株式会社ワイツー

USB-PC104シリーズ　リレー出力ユニット　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USB-PC104シリーズのリレー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。

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。

用語説明

ユニットID

ユニット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ユニットの最大数はUSB-PC104シリーズ全体で16台です。

注意事項

パソコン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、以降ユニットが動作しません。

動作環境（Windows）

PC

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[2]
Windows 7 x86, x64[2]
Windows Vista x86, x64[3]
Windows XP x64[4]
Windows XP[4]
Windows 2000[4]
Windows Me[5][6]
Windows 98, 98SE[5][6]

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. 2015年10月15日リリースの旧バージョンのドライバを使用します ↩↩
3. 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 ↩
4. 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 ↩↩↩
5. 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 ↩↩
6. YduDioOutputStatusとYduRlyOutputStatus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
   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）
2. ユニットをオープン
   YduOpen関数を使用してユニットをオープンします。
   YduOpen関数にて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クローズ（YduClose関数）するようにしてください。
3. リレー出力
   YduRlyOutput関数を使用して出力をおこないます。
   YduRlyOutputStatus関数を使用して出力状態を知ることもできます。
4. デジタル入出力（デジタル入出力がある場合）
   YduDioInput / YduDioOutput関数を使用して入出力をおこないます。
   YduDioOutputStatus関数を使用して出力状態を知ることもできます。
5. ユニットをクローズ
   YduClose関数を使用してユニットをクローズします。
6. 終了
   ユニット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
YDU_RESULT_SUCCESS (H'00000000 / 0)	正常終了	
YDU_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。
YDU_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ユニットに対して操作がおこなわれました。 YduOpen関数にてオープンされたユニットに対して操作をおこなってください。
YDU_RESULT_ALREADY_OPEN (H'CE000003 / -838860797)	オープン済み	既にオープンされているユニットに対してYduOpen関数が実行されました。
YDU_RESULT_INVALID_UNIT_ID (H'CE000004 / -838860796)	ユニットIDが不正	指定されたユニットIDが不正です。 IDは0～15を指定してください。
YDU_RESULT_CANNOT_OPEN (H'CE000006 / -838860794)	デバイスがオープンできない	デバイスが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. 型名指定が不正 5. デバイスドライバがインストールできていない
YDU_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。
YDU_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。
YDU_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。
YDU_RESULT_HARDWARE_ERROR (H'CE00000B / -838860789)	ハードウェアエラー	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. 外部との接続方法に問題がある 上記を確認しても問題が見当たらない場合は、ユニットのハードウェアに問題が発生している可能性があります。現象を弊社サポートまでご連絡ください。

エラーコード (16進数値/10進数値)	意味	対処方法
YDU_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。
YDU_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、Ydu・YduDio・YduRly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#
YduOpen	Ydu.Open
YduClose	Ydu.Close
YduDioOutput	YduDio.Output
YduDioInput	YduDio.Input
YduRlyOutput	YduRly.Output

関数一覧

関数	機能
YduOpen	ユニットのオープンをおこないます
YduClose	ユニットのクローズをおこないます

YduOpen

機能

ユニットのオープンをおこない、ユニットへのアクセスをおこなえるようにします。

書式

```
INT YduOpen(
    WORD wUnitID,
    LPCSTR lpszModelName,
    WORD wMode = YDU_OPEN_NORMAL
);
```

パラメータ

wUnitID

オープンするユニットのID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

オープンするユニットの型名を指定します（備考も参照してください）。

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

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
YDU_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
YDU_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

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（YDU_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

型番末尾に (35V) または (50V) が付加されている型番の場合、lpszModelNameには (35V) または (50V) を除いた型番を指定してください。

- 例

型番	lpszModelNameに指定する型番
RLY-24/16/32A-U (35V)	RLY-24/16/32A-U

> ⚠ YduOpen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YduClose関数でクローズしてください

使用例

- 例1
  IDが0に設定されているRLY-24A-Uをオープンします。リレー出力は全てOFFになります。
- 例2
  IDが0に設定されているRLY-24A-Uをオープンします。リレー出力の状態は関数実行前と変わりません。

C/C++

```
// 例1
int nResult;
nResult = YduOpen(0, "RLY-24A-U");

// 例2
int nResult;
nResult = YduOpen(0, "RLY-24A-U", YDU_OPEN_OUT_NOT_INIT);
```

C++/CLI

```
// 例1
int result;
result = YduOpen(0, "RLY-24A-U");

// 例2
int result;
result = YduOpen(0, "RLY-24A-U", YDU_OPEN_OUT_NOT_INIT);
```

C#

```
// 例1
int result;
result = Ydu.Open(0, "RLY-24A-U");

// 例2
int result;
result = Ydu.Open(0, "RLY-24A-U", Ydu.YDU_OPEN_OUT_NOT_INIT);
```

VB（.NET2002以降）

```
' 例1
Dim result As Integer
result = YduOpen(0, "RLY-24A-U")

' 例2
Dim result As Integer
result = YduOpen(0, "RLY-24A-U", YDU_OPEN_OUT_NOT_INIT)
```

VB6.0/VBA

```
' 例1
Dim lngResult As Long
Dim strModelName as String
strModelName = "RLY-24A-U"
lngResult = YduOpen(0, strModelName)

' 例2
Dim lngResult As Long
Dim strModelName as String
strModelName = "RLY-24A-U"
lngResult = YduOpen(0, strModelName, YDU_OPEN_OUT_NOT_INIT)
```

GCC

```
// 例1
int32_t result;
result = YduOpen(0, "RLY-24A-U");

// 例2
int32_t result;
result = YduOpen(0, "RLY-24A-U", YDU_OPEN_OUT_NOT_INIT);
```

YduClose

機能

ユニットのクローズをおこない、ユニットへのアクセスを禁止します。

書式

```
BOOL YduClose(
    WORD wUnitID
);
```

パラメータ

wUnitID

クローズするユニットのID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

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

> ⚠ YduOpen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YduClose関数でクローズしてください

使用例

ユニットIDが0のユニットのクローズ処理をおこないます。

C/C++

```
BOOL bResult;
bResult = YduClose(0);
```

C++/CLI

```
bool result;
result = YduClose(0);
```

C#

```
bool result;
result = Ydu.Close(0);
```

VB（.NET2002以降）

```
Dim result As Boolean
result = YduClose(0)
```

VB6.0/VBA

```
Dim blnResult As Boolean
blnResult = YduClose(0)
```

GCC

```
int32_t result;
result = YduClose(0);
```

関数一覧

関数	機能
YduRlyOutput	リレー出力の制御をおこないます
YduRlyOutputStatus	リレー出力の状態を読み込みます

YduRlyOutput

機能

任意の点数のリレー出力を制御します。

書式

```
INT YduRlyOutput(
    WORD wUnitID,
    PBYTE pbyData,
    WORD wStart,
    WORD wCount
);
```

パラメータ

wUnitID

出力をおこなうユニットのID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

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（YDU_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

ユニットIDが0のユニットの、RY1をOFF、RY2をON、RY3をOFF、RY4をON、RY5は現在の状態、RY6は現在の状態、RY7をOFF、RY8をONにする。

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 = YduRlyOutput(0, 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 = YduRlyOutput(0, 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 = YduRly.Output(0, 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 = YduRlyOutput(0, 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 = YduRlyOutput(0, 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 = YduRlyOutput(0, output_data, 0, 8);
```

YduRlyOutputStatus

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。

書式

```
INT YduRlyOutputStatus(
    WORD wUnitID,
    PBYTE pbyData,
    WORD wStart,
    WORD wCount
);
```

パラメータ

wUnitID

出力読み込みをおこなうユニットのID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

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（YDU_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
  ユニットIDが0のユニットのRY1からRY8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
  ユニットIDが0のユニットのRY6からRY8の出力状態を読み込みます。
  データはRY6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 = YduRlyOutputStatus(0, abyData, 0, 8);

// 例2
int nResult;
BYTE abyData[3];
nResult = YduRlyOutputStatus(0, abyData, 5, 3);
```

C++/CLI

```
// 例1
int result;
unsigned char outputStatus[8];
result = YduRlyOutputStatus(0, outputStatus, 0, 8);
```

```
// 例2
int result;
unsigned char outputStatus[3];
result = YduRlyOutputStatus(0, outputStatus, 5, 3);
```

C#

```
// 例1
int result;
byte[] outputStatus = new byte[8];
result = YduRly.OutputStatus(0, outputStatus, 0, 8);

// 例2
int result;
byte[] outputStatus = new byte[3];
result = YduRly.OutputStatus(0, outputStatus, 5, 3);
```

VB（.NET2002以降）

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

' 例2
Dim result As Integer
Dim outputStatus(2) As Byte
result = YduRlyOutputStatus(0, outputStatus, 5, 3)
```

VB6.0/VBA

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

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

GCC

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

// 例2
int32_t result;
uint8_t output_status[3];
result = YduRlyOutputStatus(0, output_status, 5, 3);
```

関数一覧

デジタル入出力ボードと組み合わせたユニットで使用できます。

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

YduDioInput

機能

任意の点数の入力端子の状態を読み込みます。

書式

```
INT YduDioInput(
    WORD wUnitID,
    PBYTE pbyData,
    WORD wStart,
    WORD wCount
);
```

パラメータ

wUnitID

入力読み込みをおこなうユニットのID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

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

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（YDU_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

ユニットIDが0のユニットのIN0からIN7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[8];
nResult = YduDioInput(0, abyData, 0, 8);
```

C++/CLI

```
int result;
unsigned char inputData[8];
result = YduDioInput(0, inputData, 0, 8);
```

C#

```
int result;
byte[] inputData = new byte[8];
result = YduDio.Input(0, inputData, 0, 8);
```

VB（.NET2002以降）

```
Dim result As Integer
Dim inputData(7) As Byte
result = YduDioInput(0, inputData, 0, 8)
```

VB6.0/VBA

```
Dim lngResult As Long
Dim bytData(7) As Byte
lngResult = YduDioInput(0, bytData(0), 0, 8)
```

GCC

```
int32_t result;
uint8_t input_data[8];
result = YduDioInput(0, input_data, 0, 8);
```

YduDioOutput

機能

任意の点数の出力端子を制御します。

書式

```
INT YduDioOutput(
    WORD wUnitID,
    PBYTE pbyData,
    WORD wStart,
    WORD wCount
);
```

パラメータ

wUnitID

出力をおこなうユニットのID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

pbyData

ユニットへ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

出力するデータ数を指定します。

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（YDU_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

ユニットIDが0のユニットへ、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 = YduDioOutput(0, 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 = YduDioOutput(0, 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 = YduDio.Output(0, 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 = YduDioOutput(0, 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 = YduDioOutput(0, 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 = YduDioOutput(0, output_data, 0, 8);
```

YduDioOutputStatus

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。

書式

```
INT YduDioOutputStatus(
    WORD wUnitID,
    PBYTE pbyData,
    WORD wStart,
    WORD wCount
);
```

パラメータ

wUnitID

出力読み込みをおこなうユニットのID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

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（YDU_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
  ユニットIDが0のユニットの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
  ユニットIDが0のユニットの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 = YduDioOutputStatus(0, abyData, 0, 8);

// 例2
int nResult;
BYTE abyData[3];
nResult = YduDioOutputStatus(0, abyData, 5, 3);
```

C++/CLI

```
// 例1
int result;
unsigned char outputStatus[8];
result = YduDioOutputStatus(0, outputStatus, 0, 8);
```

```
// 例2
int result;
unsigned char outputStatus[3];
result = YduDioOutputStatus(0, outputStatus, 5, 3);
```

C#

```
// 例1
int result;
byte[] outputStatus = new byte[8];
result = YduDio.OutputStatus(0, outputStatus, 0, 8);

// 例2
int result;
byte[] outputStatus = new byte[3];
result = YduDio.OutputStatus(0, outputStatus, 5, 3);
```

VB（.NET2002以降）

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

' 例2
Dim result As Integer
Dim outputStatus(2) As Byte
result = YduDioOutputStatus(0, outputStatus, 5, 3)
```

VB6.0/VBA

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

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

GCC

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

// 例2
int32_t result;
uint8_t output_status[3];
result = YduDioOutputStatus(0, output_status, 5, 3);
```

アクセス時間

リレー出力及びデジタル入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）

リレー出力

リレー出力関数（YduRlyOutput）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（3msec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	1010msec	101μsec	A
1	2510～2530msec	251～253μsec	B
1	5020msec	502μsec	C
1	1300msec	130μsec	D
72	2570msec	257μsec	A
72	3760～3765msec	376～376.5μsec	B
72	5020msec	502μsec	C
72	3280msec	328μsec	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	30000msec	3msec	C
72	30000msec	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

デジタル入出力

入力関数（YduDioInput）及び出力関数（YduDioOutput）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。

USB2.0

入力

入力点数	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	1010msec	101μsec	A
1	2510～2530msec	251～253μsec	B
1	5000msec	500μsec	C
1	1300msec	130μsec	D
192	7000msec	700μsec	A
192	7500～7510msec	750～751μsec	B
192	8700msec	870μsec	C
192	6950msec	695μsec	D

出力

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	1010msec	101μsec	A
1	2510～2530msec	251～253μsec	B

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	5000msec	500μsec	C
1	1300msec	130μsec	D
192	7200msec	720μsec	A
192	7500～7510msec	750～751μsec	B
192	8800msec	880μsec	C
192	7140msec	714μsec	D

USB1.1（USB1.1ハブ使用）

> 入力関数と出力関数の平均所要時間は同じです

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	30000msec	3msec	C
192	30000msec	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

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
   YduApi.h
   YduRlyApi.h
   YduResult.h
   Ydu.lib
2. YduApi.h, YduRlyApi.h, YduResult.hをプロジェクトに追加します
3. Ydu.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Ydu.libと入力します。
4. ソースファイルにYduApi.h, YduRlyApi.h, YduResult.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
   YduApi.h
   YduRlyApi.h
   YduResult.h
   Ydu.lib
2. YduApi.h, YduRlyApi.h, YduResult.h, Ydu.libをプロジェクトに追加します
3. ソースファイルにYduApi.h, YduRlyApi.h, YduResult.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

```
#include <windows.h>
#include <stdio.h>
#include "YduApi.h"
#include "YduRlyApi.h"
#include "YduResult.h"

void main()
{
    int     nResult;
    BYTE    abyOutData[24];
    BOOL    bResult;
    int     i;

    // IDが0に設定されているRLY-24A-Uをオープンします
    nResult = YduOpen(0, "RLY-24A-U");
    if(nResult != YDU_RESULT_SUCCESS){
        printf("オープンできません\n");
```

```
        return;
    }

    // RY1～24のリレーをONします
    for(i=0; i<24; i++){
        abyOutData[i]=1;
    }
    nResult = YduRlyOutput(0, abyOutData, 0, 24);

    // ユニットをクローズします
    bResult = YduClose(0);
}
```

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
   YduApiCLI.h
   YduRlyApiCLI.h
2. YduApiCLI.h, YduRlyApiCLI.hをプロジェクトに追加します
3. ソースファイルにYduApiCLI.h, YduRlyApiCLI.hをインクルードします
   （下記プログラム例を参照してください）
4. usingディレクティブを使ってYduCLIを宣言します（using namespace YduCLI;）

プログラム例

```
#include "stdafx.h"
#include "YduApiCLI.h"
#include "YduRlyApiCLI.h"

using namespace System;
using namespace YduCLI;

int main(array<System::String ^> ^args)
{
    int result;
    unsigned char outputData[24];
    int i;

    // IDが0に設定されているRLY-24A-Uをオープンします
    result = YduOpen(0, "RLY-24A-U");
    if (result != YDU_RESULT_SUCCESS) {
        Console::WriteLine("オープンできません");
        return -1;
    }

    // RY1～24のリレーをONします
    for (i = 0; i < 24; i++) {
        outputData[i] = 1;
    }
    result = YduRlyOutput(0, outputData, 0, 24);

    return 0;
}
```

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.UsbPc104.YduCs 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
    - Ydu.cs
    - YduRly.cs
  - Ydu.cs, YduRly.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YduCsを宣言します

```
using YduCs;
```

プログラム例

```
using YduCs;

static void Main()
{
    int result;
    byte[] outputData = new byte[24];
    int i;

    // IDが0に設定されているRLY-24A-Uをオープンします
    result = Ydu.Open(0, "RLY-24A-U");
    if(result != Ydu.YDU_RESULT_SUCCESS)
    {
        Console.WriteLine("オープンできません");
        return;
    }

    // RY1～24のリレーをONします
    for(i = 0; i < 24; i++)
    {
        outputData[i] = 1;
    }
    result = YduRly.Output(0, outputData, 0, 24);

    // ボードをクローズします
    Ydu.Close(0);
}
```

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
   YduApi.vb
   YduRlyApi.vb
   YduResult.vb
2. YduApi.vb, YduRlyApi.vb, YduResult.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 outputData(23) As Byte
Dim i As Integer

'IDが0に設定されているRLY-24A-Uをオープンします
result = YduOpen(0, "RLY-24A-U")
If result <> YDU_RESULT_SUCCESS Then
    MessageBox.Show("オープンできません", "", MessageBoxButtons.OK, MessageBoxIcon.Stop)
    Exit Sub
End If

'RY1～24のリレーをONします
For i = 0 To 23
    outputData(i) = 1
Next
result = YduRlyOutput(0, outputData, 0, 24)

'ユニットをクローズします
YduClose(0)
```

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
   YduBaseApi.bas
   YduRlyApi.bas
2. YduBaseApi.bas, YduRlyApi.basをプロジェクトに追加します

VBA

1. 以下のファイルをインポートします
   YduBaseApi.bas
   YduRlyApi.bas

プログラム例

```
Dim lngResult As Long
Dim strModelName As String
Dim bytOutData(0 To 23) As Byte
Dim blnResult As Boolean
Dim i As Integer

'IDが0に設定されているRLY-24A-Uをオープンします
strModelName = "RLY-24A-U"
lngResult = YduOpen(0, strModelName)
If lngResult <> YDU_RESULT_SUCCESS Then
    MsgBox "オープンできません", vbInformation
    Exit Sub
End If

'RY1～24のリレーをONします
For i = 0 To 23
    bytOutData(i) = 1
Next
lngResult = YduRlyOutput(0, bytOutData(0), 0, 24)

'ユニットをクローズします
blnResult = YduClose(0)
```

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）
   YduApi.h
   YduRlyApi.h
   YduResult.h
2. ソースファイルにYduApi.h, YduRlyApi.h, YduResult.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 -lydu
```

プログラム例

```
#include <iostream>
#include "YduApi.h"
#include "YduRlyApi.h"
#include "YduResult.h"

int main()
{
   // IDが0に設定されているRLY-24A-Uをオープンします
    uint16_t unit_id = 0;
    int32_t result = YduOpen(unit_id, "RLY-24A-U");
    if (result != YDU_RESULT_SUCCESS)
    {
        std::cout << "オープンできません" << std::endl;
        return 1;
    }

    // RY1～24のリレーをONします
    uint8_t output_data[24];
    for (uint32_t i = 0; i < 24; i++)
    {
        output_data[i] = 1;
    }
    result = YduRlyOutput(unit_id, output_data, 0, 24);

    // ユニットをクローズします
    result = YduClose(unit_id);
    return 0;
}
```

C/C++

開発環境の設定

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
   YduApi.h
   YduDioApi.h
   YduResult.h
   Ydu.lib
2. YduApi.h, YduDioApi.h, YduResult.hをプロジェクトに追加します
3. Ydu.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Ydu.libと入力します。
4. ソースファイルにYduApi.h, YduDioApi.h, YduResult.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
   YduApi.h
   YduDioApi.h
   YduResult.h
   Ydu.lib
2. YduApi.h, YduDioApi.h, YduResult.h, Ydu.libをプロジェクトに追加します
3. ソースファイルにYduApi.h, YduDioApi.h, YduResult.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

```
#include <windows.h>
#include <stdio.h>
#include "YduApi.h"
#include "YduDioApi.h"
#include "YduResult.h"

void main()
{
    int     nResult;
    BYTE    abyInData[16];
    BYTE    abyOutData[32];
    BOOL    bResult;
    int     i;

    // IDが0に設定されているRLY-8/16/32A-Uをオープンします
    nResult = YduOpen(0, "RLY-8/16/32A-U");
    if(nResult != YDU_RESULT_SUCCESS){
```

```
        printf("オープンできません\n");
        return;
    }
    // IN0～15の入力をおこないます
    nResult = YduDioInput(0, abyInData, 0, 16);
    // 入力データの表示
    for(i = 0; i < 16; i++){
        printf("IN%u : %u\n", i, abyInData[i]);
    }

    // OUT0～31の出力をONします
    memset(abyOutData, 1, 32);
    nResult = YduDioOutput(0, abyOutData, 0, 32);

    // ユニットをクローズします
    bResult = YduClose(0);
}
```

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
   YduApiCLI.h
   YduDioApiCLI.h
2. YduApiCLI.h, YduDioApiCLI.hをプロジェクトに追加します
3. ソースファイルにYduApiCLI.h, YduDioApiCLI.hをインクルードします
   （下記プログラム例を参照してください）
4. usingディレクティブを使ってYduCLIを宣言します（using namespace YduCLI;）

プログラム例

```
#include "stdafx.h"
#include "YduApiCLI.h"
#include "YduDioApiCLI.h"

using namespace System;
using namespace YduCLI;

int main(array<System::String ^> ^args)
{
    int result;
    unsigned char inputData[16];
    unsigned char outputData[32];
    int i;

    // IDが0に設定されているRLY-8/16/32A-Uをオープンします
    result = YduOpen(0, "RLY-8/16/32A-U");
    if (result != YDU_RESULT_SUCCESS) {
        Console::WriteLine("オープンできません");
        return -1;
    }

    // IN0～15の入力をおこないます
    result = YduDioInput(0, inputData, 0, 16);
    for (i = 0; i < 16; i++) {
        Console::WriteLine("IN{0:D} : {1:D}", i, inputData[i]);
    }

    // OUT0～31の出力をONします
    for (i = 0; i < 32; i++) {
        outputData[i] = 1;
    }
    result = YduDioOutput(0, outputData, 0, 32);

    // ユニットをクローズします
    YduClose(0);

    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.UsbPc104.YduCs 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
    - Ydu.cs
    - YduDio.cs
  - Ydu.cs, YduDio.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YduCsを宣言します

```
using YduCs;
```

プログラム例

```
using YduCs;

static void Main()
{
    int result;
    byte[] inputData = new byte[16];
    byte[] outputData = new byte[32];
    int i;

    // IDが0に設定されているRLY-8/16/32A-Uをオープンします
    result = Ydu.Open(0, "RLY-8/16/32A-U");
    if(result != Ydu.YDU_RESULT_SUCCESS)
    {
        Console.WriteLine("オープンできません");
        return;
    }

    // IN0～15の入力をおこないます
    result = YduDio.Input(0, inputData, 0, 16);
    for(i = 0; i < 16; i++)
    {
        Console.WriteLine("IN{0:D} : {1:D}", i, inputData[i]);
    }

    // OUT0～31の出力をONします
    for(i = 0; i < 32; i++)
    {
        outputData[i] = 1;
    }
    result = YduDio.Output(0, outputData, 0, 32);

    // ボードをクローズします
```

```
    Ydu.Close(0);
}
```

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
   YduApi.vb
   YduDioApi.vb
   YduResult.vb
2. YduApi.vb, YduDioApi.vb, YduResult.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 inputData(15) As Byte
Dim outputData(31) As Byte
Dim msgText As String
Dim i As Integer

'IDが0に設定されているRLY-8/16/32A-Uをオープンします
result = YduOpen(0, "RLY-8/16/32A-U")
If result <> YDU_RESULT_SUCCESS Then
    MessageBox.Show("オープンできません", "", MessageBoxButtons.OK, MessageBoxIcon.Stop)
    Exit Sub
End If

msgText = ""
'IN0～15の入力をおこないます
result = YduDioInput(0, inputData, 0, 16)
For i = 0 To 15
    msgText = msgText & "IN" & i & " : " & inputData(i) & ControlChars.CrLf
Next
MessageBox.Show(msgText)

'OUT0～31の出力をONします
For i = 0 To 31
    outputData(i) = 1
Next
result = YduDioOutput(0, outputData, 0, 32)

'ユニットをクローズします
YduClose(0)
```

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
   YduBaseApi.bas
   YduDioApi.bas
2. YduBaseApi.bas, YduDioApi.basをプロジェクトに追加します

VBA

1. 以下のファイルをインポートします
   YduBaseApi.bas
   YduDioApi.bas

プログラム例

```
Dim lngResult As Long
Dim strModelName As String
Dim bytInData(0 To 15) As Byte
Dim bytOutData(0 To 31) As Byte
Dim blnResult As Boolean
Dim strInData As String
Dim i As Integer

'IDが0に設定されているRLY-8/16/32A-Uをオープンします
strModelName = "RLY-8/16/32A-U"
lngResult = YduOpen(0, strModelName)
If lngResult <> YDU_RESULT_SUCCESS Then
    MsgBox "オープンできません", vbInformation
    Exit Sub
End If

'IN0～15の入力をおこないます
lngResult = YduDioInput(0, bytInData(0), 0, 16)
'入力データの表示
For i = 0 To 15
    strInData = strInData & "IN" & i & " : " & bytInData(i) & vbCrLf
Next
MsgBox strInData, vbInformation

'OUT0～31の出力をONします
For i = 0 To 31
    bytOutData(i) = 1
Next
lngResult = YduDioOutput(0, bytOutData(0), 0, 32)

'ユニットをクローズします
blnResult = YduClose(0)
```

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）
   YduApi.h
   YduDioApi.h
   YduResult.h
2. ソースファイルにYduApi.h, YduDioApi.h, YduResult.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 -lydu
```

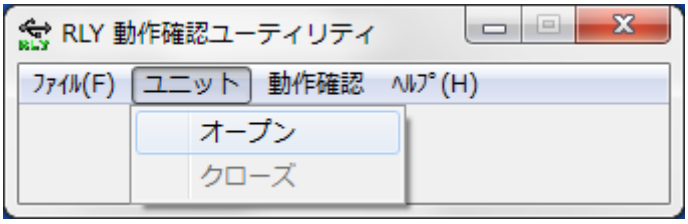
プログラム例

```
#include <iostream>
#include "YduApi.h"
#include "YduDioApi.h"
#include "YduResult.h"

int main()
{
    // IDが0に設定されているRLY-8/16/32A-Uをオープンします
    uint16_t unit_id = 0;
    int32_t result = YduOpen(unit_id, "RLY-8/16/32A-U");
    if (result != YDU_RESULT_SUCCESS)
    {
        std::cout << "オープンできません" << std::endl;
        return 1;
    }

    // IN0～15の入力をおこないます
    uint8_t input_data[16];
    result = YduDioInput(unit_id, input_data, 0, 16);
    // 入力データの表示
    for (uint32_t i = 0; i < 16; i++)
    {
        std::cout << "IN" << std::dec << i << ": " << (uint16_t)input_data[i] << std::endl;
    }

    // OUT0～31の出力をONします
    uint8_t output_data[32];
    for (uint32_t i = 0; i < 32; i++)
    {
        output_data[i] = 1;
    }
    result = YduDioOutput(unit_id, output_data, 0, 32);

    // ユニットをクローズします
    result = YduClose(unit_id);
    return 0;
}
```

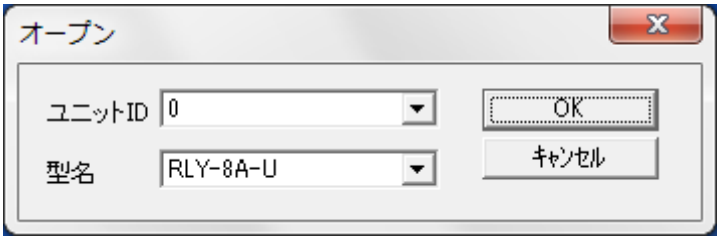
リレー出力確認ユーティリティ（Windowsのみ）

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

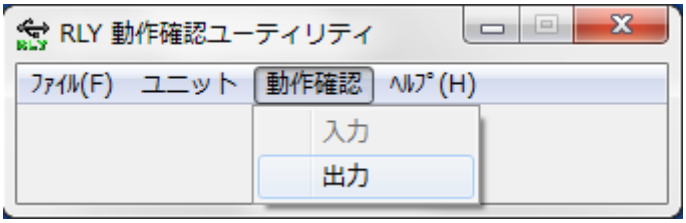
1. ユニットのオープン
   [ユニット]メニューの[オープン]をクリックしてください。



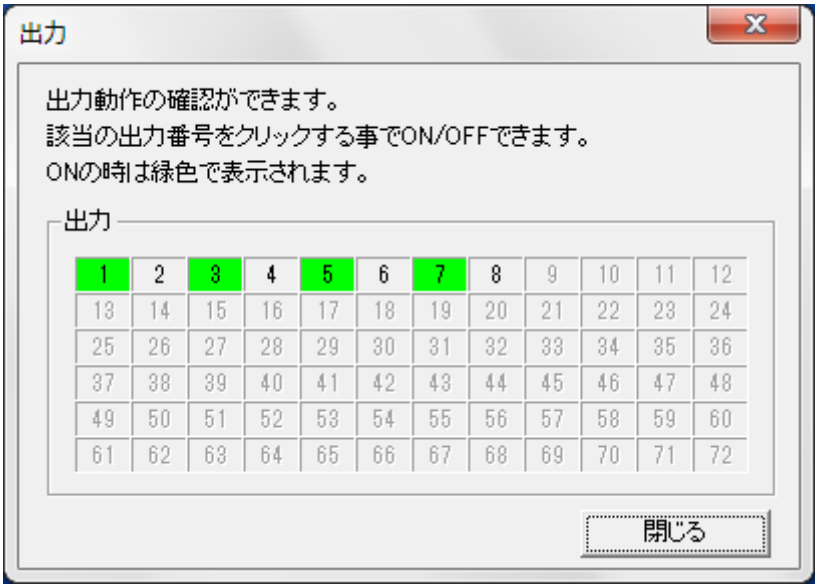
   表示された画面にてユニットID及び型名を選択してユニットのオープンをおこなってください。



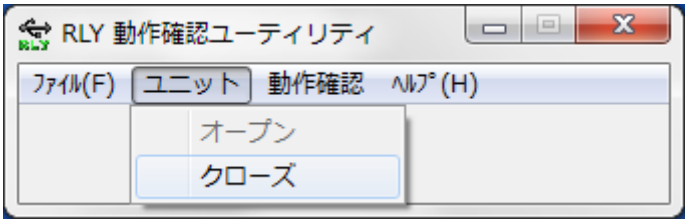
2. 出力の確認
   [動作確認]メニューの[出力]をクリックしてください。



   リレー番号の一覧が表示されていますので、ON/OFFを切り替えたいリレー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リレーOFFになります）



3. ユニットのクローズ
   [ユニット]メニューの[クローズ]をクリックしてください。

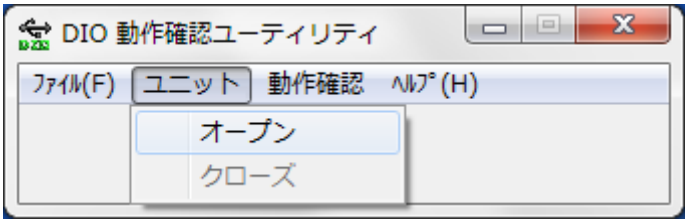


続けて他のユニットを確認したい場合は、1から実行してください。
(ユーティリティ終了時にクローズ処理をおこなうようにしていますので、クローズを実行せずにユーティリティを終了させても問題ありません)

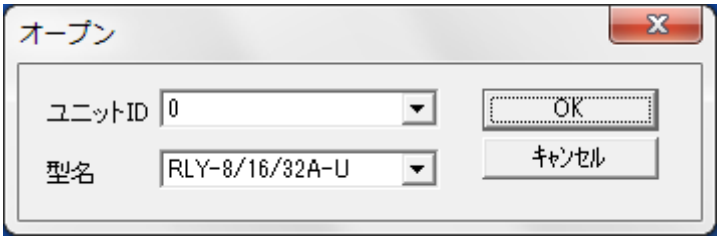
デジタル入出力確認ユーティリティ（Windowsのみ）

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。

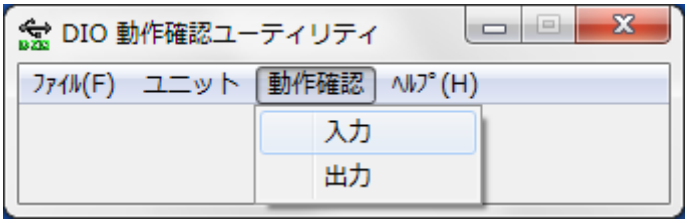
1. ユニットのオープン
   [ユニット]メニューの[オープン]をクリックしてください。



   表示された画面にてユニットID及び型名を選択してユニットのオープンをおこなってください。



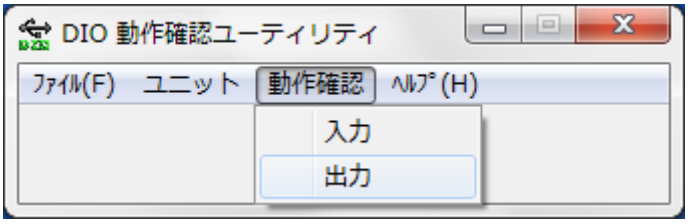
2. 入力の確認
   [動作確認]メニューの[入力]をクリックしてください。



   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毎に監視しています）

3. 出力の確認

[動作確認]メニューの[出力]をクリックしてください。

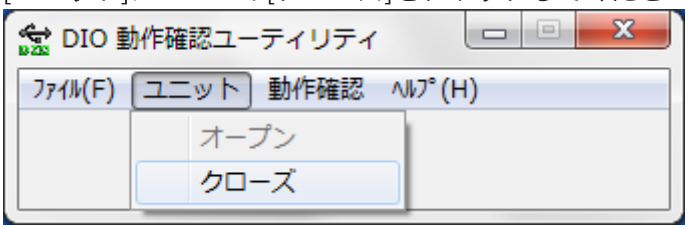


出力番号の一覧が表示されていますので、出力ON/OFFを切り替えたい出力番号をクリックしてください。ONの時には緑、OFFの時は灰色で表示されます。

（出力画面を閉じた際に全出力OFFになります）

4. ユニットのクローズ

[ユニット]メニューの[クローズ]をクリックしてください。



続けて他のユニット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。