株式会社ワイツー

UBシリーズ　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UBシリーズ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。

> **オブジェクト指向のライブラリも利用可能になりました**
>
> 詳細は、Y2.UsbIO を参照してください。

DIO-8/8B-UBT(W)について

DIO-8/8B-UBT(W)はDIO-8/8B-UBTとは端子台が違うだけの製品です。
DIO-8/8B-UBT(W)を使用される場合、本ソフトウェアマニュアルではDIO-8/8B-UBTとしてご覧ください。
YdciOpenや動作確認ユーティリティで指定する型名はDIO-8/8B-UBTとしてください。

用語説明

ボード識別スイッチ番号

ボード識別用スイッチにて設定された値
（設定方法については取扱説明書を参照してください）

基本仕様

接続台数

1台のパソコンから制御できるボードの最大数は各機種16台です（UBシリーズ全体で16台ではありません）。

機種名	最大接続数
DIO-8/8C-UBT	16台
DIO-8/8B-UBCまたはDIO-8/8B-UBT	16台
RLY-P4/2/0B-UBT	16台

> ⚠ DIO-8/8B-UBCとDIO-8/8B-UBTをそれぞれ16台接続することはできません。合わせて16台が最大です

注意事項

PCが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。
スタンバイや休止状態になると、以降ボードが動作しません。

動作環境（Windows）

PC

IBM PC/AT互換機（DOS/V機）

OS

DIO-8/8C-UBT

Windows 11 x64
Windows 10 x86, x64
Windows 10 IoT Enterprise[1]

DIO-8/8B-UBC、DIO-8/8B-UBT、RLY-P4/2/0B-UBT

Windows 11 x64
Windows 10 x86, x64
Windows 10 IoT Enterprise[1]
Windows 8.1 x86, x64[2]
Windows 8 x86, x64[3]
Windows 7 x86, x64[3]
Windows Vista x86, x64[4]
Windows XP x64[5]
Windows XP[5]
Windows 2000[5]
Windows Me[6][7]
Windows 98, 98SE[6][7]

対応言語

Microsoft Visual C++（6.0, .NET2002以降）
Microsoft Visual C#
Microsoft Visual Basic（6.0, .NET2002以降）
VBA
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. 2020年9月25日リリースの旧バージョンのドライバを使用します ↩
3. 2015年10月8日リリースの旧バージョンのドライバを使用します ↩↩
4. 2014年2月28日リリースの旧バージョンのドライバを使用します ↩
5. 2012年6月29日リリースの旧バージョンのドライバを使用します ↩↩↩
6. 2009年10月26日リリースの旧バージョンのドライバを使用します ↩↩
7. YdciDioOutputStatusとYdciRlyOutputStatusの2つの関数は使用できません ↩↩

動作環境（Linux）

PC

IBM PC/AT互換機（DOS/V機）

シングルボードコンピュータ

Raspberry Pi 5/4/3
Jetson Nano

OS

PC

Ubuntu Desktop 24.04 LTS
Ubuntu Desktop 22.04 LTS
Ubuntu Desktop 20.04 LTS
Ubuntu Desktop 18.04 LTS
Ubuntu Desktop 16.04 LTS
Debian 12（amd64, i386）
Debian 11（amd64, i386）
Debian 10（amd64, i386）
Debian 9（amd64, i386）
CentOS 8（x86_64）

Raspberry Pi 5

Raspberry Pi OS bookworm（64-bit, 32-bit）
Ubuntu Server 24.04 LTS
Ubuntu Desktop 24.04 LTS

Raspberry Pi 4

Raspberry Pi OS bookworm（64-bit, 32-bit）
Raspberry Pi OS bullseye（64-bit, 32-bit）
Raspberry Pi OS（Raspbian） buster
Ubuntu Server 24.04 LTS
Ubuntu Server 22.04 LTS（arm64, armhf）
Ubuntu Server 20.04 LTS（arm64, armhf）
Ubuntu Server 18.04 LTS（arm64, armhf）
Ubuntu Desktop 24.04 LTS
Ubuntu Desktop 22.04 LTS

Raspberry Pi 3

Raspberry Pi OS bullseye（64-bit, 32-bit）
Raspberry Pi OS（Raspbian） buster
Raspberry Pi OS（Raspbian） stretch
Ubuntu Server 22.04 LTS（arm64, armhf）
Ubuntu Server 20.04 LTS（arm64, armhf）
Ubuntu Server 18.04 LTS（arm64, armhf）

Jetson Nano

Jetson Nano Developer Kit SD Card Image

対応言語

GCC5+
Python3

必要なソフトウェアパッケージ

libusb-1.0
GCC5+

実行手順

1. 準備
   ボードとパソコンをUSBケーブルで接続してください。
   （ボードを接続後、パソコンがボードを認識するまでに数秒程度必要となる場合があります）
2. ボードをオープン
   YdciOpen関数を使用してボードをオープンします。
   YdciOpen関数にてオープンしたボードは、アプリケーション終了時に必ずクローズ（YdciClose関数）するようにしてください。
3. デジタル入出力・リレー出力
   - デジタル入力
     YdciDioInput関数を使用してデジタル入力をおこないます。
   - デジタル出力
     YdciDioOutput関数を使用してデジタル出力をおこないます。
     YdciDioOutputStatus関数を使用して出力状態を知ることもできます。
   - リレー出力
     YdciRlyOutput関数を使用してリレー出力をおこないます。
     YdciRlyOutputStatus関数を使用して出力状態を知ることもできます。
4. ボードをクローズ
   YdciClose関数を使用してボードをクローズします。
5. 終了
   USBケーブルを外し、ボードへの電源供給を止めます。

戻り値一覧

エラーコード (16進数値/10進数値)	意味	対処方法
YDCI_RESULT_SUCCESS (H'00000000 / 0)	正常終了	
YDCI_RESULT_ERROR (H'CE000001 / -838860799)	エラー	エラーが発生しました。USBケーブルが抜けている事などが考えられます。
YDCI_RESULT_NOT_OPEN (H'CE000002 / -838860798)	オープンされていない	オープンされていないボードに対して操作がおこなわれました。YdciOpen関数にてオープンされたボードに対して操作をおこなってください。
YDCI_RESULT_ALREADY_OPEN (H'CE000003 / -838860797)	オープン済み	既にオープンされているボードに対してYdciOpen関数が実行されました
YDCI_RESULT_INVALID_ID (H'CE000004 / -838860796)	IDが不正	指定されたIDが不正です。IDはYdciOpen関数で取得したIDを指定してください。
YDCI_RESULT_CANNOT_OPEN (H'CE000006 / -838860794)	ボードがオープンできない	ボードがオープンできませんでした。USBケーブルが接続されていることを確認してください。
YDCI_RESULT_PARAMETER_ERROR (H'CE000008 / -838860792)	引数不正	関数に渡された引数が不正です。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。
YDCI_RESULT_MEM_ALLOC_ERROR (H'CE000009 / -838860791)	メモリ確保エラー	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、利用可能なメモリを増やしてください。
YDCI_RESULT_MODELNAME_ERROR (H'CE00000A / -838860790)	型名指定が不正	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。型名を再度確認してください。
YDCI_RESULT_NOT_SUPPORTED (H'CE00000C / -838860788)	サポートされていない	サポートされていない関数が実行されました
YDCI_RESULT_INVALID_BOARD_SW (H'CE000013 / -838860781)	ボード識別スイッチが不正	指定されたボード識別スイッチが不正です。0～15を指定してください。
YDCI_RESULT_FATAL_ERROR (H'CEFFFFFF / -822083585)	致命的なエラー	致命的なエラーです。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。

C#での注意事項

C#で関数を使用する場合、Ydci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#
YdciOpen	Ydci.Open
YdciClose	Ydci.Close
YdciDioOutput	Ydci.DioOutput
YdciDioInput	Ydci.DioInput
YdciRlyOutput	Ydci.RlyOutput

関数一覧

関数	機能
YdciOpen	ボードのオープンをおこないます
YdciClose	ボードのクローズをおこないます

YdciOpen

機能

ボードのオープンをおこない、ボードへのアクセスをおこなえるようにします。

適用

UBシリーズ全機種

書式

```
INT YdciOpen(
    WORD wBoardSw,
    LPCSTR lpszModelName,
    PWORD pwID,
    WORD wMode = YDCI_OPEN_NORMAL
);
```

パラメータ

wBoardSw

オープンするボードのボード識別スイッチ番号を指定します。

言語	C/C++	C++/CLI	C#	VB（.NET2002以降）	VB6.0/VBA	GCC
型	WORD	unsigned short	ushort	Short	Integer	uint16_t

lpszModelName

オープンするボードの型名を指定します。

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

pwID

IDを格納するバッファ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/C++	C++/CLI	C#	VB（.NET2002以降）	VB6.0/VBA	GCC
型	PWORD	unsigned short*	out ushort	Short	Integer	uint16_t*

wMode

オープン時の動作を指定します。

定義	値	オープン時の動作
YDCI_OPEN_NORMAL （省略可能）	0	デジタル出力・リレー出力が全てOFFになります
YDCI_OPEN_OUT_NOT_INIT	1	デジタル出力・リレー出力の出力状態は変わりません

言語	C/C++	C++/CLI	C#	VB（.NET2002以降）	VB6.0/VBA	GCC
型	WORD	unsigned short	ushort	Short	Integer	uint16_t

戻り値

関数が正常に終了した場合は0（YDCI_RESULT_SUCCESS）が返ります。
オープンに失敗した場合は0以外が返りますので、その場合はエラーコードを参照してください。

言語	C/C++	C++/CLI	C#	VB（.NET2002以降）	VB6.0/VBA	GCC
型	INT	int	int	Integer	Long	int32_t

備考

wModeを省略（もしくはYDCI_OPEN_NORMALを指定）した場合は、オープン時に全ての出力をOFFします。
YDCI_OPEN_OUT_NOT_INITを指定した場合は現在の出力状態のままオープンします。

> ⚠ YdciOpen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YdciClose関数でクローズしてください

使用例

ボード識別スイッチが0に設定されているボードをオープンします。出力は全てOFFになります。

C/C++

```
int nResult;
WORD wID;
nResult = YdciOpen(0, "RLY-P4/2/0B-UBT", &wID);
```

C++/CLI

```
int result;
unsigned short id;
result = YdciOpen(0, "RLY-P4/2/0B-UBT", &id);
```

C#

```
int result;
ushort id;
result = Ydci.Open(0, "RLY-P4/2/0B-UBT", out id);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim id As Short
result = YdciOpen(0, "RLY-P4/2/0B-UBT", id)
```

VB6.0/VBA

```
Dim lngResult As Long
Dim intID As Integer
Dim strModelName as String
strModelName = "RLY-P4/2/0B-UBT"
lngResult = YdciOpen(0, strModelName, intID)
```

GCC

```
int32_t result;
uint16_t id;
result = YdciOpen(0, "RLY-P4/2/0B-UBT", &id);
```

YdciClose

機能

ボードのクローズをおこない、ボードへのアクセスを禁止します。

適用

UBシリーズ全機種

書式

```
BOOL YdciClose(
    WORD wID
);
```

パラメータ

wID

YdciOpen関数で取得したIDを指定します。

言語	C/C++	C++/CLI	C#	VB（.NET2002以降）	VB6.0/VBA	GCC
型	WORD	unsigned short	ushort	Short	Integer	uint16_t

戻り値

関数が正常に終了した場合はTRUEが返ります。
クローズに失敗した場合はFALSEが返ります。

言語	C/C++	C++/CLI	C#	VB（.NET2002以降）	VB6.0/VBA	GCC
型	BOOL	bool	bool	Boolean	Boolean	int32_t

備考

> ⚠ YdciOpen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YdciClose関数でクローズしてください

使用例

ボードのクローズ処理をおこないます。

C/C++

```
BOOL bResult;
bResult = YdciClose(wID);
```

C++/CLI

```
bool result;
result = YdciClose(id);
```

C#

```
bool result;
result = Ydci.Close(id);
```

VB（.NET2002以降）

```
Dim result As Boolean
result = YdciClose(id)
```

VB6.0/VBA

```
Dim blnResult As Boolean
blnResult = YdciClose(intID)
```

GCC

```
int32_t result;
result = YdciClose(id);
```

関数一覧

関数	機能
YdciDioInput	入力端子の読み込みをおこないます
YdciDioOutput	出力端子の制御をおこないます
YdciDioOutputStatus	出力状態の読み込みをおこないます

YdciDioInput

機能

任意の点数の入力端子の状態を読み込みます。

適用

UBシリーズ全機種

書式

```
INT YdciDioInput(
    WORD wID,
    PBYTE pbyData,
    WORD wStart,
    WORD wCount
);
```

パラメータ

wID

YdciOpen関数で取得したIDを指定します。

言語	C/C++	C++/CLI	C#	VB（.NET2002以降）	VB6.0/VBA	GCC
型	WORD	unsigned short	ushort	Short	Integer	uint16_t

pbyData

入力データを格納するバッファ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。

入力データ	状態
0	OFF
1	ON

言語	C/C++	C++/CLI	C#	VB（.NET2002以降）	VB6.0/VBA	GCC
型	PBYTE	unsigned char*	byte	Byte	Byte	uint8_t*

wStart

入力開始番号（0～）を指定します。

言語	C/C++	C++/CLI	C#	VB（.NET2002以降）	VB6.0/VBA	GCC
型	WORD	unsigned short	ushort	Short	Integer	uint16_t

wCount

入力の読み込みをおこなう数を指定します。

言語	C/C++	C++/CLI	C#	VB（.NET2002以降）	VB6.0/VBA	GCC
型	WORD	unsigned short	ushort	Short	Integer	uint16_t

戻り値

関数が正常に終了した場合は0（YDCI_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は0以外が返りますので、その場合はエラーコードを参照してください。

言語	C/C++	C++/CLI	C#	VB（.NET2002以降）	VB6.0/VBA	GCC
型	INT	int	int	Integer	Long	int32_t

使用例

IN0からIN3の入力端子の状態を読み込みます。データはIN0から順にデータバッファへ格納されます。

C/C++

```
int nResult;
BYTE abyData[4];
nResult = YdciDioInput(wID, abyData, 0, 4);
```

C++/CLI

```
int result;
unsigned char inputData[4];
result = YdciDioInput(id, inputData, 0, 4);
```

C#

```
int result;
byte[] inputData = new byte[4];
result = Ydci.DioInput(id, inputData, 0, 4);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim inputData(3) As Byte
result = YdciDioInput(id, inputData, 0, 4)
```

VB6.0/VBA

```
Dim lngResult As Long
Dim bytData(3) As Byte
lngResult = YdciDioInput(intID, bytData(0), 0, 4)
```

GCC

```
int32_t result;
uint8_t input_data[4];
result = YdciDioInput(id, input_data, 0, 4);
```

YdciDioOutput

機能

任意の点数の出力端子を制御します。

適用

DIO-8/8B-UBC、DIO-8/8B-UBT

書式

```
INT YdciDioOutput(
    WORD wID,
    PBYTE pbyData,
    WORD wStart,
    WORD wCount
);
```

パラメータ

wID

YdciOpen関数で取得したIDを指定します。

言語	C/C++	C++/CLI	C#	VB（.NET2002以降）	VB6.0/VBA	GCC
型	WORD	unsigned short	ushort	Short	Integer	uint16_t

pbyData

出力するデータを格納したバッファへのポインタを指定します。

出力データ	状態
0	OFF
1	ON
2	現在の状態を保持

言語	C/C++	C++/CLI	C#	VB（.NET2002以降）	VB6.0/VBA	GCC
型	PBYTE	unsigned char*	byte	Byte	Byte	uint8_t*

wStart

出力開始番号（0～）を指定します。

言語	C/C++	C++/CLI	C#	VB（.NET2002以降）	VB6.0/VBA	GCC
型	WORD	unsigned short	ushort	Short	Integer	uint16_t

wCount

出力する数を指定します。

言語	C/C++	C++/CLI	C#	VB（.NET2002以降）	VB6.0/VBA	GCC
型	WORD	unsigned short	ushort	Short	Integer	uint16_t

戻り値

関数が正常に終了した場合は0（YDCI_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は0以外が返りますので、その場合はエラーコードを参照してください。

言語	C/C++	C++/CLI	C#	VB（.NET2002以降）	VB6.0/VBA	GCC
型	INT	int	int	Integer	Long	int32_t

備考

PCがスタンバイモードになると、全ての出力はOFFになります。
その後スタンバイモードから復帰した場合でも元の出力状態には戻らず、OFFのままです。
出力状態を常に保持する必要がある場合はスタンバイモードにならないようにしてください。

使用例

OUT0に0、OUT1に1、OUT2に0、OUT3に1、OUT4は現在の出力状態、OUT5は現在の出力状態、OUT6に0、OUT7に1を出力する。

C/C++

```
int nResult;
BYTE abyData[8];
abyData[0] = 0;
abyData[1] = 1;
abyData[2] = 0;
abyData[3] = 1;
abyData[4] = 2;
abyData[5] = 2;
abyData[6] = 0;
abyData[7] = 1;
nResult = YdciDioOutput(wID, abyData, 0, 8);
```

C++/CLI

```
int result;
unsigned char outputData[8];
outputData[0] = 0;
outputData[1] = 1;
outputData[2] = 0;
outputData[3] = 1;
outputData[4] = 2;
outputData[5] = 2;
outputData[6] = 0;
outputData[7] = 1;
result = YdciDioOutput(id, outputData, 0, 8);
```

C#

```
int result;
byte[] outputData = new byte[8];
outputData[0] = 0;
outputData[1] = 1;
outputData[2] = 0;
outputData[3] = 1;
outputData[4] = 2;
outputData[5] = 2;
outputData[6] = 0;
outputData[7] = 1;
result = Ydci.DioOutput(id, outputData, 0, 8);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim outputData(7) As Byte
outputData(0) = 0
outputData(1) = 1
outputData(2) = 0
outputData(3) = 1
outputData(4) = 2
outputData(5) = 2
outputData(6) = 0
outputData(7) = 1
result = YdciDioOutput(id, outputData, 0, 8)
```

VB6.0/VBA

```
Dim lngResult As Long
Dim bytData(7) As Byte
bytData(0) = 0
bytData(1) = 1
bytData(2) = 0
bytData(3) = 1
bytData(4) = 2
bytData(5) = 2
bytData(6) = 0
bytData(7) = 1
lngResult = YdciDioOutput(intID, bytData(0), 0, 8)
```

GCC

```
int32_t result;
uint8_t output_data[8];
output_data[0] = 0;
output_data[1] = 1;
output_data[2] = 0;
output_data[3] = 1;
output_data[4] = 2;
output_data[5] = 2;
output_data[6] = 0;
output_data[7] = 1;
result = YdciDioOutput(id, output_data, 0, 8);
```

YdciDioOutputStatus

機能

任意の点数のデジタル出力状態を読み込みます。
現在のデジタル出力の状態を知りたい場合に使用します。

適用

DIO-8/8B-UBC、DIO-8/8B-UBT

書式

```
INT YdciDioOutputStatus(
    WORD wID,
    PBYTE pbyData,
    WORD wStart,
    WORD wCount
);
```

パラメータ

wID

YdciOpen関数で取得したIDを指定します。

言語	C/C++	C++/CLI	C#	VB（.NET2002以降）	VB6.0/VBA	GCC
型	WORD	unsigned short	ushort	Short	Integer	uint16_t

pbyData

出力データを格納するバッファへのポインタを指定します。
関数が正常に実行されると、出力データが格納されます。

出力データ	状態
0	OFF
1	ON

言語	C/C++	C++/CLI	C#	VB（.NET2002以降）	VB6.0/VBA	GCC
型	PBYTE	unsigned char*	byte	Byte	Byte	uint8_t*

wStart

出力の読み込みを開始する出力番号（0～）を指定します。

言語	C/C++	C++/CLI	C#	VB（.NET2002以降）	VB6.0/VBA	GCC
型	WORD	unsigned short	ushort	Short	Integer	uint16_t

wCount

出力の読み込みをおこなう出力点数を指定します。

言語	C/C++	C++/CLI	C#	VB（.NET2002以降）	VB6.0/VBA	GCC
型	WORD	unsigned short	ushort	Short	Integer	uint16_t

戻り値

関数が正常に終了した場合は0（YDCI_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は0以外が返りますので、その場合はエラーコードを参照してください。

言語	C/C++	C++/CLI	C#	VB（.NET2002以降）	VB6.0/VBA	GCC
型	INT	int	int	Integer	Long	int32_t

使用例

- 例1
  OUT0からOUT7の出力状態を読み込みます。データはOUT0から順にデータバッファへ格納されます。
- 例2
  OUT5からOUT7の出力状態を読み込みます。データはOUT5から順にデータバッファへ格納されます。

C/C++

```
// 例1
int nResult;
BYTE abyData[8];
nResult = YdciDioOutputStatus(wID, abyData, 0, 8);

// 例2
int nResult;
BYTE abyData[3];
nResult = YdciDioOutputStatus(wID, abyData, 5, 3);
```

C++/CLI

```
// 例1
int result;
unsigned char outputStatus[8];
```

```
result = YdciDioOutputStatus(id, outputStatus, 0, 8);

// 例2
int result;
unsigned char outputStatus[3];
result = YdciDioOutputStatus(id, outputStatus, 5, 3);
```

C#

```
// 例1
int result;
byte[] outputStatus = new byte[8];
result = Ydci.DioOutputStatus(id, outputStatus, 0, 8);

// 例2
int result;
byte[] outputStatus = new byte[3];
result = Ydci.DioOutputStatus(id, outputStatus, 5, 3);
```

VB（.NET2002以降）

```
' 例1
Dim result As Integer
Dim outputStatus(7) As Byte
result = YdciDioOutputStatus(id, outputStatus, 0, 8)

' 例2
Dim result As Integer
Dim outputStatus(2) As Byte
result = YdciDioOutputStatus(id, outputStatus, 5, 3)
```

VB6.0/VBA

```
' 例1
Dim lngResult As Long
Dim bytData(7) As Byte
lngResult = YdciDioOutputStatus(intID, bytData(0), 0, 8)

' 例2
Dim lngResult As Long
Dim bytData(2) As Byte
lngResult = YdciDioOutputStatus(intID, bytData(0), 5, 3)
```

GCC

```
// 例1
int32_t result;
uint8_t output_status[8];
result = YdciDioOutputStatus(id, output_status, 0, 8);

// 例2
int32_t result;
uint8_t output_status[3];
result = YdciDioOutputStatus(id, output_status, 5, 3);
```

関数一覧

関数	機能
YdciRlyOutput	リレー出力の制御をおこないます
YdciRlyOutputStatus	リレー出力状態の読み込みをおこないます

YdciRlyOutput

機能

任意の点数のリレー出力を制御します。

適用

RLY-P4/2/0B-UBT

書式

```
INT YdciRlyOutput(
    WORD wID,
    PBYTE pbyData,
    WORD wStart,
    WORD wCount
);
```

パラメータ

wID

YdciOpen関数で取得したIDを指定します。

言語	C/C++	C++/CLI	C#	VB（.NET2002以降）	VB6.0/VBA	GCC
型	WORD	unsigned short	ushort	Short	Integer	uint16_t

pbyData

出力するデータを格納したバッファへのポインタを指定します。

出力データ	状態
0	OFF
1	ON
2	現在の状態を保持

言語	C/C++	C++/CLI	C#	VB（.NET2002以降）	VB6.0/VBA	GCC
型	PBYTE	unsigned char*	byte	Byte	Byte	uint8_t*

wStart

出力を開始するリレー番号（1～）から1引いた値（0～）を指定します。

言語	C/C++	C++/CLI	C#	VB（.NET2002以降）	VB6.0/VBA	GCC
型	WORD	unsigned short	ushort	Short	Integer	uint16_t

wCount

出力するリレーの数を指定します。

言語	C/C++	C++/CLI	C#	VB（.NET2002以降）	VB6.0/VBA	GCC
型	WORD	unsigned short	ushort	Short	Integer	uint16_t

戻り値

関数が正常に終了した場合は0（YDCI_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は0以外が返りますので、その場合はエラーコードを参照してください。

言語	C/C++	C++/CLI	C#	VB（.NET2002以降）	VB6.0/VBA	GCC
型	INT	int	int	Integer	Long	int32_t

備考

PCがスタンバイモードになると、全ての出力はOFFになります。
その後スタンバイモードから復帰した場合でも元の出力状態には戻らず、OFFのままです。
出力状態を常に保持する必要がある場合はスタンバイモードにならないようにしてください。

使用例

RY1をOFF、RY2をON、RY3をOFF、RY4を現在の状態にする。

C/C++

```
int nResult;
BYTE abyData[4];
abyData[0] = 0;
abyData[1] = 1;
abyData[2] = 0;
abyData[3] = 2;
nResult = YdciRlyOutput(wID, abyData, 0, 4);
```

C++/CLI

```
int result;
unsigned char outputData[4];
outputData[0] = 0;
outputData[1] = 1;
outputData[2] = 0;
outputData[3] = 2;
result = YdciRlyOutput(id, outputData, 0, 4);
```

C#

```
int result;
byte[] outputData = new byte[4];
outputData[0] = 0;
outputData[1] = 1;
outputData[2] = 0;
outputData[3] = 2;
result = Ydci.RlyOutput(id, outputData, 0, 4);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim outputData(3) As Byte
outputData(0) = 0
outputData(1) = 1
outputData(2) = 0
outputData(3) = 2
result = YdciRlyOutput(id, outputData, 0, 4)
```

VB6.0/VBA

```
Dim lngResult As Long
Dim bytData(3) As Byte
bytData(0) = 0
bytData(1) = 1
bytData(2) = 0
bytData(3) = 2
lngResult = YdciRlyOutput(intID, bytData(0), 0, 4)
```

GCC

```
int32_t result;
uint8_t output_data[4];
output_data[0] = 0;
output_data[1] = 1;
output_data[2] = 0;
output_data[3] = 2;
result = YdciRlyOutput(id, output_data, 0, 4);
```

YdciRlyOutputStatus

機能

任意の点数のリレー出力状態を読み込みます。
現在のリレー出力の状態を知りたい場合に使用します。

適用

RLY-P4/2/0B-UBT

書式

```
INT YdciRlyOutputStatus(
    WORD wID,
    PBYTE pbyData,
    WORD wStart,
    WORD wCount
);
```

パラメータ

wID

YdciOpen関数で取得したIDを指定します。

言語	C/C++	C++/CLI	C#	VB（.NET2002以降）	VB6.0/VBA	GCC
型	WORD	unsigned short	ushort	Short	Integer	uint16_t

pbyData

出力データを格納するバッファへのポインタを指定します。
関数が正常に実行されると、出力データが格納されます。

出力データ	状態
0	OFF
1	ON

言語	C/C++	C++/CLI	C#	VB（.NET2002以降）	VB6.0/VBA	GCC
型	PBYTE	unsigned char*	byte	Byte	Byte	uint8_t*

wStart

出力の読み込みを開始するリレー番号（1～）から1引いた値（0～）を指定します。

言語	C/C++	C++/CLI	C#	VB（.NET2002以降）	VB6.0/VBA	GCC
型	WORD	unsigned short	ushort	Short	Integer	uint16_t

wCount

出力の読み込みをおこなうリレーの数を指定します。

言語	C/C++	C++/CLI	C#	VB（.NET2002以降）	VB6.0/VBA	GCC
型	WORD	unsigned short	ushort	Short	Integer	uint16_t

戻り値

関数が正常に終了した場合は0（YDCI_RESULT_SUCCESS）が返ります。
正常に終了しなかった場合は0以外が返りますので、その場合はエラーコードを参照してください。

言語	C/C++	C++/CLI	C#	VB（.NET2002以降）	VB6.0/VBA	GCC
型	INT	int	int	Integer	Long	int32_t

使用例

- 例1
  RY1からRY4の出力状態を読み込みます。データはRY1から順にデータバッファへ格納されます。
- 例2
  RY3からRY4の出力状態を読み込みます。データはRY3から順にデータバッファへ格納されます。

C/C++

```
// 例1
int nResult;
BYTE abyData[4];
nResult = YdciRlyOutputStatus(wID, abyData, 0, 4);

// 例2
int nResult;
BYTE abyData[2];
nResult = YdciRlyOutputStatus(wID, abyData, 2, 2);
```

C++/CLI

```
// 例1
int result;
```

```
unsigned char outputStatus[4];
result = YdciRlyOutputStatus(id, outputStatus, 0, 4);

// 例2
int result;
unsigned char outputStatus[2];
result = YdciRlyOutputStatus(id, outputStatus, 2, 2);
```

C#

```
// 例1
int result;
byte[] outputStatus = new byte[4];
result = Ydci.RlyOutputStatus(id, outputStatus, 0, 4);

// 例2
int result;
byte[] outputStatus = new byte[2];
result = Ydci.RlyOutputStatus(id, outputStatus, 2, 2);
```

VB（.NET2002以降）

```
' 例1
Dim result As Integer
Dim outputStatus(3) As Byte
result = YdciRlyOutputStatus(id, outputStatus, 0, 4)

' 例2
Dim result As Integer
Dim outputStatus(1) As Byte
result = YdciRlyOutputStatus(id, outputStatus, 2, 2)
```

VB6.0/VBA

```
' 例1
Dim lngResult As Long
Dim bytData(3) As Byte
lngResult = YdciRlyOutputStatus(intID, bytData(0), 0, 4)

' 例2
Dim lngResult As Long
Dim bytData(1) As Byte
lngResult = YdciRlyOutputStatus(intID, bytData(0), 2, 2)
```

GCC

```
// 例1
int32_t result;
uint8_t output_status[4];
result = YdciRlyOutputStatus(id, output_status, 0, 4);

// 例2
int32_t result;
uint8_t output_status[2];
result = YdciRlyOutputStatus(id, output_status, 2, 2);
```

アクセス時間

デジタル入出力及びリレー出力に要する時間は以下の通りです。
（コンピュータの性能やCPUの負荷状況によって変わります）

デジタル入出力

入力関数（YdciDioInput）及び出力関数（YdciDioOutput）を10,000回実行した場合の1回当たりの平均所要時間は以下の通りです。

> 入力関数と出力関数の平均所要時間は同じです

USB2.0

入力または出力点数	10,000回実行時間	1回当たりの実行時間	使用コンピュータ
1	1,010msec	101μsec	A
1	2,510～2,530msec	251～253μsec	B
1	5,000msec	500μsec	C
1	1,280msec	128μsec	D
8	1,010msec	101μsec	A
8	2,510～2,530msec	251～253μsec	B
8	5,000msec	500μsec	C
8	1,280msec	128μsec	D

USB1.1（USB1.1ハブ使用）

入力または出力点数	10,000回実行時間	1回当たりの実行時間	使用コンピュータ
1	30,000msec	3msec	C
8	30,000msec	3msec	C

使用コンピュータ

使用コンピュータ	OS	CPUまたはボード
A	Windows 10	Intel Core i7-9700 3.00GHz
B	Windows 7	Intel Core 2 Quad Q8400 2.66GHz
C	Windows XP	Intel CeleronM 1.5GHz
D	Raspberry Pi OS Buster	Raspberry Pi 4

リレー出力

出力関数（YdciRlyOutput）を10000回実行した場合の1回当たりの平均所要時間は以下の通りです。

> ⚠ 関数実行後、リレーの動作時間（10msec以下）後にリレーが動作します

USB2.0

出力リレー数	10,000回実行時間	1回当たりの実行時間	使用コンピュータ
1	1,010msec	101μsec	A
1	2,510～2,530msec	251～253μsec	B
1	5,000msec	500μsec	C
1	1,280msec	128μsec	D
4	1,010msec	101μsec	A
4	2,510～2,530msec	251～253μsec	B
4	5,000msec	500μsec	C
4	1,280msec	128μsec	D

USB1.1（USB1.1ハブ使用）

出力リレー数	10,000回実行時間	1回当たりの実行時間	使用コンピュータ
1	30,000msec	3msec	C

出力リレー数	10,000回実行時間	1回当たりの実行時間	使用コンピュータ
4	30,000msec	3msec	C

使用コンピュータ

使用コンピュータ	OS	CPUまたはボード
A	Windows 10	Intel Core i7-9700 3.00GHz
B	Windows 7	Intel Core 2 Quad Q8400 2.66GHz
C	Windows XP	Intel CeleronM 1.5GHz
D	Raspberry Pi OS Buster	Raspberry Pi 4

C/C++

開発環境の設定（Windows）

Visual C++ .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
   YdciApi.h
   Ydci.lib
2. YdciApi.hをプロジェクトに追加します
3. Ydci.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Ydci.libと入力します。
4. ソースファイルにYdciApi.hをインクルードします
   （下記プログラム例を参照して下さい）

Visual C++ 6.0

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
   YdciApi.h
   Ydci.lib
2. YdciApi.h, Ydci.libをプロジェクトに追加します
3. ソースファイルにYdciApi.hをインクルードします
   （下記プログラム例を参照して下さい）

プログラム例（Windows）

```
#include <windows.h>
#include <stdio.h>
#include "YdciApi.h"

void main()
{
    int nResult;
    WORD wID;
    BYTE abyInData[8];
    BYTE abyOutData[8];
    BOOL bResult;
    int i;

    // ボード識別スイッチが0のボードをオープンします
    nResult = YdciOpen(0, "DIO-8/8B-UBT", &wID);
    if(nResult != YDCI_RESULT_SUCCESS){
        printf("オープンできません\n");
        return;
    }

    // IN0～7の入力をおこないます
    nResult = YdciDioInput(wID, abyInData, 0, 8);
    // 入力データの表示
```

```
    for(i = 0; i < 8; i++){
        printf("IN%u : %u\n", i, abyInData[i]);
    }

    // OUT0～7の出力をONします
    for(i = 0; i < 8; i++){
        abyOutData[i] = 1;
    }
    nResult = YdciDioOutput(wID, abyOutData, 0, 8);

    // ボードをクローズします
    bResult = YdciClose(wID);
}
```

C++/CLI

開発環境の設定

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
   YdciApiCLI.h
2. YdciApiCLI.hをプロジェクトに追加します
3. ソースファイルにYdciApiCLI.hをインクルードします（#include "YdciApiCLI.h"）
4. usingディレクティブを使ってYdciCLIを宣言します（using namespace YdciCLI;）

プログラム例

```
#include "stdafx.h"
#include "YdciApiCLI.h"

using namespace System;
using namespace YdciCLI;

int main(array<System::String ^> ^args)
{
    int result;
    unsigned short id;
    unsigned char inputData[8];
    unsigned char outputData[8];
    int i;

    // ボード識別スイッチが0のボードをオープンします
    result = YdciOpen(0, "DIO-8/8B-UBT", &id);
    if (result != YDCI_RESULT_SUCCESS) {
        Console::WriteLine("オープンできません");
        return -1;
    }

    // IN0～7の入力をおこないます
    result = YdciDioInput(id, inputData, 0, 8);
    for (i = 0; i < 8; i++) {
        Console::WriteLine("IN{0:D} : {1:D}", i, inputData[i]);
    }

    // OUT0～7の出力をONします
    for (i = 0; i < 8; i++) {
        outputData[i] = 1;
    }
    result = YdciDioOutput(id, outputData, 0, 8);

    // ボードをクローズします
    YdciClose(id);

    return 0;
}
```

C#

開発環境の設定

プロジェクトにドライバへの参照を追加します。

- Nugetからインストールする場合
  - Y2.Ub.YdciCs をインストールします。
- ソフトウェアパックから追加する場合
  - 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
    - Ydci.cs
  - Ydci.csをプロジェクトに追加します

ソースファイルにusing ディレクティブを使ってYdciCsを宣言します

```
using YdciCs;
```

プログラム例

```
using YdciCs;

static void Main()
{
    int result;
    ushort id;
    byte[] inputData = new byte[8];
    byte[] outputData = new byte[8];
    int i;

    // ボード識別スイッチが0のボードをオープンします
    result = Ydci.Open(0, "DIO-8/8B-UBT", out id);
    if(result != Ydci.YDCI_RESULT_SUCCESS)
    {
        Console.WriteLine("オープンできません");
        return;
    }

    // IN0～7の入力をおこないます
    result = Ydci.DioInput(id, inputData, 0, 8);
    for(i = 0; i < 8; i++)
    {
        Console.WriteLine("IN{0:D} : {1:D}", i, inputData[i]);
    }

    // OUT0～7の出力をONします
    for(i = 0; i < 8; i++)
    {
        outputData[i] = 1;
    }
    result = Ydci.DioOutput(id, outputData, 0, 8);

    // ボードをクローズします
```

```
    Ydci.Close(id);
}
```

Visual Basic（.NET2002以降）

開発環境の設定

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
   YdciApi.vb
2. YdciApi.vbをプロジェクトに追加します

プログラム例

```
Dim result As Integer
Dim id As Short
Dim inputData(7) As Byte
Dim outputData(7) As Byte
Dim msgText As String
Dim i As Integer

'ボード識別スイッチが0のボードをオープンします
result = YdciOpen(0, "DIO-8/8B-UBT", id)
If result <> YDCI_RESULT_SUCCESS Then
    MessageBox.Show("オープンできません", "", MessageBoxButtons.OK, MessageBoxIcon.Stop)
    Exit Sub
End If

'IN0～7の入力をおこないます
result = YdciDioInput(id, inputData, 0, 8)
For i = 0 To 7
    msgText = msgText & "IN" & i & " : " & inputData(i) & ControlChars.CrLf
Next
MessageBox.Show(msgText)

'OUT0～7の出力をONします
For i = 0 To 7
    outputData(i) = 1
Next
result = YdciDioOutput(id, outputData, 0, 8)

'ボードをクローズします
YdciClose(id)
```

Visual Basic 6.0/VBA

開発環境の設定

VB6.0

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
   YdciApi.bas
2. YdciApi.basをプロジェクトに追加します

VBA

1. YdciApi.basをインポートします

プログラム例

```
Dim lngResult As Long
Dim strModelName As String
Dim intID As Integer
Dim bytInData(0 To 7) As Byte
Dim bytOutData(0 To 7) As Byte
Dim blnResult As Boolean
Dim strInData As String
Dim i As Integer

'ボード識別スイッチが0のボードをオープンします
strModelName = "DIO-8/8B-UBT"
lngResult = YdciOpen(0, strModelName, intID)
If lngResult <> YDCI_RESULT_SUCCESS Then
    MsgBox "オープンできません", vbInformation
    Exit Sub
End If

'IN0～7の入力をおこないます
lngResult = YdciDioInput(intID, bytInData(0), 0, 8)
'入力データの表示
For i = 0 To 7
    strInData = strInData & "IN" & i & " : " & bytInData(i) & vbCrLf
Next
MsgBox strInData, vbInformation

'OUT0～7の出力をONします
For i = 0 To 7
    bytOutData(i) = 1
Next
lngResult = YdciDioOutput(intID, bytOutData(0), 0, 8)

'ボードをクローズします
blnResult = YdciClose(intID)
```

GCC

開発環境の設定

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（GCCサンプルに含まれています）
   YdciApi.h
2. ソースファイルにYdciApi.hをインクルードします（下記プログラム例を参照して下さい）
3. リンク時はライブラリとリンクするために以下を追加してください（GCCサンプルのmakefileも参考にしてください）

```
-L/usr/lib/y2c -lydci
```

プログラム例

```
#include <iostream>
#include "YdciApi.h"

int main()
{
    // ボード識別スイッチが0のボードをオープンします
    uint16_t id;
    int32_t result = YdciOpen(0, "DIO-8/8B-UBT", &id);
    if (result != YDCI_RESULT_SUCCESS)
    {
        std::cout << "オープンできません" << std::endl;
        return 1;
    }

    // IN0～7の入力をおこないます
    uint8_t input_data[8];
    result = YdciDioInput(id, input_data, 0, 8);
    // 入力データの表示
    for (uint32_t i = 0; i < 8; i++)
    {
        std::cout << "IN" << std::dec << i << ": " << (uint16_t)input_data[i] << std::endl;
    }

    // OUT0～7の出力をONします
    uint8_t output_data[8];
    for (uint32_t i = 0; i < 8; i++)
    {
        output_data[i] = 1;
    }
    result = YdciDioOutput(id, output_data, 0, 8);

    // ボードをクローズします
    result = YdciClose(id);
    return 0;
}
```

C/C++

開発環境の設定（Windows）

Visual C++ .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
   YdciApi.h
2. YdciApi.hをプロジェクトに追加します
3. Ydci.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Ydci.libと入力します。
4. ソースファイルにYdciApi.hをインクルードします
   （下記プログラム例を参照して下さい）

Visual C++ 6.0

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
   YdciApi.h
   Ydci.lib
2. YdciApi.h, Ydci.libをプロジェクトに追加します
3. ソースファイルにYdciApi.hをインクルードします
   （下記プログラム例を参照して下さい）

プログラム例（Windows）

```
#include <windows.h>
#include <stdio.h>
#include "YdciApi.h"

void main()
{
    int nResult;
    WORD wID;
    BYTE abyInData[2];
    BYTE abyOutData[4];
    BOOL bResult;
    int i;

    // ボード識別スイッチが0のボードをオープンします
    nResult = YdciOpen(0, "RLY-P4/2/0B-UBT", &wID);
    if(nResult != YDCI_RESULT_SUCCESS){
        printf("オープンできません\n");
        return;
    }

    // IN0～1の入力をおこないます
    nResult = YdciDioInput(wID, abyInData, 0, 2);
    // 入力データの表示
    for(i = 0; i < 2; i++){
```

```
        printf("IN%u : %u\n", i, abyInData[i]);
    }

    // RY1～4のリレー出力をONします
    for(i = 0; i < 4; i++){
        abyOutData[i] = 1;
    }
    nResult = YdciRlyOutput(wID, abyOutData, 0, 4);

    // ボードをクローズします
    bResult = YdciClose(wID);
}
```

C++/CLI

開発環境の設定

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
   YdciApiCLI.h
2. YdciApiCLI.hをプロジェクトに追加します
3. ソースファイルにYdciApiCLI.hをインクルードします（#include "YdciApiCLI.h"）
4. usingディレクティブを使ってYdciCLIを宣言します（using namespace YdciCLI;）

プログラム例

```
#include "stdafx.h"
#include "YdciApiCLI.h"

using namespace System;
using namespace YdciCLI;

int main(array<System::String ^> ^args)
{
    int result;
    unsigned short id;
    unsigned char inputData[2];
    unsigned char outputData[4];
    int i;

    // ボード識別スイッチが0のボードをオープンします
    result = YdciOpen(0, "RLY-P4/2/0B-UBT", &id);
    if (result != YDCI_RESULT_SUCCESS) {
        Console::WriteLine("オープンできません");
        return -1;
    }

    // IN0～1の入力をおこないます
    result = YdciDioInput(id, inputData, 0, 2);
    for (i = 0; i < 2; i++) {
        Console::WriteLine("IN{0:D} : {1:D}", i, inputData[i]);
    }

    // RY1～4のリレー出力をONします
    for (i = 0; i < 4; i++) {
        outputData[i] = 1;
    }
    result = YdciRlyOutput(id, outputData, 0, 4);

    // ボードをクローズします
    YdciClose(id);

    return 0;
}
```

C#

開発環境の設定

プロジェクトにドライバへの参照を追加します。

- Nugetからインストールする場合
  - Y2.Ub.YdciCs をインストールします。
- ソフトウェアパックから追加する場合
  - 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
    - Ydci.cs
  - Ydci.csをプロジェクトに追加します

ソースファイルにusing ディレクティブを使ってYdciCsを宣言します

```
using YdciCs;
```

プログラム例

```
using YdciCs;

static void Main()
{
    int result;
    ushort id;
    byte[] inputData = new byte[2];
    byte[] outputData = new byte[4];
    int i;

    // ボード識別スイッチが0のボードをオープンします
    result = Ydci.Open(0, "RLY-P4/2/0B-UBT", out id);
    if(result != Ydci.YDCI_RESULT_SUCCESS)
    {
        Console.WriteLine("オープンできません");
        return;
    }

    // IN0～1の入力をおこないます
    result = Ydci.DioInput(id, inputData, 0, 2);
    for(i = 0; i < 2; i++)
    {
        Console.WriteLine("IN{0:D} : {1:D}", i, inputData[i]);
    }

    // RY1～4のリレー出力をONします
    for(i = 0; i < 4; i++)
    {
        outputData[i] = 1;
    }
    result = Ydci.RlyOutput(id, outputData, 0, 4);

    // ボードをクローズします
```

```
    Ydci.Close(id);
}
```

Visual Basic（.NET2002以降）

開発環境の設定

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
   YdciApi.vb
2. YdciApi.vbをプロジェクトに追加します

プログラム例

```
Dim result As Integer
Dim id As Short
Dim inputData(1) As Byte
Dim outputData(3) As Byte
Dim msgText As String
Dim i As Integer

'ボード識別スイッチが0のボードをオープンします
result = YdciOpen(0, "RLY-P4/2/0B-UBT", id)
If result <> YDCI_RESULT_SUCCESS Then
    MessageBox.Show("オープンできません", "", MessageBoxButtons.OK, MessageBoxIcon.Stop)
    Exit Sub
End If

'IN0～1の入力をおこないます
result = YdciDioInput(id, inputData, 0, 2)
For i = 0 To 1
    msgText = msgText & "IN" & i & " : " & inputData(i) & ControlChars.CrLf
Next
MessageBox.Show(msgText)

'RY1～4の出力をONします
For i = 0 To 3
    outputData(i) = 1
Next
result = YdciRlyOutput(id, outputData, 0, 4)

'ボードをクローズします
YdciClose(id)
```

Visual Basic 6.0/VBA

開発環境の設定

VB6.0

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
   YdciApi.bas
2. YdciApi.basをプロジェクトに追加します

VBA

1. YdciApi.basをインポートします

プログラム例

```
Dim lngResult As Long
Dim strModelName As String
Dim intID As Integer
Dim bytInData(0 To 1) As Byte
Dim bytOutData(0 To 3) As Byte
Dim blnResult As Boolean
Dim strInData As String
Dim i As Integer

'ボード識別スイッチが0のボードをオープンします
strModelName = "RLY-P4/2/0B-UBT"
lngResult = YdciOpen(0, strModelName, intID)
If lngResult <> YDCI_RESULT_SUCCESS Then
    MsgBox "オープンできません", vbInformation
    Exit Sub
End If

'IN0～1の入力をおこないます
lngResult = YdciDioInput(intID, bytInData(0), 0, 2)
'入力データの表示
For i = 0 To 1
    strInData = strInData & "IN" & i & " : " & bytInData(i) & vbCrLf
Next
MsgBox strInData, vbInformation

'RY1～4のリレー出力をONします
For i = 0 To 3
    bytOutData(i) = 1
Next
lngResult = YdciRlyOutput(intID, bytOutData(0), 0, 4)

'ボードをクローズします
blnResult = YdciClose(intID)
```

GCC

開発環境の設定

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（GCCサンプルに含まれています）
   YdciApi.h
2. ソースファイルにYdciApi.hをインクルードします（下記プログラム例を参照して下さい）
3. リンク時はライブラリとリンクするために以下を追加してください（GCCサンプルのmakefileも参考にしてください）

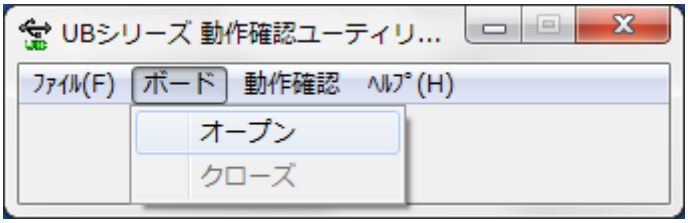
```
-L/usr/lib/y2c -lydci
```

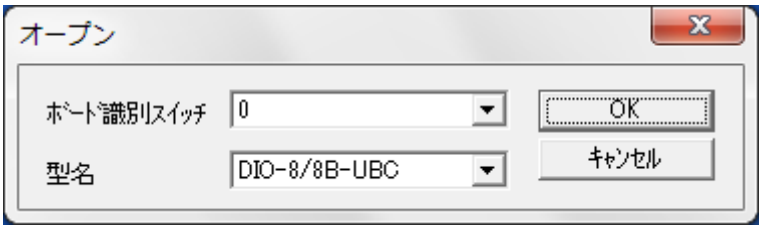
プログラム例

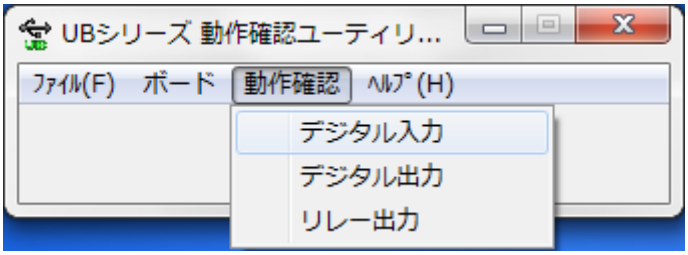
```
#include <iostream>
#include "YdciApi.h"

int main()
{
    // ボード識別スイッチが0のボードをオープンします
    uint16_t id;
    int32_t result = YdciOpen(0, "RLY-P4/2/0B-UBT", &id);
    if (result != YDCI_RESULT_SUCCESS)
    {
        std::cout << "オープンできません" << std::endl;
        return 1;
    }

    // IN0～1の入力をおこないます
    uint8_t input_data[2];
    result = YdciDioInput(id, input_data, 0, 2);
    // 入力データの表示
    for (uint32_t i = 0; i < 2; i++)
    {
        std::cout << "IN" << std::dec << i << ": " << (uint16_t)input_data[i] << std::endl;
    }

    // RY1～4のリレー出力をONします
    uint8_t output_data[4];
    for (uint32_t i = 0; i < 4; i++)
    {
        output_data[i] = 1;
    }
    result = YdciRlyOutput(id, output_data, 0, 4);

    // ボードをクローズします
    result = YdciClose(id);
    return 0;
}
```

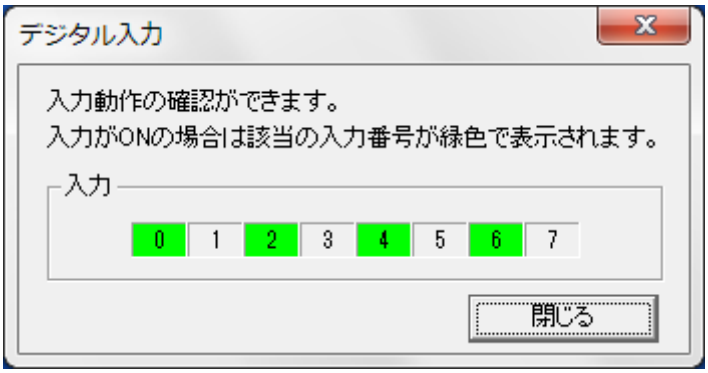
動作確認ユーティリティ（Windowsのみ）

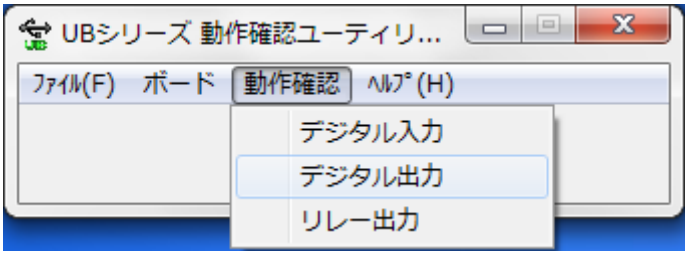
全入出力の確認ができます。
ユーティリティを起動後、以下の手順で入出力確認ができます。

1. ボードのオープン
   [ボード]メニューの[オープン]をクリックしてください。
   

   表示された画面にてボード識別スイッチ及び型名を選択してボードのオープンをおこなってください。
   

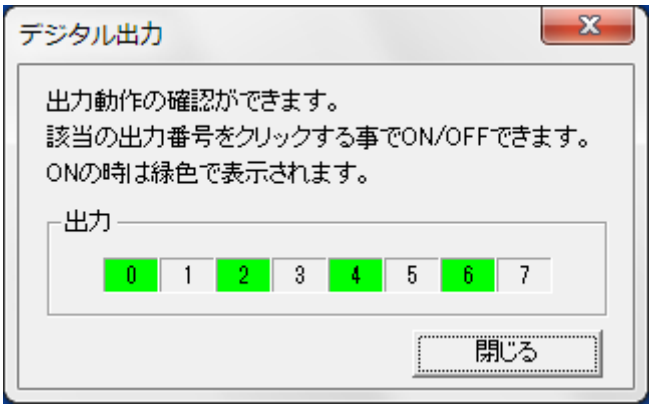
2. デジタル入力の確認
   [動作確認]メニューの[デジタル入力]をクリックしてください。
   

   入力番号の一覧が表示されており、該当の入力がONの場合は緑、OFFの場合は灰色で表示されます。
   （入力の状態を100msec間隔で監視しています）
   

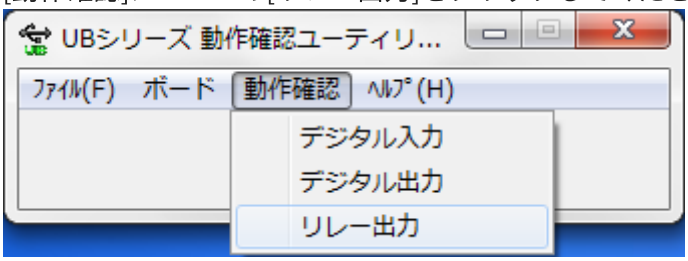
3. デジタル出力の確認
   [動作確認]メニューの[デジタル出力]をクリックしてください。
   

   出力番号の一覧が表示されていますので、出力ON/OFFを切り替えたい出力番号をクリックしてくださ

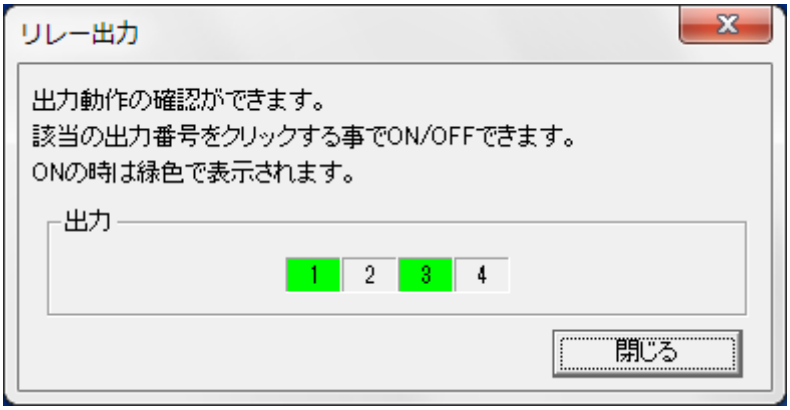
い。
ONの時には緑、OFFの時は灰色で表示されます。
（出力画面を閉じた際に全出力OFFになります）



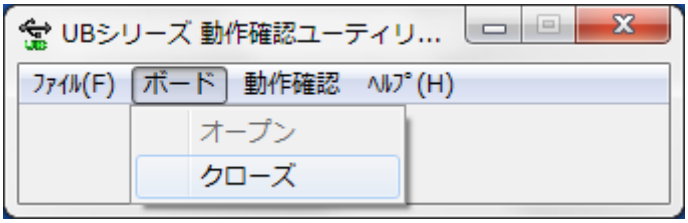
4. リレー出力の確認
[動作確認]メニューの[リレー出力]をクリックしてください。



出力番号の一覧が表示されていますので、出力ON/OFFを切り替えたい出力番号をクリックしてください。
ONの時には緑、OFFの時は灰色で表示されます。
（出力画面を閉じた際に全出力OFFになります）



5. ボードのクローズ
[ボード]メニューの[クローズ]をクリックしてください。



続けて他のボードを確認したい場合は、1から実行してください。
（ユーティリティ終了時にクローズ処理をおこなうようにしていますので、クローズを実行せずにユーティリティを終了させても問題ありません）

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。