如何在 C#.2005 中使用 ICPDAS I/O Card 的 DLL 檔案

本文件說明如何在 C#.Net 程式中引入 ICPDAS I/O Card 的 DLL 檔案。

[下載安裝 DLL 驅動程式與 VC 範例程式]

多年來, ICPDAS 完整的提供了全系列 PCI 與 ISA BUS I/O Card 在 Windows 上的 driver 與 DLL 函式庫, 並提供 Microsoft Visual C++, Visual Basic, Borland C++ builder and Delphi 的範例程式暨原始碼幫助使用者開發自己的程式。以下，將介紹如何在 C#.Net application 中呼叫 DLL 的函式。

本文件以 Win2000/XP 作業系統, 搭配 PIO-D56 爲範例。在開始之前請先安裝 Win2000/XP 的 DLL/OCX 驅動程式。以下爲驅動程式 pio_dio_win2k_v207.exe 的 ftp 位址

ftp://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/pio-dio/dll_ocx/win2k_xp/

或者您可以在以下的 CD 目錄中找到:

CD: \NAPDOS\PCI\PIO-DIO\DLL_OCX\Win2K_XP

安裝 DLL/OCX driver 後，請從下面的 ftp 網址下載 VC demo 程式, 解壓縮後, 選用合適的範例程式

ftp://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/pio-dio/dll_ocx/demo/

這個檔案也可以在以下的 CD 目錄找到

\NAPDOS\PCI\PIO-DIO\DLL_OCX\Demo\

選好的 VC demo 程式碼可以直接複製到 C#.Net 上再加以修改

[從 Visual C++ 6.0 範例程式修改]

在下面的 ftp 網址下載 dll_vc6_xxxxxx.exe 檔案:

ftp://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/pio-dio/dll_ocx/demo/

或者在下列的 CD 目錄下找到
CD:\NAPDOS\PCI\PIO-DIO\DLL_OCX\Demo

執行這個自動解壓縮檔，參考其中的 PIODIO.H 檔案與 VC 範例程式的架構建立 C#.Net 專案。使用 "DllImport" 關鍵字將 PIODIO.H 中宣告的函式轉換成 C#.Net 的格式。

舉例來說，PIODIO.H 中 PIODIO_InputByte(ushort wBaseAddr) 函式宣告方式如下。

EXPORTS 函數型態 CALLBACK 函數名稱(資料型態 變數);

將上列程式碼改寫爲

[DllImport("xxxx.dll",EntryPoint="函數名稱")]
public static extern 函數型態 新函數名稱(資料型態 變數);

參考範例：

Example:

```
EXPORTS WORD CALLBACK PIODIO_InputByte(DWORD wBase);

將上列程式碼改寫爲

[DllImport("Piodio.dll",EntryPoint="PIODIO_InputByte")]
        public static extern ushort InputByte(uint wBase);
```

再加入類別中。

修改函式宣告之後，您可以使用以下的方法來呼叫函式

InVal1 = PIODIO. InputByte(wBaseAddr + 0xC0);
InVal2 = PIODIO. InputByte(wBaseAddr + 0xC4);
InVal3 = PIODIO. InputByte(wBaseAddr + 0xC8);

[參考資料 ： 資料型態映照表]

Bytes	VC++ 6 data type	C# data type
2	Short WORD	Short Ushort
4	Int unsigned int	Uint
4	unsigned long DWORD	Uint
4	Float	Float
8	Double	double
4	Int *	out int
4	float *	out float

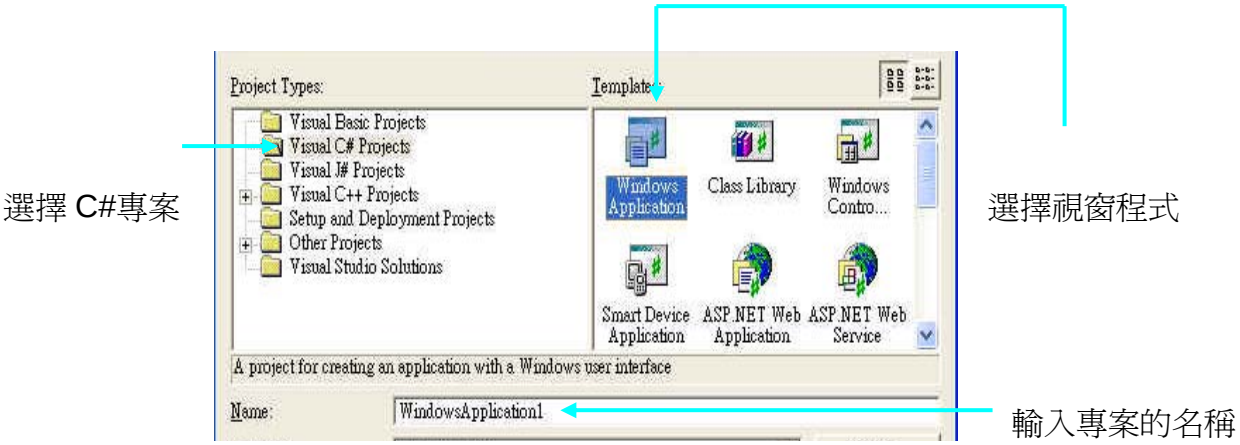
參考文件 ：**PIODIO** 軟體手冊, 介紹 **PIODIO.dll** 所有函式功能與參數

ftp://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/pio-dio/manual/pio-dio_dll_software_manual.pdf

以下爲更詳細的步驟:

步驟 1.

開啓 Visual Studio.Net 然後點選 File->New ->Project。如下圖所示去建立一個新專案



步驟 2.

在程式的開頭增加以下的程式碼

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Runtime.InteropServices;
using System.Threading;
```

步驟 3.

修改 PIODIO.h 的 function 宣告，完成後加入類別（程式）中。

PIODIO.h 的相關函式原始定義如下:

```
// Driver functions
```

```
EXPORTS WORD   CALLBACK PIODIO_DriverInit(void);
EXPORTS void   CALLBACK PIODIO_DriverClose(void);
EXPORTS WORD   CALLBACK PIODIO_SearchCard(WORD *wBoards, DWORD dwPIOCardID);
EXPORTS WORD   CALLBACK PIODIO_GetDriverVersion(WORD *wDriverVersion);
EXPORTS WORD   CALLBACK PIODIO_GetConfigAddressSpace(
   WORD  wBoardNo, DWORD *wAddrBase, WORD *wIrqNo, WORD *wSubVendor, WORD *wSubDevice
   WORD *wSubAux, WORD *wSlotBus,  WORD *wSlotDevice);
EXPORTS WORD   CALLBACK PIODIO_ActiveBoard( WORD wBoardNo );
EXPORTS WORD   CALLBACK PIODIO_WhichBoardActive(void);

// DIO functions
EXPORTS void   CALLBACK PIODIO_OutputWord(DWORD wPortAddress, DWORD wOutData);
EXPORTS void   CALLBACK PIODIO_OutputByte(DWORD wPortAddr, WORD bOutputValue);
EXPORTS DWORD  CALLBACK PIODIO_InputWord(DWORD wPortAddress);
EXPORTS WORD   CALLBACK PIODIO_InputByte(DWORD wPortAddr);

// Interrupt functions
EXPORTS WORD   CALLBACK PIODIO_IntInstall( WORD wBoardNo, HANDLE *hEvent,
   WORD wInterruptSource, WORD wActiveMode);
EXPORTS WORD   CALLBACK PIODIO_IntRemove(void);
EXPORTS WORD   CALLBACK PIODIO_IntResetCount(void);
EXPORTS WORD   CALLBACK PIODIO_IntGetCount(DWORD *dwIntCount);

// PIOD48 Counter functions
EXPORTS void   CALLBACK PIOD48_SetCounter
   (DWORD dwBase, WORD wCounterNo, WORD bCounterMode, DWORD wCounterValue);
EXPORTS DWORD  CALLBACK PIOD48_ReadCounter
   (DWORD dwBase, WORD wCounterNo, WORD bCounterMode);
EXPORTS void   CALLBACK PIOD48_SetCounterA
   (WORD wCounterNo, WORD bCounterMode, DWORD wCounterValue);
EXPORTS DWORD  CALLBACK PIOD48_ReadCounterA(WORD wCounterNo, WORD bCounterMode);

// PIOD48 Interrupt functions
EXPORTS WORD   CALLBACK PIOD48_IntInstall
   (WORD wBoardNo, HANDLE *hEvent, WORD wIrqMask, WORD wActiveMode);
EXPORTS WORD   CALLBACK PIOD48_IntRemove();
EXPORTS WORD   CALLBACK PIOD48_IntGetActiveFlag (WORD *bActiveHighFlag, WORD *bActiveLowFlag);
EXPORTS WORD   CALLBACK PIOD48_IntGetCount(DWORD *dwIntCount);
```

```
// PIOD64 Counter functions
EXPORTS void   CALLBACK PIOD64_SetCounter
   (DWORD dwBase, WORD wCounterNo, WORD bCounterMode, DWORD wCounterValue);
EXPORTS DWORD  CALLBACK PIOD64_ReadCounter
   (DWORD dwBase, WORD wCounterNo, WORD bCounterMode);
EXPORTS void   CALLBACK PIOD64_SetCounterA
   (WORD wCounterNo, WORD bCounterMode, DWORD wCounterValue);
EXPORTS DWORD  CALLBACK PIOD64_ReadCounterA(WORD wCounterNo, WORD bCounterMode);

// PIOD48 Frequency Measurement functions
EXPORTS DWORD  CALLBACK PIOD48_Freq(DWORD dwBase);
EXPORTS DWORD  CALLBACK PIOD48_FreqA();
```

宣告一個類別並且將使用到的函式匯入到程式中:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Runtime.InteropServices;

namespace PIODIO_Ns
{
    public class PIODIO

    {
        //****************
        //PIODIO CARD ID
        //****************

        public const uint PIOD_24=0x800140;
        public const uint PIOD_48=0x800130;
        public const uint PIOD_56=0x800140;
        public const uint PIOD_64=0x800120;
        public const uint PIOD_96=0x800110;
        public const uint PIOD_144=0x800100;
        public const uint PIOD_168=0x98800150;
```

```
public const uint PIOD_168A=0x800150;

//****************
//Error Code
//****************

public const uint NoError = 0;
public const uint DriverOpenError = 1;
public const uint DriverNoOpen = 2;
public const uint GetDriverVersionError = 3;
public const uint InstallIrqError = 4;
public const uint ClearIntCountError = 5;
public const uint GetIntCountError = 6;
public const uint RegisterApcError = 7;
public const uint RemoveIrqError = 8;
public const uint FindBoardError = 9;
public const uint ExceedBoardNumber = 10;
public const uint ResetError = 11;
public const uint IrqMaskError = 12;
public const uint ActiveModeError = 13;
public const uint GetActiveFlagError = 14;
public const uint ActiveFlagEndOfQueue = 15;

//*****************
//PIODIO ActiveMode
//*****************
// to trigger a interrupt when low -> high
public const uint ActiveHigh =1;
// to trigger a interrupt when high -> low
public const uint  ActiveLow=0;

//*********************************
//define the interrupt signal source
//*********************************
public const uint PIOD144_P2C0 = 0;   // pin29 of CN1(37 pin D-type, pin1 to pin37)
public const uint PIOD144_P2C1 = 1;   // pin28 of CN1(37 pin D-type, pin1 to pin37)
public const uint PIOD144_P2C2 = 2;   // pin27 of CN1(37 pin D-type, pin1 to pin37)
public const uint PIOD144_P2C3 = 3;   // pin26 of CN1(37 pin D-type, pin1 to pin37)
```

```
//*********************************
// Interrupt Channel for PIO-D48
//*********************************
public const uint PIOD48_INTCH0 = 1;  // INT_CHAN_0
public const uint PIOD48_INTCH1 = 2;  // INT_CHAN_1
public const uint PIOD48_INTCH2 = 4;  // INT_CHAN_2
public const uint PIOD48_INTCH3 = 8;  // INT_CHAN_3

//*********************************
//Test functions
//*********************************

[DllImport ("Piodio.dll",EntryPoint ="PIODIO_FloatSub")]
 public static extern float FloatSub(float fA,float fB);
[DllImport ("Piodio.dll",EntryPoint ="PIODIO_ShortSub")]
 public static extern short ShortSub(short nA,short nB);

[DllImport ("Piodio.dll",EntryPoint ="PIODIO_GetDllVersion")]
 public static extern ushort GetDllVersion();

//**************
// PIODIO Driver
//**************
[DllImport("Piodio.dll",EntryPoint="PIODIO_DriverInit")]
 public static extern ushort DriverInit();

 [DllImport("Piodio.dll",EntryPoint="PIODIO_DriverClose")]
 public static extern void DriverClose();
 [DllImport("Piodio.dll",EntryPoint="PIODIO_SearchCard")]
 public static extern ushort SearchCard(out ushort wBoards, uint dwPIOCardID);
 [DllImport ("Piodio.dll",EntryPoint ="PIODIO_GetDriverVision")]
 public static extern ushort GetDriverVersion(out ushort wDriverVersion);

 [DllImport("Piodio.dll",EntryPoint="PIODIO_GetConfigAddressSpace")]
 public static extern ushort GetConfigAddressSpace(
     ushort wBoardNo, out uint wAddrBase, out ushort wIrqNo,
     out ushort wSubVendor, out ushort wSubDevice, out ushort wSubAux,
```

```
    out ushort wSlotBus, out ushort wSlotDevice);
[DllImport("Piodio.dll",EntryPoint="PIODIO_ActiveBoard")]
public static extern ushort ActiveBoard(ushort wBoardNo);
[DllImport("Piodio.dll",EntryPoint="PIODIO_WhichBoardActive")]
public static extern ushort WhichBoardActive();

// ***************************************
[DllImport("Piodio.dll",EntryPoint="PIODIO_OutputByte")]
public static extern void OutputByte(uint wBaseAddr, ushort bOutputValue);
[DllImport("Piodio.dll",EntryPoint="PIODIO_InputByte")]
public static extern ushort InputByte(uint wBaseAddr);

//********************
//PIODIO Interrupt
//********************

[DllImport("Piodio.dll", EntryPoint = "PIODIO_IntInstall")]
public static extern ushort IntInstall(ushort wBoardNo, out uint hEvent, _
ushort wInterruptSource, ushort wActiveMode);
[DllImport("Piodio.dll", EntryPoint = "PIODIO_IntRemove")]
public static extern ushort IntRemove();

[DllImport("Piodio.dll", EntryPoint = "PIODIO_IntGetCount")]
public static extern ushort IntGetCount(out uint dwIntCount);

[DllImport("Piodio.dll", EntryPoint = "PIODIO_IntResetCount")]
public static extern ushort IntResetCount();

//********************
//PIODIO_48 Frequency
//********************
[DllImport("Piodio.dll")]
public static extern uint PIOD48_Freq(uint wBaseAddr);

//*********************
//PIODIO_48 Counter
//*********************
[DllImport("Piodio.dll")]
```

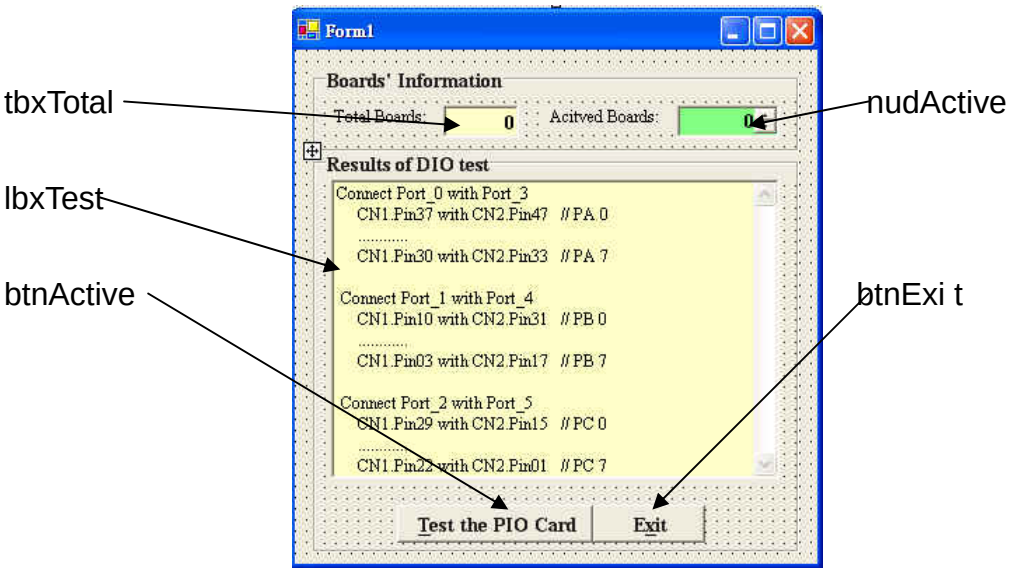
```
public static extern void PIOD48_SetCounter(uint dwBase,ushort wCounterNo,_
ushort bCounterMode,uint wCounterValue );
[DllImport("Piodio.dll")]
public static extern uint PIOD48_ReadCounter(uint dwBase,ushort wCounterNo,_
ushort bCounterMode);
[DllImport ("Piodio.dll")]
public static extern void PIOD48_SetCounterA(ushort wCounterNo, _
ushort bCounterMode,uint wCounterValue);
[DllImport ("Piodio.dll")]
public static extern uint PIOD48_ReadCounterA(ushort wCounterNo,ushort bCounterMode);

//**********************
//PIODIO_48 Interrupt
//**********************
[DllImport ("Piodio.dll")]
public static extern ushort PIOD48_IntInstall(ushort wBoardNo, out uint hEvent,_
ushort  wIrqMask, ushort  wActiveMode);

[DllImport ("Piodio.dll")]
public static extern ushort PIOD48_IntRemove();
[DllImport ("Piodio.dll")]
public static extern ushort PIOD48_IntGetActiveFlag(out ushort bActiveHighFlag, _
out ushort bActiveLowFlag);
[DllImport ("Piodio.dll")]
public static extern ushort PIOD48_IntGetCount(out uint dwIntCount);

//********************
//PIODIO_64 Counter
//********************
[DllImport("Piodio.dll")]
public static extern void  PIOD64_SetCounter(uint dwBase,ushort wCounterNo,_
ushort bCounterMode,uint wCounterValue);
[DllImport("Piodio.dll")]
public static extern uint PIOD64_ReadCounter(uint dwBase,ushort wCounterNo,_
ushort bCounterMode);

[DllImport("Piodio.dll")]
public static extern void PIOD64_SetCounterA(ushort wCounterNo, _
```

```
        ushort bCounterMode,uint wCounterValue);
        [DllImport("Piodio.dll")]
        public static extern uint PIOD64_ReadCounterA(ushort wCounterNo, ushort bCounterMode);

        // ****************************************
        private int DriverOpened = 0;

        public PIODIO()//constroctor
        {
            DriverOpened = 0;
        }
        ~PIODIO()
        {
            if (DriverOpened != 0)
            {
                DriverOpened = 0;
                DriverClose();
            }
        }
    }
}
```

步驟 4.

設計程式和 DLL 函式應用

設計介面:



函式應用:

```
namespace PIOD56_Demo
{
    public partial class Form1 : Form
    {
        public uint wBaseAddr;
        public ushort wInitialCode, wTotalBoards, wIrq, wSubVendor, wSubDevice, wSubAux,
wSlotBus, wSlotDevice;
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {

            btnActive.Enabled = false;
            if ((wInitialCode = PIODIO .DriverInit ()) != 0)
            {
                MessageBox.Show("Driver initialize error!!!");
                return;
            }

            if ((wInitialCode = PIODIO.SearchCard (out wTotalBoards, PIODIO.PIOD_56)) != 0)
            {
                MessageBox.Show("SearchCard Error");

                return;
            }

            tbxTotal.Text = wTotalBoards.ToString();
            nudActive.Maximum = wTotalBoards - 1;
            nudActive.Minimum = 0;
            btnActive.Enabled = true;
        }
```

```
        private void btnExit_Click(object sender, EventArgs e)
        {
            PIODIO.DriverClose();
            Close();
        }

        private void btnActive_Click(object sender, EventArgs e)
        {

            ushort InVal0,InVal1,InVal2,wRst;
            lbxTest .Items .Clear ();
            if (Convert.ToInt16(nudActive.Value) < 0 || Convert.ToInt16(nudActive.Value) >
Convert.ToInt16(tbxTotal.Text ))
            {
                lbxTest.Items.Add("Invalid board number,Please Retry!!!");
                btnActive.Enabled = false;
                return;
            }
            wRst = PIODIO.GetConfigAddressSpace((ushort)Convert.ToInt16(nudActive.Value), out
wBaseAddr, out wIrq, out wSubVendor, out wSubDevice, out wSubAux, out wSlotBus, out wSlotDevice);

            if(wRst !=0)
            {
                MessageBox .Show ("Get Config-Address-Space Error!!");
                btnActive.Enabled = false;
                return ;

            }
            //*************************//
            //Enable all DI/DO port    //
            //*************************//
            lbxTest.Items.Add("Enable All DI/DO");
            PIODIO.OutputByte(wBaseAddr, (ushort)1); //Enable I/O function
            lbxTest.Items.Add("");
            lbxTest.Items.Add("Setting Port 0 to Output-Mode and Port 1, 2 to Input-Mode");
            PIODIO.OutputByte(wBaseAddr + 0xCC, (ushort)0x01); //Setting Port 0 Output
```

```
            ushort ii = 1;
            while (ii <= (ushort)0x80)
            {
                PIODIO.OutputByte((wBaseAddr + 0xC0), (ushort)ii);
                InVal1 = PIODIO.InputByte(wBaseAddr + 0xC4);
                InVal2 = PIODIO.InputByte(wBaseAddr + 0xC8);

                lbxTest.Items.Add("Output Port 0 (Hex)= " + Convert.ToString(ii, 16));
                lbxTest.Items.Add("Input Port 1,2 (Hex)= " + Convert.ToString(InVal1, 16) +"
"+Convert.ToString(InVal2, 16));
                Thread.Sleep(100);
                Application.DoEvents();
                ii*= 2;
            }


            lbxTest.Items.Add("");
            lbxTest.Items.Add("Setting Port 1 to Output-Mode and Port 0, 2 to Input-Mode");
            PIODIO.OutputByte(wBaseAddr + 0xCC, (ushort)0x02); //Setting Port 1 Output
            ii = 1;
            while (ii <= (ushort)0x80)
            {
               PIODIO.OutputByte(wBaseAddr + 0xC4, (ushort)ii);
                InVal0 = PIODIO.InputByte(wBaseAddr + 0xC0);
                InVal2 = PIODIO.InputByte(wBaseAddr + 0xC8);

                lbxTest.Items.Add("Output Port 1 (Hex)= " + Convert.ToString(ii, 16));
                lbxTest.Items.Add("Input Port 0,2 (Hex)= " + Convert.ToString(InVal0, 16) + "
" + Convert.ToString(InVal2, 16));
                Thread.Sleep(100);
                Application.DoEvents();
                ii *= 2;
            }
            lbxTest.Items.Add("");
            lbxTest.Items.Add("Setting Port 2 to Output-Mode and Port 0, 1 to Input-Mode");
            PIODIO.OutputByte(wBaseAddr + 0xCC, (ushort)0x04); //Setting Port 2 Output
            ii=1;
            while (ii <= (ushort)0x80)
```

```
            {
                PIODIO.OutputByte(wBaseAddr + 0xC8, (ushort)ii);
                InVal0 = PIODIO.InputByte(wBaseAddr + 0xC0);
                InVal1 = PIODIO.InputByte(wBaseAddr + 0xC4);

                lbxTest.Items.Add("Output Port 2 (Hex)= " + Convert.ToString(ii, 16));
                lbxTest.Items.Add("Input Port 0,1 (Hex)= " + Convert.ToString(InVal0, 16) + "
" + Convert.ToString(InVal1, 16));
                Thread.Sleep(100);
                Application.DoEvents();
                ii *= 2;
            }
            lbxTest.Items .Add ("");
            lbxTest.Items.Add("Digital-Input/Digital-Output (CON1 and CON2)");

            ii = 1;
            while (ii <= (ushort)0x80)
            {
                PIODIO.OutputByte(wBaseAddr + 0xD0, (ushort)ii);
                PIODIO.OutputByte(wBaseAddr + 0xD4, (ushort)ii);
                InVal1 = PIODIO.InputByte(wBaseAddr + 0xD0);
                InVal2 = PIODIO.InputByte(wBaseAddr + 0xD4);

                lbxTest.Items.Add("Digital-Output (Hex)= "+Convert.ToString (ii,16)+"
"+Convert .ToString(ii,16));
                lbxTest.Items.Add("Digital-Input(Hex)= " + Convert.ToString(InVal1, 16) + " "
+ Convert.ToString(InVal2, 16));
                Thread.Sleep(100);
                Application.DoEvents();
                ii *= 2;
            }
        }
    }
}
```