Using the ICPDAS I/O card DLL in C#

This document describes how to use the ICPDAS I/O card DLL file in a C# application.

[DLL driver and demo file related information]

In the past, ICPDAS has provided the relevant DLL files for our various I/O cards for use in applications that were developed by our customers using Microsoft Visual C++, Visual Basic, Borland C++ builder and Delphi. By following the instructions in this document, it will be possible to use our DLL in a C# application.

The following instructions will use the PIO-D48 add-on card in Win2000/XP as a demo. Before this issue, please install the DLL/OCX driver for Win2000/XP first. Download the pio_dio_win2k_v205.exe file from the ftp site:

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

or from the attached CD path:

\NAPDOS\PCI\PIO-DIO\DLL_OCX\Driver\

After installing the DLL/OCX driver, select the suitable demo and download the existing VC sample program from the ftp site:

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

or from the attached CD path:

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

The source code of VC sample programs can then be copied, pasted and modified to C# code.

[To modify from Visual C++ 6.0]

Download the dll_vc6_xxxxxx.exe file from the ftp site:

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

or from the attached CD path:

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

Extract the file to a local directory. Refer to the PIODIO.h file and the program structure of the existing sample programs to create your C# project. Insert the name of the functions that will be used in your application into your class using "DllImport." For example, imagine an application has a class named PIODIO that will use a function called "PIODIO_InputByte(ushort wBaseAddr)," which is declared in the PIODIO.h file.

To import

EXPORTS WORD CALLBACK PIODIO_InputByte(DWORD wPortAddr);

into the class, insert the following two lines into your code:

```
[DllImport("Piodio.dll")]
    public static extern ushort PIODIO_InputByte(ushort wBaseAddr);
```

Refer to

ftp://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/piso-dio/manual/PISO-DIO_Win32_SDK_Manual.pdf

for more information about using ICPDAS functions.

After adding the above two lines, the function can be called in an application in the following manner:

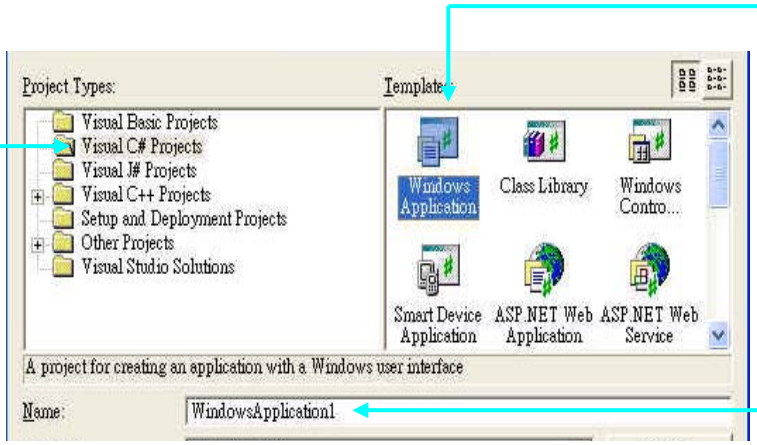
```
InVal1 = PIODIO.PIODIO_InputByte((ushort)(wBaseAddr + 0xC0));
InVal2 = PIODIO.PIODIO_InputByte((ushort)(wBaseAddr + 0xC4));
InVal3 = PIODIO.PIODIO_InputByte((ushort)(wBaseAddr + 0xC8));
```

A detailed description of the procedure is as follows:

Step 1.

Start Visual Studio.Net and select File->New ->Project. Refer to the following figure for details of how to create a new project.

Select the C# Project.



Select the Windows application.

Enter the project name here.

Step 2.

Add the following lines at the start of the code.

```
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;
```

Step 3.

Import the function declaration from the PIODIO.h file into the code.

The function declarations in the original PIODIO.h file:

```
// 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();
```

Declare a class and Import the function to be used in the application:

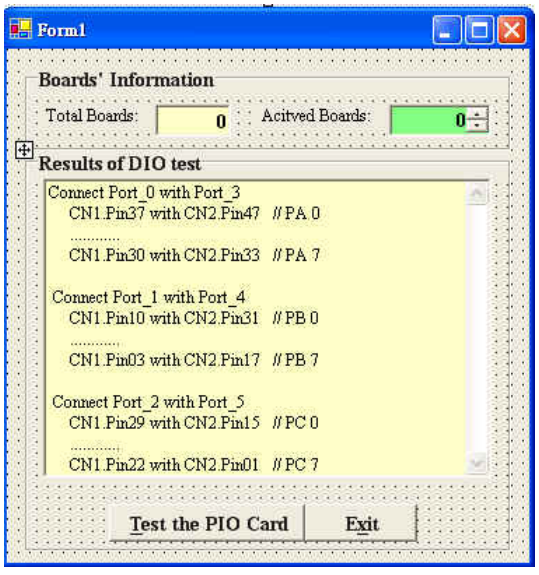
```
public class PIODIO
{
     [DllImport("Piodio.dll")]
     public static extern ushort PIODIO_DriverInit();
     [DllImport("Piodio.dll")]
     public static extern void PIODIO_DriverClose();
     [DllImport("Piodio.dll")]
     public static extern ushort PIODIO_SearchCard(out ushort wBoards, uint dwPIOCardID);
     [DllImport("Piodio.dll")]
     public static extern ushort PIODIO_GetConfigAddressSpace(
          ushort wBoardNo, out ushort wAddrBase, out ushort wIrqNo,
          out ushort wSubVendor, out ushort wSubDevice, out ushort wSubAux,
          out ushort wSlotBus, out ushort wSlotDevice);

     // ****************************************
     [DllImport("Piodio.dll")]
     public static extern void PIODIO_OutputByte(ushort wBaseAddr, ushort bOutputValue);
     [DllImport("Piodio.dll")]
```

```
        public static extern ushort PIODIO_InputByte(ushort wBaseAddr);
        // ****************************************
        private int   DriverOpened = 0;
        // ****************************************
        // ****************************************
        public PIODIO()//constroctor
        {
              DriverOpened = 0;
        }
        ~PIODIO()
        {
              if ( DriverOpened != 0 )
              {
                    DriverOpened = 0;
                    PIODIO_DriverClose();
              }
        }
}
```

Step 4.

Designing the application and using the DLL functions.

Designing the GUI:



Using the function:

```
namespace PIOD48_DIO
{
    public class Form1 : System.Windows.Forms.Form
    {
        private System.Windows.Forms.GroupBox groupBox1;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.GroupBox groupBox2;
        private System.Windows.Forms.ListBox lbxRst;
        private System.Windows.Forms.Button btnTestDIO;
        private System.Windows.Forms.Button btnExit;
        private System.Windows.Forms.TextBox tbxBoards;
        private System.Windows.Forms.NumericUpDown nudActiveBoards;

        private System.ComponentModel.Container components = null;
        public ushort wInitialCode, wTotalBoards, wBaseAddr, wIrq, wSubVendor, wSubDevice,
            wSubAux, wSlotBus, wSlotDevice;
        PIODIO piodio = null;

        public Form1()
        {
            InitializeComponent();
        }

        protected override void Dispose( bool disposing )
        {
            …
        }

        #region Windows Form Designer generated code
        private void InitializeComponent()
        {
            …
        }
        #endregion
```

```
[STAThread]
static void Main()
{
      Application.Run(new Form1());
}

private void Form1_Load(object sender, System.EventArgs e)
{
      piodio = new PIODIO();
      const uint PIO_D48 = 0x800130;

      if((wInitialCode = PIODIO.PIODIO_DriverInit()) != 0)
      {
            MessageBox.Show("Driver initialize error!!!");
            return;
      }
      btnTestDIO.Enabled = true;
      if((wInitialCode = PIODIO.PIODIO_SearchCard(out wTotalBoards, PIO_D48)) != 0)
      {
            MessageBox.Show("Search Card Error!!");
            return;
      }
      tbxBoards.Text = wTotalBoards.ToString();
      nudActiveBoards.Maximum = 0;
      nudActiveBoards.Minimum = wTotalBoards - 1;
}

private void btnExit_Click(object sender, System.EventArgs e)
{
      Close();
}

private void btnTestDIO_Click(object sender, System.EventArgs e)
{
      uint i;
      ushort InVal1, InVal2, InVal3, wRst;

      lbxRst.Items.Clear();
```

```
            if(Convert.ToInt16(nudActiveBoards.Value) != 0 ||
             Convert.ToInt16(nudActiveBoards.Value) > Convert.ToInt16(tbxBoards.Text))
            {
                  lbxRst.Items.Add("Invalid board number, Pls retry!!");
                  return;
            }
            wRst = PIODIO.PIODIO_GetConfigAddressSpace(
                  (ushort)Convert.ToInt16(nudActiveBoards.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 !!");
                  return;
            }

            //************************************************************
            // step 1: enable all DI/DO port
            //************************************************************
            lbxRst.Items.Add("Enable All DI/DO ");
            PIODIO.PIODIO_OutputByte(wBaseAddr,(ushort)1);//Enable DI/O function
            lbxRst.Items.Add("");

            lbxRst.Items.Add("Setting Port 0, 1, 2 to Output-Mode");
            //set 8254-1 PA,Pb,PC(port0,1,2) as DO
            PIODIO.PIODIO_OutputByte((ushort)(wBaseAddr + 0xCC), (ushort)0x80);
            lbxRst.Items.Add("Setting Port 3, 4, 5 to Input-Mode");
            //set 8254-2 PA,Pb,PC(port3,4,5) as DI
            PIODIO.PIODIO_OutputByte((ushort)(wBaseAddr + 0xDC), (ushort)0x9B);
lbxRst.Items.Add("");

            ushort ii = 1;
            while (ii <= (ushort)0x80)
            {
                  PIODIO.PIODIO_OutputByte((ushort)(wBaseAddr + 0xc0), ii); // Port 0
                  PIODIO.PIODIO_OutputByte((ushort)(wBaseAddr + 0xc4), ii); // Port 1
                  PIODIO.PIODIO_OutputByte((ushort)(wBaseAddr + 0xc8), ii); // Port 2
                  lbxRst.Items.Add("Output Port 2, 1, 0 (Hex)=  "
```

```
            + Convert.ToString(ii,16) + " "
            + Convert.ToString(ii,16) + " "
            + Convert.ToString(ii,16));

      InVal1 = PIODIO.PIODIO_InputByte((ushort)(wBaseAddr + 0xD0)); // Port 3
      InVal2 = PIODIO.PIODIO_InputByte((ushort)(wBaseAddr + 0xD4)); // Port 4
      InVal3 = PIODIO.PIODIO_InputByte((ushort)(wBaseAddr + 0xD8)); // Port 5
      lbxRst.Items.Add("  Input Port 5, 4, 3 (Hex)=  "
            + Convert.ToString(InVal3, 16) + " "
            + Convert.ToString(InVal2, 16) + " "
            + Convert.ToString(InVal1));

      lbxRst.Items.Add("Delay 100 mSec.");
      lbxRst.Items.Add("");
      Thread.Sleep(100);
      ii <<= 2;
}

lbxRst.Items.Add("");
lbxRst.Items.Add("");
lbxRst.Items.Add("Setting Port 3, 4, 5 to Output-Mode");
PIODIO.PIODIO_OutputByte((ushort)(wBaseAddr + 0xDC), (ushort)0x80);
lbxRst.Items.Add("Setting Port 0, 1, 2 to Input-Mode");
PIODIO.PIODIO_OutputByte((ushort)(wBaseAddr + 0xCC), (ushort)0x9B);
lbxRst.Items.Add("");

ii = 1;
while (ii <= 0x80)
{
      PIODIO.PIODIO_OutputByte((ushort)(wBaseAddr + 0xD0), ii); // Port 3
      PIODIO.PIODIO_OutputByte((ushort)(wBaseAddr + 0xD4), ii); // Port 4
      PIODIO.PIODIO_OutputByte((ushort)(wBaseAddr + 0xD8), ii); // Port 5
      lbxRst.Items.Add( "Output Port 5, 4, 3 (Hex)=  "
            + Convert.ToString(ii, 16) + " "
            + Convert.ToString(ii, 16) + " "
            + Convert.ToString(ii, 16));

      InVal1 = PIODIO.PIODIO_InputByte((ushort)(wBaseAddr + 0xC0)); // Port 0
```

```
                InVal2 = PIODIO.PIODIO_InputByte((ushort)(wBaseAddr + 0xC4)); // Port 1
                InVal3 = PIODIO.PIODIO_InputByte((ushort)(wBaseAddr + 0xC8)); // Port 2
                lbxRst.Items.Add("  Input Port 2, 1, 0 (Hex)=  "
                    + Convert.ToString(InVal3, 16) + " "
                    + Convert.ToString(InVal2, 16) + " "
                    + Convert.ToString(InVal1, 16));

                lbxRst.Items.Add("Delay 100 mSec.");
                lbxRst.Items.Add("");
                Thread.Sleep(100);
                ii <<= 2;
            }
            lbxRst.Items.Add("   The End  ");
        }
    }
}
```

[Data type mapping table]

VC++ 6 data type	C# data type
int	short
unsigned int WORD	ushort
unsigned long DWORD	uint
Float	float
double	double
Int *	ref int
float *	ref float

Writer: Tony Lee (2006/01)