(System::Object^ sender, System::EventArgs^ e)
{
    dataTextBox->ResetText();
    Application::DoEvents();

    // データバッファの設定
    int result = YdxDiSetBuffer(id, 0); // FIFOバッファ
    if(result != 0)
    {
        ResultShow("YdxDiSetBuffer", result);
        return;
    }
```

```
// サンプリングクロックの設定
result = YdxDiSetClock(id, 0);  // 内部クロック
if(result != 0)
{
    ResultShow("YdxDiSetClock", result);
    return;
}

// 内部クロック周期の設定
result = YdxDiSetClockInternal(id, 1000); // 1000μsec
if(result != 0)
{
    ResultShow("YdxDiSetClockInternal", result);
    return;
}

// サンプリング開始条件の設定
result = YdxDiSetStartCondition(id, 0, 0);  // ソフトウェア
if(result != 0)
{
    ResultShow("YdxDiSetStartCondition", result);
    return;
}

// サンプリング停止条件の設定
result = YdxDiSetStopCondition(id, 0, 0);   // サンプル数
if(result != 0)
{
    ResultShow("YdxDiSetStopCondition", result);
    return;
}

// サンプリング停止条件（サンプル数）の設定
const int SAMPLE_NUM = 1000;    // 1000個
result = YdxDiSetStopSampleNum(id, SAMPLE_NUM);
if(result != 0)
{
    ResultShow("YdxDiSetStopSampleNum", result);
    return;
}

// データをクリア
result = YdxDiClearData(id);
if(result != 0)
{
    ResultShow("YdxDiClearData", result);
    return;
}

// デジタル入力動作を開始
result = YdxDiStart(id);
if(result != 0)
{
    ResultShow("YdxDiStart", result);
    return;
}

// 動作終了待ち
int status, sampleCount, repeatCount;
//動作中ステータスがOFFになるまでポーリング
do
{
    //ステータスの取得
```

```
        result = YdxDiGetStatus(id, &status, &sampleCount, &repeatCount);
        if(result != 0)
        {
            ResultShow("YdxDiGetStatus", result);
            return;
        }

        statusTextBox->Text = status.ToString("X")->PadLeft(8, '0') + "h";
        sampleCountTextBox->Text = sampleCount.ToString();
        repeatCountTextBox->Text = repeatCount.ToString();
        Application::DoEvents();
        if((status & YDX_STATUS_COMMUNICATE_ERR) != 0)
        {
            MessageBox::Show("通信エラーが発生しました", "", MessageBoxButtons::OK,
MessageBoxIcon::Hand);
            return;
        }
        if((status & YDX_STATUS_HARDWARE_ERR) != 0)
        {
            MessageBox::Show("ハードウェアエラーが発生しました", "", MessageBoxButtons::OK,
MessageBoxIcon::Hand);
            return;
        }
        if((status & YDX_STATUS_OVERRUN_ERR) != 0)
        {
            MessageBox::Show("オーバランエラーが発生しました", "", MessageBoxButtons::OK,
MessageBoxIcon::Hand);
            return;
        }
        if((status & YDX_STATUS_SAMPLE_CLOCK_ERR) != 0)
        {
            MessageBox::Show("サンプリングクロックエラーが発生しました", "", MessageBoxButtons::OK,
MessageBoxIcon::Hand);
            return;
        }
    } while ((status & YDX_STATUS_BUSY) != 0);

    // データの読み出し
    int data[SAMPLE_NUM];
    result = YdxDiGetData(id, &sampleCount, data);
    if(result != 0)
    {
        ResultShow("YdxDiGetData", result);
        if((result != YDX_RESULT_DI_EXCEED_DATA_NUM) && (result !=
YDX_RESULT_DI_EXCEED_BUF_SIZ))
            return;
    }

    // 表示
    String^ txt = "";
    for (int sampleIndex = 0; sampleIndex < sampleCount; sampleIndex++)
    {
        txt += (sampleIndex + 1).ToString()->PadLeft(5) + " : ";
        // 2進数表記
        txt += Convert::ToString(data[sampleIndex] >> 16 & 0x0f, 2)->PadLeft(4, '0') + " ";
        txt += Convert::ToString(data[sampleIndex] >> 8 & 0x0f, 2)->PadLeft(4, '0') + " ";
        txt += Convert::ToString(data[sampleIndex] >> 4 & 0x0f, 2)->PadLeft(4, '0') + " ";
        txt += Convert::ToString(data[sampleIndex] & 0x0f, 2)->PadLeft(4, '0');
        // 16進数表記
        txt += " (" + data[sampleIndex].ToString("X")->PadLeft(4) + "h)";
        txt += Environment::NewLine;
    }
```

```
    dataTextBox->Text = txt;
}
```

クローズ

```
private: System::Void closeButton_Click(System::Object^ sender, System::EventArgs^ e)
{
    unitSwitchComboBox->Enabled = true;
    modelNameComboBox->Enabled = true;
    int result = YdxClose(id);
    if(result != 0)
        ResultShow("YdxClose", result);
    else
        ResultShow("クローズ", result);
}
```

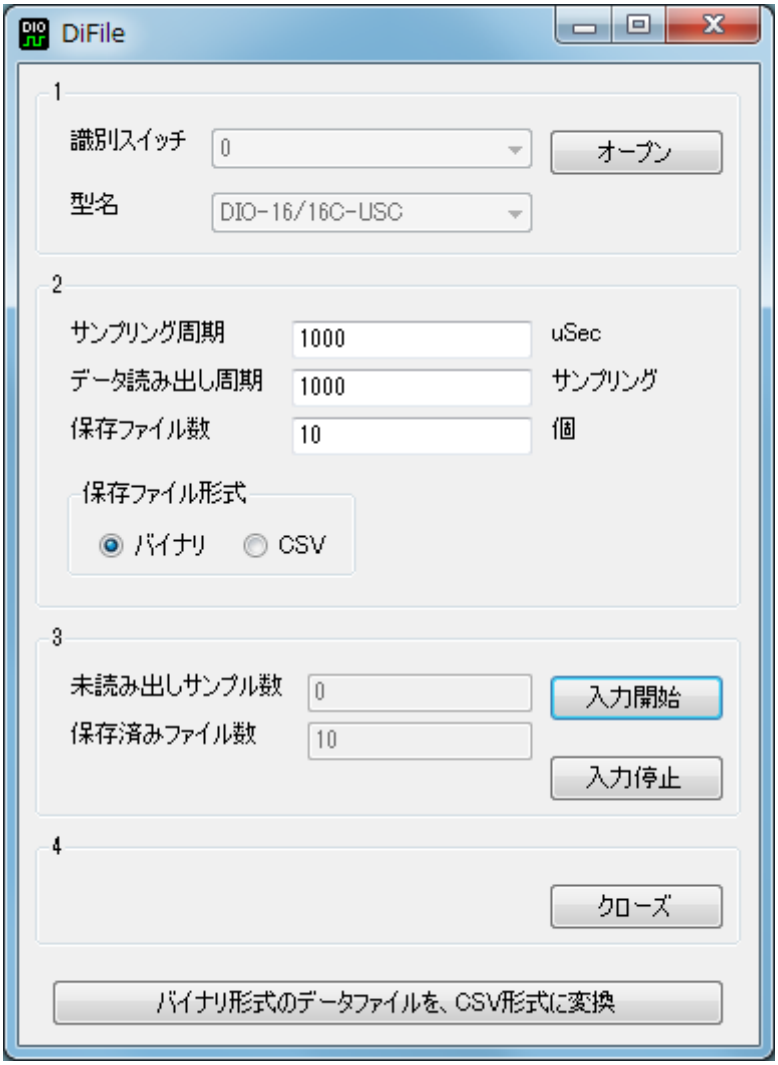
フォームクローズ

```
private: System::Void Form1_FormClosing(System::Object^ sender,
System::Windows::Forms::FormClosingEventArgs^ e)
{
    int result = YdxClose(id);
    if((result != 0) && (result != YDX_RESULT_NOT_OPEN))
    {
        ResultShow("YdxClose", result);
    }
}
```

DiFile

高機能デジタル入力 のサンプルプログラムです。
デジタル入力を連続でおこない、ファイルに保存します。

画面



1. オープン

   ユニットのオープンをします。

2. 設定

   設定をします。

3. 入力開始／入力停止

   入力の開始および停止をします。

   - 「入力開始」ボタンがクリックされると、「データファイル保存先の確認」→「動作条件の設定」→「動作開始」→「状態監視」という手順が実行されます。
   - データバッファに「データ読み出し周期」に設定されたサンプル数以上のデータが貯まると、読み出しをおこないファイルに保存します。

ファイル数が「保存ファイル数」に達するか「入力停止」ボタンがクリックされるまで、上記動作を繰り返します。

4. クローズ

   ユニットのクローズをします。
   オープンをした場合は、必ず実行する必要があります。

5. バイナリ形式のデータファイルを、CSV形式に変換

   バイナリ形式で保存したデータファイルを、CSV形式に変換します。
   ユニットの動作状態には関わりなく、いつでもおこなう事が可能です。
   （オープン・クローズどちらの状態でも構いません）

備考

サンプリング周期を短く設定した場合、処理（主にファイル保存）が間に合わなくなると、「未読み出しサンプル数」が「データ読み出し周期」に設定したサンプル数を超えて増えていきます。
データバッファが満杯になるとオーバランエラーが発生します。

データファイルのファイル名は、0からの連番です。

> ⚠ ファイル数が「保存ファイル数」に達するか「入力停止」されるまで保存を繰り返しますので、ディスク容量が不足しないように注意してください。

保存形式は「CSV」よりも「バイナリ」が高速です。
保存形式は「バイナリ」で高速におこない、停止後に「バイナリ形式のデータファイルを、CSV形式に変換」ボタンをクリックする事で、CSVファイルに変換する事もできます。

サンプルソース

- C#

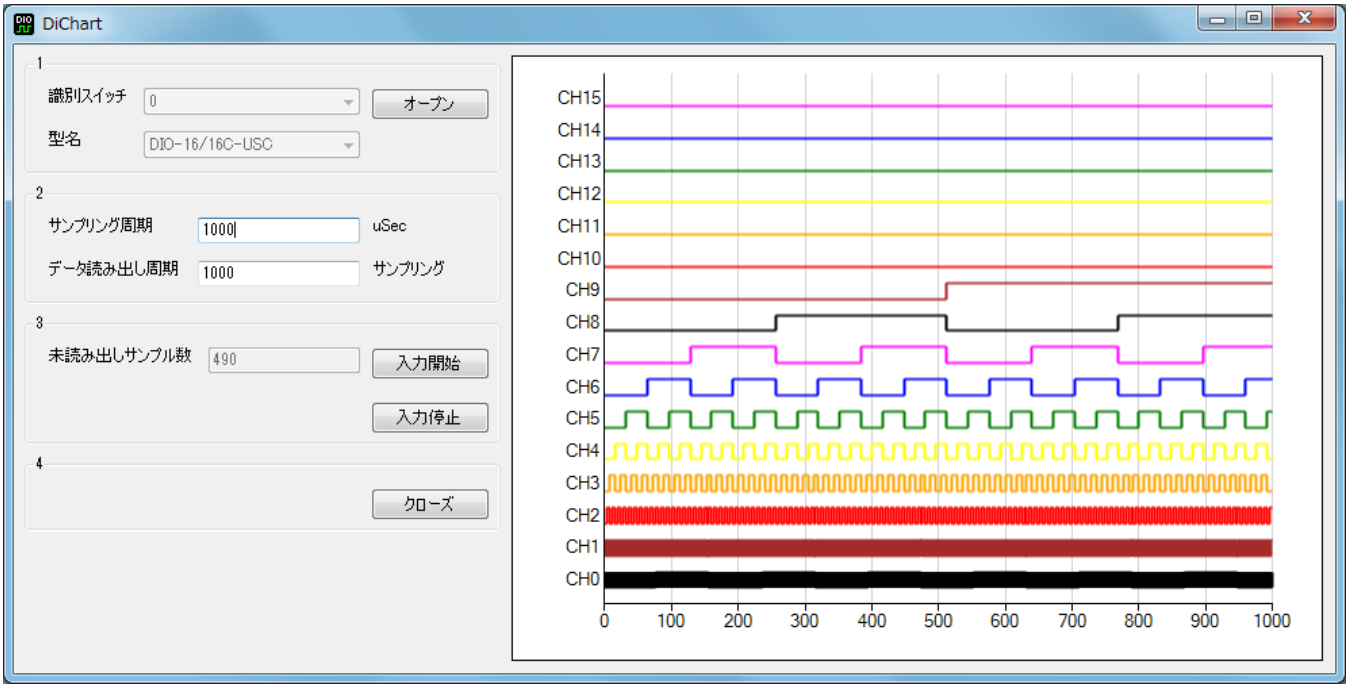
  ソフトウェアパック に付属しているサンプルプログラムのソースファイルを参照してください。

サンプルプログラム >

DiChart

高機能デジタル入力 のサンプルプログラムです。
デジタル入力を連続でおこない、波形をグラフ表示します。

画面



1. オープン

   ユニットのオープンをします。

2. 設定

   設定をします。

3. 入力開始／入力停止

   入力の開始および停止をします。

   - 「入力開始」ボタンがクリックされると、「動作条件の設定」→「動作開始」→「状態監視」という手順が実行されます。
   - データバッファに「データ読み出し周期」に設定されたサンプル数以上のデータが貯まると、読み出しをおこない波形をグラフ表示します。
   - 「入力停止」ボタンがクリックされるまで、上記動作を繰り返します。

4. クローズ

   ユニットのクローズをします。
   オープンをした場合は、必ず実行する必要があります。

備考

サンプリング周期を短く設定した場合、処理（主にグラフ描画）が間に合わなくなると、「未読み出しサンプル数」が「データ読み出し周期」に設定したサンプル数を超えて増えていきます。

データバッファが満杯になるとオーバランエラーが発生します。

サンプルソース

- C#

  ソフトウェアパック に付属しているサンプルプログラムのソースファイルを参照してください。

開発環境について

グラフ表示には「Chart Controls for Microsoft .NET Framework」を使用しています。
VisualStudio2008で使用する場合「Chart Controls for Microsoft .NET Framework」をインストールする必要があります。
VisualStudio2010以降で使用する場合は、インストールは不要です。

DoPolling

高機能デジタル出力 のサンプルプログラムです。
デジタル出力を1000回おこないます。
動作状態の監視をポーリングでおこなっています。

画面

1. オープン

   ユニットのオープンをします。

2. 出力開始

   出力を開始します。
   「動作条件の設定」→「動作開始」→「状態監視（動作が停止するまで）」という手順が実行されます。

3. クローズ

   ユニットのクローズをします。
   オープンをした場合は、必ず実行する必要があります。

サンプルソース

- C#
- Visual Basic（.NET2002以降）
- Visual Basic 6.0
- C++/CLI

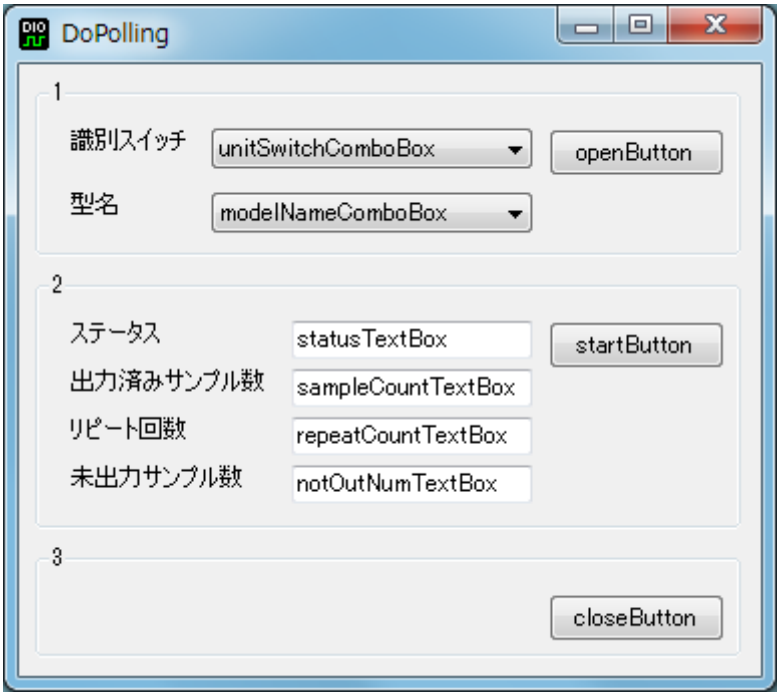
C#

開発環境の設定

1. Ydx.cs をプロジェクトフォルダにコピーします。
2. Ydx.cs をプロジェクトに追加します。
3. ソースファイルにusing ディレクティブを使ってYdxCsを宣言します。

```
using YdxCs;
```

コントロール



変数

```
private int id;
```

実行結果の表示

```
private void ResultShow(string title, int resultCode)
{
    string resultString;
    Ydx.CnvResultToString(resultCode, out resultString);
    switch (resultCode)
    {
        case 0:
        case Ydx.YDX_RESULT_DI_EXCEED_DATA_NUM:
        case Ydx.YDX_RESULT_DI_EXCEED_BUF_SIZ:
            MessageBox.Show(resultString, title, MessageBoxButtons.OK,
MessageBoxIcon.Asterisk);
            break;
        default:
```

```
            MessageBox.Show(resultString, title, MessageBoxButtons.OK, MessageBoxIcon.Hand);
            break;
    }
}
```

フォームロード

```
private void Form1_Load(object sender, EventArgs e)
{
    // ユニット識別スイッチ
    unitSwitchComboBox.ResetText();
    unitSwitchComboBox.Items.AddRange(new string[] { "0", "1", "2", "3", "4", "5", "6", "7",
"8", "9", "A", "B", "C", "D", "E", "F" });
    unitSwitchComboBox.SelectedIndex = 0;

    // 型名
    modelNameComboBox.ResetText();
    modelNameComboBox.Items.AddRange(new string[] { "DIO-16/16C-USC", "DIO-16/16D-UBC",
"DIO-16/16D-USC" });
    modelNameComboBox.SelectedIndex = 0;
}
```

オープン

```
private void openButton_Click(object sender, EventArgs e)
{
    int result = Ydx.Open(unitSwitchComboBox.SelectedIndex, modelNameComboBox.Text, 0, out
id);
    if(result != 0)
        ResultShow("YdxOpen", result);
    else
    {
        unitSwitchComboBox.Enabled = false;
        modelNameComboBox.Enabled = false;
        ResultShow("オープン", result);
    }
}
```

出力開始

```
private void startButton_Click(object sender, EventArgs e)
{
    int result;

    // データバッファの設定
    result = Ydx.DoSetBuffer(id, 0);   // FIFOバッファ
    if(result != 0)
    {
        ResultShow("YdxDoSetBuffer", result);
        return;
    }

    // チャネルの設定
    for (int channel = 0; channel < 16; channel++)
    {
        result = Ydx.DoSetChannel(id, channel, 1);  // 高機能デジタル出力モード
        if(result != 0)
        {
```

```
            ResultShow("YdxDoSetChannel", result);
            return;
        }
    }

    // サンプリングクロックの設定
    result = Ydx.DoSetClock(id, 0);     // 内部クロック
    if(result != 0)
    {
        ResultShow("YdxDoSetClock", result);
        return;
    }

    // 内部クロック周期の設定
    result = Ydx.DoSetClockInternal(id, 1000); // 1000μsec
    if(result != 0)
    {
        ResultShow("YdxDoSetClockInternal", result);
        return;
    }

    // データの設定
    const int SAMPLE_NUM = 1000;     // サンプル数
    int[] data = new int[SAMPLE_NUM];
    for (int i = 0; i < SAMPLE_NUM; i++)
    {
        data[i] = i;
    }

    result = Ydx.DoSetData(id, SAMPLE_NUM, data);
    if(result != 0)
    {
        ResultShow("YdxDoSetData", result);
        return;
    }

    // サンプリング開始条件の設定
    result = Ydx.DoSetStartCondition(id, 0, 0);    // ソフトウェア
    if(result != 0)
    {
        ResultShow("YdxDoSetStartCondition", result);
        return;
    }

    // サンプリング停止条件の設定
    result = Ydx.DoSetStopCondition(id, 0, 0);     // データ終了
    if(result != 0)
    {
        ResultShow("YdxDoSetStopCondition", result);
        return;
    }

    // デジタル出力動作を開始
    result = Ydx.DoStart(id);
    if(result != 0)
    {
        ResultShow("YdxDoStart", result);
        return;
    }

    // 動作終了待ち
    int status, sampleCount, repeatCount, notOutNum;
    //動作中ステータスがOFFになるまでポーリング
    do
```

```
    {
        Application.DoEvents();

        // ステータスの取得
        result = Ydx.DoGetStatus(id, out status, out sampleCount, out repeatCount, out
notOutNum);
        if(result != 0)
        {
            ResultShow("YdxDoGetStatus", result);
            return;
        }

        statusTextBox.Text = status.ToString("X").PadLeft(8, '0') + "h";
        sampleCountTextBox.Text = sampleCount.ToString();
        repeatCountTextBox.Text = repeatCount.ToString();
        notOutNumTextBox.Text = notOutNum.ToString();

        if((status & Ydx.YDX_STATUS_COMMUNICATE_ERR) != 0)
        {
            MessageBox.Show("通信エラーが発生しました", "", MessageBoxButtons.OK,
MessageBoxIcon.Hand);
            return;
        }

        if((status & Ydx.YDX_STATUS_HARDWARE_ERR) != 0)
        {
            MessageBox.Show("ハードウェアエラーが発生しました", "", MessageBoxButtons.OK,
MessageBoxIcon.Hand);
            return;
        }

        if((status & Ydx.YDX_STATUS_SAMPLE_CLOCK_ERR) != 0)
        {
            MessageBox.Show("サンプリングクロックエラーが発生しました", "", MessageBoxButtons.OK,
MessageBoxIcon.Hand);
            return;
        }
    } while ((status & Ydx.YDX_STATUS_BUSY) != 0);

    ResultShow("出力", 0);
}
```

クローズ

```
private void closeButton_Click(object sender, EventArgs e)
{
    unitSwitchComboBox.Enabled = true;
    modelNameComboBox.Enabled = true;
    int result = Ydx.Close(id);
    if(result != 0)
        ResultShow("YdxClose", result);
    else
        ResultShow("クローズ", result);
}
```

フォームクローズ

```
private void Form1_Closing(object sender, System.ComponentModel.CancelEventArgs e)
{
    int result = Ydx.Close(id);
```

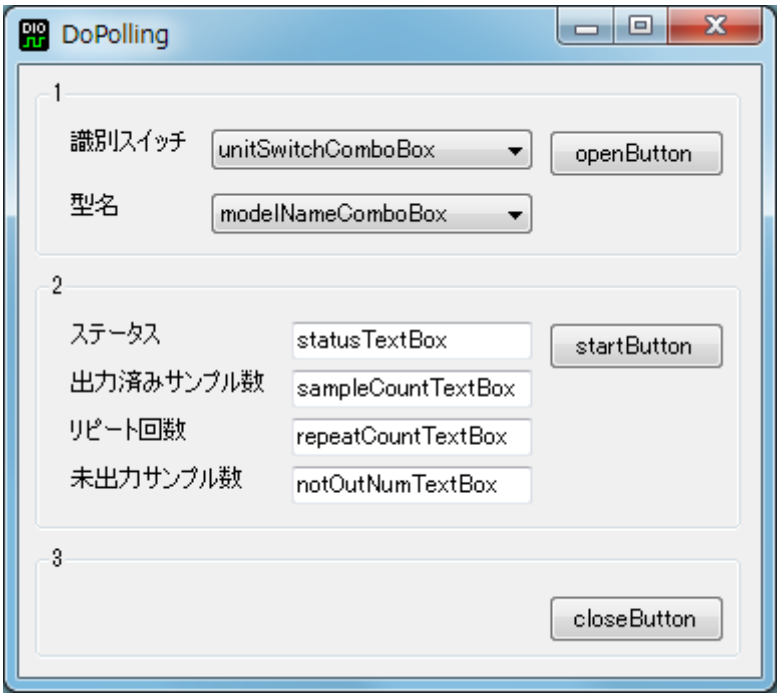
```
    if((result != 0) && (result != Ydx.YDX_RESULT_NOT_OPEN))
    {
        ResultShow("YdxClose", result);
    }
}
```

Visual Basic（.NET2002以降）

開発環境の設定

1. Ydx.vb をプロジェクトフォルダにコピーします。
2. Ydx.vb をプロジェクトに追加します。

コントロール



変数

```
Dim id As Integer
Dim result As Integer
```

実行結果の表示

```
Private Sub ResultShow(ByVal title As String, ByVal resultCode As Integer)
    Dim resultString As New StringBuilder(256)
    YdxCnvResultToString(resultCode, resultString)
    Select Case resultCode
        Case 0, Ydx.YDX_RESULT_DI_EXCEED_DATA_NUM, Ydx.YDX_RESULT_DI_EXCEED_BUF_SIZ
            MessageBox.Show(resultString.ToString(), title, MessageBoxButtons.OK,
MessageBoxIcon.Asterisk)
        Case Else
            MessageBox.Show(resultString.ToString(), title, MessageBoxButtons.OK,
MessageBoxIcon.Hand)
    End Select
End Sub
```

フォームロード

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
    ' ユニット識別スイッチ
    unitSwitchComboBox.ResetText()
    unitSwitchComboBox.Items.AddRange(New String() { "0", "1", "2", "3", "4", "5", "6", "7",
"8", "9", "A", "B", "C", "D", "E", "F" })
    unitSwitchComboBox.SelectedIndex = 0

    ' 型名
    modelNameComboBox.ResetText()
    modelNameComboBox.Items.AddRange(New String() { "DIO-16/16C-USC", "DIO-16/16D-UBC",
"DIO-16/16D-USC" })
    modelNameComboBox.SelectedIndex = 0
End Sub
```

オープン

```
Private Sub openButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles openButton.Click
    result = YdxOpen(unitSwitchComboBox.SelectedIndex, modelNameComboBox.Text, 0, id)
    If result <> 0 Then
        ResultShow("YdxOpen", result)
    Else
        unitSwitchComboBox.Enabled = False
        modelNameComboBox.Enabled = False
        ResultShow("オープン", result)
    End If
End Sub
```

出力開始

```
Private Sub startButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles startButton.Click
    ' データバッファの設定
    result = YdxDoSetBuffer(id, 0) ' FIFOバッファ
    If result <> 0 Then
        ResultShow("YdxDoSetBuffer", result)
        Exit Sub
    End If

    ' チャネルの設定
    For channel As Integer = 0 To 15
        result = YdxDoSetChannel(id, channel, 1) ' 高機能デジタル出力モード
        If result <> 0 Then
            ResultShow("YdxDoSetChannel", result)
            Exit Sub
        End If
    Next

    ' サンプリングクロックの設定
    result = YdxDoSetClock(id, 0)     ' 内部クロック
    If result <> 0 Then
        ResultShow("YdxDoSetClock", result)
        Exit Sub
    End If

    ' 内部クロック周期の設定
    result = YdxDoSetClockInternal(id, 1000) ' 1000μsec
    If result <> 0 Then
```

```
        ResultShow("YdxDoSetClockInternal", result)
        Exit Sub
    End If

    ' データの設定
    Const SAMPLE_NUM As Integer = 1000  ' サンプル数
    Dim data(SAMPLE_NUM) As Integer
    For i As Integer = 0 To SAMPLE_NUM - 1
        data(i) = i
    Next
    result = YdxDoSetData(id, SAMPLE_NUM, data)
    If result <> 0 Then
        ResultShow("YdxDoSetData", result)
        Exit Sub
    End If

    ' サンプリング開始条件の設定
    result = YdxDoSetStartCondition(id, 0, 0)   ' ソフトウェア
    If result <> 0 Then
        ResultShow("YdxDoSetStartCondition", result)
        Exit Sub
    End If

    ' サンプリング停止条件の設定
    result = YdxDoSetStopCondition(id, 0, 0)    ' データ終了
    If result <> 0 Then
        ResultShow("YdxDoSetStopCondition", result)
        Exit Sub
    End If

    ' デジタル出力動作を開始
    result = YdxDoStart(id)
    If result <> 0 Then
        ResultShow("YdxDoStart", result)
        Exit Sub
    End If

    ' 動作終了待ち
    Dim status, sampleCount, repeatCount, notOutNum As Integer
    '動作中ステータスがOFFになるまでポーリング
    Do
        Application.DoEvents()

        'ステータスの取得
        result = YdxDoGetStatus(id, status, sampleCount, repeatCount, notOutNum)
        If result <> 0 Then
            ResultShow("YdxDoGetStatus", result)
            Exit Sub
        End If
        statusTextBox.Text = status.ToString("X").PadLeft(8, "0"c) & "h"
        sampleCountTextBox.Text = sampleCount.ToString()
        repeatCountTextBox.Text = repeatCount.ToString()
        notOutNumTextBox.Text = notOutNum.ToString()

        if(status And YDX_STATUS_COMMUNICATE_ERR) <> 0 Then
            MessageBox.Show("通信エラーが発生しました", "", MessageBoxButtons.OK,
MessageBoxIcon.Hand)
            Exit Sub
        End If
        if(status And YDX_STATUS_HARDWARE_ERR) <> 0 Then
            MessageBox.Show("ハードウェアエラーが発生しました", "", MessageBoxButtons.OK,
MessageBoxIcon.Hand)
            Exit Sub
        End If
```

```
        if(status And YDX_STATUS_SAMPLE_CLOCK_ERR) <> 0 Then
            MessageBox.Show("サンプリングクロックエラーが発生しました", "", MessageBoxButtons.OK,
MessageBoxIcon.Hand)
            Exit Sub
        End If
    Loop While(status And YDX_STATUS_BUSY) <> 0
    ResultShow("出力", 0)
End Sub
```

クローズ

```
Private Sub closeButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles closeButton.Click
    unitSwitchComboBox.Enabled = True
    modelNameComboBox.Enabled = True
    result = YdxClose(id)
    If result <> 0 Then
        ResultShow("YdxClose", result)
    Else
        ResultShow("クローズ", result)
    End If
End Sub
```

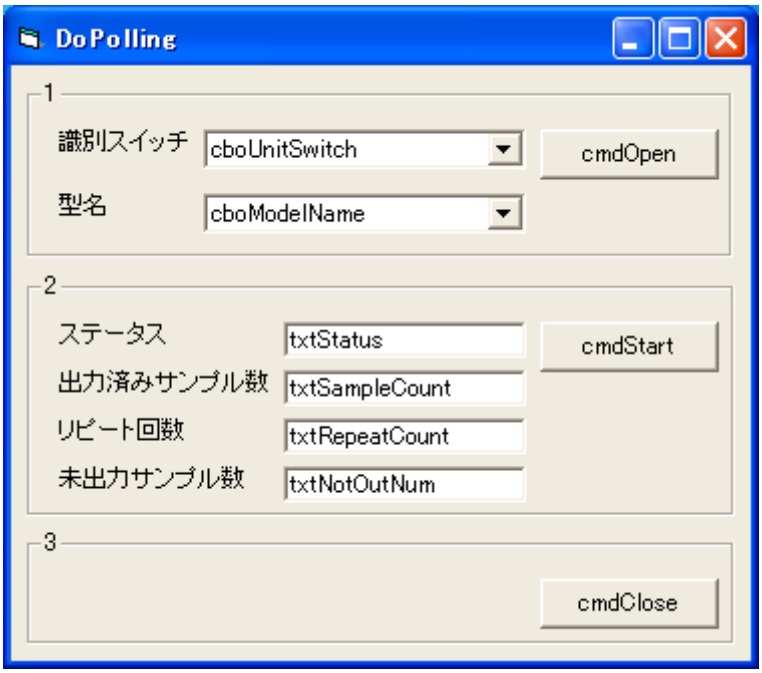
フォームクローズ

```
Private Sub Form1_Closing(ByVal sender As Object, ByVal e As
System.ComponentModel.CancelEventArgs) Handles MyBase.Closing
    result = YdxClose(id)
    If result <> 0 And result <> YDX_RESULT_NOT_OPEN Then
        ResultShow("YdxClose", result)
    End If
End Sub
```

DoPolling

Visual Basic 6.0 サンプル

コントロール



変数

```
Dim id As Long
Dim result As Long
```

実行結果の表示

```
Private Sub ResultShow(ByVal title As String, ByVal resultCode As Long)
    Dim resultString As String
    Call YdxCnvResultToString(resultCode, resultString)
    Select Case resultCode
        Case 0, Ydx.YDX_RESULT_DI_EXCEED_DATA_NUM, Ydx.YDX_RESULT_DI_EXCEED_BUF_SIZ
            MsgBox resultString, vbInformation, title
        Case Else
            MsgBox resultString, vbCritical, title
    End Select
End Sub
```

フォームロード

```
Private Sub Form_Load()
    ' ユニット識別スイッチ
    cboUnitSwitch.AddItem "0"
    cboUnitSwitch.AddItem "1"
    cboUnitSwitch.AddItem "2"
    cboUnitSwitch.AddItem "3"
    cboUnitSwitch.AddItem "4"
```

```
    cboUnitSwitch.AddItem "5"
    cboUnitSwitch.AddItem "6"
    cboUnitSwitch.AddItem "7"
    cboUnitSwitch.AddItem "8"
    cboUnitSwitch.AddItem "9"
    cboUnitSwitch.AddItem "A"
    cboUnitSwitch.AddItem "B"
    cboUnitSwitch.AddItem "C"
    cboUnitSwitch.AddItem "D"
    cboUnitSwitch.AddItem "E"
    cboUnitSwitch.AddItem "F"
    cboUnitSwitch.ListIndex = 0

    ' 型名
    cboModelName.AddItem "DIO-16/16C-USC"
    cboModelName.AddItem "DIO-16/16D-UBC"
    cboModelName.AddItem "DIO-16/16D-USC"
    cboModelName.ListIndex = 0
End Sub
```

オープン

```
Private Sub cmdOpen_Click()
    result = YdxOpen(cboUnitSwitch.ListIndex, cboModelName.Text, 0, id)
    If result <> 0 Then
        Call ResultShow("YdxOpen", result)
    Else
        cboUnitSwitch.Enabled = False
        cboModelName.Enabled = False
        Call ResultShow("オープン", result)
    End If
End Sub
```

出力開始

```
Private Sub cmdStart_Click()
    ' データバッファの設定
    result = YdxDoSetBuffer(id, 0)    ' FIFOバッファ
    If result <> 0 Then
        Call ResultShow("YdxDoSetBuffer", result)
        Exit Sub
    End If

    ' チャネルの設定
    Dim channel As Long
    For channel = 0 To 15
        result = YdxDoSetChannel(id, channel, 1) ' 高機能デジタル出力モード
        If result <> 0 Then
            Call ResultShow("YdxDoSetChannel", result)
            Exit Sub
        End If
    Next

    ' サンプリングクロックの設定
    result = YdxDoSetClock(id, 0)     ' 内部クロック
    If result <> 0 Then
        Call ResultShow("YdxDoSetClock", result)
        Exit Sub
    End If
```

```
    ' 内部クロック周期の設定
    result = YdxDoSetClockInternal(id, 1000) ' 1000μsec
    If result <> 0 Then
        Call ResultShow("YdxDoSetClockInternal", result)
        Exit Sub
    End If

    ' データの設定
    Const SAMPLE_NUM As Long = 1000  ' サンプル数
    Dim data(SAMPLE_NUM) As Long
    Dim i As Long
    For i = 0 To SAMPLE_NUM - 1
        data(i) = i
    Next
    result = YdxDoSetData(id, SAMPLE_NUM, data(0))
    If result <> 0 Then
        Call ResultShow("YdxDoSetData", result)
        Exit Sub
    End If

    ' サンプリング開始条件の設定
    result = YdxDoSetStartCondition(id, 0, 0)    ' ソフトウェア
    If result <> 0 Then
        Call ResultShow("YdxDoSetStartCondition", result)
        Exit Sub
    End If

    ' サンプリング停止条件の設定
    result = YdxDoSetStopCondition(id, 0, 0)     ' データ終了
    If result <> 0 Then
        Call ResultShow("YdxDoSetStopCondition", result)
        Exit Sub
    End If

    ' デジタル出力動作を開始
    result = YdxDoStart(id)
    If result <> 0 Then
        Call ResultShow("YdxDoStart", result)
        Exit Sub
    End If

    ' 動作終了待ち
    Dim status, sampleCount, repeatCount, notOutNum As Long
    '動作中ステータスがOFFになるまでポーリング
    Do
        DoEvents

        'ステータスの取得
        result = YdxDoGetStatus(id, status, sampleCount, repeatCount, notOutNum)
        If result <> 0 Then
            Call ResultShow("YdxDoGetStatus", result)
            Exit Sub
        End If
        txtStatus.Text = Right("0000000" & Hex(status), 8) & "h"
        txtSampleCount.Text = Format(sampleCount)
        txtRepeatCount.Text = Format(repeatCount)
        txtNotOutNum.Text = Format(notOutNum)

        if(status And YDX_STATUS_COMMUNICATE_ERR) <> 0 Then
            MsgBox "通信エラーが発生しました", vbCritical
            Exit Sub
        End If
        if(status And YDX_STATUS_HARDWARE_ERR) <> 0 Then
            MsgBox "ハードウェアエラーが発生しました", vbCritical
```

```
            Exit Sub
        End If
        if(status And YDX_STATUS_SAMPLE_CLOCK_ERR) <> 0 Then
            MsgBox "サンプリングクロックエラーが発生しました", vbCritical
            Exit Sub
        End If
    Loop While(status And YDX_STATUS_BUSY) <> 0
    Call ResultShow("出力", 0)
End Sub
```

クローズ

```
Private Sub cmdClose_Click()
    cboUnitSwitch.Enabled = True
    cboModelName.Enabled = True
    result = YdxClose(id)
    If result <> 0 Then
        Call ResultShow("YdxClose", result)
    Else
        Call ResultShow("クローズ", result)
    End If
End Sub
```

フォームアンロード

```
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    result = YdxClose(id)
    If result <> 0 And result <> YDX_RESULT_NOT_OPEN Then
        Call ResultShow("YdxClose", result)
    End If
End Sub
```

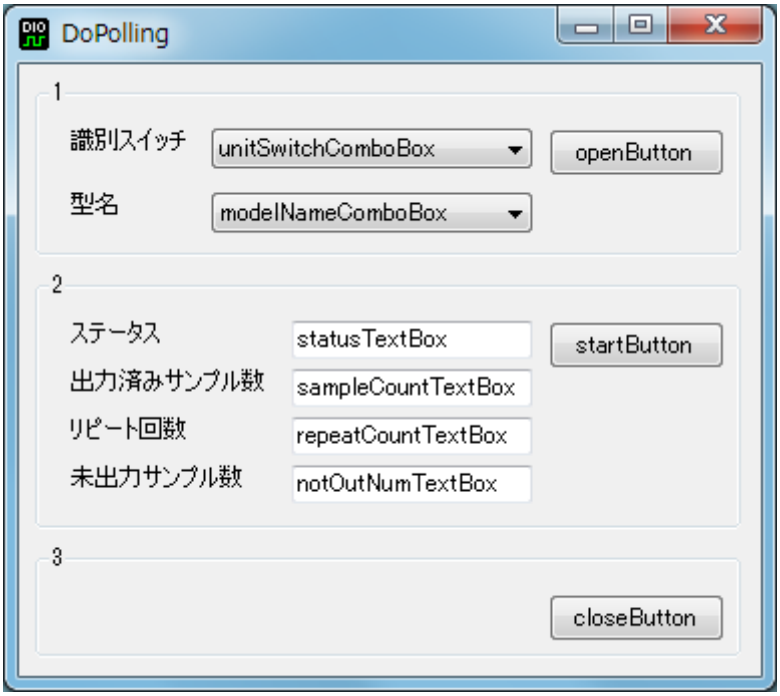
C++/CLI

開発環境の設定

1. YdxCLI.h をプロジェクトフォルダにコピーします。
2. YdxCLI.h をプロジェクトに追加します。
3. ソースファイルに YdxCLI.h をインクルードします。
4. usingディレクティブを使ってYdxCLIを宣言します。

```
using namespace YdxCLI;
```

コントロール



変数

```
int id;
```

実行結果の表示

```
private: System::Void ResultShow(String^ title, int resultCode)
{
    StringBuilder ^resultString = gcnew StringBuilder(256);
    YdxCnvResultToString(resultCode, resultString);
    switch (resultCode)
    {
    case 0:
    case YDX_RESULT_DI_EXCEED_DATA_NUM:
    case YDX_RESULT_DI_EXCEED_BUF_SIZ:
        MessageBox::Show(resultString->ToString(), title, MessageBoxButtons::OK,
MessageBoxIcon::Asterisk);
```

```
        break;
    default:
        MessageBox::Show(resultString->ToString(), title, MessageBoxButtons::OK,
MessageBoxIcon::Hand);
        break;
     }
}
```

フォームロード

```
private: System::Void Form1_Load(System::Object^ sender, System::EventArgs^ e)
{
    // ユニット識別スイッチ
    unitSwitchComboBox->ResetText();
    unitSwitchComboBox->Items->AddRange(gcnew array<String ^>{"0", "1", "2", "3", "4", "5",
"6", "7", "8", "9", "A", "B", "C", "D", "E", "F"});
    unitSwitchComboBox->SelectedIndex = 0;

    // 型名
    modelNameComboBox->ResetText();
    modelNameComboBox->Items->AddRange(gcnew array<String ^>{"DIO-16/16C-USC", "DIO-16/16D-
UBC", "DIO-16/16D-USC"});
    modelNameComboBox->SelectedIndex = 0;
}
```

オープン

```
private: System::Void openButton_Click(System::Object^ sender, System::EventArgs^ e)
{
    int getId;
    int result = YdxOpen(unitSwitchComboBox->SelectedIndex, modelNameComboBox->Text, 0,
&getId);
    if(result != 0)
        ResultShow("YdxOpen", result);
    else {
        unitSwitchComboBox->Enabled = false;
        modelNameComboBox->Enabled = false;
        ResultShow("オープン", result);
        id = getId;
    }
}
```

出力開始

```
private: System::Void startButton_Click(System::Object^ sender, System::EventArgs^ e)
{
    int result;

    // データバッファの設定
    result = YdxDoSetBuffer(id, 0);   // FIFOバッファ
    if(result != 0)
    {
        ResultShow("YdxDoSetBuffer", result);
        return;
    }

    // チャネルの設定
    for (int channel = 0; channel < 16; channel++)
```

```
    {
        result = YdxDoSetChannel(id, channel, 1); // 高機能デジタル出力モード
        if(result != 0)
        {
            ResultShow("YdxDoSetChannel", result);
            return;
        }
    }

    // サンプリングクロックの設定
    result = YdxDoSetClock(id, 0);     // 内部クロック
    if(result != 0)
    {
        ResultShow("YdxDoSetClock", result);
        return;
    }

    // 内部クロック周期の設定
    result = YdxDoSetClockInternal(id, 1000); // 1000μsec
    if(result != 0)
    {
        ResultShow("YdxDoSetClockInternal", result);
        return;
    }

    // データの設定
    const int SAMPLE_NUM = 1000;  // サンプル数
    int data[SAMPLE_NUM];
    for (int i = 0; i < SAMPLE_NUM; i++)
    {
        data[i] = i;
    }
    result = YdxDoSetData(id, SAMPLE_NUM, data);
    if(result != 0)
    {
        ResultShow("YdxDoSetData", result);
        return;
    }

    // サンプリング開始条件の設定
    result = YdxDoSetStartCondition(id, 0, 0);    // ソフトウェア
    if(result != 0)
    {
        ResultShow("YdxDoSetStartCondition", result);
        return;
    }

    // サンプリング停止条件の設定
    result = YdxDoSetStopCondition(id, 0, 0);     // データ終了
    if(result != 0)
    {
        ResultShow("YdxDoSetStopCondition", result);
        return;
    }

    // デジタル出力動作を開始
    result = YdxDoStart(id);
    if(result != 0)
    {
        ResultShow("YdxDoStart", result);
        return;
    }

    // 動作終了待ち
```

```
    int status, sampleCount, repeatCount, notOutNum;
    //動作中ステータスがOFFになるまでポーリング
    do
    {
        Application::DoEvents();

        //ステータスの取得
        result = YdxDoGetStatus(id, &status, &sampleCount, &repeatCount, &notOutNum);
        if(result != 0)
        {
            ResultShow("YdxDoGetStatus", result);
            return;
        }

        statusTextBox->Text=status.ToString("X")->PadLeft(8,'0')+"h";
        sampleCountTextBox->Text = sampleCount.ToString();
        repeatCountTextBox->Text = repeatCount.ToString();
        notOutNumTextBox->Text = notOutNum.ToString();

        if((status & YDX_STATUS_COMMUNICATE_ERR) != 0)
        {
            MessageBox::Show("通信エラーが発生しました", "", MessageBoxButtons::OK,
MessageBoxIcon::Hand);
            return;
        }

        if((status & YDX_STATUS_HARDWARE_ERR) != 0)
        {
            MessageBox::Show("ハードウェアエラーが発生しました", "", MessageBoxButtons::OK,
MessageBoxIcon::Hand);
            return;
        }

        if((status & YDX_STATUS_SAMPLE_CLOCK_ERR) != 0)
        {
            MessageBox::Show("サンプリングクロックエラーが発生しました", "", MessageBoxButtons::OK,
MessageBoxIcon::Hand);
            return;
        }
    } while ((status & YDX_STATUS_BUSY) != 0);
    ResultShow("出力", 0);
}
```

クローズ

```
private: System::Void closeButton_Click(System::Object^ sender, System::EventArgs^ e)
{
    unitSwitchComboBox->Enabled = true;
    modelNameComboBox->Enabled = true;
    int result = YdxClose(id);
    if(result != 0)
        ResultShow("YdxClose", result);
    else
        ResultShow("クローズ", result);
}
```

フォームクローズ

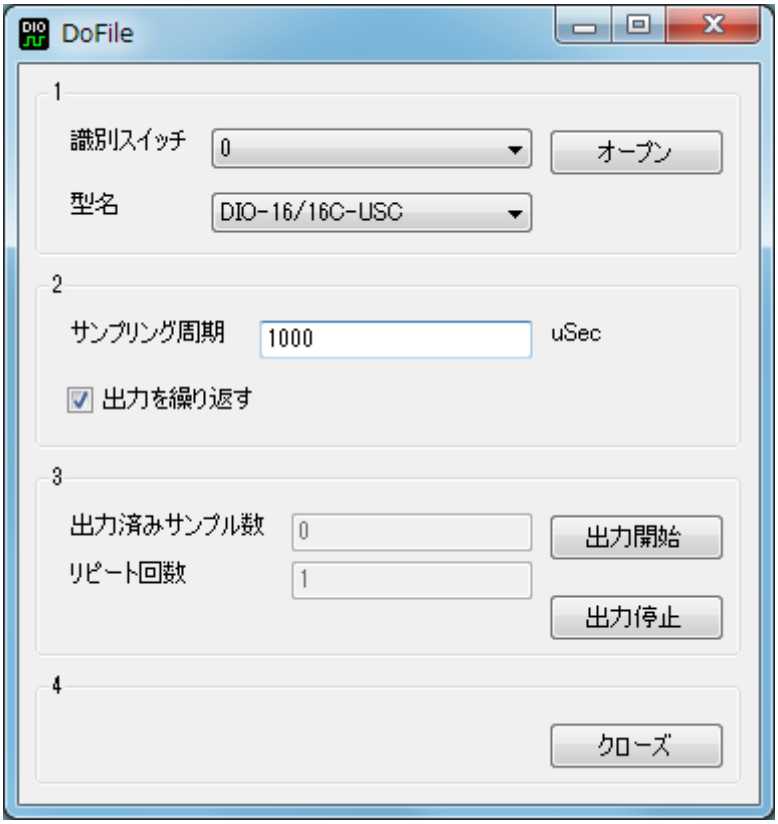
```
private: System::Void Form1_FormClosing(System::Object^ sender,
System::Windows::Forms::FormClosingEventArgs^ e)
```

```
{
    int result = YdxClose(id);
    if((result != 0) && (result != YDX_RESULT_NOT_OPEN))
    {
        ResultShow("YdxClose", result);
    }
}
```

DoFile

高機能デジタル出力 のサンプルプログラムです。
データをCSVファイルから読み出し、デジタル出力をおこないます。

画面



1. オープン

   ユニットのオープンをします。

2. 設定

   設定をします。

3. 出力開始／出力停止

   出力の開始および停止をします。

   「出力開始」ボタンがクリックされると、「動作条件の設定」→「動作開始」→「状態監視」という手順が実行されます。
   「出力を繰り返す」がチェックされていた場合、同じ波形を繰り返し出力します。
   「出力を繰り返す」がチェックされていなかった場合、波形を1回出力すると停止します。

4. クローズ

   ユニットのクローズをします。
   オープンをした場合は、必ず実行する必要があります。

備考

データは実行ファイルと同じフォルダにある「DoData.csv」です。
このファイルの内容を変更する事で、任意の波形を出力する事が可能になります。

サンプルソース

- C#

  ソフトウェアパック に付属しているサンプルプログラムのソースファイルを参照してください。

C/C++の開発環境の設定

VC（.NET2002以降）

1. 以下のファイルをプロジェクトフォルダにコピーします。

   Ydx.h
   Ydx.lib

2. Ydx.hをプロジェクトに追加します。

3. Ydx.libを以下の手順でプロジェクトに追加します。

   メニューの[プロジェクト]-[プロパティ]を選択し、プロパティページのダイアログを開きます。
   ダイアログの左ペインで[構成プロパティ]-[リンカ]-[入力]を選択します。
   右ペインの[追加の依存ファイル]にYdx.libと入力します。

4. ソースファイルにYdx.hをインクルードします。

VC6

1. 以下のファイルをプロジェクトフォルダにコピーします。

   Ydx.h
   Ydx.lib

2. Ydx.h, Ydx.libをプロジェクトに追加します。

3. ソースファイルにYdx.hをインクルードします。

実行手順

- 簡易デジタル入力

  デジタル入力を１回おこないたい場合
- 高機能デジタル入力

  連続サンプリングなど、色々な条件でデジタル入力をおこないたい場合
- 簡易デジタル出力

  デジタル出力を１回おこないたい場合
- 高機能デジタル出力

  連続サンプリングなど、色々な条件でデジタル出力をおこないたい場合

簡易デジタル入力

デジタル入力を１回おこないたい場合、簡易デジタル入力（YdxDiInput関数）を使用すると便利です。
実行手順は以下のとおりです。

準備

ユニットとパソコンをUSBケーブルで接続し、ユニットへ電源を供給してください。
（ユニットへ電源供給後、パソコンがユニットを認識するまでに数秒程度必要となる場合があります）

ユニットをオープン

YdxOpen関数 を使用してユニットをオープンします。

設定

YdxDiSetFilter関数 を使用して、デジタル入力フィルタを設定します。
初期値は、フィルタなしです。
（初期値のまま動作させる場合は、設定を省略する事が可能です）

入力

YdxDiInput関数 を使用して、デジタル入力をおこないます。

ボードのクローズ

YdxClose関数 を使用してボードをクローズします。

終了

ボードへの電源供給を止めます。

参考

- DioBit

  簡易デジタル入出力のサンプルプログラムです。
  デジタル入力には YdxDiInputBit関数 を使用しています。

高機能デジタル入力

準備

ユニットとパソコンをUSBケーブルで接続し、ユニットへ電源を供給してください。
（ユニットへ電源供給後、パソコンがユニットを認識するまでに数秒程度必要となる場合があります）

ユニットをオープン

YdxOpen関数 を使用してユニットをオープンします。

設定

各種設定をおこないます。
初期値のまま動作させる場合には省略する事が可能です。

- YdxDiSetBuffer関数 を使用して、データバッファ 形式を設定します。

  FIFOバッファ形式またはリングバッファ形式が選択できます。
  初期値は、FIFOバッファ形式です。
- YdxDiSetClock関数 を使用して、サンプリングクロック を設定します。

  設定されたサンプリングクロックのタイミングでデジタル入力がおこなわれます。
  内部クロックまたは外部クロックが選択できます。
  初期値は、内部クロック（周期1msec）です。
  - 内部クロック

    設定された周期でデジタル入力をおこないます。
    YdxDiSetClockInternal関数 を使用して、周期を設定します。
  - 外部クロック

    外部入力のタイミングでデジタル入力をおこないます。
    YdxDiSetClockExternal関数 を使用して、使用するデジタル入力チャネルと入力タイミングを設定します。
- YdxDiSetStartCondition関数 を使用して、サンプリング開始条件 を設定します。

  入力動作開始後、指定した開始条件を待ってからサンプリングがおこなわれます。
  ソフトウェア（自動）・外部トリガから選択できます。
  初期値は、ソフトウェア（自動）です。
  - ソフトウェア（自動）

    すぐにサンプリングを開始します。
  - 外部トリガ

    外部デジタル入力が、指定した状態になった時に、サンプリングを開始します。
    YdxDiSetStartExternal関数 で、使用するデジタル入力チャネルと入力タイミングを設定します。
- YdxDiSetStopCondition関数 を使用して、サンプリング停止条件 を設定します。

  サンプル数・外部トリガ・ソフトウェアから選択できます。
  初期値は、サンプル数（1000回）です。

- サンプル数

  指定した回数のサンプリングがおこなわれるとサンプリングを停止します。
  YdxDiSetStopSampleNum関数 を使用して、回数を指定します。

- 外部トリガ

  外部デジタル入力が、指定した状態になった時に、サンプリングを停止します。
  YdxDiSetStopExternal関数 で、使用するデジタル入力チャネルと入力タイミングを設定します。

- YdxDiSetRepeat関数 を使用して、リピート 回数を設定します。

  リピートとは、サンプリング開始条件（YdxDiSetStartCondition関数 で設定）からサンプリング停止条件（YdxDiSetStopCondition関数 で設定）までの動作を、繰り返しおこなう事です。
  初期値は、1回です。

入力動作の開始

以下のいずれかに該当する場合は、デジタル入力動作を開始する前に YdxDiClearData関数 を実行してください。

- データバッファに残っているデータを破棄したい場合
- 指定したリピート回数で動作終了済みの場合
- 状態（動作済みリピート回数） を0に戻したい場合

YdxDiStart関数 を使用して、デジタル入力動作を開始します。

動作状態の監視

YdxDiGetStatus関数 を使用して、動作状態を監視する事ができます。

入力動作の停止

設定された条件でのサンプリングが終了すると、デジタル入力動作は自動的に停止します。
エラー（サンプリングクロックエラー・オーバランエラー・ハードウェアエラー・通信エラー）が発生した場合も自動的に停止します。
また、YdxDiStop関数 を使用して停止する事も可能です。

データの取得

YdxDiGetData関数 を使用して、データを取得します。
データバッファがFIFO形式の場合は、動作中 もデータの取得が可能です。

ボードのクローズ

YdxClose関数 を使用してボードをクローズします。

終了

ボードへの電源供給を止めます。

参考

- 機能説明（高機能デジタル入力）
- サンプルプログラム（DiPolling）

  高機能デジタル入力のサンプルプログラムです。
  デジタル入力を1000回おこない、データを表示します。
  動作状態の監視をポーリングでおこなっています。

簡易デジタル出力

デジタル出力を１回おこないたい場合、簡易デジタル出力（YdxDoOutput関数）を使用すると便利です。
実行手順は以下のとおりです。

準備

ユニットとパソコンをUSBケーブルで接続し、ユニットへ電源を供給してください。
（ユニットへ電源供給後、パソコンがユニットを認識するまでに数秒程度必要となる場合があります）

ユニットのオープン

YdxOpen関数 を使用してユニットをオープンします。

出力

YdxDoOutput関数 を使用して、デジタル出力をおこないます。

ボードのクローズ

YdxClose関数 を使用してボードをクローズします。

終了

ボードへの電源供給を止めます。

参考

- DioBit

  簡易デジタル出力のサンプルプログラムです。
  デジタル出力には YdxDoOutputBit関数 を使用しています。

高機能デジタル出力

準備

ユニットとパソコンをUSBケーブルで接続し、ユニットへ電源を供給してください。
（ユニットへ電源供給後、パソコンがユニットを認識するまでに数秒程度必要となる場合があります）

ユニットのオープン

YdxOpen関数 を使用してユニットをオープンします。

設定

各種設定をおこないます。
初期値のまま動作させる場合には省略する事が可能です。

- YdxDoSetBuffer関数 を使用して、データバッファ 形式を設定します。

  FIFOバッファ形式またはリングバッファ形式が選択できます。
  初期値は、FIFOバッファ形式です。
- YdxDoSetChannel関数 を使用して、高機能デジタル出力モード（サンプリング出力をおこなう）として動作させるチャネルを設定します。

  初期値では、全てのチャネルが簡易デジタル出力モード（サンプリング出力をおこなわない）になっています。
- YdxDoSetClock関数 を使用して、サンプリングクロック を設定します。

  設定されたサンプリングクロックのタイミングでデジタル出力がおこなわれます。
  内部クロックまたは外部クロックが選択できます。
  初期値は、内部クロック（周期1msec）です。
  - 内部クロック

    設定された周期でデジタル出力をおこないます。
    YdxDoSetClockInternal関数 を使用して、周期を設定します。
  - 外部クロック

    外部出力のタイミングでデジタル出力をおこないます。
    YdxDoSetClockExternal関数 を使用して、使用するデジタル入力チャネルと入力タイミングを設定します。
- YdxDoSetStartCondition関数 を使用して、サンプリング開始条件 を設定します。

  出力動作開始後、指定した開始条件を待ってからサンプリングがおこなわれます。
  ソフトウェア（自動）・外部トリガから選択できます。
  初期値は、ソフトウェア（自動）です。
  - ソフトウェア（自動）

    すぐにサンプリングを開始します。
  - 外部トリガ

外部デジタル入力が、指定した状態になった時に、サンプリングを開始します。
YdxDoSetStartExternal関数 で、使用するデジタル入力チャネルと入力タイミングを設定します。

- YdxDoSetStopCondition関数 を使用して、サンプリング停止条件 を設定します。

  データ終了・外部トリガから選択できます。
  初期値は、データ終了です。

  - データ終了

    設定されたデータを全て出力するとサンプリングを停止します。

  - 外部トリガ

    外部デジタル入力が、指定した状態になった時に、サンプリングを停止します。
    YdxDoSetStopExternal関数 で、使用するデジタル入力チャネルと入力タイミングを設定します。

- YdxDoSetRepeat関数 を使用して、リピート 回数を設定します。

  リピートとは、サンプリング開始条件（YdxDoSetStartCondition関数 で設定）からサンプリング停止条件（YdxDoSetStopCondition関数 で設定）までの動作を、繰り返しおこなう事です。
  初期値は、無限です。

データの設定

以下のいずれかに該当する場合は、データを設定する前に YdxDoClearData関数 を実行してください。

- データバッファに残っているデータを破棄したい場合
- 指定したリピート回数で動作終了済みの場合
- 状態（動作済みリピート回数） を0に戻したい場合

ただしデータバッファ形式がリングバッファ形式に設定されている場合は実行不要です。
（データ設定時に自動的にクリアが実行される為）

YdxDoSetData関数 を使用して、データを設定します。
データバッファがFIFO形式の場合は、動作中 もデータの追加が可能です。

出力動作の開始

YdxDoStart関数 を使用して、デジタル出力動作を開始します。

動作状態の監視

YdxDoGetStatus関数 を使用して、動作状態を監視する事ができます。

出力動作の停止

設定された条件でのサンプリングが全て終了すると、デジタル出力動作は自動的に停止します。
エラー（サンプリングクロックエラー・ハードウェアエラー・通信エラー）が発生した場合も自動的に停止します。
また、YdxDoStop関数 を使用して停止する事も可能です。

ボードのクローズ

YdxClose関数 を使用してボードをクローズします。

終了

ボードへの電源供給を止めます。

参考

- 機能説明（高機能デジタル出力）
- DoPolling

  高機能デジタル出力のサンプルプログラムです。
  デジタル出力を1000回おこないます。
  動作状態の監視をポーリングでおこなっています。

戻り値一覧

戻り値 (16進数値/10進数値)	意味	対処方法
YDX_RESULT_SUCCESS (00000000h / 0)	正常終了	
YDX_RESULT_NOT_OPEN (CE000002h / -838860798)	オープンされていません	YdxOpen関数 にてオープンをしてください。
YDX_RESULT_ALREADY_OPEN (CE000003h / -838860797)	既にオープンされています	
YDX_RESULT_INVALID_ID (CE000004h / -838860796)	指定されたIDが不正です	YdxOpen関数 で取得したIDを指定してください。
YDX_RESULT_CANNOT_OPEN (CE000006h / -838860794)	オープンできませんでした	以下の原因などが考えられます。 1. 供給電源電圧が定格範囲を外れてしまっている。 （供給電源の電流容量不足による電圧低下など） 2. USBケーブルまたは電源ケーブルの接続不良 3. ユニット識別スイッチの設定が間違っている 4. デバイスドライバがインストールできていない 上記を確認しても問題が見当たらない場合は、どのような状況で発生したかを弊社サポートまでご連絡ください。
YDX_RESULT_MEM_ALLOC_ERROR (CE000009h / -838860791)	利用可能なメモリが不足しています	不要なアプリケーションを終了するなどして、利用可能なメモリを増やしてください。
YDX_RESULT_MODELNAME_ERROR (CE00000Ah / -838860790)	指定された型名は存在していないか、サポートしていません	型名に間違いがないか確認してください。
YDX_RESULT_HARDWARE_ERROR (CE00000Bh / -838860789)	ハードウェアにエラーが発生しました	ハードウェアが故障している可能性があります。 どのような状況で発生したかを弊社サポートまでご連絡ください。
YDX_RESULT_NOT_SUPPORTED (CE00000Ch / -838860788)	サポートされていない関数が実行されました	

戻り値 (16進数値/10進数値)	意味	対処方法
YDX_RESULT_INVALID_UNIT_SW (CE000020h / -838860768)	指定されたユニット識別スイッチの値が不正です	0～15を指定してください。
YDX_RESULT_UNIT_NUM_OVER (CE000021h / -838860767)	接続可能な台数を超えました	同時にオープンできるユニットは32台まで（同一機種は16台まで）です。
YDX_RESULT_DISCONNECT (CE000022h / -838860766)	通信が切断されました	以下の原因などが考えられます。 1. 動作中に供給電源電圧が定格範囲を外れてしまっている （供給電源の電流容量不足による電圧低下など） 2. USBケーブルまたは電源ケーブルの接続不良 3. パソコンがスリープ（スタンバイ）や休止状態になった 上記を確認しても問題が見当たらない場合は、どのような状況で発生したかを弊社サポートまでご連絡ください。 再び使用する場合は、YdxClose関数 でクローズをしてから、YdxOpen関数 でオープンをしなおしてください。
YDX_RESULT_COMMUNICATE_ERROR (CE000030h / -838860752)	通信エラーが発生しました	USB通信に異常が発生しました。 近くにノイズ要因がないか確認してください。 確認しても問題が見当たらない場合は、どのような状況で発生したかを弊社サポートまでご連絡ください。
YDX_RESULT_COMMUNICATE_TIMEOUT (CE00003Fh / -838860737)	通信タイムアウトが発生しました	
YDX_RESULT_PARAMETER1_ERROR (CE000041h / -838860735)	引数1が不正です	関数仕様やサンプルプログラムを参照し、引数の確認をしてください。
YDX_RESULT_PARAMETER2_ERROR (CE000042h / -838860734)	引数2が不正です	
YDX_RESULT_PARAMETER3_ERROR (CE000043h / -838860733)	引数3が不正です	
YDX_RESULT_PARAMETER4_ERROR (CE000044h / -838860732)	引数4が不正です	
YDX_RESULT_PARAMETER5_ERROR (CE000045h / -838860731)	引数5が不正です	
YDX_RESULT_PARAMETER1_NULL (CE000049h / -838860727)	引数1がNULLです	

戻り値 (16進数値/10進数値)	意味	対処方法
YDX_RESULT_PARAMETER2_NULL (CE00004Ah / -838860726)	引数2がNULLです	
YDX_RESULT_PARAMETER3_NULL (CE00004Bh / -838860725)	引数3がNULLです	
YDX_RESULT_PARAMETER4_NULL (CE00004Ch / -838860724)	引数4がNULLです	
YDX_RESULT_PARAMETER5_NULL (CE00004Dh / -838860723)	引数5がNULLです	
YDX_RESULT_DI_BUSY (CE0000A0h / -838860640)	デジタル入力が 動作中 です	
YDX_RESULT_DI_ERROR (CE0000A1h / -838860639)	デジタル入力動作 はエラー停止しています	YdxDiReset関数 を実行して、エラーを解除してください。
YDX_RESULT_DI_END (CE0000A2h / -838860638)	デジタル入力動作は終了しています	指定されたリピート回数の動作は終了しています。 同じ条件で再度動作させたい場合は、YdxDiClearData関数 を実行して動作済みリピート回数をクリアしてください。
YDX_RESULT_DI_EXCEED_DATA_NUM (CE0000A8h / -838860632)	変換されたデータ数を超えるデータを取得しようとしました。 sampleNumを最大値にしてデータを取得しました	
YDX_RESULT_DI_EXCEED_BUF_SIZ (CE0000A9h / -838860631)	データバッファ容量を超えるデータを取得しようとしました。 sampleNumを最大値にしてデータを取得しました	
YDX_RESULT_DO_BUSY (CE0000B0h / -838860624)	デジタル出力が 動作中 です	
YDX_RESULT_DO_ERROR (CE0000B1h / -838860623)	デジタル出力動作 はエラー停止しています	YdxDoReset関数 を実行して、エラーを解除してください。

戻り値 (16進数値/10進数値)	意味	対処方法
YDX_RESULT_DO_END (CE0000B2h / -838860622)	デジタル出力動作は終了しています	指定されたリピート回数の動作は終了しています。 同じ条件で再度動作させたい場合は、YdxDoClearData関数 を実行して動作済みリピート回数をクリアしてください。
YDX_RESULT_DO_NO_CHANNEL (CE0000B3h / -838860621)	デジタル出力チャネルが設定されていません	YdxDoSetChannel関数 で、高機能デジタル出力をおこなうチャネルを指定してください。
YDX_RESULT_DO_NO_DATA (CE0000B8h / -838860616)	データが設定されていません	YdxDoSetData関数 で、データを設定してください。
YDX_RESULT_DO_EXCEED_BUF_SIZ (CE0000B9h / -838860615)	データバッファ容量を超えるデータを設定しようとしました	
YDX_RESULT_FATAL_ERROR (CEFFFFFFh / -822083585)	致命的なエラー	どのような状況で発生したかを弊社サポートまでご連絡ください。

C#での注意事項

C#で関数を使用する場合、Ydxの後に「.」（ドット）を付加して使用します。
（サンプルプログラムも参照してください）

C#以外	C#
YdxOpen	Ydx.Open
YdxClose	Ydx.Close

基本関数一覧

関数	機能
YdxOpen	ユニットへのアクセスが出来るようにします。
YdxClose	ユニットへのアクセスを終了します。
YdxCnvResultToString	関数戻り値の内容を文字列に変換します。

YdxOpen

機能

ユニットへのアクセスが出来るようにします。

書式

```
INT YdxOpen(
    INT unitSwitch,
    const char* modelName,
    INT mode,
    INT* id
);
```

パラメータ

unitSwitch

オープンするユニットのユニット識別スイッチ番号を指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

modelName

オープンするユニットの型名を指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	string	String	String	String^	const char*

mode

オープン時の動作を指定します。

値	意味
0	デジタル出力を全てOFFにします。
1	デジタル出力の状態を変えません。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

id

IDを格納する変数へのポインタを指定します。
以降はこの変数に格納された値を使用して関数へアクセスします。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

> ⚠ YdxOpen関数でオープンしたボードは、アプリケーション終了時に必ず YdxClose関数 でクローズしてください。

使用例

ユニット識別スイッチ「0」・型名「DIO-16/16C-USC」のユニットをオープンします。

C#

```
int result;
int id;
result = Ydx.Open(0, "DIO-16/16C-USC", 0, out id);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim id As Integer
result = YdxOpen(0, "DIO-16/16C-USC", 0, id)
```

VB6.0

```
Dim result As Long
Dim id As Long
result = YdxOpen(0, "DIO-16/16C-USC", 0, id)
```

C++/CLI

```
int result;
int id;
result = YdxOpen(0, "DIO-16/16C-USC", 0, &id);
```

C/C++

```
INT result;
INT id;
result = YdxOpen(0, "DIO-16/16C-USC", 0, &id);
```

YdxClose

機能

ユニットへのアクセスを終了します。

書式

```
INT YdxClose(
    INT id
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

> ⚠ YdxOpen関数 でオープンしたユニットは、アプリケーション終了時に必ずYdxClose関数でクローズしてください。

使用例

ユニットをクローズします。

C#

```
int result;
result = Ydx.Close(id);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxClose(id)
```

VB6.0

```
Dim result As Long
result = YdxClose(id)
```

C++/CLI

```
int result;
result = YdxClose(id);
```

C/C++

```
INT result;
result = YdxClose(id);
```

YdxCnvResultToString

機能

関数戻り値の内容を文字列に変換します。

書式

```
INT YdxCnvResultToString(
    INT resultCode,
    char* resultString
);
```

パラメータ

resultCode

戻り値を指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

resultString

文字列を格納する変数へのポインタを指定します。
バッファは256バイト確保してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out string	StringBuilder	String	StringBuilder^	char*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

使用例

戻り値「CE000002h」の内容を文字列に変換します。
resultStringには「オープンされていません」が格納されます。

C#

```
int result;
string resultString;
result = Ydx.CnvResultToString(unchecked((int)0xCE000002), out resultString);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim resultString As New StringBuilder(256)
result = YdxCnvResultToString(&HCE000002, resultString)
```

VB6.0

```
Dim result As Long
Dim resultString As String
result = YdxCnvResultToString(&HCE000002, resultString)
```

C++/CLI

```
int result;
StringBuilder ^resultString = gcnew StringBuilder(256);
result = YdxCnvResultToString(0xCE000002, resultString);
```

C/C++

```
INT result;
char resultString[256];
result = YdxCnvResultToString(0xCE000002, resultString);
```

デジタル入力関数一覧

簡易デジタル入力 ・ 高機能デジタル入力　共用関数

関数	機能
YdxDiSetFilter	デジタル入力フィルタを設定します。
YdxDiGetFilter	デジタル入力フィルタの設定を取得します。

簡易デジタル入力　専用関数

関数	機能
YdxDiInput	デジタル入力端子の状態を読み込みます。
YdxDiInputBit	デジタル入力端子の状態を読み込みます。 データは、ビットごとに読み込みます。

高機能デジタル入力　専用関数

設定

関数	機能
YdxDiSetBuffer	データバッファ の形式を設定します。
YdxDiSetCheckSampleNum	監視サンプル数 を設定します。
YdxDiSetClock	サンプリングクロック の種類を設定します。
YdxDiSetClockInternal	内部クロックを設定します。
YdxDiSetClockExternal	外部クロックを設定します。
YdxDiSetRepeat	リピート を設定します。
YdxDiSetStartCondition	サンプリング開始条件 を設定します。
YdxDiSetStartExternal	サンプリング開始条件（外部トリガ）を設定します。

関数	機能
YdxDiSetStopCondition	サンプリング停止条件 を設定します。
YdxDiSetStopSampleNum	サンプリング停止条件（サンプル数）を設定します。
YdxDiSetStopExternal	サンプリング停止条件（外部トリガ）を設定します。

設定の取得

関数	機能
YdxDiGetBuffer	データバッファの設定を取得します。
YdxDiGetCheckSampleNum	監視サンプル数の設定を取得します。
YdxDiGetClock	サンプリングクロックの設定を取得します。
YdxDiGetClockInternal	内部クロックの設定を取得します。
YdxDiGetClockExternal	外部クロックの設定を取得します。
YdxDiGetRepeat	リピートの設定を取得します。
YdxDiGetStartCondition	サンプリング開始条件の設定を取得します。
YdxDiGetStartExternal	サンプリング開始条件（外部トリガ）の設定を取得します。
YdxDiGetStopCondition	サンプリング停止条件の設定を取得します。
YdxDiGetStopSampleNum	サンプリング停止条件（サンプル数）の設定を取得します。
YdxDiGetStopExternal	サンプリング停止条件（外部トリガ）の設定を取得します。

動作の開始・停止

関数	機能
YdxDiStart	デジタル入力動作 を開始します。
YdxDiStop	デジタル入力動作を停止します。

リセット

関数	機能
YdxDiReset	デジタル入力機能をリセットします。

状態の取得

関数	機能
YdxDiGetStatus	現在の状態を取得します。

データの取得・クリア

関数	機能
YdxDiGetData	データを取得します。
YdxDiClearData	データをクリアします。

YdxDiSetFilter

機能

デジタル入力フィルタを設定します。

書式

```
INT YdxDiSetFilter(
    INT id,
    INT channel,
    INT coefficient
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

channel

フィルタの設定をするチャネルを指定します。
設定範囲は-1～15です。
-1に設定した場合は、全てのチャネルとなります。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

coefficient

フィルタ時間を指定します。
指定した時間未満のパルスは除去されます。
ただし、指定した時間の遅延が発生します。
単位は「μsec」です。
設定範囲は0～65,535 [μsec]、初期値は0です。
0に設定した場合は、フィルタなしとなります。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

使用例

デジタル入力チャネル2（IN2）に、10msecのフィルタを設定します。

C#

```
int result;
result = Ydx.DiSetFilter(id, 2, 10000);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxDiSetFilter(id, 2, 10000)
```

VB6.0

```
Dim result As Long
result = YdxDiSetFilter(id, 2, 10000)
```

C++/CLI

```
int result;
result = YdxDiSetFilter(id, 2, 10000);
```

C/C++

```
INT result;
result = YdxDiSetFilter(id, 2, 10000);
```

YdxDiGetFilter

機能

デジタル入力フィルタの設定を取得します。

書式

```
INT YdxDiGetFilter(
    INT id,
    INT channel,
    INT* coefficient
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

channel

設定を取得するチャネルを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

coefficient

設定値を格納する変数へのポインタを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。

詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

パラメータの詳細については、YdxDiSetFilter関数 を参照してください。

使用例

デジタル入力チャネル2（IN2）の、フィルタ設定値を取得します。

C#

```
int result;  int coefficient;
result = Ydx.DiGetFilter(id, 2, out coefficient);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim coefficient As Integer
result = YdxDiGetFilter(id, 2, coefficient)
```

VB6.0

```
Dim result As Long
Dim coefficient As Long
result = YdxDiGetFilter(id, 2, coefficient)
```

C++/CLI

```
int result;
int coefficient;
result = YdxDiGetFilter(id, 2, &coefficient);
```

C/C++

```
INT result;
INT coefficient;
result = YdxDiGetFilter(id, 2, &coefficient);
```

YdxDiInput

機能

デジタル入力端子の状態を読み込みます。

書式

```
INT YdxDiInput(
    INT id,
    INT mode,
    INT* data
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

mode

デジタル入力フィルタ通過前と通過後どちらの状態を読み出すかを選択します。

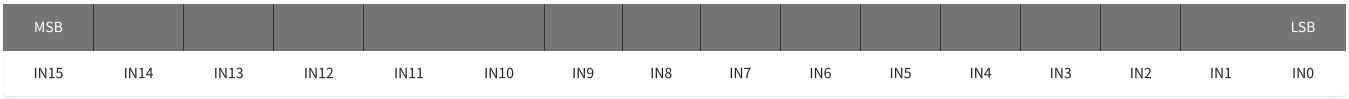
値	意味
0	フィルタ通過後の状態
1	フィルタ通過前の状態

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

data

入力データを格納する変数へのポインタを指定します。
データは、以下のフォーマットで格納されます。

MSB														LSB



言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

デジタル入力フィルタは、YdxDiSetFilter関数 で設定できます。

使用例

デジタル入力端子の状態（デジタル入力フィルタ通過前の状態）を読み込みます。

C#

```
int result;
int data;
result = Ydx.DiInput(id, 1, out data);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim data As Integer
result = YdxDiInput(id, 1, data)
```

VB6.0

```
Dim result As Long
Dim data As Long
result = YdxDiInput(id, 1, data)
```

C++/CLI

```
int result;
int data;
result = YdxDiInput(id, 1, &data);
```

C/C++

```
INT result;
INT data;
result = YdxDiInput(id, 1, &data);
```

YdxDiInputBit

機能

デジタル入力端子の状態を読み込みます。
データは、ビットごとに読み込みます。

書式

```
INT YdxDiInputBit(
    INT id,
    INT start,
    INT num,
    INT mode,
    INT* data
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

start

入力開始チャネルを指定します。
設定範囲は0～15です。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

num

読み込みをするチャネル数を指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

mode

デジタル入力フィルタ通過前と通過後どちらの状態を読み出すかを選択します。

値	意味
0	フィルタ通過後の状態
1	フィルタ通過前の状態

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

data

入力データを格納する変数へのポインタを指定します。
関数が正常に実行されると、入力データが格納されます。

値	意味
0	OFF
1	ON

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int[]	Integer	Long	int*	INT*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

デジタル入力フィルタは、YdxDiSetFilter関数 で設定できます。

使用例

デジタル入力チャネル2～5（IN2～IN5）の端子の状態（デジタル入力フィルタ通過前の状態）を読み込みます。
データはIN2から順にバッファへ格納されます。

C#

```
int result;
int[] data = new int[4];
result = Ydx.DiInputBit(id, 2, 4, 1, data);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim data(3) As Integer
result = YdxDiInputBit(id, 2, 4, 1, data)
```

VB6.0

```
Dim result As Long
Dim data(3) As Long
result = YdxDiInputBit(id, 2, 4, 1, data(0))
```

C++/CLI

```
int result;
int data[4];
result = YdxDiInputBit(id, 2, 4, 1, data);
```

C/C++

```
INT result;
INT data[4];
result = YdxDiInputBit(id, 2, 4, 1, data);
```

YdxDiSetBuffer

機能

データバッファ の形式を設定します。

書式

```
INT YdxDiSetBuffer(
    INT id,
    INT bufferType
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

bufferType

データバッファの形式を指定します。

値	意味	説明
0	FIFOバッファ形式	読み出しは、古いデータから順におこなわれます。 読み出されたデータは、バッファから破棄されます。 動作中 にデータを読み出す事が可能です。 読み出されていないデータがバッファに満杯の状態でサンプリングがおこなわれるとオーバランエラーが発生します。
1	リングバッファ形式	読み出しは、新しいデータからおこなわれます。 読み出されたデータは、バッファから破棄されません。 （再度読み出す事が可能） 動作中 にデータを読み出す事はできません。 読み出されていないデータがバッファに満杯の状態でサンプリングがおこなわれると古いデータに上書きして記憶されます。

初期値は0です。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

> ⚠ データバッファにデータが残った状態のまま、本関数により設定を変更した場合、データはクリアされます。

本関数は、デジタル入力が 動作中 には実行できません。

使用例

データバッファを、リングバッファ形式に設定します。

C#

```
int result;
result = Ydx.DiSetBuffer(id, 1);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxDiSetBuffer(id, 1)
```

VB6.0

```
Dim result As Long
result = YdxDiSetBuffer(id, 1)
```

C++/CLI

```
int result;
result = YdxDiSetBuffer(id, 1);
```

C/C++

```
INT result;
result = YdxDiSetBuffer(id, 1);
```

YdxDiSetCheckSampleNum

機能

監視サンプル数 を設定します。

書式

```
INT YdxDiSetCheckSampleNum(
    INT id,
    INT sampleNum
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

sampleNum

監視サンプル数を指定します。
設定範囲は1～2,147,483,647、初期値は500です。

データバッファのデータが、監視サンプル数以上になった場合、以下の動作となります。

- YdxDiGetStatus関数 で、ステータスを読み出した時、監視サンプル数ビットがオンになります。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

本関数は、デジタル入力が 動作中 には実行できません。

使用例

監視サンプル数を、2000に設定します。

C#

```
int result;
result = Ydx.DiSetCheckSampleNum(id, 2000);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxDiSetCheckSampleNum(id, 2000)
```

VB6.0

```
Dim result As Long
result = YdxDiSetCheckSampleNum(id, 2000)
```

C++/CLI

```
int result;
result = YdxDiSetCheckSampleNum(id, 2000);
```

C/C++

```
INT result;
result = YdxDiSetCheckSampleNum(id, 2000);
```

YdxDiSetClock

機能

サンプリングクロック の種類を設定します。

書式

```
INT YdxDiSetClock(
    INT id,
    INT clockType
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

clockType

クロックの種類を指定します。

値	意味
0	内部クロック
1	外部クロック

初期値は0です。
「内部クロック」を指定する場合、YdxDiSetClockInternal関数 で、周期の設定をしてください。
「外部クロック」を指定する場合、YdxDiSetClockExternal関数 で、使用するデジタル入力チャネルと入力タイミングの設定をしてください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

本関数は、デジタル入力が 動作中 には実行できません。

使用例

サンプリングクロックとして、外部クロックを使用します。

C#

```
int result;
result = Ydx.DiSetClock(id, 1);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxDiSetClock(id, 1)
```

VB6.0

```
Dim result As Long
result = YdxDiSetClock(id, 1)
```

C++/CLI

```
int result;
result = YdxDiSetClock(id, 1);
```

C/C++

```
INT result;
result = YdxDiSetClock(id, 1);
```

YdxDiSetClockInternal

機能

内部クロックを設定します。

書式

```
INT YdxDiSetClockInternal(
    INT id,
    double period
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

period

周期を指定します。
単位は「μsec」です。
設定範囲は0.1～60,000,000 [μsec]、初期値は1,000 [μsec]です。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	double	Double	Double	double	double

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

YdxDiSetClock関数 で、クロックの種類として「内部クロック」を選択した場合にのみ設定が有効になります。
クロックの種類として「内部クロック」を選択しない場合は、本関数を実行する必要はありません。
入力回路の応答時間については機種によって異なりますので、取扱説明書を参照してください。

本関数は、デジタル入力が 動作中 には実行できません。

使用例

内部クロックを、2msec周期に設定します。

C#

```
int result;
result = Ydx.DiSetClockInternal(id, 2000);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxDiSetClockInternal(id, 2000)
```

VB6.0

```
Dim result As Long
result = YdxDiSetClockInternal(id, 2000)
```

C++/CLI

```
int result;
result = YdxDiSetClockInternal(id, 2000);
```

C/C++

```
INT result;
result = YdxDiSetClockInternal(id, 2000);
```

YdxDiSetClockExternal

機能

外部クロックを設定します。

書式

```
INT YdxDiSetClockExternal(
    INT id,
    INT diChannel,
    INT edge
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

diChannel

外部クロックとして使用するデジタル入力チャネルを指定します。
設定範囲は0～15、初期値は0です。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

edge

入力タイミングを指定します。

値	意味
0	立ち上がりエッジセンス（OFF→ON）
1	立ち下がりエッジセンス（ON→OFF）
2	両エッジセンス（OFF→ON と ON→OFF の両方）

初期値は0です。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

YdxDiSetClock関数 で、クロックの種類として「外部クロック」を選択した場合にのみ設定が有効になります。
クロックの種類として「外部クロック」を選択しない場合は、本関数を実行する必要はありません。

本関数は、デジタル入力が 動作中 には実行できません。

使用例

外部クロックを、デジタル入力チャネル2（IN2）の立ち下りエッジに設定します。

C#

```
int result;
result = Ydx.DiSetClockExternal(id, 2, 1);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxDiSetClockExternal(id, 2, 1)
```

VB6.0

```
Dim result As Long
result = YdxDiSetClockExternal(id, 2, 1)
```

C++/CLI

```
int result;
result = YdxDiSetClockExternal(id, 2, 1);
```

C/C++

```
INT result;
result = YdxDiSetClockExternal(id, 2, 1);
```

YdxDiSetRepeat

機能

リピート を設定します。

書式

```
INT YdxDiSetRepeat(
    INT id,
    INT repeatNum
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

repeatNum

リピート回数を設定します。
設定範囲は0～2,147,483,647[回]、初期値は1[回]です。
0に設定した場合は、リピート回数は無限となります。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

リピートとは、サンプリング開始条件（YdxDiSetStartCondition関数 で設定）からサンプリング停止条件（YdxDiSetStopCondition関数 で設定）までの動作を、繰り返しおこなう事です。

本関数は、デジタル入力が 動作中 には実行できません。

使用例

リピートを、10回に設定します。

C#

```
int result;
result = Ydx.DiSetRepeat(id, 10);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxDiSetRepeat(id, 10)
```

VB6.0

```
Dim result As Long
result = YdxDiSetRepeat(id, 10)
```

C++/CLI

```
int result;
result = YdxDiSetRepeat(id, 10);
```

C/C++

```
INT result;
result = YdxDiSetRepeat(id, 10);
```

YdxDiSetStartCondition

機能

サンプリング開始条件 を設定します。

書式

```
INT YdxDiSetStartCondition(
    INT id,
    INT condition,
    INT delay
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

condition

サンプリング開始条件を指定します。

値	意味
0	自動（ソフトウェア）
1	外部トリガ

初期値は0です。
「外部トリガ」を指定する場合、YdxDiSetStartExternal関数 で、使用するデジタル入力チャネルとモードの設定をしてください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

delay

遅延回数を指定します。
本機種では0（遅延なし）しか設定できません。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

本関数は、デジタル入力が 動作中 には実行できません。

使用例

サンプリング開始条件を、外部トリガに設定します。

C#

```
int result;
result = Ydx.DiSetStartCondition(id, 1, 0);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxDiSetStartCondition(id, 1, 0)
```

VB6.0

```
Dim result As Long
result = YdxDiSetStartCondition(id, 1, 0)
```

C++/CLI

```
int result;
result = YdxDiSetStartCondition(id, 1, 0);
```

C/C++

```
INT result;
result = YdxDiSetStartCondition(id, 1, 0);
```

YdxDiSetStartExternal

機能

サンプリング開始条件（外部トリガ）を設定します。

書式

```
INT YdxDiSetStartExternal(
    INT id,
    INT diChannel,
    INT mode
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

diChannel

外部トリガとして使用するデジタル入力チャネルを指定します。
設定範囲は0～15、初期値は1です。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

mode

動作モードを指定します。

値	意味
0	立ち上がりエッジセンス（OFF→ONに変化した時に、条件成立）
1	立ち下がりエッジセンス（ON→OFFに変化した時に、条件成立）
2	両エッジセンス（ON→OFF または OFF→ONに変化した時に、条件成立）

値	意味
3	ハイレベルセンス（ONの時に、条件成立。最初からONだった場合も、条件成立）
4	ローレベルセンス（OFFの時に、条件成立。最初からOFFだった場合も、条件成立）

初期値は0です。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

YdxDiSetStartCondition関数 で、サンプリング開始条件として「外部トリガ」を選択した場合にのみ設定が有効になります。
サンプリング開始条件として「外部トリガ」を選択しない場合は、本関数を実行する必要はありません。

本関数は、デジタル入力が 動作中 には実行できません。

使用例

サンプリング開始条件（外部トリガ）を設定します。
外部トリガとして使用するデジタル入力チャネルはチャネル1、動作モードは両エッジセンスに設定します。

C#

```
int result;
result = Ydx.DiSetStartExternal(id, 1, 2);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxDiSetStartExternal(id, 1, 2)
```

VB6.0

```
Dim result As Long
result = YdxDiSetStartExternal(id, 1, 2)
```

C++/CLI

```
int result;
result = YdxDiSetStartExternal(id, 1, 2);
```

C/C++

```
INT result;
result = YdxDiSetStartExternal(id, 1, 2);
```

YdxDiSetStopCondition

機能

サンプリング停止条件 を設定します。

書式

```
INT YdxDiSetStopCondition(
    INT id,
    INT condition,
    INT delay
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

condition

サンプリング停止条件を指定します。

値	意味
0	サンプル数
1	外部トリガ
2	ソフトウェア（YdxDiStop関数）

初期値は0です。
* 「サンプル数」を指定する場合、YdxDiSetStopSampleNum関数 で、回数の設定をしてください。
* 「外部トリガ」を指定する場合、YdxDiSetStopExternal関数 で、使用するデジタル入力チャネルとモードの設定をしてください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

delay

遅延回数を指定します。
本機種では0（遅延なし）しか設定できません。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

本関数は、デジタル入力が 動作中 には実行できません。

使用例

サンプリング停止条件を、外部トリガに設定します。

C#

```
int result;
result = Ydx.DiSetStopCondition(id, 1, 0);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxDiSetStopCondition(id, 1, 0)
```

VB6.0

```
Dim result As Long
result = YdxDiSetStopCondition(id, 1, 0)
```

C++/CLI

```
int result;
result = YdxDiSetStopCondition(id, 1, 0);
```

C/C++

```
INT result;
result = YdxDiSetStopCondition(id, 1, 0);
```

YdxDiSetStopSampleNum

機能

サンプリング停止条件（サンプル数）を設定します。

書式

```
INT YdxDiSetStopSampleNum(
    INT id,
    INT sampleNum
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

sampleNum

サンプル数を指定します。
設定範囲は1～2,147,483,647[回]、初期値は1,000[回]です。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

YdxDiSetStopCondition関数 で、サンプリング停止条件として「サンプル数」を選択した場合にのみ設定が有効になります。
サンプリング停止条件として「サンプル数」を選択しない場合は、本関数を実行する必要はありません。

本関数は、デジタル入力が 動作中 には実行できません。

使用例

サンプリング停止条件（サンプル数）を設定します。
サンプル数は1200回に設定します。

C#

```
int result;
result = Ydx.DiSetStopSampleNum(id, 1200);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxDiSetStopSampleNum(id, 1200)
```

VB6.0

```
Dim result As Long
result = YdxDiSetStopSampleNum(id, 1200)
```

C++/CLI

```
int result;
result = YdxDiSetStopSampleNum(id, 1200);
```

C/C++

```
INT result;
result = YdxDiSetStopSampleNum(id, 1200);
```

YdxDiSetStopExternal

機能

サンプリング停止条件（外部トリガ）を設定します。

書式

```
INT YdxDiSetStopExternal(
    INT id,
    INT diChannel,
    INT mode
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

diChannel

外部トリガとして使用するデジタル入力チャネルを指定します。
設定範囲は0～15、初期値は2です。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

mode

動作モードを指定します。

値	意味
0	立ち上がりエッジセンス（OFF→ONに変化した時に、条件成立）
1	立ち下がりエッジセンス（ON→OFFに変化した時に、条件成立）
2	両エッジセンス（OFF→ON または ON→OFFに変化した時に、条件成立）

値	意味
3	ハイレベルセンス（ONの時に、条件成立。最初からONだった場合も、条件成立）
4	ローレベルセンス（OFFの時に、条件成立。最初からOFFだった場合も、条件成立）

初期値は0です。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

YdxDiSetStopCondition関数 で、サンプリング停止条件として「外部トリガ」を選択した場合にのみ設定が有効になります。
サンプリング停止条件として「外部トリガ」を選択しない場合は、本関数を実行する必要はありません。

本関数は、デジタル入力が 動作中 には実行できません。

使用例

サンプリング停止条件（外部トリガ）を設定します。
外部トリガとして使用するデジタル入力チャネルはチャネル1、動作モードは両エッジセンスに設定します。

C#

```
int result;
result = Ydx.DiSetStopExternal(id, 1, 2);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxDiSetStopExternal(id, 1, 2)
```

VB6.0

```
Dim result As Long
result = YdxDiSetStopExternal(id, 1, 2)
```

C++/CLI

```
int result;
result = YdxDiSetStopExternal(id, 1, 2);
```

C/C++

```
INT result;
result = YdxDiSetStopExternal(id, 1, 2);
```

YdxDiGetBuffer

機能

データバッファの設定を取得します。

書式

```
INT YdxDiGetBuffer(
    INT id,
    INT* bufferType
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

bufferType

データバッファの形式を格納する変数へのポインタを指定します。

値	意味
0	FIFOバッファ形式
1	リングバッファ形式

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

パラメータの詳細については、YdxDiSetBuffer関数 を参照してください。

使用例

データバッファの設定を取得します。

C#

```
int result;
int bufferType;
result = Ydx.DiGetBuffer(id, out bufferType);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim bufferType As Integer
result = YdxDiGetBuffer(id, bufferType)
```

VB6.0

```
Dim result As Long
Dim bufferType As Long
result = YdxDiGetBuffer(id, bufferType)
```

C++/CLI

```
int result;
int bufferType;
result = YdxDiGetBuffer(id, &bufferType);
```

C/C++

```
INT result;
INT bufferType;
result = YdxDiGetBuffer(id, &bufferType);
```

YdxDiGetCheckSampleNum

機能

監視サンプル数の設定を取得します。

書式

```
INT YdxDiGetCheckSampleNum(
    INT id,
    INT* sampleNum
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

sampleNum

監視サンプル数を格納する変数へのポインタを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

パラメータの詳細については、YdxDiSetCheckSampleNum関数 を参照してください。

使用例

監視サンプル数の設定を取得します。

C#

```
int result;
int sampleNum;
result = Ydx.DiGetCheckSampleNum(id, out sampleNum);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim sampleNum As Integer
result = YdxDiGetCheckSampleNum(id, sampleNum)
```

VB6.0

```
Dim result As Long
Dim sampleNum As Long
result = YdxDiGetCheckSampleNum(id, sampleNum)
```

C++/CLI

```
int result;
int sampleNum;
result = YdxDiGetCheckSampleNum(id, &sampleNum);
```

C/C++

```
INT result;
INT sampleNum;
result = YdxDiGetCheckSampleNum(id, &sampleNum);
```

YdxDiGetClock

機能

サンプリングクロックの設定を取得します。

書式

```
INT YdxDiGetClock(
    INT id,
    INT* clockType
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

clockType

クロックの種類を格納する変数へのポインタを指定します。

値	意味
0	内部クロック
1	外部クロック

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

パラメータの詳細については、YdxDiSetClock関数 を参照してください。

使用例

サンプリングクロックの設定を取得します。

C#

```
int result;
int clockType;
result = Ydx.DiGetClock(id, out clockType);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim clockType As Integer
result = YdxDiGetClock(id, clockType)
```

VB6.0

```
Dim result As Long
Dim clockType As Long
result = YdxDiGetClock(id, clockType)
```

C++/CLI

```
int result;
int clockType;
result = YdxDiGetClock(id, &clockType);
```

C/C++

```
INT result;
INT clockType;
result = YdxDiGetClock(id, &clockType);
```

YdxDiGetClockInternal

機能

内部クロックの設定を取得します。

書式

```
INT YdxDiGetClockInternal(
    INT id,
    double* period
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

period

周期を格納する変数へのポインタを指定します。
単位は「μsec」です。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out double	Double	Double	double*	double*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

パラメータの詳細については、YdxDiSetClockInternal関数 を参照してください。

使用例

内部クロックの設定を取得します。

C#

```
int result;
double period;
result = Ydx.DiGetClockInternal(id, out period);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim period As Double
result = YdxDiGetClockInternal(id, period)
```

VB6.0

```
Dim result As Long
Dim period As Double
result = YdxDiGetClockInternal(id, period)
```

C++/CLI

```
int result;
double period;
result = YdxDiGetClockInternal(id, &period);
```

C/C++

```
INT result;
double period;
result = YdxDiGetClockInternal(id, &period);
```

YdxDiGetClockExternal

機能

外部クロックの設定を取得します。

書式

```
INT YdxDiGetClockExternal(
    INT id,
    INT* diChannel,
    INT* edge
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

diChannel

外部クロックとして使用するデジタル入力チャネルを格納する変数へのポインタを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

edge

入力タイミングを格納する変数へのポインタを指定します。

値	意味
0	立ち上がりエッジセンス（OFF→ON）
1	立ち下がりエッジセンス（ON→OFF）
2	両エッジセンス（OFF→ON と ON→OFF の両方）

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

パラメータの詳細については、YdxDiSetClockExternal関数 を参照してください。

使用例

外部クロックの設定を取得します。

C#

```
int result;
int diChannel;
int edge;
result = Ydx.DiGetClockExternal(id, out diChannel, out edge);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim diChannel As Integer
Dim edge As Integer
result = YdxDiGetClockExternal(id, diChannel, edge)
```

VB6.0

```
Dim result As Long
Dim diChannel As Long
Dim edge As Long
result = YdxDiGetClockExternal(id, diChannel, edge)
```

C++/CLI

```
int result;
int diChannel;
int edge;
result = YdxDiGetClockExternal(id, &diChannel, &edge);
```

C/C++

```
INT result;
INT diChannel;
INT edge;
result = YdxDiGetClockExternal(id, &diChannel, &edge);
```

YdxDiGetRepeat

機能

リピートの設定を取得します。

書式

```
INT YdxDiGetRepeat(
    INT id,
    INT* repeatNum
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

repeatNum

リピート設定回数を格納する変数へのポインタを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

パラメータの詳細については、YdxDiSetRepeat関数 を参照してください。

使用例

リピートの設定を取得します。

C#

```
int result;
int repeatNum;
result = Ydx.DiGetRepeat(id, out repeatNum);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim repeatNum As Integer
result = YdxDiGetRepeat(id, repeatNum)
```

VB6.0

```
Dim result As Long
Dim repeatNum As Long
result = YdxDiGetRepeat(id, repeatNum)
```

C++/CLI

```
int result;
int repeatNum;
result = YdxDiGetRepeat(id, &repeatNum);
```

C/C++

```
INT result;
INT repeatNum;
result = YdxDiGetRepeat(id, &repeatNum);
```

YdxDiGetStartCondition

機能

サンプリング開始条件の設定を取得します。

書式

```
INT YdxDiGetStartCondition(
    INT id,
    INT* condition,
    INT* delay
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

condition

開始条件を格納する変数へのポインタを指定します。

値	意味
0	自動（ソフトウェア）
1	外部トリガ

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

delay

遅延回数を格納する変数へのポインタを指定します。
本機種では遅延回数に0（遅延なし）しか設定できない為、必ず0が格納されます。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

パラメータの詳細については、YdxDiSetStartCondition関数 を参照してください。

使用例

サンプリング開始条件の設定を取得します。

C#

```
int result;
int condition;
int delay;
result = Ydx.DiGetStartCondition(id, out condition, out delay);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim condition As Integer
Dim delay As Integer
result = YdxDiGetStartCondition(id, condition, delay)
```

VB6.0

```
Dim result As Long
Dim condition As Long
Dim delay As Long
result = YdxDiGetStartCondition(id, condition, delay)
```

C++/CLI

```
int result;
int condition;
int delay;
result = YdxDiGetStartCondition(id, &condition, &delay);
```

C/C++

```
INT result;
INT condition;
INT delay;
result = YdxDiGetStartCondition(id, &condition, &delay);
```

YdxDiGetStartExternal

機能

サンプリング開始条件（外部トリガ）の設定を取得します。

書式

```
INT YdxDiGetStartExternal(
    INT id,
    INT* diChannel,
    INT* mode
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

diChannel

外部トリガとして使用するデジタル入力チャネルを格納する変数へのポインタを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

mode

動作モードを格納する変数へのポインタを指定します。

値	意味
0	立ち上がりエッジセンス（OFF→ONに変化した時に、条件成立）
1	立ち下がりエッジセンス（ON→OFFに変化した時に、条件成立）
2	両エッジセンス（OFF→ON または ON→OFFに変化した時に、条件成立）

値	意味
3	ハイレベルセンス（ONの時に、条件成立。最初からONだった場合も、条件成立）
4	ローレベルセンス（OFFの時に、条件成立。最初からOFFだった場合も、条件成立）

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

パラメータの詳細については、YdxDiSetStartExternal関数 を参照してください。

使用例

サンプリング開始条件（外部トリガ）の設定を取得します。

C#

```
int result;
int diChannel;
int mode;
result = Ydx.DiGetStartExternal(id, out diChannel, out mode);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim diChannel As Integer
Dim mode As Integer
result = YdxDiGetStartExternal(id, diChannel, mode)
```

VB6.0

```
Dim result As Long
Dim diChannel As Long
Dim mode As Long
result = YdxDiGetStartExternal(id, diChannel, mode)
```

C++/CLI

```
int result;
int diChannel;
int mode;
result = YdxDiGetStartExternal(id, &diChannel, &mode);
```

C/C++

```
INT result;
INT diChannel;
INT mode;
result = YdxDiGetStartExternal(id, &diChannel, &mode);
```

YdxDiGetStopCondition

機能

サンプリング停止条件の設定を取得します。

書式

```
INT YdxDiGetStopCondition(
    INT id,
    INT* condition,
    INT* delay
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

condition

停止条件を格納する変数へのポインタを指定します。

値	意味
0	サンプル数
1	外部トリガ
2	ソフトウェア（YdxDiStop関数）

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

delay

遅延回数を格納する変数へのポインタを指定します。
本機種では遅延回数に0（遅延なし）しか設定できない為、必ず0が格納されます。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

パラメータの詳細については、YdxDiSetStopCondition関数 を参照してください。

使用例

サンプリング停止条件の設定を取得します。

C#

```
int result;
int condition;
int delay;
result = Ydx.DiGetStopCondition(id, out condition, out delay);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim condition As Integer
Dim delay As Integer
result = YdxDiGetStopCondition(id, condition, delay)
```

VB6.0

```
Dim result As Long
Dim condition As Long
Dim delay As Long
result = YdxDiGetStopCondition(id, condition, delay)
```

C++/CLI

```
int result;
int condition;
int delay;
result = YdxDiGetStopCondition(id, &condition, &delay);
```

C/C++

```
INT result;
INT condition;
INT delay;
result = YdxDiGetStopCondition(id, &condition, &delay);
```

YdxDiGetStopSampleNum

機能

サンプリング停止条件（サンプル数）の設定を取得します。

書式

```
INT YdxDiGetStopSampleNum(
    INT id,
    INT* sampleNum
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

sampleNum

サンプル数を格納する変数へのポインタを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

パラメータの詳細については、YdxDiSetStopSampleNum関数 を参照してください。

使用例

サンプリング停止条件（サンプル数）の設定を取得します。

C#

```
int result;
int sampleNum;
result = Ydx.DiGetStopSampleNum(id, out sampleNum);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim sampleNum As Integer
result = YdxDiGetStopSampleNum(id, sampleNum)
```

VB6.0

```
Dim result As Long
Dim sampleNum As Long
result = YdxDiGetStopSampleNum(id, sampleNum)
```

C++/CLI

```
int result;
int sampleNum;
result = YdxDiGetStopSampleNum(id, &sampleNum);
```

C/C++

```
INT result;
INT sampleNum;
result = YdxDiGetStopSampleNum(id, &sampleNum);
```

YdxDiGetStopExternal

機能

サンプリング停止条件（外部トリガ）の設定を取得します。

書式

```
INT YdxDiGetStopExternal(
    INT id,
    INT* diChannel,
    INT* mode
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

diChannel

外部トリガとして使用するデジタル入力チャネルを格納する変数へのポインタを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

mode

動作モードを格納する変数へのポインタを指定します。

値	意味
0	立ち上がりエッジセンス（OFF→ONに変化した時に、条件成立）
1	立ち下がりエッジセンス（ON→OFFに変化した時に、条件成立）
2	両エッジセンス（OFF→ON または ON→OFFに変化した時に、条件成立）

値	意味
3	ハイレベルセンス（ONの時に、条件成立。最初からONだった場合も、条件成立）
4	ローレベルセンス（OFFの時に、条件成立。最初からOFFだった場合も、条件成立）

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

パラメータの詳細については、YdxDiSetStopExternal関数 を参照してください。

使用例

サンプリング停止条件（外部トリガ）の設定を取得します。

C#

```
int result;
int diChannel;
int mode;
result = Ydx.DiGetStopExternal(id, out diChannel, out mode);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim diChannel As Integer
Dim mode As Integer
result = YdxDiGetStopExternal(id, diChannel, mode)
```

VB6.0

```
Dim result As Long
Dim diChannel As Long
Dim mode As Long
result = YdxDiGetStopExternal(id, diChannel, mode)
```

C++/CLI

```
int result;
int diChannel;
int mode;
result = YdxDiGetStopExternal(id, &diChannel, &mode);
```

C/C++

```
INT result;
INT diChannel;
INT mode;
result = YdxDiGetStopExternal(id, &diChannel, &mode);
```

YdxDiStart

機能

デジタル入力動作 を開始します。

書式

```
INT YdxDiStart(
    INT id
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

本関数は、デジタル入力が 動作中 には実行できません。

使用例

デジタル入力動作を開始します。

C#

```
int result;
result = Ydx.DiStart(id);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxDiStart(id)
```

VB6.0

```
Dim result As Long
result = YdxDiStart(id)
```

C++/CLI

```
int result;
result = YdxDiStart(id);
```

C/C++

```
INT result;
result = YdxDiStart(id);
```

YdxDiStop

機能

デジタル入力動作 を停止します。

書式

```
INT YdxDiStop(
    INT id
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

使用例

デジタル入力動作を停止します。

C#

```
int result;
result = Ydx.DiStop(id);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxDiStop(id)
```

VB6.0

```
Dim result As Long
result = YdxDiStop(id)
```

C++/CLI

```
int result;
result = YdxDiStop(id);
```

C/C++

```
INT result;
result = YdxDiStop(id);
```

YdxDiReset

機能

デジタル入力機能をリセットします。

書式

```
INT YdxDiReset(
    INT id
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

本関数が実行されると、以下の状態になります。

- デジタル入力が 動作中 の場合、動作は停止されます。
- データが、クリアされます。
- 状態 （ステータス）の全てのビットが、0になります。
- 状態（サンプル数）が、0になります。
- 状態（動作済みリピート回数）が、0になります。
- 設定値は、全て初期化されます。

使用例

デジタル入力機能をリセットします。

C#

```
int result;
result = Ydx.DiReset(id);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxDiReset(id)
```

VB6.0

```
Dim result As Long
result = YdxDiReset(id)
```

C++/CLI

```
int result;
result = YdxDiReset(id);
```

C/C++

```
INT result;
result = YdxDiReset(id);
```

YdxDiGetStatus

機能

現在の状態を取得します。

書式

```
INT YdxDiGetStatus(
    INT id,
    INT* status,
    INT* sampleCount,
    INT* repeatCount
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

status

ステータスを格納する変数へのポインタを指定します。
ビットごとに意味を持っていて、論理和された結果が格納されます。

値	定義名	ステータス
00000001h	YDX_STATUS_BUSY	動作中 動作が開始されると（YdxDiStart関数 が実行されると）オンになります。 動作が終了するとオフに戻ります。
00000002h	YDX_STATUS_SAMPLE_NUM	監視サンプル数 データバッファのデータが、監視サンプル数以上になった場合にオンになります。 監視サンプル数は、YdxDiSetCheckSampleNum関数 で変更できます。
00000004h	YDX_STATUS_START_TRIG	開始条件 成立済み 開始条件が成立するとオンになります。 リピートにより再び開始条件待ちになるとオフに戻ります。

値	定義名	ステータス
00000008h	YDX_STATUS_STOP_TRIG	停止条件 成立済み 停止条件が成立するとオンになります。 リピートにより開始条件が成立するとオフに戻ります。
00010000h	YDX_STATUS_SAMPLE_CLOCK_ERR	サンプリングクロック エラー発生 外部クロック使用時に、周期が早すぎる外部クロックが入力された場合にオンになります。 外部クロックの周期に問題がないか、チャタリング・ノイズが含まれていないかを確認してください。
00020000h	YDX_STATUS_OVERRUN_ERR	オーバランエラー発生 データバッファがFIFOバッファ形式に設定されている場合に、データバッファ容量を超えてサンプリングがおこなわれた場合にオンになります。 定期的に YdxDiGetData関数 を使用してデータを読み出して、データバッファが満杯にならないようにしてください。
00040000h	YDX_STATUS_HARDWARE_ERR	ハードウェアエラー発生 ユニット内部回路に異常が検出した場合にオンになります。 通常ありえませんが、もし発生した場合は、どのような状況で発生したかを弊社サポートまでご連絡ください。
00080000h	YDX_STATUS_COMMUNICATE_ERR	通信エラー発生 USB通信に異常が検出された場合にオンになります。 近くにノイズ要因がないか確認してください。 確認しても問題が見当たらない場合は、どのような状況で発生したかを弊社サポートまでご連絡ください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

sampleCount

サンプル数を格納する変数へのポインタを指定します。

データバッファがFIFOバッファ形式に設定されている場合、データバッファに記憶されてまだ読み出されていないサンプル数が格納されます。
データバッファがリングバッファ形式に設定されている場合、データバッファに記憶されているサンプル数が格納されます。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

repeatCount

動作済みリピート回数を格納する変数へのポインタを指定します。

2,147,483,647回を超えた場合、0に戻ってカウントされます。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

使用例

現在の状態を取得します。

C#

```
int result;
int status;
int sampleCount;
int repeatCount;
result = Ydx.DiGetStatus(id, out status, out sampleCount, out repeatCount);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim status As Integer
Dim sampleCount1 As Integer
Dim repeatCount As Integer
result = YdxDiGetStatus(id, status, sampleCount, repeatCount)
```

VB6.0

```
Dim result As Long
Dim status As Long
Dim sampleCount1 As Long
Dim repeatCount As Long
result = YdxDiGetStatus(id, status, sampleCount, repeatCount)
```

C++/CLI

```
int result;
int status;
int sampleCount;
int repeatCount;
result = YdxDiGetStatus(id, &status, &sampleCount, &repeatCount);
```

C/C++

```
INT result;
INT status;
INT sampleCount;
INT repeatCount;
result = YdxDiGetStatus(id, &status, &sampleCount, &repeatCount);
```

YdxDiGetData

機能

データを取得します。

書式

```
INT YdxDiGetData(
    INT id,
    INT* sampleNum,
    INT* data
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

sampleNum

読み出したいサンプル数を格納した変数へのポインタを指定します。
関数実行後には、実際に読み出されたサンプル数が格納されます。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	ref int	Integer	Long	int*	INT*

data

データを格納する変数へのポインタを指定します。
データは、以下のフォーマットで格納されます。

サンプリング	MSB															LSB
1回目	IN15	IN14	IN13	IN12	IN11	IN10	IN9	IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1	IN0
2回目	IN15	IN14	IN13	IN12	IN11	IN10	IN9	IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1	IN0
3回目	IN15	IN14	IN13	IN12	IN11	IN10	IN9	IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1	IN0
・	IN15	IN14	IN13	IN12	IN11	IN10	IN9	IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1	IN0
・	IN15	IN14	IN13	IN12	IN11	IN10	IN9	IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1	IN0
・	IN15	IN14	IN13	IN12	IN11	IN10	IN9	IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1	IN0

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int[]	Integer	Long	int*	INT*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

データバッファ形式によって以下の動作となります。

FIFOバッファ形式の場合

- 古いデータを指定したサンプル数（sampleNum）読み出します。
- 読み出されたデータはデータバッファから破棄されます。
- デジタル入力が 動作中 でも実行できます。

リングバッファ形式の場合

- 新しいデータを指定したサンプル数（sampleNum）読み出します。
- 読み出されたデータはデータバッファから破棄されません。
  （再度読み出す事が可能）
- デジタル入力が 動作中 は実行できません。

使用例

1000サンプリング分のデータを取得します。

C#

```
int result;
int sampleNum = 1000;
int[] data = new int[1000];
result = Ydx.DiGetData(id, ref sampleNum, data);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim sampleNum As Integer = 1000
Dim data(999) As Integer
result = YdxDiGetData(id, sampleNum, data)
```

VB6.0

```
Dim result As Long
Dim sampleNum As Long
Dim data(999) As Long
sampleNum = 1000
result = YdxDiGetData(id, sampleNum, data(0))
```

C++/CLI

```
int result;
int sampleNum = 1000;
int data[1000];
result = YdxDiGetData(id, &sampleNum, data);
```

C/C++

```
INT result;
INT sampleNum = 1000;
INT data[1000];
result = YdxDiGetData(id, &sampleNum, data);
```

YdxDiClearData

機能

データをクリアします。

書式

```
INT YdxDiClearData(
    INT id
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

本関数が実行されると、以下の状態になります。

- データが、クリアされます。
- 状態 （ステータス）の「監視サンプル数」ビットが、0になります。
- 状態（サンプル数）が、0になります。
- 状態（動作済みリピート回数）が、0になります。

本関数は、デジタル入力が 動作中 には実行できません。

使用例

データをクリアします。

C#

```
int result;
result = Ydx.DiClearData(id);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxDiClearData(id)
```

VB6.0

```
Dim result As Long
result = YdxDiClearData(id)
```

C++/CLI

```
int result;
result = YdxDiClearData(id);
```

C/C++

```
INT result;
result = YdxDiClearData(id);
```

デジタル出力関数一覧

簡易デジタル出力 ・ 高機能デジタル出力　共用関数

関数	機能
YdxDoGetOutput	デジタル出力状態を取得します。
YdxDoGetOutputBit	デジタル出力状態を取得します。 データは、ビットごとに読み込みます。
YdxDoSetChannel	チャネルの動作モードを設定します。
YdxDoGetChannel	チャネルの動作モードの設定を取得します。

簡易デジタル出力　専用関数

関数	機能
YdxDoOutput	デジタル出力端子を制御します。
YdxDoOutputBit	デジタル出力端子を制御します。 データは、ビットごとに指定します。

高機能デジタル出力　専用関数

設定

関数	機能
YdxDoSetBuffer	データバッファ の形式を設定します。
YdxDoSetCheckSampleNum	監視サンプル数 を設定します。
YdxDoSetClock	サンプリングクロック の種類を設定します。
YdxDoSetClockInternal	内部クロックを設定します。
YdxDoSetClockExternal	外部クロックを設定します。
YdxDoSetRepeat	リピート を設定します。

関数	機能
YdxDoSetStartCondition	サンプリング開始条件 を設定します。
YdxDoSetStartExternal	サンプリング開始条件（ 外部トリガ ）を設定します。
YdxDoSetStopCondition	サンプリング停止条件 を設定します。
YdxDoSetStopExternal	サンプリング停止条件（外部トリガ）を設定します。

設定の取得

関数	機能
YdxDoGetBuffer	データバッファの設定を取得します。
YdxDoGetCheckSampleNum	監視サンプル数の設定を取得します。
YdxDoGetClock	サンプリングクロックの設定を取得します。
YdxDoGetClockInternal	内部クロックの設定を取得します。
YdxDoGetClockExternal	外部クロックの設定を取得します。
YdxDoGetRepeat	リピートの設定を取得します。
YdxDoGetSampleNum	データ設定済みのサンプル数を取得します。
YdxDoGetStartCondition	サンプリング開始条件の設定を取得します。
YdxDoGetStartExternal	サンプリング開始条件（外部トリガ）の設定を取得します。
YdxDoGetStopCondition	サンプリング停止条件の設定を取得します。
YdxDoGetStopExternal	サンプリング停止条件（外部トリガ）の設定を取得します。

データの設定・クリア

関数	機能
YdxDoSetData	データを設定します。

関数	機能
YdxDoClearData	データをクリアします。

動作開始・停止

関数	機能
YdxDoStart	デジタル出力動作 を開始します。
YdxDoStop	デジタル出力動作を停止します。

リセット

関数	機能
YdxDoReset	デジタル出力機能をリセットします。

状態の取得

関数	機能
YdxDoGetStatus	現在の状態を取得します。

YdxDoGetOutput

機能

デジタル出力状態を取得します。

書式

```
INT YdxDoGetOutput(
    INT id,
    INT* data
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

data

状態を格納する変数へのポインタを指定します。
データは、以下のフォーマットで格納されます。

MSB															LSB
OUT15	OUT14	OUT13	OUT12	OUT11	OUT10	OUT9	OUT8	OUT7	OUT6	OUT5	OUT4	OUT3	OUT2	OUT1	OUT0

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

使用例

デジタル出力状態を取得します。

C#

```
int result;
int data;
result = Ydx.DoGetOutput(id, out data);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim data As Integer
result = YdxDoGetOutput(id, data)
```

VB6.0

```
Dim result As Long
Dim data As Long
result = YdxDoGetOutput(id, data)
```

C++/CLI

```
int result;
int data;
result = YdxDoGetOutput(id, &data);
```

C/C++

```
INT result;
INT data;
result = YdxDoGetOutput(id, &data);
```

YdxDoGetOutputBit

機能

デジタル出力状態を取得します。
データは、ビットごとに読み込みます。

書式

```
INT YdxDoGetOutputBit(
    INT id,
    INT start,
    INT num,
    INT* data
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

start

状態の読み込みを開始する出力チャネルを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

num

状態の読み込みをするチャネル数を指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

data

状態を格納する変数へのポインタを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int[]	Integer	Long	int*	INT*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

使用例

OUT2～OUT4の状態を取得します。

C#

```
int result;
int[] data = new int[3];
result = Ydx.DoGetOutputBit(id, 2, 3, data);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim data(2) As Integer
result = YdxDoGetOutputBit(id, 2, 3, data)
```

VB6.0

```
Dim result As Long
Dim data(2) As Long
result = YdxDoGetOutputBit(id, 2, 3, data(0))
```

C++/CLI

```
int result;
int data[3];
result = YdxDoGetOutputBit(id, 2, 3, data);
```

C/C++

```
INT result;
INT data[3];
result = YdxDoGetOutputBit(id, 2, 3, data);
```

YdxDoSetChannel

機能

チャネルの動作モードを設定します。

書式

```
INT YdxDoSetChannel(
    INT id,
    INT channel,
    INT mode
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

channel

設定をするチャネルを指定します。
設定範囲は0～15です。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

mode

動作モードを指定します。

値	意味
0	サンプリング無効（簡易デジタル出力）モード
1	サンプリング有効（高機能デジタル出力）モード

初期値は全てのチャネルが0です。
「簡易デジタル出力モード」に設定したチャネルは

- 簡易デジタル出力関数（YdxDoOutput関数 ・ YdxDoOutputBit関数）によってデジタル出力の制御がおこなえるようになります。
- 高機能デジタル出力（サンプリング出力）の影響は受けなくなります。

「高機能デジタル出力モード」に設定したチャネルは

- 高機能デジタル出力（サンプリング出力）がおこなえるようになります。
- 簡易デジタル出力関数（YdxDoOutput関数・YdxDoOutputBit関数）の影響は受けなくなります。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

本関数は、デジタル出力が 動作中 には実行できません。

使用例

チャネル2（DOUT2）を、高機能デジタル出力モードに設定します。

C#

```
int result;
result = Ydx.DoSetChannel(id, 2, 1);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxDoSetChannel(id, 2, 1)
```

VB6.0

```
Dim result As Long
result = YdxDoSetChannel(id, 2, 1)
```

C++/CLI

```
int result;
result = YdxDoSetChannel(id, 2, 1);
```

C/C++

```
INT result;
result = YdxDoSetChannel(id, 2, 1);
```

YdxDoGetChannel

機能

チャネルの動作モードの設定を取得します。

書式

```
INT YdxDoGetChannel(
    INT id,
    INT channel,
    INT* mode
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

channel

設定を取得するチャネルを指定します。
設定範囲は0～15です。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

mode

動作モードを格納する変数へのポインタを指定します。

値	意味
0	簡易デジタル出力モード
1	高機能デジタル出力モード

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

パラメータの詳細については、YdxDoSetChannel関数 を参照してください。

使用例

チャネル2の、動作モードを取得します。

C#

```
int result;
int mode;
result = Ydx.DoGetChannel(id, 2, out mode);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim mode As Integer
result = YdxDoGetChannel(id, 2, mode)
```

VB6.0

```
Dim result As Long
Dim mode As Long
result = YdxDoGetChannel(id, 2, mode)
```

C++/CLI

```
int result;
int mode;
result = YdxDoGetChannel(id, 2, &mode);
```

C/C++

```
INT result;
INT mode;
result = YdxDoGetChannel(id, 2, &mode);
```

YdxDoOutput

機能

デジタル出力端子を制御します。

書式

```
INT YdxDoOutput(
    INT id,
    INT data
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

data

出力データを指定します。
データは、以下のフォーマットで指定します。

MSB															LSB
OUT15	OUT14	OUT13	OUT12	OUT11	OUT10	OUT9	OUT8	OUT7	OUT6	OUT5	OUT4	OUT3	OUT2	OUT1	OUT0

設定範囲は、0～FFFFh（0～65535）です。
ただし、出力が更新されるのは、チャネル設定 で簡易デジタル出力モードになっているチャネルのみです。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

使用例

OUT0とOUT13をON、その他の出力端子をOFFにします。

C#

```
int result;
result = Ydx.DoOutput(id, 0x2001);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxDoOutput(id, &H2001)
```

VB6.0

```
Dim result As Long
result = YdxDoOutput(id, &H2001)
```

C++/CLI

```
int result;
result = YdxDoOutput(id, 0x2001);
```

C/C++

```
INT result;
result = YdxDoOutput(id, 0x2001);
```

YdxDoOutputBit

機能

デジタル出力端子を制御します。
データは、ビットごとに指定します。

書式

```
INT YdxDoOutputBit(
    INT id,
    INT start,
    INT num,
    INT* data
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

start

出力開始チャネルを指定します。
設定範囲は0～15です。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

num

出力をするチャネル数を指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

data

出力データを格納した変数へのポインタを指定します。

値	意味
0	OFF
1	ON
2	現在の状態を保持

ただし、出力が更新されるのは、チャネル設定 で簡易デジタル出力モードになっているチャネルのみです。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int[]	Integer	Long	int*	INT*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

使用例

OUT2とOUT4をONにします。
その他の出力端子は現在の状態を保持します。

C#

```
int result;
int[] data = new int[3];
data[0] = 1;
data[1] = 2;
data[2] = 1;
result = Ydx.DoOutputBit(id, 2, 3, data);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim data(2) As Integer
data(0) =
1
```

```
data(1) = 2
data(2) = 1
result = YdxDoOutputBit(id, 2, 3, data)
```

VB6.0

```
Dim result As Long
Dim data(2) As Long
data(0) = 1
data(1) = 2
data(2) = 1
result = YdxDoOutputBit(id, 2, 3, data(0))
```

C++/CLI

```
int result;
int data[3];
data[0] = 1;
data[1] = 2;
data[2] = 1;
result = YdxDoOutputBit(id, 2, 3, data);
```

C/C++

```
INT result;
INT data[3];
data[0] = 1;
data[1] = 2;
data[2] = 1;
result = YdxDoOutputBit(id, 2, 3, data);
```

YdxDoSetBuffer

機能

データバッファ の形式を設定します。

書式

```
INT YdxDoSetBuffer(
    INT id,
    INT bufferType
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

bufferType

データバッファの形式を指定します。

値	意味	説明
0	FIFOバッファ形式	出力は、先に設定したデータから順におこなわれます。 出力したデータは、バッファから破棄されます。 動作中 にデータを設定（追加）する事が可能です。
1	リングバッファ形式	データを最後まで出力すると、先頭に戻って繰り返し出力がおこなわれます。 出力したデータは、バッファから破棄されません。 動作中にデータを設定する事はできません。

初期値は0です。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

> ⚠ データバッファにデータが残った状態のまま、本関数により設定を変更した場合、データはクリアされます。

本関数は、デジタル出力が 動作中 には実行できません。

使用例

データバッファを、リングバッファ形式に設定します。

C#

```
int result;
result = Ydx.DoSetBuffer(id, 1);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxDoSetBuffer(id, 1)
```

VB6.0

```
Dim result As Long
result = YdxDoSetBuffer(id, 1)
```

C++/CLI

```
int result;
result = YdxDoSetBuffer(id, 1);
```

C/C++

```
INT result;
result = YdxDoSetBuffer(id, 1);
```

YdxDoSetCheckSampleNum

機能

監視サンプル数 を設定します。

書式

```
INT YdxDoSetCheckSampleNum(
    INT id,
    INT sampleNum
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

sampleNum

監視サンプル数を指定します。
設定範囲は1～2,147,483,647、初期値は500です。
未出力サンプル数（データバッファにデータが残っているサンプル数）が、監視サンプル数以下になった場合、YdxDoGetStatus関数 でステータスを読み出した時、監視サンプル数ビットが1になります。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

本関数は、デジタル出力が 動作中 には実行できません。

使用例

監視サンプル数を、2000に設定します。

C#

```
int result;
result = Ydx.DoSetCheckSampleNum(id, 2000);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxDoSetCheckSampleNum(id, 2000)
```

VB6.0

```
Dim result As Long
result = YdxDoSetCheckSampleNum(id, 2000)
```

C++/CLI

```
int result;
result = YdxDoSetCheckSampleNum(id, 2000);
```

C/C++

```
INT result;
result = YdxDoSetCheckSampleNum(id, 2000);
```

YdxDoSetClock

機能

サンプリングクロック の種類を設定します。

書式

```
INT YdxDoSetClock(
    INT id,
    INT clockType
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

clockType

クロックの種類を指定します。

値	意味
0	内部クロック
1	外部クロック

初期値は0です。

「内部クロック」を指定する場合、YdxDoSetClockInternal関数 で、周期の設定をしてください。
「外部クロック」を指定する場合、YdxDoSetClockExternal関数 で、使用するデジタル入力チャネルと入力タイミングの設定をしてください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

本関数は、デジタル出力が 動作中 には実行できません。

使用例

サンプリングクロックとして、外部クロックを使用します。

C#

```
int result;
result = Ydx.DoSetClock(id, 1);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxDoSetClock(id, 1)
```

VB6.0

```
Dim result As Long
result = YdxDoSetClock(id, 1)
```

C++/CLI

```
int result;
result = YdxDoSetClock(id, 1);
```

C/C++

```
INT result;
result = YdxDoSetClock(id, 1);
```

YdxDoSetClockInternal

機能

内部クロックを設定します。

書式

```
INT YdxDoSetClockInternal(
    INT id,
    double period
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

period

周期を指定します。
単位は「μsec」です。
設定範囲は0.1～60,000,000 [μsec]、初期値は1,000 [μsec]です。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	double	Double	Double	double	double

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

YdxDoSetClock関数 で、クロックの種類として「内部クロック」を選択した場合にのみ設定が有効になります。
クロックの種類として「内部クロック」を選択しない場合は、本関数を実行する必要はありません。

出力回路の応答時間については機種によって異なりますので、取扱説明書を参照してください。

本関数は、デジタル出力が 動作中 には実行できません。

使用例

内部クロックを、2msec周期に設定します。

C#

```
int result;
result = Ydx.DoSetClockInternal(id, 2000);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxDoSetClockInternal(id, 2000)
```

VB6.0

```
Dim result As Long
result = YdxDoSetClockInternal(id, 2000)
```

C++/CLI

```
int result;
result = YdxDoSetClockInternal(id, 2000);
```

C/C++

```
INT result;
result = YdxDoSetClockInternal(id, 2000);
```

YdxDoSetClockExternal

機能

外部クロックを設定します。

書式

```
INT YdxDoSetClockExternal(
    INT id,
    INT diChannel,
    INT edge
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

diChannel

外部クロックとして使用するデジタル入力チャネルを指定します。
設定範囲は0～15、初期値は3です。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

edge

入力タイミングを指定します。

値	意味
0	立ち上がりエッジセンス（OFF→ON）
1	立ち下がりエッジセンス（ON→OFF）
2	両エッジセンス（OFF→ON と ON→OFF の両方）

初期値は0です。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

YdxDoSetClock関数 で、クロックの種類として「外部クロック」を選択した場合にのみ設定が有効になります。
クロックの種類として「外部クロック」を選択しない場合は、本関数を実行する必要はありません。

本関数は、デジタル出力が 動作中 には実行できません。

使用例

外部クロックを、デジタル入力チャネル2（IN2）の立ち下りエッジに設定します。

C#

```
int result;
result = Ydx.DoSetClockExternal(id, 2, 1);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxDoSetClockExternal(id, 2, 1)
```

VB6.0

```
Dim result As Long
result = YdxDoSetClockExternal(id, 2, 1)
```

C++/CLI

```
int result;
result = YdxDoSetClockExternal(id, 2, 1);
```

C/C++

```
INT result;
result = YdxDoSetClockExternal(id, 2, 1);
```

YdxDoSetRepeat

機能

リピート を設定します。

書式

```
INT YdxDoSetRepeat(
    INT id,
    INT repeatNum
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

repeatNum

リピート回数を設定します。
設定範囲は0～2,147,483,647[回]、初期値は0です。
0に設定した場合は、リピート回数は無限となります。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

リピートとは、サンプリング開始条件（YdxDoSetStartCondition関数 で設定）からサンプリング停止条件（YdxDoSetStopCondition関数 で設定）までの動作を、繰り返しおこなう事です。

本関数は、デジタル出力が 動作中 には実行できません。

使用例

リピートを、10回に設定します。

C#

```
int result;
result = Ydx.DoSetRepeat(id, 10);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxDoSetRepeat(id, 10)
```

VB6.0

```
Dim result As Long
result = YdxDoSetRepeat(id, 10)
```

C++/CLI

```
int result;
result = YdxDoSetRepeat(id, 10);
```

C/C++

```
INT result;
result = YdxDoSetRepeat(id, 10);
```

YdxDoSetStartCondition

機能

サンプリング開始条件 を設定します。

書式

```
INT YdxDoSetStartCondition(
    INT id,
    INT condition,
    INT delay
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

condition

サンプリング開始条件を指定します。

値	意味
0	自動（ソフトウェア）
1	外部トリガ

初期値は0です。

「外部トリガ」を指定する場合、YdxDoSetStartExternal関数 で、使用するデジタル入力チャネルとモードの設定をしてください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

delay

遅延回数を指定します。
本機種では0（遅延なし）しか設定できません。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

本関数は、デジタル出力が 動作中 には実行できません。

使用例

サンプリング開始条件を、外部トリガに設定します。

C#

```
int result;
result = Ydx.DoSetStartCondition(id, 1, 0);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxDoSetStartCondition(id, 1, 0)
```

VB6.0

```
Dim result As Long
result = YdxDoSetStartCondition(id, 1, 0)
```

C++/CLI

```
int result;
result = YdxDoSetStartCondition(id, 1, 0);
```

C/C++

```
INT result;
result = YdxDoSetStartCondition(id, 1, 0);
```

YdxDoSetStartExternal

機能

サンプリング開始条件（外部トリガ）を設定します。

書式

```
INT YdxDoSetStartExternal(
    INT id,
    INT diChannel,
    INT mode
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

diChannel

外部トリガとして使用するデジタル入力チャネルを指定します。
設定範囲は0～15、初期値は4です。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

mode

動作モードを指定します。

値	意味
0	立ち上がりエッジセンス（OFF→ONに変化した時に、条件成立）
1	立ち下がりエッジセンス（ON→OFFに変化した時に、条件成立）
2	両エッジセンス（ON→OFF または OFF→ONに変化した時に、条件成立）

値	意味
3	ハイレベルセンス（ONの時に、条件成立。最初からONだった場合も、条件成立）
4	ローレベルセンス（OFFの時に、条件成立。最初からOFFだった場合も、条件成立）

初期値は0です。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

YdxDoSetStartCondition関数 で、サンプリング開始条件として「外部トリガ」を選択した場合にのみ設定が有効になります。
サンプリング開始条件として「外部トリガ」を選択しない場合は、本関数を実行する必要はありません。

本関数は、デジタル出力が 動作中 には実行できません。

使用例

サンプリング開始条件（外部トリガ）を設定します。
外部トリガとして使用するデジタル入力チャネルはチャネル1、動作モードは両エッジセンスに設定します。

C#

```
int result;
result = Ydx.DoSetStartExternal(id, 1, 2);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxDoSetStartExternal(id, 1, 2)
```

VB6.0

```
Dim result As Long
result = YdxDoSetStartExternal(id, 1, 2)
```

C++/CLI

```
int result;
result = YdxDoSetStartExternal(id, 1, 2);
```

C/C++

```
INT result;
result = YdxDoSetStartExternal(id, 1, 2);
```

YdxDoSetStopCondition

機能

サンプリング停止条件 を設定します。

書式

```
INT YdxDoSetStopCondition(
    INT id,
    INT condition,
    INT delay
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

condition

サンプリング停止条件を指定します。

値	意味
0	データ終了
1	外部トリガ

初期値は0です。

「外部トリガ」を指定する場合、YdxDoSetStopExternal関数 で、使用するデジタル入力チャネルとモードの設定をしてください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

delay

遅延回数を指定します。
本機種では0（遅延なし）しか設定できません。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

本関数は、デジタル出力が 動作中 には実行できません。

使用例

サンプリング停止条件を、外部トリガに設定します。

C#

```
int result;
result = Ydx.DoSetStopCondition(id, 1, 0);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxDoSetStopCondition(id, 1, 0)
```

VB6.0

```
Dim result As Long
result = YdxDoSetStopCondition(id, 1, 0)
```

C++/CLI

```
int result;
result = YdxDoSetStopCondition(id, 1, 0);
```

C/C++

```
INT result;
result = YdxDoSetStopCondition(id, 1, 0);
```

YdxDoSetStopExternal

機能

サンプリング停止条件（外部トリガ）を設定します。

書式

```
INT YdxDoSetStopExternal(
    INT id,
    INT diChannel,
    INT mode
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

diChannel

外部トリガとして使用するデジタル入力チャネルを指定します。
設定範囲は0～15、初期値は5です。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

mode

動作モードを指定します。

値	意味
0	立ち上がりエッジセンス（OFF→ONに変化した時に、条件成立）
1	立ち下がりエッジセンス（ON→OFFに変化した時に、条件成立）
2	両エッジセンス（OFF→ON または ON→OFFに変化した時に、条件成立）

値	意味
3	ハイレベルセンス（ONの時に、条件成立。最初からONだった場合も、条件成立）
4	ローレベルセンス（OFFの時に、条件成立。最初からOFFだった場合も、条件成立）

初期値は0です。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

YdxDoSetStopCondition関数 で、サンプリング停止条件として「外部トリガ」を選択した場合にのみ設定が有効になります。
サンプリング停止条件として「外部トリガ」を選択しない場合は、本関数を実行する必要はありません。

本関数は、デジタル出力が 動作中 には実行できません。

使用例

サンプリング停止条件（外部トリガ）を設定します。
外部トリガとして使用するデジタル入力チャネルはチャネル1、動作モードは両エッジセンスに設定します。

C#

```
int result;
result = Ydx.DoSetStopExternal(id, 1, 2);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxDoSetStopExternal(id, 1, 2)
```

VB6.0

```
Dim result As Long
result = YdxDoSetStopExternal(id, 1, 2)
```

C++/CLI

```
int result;
result = YdxDoSetStopExternal(id, 1, 2);
```

C/C++

```
INT result;
result = YdxDoSetStopExternal(id, 1, 2);
```

YdxDoGetBuffer

機能

データバッファの設定を取得します。

書式

```
INT YdxDoGetBuffer(
    INT id,
    INT* bufferType
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

bufferType

データバッファの形式を格納する変数へのポインタを指定します。

値	意味
0	FIFOバッファ形式
1	リングバッファ形式

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

パラメータの詳細については、YdxDoSetBuffer関数 を参照してください。

使用例

データバッファの設定を取得します。

C#

```
int result;
int bufferType;
result = Ydx.DoGetBuffer(id, out bufferType);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim bufferType As Integer
result = YdxDoGetBuffer(id, bufferType)
```

VB6.0

```
Dim result As Long
Dim bufferType As Long
result = YdxDoGetBuffer(id, bufferType)
```

C++/CLI

```
int result;
int bufferType;
result = YdxDoGetBuffer(id, &bufferType);
```

C/C++

```
INT result;
INT bufferType;
result = YdxDoGetBuffer(id, &bufferType);
```

YdxDoGetCheckSampleNum

機能

監視サンプル数の設定を取得します。

書式

```
INT YdxDoGetCheckSampleNum(
    INT id,
    INT* sampleNum
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

sampleNum

監視サンプル数を格納する変数へのポインタを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

パラメータの詳細については、YdxDoSetCheckSampleNum関数 を参照してください。

使用例

監視サンプル数の設定を取得します。

C#

```
int result;
int sampleNum;
result = Ydx.DoGetCheckSampleNum(id, out sampleNum);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim sampleNum As Integer
result = YdxDoGetCheckSampleNum(id, sampleNum)
```

VB6.0

```
Dim result As Long
Dim sampleNum As Long
result = YdxDoGetCheckSampleNum(id, sampleNum)
```

C++/CLI

```
int result;
int sampleNum;
result = YdxDoGetCheckSampleNum(id, &sampleNum);
```

C/C++

```
INT result;
INT sampleNum;
result = YdxDoGetCheckSampleNum(id, &sampleNum);
```

YdxDoGetClock

機能

サンプリングクロックの設定を取得します。

書式

```
INT YdxDoGetClock(
    INT id,
    INT* clockType
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

clockType

クロックの種類を格納する変数へのポインタを指定します。

値	意味
0	内部クロック
1	外部クロック

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

パラメータの詳細については、YdxDoSetClock関数 を参照してください。

使用例

サンプリングクロックの設定を取得します。

C#

```
int result;
int clockType;
result = Ydx.DoGetClock(id, out clockType);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim clockType As Integer
result = YdxDoGetClock(id, clockType)
```

VB6.0

```
Dim result As Long
Dim clockType As Long
result = YdxDoGetClock(id, clockType)
```

C++/CLI

```
int result;
int clockType;
result = YdxDoGetClock(id, &clockType);
```

C/C++

```
INT result;
INT clockType;
result = YdxDoGetClock(id, &clockType);
```

YdxDoGetClockInternal

機能

内部クロックの設定を取得します。

書式

```
INT YdxDoGetClockInternal(
    INT id,
    double* period
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

period

周期を格納する変数へのポインタを指定します。
単位は「μsec」です。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out double	Double	Double	double*	double*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

パラメータの詳細については、YdxDoSetClockInternal関数 を参照してください。

使用例

内部クロックの設定を取得します。

C#

```
int result;
double period;
result = Ydx.DoGetClockInternal(id, out period);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim period As Double
result = YdxDoGetClockInternal(id, period)
```

VB6.0

```
Dim result As Long
Dim period As Double
result = YdxDoGetClockInternal(id, period)
```

C++/CLI

```
int result;
double period;
result = YdxDoGetClockInternal(id, &period);
```

C/C++

```
INT result;
double period;
result = YdxDoGetClockInternal(id, &period);
```

YdxDoGetClockExternal

機能

外部クロックの設定を取得します。

書式

```
INT YdxDoGetClockExternal(
    INT id,
    INT* diChannel,
    INT* edge
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

diChannel

外部クロックとして使用するデジタル入力チャネルを格納する変数へのポインタを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

edge

入力タイミングを格納する変数へのポインタを指定します。

値	意味
0	立ち上がりエッジセンス（OFF→ON）
1	立ち下がりエッジセンス（ON→OFF）
2	両エッジセンス（OFF→ON と ON→OFF の両方）

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

パラメータの詳細については、YdxDoSetClockExternal関数 を参照してください。

使用例

外部クロックの設定を取得します。

C#

```
int result;
int diChannel;
int edge;
result = Ydx.DoGetClockExternal(id, out diChannel, out edge);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim diChannel As Integer
Dim edge As Integer
result = YdxDoGetClockExternal(id, diChannel, edge)
```

VB6.0

```
Dim result As Long
Dim diChannel As Long
Dim edge As Long
result = YdxDoGetClockExternal(id, diChannel, edge)
```

C++/CLI

```
int result;
int diChannel;
int edge;
result = YdxDoGetClockExternal(id, &diChannel, &edge);
```

C/C++

```
INT result;
INT diChannel;
INT edge;
result = YdxDoGetClockExternal(id, &diChannel, &edge);
```

YdxDoGetRepeat

機能

リピートの設定を取得します。

書式

```
INT YdxDoGetRepeat(
    INT id,
    INT* repeatNum
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

repeatNum

リピート設定回数を格納する変数へのポインタを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

パラメータの詳細については、YdxDoSetRepeat関数 を参照してください。

使用例

リピートの設定を取得します。

C#

```
int result;
int repeatNum;
result = Ydx.DoGetRepeat(id, out repeatNum);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim repeatNum As Integer
result = YdxDoGetRepeat(id, repeatNum)
```

VB6.0

```
Dim result As Long
Dim repeatNum As Long
result = YdxDoGetRepeat(id, repeatNum)
```

C++/CLI

```
int result;
int repeatNum;
result = YdxDoGetRepeat(id, &repeatNum);
```

C/C++

```
INT result;
INT repeatNum;
result = YdxDoGetRepeat(id, &repeatNum);
```

YdxDoGetSampleNum

機能

データ設定済みのサンプル数を取得します。

書式

```
INT YdxDoGetSampleNum(
    INT id,
    INT* sampleNum
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

sampleNum

サンプル数を格納する変数へのポインタを指定します。
2,147,483,647を超えた場合、0に戻ってカウントがおこなわれています。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

使用例

データ設定済みのサンプル数を取得します。

C#

```
int result;
int sampleNum;
result = Ydx.DoGetSampleNum(id, out sampleNum);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim sampleNum As Integer
result = YdxDoGetSampleNum(id, sampleNum)
```

VB6.0

```
Dim result As Long
Dim sampleNum As Long
result = YdxDoGetSampleNum(id, sampleNum)
```

C++/CLI

```
int result;
int sampleNum;
result = YdxDoGetSampleNum(id, &sampleNum);
```

C/C++

```
INT result;
INT sampleNum;
result = YdxDoGetSampleNum(id, &sampleNum);
```

YdxDoGetStartCondition

機能

サンプリング開始条件の設定を取得します。

書式

```
INT YdxDoGetStartCondition(
    INT id,
    INT* condition,  INT* delay
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

condition

開始条件を格納する変数へのポインタを指定します。

値	意味
0	自動（ソフトウェア）
1	外部トリガ

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

delay

遅延回数を格納する変数へのポインタを指定します。
本機種では遅延回数に0（遅延なし）しか設定できない為、必ず0が格納されます。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

パラメータの詳細については、YdxDoSetStartCondition関数 を参照してください。

使用例

サンプリング開始条件の設定を取得します。

C#

```
int result;
int condition;
int delay;
result = Ydx.DoGetStartCondition(id, out condition, out delay);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim condition As Integer
Dim delay As Integer
result = YdxDoGetStartCondition(id, condition, delay)
```

VB6.0

```
Dim result As Long
Dim condition As Long
Dim delay As Long
result = YdxDoGetStartCondition(id, condition, delay)
```

C++/CLI

```
int result;
int condition;
int delay;
result = YdxDoGetStartCondition(id, &condition, &delay);
```

C/C++

```
INT result;
INT condition;
INT delay;
result = YdxDoGetStartCondition(id, &condition, &delay);
```

YdxDoGetStartExternal

機能

サンプリング開始条件（外部トリガ）の設定を取得します。

書式

```
INT YdxDoGetStartExternal(
    INT id,
    INT* diChannel,
    INT* mode
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

diChannel

外部トリガとして使用するデジタル入力チャネルを格納する変数へのポインタを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

mode

動作モードを格納する変数へのポインタを指定します。

値	意味
0	立ち上がりエッジセンス（OFF→ONに変化した時に、条件成立）
1	立ち下がりエッジセンス（ON→OFFに変化した時に、条件成立）
2	両エッジセンス（OFF→ON または ON→OFFに変化した時に、条件成立）

値	意味
3	ハイレベルセンス（ONの時に、条件成立。最初からONだった場合も、条件成立）
4	ローレベルセンス（OFFの時に、条件成立。最初からOFFだった場合も、条件成立）

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

パラメータの詳細については、YdxDoSetStartExternal関数 を参照してください。

使用例

サンプリング開始条件（外部トリガ）の設定を取得します。

C#

```
int result;
int diChannel;
int mode;
result = Ydx.DoGetStartExternal(id, out diChannel, out mode);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim diChannel As Integer
Dim mode As Integer
result = YdxDoGetStartExternal(id, diChannel, mode)
```

VB6.0

```
Dim result As Long
Dim diChannel As Long
Dim mode As Long
result = YdxDoGetStartExternal(id, diChannel, mode)
```

C++/CLI

```
int result;
int diChannel;
int mode;
result = YdxDoGetStartExternal(id, &diChannel, &mode);
```

C/C++

```
INT result;
INT diChannel;
INT mode;
result = YdxDoGetStartExternal(id, &diChannel, &mode);
```

YdxDoGetStopCondition

機能

サンプリング停止条件の設定を取得します。

書式

```
INT YdxDoGetStopCondition(
    INT id,
    INT* condition,
    INT* delay
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

condition

停止条件を格納する変数へのポインタを指定します。

値	意味
0	出力完了（データ終了）
1	外部トリガ

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

delay

遅延回数を格納する変数へのポインタを指定します。
本機種では遅延回数に0（遅延なし）しか設定できない為、必ず0が格納されます。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

パラメータの詳細については、YdxDoSetStopCondition関数 を参照してください。

使用例

サンプリング停止条件の設定を取得します。

C#

```
int result;
int condition;
int delay;
result = Ydx.DoGetStopCondition(id, out condition, out delay);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim condition As Integer
Dim delay As Integer
result = YdxDoGetStopCondition(id, condition, delay)
```

VB6.0

```
Dim result As Long
Dim condition As Long
Dim delay As Long
result = YdxDoGetStopCondition(id, condition, delay)
```

C++/CLI

```
int result;
int condition;
int delay;
result = YdxDoGetStopCondition(id, &condition, &delay);
```

C/C++

```
INT result;
INT condition;
INT delay;
result = YdxDoGetStopCondition(id, &condition, &delay);
```

YdxDoGetStopExternal

機能

サンプリング停止条件（外部トリガ）の設定を取得します。

書式

```
INT YdxDoGetStopExternal(
    INT id,
    INT* diChannel,
    INT* mode
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

diChannel

外部トリガとして使用するデジタル入力チャネルを格納する変数へのポインタを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

mode

動作モードを格納する変数へのポインタを指定します。

値	意味
0	立ち上がりエッジセンス（OFF→ONに変化した時に、条件成立）
1	立ち下がりエッジセンス（ON→OFFに変化した時に、条件成立）
2	両エッジセンス（OFF→ON または ON→OFFに変化した時に、条件成立）

値	意味
3	ハイレベルセンス（ONの時に、条件成立。最初からONだった場合も、条件成立）
4	ローレベルセンス（OFFの時に、条件成立。最初からOFFだった場合も、条件成立）

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

パラメータの詳細については、YdxDoSetStopExternal関数 を参照してください。

使用例

サンプリング停止条件（外部トリガ）の設定を取得します。

C#

```
int result;
int diChannel;
int mode;
result = Ydx.DoGetStopExternal(id, out diChannel, out mode);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim diChannel As Integer
Dim mode As Integer
result = YdxDoGetStopExternal(id, diChannel, mode)
```

VB6.0

```
Dim result As Long
Dim diChannel As Long
Dim mode As Long
result = YdxDoGetStopExternal(id, diChannel, mode)
```

C++/CLI

```
int result;
int diChannel;
int mode;
result = YdxDoGetStopExternal(id, &diChannel, &mode);
```

C/C++

```
INT result;
INT diChannel;
INT mode;
result = YdxDoGetStopExternal(id, &diChannel, &mode);
```

YdxDoSetData

機能

データを設定します。

書式

```
INT YdxDoSetData(
    INT id,
    INT sampleNum,
    INT* data
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

sampleNum

データを設定するサンプル数を指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

data

データを格納した変数へのポインタを指定します。
データは、以下のフォーマットで指定します。

サンプリング	MSB														LSB
1回目	OUT15	OUT14	OUT13	OUT12	OUT11	OUT10	OUT9	OUT8	OUT7	OUT6	OUT5	OUT4	OUT3	OUT2	OUT1	OUT0
2回目	OUT15	OUT14	OUT13	OUT12	OUT11	OUT10	OUT9	OUT8	OUT7	OUT6	OUT5	OUT4	OUT3	OUT2	OUT1	OUT0
3回目	OUT15	OUT14	OUT13	OUT12	OUT11	OUT10	OUT9	OUT8	OUT7	OUT6	OUT5	OUT4	OUT3	OUT2	OUT1	OUT0
・	OUT15	OUT14	OUT13	OUT12	OUT11	OUT10	OUT9	OUT8	OUT7	OUT6	OUT5	OUT4	OUT3	OUT2	OUT1	OUT0
・	OUT15	OUT14	OUT13	OUT12	OUT11	OUT10	OUT9	OUT8	OUT7	OUT6	OUT5	OUT4	OUT3	OUT2	OUT1	OUT0
・	OUT15	OUT14	OUT13	OUT12	OUT11	OUT10	OUT9	OUT8	OUT7	OUT6	OUT5	OUT4	OUT3	OUT2	OUT1	OUT0

設定範囲は、0～FFFFh（0～65535）です。
ただし、サンプリング時に出力が更新されるのは、チャネル設定 で高機能デジタル出力モードになっているチャネルのみです。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int[]	Integer	Long	int*	INT*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

データバッファにデータが残った状態のまま、本関数を実行した場合

- データバッファがリングバッファ形式に設定されている時は、残っていたデータは破棄されます。
- データバッファがFIFOバッファ形式に設定されている時は、残っていたデータの後に追加されます。

> ⚠ データバッファにデータが残った状態のまま、YdxDoSetBuffer関数 により設定が変更された場合、データはクリアされます。
> 設定を変更する場合は本関数の実行前におこなってください。

本関数は、データバッファがリングバッファ形式に設定されている時は、デジタル出力が 動作中 は実行できません。
（データバッファがFIFOバッファ形式に設定されている時は、デジタル出力が動作中でも実行できます）

使用例

1000サンプリング分のデータを設定します。
（データは、0, 1, 2 ・・・にする場合）

C#

```
int result;
int[] data = new int[1000];
int i;
for (i = 0; i < 1000; i++)
{
    data[i] = i;
```

```
}
result = Ydx.DoSetData(id, 1000, data);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim data(999) As Integer
Dim i As Integer
For i = 0 To 999
    data(i) = i
Next
result = YdxDoSetData(id, 1000, data)
```

VB6.0

```
Dim result As Long
Dim data(999) As Long
Dim i As Long
For i = 0 To 999
    data(i) = i
Next
result = YdxDoSetData(id, 1000, data(0))
```

C++/CLI

```
int result;
int data[1000];
int i;
for (i = 0; i < 1000; i++)
{
    data[i] = i;
}
result = YdxDoSetData(id, 1000, data);
```

C/C++

```
INT result;
INT data[1000];
INT i;
for (i = 0; i < 1000; i++)
{
    data[i] = i;
}
result = YdxDoSetData(id, 1000, data);
```

YdxDoClearData

機能

データをクリアします。

書式

```
INT YdxDoClearData(
    INT id
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

本関数が実行されると、以下の状態になります。

- データが、クリアされます。
- 状態 （ステータス）の「監視サンプル数」ビットが、0になります。
- 状態（出力済みサンプル数）が、0になります。
- 状態（動作済みリピート回数）が、0になります。
- 状態（未出力サンプル数）が、0になります。

本関数は、デジタル出力が 動作中 には実行できません。

使用例

データをクリアします。

C#

```
int result;
result = Ydx.DoClearData(id);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxDoClearData(id)
```

VB6.0

```
Dim result As Long
result = YdxDoClearData(id)
```

C++/CLI

```
int result;
result = YdxDoClearData(id);
```

C/C++

```
INT result;
result = YdxDoClearData(id);
```

YdxDoStart

機能

デジタル出力動作 を開始します。

書式

```
INT YdxDoStart(
    INT id
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

本関数は、デジタル出力が 動作中 には実行できません。

使用例

デジタル出力動作を開始します。

C#

```
int result;
result = Ydx.DoStart(id);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxDoStart(id)
```

VB6.0

```
Dim result As Long
result = YdxDoStart(id)
```

C++/CLI

```
int result;
result = YdxDoStart(id);
```

C/C++

```
INT result;
result = YdxDoStart(id);
```

YdxDoStop

機能

デジタル出力動作 を停止します。

書式

```
INT YdxDoStop(
    INT id
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

使用例

デジタル出力動作を停止します。

C#

```
int result;
result = Ydx.DoStop(id);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxDoStop(id)
```

VB6.0

```
Dim result As Long
result = YdxDoStop(id)
```

C++/CLI

```
int result;
result = YdxDoStop(id);
```

C/C++

```
INT result;
result = YdxDoStop(id);
```

YdxDoReset

機能

デジタル出力機能をリセットします。

書式

```
INT YdxDoReset(
    INT id
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

備考

本関数が実行されると、以下の状態になります。

- デジタル出力が 動作中 の場合、動作は停止されます。
- データが、クリアされます。
- 状態 （ステータス）の全てのビットが、0になります。
- 状態（出力済みサンプル数）が、0になります。
- 状態（動作済みリピート回数）が、0になります。
- 状態（未出力サンプル数）が、0になります。
- 設定値は、全て初期化されます。

出力の状態は変わりません。（機能・設定値のみリセットされます）
出力の状態も初期化したい場合は、本関数の実行後に YdxDoOutput関数 で出力データを0にしてください。

使用例

デジタル出力機能をリセットします。

C#

```
int result;
result = Ydx.DoReset(id);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxDoReset(id)
```

VB6.0

```
Dim result As Long
result = YdxDoReset(id)
```

C++/CLI

```
int result;
result = YdxDoReset(id);
```

C/C++

```
INT result;
result = YdxDoReset(id);
```

YdxDoGetStatus

機能

現在の状態を取得します。

書式

```
INT YdxDoGetStatus(
    INT id,
    INT* status,
    INT* sampleCount,
    INT* repeatCount,
    INT* notOutNum
);
```

パラメータ

id

YdxOpen関数 で取得したIDを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

status

ステータスを格納する変数へのポインタを指定します。
ビットごとに意味を持っていて、論理和された結果が格納されます。

値	定義名	ステータス
00000001h	YDX_STATUS_BUSY	動作中 動作が開始されると（YdxDoStart関数 が実行されると）オンになります。 動作が終了するとオフに戻ります。
00000002h	YDX_STATUS_SAMPLE_NUM	監視サンプル数 未出力サンプル数（データバッファにデータが残っているサンプル数）が監視サンプル数以下になるとオンになります。 監視サンプル数は、YdxDoSetCheckSampleNum関数 で変更できます。
00000004h	YDX_STATUS_START_TRIG	開始条件 成立済み 開始条件が成立するとオンになります。 リピートにより再び開始条件待ちになるとオフに戻ります。

値	定義名	ステータス
00000008h	YDX_STATUS_STOP_TRIG	停止条件 成立済み 停止条件が成立するとオンになります。 リピートにより開始条件が成立するとオフに戻ります。
00010000h	YDX_STATUS_SAMPLE_CLOCK_ERR	サンプリングクロック エラー発生 外部クロック使用時に、周期が早すぎる外部クロックが入力された場合にオンになります。 外部クロックの周期に問題がないか、チャタリング・ノイズが含まれていないか確認してください。
00040000h	YDX_STATUS_HARDWARE_ERR	ハードウェアエラー発生 ユニット内部回路に異常が検出した場合にオンになります。 通常ありえませんが、もし発生した場合は、どのような状況で発生したかを弊社サポートまでご連絡ください。
00080000h	YDX_STATUS_COMMUNICATE_ERR	通信エラー発生 USB通信に異常が検出された場合にオンになります。 近くにノイズ要因がないか確認してください。 確認しても問題が見当たらない場合は、どのような状況で発生したかを弊社サポートまでご連絡ください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

sampleCount

出力済みサンプル数を格納する変数へのポインタを指定します。
データバッファがFIFOバッファ形式に設定されている場合、2,147,483,647回を超えると0に戻ってカウントされます。
データバッファがリングバッファ形式に設定されている場合、リングバッファを一周するごとに0に戻ってカウントされます。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

repeatCount

動作済みリピート回数を格納する変数へのポインタを指定します。
2,147,483,647回を超えた場合、0に戻ってカウントされます。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

notOutNum

未出力サンプル数（データバッファにデータが残っているサンプル数）を格納する変数へのポインタを指定します。
リングバッファ形式に設定されている場合は、データを出力しても減算されません。
（データバッファにそのままデータが残る為）

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	out int	Integer	Long	int*	INT*

戻り値

関数が正常に終了した場合は、0（YDX_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は、0以外が返ります。
詳細は、戻り値一覧 を参照してください。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	int	Integer	Long	int	INT

使用例

現在の状態を取得します。

C#

```
int result;
int status;
int sampleCount;
int repeatCount;
int notOutNum;
result = Ydx.DoGetStatus(id, out status, out sampleCount, out repeatCount, out notOutNum);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim status As Integer
Dim sampleCount1 As Integer
Dim repeatCount As Integer
Dim notOutNum As Integer
result = YdxDoGetStatus(id, status, sampleCount, repeatCount, notOutNum)
```

VB6.0

```
Dim result As Long
Dim status As Long
Dim sampleCount1 As Long
Dim repeatCount As Long
Dim notOutNum As Long
result = YdxDoGetStatus(id, status, sampleCount, repeatCount, notOutNum)
```

C++/CLI

```
int result;
int status;
int sampleCount;
int repeatCount;
result = YdxDoGetStatus(id, &status, &sampleCount, &repeatCount);
```

C/C++

```
INT result;
INT status;
INT sampleCount;
INT repeatCount;
INT notOutNum;
result = YdxDoGetStatus(id, &status, &sampleCount, &repeatCount, &notOutNum);
```

ソフトウェアパック

弊社ホームページよりダウンロードする（※）、サンプルプログラム・ユーティリティなどをひとまとめにしたもの
※ CD-ROMでの提供も可能（有償）

デジタル入力動作

デジタル入力動作とは、高機能デジタル入力 での動作開始から動作停止までの一連の動作の事です。

参考

- 用語説明（動作中）
- 機能説明（開始条件・停止条件・リピート）

デジタル入力動作中

動作中とは、動作開始から動作停止するまでの期間です。
開始条件までの待機・サンプリングの期間が含まれます。

動作開始は、YdxDiStart関数 の実行によっておこなわれます。

動作停止は、以下のいずれかの要因によっておこなわれます。

- 設定された条件でのサンプリングが終了
- エラー（サンプリングエラー・オーバランエラー・ハードウェアエラー・通信エラー）
- YdxDiStop関数 の実行

参考

- 機能説明（開始条件・停止条件・リピート）

デジタル出力動作

デジタル出力動作とは、高機能デジタル出力 での動作開始から動作停止までの一連の動作の事です。

参考

- 用語説明（動作中）
- 機能説明（開始条件・停止条件・リピート）

デジタル出力動作中

動作中とは、動作開始から動作停止するまでの期間です。
開始条件までの待機・サンプリングの期間が含まれます。

動作開始は、YdxDoStart関数 の実行によっておこなわれます。

動作停止は、以下のいずれかの要因によっておこなわれます。

- 設定された条件でのサンプリングが終了
- エラー（サンプリングエラー・ハードウェアエラー・通信エラー）
- YdxDoStop関数 の実行

参考

- 機能説明（開始条件・停止条件・リピート）

注意事項

本製品及び本書の内容については、改良のために予告なく変更することがあります。

本製品を運用した結果の他への影響については、上記にかかわらず責任を負いかねますのでご了承ください。

本製品は人命にかかわる設備や機器、及び高度な信頼性を必要とする設備や機器としての使用またはこれらに組み込んでの使用は意図されておりません。
これら、設備や機器、制御システムなどに本製品を使用され、本製品の故障により人身事故、火災事故、損害などが生じても、弊社ではいかなる責任も負いかねます。
設備や機器、制御システムなどにおいて、安全設計に万全を期されるようご注意願います。

本製品は「外国為替及び外国貿易法」の規定により戦略物資等輸出規制製品に該当する場合があります。
国外に持ち出す際には、日本国政府の輸出許可申請などの手続きが必要になる場合があります。

本製品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、弊社は一切の責任を負いかねます。
また、弊社は本製品に関し、日本国外への技術サポート等をおこなっておりませんので、予めご了承ください。