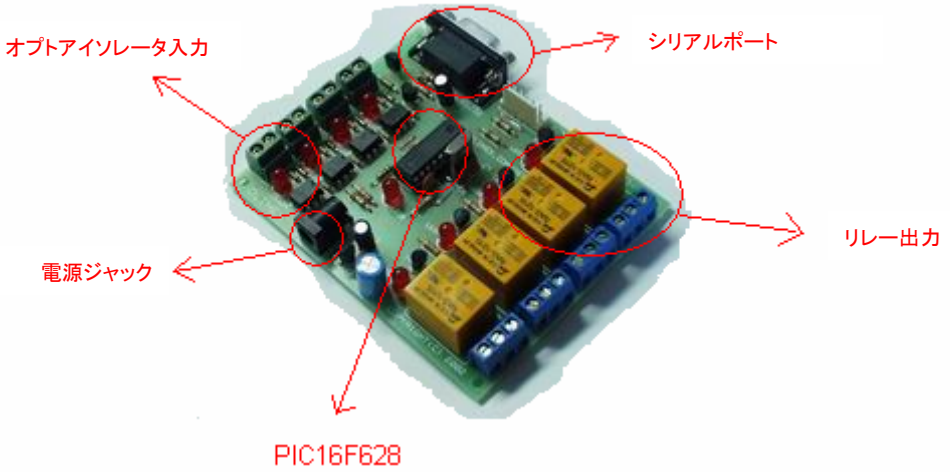
PIC-I/O　これは便利！すぐに使える！リレー、オプトアイソレータ、LED 各 4 個、RS232 内蔵。18 ピン PIC 用 I/O ボード完成品

PIC-I/O アプリケーションノート

イントロダクション：

OLIMEX社のPIC-I/Oは、小さくても、かなり使える開発ボードです。4つのオプトアイソレータ付きの入力ポート、4つのリレー（220V、10A）による出力ポートが搭載されているので、自宅でいろんな機器をオン、オフできます。（注意：AC100Vを取り扱う場合、特に敷設する場合は、第2種電気工事士の資格が必要です。詳しくはマイコンキットドットコムのWEBをご覧ください。）
このボードは、PLC（高速電力線コントローラ）モジュールのように、家の中で離れた場所（部屋）の機器をRS232を通してPCから制御できるボードです。入力チャネルからの信号はPICにより解析され、そして処理されます。同時にPICは入力から、またはPCからの制御信号により出力を制御します。
この短いアプリケーションノートでは、PC　から、グラフィカル・ユーザー・インターフェイス（GUI）を使用して出力をオン、オフする方法を説明しています。このアプリケーションノートで説明しているサンプルソフトウェアを使えば、PC から簡単に機器（照明、小さなモーター、スプリンクラーなど）を制御できます。



このアプリケーションノートで使う物：

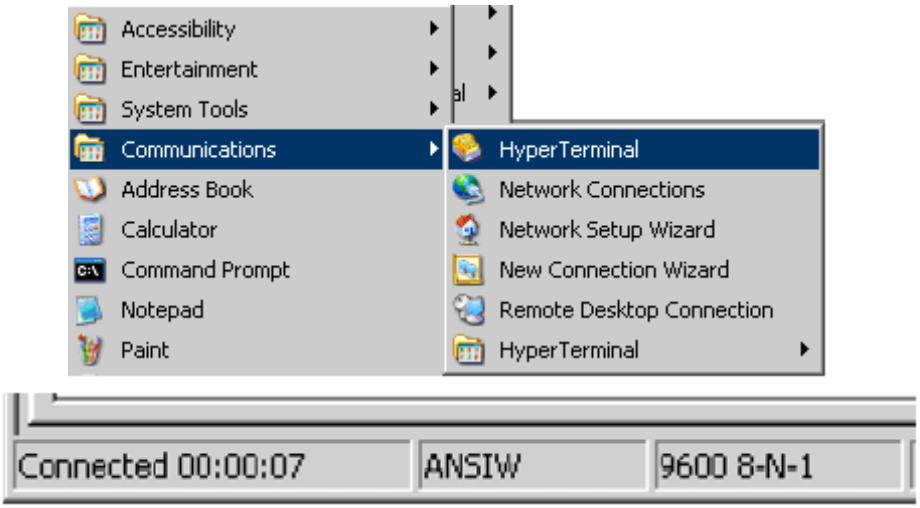
1・・PIC-I/O　開発ボード
1・・16F628マイクロコントローラ（PIC-I/Oに互換性のあるPICであれば、これ以外でも使えると思いますが動作確認は16F628でしか行っておりません。）
1・・PIC-PG1プログラマ（PICをプログラムするため）
1・・シリアルケーブル（作業しやすい適当なもの）
1・・12V電源
1・・Visual Basic 6.0
“stdio.c”はhttp://www.olimex.cl/soft/pic/stdio.c　からダウンロードできます。

このアプリケーションノートで行うこと:

1. PCからPICにコマンドを送ります。
2. PICはこのコマンドを解釈します。
3. PICは受け取ったコマンドに従って出力を制御します。

PIC側が行うこと:

まず、PCとPIC-I/Oボードとの間でやりとりされる「プロトコル」を定義しなければなりません。
ここで使われる「プロトコル」(コマンド)は以下のとおりです:
O <出力チャンネル番号> <出力チャンネル状態>
各項の説明:
「O」(オー)は、英大文字の「O」(オー)です。
<出力チャンネル番号>は、オンオフする出力を識別するチャンネル番号です。入力可能な値は、1、2、3、4です。
<出力チャンネル状態>は、チャンネルの状態を示します。その状態は「A」でアクティブ(オン)、「D」でディアクティブ(オフ)であることを示します。
ここで2つの例を紹介します:コマンド「O1A」は、チャンネル1番の出力をオン(アクティブ)にします。コマンド「O4D」は、チャンネル4番の出力をオフ(ディアクティブ)にします。
このコマンドはシリアルケーブルを通してPCからPIC-I/Oボードに送られます。このテストでは、PCに標準で付属されているハイパーターミナルを使用します。その設定は、9600、8、N、1としてください。



ここでは、PIC がどのように、このコマンドを受けて動くのかを説明します。このドキュメントの C ソースコードは、コピーしてそのまま使えます。なぜ C 言語を使うのか？それは、アセンブラよりも取り扱いが簡単だからです。下記のソースコードをコンパイルするために cc5x を使用しています。もし、PIC マイクロコントローラの C ソースコードのコンパイル方法がわからない場合は、下記の WEB を参照してください(英語)。

http://www.sparkfun.com/tutorial/Setup_Space/setting_up_your_space.htm

このソースコードで行っていることは、いつ PIC が PC からのコマンドを受け取ったかをインターラプトにより確認する処理です。最後の 3 文字を保存し、次に PIC は「0x0D」（ASCII コードで改行の意味）コードを受け取ったときに、そのシーケンスが正しいシーケンスかどうかチェックします。

```
.
#define HS_Osc
#define Serial_Out_BB RB2
#define Serial_In_BB RB7
#define Baud_9600

#include "c:¥cc5x¥16F628.h" //Compiler specific header file
#include "c:¥cc5x¥int16CXX.H" //General Interrupts header file

#pragma origin 4 //Manditory when interrupts are used

#define TRUE 1
#define FALSE 0

bit print_it;
bit start_record;

uns8 data_in;
uns8 data_last1;
uns8 data_last2;
uns8 data_last3;

uns8 memory_array[8];
uns8 mem_spot;

interrupt serverX(void)
{
        int_save_registers
        char sv_FSR = FSR; // save FSR if required

        if(RCIF) //UART Recieve Interrupt
        {
                data_in = RCREG;

                print_it = TRUE;

                //No clearing RCIF, must clear RCREG
                RCREG = 0;
        }

        FSR = sv_FSR; // restore FSR if saved
        int_restore_registers
}

#include "c:¥cc5x¥Delay.c" // Delays
#include "c:¥cc5x¥stdio.c" // Software and Hardware based RS232 Signals

#pragma config |= 0x3F02 //HS Oscillator, CProtect off, WDT off

void main()
{
```

```
PORTA = 0b.0000.0000;
TRISA = 0b.0000.0000; //0 = Output, 1 = Input

PORTB = 0b.0000.0000;
TRISB = 0b.0000.0010; //0 = Output, 1 = Input - RB1 on the 16F628 is RX of the UART

uns8 x;

RB5 = 1; //Turn on the on-board LED
mem_spot = 0;
uns16 button_monitor = 0;

enable_uart_TX(0); //Turn on Transmit UART with TX Interupts Disabled
enable_uart_RX(1); //Turn on Receive UART with RX Interupts Enabled

char aux;
aux = 0x0000;

while(1)
{

	if (print_it == TRUE) //We have something to record
	{
		GIE = 0;

		print_it = FALSE;
		button_monitor = 0; //Reset the LED blinker


		//Store the incoming data to the memory array
		memory_array[mem_spot] = data_in;
		mem_spot++;

		if (mem_spot > 8) mem_spot = 0;


		if (data_in == 0x0D && mem_spot>3) //0x0D is ENTER in ASCII code
		{
			rs_out(12);

			data_last1 = memory_array[mem_spot-2];
			data_last2 = memory_array[mem_spot-3];
			data_last3 = memory_array[mem_spot-4];

			if(data_last3 == 'O')
			{
				switch (data_last2) {
				case '1':
					if (data_last1 == 'A')
					{
						aux |= 0x08;
					}
					if (data_last1 == 'D')
					{
						aux &= 0xF7;
					}
					PORTA = aux;
				break;

				case '2':
					if (data_last1 == 'A')
					{
						aux |= 0x04;
					}
					if (data_last1 == 'D')
					{
```

```
                                    aux &= 0xFB;
                                }
                                PORTA = aux;
                            break;

                            case '3':
                                if (data_last1 == 'A')
                                {
                                    aux |= 0x02;
                                }
                                if (data_last1 == 'D')
                                {
                                    aux &= 0xFD;
                                }
                                PORTA = aux;
                            break;

                            case '4':
                                if (data_last1 == 'A')
                                {
                                    aux |= 0x01;
                                }
                                if (data_last1 == 'D')
                                {
                                    aux &= 0xFE;
                                }
                                PORTA = aux;
                            break;

                            default:
                            break;
                            }
                        }
                    mem_spot = 0;
                    }
                    GIE = 1;

                    CREN = 0;
                    CREN = 1;
                }

                //Blinks the LED when the counter called button_monitor rolls over
                if(button_monitor == 0xFFFF)
                    RB5 ^= 1;
                button_monitor++;
            }
}
```

PC側が行うこと:

前述したようにPCはハイパーターミナルを使用してPICにコマンドを送ります。でも、もっと使いやすいグラフィカルなユーザーインターフェイスを使いたいと思います。そこで、Visual Basic 6を使用して、簡単なグラフィク・ユーザー・インターフェイスを作ります。このソフトでは、各リレーにオン/オフスイッチとCOMポートセレクタを付けています。
まず、オン/オフボタンの処理ルーチンを説明します。

```
Private Sub OutOn_Click(Index As Integer)
        MSComm1.Output = "O"
Delay
        MSComm1.Output = Chr$(Index + &H30 + 1)
Delay
        MSComm1.Output = "A"
Delay
        MSComm1.Output = Chr$(13)
        LedOn (4 + Index)
End Sub
```

ご覧になるとわかるように、PICに一文字送るごとに少しウェイト時間を入れてPICの処理を待っています。

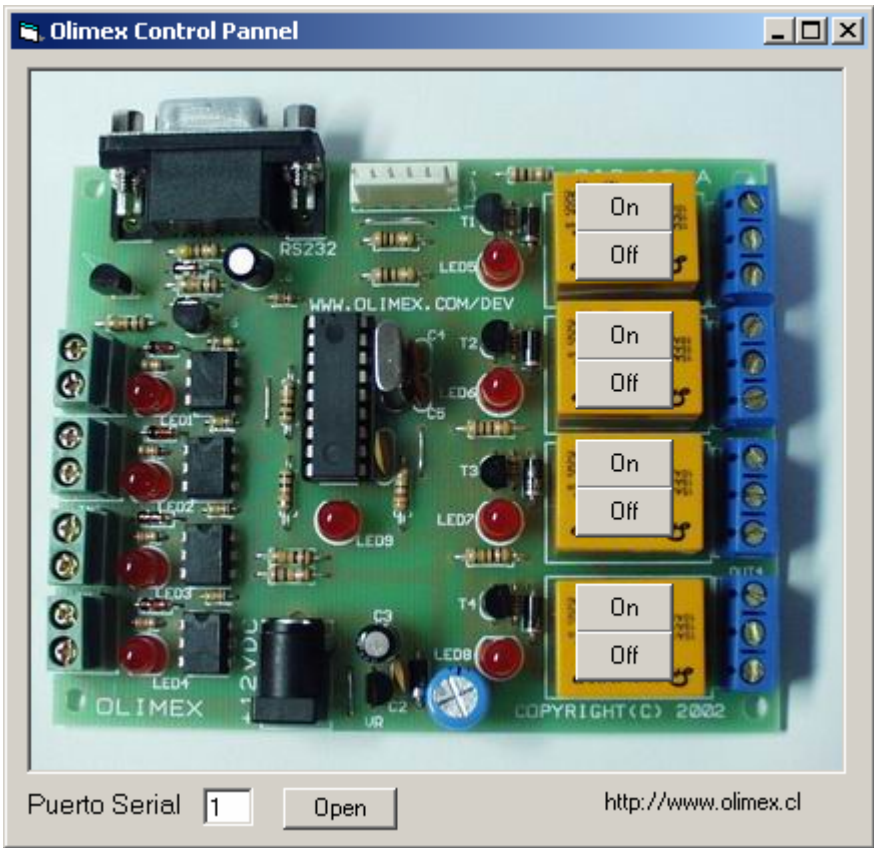
次のような命令文となります：

```
MSComm1.Output = Chr$(Index + &H30 + 1)
```

各ボタンには番号が付いており、それぞれのルーチンを呼ぶときに、そのボタンの番号がパラメータとして使われます。上記の命令は、この番号をASCIIに変換し、シリアルポートをとおして送ります。（ASCIIコードについては　www.asciitable.com　を参照してください）

```
MSComm1.Output = Chr$(13)
```

シリアルポートに「Enter（改行）」文字を送ります（ASCII表で13は0x0D、Enter（改行）の意味）。



VB6でシリアルポートを制御する場合は、このルーチンを使用してください：

```
Private Sub Command1_Click()
    If Command1.Caption = "Open" Then
        MSComm1.CommPort = Text1.Text
        MSComm1.PortOpen = True
        Command1.Caption = "Close"
    Else
        MSComm1.PortOpen = False
        Command1.Caption = "Open"
    End If
End Sub

Private Sub CheckRxData(incomingChars As String)

    crPos = InStr(1, incomingChars, Chr$(13), vbBinaryCompare)
    If crPos <> 0 Then
        rxLine = Mid(incomingChars, 1, crPos - 1)

        For i = 2 To 5
            If Mid(rxLine, i, 1) = "A" Then
                LedOn (i - 2)
            Else
                LedOff (i - 2)
            End If
        Next i
    End If
End Sub

Private Sub MSComm1_OnComm()

    Select Case MSComm1.CommEvent
        Case comEventBreak
            Debug.Print "comEventBreak"
        Case comEventFrame
            Debug.Print "comEventFrame"
        Case comEventOverrun
            Debug.Print "comEventOverrun"
        Case comEventRxOver
            Debug.Print "comEventRxOver"
        Case comEventRxParity
            Debug.Print "comEventRxParity"
        Case comEventTxFull
            Debug.Print "comEventTxFull"
        Case comEventDCB
            Debug.Print "comEventDCB"
        Case comEvCD
            Debug.Print "comEvCD"
        Case comEvCTS
            Debug.Print "comEvCTS"
        Case comEvDSR
            Debug.Print "comEvDSR"
        Case comEvRing
            Debug.Print "comEvRing"
        Case comEvReceive
            CheckRxData (MSComm1.Input)
        Case comEvSend
            Debug.Print "comEvSend"
        Case comEvEOF
            Debug.Print "comEvEOF"
    End Select
End Sub
```

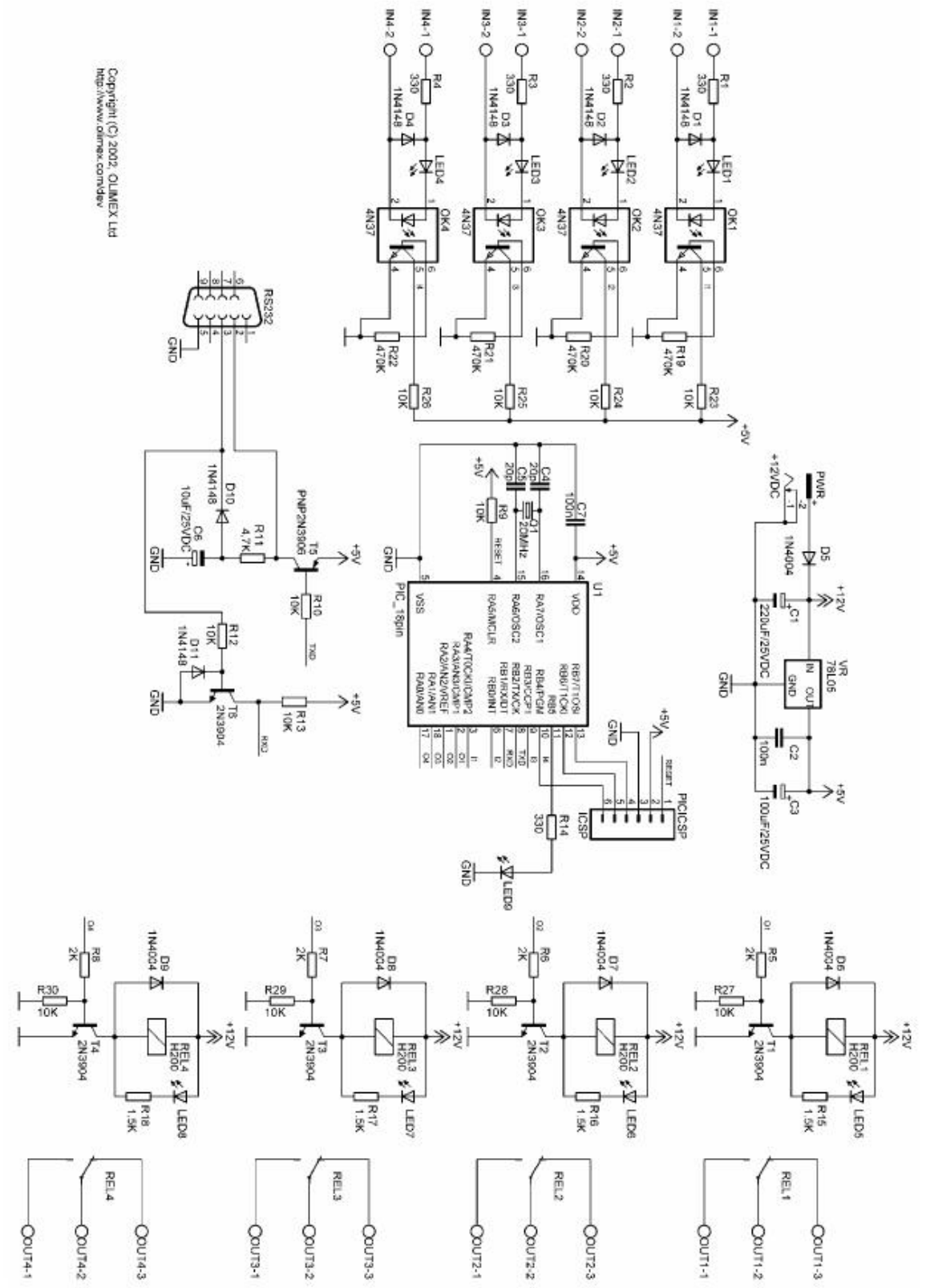
必要なファイルはすべて以下のWEBからダウンロードしてください：

http://www.olimex.cl/pic-io.zip

このインターフェイスを使う場合は、その使用しようとしているPCのCOMポートは使用可能であることを確認してください。ポートを選択した後は、「open」ボタンをクリックし、次に任意のオン/オフボタンをクリックしてください。

PIC-I/O　これは便利！すぐに使える！リレー、オプトアイソレータ、LED 各 4 個、RS232 内蔵。18 ピン PIC 用 I/O ボード完成品

ハードウェア：



PIC-I/O　これは便利！すぐに使える！リレー、オプトアイソレータ、LED 各 4 個、RS232 内蔵。18 ピン PIC 用 I/O ボード完成品

重要: