

Phần 3:

Viết Chương Trình C# giao tiếp với thiết bị USB

1. Tạo Driver cho thiết bị và tạo các thư viện hàm giao tiếp USB:

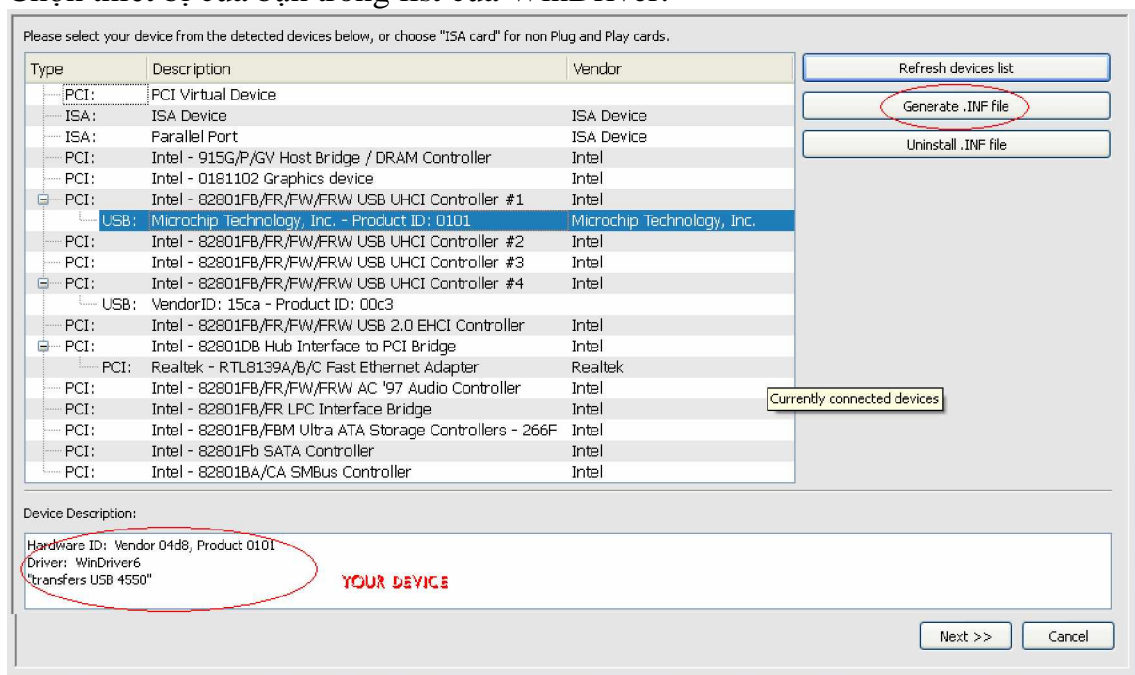
Công việc đầu tiên mà bạn phải làm với WinDriver để tạo Driver và các thư viện phần mềm cho chương trình PC của bạn. Các bước tiến hành bao gồm:

1. Gắn thiết bị của bạn vào cổng USB trên máy tính computer:
2. Chạy DriverWizard và chọn thiết bị của bạn:



hình 1: Mở hoặc tạo một WinDriver Project

Chọn thiết bị của bạn trong list của WinDriver:



Chọn phát ra file Driver và đánh tên cho thiết bị của bạn:

Vendor ID: 04d8 Device ID: 0101

Manufacturer name: Microchip Technology, Inc.

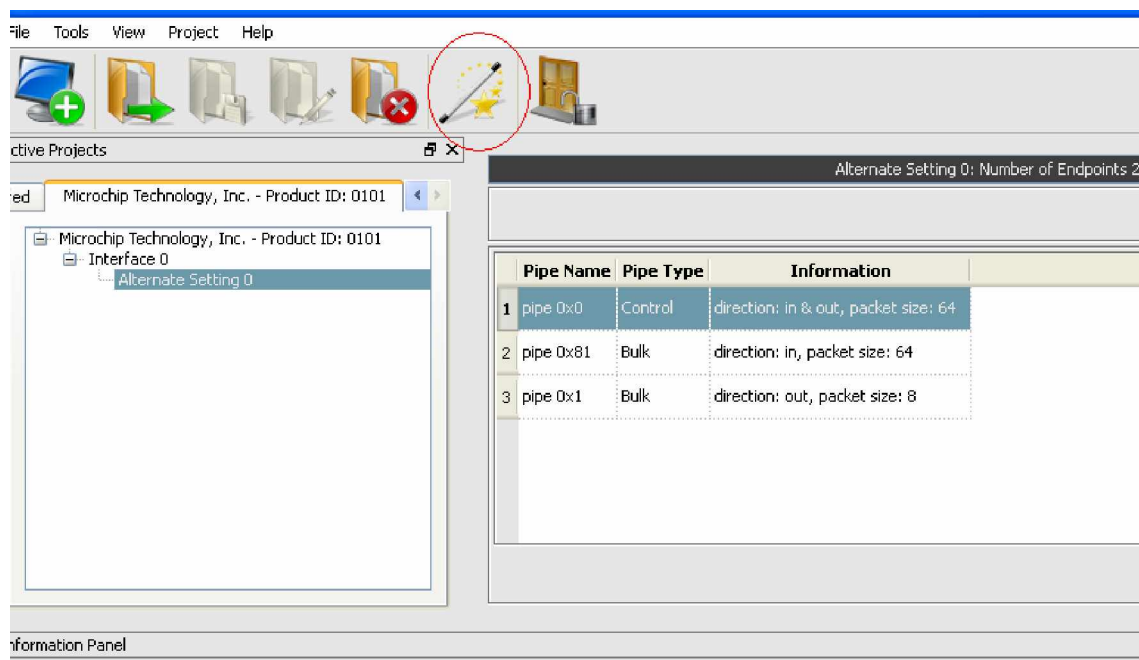
Device name: **DEVICE**

Device Class: OTHER

WinDriver's unique Class.

Use this option for a non-standard type of device.
WinDriver will set a new Class type for your device.

Tiếp tục chọn *next* ta được:



Tại đây bạn có thể kiểm tra việc truyền nhận qua USB xem thiết bị của bạn đã hoạt động chưa. Việc tiếp theo là tạo ra giao diện ban đầu và thư viện trên C#. Click vào biểu tượng *Generate Code* và chọn ngôn ngữ mà bạn sử dụng, hiện nay ngôn ngữ C# thông dụng với nhiều người và có nhiều tiện ích hơn so với VB nên ở đây tôi chọn như sau:

Select Code Generation Options

In which language do you want your code to be generated?

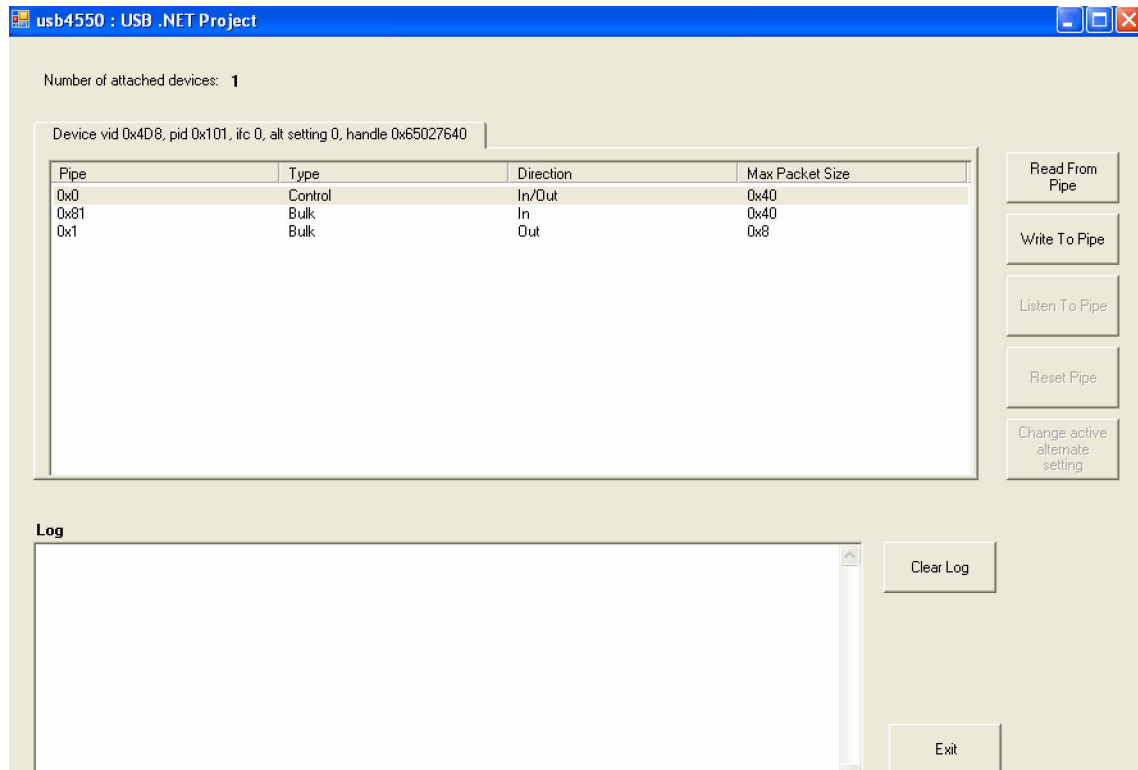
C# .NET

Generate project makefile for:

- ☐ MS Developer Studio .NET 2003
- ☒ MS Developer Studio .NET 2005 (for X86)
- ☐ MS Developer Studio .NET 2005 (for AMD64)

Công việc còn lại chỉ là OK và chờ máy chạy hihi.

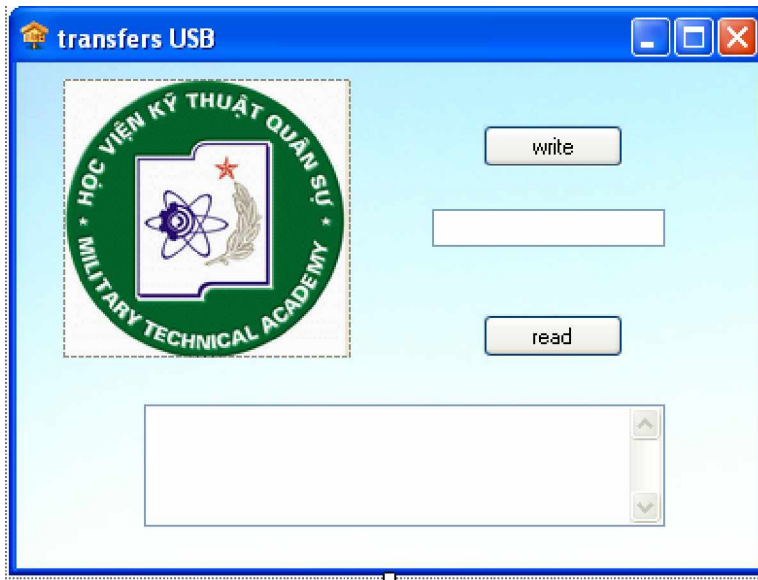
Chạy Project trên Visual Studio 2005 bạn có thể thấy WinDriver đã tạo cho ta một giao diện ban đầu đủ để sử dụng các giao tiếp USB, bạn có thể sử dụng luôn giao diện này để thực hiện theo Project của bạn:



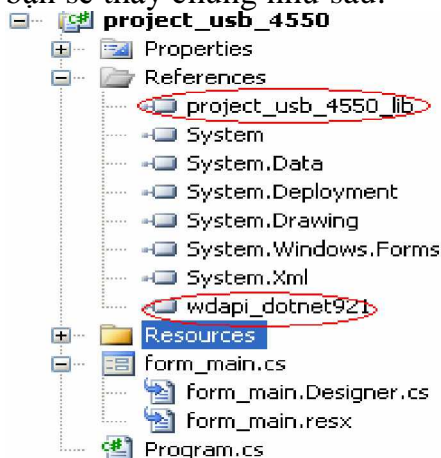
Chú ý rằng bạn phải *Project/ Set as Startup Project* trước đã.

Nhưng để giúp cho người mới học có thể hiểu được một cách nhanh chóng tôi xin tự tạo một giao diện khác đơn giản ngắn gọn hơn bằng việc sử dụng các thư viện mà WinDriver vừa tạo ra cho chúng ta. Đó là hai thư viện *project_usb_4550_lib.dll* (thư viện này do ta đặt nó cùng với tên Project khi tạo ra) và *wdapi_dotnet921.dll* hai thư viện này chứa các lớp và các ủy quyền rất hữu ích giúp cho việc quản lý và thực hiện giao tiếp bằng USB.

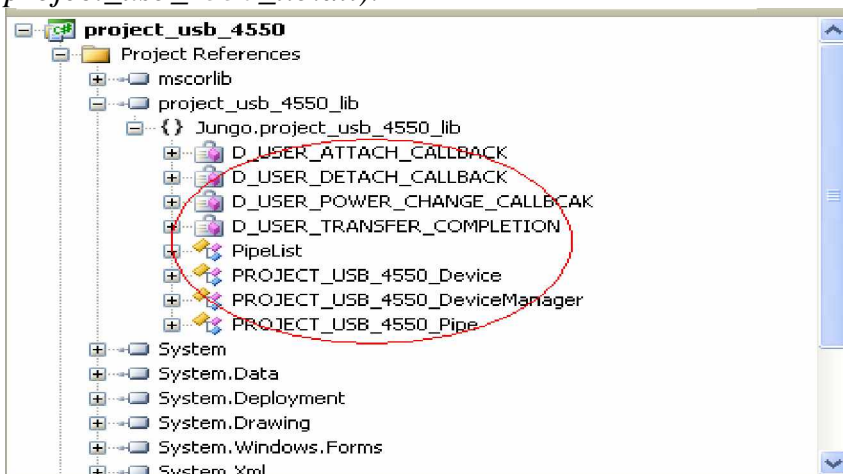
Công việc tiếp theo của chúng ta là hãy tạm quên đi cái giao diện mà WinDriver vừa tạo ra ở trên, và bắt tay vào tạo một giao diện của chính chúng ta. Với phương châm càng đơn giản càng tốt nên hướng dẫn này tôi chỉ tạo một giao diện chỉ với 2 Button và 2 textBox dùng để điều khiển như sau:



Vậy là song bước đầu tiên, bước tiếp theo là add 2 thư viện ở trên vào trong project của mình và nghiên cứu cách sử dụng chúng. Nếu bạn add thành công thì bạn sẽ thấy chúng như sau:



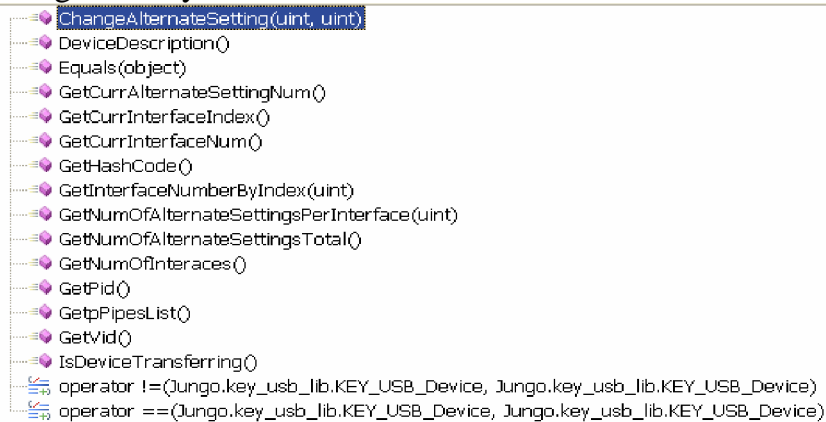
Ta hãy xem chúng có những gì (việc sử dụng chủ yếu trên thư viện *project_usb_4550_lib.dll*):



Thư viện này chứa 4 uỷ quyền (delegates) dành để tạo các sự kiện quan trọng với bus USB và 4 lớp, trong đó 3 lớp để điều khiển thiết bị USB và một lớp để

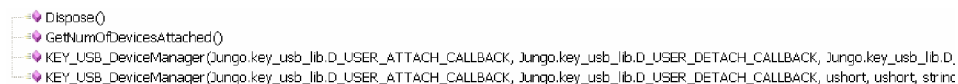
quản lý danh sách các thiết bị USB khác nhau được gắn vào PC. Chúng ta hãy quan sát cụ thể hơn với các uỷ quyền và các lớp này:

- Uỷ quyền **D_USER_ATTACH_CALLBACK** sử dụng để gắn sự kiện có thiết bị USB thích hợp gắn vào bus USB với một hàm callback do bạn tạo ra thường là khởi tạo việc truyền nhận dữ liệu.
- Uỷ quyền **D_USER_DETACH_CALLBACK** Được sử dụng để gắn sự kiện thiết bị USB đã được tháo ra với một hàm gọi lại do bạn xây dựng thường là thông báo cho người sử dụng về việc Disconnect và kết thúc việc truyền nhận
- Uỷ quyền **D_USER_TRANSFER_COMPLETION** khá hữu ích vì nó sẽ gắn sự kiện truyền nhận kết thúc vào một hàm xử lý đọc viết dữ liệu usb do bạn xây dựng.
- Uỷ quyền **D_USER_POWER_CHANGE_CALLBACK** ít được sử dụng hơn vì nó liên quan đến việc báo động thay đổi nguồn trên bus USB, bạn có thể tận dụng để thực hiện các biện pháp bảo đảm cắt nguồn bus USB khi có sự cố.
- Lớp **KEY_USB_Device** chứa các hàm để thực hiện việc cài đặt cấu hình khác nhau cho thiết bị USB hay hàm đọc các số hiệu VP, IP và hàm theo dõi trạng thái truyền nhận của thiết bị:



```
ChangeAlternateSetting(uint, uint)
DeviceDescription()
Equals(object)
GetCurrAlternateSettingNum()
GetCurrInterfaceIndex()
GetCurrInterfaceNum()
GetHashCode()
GetInterfaceNumberByIndex(uint)
GetNumOfAlternateSettingsPerInterface(uint)
GetNumOfAlternateSettingsTotal()
GetNumOfInterfaces()
GetPid()
GetPipesList()
GetVid()
IsDeviceTransferring()
operator !=(Jungo.key_usb_lib.KEY_USB_Device, Jungo.key_usb_lib.KEY_USB_Device)
operator ==(Jungo.key_usb_lib.KEY_USB_Device, Jungo.key_usb_lib.KEY_USB_Device)
```

- Lớp **KEY_USB_DeviceManager** thực hiện việc quản lý thiết bị USB và về số lượng thiết bị USB được gắn vào:



```
Dispose()
GetNumOfDevicesAttached()
KEY_USB_DeviceManager(Jungo.key_usb_lib.D_USER_ATTACH_CALLBACK, Jungo.key_usb_lib.D_USER_DETACH_CALLBACK, Jungo.key_usb_lib.D_USER_TRANSFER_COMPLETION_CALLBACK, ushort, ushort, string)
KEY_USB_DeviceManager(Jungo.key_usb_lib.D_USER_ATTACH_CALLBACK, Jungo.key_usb_lib.D_USER_DETACH_CALLBACK, ushort, ushort, string)
```

- Lớp **KEY_USB_Pipe** có số lượng hàm tương đối lớn và chuyên dụng để đọc viết dữ liệu kiểm tra trạng thái truyền nhận, kiểm soát việc khởi động và dừng các đường ống...
- Lớp **Pipe_list** thực hiện quản lý các đường ống vào ra của thiết bị. Tập các hàm này khá linh hoạt và đầy đủ cho mọi ứng dụng USB và bạn cũng không cần phải tìm hiểu kỹ tất cả các hàm đó vì chúng ta chỉ cần làm việc với một số hàm nhất định là đủ để giao tiếp.

Bây giờ là đến công việc khó nhất đây, ta phải viết code làm sao cho khi gắn thiết bị vào thì nó phải nhận được và thực hiện truyền phát dữ liệu.

Nguyễn Ngọc Hưng Lớp TKTL 40 HVKTQS

Ý tưởng của tôi là dùng Button1(tên là *btwrite*) để thực hiện việc truyền dữ liệu được lấy trong textBox1(tên là *txttrans*) từ PC xuống VĐK, Button2(*btread*) dùng để lấy dữ liệu từ VĐK lên PC hiển thị ở textBox2(*txtLog*).

Về thuật toán thực hiện chương trình chúng ta chỉ hạn chế 3 chức năng cơ bản là đọc, viết dữ liệu và phát hiện thiết bị gắn vào hay tháo ra (hai sự kiện nhận biết thiết bị đều được hiển thị trên *txtLog*).

Phần khai báo USING của chương trình chúng ta cần một số khai báo như sau:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Runtime.InteropServices;
using System.Runtime.CompilerServices;

using Jungo.wdapi_dotnet;           //sử dụng thư viện wdapi_dotnet
using Jungo.project_usb_4550_lib;    //sử dụng thư viện project_usb_4550_lib
using wdu_err=Jungo.wdapi_dotnet.WD_ERROR_CODES; //sử dụng việc khai báo lỗi

using DWORD = System.UInt32;
using WORD = System.UInt16;
using UCHAR = System.Byte;
```

phần khai báo các biến của form chính chúng ta thực hiện một số khai báo hàng biến và tạo các đối tượng sau:

```
private const string DEFAULT_LICENSE_STRING =
"6C3CC2CFE89E7AD04238DF2EF24449E848CDC187.NguyenNgocHung";
// TODO: If you have renamed the WinDriver kernel module (windrvr6.sys),
// change the driver name below accordingly
private const string DEFAULT_DRIVER_NAME = "windrvr6";
private const WORD DEFAULT_VENDOR_ID = 0x04D8;
private const WORD DEFAULT_PRODUCT_ID = 0x0101;
public const DWORD TIME_OUT = 30000;

private PROJECT_USB_4550_DeviceManager uDevManager; //khai báo khởi tạo
việc quản lý đối tượng USB
private PROJECT_USB_4550_Device usb_device; //khai báo đối tượng USB
private PROJECT_USB_4550_Pipe usb_pipe_trs; //khai báo đối tượng ống truyền
private PROJECT_USB_4550_Pipe usb_pipe_rev; //khai báo đối tượng ống nhận
private delegate void D_ATTACH_GUI_CALLBACK(PROJECT_USB_4550_Device pDev);
//khai báo ủy quyền thông báo thiết bị gắn vào
private delegate void D_DETACH_GUI_CALLBACK(PROJECT_USB_4550_Device pDev);
//khai báo ủy quyền thông báo thiết bị tháo ra
delegate void safeLogTextCallBack(string sMsg); //khai báo ủy quyền phục vụ
hiển thị
```

Chúng ta đã có 1 đối tượng quản lý USB, 1 đối tượng thiết bị USB, 2 đối tượng đường ống TRANSFER và RECEIVER, các kết nối, khai báo driver điều khiển và các ủy quyền cần thiết. Tiếp theo chúng ta sẽ xây dựng các hàm của form theo các sự kiện chính. Sự kiện đầu tiên chúng ta cần quan tâm là “form load” vì nó xảy ra đầu tiên trong chuỗi sự kiện vì vậy có thể tận dụng để khởi tạo một số biến, đối tượng cho chương trình.

```
private void form_main_Load(object sender, EventArgs e)
{
    // đăng ký hai hàm sự kiện ATTACH_CALLBACK & DETACH_CALLBACK
```


Nguyễn Ngọc Hưng Lớp TKTL 40 HVKTQS

```
D_USER_ATTACH_CALLBACK dDeviceAttachCb = new
    D_USER_ATTACH_CALLBACK(UserDeviceAttach);
D_USER_DETACH_CALLBACK dDeviceDetachCb = new
    D_USER_DETACH_CALLBACK(UserDeviceDetach);

//khởi tạo đối tượng quản lý thiết bị USB
uDevManager = new
PROJECT_USB_4550_DeviceManager(dDeviceAttachCb, dDeviceDetachCb,
    DEFAULT_VENDOR_ID, DEFAULT_PRODUCT_ID, DEFAULT_DRIVER_NAME,
    DEFAULT_LICENSE_STRING);
}
```

Hàm 2 callback xử lý 2 sự kiện Attach và Detach theo cơ cấu trung gian được gọi đến 2 hàm UserDeviceAttach & UserDeviceDetach để khởi tạo các đối tượng **AttachCb** và **DetachCb**. Trong hàm này còn thực hiện việc khởi tạo thiết bị USB và hai đường ống TRANSFER & RECEIVER.

```
private void UserDeviceAttach(PROJECT_USB_4550_Device pDev)
{
    D_ATTACH_GUI_CALLBACK AttachCb = new
        D_ATTACH_GUI_CALLBACK(DeviceAttachGuiCb);

    // khởi tạo thiết bị USB
    usb_device = pDev;
    // khởi tạo đường ống nhận và truyền
    usb_pipe_rev =
    (PROJECT_USB_4550_Pipe)usb_device.GetpPipesList()[1];
    usb_pipe_trs =
    (PROJECT_USB_4550_Pipe)usb_device.GetpPipesList()[2];

    Invoke(AttachCb, new object[] {pDev});
}
```

Và:

```
private void UserDeviceDetach(PROJECT_USB_4550_Device pDev)
{
    D_DETACH_GUI_CALLBACK DetachCb = new
        D_DETACH_GUI_CALLBACK(DeviceDetachGuiCb);
    Invoke(DetachCb, new object[] { pDev });
}
```

Các hàm DeviceAttachGuiCb & DeviceDetachGuiCb chỉ là các hàm hiển thị trạng thái thiết bị gắn vào hay tháo ra.

```
private void DeviceAttachGuiCb(PROJECT_USB_4550_Device pDev)
{
    TraceMsg("Found new device\r\n");
    TraceMsg("Your new device is installed and ready to use\r\n");
}
private void DeviceDetachGuiCb(PROJECT_USB_4550_Device pDev)
{
    TraceMsg("Your device has detached\r\n");
}
```

Để thuận lợi cho việc hiển thị trạng thái chương trình chúng ta nên bổ xung một số hàm điều khiển việc xuất thông báo ví dụ 3 hàm dưới đây:

```
private void SafeLogText(string sMsg)
{
    // InvokeRequired compares the thread ID of the
    // calling thread to the thread ID of the creating thread.
    // If these threads are different, it returns true.
    if (txtLog.InvokeRequired)
    {

```

```
        safeLogTextCallBack cb = new
safeLogTextCallBack(SafeLogText);
        this.Invoke(cb, new object[] { sMsg });
    }
    else
    {
        txtLog.AppendText(sMsg);
    }
}
public void TraceMsg(string sMsg)
{
    SafeLogText(sMsg);
}
```

Bây giờ ta bắt tay vào việc xây dựng hàm dùng để truyền nhận dữ liệu. Khi nhấn vào button *write* thì thực hiện truyền một byte xuống VDK:

```
private void btwrite_Click(object sender, EventArgs e)
{
    SingleTransfer();
}
```

Hàm `SingleTransfer()`; để thực hiện viết sẽ lấy 1 byte đã nhập vào *txttrans* và chuyển sang dạng byte để ở trong *buffer* sau đó gọi hàm truyền không đồng bộ:

```
usb_pipe_trs.UsbPipeTransferAsync(false, 0, buffer,
    dwBuffSize, TIME_OUT, new
D_USER_TRANSFER_COMPLETION(Transfer_trs_Completion));
```

Khi nhấn vào button *read* thì thực hiện nhận một byte từ VDK lên PC và hiển thị ra *txtLog*.

```
private void btread_Click(object sender, EventArgs e)
{
    SingleReceiver();
}
```

Hàm `SingleReceiver()`; tương tự như hàm `SingleTransfer()`; để thực hiện việc nhận dữ liệu vào bộ đệm:

```
usb_pipe_rev.UsbPipeTransferAsync(true, 0, buffer,
    dwBuffSize, TIME_OUT, new
D_USER_TRANSFER_COMPLETION(Transfer_rev_Completion));
```

Các hàm `Transfer_trs_Completion` & `Transfer_rev_Completion` thực hiện việc thông báo hoàn thành truyền hoặc nhận. Hàm `DisplayHexBuffer` để hỗ trợ việc hiển thị giá trị thập lục phân.

Sau đây là toàn bộ mã của chương trình:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Runtime.InteropServices;
using System.Runtime.CompilerServices;

using Jungo.wdapi_dotnet;
using Jungo.project_usb_4550_lib;
using wdu_err = Jungo.wdapi_dotnet.WD_ERROR_CODES;

using DWORD = System.UInt32;
```


Nguyễn Ngọc Hưng Lớp TKTL 40 HVKTQS

```
using WORD = System.UInt16;
using UCHAR = System.Byte;

namespace project_usb_4550
{
    public partial class form_main : Form
    {
        private const string DEFAULT_LICENSE_STRING =
"6C3CC2CFE89E7AD04238DF2EF24449E848CDC187.NguyenNgocHung";
        private const string DEFAULT_DRIVER_NAME = "windrvr6";
        private const WORD DEFAULT_VENDOR_ID = 0x04D8;
        private const WORD DEFAULT_PRODUCT_ID = 0x0101;
        public const DWORD TIME_OUT = 30000;

        private PROJECT_USB_4550_DeviceManager uDevManager;
        private PROJECT_USB_4550_Device usb_device;
        private PROJECT_USB_4550_Pipe usb_pipe_trs;
        private PROJECT_USB_4550_Pipe usb_pipe_rev;
        private delegate void D_ATTACH_GUI_CALLBACK(PROJECT_USB_4550_Device
pDev);
        private delegate void D_DETACH_GUI_CALLBACK(PROJECT_USB_4550_Device
pDev);
        delegate void safeLogTextCallBack(string sMsg);
        public form_main()
        {
            InitializeComponent();
        }
        private void form_main_Load(object sender, EventArgs e)
        {
            D_USER_ATTACH_CALLBACK dDeviceAttachCb = new
                D_USER_ATTACH_CALLBACK(UserDeviceAttach);
            D_USER_DETACH_CALLBACK dDeviceDetachCb = new
                D_USER_DETACH_CALLBACK(UserDeviceDetach);

            uDevManager = new
PROJECT_USB_4550_DeviceManager(dDeviceAttachCb, dDeviceDetachCb,
                DEFAULT_VENDOR_ID, DEFAULT_PRODUCT_ID, DEFAULT_DRIVER_NAME,
                DEFAULT_LICENSE_STRING);
        }

        private void UserDeviceAttach(PROJECT_USB_4550_Device pDev)
        {
            D_ATTACH_GUI_CALLBACK AttachCb = new
                D_ATTACH_GUI_CALLBACK(DeviceAttachGuiCb);

            usb_device = pDev;
            usb_pipe_rev =
(PROJECT_USB_4550_Pipe)usb_device.GetpPipesList()[1];
            usb_pipe_trs =
(PROJECT_USB_4550_Pipe)usb_device.GetpPipesList()[2];

            Invoke(AttachCb, new object[] {pDev});
        }
        private void DeviceAttachGuiCb(PROJECT_USB_4550_Device pDev)
        {
            TraceMsg("Found new device\r\n");
            TraceMsg("Your new device is installed and ready to use\r\n");
        }
        private void UserDeviceDetach(PROJECT_USB_4550_Device pDev)
        {
            D_DETACH_GUI_CALLBACK DetachCb = new
                D_DETACH_GUI_CALLBACK(DeviceDetachGuiCb);
            Invoke(DetachCb, new object[] { pDev });
        }
    }
}
```

Nguyễn Ngọc Hưng Lớp TKTL 40 HVKTQS

```
private void DeviceDetachGuiCb(PROJECT_USB_4550_Device pDev)
{
    TraceMsg("Your device has detached\r\n");
}

private void SafeLogText(string sMsg)
{
    // InvokeRequired compares the thread ID of the
    // calling thread to the thread ID of the creating thread.
    // If these threads are different, it returns true.
    if (txtLog.InvokeRequired)
    {
        safeLogTextCallBack cb = new
safeLogTextCallBack(SafeLogText);
        this.Invoke(cb, new object[] { sMsg });
    }
    else
    {
        txtLog.AppendText(sMsg);
    }
}

public void TraceMsg(string sMsg)
{
    SafeLogText(sMsg);
}

private void btwrite_Click(object sender, EventArgs e)
{
    SingleTransfer();
}

private void SingleTransfer()
{
    DWORD dwBuffSize = 1;
    byte[] buffer = new byte[1];
    string txt = txttrans.Text;
    buffer[0] = Convert.ToByte(txt[0]);
    usb_pipe_trs.UsbPipeTransferAsync(false, 0, buffer,
dwBuffSize, TIME_OUT, new
D_USER_TRANSFER_COMPLETION(Transfer_trs_Completion));
}

private void Transfer_trs_Completion(PROJECT_USB_4550_Pipe pipe)
{
    if (pipe.GetTransferStatus() ==
(DWORD)wdu_err.WD_STATUS_SUCCESS)
    {
        TraceMsg("Transfer completed successfully!\r\n ");
    }
    else
    {
        TraceMsg("tranfer fail\r\n");
    }
}

private void btread_Click(object sender, EventArgs e)
{
    SingleReceiver();
}

private void SingleReceiver()
{
    DWORD dwBuffSize = 1;
    byte[] buffer = new byte[1];

    usb_pipe_rev.UsbPipeTransferAsync(true, 0, buffer,
```

Nguyễn Ngọc Hưng Lớp TKTL 40 HVKTQS

```
        dwBuffSize, TIME_OUT, new
D_USER_TRANSFER_COMPLETION(Transfer_rev_Completion));
    }

    private void Transfer_rev_Completion(PROJECT_USB_4550_Pipe pipe)
    {
        if (pipe.GetTransferStatus() ==
(DWORD)wdu_err.WD_STATUS_SUCCESS)
        {
            TraceMsg(DisplayHexBuffer(pipe.GetBuffer(),1));
        }
        else
        {
            TraceMsg("receiver failure!\r\n");
        }
    }

    private string DisplayHexBuffer(byte[] buff, DWORD dwBuffSize)
    {
        string display = "";
        for (DWORD i = 0; i < dwBuffSize; i++)
            display = string.Concat(display, buff[i].ToString("X"), " ");

        display = string.Concat(display, Environment.NewLine);
        return display;
    }
}
```