株式会社ワイツー

DAQ-Xシリーズ アナログ入出力ユニット
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 アナログ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。

簡易アナログ入力

指定したチャネルのアナログ入力を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。

- YdxAiInput関数
- YdxAiInputVolt関数

一方、連続サンプリングなど色々な条件でアナログ入力をおこないたい場合は 高機能アナログ入力 を使用します。

参考

- 実行手順（簡易アナログ入力）
- サンプルプログラム（AiInputVolt）

高機能アナログ入力

連続サンプリングなど、色々な条件でアナログ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、アナログ入力を１回おこないたい場合は 簡易アナログ入力 を使用します。

参考

- 実行手順（高機能アナログ入力）
- サンプルプログラム（AiPolling）

  高機能アナログ入力のサンプルプログラムです。
  動作状態の監視をポーリングでおこなっています。
- サンプルプログラム（AiEvent）

  高機能アナログ入力のサンプルプログラムです。
  動作状態の監視をイベント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 YdxAiSetBuffer関数 を使用します。
- データの読み出しには YdxAiGetData関数 または YdxAiGetDataVolt関数 を使用します。
- データをクリアするには YdxAiClearData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。

- YdxAiSetClock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。

- YdxAiSetClockInternal関数

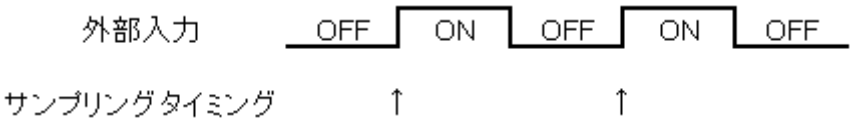
外部クロック

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。

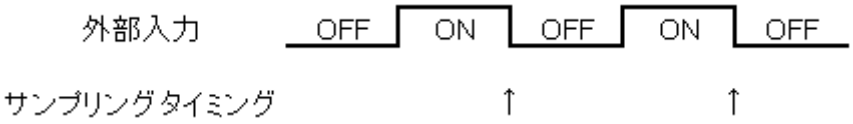
- YdxAiSetClockExternal関数

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。

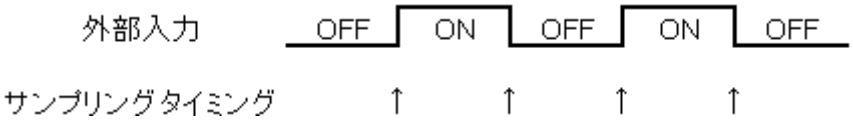
立ち上がりエッジセンス



立ち下がりエッジセンス



両エッジセンス



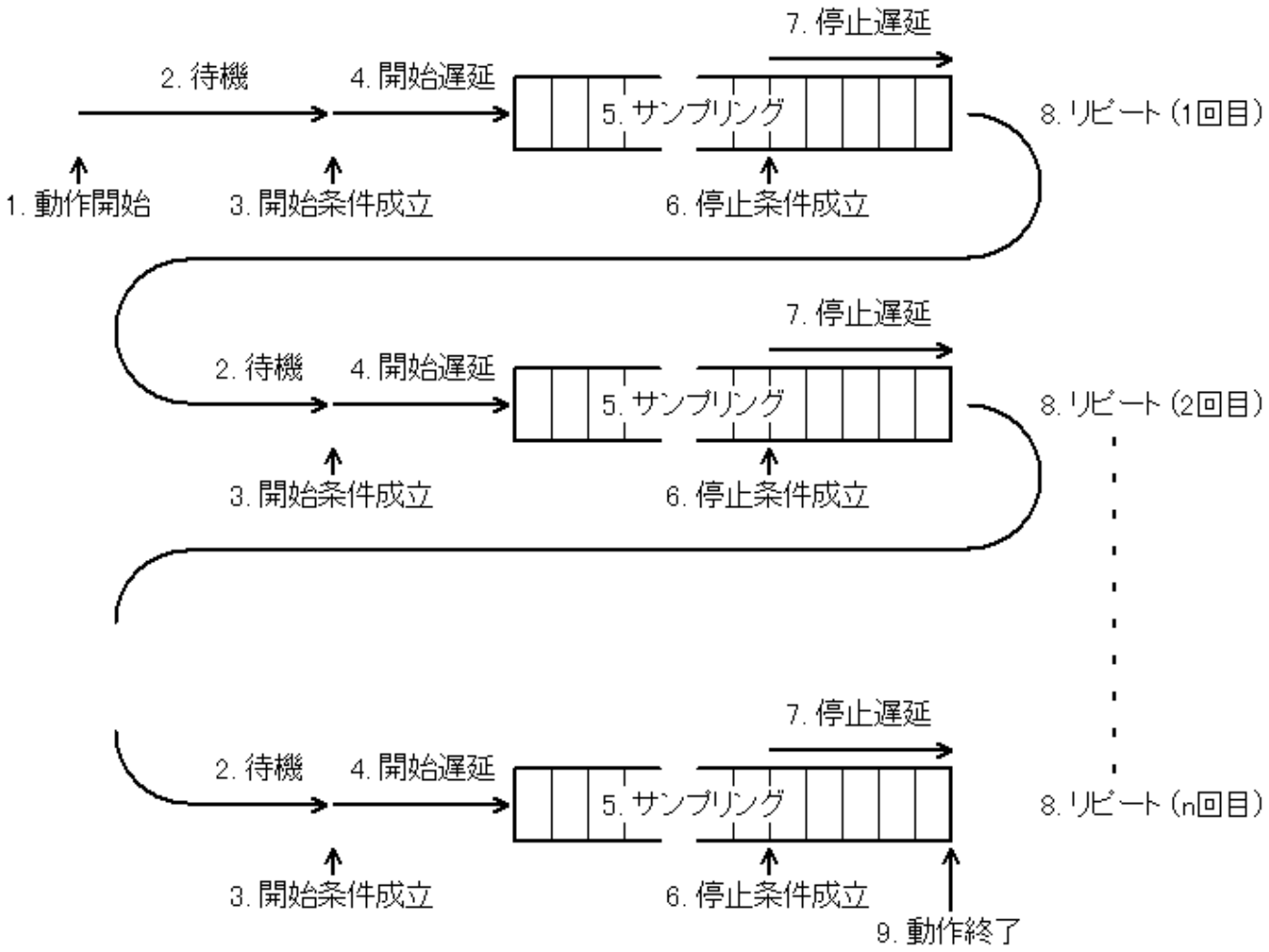
開始条件・停止条件・リピート

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 YdxAiSetStartCondition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 YdxAiSetStopCondition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 YdxAiSetRepeat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。
   - YdxAiStart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. 遅延回数のサンプリングクロックを待ってからサンプリングを開始します。
5. サンプリングをおこないます。
6. 停止条件の成立を検出します。

7. 遅延回数のサンプリング後にサンプリングを停止します。
8. リピート設定回数分、2〜7を繰り返します。
9. リピートが完了したら、動作を終了します。

外部トリガ

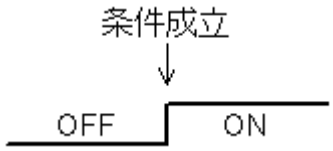
外部トリガとは、外部からのデジタル入力を サンプリング開始条件・停止条件 として使用する事です。
外部トリガとして使用するデジタル入力チャネルと動作モードの設定には、以下の関数を使用します。

- サンプリング開始条件
  - YdxAiSetStartExternal関数
- サンプリング停止条件
  - YdxAiSetStopExternal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。

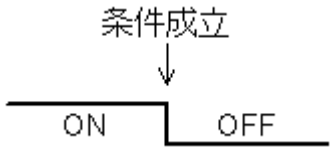
立ち上がりエッジセンス

OFF→ONに変化した時に、条件成立。



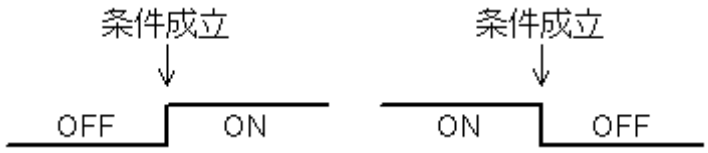
立ち下がりエッジセンス

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）

レベル比較トリガ（アナログ入力トリガ）

アナログ入力トリガとは、外部からのアナログ入力を サンプリング開始条件・停止条件 として使用する事です。
レベル比較トリガは、アナログ入力状態がしきい値以上またはしきい値以下になった時に条件成立とします。

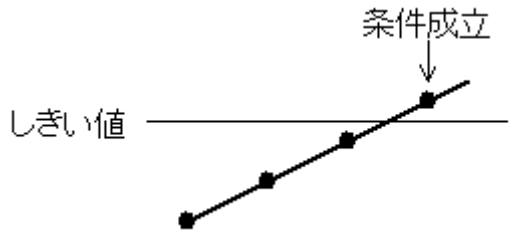
アナログ入力トリガとして使用するアナログ入力チャネルと動作モードの設定には、以下の関数を使用します。

- サンプリング開始条件
  - YdxAiSetStartLevel関数 または YdxAiSetStartLevelVolt関数
- サンプリング停止条件
  - YdxAiSetStopLevel関数 または YdxAiSetStopLevelVolt関数

動作モード

立ち上がりエッジセンス

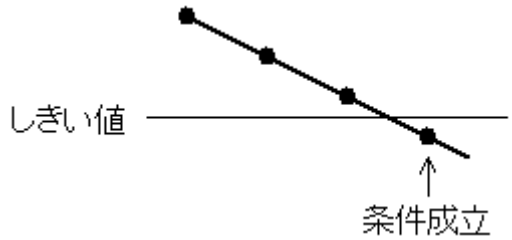
しきい値未満から、しきい値以上に変化した時に、条件成立。



※ 図は、太線がアナログ入力信号、黒丸がサンプリングタイミングを表しています。

立ち下がりエッジセンス

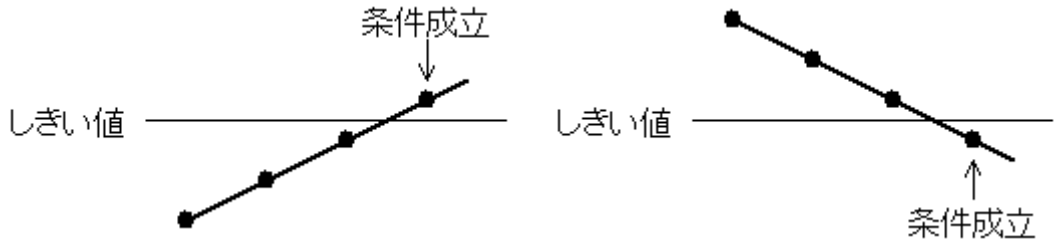
しきい値を超える値から、しきい値以下に変化した時に、条件成立。



※ 図は、太線がアナログ入力信号、黒丸がサンプリングタイミングを表しています。

両エッジセンス

「しきい値未満から、しきい値以上に変化した時」または「しきい値を超える値から、しきい値以下に変化した時」に、条件成立。



※ 図は、太線がアナログ入力信号、黒丸がサンプリングタイミングを表しています。

ハイレベルセンス

しきい値以上の時に、条件成立。
最初からしきい値以上だった場合も、条件成立。
（最初、しきい値未満だった場合は、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

しきい値以下の時に、条件成立。
最初からしきい値以下だった場合も、条件成立。
（最初、しきい値を超える値だった場合は、立ち下がりエッジセンスと同じタイミング）

インレンジ比較トリガ（アナログ入力トリガ）

アナログ入力トリガとは、外部からのアナログ入力を サンプリング開始条件・停止条件 として使用する事です。
インレンジ比較トリガは、アナログ入力状態が２つのしきい値の範囲内になった時に条件成立とします。

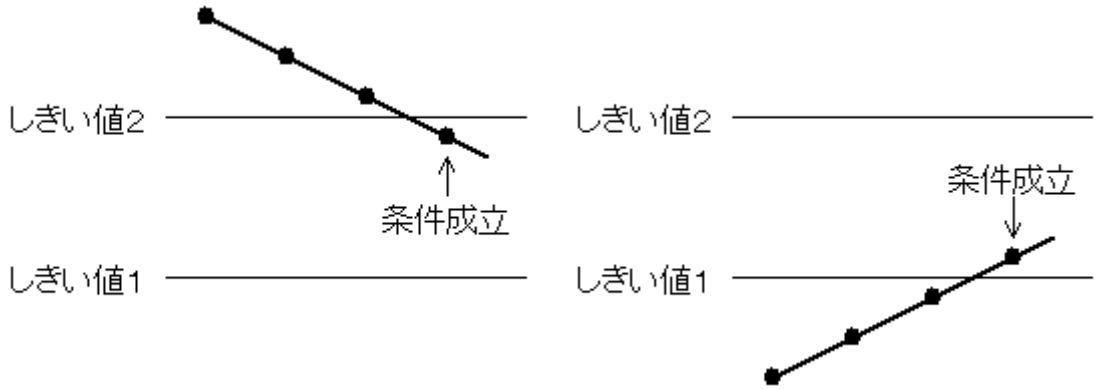
アナログ入力トリガとして使用するアナログ入力チャネルと動作モードの設定には、以下の関数を使用します。

- サンプリング開始条件
  - YdxAiSetStartInRange関数 または YdxAiSetStartInRangeVolt関数
- サンプリング停止条件用
  - YdxAiSetStopInRange関数 または YdxAiSetStopInRangeVolt関数

動作モード

エッジセンス

２つのしきい値の範囲外から範囲内に変化した時に、条件成立。



※ 図は、太線がアナログ入力信号、黒丸がサンプリングタイミングを表しています。

レベルセンス

２つのしきい値の範囲内の時に、条件成立。
最初から範囲内だった場合も、条件成立。
（最初、範囲外だった場合は、エッジセンスと同じタイミング）

アウトレンジ比較トリガ（アナログ入力トリガ）

アナログ入力トリガとは、外部からのアナログ入力を サンプリング開始条件・停止条件 として使用する事です。
アウトレンジ比較トリガは、アナログ入力状態が２つのしきい値の範囲外になった時に条件成立とします。

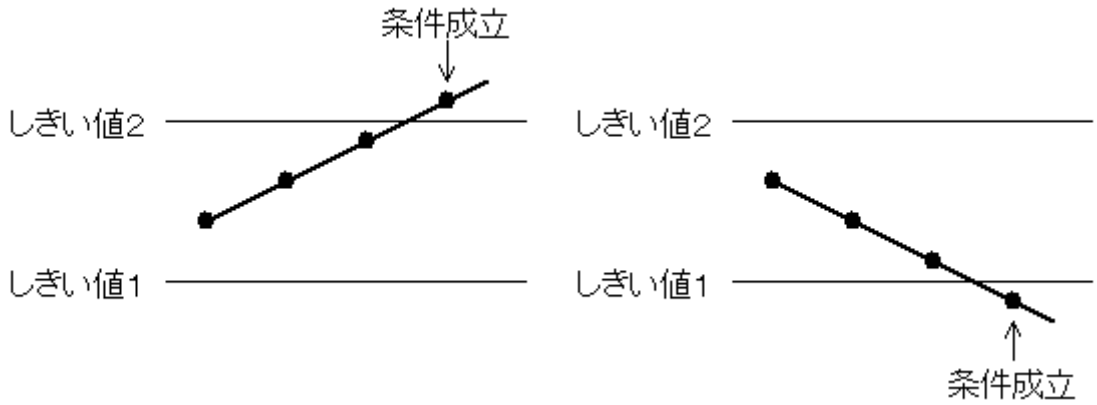
アナログ入力トリガとして使用するアナログ入力チャネルと動作モードの設定には、以下の関数を使用します。

- サンプリング開始条件
  - YdxAiSetStartOutRange関数 または YdxAiSetStartOutRangeVolt関数
- サンプリング停止条件
  - YdxAiSetStopOutRange関数 または YdxAiSetStopOutRangeVolt関数

動作モード

エッジセンス

２つのしきい値の範囲内から、範囲外に変化した時に、条件成立。



※ 図は、太線がアナログ入力信号、黒丸がサンプリングタイミングを表しています。

レベルセンス

２つのしきい値の範囲外の時に、条件成立。
最初から範囲外だった場合も、条件成立。
（最初、範囲内だった場合は、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したり、イベントを発生させる事ができます。

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。

- YdxAiGetStatus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。
- YdxAiSetEvent関数 で、監視サンプル数イベントを有効に設定してある場合、イベントが発生します。

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

簡易アナログ出力

指定したチャネルのアナログ出力を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。

- YdxAoOutput関数
- YdxAoOutputVolt関数

一方、連続サンプリングなど色々な条件でアナログ出力をおこないたい場合は 高機能アナログ出力 を使用します。

参考

- 実行手順（簡易アナログ出力）
- サンプルプログラム（AoOutputVolt）

高機能アナログ出力

連続サンプリングなど、色々な条件でアナログ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、アナログ出力を１回おこないたい場合は 簡易アナログ出力 を使用します。

参考

- 実行手順（高機能アナログ出力）
- サンプルプログラム（AoPolling）

  高機能アナログ出力のサンプルプログラムです。
  動作状態の監視をポーリングでおこなっています。
- サンプルプログラム（AoEvent）

  高機能アナログ出力のサンプルプログラムです。
  動作状態の監視をイベント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、YdxAoSetBuffer関数 を使用します。
- データの設定には YdxAoSetData関数 または YdxAoSetDataVolt関数 を使用します。
- データをクリアするには YdxAoClearData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。

- YdxAoSetClock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。

- YdxAoSetClockInternal関数

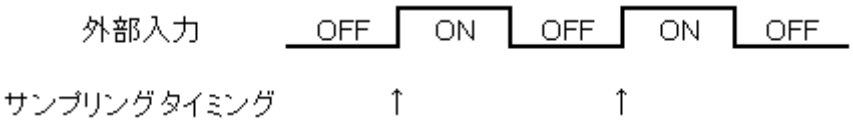
外部クロック

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。

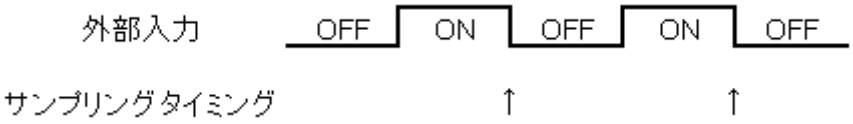
- YdxAoSetClockExternal関数

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。

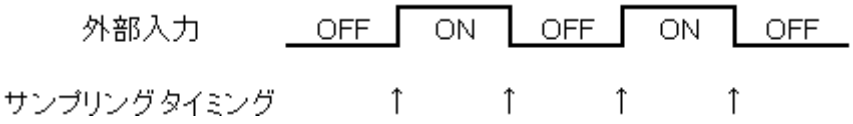
立ち上がりエッジセンス



立ち下がりエッジセンス



両エッジセンス



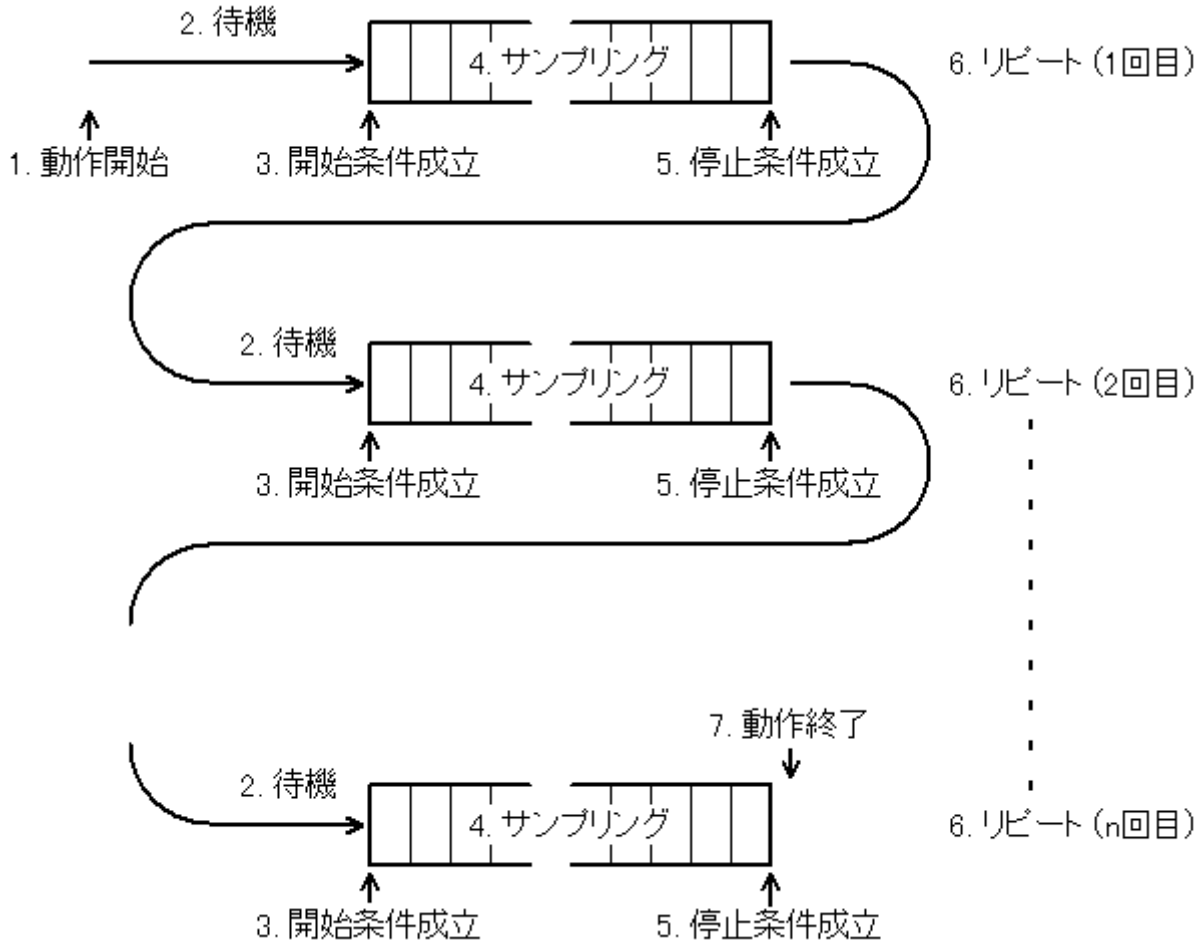
開始条件・停止条件・リピート

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 YdxAoSetStartCondition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 YdxAoSetStopCondition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 YdxAoSetRepeat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。
   - YdxAoStart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。

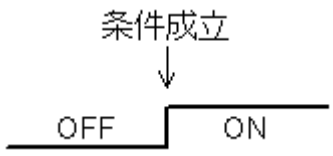
外部トリガとして使用するデジタル入力チャネルと動作モードの設定には、以下の関数を使用します。

- サンプリング開始条件
  - YdxAoSetStartExternal関数
- サンプリング停止条件
  - YdxAoSetStopExternal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。

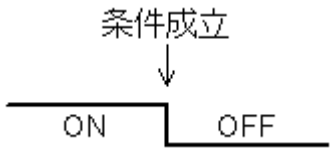
立ち上がりエッジセンス

OFF→ONに変化した時に、条件成立。



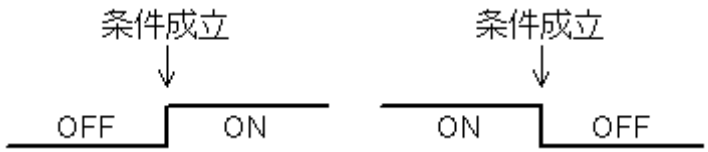
立ち下がりエッジセンス

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）

レベル比較トリガ
アナログ入力トリガ

アナログ入力トリガとは、外部からのアナログ入力を サンプリング開始条件・停止条件 として使用する事です。
レベル比較トリガは、アナログ入力状態がしきい値以上またはしきい値以下になった時に条件成立とします。

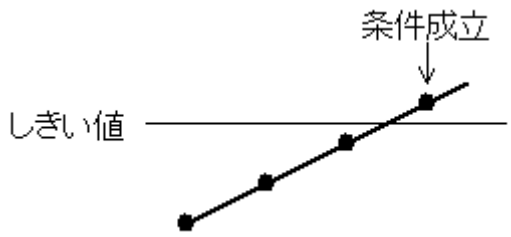
アナログ入力トリガとして使用するアナログ入力チャネルと動作モードの設定には、以下の関数を使用します。

- サンプリング開始条件
  - YdxAoSetStartLevel関数
  - YdxAoSetStartLevelVolt関数
- サンプリング停止条件
  - YdxAoSetStopLevel関数
  - YdxAoSetStopLevelVolt関数

動作モード

立ち上がりエッジセンス

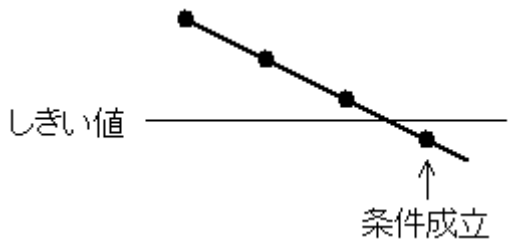
しきい値未満から、しきい値以上に変化した時に、条件成立。



※ 図は、太線がアナログ入力信号、黒丸がサンプリングタイミングを表しています。

立ち下がりエッジセンス

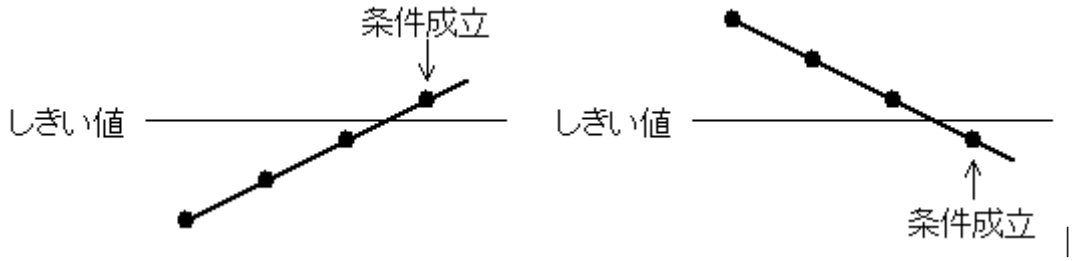
しきい値を超える値から、しきい値以下に変化した時に、条件成立。



※ 図は、太線がアナログ入力信号、黒丸がサンプリングタイミングを表しています。

両エッジセンス

「しきい値未満から、しきい値以上に変化した時」または「しきい値を超える値から、しきい値以下に変化した時」に、条件成立。



※ 図は、太線がアナログ入力信号、黒丸がサンプリングタイミングを表しています。

ハイレベルセンス

しきい値以上の時に、条件成立。
最初からしきい値以上だった場合も、条件成立。
（最初、しきい値未満だった場合は、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

しきい値以下の時に、条件成立。
最初からしきい値以下だった場合も、条件成立。
（最初、しきい値を超える値だった場合は、立ち下がりエッジセンスと同じタイミング）

インレンジ比較トリガ（アナログ入力トリガ）

アナログ入力トリガとは、外部からのアナログ入力を サンプリング開始条件・停止条件 として使用する事です。
インレンジ比較トリガは、アナログ入力状態が２つのしきい値の範囲内になった時に条件成立とします。

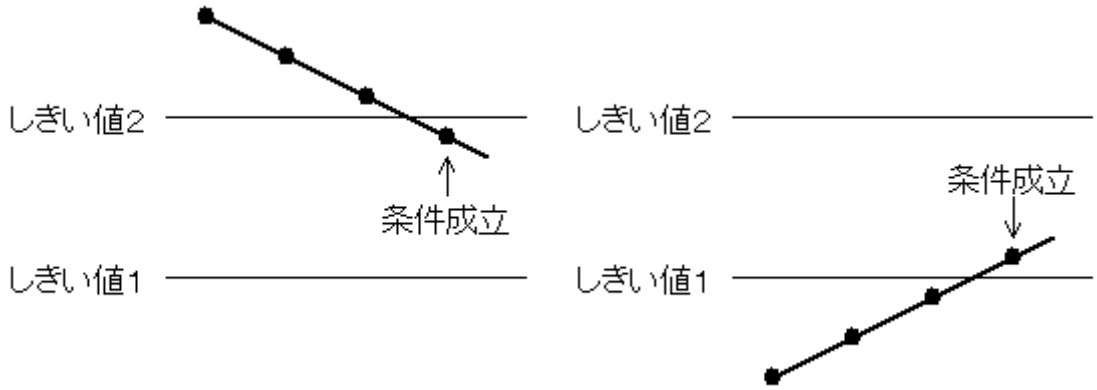
アナログ入力トリガとして使用するアナログ入力チャネルと動作モードの設定には、以下の関数を使用します。

- サンプリング開始条件
  - YdxAoSetStartInRange関数 または YdxAoSetStartInRangeVolt関数
- サンプリング停止条件
  - YdxAoSetStopInRange関数 または YdxAoSetStopInRangeVolt関数

動作モード

エッジセンス

２つのしきい値の範囲外から範囲内に変化した時に、条件成立。



※ 図は、太線がアナログ入力信号、黒丸がサンプリングタイミングを表しています。

レベルセンス

２つのしきい値の範囲内の時に、条件成立。
最初から範囲内だった場合も、条件成立。
（最初、範囲外だった場合は、エッジセンスと同じタイミング）

アウトレンジ比較
（アナログ入力トリガ）

アナログ入力トリガとは、外部からのアナログ入力を サンプリング開始条件・停止条件 として使用する事です。
アウトレンジ比較トリガは、アナログ入力状態が２つのしきい値の範囲外になった時に条件成立とします。

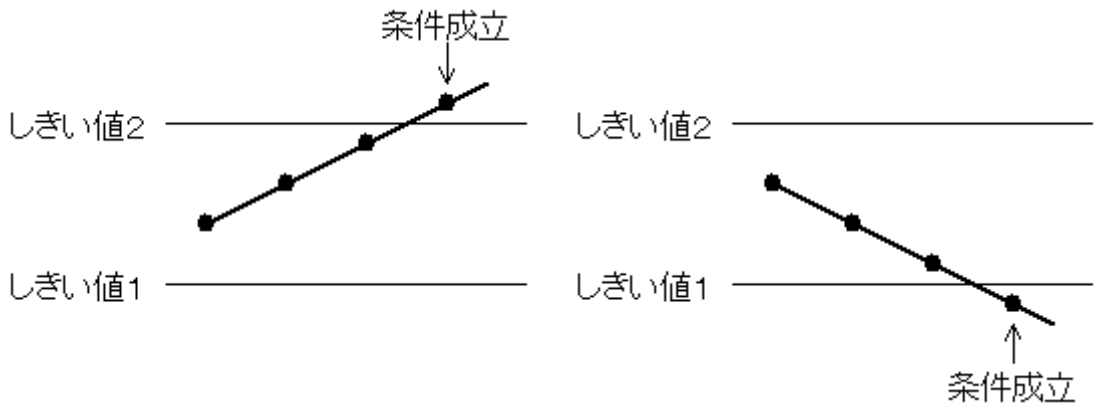
アナログ入力トリガとして使用するアナログ入力チャネルと動作モードの設定には、以下の関数を使用します。

- サンプリング開始条件
  - YdxAoSetStartOutRange関数
  - YdxAoSetStartOutRangeVolt関数
- サンプリング停止条件
- YdxAoSetStopOutRange関数
- YdxAoSetStopOutRangeVolt関数

動作モード

エッジセンス

２つのしきい値の範囲内から、範囲外に変化した時に、条件成立。



※ 図は、太線がアナログ入力信号、黒丸がサンプリングタイミングを表しています。

レベルセンス

２つのしきい値の範囲外の時に、条件成立。
最初から範囲外だった場合も、条件成立。
（最初、範囲内だった場合は、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したり、イベントを発生させる事ができます。

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。

- YdxAoGetStatus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。
- YdxAoSetEvent関数 で、監視サンプル数イベントを有効に設定してある場合、イベントが発生します。

監視サンプル数は、YdxAoSetCheckSampleNum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

アナログ入力・アナログ出力・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。

- アナログ入力

  入力電圧値を100msec毎に表示します。

- アナログ出力

  出力したい電圧値を入力してから「出力」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

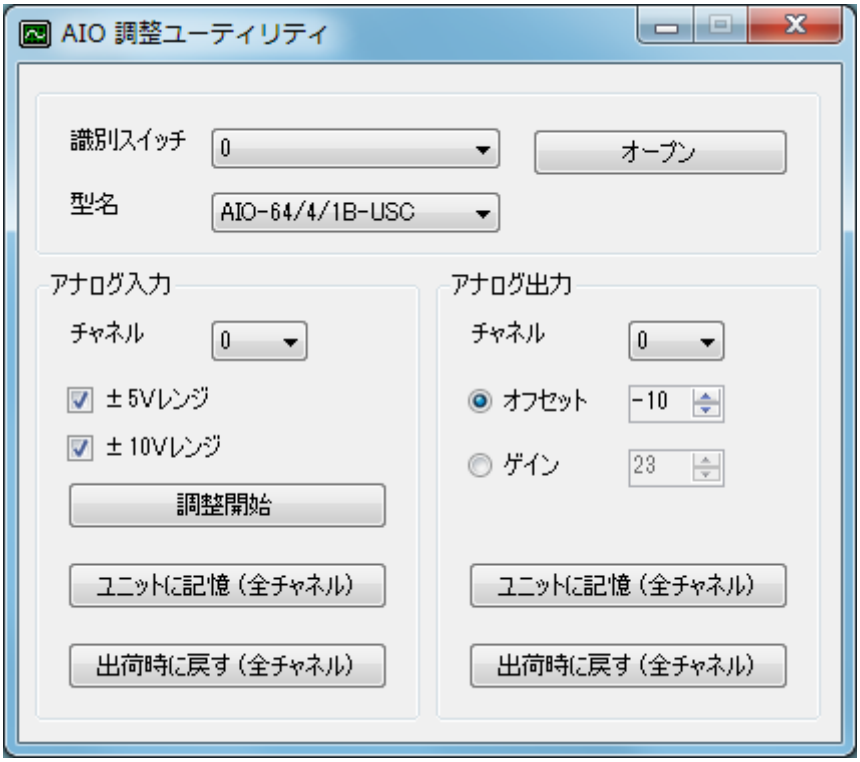
  出力ON/OFFを切り替えるには番号をクリックしてください。
  出力がONの場合は緑、OFFの場合は灰色で表示します。

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。

アナログ入出力調整用ユーティリティ

弊社にて調整をして出荷していますが、お客様の環境で再度調整が必要な場合は、このユーティリティを使用して調整する事ができます。
調整をする際は、ユニットを30分以上通電させた後に行ってください。



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。

アナログ入力

基準電圧発生器を用意してください。

1. 調整をしたいチャネルを選択して、基準電圧発生器を接続してください。
2. 調整をしたいレンジを選択して（複数選択可能）「調整開始」をクリックしてください。
3. 「CHxに0Vを入力してから、OKをクリックしてください」と表示されますので、0Vを入力して「OK」をクリックしてください。
4. 「CHxに+10Vを入力してから、OKをクリックしてください」と表示されますので、10Vを入力して「OK」をクリックしてください。（±10Vレンジを選択した場合）
   「CHxに+5Vを入力してから、OKをクリックしてください」と表示されますので、5Vを入力して「OK」をクリックしてください。（±5Vレンジを選択した場合）
5. 「調整が完了しました」と表示されれば、調整は終了です。

他のチャネルも調整したい場合は、1～5を繰り返してください。

「ユニットに記憶（全チャネル）」をクリックすると、調整内容がユニットに記憶されます。
選択されているチャネルに関わらず、全チャネルの調整内容が記憶されます。

「出荷時に戻す（全チャネル）」をクリックすると、出荷時の調整値に戻ります。
選択されているチャネルに関わらず、全チャネルが出荷時の調整値に戻ります。

アナログ出力

精度の良いマルチメーターを用意してください。

1. 調整をしたいチャネルを選択して、マルチメーターを接続してください。
2. オフセットまたはゲインを選択してください。（オフセットを調整後、ゲインを調整してください）
3. オフセットの場合は0V、ゲインの場合は+10Vに近づくように値を△▽ボタンで調整してください。

他のチャネルも調整したい場合は、1〜3を繰り返してください。

「ユニットに記憶（全チャネル）」をクリックすると、調整内容がユニットに記憶されます。
選択されているチャネルに関わらず、全チャネルの調整内容（変更された値のみ）が記憶されます。

「出荷時に戻す（全チャネル）」をクリックすると、出荷時の調整値に戻ります。
選択されているチャネルに関わらず、全チャネルが出荷時の調整値に戻ります。

その他

調整をおこなった後、ユニットに記憶をせずにユーティリティを閉じようとすると「調整値をユニットに記憶させますか?」と表示されます。
「はい」をクリックすると、ユニットに調整値を記憶させてから、終了します。
「いいえ」をクリックすると、調整値は破棄されます。

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。

アナログ入力

サンプル名	動作概要
AiInputVolt	簡易アナログ入力 のサンプルプログラムです。 アナログ入力を1回おこない、電圧値で表示します。
AiPolling	高機能アナログ入力 のサンプルプログラムです。 アナログ入力を1000回おこない、電圧値で表示します。 動作状態の監視をポーリングでおこなっています。
AiEvent	高機能アナログ入力 のサンプルプログラムです。 アナログ入力を1000回おこない、電圧値で表示します。 動作状態の監視をイベントでおこなっています。
AiFile	高機能アナログ入力 のサンプルプログラムです。 アナログ入力を連続でおこない、ファイルに保存します。
AiChart	高機能アナログ入力 のサンプルプログラムです。 アナログ入力を連続でおこない、波形をグラフ表示します。

アナログ出力

サンプル名	動作概要
AoOutputVolt	簡易アナログ出力 のサンプルプログラムです。 アナログ出力を1回おこないます。
AoPolling	高機能アナログ出力 のサンプルプログラムです。 アナログ出力を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。
AoEvent	高機能アナログ出力 のサンプルプログラムです。 アナログ出力を1000回おこないます。 動作状態の監視をイベントでおこなっています。
AoFile	高機能アナログ出力 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、アナログ出力をおこないます。

デジタル入出力

サンプル名	動作概要
Dio	デジタル入力・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

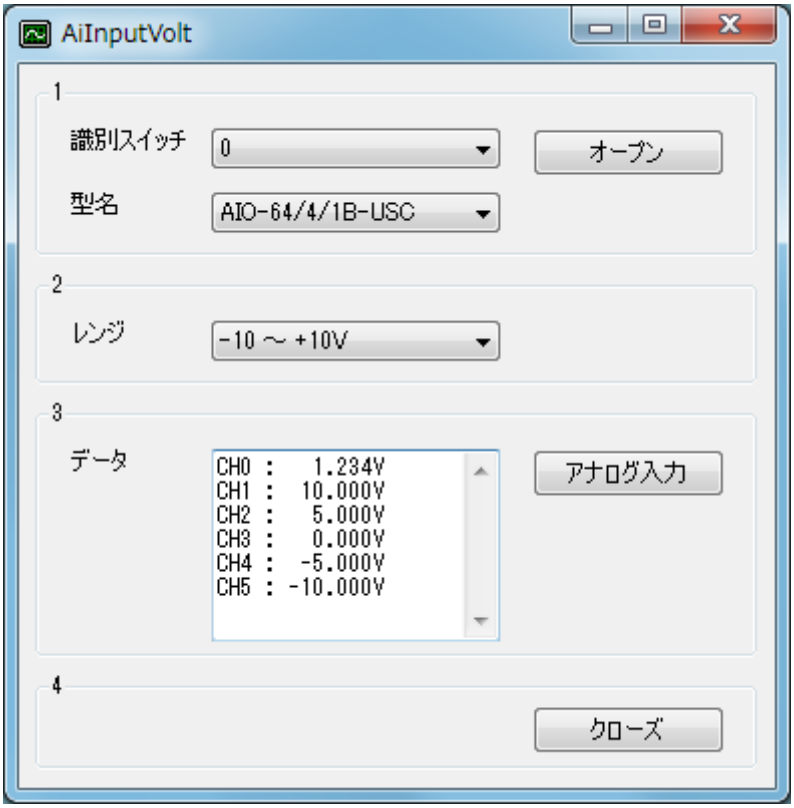
  Visual Basic .NET2003のプロジェクトを変換して使用してください。
- Visual C++ 2008以降

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

AiInputVolt

簡易アナログ入力 のサンプルプログラムです。
アナログ入力を1回おこない、電圧値で表示します。

画面



1. オープン

   ユニットのオープンをします。

2. 設定

   入力レンジを設定します。

3. アナログ入力

   アナログ入力を1回おこない、電圧値で表示します。

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

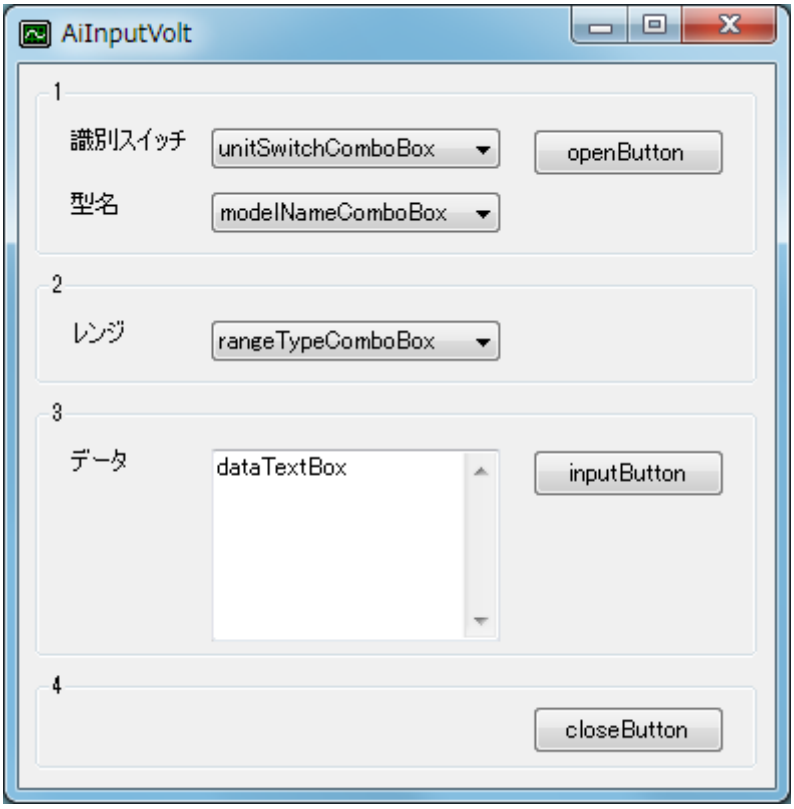
C#

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_AI_EXCEED_DATA_NUM:
        case Ydx.YDX_RESULT_AI_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[] { "AIO-64/4/1B-USC", "AIO-60/4/1B-USC" });
    modelNameComboBox.SelectedIndex = 0;

    // レンジ
    rangeTypeComboBox.ResetText();
    rangeTypeComboBox.Items.AddRange(new string[] { "-10 ～ +10V", "-5 ～ +5V" });
}
```

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;
        rangeTypeComboBox.SelectedIndexChanged -= new
EventHandler(rangeTypeComboBox_SelectedIndexChanged);
        rangeTypeComboBox.SelectedIndex = 0;
        rangeTypeComboBox.SelectedIndexChanged += new
EventHandler(rangeTypeComboBox_SelectedIndexChanged);
        ResultShow("オープン", result);
    }
}
```

レンジ切り替え

```
private void rangeTypeComboBox_SelectedIndexChanged(object sender, EventArgs e)
{
    int result = Ydx.AiSetRange(id, rangeTypeComboBox.SelectedIndex);
    if (result != 0)
        ResultShow("YdxAiSetRange", result);
}
```

アナログ入力

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

    const int CHANNEL_NUM = 6;
    float[] data = new float[CHANNEL_NUM];
    int result = Ydx.AiInputVolt(id, 0, CHANNEL_NUM, data);
    if (result != 0)
    {
        ResultShow("YdxAiInputVolt", result);
        return;
    }

    string txt = "";
    for (int channel = 0; channel < CHANNEL_NUM; channel++)
    {
        txt += "CH" + channel.ToString() + " : " + data[channel].ToString("
0.000;-0.000").PadLeft(7) + "V" + 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

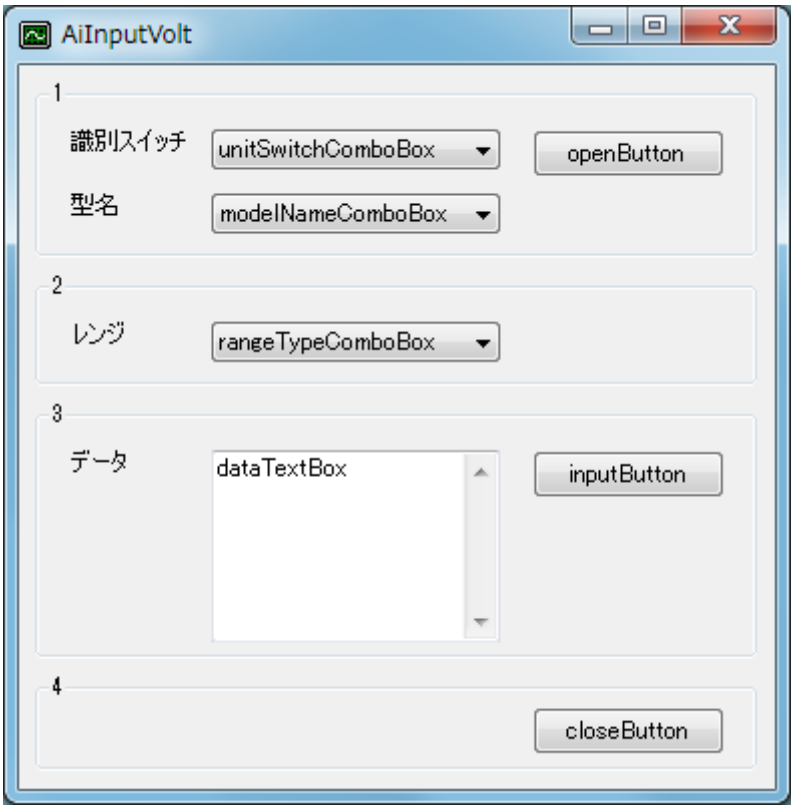
```
private void Form1_Closing(object sender, System.ComponentModel.CancelEventArgs e)
{
    int result = Ydx.Close(id);
    if ((result != 0) && (result != Ydx.YDX_RESULT_NOT_OPEN))
    {
        ResultShow("YdxClose", result);
    }
}
```

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 Short
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() { "AIO-64/4/1B-USC", "AIO-60/4/1B-USC" })
    modelNameComboBox.SelectedIndex = 0

    ' レンジ
    rangeTypeComboBox.ResetText()
    rangeTypeComboBox.Items.AddRange(New String() { "-10 ～ +10V", "-5 ～ +5V" })
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
        RemoveHandler rangeTypeComboBox.SelectedIndexChanged, AddressOf
rangeTypeComboBox_SelectedIndexChange
        rangeTypeComboBox.SelectedIndex = 0
        AddHandler rangeTypeComboBox.SelectedIndexChanged, AddressOf
rangeTypeComboBox_SelectedIndexChange
        ResultShow("オープン", result)
    End If
End Sub
```

レンジ切り替え

```
Private Sub rangeTypeComboBox_SelectedIndexChange(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles rangeTypeComboBox.SelectedIndexChanged
    result = YdxAiSetRange(id, rangeTypeComboBox.SelectedIndex)
    If result <> 0 Then
        ResultShow("YdxAiSetRange", result)
    End If
End Sub
```

アナログ入力

```
Private Sub inputButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles inputButton.Click
    dataTextBox.ResetText()
    Application.DoEvents()
    Const CHANNEL_NUM As Integer = 6
    Dim data(CHANNEL_NUM - 1) As Single
```

```
    result = YdxAiInputVolt(id, 0, CHANNEL_NUM, data)
    If result <> 0 Then
        ResultShow("YdxAiInput", result)
        Exit Sub
    End If

    Dim txt As String = ""
    Dim channel As Integer
    For channel = 0 To CHANNEL_NUM - 1
        txt += "CH" + channel.ToString() + " : " + data(channel).ToString("
0.000;-0.000").PadLeft(7) + "V" + 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

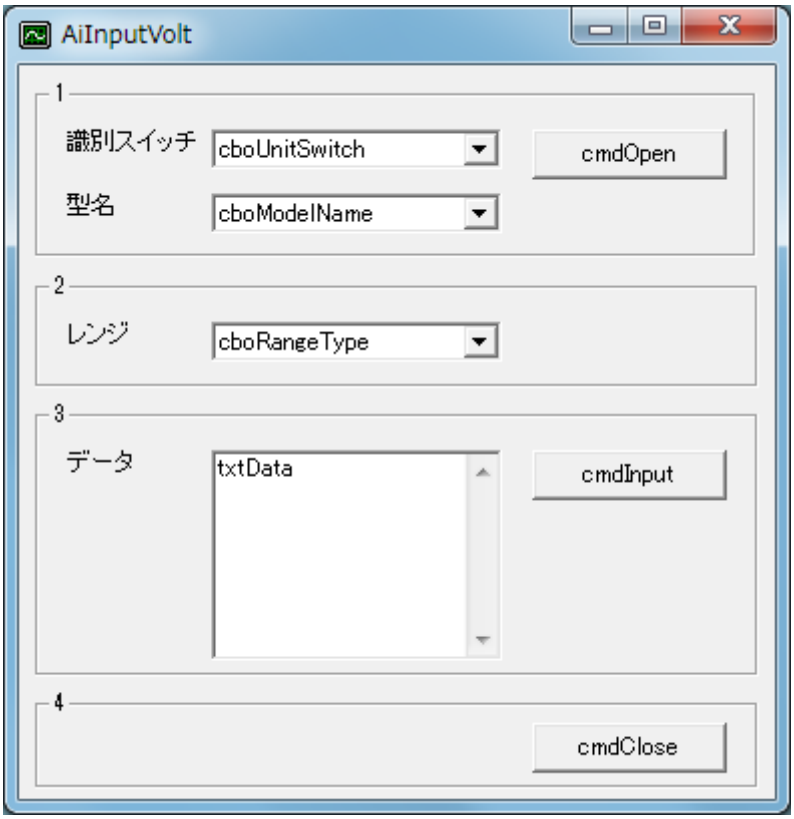
```
Private Sub Form1_Closing(ByVal sender As Object, ByVal e As
System.ComponentModel.CancelEventArgs) Handles MyBase.Closing
    result = YdxClose(id)
    If result <> 0 And result <> YDX_RESULT_NOT_OPEN Then
        ResultShow("YdxClose", result)
    End If
End Sub
```

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_AI_EXCEED_DATA_NUM, Ydx.YDX_RESULT_AI_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 "AIO-64/4/1B-USC"
    cboModelName.AddItem "AIO-60/4/1B-USC"
    cboModelName.ListIndex = 0

    ' レンジ
    cboRangeType.AddItem "-10 ～ +10V"
    cboRangeType.AddItem "-5 ～ +5V"
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
        cboRangeType.ListIndex = 0
        Call ResultShow("オープン", result)
    End If
End Sub
```

レンジ切り替え

```
Private Sub cboRangeType_Click()
    result = YdxAiSetRange(id, cboRangeType.ListIndex)
    If result <> 0 Then
        Call ResultShow("YdxAiSetRange", result)
    End If
End Sub
```

アナログ入力

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

    Const CHANNEL_NUM As Long = 6
    Dim data(CHANNEL_NUM - 1) As Single
    result = YdxAiInputVolt(id, 0, CHANNEL_NUM, data(0))
    If result <> 0 Then
        Call ResultShow("YdxAiInput", result)
        Exit Sub
    End If

    Dim txt As String
    txt = ""
    Dim channel As Long
    For channel = 0 To CHANNEL_NUM - 1
        txt = txt + "CH" + Format(channel) + " : " + Right(" " & Format(data(channel), "
0.000;-0.000"), 7) + "V" + 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

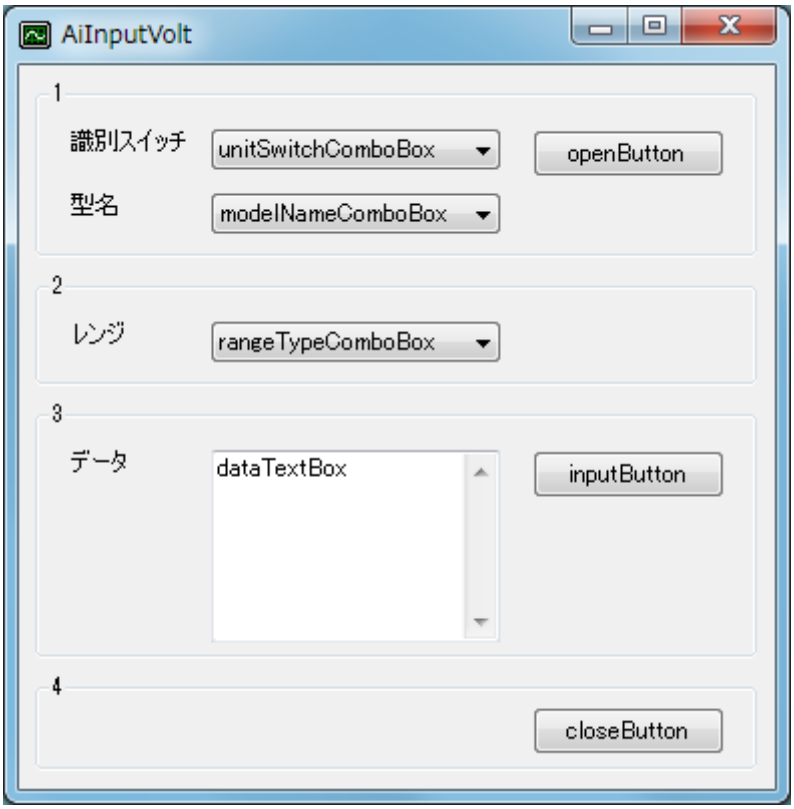
```
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    result = YdxClose(id)
    If result <> 0 And result <> YDX_RESULT_NOT_OPEN Then
        Call ResultShow("YdxClose", result)
    End If
End Sub
```

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。

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_AI_EXCEED_DATA_NUM:
    case YDX_RESULT_AI_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^> { "AIO-64/4/1B-USC", "AIO-
60/4/1B-USC" });
    modelNameComboBox->SelectedIndex = 0;

    // レンジ
    rangeTypeComboBox->ResetText();
    rangeTypeComboBox->Items->AddRange(gcnew array<String^> { "-10 ～ +10V", "-5 ～ +5V" });
}
```

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;
        rangeTypeComboBox->SelectedIndexChanged -= gcnew EventHandler(this,
&Form1::rangeTypeComboBox_SelectedIndexChanged);
        rangeTypeComboBox->SelectedIndex = 0;
        rangeTypeComboBox->SelectedIndexChanged += gcnew
EventHandler(this,&Form1::rangeTypeComboBox_SelectedIndexChanged);
        ResultShow("オープン", result);
        id = getId;
    }
}
```

レンジ切り替え

```
private: System::Void rangeTypeComboBox_SelectedIndexChanged(System::Object^ sender,
System::EventArgs^ e)
{
    int result = YdxAiSetRange(id, rangeTypeComboBox->SelectedIndex);
```

```
    if (result != 0)
        ResultShow("YdxAiSetRange", result);
}
```

アナログ入力

```
private: System::Void inputButton_Click(System::Object^ sender, System::EventArgs^ e)
{
    const int CHANNEL_NUM = 6;
    float data[CHANNEL_NUM];
    int result = YdxAiInputVolt(id, 0, CHANNEL_NUM, data);
    if (result != 0)
    {
        ResultShow("YdxAiInputVolt", result);
        return;
    }

    String^ txt = "";
    for (int channel = 0; channel < CHANNEL_NUM; channel++)
    {
        txt += "CH" + channel.ToString() + " : " + data[channel].ToString(" 0.000;-0.000")-
>PadLeft(7) + "V" + 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);
}
```

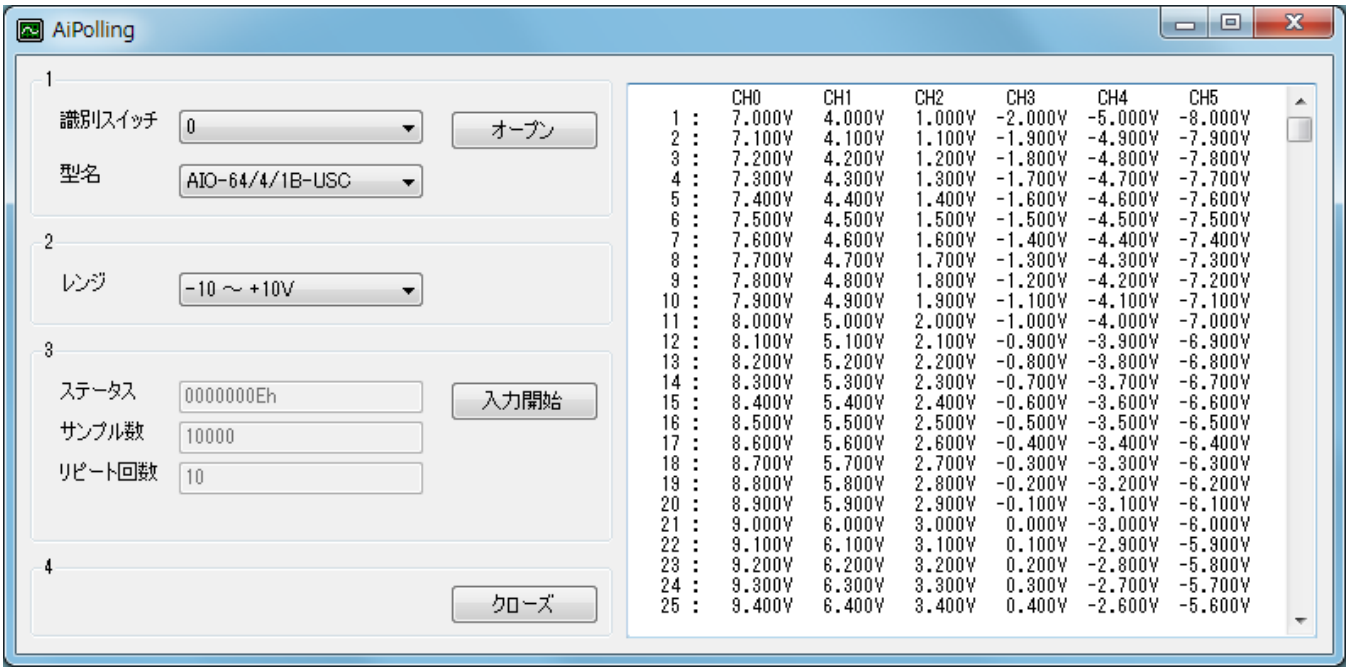
フォームクローズ

```
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);
    }
}
```

AiPolling

高機能アナログ入力 のサンプルプログラムです。
アナログ入力を1000回おこない、電圧値で表示します。
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. 設定

   入力レンジを設定をします。

3. 入力開始

   アナログ入力動作を開始します。

   - 「動作条件の設定」→「動作開始」→「ポーリングによる状態監視」という手順が実行されます。
   - 全てのアナログ入力チャネルを、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。

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

「入力開始」後は、動作が停止するまで、動作状態の読み出しを繰り返しおこない監視（ポーリング）しています。
その為、パソコンに負荷がかかります。

タイマ等を使用して動作状態の読み出し頻度を減らす事で、負荷を軽減できます。
また、ポーリングの代わりにイベントを使用する事で、パソコンにかかる負荷を大幅に軽減できます。

サンプルソース

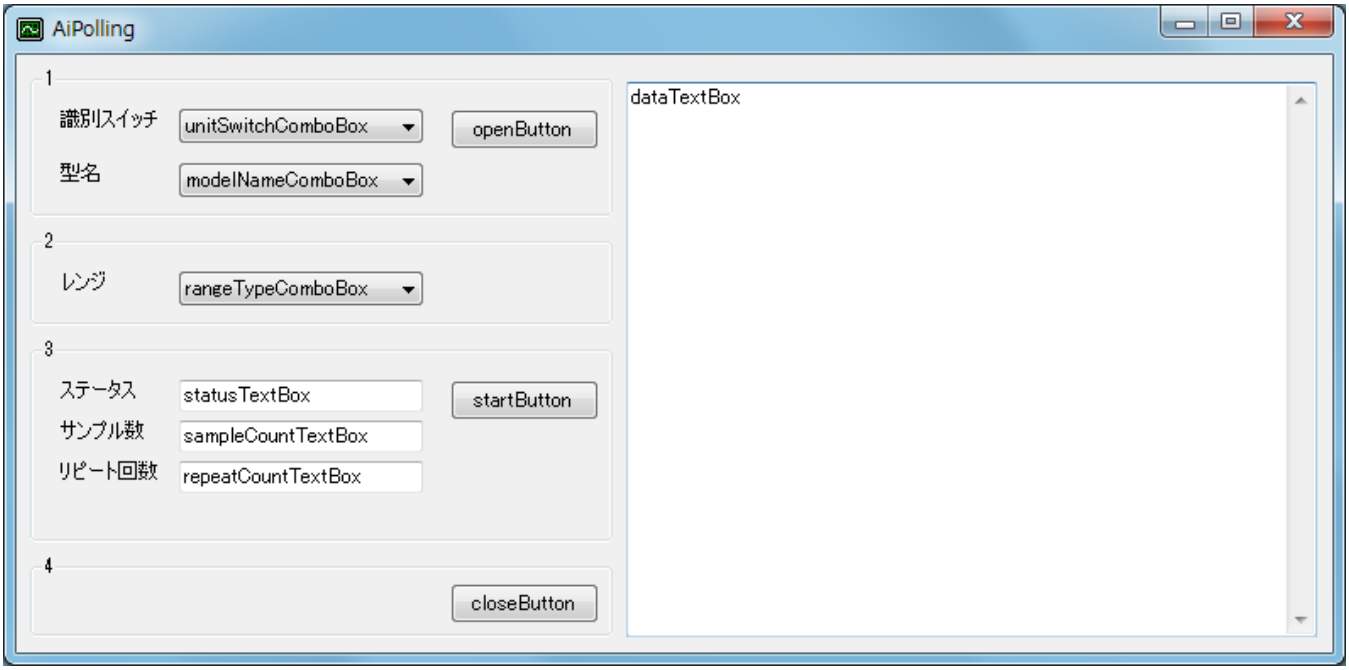
- C#
- Visual Basic（.NET2002以降）
- Visual Basic 6.0
- C++/CLI

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_AI_EXCEED_DATA_NUM:
        case Ydx.YDX_RESULT_AI_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[] { "AIO-64/4/1B-USC", "AIO-60/4/1B-USC" });
    modelNameComboBox.SelectedIndex = 0;

    // レンジ
    rangeTypeComboBox.ResetText();
    rangeTypeComboBox.Items.AddRange(new string[] { "-10 ～ +10V", "-5 ～ +5V" });
}
```

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;
        rangeTypeComboBox.SelectedIndexChanged -= new
EventHandler(rangeTypeComboBox_SelectedIndexChanged);
        rangeTypeComboBox.SelectedIndex = 0;
        rangeTypeComboBox.SelectedIndexChanged += new
EventHandler(rangeTypeComboBox_SelectedIndexChanged);
        ResultShow("オープン", result);
    }
}
```

レンジ切り替え

```
private void rangeTypeComboBox_SelectedIndexChanged(object sender, EventArgs e)
{
    int result = Ydx.AiSetRange(id, rangeTypeComboBox.SelectedIndex);
    if (result != 0)
        ResultShow("YdxAiSetRange", 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.AiSetBuffer(id, 0);   // FIFOバッファ
    if (result != 0)
    {
        ResultShow("YdxAiSetBuffer", result);
        return;
    }

    // チャネルの設定
    const int CHANNEL_NUM = 6;  // 6チャネルを有効にする
    for (int channel = 0; channel < CHANNEL_NUM; channel++)
    {
        result = Ydx.AiSetChannel(id, channel, 1);
        if (result != 0)
        {
            ResultShow("YdxAiSetChannel", result);
            return;
        }
    }

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

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

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

    // サンプリング停止条件の設定
    result = Ydx.AiSetStopCondition(id, 0, 0);    // サンプル数
    if (result != 0)
    {
        ResultShow("YdxAiSetStopCondition", 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 = Ydx.AiSetStopSampleNum(id, SAMPLE_NUM);
    if (result != 0)
    {
        ResultShow("YdxAiSetStopSampleNum", result);
```

```
        return;
    }

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

    // アナログ入力動作を開始
    result = Ydx.AiStart(id);
    if (result != 0)
    {
        ResultShow("YdxAiStart", 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.AiGetStatus(id, out status, out sampleCount, out repeatCount);
        if (result != 0)
        {
            ResultShow("YdxAiGetStatus", 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
    float[] data = new float[sampleCount * CHANNEL_NUM];   // データ個数は、サンプル数 * 有効チャネル数
    int sampleNum = sampleCount;
    result = Ydx.AiGetDataVolt(id, ref sampleNum, data);
    if (result != 0)
    {
        ResultShow("YdxAiGetDataVolt", result);
        if ((result != Ydx.YDX_RESULT_AI_EXCEED_DATA_NUM) && (result != Ydx.YDX_RESULT_AI_EXCEED_BUF_SIZ))
            return;
    }

    // 表示
    string txt = "    ";
    for (int channel = 0; channel < CHANNEL_NUM; channel++)
    {
        txt += "      CH" + channel.ToString();
    }

    txt += Environment.NewLine;
    for (int sampleIndex = 0; sampleIndex < sampleNum; sampleIndex++)
    {
        txt += (sampleIndex + 1).ToString().PadLeft(5) + " : ";
        for (int channel = 0; channel < CHANNEL_NUM; channel++)
        {
            txt += data[sampleIndex * CHANNEL_NUM + channel].ToString("0.000;-0.000").PadLeft(7) + "V ";
        }
        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);
}
```

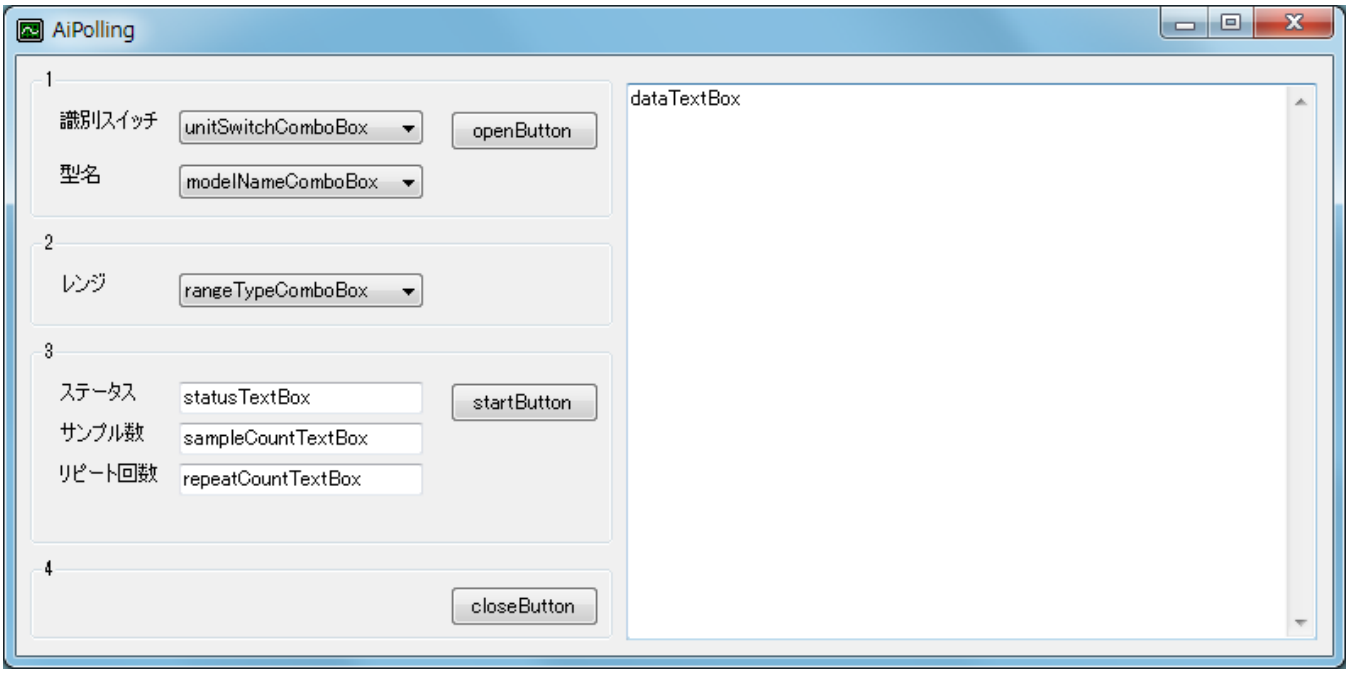
フォームクローズ

```
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);
}
```

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 Short
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() { "AIO-64/4/1B-USC", "AIO-60/4/1B-USC" })
    modelNameComboBox.SelectedIndex = 0

    ' レンジ
    rangeTypeComboBox.ResetText()
    rangeTypeComboBox.Items.AddRange(New String() { "-10 ～ +10V", "-5 ～ +5V" })
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
        RemoveHandler rangeTypeComboBox.SelectedIndexChanged, AddressOf
rangeTypeComboBox_SelectedIndexChange
        rangeTypeComboBox.SelectedIndex = 0
        AddHandler rangeTypeComboBox.SelectedIndexChanged, AddressOf
rangeTypeComboBox_SelectedIndexChange
        ResultShow("オープン", result)
    End If
End Sub
```

レンジ切り替え

```
Private Sub rangeTypeComboBox_SelectedIndexChange(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles rangeTypeComboBox.SelectedIndexChanged
    result = YdxAiSetRange(id, rangeTypeComboBox.SelectedIndex)
    If result <> 0 Then
        ResultShow("YdxAiSetRange", 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 = YdxAiSetBuffer(id, 0)    ' FIFOバッファ
    If result <> 0 Then
```

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

    ' チャネルの設定
    Const CHANNEL_NUM As Integer = 6  ' 6チャネルを有効にする
    For channel As Integer = 0 To CHANNEL_NUM - 1
        result = YdxAiSetChannel(id, channel, 1)
        If result <> 0 Then
            ResultShow("YdxAiSetChannel", result)
            Exit Sub
        End If
    Next

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

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

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

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

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

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

    ' アナログ入力動作を開始
    result = YdxAiStart(id)
    If result <> 0 Then
        ResultShow("YdxAiStart", 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 = YdxAiGetStatus(id, status, sampleCount, repeatCount)
        If result <> 0 Then
            ResultShow("YdxAiGetStatus", 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 * CHANNEL_NUM) As Single    ' データ個数は、サンプル数 * 有効チャネル数
    Dim sampleNum As Integer = sampleCount
    result = YdxAiGetDataVolt(id, sampleNum, data)
    If result <> 0 Then
        ResultShow("YdxAiGetDataVolt", result)
        If result <> YDX_RESULT_AI_EXCEED_DATA_NUM And result <>
YDX_RESULT_AI_EXCEED_BUF_SIZ Then
            Exit Sub
        End If
    End If

    ' 表示
    Dim txt As String = "     "
    For channel As Integer = 0 To CHANNEL_NUM - 1
        txt += "       CH" + channel.ToString()
    Next

    txt += Environment.NewLine
    For sampleIndex As Integer = 0 To sampleNum - 1
        txt += (sampleIndex + 1).ToString().PadLeft(5) + " : "
        For channel As Integer = 0 To CHANNEL_NUM - 1
            txt += data(sampleIndex * CHANNEL_NUM + channel).ToString("
0.000;-0.000").PadLeft(7) + "V "
```

```
        Next
        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

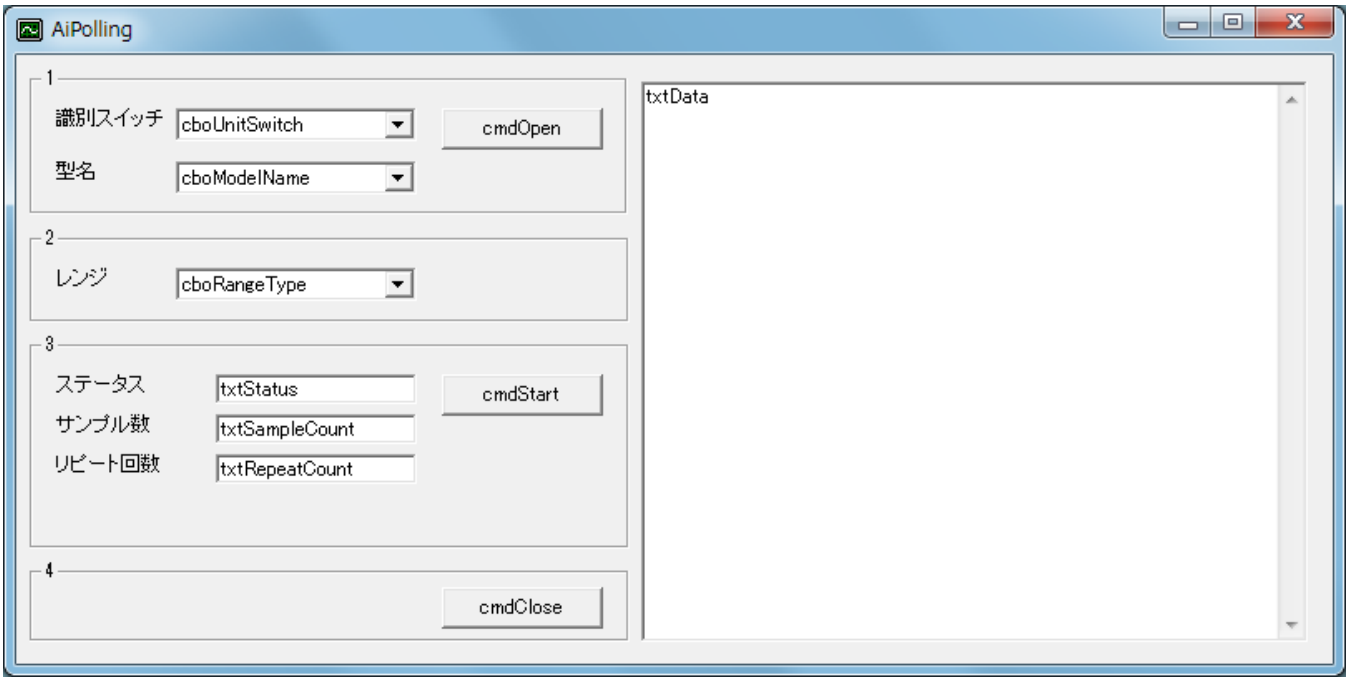
```
Private Sub Form1_Closing(ByVal sender As Object, ByVal e As
System.ComponentModel.CancelEventArgs) Handles MyBase.Closing
    result = YdxClose(id)
    If result <> 0 And result <> YDX_RESULT_NOT_OPEN Then
        ResultShow("YdxClose", result)
    End If
End Sub
```

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_AI_EXCEED_DATA_NUM, Ydx.YDX_RESULT_AI_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 "AIO-64/4/1B-USC"
    cboModelName.AddItem "AIO-60/4/1B-USC"
    cboModelName.ListIndex = 0

    ' レンジ
    cboRangeType.AddItem "-10 ～ +10V"
    cboRangeType.AddItem "-5 ～ +5V"
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
        cboRangeType.ListIndex = 0
        Call ResultShow("オープン", result)
    End If
End Sub
```

レンジ切り替え

```
Private Sub cboRangeType_Click()
    result = YdxAiSetRange(id, cboRangeType.ListIndex)
    If result <> 0 Then
        Call ResultShow("YdxAiSetRange", 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 = YdxAiSetBuffer(id, 0)    ' FIFOバッファ
    If result <> 0 Then
        Call ResultShow("YdxAiSetBuffer", result)
        Exit Sub
    End If

    ' チャネルの設定
    Const CHANNEL_NUM As Long = 6  ' 6チャネルを有効にする
    Dim channel As Long
    For channel = 0 To CHANNEL_NUM - 1
        result = YdxAiSetChannel(id, channel, 1)
        If result <> 0 Then
            Call ResultShow("YdxAiSetChannel", result)
            Exit Sub
        End If
    Next

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

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

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

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

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

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

    ' アナログ入力動作を開始
    result = YdxAiStart(id)
    If result <> 0 Then
        Call ResultShow("YdxAiStart", 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 = YdxAiGetStatus(id, status, sampleCount, repeatCount)
        If result <> 0 Then
            Call ResultShow("YdxAiGetStatus", 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(SAMPLE_NUM * CHANNEL_NUM) As Single    ' データ個数は、サンプル数 * 有効チャネル数
    Dim sampleNum As Long
    sampleNum = SAMPLE_NUM
    result = YdxAiGetDataVolt(id, sampleNum, data(0))
    If result <> 0 Then
        Call ResultShow("YdxAiGetDataVolt", result)
        If result <> YDX_RESULT_AI_EXCEED_DATA_NUM And result <>
YDX_RESULT_AI_EXCEED_BUF_SIZ Then
            Exit Sub
        End If
    End If

    ' 表示
    Dim txt As String
    txt = "     "
    For channel = 0 To CHANNEL_NUM - 1
        txt = txt + "       CH" + Format(channel)
    Next

    txt = txt + vbCrLf
    Dim sampleIndex As Long
    For sampleIndex = 0 To sampleNum - 1
        txt = txt + Right("     " & Str(sampleIndex + 1), 5) + " : "
```

```
        For channel = 0 To CHANNEL_NUM - 1
            txt = txt + Right(" " & Format(data(sampleIndex * CHANNEL_NUM + channel), "
0.000;-0.000"), 7) + "V "
        Next
        txt = txt + 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
```

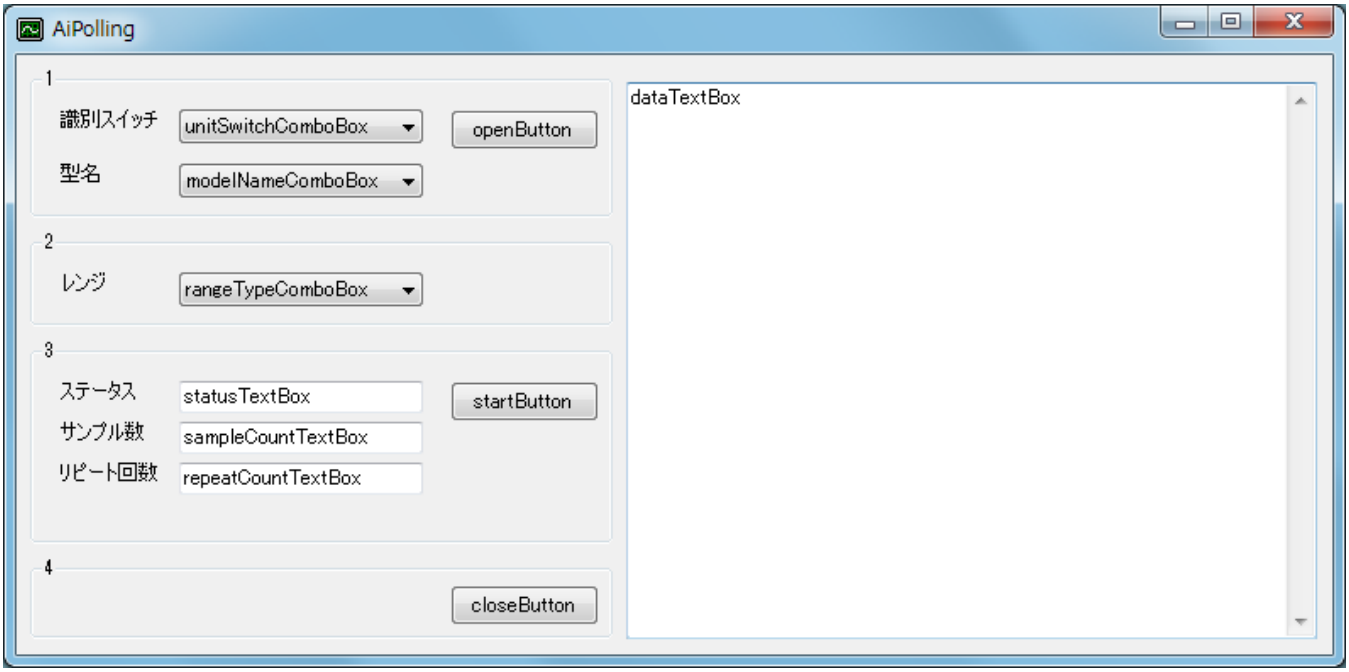
C++/CLI

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_AI_EXCEED_DATA_NUM:
    case YDX_RESULT_AI_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^> { "AIO-64/4/1B-USC", "AIO-
60/4/1B-USC" });
    modelNameComboBox->SelectedIndex = 0;

    // レンジ
    rangeTypeComboBox->ResetText();
    rangeTypeComboBox->Items->AddRange(gcnew array<String^> { "-10 ～ +10V", "-5 ～ +5V" });
}
```

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;
        rangeTypeComboBox->SelectedIndexChanged -= gcnew EventHandler(this,
&Form1::rangeTypeComboBox_SelectedIndexChanged);
        rangeTypeComboBox->SelectedIndex = 0;
        rangeTypeComboBox->SelectedIndexChanged += gcnew
EventHandler(this,&Form1::rangeTypeComboBox_SelectedIndexChanged);
        ResultShow("オープン", result);
        id = getId;
    }
}
```

レンジ切り替え

```
private: System::Void rangeTypeComboBox_SelectedIndexChanged(System::Object^ sender,
System::EventArgs^ e)
{
    int result = YdxAiSetRange(id, rangeTypeComboBox->SelectedIndex);
    if (result != 0)
```

```
        ResultShow("YdxAiSetRange", result);
}
```

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

    // チャネルの設定
    const int CHANNEL_NUM = 6;  // 6チャネルを有効にする
    for (int channel = 0; channel < CHANNEL_NUM; channel++)
    {
        result = YdxAiSetChannel(id, channel, 1);
        if (result != 0)
        {
            ResultShow("YdxAiSetChannel", result);
            return;
        }
    }

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

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

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

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

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

    // アナログ入力動作を開始
    result = YdxAiStart(id);
    if (result != 0)
    {
        ResultShow("YdxAiStart", 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 = YdxAiGetStatus(id, &status, &sampleCount, &repeatCount);
        if (result != 0)
        {
            ResultShow("YdxAiGetStatus", 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
    float data[SAMPLE_NUM * CHANNEL_NUM];   // データ個数は、サンプル数 * 有効チャネル数
    int sampleNum = SAMPLE_NUM;
    result = YdxAiGetDataVolt(id, &sampleNum, data);
    if (result != 0)
    {
        ResultShow("YdxAiGetDataVolt", result);
        if ((result != YDX_RESULT_AI_EXCEED_DATA_NUM) && (result !=
YDX_RESULT_AI_EXCEED_BUF_SIZ))
            return;
    }

    // 表示
    String^ txt = "    ";
    for (int channel = 0; channel < CHANNEL_NUM; channel++)
    {
        txt += "      CH" + channel.ToString();
    }

    txt += Environment::NewLine;
    for (int sampleIndex = 0; sampleIndex < sampleNum; sampleIndex++)
    {
        txt += (sampleIndex + 1).ToString()->PadLeft(5) + " : ";
        for (int channel = 0; channel < CHANNEL_NUM; channel++)
        {
            txt += data[sampleIndex * CHANNEL_NUM + channel].ToString(" 0.000;-0.000")-
>PadLeft(7) + "V ";
        }
        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

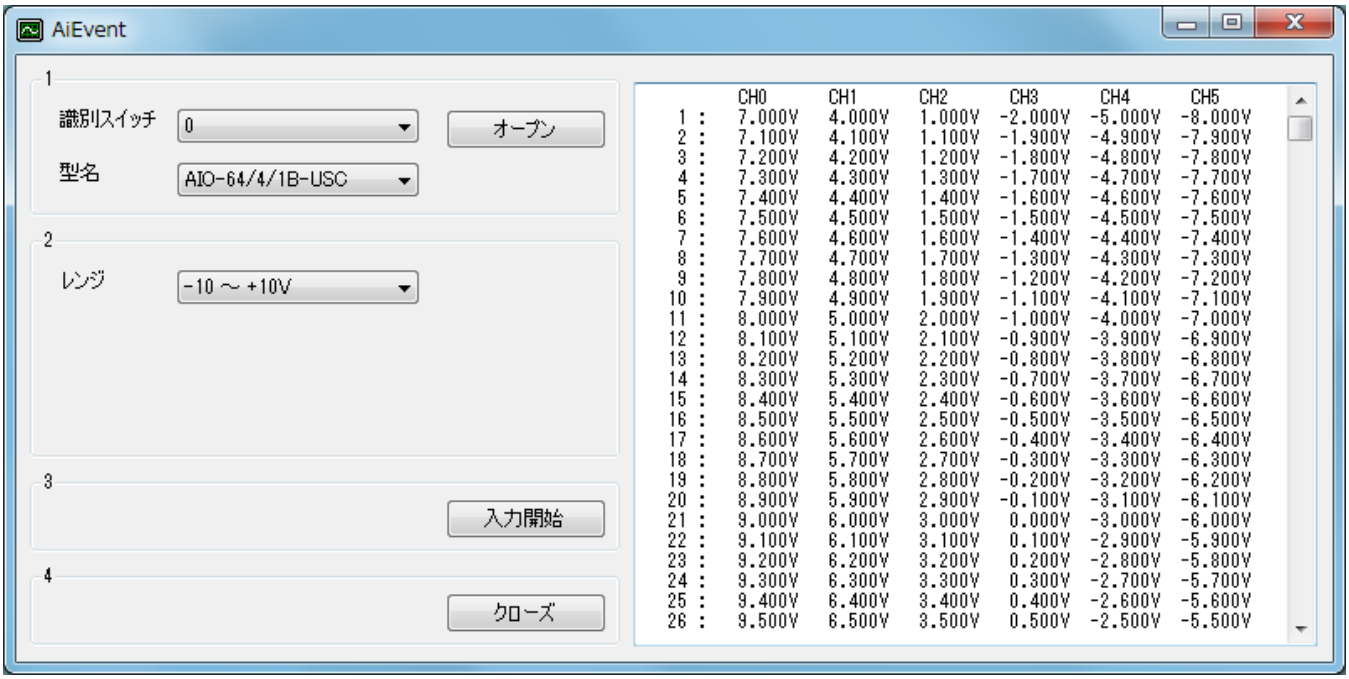
```
private: System::Void Form1_FormClosing(System::Object^ sender,
System::Windows::Forms::FormClosingEventArgs^ e)
{
    int result = YdxClose(id);
    if ((result != 0) && (result != YDX_RESULT_NOT_OPEN))
    {
        ResultShow("YdxClose", result);
```

```
    }
}
```

AiEvent

高機能アナログ入力 のサンプルプログラムです。
アナログ入力を1000回おこない、電圧値で表示します。
動作状態の監視をイベント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. 設定

   入力レンジを設定します。

3. 入力開始

   アナログ入力動作を開始します。
   「動作条件の設定」→「動作開始」→「イベント待ち」という手順が実行されます。

   全てのアナログ入力チャネルを、周期 1msecで1000回サンプリングします。
   動作が停止するとイベントが発生しますので、データを読み出して、表示します。

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

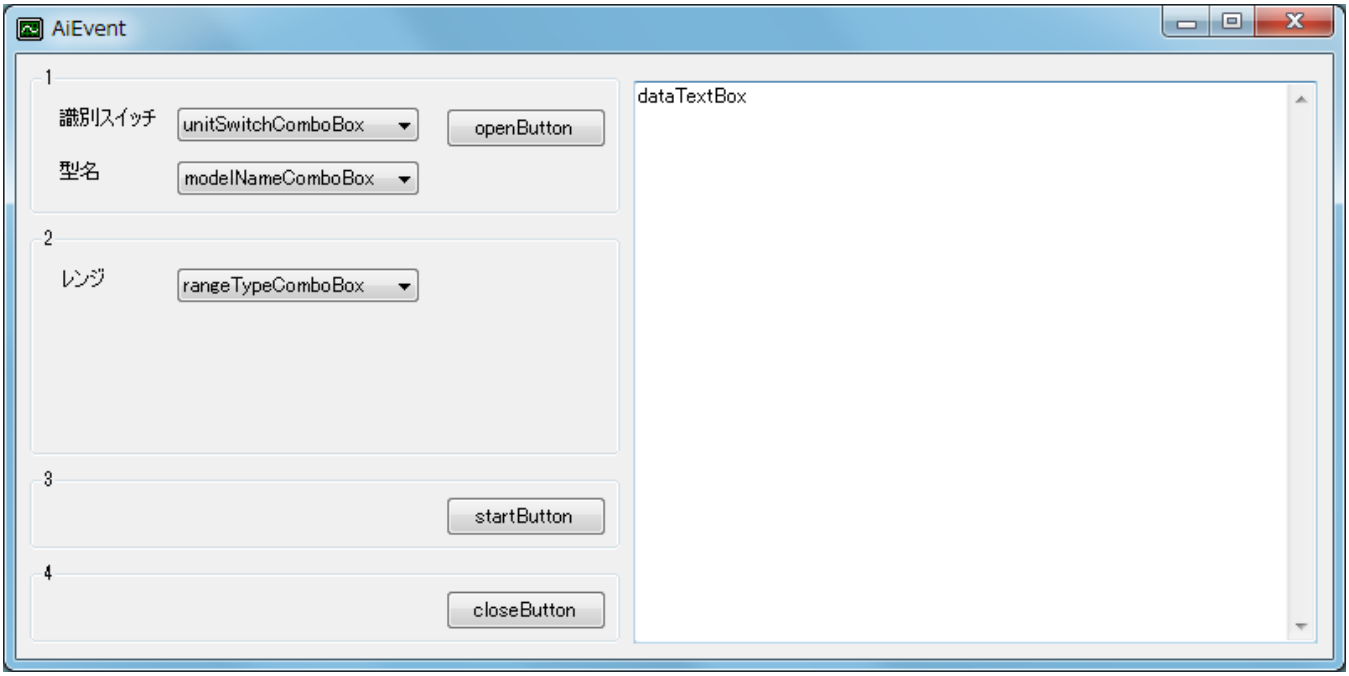
C#

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_AI_EXCEED_DATA_NUM:
        case Ydx.YDX_RESULT_AI_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[] { "AIO-64/4/1B-USC", "AIO-60/4/1B-USC" });
    modelNameComboBox.SelectedIndex = 0;

    // レンジ
    rangeTypeComboBox.ResetText();
    rangeTypeComboBox.Items.AddRange(new string[] { "-10 ～ +10V", "-5 ～ +5V" });
}
```

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;
        rangeTypeComboBox.SelectedIndexChanged -= new
EventHandler(rangeTypeComboBox_SelectedIndexChanged);
        rangeTypeComboBox.SelectedIndex = 0;
        rangeTypeComboBox.SelectedIndexChanged += new
EventHandler(rangeTypeComboBox_SelectedIndexChanged);
        ResultShow("オープン", result);
    }
}
```

レンジ切り替え

```
private void rangeTypeComboBox_SelectedIndexChanged(object sender, EventArgs e)
{
    int result = Ydx.AiSetRange(id, rangeTypeComboBox.SelectedIndex);
    if (result != 0)
        ResultShow("YdxAiSetRange", 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.AiSetBuffer(id, 0); // FIFOバッファ
    if (result != 0)
    {
        ResultShow("YdxAiSetBuffer", result);
        return;
    }

    // チャネルの設定
    const int CHANNEL_NUM = 6; // 6チャネルを有効にする
    for (int channel = 0; channel < CHANNEL_NUM; channel++)
    {
        result = Ydx.AiSetChannel(id, channel, 1);
        if (result != 0)
        {
            ResultShow("YdxAiSetChannel", result);
            return;
        }
    }

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

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

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

    // サンプリング停止条件の設定
    result = Ydx.AiSetStopCondition(id, 0, 0); // サンプル数
    if (result != 0)
    {
        ResultShow("YdxAiSetStopCondition", 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 = Ydx.AiSetStopSampleNum(id, SAMPLE_NUM);
    if (result != 0)
    {
        ResultShow("YdxAiSetStopSampleNum", result);
```

```
        return;
    }

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

    // イベントオブジェクト作成
    AutoResetEvent hEvent = new AutoResetEvent(false);

    // イベントの設定
    result = Ydx.AiSetEvent(id,
        Ydx.YDX_EVENT_COMMUNICATE_ERR |
        Ydx.YDX_EVENT_HARDWARE_ERR |
        Ydx.YDX_EVENT_OVERRUN_ERR |
        Ydx.YDX_EVENT_SAMPLE_CLOCK_ERR |
        Ydx.YDX_EVENT_STOP,
        hEvent.Handle);
    if (result != 0)
    {
        ResultShow("YdxAiSetEvent", result);
        hEvent.Close();
        return;
    }

    // アナログ入力動作を開始
    result = Ydx.AiStart(id);
    if (result != 0)
    {
        ResultShow("YdxAiStart", result);
        hEvent.Close();
        return;
    }

    // イベント発生待ち
    hEvent.WaitOne();
    hEvent.Close();

    // ステータスの取得
    int factor, sampleCount, repeatCount;
    result = Ydx.AiGetEventStatus(id, out factor, out sampleCount, out repeatCount);
    if (result != 0)
    {
        ResultShow("YdxAiGetEventStatus", result);
        return;
    }

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

    if ((factor & Ydx.YDX_EVENT_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 ((factor & Ydx.YDX_EVENT_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 ((factor & Ydx.YDX_EVENT_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;
    }

    // データの読み出し
    float[] data = new float[sampleCount * CHANNEL_NUM]; // データ個数は、サンプル数 * 有効チャネ
ル数
    int sampleNum = sampleCount;
    result = Ydx.AiGetDataVolt(id, ref sampleNum, data);
    if (result != 0)
    {
        ResultShow("YdxAiGetDataVolt", result);
        if ((result != Ydx.YDX_RESULT_AI_EXCEED_DATA_NUM) && (result !=
Ydx.YDX_RESULT_AI_EXCEED_BUF_SIZ))
            return;
    }

    // 表示
    string txt = " ";
    for (int channel = 0; channel < CHANNEL_NUM; channel++)
    {
        txt += " CH" + channel.ToString();
    }

    txt += Environment.NewLine;
    for (int sampleIndex = 0; sampleIndex < sampleNum; sampleIndex++)
    {
        txt += (sampleIndex + 1).ToString().PadLeft(5) + " : ";
        for (int channel = 0; channel < CHANNEL_NUM; channel++)
        {
            txt += data[sampleIndex * CHANNEL_NUM + channel].ToString("
0.000;-0.000").PadLeft(7) + "V ";
        }
        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

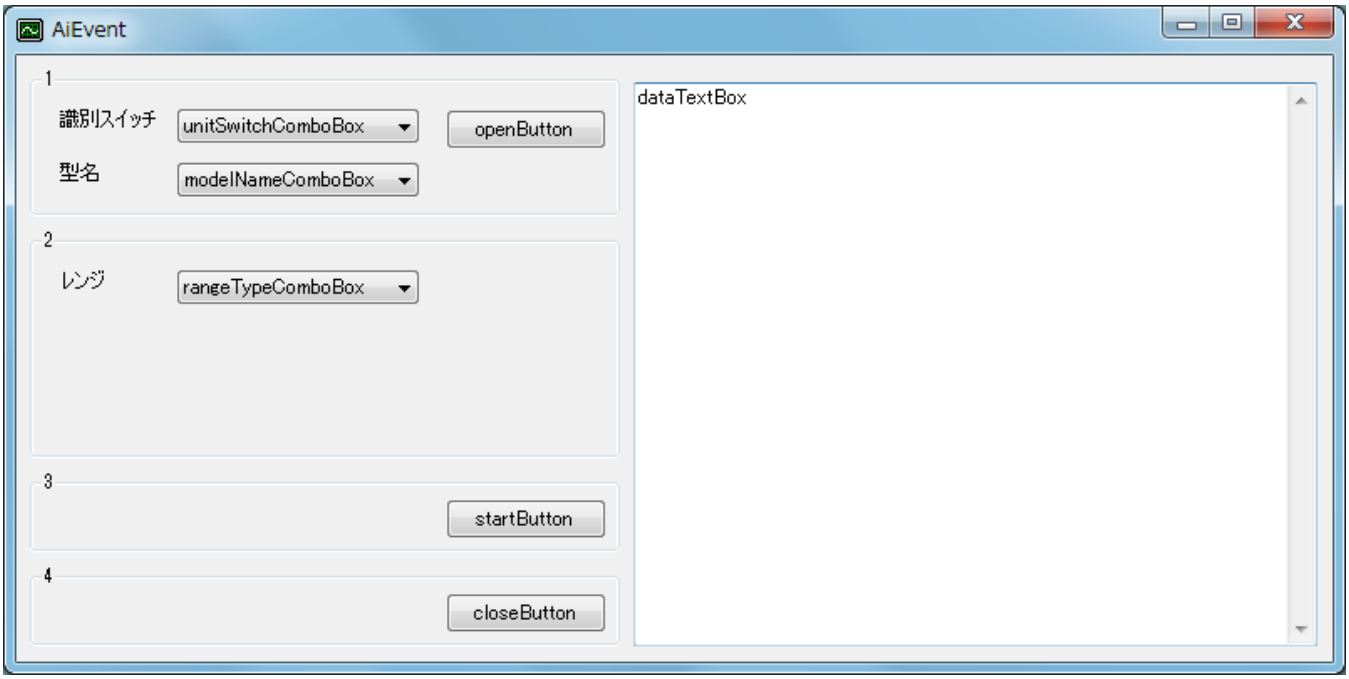
```
private void Form1_Closing(object sender, System.ComponentModel.CancelEventArgs e)
{
    int result = Ydx.Close(id);
    if ((result != 0) && (result != Ydx.YDX_RESULT_NOT_OPEN))
    {
        ResultShow("YdxClose", result);
    }
}
```

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 Short
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() { "AIO-64/4/1B-USC", "AIO-60/4/1B-USC" })
    modelNameComboBox.SelectedIndex = 0

    ' レンジ
    rangeTypeComboBox.ResetText()
    rangeTypeComboBox.Items.AddRange(New String() { "-10 ～ +10V", "-5 ～ +5V" })
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
        RemoveHandler rangeTypeComboBox.SelectedIndexChanged, AddressOf
rangeTypeComboBox_SelectedIndexChange
        rangeTypeComboBox.SelectedIndex = 0
        AddHandler rangeTypeComboBox.SelectedIndexChanged, AddressOf
rangeTypeComboBox_SelectedIndexChange
        ResultShow("オープン", result)
    End If
End Sub
```

レンジ切り替え

```
Private Sub rangeTypeComboBox_SelectedIndexChange(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles rangeTypeComboBox.SelectedIndexChanged
    result = YdxAiSetRange(id, rangeTypeComboBox.SelectedIndex)
    If result <> 0 Then
        ResultShow("YdxAiSetRange", 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 = YdxAiSetBuffer(id, 0)    ' FIFOバッファ
    If result <> 0 Then
```

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

    ' チャネルの設定
    Const CHANNEL_NUM As Integer = 6  ' 6チャネルを有効にする
    For channel As Integer = 0 To CHANNEL_NUM - 1
        result = YdxAiSetChannel(id, channel, 1)
        If result <> 0 Then
            ResultShow("YdxAiSetChannel", result)
            Exit Sub
        End If
    Next

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

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

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

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

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

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

    ' イベントオブジェクト作成
    Dim hEvent As AutoResetEvent = New AutoResetEvent(False)

    ' イベントの設定
    result = YdxAiSetEvent(id, _
        YDX_EVENT_COMMUNICATE_ERR Or _
        YDX_EVENT_HARDWARE_ERR Or _
```

```
        YDX_EVENT_SAMPLE_CLOCK_ERR Or _
        YDX_EVENT_OVERRUN_ERR Or _
        YDX_EVENT_STOP, _
        hEvent.Handle)
    If result <> 0 Then
        ResultShow("YdxAiSetEvent", result)
        hEvent.Close()
        Exit Sub
    End If

    ' アナログ入力動作を開始
    result = YdxAiStart(id)
    If result <> 0 Then
        ResultShow("YdxAiStart", result)
        hEvent.Close()
        Exit Sub
    End If

    ' イベント発生待ち
    hEvent.WaitOne()
    hEvent.Close()

    ' ステータスの取得
    Dim factor, sampleCount, repeatCount As Integer
    result = YdxAiGetEventStatus(id, factor, sampleCount, repeatCount)
    If result <> 0 Then
        ResultShow("YdxAiGetEventStatus", result)
        Exit Sub
    End If

    If (factor 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 (factor 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 (factor 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

    If (factor 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

    ' データの読み出し
    Dim data(sampleCount * CHANNEL_NUM) As Single    ' データ個数は、サンプル数 * 有効チャネル数
    Dim sampleNum As Integer = sampleCount
    result = YdxAiGetDataVolt(id, sampleNum, data)
    If result <> 0 Then
        ResultShow("YdxAiGetDataVolt", result)
        If result <> YDX_RESULT_AI_EXCEED_DATA_NUM And result <>
YDX_RESULT_AI_EXCEED_BUF_SIZ Then
            Exit Sub
        End If
```

```
    End If

    Dim txt As String = "    "
    For channel As Integer = 0 To CHANNEL_NUM - 1
        txt += "      CH" + channel.ToString()
    Next

    txt += Environment.NewLine
    For sampleIndex As Integer = 0 To sampleNum - 1
        txt += (sampleIndex + 1).ToString().PadLeft(5) + " : "
        For channel As Integer = 0 To CHANNEL_NUM - 1
            txt += data(sampleIndex * CHANNEL_NUM + channel).ToString("
0.000;-0.000").PadLeft(7) + "V "
        Next
        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

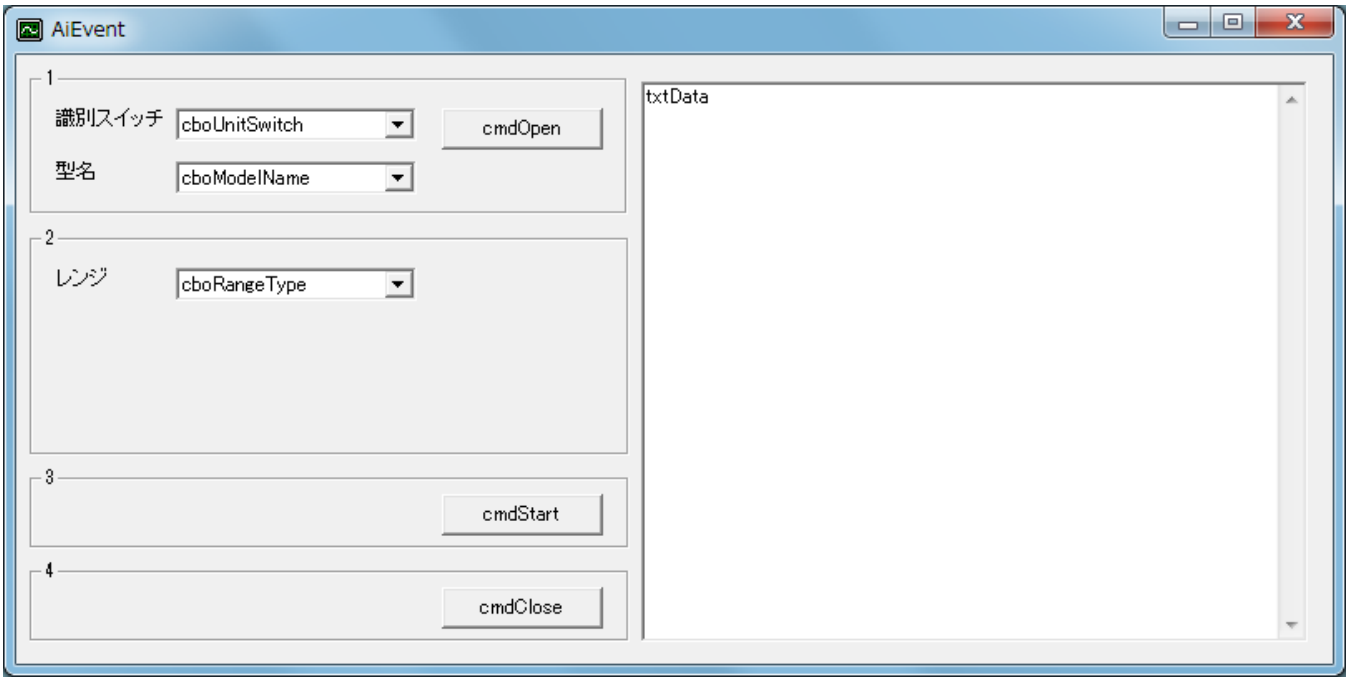
```
Private Sub Form1_Closing(ByVal sender As Object, ByVal e As
System.ComponentModel.CancelEventArgs) Handles MyBase.Closing
    result = YdxClose(id)
    If result <> 0 And result <> YDX_RESULT_NOT_OPEN Then
        ResultShow("YdxClose", result)
    End If
End Sub
```

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



定義

```
Private Declare Function CloseHandle Lib "kernel32" (ByVal hObject As Long) As Long
Private Declare Function CreateEvent Lib "kernel32" Alias "CreateEventA" (ByVal
lpEventAttributes As Long, ByVal bManualReset As Long, ByVal bInitialState As Long, ByVal
lpName As String) As Long
Private Declare Function WaitForSingleObject Lib "kernel32" (ByVal hHandle As Long, ByVal
dwMilliseconds As Long) As Long
Private Const INFINITE = &HFFFF
```

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_AI_EXCEED_DATA_NUM, Ydx.YDX_RESULT_AI_EXCEED_BUF_SIZ
```

```
            MsgBox resultString, vbInformation, title
        Case Else
            MsgBox resultString, vbCritical, title
    End Select
```

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 "AIO-64/4/1B-USC"
    cboModelName.AddItem "AIO-60/4/1B-USC"
    cboModelName.ListIndex = 0

    ' レンジ
    cboRangeType.AddItem "-10 ～ +10V"
    cboRangeType.AddItem "-5 ～ +5V"
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
        cboRangeType.ListIndex = 0
        Call ResultShow("オープン", result)
    End If
End Sub
```

レンジ切り替え

```
Private Sub cboRangeType_Click()
    result = YdxAiSetRange(id, cboRangeType.ListIndex)
    If result <> 0 Then
        Call ResultShow("YdxAiSetRange", 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 = YdxAiSetBuffer(id, 0)    ' FIFOバッファ
    If result <> 0 Then
        Call ResultShow("YdxAiSetBuffer", result)
        Exit Sub
    End If

    ' チャネルの設定
    Const CHANNEL_NUM As Long = 6   ' 6チャネルを有効にする
    Dim channel As Long
    For channel = 0 To CHANNEL_NUM - 1
        result = YdxAiSetChannel(id, channel, 1)
        If result <> 0 Then
            Call ResultShow("YdxAiSetChannel", result)
            Exit Sub
        End If
    Next

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

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

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

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

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

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

' イベントオブジェクト作成
Dim hEvent As Long
hEvent = CreateEvent(0, True, False, 0)
If hEvent = 0 Then
    MsgBox "イベントオブジェクトの作成に失敗しました", vbCritical
    Exit Sub
End If

' イベントの設定
result = YdxAiSetEvent(id, _
    YDX_EVENT_COMMUNICATE_ERR Or _
    YDX_EVENT_HARDWARE_ERR Or _
    YDX_EVENT_OVERRUN_ERR Or _
    YDX_EVENT_SAMPLE_CLOCK_ERR Or _
    YDX_EVENT_STOP, _
    hEvent)
If result <> 0 Then
    Call ResultShow("YdxAiSetEvent", result)
    CloseHandle hEvent
    Exit Sub
End If

' アナログ入力動作を開始
result = YdxAiStart(id)
If result <> 0 Then
    Call ResultShow("YdxAiStart", result)
    CloseHandle hEvent
    Exit Sub
End If

' イベント発生待ち
Call WaitForSingleObject(hEvent, INFINITE)
CloseHandle hEvent

' ステータスの取得
Dim factor, sampleCount, repeatCount As Long
result = YdxAiGetEventStatus(id, factor, sampleCount, repeatCount)
If result <> 0 Then
    Call ResultShow("YdxAiGetEventStatus", result)
    Exit Sub
End If

If (factor 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 (factor 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 (factor 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 (factor 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

    ' データの読み出し
    Dim data(SAMPLE_NUM * CHANNEL_NUM) As Single    ' データ個数は、サンプル数 * 有効チャネル数
    Dim sampleNum As Long
    sampleNum = SAMPLE_NUM
    result = YdxAiGetDataVolt(id, sampleNum, data(0))
    If result <> 0 Then
        Call ResultShow("YdxAiGetDataVolt", result)
        If result <> YDX_RESULT_AI_EXCEED_DATA_NUM And result <>
YDX_RESULT_AI_EXCEED_BUF_SIZ Then
            Exit Sub
        End If
    End If

    ' 表示
    Dim txt As String
    txt = "     "
    For channel = 0 To CHANNEL_NUM - 1
        txt = txt + "       CH" + Format(channel)
    Next

    txt = txt + vbCrLf
    Dim sampleIndex As Long
    For sampleIndex = 0 To sampleNum - 1
        txt = txt + Right("    " + Str(sampleIndex + 1), 5) + " : "
        For channel = 0 To CHANNEL_NUM - 1
            txt = txt + Right(" " & Format(data(sampleIndex * CHANNEL_NUM + channel), "
0.000;-0.000"), 7) + "V "
        Next
        txt = txt + 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
```

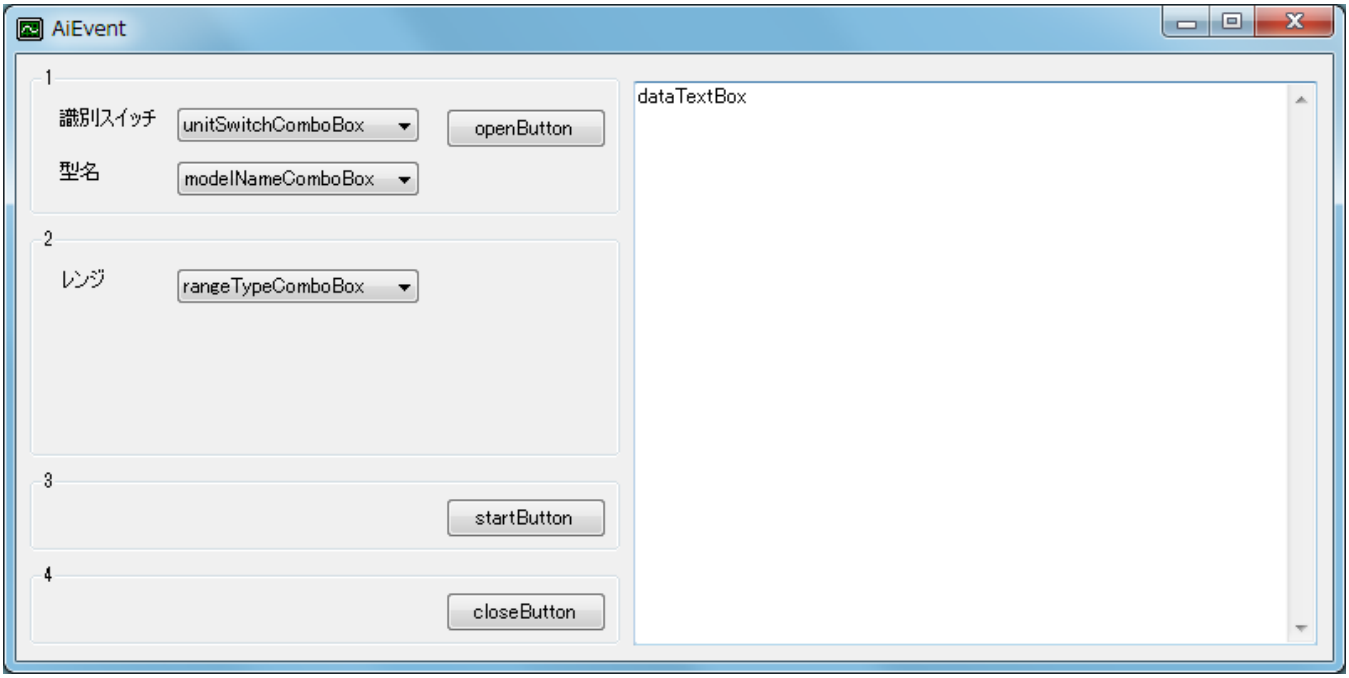
C++/CLI

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_AI_EXCEED_DATA_NUM:
    case YDX_RESULT_AI_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^> { "AIO-64/4/1B-USC", "AIO-
60/4/1B-USC" });
    modelNameComboBox->SelectedIndex = 0;

    // レンジ
    rangeTypeComboBox->ResetText();
    rangeTypeComboBox->Items->AddRange(gcnew array<String^> { "-10 ～ +10V", "-5 ～ +5V" });
}
```

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;
        rangeTypeComboBox->SelectedIndexChanged -= gcnew EventHandler(this,
&Form1::rangeTypeComboBox_SelectedIndexChanged);
        rangeTypeComboBox->SelectedIndex = 0;
        rangeTypeComboBox->SelectedIndexChanged += gcnew
EventHandler(this,&Form1::rangeTypeComboBox_SelectedIndexChanged);
        ResultShow("オープン", result);
        id = getId;
    }
}
```

レンジ切り替え

```
private: System::Void rangeTypeComboBox_SelectedIndexChanged(System::Object^ sender,
System::EventArgs^ e)
{
    int result = YdxAiSetRange(id, rangeTypeComboBox->SelectedIndex);
    if (result != 0)
```

```
        ResultShow("YdxAiSetRange", result);
}
```

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

    // チャネルの設定
    const int CHANNEL_NUM = 6;  // 6チャネルを有効にする
    for (int channel = 0; channel < CHANNEL_NUM; channel++)
    {
        result = YdxAiSetChannel(id, channel, 1);
        if (result != 0)
        {
            ResultShow("YdxAiSetChannel", result);
            return;
        }
    }

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

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

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

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

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

    // イベントオブジェクト作成
    AutoResetEvent^ eventHandle = gcnew AutoResetEvent(false);

    // イベントの設定
    result = YdxAiSetEvent(id,
        YDX_EVENT_COMMUNICATE_ERR |
        YDX_EVENT_HARDWARE_ERR |
        YDX_EVENT_SAMPLE_CLOCK_ERR |
        YDX_EVENT_OVERRUN_ERR |
        YDX_EVENT_STOP,
        eventHandle->Handle);
    if (result != 0)
    {
        ResultShow("YdxAiSetEvent", result);
        eventHandle->Close();
        return;
    }

    // アナログ入力動作を開始
    result = YdxAiStart(id);
    if (result != 0)
    {
        ResultShow("YdxAiStart", result);
        eventHandle->Close();
        return;
    }

    // イベント発生待ち
    eventHandle->WaitOne();
    eventHandle->Close();

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

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

```
    if ((factor & YDX_EVENT_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 ((factor & YDX_EVENT_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 ((factor & YDX_EVENT_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;
    }

    // データの読み出し
    float data[SAMPLE_NUM * CHANNEL_NUM];   // データ個数は、サンプル数 * 有効チャネル数
    int sampleNum = SAMPLE_NUM;
    result = YdxAiGetDataVolt(id, &sampleNum, data);
    if (result != 0)
    {
        ResultShow("YdxAiGetDataVolt", result);
        if ((result != YDX_RESULT_AI_EXCEED_DATA_NUM) && (result !=
YDX_RESULT_AI_EXCEED_BUF_SIZ))
            return;
    }

    // 表示
    String^ txt = "    ";
    for (int channel = 0; channel < CHANNEL_NUM; channel++)
    {
        txt += "      CH" + channel.ToString();
    }

    txt += Environment::NewLine;
    for (int sampleIndex = 0; sampleIndex < sampleNum; sampleIndex++)
    {
        txt += (sampleIndex + 1).ToString()->PadLeft(5) + " : ";
        for (int channel = 0; channel < CHANNEL_NUM; channel++)
        {
            txt += data[sampleIndex * CHANNEL_NUM + channel].ToString(" 0.000;-0.000")-
>PadLeft(7) + "V ";
        }
        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);
}
```

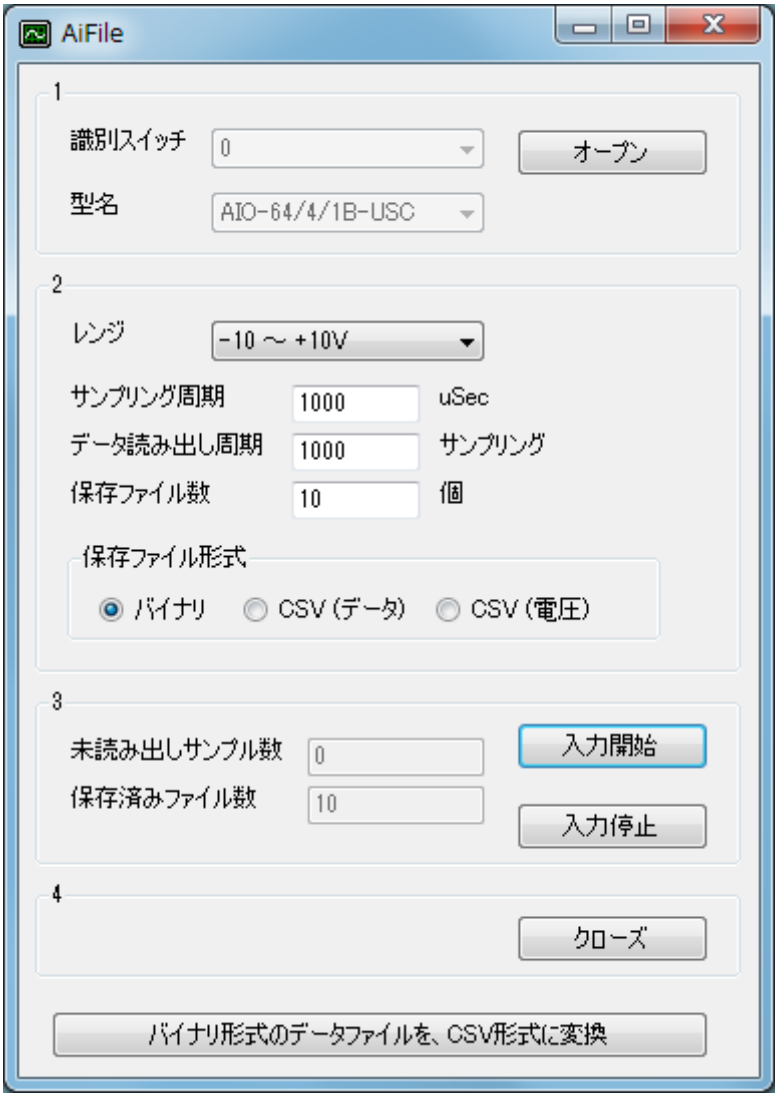
フォームクローズ

```
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);
    }
}
```

AiFile

高機能アナログ入力 のサンプルプログラムです。
アナログ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「バイナリ」が最も高速で、「CSV（データ）」「CSV（電圧）」の順に遅くなっていきます。
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

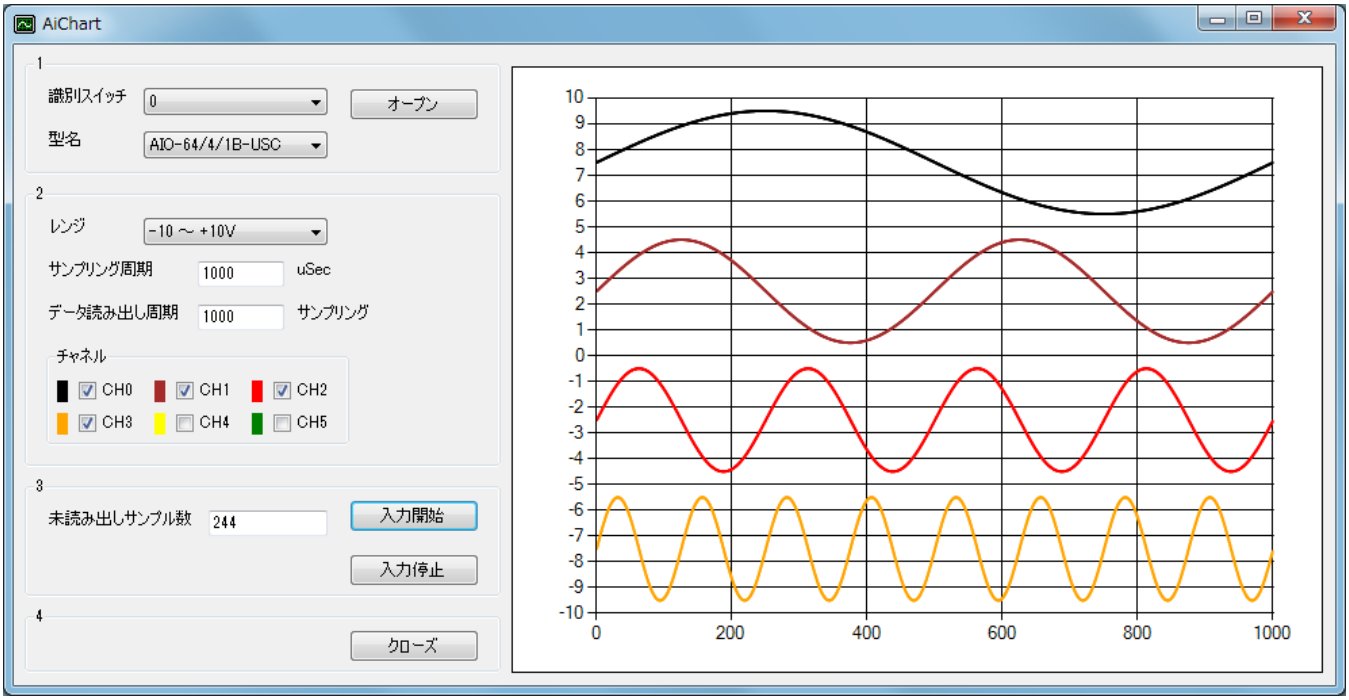
- C#

  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。

AiChart

高機能アナログ入力 のサンプルプログラムです。
アナログ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。

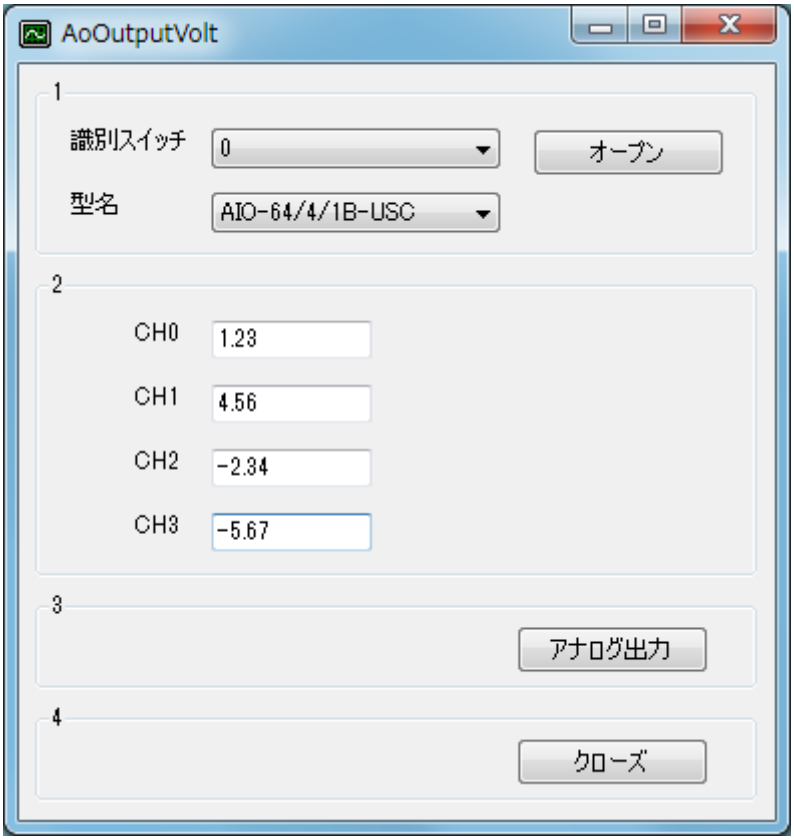
開発環境について

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。

AoOutputVolt

簡易アナログ出力 のサンプルプログラムです。
アナログ出力を1回おこないます。

画面



1. オープン

   ユニットのオープンをします。

2. 設定

   出力したい電圧を設定します。

3. アナログ出力

   アナログ出力を1回おこないます。

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 .NET
- Visual Basic 6.0
- C++/CLI

サンプルプログラム > アナログ出力 > AoOutputVolt >

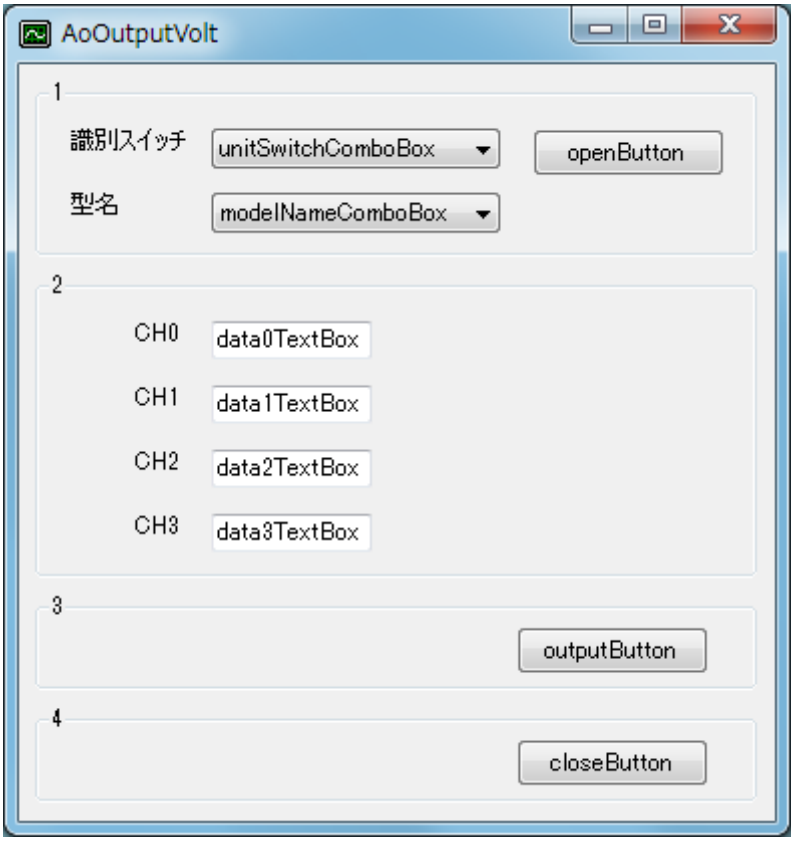
C#

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_AI_EXCEED_DATA_NUM:
```

```
        case Ydx.YDX_RESULT_AI_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[] { "AIO-64/4/1B-USC", "AIO-04/4/1B-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);
    }
}
```

アナログ出力

```
private void outputButton_Click(object sender, EventArgs e)
{
    const int CHANNEL_NUM = 4;
    float[] data = new float[CHANNEL_NUM];
    string dataText = "";
    for (int i = 0; i < CHANNEL_NUM; i++)
    {
        switch (i)
        {
            case 0:
                dataText = data0TextBox.Text;
                break;
            case 1:
                dataText = data1TextBox.Text;
```

```
                break;
            case 2:
                dataText = data2TextBox.Text;
                break;
            case 3:
                dataText = data3TextBox.Text;
                break;
        }

        double doubleData;
        if (!double.TryParse(dataText, System.Globalization.NumberStyles.Float, null, out
doubleData))
        {
            MessageBox.Show("CH" + i.ToString() + "のデータが不正です", "",
MessageBoxButtons.OK, MessageBoxIcon.Hand);
            return;
        }

        data[i] = (float)doubleData;
    }

    int result = Ydx.AoOutputVolt(id, 0, CHANNEL_NUM, 0, data);
    ResultShow("アナログ出力", 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

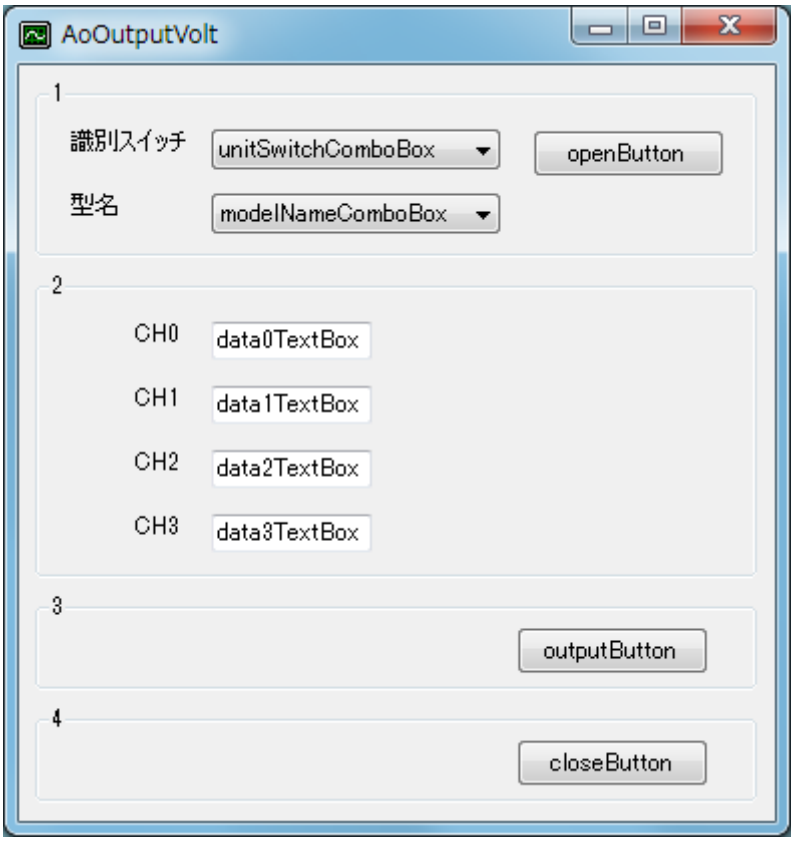
```
private void Form1_Closing(object sender, System.ComponentModel.CancelEventArgs e)
{
    int result = Ydx.Close(id);
    if ((result != 0) && (result != Ydx.YDX_RESULT_NOT_OPEN))
    {
        ResultShow("YdxClose", result);
    }
}
```

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 Short
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() { "AIO-64/4/1B-USC", "AIO-04/4/1B-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
```

アナログ出力

```
Private Sub outputButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles outputButton.Click
    Const CHANNEL_NUM As Integer = 4
    Dim data(CHANNEL_NUM) As Single
    Dim dataText As String = ""
    For i As Integer = 0 To CHANNEL_NUM - 1
        Select Case (i)
            Case 0
                dataText = data0TextBox.Text
            Case 1
                dataText = data1TextBox.Text
            Case 2
                dataText = data2TextBox.Text
            Case 3
                dataText = data3TextBox.Text
        End Select

        Dim doubleData As Double
        If Double.TryParse(dataText, Globalization.NumberStyles.Float, Nothing, doubleData)
= False Then
            MessageBox.Show("CH" + i.ToString() + "のデータが不正です", "",
MessageBoxButtons.OK, MessageBoxIcon.Hand)
            Exit Sub
        End If
```

```
        data(i) = CType(doubleData, Single)
    Next

    result = YdxAoOutputVolt(id, 0, CHANNEL_NUM, 0, data)
    ResultShow("アナログ出力", 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

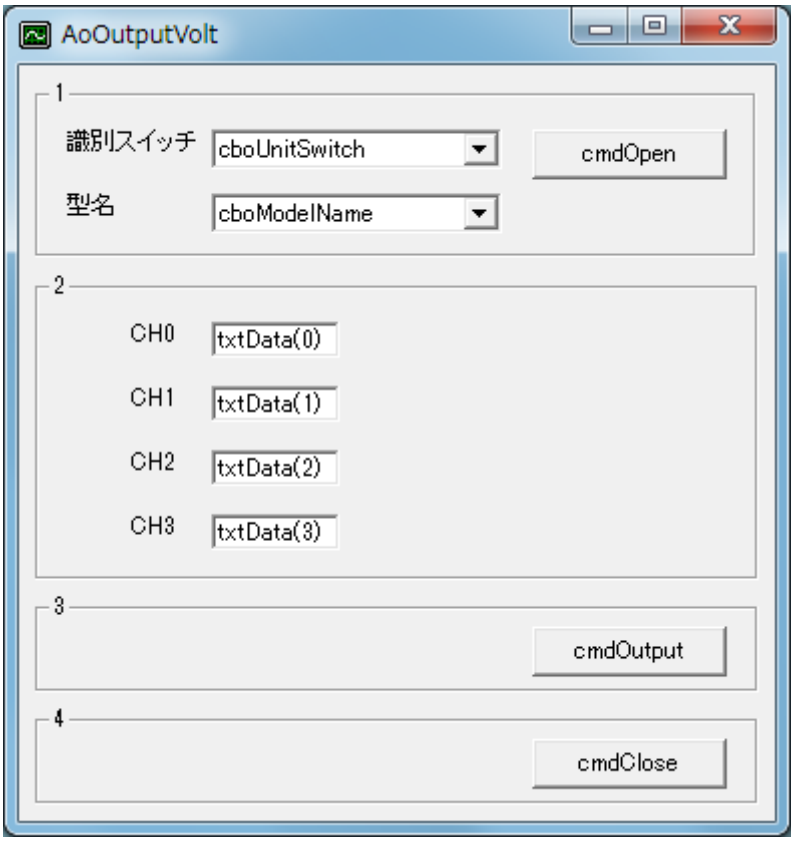
```
Private Sub Form1_Closing(ByVal sender As Object, ByVal e As
System.ComponentModel.CancelEventArgs) Handles MyBase.Closing
    result = YdxClose(id)
    If result <> 0 And result <> YDX_RESULT_NOT_OPEN Then
        ResultShow("YdxClose", result)
    End If
End Sub
```

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_AI_EXCEED_DATA_NUM, Ydx.YDX_RESULT_AI_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 "AIO-64/4/1B-USC"
    cboModelName.AddItem "AIO-04/4/1B-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
```

アナログ出力

```
Private Sub cmdOutput_Click()
    Const CHANNEL_NUM As Long = 4
    Dim data(CHANNEL_NUM) As Single
    Dim i As Long
    For i = 0 To CHANNEL_NUM - 1
        data(i) = txtData(i).Text
    Next

    result = YdxAoOutputVolt(id, 0, CHANNEL_NUM, 0, data(0))
    Call ResultShow("アナログ出力", 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

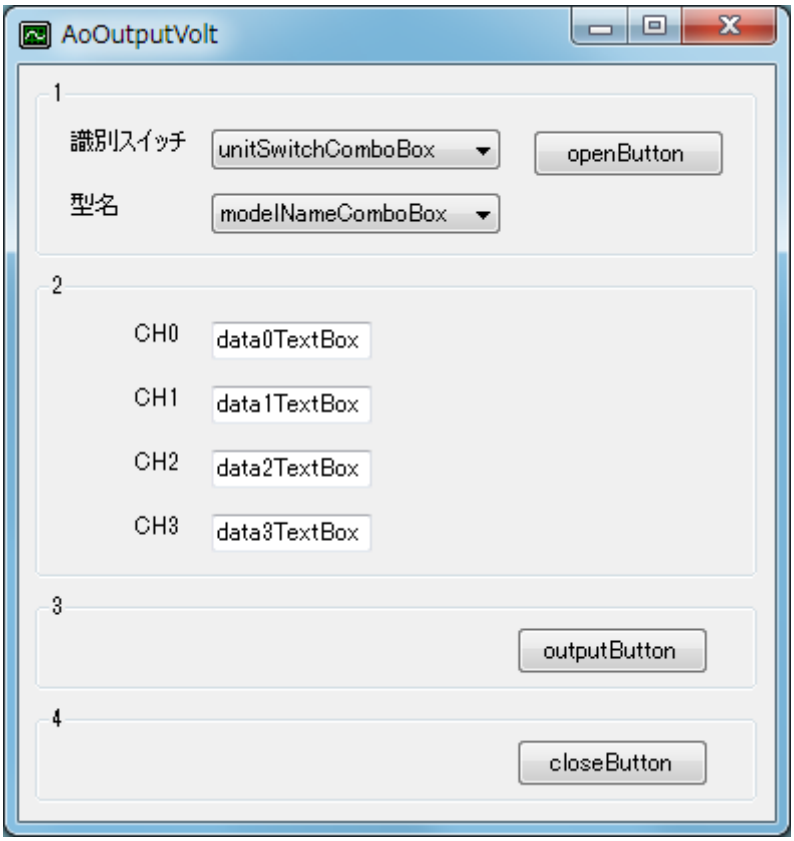
```
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    result = YdxClose(id)
    If result <> 0 And result <> YDX_RESULT_NOT_OPEN Then
        Call ResultShow("YdxClose", result)
    End If
End Sub
```

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_AI_EXCEED_DATA_NUM:
    case YDX_RESULT_AI_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^> { "AIO-64/4/1B-USC", "AIO-
04/4/1B-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;
    }
}
```

アナログ出力

```
private: System::Void outputButton_Click(System::Object^ sender, System::EventArgs^ e)
{
    const int CHANNEL_NUM = 4;
    float data[CHANNEL_NUM];
    String^ dataText = "";
    for (int i = 0; i < CHANNEL_NUM; i++)
    {
        switch (i)
        {
        case 0:
            dataText = data0TextBox->Text;
            break;
```

```
        case 1:
            dataText = data1TextBox->Text;
            break;
        case 2:
            dataText = data2TextBox->Text;
            break;
        case 3:
            dataText = data3TextBox->Text;
            break;
        }

        if (!float::TryParse(dataText, data[i]))
        {
            MessageBox::Show("CH" + i.ToString() + "のデータが不正です", "",
MessageBoxButtons::OK, MessageBoxIcon::Hand);
            return;
        }
    }

    int result = YdxAoOutputVolt(id, 0, CHANNEL_NUM, 0, data);
    ResultShow("アナログ出力", 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);
    }
}
```

AoPolling

高機能アナログ出力 のサンプルプログラムです。
アナログ出力を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」→「ポーリングによる状態監視」という手順が実行されます。
   全てのアナログ出力チャネルに、サンプリング周期 1msec・1000サンプリングの正弦波を出力します。
   出力波形は、1Hzの正弦波1周期・振幅は±10Vです。

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。

備考

「出力開始」後は、動作が停止するまで、動作状態の読み出しを繰り返しおこない監視（ポーリング）しています。
その為、パソコンに負荷がかかります。
タイマ等を使用して動作状態の読み出し頻度を減らす事で、負荷を軽減できます。
また、ポーリングの代わりにイベントを使用して、パソコンにかかる負荷を大幅に軽減できます。

サンプルソース

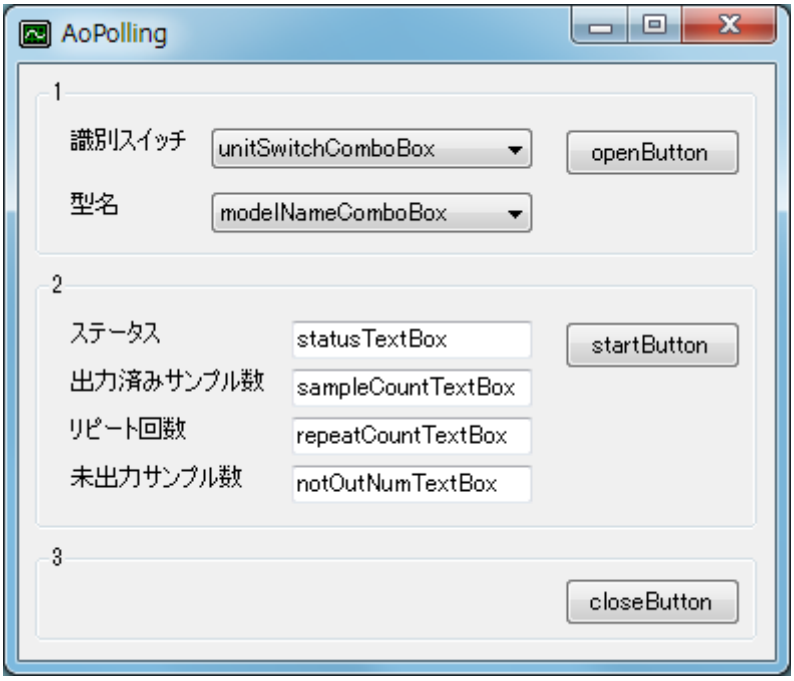
- C#
- Visual Basic（.NET2002以降）
- Visual Basic 6.0
- C++/CLI

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_AI_EXCEED_DATA_NUM:
        case Ydx.YDX_RESULT_AI_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[] { "AIO-64/4/1B-USC", "AIO-04/4/1B-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);
    }
}
```

アナログ出力

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

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

    // チャネルの設定
    const int CHANNEL_NUM = 4;  // 4チャネルを有効にする
    for (int channel = 0; channel < CHANNEL_NUM; channel++)
    {
        result = Ydx.AoSetChannel(id, channel, 1);
        if (result != 0)
        {
```

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

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

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

    // データの設定
    const int SAMPLE_NUM = 1000;  // サンプル数
    float[] data = new float[SAMPLE_NUM * CHANNEL_NUM];   // データ個数は、サンプル数 * 有効チャネル数
    for (int i = 0; i < SAMPLE_NUM; i++)
    {
        float waveData = (float)(10 * Math.Sin(2 * Math.PI * i / SAMPLE_NUM));    // 正弦波
        for (int channel = 0; channel < CHANNEL_NUM; channel++)
        {
            data[i * CHANNEL_NUM + channel] = waveData;
        }
    }

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

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

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

    // アナログ出力動作を開始
    result = Ydx.AoStart(id);
    if (result != 0)
    {
        ResultShow("YdxAoStart", 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.AoGetStatus(id, out status, out sampleCount, out repeatCount, out
notOutNum);
        if (result != 0)
        {
            ResultShow("YdxAoGetStatus", 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("アナログ出力", 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

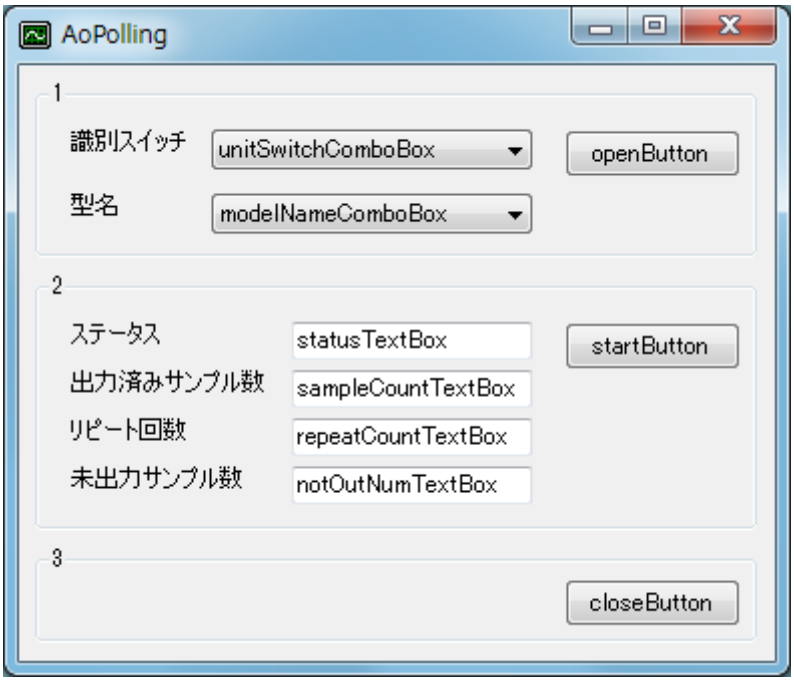
```
private void Form1_Closing(object sender, System.ComponentModel.CancelEventArgs e)
{
    int result = Ydx.Close(id);
    if ((result != 0) && (result != Ydx.YDX_RESULT_NOT_OPEN))
    {
        ResultShow("YdxClose", result);
    }
}
```

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 Short
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() { "AIO-64/4/1B-USC", "AIO-04/4/1B-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 = YdxAoSetBuffer(id, 0) ' FIFOバッファ
    If result <> 0 Then
        ResultShow("YdxAoSetBuffer", result)
        Exit Sub
    End If

    ' チャネルの設定
    Const CHANNEL_NUM As Integer = 4  ' 4チャネルを有効にする
    For channel As Integer = 0 To CHANNEL_NUM - 1
        result = YdxAoSetChannel(id, channel, 1)
        If result <> 0 Then
            ResultShow("YdxAoSetChannel", result)
            Exit Sub
        End If
    Next

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

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

```
        ResultShow("YdxAoSetClockInternal", 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 * CHANNEL_NUM) As Single   ' データ個数は、サンプル数 * 有効チャネル数
    For i As Integer = 0 To SAMPLE_NUM - 1
        Dim waveData As Single = CType(10 * Math.Sin(2 * Math.PI * i / SAMPLE_NUM), Single)
' 正弦波
        For channel As Integer = 0 To CHANNEL_NUM - 1
            data(i * CHANNEL_NUM + channel) = waveData
        Next
    Next

    result = YdxAoSetDataVolt(id, SAMPLE_NUM, data)
    If result <> 0 Then
        ResultShow("YdxAoSetDataVolt", result)
        Exit Sub
    End If

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

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

    ' アナログ出力動作を開始
    result = YdxAoStart(id)
    If result <> 0 Then
        ResultShow("YdxAoStart", 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 = YdxAoGetStatus(id, status, sampleCount, repeatCount, notOutNum)
        If result <> 0 Then
            ResultShow("YdxAoGetStatus", 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("アナログ出力", 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

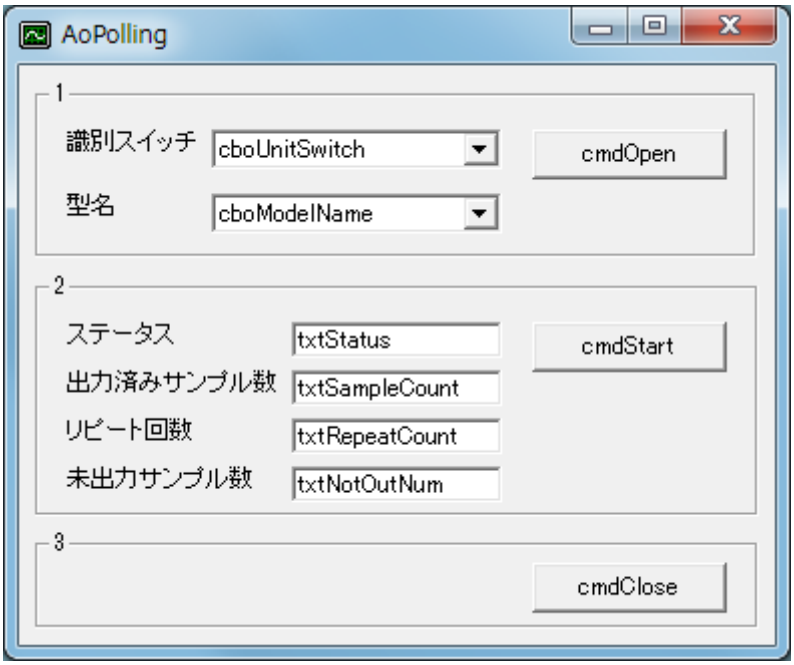
```
Private Sub Form1_Closing(ByVal sender As Object, ByVal e As
System.ComponentModel.CancelEventArgs) Handles MyBase.Closing
    result = YdxClose(id)
    If result <> 0 And result <> YDX_RESULT_NOT_OPEN Then
        ResultShow("YdxClose", result)
    End If
End Sub
```

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_AI_EXCEED_DATA_NUM, Ydx.YDX_RESULT_AI_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 "AIO-64/4/1B-USC"
    cboModelName.AddItem "AIO-04/4/1B-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()
    Dim result As Long

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

    ' チャネルの設定
    Const CHANNEL_NUM As Long = 4   ' 4チャネルを有効にする
    Dim channel As Long
    For channel = 0 To CHANNEL_NUM - 1
        result = YdxAoSetChannel(id, channel, 1)
        If result <> 0 Then
            Call ResultShow("YdxAoSetChannel", result)
            Exit Sub
        End If
    Next

    ' サンプリングクロックの設定
```

```
    result = YdxAoSetClock(id, 0)     ' 内部クロック
    If result <> 0 Then
        Call ResultShow("YdxAoSetClock", result)
        Exit Sub
    End If

    ' 内部クロック周期の設定
    result = YdxAoSetClockInternal(id, 1000) ' 1000μsec
    If result <> 0 Then
        Call ResultShow("YdxAoSetClockInternal", 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 * CHANNEL_NUM) As Single    ' データ個数は、サンプル数 * 有効チャネル数
    Dim i As Long
    For i = 0 To SAMPLE_NUM - 1
        Dim waveData As Single
        waveData = 10 * Math.Sin(2 * 3.14159265358979 * i / SAMPLE_NUM)    ' 正弦波
        For channel = 0 To CHANNEL_NUM - 1
            data(i * CHANNEL_NUM + channel) = waveData
        Next
    Next

    result = YdxAoSetDataVolt(id, SAMPLE_NUM, data(0))
    If result <> 0 Then
        Call ResultShow("YdxAoSetDataVolt", result)
        Exit Sub
    End If

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

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

    ' アナログ出力動作を開始
    result = YdxAoStart(id)
    If result <> 0 Then
        Call ResultShow("YdxAoStart", 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 = YdxAoGetStatus(id, status, sampleCount, repeatCount, notOutNum)
        If result <> 0 Then
            Call ResultShow("YdxAoGetStatus", 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("アナログ出力", 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
```

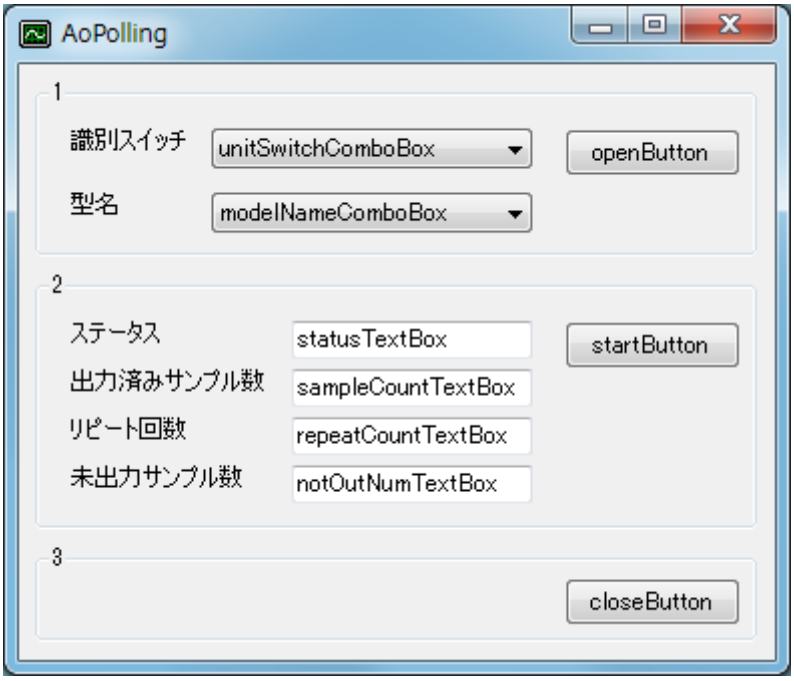
C++/CLI

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_AI_EXCEED_DATA_NUM:
    case YDX_RESULT_AI_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^> { "AIO-64/4/1B-USC", "AIO-
04/4/1B-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 = YdxAoSetBuffer(id, 0);   // FIFOバッファ
    if (result != 0)
    {
        ResultShow("YdxAoSetBuffer", result);
        return;
    }

    // チャネルの設定
    const int CHANNEL_NUM = 4;  // 4チャネルを有効にする
```

```
    for (int channel = 0; channel < CHANNEL_NUM; channel++)
    {
        result = YdxAoSetChannel(id, channel, 1);
        if (result != 0)
        {
            ResultShow("YdxAoSetChannel", result);
            return;
        }
    }

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

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

    // データの設定
    const int SAMPLE_NUM = 1000;  // サンプル数
    float data[SAMPLE_NUM * CHANNEL_NUM];   // データ個数は、サンプル数 * 有効チャネル数
    for (int i = 0; i < SAMPLE_NUM; i++)
    {
        float waveData = (float)(10 * Math::Sin(2 * Math.PI * i / SAMPLE_NUM));    // 正弦波
        for (int channel = 0; channel < CHANNEL_NUM; channel++)
        {
            data[i * CHANNEL_NUM + channel] = waveData;
        }
    }
    result = YdxAoSetDataVolt(id, SAMPLE_NUM, data);
    if (result != 0)
    {
        ResultShow("YdxAoSetDataVolt", result);
        return;
    }

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

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

    // アナログ出力動作を開始
    result = YdxAoStart(id);
    if (result != 0)
    {
```

```
        ResultShow("YdxAoStart", 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 = YdxAoGetStatus(id, &status, &sampleCount, &repeatCount, &notOutNum);
        if (result != 0)
        {
            ResultShow("YdxAoGetStatus", 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("アナログ出力", 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);
}
```

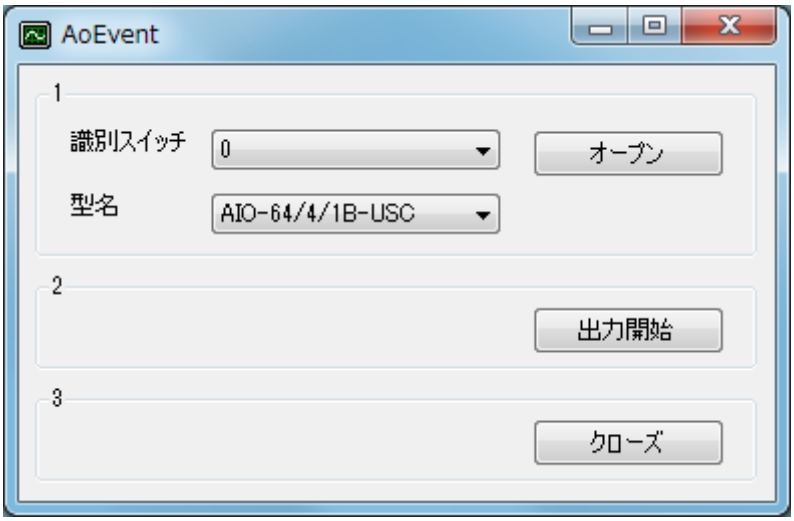
フォームクローズ

```
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);
    }
}
```

AoEvent

高機能アナログ出力 のサンプルプログラムです。
アナログ出力を1000回おこないます。
動作状態の監視をイベント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」→「イベント待ち」という手順が実行されます。
   全てのアナログ出力チャネルに、サンプリング周期 1msec・1000サンプリングの正弦波を出力します。
   出力波形は、1Hzの正弦波1周期・振幅は±10Vです。

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

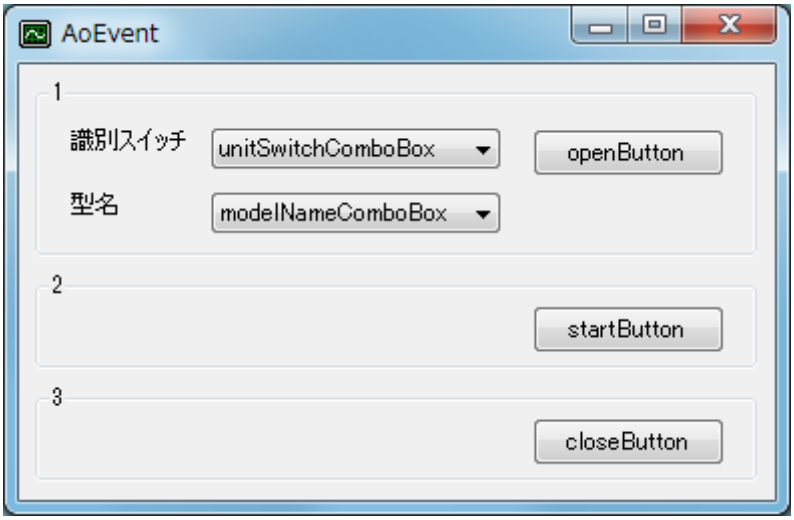
- C#
- Visual Basic（.NET2002以降）
- C++/CLI
- Visual Basic 6.0

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_AI_EXCEED_DATA_NUM:
        case Ydx.YDX_RESULT_AI_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[] { "AIO-64/4/1B-USC", "AIO-04/4/1B-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.AoSetBuffer(id, 0);   // FIFOバッファ
    if (result != 0)
    {
        ResultShow("YdxAoSetBuffer", result);
        return;
    }

    // チャネルの設定
    const int CHANNEL_NUM = 4;  // 4チャネルを有効にする
    for (int channel = 0; channel < CHANNEL_NUM; channel++)
    {
        result = Ydx.AoSetChannel(id, channel, 1);
        if (result != 0)
        {
            ResultShow("YdxAoSetChannel", result);
            return;
        }
    }

    // サンプリングクロックの設定
```

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

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

    // データの設定
    const int SAMPLE_NUM = 1000;  // サンプル数
    float[] data = new float[SAMPLE_NUM * CHANNEL_NUM];    // データ個数は、サンプル数 * 有効チャネル数
    for (int i = 0; i < SAMPLE_NUM; i++)
    {
        float waveData = (float)(10 * Math.Sin(2 * Math.PI * i / SAMPLE_NUM));     // 正弦波
        for (int channel = 0; channel < CHANNEL_NUM; channel++)
        {
            data[i * CHANNEL_NUM + channel] = waveData;
        }
    }
    result = Ydx.AoSetDataVolt(id, SAMPLE_NUM, data);
    if (result != 0)
    {
        ResultShow("YdxAoSetDataVolt", result);
        return;
    }

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

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

    // イベントオブジェクト作成
    AutoResetEvent hEvent = new AutoResetEvent(false);

    // イベントの設定
    result = Ydx.AoSetEvent(id,
        Ydx.YDX_EVENT_COMMUNICATE_ERR |
        Ydx.YDX_EVENT_HARDWARE_ERR |
        Ydx.YDX_EVENT_SAMPLE_CLOCK_ERR |
        Ydx.YDX_EVENT_STOP,
        hEvent.Handle);
    if (result != 0)
    {
        ResultShow("YdxAoSetEvent", result);
        hEvent.Close();
```

```
        return;
    }

    // アナログ出力動作を開始
    result = Ydx.AoStart(id);
    if (result != 0)
    {
        ResultShow("YdxAoStart", result);
        hEvent.Close();
        return;
    }

    // イベント発生待ち
    hEvent.WaitOne();
    hEvent.Close();

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

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

    if ((factor & Ydx.YDX_EVENT_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 ((factor & Ydx.YDX_EVENT_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;
    }

    ResultShow("アナログ出力", 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);
}
```

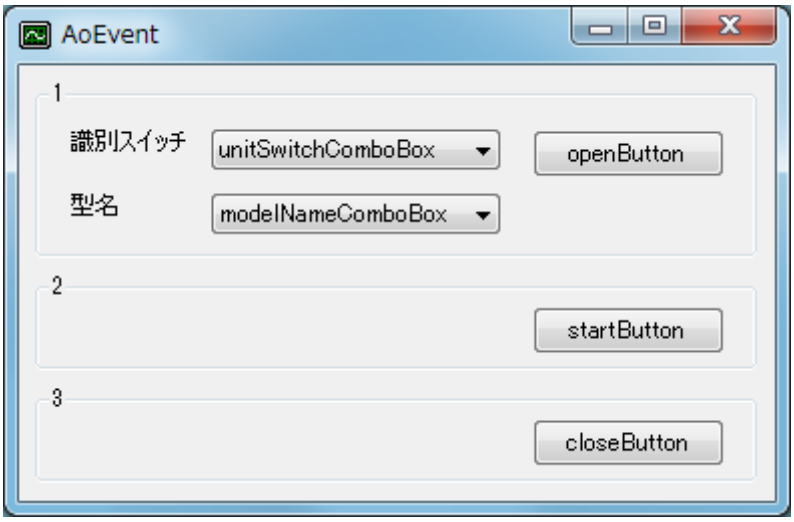
フォームクローズ

```
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);
    }
}
```

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 Short
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() { "AIO-64/4/1B-USC", "AIO-04/4/1B-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 = YdxAoSetBuffer(id, 0)    ' FIFOバッファ
    If result <> 0 Then
        ResultShow("YdxAoSetBuffer", result)
        Exit Sub
    End If

    ' チャネルの設定
    Const CHANNEL_NUM As Integer = 4  ' 4チャネルを有効にする
    For channel As Integer = 0 To CHANNEL_NUM - 1
        result = YdxAoSetChannel(id, channel, 1)
        If result <> 0 Then
            ResultShow("YdxAoSetChannel", result)
            Exit Sub
        End If
    Next

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

    ' 内部クロック周期の設定
    result = YdxAoSetClockInternal(id, 1000) ' 1000μsec
    If result <> 0 Then
        ResultShow("YdxAoSetClockInternal", 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 * CHANNEL_NUM) As Single    ' データ個数は、サンプル数 * 有効チャネル数
    For i As Integer = 0 To SAMPLE_NUM - 1
        Dim waveData As Single = CType(10 * Math.Sin(2 * Math.PI * i / SAMPLE_NUM), Single)
' 正弦波
        For channel As Integer = 0 To CHANNEL_NUM - 1
            data(i * CHANNEL_NUM + channel) = waveData
        Next
    Next
    result = YdxAoSetDataVolt(id, SAMPLE_NUM, data)
    If result <> 0 Then
        ResultShow("YdxAoSetDataVolt", result)
        Exit Sub
    End If

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

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

    ' イベントオブジェクト作成
    Dim hEvent As AutoResetEvent = New AutoResetEvent(False)

    ' イベントの設定
    result = YdxAoSetEvent(id, _
        YDX_EVENT_COMMUNICATE_ERR Or _
        YDX_EVENT_HARDWARE_ERR Or _
        YDX_EVENT_SAMPLE_CLOCK_ERR Or _
        YDX_EVENT_STOP, _
        hEvent.Handle)
    If result <> 0 Then
        ResultShow("YdxAoSetEvent", result)
        hEvent.Close()
        Exit Sub
    End If

    ' アナログ出力動作を開始
    result = YdxAoStart(id)
    If result <> 0 Then
        ResultShow("YdxAoStart", result)
        hEvent.Close()
        Exit Sub
    End If

    ' イベント発生待ち
    hEvent.WaitOne()
    hEvent.Close()

    ' ステータスの取得
    Dim factor, sampleCount, repeatCount, notOutNum As Integer
    result = YdxAoGetEventStatus(id, factor, sampleCount, repeatCount, notOutNum)
    If result <> 0 Then
        ResultShow("YdxAoGetEventStatus", result)
        Exit Sub
    End If
```

```
    If (factor 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 (factor 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 (factor 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

    ResultShow("アナログ出力", 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

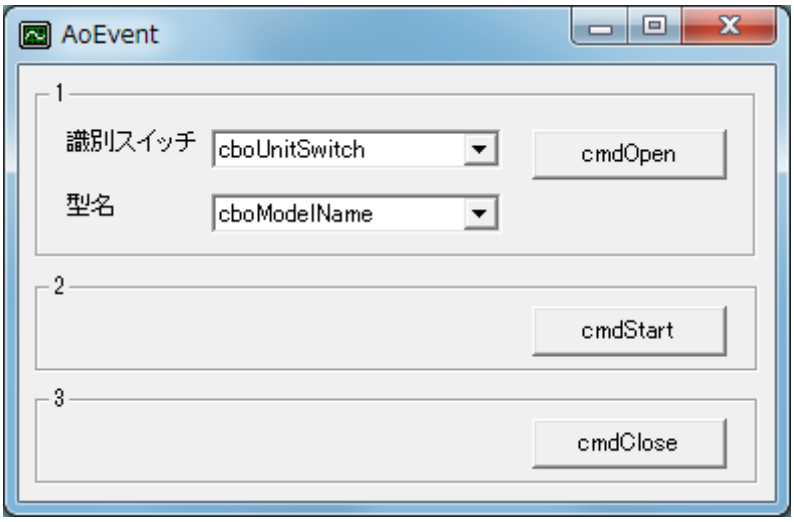
```
Private Sub Form1_Closing(ByVal sender As Object, ByVal e As
System.ComponentModel.CancelEventArgs) Handles MyBase.Closing
    result = YdxClose(id)
    If result <> 0 And result <> YDX_RESULT_NOT_OPEN Then
        ResultShow("YdxClose", result)
    End If
End Sub
```

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



定義

```
Private Declare Function CloseHandle Lib "kernel32" (ByVal hObject As Long) As Long
Private Declare Function CreateEvent Lib "kernel32" Alias "CreateEventA" (ByVal
lpEventAttributes As Long, ByVal bManualReset As Long, ByVal bInitialState As Long, ByVal
lpName As String) As Long
Private Declare Function WaitForSingleObject Lib "kernel32" (ByVal hHandle As Long, ByVal
dwMilliseconds As Long) As Long
Private Const INFINITE = &HFFFF
```

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_AI_EXCEED_DATA_NUM, Ydx.YDX_RESULT_AI_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 "AIO-64/4/1B-USC"
    cboModelName.AddItem "AIO-04/4/1B-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 = YdxAoSetBuffer(id, 0)    ' FIFOバッファ
    If result <> 0 Then
        Call ResultShow("YdxAoSetBuffer", result)
        Exit Sub
    End If

    ' チャネルの設定
    Const CHANNEL_NUM As Long = 4   ' 4チャネルを有効にする
    Dim channel As Long
    For channel = 0 To CHANNEL_NUM - 1
        result = YdxAoSetChannel(id, channel, 1)
        If result <> 0 Then
```

```
            Call ResultShow("YdxAoSetChannel", result)
            Exit Sub
        End If
    Next

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

    ' 内部クロック周期の設定
    result = YdxAoSetClockInternal(id, 1000) ' 1000μsec
    If result <> 0 Then
        Call ResultShow("YdxAoSetClockInternal", 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 * CHANNEL_NUM) As Single    ' データ個数は、サンプル数 * 有効チャネル数
    Dim i As Long
    For i = 0 To SAMPLE_NUM - 1
        Dim waveData As Single
        waveData = 10 * Math.Sin(2 * 3.14159265358979 * i / SAMPLE_NUM)     ' 正弦波
        For channel = 0 To CHANNEL_NUM - 1
            data(i * CHANNEL_NUM + channel) = waveData
        Next
    Next

    result = YdxAoSetDataVolt(id, SAMPLE_NUM, data(0))
    If result <> 0 Then
        Call ResultShow("YdxAoSetDataVolt", result)
        Exit Sub
    End If

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

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

    ' イベントオブジェクト作成
    Dim hEvent As Long
    hEvent = CreateEvent(0, False, False, 0)
    If hEvent = 0 Then
        MsgBox "イベントオブジェクトの作成に失敗しました", vbCritical
        Exit Sub
    End If

    ' イベントの設定
    result = YdxAoSetEvent(id, _
        YDX_EVENT_COMMUNICATE_ERR Or _
        YDX_EVENT_HARDWARE_ERR Or _
        YDX_EVENT_SAMPLE_CLOCK_ERR Or _
```

```
        YDX_EVENT_STOP, _
        hEvent)
    If result <> 0 Then
        Call ResultShow("YdxAoSetEvent", result)
        CloseHandle hEvent
        Exit Sub
    End If

    ' アナログ出力動作を開始
    result = YdxAoStart(id)
    If result <> 0 Then
        Call ResultShow("YdxAoStart", result)
        CloseHandle hEvent
        Exit Sub
    End If

    ' イベント発生待ち
    Call WaitForSingleObject(hEvent, INFINITE)
    CloseHandle hEvent

    ' ステータスの取得
    Dim factor, sampleCount, repeatCount, notOutNum As Long
    result = YdxAoGetEventStatus(id, factor, sampleCount, repeatCount, notOutNum)
    If result <> 0 Then
        Call ResultShow("YdxAoGetEventStatus", result)
        Exit Sub
    End If

    If (factor 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 (factor 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 (factor 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

    Call ResultShow("アナログ出力", 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
```

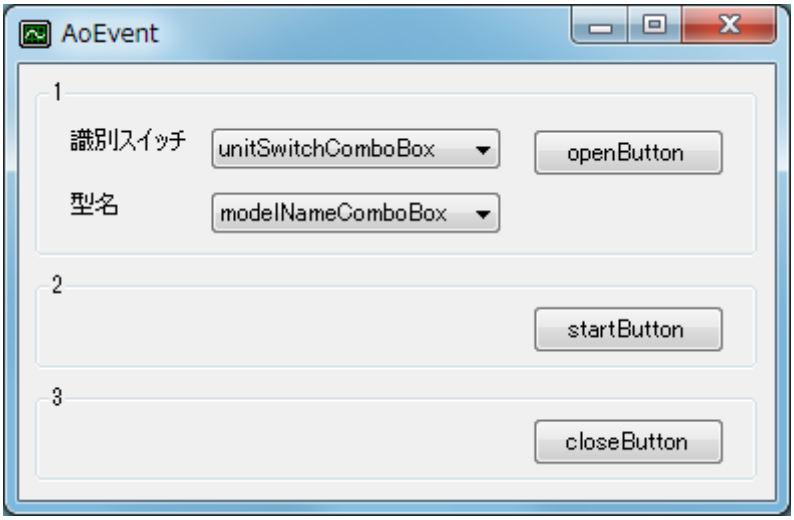
C++/CLI

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_AI_EXCEED_DATA_NUM:
    case YDX_RESULT_AI_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^> { "AIO-64/4/1B-USC", "AIO-
04/4/1B-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 = YdxAoSetBuffer(id, 0);   // FIFOバッファ
    if (result != 0)
    {
        ResultShow("YdxAoSetBuffer", result);
        return;
    }

    // チャネルの設定
    const int CHANNEL_NUM = 4;  // 4チャネルを有効にする
    for (int channel = 0; channel < CHANNEL_NUM; channel++)
    {
        result = YdxAoSetChannel(id, channel, 1);
        if (result != 0)
        {
```

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

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

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

    // データの設定
    const int SAMPLE_NUM = 1000;  // サンプル数
    float data[SAMPLE_NUM * CHANNEL_NUM];   // データ個数は、サンプル数 * 有効チャネル数
    for (int i = 0; i < SAMPLE_NUM; i++)
    {
        float waveData = (float)(10 * Math::Sin(2 * Math.PI * i / SAMPLE_NUM));    // 正弦波
        for (int channel = 0; channel < CHANNEL_NUM; channel++)
        {
            data[i * CHANNEL_NUM + channel] = waveData;
        }
    }
    result = YdxAoSetDataVolt(id, SAMPLE_NUM, data);
    if (result != 0)
    {
        ResultShow("YdxAoSetDataVolt", result);
        return;
    }

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

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

    // イベントオブジェクト作成
    AutoResetEvent^ hEvent = gcnew AutoResetEvent(false);

    // イベントの設定
    result = YdxAoSetEvent(id,
        YDX_EVENT_COMMUNICATE_ERR |
        YDX_EVENT_HARDWARE_ERR |
        YDX_EVENT_SAMPLE_CLOCK_ERR |
        YDX_EVENT_STOP,
```

```
        hEvent->Handle);
    if (result != 0)
    {
        ResultShow("YdxAoSetEvent", result);
        hEvent->Close();
        return;
    }

    // アナログ出力動作を開始
    result = YdxAoStart(id);
    if (result != 0)
    {
        ResultShow("YdxAoStart", result);
        hEvent->Close();
        return;
    }

    // イベント発生待ち
    hEvent->WaitOne();
    hEvent->Close();

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

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

    ResultShow("アナログ出力", 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);
}
```

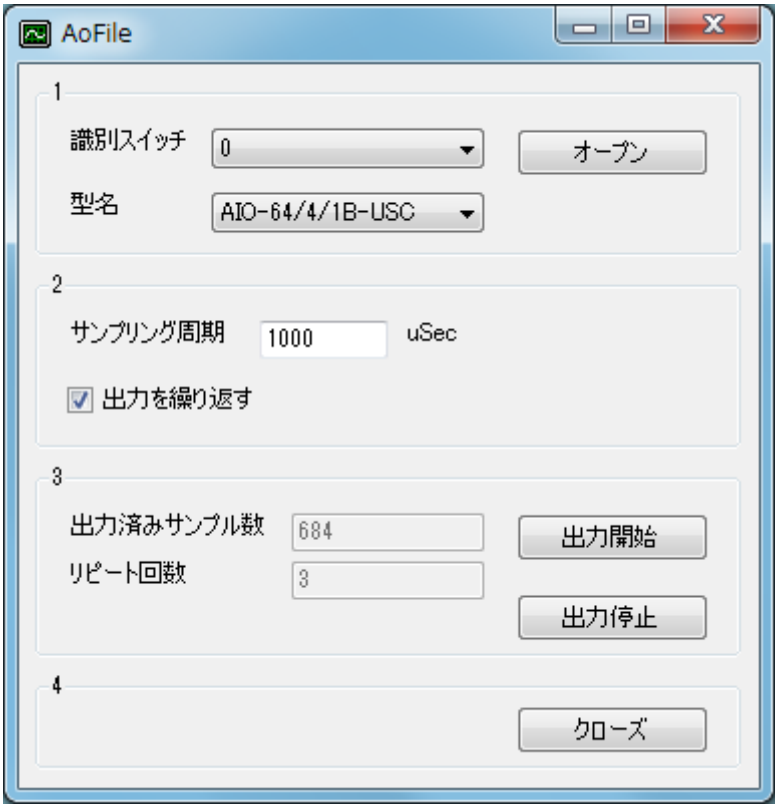
フォームクローズ

```
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);
    }
}
```

AoFile

高機能アナログ出力 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、アナログ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」→「ポーリングによる状態監視」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

波形データは実行ファイルと同じフォルダにある「WaveData.csv」です。
初期状態では、以下の波形データになっています。

チャネル	波形
0	振幅±10Vの正弦波
1	振幅±10Vの三角波
2	振幅±5Vの正弦波
3	振幅±5Vの三角波

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。
データ数は最大1,000,000サンプル分まで設定可能です。

サンプルソース

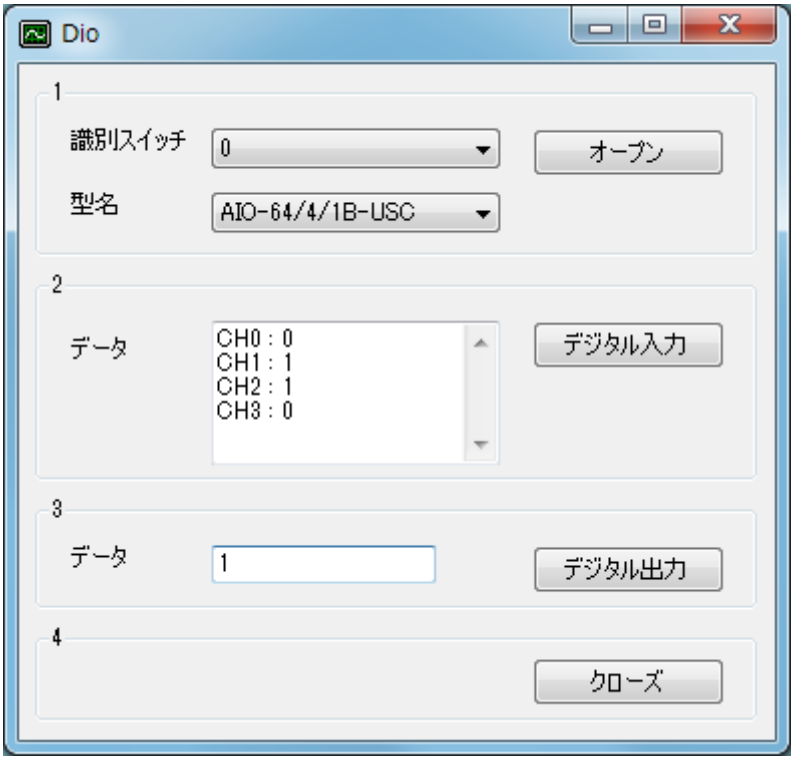
- C#

  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。

Dio

デジタル入力・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. 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

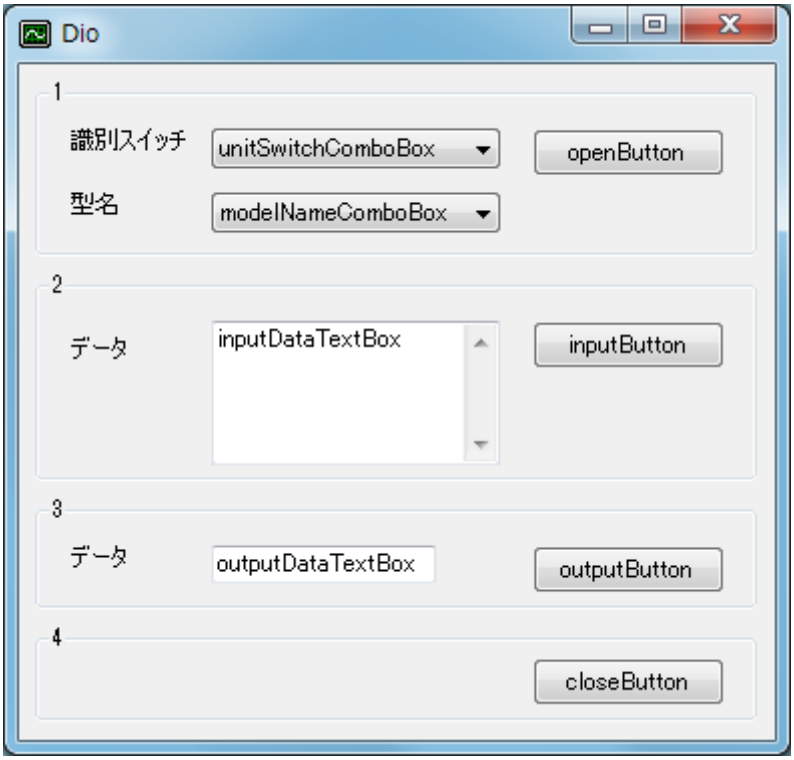
C#

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_AI_EXCEED_DATA_NUM:
        case Ydx.YDX_RESULT_AI_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[] { "AIO-64/4/1B-USC", "AIO-60/4/1B-USC",
"AIO-04/4/1B-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);
    }
}
```

デジタル入力

```
private void inputButton_Click(object sender, EventArgs e)
{
    inputDataTextBox.ResetText();
    Application.DoEvents();
    const int DI_CHANNEL_NUM = 4;
    int[] data = new int[DI_CHANNEL_NUM];
    int result = Ydx.DiInputBit(id, 0, DI_CHANNEL_NUM, 0, data);
    if (result != 0)
    {
        ResultShow("YdxDiInputBit", result);
        return;
    }

    string txt = "";
    for (int channel = 0; channel < DI_CHANNEL_NUM; channel++)
    {
```

```
        txt += "CH" + channel.ToString() + " : " + data[channel].ToString() +
Environment.NewLine;
    }

    inputDataTextBox.Text = txt;
}
```

デジタル出力

```
private void outputButton_Click(object sender, EventArgs e)
{
    int[] data = new int[1];
    double doubleData;
    if (!double.TryParse(outputDataTextBox.Text,System.Globalization.NumberStyles.Integer,
null, out doubleData))
    {
        MessageBox.Show("データが不正です", "", MessageBoxButtons.OK, MessageBoxIcon.Hand);
        return;
    }

    data[0] = (int)doubleData;
    int result = Ydx.DoOutputBit(id, 0, 1, data);
    ResultShow("デジタル出力", 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

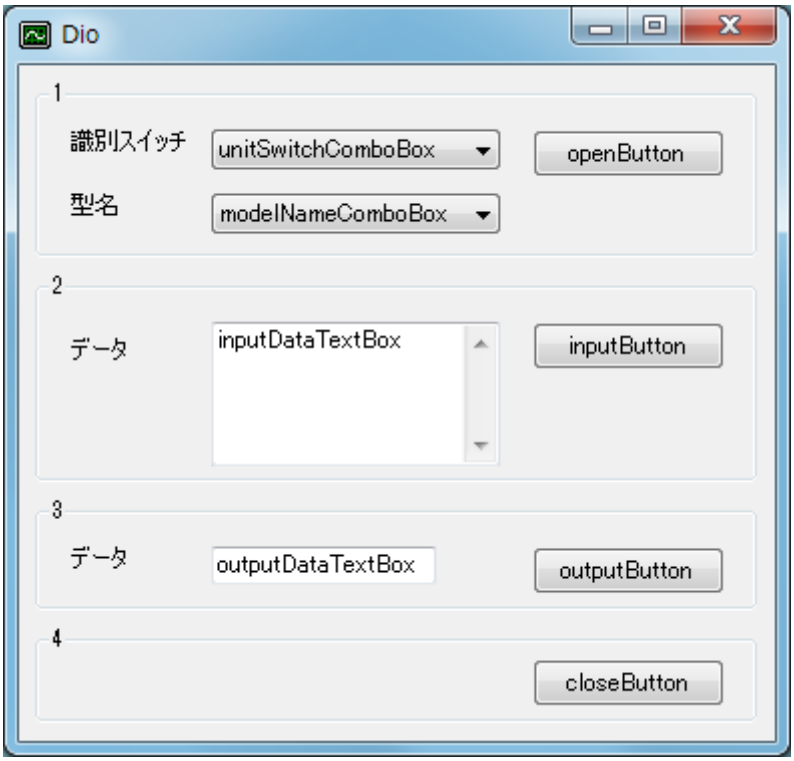
```
private void Form1_Closing(object sender, System.ComponentModel.CancelEventArgs e)
{
    int result = Ydx.Close(id);
    if ((result != 0) && (result != Ydx.YDX_RESULT_NOT_OPEN))
    {
        ResultShow("YdxClose", result);
    }
}
```

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 Short
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() { "AIO-64/4/1B-USC", "AIO-60/4/1B-USC",
"AIO-04/4/1B-USC" })
    modelNameComboBox.SelectedIndex
```

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
```

デジタル入力

```
Private Sub inputButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles inputButton.Click
    inputDataTextBox.ResetText()
    Application.DoEvents()
    Const DI_CHANNEL_NUM As Integer = 4
    Dim data(DI_CHANNEL_NUM - 1) As Integer
    result = YdxDiInputBit(id, 0, DI_CHANNEL_NUM, 0, data)
    If result <> 0 Then
        ResultShow("YdxDiInputBit", result)
        Exit Sub
    End If

    Dim txt As String = ""
    Dim channel As Integer
    For channel = 0 To DI_CHANNEL_NUM - 1
        txt += "CH" + channel.ToString() + " : " + data(channel).ToString() +
Environment.NewLine
    Next

    inputDataTextBox.Text = txt
End Sub
```

デジタル出力

```
Private Sub outputButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles outputButton.Click
    Dim data(0) As Integer
    Dim doubleData As Double
    If Double.TryParse(outputDataTextBox.Text, Globalization.NumberStyles.Integer, Nothing,
doubleData) = False Then
        MessageBox.Show("データが不正です", "", MessageBoxButtons.OK, MessageBoxIcon.Hand)
        Exit Sub
    End If

    data(0) = CType(doubleData, Integer)
    result = YdxDoOutputBit(id, 0, 1, data)
    ResultShow("デジタル出力", 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

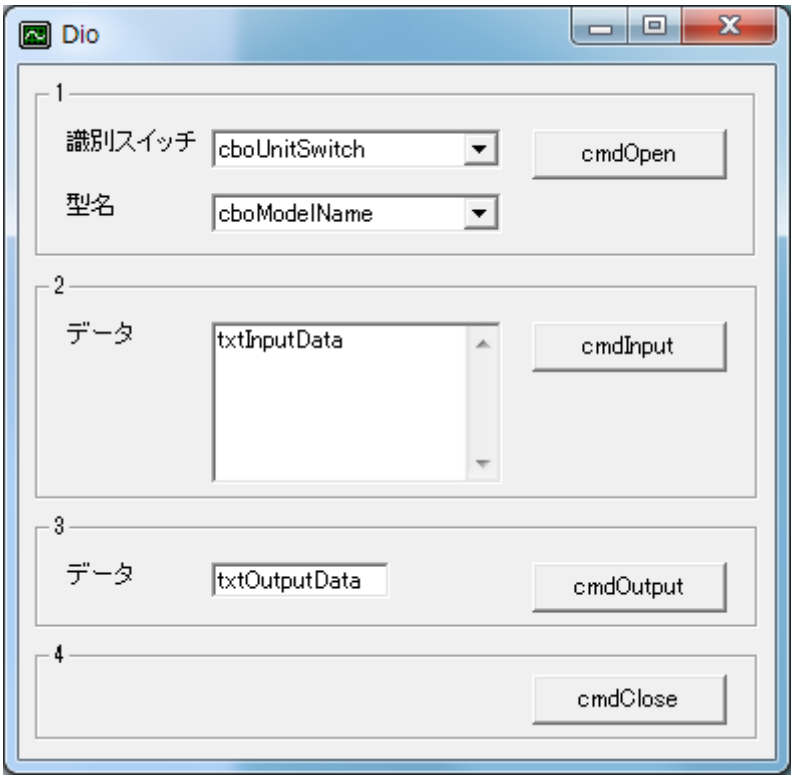
```
Private Sub Form1_Closing(ByVal sender As Object, ByVal e As
System.ComponentModel.CancelEventArgs) Handles MyBase.Closing
    result = YdxClose(id)
    If result <> 0 And result <> YDX_RESULT_NOT_OPEN Then
        ResultShow("YdxClose", result)
    End If
End Sub
```

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_AI_EXCEED_DATA_NUM, Ydx.YDX_RESULT_AI_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 "AIO-64/4/1B-USC"
    cboModelName.AddItem "AIO-60/4/1B-USC"
    cboModelName.AddItem "AIO-04/4/1B-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
```

デジタル入力

```
Private Sub cmdInput_Click()
    txtInputData.Text = ""
    DoEvents

    Const DI_CHANNEL_NUM As Long = 4
    Dim data(DI_CHANNEL_NUM - 1) As Long
    result = YdxDiInputBit(id, 0, DI_CHANNEL_NUM, 0, data(0))
    If result <> 0 Then
        Call ResultShow("YdxDiInputBit", result)
        Exit Sub
    End If

    Dim txt As String
    txt = ""
    Dim channel As Long
    For channel = 0 To DI_CHANNEL_NUM - 1
        txt = txt + "CH" + Format(channel) + " : " + Format(data(channel)) + vbCrLf
```

```
    Next

    txtInputData.Text = txt
End Sub
```

デジタル出力

```
Private Sub cmdOutput_Click()
    Dim data(0) As Long
    data(0) = txtOutputData.Text
    result = YdxDoOutputBit(id, 0, 1, data(0))
    Call ResultShow("デジタル出力", 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

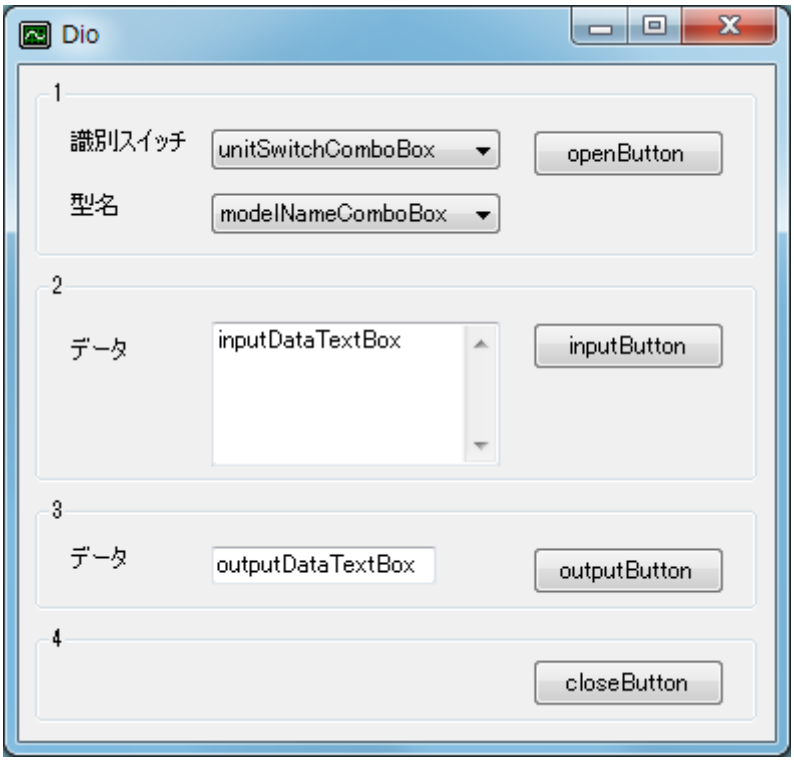
```
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    result = YdxClose(id)
    If result <> 0 And result <> YDX_RESULT_NOT_OPEN Then
        Call ResultShow("YdxClose", result)
    End If
End Sub
```

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_AI_EXCEED_DATA_NUM:
    case YDX_RESULT_AI_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^> { "AIO-64/4/1B-USC", "AIO-
60/4/1B-USC", "AIO-04/4/1B-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;
    }
}
```

デジタル入力

```
private: System::Void inputButton_Click(System::Object^ sender, System::EventArgs^ e)
{
    inputDataTextBox->ResetText();
    Application::DoEvents();
    const int DI_CHANNEL_NUM = 4;
    int data[DI_CHANNEL_NUM];
    int result = YdxDiInputBit(id, 0, DI_CHANNEL_NUM, 0, data);
    if (result != 0)
    {
        ResultShow("YdxDiInputBit", result);
        return;
    }

    String^ txt = "";
```

```
    for (int channel = 0; channel < DI_CHANNEL_NUM; channel++)
    {
        txt += "CH" + channel.ToString() + " : " + data[channel].ToString() +
Environment::NewLine;
    }
    inputDataTextBox->Text = txt;
}
```

デジタル出力

```
private: System::Void outputButton_Click(System::Object^ sender, System::EventArgs^ e)
{
    int data[1];
    if (!int::TryParse(outputDataTextBox->Text, data[0]))
    {
        MessageBox::Show("データが不正です", "", MessageBoxButtons::OK, MessageBoxIcon::Hand);
        return;
    }

    int result = YdxDoOutputBit(id, 0, 1, data);
    ResultShow("デジタル出力", 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);
    }
}
```

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

- 簡易アナログ入力

  アナログ入力を１回おこないたい場合
- 高機能アナログ入力

  連続サンプリングなど、色々な条件でアナログ入力をおこないたい場合
- 簡易アナログ出力

  アナログ出力を１回おこないたい場合
- 高機能アナログ出力

  連続サンプリングなど、色々な条件でアナログ出力をおこないたい場合

簡易アナログ入力

アナログ入力を１回おこないたい場合、簡易アナログ入力（YdxAiInput関数 または YdxAiInputVolt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

YdxAiSetRange関数 を使用して、入力レンジを設定します。
「±10V」または「±5V」を選択できます。
初期値は、±10V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

YdxAiInput関数 または YdxAiInputVolt関数 を使用して、アナログ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

- AiInputVolt

  簡易アナログ入力のサンプルプログラムです。
  アナログ入力には YdxAiInputVolt関数 を使用しています。

高機能アナログ入力

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。

- YdxAiSetRange関数 を使用して、入力レンジを設定します。

  ±10Vまたは±5Vが選択できます。
  初期値は、±10Vです。

- YdxAiSetBuffer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。

- YdxAiSetChannel関数 を使用して、サンプリングをおこなうチャネルを設定します。

  初期値では、サンプリングはチャネル0のみ有効（サンプリングをおこなう）になっています。

- YdxAiSetClock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アナログ入力（AD変換）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

    設定された周期でアナログ入力（AD変換）をおこないます。
    YdxAiSetClockInternal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アナログ入力（AD変換）をおこないます。
    YdxAiSetClockExternal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。

- YdxAiSetStartCondition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・アナログ入力トリガ（レンジ比較／インレンジ比較／アウトレンジ比較）から選択できます。
  また遅延回数の設定もおこなえます。 初期値は、ソフトウェア（自動）、遅延回数0回です。

  - 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。
  YdxAiSetStartExternal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。

- アナログ入力トリガ（レベル比較）

  アナログ入力状態が、しきい値以上またはしきい値以下になった時に、サンプリングを開始します。
  YdxAiSetStartLevel関数 または YdxAiSetStartLevelVolt関数 を使用して、アナログ入力チャネル・しきい値・動作モードを指定します。

- アナログ入力トリガ（インレンジ比較）

  アナログ入力状態が、２つのしきい値の範囲内になった時に、サンプリングを開始します。
  YdxAiSetStartInRange関数 または YdxAiSetStartInRangeVolt関数 を使用して、アナログ入力チャネル・しきい値・動作モードを指定します。

- アナログ入力トリガ（アウトレンジ比較）

  アナログ入力状態が、２つのしきい値の範囲外になった時に、サンプリングを開始します。
  YdxAiSetStartOutRange関数 または YdxAiSetStartOutRangeVolt関数 を使用して、アナログ入力チャネル・しきい値・動作モードを指定します。

- YdxAiSetStopCondition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・アナログ入力トリガ（レンジ比較／インレンジ比較／アウトレンジ比較）・ソフトウェアから選択できます。
  また遅延回数の設定もおこなえます。
  初期値は、サンプル数（1000回）、遅延回数0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。
    YdxAiSetStopSampleNum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。
    YdxAiSetStopExternal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。

  - アナログ入力トリガ（レベル比較）

    アナログ入力状態が、しきい値以上またはしきい値以下になった時に、サンプリングを停止します。
    YdxAiSetStopLevel関数 または YdxAiSetStopLevelVolt関数 を使用して、アナログ入力チャネル・しきい値・動作モードを指定します。

  - アナログ入力トリガ（インレンジ比較）

    アナログ入力状態が、２つのしきい値の範囲内になった時に、サンプリングを停止します。
    YdxAiSetStopInRange関数 または YdxAiSetStopInRangeVolt関数 を使用して、アナログ入力チャネル・しきい値・動作モードを指定します。

  - アナログ入力トリガ（アウトレンジ比較）

    アナログ入力状態が、２つのしきい値の範囲外になった時に、サンプリングを停止します。
    YdxAiSetStopOutRange関数 または YdxAiSetStopOutRangeVolt関数 を使用して、アナログ入力チャネル・しきい値・動作モードを指定します。

- YdxAiSetRepeat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（YdxAiSetStartCondition関数 で設定）からサンプリング停止条件（YdxAiSetStopCondition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。

- YdxAiSetEvent関数 を使用して、イベントを設定します。

  指定した要因となった時に、イベントを発生させる事ができます。
  イベントを使用しない場合は設定不要です。

入力動作の開始

以下のいずれかに該当する場合は、アナログ入力動作を開始する前に YdxAiClearData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

YdxAiStart関数 を使用して、アナログ入力動作を開始します。

動作状態の監視

YdxAiGetStatus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、アナログ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、YdxAiStop関数 を使用して停止する事も可能です。

データの取得

YdxAiGetData関数 または YdxAiGetDataVolt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

- AiPolling

  高機能アナログ入力のサンプルプログラムです。
  動作状態の監視をポーリングでおこなっています。

- AiEvent

  高機能アナログ入力のサンプルプログラムです。
  動作状態の監視をイベントでおこなっています。

簡易アナログ出力

アナログ出力を１回おこないたい場合、簡易アナログ出力（YdxAoOutput関数 または YdxAoOutputVolt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

YdxAoOutput関数 または YdxAoOutputVolt関数 を使用して、アナログ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

- AoOutputVolt

  簡易アナログ出力のサンプルプログラムです。
  アナログ出力には YdxAoOutputVolt関数 を使用しています。

高機能アナログ出力

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。

- YdxAoSetBuffer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。
- YdxAoSetChannel関数 を使用して、サンプリングをおこなうチャネルを設定します。

  初期値では、サンプリングはチャネル0のみ有効（サンプリングをおこなう）になっています。
- YdxAoSetClock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アナログ出力（DA変換）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

    設定された周期でアナログ出力（DA変換）をおこないます。
    YdxAoSetClockInternal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アナログ出力（DA変換）をおこないます。
    YdxAoSetClockExternal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。
- YdxAoSetStartCondition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・アナログ入力トリガ（レンジ比較／インレンジ比較／アウトレンジ比較）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。
    YdxAoSetStartExternal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。

- アナログ入力トリガ（レベル比較）

  アナログ入力状態が、しきい値以上またはしきい値以下になった時に、サンプリングを開始します。
  YdxAoSetStartLevel関数 または YdxAoSetStartLevelVolt関数 を使用して、アナログ入力チャネル・しきい値・動作モードを指定します。

- アナログ入力トリガ（インレンジ比較）

  アナログ入力状態が、２つのしきい値の範囲内になった時に、サンプリングを開始します。
  YdxAoSetStartInRange関数 または YdxAoSetStartInRangeVolt関数 を使用して、アナログ入力チャネル・しきい値・動作モードを指定します。

- アナログ入力トリガ（アウトレンジ比較）

  アナログ入力状態が、２つのしきい値の範囲外になった時に、サンプリングを開始します。
  YdxAoSetStartOutRange関数 または YdxAoSetStartOutRangeVolt関数 を使用して、アナログ入力チャネル・しきい値・動作モードを指定します。

- YdxAoSetStopCondition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・アナログ入力トリガ（レンジ比較／インレンジ比較／アウトレンジ比較）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。
    YdxAoSetStopExternal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。

  - アナログ入力トリガ（レベル比較）

    アナログ入力状態が、しきい値以上またはしきい値以下になった時に、サンプリングを停止します。
    YdxAoSetStopLevel関数 または YdxAoSetStopLevelVolt関数 を使用して、アナログ入力チャネル・しきい値・動作モードを指定します。

  - アナログ入力トリガ（インレンジ比較）

    アナログ入力状態が、２つのしきい値の範囲内になった時に、サンプリングを停止します。
    YdxAoSetStopInRange関数 または YdxAoSetStopInRangeVolt関数 を使用して、アナログ入力チャネル・しきい値・動作モードを指定します。

  - アナログ入力トリガ（アウトレンジ比較）

    アナログ入力状態が、２つのしきい値の範囲外になった時に、サンプリングを停止します。
    YdxAoSetStopOutRange関数 または YdxAoSetStopOutRangeVolt関数 を使用して、アナログ入力チャネル・しきい値・動作モードを指定します。

- YdxAoSetRepeat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（YdxAoSetStartCondition関数 で設定）からサンプリング停止条件（YdxAoSetStopCondition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。

- YdxAoSetEvent関数 を使用して、イベントを設定します。

  指定した要因となった時に、イベントを発生させる事ができます。
  イベントを使用しない場合は設定不要です。

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 YdxAoClearData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）

YdxAoSetData関数 または YdxAoSetDataVolt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

YdxAoStart関数 を使用して、アナログ出力動作を開始します。

動作状態の監視

YdxAoGetStatus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、アナログ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、YdxAoStop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

- AoPolling

  高機能アナログ出力のサンプルプログラムです。
  動作状態の監視をポーリングでおこなっています。

- AoEvent

  高機能アナログ出力のサンプルプログラムです。
  動作状態の監視をイベント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_AI_BUSY (CE000080h / -838860672)	アナログ入力が 動作中 です	
YDX_RESULT_AI_ERROR (CE000081h / -838860671)	アナログ入力動作 はエラー停止しています	YdxAiReset関数 を実行して、エラーを解除してください。
YDX_RESULT_AI_END (CE000082h / -838860670)	アナログ入力動作 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、YdxAiClearData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_AI_NO_CHANNEL (CE000083h / -838860669)	アナログ入力チャネルが設定されていません	YdxAiSetChannel関数 で、有効にするチャネルを指定してください。
YDX_RESULT_AI_EXCEED_DATA_NUM (CE000088h / -838860664)	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_AI_EXCEED_BUF_SIZ (CE000089h / -838860663)	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_AO_BUSY (CE000090h / -838860656)	アナログ出力が 動作中 です	

戻り値 (16進数値/10進数値)	意味	対処方法
YDX_RESULT_AO_ERROR (CE000091h / -838860655)	アナログ出力動作 は エラー停止しています	YdxAoReset関数 を実行して、エラーを解除してください。
YDX_RESULT_AO_END (CE000092h / -838860654)	アナログ出力動作 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、YdxAoClearData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_AO_NO_CHANNEL (CE000093h / -838860653)	アナログ出力チャネルが設定されていません	YdxAoSetChannel関数 で、有効にするチャネルを指定してください。
YDX_RESULT_AO_NO_DATA (CE000098h / -838860648)	データが設定されていません	YdxAoSetData関数 または YdxAoSetDataVolt関数 で、データを設定してください。
YDX_RESULT_AO_EXCEED_BUF_SIZ (CE000099h / -838860647)	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、アナログ出力を全て0Vにします。
1	デジタル出力の状態を変えません。（アナログ出力は全て0Vになります）

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「AIO-64/4/1B-USC」のユニットをオープンします。

C#

```
int result;
int id;
result = Ydx.Open(0, "AIO-64/4/1B-USC", 0, out id);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim id As Integer
result = YdxOpen(0, "AIO-64/4/1B-USC", 0, id)
```

VB6.0

```
Dim result As Long
Dim id As Long
result = YdxOpen(0, "AIO-64/4/1B-USC", 0, id)
```

C++/CLI

```
int result;
int id;
result = YdxOpen(0, "AIO-64/4/1B-USC", 0, &id);
```

C/C++

```
INT result;
INT id;
result = YdxOpen(0, "AIO-64/4/1B-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);
```

アナログ入力関数一覧

簡易アナログ入力 ・ 高機能アナログ入力　共用関数

関数	機能
YdxAiSetRange	入力レンジを設定します。
YdxAiGetRange	入力レンジの設定を取得します。

簡易アナログ入力　専用関数

関数	機能
YdxAiInput	アナログ入力端子の状態を読み込みます。 データは、バイナリ値で読み込みます。
YdxAiInputVolt	アナログ入力端子の状態を読み込みます。 データは、電圧値で読み込みます。

高機能アナログ入力　専用関数

設定

関数	機能
YdxAiSetBuffer	データバッファ の形式を設定します。
YdxAiSetChannel	チャネルの有効（サンプリングをおこなう）／無効（サンプリングをおこなわない）を設定します。
YdxAiSetCheckSampleNum	監視サンプル数 を設定します。
YdxAiSetClock	サンプリングクロック の種類を設定します。
YdxAiSetClockInternal	内部クロックを設定します。
YdxAiSetClockExternal	外部クロックを設定します。
YdxAiSetEvent	イベントを設定します。
YdxAiSetRepeat	リピート を設定します。
YdxAiSetStartCondition	サンプリング開始条件 を設定します。

関数	機能
YdxAiSetStartExternal	サンプリング開始条件（外部トリガ）を設定します。
YdxAiSetStartLevel	サンプリング開始条件（アナログ入力トリガ　レベル比較）を設定します。 しきい値は、バイナリ値で指定します。
YdxAiSetStartLevelVolt	サンプリング開始条件（アナログ入力トリガ　レベル比較）を設定します。 しきい値は、電圧値で指定します。
YdxAiSetStartInRange	サンプリング開始条件（アナログ入力トリガ　インレンジ比較）を設定します。 しきい値は、バイナリ値で指定します。
YdxAiSetStartInRangeVolt	サンプリング開始条件（アナログ入力トリガ　インレンジ比較）を設定します。 しきい値は、電圧値で指定します。
YdxAiSetStartOutRange	サンプリング開始条件（アナログ入力トリガ　アウトレンジ比較）を設定します。 しきい値は、バイナリ値で指定します。
YdxAiSetStartOutRangeVolt	サンプリング開始条件（アナログ入力トリガ　アウトレンジ比較）を設定します。 しきい値は、電圧値で指定します。
YdxAiSetStopCondition	サンプリング停止条件 を設定します。
YdxAiSetStopSampleNum	サンプリング停止条件（サンプル数）を設定します。
YdxAiSetStopExternal	サンプリング停止条件（外部トリガ）を設定します。
YdxAiSetStopLevel	サンプリング停止条件（アナログ入力トリガ　レベル比較）を設定します。 しきい値は、バイナリ値で指定します。
YdxAiSetStopLevelVolt	サンプリング停止条件（アナログ入力トリガ　レベル比較）を設定します。 しきい値は、電圧値で指定します。
YdxAiSetStopInRange	サンプリング停止条件（アナログ入力トリガ　インレンジ比較）を設定します。 しきい値は、バイナリ値で指定します。
YdxAiSetStopInRangeVolt	サンプリング停止条件（アナログ入力トリガ　インレンジ比較）を設定します。 しきい値は、電圧値で指定します。
YdxAiSetStopOutRange	サンプリング停止条件（アナログ入力トリガ　アウトレンジ比較）を設定します。 しきい値は、バイナリ値で指定します。
YdxAiSetStopOutRangeVolt	サンプリング停止条件（アナログ入力トリガ　アウトレンジ比較）を設定します。 しきい値は、電圧値で指定します。

設定の取得

関数	機能
YdxAiGetBuffer	データバッファの設定を取得します。

関数	機能
YdxAiGetChannel	チャネルの有効（サンプリングをおこなう）／無効（サンプリングをおこなわない）の設定を取得します。
YdxAiGetCheckSampleNum	監視サンプル数の設定を取得します。
YdxAiGetClock	サンプリングクロックの設定を取得します。
YdxAiGetClockInternal	内部クロックの設定を取得します。
YdxAiGetClockExternal	外部クロックの設定を取得します。
YdxAiGetEvent	イベントの設定を取得します。
YdxAiGetRepeat	リピートの設定を取得します。
YdxAiGetStartCondition	サンプリング開始条件の設定を取得します。
YdxAiGetStartExternal	サンプリング開始条件（外部トリガ）の設定を取得します。
YdxAiGetStartLevel	サンプリング開始条件（アナログ入力トリガ　レベル比較）の設定を取得します。 しきい値は、バイナリ値で取得します。
YdxAiGetStartLevelVolt	サンプリング開始条件（アナログ入力トリガ　レベル比較）の設定を取得します。 しきい値は、電圧値で取得します。
YdxAiGetStartInRange	サンプリング開始条件（アナログ入力トリガ　インレンジ比較）の設定を取得します。 しきい値は、バイナリ値で取得します。
YdxAiGetStartInRangeVolt	サンプリング開始条件（アナログ入力トリガ　インレンジ比較）の設定を取得します。 しきい値は、電圧値で取得します。
YdxAiGetStartOutRange	サンプリング開始条件（アナログ入力トリガ　アウトレンジ比較）の設定を取得します。 しきい値は、バイナリ値で取得します。
YdxAiGetStartOutRangeVolt	サンプリング開始条件（アナログ入力トリガ　アウトレンジ比較）の設定を取得します。 しきい値は、電圧値で取得します。
YdxAiGetStopCondition	サンプリング停止条件の設定を取得します。
YdxAiGetStopSampleNum	サンプリング停止条件（サンプル数）の設定を取得します。
YdxAiGetStopExternal	サンプリング停止条件（外部トリガ）の設定を取得します。
YdxAiGetStopLevel	サンプリング停止条件（アナログ入力トリガ　レベル比較）の設定を取得します。 しきい値は、バイナリ値で取得します。
YdxAiGetStopLevelVolt	サンプリング停止条件（アナログ入力トリガ　レベル比較）の設定を取得します。 しきい値は、電圧値で取得します。

関数	機能
YdxAiGetStopInRange	サンプリング停止条件（アナログ入力トリガ　インレンジ比較）の設定を取得します。 しきい値は、バイナリ値で取得します。
YdxAiGetStopInRangeVolt	サンプリング停止条件（アナログ入力トリガ　インレンジ比較）の設定を取得します。 しきい値は、電圧値で取得します。
YdxAiGetStopOutRange	サンプリング停止条件（アナログ入力トリガ　アウトレンジ比較）の設定を取得します。 しきい値は、バイナリ値で取得します。
YdxAiGetStopOutRangeVolt	サンプリング停止条件（アナログ入力トリガ　アウトレンジ比較）の設定を取得します。 しきい値は、電圧値で取得します。

動作の開始・停止

関数	機能
YdxAiStart	アナログ入力動作 を開始します。
YdxAiStop	アナログ入力動作 を停止します。

リセット

関数	機能
YdxAiReset	アナログ入力機能をリセットします。

状態の取得

関数	機能
YdxAiGetStatus	現在の状態を取得します。
YdxAiGetEventStatus	イベント発生要因を取得します。

データの取得・クリア

関数	機能
YdxAiGetData	データをバイナリ値で取得します。
YdxAiGetDataVolt	データを電圧値で取得します。

関数	機能
YdxAiClearData	データをクリアします。

YdxAiSetRange

機能

入力レンジを設定します。

書式

```
INT YdxAiSetRange(
    INT id,
    INT rangeType
);
```

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

rangeType

入力レンジを指定します。

値	意味
0	±10V
1	±5V

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、YdxAiStart関数 実行時にデータはクリアされます。

本関数は、アナログ入力が 動作中 には実行できません。

使用例

入力レンジを、±5Vに設定します。

C#

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

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAiGetRange

機能

入力レンジの設定を取得します。

書式

```
INT YdxAiGetRange(
    INT id,
    INT* rangeType
);
```

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

rangeType

入力レンジを格納する変数へのポインタを指定します。

値	意味
0	±10V
1	±5V

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、YdxAiSetRange関数 を参照してください。

使用例

入力レンジの設定を取得します。

C#

```
int result;
int rangeType;
result = Ydx.AiGetRange(id, out rangeType);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim rangeType As Integer
result = YdxAiGetRange(id, rangeType)
```

VB6.0

```
Dim result As Long
Dim rangeType As Long
result = YdxAiGetRange(id, rangeType)
```

C++/CLI

```
int result;
int rangeType;
result = YdxAiGetRange(id, &rangeType);
```

C/C++

```
INT result;
INT rangeType;
result = YdxAiGetRange(id, &rangeType);
```

YdxAiInput

機能

アナログ入力端子の状態を読み込みます。
データは、バイナリ値で読み込みます。

書式

```
INT YdxAiInput(
    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

入力開始チャネルを指定します。
設定範囲は0～5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

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。

データの値の範囲は、-30000～30000です。
電圧値への換算式は以下のとおりです。

- 電圧値 = data / 30000 * 10（±10Vレンジの場合）
- 電圧値 = data / 30000 * 5（±5Vレンジの場合）

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、アナログ入力が 動作中 には実行できません。

使用例

アナログ入力チャネル0～3（AIN0～AIN3）の端子の状態を読み込みます。
データはAIN0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.AiInput(id, 0, 4, data);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim data(3) As Integer
result = YdxAiInput(id, 0, 4, data)
```

VB6.0

```
Dim result As Long
Dim data(3) As Long
result = YdxAiInput(id, 0, 4, data(0))
```

C++/CLI

```
int result;
int data[4];
result = YdxAiInput(id, 0, 4, data);
```

C/C++

```
INT result;
INT data[4];
result = YdxAiInput(id, 0, 4, data);
```

YdxAiInputVolt

機能

アナログ入力端子の状態を読み込みます。
データは、電圧値で読み込みます。

書式

```
INT YdxAiInputVolt(
    INT id,
    INT start,
    INT num,
    FLOAT* volt
);
```

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0～5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

volt

入力データ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	float[]	Single	Single	float*	FLOAT*

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、アナログ入力が 動作中 には実行できません。

使用例

アナログ入力チャネル0～3（AIN0～AIN3）の端子の状態を読み込みます。
データはAIN0から順にバッファへ格納されます。

C#

```
int result;
float[] volt = new float[4];
result = Ydx.AiInputVolt(id, 0, 4, volt);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim volt(3) As Single
result = YdxAiInputVolt(id, 0, 4, volt)
```

VB6.0

```
Dim result As Long
Dim volt(3) As Single
result = YdxAiInputVolt(id, 0, 4, volt(0))
```

C++/CLI

```
int result;
float volt[4];
result = YdxAiInputVolt(id, 0, 4, volt);
```

C/C++

```
INT result;
float volt[4];
result = YdxAiInputVolt(id, 0, 4, volt);
```

YdxAiSetBuffer

機能

データバッファ の形式を設定します。

書式

```
INT YdxAiSetBuffer(
    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、アナログ入力が 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.AiSetBuffer(id, 1);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAiSetChannel

機能

チャネルの有効（サンプリングをおこなう）／無効（サンプリングをおこなわない）を設定します。

高機能アナログ入力時、有効に設定されたチャネルのみサンプリングをおこないます。
（無効に設定されたチャネルはサンプリングをおこないません）

書式

```
INT YdxAiSetChannel(
    INT id,
    INT channel,
    INT enable
);
```

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0～5です。

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

enable

有効/無効を指定します。

値	意味
0	無効（サンプリングをおこなわない）
1	有効（サンプリングをおこなう）

初期値はチャネル0は1・その他のチャネルは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、YdxAiStart関数 実行時にデータはクリアされます。

本関数は、アナログ入力が 動作中 には実行できません。

使用例

チャネル2（AIN2）を、有効に設定します。

C#

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

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAiSetCheckSampleNum

機能

監視サンプル数 を設定します。

書式

```
INT YdxAiSetCheckSampleNum(
    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。

- YdxAiGetStatus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。
- YdxAiSetEvent関数 で、監視サンプル数イベントを有効に設定してある場合、イベントが発生します。

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++

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、アナログ入力が 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.AiSetCheckSampleNum(id, 2000);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAiSetClock

機能

サンプリングクロック の種類を設定します。

書式

```
INT YdxAiSetClock(
    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、YdxAiSetClockInternal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、YdxAiSetClockExternal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、アナログ入力が 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.AiSetClock(id, 1);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAiSetClockInternal

機能

内部クロックを設定します。

書式

```
INT YdxAiSetClockInternal(
    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5.5～60,000,000 [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

YdxAiSetClock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。

本関数は、アナログ入力が 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.AiSetClockInternal(id, 2000);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAiSetClockExternal

機能

外部クロックを設定します。

書式

```
INT YdxAiSetClockExternal(
    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0～3、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

YdxAiSetClock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、アナログ入力が 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.AiSetClockExternal(id, 2, 1);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAiSetEvent

機能

イベントを設定します。

書式

```
INT YdxAiSetEvent(
    INT id,
    INT* mask,
    HANDLE* hEvent
);
```

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

mask

イベント要因を指定します。
指定した要因となった時に、イベントを発生させる事ができます。

値	定義名	イベント要因
00000001h	YDX_EVENT_STOP	動作終了 　動作が終了するとイベントを発生させます。
00000002h	YDX_EVENT_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イベントを発生させます。 　監視サンプル数は、YdxAiSetCheckSampleNum関数 で変更できます。
00000004h	YDX_EVENT_START_TRIG	開始条件成立 　開始条件が成立するとイベントを発生させます。 　リピートの場合、開始条件が成立するたびにイベントを発生させます。

値	定義名	イベント要因
00000008h	YDX_EVENT_STOP_TRIG	停止条件成立 　停止条件が成立するとイベントを発生させます。 　リピートの場合、停止条件が成立するたびにイベントを発生させます。
00010000h	YDX_EVENT_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イベントを発生させます。
00020000h	YDX_EVENT_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イベントを発生させます。
00040000h	YDX_EVENT_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イベントを発生させます。
00080000h	YDX_EVENT_COMMUNICATE_ERR	通信エラー発生 　USB通信に異常が検出された場合にイベントを発生させます。

論理和をおこなう事で、複数の要因を組み合わせて指定できます。

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

hEvent

イベントオブジェクトのハンドルを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	IntPtr	IntPtr	Long	IntPtr	HANDLE

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、アナログ入力が 動作中 には実行できません。

使用例

イベントを設定します。
イベント要因を「監視サンプル数」と「動作終了」に設定します。

C#

```
int result;
AutoResetEvent hEvent = new AutoResetEvent(false);
result = Ydx.AiSetEvent(id, YDX_EVENT_SAMPLE_NUM| YDX_EVENT_STOP, hEvent);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim hEvent As AutoResetEvent = New AutoResetEvent(False)
result = YdxAiSetEvent(id, YDX_EVENT_SAMPLE_NUM| YDX_EVENT_STOP, hEvent)
```

VB6.0

```
Dim result As Long
Dim hEvent As Long
hEvent = CreateEvent(0, False, False, 0)
result = YdxAiSetEvent(id, YDX_EVENT_SAMPLE_NUM| YDX_EVENT_STOP, hEvent)
```

C++/CLI

```
int result;
AutoResetEvent^ hEvent = gcnew AutoResetEvent(false);
result = YdxAiSetEvent(id, YDX_EVENT_SAMPLE_NUM| YDX_EVENT_STOP, hEvent->Handle);
```

C/C++

```
INT result;
HANDLE hEvent = CreateEvent(NULL, FALSE, FALSE, NULL);
result = YdxAiSetEvent(id, YDX_EVENT_SAMPLE_NUM| YDX_EVENT_STOP, hEvent);
```

参考

- AiEvent

  高機能アナログ入力のサンプルプログラムです。
  動作状態の監視をイベントでおこなっています。

YdxAiSetRepeat

機能

リピート を設定します。

書式

```
INT YdxAiSetRepeat(
    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（YdxAiSetStartCondition関数 で設定）からサンプリング停止条件（YdxAiSetStopCondition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、アナログ入力が 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.AiSetRepeat(id, 10);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAiSetStartCondition

機能

サンプリング開始条件 を設定します。

書式

```
INT YdxAiSetStartCondition(
    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
2	アナログ入力トリガ（レベル比較）
3	アナログ入力トリガ（インレンジ比較）
4	アナログ入力トリガ（アウトレンジ比較）

初期値は0です。

- 「外部トリガ」を指定する場合、YdxAiSetStartExternal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。

- 「アナログ入力トリガ（レベル比較）」を指定する場合、YdxAiSetStartLevel関数 （または YdxAiSetStartLevelVolt関数 ）で、使用するチャネルとモードの設定をしてください。
- 「アナログ入力トリガ（インレンジ比較）」を指定する場合、YdxAiSetStartInRange関数 （または YdxAiSetStartInRangeVolt関数 ）で、使用するチャネルとモードの設定をしてください。
- 「アナログ入力トリガ（アウトレンジ比較）」を指定する場合、YdxAiSetStartOutRange関数 （または YdxAiSetStartOutRangeVolt関数 ）で、使用する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。
設定範囲は0～2,147,483,647[回]、初期値は0[回]です。

サンプリング開始条件成立後、遅延回数のサンプリングクロックを待ってから、サンプリングを開始します。

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、アナログ入力が 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・遅延回数20回に設定します。

C#

```
int result;
result = Ydx.AiSetStartCondition(id, 1, 20);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxAiSetStartCondition(id, 1, 20)
```

VB6.0

```
Dim result As Long
result = YdxAiSetStartCondition(id, 1, 20)
```

C++/CLI

```
int result;
result = YdxAiSetStartCondition(id, 1, 20);
```

C/C++

```
INT result;
result = YdxAiSetStartCondition(id, 1, 20);
```

YdxAiSetStartExternal

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 YdxAiSetStartExternal(
    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0～3、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

YdxAiSetStartCondition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、アナログ入力が 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.AiSetStartExternal(id, 1, 2);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAiSetStartLevel

機能

サンプリング開始条件（アナログ入力トリガ　レベル比較）を設定します。
しきい値は、バイナリ値で指定します。

書式

```
INT YdxAiSetStartLevel(
    INT id,
    INT channel,
    INT level,
    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

比較をするチャネルを指定します。
設定範囲は0～5、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

level

しきい値を指定します。
設定範囲は-30,000～30,000、初期値は15,000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（しきい値未満から、しきい値以上に変化した時に、条件成立）
1	立ち下がりエッジセンス（しきい値を超える値から、しきい値以下に変化した時に、条件成立）
2	両エッジセンス
3	ハイレベルセンス（しきい値以上の時に、条件成立。最初からしきい値以上だった場合も、条件成立）
4	ローレベルセンス（しきい値以下の時に、条件成立。最初からしきい値以下だった場合も、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

YdxAiSetStartCondition関数 で、サンプリング開始条件として「アナログ入力トリガ（レベル比較）」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「アナログ入力トリガ（レベル比較）」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、アナログ入力が 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（アナログ入力トリガ　レベル比較）を設定します。
比較をするチャネルは1、しきい値は23,000、動作モードはローレベルセンスに設定します。

C#

```
int result;
result = Ydx.AiSetStartLevel(id, 1, 23000, 4);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxAiSetStartLevel(id, 1, 23000, 4)
```

VB6.0

```
Dim result As Long
result = YdxAiSetStartLevel(id, 1, 23000, 4)
```

C++/CLI

```
int result;
result = YdxAiSetStartLevel(id, 1, 23000, 4);
```

C/C++

```
INT result;
result = YdxAiSetStartLevel(id, 1, 23000, 4);
```

YdxAiSetStartLevelVolt

機能

サンプリング開始条件（アナログ入力トリガ　レベル比較）を設定します。
しきい値は、電圧値で指定します。

書式

```
INT YdxAiSetStartLevelVolt(
    INT id,
    INT channel,
    float volt,
    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

比較をするチャネルを指定します。
設定範囲は0～5、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

volt

しきい値を指定します。
単位は「V」です。
設定範囲は-10～10 [V]、初期値は5 [V]です。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	float	Single	Single	float	FLOAT

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（しきい値未満から、しきい値以上に変化した時に、条件成立）
1	立ち下がりエッジセンス（しきい値を超える値から、しきい値以下に変化した時に、条件成立）
2	両エッジセンス
3	ハイレベルセンス（しきい値以上の時に、条件成立。最初からしきい値以上だった場合も、条件成立）
4	ローレベルセンス（しきい値以下の時に、条件成立。最初からしきい値以下だった場合も、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

YdxAiSetStartCondition関数 で、サンプリング開始条件として「アナログ入力トリガ（レベル比較）」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「アナログ入力トリガ（レベル比較）」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。

> ⚠ しきい値は、バイナリ値に換算されて、内部に記憶されます。
> 本関数の実行後に 入力レンジが変更 されても再換算はおこなわれませんので、入力レンジ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、アナログ入力が 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（アナログ入力トリガ　レベル比較）を設定します。
比較をするチャネルは1、しきい値は2.3V、動作モードはローレベルセンスに設定します。

C#

```
int result;
result = Ydx.AiSetStartLevelVolt(id, 1, 2.3F, 4);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxAiSetStartLevelVolt(id, 1, 2.3, 4)
```

VB6.0

```
Dim result As Long
result = YdxAiSetStartLevelVolt(id, 1, 2.3, 4)
```

C++/CLI

```
int result;
result = YdxAiSetStartLevelVolt(id, 1, 2.3, 4);
```

C/C++

```
INT result;
result = YdxAiSetStartLevelVolt(id, 1, 2.3, 4);
```

YdxAiSetStartInRange

機能

サンプリング開始条件（アナログ入力トリガ　インレンジ比較）を設定します。
しきい値は、バイナリ値で指定します。

書式

```
INT YdxAiSetStartInRange(
    INT id,
    INT channel,
    INT level1,
    INT level2,
    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

比較をするチャネルを指定します。
設定範囲は0～5、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

level1・level2

しきい値を指定します。
設定範囲は-30,000～30,000、初期値はlevel1=-15,000・level2=15,000です。

「level1≦データ≦level2」または「level2≦データ≦level1」となった時に、条件が成立します。

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

mode

動作モードを指定します。

値	意味
0	エッジセンス（インレンジでない状態から、インレンジに変化した時に、条件成立）
1	レベルセンス（インレンジの時に、条件成立。最初からインレンジだった場合も、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

YdxAiSetStartCondition関数 で、サンプリング開始条件として「アナログ入力トリガ（インレンジ比較）」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「アナログ入力トリガ（インレンジ比較）」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、アナログ入力が 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（アナログ入力トリガ　インレンジ比較）を設定します。
比較をするチャネルは1、しきい値は-2,300と4,500、動作モードはレベルセンスに設定します。

C#

```
int result;
result = Ydx.AiSetStartInRange(id, 1, -2300, 4500, 1);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxAiSetStartInRange(id, 1, -2300, 4500, 1)
```

VB6.0

```
Dim result As Long
result = YdxAiSetStartInRange(id, 1, -2300, 4500, 1)
```

C++/CLI

```
int result;
result = YdxAiSetStartInRange(id, 1, -2300, 4500, 1);
```

C/C++

```
INT result;
result = YdxAiSetStartInRange(id, 1, -2300, 4500, 1);
```

YdxAiSetStartInRangeVolt

機能

サンプリング開始条件（アナログ入力トリガ　インレンジ比較）を設定します。
しきい値は、電圧値で指定します。

書式

```
INT YdxAiSetStartInRangeVolt(
    INT id,
    INT channel,
    float volt1,
    float volt2,
    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

比較をするチャネルを指定します。
設定範囲は0～5、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

volt1・volt2

しきい値を指定します。
単位は「V」です。
設定範囲は-10～10 [V]、初期値はvolt1=-5 [V]・volt2=5 [V]です。

「volt1≦電圧≦volt2」または「volt2≦電圧≦volt1」となった時に、条件が成立します。

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	float	Single	Single	float	FLOAT

mode

動作モードを指定します。

値	意味
0	エッジセンス（インレンジでない状態から、インレンジに変化した時に、条件成立）
1	レベルセンス（インレンジの時に、条件成立。最初からインレンジだった場合も、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

YdxAiSetStartCondition関数 で、サンプリング開始条件として「アナログ入力トリガ（インレンジ比較）」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「アナログ入力トリガ（インレンジ比較）」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。

> ⚠ しきい値は、バイナリ値に換算されて、内部に記憶されます。
> 本関数の実行後に 入力レンジが変更 されても再換算はおこなわれませんので、入力レンジ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、アナログ入力が 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（アナログ入力トリガ　インレンジ比較）を設定します。
比較をするチャネルは1、しきい値は-2.3Vと4.5V、動作モードはレベルセンスに設定します。

C#

```
int result;
result = Ydx.AiSetStartInRangeVolt(id, 1, -2.3F, 4.5F, 1);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxAiSetStartInRangeVolt(id, 1, -2.3, 4.5, 1)
```

VB6.0

```
Dim result As Long
result = YdxAiSetStartInRangeVolt(id, 1, -2.3, 4.5, 1)
```

C++/CLI

```
int result;
result = YdxAiSetStartInRangeVolt(id, 1, -2.3, 4.5, 1);
```

C/C++

```
INT result;
result = YdxAiSetStartInRangeVolt(id, 1, -2.3, 4.5, 1);
```

YdxAiSetStartOutRange

機能

サンプリング開始条件（アナログ入力トリガ　アウトレンジ比較）を設定します。
しきい値は、バイナリ値で指定します。

書式

```
INT YdxAiSetStartOutRange(
    INT id,
    INT channel,
    INT level1,
    INT level2,
    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

比較をするチャネルを指定します。
設定範囲は0～5、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

level1・level2

しきい値を指定します。
設定範囲は-30,000～30,000、初期値はlevel1=-15,000・level2=15,000です。

- level1≦level2の場合

  「データ≦level1 または level2≦データ」となった時に、条件が成立します。

- level2≦level1の場合

  「データ≦level2 または level1≦データ」となった時に、条件が成立します。

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	エッジセンス（アウトレンジでない状態から、アウトレンジに変化した時に、条件成立）
1	レベルセンス（アウトレンジの時に、条件成立。最初からアウトレンジだった場合も、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

YdxAiSetStartCondition関数 で、サンプリング開始条件として「アナログ入力トリガ（アウトレンジ比較）」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「アナログ入力トリガ（アウトレンジ比較）」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、アナログ入力が 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（アナログ入力トリガ　アウトレンジ比較）を設定します。
比較をするチャネルは1、しきい値は-2,300と4,500、動作モードはレベルセンスに設定します。

C#

```
int result;
result = Ydx.AiSetStartOutRange(id, 1, -2300, 4500, 1);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxAiSetStartOutRange(id, 1, -2300, 4500, 1)
```

VB6.0

```
Dim result As Long
result = YdxAiSetStartOutRange(id, 1, -2300, 4500, 1)
```

C++/CLI

```
int result;
result = YdxAiSetStartOutRange(id, 1, -2300, 4500, 1);
```

C/C++

```
INT result;
result = YdxAiSetStartOutRange(id, 1, -2300, 4500, 1);
```

YdxAiSetStartOutRangeVolt

機能

サンプリング開始条件（アナログ入力トリガ　アウトレンジ比較）を設定します。
しきい値は、電圧値で指定します。

書式

```
INT YdxAiSetStartOutRangeVolt(
    INT id,
    INT channel,
    float volt1,
    float volt2,
    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

比較をするチャネルを指定します。
設定範囲は0～5、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

volt1・volt2

しきい値を指定します。
単位は「V」です。
設定範囲は-10～10 [V]、初期値はvolt1=-5 [V]・volt2=5 [V]です。

- volt1≦volt2の場合

  「電圧≦volt1 または volt2≦電圧」となった時に、条件が成立します。

- volt2≦volt1の場合

「電圧≦volt2 または volt1≦電圧」となった時に、条件が成立します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	float	Single	Single	float	FLOAT

mode

動作モードを指定します。

値	意味
0	エッジセンス（アウトレンジでない状態から、アウトレンジに変化した時に、条件成立）
1	レベルセンス（アウトレンジの時に、条件成立。最初からアウトレンジだった場合も、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

YdxAiSetStartCondition関数 で、サンプリング開始条件として「アナログ入力トリガ（アウトレンジ比較）」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「アナログ入力トリガ（アウトレンジ比較）」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。

> ⚠ しきい値は、バイナリ値に換算されて、内部に記憶されます。
> 本関数の実行後に 入力レンジが変更 されても再換算はおこなわれませんので、入力レンジ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、アナログ入力が 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（アナログ入力トリガ　アウトレンジ比較）を設定します。
比較をするチャネルは1、しきい値は-2.3Vと4.5V、動作モードはレベルセンスに設定します。

C#

```
int result;
result = Ydx.AiSetStartOutRangeVolt(id, 1, -2.3F, 4.5F, 1);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxAiSetStartOutRangeVolt(id, 1, -2.3, 4.5, 1)
```

VB6.0

```
Dim result As Long
result = YdxAiSetStartOutRangeVolt(id, 1, -2.3, 4.5, 1)
```

C++/CLI

```
int result;
result = YdxAiSetStartOutRangeVolt(id, 1, -2.3, 4.5, 1);
```

C/C++

```
INT result;
result = YdxAiSetStartOutRangeVolt(id, 1, -2.3, 4.5, 1);
```

YdxAiSetStopCondition

機能

サンプリング停止条件 を設定します。

書式

```
INT YdxAiSetStopCondition(
    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	アナログ入力トリガ（レベル比較）
3	アナログ入力トリガ（インレンジ比較）
4	アナログ入力トリガ（アウトレンジ比較）
5	ソフトウェア（YdxAiStop関数）

初期値は0です。

- 「サンプル数」を指定する場合、YdxAiSetStopSampleNum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、YdxAiSetStopExternal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。
- 「アナログ入力トリガ（レベル比較）」を指定する場合、YdxAiSetStopLevel関数 （または YdxAiSetStopLevelVolt関数 ）で、使用するチャネルとモードの設定をしてください。
- 「アナログ入力トリガ（インレンジ比較）」を指定する場合、YdxAiSetStopInRange関数 （または YdxAiSetStopInRangeVolt関数 ）で、使用するチャネルとモードの設定をしてください。
- 「アナログ入力トリガ（アウトレンジ比較）」を指定する場合、YdxAiSetStopOutRange関数 （または YdxAiSetStopOutRangeVolt関数 ）で、使用する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。
設定範囲は0～2,147,483,647[回]、初期値は0[回]です。
サンプリング停止条件成立後、遅延回数のサンプリング後に、サンプリングを停止します。
ただし、停止条件がソフトウェアの場合は遅延はおこなわれず、YdxAiStop関数実行後に即時停止します。

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、アナログ入力が 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・遅延回数20回に設定します。

C#

```
int result;
result = Ydx.AiSetStopCondition(id, 1, 20);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxAiSetStopCondition(id, 1, 20)
```

VB6.0

```
Dim result As Long
result = YdxAiSetStopCondition(id, 1, 20)
```

C++/CLI

```
int result;
result = YdxAiSetStopCondition(id, 1, 20);
```

C/C++

```
INT result;
result = YdxAiSetStopCondition(id, 1, 20);
```

YdxAiSetStopSampleNum

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 YdxAiSetStopSampleNum(
    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

YdxAiSetStopCondition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、アナログ入力が 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.AiSetStopSampleNum(id, 1200);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAiSetStopExternal

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 YdxAiSetStopExternal(
    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0～3、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

YdxAiSetStopCondition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、アナログ入力が 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.AiSetStopExternal(id, 1, 2);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAiSetStopLevel

機能

サンプリング停止条件（アナログ入力トリガ　レベル比較）を設定します。
しきい値は、バイナリ値で指定します。

書式

```
INT YdxAiSetStopLevel(
    INT id,
    INT channel,
    INT level,
    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

比較をするチャネルを指定します。
設定範囲は0～5、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

level

しきい値を指定します。
設定範囲は-30,000～30,000、初期値は15,000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（しきい値未満から、しきい値以上に変化した時に、条件成立）
1	立ち下がりエッジセンス（しきい値を超える値から、しきい値以下に変化した時に、条件成立）
2	両エッジセンス
3	ハイレベルセンス（しきい値以上の時に、条件成立。最初からしきい値以上だった場合も、条件成立）
4	ローレベルセンス（しきい値以下の時に、条件成立。最初からしきい値以下だった場合も、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

YdxAiSetStopCondition関数 で、サンプリング停止条件として「アナログ入力トリガ（レベル比較）」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「アナログ入力トリガ（レベル比較）」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、アナログ入力が 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（アナログ入力トリガ　レベル比較）を設定します。
比較をするチャネルは1、しきい値は23,000、動作モードはローレベルセンスに設定します。

C#

```
int result;
result = Ydx.AiSetStopLevel(id, 1, 23000, 4);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxAiSetStopLevel(id, 1, 23000, 4)
```

VB6.0

```
Dim result As Long
result = YdxAiSetStopLevel(id, 1, 23000, 4)
```

C++/CLI

```
int result;
result = YdxAiSetStopLevel(id, 1, 23000, 4);
```

C/C++

```
INT result;
result = YdxAiSetStopLevel(id, 1, 23000, 4);
```

YdxAiSetStopLevelVolt

機能

サンプリング停止条件（アナログ入力トリガ　レベル比較）を設定します。
しきい値は、電圧値で指定します。

書式

```
INT YdxAiSetStopLevelVolt(
    INT id,
    INT channel,
    float volt,
    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

比較をするチャネルを指定します。
設定範囲は0～5、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

volt

しきい値を指定します。
単位は「V」です。
設定範囲は-10～10 [V]、初期値は5 [V]です。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	float	Single	Single	float	FLOAT

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（しきい値未満から、しきい値以上に変化した時に、条件成立）
1	立ち下がりエッジセンス（しきい値を超える値から、しきい値以下に変化した時に、条件成立）
2	両エッジセンス
3	ハイレベルセンス（しきい値以上の時に、条件成立。最初からしきい値以上だった場合も、条件成立）
4	ローレベルセンス（しきい値以下の時に、条件成立。最初からしきい値以下だった場合も、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

YdxAiSetStopCondition関数 で、サンプリング停止条件として「アナログ入力トリガ（レベル比較）」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「アナログ入力トリガ（レベル比較）」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。

> ⚠ しきい値は、バイナリ値に換算されて、内部に記憶されます。
> 本関数の実行後に 入力レンジが変更 されても再換算はおこなわれませんので、入力レンジ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、アナログ入力が 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（アナログ入力トリガ　レベル比較）を設定します。
比較をするチャネルは1、しきい値は2.3V、動作モードはローレベルセンスに設定します。

C#

```
int result;
result = Ydx.AiSetStopLevelVolt(id, 1, 2.3F, 4);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxAiSetStopLevelVolt(id, 1, 2.3, 4)
```

VB6.0

```
Dim result As Long
result = YdxAiSetStopLevelVolt(id, 1, 2.3, 4)
```

C++/CLI

```
int result;
result = YdxAiSetStopLevelVolt(id, 1, 2.3, 4);
```

C/C++

```
INT result;
result = YdxAiSetStopLevelVolt(id, 1, 2.3, 4);
```

YdxAiSetStopInRange

機能

サンプリング停止条件（アナログ入力トリガ　インレンジ比較）を設定します。
しきい値は、バイナリ値で指定します。

書式

```
INT YdxAiSetStopInRange(
    INT id,
    INT channel,
    INT level1,
    INT level2,
    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

比較をするチャネルを指定します。
設定範囲は0～5、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

level1・level2

しきい値を指定します。
設定範囲は-30,000～30,000、初期値はlevel1=-15,000・level2=15,000です。
「level1≦データ≦level2」または「level2≦データ≦level1」となった時に、条件が成立します。

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	エッジセンス（インレンジでない状態から、インレンジに変化した時に、条件成立）
1	レベルセンス（インレンジの時に、条件成立。最初からインレンジだった場合も、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

YdxAiSetStopCondition関数 で、サンプリング停止条件として「アナログ入力トリガ（インレンジ比較）」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「アナログ入力トリガ（インレンジ比較）」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、アナログ入力が 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（アナログ入力トリガ　インレンジ比較）を設定します。
比較をするチャネルは1、しきい値は-2,300と4,500、動作モードはレベルセンスに設定します。

C#

```
int result;
result = Ydx.AiSetStopInRange(id, 1, -2300, 4500, 1);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxAiSetStopInRange(id, 1, -2300, 4500, 1)
```

VB6.0

```
Dim result As Long
result = YdxAiSetStopInRange(id, 1, -2300, 4500, 1)
```

C++/CLI

```
int result;
result = YdxAiSetStopInRange(id, 1, -2300, 4500, 1);
```

C/C++

```
INT result;
result = YdxAiSetStopInRange(id, 1, -2300, 4500, 1);
```

YdxAiSetStopInRangeVolt

機能

サンプリング停止条件（アナログ入力トリガ　インレンジ比較）を設定します。
しきい値は、電圧値で指定します。

書式

```
INT YdxAiSetStopInRangeVolt(
    INT id,
    INT channel,
    float volt1,
    float volt2,
    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

比較をするチャネルを指定します。
設定範囲は0～5、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

volt1・volt2

しきい値を指定します。
単位は「V」です。
設定範囲は-10～10 [V]、初期値はvolt=-5 [V]・volt2=5 [V]です。
「volt1≦電圧≦volt2」または「volt2≦電圧≦volt1」となった時に、条件が成立します。

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	float	Single	Single	float	FLOAT

mode

動作モードを指定します。

値	意味
0	エッジセンス（インレンジでない状態から、インレンジに変化した時に、条件成立）
1	レベルセンス（インレンジの時に、条件成立。最初からインレンジだった場合も、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

YdxAiSetStopCondition関数 で、サンプリング停止条件として「アナログ入力トリガ（インレンジ比較）」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「アナログ入力トリガ（インレンジ比較）」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。

> ⚠ しきい値は、バイナリ値に換算されて、内部に記憶されます。
> 本関数の実行後に 入力レンジが変更 されても再換算はおこなわれませんので、入力レンジ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、アナログ入力が 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（アナログ入力トリガ　インレンジ比較）を設定します。
比較をするチャネルは1、しきい値は-2.3Vと4.5V、動作モードはレベルセンスに設定します。

C#

```
int result;
result = Ydx.AiSetStopInRangeVolt(id, 1, -2.3F, 4.5F, 1);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxAiSetStopInRangeVolt(id, 1, -2.3, 4.5, 1)
```

VB6.0

```
Dim result As Long
result = YdxAiSetStopInRangeVolt(id, 1, -2.3, 4.5, 1)
```

C++/CLI

```
int result;
result = YdxAiSetStopInRangeVolt(id, 1, -2.3, 4.5, 1);
```

C/C++

```
INT result;
result = YdxAiSetStopInRangeVolt(id, 1, -2.3, 4.5, 1);
```

YdxAiSetStopOutRange

機能

サンプリング停止条件（アナログ入力トリガ　アウトレンジ比較）を設定します。
しきい値は、バイナリ値で指定します。

書式

```
INT YdxAiSetStopOutRange(
    INT id,
    INT channel,
    INT level1,
    INT level2,
    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

比較をするチャネルを指定します。
設定範囲は0～5、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

level1・level2

しきい値を指定します。
設定範囲は-30,000～30,000、初期値はlevel1=-15,000・level2=15,000です。

- level1≦level2の場合

  「データ≦level1 または level2≦データ」となった時に、条件が成立します。

- level2≦level1の場合

  「データ≦level2 または level1≦データ」となった時に、条件が成立します。

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	エッジセンス（アウトレンジでない状態から、アウトレンジに変化した時に、条件成立）
1	レベルセンス（アウトレンジの時に、条件成立。最初からアウトレンジだった場合も、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

YdxAiSetStopCondition関数 で、サンプリング停止条件として「アナログ入力トリガ（アウトレンジ比較）」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「アナログ入力トリガ（アウトレンジ比較）」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、アナログ入力が 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（アナログ入力トリガ　アウトレンジ比較）を設定します。
比較をするチャネルは1、しきい値は-2,300と4,500、動作モードはレベルセンスに設定します。

C#

```
int result;
result = Ydx.AiSetStopOutRange(id, 1, -2300, 4500, 1);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxAiSetStopOutRange(id, 1, -2300, 4500, 1)
```

VB6.0

```
Dim result As Long
result = YdxAiSetStopOutRange(id, 1, -2300, 4500, 1)
```

C++/CLI

```
int result;
result = YdxAiSetStopOutRange(id, 1, -2300, 4500, 1);
```

C/C++

```
INT result;
result = YdxAiSetStopOutRange(id, 1, -2300, 4500, 1);
```

YdxAiSetStopOutRangeVolt

機能

サンプリング停止条件（アナログ入力トリガ　アウトレンジ比較）を設定します。
しきい値は、電圧値で指定します。

書式

```
INT YdxAiSetStopOutRangeVolt(
    INT id,
    INT channel,
    float volt1,
    float volt2,
    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

比較をするチャネルを指定します。
設定範囲は0～5、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

volt1・volt2

しきい値を指定します。
単位は「V」です。
設定範囲は-10～10 [V]、初期値はvolt1=-5 [V]・volt2=5 [V]です。

- volt1≦volt2の場合

  「電圧≦volt1 または volt2≦電圧」となった時に、条件が成立します。

- volt2≦volt1の場合

「電圧≦volt2 または volt1≦電圧」となった時に、条件が成立します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	float	Single	Single	float	FLOAT

mode

動作モードを指定します。

値	意味
0	エッジセンス（アウトレンジでない状態から、アウトレンジに変化した時に、条件成立）
1	レベルセンス（アウトレンジの時に、条件成立。最初からアウトレンジだった場合も、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

YdxAiSetStopCondition関数 で、サンプリング停止条件として「アナログ入力トリガ（アウトレンジ比較）」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「アナログ入力トリガ（アウトレンジ比較）」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。

> ⚠ しきい値は、バイナリ値に換算されて、内部に記憶されます。
> 本関数の実行後に 入力レンジが変更 されても再換算はおこなわれませんので、入力レンジ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、アナログ入力が 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（アナログ入力トリガ　アウトレンジ比較）を設定します。
比較をするチャネルは1、しきい値は-2.3Vと4.5V、動作モードはレベルセンスに設定します。

C#

```
int result;
result = Ydx.AiSetStopOutRangeVolt(id, 1, -2.3F, 4.5F, 1);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxAiSetStopOutRangeVolt(id, 1, -2.3, 4.5, 1)
```

VB6.0

```
Dim result As Long
result = YdxAiSetStopOutRangeVolt(id, 1, -2.3, 4.5, 1)
```

C++/CLI

```
int result;
result = YdxAiSetStopOutRangeVolt(id, 1, -2.3, 4.5, 1);
```

C/C++

```
INT result;
result = YdxAiSetStopOutRangeVolt(id, 1, -2.3, 4.5, 1);
```

YdxAiGetBuffer

機能

データバッファの設定を取得します。

書式

```
INT YdxAiGetBuffer(
    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、YdxAiSetBuffer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.AiGetBuffer(id, out bufferType);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAiGetChannel

機能

チャネルの有効（サンプリングをおこなう）／無効（サンプリングをおこなわない）の設定を取得します。

書式

```
INT YdxAiGetChannel(
    INT id,
    INT channel,
    INT* enable
);
```

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0～5です。

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

enable

有効/無効の設定を格納する変数へのポインタを指定します。

値	意味
0	無効（サンプリングをおこなわない）
1	有効（サンプリングをおこなう）

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、YdxAiSetChannel関数 を参照してください。

使用例

チャネル2の、有効/無効の設定を取得します。

C#

```
int result;
int enable;
result = Ydx.AiGetChannel(id, 2, out enable);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim enable As Integer
result = YdxAiGetChannel(id, 2, enable)
```

VB6.0

```
Dim result As Long
Dim enable As Long
result = YdxAiGetChannel(id, 2, enable)
```

C++/CLI

```
int result;
int enable;
result = YdxAiGetChannel(id, 2, &enable);
```

C/C++

```
INT result;
INT enable;
result = YdxAiGetChannel(id, 2, &enable);
```

YdxAiGetCheckSampleNum

機能

監視サンプル数の設定を取得します。

書式

```
INT YdxAiGetCheckSampleNum(
    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、YdxAiSetCheckSampleNum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.AiGetCheckSampleNum(id, out sampleNum);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAiGetClock

機能

サンプリングクロックの設定を取得します。

書式

```
INT YdxAiGetClock(
    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、YdxAiSetClock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.AiGetClock(id, out clockType);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAiGetClockInternal

機能

内部クロックの設定を取得します。

書式

```
INT YdxAiGetClockInternal(
    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、YdxAiSetClockInternal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.AiGetClockInternal(id, out period);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAiGetClockExternal

機能

外部クロックの設定を取得します。

書式

```
INT YdxAiGetClockExternal(
    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、YdxAiSetClockExternal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.AiGetClockExternal(id, out diChannel, out edge);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAiGetEvent

機能

イベントの設定を取得します。

書式

```
INT YdxAiGetEvent(
    INT id,
    INT* mask,
    HANDLE* hEvent
);
```

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

mask

イベント要因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	イベント要因
00000001h	YDX_EVENT_STOP	動作終了
00000002h	YDX_EVENT_SAMPLE_NUM	サンプル数
00000004h	YDX_EVENT_START_TRIG	開始条件成立
00000008h	YDX_EVENT_STOP_TRIG	停止条件成立
00010000h	YDX_EVENT_SAMPLE_CLOCK_ERR	サンプリングクロックエラー
00020000h	YDX_EVENT_OVERRUN_ERR	オーバランエラー
00040000h	YDX_EVENT_HARDWARE_ERR	ハードウェアエラー

値	定義名	イベント要因
00080000h	YDX_EVENT_COMMUNICATE_ERR	通信エラー

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

hEvent

イベントオブジェクトのハンド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 IntPtr	IntPtr	Long	IntPtr*	HANDLE*

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、YdxAiSetEvent関数 を参照してください。

使用例

イベントの設定を読み出します。

C#

```
int result;
int mask;
IntPtr hEvent;
result = Ydx.AiGetEvent(id, out mask, out hEvent);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim mask As Integer
Dim hEvent As IntPtr
result = YdxAiGetEvent(id, mask, hEvent)
```

VB6.0

```
Dim result As Long
Dim mask As Long
Dim hEvent As Long
result = YdxAiGetEvent(id, mask, hEvent)
```

C++/CLI

```
int result;
int mask;
IntPtr hEvent;
result = YdxAiGetEvent(id, &mask, &hEvent);
```

C/C++

```
INT result;
INT mask;
HANDLE hEvent;
result = YdxAiGetEvent(id, &mask, &hEvent);
```

YdxAiGetRepeat

機能

リピートの設定を取得します。

書式

```
INT YdxAiGetRepeat(
    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、YdxAiSetRepeat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.AiGetRepeat(id, out repeatNum);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAiGetStartCondition

機能

サンプリング開始条件の設定を取得します。

書式

```
INT YdxAiGetStartCondition(
    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
2	アナログ入力トリガ（レベル比較）
3	アナログ入力トリガ（インレンジ比較）
4	アナログ入力トリガ（アウトレンジ比較）

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。

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、YdxAiSetStartCondition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.AiGetStartCondition(id, out condition, out delay);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAiGetStartExternal

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 YdxAiGetStartExternal(
    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、YdxAiSetStartExternal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.AiGetStartExternal(id, out diChannel, out mode);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAiGetStartLevel

機能

サンプリング開始条件（アナログ入力トリガ　レベル比較）の設定を取得します。
しきい値は、バイナリ値で取得します。

書式

```
INT YdxAiGetStartLevel(
    INT id,
    INT* channel,
    INT* level,
    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

比較をする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*

level

しきい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*

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（しきい値未満から、しきい値以上に変化した時に、条件成立）
1	立ち下がりエッジセンス（しきい値を超える値から、しきい値以下に変化した時に、条件成立）
2	両エッジセンス
3	ハイレベルセンス（しきい値以上の時に、条件成立。最初からしきい値以上だった場合も、条件成立）
4	ローレベルセンス（しきい値以下の時に、条件成立。最初からしきい値以下だった場合も、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、YdxAiSetStartLevel関数 を参照してください。

使用例

サンプリング開始条件（アナログ入力トリガ　レベル比較）の設定を取得します。

C#

```
int result;
int channel;
int level;
int mode;
result = Ydx.AiGetStartLevel(id, out channel, out level, out mode);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim channel As Integer
Dim level1 As Integer
Dim mode As Integer
result = YdxAiGetStartLevel(id, channel, level, mode)
```

VB6.0

```
Dim result As Long
Dim channel As Long
Dim level1 As Long
Dim mode As Long
result = YdxAiGetStartLevel(id, channel, level, mode)
```

C++/CLI

```
int result;
int channel;
int level;
int mode;
result = YdxAiGetStartLevel(id, &channel, &level, &mode);
```

C/C++

```
INT result;
INT channel;
INT level;
INT mode;
result = YdxAiGetStartLevel(id, &channel, &level, &mode);
```

YdxAiGetStartLevelVolt

機能

サンプリング開始条件（アナログ入力トリガ　レベル比較）の設定を取得します。
しきい値は、電圧値で取得します。

書式

```
INT YdxAiGetStartLevelVolt(
    INT id,
    INT* channel,
    float* volt,
    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

比較をする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*

volt

しきい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 float	Single	Single	float*	FLOAT*

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（しきい値未満から、しきい値以上に変化した時に、条件成立）
1	立ち下がりエッジセンス（しきい値を超える値から、しきい値以下に変化した時に、条件成立）
2	両エッジセンス
3	ハイレベルセンス（しきい値以上の時に、条件成立。最初からしきい値以上だった場合も、条件成立）
4	ローレベルセンス（しきい値以下の時に、条件成立。最初からしきい値以下だった場合も、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、YdxAiSetStartLevelVolt関数 を参照してください。

使用例

サンプリング開始条件（アナログ入力トリガ　レベル比較）の設定を取得します。

C#

```
int result;
int channel;
float volt;
int mode;
result = Ydx.AiGetStartLevelVolt(id, out channel, out volt, out mode);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim channel As Integer
Dim volt As Single
Dim mode As Integer
result = YdxAiGetStartLevelVolt(id, channel, volt, mode)
```

VB6.0

```
Dim result As Long
Dim channel As Long
Dim volt As Single
Dim mode As Long
result = YdxAiGetStartLevelVolt(id, channel, volt, mode)
```

C++/CLI

```
int result;
int channel;
float volt;
int mode;
result = YdxAiGetStartLevelVolt(id, &channel, &volt, &mode);
```

C/C++

```
INT result;
INT channel;
float volt;
INT mode;
result = YdxAiGetStartLevelVolt(id, &channel, &volt, &mode);
```

YdxAiGetStartInRange

機能

サンプリング開始条件（アナログ入力トリガ　インレンジ比較）の設定を取得します。
しきい値は、バイナリ値で取得します。

書式

```
INT YdxAiGetStartInRange(
    INT id,
    INT* channel,
    INT* level1,
    INT* level2,
    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

比較をする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*

level1・level2

しきい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*

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	エッジセンス（インレンジでない状態から、インレンジに変化した時に、条件成立）
1	レベルセンス（インレンジの時に、条件成立。最初からインレンジだった場合も、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、YdxAiSetStartInRange関数 を参照してください。

使用例

サンプリング開始条件（アナログ入力トリガ　インレンジ比較）の設定を取得します。

C#

```
int result;
int channel;
int level1;
int level2;
int mode;
result = Ydx.AiGetStartInRange(id, out channel, out level1, out level2, out mode);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim channel As Integer
Dim level1 As Integer
Dim level2 As Integer
```

```
Dim mode As Integer
result = YdxAiGetStartInRange(id, channel, level1, level2, mode)
```

VB6.0

```
Dim result As Long
Dim channel As Long
Dim level1 As Long
Dim level2 As Long
Dim mode As Long
result = YdxAiGetStartInRange(id, channel, level1, level2, mode)
```

C++/CLI

```
int result;
int channel;
int level1;
int level2;
int mode;
result = YdxAiGetStartInRange(id, &channel, &level1, &level2, &mode);
```

C/C++

```
INT result;
INT channel;
INT level1;
INT level2;
INT mode;
result = YdxAiGetStartInRange(id, &channel, &level1, &level2, &mode);
```

YdxAiGetStartInRangeVolt

機能

サンプリング開始条件（アナログ入力トリガ　インレンジ比較）の設定を取得します。
しきい値は、電圧値で取得します。

書式

```
INT YdxAiGetStartInRangeVolt(
    INT id,
    INT* channel,
    float* volt1,
    float* volt2,
    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

比較をする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*

volt1・volt2

しきい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 float	Single	Single	float*	FLOAT*

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	エッジセンス（インレンジでない状態から、インレンジに変化した時に、条件成立）
1	レベルセンス（インレンジの時に、条件成立。最初からインレンジだった場合も、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、YdxAiSetStartInRangeVolt関数 を参照してください。

使用例

サンプリング開始条件（アナログ入力トリガ　インレンジ比較）の設定を取得します。

C#

```
int result;
int channel;
float volt1;
float volt2;
int mode;
result = Ydx.AiGetStartInRangeVolt(id, out channel, out volt1, out volt2, out mode);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim channel As Integer
Dim volt1 As Single
Dim volt2 As Single
```

```
Dim mode As Integer
result = YdxAiGetStartInRangeVolt(id, channel, volt1, volt2, mode)
```

VB6.0

```
Dim result As Long
Dim channel As Long
Dim volt1 As Single
Dim volt2 As Single
Dim mode As Long
result = YdxAiGetStartInRangeVolt(id, channel, volt1, volt2, mode)
```

C++/CLI

```
int result;
int channel;
float volt1;
float volt2;
int mode;
result = YdxAiGetStartInRangeVolt(id, &channel, &volt1, &volt2, &mode);
```

C/C++

```
INT result;
INT channel;
float volt1;
float volt2;
INT mode;
result = YdxAiGetStartInRangeVolt(id, &channel, &volt1, &volt2, &mode);
```

YdxAiGetStartOutRange

機能

サンプリング開始条件（アナログ入力トリガ　アウトレンジ比較）の設定を取得します。
しきい値は、バイナリ値で取得します。

書式

```
INT YdxAiGetStartOutRange(
    INT id,
    INT* channel,
    INT* level1,
    INT* level2,
    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

比較をする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*

level1・level2

しきい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*

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	エッジセンス（アウトレンジでない状態から、アウトレンジに変化した時に、条件成立）
1	レベルセンス（アウトレンジの時に、条件成立。最初からアウトレンジだった場合も、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、YdxAiSetStartOutRange関数 を参照してください。

使用例

サンプリング開始条件（アナログ入力トリガ　アウトレンジ比較）の設定を取得します。

C#

```
int result;
int channel;
int level1;
int level2;
int mode;
result = Ydx.AiGetStartOutRange(id, out channel, out level1, out level2, out mode);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim channel As Integer
Dim level1 As Integer
Dim level2 As Integer
```

```
Dim mode As Integer
result = YdxAiGetStartOutRange(id, channel, level1, level2, mode)
```

VB6.0

```
Dim result As Long
Dim channel As Long
Dim level1 As Long
Dim level2 As Long
Dim mode As Long
result = YdxAiGetStartOutRange(id, channel, level1, level2, mode)
```

C++/CLI

```
int result;
int channel;
int level1;
int level2;
int mode;
result = YdxAiGetStartOutRange(id, &channel, &level1, &level2, &mode);
```

C/C++

```
INT result;
INT channel;
INT level1;
INT level2;
INT mode;
result = YdxAiGetStartOutRange(id, &channel, &level1, &level2, &mode);
```

YdxAiGetStartOutRangeVolt

機能

サンプリング開始条件（アナログ入力トリガ　アウトレンジ比較）の設定を取得します。
しきい値は、電圧値で取得します。

書式

```
INT YdxAiGetStartOutRangeVolt(
    INT id,
    INT* channel,
    float* volt1,
    float* volt2,
    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

比較をする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*

volt1・volt2

しきい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 float	Single	Single	float*	FLOAT*

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	エッジセンス（アウトレンジでない状態から、アウトレンジに変化した時に、条件成立）
1	レベルセンス（アウトレンジの時に、条件成立。最初からアウトレンジだった場合も、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、YdxAiSetStartOutRangeVolt関数 を参照してください。

使用例

サンプリング開始条件（アナログ入力トリガ　アウトレンジ比較）の設定を取得します。

C#

```
int result;
int channel;
float volt1;
float volt2;
int mode;
result = Ydx.AiGetStartOutRangeVolt(id, out channel, out volt1, out volt2, out mode);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim channel As Integer
Dim volt1 As Single
Dim volt2 As Single
```

```
Dim mode As Integer
result = YdxAiGetStartOutRangeVolt(id, channel, volt1, volt2, mode)
```

VB6.0

```
Dim result As Long
Dim channel As Long
Dim volt1 As Single
Dim volt2 As Single
Dim mode As Long
result = YdxAiGetStartOutRangeVolt(id, channel, volt1, volt2, mode)
```

C++/CLI

```
int result;
int channel;
float volt1;
float volt2;
int mode;
result = YdxAiGetStartOutRangeVolt(id, &channel, &volt1, &volt2, &mode);
```

C/C++

```
INT result;
INT channel;
float volt1;
float volt2;
INT mode;
result = YdxAiGetStartOutRangeVolt(id, &channel, &volt1, &volt2, &mode);
```

YdxAiGetStopCondition

機能

サンプリング停止条件の設定を取得します。

書式

```
INT YdxAiGetStopCondition(
    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	アナログ入力トリガ（レベル比較）
3	アナログ入力トリガ（インレンジ比較）
4	アナログ入力トリガ（アウトレンジ比較）
5	ソフトウェア（YdxAiStop関数）

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*

delay

遅延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、YdxAiSetStopCondition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.AiGetStopCondition(id, out condition, out delay);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAiGetStopSampleNum

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 YdxAiGetStopSampleNum(
    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、YdxAiSetStopSampleNum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.AiGetStopSampleNum(id, out sampleNum);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAiGetStopExternal

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 YdxAiGetStopExternal(
    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、YdxAiSetStopExternal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.AiGetStopExternal(id, out diChannel, out mode);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAiGetStopLevel

機能

サンプリング停止条件（アナログ入力トリガ　レベル比較）の設定を取得します。
しきい値は、バイナリ値で取得します。

書式

```
INT YdxAiGetStopLevel(
    INT id,
    INT* channel,
    INT* level,
    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

比較をする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*

level

しきい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*

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（しきい値未満から、しきい値以上に変化した時に、条件成立）
1	立ち下がりエッジセンス（しきい値を超える値から、しきい値以下に変化した時に、条件成立）
2	両エッジセンス
3	ハイレベルセンス（しきい値以上の時に、条件成立。最初からしきい値以上だった場合も、条件成立）
4	ローレベルセンス（しきい値以下の時に、条件成立。最初からしきい値以下だった場合も、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、YdxAiSetStopLevel関数 を参照してください。

使用例

サンプリング停止条件（アナログ入力トリガ　レベル比較）の設定を取得します。

C#

```
int result;
int channel;
int level;
int mode;
result = Ydx.AiGetStopLevel(id, out channel, out level, out mode);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim channel As Integer
Dim level1 As Integer
Dim mode As Integer
result = YdxAiGetStopLevel(id, channel, level, mode)
```

VB6.0

```
Dim result As Long
Dim channel As Long
Dim level1 As Long
Dim mode As Long
result = YdxAiGetStopLevel(id, channel, level, mode)
```

C++/CLI

```
int result;
int channel;
int level;
int mode;
result = YdxAiGetStopLevel(id, &channel, &level, &mode);
```

C/C++

```
INT result;
INT channel;
INT level;
INT mode;
result = YdxAiGetStopLevel(id, &channel, &level, &mode);
```

YdxAiGetStopLevelVolt

機能

サンプリング停止条件（アナログ入力トリガ　レベル比較）の設定を取得します。
しきい値は、電圧値で取得します。

書式

```
INT YdxAiGetStopLevelVolt(
    INT id,
    INT* channel,
    float* volt,
    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

比較をする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*

volt

しきい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 float	Single	Single	float*	FLOAT*

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（しきい値未満から、しきい値以上に変化した時に、条件成立）
1	立ち下がりエッジセンス（しきい値を超える値から、しきい値以下に変化した時に、条件成立）
2	両エッジセンス
3	ハイレベルセンス（しきい値以上の時に、条件成立。最初からしきい値以上だった場合も、条件成立）
4	ローレベルセンス（しきい値以下の時に、条件成立。最初からしきい値以下だった場合も、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、YdxAiSetStopLevelVolt関数 を参照してください。

使用例

サンプリング停止条件（アナログ入力トリガ　レベル比較）の設定を取得します。

C#

```
int result;
int channel;
float volt;
int mode;
result = Ydx.AiGetStopLevelVolt(id, out channel, out volt, out mode);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim channel As Integer
Dim volt As Single
Dim mode As Integer
result = YdxAiGetStopLevelVolt(id, channel, volt, mode)
```

VB6.0

```
Dim result As Long
Dim channel As Long
Dim volt As Single
Dim mode As Long
result = YdxAiGetStopLevelVolt(id, channel, volt, mode)
```

C++/CLI

```
int result;
int channel;
float volt;
int mode;
result = YdxAiGetStopLevelVolt(id, &channel, &volt, &mode);
```

C/C++

```
INT result;
INT channel;
float volt;
INT mode;
result = YdxAiGetStopLevelVolt(id, &channel, &volt, &mode);
```

YdxAiGetStopInRange

機能

サンプリング停止条件（アナログ入力トリガ　インレンジ比較）の設定を取得します。
しきい値は、バイナリ値で取得します。

書式

```
INT YdxAiGetStopInRange(
    INT id,
    INT* channel,
    INT* level1,
    INT* level2,
    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

比較をする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*

level1・level2

しきい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*

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	エッジセンス（インレンジでない状態から、インレンジに変化した時に、条件成立）
1	レベルセンス（インレンジの時に、条件成立。最初からインレンジだった場合も、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、YdxAiSetStopInRange関数 を参照してください。

使用例

サンプリング停止条件（アナログ入力トリガ　インレンジ比較）の設定を取得します。

C#

```
int result;
int channel;
int level1;
int level2;
int mode;
result = Ydx.AiGetStopInRange(id, out channel, out level1, out level2, out mode);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim channel As Integer
Dim level1 As Integer
Dim level2 As Integer
```

```
Dim mode As Integer
result = YdxAiGetStopInRange(id, channel, level1, level2, mode)
```

VB6.0

```
Dim result As Long
Dim channel As Long
Dim level1 As Long
Dim level2 As Long
Dim mode As Long
result = YdxAiGetStopInRange(id, channel, level1, level2, mode)
```

C++/CLI

```
int result;
int channel;
int level1;
int level2;
int mode;
result = YdxAiGetStopInRange(id, &channel, &level1, &level2, &mode);
```

C/C++

```
INT result;
INT channel;
INT level1;
INT level2;
INT mode;
result = YdxAiGetStopInRange(id, &channel, &level1, &level2, &mode);
```

YdxAiGetStopInRangeVolt

機能

サンプリング停止条件（アナログ入力トリガ　インレンジ比較）の設定を取得します。
しきい値は、電圧値で取得します。

書式

```
INT YdxAiGetStopInRangeVolt(
    INT id,
    INT* channel,
    float* volt1,
    float* volt2,
    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

比較をする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*

volt1・volt2

しきい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 float	Single	Single	float*	FLOAT*

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	エッジセンス（インレンジでない状態から、インレンジに変化した時に、条件成立）
1	レベルセンス（インレンジの時に、条件成立。最初からインレンジだった場合も、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、YdxAiSetStopInRangeVolt関数 を参照してください。

使用例

サンプリング停止条件（アナログ入力トリガ　インレンジ比較）の設定を取得します。

C#

```
int result;
int channel;
float volt1;
float volt2;
int mode;
result = Ydx.AiGetStopInRangeVolt(id, out channel, out volt1, out volt2, out mode);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim channel As Integer
Dim volt1 As Single
Dim volt2 As Single
```

```
Dim mode As Integer
result = YdxAiGetStopInRangeVolt(id, channel, volt1, volt2, mode)
```

VB6.0

```
Dim result As Long
Dim channel As Long
Dim volt1 As Single
Dim volt2 As Single
Dim mode As Long
result = YdxAiGetStopInRangeVolt(id, channel, volt1, volt2, mode)
```

C++/CLI

```
int result;
int channel;
float volt1;
float volt2;
int mode;
result = YdxAiGetStopInRangeVolt(id, &channel, &volt1, &volt2, &mode);
```

C/C++

```
INT result;
INT channel;
float volt1;
float volt2;
INT mode;
result = YdxAiGetStopInRangeVolt(id, &channel, &volt1, &volt2, &mode);
```

YdxAiGetStopOutRange

機能

サンプリング停止条件（アナログ入力トリガ　アウトレンジ比較）の設定を取得します。
しきい値は、バイナリ値で取得します。

書式

```
INT YdxAiGetStopOutRange(
    INT id,
    INT* channel,
    INT* level1,
    INT* level2,
    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

比較をする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*

level1・level2

しきい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*

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	エッジセンス（アウトレンジでない状態から、アウトレンジに変化した時に、条件成立）
1	レベルセンス（アウトレンジの時に、条件成立。最初からアウトレンジだった場合も、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、YdxAiSetStopOutRange関数 を参照してください。

使用例

サンプリング停止条件（アナログ入力トリガ　アウトレンジ比較）の設定を取得します。

C#

```
int result;
int channel;
int level1;
int level2;
int mode;
result = Ydx.AiGetStopOutRange(id, out channel, out level1, out level2, out mode);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim channel As Integer
Dim level1 As Integer
Dim level2 As Integer
```

```
Dim mode As Integer
result = YdxAiGetStopOutRange(id, channel, level1, level2, mode)
```

VB6.0

```
Dim result As Long
Dim channel As Long
Dim level1 As Long
Dim level2 As Long
Dim mode As Long
result = YdxAiGetStopOutRange(id, channel, level1, level2, mode)
```

C++/CLI

```
int result;
int channel;
int level1;
int level2;
int mode;
result = YdxAiGetStopOutRange(id, &channel, &level1, &level2, &mode);
```

C/C++

```
INT result;
INT channel;
INT level1;
INT level2;
INT mode;
result = YdxAiGetStopOutRange(id, &channel, &level1, &level2, &mode);
```

YdxAiGetStopOutRangeVolt

機能

サンプリング停止条件（アナログ入力トリガ　アウトレンジ比較）の設定を取得します。
しきい値は、電圧値で取得します。

書式

```
INT YdxAiGetStopOutRangeVolt(
    INT id,
    INT* channel,
    float* volt1,
    float* volt2,
    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

比較をする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*

volt1・volt2

しきい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 float	Single	Single	float*	FLOAT*

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	エッジセンス（アウトレンジでない状態から、アウトレンジに変化した時に、条件成立）
1	レベルセンス（アウトレンジの時に、条件成立。最初からアウトレンジだった場合も、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、YdxAiSetStopOutRangeVolt関数 を参照してください。

使用例

サンプリング停止条件（アナログ入力トリガ　アウトレンジ比較）の設定を取得します。

C#

```
int result;
int channel;
float volt1;
float volt2;
int mode;
result = Ydx.AiGetStopOutRangeVolt(id, out channel, out volt1, out volt2, out mode);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim channel As Integer
Dim volt1 As Single
Dim volt2 As Single
```

```
Dim mode As Integer
result = YdxAiGetStopOutRangeVolt(id, channel, volt1, volt2, mode)
```

VB6.0

```
Dim result As Long
Dim channel As Long
Dim volt1 As Single
Dim volt2 As Single
Dim mode As Long
result = YdxAiGetStopOutRangeVolt(id, channel, volt1, volt2, mode)
```

C++/CLI

```
int result;
int channel;
float volt1;
float volt2;
int mode;
result = YdxAiGetStopOutRangeVolt(id, &channel, &volt1, &volt2, &mode);
```

C/C++

```
INT result;
INT channel;
float volt1;
float volt2;
INT mode;
result = YdxAiGetStopOutRangeVolt(id, &channel, &volt1, &volt2, &mode);
```

YdxAiStart

機能

アナログ入力動作 を開始します。

書式

```
INT YdxAiStart(
    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

⚠ データバッファにデータが残った状態のまま、YdxAiSetChannel関数 により設定が変更された場合、本関数実行時にデータはクリアされます。
データバッファにデータが残った状態のまま、YdxAiSetRange関数 により設定が変更された場合、本関数実行時にデータはクリアされます。

本関数は、アナログ入力が 動作中 には実行できません。

使用例

アナログ入力動作を開始します。

C#

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

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAiStop

機能

アナログ入力動作 を停止します。

書式

```
INT YdxAiStop(
    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

アナログ入力動作を停止します。

C#

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

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAiReset

機能

アナログ入力機能をリセットします。

書式

```
INT YdxAiReset(
    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。

- アナログ入力が 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

アナログ入力機能をリセットします。

C#

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

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAiGetStatus

機能

現在の状態を取得します。

書式

```
INT YdxAiGetStatus(
    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（YdxAiStart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、YdxAiSetCheckSampleNum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 YdxAiGetData関数 または YdxAiGetDataVolt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.AiGetStatus(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 = YdxAiGetStatus(id, status, sampleCount, repeatCount)
```

VB6.0

```
Dim result As Long
Dim status As Long
Dim sampleCount1 As Long
Dim repeatCount As Long
result = YdxAiGetStatus(id, status, sampleCount, repeatCount)
```

C++/CLI

```
int result;
int status;
int sampleCount;
int repeatCount;
result = YdxAiGetStatus(id, &status, &sampleCount, &repeatCount);
```

C/C++

```
INT result;
INT status;
INT sampleCount;
INT repeatCount;
result = YdxAiGetStatus(id, &status, &sampleCount, &repeatCount);
```

YdxAiGetEventStatus

機能

イベント発生要因を取得します。

書式

```
INT YdxAiGetEventStatus(
    INT id,
    INT* factor,
    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

factor

イベント発生要因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	イベント要因
00000001h	YDX_EVENT_STOP	動作終了 　動作が終了するとオンになります。
00000002h	YDX_EVENT_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るとオンになります。 　監視サンプル数は、YdxAiSetCheckSampleNum関数 で変更できます。
00000004h	YDX_EVENT_START_TRIG	開始条件 成立 　開始条件が成立するとオンになります。
00000008h	YDX_EVENT_STOP_TRIG	停止条件 成立 　停止条件が成立するとオンになります。

値	定義名	イベント要因
00010000h	YDX_EVENT_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るとオンになります。 　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_EVENT_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るとオンになります。 　定期的に YdxAiGetData関数 または YdxAiGetDataVolt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_EVENT_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されるとオンになります。 　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_EVENT_COMMUNICATE_ERR	通信エラー発生 　USB通信に異常が検出されるとオンになります。 　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

イベント発生時の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

イベント発生時の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

イベント発生要因を読み出します。

C#

```
int result;
int factor;
int sampleCount;
int repeatCount;
result = Ydx.AiGetEventStatus(id, out factor, out sampleCount, out repeatCount);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim factor As Integer
Dim sampleCount As Integer
Dim repeatCount As Integer
result = YdxAiGetEventStatus(id, factor, sampleCount, repeatCount)
```

VB6.0

```
Dim result As Long
Dim factor As Long
Dim sampleCount As Long
Dim repeatCount As Long
result = YdxAiGetEventStatus(id, factor, sampleCount, repeatCount)
```

C++/CLI

```
int result;
int factor;
int sampleCount;
```

```
int repeatCount;
result = YdxAiGetEventStatus(id, &factor, &sampleCount, &repeatCount);
```

C/C++

```
INT result;
INT factor;
INT sampleCount;
INT repeatCount;
result = YdxAiGetEventStatus(id, &factor, &sampleCount, &repeatCount);
```

参考

- AiEvent

  高機能アナログ入力のサンプルプログラムです。
  動作状態の監視をイベントでおこなっています。

YdxAiGetData

機能

データをバイナリ値で取得します。

書式

```
INT YdxAiGetData(
    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。
データ数ではなく、サンプル数で指定してください。
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。
（sampleNum * 有効なチャネル 数）個分の配列を用意してください。

（例）サンプル数=1000、有効チャネル=CH0・CH1・CH4の場合、データは以下の順番で格納されます。

データ	CH0	CH1	CH4	CH0	CH1	CH4	・・・	CH0	CH1	CH4	CH0	CH1	CH4
サンプル数	1	1	1	2	2	2	・・・	999	999	999	1000	1000	1000
データ数	1	2	3	4	5	6	・・・	2995	2996	2997	2998	2999	3000

データの値の範囲は、-30000～30000です。
電圧値への換算式は以下のとおりです。

- 電圧値 = data / 30000 * 10（±10Vレンジの場合）
- 電圧値 = data / 30000 * 5（±5Vレンジの場合）

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。
- アナログ入力が 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）
- アナログ入力が 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。
（有効なチャネル数は、6チャネルの場合）

C#

```
int result;
int sampleNum = 1000;
int[] data = new int[6000];
result = Ydx.AiGetData(id, ref sampleNum, data);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim sampleNum As Integer = 1000
Dim data(5999) As Integer
result = YdxAiGetData(id, sampleNum, data)
```

VB6.0

```
Dim result As Long
Dim sampleNum As Long
Dim data(5999) As Long
sampleNum = 1000
result = YdxAiGetData(id, sampleNum, data(0))
```

C++/CLI

```
int result;
int sampleNum = 1000;
int data[6000];
result = YdxAiGetData(id, &sampleNum, data);
```

C/C++

```
INT result;
INT sampleNum = 1000;
INT data[6000];
result = YdxAiGetData(id, &sampleNum, data);
```

YdxAiGetDataVolt

機能

データを電圧値で取得します。

書式

```
INT YdxAiGetDataVolt(
    INT id,
    INT* sampleNum,
    FLOAT* volt
);
```

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。
データ数ではなく、サンプル数で指定してください。
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*

volt

データを格納する変数へのポインタを指定します。
（sampleNum * 有効なチャネル 数）個分の配列を用意してください。
（例）サンプル数=1000、有効チャネル=CH0・CH1・CH4の場合、データは以下の順番で格納されます。

データ	CH0	CH1	CH4	CH0	CH1	CH4	・・・	CH0	CH1	CH4	CH0	CH1	CH4
サンプル数	1	1	1	2	2	2	・・・	999	999	999	1000	1000	1000
データ数	1	2	3	4	5	6	・・・	2995	2996	2997	2998	2999	3000

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	float[]	Single	Single	float*	FLOAT*

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バッファから破棄されます。
- アナログ入力が 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バッファから破棄されません。
  （再度読み出す事が可能）
- アナログ入力が 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。
（有効なチャネル数は、6チャネルの場合）

C#

```
int result;
int sampleNum = 1000;
float[] volt = new float[6000];
result = Ydx.AiGetDataVolt(id, ref sampleNum, volt);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim sampleNum As Integer = 1000
Dim volt(5999) As Single
result = YdxAiGetDataVolt(id, sampleNum, volt)
```

VB6.0

```
Dim result As Long
Dim sampleNum As Long
Dim volt(5999) As Single
sampleNum = 1000
result = YdxAiGetDataVolt(id, sampleNum, volt(0))
```

C++/CLI

```
int result;
int sampleNum = 1000;
float volt[6000];
result = YdxAiGetDataVolt(id, &sampleNum, volt);
```

C/C++

```
INT result;
INT sampleNum = 1000;
float volt[6000];
result = YdxAiGetDataVolt(id, &sampleNum, volt);
```

YdxAiClearData

機能

データをクリアします。

書式

```
INT YdxAiClearData(
    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、アナログ入力が 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.AiClearData(id);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

アナログ出力関数一覧

簡易アナログ出力　専用関数

関数	機能
YdxAoOutput	アナログ出力端子を制御します。 データは、バイナリ値で指定します。
YdxAoOutputVolt	アナログ出力端子を制御します。 データは、電圧値で指定します。

高機能アナログ出力　専用関数

設定

関数	機能
YdxAoSetBuffer	データバッファ の形式を設定します。
YdxAoSetChannel	チャネルの有効（サンプリングをおこなう）／無効（サンプリングをおこなわない）を設定します。
YdxAoSetCheckSampleNum	監視サンプル数 を設定します。
YdxAoSetClock	サンプリングクロック の種類を設定します。
YdxAoSetClockInternal	内部クロックを設定します。
YdxAoSetClockExternal	外部クロックを設定します。
YdxAoSetEvent	イベントを設定します。
YdxAoSetRepeat	リピート を設定します。
YdxAoSetStartCondition	サンプリング開始条件 を設定します。
YdxAoSetStartExternal	サンプリング開始条件（外部トリガ）を設定します。
YdxAoSetStartLevel	サンプリング開始条件（アナログ入力トリガ　レベル比較）を設定します。 しきい値は、バイナリ値で指定します。
YdxAoSetStartLevelVolt	サンプリング開始条件（アナログ入力トリガ　レベル比較）を設定します。 しきい値は、電圧値で指定します。
YdxAoSetStartInRange	サンプリング開始条件（アナログ入力トリガ　インレンジ比較）を設定します。 しきい値は、バイナリ値で指定します。

関数	機能
YdxAoSetStartInRangeVolt	サンプリング開始条件（アナログ入力トリガ　インレンジ比較）を設定します。 しきい値は、電圧値で指定します。
YdxAoSetStartOutRange	サンプリング開始条件（アナログ入力トリガ　アウトレンジ比較）を設定します。 しきい値は、バイナリ値で指定します。
YdxAoSetStartOutRangeVolt	サンプリング開始条件（アナログ入力トリガ　アウトレンジ比較）を設定します。 しきい値は、電圧値で指定します。
YdxAoSetStopCondition	サンプリング停止条件 を設定します。
YdxAoSetStopExternal	サンプリング停止条件（外部トリガ）を設定します。
YdxAoSetStopLevel	サンプリング停止条件（アナログ入力トリガ　レベル比較）を設定します。 しきい値は、バイナリ値で指定します。
YdxAoSetStopLevelVolt	サンプリング停止条件（アナログ入力トリガ　レベル比較）を設定します。 しきい値は、電圧値で指定します。
YdxAoSetStopInRange	サンプリング停止条件（アナログ入力トリガ　インレンジ比較）を設定します。 しきい値は、バイナリ値で指定します。
YdxAoSetStopInRangeVolt	サンプリング停止条件（アナログ入力トリガ　インレンジ比較）を設定します。 しきい値は、電圧値で指定します。
YdxAoSetStopOutRange	サンプリング停止条件（アナログ入力トリガ　アウトレンジ比較）を設定します。 しきい値は、バイナリ値で指定します。
YdxAoSetStopOutRangeVolt	サンプリング停止条件（アナログ入力トリガ　アウトレンジ比較）を設定します。 しきい値は、電圧値で指定します。

設定の取得

関数	機能
YdxAoGetBuffer	データバッファの設定を取得します。
YdxAoGetChannel	チャネルの有効（サンプリングをおこなう）／無効（サンプリングをおこなわない）の設定を取得します。
YdxAoGetCheckSampleNum	監視サンプル数の設定を取得します。
YdxAoGetClock	サンプリングクロックの設定を取得します。
YdxAoGetClockInternal	内部クロックの設定を取得します。
YdxAoGetClockExternal	外部クロックの設定を取得します。
YdxAoGetEvent	イベントの設定を取得します。

関数	機能
YdxAoGetRepeat	リピートの設定を取得します。
YdxAoGetSampleNum	データ設定済みのサンプル数を取得します。
YdxAoGetStartCondition	サンプリング開始条件の設定を取得します。
YdxAoGetStartExternal	サンプリング開始条件（外部トリガ）の設定を取得します。
YdxAoGetStartLevel	サンプリング開始条件（アナログ入力トリガ　レベル比較）の設定を取得します。 しきい値は、バイナリ値で取得します。
YdxAoGetStartLevelVolt	サンプリング開始条件（アナログ入力トリガ　レベル比較）の設定を取得します。 しきい値は、電圧値で取得します。
YdxAoGetStartInRange	サンプリング開始条件（アナログ入力トリガ　インレンジ比較）の設定を取得します。 しきい値は、バイナリ値で取得します。
YdxAoGetStartInRangeVolt	サンプリング開始条件（アナログ入力トリガ　インレンジ比較）の設定を取得します。 しきい値は、電圧値で取得します。
YdxAoGetStartOutRange	サンプリング開始条件（アナログ入力トリガ　アウトレンジ比較）の設定を取得します。 しきい値は、バイナリ値で取得します。
YdxAoGetStartOutRangeVolt	サンプリング開始条件（アナログ入力トリガ　アウトレンジ比較）の設定を取得します。 しきい値は、電圧値で取得します。
YdxAoGetStopCondition	サンプリング停止条件の設定を取得します。
YdxAoGetStopExternal	サンプリング停止条件（外部トリガ）の設定を取得します。
YdxAoGetStopLevel	サンプリング停止条件（アナログ入力トリガ　レベル比較）の設定を取得します。 しきい値は、バイナリ値で取得します。
YdxAoGetStopLevelVolt	サンプリング停止条件（アナログ入力トリガ　レベル比較）の設定を取得します。 しきい値は、電圧値で取得します。
YdxAoGetStopInRange	サンプリング停止条件（アナログ入力トリガ　インレンジ比較）の設定を取得します。 しきい値は、バイナリ値で取得します。
YdxAoGetStopInRangeVolt	サンプリング停止条件（アナログ入力トリガ　インレンジ比較）の設定を取得します。 しきい値は、電圧値で取得します。
YdxAoGetStopOutRange	サンプリング停止条件（アナログ入力トリガ　アウトレンジ比較）の設定を取得します。 しきい値は、バイナリ値で取得します。
YdxAoGetStopOutRangeVolt	サンプリング停止条件（アナログ入力トリガ　アウトレンジ比較）の設定を取得します。 しきい値は、電圧値で取得します。

データの設定・クリア

関数	機能
YdxAoSetData	データをバイナリ値で設定します。
YdxAoSetDataVolt	データを電圧値で設定します。
YdxAoClearData	データをクリアします。

動作の開始・停止

関数	機能
YdxAoStart	アナログ出力動作 を開始します。
YdxAoStop	アナログ出力動作 を停止します。

リセット

関数	機能
YdxAoReset	アナログ出力機能をリセットします。

状態の取得

関数	機能
YdxAoGetStatus	現在の状態を取得します。
YdxAoGetEventStatus	イベント発生要因を取得します。

YdxAoOutput

機能

アナログ出力端子を制御します。
データは、バイナリ値で指定します。

書式

```
INT YdxAoOutput(
    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

出力開始チャネルを指定します。
設定範囲は0～3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

mode

動作モードを指定します。

値	意味
0	出力の更新を、すぐにおこないます。
1	出力の更新を、次のサンプリング周期でおこないます。 （アナログ出力のサンプリングが停止中・待機中は、出力は更新されません） ただし、出力が更新されるのは、チャネル設定 で無効になっているチャネルのみです。 （有効になっているチャネルは、データバッファからのデータで更新されます） アナログ出力が 動作中 に実行する場合は、1を指定してください。

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

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。
データの値の範囲は、-32768～32767です。
電圧値からの換算式は以下のとおりです。
data = 電圧値 * 32767 / 10

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、modeに0を指定する場合は、アナログ出力が 動作中 には実行できません。
（modeに1を指定する場合は、アナログ出力が動作中でも実行できます）

使用例

アナログ出力チャネル0～3（AOUT0～AOUT3）に、15000を出力します。

C#

```
int result;
int[] data = new int[4];
int i;
for (i = 0; i < 4; i++)
{
    data[i] = 15000;
}
result = Ydx.AoOutput(id, 0, 4, 0, data);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim data(3) As Integer
Dim i As Integer
For i = 0 To 3
    data(i) = 15000
Next
result = YdxAoOutput(id, 0, 4, 0, data)
```

VB6.0

```
Dim result As Long
Dim data(3) As Long
For i = 0 To 3
    data(i) = 15000
Next
result = YdxAoOutput(id, 0, 4, 0, data(0))
```

C++/CLI

```
int result;
int data[4];
int i;
for (i = 0; i < 4; i++)
{
    data[i] = 15000;
}
result = YdxAoOutput(id, 0, 4, 0, data);
```

C/C++

```
INT result;
INT data[4];
INT i;
for (i = 0; i < 4; i++)
{
    data[i] = 15000;
}
result = YdxAoOutput(id, 0, 4, 0, data);
```

YdxAoOutputVolt

機能

アナログ出力端子を制御します。
データは、電圧値で指定します。

書式

```
INT YdxAoOutputVolt(
    INT id,
    INT start,
    INT num,
    INT mode,
    FLOAT* volt
);
```

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0～3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

mode

動作モードを指定します。
　0 ：出力の更新を、すぐにおこないます。
　1 ：出力の更新を、次のサンプリング周期でおこないます。
　　（アナログ出力のサンプリングが停止中・待機中は、出力は更新されません）
　　ただし、出力が更新されるのは、チャネル設定 で無効になっているチャネルのみです。
　　（有効になっているチャネルは、データバッファからのデータで更新されます）
　　アナログ出力が 動作中 に実行する場合は、1を指定してください。

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

volt

出力データを格納した変数へのポインタを指定します。
データの値の範囲は、-10～10 [V]です。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	float[]	Single	Single	float*	FLOAT*

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、modeに0を指定する場合は、アナログ出力が 動作中 には実行できません。
　（modeに1を指定する場合は、アナログ出力が動作中でも実行できます）

使用例

アナログ出力チャネル0～3（AOUT0～AOUT3）に、5.6Vを出力します。

C#

```
int result;
float[] volt = new float[4];
int i;
for (i = 0; i < 4; i++)
{
    data[i] = 5.6F;
}
result = Ydx.AoOutputVolt(id, 0, 4, 0, volt);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim volt(3) As Single
Dim i As Integer
For i = 0 To 3
    data(i) = 5.6
Next
result = YdxAoOutputVolt(id, 0, 4, 0, volt)
```

VB6.0

```
Dim result As Long
Dim volt(3) As Single
For i = 0 To 3
    data(i) = 5.6
Next
result = YdxAoOutputVolt(id, 0, 4, 0, volt(0))
```

C++/CLI

```
int result;
float volt[4];
int i;
for (i = 0; i < 4; i++)
{
    volt[i] = 5.6;
}
result = YdxAoOutputVolt(id, 0, 4, 0, volt);
```

C/C++

```
INT result;
float volt[4];
INT i;
for (i = 0; i < 4; i++)
{
    volt[i] = 5.6;
}
result = YdxAoOutputVolt(id, 0, 4, 0, volt);
```

YdxAoSetBuffer

機能

データバッファ の形式を設定します。

書式

```
INT YdxAoSetBuffer(
    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、アナログ出力が 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.AoSetBuffer(id, 1);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAoSetChannel

機能

チャネルの有効（サンプリングをおこなう）／無効（サンプリングをおこなわない）を設定します。

高機能アナログ出力時、有効に設定されたチャネルのみサンプリングをおこないます。
（無効に設定されたチャネルはサンプリングをおこないません）

書式

```
INT YdxAoSetChannel(
    INT id,
    INT channel,
    INT enable
);
```

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0～3です。

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

enable

有効/無効を指定します。

値	意味
0	無効（サンプリングをおこなわない）
1	有効（サンプリングをおこなう）

初期値はチャネル0は1・その他のチャネルは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、アナログ出力が 動作中 には実行できません。

使用例

チャネル2（AOUT2）を、有効に設定します。

C#

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

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAoSetCheckSampleNum

機能

監視サンプル数 を設定します。

書式

```
INT YdxAoSetCheckSampleNum(
    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、以下の動作となります。
・ YdxAoGetStatus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。
・ YdxAoSetEvent関数 で、監視サンプル数イベントを有効に設定してある場合、イベントが発生します。

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++

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、アナログ出力が 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.AoSetCheckSampleNum(id, 2000);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAoSetClock

機能

サンプリングクロック の種類を設定します。

書式

```
INT YdxAoSetClock(
    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、YdxAoSetClockInternal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、YdxAoSetClockExternal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、アナログ出力が 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.AoSetClock(id, 1);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAoSetClockInternal

機能

内部クロックを設定します。

書式

```
INT YdxAoSetClockInternal(
    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6.6～60,000,000 [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

YdxAoSetClock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。

本関数は、アナログ出力が 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.AoSetClockInternal(id, 2000);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAoSetClockExternal

機能

外部クロックを設定します。

書式

```
INT YdxAoSetClockExternal(
    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0～3、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

YdxAoSetClock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、アナログ出力が 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.AoSetClockExternal(id, 2, 1);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAoSetEvent

機能

イベントを設定します。

書式

```
INT YdxAoSetEvent(
    INT id,
    INT* mask,
    HANDLE* hEvent
);
```

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

mask

イベント要因を指定します。
指定した要因となった時に、イベントを発生させる事ができます。

値	定義名	イベント要因
00000001h	YDX_EVENT_STOP	動作終了 作が終了するとイベントを発生させます。
00000002h	YDX_EVENT_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イベントを発生させます。 視サンプル数は、YdxAoSetCheckSampleNum関数 で変更できます。
00000004h	YDX_EVENT_START_TRIG	開始条件成立 始条件が成立するとイベントを発生させます。 リピートの場合、開始条件が成立するたびにイベントを発生させます。

値	定義名	イベント要因
00000008h	YDX_EVENT_STOP_TRIG	停止条件成立 止条件が成立するとイベントを発生させます。 リピートの場合、停止条件が成立するたびにイベントを発生させます。
00010000h	YDX_EVENT_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るとイベントを発生させます。
00040000h	YDX_EVENT_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されるとイベントを発生させます。
00080000h	YDX_EVENT_COMMUNICATE_ERR	通信エラー発生 USB通信に異常が検出されるとイベントを発生させます。

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

hEvent

イベントオブジェクトのハンドルを指定します。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	IntPtr	IntPtr	Long	IntPtr	HANDLE

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、アナログ出力が 動作中 には実行できません。

使用例

イベントを設定します。
イベント要因を「監視サンプル数」と「動作終了」に設定します。

C#

```
int result;
AutoResetEvent hEvent = new AutoResetEvent(false);
result = Ydx.AoSetEvent(id, YDX_EVENT_SAMPLE_NUM| YDX_EVENT_STOP, hEvent);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim hEvent As AutoResetEvent = New AutoResetEvent(False)
result = YdxAoSetEvent(id, YDX_EVENT_SAMPLE_NUM| YDX_EVENT_STOP, hEvent)
```

VB6.0

```
Dim result As Long
Dim hEvent As Long
hEvent = CreateEvent(0, False, False, 0)
result = YdxAoSetEvent(id, YDX_EVENT_SAMPLE_NUM| YDX_EVENT_STOP, hEvent)
```

C++/CLI

```
int result;
AutoResetEvent^ hEvent = gcnew AutoResetEvent(false);
result = YdxAoSetEvent(id, YDX_EVENT_SAMPLE_NUM| YDX_EVENT_STOP, hEvent->Handle);
```

C/C++

```
INT result;
HANDLE hEvent = CreateEvent(NULL, FALSE, FALSE, NULL);
result = YdxAoSetEvent(id, YDX_EVENT_SAMPLE_NUM| YDX_EVENT_STOP, hEvent);
```

参考

- AoEvent

  高機能アナログ出力のサンプルプログラムです。
  動作状態の監視をイベントでおこなっています。

YdxAoSetRepeat

機能

リピート を設定します。

書式

```
INT YdxAoSetRepeat(
    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（YdxAoSetStartCondition関数 で設定）からサンプリング停止条件（YdxAoSetStopCondition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、アナログ出力が 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.AoSetRepeat(id, 10);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAoSetStartCondition

機能

サンプリング開始条件 を設定します。

書式

```
INT YdxAoSetStartCondition(
    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
2	アナログ入力トリガ（レベル比較）
3	アナログ入力トリガ（インレンジ比較）
4	アナログ入力トリガ（アウトレンジ比較）

初期値は0です。

「外部トリガ」を指定する場合、YdxAoSetStartExternal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。
「アナログ入力トリガ（レベル比較）」を指定する場合、YdxAoSetStartLevel関数 （または

YdxAoSetStartLevelVolt関数 ）で、使用するアナログ入力チャネルとモードの設定をしてください。
「アナログ入力トリガ（インレンジ比較）」を指定する場合、YdxAoSetStartInRange関数 （または YdxAoSetStartInRangeVolt関数 ）で、使用するアナログ入力チャネルとモードの設定をしてください。
「アナログ入力トリガ（アウトレンジ比較）」を指定する場合、YdxAoSetStartOutRange関数 （または YdxAoSetStartOutRangeVolt関数 ）で、使用するアナログ入力チャネルとモードの設定をしてください。
「アナログ入力トリガ（レベル比較）」「アナログ入力トリガ（インレンジ比較）」「アナログ入力トリガ（アウトレンジ比較）」を指定した場合、比較判定をさせるには、アナログ入力も動作させる必要があります。

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、アナログ出力が 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.AoSetStartCondition(id, 1, 0);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAoSetStartExternal

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 YdxAoSetStartExternal(
    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0～3、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

YdxAoSetStartCondition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、アナログ出力が 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.AoSetStartExternal(id, 1, 2);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAoSetStartLevel

機能

サンプリング開始条件（アナログ入力トリガ　レベル比較）を設定します。
しきい値は、バイナリ値で指定します。

書式

```
INT YdxAoSetStartLevel(
    INT id,
    INT aiChannel,
    INT level,
    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

aiChannel

比較をするアナログ入力チャネルを指定します。
設定範囲は0～5、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

level

しきい値を指定します。
設定範囲は-30,000～30,000、初期値は15,000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（しきい値未満から、しきい値以上に変化した時に、条件成立）
1	立ち下がりエッジセンス（しきい値を超える値から、しきい値以下に変化した時に、条件成立）
2	両エッジセンス
3	ハイレベルセンス（しきい値以上の時に、条件成立。最初からしきい値以上だった場合も、条件成立）
4	ローレベルセンス（しきい値以下の時に、条件成立。最初からしきい値以下だった場合も、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

YdxAoSetStartCondition関数 で、サンプリング開始条件として「アナログ入力トリガ（レベル比較）」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「アナログ入力トリガ（レベル比較）」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、アナログ出力が 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（アナログ入力トリガ　レベル比較）を設定します。
比較をするチャネルは1、しきい値は23,000、動作モードはローレベルセンスに設定します。

C#

```
int result;
result = Ydx.AoSetStartLevel(id, 1, 23000, 4);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxAoSetStartLevel(id, 1, 23000, 4)
```

VB6.0

```
Dim result As Long
result = YdxAoSetStartLevel(id, 1, 23000, 4)
```

C++/CLI

```
int result;
result = YdxAoSetStartLevel(id, 1, 23000, 4);
```

C/C++

```
INT result;
result = YdxAoSetStartLevel(id, 1, 23000, 4);
```

YdxAoSetStartLevelVolt

機能

サンプリング開始条件（アナログ入力トリガ　レベル比較）を設定します。
しきい値は、電圧値で指定します。

書式

```
INT YdxAoSetStartLevelVolt(
    INT id,
    INT aiChannel,
    float volt,
    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

aiChannel

比較をするアナログ入力チャネルを指定します。
設定範囲は0～5、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

volt

しきい値を指定します。
単位は「V」です。
設定範囲は-10～10 [V]、初期値は5 [V]です。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	float	Single	Single	float	FLOAT

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（しきい値未満から、しきい値以上に変化した時に、条件成立）
1	立ち下がりエッジセンス（しきい値を超える値から、しきい値以下に変化した時に、条件成立）
2	両エッジセンス
3	ハイレベルセンス（しきい値以上の時に、条件成立。最初からしきい値以上だった場合も、条件成立）
4	ローレベルセンス（しきい値以下の時に、条件成立。最初からしきい値以下だった場合も、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

YdxAoSetStartCondition関数 で、サンプリング開始条件として「アナログ入力トリガ（レベル比較）」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「アナログ入力トリガ（レベル比較）」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。

> ⚠ しきい値は、バイナリ値に換算されて、内部に記憶されます。
> 本関数の実行後に 入力レンジが変更 されても再換算はおこなわれませんので、入力レンジ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、アナログ出力が 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（アナログ入力トリガ　レベル比較）を設定します。
比較をするチャネルは1、しきい値は2.3V、動作モードはローレベルセンスに設定します。

C#

```
int result;
result = Ydx.AoSetStartLevelVolt(id, 1, 2.3F, 4);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxAoSetStartLevelVolt(id, 1, 2.3, 4)
```

VB6.0

```
Dim result As Long
result = YdxAoSetStartLevelVolt(id, 1, 2.3, 4)
```

C++/CLI

```
int result;
result = YdxAoSetStartLevelVolt(id, 1, 2.3, 4);
```

C/C++

```
INT result;
result = YdxAoSetStartLevelVolt(id, 1, 2.3, 4);
```

YdxAoSetStartInRange

機能

サンプリング開始条件（アナログ入力トリガ　インレンジ比較）を設定します。
しきい値は、バイナリ値で指定します。

書式

```
INT YdxAoSetStartInRange(
    INT id,
    INT aiChannel,
    INT level1,
    INT level2,
    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

aiChannel

比較をするアナログ入力チャネルを指定します。
設定範囲は0～5、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

level1・level2

しきい値を指定します。
設定範囲は-30,000～30,000、初期値はlevel1=-15,000・level2=15,000です。
「level1≦データ≦level2」または「level2≦データ≦level1」となった時に、条件が成立します。

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	エッジセンス（インレンジでない状態から、インレンジに変化した時に、条件成立）
1	レベルセンス（インレンジの時に、条件成立。最初からインレンジだった場合も、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

YdxAoSetStartCondition関数 で、サンプリング開始条件として「アナログ入力トリガ（インレンジ比較）」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「アナログ入力トリガ（インレンジ比較）」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、アナログ出力が 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（アナログ入力トリガ　インレンジ比較）を設定します。
比較をするチャネルは1、しきい値は-2,300と4,500、動作モードはレベルセンスに設定します。

C#

```
int result;
result = Ydx.AoSetStartInRange(id, 1, -2300, 4500, 1);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxAoSetStartInRange(id, 1, -2300, 4500, 1)
```

VB6.0

```
Dim result As Long
result = YdxAoSetStartInRange(id, 1, -2300, 4500, 1)
```

C++/CLI

```
int result;
result = YdxAoSetStartInRange(id, 1, -2300, 4500, 1);
```

C/C++

```
INT result;
result = YdxAoSetStartInRange(id, 1, -2300, 4500, 1);
```

YdxAoSetStartInRangeVolt

機能

サンプリング開始条件（アナログ入力トリガ　インレンジ比較）を設定します。
しきい値は、電圧値で指定します。

書式

```
INT YdxAoSetStartInRangeVolt(
    INT id,
    INT aiChannel,
    float volt1,
    float volt2,
    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

aiChannel

比較をするアナログ入力チャネルを指定します。
設定範囲は0～5、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

volt1・volt2

しきい値を指定します。
単位は「V」です。
設定範囲は-10～10 [V]、初期値はvolt1=-5 [V]・volt2=5 [V]です。
「volt1≦電圧≦volt2」または「volt2≦電圧≦volt1」となった時に、条件が成立します。

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	float	Single	Single	float	FLOAT

mode

動作モードを指定します。

値	意味
0	エッジセンス（インレンジでない状態から、インレンジに変化した時に、条件成立）
1	レベルセンス（インレンジの時に、条件成立。最初からインレンジだった場合も、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

YdxAoSetStartCondition関数 で、サンプリング開始条件として「アナログ入力トリガ（インレンジ比較）」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「アナログ入力トリガ（インレンジ比較）」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。

> ⚠ しきい値は、バイナリ値に換算されて、内部に記憶されます。
> 本関数の実行後に 入力レンジが変更 されても再換算はおこなわれませんので、入力レンジ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、アナログ出力が 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（アナログ入力トリガ　インレンジ比較）を設定します。
比較をするチャネルは1、しきい値は-2.3Vと4.5V、動作モードはレベルセンスに設定します。

C#

```
int result;
result = Ydx.AoSetStartInRangeVolt(id, 1, -2.3F, 4.5F, 1);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxAoSetStartInRangeVolt(id, 1, -2.3, 4.5, 1)
```

VB6.0

```
Dim result As Long
result = YdxAoSetStartInRangeVolt(id, 1, -2.3, 4.5, 1)
```

C++/CLI

```
int result;
result = YdxAoSetStartInRangeVolt(id, 1, -2.3, 4.5, 1);
```

C/C++

```
INT result;
result = YdxAoSetStartInRangeVolt(id, 1, -2.3, 4.5, 1);
```

YdxAoSetStartOutRange

機能

サンプリング開始条件（アナログ入力トリガ　アウトレンジ比較）を設定します。
しきい値は、バイナリ値で指定します。

書式

```
INT YdxAoSetStartOutRange(
    INT id,
    INT aiChannel,
    INT level1,
    INT level2,
    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

aiChannel

比較をするアナログ入力チャネルを指定します。
設定範囲は0～5、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

level1・level2

しきい値を指定します。
設定範囲は-30,000～30,000、初期値はlevel1=-15,000・level2=15,000です。
「データ≦level1 または level2≦データ」となった時に、条件が成立します。（level1≦level2の場合）「データ≦level2 または level1≦データ」となった時に、条件が成立します。（level2≦level1の場合）

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

mode

動作モードを指定します。

値	意味
0	エッジセンス（アウトレンジでない状態から、アウトレンジに変化した時に、条件成立）
1	レベルセンス（アウトレンジの時に、条件成立。最初からアウトレンジだった場合も、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

YdxAoSetStartCondition関数 で、サンプリング開始条件として「アナログ入力トリガ（アウトレンジ比較）」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「アナログ入力トリガ（アウトレンジ比較）」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、アナログ出力が 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（アナログ入力トリガ　アウトレンジ比較）を設定します。
比較をするチャネルは1、しきい値は-2,300と4,500、動作モードはレベルセンスに設定します。

C#

```
int result;
result = Ydx.AoSetStartOutRange(id, 1, -2300, 4500, 1);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxAoSetStartOutRange(id, 1, -2300, 4500, 1)
```

VB6.0

```
Dim result As Long
result = YdxAoSetStartOutRange(id, 1, -2300, 4500, 1)
```

C++/CLI

```
int result;
result = YdxAoSetStartOutRange(id, 1, -2300, 4500, 1);
```

C/C++

```
INT result;
result = YdxAoSetStartOutRange(id, 1, -2300, 4500, 1);
```

YdxAoSetStartOutRangeVolt

機能

サンプリング開始条件（アナログ入力トリガ　アウトレンジ比較）を設定します。
しきい値は、電圧値で指定します。

書式

```
INT YdxAoSetStartOutRangeVolt(
    INT id,
    INT aiChannel,
    float volt1,
    float volt2,
    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

aiChannel

比較をするアナログ入力チャネルを指定します。
設定範囲は0～5、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

volt1・volt2

しきい値を指定します。
単位は「V」です。
設定範囲は-10～10 [V]、初期値はvolt1=-5 [V]・volt2=5 [V]です。
「電圧≦volt1 または volt2≦電圧」となった時に、条件が成立します。（volt1≦volt2の場合）「電圧≦volt2 または volt1≦電圧」となった時に、条件が成立します。（volt2≦volt1の場合）

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	float	Single	Single	float	FLOAT

mode

動作モードを指定します。

値	意味
0	エッジセンス（アウトレンジでない状態から、アウトレンジに変化した時に、条件成立）
1	レベルセンス（アウトレンジの時に、条件成立。最初からアウトレンジだった場合も、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

YdxAoSetStartCondition関数 で、サンプリング開始条件として「アナログ入力トリガ（アウトレンジ比較）」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「アナログ入力トリガ（アウトレンジ比較）」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。

> ⚠ しきい値は、バイナリ値に換算されて、内部に記憶されます。
> 本関数の実行後に 入力レンジが変更 されても再換算はおこなわれませんので、入力レンジ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、アナログ出力が 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（アナログ入力トリガ　アウトレンジ比較）を設定します。
比較をするチャネルは1、しきい値は-2.3Vと4.5V、動作モードはレベルセンスに設定します。

C#

```
int result;
result = Ydx.AoSetStartOutRangeVolt(id, 1, -2.3F, 4.5F, 1);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxAoSetStartOutRangeVolt(id, 1, -2.3, 4.5, 1)
```

VB6.0

```
Dim result As Long
result = YdxAoSetStartOutRangeVolt(id, 1, -2.3, 4.5, 1)
```

C++/CLI

```
int result;
result = YdxAoSetStartOutRangeVolt(id, 1, -2.3, 4.5, 1);
```

C/C++

```
INT result;
result = YdxAoSetStartOutRangeVolt(id, 1, -2.3, 4.5, 1);
```

YdxAoSetStopCondition

機能

サンプリング停止条件 を設定します。

書式

```
INT YdxAoSetStopCondition(
    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
2	アナログ入力トリガ（レベル比較）
3	アナログ入力トリガ（インレンジ比較）
4	アナログ入力トリガ（アウトレンジ比較）

初期値は0です。

「外部トリガ」を指定する場合、YdxAoSetStopExternal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。
「アナログ入力トリガ（レベル比較）」を指定する場合、YdxAoSetStopLevel関数 （または

YdxAoSetStopLevelVolt関数）で、使用するアナログ入力チャネルとモードの設定をしてください。
「アナログ入力トリガ（インレンジ比較）」を指定する場合、YdxAoSetStopInRange関数（またはYdxAoSetStopInRangeVolt関数）で、使用するアナログ入力チャネルとモードの設定をしてください。
「アナログ入力トリガ（アウトレンジ比較）」を指定する場合、YdxAoSetStopOutRange関数（またはYdxAoSetStopOutRangeVolt関数）で、使用するアナログ入力チャネルとモードの設定をしてください。
「アナログ入力トリガ（レベル比較）」「アナログ入力トリガ（インレンジ比較）」「アナログ入力トリガ（アウトレンジ比較）」を指定した場合、比較判定をさせるには、アナログ入力も動作させる必要があります。

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、アナログ出力が 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.AoSetStopCondition(id, 1, 0);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAoSetStopExternal

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 YdxAoSetStopExternal(
    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0～3、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

YdxAoSetStopCondition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、アナログ出力が 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.AoSetStopExternal(id, 1, 2);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAoSetStopLevel

機能

サンプリング停止条件（アナログ入力トリガ　レベル比較）を設定します。
しきい値は、バイナリ値で指定します。

書式

```
INT YdxAoSetStopLevel(
    INT id,
    INT aiChannel,
    INT level,
    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

aiChannel

比較をするアナログ入力チャネルを指定します。
設定範囲は0～5、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

level

しきい値を指定します。
設定範囲は-30,000～30,000、初期値は15,000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（しきい値未満から、しきい値以上に変化した時に、条件成立）
1	立ち下がりエッジセンス（しきい値を超える値から、しきい値以下に変化した時に、条件成立）
2	両エッジセンス
3	ハイレベルセンス（しきい値以上の時に、条件成立。最初からしきい値以上だった場合も、条件成立）
4	ローレベルセンス（しきい値以下の時に、条件成立。最初からしきい値以下だった場合も、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

YdxAoSetStopCondition関数 で、サンプリング停止条件として「アナログ入力トリガ（レベル比較）」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「アナログ入力トリガ（レベル比較）」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、アナログ出力が 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（アナログ入力トリガ　レベル比較）を設定します。
比較をするチャネルは1、しきい値は23,000、動作モードはローレベルセンスに設定します。

C#

```
int result;
result = Ydx.AoSetStopLevel(id, 1, 23000, 4);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxAoSetStopLevel(id, 1, 23000, 4)
```

VB6.0

```
Dim result As Long
result = YdxAoSetStopLevel(id, 1, 23000, 4)
```

C++/CLI

```
int result;
result = YdxAoSetStopLevel(id, 1, 23000, 4);
```

C/C++

```
INT result;
result = YdxAoSetStopLevel(id, 1, 23000, 4);
```

YdxAoSetStopLevelVolt

機能

サンプリング停止条件（アナログ入力トリガ　レベル比較）を設定します。
しきい値は、電圧値で指定します。

書式

```
INT YdxAoSetStopLevelVolt(
    INT id,
    INT aiChannel,
    float volt,
    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

aiChannel

比較をするアナログ入力チャネルを指定します。
設定範囲は0～5、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

volt

しきい値を指定します。
単位は「V」です。
設定範囲は-10～10 [V]、初期値は5 [V]です。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	float	Single	Single	float	FLOAT

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（しきい値未満から、しきい値以上に変化した時に、条件成立）
1	立ち下がりエッジセンス（しきい値を超える値から、しきい値以下に変化した時に、条件成立）
2	両エッジセンス
3	ハイレベルセンス（しきい値以上の時に、条件成立。最初からしきい値以上だった場合も、条件成立）
4	ローレベルセンス（しきい値以下の時に、条件成立。最初からしきい値以下だった場合も、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

YdxAoSetStopCondition関数 で、サンプリング停止条件として「アナログ入力トリガ（レベル比較）」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「アナログ入力トリガ（レベル比較）」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。

> ⚠ しきい値は、バイナリ値に換算されて、内部に記憶されます。
> 本関数の実行後に 入力レンジが変更 されても再換算はおこなわれませんので、入力レンジ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、アナログ出力が 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（アナログ入力トリガ　レベル比較）を設定します。
比較をするチャネルは1、しきい値は2.3V、動作モードはローレベルセンスに設定します。

C#

```
int result;
result = Ydx.AoSetStopLevelVolt(id, 1, 2.3F, 4);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxAoSetStopLevelVolt(id, 1, 2.3, 4)
```

VB6.0

```
Dim result As Long
result = YdxAoSetStopLevelVolt(id, 1, 2.3, 4)
```

C++/CLI

```
int result;
result = YdxAoSetStopLevelVolt(id, 1, 2.3, 4);
```

C/C++

```
INT result;
result = YdxAoSetStopLevelVolt(id, 1, 2.3, 4);
```

YdxAoSetStopInRange

機能

サンプリング停止条件（アナログ入力トリガ　インレンジ比較）を設定します。
しきい値は、バイナリ値で指定します。

書式

```
INT YdxAoSetStopInRange(
    INT id,
    INT aiChannel,
    INT level1,
    INT level2,
    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

aiChannel

比較をするアナログ入力チャネルを指定します。
設定範囲は0～5、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

level1・level2

しきい値を指定します。
設定範囲は-30,000～30,000、初期値はlevel1=-15,000・level2=15,000です。
「level1≦データ≦level2」または「level2≦データ≦level1」となった時に、条件が成立します。

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	エッジセンス（インレンジでない状態から、インレンジに変化した時に、条件成立）
1	レベルセンス（インレンジの時に、条件成立。最初からインレンジだった場合も、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

YdxAoSetStopCondition関数 で、サンプリング停止条件として「アナログ入力トリガ（インレンジ比較）」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「アナログ入力トリガ（インレンジ比較）」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、アナログ出力が 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（アナログ入力トリガ　インレンジ比較）を設定します。
比較をするチャネルは1、しきい値は-2,300と4,500、動作モードはレベルセンスに設定します。

C#

```
int result;
result = Ydx.AoSetStopInRange(id, 1, -2300, 4500, 1);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxAoSetStopInRange(id, 1, -2300, 4500, 1)
```

VB6.0

```
Dim result As Long
result = YdxAoSetStopInRange(id, 1, -2300, 4500, 1)
```

C++/CLI

```
int result;
result = YdxAoSetStopInRange(id, 1, -2300, 4500, 1);
```

C/C++

```
INT result;
result = YdxAoSetStopInRange(id, 1, -2300, 4500, 1);
```

YdxAoSetStopInRangeVolt

機能

サンプリング停止条件（アナログ入力トリガ　インレンジ比較）を設定します。
しきい値は、電圧値で指定します。

書式

```
INT YdxAoSetStopInRangeVolt(
    INT id,
    INT aiChannel,
    float volt1,
    float volt2,
    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

aiChannel

比較をするアナログ入力チャネルを指定します。
設定範囲は0～5、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

volt1・volt2

しきい値を指定します。
単位は「V」です。
設定範囲は-10～10 [V]、初期値はvolt1=-5 [V]・volt2=5 [V]です。
「volt1≦電圧≦volt2」または「volt2≦電圧≦volt1」となった時に、条件が成立します。

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	float	Single	Single	float	FLOAT

mode

動作モードを指定します。

値	意味
0	エッジセンス（インレンジでない状態から、インレンジに変化した時に、条件成立）
1	レベルセンス（インレンジの時に、条件成立。最初からインレンジだった場合も、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

YdxAoSetStopCondition関数 で、サンプリング停止条件として「アナログ入力トリガ（インレンジ比較）」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「アナログ入力トリガ（インレンジ比較）」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。

> ⚠ しきい値は、バイナリ値に換算されて、内部に記憶されます。
> 本関数の実行後に 入力レンジが変更 されても再換算はおこなわれませんので、入力レンジ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、アナログ出力が 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（アナログ入力トリガ　インレンジ比較）を設定します。
比較をするチャネルは1、しきい値は-2.3Vと4.5V、動作モードはレベルセンスに設定します。

C#

```
int result;
result = Ydx.AoSetStopInRangeVolt(id, 1, -2.3F, 4.5F, 1);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxAoSetStopInRangeVolt(id, 1, -2.3, 4.5, 1)
```

VB6.0

```
Dim result As Long
result = YdxAoSetStopInRangeVolt(id, 1, -2.3, 4.5, 1)
```

C++/CLI

```
int result;
result = YdxAoSetStopInRangeVolt(id, 1, -2.3, 4.5, 1);
```

C/C++

```
INT result;
result = YdxAoSetStopInRangeVolt(id, 1, -2.3, 4.5, 1);
```

YdxAoSetStopOutRange

機能

サンプリング停止条件（アナログ入力トリガ　アウトレンジ比較）を設定します。
しきい値は、バイナリ値で指定します。

書式

```
INT YdxAoSetStopOutRange(
    INT id,
    INT aiChannel,
    INT level1,
    INT level2,
    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

aiChannel

比較をするアナログ入力チャネルを指定します。
設定範囲は0～5、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

level1・level2

しきい値を指定します。
設定範囲は-30,000～30,000、初期値はlevel1=-15,000・level2=15,000です。
「データ≦level1 または level2≦データ」となった時に、条件が成立します。（level1≦level2の場合）「データ≦level2 または level1≦データ」となった時に、条件が成立します。（level2≦level1の場合）

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

mode

動作モードを指定します。

値	意味
0	エッジセンス（アウトレンジでない状態から、アウトレンジに変化した時に、条件成立）
1	レベルセンス（アウトレンジの時に、条件成立。最初からアウトレンジだった場合も、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

YdxAoSetStopCondition関数 で、サンプリング停止条件として「アナログ入力トリガ（アウトレンジ比較）」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「アナログ入力トリガ（アウトレンジ比較）」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、アナログ出力が 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（アナログ入力トリガ　アウトレンジ比較）を設定します。
比較をするチャネルは1、しきい値は-2,300と4,500、動作モードはレベルセンスに設定します。

C#

```
int result;
result = Ydx.AoSetStopOutRange(id, 1, -2300, 4500, 1);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxAoSetStopOutRange(id, 1, -2300, 4500, 1)
```

VB6.0

```
Dim result As Long
result = YdxAoSetStopOutRange(id, 1, -2300, 4500, 1)
```

C++/CLI

```
int result;
result = YdxAoSetStopOutRange(id, 1, -2300, 4500, 1);
```

C/C++

```
INT result;
result = YdxAoSetStopOutRange(id, 1, -2300, 4500, 1);
```

YdxAoSetStopOutRangeVolt

機能

サンプリング停止条件（アナログ入力トリガ　アウトレンジ比較）を設定します。
しきい値は、電圧値で指定します。

書式

```
INT YdxAoSetStopOutRangeVolt(
    INT id,
    INT aiChannel,
    float volt1,
    float volt2,
    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

aiChannel

比較をするアナログ入力チャネルを指定します。
設定範囲は0～5、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

volt1・volt2

しきい値を指定します。
単位は「V」です。
設定範囲は-10～10 [V]、初期値はvolt1=-5 [V]・volt2=5 [V]です。
「電圧≦volt1 または volt2≦電圧」となった時に、条件が成立します。（volt1≦volt2の場合）「電圧≦volt2 または volt1≦電圧」となった時に、条件が成立します。（volt2≦volt1の場合）

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	float	Single	Single	float	FLOAT

mode

動作モードを指定します。

値	意味
0	エッジセンス（アウトレンジでない状態から、アウトレンジに変化した時に、条件成立）
1	レベルセンス（アウトレンジの時に、条件成立。最初からアウトレンジだった場合も、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

YdxAoSetStopCondition関数 で、サンプリング停止条件として「アナログ入力トリガ（アウトレンジ比較）」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「アナログ入力トリガ（アウトレンジ比較）」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。

> ⚠ しきい値は、バイナリ値に換算されて、内部に記憶されます。
> 本関数の実行後に 入力レンジが変更 されても再換算はおこなわれませんので、入力レンジ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、アナログ出力が 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（アナログ入力トリガ　アウトレンジ比較）を設定します。
比較をするチャネルは1、しきい値は-2.3Vと4.5V、動作モードはレベルセンスに設定します。

C#

```
int result;
result = Ydx.AoSetStopOutRangeVolt(id, 1, -2.3F, 4.5F, 1);
```

VB（.NET2002以降）

```
Dim result As Integer
result = YdxAoSetStopOutRangeVolt(id, 1, -2.3, 4.5, 1)
```

VB6.0

```
Dim result As Long
result = YdxAoSetStopOutRangeVolt(id, 1, -2.3, 4.5, 1)
```

C++/CLI

```
int result;
result = YdxAoSetStopOutRangeVolt(id, 1, -2.3, 4.5, 1);
```

C/C++

```
INT result;
result = YdxAoSetStopOutRangeVolt(id, 1, -2.3, 4.5, 1);
```

YdxAoGetBuffer

機能

データバッファの設定を取得します。

書式

```
INT YdxAoGetBuffer(
    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、YdxAoSetBuffer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.AoGetBuffer(id, out bufferType);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAoGetChannel

機能

チャネルの有効（サンプリングをおこなう）／無効（サンプリングをおこなわない）の設定を取得します。

書式

```
INT YdxAoGetChannel(
    INT id,
    INT channel,
    INT* enable
);
```

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0～3です。

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

enable

有効/無効の設定を格納する変数へのポインタを指定します。

値	意味
0	無効（サンプリングをおこなわない）
1	有効（サンプリングをおこなう）

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、YdxAoSetChannel関数 を参照してください。

使用例

チャネル2の、有効/無効の設定を取得します。

C#

```
int result;
int enable;
result = Ydx.AoGetChannel(id, 2, out enable);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim enable As Integer
result = YdxAoGetChannel(id, 2, enable)
```

VB6.0

```
Dim result As Long
Dim enable As Long
result = YdxAoGetChannel(id, 2, enable)
```

C++/CLI

```
int result;
int enable;
result = YdxAoGetChannel(id, 2, &enable);
```

C/C++

```
INT result;
INT enable;
result = YdxAoGetChannel(id, 2, &enable);
```

YdxAoGetCheckSampleNum

機能

監視サンプル数の設定を取得します。

書式

```
INT YdxAoGetCheckSampleNum(
    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、YdxAoSetCheckSampleNum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.AoGetCheckSampleNum(id, out sampleNum);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAoGetClock

機能

サンプリングクロックの設定を取得します。

書式

```
INT YdxAoGetClock(
    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、YdxAoSetClock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.AoGetClock(id, out clockType);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAoGetClockInternal

機能

内部クロックの設定を取得します。

書式

```
INT YdxAoGetClockInternal(
    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、YdxAoSetClockInternal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.AoGetClockInternal(id, out period);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAoGetClockExternal

機能

外部クロックの設定を取得します。

書式

```
INT YdxAoGetClockExternal(
    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、YdxAoSetClockExternal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.AoGetClockExternal(id, out diChannel, out edge);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAoGetEvent

機能

イベントの設定を取得します。

書式

```
INT YdxAoGetEvent(
    INT id,
    INT* mask,
    HANDLE* hEvent
);
```

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

mask

イベント要因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	イベント要因
00000001h	YDX_EVENT_STOP	動作終了
00000002h	YDX_EVENT_SAMPLE_NUM	監視サンプル数
00000004h	YDX_EVENT_START_TRIG	開始条件成立
00000008h	YDX_EVENT_STOP_TRIG	停止条件成立
00010000h	YDX_EVENT_SAMPLE_CLOCK_ERR	サンプリングクロックエラー発生
00040000h	YDX_EVENT_HARDWARE_ERR	ハードウェアエラー発生
00080000h	YDX_EVENT_COMMUNICATE_ERR	通信エラー発生

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

hEvent

イベントオブジェクトのハンド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 IntPtr	IntPtr	Long	IntPtr*	HANDLE*

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、YdxAoSetEvent関数 を参照してください。

使用例

イベントの設定を読み出します。

C#

```
int result;
int mask;
IntPtr hEvent;
result = Ydx.AoGetEvent(id, out mask, out hEvent);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim mask As Integer
Dim hEvent As IntPtr
result = YdxAoGetEvent(id, mask, hEvent)
```

VB6.0

```
Dim result As Long
Dim mask As Long
Dim hEvent As Long
result = YdxAoGetEvent(id, mask, hEvent)
```

C++/CLI

```
int result;
int mask;
IntPtr hEvent;
result = YdxAoGetEvent(id, &mask, &hEvent);
```

C/C++

```
INT result;
INT mask;
HANDLE hEvent;
result = YdxAoGetEvent(id, &mask, &hEvent);
```

YdxAoGetRepeat

機能

リピートの設定を取得します。

書式

```
INT YdxAoGetRepeat(
    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、YdxAoSetRepeat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.AoGetRepeat(id, out repeatNum);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAoGetSampleNum

機能

データ設定済みのサンプル数を取得します。

書式

```
INT YdxAoGetSampleNum(
    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.AoGetSampleNum(id, out sampleNum);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAoGetStartCondition

機能

サンプリング開始条件の設定を取得します。

書式

```
INT YdxAoGetStartCondition(
    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
2	アナログ入力トリガ（レベル比較）
3	アナログ入力トリガ（インレンジ比較）
4	アナログ入力トリガ（アウトレンジ比較）

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、YdxAoSetStartCondition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.AoGetStartCondition(id, out condition, out delay);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAoGetStartExternal

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 YdxAoGetStartExternal(
    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、YdxAoSetStartExternal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.AoGetStartExternal(id, out diChannel, out mode);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAoGetStartLevel

機能

サンプリング開始条件（アナログ入力トリガ　レベル比較）の設定を取得します。
しきい値は、バイナリ値で取得します。

書式

```
INT YdxAoGetStartLevel(
    INT id,
    INT* aiChannel,
    INT* level,
    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

aiChannel

比較をするアナログ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*

level

しきい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*

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（しきい値未満から、しきい値以上に変化した時に、条件成立）
1	立ち下がりエッジセンス（しきい値を超える値から、しきい値以下に変化した時に、条件成立）
2	両エッジセンス
3	ハイレベルセンス（しきい値以上の時に、条件成立。最初からしきい値以上だった場合も、条件成立）
4	ローレベルセンス（しきい値以下の時に、条件成立。最初からしきい値以下だった場合も、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、YdxAoSetStartLevel関数 を参照してください。

使用例

サンプリング開始条件（アナログ入力トリガ　レベル比較）の設定を取得します。

C#

```
int result;
int aiChannel;
int level;
int mode;
result = Ydx.AoGetStartLevel(id, out aiChannel, out level, out mode);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim aiChannel As Integer
Dim level1 As Integer
Dim mode As Integer
result = YdxAoGetStartLevel(id, aiChannel, level, mode)
```

VB6.0

```
Dim result As Long
Dim aiChannel As Long
Dim level1 As Long
Dim mode As Long
result = YdxAoGetStartLevel(id, aiChannel, level, mode)
```

C++/CLI

```
int result;
int aiChannel;
int level;
int mode;
result = YdxAoGetStartLevel(id, &aiChannel, &level, &mode);
```

C/C++

```
INT result;
INT aiChannel;
INT level;
INT mode;
result = YdxAoGetStartLevel(id, &aiChannel, &level, &mode);
```

YdxAoGetStartLevelVolt

機能

サンプリング開始条件（アナログ入力トリガ　レベル比較）の設定を取得します。
しきい値は、電圧値で取得します。

書式

```
INT YdxAoGetStartLevelVolt(
    INT id,
    INT* aiChannel,
    float* volt,
    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

aiChannel

比較をするアナログ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*

volt

しきい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 float	Single	Single	float*	FLOAT*

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（しきい値未満から、しきい値以上に変化した時に、条件成立）
1	立ち下がりエッジセンス（しきい値を超える値から、しきい値以下に変化した時に、条件成立）
2	両エッジセンス
3	ハイレベルセンス（しきい値以上の時に、条件成立。最初からしきい値以上だった場合も、条件成立）
4	ローレベルセンス（しきい値以下の時に、条件成立。最初からしきい値以下だった場合も、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、YdxAoSetStartLevelVolt関数 を参照してください。

使用例

サンプリング開始条件（アナログ入力トリガ　レベル比較）の設定を取得します。

C#

```
int result;
int aiChannel;
float volt;
int mode;
result = Ydx.AoGetStartLevelVolt(id, out aiChannel, out volt, out mode);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim aiChannel As Integer
Dim volt As Single
Dim mode As Integer
result = YdxAoGetStartLevelVolt(id, aiChannel, volt, mode)
```

VB6.0

```
Dim result As Long
Dim aiChannel As Long
Dim volt As Single
Dim mode As Long
result = YdxAoGetStartLevelVolt(id, aiChannel, volt, mode)
```

C++/CLI

```
int result;
int aiChannel;
float volt;
int mode;
result = YdxAoGetStartLevelVolt(id, &aiChannel, &volt, &mode);
```

C/C++

```
INT result;
INT aiChannel;
float volt;
INT mode;
result = YdxAoGetStartLevelVolt(id, &aiChannel, &volt, &mode);
```

YdxAoGetStartInRange

機能

サンプリング開始条件（アナログ入力トリガ　インレンジ比較）の設定を取得します。
しきい値は、バイナリ値で取得します。

書式

```
INT YdxAoGetStartInRange(
    INT id,
    INT* aiChannel,
    INT* level1,
    INT* level2,
    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

aiChannel

比較をするアナログ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*

level1・level2

しきい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*

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	エッジセンス（インレンジでない状態から、インレンジに変化した時に、条件成立）
1	レベルセンス（インレンジの時に、条件成立。最初からインレンジだった場合も、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、YdxAoSetStartInRange関数 を参照してください。

使用例

サンプリング開始条件（アナログ入力トリガ　インレンジ比較）の設定を取得します。

C#

```
int result;
int aiChannel;
int level1;
int level2;
int mode;
result = Ydx.AoGetStartInRange(id, out aiChannel, out level1, out level2, out mode);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim aiChannel As Integer
Dim level1 As Integer
Dim level2 As Integer
```

```
Dim mode As Integer
result = YdxAoGetStartInRange(id, aiChannel, level1, level2, mode)
```

VB6.0

```
Dim result As Long
Dim aiChannel As Long
Dim level1 As Long
Dim level2 As Long
Dim mode As Long
result = YdxAoGetStartInRange(id, aiChannel, level1, level2, mode)
```

C++/CLI

```
int result;
int aiChannel;
int level1;
int level2;
int mode;
result = YdxAoGetStartInRange(id, &aiChannel, &level1, &level2, &mode);
```

C/C++

```
INT result;
INT aiChannel;
INT level1;
INT level2;
INT mode;
result = YdxAoGetStartInRange(id, &aiChannel, &level1, &level2, &mode);
```

YdxAoGetStartInRangeVolt

機能

サンプリング開始条件（アナログ入力トリガ　インレンジ比較）の設定を取得します。
しきい値は、電圧値で取得します。

書式

```
INT YdxAoGetStartInRangeVolt(
    INT id,
    INT* aiChannel,
    float* volt1,
    float* volt2,
    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

aiChannel

比較をするアナログ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*

volt1・volt2

しきい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 float	Single	Single	float*	FLOAT*

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	エッジセンス（インレンジでない状態から、インレンジに変化した時に、条件成立）
1	レベルセンス（インレンジの時に、条件成立。最初からインレンジだった場合も、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、YdxAoSetStartInRangeVolt関数 を参照してください。

使用例

サンプリング開始条件（アナログ入力トリガ　インレンジ比較）の設定を取得します。

C#

```
int result;
int aiChannel;
float volt1;
float volt2;
int mode;
result = Ydx.AoGetStartInRangeVolt(id, out aiChannel, out volt1, out volt2, out mode);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim aiChannel As Integer
Dim volt1 As Single
Dim volt2 As Single
```

```
Dim mode As Integer
result = YdxAoGetStartInRangeVolt(id, aiChannel, volt1, volt2, mode)
```

VB6.0

```
Dim result As Long
Dim aiChannel As Long
Dim volt1 As Single
Dim volt2 As Single
Dim mode As Long
result = YdxAoGetStartInRangeVolt(id, aiChannel, volt1, volt2, mode)
```

C++/CLI

```
int result;
int aiChannel;
float volt1;
float volt2;
int mode;
result = YdxAoGetStartInRangeVolt(id, &aiChannel, &volt1, &volt2, &mode);
```

C/C++

```
INT result;
INT aiChannel;
float volt1;
float volt2;
INT mode;
result = YdxAoGetStartInRangeVolt(id, &aiChannel, &volt1, &volt2, &mode);
```

YdxAoGetStartOutRange

機能

サンプリング開始条件（アナログ入力トリガ　アウトレンジ比較）の設定を取得します。
しきい値は、バイナリ値で取得します。

書式

```
INT YdxAoGetStartOutRange(
    INT id,
    INT* aiChannel,
    INT* level1,
    INT* level2,
    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

aiChannel

比較をするアナログ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*

level1・level2

しきい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*

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	エッジセンス（アウトレンジでない状態から、アウトレンジに変化した時に、条件成立）
1	レベルセンス（アウトレンジの時に、条件成立。最初からアウトレンジだった場合も、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、YdxAoSetStartOutRange関数 を参照してください。

使用例

サンプリング開始条件（アナログ入力トリガ　アウトレンジ比較）の設定を取得します。

C#

```
int result;
int aiChannel;
int level1;
int level2;
int mode;
result = Ydx.AoGetStartOutRange(id, out aiChannel, out level1, out level2, out mode);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim aiChannel As Integer
Dim level1 As Integer
Dim level2 As Integer
```

```
Dim mode As Integer
result = YdxAoGetStartOutRange(id, aiChannel, level1, level2, mode)
```

VB6.0

```
Dim result As Long
Dim aiChannel As Long
Dim level1 As Long
Dim level2 As Long
Dim mode As Long
result = YdxAoGetStartOutRange(id, aiChannel, level1, level2, mode)
```

C++/CLI

```
int result;
int aiChannel;
int level1;
int level2;
int mode;
result = YdxAoGetStartOutRange(id, &aiChannel, &level1, &level2, &mode);
```

C/C++

```
INT result;
INT aiChannel;
INT level1;
INT level2;
INT mode;
result = YdxAoGetStartOutRange(id, &aiChannel, &level1, &level2, &mode);
```

YdxAoGetStartOutRangeVolt

機能

サンプリング開始条件（アナログ入力トリガ　アウトレンジ比較）の設定を取得します。
しきい値は、電圧値で取得します。

書式

```
INT YdxAoGetStartOutRangeVolt(
    INT id,
    INT* aiChannel,
    float* volt1,
    float* volt2,
    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

aiChannel

比較をするアナログ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*

volt1・volt2

しきい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 float	Single	Single	float*	FLOAT*

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	エッジセンス（アウトレンジでない状態から、アウトレンジに変化した時に、条件成立）
1	レベルセンス（アウトレンジの時に、条件成立。最初からアウトレンジだった場合も、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、YdxAoSetStartOutRangeVolt関数 を参照してください。

使用例

サンプリング開始条件（アナログ入力トリガ　アウトレンジ比較）の設定を取得します。

C#

```
int result;
int aiChannel;
float volt1;
float volt2;
int mode;
result = Ydx.AoGetStartOutRangeVolt(id, out aiChannel, out volt1, out volt2, out mode);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim aiChannel As Integer
Dim volt1 As Single
Dim volt2 As Single
```

```
Dim mode As Integer
result = YdxAoGetStartOutRangeVolt(id, aiChannel, volt1, volt2, mode)
```

VB6.0

```
Dim result As Long
Dim aiChannel As Long
Dim volt1 As Single
Dim volt2 As Single
Dim mode As Long
result = YdxAoGetStartOutRangeVolt(id, aiChannel, volt1, volt2, mode)
```

C++/CLI

```
int result;
int aiChannel;
float volt1;
float volt2;
int mode;
result = YdxAoGetStartOutRangeVolt(id, &aiChannel, &volt1, &volt2, &mode);
```

C/C++

```
INT result;
INT aiChannel;
float volt1;
float volt2;
INT mode;
result = YdxAoGetStartOutRangeVolt(id, &aiChannel, &volt1, &volt2, &mode);
```

YdxAoGetStopCondition

機能

サンプリング停止条件の設定を取得します。

書式

```
INT YdxAoGetStopCondition(
    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
2	アナログ入力トリガ（レベル比較）
3	アナログ入力トリガ（インレンジ比較）
4	アナログ入力トリガ（アウトレンジ比較）

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、YdxAoSetStopCondition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.AoGetStopCondition(id, out condition, out delay);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAoGetStopExternal

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 YdxAoGetStopExternal(
    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、YdxAoSetStopExternal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.AoGetStopExternal(id, out diChannel, out mode);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAoGetStopLevel

機能

サンプリング停止条件（アナログ入力トリガ　レベル比較）の設定を取得します。
しきい値は、バイナリ値で取得します。

書式

```
INT YdxAoGetStopLevel(
    INT id,
    INT* aiChannel,
    INT* level,
    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

aiChannel

比較をするアナログ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*

level

しきい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*

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（しきい値未満から、しきい値以上に変化した時に、条件成立）
1	立ち下がりエッジセンス（しきい値を超える値から、しきい値以下に変化した時に、条件成立）
2	両エッジセンス
3	ハイレベルセンス（しきい値以上の時に、条件成立。最初からしきい値以上だった場合も、条件成立）
4	ローレベルセンス（しきい値以下の時に、条件成立。最初からしきい値以下だった場合も、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、YdxAoSetStopLevel関数 を参照してください。

使用例

サンプリング停止条件（アナログ入力トリガ　レベル比較）の設定を取得します。

C#

```
int result;
int aiChannel;
int level;
int mode;
result = Ydx.AoGetStopLevel(id, out aiChannel, out level, out mode);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim aiChannel As Integer
Dim level1 As Integer
Dim mode As Integer
result = YdxAoGetStopLevel(id, aiChannel, level, mode)
```

VB6.0

```
Dim result As Long
Dim aiChannel As Long
Dim level1 As Long
Dim mode As Long
result = YdxAoGetStopLevel(id, aiChannel, level, mode)
```

C++/CLI

```
int result;
int aiChannel;
int level;
int mode;
result = YdxAoGetStopLevel(id, &aiChannel, &level, &mode);
```

C/C++

```
INT result;
INT aiChannel;
INT level;
INT mode;
result = YdxAoGetStopLevel(id, &aiChannel, &level, &mode);
```

YdxAoGetStopLevelVolt

機能

サンプリング停止条件（アナログ入力トリガ　レベル比較）の設定を取得します。
しきい値は、電圧値で取得します。

書式

```
INT YdxAoGetStopLevelVolt(
    INT id,
    INT* aiChannel,
    float* volt,
    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

aiChannel

比較をするアナログ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*

volt

しきい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 float	Single	Single	float*	FLOAT*

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（しきい値未満から、しきい値以上に変化した時に、条件成立）
1	立ち下がりエッジセンス（しきい値を超える値から、しきい値以下に変化した時に、条件成立）
2	両エッジセンス
3	ハイレベルセンス（しきい値以上の時に、条件成立。最初からしきい値以上だった場合も、条件成立）
4	ローレベルセンス（しきい値以下の時に、条件成立。最初からしきい値以下だった場合も、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、YdxAoSetStopLevelVolt関数 を参照してください。

使用例

サンプリング停止条件（アナログ入力トリガ　レベル比較）の設定を取得します。

C#

```
int result;
int aiChannel;
float volt;
int mode;
result = Ydx.AoGetStopLevelVolt(id, out aiChannel, out volt, out mode);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim aiChannel As Integer
Dim volt As Single
Dim mode As Integer
result = YdxAoGetStopLevelVolt(id, aiChannel, volt, mode)
```

VB6.0

```
Dim result As Long
Dim aiChannel As Long
Dim volt As Single
Dim mode As Long
result = YdxAoGetStopLevelVolt(id, aiChannel, volt, mode)
```

C++/CLI

```
int result;
int aiChannel;
float volt;
int mode;
result = YdxAoGetStopLevelVolt(id, &aiChannel, &volt, &mode);
```

C/C++

```
INT result;
INT aiChannel;
float volt;
INT mode;
result = YdxAoGetStopLevelVolt(id, &aiChannel, &volt, &mode);
```

YdxAoGetStopInRange

機能

サンプリング停止条件（アナログ入力トリガ　インレンジ比較）の設定を取得します。
しきい値は、バイナリ値で取得します。

書式

```
INT YdxAoGetStopInRange(
    INT id,
    INT* aiChannel,
    INT* level1,
    INT* level2,
    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

aiChannel

比較をするアナログ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*

level1・level2

しきい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*

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	エッジセンス（インレンジでない状態から、インレンジに変化した時に、条件成立）
1	レベルセンス（インレンジの時に、条件成立。最初からインレンジだった場合も、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、YdxAoSetStopInRange関数 を参照してください。

使用例

サンプリング停止条件（アナログ入力トリガ　インレンジ比較）の設定を取得します。

C#

```
int result;
int aiChannel;
int level1;
int level2;
int mode;
result = Ydx.AoGetStopInRange(id, out aiChannel, out level1, out level2, out mode);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim aiChannel As Integer
Dim level1 As Integer
Dim level2 As Integer
```

```
Dim mode As Integer
result = YdxAoGetStopInRange(id, aiChannel, level1, level2, mode)
```

VB6.0

```
Dim result As Long
Dim aiChannel As Long
Dim level1 As Long
Dim level2 As Long
Dim mode As Long
result = YdxAoGetStopInRange(id, aiChannel, level1, level2, mode)
```

C++/CLI

```
int result;
int aiChannel;
int level1;
int level2;
int mode;
result = YdxAoGetStopInRange(id, &aiChannel, &level1, &level2, &mode);
```

C/C++

```
INT result;
INT aiChannel;
INT level1;
INT level2;
INT mode;
result = YdxAoGetStopInRange(id, &aiChannel, &level1, &level2, &mode);
```

YdxAoGetStopInRangeVolt

機能

サンプリング停止条件（アナログ入力トリガ　インレンジ比較）の設定を取得します。
しきい値は、電圧値で取得します。

書式

```
INT YdxAoGetStopInRangeVolt(
    INT id,
    INT* aiChannel,
    float* volt1,
    float* volt2,
    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

aiChannel

比較をするアナログ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*

volt1・volt2

しきい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 float	Single	Single	float*	FLOAT*

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	エッジセンス（インレンジでない状態から、インレンジに変化した時に、条件成立）
1	レベルセンス（インレンジの時に、条件成立。最初からインレンジだった場合も、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、YdxAoSetStopInRangeVolt関数 を参照してください。

使用例

サンプリング停止条件（アナログ入力トリガ　インレンジ比較）の設定を取得します。

C#

```
int result;
int aiChannel;
float volt1;
float volt2;
int mode;
result = Ydx.AoGetStopInRangeVolt(id, out aiChannel, out volt1, out volt2, out mode);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim aiChannel As Integer
Dim volt1 As Single
Dim volt2 As Single
```

```
Dim mode As Integer
result = YdxAoGetStopInRangeVolt(id, aiChannel, volt1, volt2, mode)
```

VB6.0

```
Dim result As Long
Dim aiChannel As Long
Dim volt1 As Single
Dim volt2 As Single
Dim mode As Long
result = YdxAoGetStopInRangeVolt(id, aiChannel, volt1, volt2, mode)
```

C++/CLI

```
int result;
int aiChannel;
float volt1;
float volt2;
int mode;
result = YdxAoGetStopInRangeVolt(id, &aiChannel, &volt1, &volt2, &mode);
```

C/C++

```
INT result;
INT aiChannel;
float volt1;
float volt2;
INT mode;
result = YdxAoGetStopInRangeVolt(id, &aiChannel, &volt1, &volt2, &mode);
```

YdxAoGetStopOutRange

機能

サンプリング停止条件（アナログ入力トリガ　アウトレンジ比較）の設定を取得します。
しきい値は、バイナリ値で取得します。

書式

```
INT YdxAoGetStopOutRange(
    INT id,
    INT* aiChannel,
    INT* level1,
    INT* level2,
    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

aiChannel

比較をするアナログ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*

level1・level2

しきい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*

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	エッジセンス（アウトレンジでない状態から、アウトレンジに変化した時に、条件成立）
1	レベルセンス（アウトレンジの時に、条件成立。最初からアウトレンジだった場合も、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、YdxAoSetStopOutRange関数 を参照してください。

使用例

サンプリング停止条件（アナログ入力トリガ　アウトレンジ比較）の設定を取得します。

C#

```
int result;
int aiChannel;
int level1;
int level2;
int mode;
result = Ydx.AoGetStopOutRange(id, out aiChannel, out level1, out level2, out mode);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim aiChannel As Integer
Dim level1 As Integer
Dim level2 As Integer
```

```
Dim mode As Integer
result = YdxAoGetStopOutRange(id, aiChannel, level1, level2, mode)
```

VB6.0

```
Dim result As Long
Dim aiChannel As Long
Dim level1 As Long
Dim level2 As Long
Dim mode As Long
result = YdxAoGetStopOutRange(id, aiChannel, level1, level2, mode)
```

C++/CLI

```
int result;
int aiChannel;
int level1;
int level2;
int mode;
result = YdxAoGetStopOutRange(id, &aiChannel, &level1, &level2, &mode);
```

C/C++

```
INT result;
INT aiChannel;
INT level1;
INT level2;
INT mode;
result = YdxAoGetStopOutRange(id, &aiChannel, &level1, &level2, &mode);
```

YdxAoGetStopOutRangeVolt

機能

サンプリング停止条件（アナログ入力トリガ　アウトレンジ比較）の設定を取得します。
しきい値は、電圧値で取得します。

書式

```
INT YdxAoGetStopOutRangeVolt(
    INT id,
    INT* aiChannel,
    float* volt1,
    float* volt2,
    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

aiChannel

比較をするアナログ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*

volt1・volt2

しきい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 float	Single	Single	float*	FLOAT*

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	エッジセンス（アウトレンジでない状態から、アウトレンジに変化した時に、条件成立）
1	レベルセンス（アウトレンジの時に、条件成立。最初からアウトレンジだった場合も、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、YdxAoSetStopOutRangeVolt関数 を参照してください。

使用例

サンプリング停止条件（アナログ入力トリガ　アウトレンジ比較）の設定を取得します。

C#

```
int result;
int aiChannel;
float volt1;
float volt2;
int mode;
result = Ydx.AoGetStopOutRangeVolt(id, out aiChannel, out volt1, out volt2, out mode);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim aiChannel As Integer
Dim volt1 As Single
Dim volt2 As Single
```

```
Dim mode As Integer
result = YdxAoGetStopOutRangeVolt(id, aiChannel, volt1, volt2, mode)
```

VB6.0

```
Dim result As Long
Dim aiChannel As Long
Dim volt1 As Single
Dim volt2 As Single
Dim mode As Long
result = YdxAoGetStopOutRangeVolt(id, aiChannel, volt1, volt2, mode)
```

C++/CLI

```
int result;
int aiChannel;
float volt1;
float volt2;
int mode;
result = YdxAoGetStopOutRangeVolt(id, &aiChannel, &volt1, &volt2, &mode);
```

C/C++

```
INT result;
INT aiChannel;
float volt1;
float volt2;
INT mode;
result = YdxAoGetStopOutRangeVolt(id, &aiChannel, &volt1, &volt2, &mode);
```

YdxAoSetData

機能

データをバイナリ値で設定します。

書式

```
INT YdxAoSetData(
    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。
データ数ではなく、サンプル数で指定してください。

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

data

データを格納した変数へのポインタを指定します。
（sampleNum * 有効なチャネル 数）個分の配列を指定してください。

（例）サンプル数=1000、有効チャネル=CH0・CH1・CH3の場合、データは以下の順番で格納してください。

データ	CH0	CH1	CH3	CH0	CH1	CH3	・・・	CH0	CH1	CH3	CH0	CH1	CH3
サンプル数	1	1	1	2	2	2	・・・	999	999	999	1000	1000	1000
データ数	1	2	3	4	5	6	・・・	2995	2996	2997	2998	2999	3000

データの値の範囲は、-32768～32767です。
電圧値からの換算式は以下のとおりです。
data = 電圧値 * 32767 / 10

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、YdxAoSetBuffer関数・YdxAoSetChannel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、アナログ出力が 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、アナログ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。
（有効なチャネル数は4チャネルになっていて、データは15,000にする場合）

C#

```
int result;
int[] data = new int[4000];
int i;
for (i = 0; i < 4000; i++)
{
    data[i] = 15000;
```

```
}
result = Ydx.AoSetData(id, 1000, data);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim data(3999) As Integer
Dim i As Integer
For i = 0 To 3999
    data(i) = 15000
Next
result = YdxAoSetData(id, 1000, data)
```

VB6.0

```
Dim result As Long
Dim data(3999) As Long
Dim i As Long
For i = 0 To 3999
    data(i) = 15000
Next
result = YdxAoSetData(id, 1000, data(0))
```

C++/CLI

```
int result;
int data[4000];
int i;
for (i = 0; i < 4000; i++)
{
    data[i] = 15000;
}
result = YdxAoSetData(id, 1000, data);
```

C/C++

```
INT result;
INT data[4000];
INT i;
for (i = 0; i < 4000; i++)
{
    data[i] = 15000;
}
result = YdxAoSetData(id, 1000, data);
```

YdxAoSetDataVolt

機能

データを電圧値で設定します。

書式

```
INT YdxAoSetDataVolt(
    INT id,
    INT sampleNum,
    FLOAT* volt
);
```

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。
データ数ではなく、サンプル数で指定してください。

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

volt

データを格納した変数へのポインタを指定します。
（sampleNum * 有効なチャネル 数）個分の配列を指定してください。

（例）サンプル数=1000、有効チャネル=CH0・CH1・CH3の場合、データは以下の順番で格納してください。

データ	CH0	CH1	CH3	CH0	CH1	CH3	・・・	CH0	CH1	CH3	CH0	CH1	CH3
サンプル数	1	1	1	2	2	2	・・・	999	999	999	1000	1000	1000
データ数	1	2	3	4	5	6	・・・	2995	2996	2997	2998	2999	3000

単位は「V」です。
データの値の範囲は、-10～10 [V]です。

言語	C#	VB（.NET2002以降）	VB6.0	C++/CLI	C/C++
型	float[]	Single	Single	float*	FLOAT*

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、YdxAoSetBuffer関数 ・ YdxAoSetChannel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、アナログ出力が 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、アナログ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。
（有効なチャネル数は4チャネル、データは5.6Vの場合）

C#

```
int result;
float[] volt = new float[4000];
int i;
for (i = 0; i < 4000; i++)
{
    volt[i] = 5.6F;
```

```
}
result = Ydx.AoSetDataVolt(id, 1000, volt);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim volt(3999) As Single
Dim i As Integer
For i = 0 To 3999
    volt(i) = 5.6
Next
result = YdxAoSetDataVolt(id, 1000, volt)
```

VB6.0

```
Dim result As Long
Dim volt(3999) As Single
Dim i As Long
For i = 0 To 3999
    volt(i) = 5.6
Next
result = YdxAoSetDataVolt(id, 1000, volt(0))
```

C++/CLI

```
int result;
float volt[4000];
int i;
for (i = 0; i < 4000; i++)
{
    volt[i] = 5.6;
}
result = YdxAoSetDataVolt(id, 1000, volt);
```

C/C++

```
INT result;
float volt[4000];
INT i;
for (i = 0; i < 4000; i++)
{
    volt[i] = 5.6;
}
result = YdxAoSetDataVolt(id, 1000, volt);
```

YdxAoClearData

機能

データをクリアします。

書式

```
INT YdxAoClearData(
    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、アナログ出力が 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.AoClearData(id);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAoStart

機能

アナログ出力動作 を開始します。

書式

```
INT YdxAoStart(
    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、アナログ出力が 動作中 には実行できません。

使用例

アナログ出力動作を開始します。

C#

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

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAoStop

機能

アナログ出力動作 を停止します。

書式

```
INT YdxAoStop(
    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

アナログ出力動作を停止します。

C#

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

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAoReset

機能

アナログ出力機能をリセットします。

書式

```
INT YdxAoReset(
    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。

- アナログ出力が 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。

使用例

アナログ出力機能をリセットします。

C#

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

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

YdxAoGetStatus

機能

現在の状態を取得します。

書式

```
INT YdxAoGetStatus(
    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（YdxAoStart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、YdxAoSetCheckSampleNum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.AoGetStatus(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 = YdxAoGetStatus(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 = YdxAoGetStatus(id, status, sampleCount, repeatCount, notOutNum)
```

C++/CLI

```
int result;
int status;
int sampleCount;
int repeatCount;
result = YdxAoGetStatus(id, &status, &sampleCount, &repeatCount);
```

C/C++

```
INT result;
INT status;
INT sampleCount;
INT repeatCount;
INT notOutNum;
result = YdxAoGetStatus(id, &status, &sampleCount, &repeatCount, &notOutNum);
```

YdxAoGetEventStatus

機能

イベント発生要因を取得します。

書式

```
INT YdxAoGetEventStatus(
    INT id,
    INT* factor,
    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

factor

イベント発生要因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	イベント要因
00000001h	YDX_EVENT_STOP	動作終了 動作が終了するとオンになります。
00000002h	YDX_EVENT_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、YdxAoSetCheckSampleNum関数 で変更できます。
00000004h	YDX_EVENT_START_TRIG	開始条件 成立 開始条件が成立するとオンになります。
00000008h	YDX_EVENT_STOP_TRIG	停止条件 成立 停止条件が成立するとオンになります。

値	定義名	イベント要因
00010000h	YDX_EVENT_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るとオンになります。 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_EVENT_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されるとオンになります。 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_EVENT_COMMUNICATE_ERR	通信エラー発生 USB通信に異常が検出されるとオンになります。 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

イベント発生時の出力済みサンプル回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

イベント発生時の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

イベント発生時の未出力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

イベント発生要因を読み出します。

C#

```
int result;
int factor;
int sampleCount;
int repeatCount;
int notOutNum;
result = Ydx.AoGetEventStatus(id, out factor, out sampleCount, out repeatCount, out
notOutNum);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim factor As Integer
Dim sampleCount As Integer
Dim repeatCount As Integer
Dim notOutNum As Integer
result = YdxAoGetEventStatus(id, factor, sampleCount, repeatCount, notOutNum)
```

VB6.0

```
Dim result As Long
Dim factor As Long
Dim sampleCount As Long
Dim repeatCount As Long
Dim notOutNum As Long
result = YdxAoGetEventStatus(id, factor, sampleCount, repeatCount, notOutNum)
```

C++/CLI

```
int result;
int factor;
int sampleCount;
int repeatCount;
int notOutNum;
result = YdxAoGetEventStatus(id, &factor, &sampleCount, &repeatCount, &notOutNum);
```

C/C++

```
INT result;
INT factor;
INT sampleCount;
INT repeatCount;
INT notOutNum;
result = YdxAoGetEventStatus(id, &factor, &sampleCount, &repeatCount, &notOutNum);
```

参考

- AoEvent

  高機能アナログ出力のサンプルプログラムです。
  動作状態の監視をイベントでおこなっています。

デジタル入力関数一覧

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

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-1～3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。

			LSB
IN3	IN2	IN1	IN0

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0～3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0～3（IN0～IN3）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IN0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, 0, 4, 1, data);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

```
int result;
int data[4];
result = YdxDiInputBit(id, 0, 4, 1, data);
```

C/C++

```
INT result;
INT data[4];
result = YdxDiInputBit(id, 0, 4, 1, data);
```

デジタル出力関数一覧

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。
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。

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。
設定範囲は、0または1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をONにします。

C#

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

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

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

C/C++

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

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。
本機種では、必ず0を指定してください。

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

num

出力するチャネル数を指定します。
本機種では、必ず1を指定してください。

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

data

出力する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

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

デジタル出力チャネル0（OUT0）を、ONにします。

C#

```
int result;
int[] data = new int[1];
data[0] = 1;
result = Ydx.DoOutputBit(id, 0, 1, data);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

```
int result;
int data[1];
data[0] = 0;
result = YdxDoOutputBit(id, 0, 1, data);
```

C/C++

```
INT result;
INT data[1];
data[0] = 0;
result = YdxDoOutputBit(id, 0, 1, data);
```

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。

値	意味
0	OFF
1	ON

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。
本機種では、必ず0を指定してください。

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

num

状態の読み込みをするチャネル数を指定します。
本機種では、必ず1を指定してください。

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

data

状態を格納する変数へのポインタを指定します。

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

使用例

デジタル出力チャネル0（OUT0）の状態を読み込みます。

C#

```
int result;
int[] data = new int[1];
result = Ydx.DoGetOutputBit(id, 0, 1, data);
```

VB（.NET2002以降）

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

VB6.0

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

C++/CLI

```
int result;
int data[1];
result = YdxDoGetOutputBit(id, 0, 1, data);
```

C/C++

```
INT result;
INT data[1];
result = YdxDoGetOutputbit(id, 0, 1, data);
```

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）

アナログ入力動作

アナログ入力動作とは、高機能アナログ入力 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）

アナログ入力動作中

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・開始遅延・サンプリングの期間が含まれます。

動作開始は、YdxAiStart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）
- YdxAiStop関数 の実行

参考

- 機能説明（開始条件・停止条件・リピート）

アナログ出力動作

アナログ出力動作とは、高機能アナログ出力 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）

アナログ出力動作中

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、YdxAoStart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）
- YdxAoStop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。