
Using the ICPDAS I/O card DLL in C# 2005

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 various I/O cards for users to drive I/O cards in Microsoft Visual C++, Visual Basic, Borland C++ builder and Delphi. By following the instructions in this document, it will be possible to use the DLLs in a C# application.

The following instructions will use the PIO-D56 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_v207.exe file from the ftp site:

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

or from the attached CD path:

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

After installing the DLL/OCX driver, 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:

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

The source code of VC sample programs can 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:

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

Extract the file to a local directory and select a suitable demo. Refer to the PIODIO.h file and the program structure to create your C# project. Insert the declaration of the functions into your class by using "DllImport" decoration.

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 FunctionType CALLBACK FunctionName(DataType arg1);

into the class, please modify the declaration as:

[DllImport("XXXX.dll",EntryPoint="FunctionName")]

public static extern Datatype NewFunctionName(Datetype arg1);

Refer to example

Example :

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

Convert to C# 2005

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

add to class .

[Data type mapping table]

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

Refer to

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

for more information about the functions in PIODIO.dll.

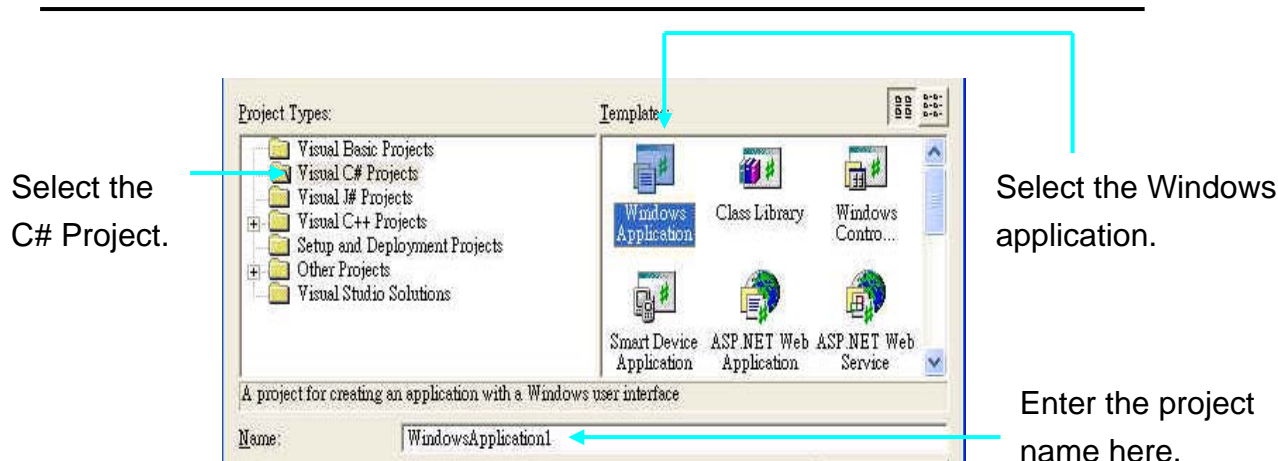
After adding the above two lines, the function can be called in an application in the following manner:

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



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:

```

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_GetDriverVersion")]
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;

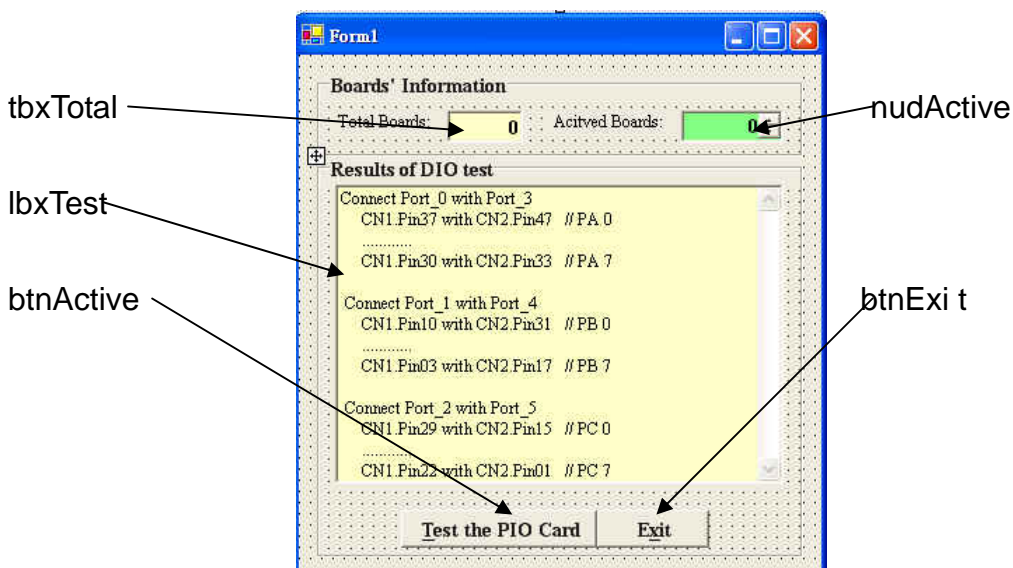
public PIODIO();//constroctor
{
    DriverOpened = 0;
}
~PIODIO()
{
    if (DriverOpened != 0)
    {
        DriverOpened = 0;
        DriverClose();
    }
}
}
}
}

```

Step 4.

Design the application and use the DLL functions.

Designing the GUI:



Using the function:

```
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;
    }
}
}
}

```