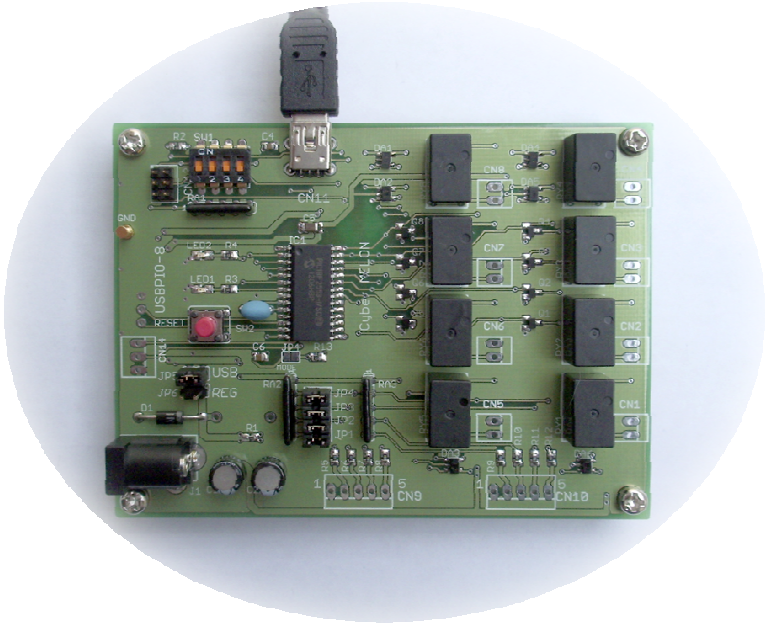
ユーザーズマニュアル

USB I/O Controller

USBPIO-8



Cyber MELON

株式会社　インターネット

ユーザーズマニュアル履歴

Rev.	改訂日付	内　容
1.0	2012/ 3/19	初版リリース
1.01	2012/7/18	DLL 呼び出し規約の説明追加
1.02	2013/6/28	入力回路定数修正
1.03	2014/4/30	寸法図追加

☆本マニュアルの最新版は当社ホームページからダウンロードいただけます。

目次

５．その他

1. はじめに

このたびはサイバーメロン・マイコンボードシリーズ USBPIO-8 をお買いあげいただきまして、誠に有り難うございます。
USBPIO-8 は USB でリレーを制御したりボード入力ピンの状態を USB で取り込むことのできる制御ボードです。
ご使用に当たりましては本書を良くお読みいただき正しい取り扱い方法をご理解の上ご使用いただきますようお願い致します。

ご注意事項

1. 本製品の仕様、および本書の内容に関して事前の予告なく変更することがありますのでご了承ください。
2. 本製品または付属プログラムの使用によるお客様の損害、および第三者からのいかなる請求につきましても当社はその責任を負いかねますので予めご了承ください。
3. 本製品に付属のソフトウェア・ライブラリおよびサンプルプログラムはその動作を完全保証するものではありません。製品に組み込んで使用される場合にはユーザ様にて十分なテストと検証をお願いします。
   ソフトウェアの最新版はユーザー登録後、当社ホームページからダウンロードしていただけます。
4. 本製品および本書に関し、営利目的での複製、引用、配布は禁止されています。

ご使用に当たって

1. 梱包品の内容をまずご確認ください。(8 ページ参照)
2. ご使用になる前に下記の安全についての注意を必ずお読みください。
3. 通電する前に、本製品の使い方を十分ご確認いただき、正しい接続と設定をご確認ください。

安全についてのご注意

1. 本製品を医療機器など人命に関わる装置や高度な信頼性・安全性を要求される装置へ搭載することはご遠慮ください。
   その他の装置に搭載する場合でもユーザー様にて十分な信頼性試験・評価をおこなった上で搭載してください。
   また非常停止や緊急時の制御は外部の独立した回路にておこなってください。
2. 本製品の改造使用は発熱、火災などの原因となり危険ですのでご遠慮ください。
3. 本製品のマニュアル記載環境以外でのご使用は故障、動作不良などの原因になりますのでご遠慮ください。
4. 本製品は導電部分が露出しておりますので、金属パーツなどショートの可能性のあるもの、液体のこぼれる可能性のある場所の近くでの使用はお控えください。
   また装置に組み込む場合も絶縁に関しては十分な注意を払ってください。
5. 電源は必ず本製品専用（指定）のものをお使いください。
   電圧、極性、プラグ形状など異なるものをご使用になりますと故障の原因となるばかりでなく、火災など重大事故に繋がる危険性があります。
6. 本製品に触れる前には体から静電気を除去してください。
7. 本製品には落下など強い衝撃を与えないでください。

使用環境

■ 以下の環境でのご使用はお控えください。

・強い電磁界や静電気などのある環境
・直射日光の当たる場所、高温になる場所
・氷結や結露のある場所、湿度の異常に高い場所
・薬品や油、塩分などのかかる場所
・可燃性の気体、液体などに触れる場所
・振動の多い場所、本製品が静止できない場所
・基板のショートを引き起こす可能性のある場所

規格取得など

■　本製品は UL CSA 規格、CCC 認証など取得しておりません。
装置に組み込む場合は各安全規格への適合性をユーザー様で
ご確認いただき、対応して頂きますようお願いします。

製品保証と修理

・本製品の保証は商品到着後１０日以内の初期不良のみ無償交換
とさせていただきます。
本マニュアルに記載するテスト手順にて正しく動作しない場合は
ただちに電源を切って、当社ホームページのサポートからご連絡
ください。　折り返し交換手順をご案内いたします。
・保証期間中であってもユーザー様の責となる故障（落下や電源の
誤接続など）は有料修理になります。
・その他の故障やクレームにつきましても当社ホームページ
http://www.cyber-melon.com　お問い合わせコーナーよりご連絡ください。

梱包内容をご確認ください

1. USBPIO-8 ボード本体　　1

2. 説明書 CD-ROM　　1

3. USB ケーブル(A-Bmini)　1

4. スペーサーとビス　　1 式

動作チェック

・本ボードの動作および動作チェックには以下の環境が必要です。

Windows パソコン

OS: WindowsXP 以上

USB の空きポート

・ USBPIO-8 を付属ケーブルでパソコンと接続してください。

. ４章-1 の手順にしたがって USB ドライバーをインストールし、接続テストを おこなってください。

もし動作異常が認められた場合は電源をはずして当社ホームページ
http://www.cyber-melon.com のサポートから症状をご連絡ください。
対処方法をメールまたは電話でご連絡いたします。

2. 概要

USBPIO-8 はUSB でI/O ポートを制御するリレー搭載の組み込みに最適なボードです。
パソコンなどと USB 接続して簡単なコマンドを送ることでリレー出力と接点入力
制御をおこなうことができます。
最大１６枚のボード(128 ビット分) を番号で管理することができます。

2.1　特長

・USBPIO-8 には8 ビット のデジタルI/O ポートを有し、ビットごとに入出力を
　選択可能です。
・出力は８個の内蔵リレーに接続されており、入力は抵抗で5V にプルアップされた
　入力ピンに接続されていますのでスイッチやリレー接点の状態を読み込むことが
　できます。　(**3.1** ブロック図を参照　)
・VB.NET、 C#.NET、 C++(MFC)　のわかりやすい制御サンプルプログラム・ソースと
　ラッパークラスの提供。
・回路図付属。

2.2 ボード仕様

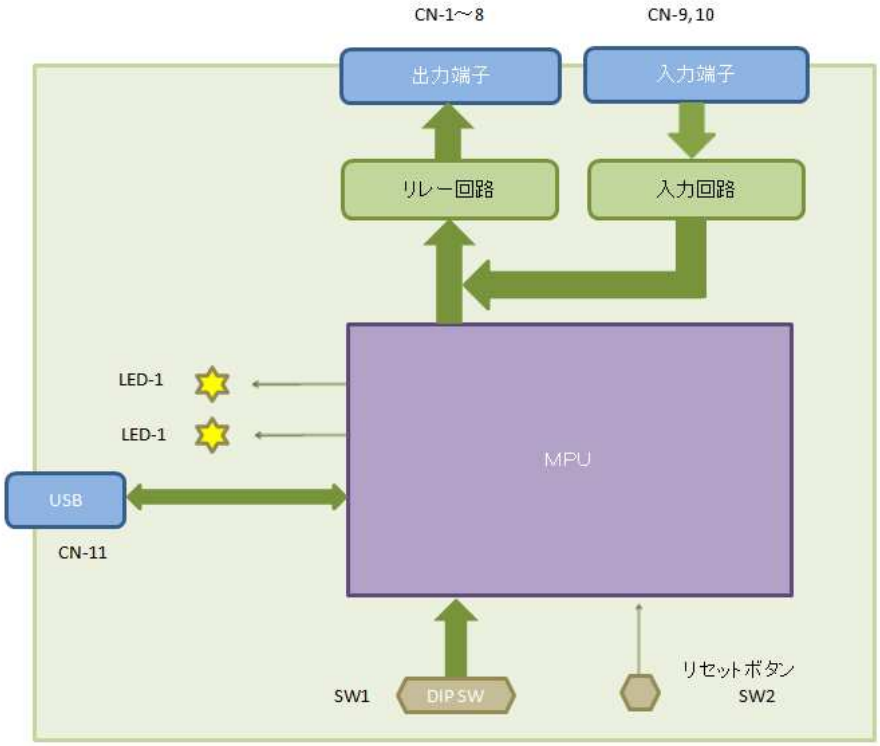
項 目	仕　様
電源電圧	USB 5V（または外部 DC5V 入力ジャンパー切替え）
入出力	USB コネクタ(MiniB タイプ)
	DIP スイッチ（4 極）
	リセットスイッチ
	接点入力用コネクター端子(5 ピン x 2 組　***1)**
	リレー出力端子(2 ピン x 8 組)　　***1)**
	LED　2 系統
ボード寸法	100.4(W) x 74.3(D)x 12.6(H) mm（足なし基板のみ）
消費電流	60mA（全リレーOFF 時）～ 360mA（全リレーON 時）
重量	60g（M3x10mm 長ナット x 4 込み）

***1)**　入出力用コネクターは実装されておりませんのでお客様のご都合の良い
コネクターを実装するか、配線を直付けしてください。(2.54mm ピッチ、1.0φ穴)
使用コネクター例：　日圧　Ｂ５Ｂ－ＸＨ－Ａ（ＬＦ）（ＳＮ）など

3．ハードウェア

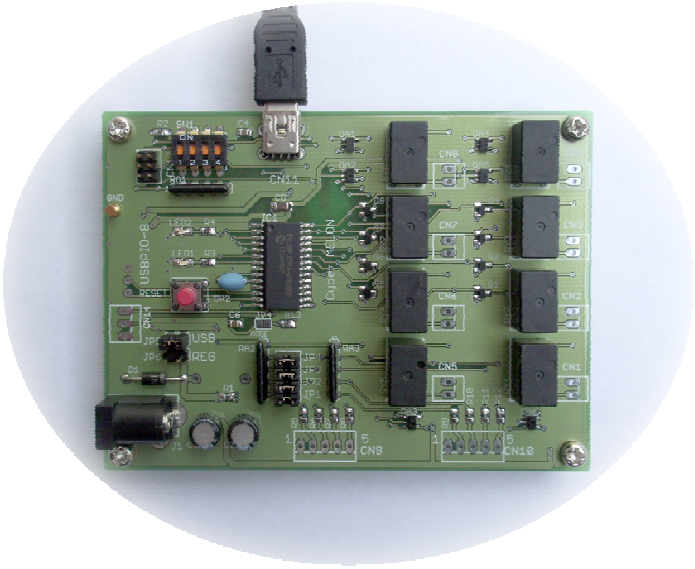
3.1 ブロック図

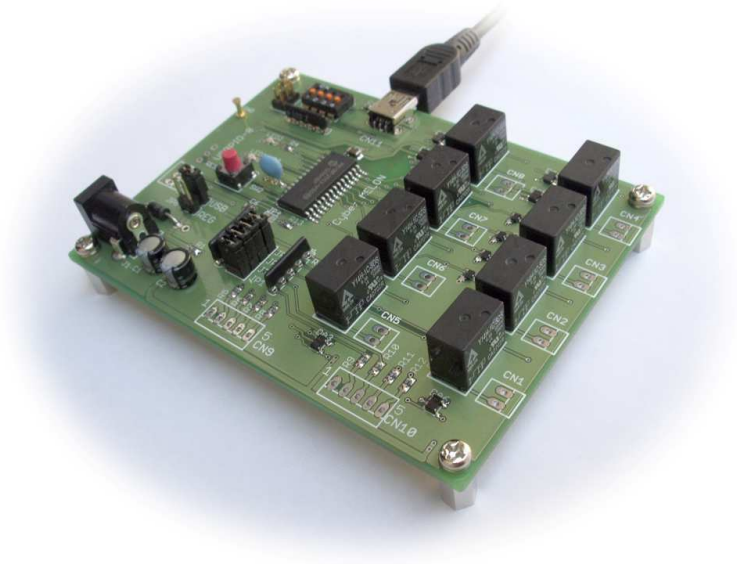
USBPIO-8 ボード・ブロックダイアグラム



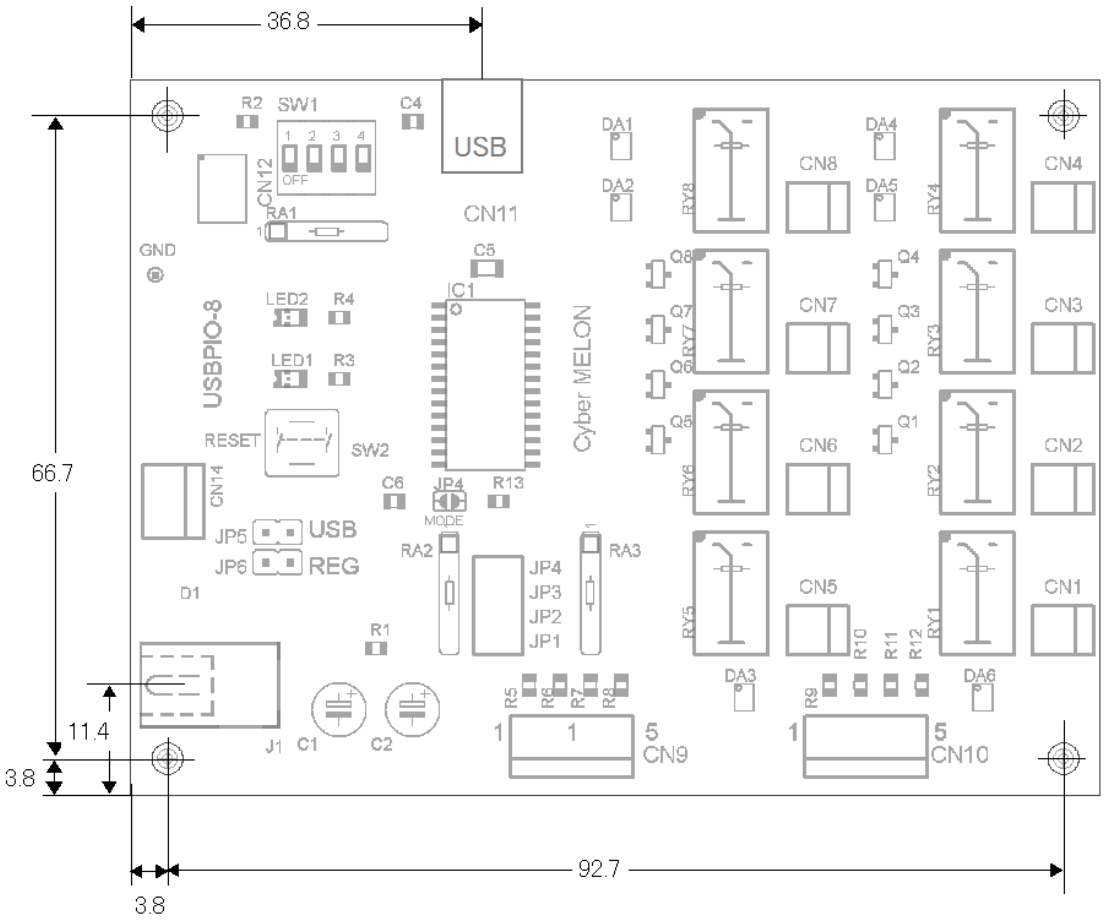
3.2 外観・接続図

USBPIO-8 の外観



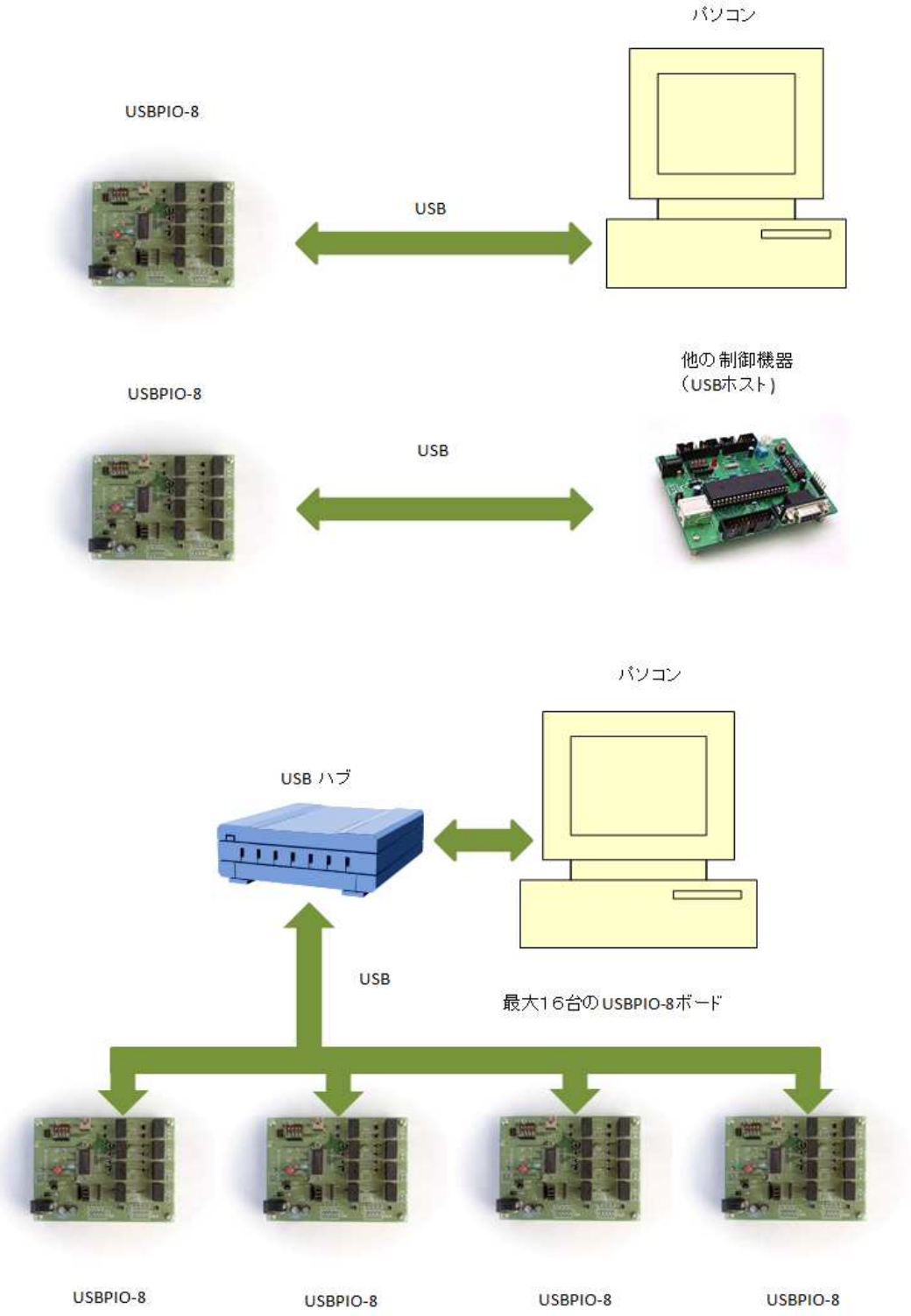


外形寸法　100.4(W) x 74.3(H)

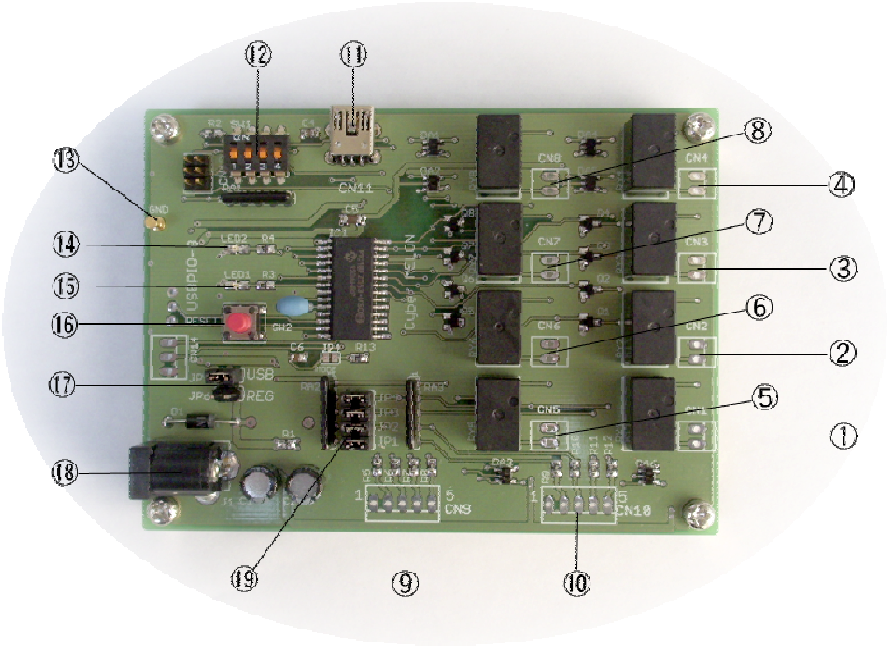


接続図

USBPIO-8 は USB ターゲット・デバイスとして動作します。



3.3 各部名称と機能



①　～⑧　リレー出力端子	リレーのメーク接点出力(CH1～8)です。
⑨～⑩択ジャンパー	ボードの電源を USB にするか 外部電源にするかを選択します。工場出荷時は外部電源（電源ジャック側）になっています。
⑪ USB コネクター	USB ケーブルを接続します。
⑫ DIP スイッチ	複数ボード接続時のボード番号を決めます。
⑬ GND 端子	グラウンドに接続されています。
⑭ LED2	エラー時に点滅します。
⑮ LED1	ゆっくり点滅して動作中を示します。
⑯ リセットスイッチ	ボードをリセットします。(通常必要ありません)
⑰ 電源選択ジャンパー	USB 電源/ 外部電源を切り替えます。
⑱ 外部 5V 電源コネクター	外部スイッチング電源を使う場合、ここに接続します。
⑲ プルアップ・ジャンパー	CH1～4 の入力プルアップを有効にします。 CH5～8 は常にプルアップされています。

3.4 コネクター・ピン配置

CN1～CN8　リレー出力端子

PIN	機 能	PIN	機 能
1A	リレー1 Common	1B	リレー1 Make
2A	リレー2 Common	2B	リレー2 Make
3A	リレー3 Common	3B	リレー3 Make
4A	リレー4 Common	4B	リレー4 Make
5A	リレー5 Common	5B	リレー5 Make
6A	リレー6 Common	6B	リレー6 Make
7A	リレー7 Common	7B	リレー7 Make
8A	リレー8 Common	8B	リレー8 Make

CN-9　接点入力コネクタ

PIN	機 能	PIN	機 能
1	GND	2	CH-1 入力
3	CH-2 入力	4	CH-3 入力
5	CH-4 入力		

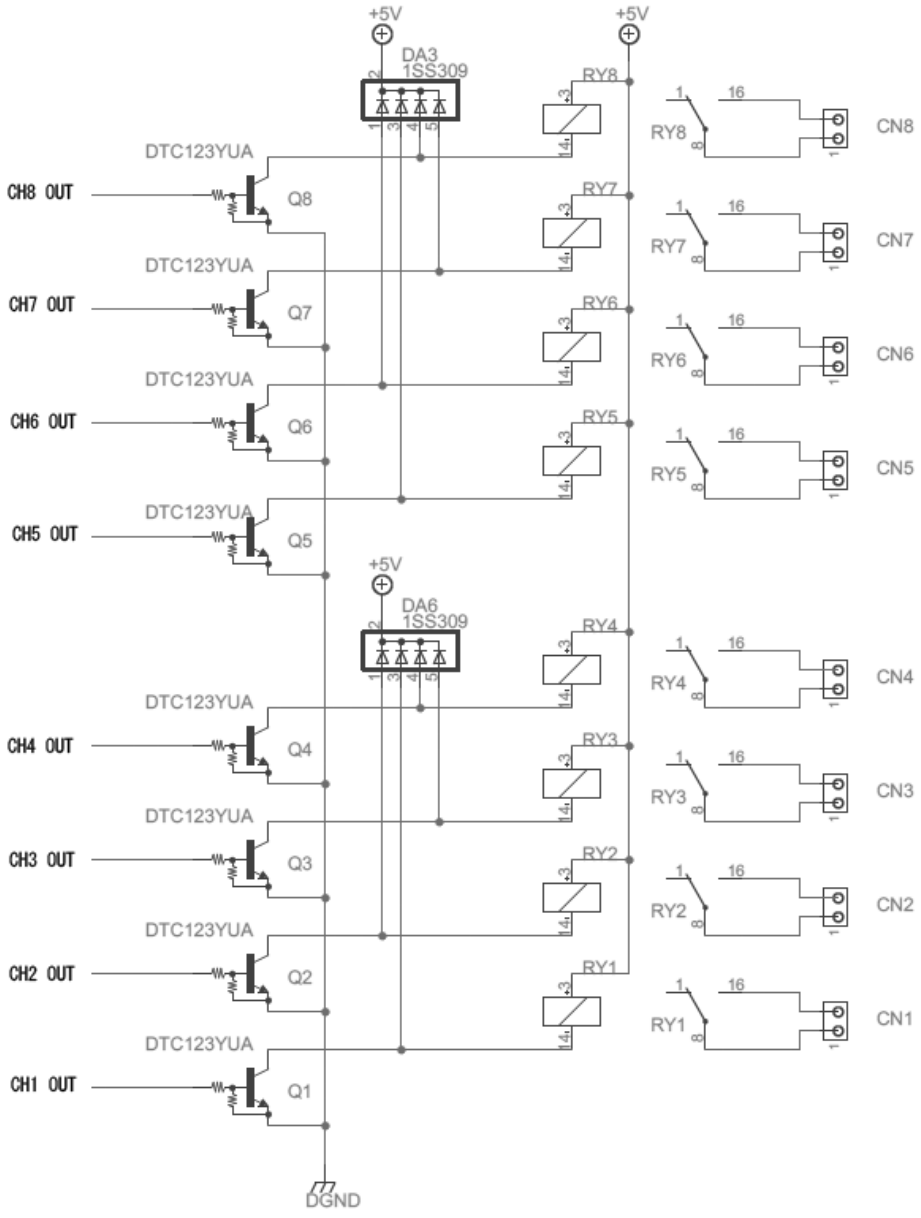
CN-10　接点入力コネクタ

PIN	機 能	PIN	機 能
1	GND	2	CH-5 入力
3	CH-6 入力	4	CH-7 入力
5	CH-8 入力		

3.5　入出力回路

3.5.1　出力回路

出力は '1' を出力すると対応するリレーが ON になり、'0' を出力すると OFF になります。　CH-1 がビット 0 に、CH-8 がビット 7 に対応します。
以下に出力部の回路図を示します。

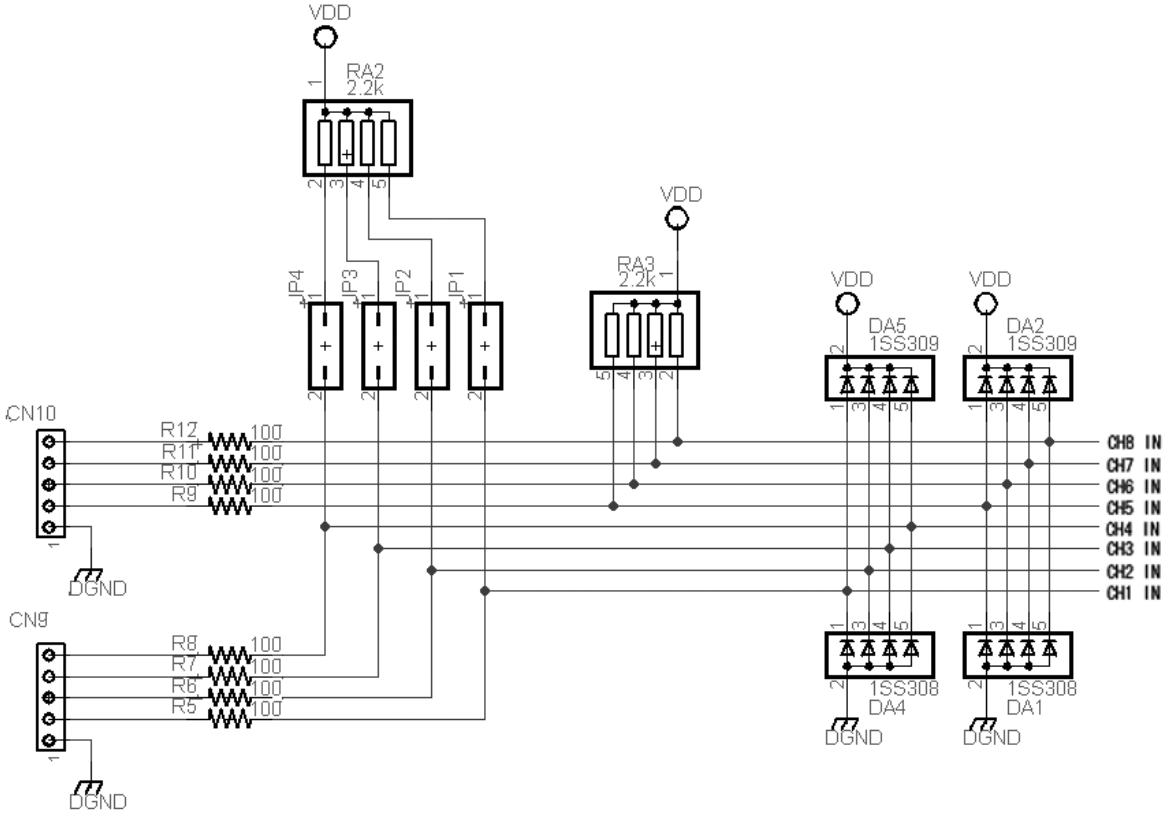


3.5.2　入力回路

入力ピンはオープンまたは 5V で'1'、 GND に落とすことで'0' となります。

以下に入力部の回路図を示します。

入力コネクター CN9, CN10 のピンアサインは **3.4** を参照してください。



注）ポートの方向を「入力」に切り替えた際、入力ピンがオープンの場合はプルアップ抵抗で入力が'1' になるため、同じチャンネルに接続されたリレーは ON になりますが入力側に影響はありません。　また同様の理由で入力ピンの状態変化に応じて出力リレーが ON/OFF しますが不良ではありません。

3.6 電気的特性

以下にポート入/出力の電気的特性を示します。

出力リレーの定格

項目	値	条件
AC 接点容量	0.5A / 125VAC	
DC 接点容量	1A / 24V	

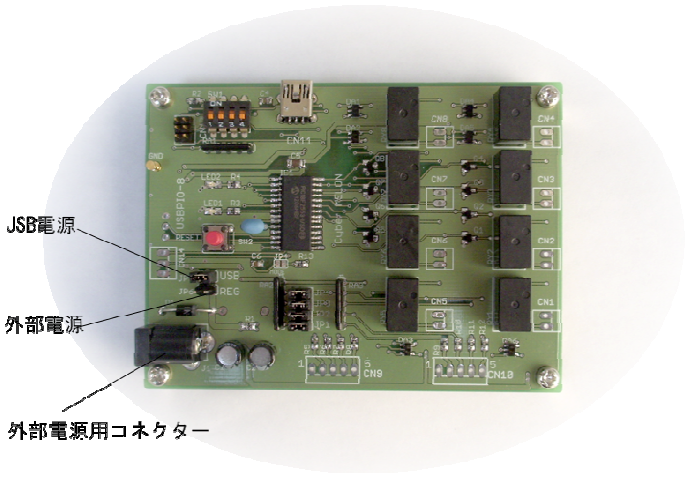
オンボード・リレーの詳細仕様は付属 CD-ROM の Documents フォルダーにある
仕様書(Y14_SeriesRelay.pdf)を参照してください。

入力の定格

項目	値	条件
最大入力電圧	5.5V	
入力インピーダンス	10kΩ	プルアップあり

3.7　電源について

・USBPIO-8 ボードの電源はＵＳＢから供給することも、またオプションの外部５Ｖスイッチング電源を使うことも可能です。
出荷時の初期設定は USB 電源です。
USB2.0 規格では 500mA を供給できますが、もし USB の電源供給能力が不足（ 360mA 以下）の場合、オプションの外部スイッチング電源（５V)から電源を供給することができます。
外部から供給する場合はジャンパープラグを図に従って差し替えてください。
外部電源は必ず当社指定のオプション品をご利用ください。



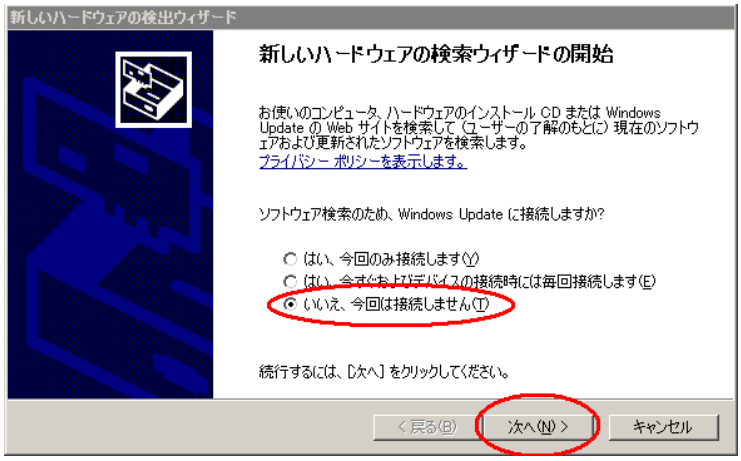
4．ソフトウェア

4.1 ドライバーのインストール

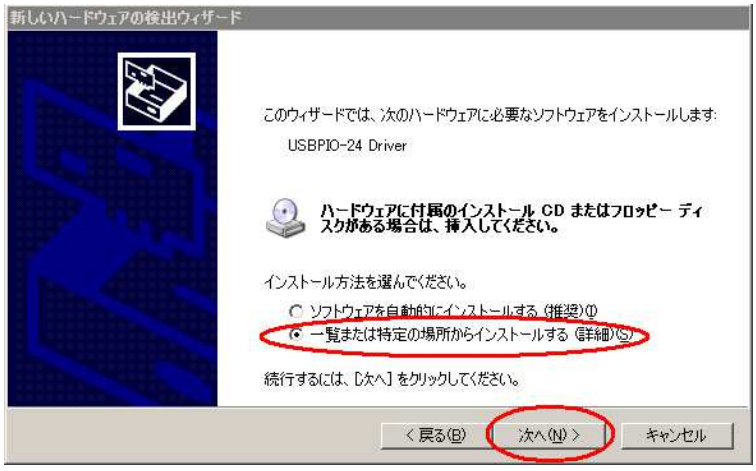
USBPIO-8 を USB で動作させるためにはパソコンに本ボード用のＵＳＢドライバーをインストールする必要があります。以下にその手順を説明します。
パソコンの OS が WindowsXP の場合は 4.1.1、 Vista の場合は 4.1.2 、Windows7 の場合は 4.1.3 を参照してください。
尚、ドライバーは USBPIO-24 と共通ですので ″USBPIO-24″ の表示になります。

4.1.1　WindowsXP の場合

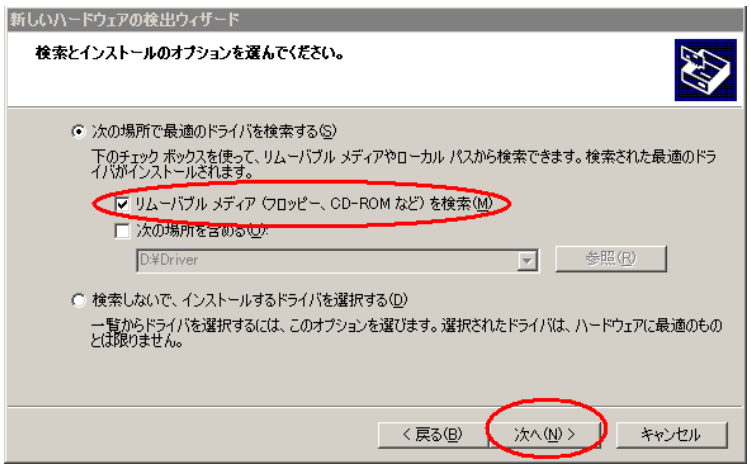
PC と USB ケーブルを接続し、USBPIO-8 の電源を入れます。
下記のダイアログが表示されますので赤丸の項目を選択して「次へ」をクリックします。



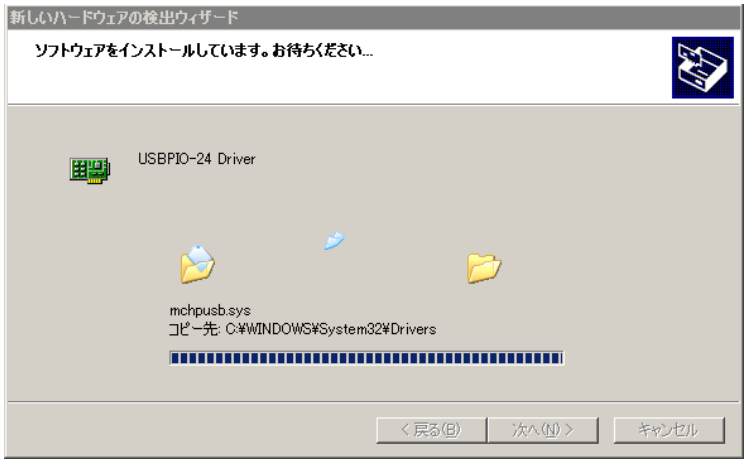
「一覧または特定の場所からインストールする」を選択して「次へ」をクリックします。

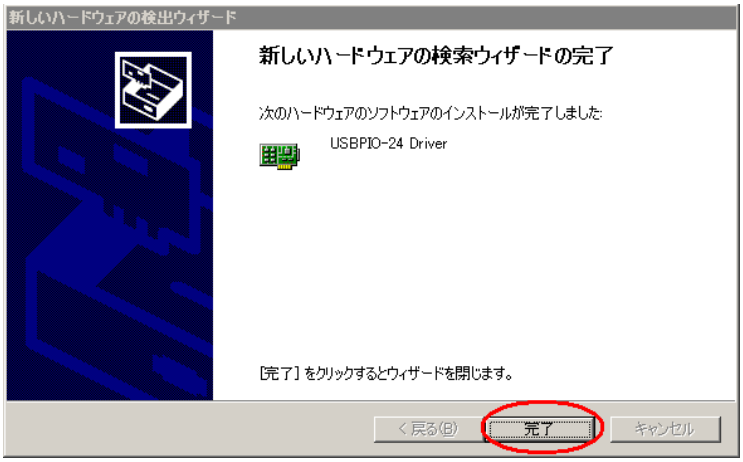


付属の CD-ROM を挿入し、赤丸の項目を選択して「次へ」をクリックします。



自動的にインストールが始まります。もし始まらない場合は「次の場所を含める」にチェックを入れて CD-ROM の “Driver32”フォルダー(32bit OS の場合) または ″Driver64″フォルダー(64bit OS の場合) を「参照」ボタンで選択します。

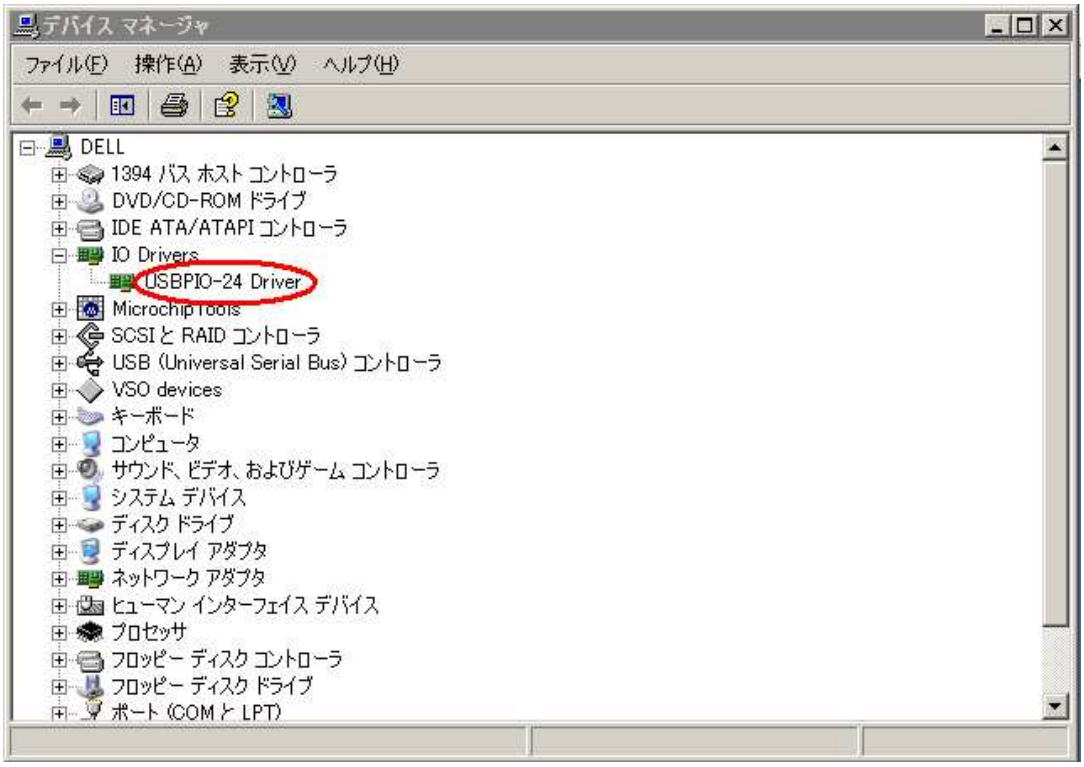




USBPIO-8 ドライバーのインストールが完了しました。

■　正常にインストールできていることを確認します。

スタートメニューからコントロールパネルを開き、
システム → ハードウェア タブ→デバイスマネージャ を開きます。

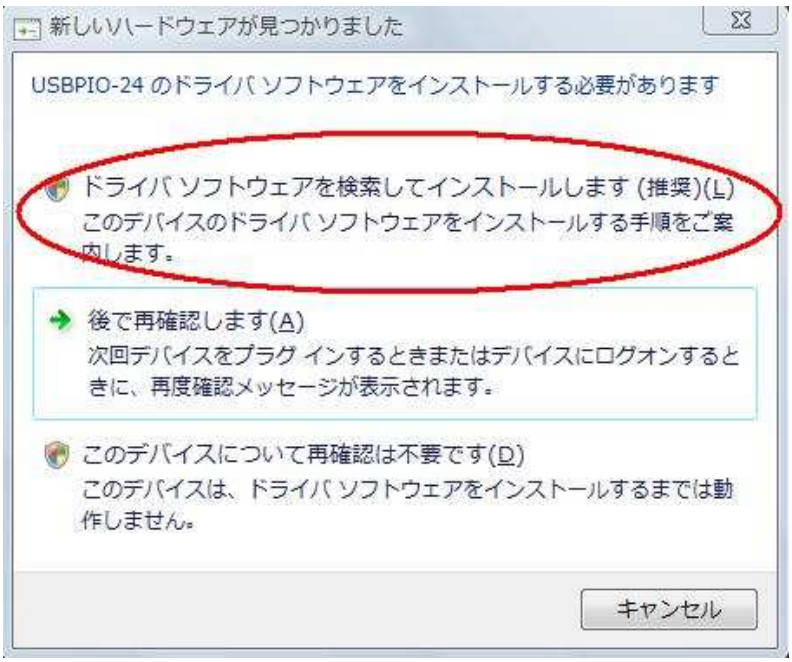


USBPIO-8 を接続した状態で上のように **IO Drivers** の項目に “USBPIO-24 Driver” が表示されていれば正しくボードが認識されています。
(ドライバーは USBPIO-24 と共通です)

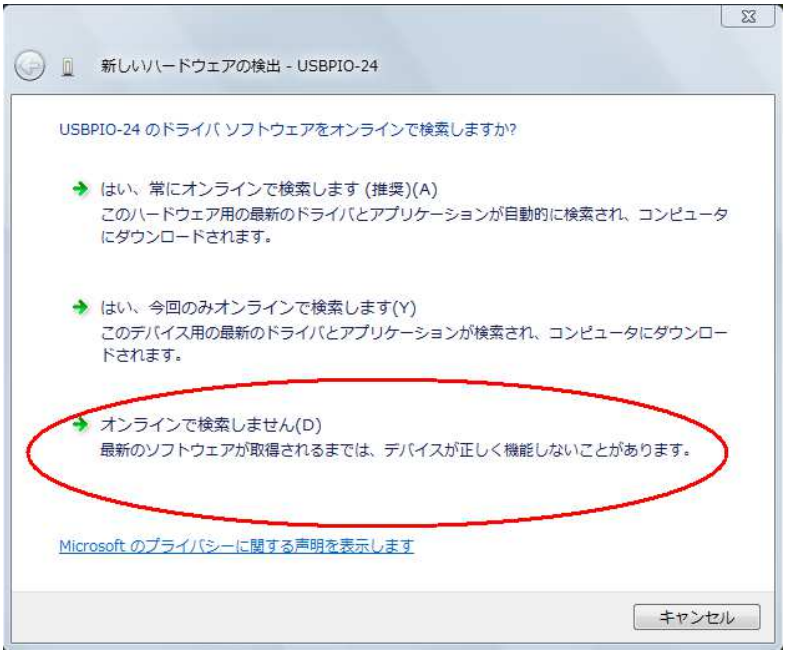
4.1.2 Windows Vista の場合

PC と USB ケーブルを接続し、USBPIO-8 の電源を入れます。

下記のダイアログが表示されますので赤丸の項目を選択します。

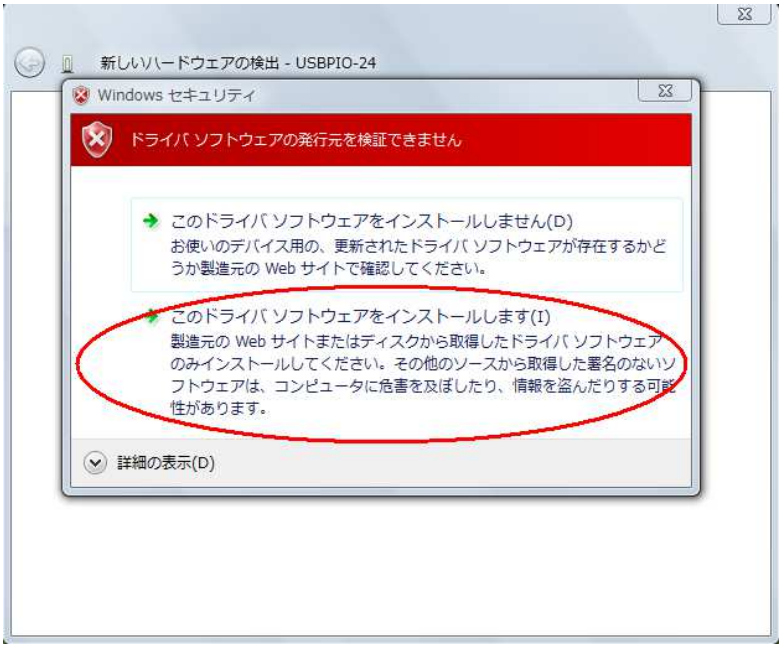


「オンラインで検索しません」を選択します。

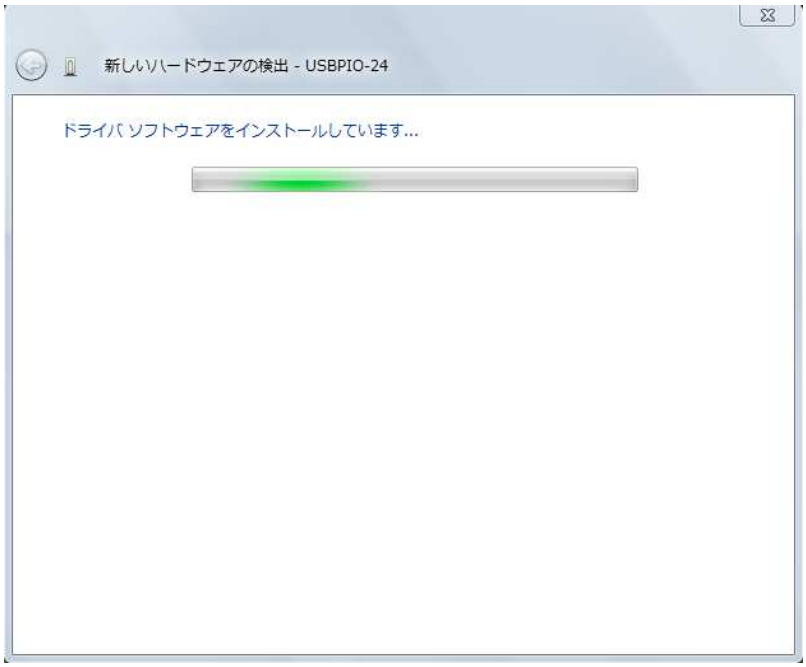


下の画面が出たら付属の CD-ROM を挿入します。

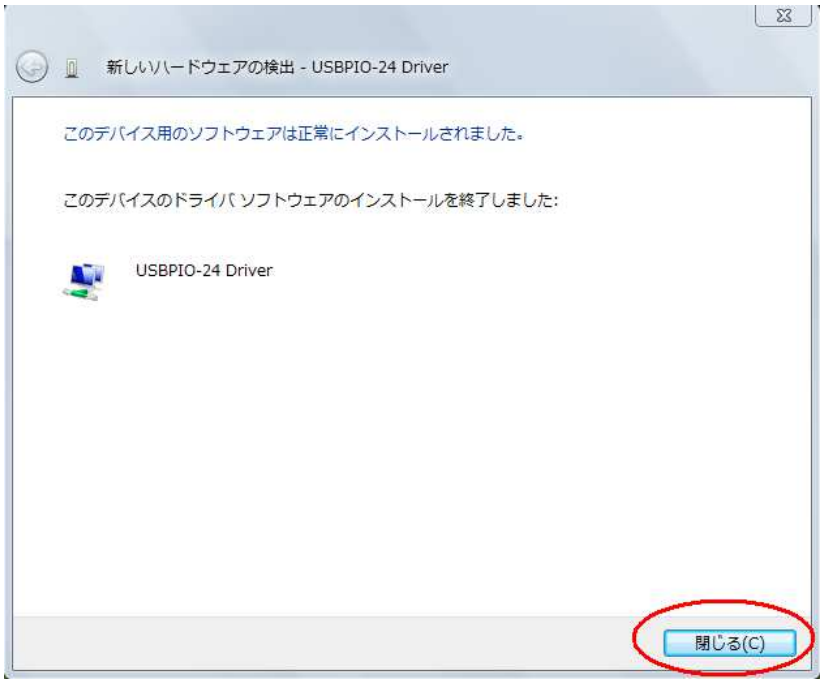
下の警告が出たら「このドライバソフトウェアをインストールします」を選択します。



インストールが開始されます。

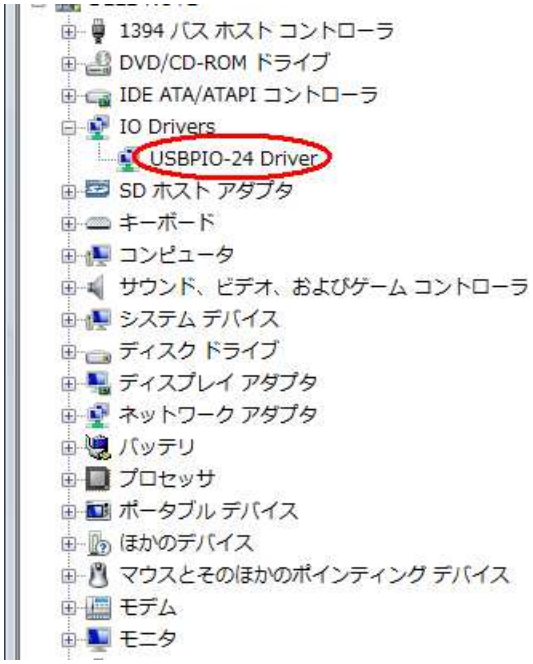


インストールが終了しました。



■　正常にインストールできていることを確認します。

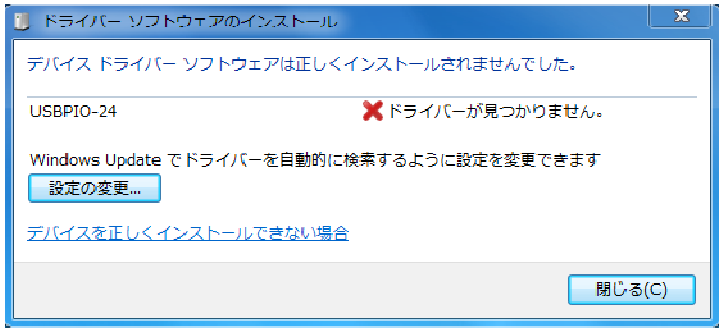
スタートメニューからコントロールパネルを開き、
システム → デバイスマネージャ　　を開きます。



USBPIO-8 を接続した状態で上のように **IO Drivers** の項目に "USBPIO-24 Driver" が表示されていれば正しくボードが認識されています。
(USBPIO-8 ドライバーは USBPIO-24 と共通です)

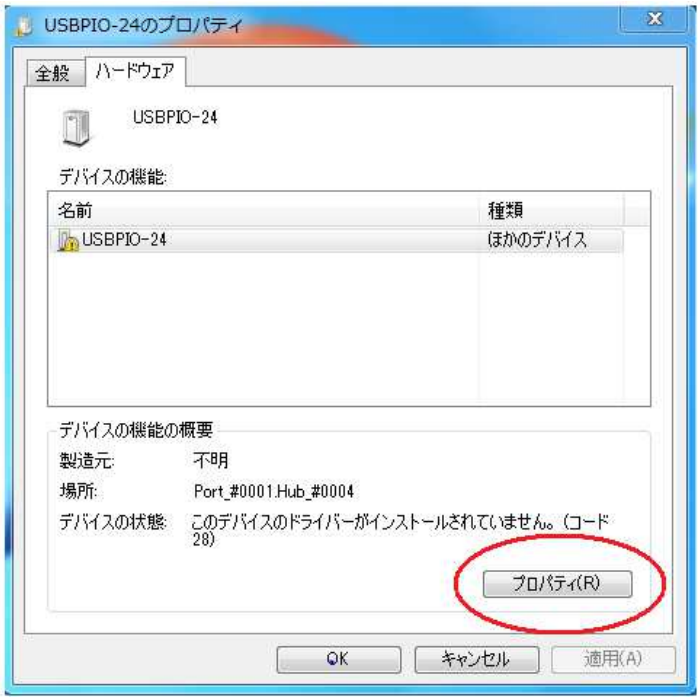
4.1.3 Windows7 の場合

PC と USB ケーブルを接続し、USBPIO-8 の電源を入れます。
一旦このようなエラーが出ます。「閉じる」で終了します。

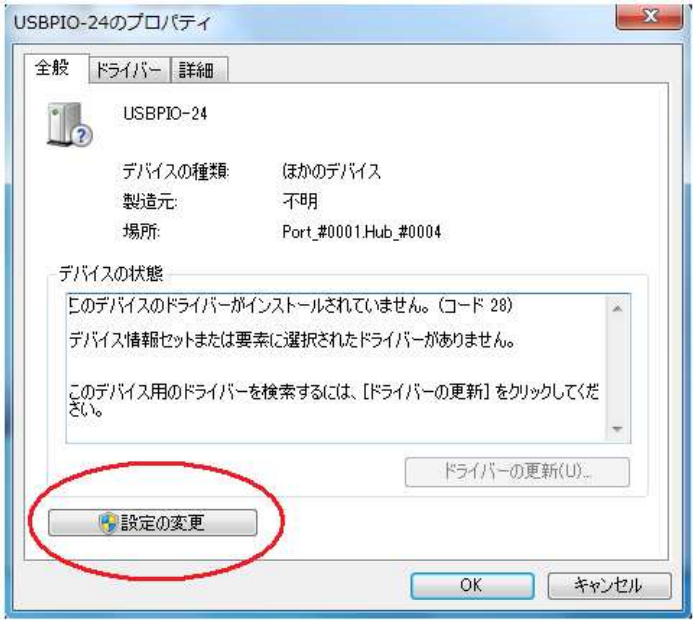


コントロールパネルから「デバイスとプリンターの表示」を選択します。

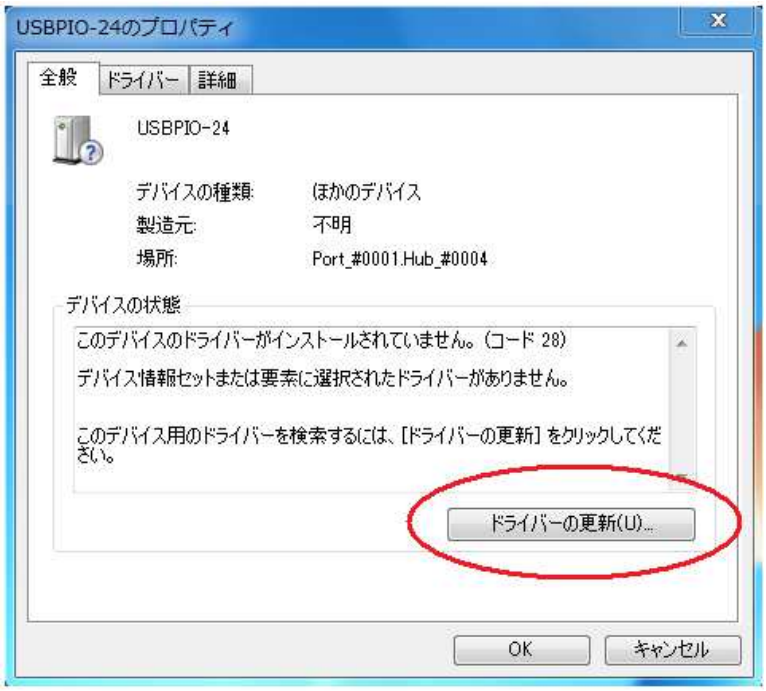
USBPIO-24 のアイコンを右クリックして「ハードウェア」タブの「プロパティー」を開きます。　　右下の「プロパティー」ボタンをクリックします。



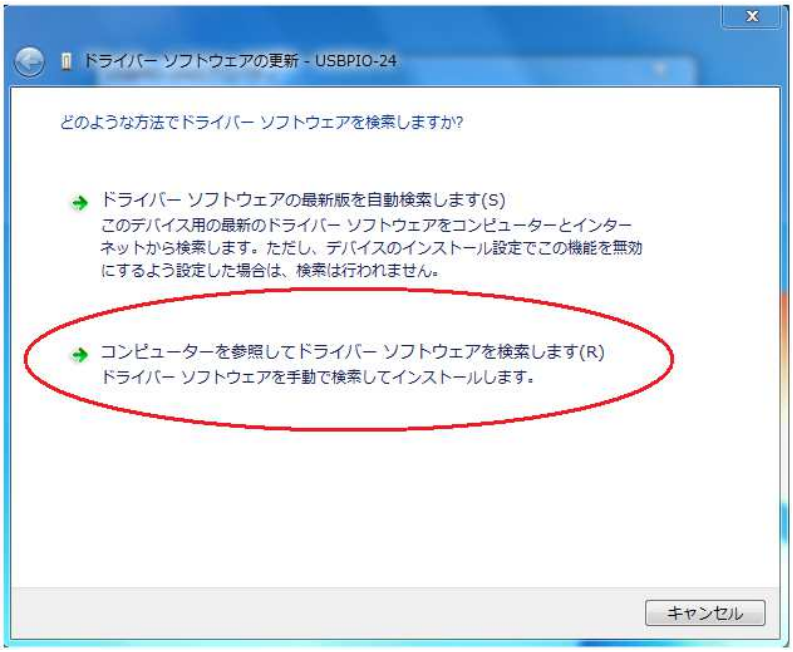
「設定の変更」をクリックします。



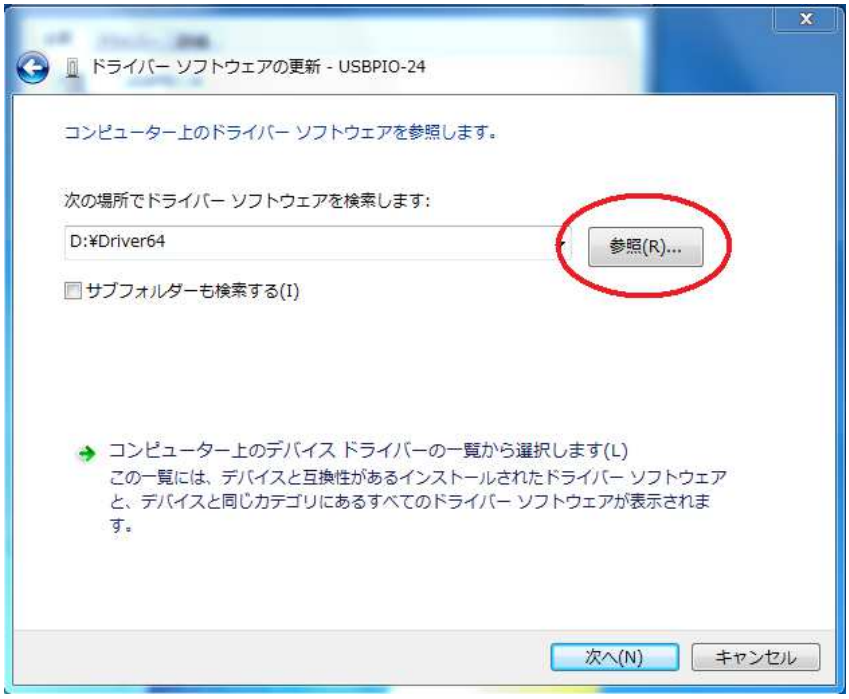
「ドライバーの更新」をクリックします。



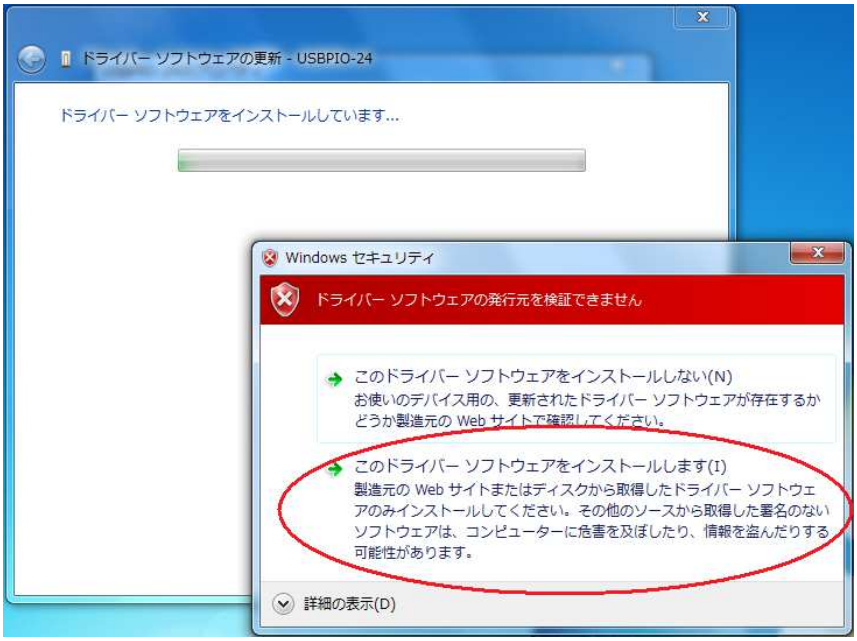
「コンピュータを参照してドライバーソフトウェアを検索します」をクリック



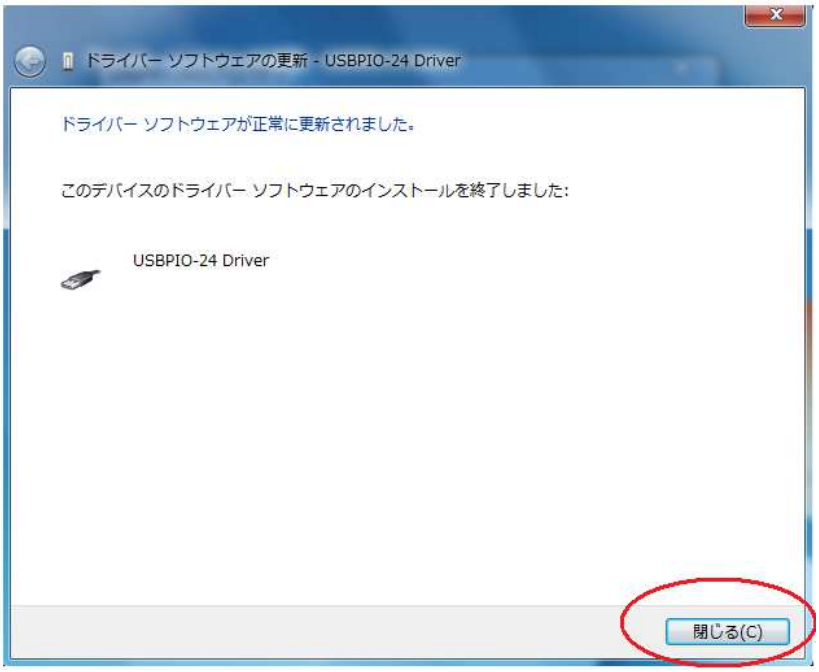
参照ボタンで Driver フォルダーを選択します。
64bit OS の場合は "Driver64" フォルダー、32bit OS の場合は"Driver32"
フォルダーを CD-ROM から選択します。



「このドライバーソフトウェアをインストールします」をクリック。



完了画面が出たら「閉じる」をクリックします。



先ほど開いたデバイス画面で黄色い△マークが消えていることを確認します。

4.2 コマンド一覧

USBPIO-8 は USB から簡単な文字列のコマンドを受け取って８ビットのポートを制御します。

4.2.1　コマンドの形式

コマンドはすべて ASCII 文字列で**数値はすべてヘキサ（１６進数）文字にて送ります。**
最初の２文字がコマンドで、あとポートの番号(USBPIO-8 では"00"固定) 、制御データなどのパラメータが続きます。
モード設定で「チェックサムを有効にする」指定をした場合はコマンドの後に
チェックサム（それまでの文字をヘキサ値で合計した数値の下８ビットを２桁ヘキサ文字にしたもの）を追加します。
チェックサムの計算例は 4.2.2 の"MD" コマンドの項目を参照してください。
コマンド、パラメータ（、チェックサム）の後にはデリミタとして CR（キャリッジ・リターンコード = 0x0d)を付加してください。
USBPIO-8 は CR を受け取った時点でコマンドを解析して実行します。

コマンド実行に成功すれば"OK" 失敗すれば "NGxx" の文字列を返します。
（xx はエラー番号）応答のデリミタも CR です。
ただしモード設定で「応答なし」を指定した場合、USBPIO-8 はコマンドに対する応答を返しません。(ポート入力モードの場合を除く)

4.2.2　コマンドの詳細仕様

■ **モード設定コマンド**　**形式：MDmm**

最初の２文字"MD"がコマンド、続く２文字でモードを指定します。

パラメータ mm で bit-7 (0x80)が１のときチェックサム有効となり、
USBPIO-8 はサムチェックの計算をおこないます。
この場合次から送るコマンドには XXppcc の形式（ XX はコマンド, pp は
パラメータ、 cc はチェックサムコード）になります。
USBPIO-8 で cc をチェックして正しくなければエラーを返してコマンドは実行
されません。
bit-6(0x40)が１のときは「応答なし」モードとなり、コマンドに対して
応答を返しません。（ ポート入力コマンドを除く）

コマンド	MDmm	
機能	モードを設定する	
引数	値	内 容
mm	80	チェックサム有効
	40	応答なし
		更に bit-7 (0x80)が１のとき：チェックサム有効 bit-6 (0x40)が １のとき：応答なし
応答		"OK"： 成功、"NGxx"： 失敗
使用例	MDC0 MD40	チェックサム有効＋応答なし チェックサムなし＋応答なし
備考	電源投入時は 00（ チェックサムなし＋応答あり）です。 途中でモードを変更することも可能です。	

・チェックサムの計算法

ＭＤコマンドで「チェックサム有効」を指定した場合は、コマンド文字列の末尾
（ターミネータ 'CR' の前）にチェックサムを追加します。
チェックサムはコマンド文字列の各文字のアスキー値を合計したものの下８ビットを
２桁のヘキサで表します。
例えばボードに送信するコマンドが ポートの読み込みの場合、コマンド文字列は
"PI00" ですので、各文字の ASCII 値を合計すると
　P　　I　　0　　0

0x50 + 0x49 + 0x30 + 0x30 = 0xF9 となるので送信する文字列は チェクサムの
２文字を追加して ″PI00F9{CR}″ となります。({CR} はターミネータの 0x0d)

・ご注意

モード設定コマンド自身は最初ボード側で受け取ったとき、チェックサムの検査を
しません。

■ **ポート方向設定コマンド**　**形式：PDppvv**

最初の２文字"PD" がコマンド、続く２文字でポート番号（"00"に固定）を指定し
最後の２文字でそのポートの方向を指定します。
モード設定で「チェックサム有効」を指定したときは **PDppvvcc**（cc はチェックサム）
の形式になります。

pp　00　　（ USBPIO-24 と互換のために必要です ）

vv　1 のビットが入力、0 のビットが出力に設定されます。
　　ビット 0(0x01) が CH-1 に対応し、ビット 7(0x80)が CH-8 に対応します。

コマンド	PDppvv	
機能	各ポートの入出力方向を設定する	
引数	値	内 容
pp	00	固定値
vv	00..FF	1 のビットが入力、0 のビットが出力
応答		“OK”： 成功、”NGxx”： 失敗
使用例	PD00FF PD00C0 PD005F	ポートをすべて入力に設定 ポートの bit-6,7 を入力、他のビットを出力に ポートの bit-5,7 を出力、他のビットを入力に
備考		

■ データ出力コマンド　形式：POppvv

最初の２文字"PO"がコマンド、続く２文字でポート番号（"00"に固定）を指定し
最後の２文字でデータを指定します。

注）"PO "の２文字目は「ゼロ」ではなく OUT の「オー」です。

モード設定で「チェックサム有効」を指定したときは **POppvvcc**（cc はチェックサム）の形式になります。

pp　00　　（ USBPIO-24 と互換のために必要です ）

vv　00～FF のヘキサ値をポートに設定します。

ビット 0（ 0x01）が CH-1 に対応し、ビット 7(0x80)が CH-8 に対応します。

コマンド	POppvv	
機能	ポートにデータをラッチ（出力）する	
引数	値	内 容
pp	00	（固定値）
vv	00..FF	出力値
応答		"OK"： 成功、" NGxx"： 失敗
使用例	PO00FF PO00C0 PO005F	ポートをすべて 1 に ポートの bit-6,7 を 1 に、他のビットを 0 に ポートの bit-5,7 を 1 に、他のビットを 0 に
備考	ポート方向設定コマンドで「出力」に設定されたビットに対してのみ有効です。 '1' でリレーが ON （メーク）になります。	

■ データ入力コマンド　形式：PIpp

最初の２文字"PI" がコマンド、続く２文字でポート番号("00"に固定) を指定します。

モード設定で「チェックサム有効」を指定したときは **PIppcc**（cc はチェックサム）の形式になります。

pp　00　　（ USBPIO-24 と互換のために必要です ）

コマンド	PIpp	
機能	ポートからデータを入力する	
引数	値	内 容
pp	00	(固定値)
応答	PIppvv	vv はポート入力値(00..FF)
使用例	PI00	ポートから読み込む
備考	ポート方向設定コマンドで「入力」に設定されたビットに対してのみ有効です。他のビットは取り込み時にマスクしてください。	

■ DIP スイッチ読み込みコマンド　形式：CH

DIP スイッチの設定を読み込みます。
ボードの番号区別などに利用することができます。
最初の２文字"CH" がコマンドです。
モード設定で「チェックサム有効」を指定したときは CHcc
(cc はチェックサム) の形式になります。

応答は ON のビット が 1、OFF のビット が 0 になります。
DIP スイッチ上の番号とビットの対応は
1　bit-0
2　bit-1
3　bit-2
4　bit-3
となります。

コマンド	CH	
機能	DIP スイッチの設定値を読み込みます。	
引数	値	内 容
応答	CHvv	vv が DIP スイッチ読み込み値(00..0F)
使用例	CH	
備考		

4.2.2　エラーコード一覧

コマンドにエラーがあった場合（モード設定で「応答なし」を設定いなければ）
"NGxx"（xx は下記のエラー番号）を返します。

エラー名	エラー番号	内　容
NOERROR	00	エラーなし（成功）
ERR_ILLEGAL_COMMAND	01	存在しないコマンド
ERR_ILLEGAL_PORT	02	不正なポート番号指定
ERR_ILLEGAL_VALUE	03	不正なパラメータ
ERR_ILLEGAL_MODE	04	不正なモード指定
ERR_ILLEGAL_FORMAT	05	不正なフォーマット
ERR_CHECKSUM	06	チェックサムエラー

4.3 サンプルプログラムの実行

・サンプルプログラムは付属 CD-ROM の"Sample Programs" フォルダーに以下の3本が入っています。

１．USBPIO-8TestVB

USBPIO-8 を USB で制御する VB.NET のサンプルです。

2. USBPIO-8TestCS

USBPIO-8 を USB で制御する VC#.NET のサンプルです。

3. USBPIO-8TestCPP

USBPIO-8 を USB で制御する VC++ (MFC) のサンプルです。

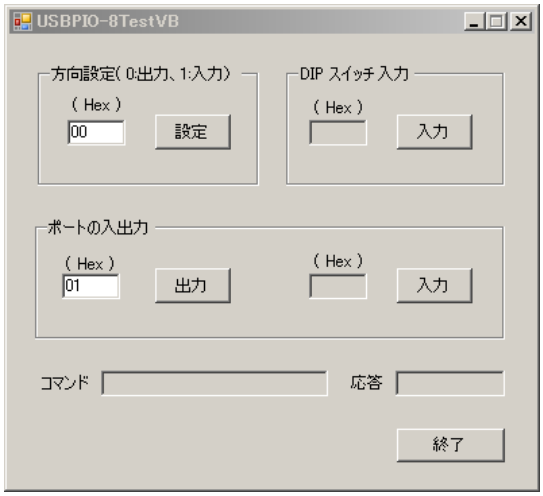
・プログラムの実行には .NET Framework 2.0 以上の環境が必要です。
OS が Vista 以降の場合はそのままお使いいただけますが、Windows XP で .NET Framework がインストールされていない場合は 付属 CD-ROM の redist フォルダーにある **vcredist_x86.exe** を実行して再頒布可能パッケージ(x86)を予めインストールしてください。

・もしサンプルプログラムを改造する場合はコンパイルに VisualStudio 2005 以降の開発環境が必要です。
サンプルプログラムは古い環境でもコンパイルできるよう VisualStudio 2005 で作成されています。　より新しい環境ではプロジェクト起動時にプロジェクトを自動変換しますので、それを許可してください

・CD-ROM の "Sample Programs" フォルダーをどこかハードディスクの適当な場所にコピーしてお使いください。

4.3.1　USBPIO-8TestVB の実行

USBPIO-8 ボードをパソコンと USB ケーブルで接続し、電源を入れて
Sample Programs の下の bin フォルダーの USBPIO-8TestVB.exe を実行します。

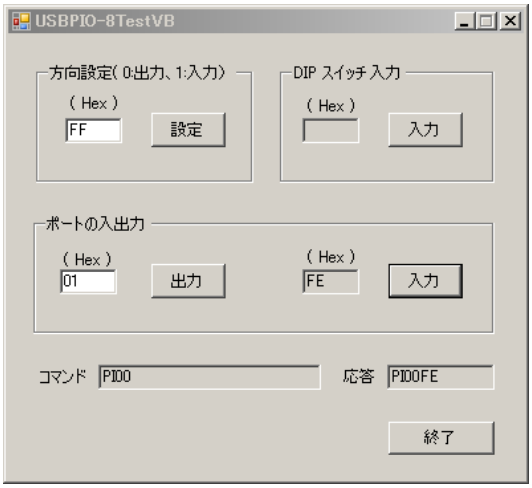


・ポートの方向を２桁へキサで入力し、設定ボタンを押してコマンドを送ります。
　'0' に指定したビットが「出力」に、'1' に指定したビットが「入力」になります。
　例えば 'F0' を指定すると ビット 0～3 (CH-1～4)が出力に、ビット 4～7 (CH-5～8)
　が入力になります。
　送ったコマンド内容は画面下の「コマンド」のところにそのまま表示されます。
　それに対する USBPIO-8 から受け取った応答が「応答」のところに表示されます。
　（デリミタの CR（キャリッジ・リターンコード）は表示されません）

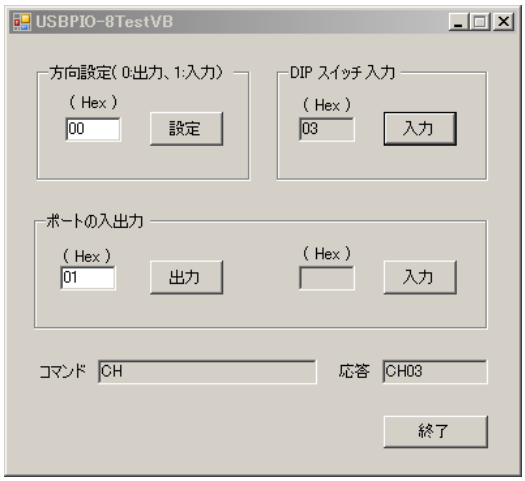
注）　パワーオン直後のボードの各ポートはすべて入力に設定されます。

・「出力」ボタンをクリックすると出力に設定したビットのリレーが ON（メーク）します。
例えば 0～7 全てのビットが出力に設定されているとして"41"（ヘキサ）を指定した場合ビット 6 (CH-7)とビット 0（CH-1）のリレーが ON になります。

・「入力」ボタンをクリックすると入力に設定したビットを読み込んでヘキサで表示します。
例えば 0～7 全てのビットが入力に設定されているとしてビット 0 (CH-1)のスイッチが押されて（短絡）いる場合、"FE"が返ってきます。



・DIP スイッチ入力ボタンをクリックすると DIP スイッチの ON のビットが'1' OFF のビットが'0' として返ってきます。
例えば スイッチ 1～4 の 1, 2 が ON の場合 "03" が返ってきます。



4.3.2　USBPIO-8TestCS の実行

内容は USBPIO-8TestVB と同等ですがソースは C# 言語で書かれています。

4.3.3　USBPIO-8TestCPP の実行

内容は USBPIO-8TestVB と同等ですがソースは C++ 言語で書かれています。

4.4　複数ボードの制御

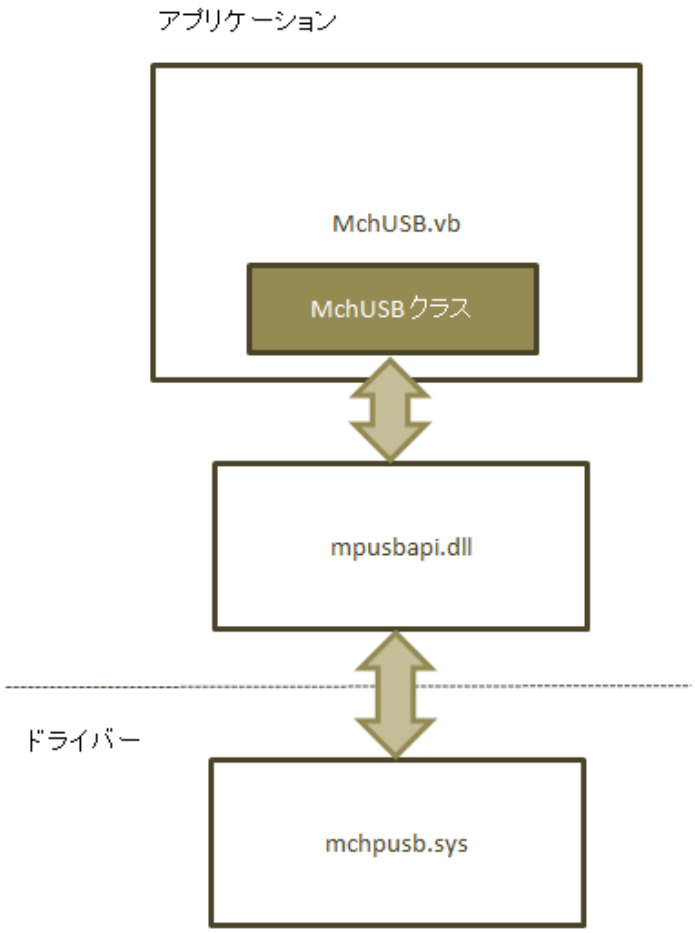
接続順に依存しないでボードを識別するためにはボードの DIP スイッチをユーザープログラムで読み込んでデバイス ID とボードを関連付ける必要があります。
具体的にはデバイス ID （0～15)を指定して USB を（0,1,2.. と）順に Open し、
そのとき読み込んだチャンネル番号(DIP スイッチ読み取りコマンドで検出）の
対応表を作ります。
アクセスしたいボードのチャンネル番号から表を検索してデバイス ID を取得し、
そのデバイス ID を引数に USB の読み書きをおこないます。
付属のサンプルでは 1 台だけの制御をしている為 ID はすべて'0' を使用しています。

4.5　サンプルプログラム解説

4.5.1　USB による I/O 制御(VB 版)

付属 CD-ROM の Sample Programs フォルダーの USBPIO-8TestVB は VB.NET で作成した USB による I/O 制御サンプルです。
使い方は 4.3 に書いた通りですが、ここではプログラムの USB 制御コアについて説明します。
プロジェクトの起動は USBPIO-8TestVB　フォルダーの USBPIO-8TestVB.vbproj　をダブルクリックしておこなえます。
以下にＰＣ側のプログラム階層図を示します。

ＰＣプログラム階層図



*) 64bit OS の場合、ドライバーは mchusb64.sys になります。

Form1.vb から MchUSB.vb の MchUSB クラスを呼び出して USB の入出力をおこないます。
以下は Form1.vb での主な制御関数です。

関数名	機 能
GetReply	USB から応答を受け取る
ButtonXXX_Click	各ボタンの応答関数(XXX はボタン名)

MchUSB.Write 関数によって USB ドライバーにコマンドが送られて USBPIO-8 ボードがこれを受け取り I/O 制御がおこなわれます。
USB 制御のコアは MchUSB.vb ファイルに MchUSB クラスとしてカプセル化されています。
このクラスはマイクロチップ社提供の DLL　mcusbapi.dll を経由して USB ドライバーを呼び出します。　以下は MchUSB クラスの主な関数です。

MchUSB クラスの概要

関数名	機 能
IsOpen	指定のパイプがオープンされているか調べる
Open	入出力のパイプをオープンする
Close	入出力のパイプをクローズする
Write	出力パイプにデータを書き込む
ReadLine	入力パイプからデータを CR まで読み出す

USB の利用手順は

１．MchUSB クラスのインスタンスを作成する
　サンプルプログラムでは Form1.vb の 12 行目あたりにある
　　usbpio = New MchUSB
　で MchUSB クラスのインスタンスを定義しています。
２．Open 関数でパイプをオープンする。
３．ReadLine/Write 関数でパイプに対して読み書きをおこなう
４．使用が終われば Close 関数でパイプを閉じる
となります。
以下は MchUSB クラスの関数仕様です。

MchUSB クラスの関数仕様

関数名	Open	
機能	パイプのオープン	
引数	タイプ	内 容
iDevice	Integer	複数のボードを扱う場合のデバイス番号(0..15)
戻り値	Boolean	true: 成功、false:失敗
備考		

関数名	Close	
機能	パイプのクローズ	
引数	タイプ	内 容
iDevice	Integer	複数のボードを扱う場合のデバイス番号(0..15)
戻り値	Boolean	true: 成功、false:失敗
備考		

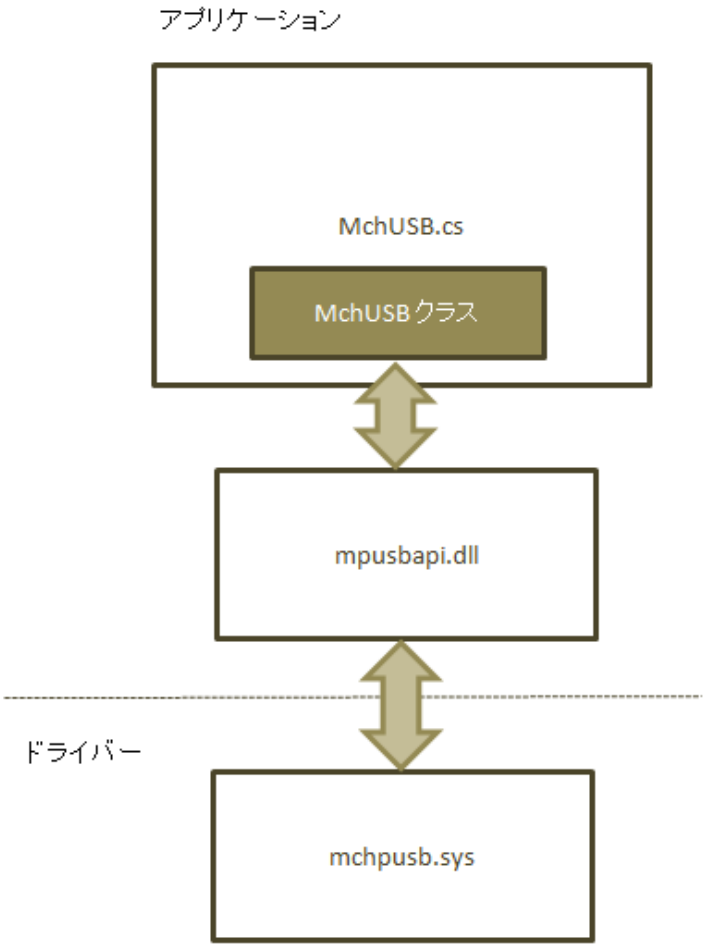
関数名	Write	
機能	パイプへの書き込み	
引数	タイプ	内 容
iDevice	Integer	複数のボードを扱う場合のデバイス番号(0..7)
sCommand	String	USB に送るコマンド文字列
戻り値	Boolean	true: 成功、false:失敗
備考	USBPIO-8 側エンドポイント（チップ内バッファ）のサイズは 64 バイト	

関数名	ReadLine	
機能	パイプの読み出し	
引数	タイプ	内 容
iDevice	Integer	複数のボードを扱う場合のデバイス番号(0..15)
nTimeout	Integer	タイムアウト (msec)
戻り値	String	USB からの応答文字列
備考		

関数名	IsOpen	
機能	パイプのオープン状態を調べる	
引数	タイプ	内 容
iDevice	Integer	複数のボードを扱う場合のデバイス番号(0..15)
戻り値	Boolean	true: オープン、false:非オープン
備考		

4.5.2　USB による I/O 制御(C# 版)

付属 CD-ROM の Sample Programs フォルダーの USBPIO-8TestCS は C#.NET で作成した USB による I/O 制御サンプルです。
使い方は 4.3 に書いた通りですが、ここではプログラムの USB 制御コアについて説明します。
プロジェクトの起動は USBPIO-8TestCS　フォルダーの USBPIO-8TestCS.csproj　をダブルクリックしておこなえます。
以下にＰＣ側のプログラム階層図を示します。



*) 64bit OS の場合、ドライバーは mchusb64.sys になります。

Form1.cs から MchUSB.cs の MchUSB クラスを呼び出して USB の入出力をおこないます。
以下は Form1.cs での主な制御関数です。

関数名	機 能
GetReply	USB に送出する
buttonXXX_Click	各ボタンの応答関数(XXX はボタン名)

MchUSB.Write 関数によって USB ドライバーにコマンドが送られて USBPIO-8 ボードがこれを受け取り I/O 制御がおこなわれます。
USB 制御のコアは MchUSB.cs ファイルに MchUSB クラスとしてカプセル化されています。
このクラスはマイクロチップ社提供の DLL　mcusbapi.dll を経由して USB ドライバーを呼び出します。　以下は MchUSB クラスの主な関数です。

MchUSB クラスの概要

関数名	機 能
IsOpen	USB がオープンされているか調べる
Open	入出力のパイプをオープンする
Close	入出力のパイプをクローズする
Write	出力パイプにデータを書き込む
ReadLine	入力パイプからデータを CR まで読み出す

USB の利用手順は

1. MchUSB クラスのインスタンスを作成する
   サンプルプログラムでは Form1.cs の 38 行目あたりにある
   ```
   usbpio = new MchUSB();
   ```
   で MchUSB クラスのインスタンスを定義しています。
2. Open 関数でパイプをオープンする。
3. ReadLine/Write 関数でパイプに対して読み書きをおこなう
4. 使用が終われば Close でパイプを閉じる

となります。
以下は MchUSB クラスの関数仕様です。

MchUSB クラスの関数仕様

関数名	Open	
機能	パイプのオープン	
引数	タイプ	内 容
iDevice	int	複数のボードを扱う場合のデバイス番号(0..15)
戻り値	Boolean	true: 成功、false:失敗
備考		

関数名	Close	
機能	パイプのクローズ	
引数	タイプ	内 容
iDevice	int	複数のボードを扱う場合のデバイス番号(0..15)
戻り値	Boolean	true: 成功、false:失敗
備考		

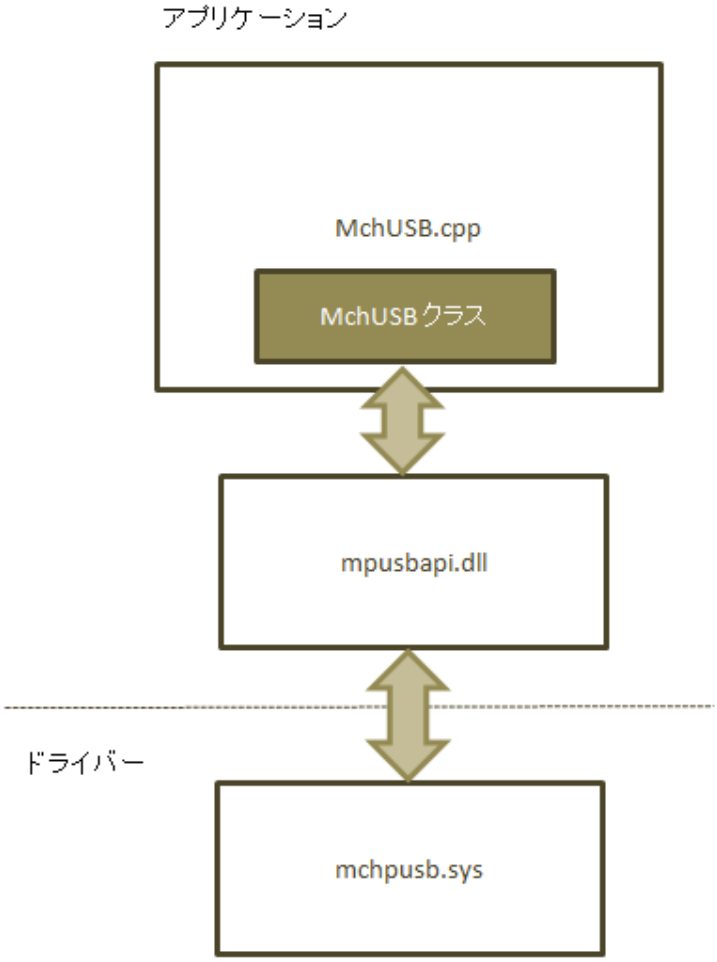
関数名	Write	
機能	パイプへの書き込み	
引数	タイプ	内 容
iDevice	int	複数のボードを扱う場合のデバイス番号(0..15)
sCommand	String	コマンド文字列
戻り値	Boolean	true: 成功、false:失敗
備考	USBPIO-8 側エンドポイント（チップ内バッファ）のサイズは 64 バイト	

関数名	ReadLine	
機能	パイプの読み出し	
引数	タイプ	内 容
iDevice	int	複数のボードを扱う場合のデバイス番号(0..7)
nTimeout	int	タイムアウト (msec)
戻り値	String	USB から読み出した文字列
備考		

関数名	IsOpen	
機能	パイプのオープン状態を調べる	
引数	タイプ	内 容
iDevice	Integer	複数のボードを扱う場合のデバイス番号(0..15)
戻り値	Boolean	true: オープン、false:非オープン
備考		

4.5.3　USB による I/O 制御(C++ 版)

付属 CD-ROM の Sample Programs フォルダーの USBPIO-8TestCPP は C++(MFC) で作成した USB による I/O 制御サンプルです。
使い方は 4.3 に書いた通りですが、ここではプログラムの USB 制御コアについて説明します。
プロジェクトの起動は USBPIO-8TestCPP　フォルダーの USBPIO-8TestCPP.vcproj　をダブルクリックしておこなえます。
以下にＰＣ側のプログラム階層図を示します。



*) 64bit OS の場合、ドライバーは mchusb64.sys になります。

USBPIO-8TestCPPDlg.cpp から MchUSB.cpp のクラスを呼び出して USB の入出力をおこないます。
以下は USBPIO-8TestCPPDlg.cpp での主な制御関数です。

関数名	機 能
SendCommand	USB にコマンドを送る
OnBnClickedButtonXXX	各ボタンの応答関数(XXX はボタン名)

SendCommand 関数によって USB ドライバーにコマンドが送られて USBPIO-8 ボードがこれを受け取り I/O 制御がおこなわれます。
USB 制御のコアは MchUSB.cpp ファイルに CMchUSB クラスとしてカプセル化されています。
このクラスはマイクロチップ社提供の DLL　mcusbapi.dll を経由して USB ドライバーを呼び出します。　以下はこのクラスの主な関数です。

CMchUSB クラスの概要

関数名	機 能
Initialize	USB の初期化　プログラムの開始時に一度だけ呼んでください
Open	入出力のパイプをオープンする
Close	入出力のパイプをクローズする
Write	出力パイプにデータを書き込む
Read	入力パイプからデータを読み出す

USB の利用手順は

1． CMchUSB クラスのインスタンスを作成する
　サンプルプログラムでは MchUSB.cpp の 32 行目あたりにある
　　CMchUSB　　usbpio;
　でグローバルインスタンスを定義しています。
2． Initialize で USB の初期化をおこなう。
3． Open 関数でパイプをオープンする。
4． Read/Write 関数でパイプに対して読み書きをおこなう
5． 使用が終われば Close でパイプを閉じる
となります。

以下は CMchUSB クラスの関数仕様です。

CMchUSB クラスの関数仕様

関数名	Initialize	
機能	USB の初期化	
引数	タイプ	内 容
なし		
戻り値	BOOL	TRUE: 成功、FALSE:失敗
備考		

関数名	Open	
機能	パイプのオープン	
引数	タイプ	内 容
iDevice	int	複数のボードを扱う場合のデバイス番号(0..7)
戻り値	BOOL	TRUE: 成功、FALSE:失敗
備考		

関数名	Close	
機能	パイプのクローズ	
引数	タイプ	内 容
iDevice	int	複数のボードを扱う場合のデバイス番号(0..7)
戻り値	BOOL	TRUE: 成功、FALSE:失敗
備考		

関数名	Write	
機能	パイプへの書き込み	
引数	タイプ	内 容
iDevice	int	複数のボードを扱う場合のデバイス番号(0..7)
pData	BYTE*	データバッファへのポインタ
nLength	int	データのバイト数
戻り値	int	-1: エラー、> 0: 書き込んだバイト数
備考	USBPIO-24 側エンドポイント（チップ内バッファ）のサイズは６４バイト	

関数名	Read	
機能	パイプの読み出し	
引数	タイプ	内 容
iDevice	int	複数のボードを扱う場合のデバイス番号(0..7)
pData	BYTE*	データバッファへのポインタ
nLength	int	データのバイト数
nTimeout	int	タイムアウト（msec）
戻り値	int	-1: エラー、>=0: 読み込んだバイト数
備考		

4.6　mpusbapi.dll の関数一覧

サンプルの MchUSB クラスを使わずに DLL を直接コールする場合の関数一覧です。
使い方は 4.5.1 ～4.5.3 の MchUSB クラス各言語サンプルソースを参照してください。

関数名	MPUSBGetDeviceCount	
機能	接続されているデバイスの数を返す	
引数	タイプ	内 容
pVID_PID	char*	ベンダーID, プロダクト ID "vid_04d8&pid_ff33"
戻り値	int	接続されているデバイスの数
備考		

関数名	MPUSBOpen	
機能	パイプのオープン	
引数	タイプ	内 容
iDevice	int	デバイス ID
pVID_PID	char*	ベンダーID, プロダクト ID "vid_04d8&pid_ff33"
pEP	char*	エンドポイントの名前 "¥MCHP_EP1"
iDir	int	Write = 0、 Read = 1
iReserved	int	未使用
戻り値	int	パイプのハンドル
備考		

関数名	MPUSBRead	
機能	パイプのリード	
引数	タイプ	内 容
iHandle	int	パイプのハンドル
pData	char*	データバッファへのポインタ
iLen	int	データバッファ長
pLen	int*	読み込んだデータ長を返す変数へのポインタ
iTimeout	int	タイムアウト時間の指定(msec)
戻り値	int	1: 成功、0:失敗
備考		

関数名	MPUSBWrite	
機能	パイプのリード	
引数	タイプ	内 容
iHandle	int	パイプのハンドル
pData	char*	データバッファへのポインタ
iLen	int	データバッファ長
pLen	int*	読み込んだデータ長を返す変数へのポインタ
iTimeout	int	タイムアウト時間の指定(msec)
戻り値	int	1: 成功、0:失敗
備考		

関数名	MPUSBClose	
機能	パイプのクローズ	
引数	タイプ	内 容
iPipe	int	パイプのハンドル
戻り値	int	1: 成功、0:失敗
備考		

５．その他

5.1　64bit OS 対応について

64bit OS 環境では mpusbapi.dll が正しくインストールされない場合があるようですので、その場合は実行ファイル（ EXE ）と同じフォルダーに DLL をコピーしてお使いください。

mpusbapi.dll は USB ドライバー(64bit 版は mchpusb64.sys)をアプリから簡単に呼ぶだめの仕掛け(関数をカプセル化した DLL)で これを経由して USB ドライバーを呼び出します。
64bit パソコンでは USB ドライバーそのものは 64bit 版でなければなりませんが、アプリは 32, 64 どちらからでもドライバーを呼ぶ ことができます。

しかし現在 DLL が 32bit 版しかないため 32bit のアプリ からしか呼び出すことができません。 (64bit アプリから 32bit DLL を呼び出すとエラーになります)

・ したがってユーザー様のプログラムで mpusbapi.dll を使って USBPIO-8 を 制御する場合は 64bit 環境でもアプリは x86(32bit アプリ) として コンパイルしてください。
USB アプリを 32bit(x86) でコンパイルする方法を VB.NET の場合について 以下に説明します。

- VB プロジェクトを立ち上げます。
- 「プロジェクト」メニューの一番下の「xxx のプロパティー」を開きます。
- 左の「コンパイル」タブをクリックして「詳細コンパイルオプション」
  ボタンをクリック。
- 下の「ターゲット CPU」から 「x86」を選択。
- 「OK」ボタンをクリック。

これで「ビルド」メニューでリビルドすれば 32bit アプリでコンパイルされます。

5.2　DLL 呼び出しについて

・ＶＢで制御プログラムを作成する場合、 DLL の関数定義は付属のサンプルプログラムのように文字セットは Ansi で定義してください。　Auto にすると引数のマーシャリングが正しくおこなわれず、エラーになることがあります。

・　また　Public Declare Ansi Function _MPUSBOpen(・・・・
のように関数定義すると DLL の呼び出し規約が cdecl　のため VB 2010 以降ではエラーになります。　サンプルのように面倒でも DllImport を使ってサンプルのように CallingConvention.Cdecl　を指定してください。

```
<DllImport("mpusbapi.dll",CharSet:=CharSet.Ansi,CallingConvention:=CallingConvention.Cdecl)>
    Public Shared Function _MPUSBOpen(ByVal iInstance As Integer, ByVal pVID_PID As String,
                                      ByVal pEP As String, ByVal iDir As Integer,
                                      ByVal iReserved As Integer) As Integer
    End Function
```

5.3　USB の速度について

USB 自体の転送速度は Full Speed 対応で最大 12Mbps ということになっていますが、コマンドを送ってからポートに反映されるまでの時間はパソコンのドライバーとプログラムの条件などによって異なりますのでユーザー様にてご確認願います。
ハンドシェークなしで連続転送した場合、ＰＣ側が速すぎるとオーバーフローする可能性がありますので応答(OK, NG など) を受け取ってから次のコマンドを送ることを推奨いたします。

5.4　オプション製品

・外部 5V スイッチング電源
USB から電源を供給できない場合にご購入ください。

本書の改訂版は当社ホームページの該当製品コーナーよりダウンロードしてください。

Cyber MELON

株式会社インターネット

〒665-0841

兵庫県宝塚市御殿山 2-25-39

http://www.cyber-melon.com

e-mail: info#cyber-melon.com

(# を @ に置き換えてください)