MICROCHIP

Regional Training Centers

COM3201T v2.0

Introduction HID Custom Class & Windows Application Programming

Authors : Calvin Ho
Modify : Adam Syu

MICROCHIP
Regional Training Centers

Objectives

- **Understand the basics of USB and HID class.**
- **Understand the advantages and disadvantages of using the**
- **HID class.**
- **Learn what USB examples that Microchip has provided**
- **To know what's the key point when using Microchip USB stack.**
- **Gain experience about how we can …**
  - Find the proper firmware example as reference.
  - Modify the hardware profile to accommodate your hardware.
  - Run the example and discover it's result.
  - Handle the running example by proper PC GUI.

MICROCHIP
Regional Training Centers

Agenda

- Part 1
  - USB Concepts & Specification Quick Review
- Part 2
  - Introduction USB HID Class
- Part3
  - Exercise
    - Lab1 Case Study
    - Lab2 Command & Response Style
    - Lab3 Auto Report Style

MICROCHIP

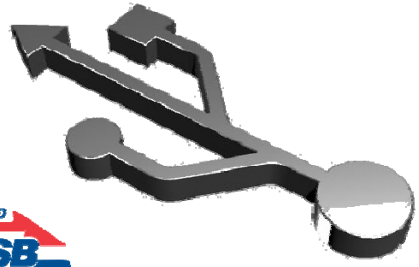
Regional Training Centers

USB Concepts and Specification Quick Review

MICROCHIP
Regional Training Centers

A Little History about USB

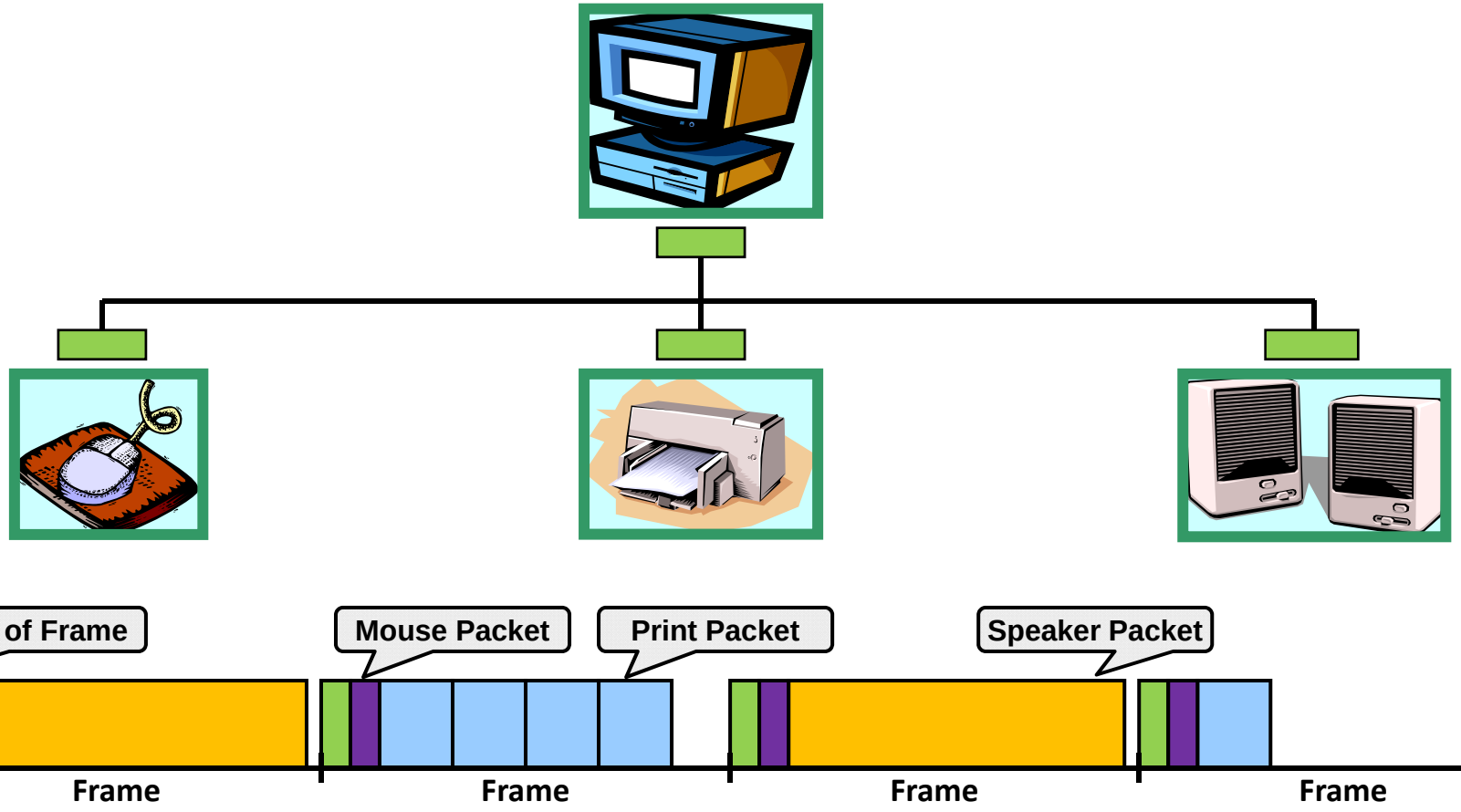
- USB was co-developed by a group of companies….
  - Compaq, Intel, Microsoft, NEC
- Who wanted to make it much easier to add/remove peripheral devices from PCs
- Jan.,1996 - USB 1.0 ***1.5Mbps,Low Speed***
- Sep.,1998 - USB 1.1 ***12Mbps,Full Speed***
- Apr.,2000 - USB 2.0 ***480Mbps,Hi Speed***
- 2001 - USB OTG (USB On The Go)
- Nov. 2008 - USB 3.0 ***5Gbps,SuperSpeed***
- Jul. 2013 - USB 3.1 ***10Gbps,SuperSpeed+***
  - ***Include Power Delivery feature (Max. Power : 100W)***



MICROCHIP
Regional Training Centers

Basics

- USB is a “Single Master + Multiple Slaves” polled bus.



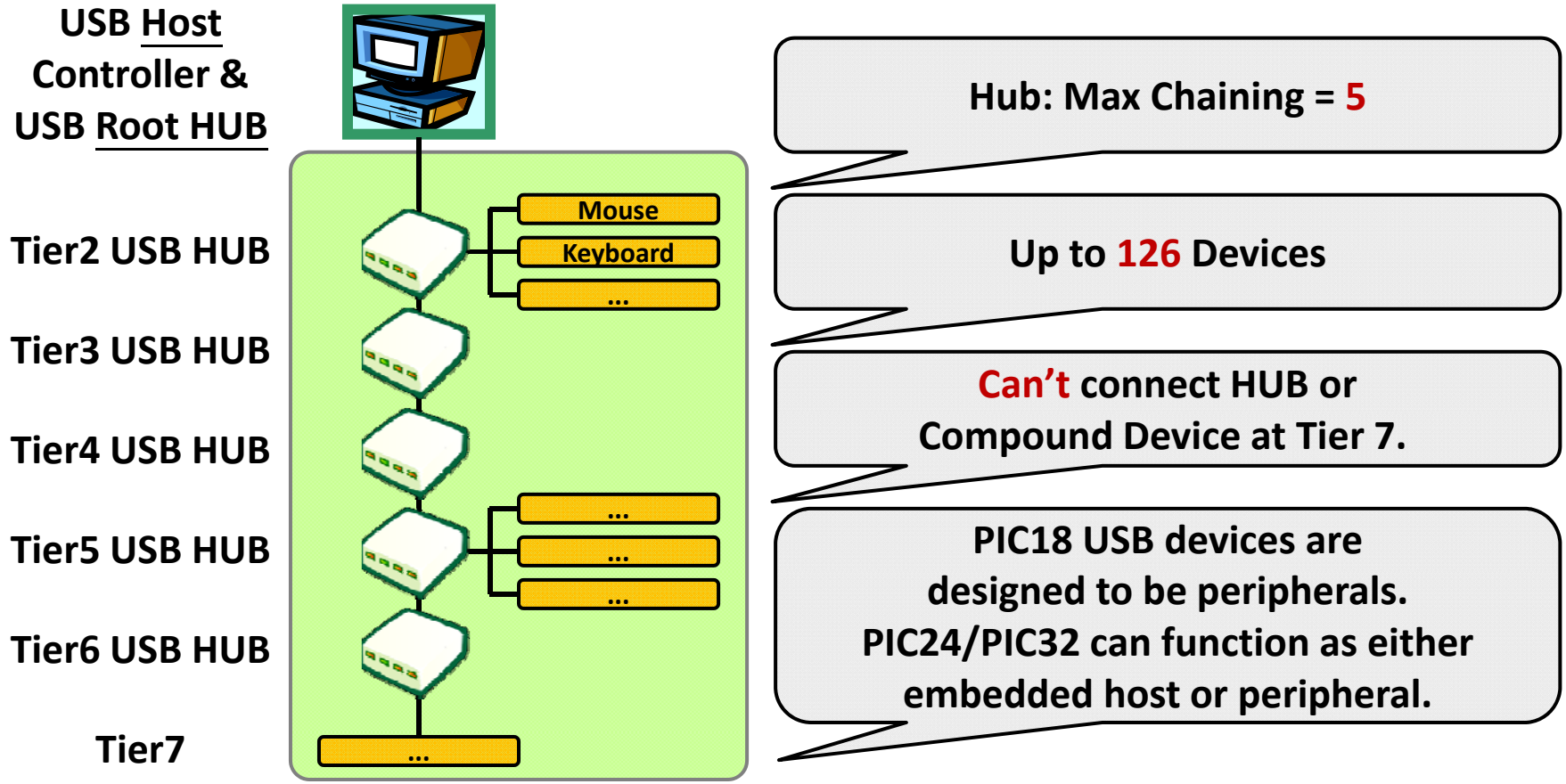
MICROCHIP
Regional Training Centers

USB Device Types

- **Peripheral (also called "Function")**
  - Provides a functionality (capability) to the host
  - i.e. data acquisition
- **Hub**
  - Repeats traffic (both directions), manages power
- **Compound Device**
  - Contains a hub and 1 or more peripheral
  - Host treats hub and peripheral function separately (each has its own address)
  - i.e. USB keyboard with 1-port hub
- **Composite Device**
  - Has multiple interfaces active at the same time
  - Host loads a driver for each interface
  - i.e. video camera (both audio & video interfaces active)

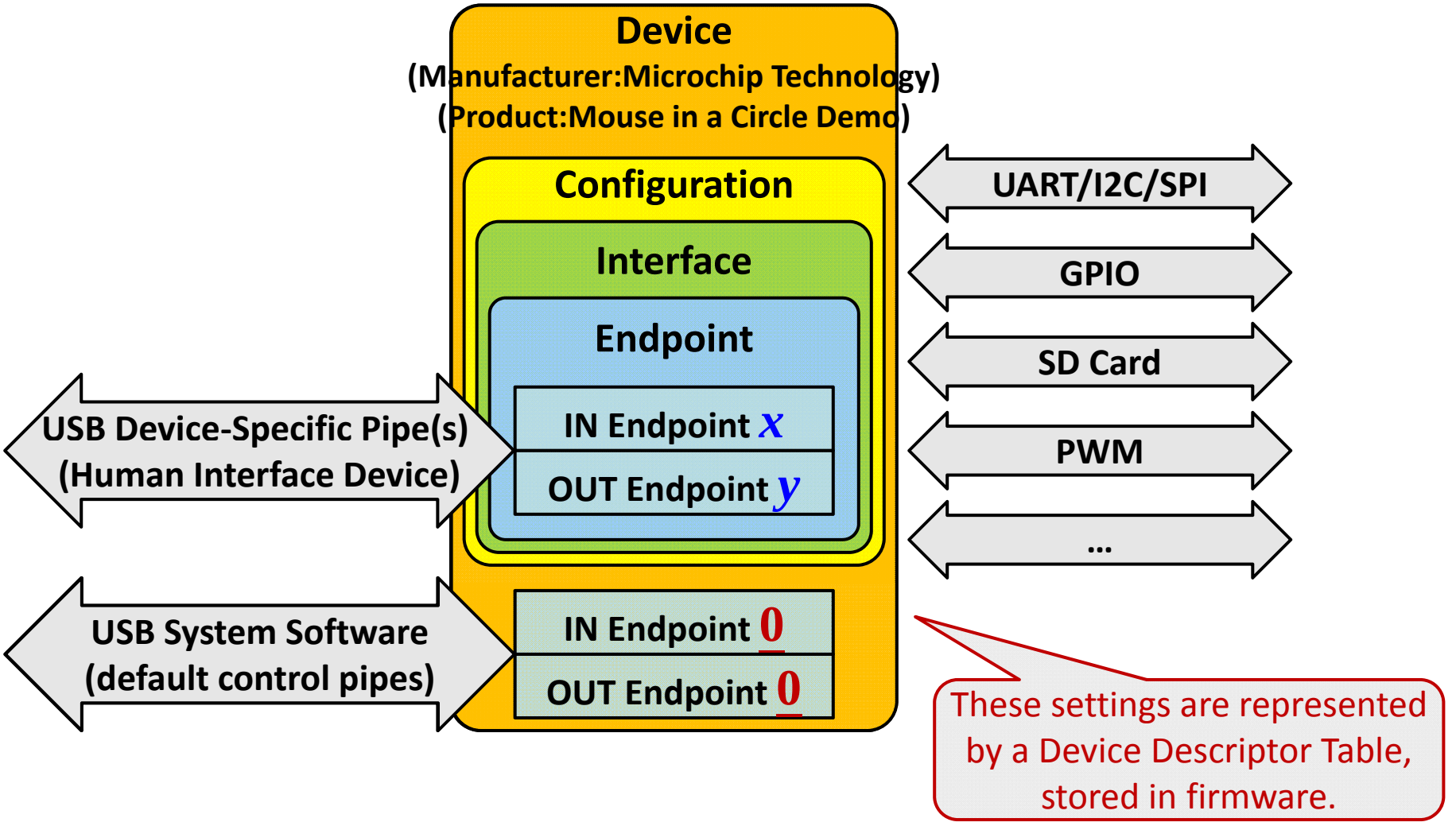
MICROCHIP
Regional Training Centers

Physical Bus Topology



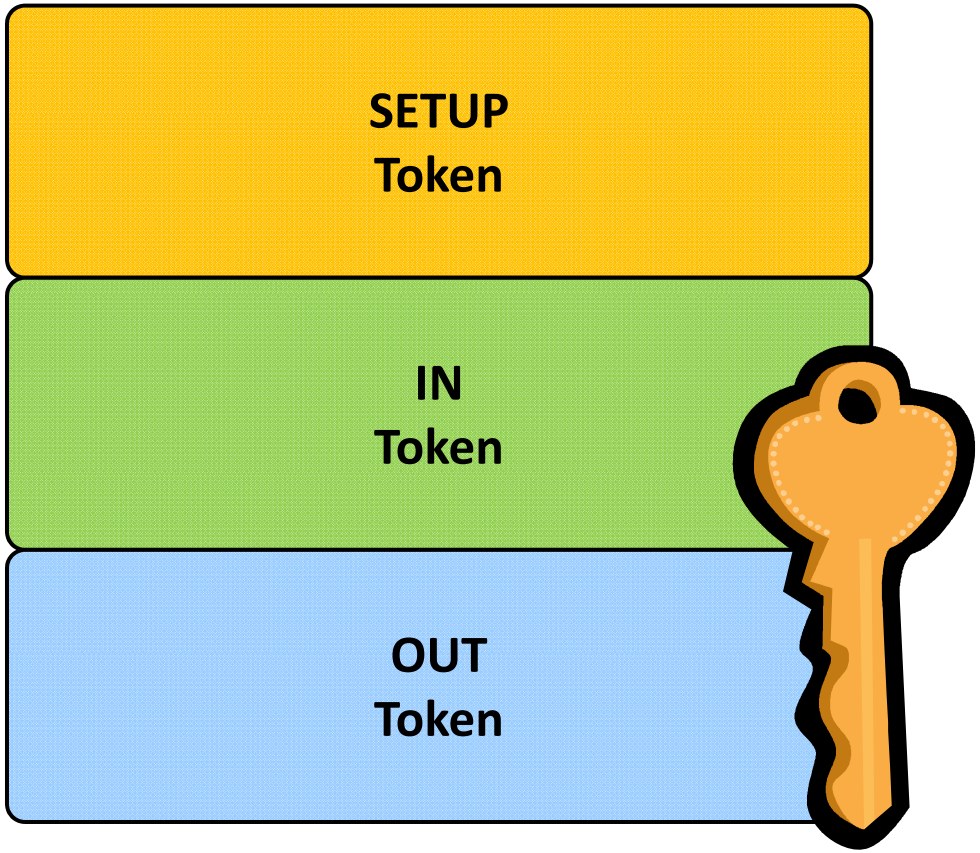
MICROCHIP
Regional Training Centers

The "Logical" Device



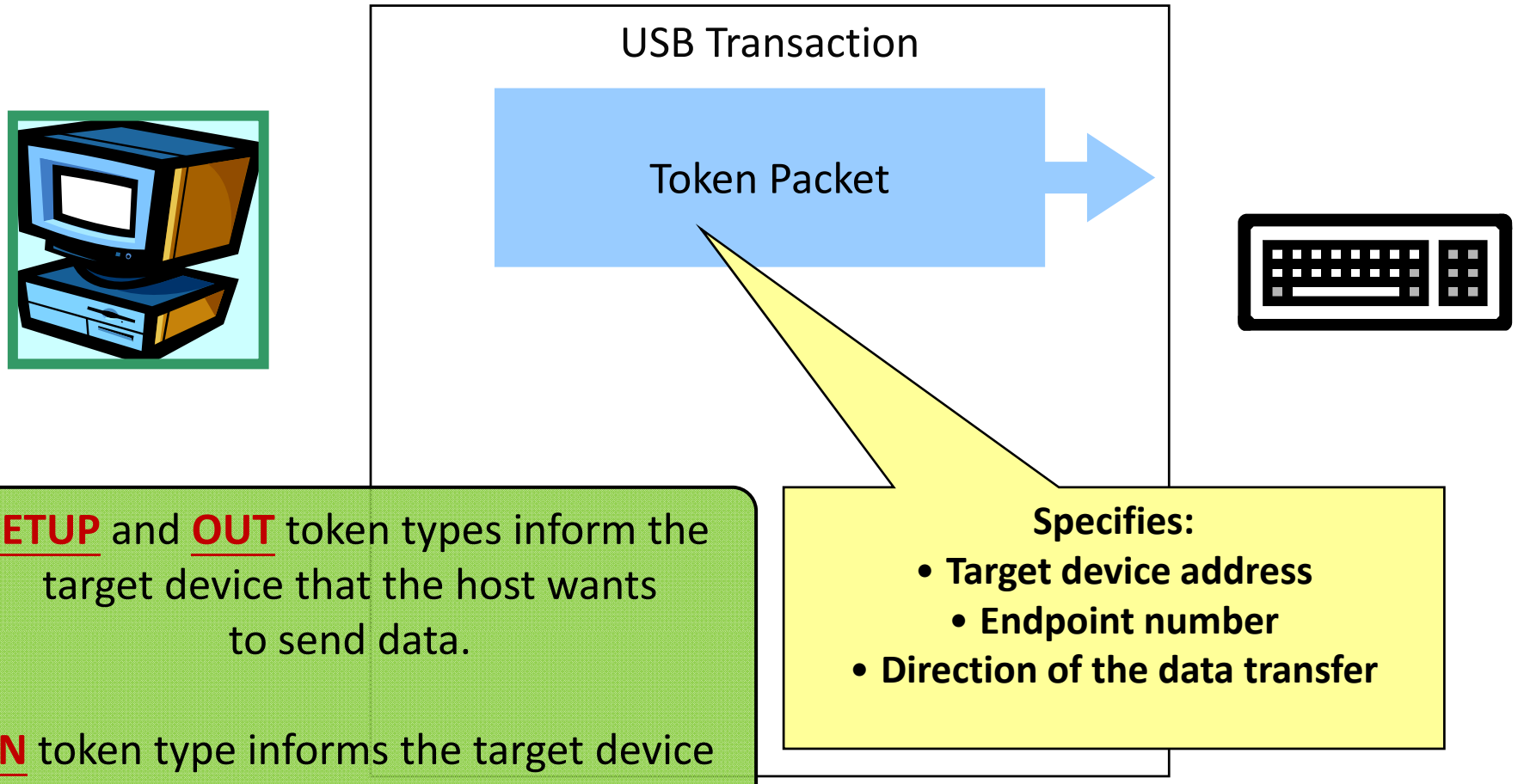
MICROCHIP
Regional Training Centers

Key: Token Types



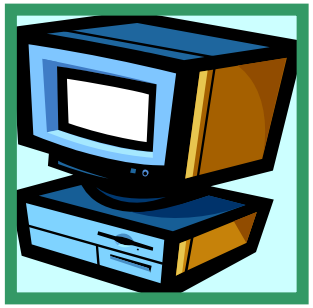
MICROCHIP
Regional Training Centers

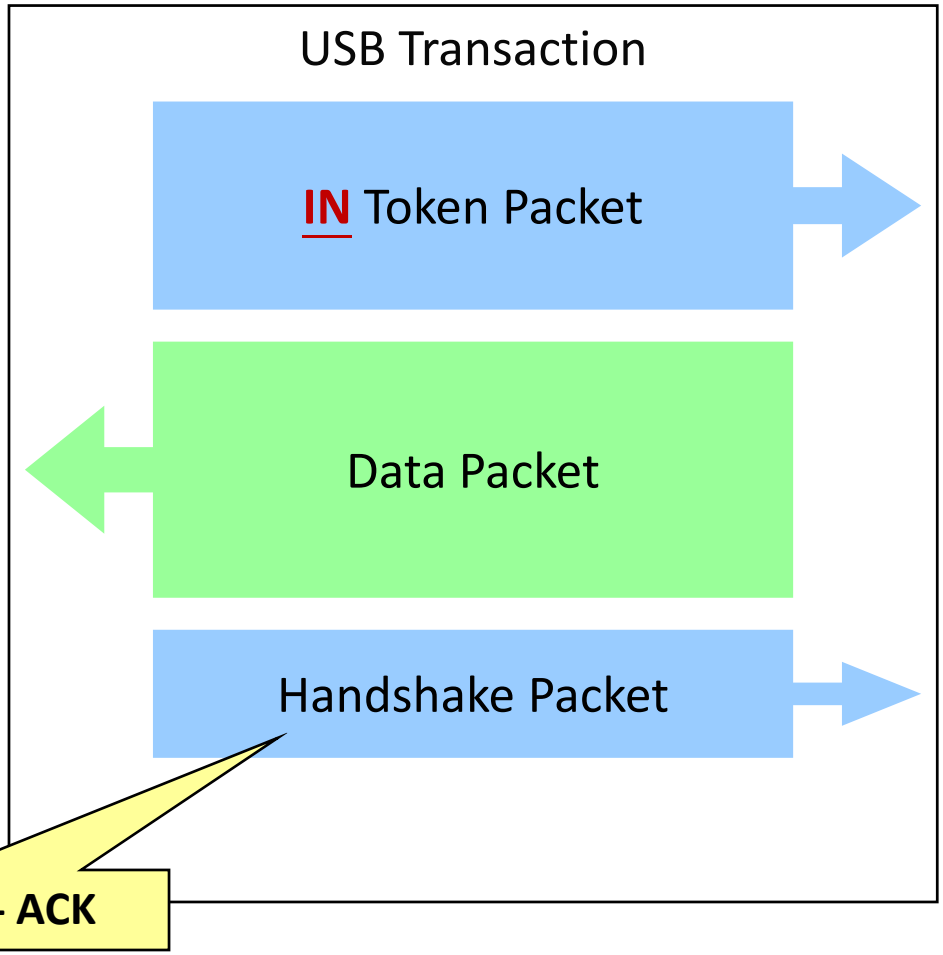
USB Transaction

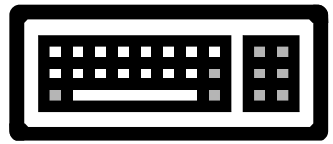


MICROCHIP
Regional Training Centers

USB Transaction Example

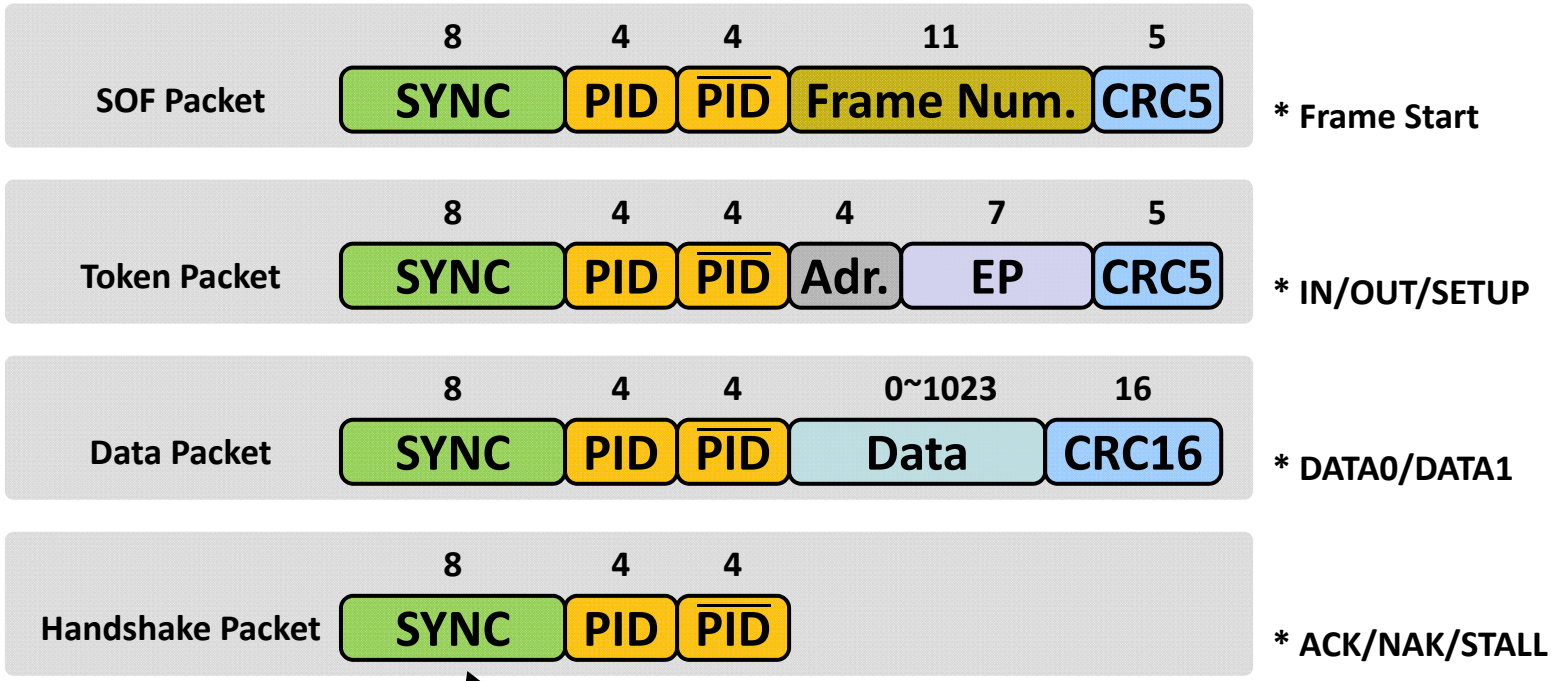




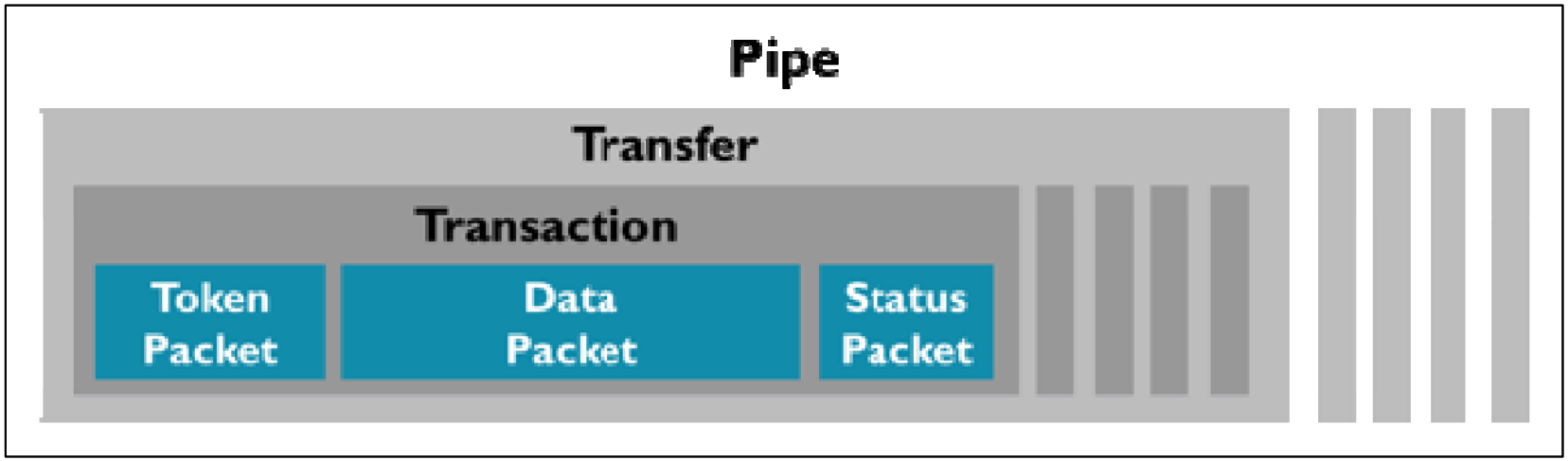


MICROCHIP
Regional Training Centers

Format of USB Packets



Packet, Frame, Transaction, Transfer & Pipe



* Refer from http://www.keil.com/pack/doc/mw/USB/html/_u_s_b__protocol.html

MICROCHIP
Regional Training Centers

Transfers Types

- **Bulk** (MSD, CDC, etc..)
  - Transfers data when there is time
  - **Highest throughput, but unbounded latency**
- **Interrupt** (HID Class)
  - Transfers a small amount of data with **guaranteed latency**
  - Low throughput, low latency.
- **Control** (All Class Need)
  - Performs setup and control of the device (Enumeration).
  - Message-based, initiated from the host.
  - Can also transfer data, but requires protocol overhead.
- **Isochronous** (Audio, Video Class)
  - **Transfers data quickly but not guaranteed**
  - Low latency, high throughput.

MICROCHIP
Regional Training Centers

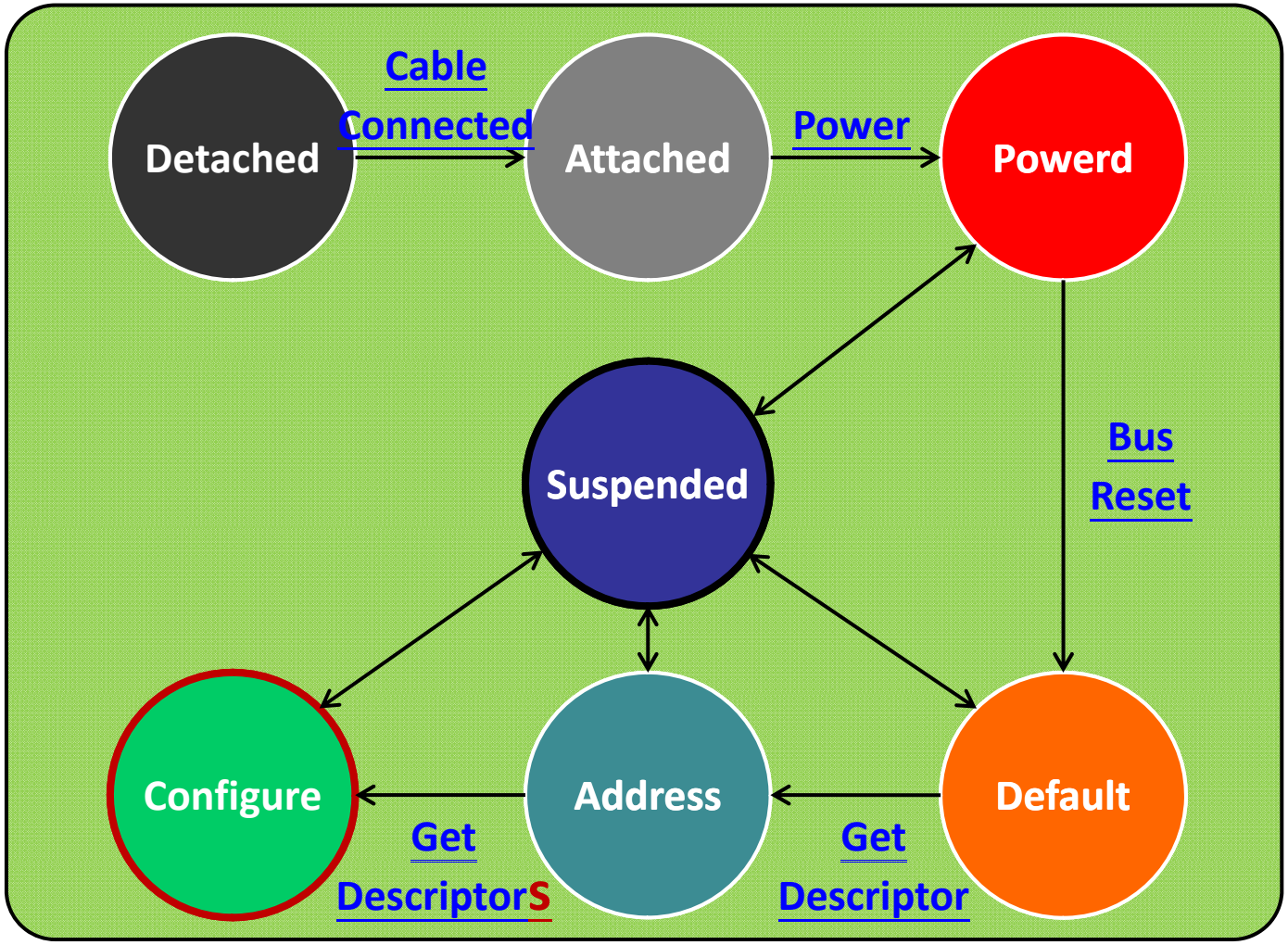
Summary - Data Transfer Types

Transfer/ Endpoint Type	Polling Interval	% Reserved BW/Frame for all transfers of this type	Max. # Data Bytes/Frame/Endpoint (Max# transactions per frame @ Max Ep Size)*	Data Integrity
Interrupt	Fixed, Periodic	10	64 (1 x 64)	Yes
Isochronous	Fixed, Periodic	90	1,023 (1 x 1023)	No
Bulk	Variable, Uses Free Bandwidth	0	1,216 (19 x 64)	Yes
Control	Variable	10	832 (13 x 64)	Yes

***Assumes transfers use maximum packet sizes allowed per EP type**

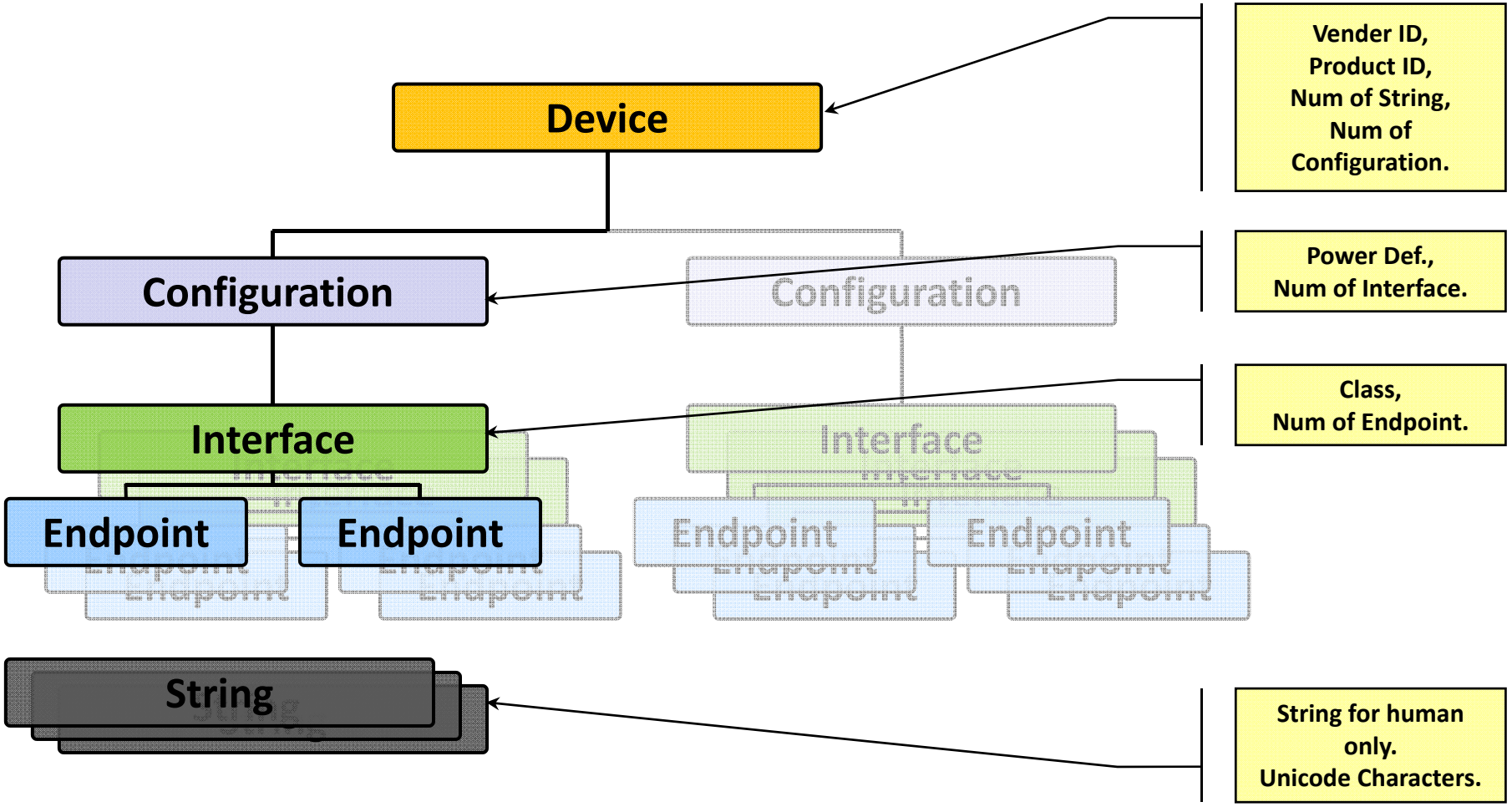
MICROCHIP
Regional Training Centers

The Enumeration Process



MICROCHIP
Regional Training Centers

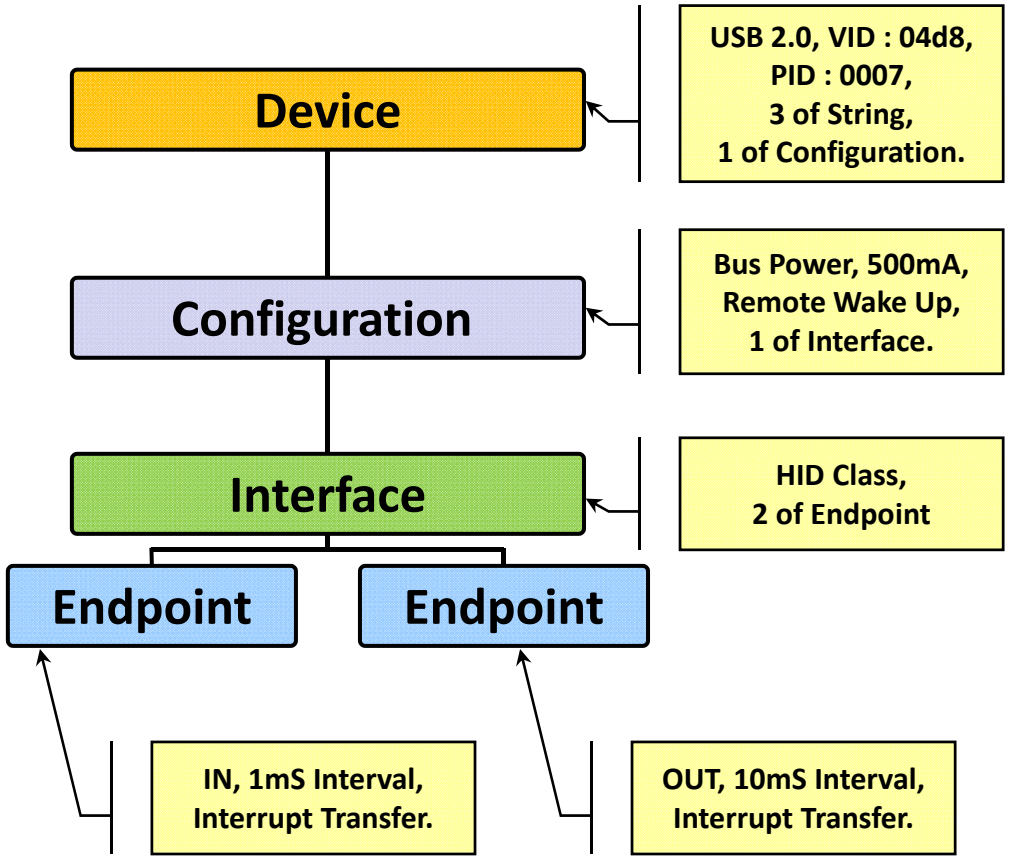
Descriptors



*** Descriptors are typically stored in non-volatile/Flash memory**

MICROCHIP
Regional Training Centers

Descriptors Example



String1
Microchip Technology Inc.

String2
HID Basic Demo

String3
Rev.A00

MICROCHIP
Regional Training Centers

MCHPFSUSB Software Framework

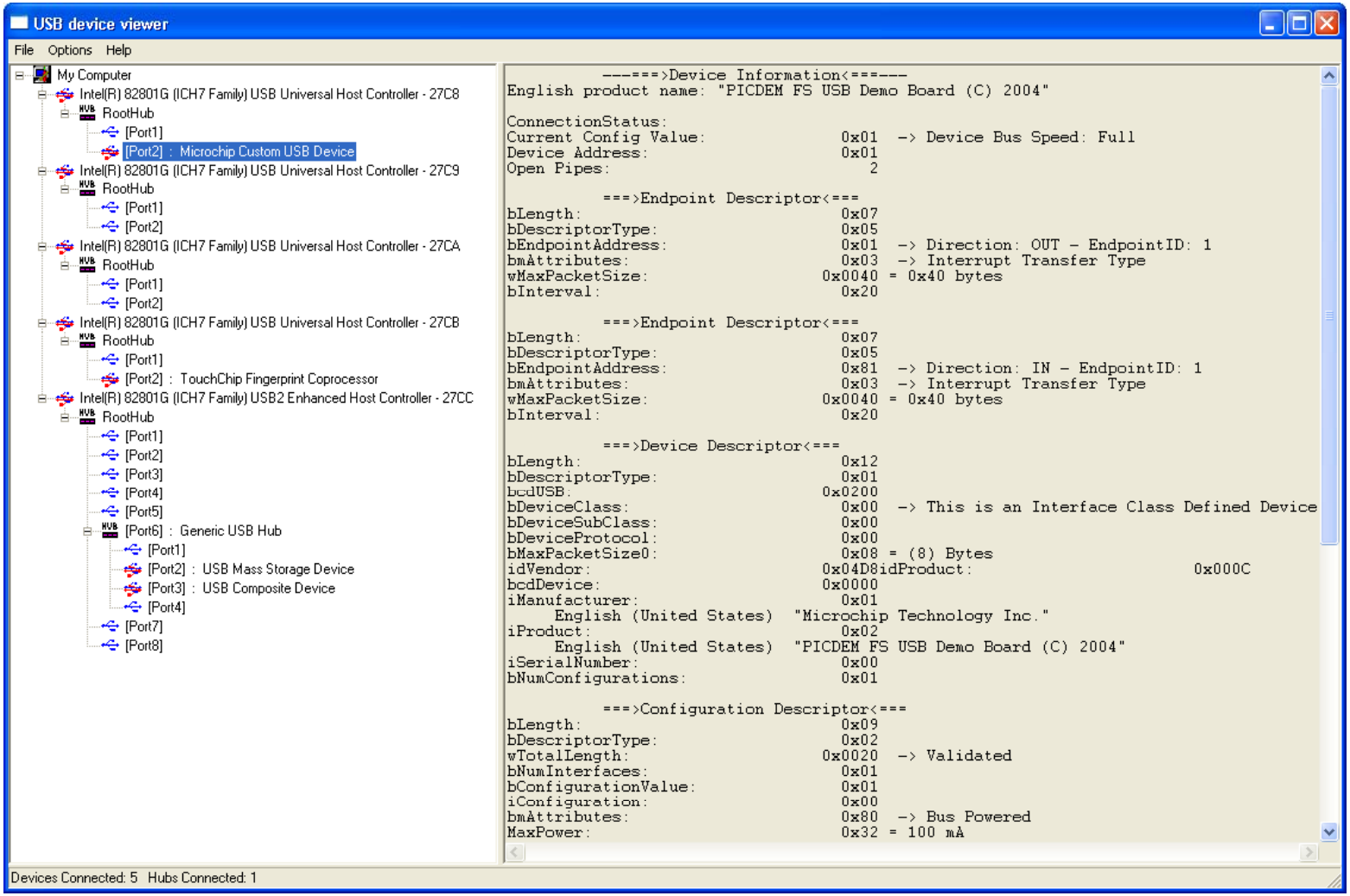
Device Descriptor Table

- **usb_descriptors.c**
  Descriptors :
  VID & PID,
  Class Specific, etc.

```
/* Device Descriptor */
ROM USB_DEVICE_DESCRIPTOR device_dsc={
   0x12,    // Size of this descriptor in bytes
   USB_DESCRIPTOR_DEVICE,
   …
 }
/* Configuration 1 Descriptor */
ROM BYTE configDescriptor1[]={
   /* Configuration Descriptor */
   0x09,//sizeof(USB_CFG_DSC),    // Size of this descriptor in bytes
   …
   /* Interface Descriptor */
   0x09,//sizeof(USB_INTF_DSC),   // Size of this descriptor in bytes
   USB_DESCRIPTOR_INTERFACE,            // INTERFACE descriptor type
    …
 /* HID Class-Specific Descriptor */
   0x09,//sizeof(USB_HID_DSC)+3,    // Size of this descriptor in bytes
   DSC_HID,             // HID descriptor type
   DESC_CONFIG_WORD(0x0111),
    …
 /* Endpoint Descriptor */
   0x07,/*sizeof(USB_EP_DSC)*/
   USB_DESCRIPTOR_ENDPOINT,    //Endpoint Descriptor
 }
```

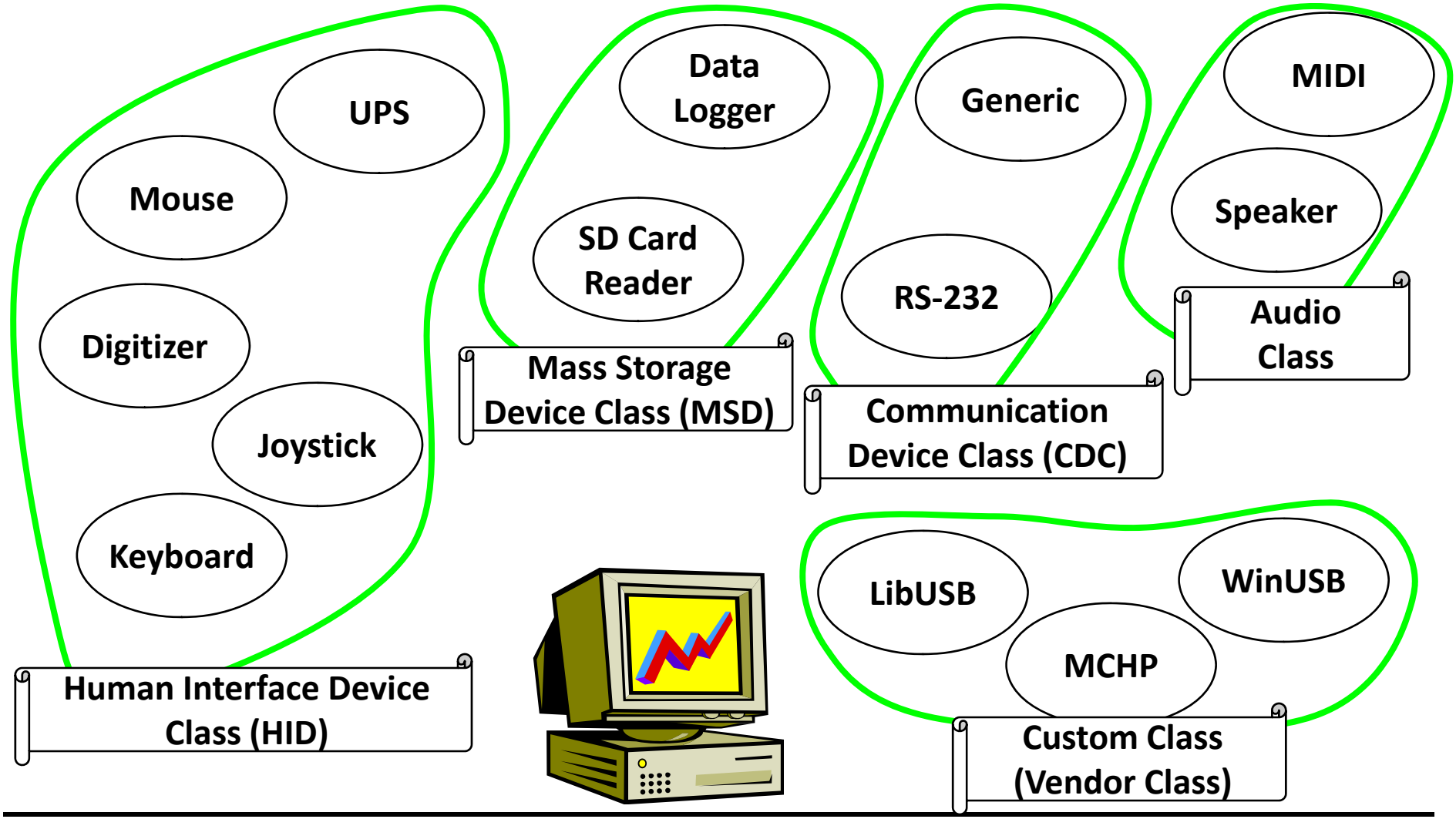
MICROCHIP
Regional Training Centers

Viewing Descriptor Information



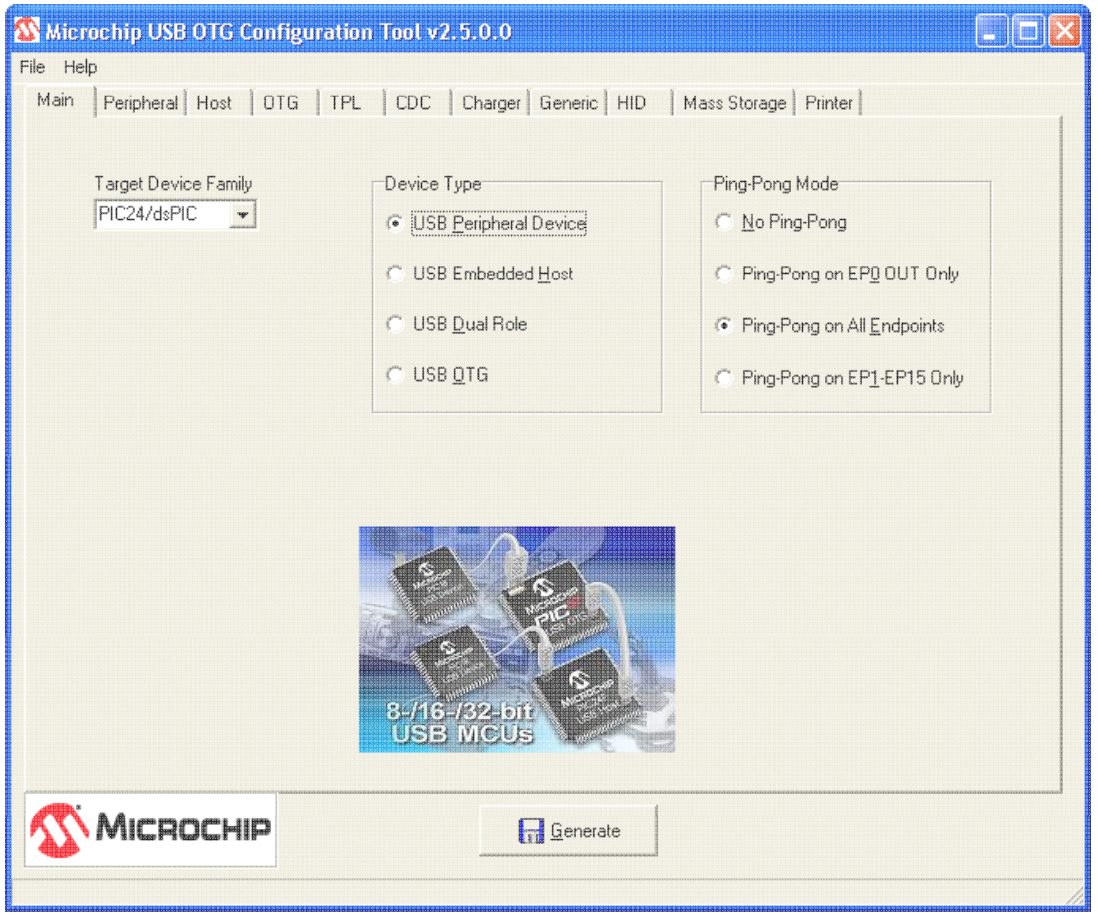
MCHPFSUSB Software Framework

Available Device Class



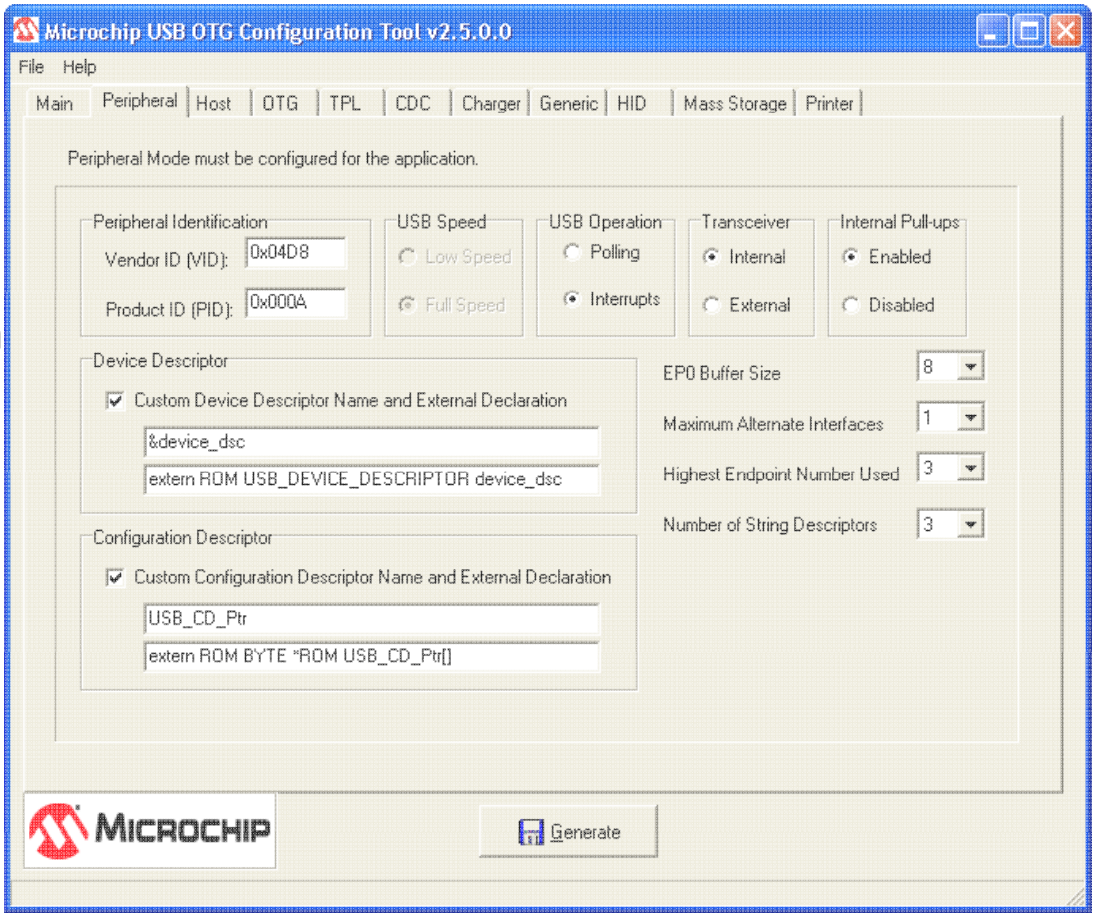
USB Stack Configuration Tool

- **Main Page:**
  - **Target Device**
  - **Device Type**
  - **Ping-Pong Mode**



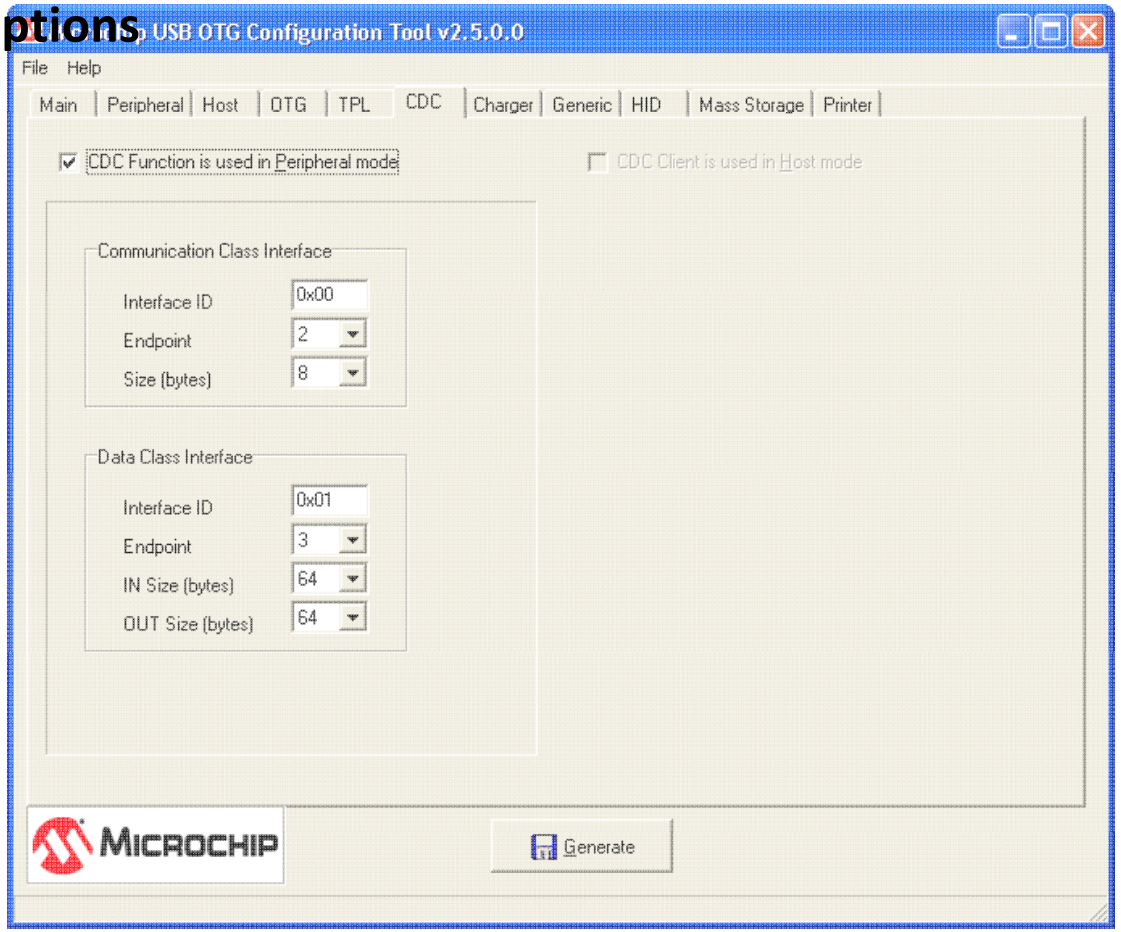
USB Stack Configuration Tool

- **Peripheral:**
  - VID &PID
  - Speed
  - USB operation
  - Transceiver options
  - Device and configuration
  - descriptors pointers
  - Endpoint 0 Buffer Size
  - # interfaces
  - # strings



USB Stack Configuration Tool

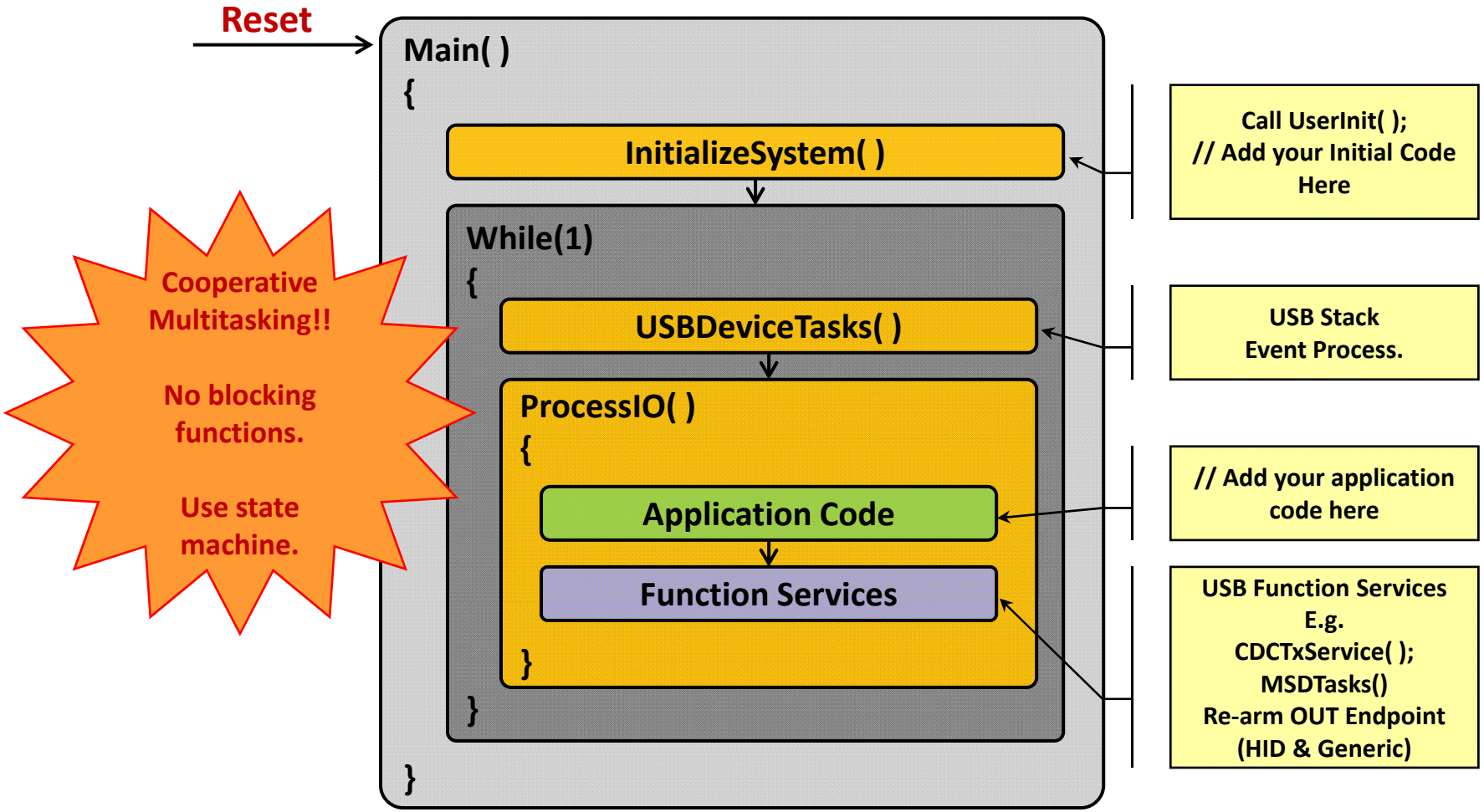
- **Function:**
  - Class specific interface options
  - Endpoint used
  - Endpoint configuration



MICROCHIP
Regional Training Centers

MCHPFSUSB Framework

Polled Base Flow



MICROCHIP
Regional Training Centers

Polled Base Code Example

- main.c

```
#include "./USB/usb.h"
#include "./USB/usb_function_cdc.h"
#include "HardwareProfile.h"
void main( void )
{
   InitializeSystem( );
   while(1)
   {
      USBDeviceTasks( );
      ProcessIO();
   }
}
static void InitializeSystem(void)
{
   ...
   UserInit( );
}
void ProcessIO(void)
{
   if((USBDeviceState < CONFIGURED_STATE)||(USBSuspendControl==1))
      return;
      ...
   CDCTxService();
}
```

// main loop

// Add your Initial Code Here

// Add your application code here

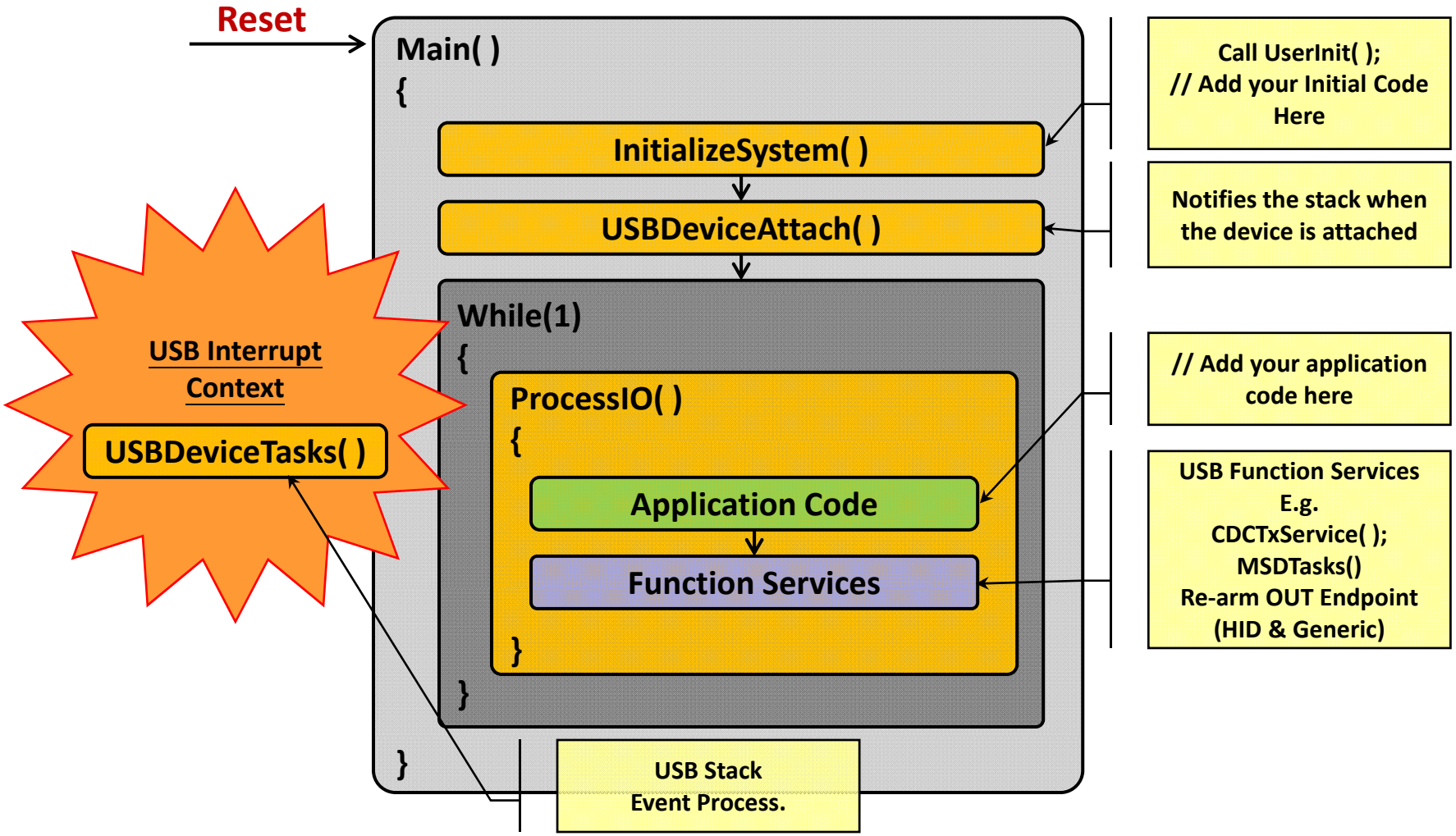
No blocking functions.

Use state machine.

MICROCHIP
Regional Training Centers

MCHPFSUSB Framework

Interrupt Base Flow



MICROCHIP
Regional Training Centers

Interrupt Base Code Example

- main.c

```
#include "./USB/usb.h"
#include "./USB/usb_function_cdc.h"
#include "HardwareProfile.h"
void main( void )
{
   InitializeSystem( );
   USBDeviceAttach();
   while(1)
   {
      ProcessIO();
   }
}
static void InitializeSystem(void)
{
   ...
   UserInit( );
}
void ProcessIO(void)
{
   if((USBDeviceState < CONFIGURED_STATE)||(USBSuspendControl==1))
      return;
      ...
   CDCTxService();
}
```

// main loop

// Add your Initial Code Here

// Add your application code here

```
void USBISR( void )
{
   USBDeviceTasks( );
}
```

USB Interrupt Context

MICROCHIP

Regional Training Centers

The HID Device Class

MICROCHIP
Regional Training Centers

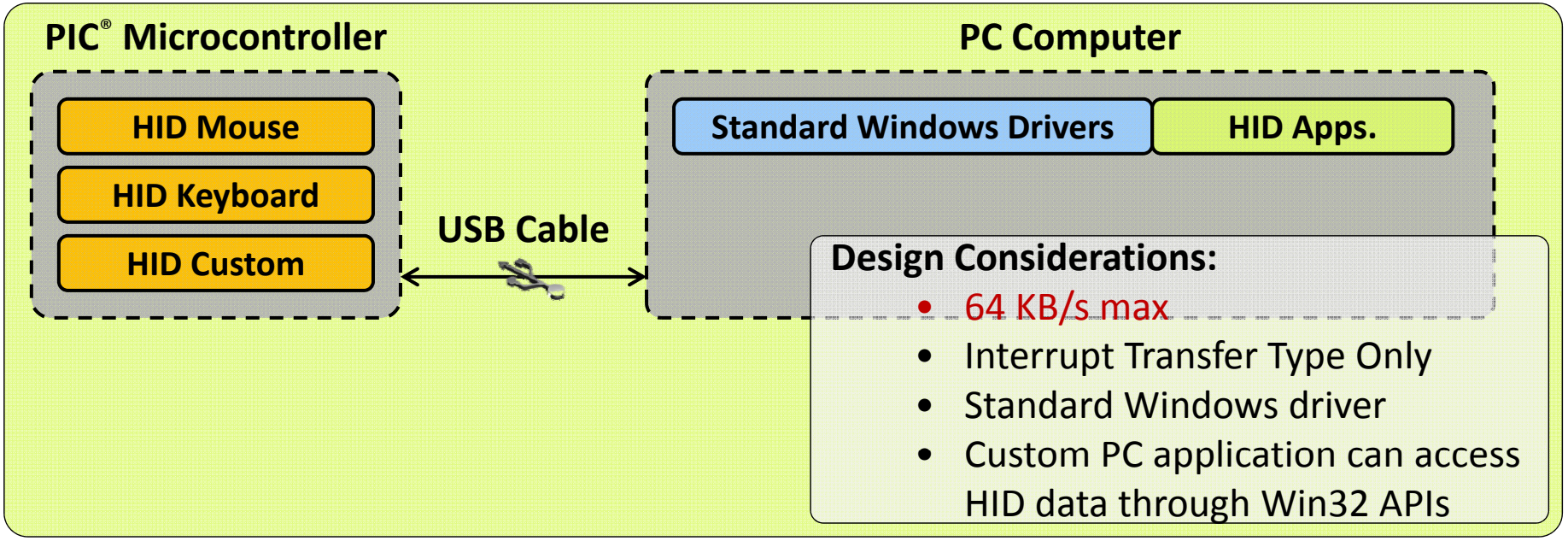
HID Device Class

- **HID : Human Interface Device**
  - Specification defined by USB-IF
    http://www.usb.org/developers/hidpage/
- **Designed for anything a human interacts with which might be connected to a computer**
  - keyboards, mice, joysticks, buttons/LEDs, digitizers.
- **Designed for devices that operate in "human time"**
  - Low rate data (64 kB/Sec), Limited latency (1 ms)
  - Devices can be of a standard type or of a vendor (Custom) defined type.
  - Devices can have standard data fields and vendor-defined data fields in the same report.

MICROCHIP
Regional Training Centers

HID Device Class

- **HID is an interface-level class**
  - Devices can have a HID interface along with other class interfaces (or even vendor-defined interfaces)
- **Devices can have multiple HID interfaces**
  - For example : Keyboard + Mouse (Composite Device)



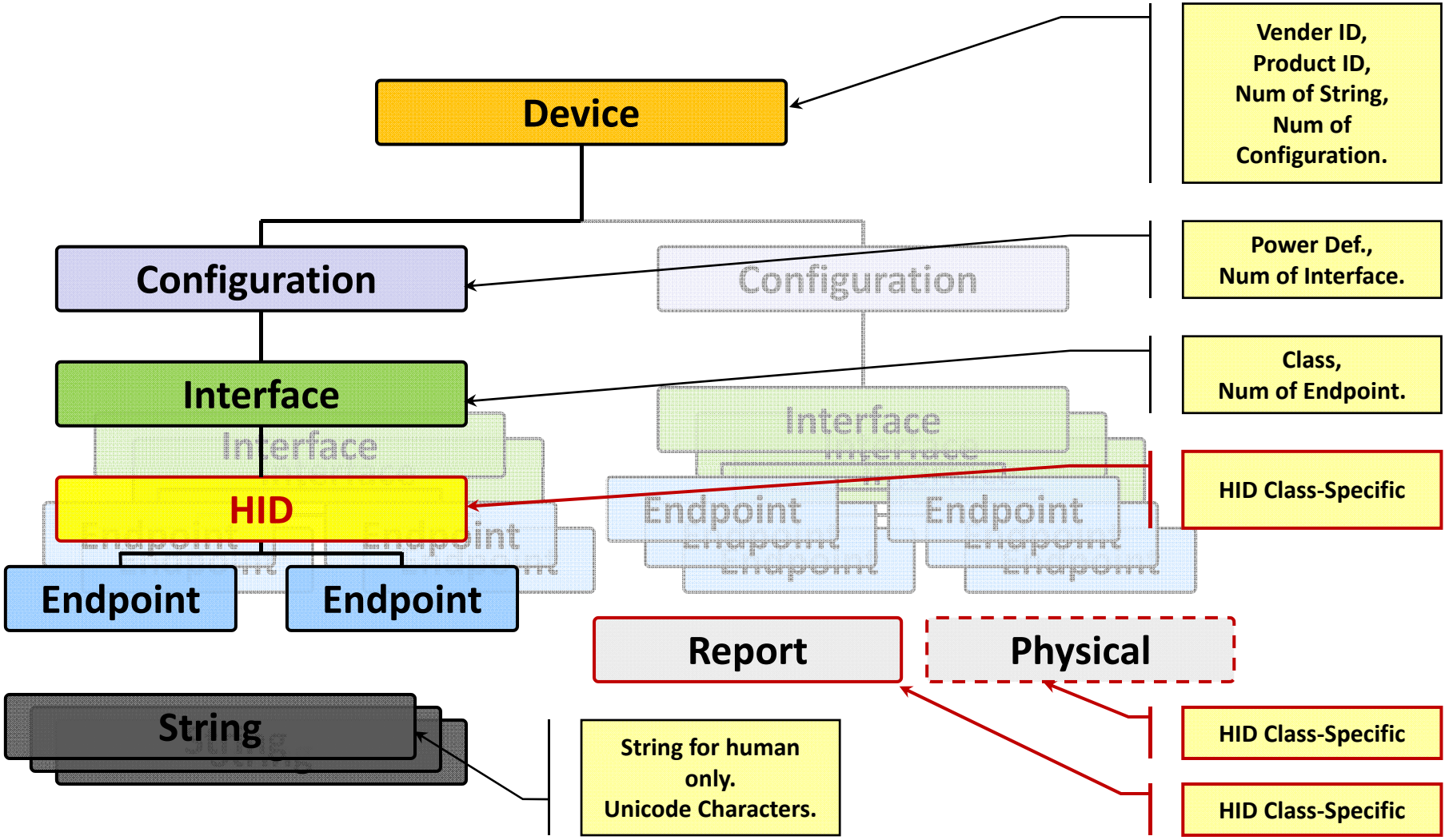
MICROCHIP
Regional Training Centers

HID Device Class

- **HID devices have Below Endpoint:**
  - Control endpoint (**SETUP**)
  - Interrupt **IN** endpoint
  - Interrupt **OUT** endpoint (Optional)
- **Remember, interrupt endpoints are 64k Bytes/Sec**
  - One 64 bytes transaction per 1ms frame.(Full Speed)
  - Theoretical maximum, less often in practice.
  - Plenty of bandwidth for human interaction.

MICROCHIP
Regional Training Centers

HID Class Descriptors



MICROCHIP
Regional Training Centers

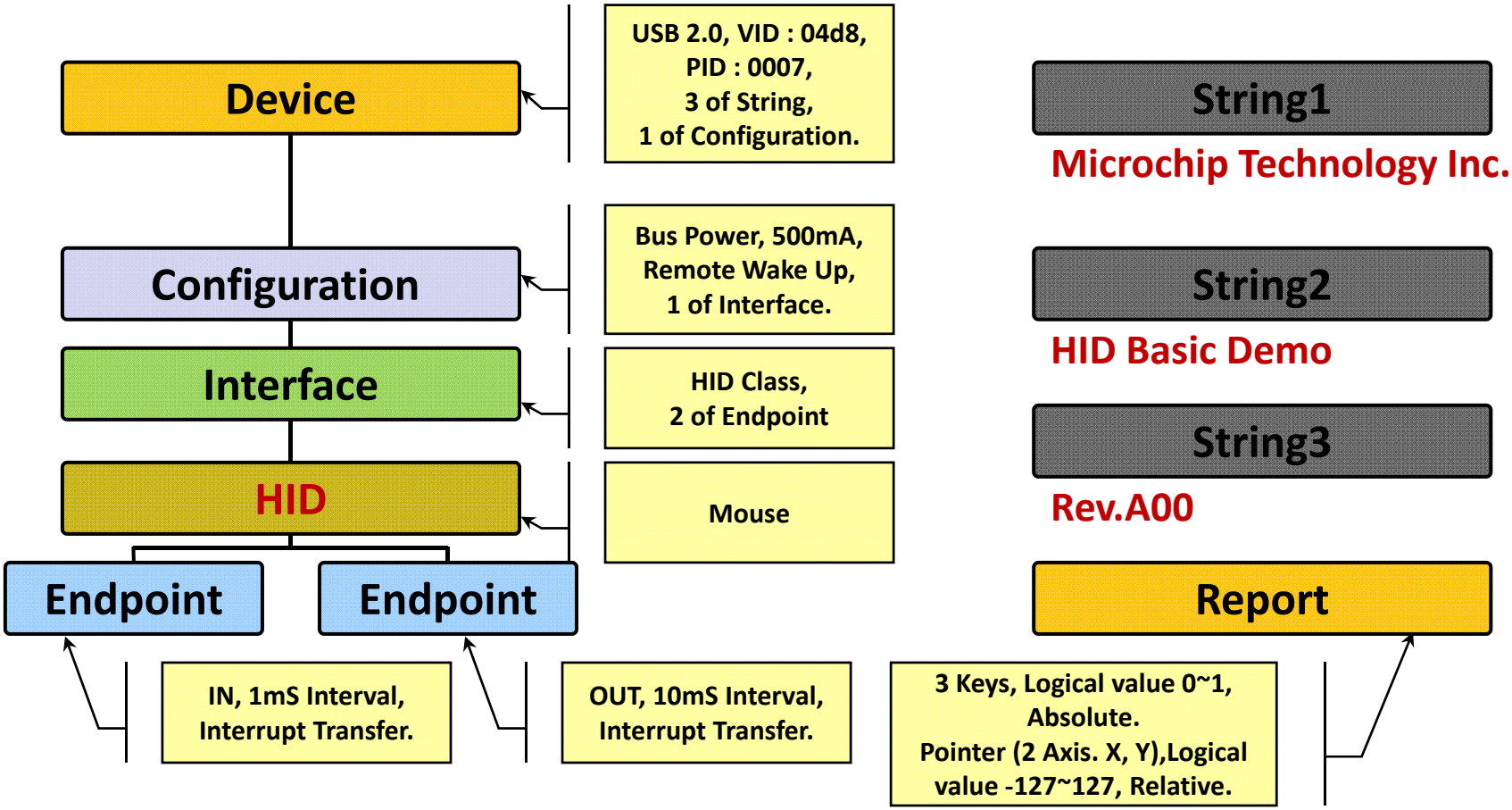
HID Device Class

Report Descriptor

- **HID devices contain a HID Report Descriptor which describes the format of the data transferred between the device and host.**
- **The host uses the Report Descriptor to determine how to parse data received from the device. Describes the format of reports that the device expects or will send.**
- **Uses constants which come from the HID Usage Tables (HUT) document. See USB-IF HID page at http://USB.org**
- **E.g.:**
  - The Report descriptor defines that bit 0 of byte 1 of the INPUT report represents a left mouse button state.
  - The host uses the Report Descriptor to know how to format data to send to the device.
  - The Report descriptor defines that bit 0 of byte 1 of the OUTPUT report represents caps lock LED state.

MICROCHIP
Regional Training Centers

Descriptors Example (Mouse)



MICROCHIP
Regional Training Centers

HID Device Class
Report Descriptor

- **Reports Descriptor**
  - Data is transferred to and from HID devices using reports.
  - Devices can have multiple reports of each type if desired.
- **3 Types of Reports:**
  - **Input**
    Device to Host on Interrupt **IN** endpoint. Used for normal data.
    E.g. : Key press, button click.
  - **Output**
    Host to Device on Interrupt **OUT** endpoint or control endpoint.
    Used for normal data.
    E.g. : Caps lock LED.
  - **Feature**
    Sent or Requested by the Host using Control endpoint Bi-directional,
    Used for configuration data.
    Many devices do not have feature reports, but use input and output reports for everything.

MCHPFSUSB Software Framework

Report Descriptor Table

- usb_descriptors.c // HID Mouse

```
//Class specific descriptor - HID
ROM struct{BYTE report[HID_RPT01_SIZE];}hid_rpt01={
{
   0x05, 0x01, /* Usage Page (Generic Desktop)          */
   0x09, 0x02, /* Usage (Mouse)                        */
   0xA1, 0x01, /* Collection (Application)              */
   0x09, 0x01, /*  Usage (Pointer)                     */
   0xA1, 0x00, /*  Collection (Physical)               */
   0x05, 0x09, /*     Usage Page (Buttons)             */
   0x19, 0x01, /*     Usage Minimum (01)               */
   0x29, 0x03, /*     Usage Maximum (03)               */
   0x15, 0x00, /*     Logical Minimum (0)              */
   0x25, 0x01, /*     Logical Maximum (1)              */
   0x95, 0x03, /*     Report Count (3)                 */
   0x75, 0x01, /*     Report Size (1)                  */
   0x81, 0x02, /*     Input (Data, Variable, Absolute) */
   0x95, 0x01, /*     Report Count (1)                 */
   0x75, 0x05, /*     Report Size (5)                  */
   0x81, 0x01, /*     Input (Constant)   ;5 bit padding */
   0x05, 0x01, /*     Usage Page (Generic Desktop)     */
   0x09, 0x30, /*     Usage (X)                        */
   0x09, 0x31, /*     Usage (Y)                        */
   ...
}};
```

MICROCHIP
Regional Training Centers

Subclass

- **Subclass (Device Descriptor Level)**
  - subclasses were intended to be used to identify the specific protocols of different types of HID class.
  - current HID class does not use subclass to define most protocol.
  - **HID class device identifies it's data protocol and type of data provided within its report descriptor.**
  - Report descriptor is loaded and parsed by the HID class driver.

MICROCHIP
Regional Training Centers

Subclass

- The parser for Report descriptor requires a significant amount of code!
- A simpler method needed for device requiring BIOS support (Boot device).
- HID class device use the subclass part to indicate devices that support a predefined protocol.
- For either mouse devices or keyboards
  The *bInterfaceSubClass* number of Interface Descriptor declares whether a device support a boot interface.

Subclass Codes	Description
0	No Subclass
1	Boot Interface
2~255	Reserved

MICROCHIP
Regional Training Centers

Protocols

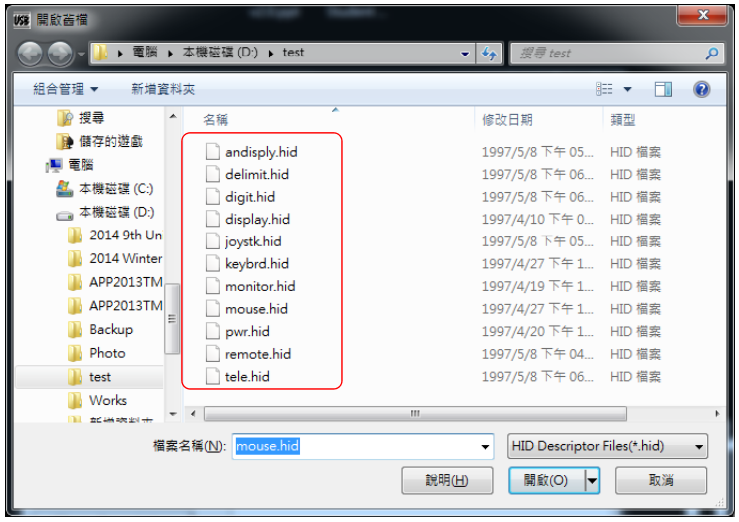
- **Protocol (Device Descriptor Level)**
  - A verity of protocols are supported HID device
    The ***bInterfaceProtocol*** value at Interface Descriptor only has meaning. **if The *bInterfaceSubClass* member of Interface Descriptor declares that the device supports a boot interface Otherwise , it's 0.**

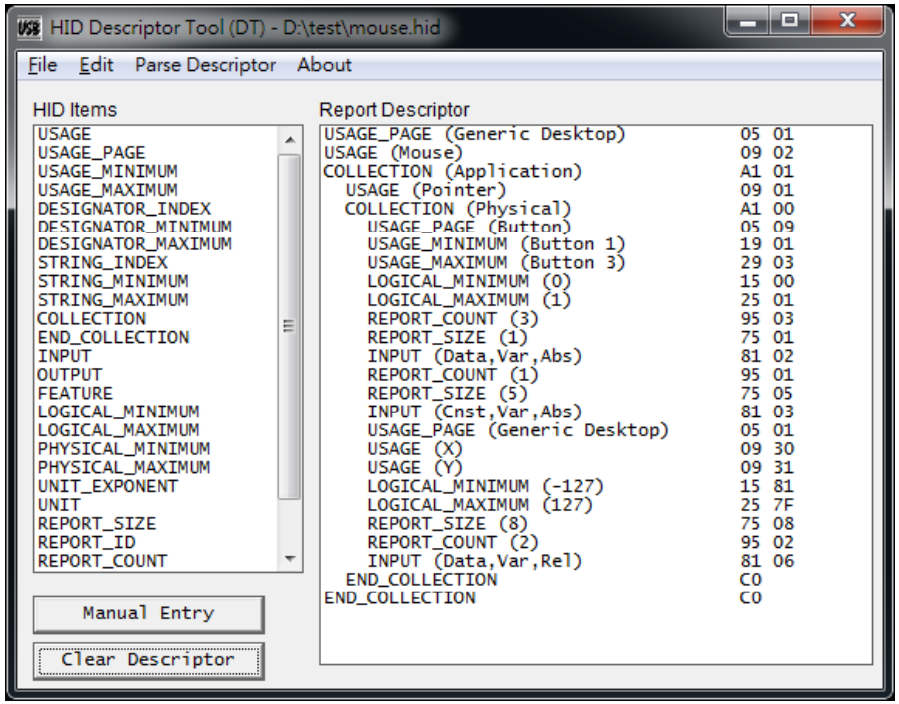
Protocol Codes	Description
0	None
1	Keyboard
2	Mouse
3~255	Reserved

MICROCHIP
Regional Training Centers

HID Report Descriptor Tool

- Make HID descriptors, View example HID descriptors !
- Parse Descriptors & Shows some errors!
- Export as C and ASM structures
- Some trivial modification to the generated code is needed to work with Microchip MLA.



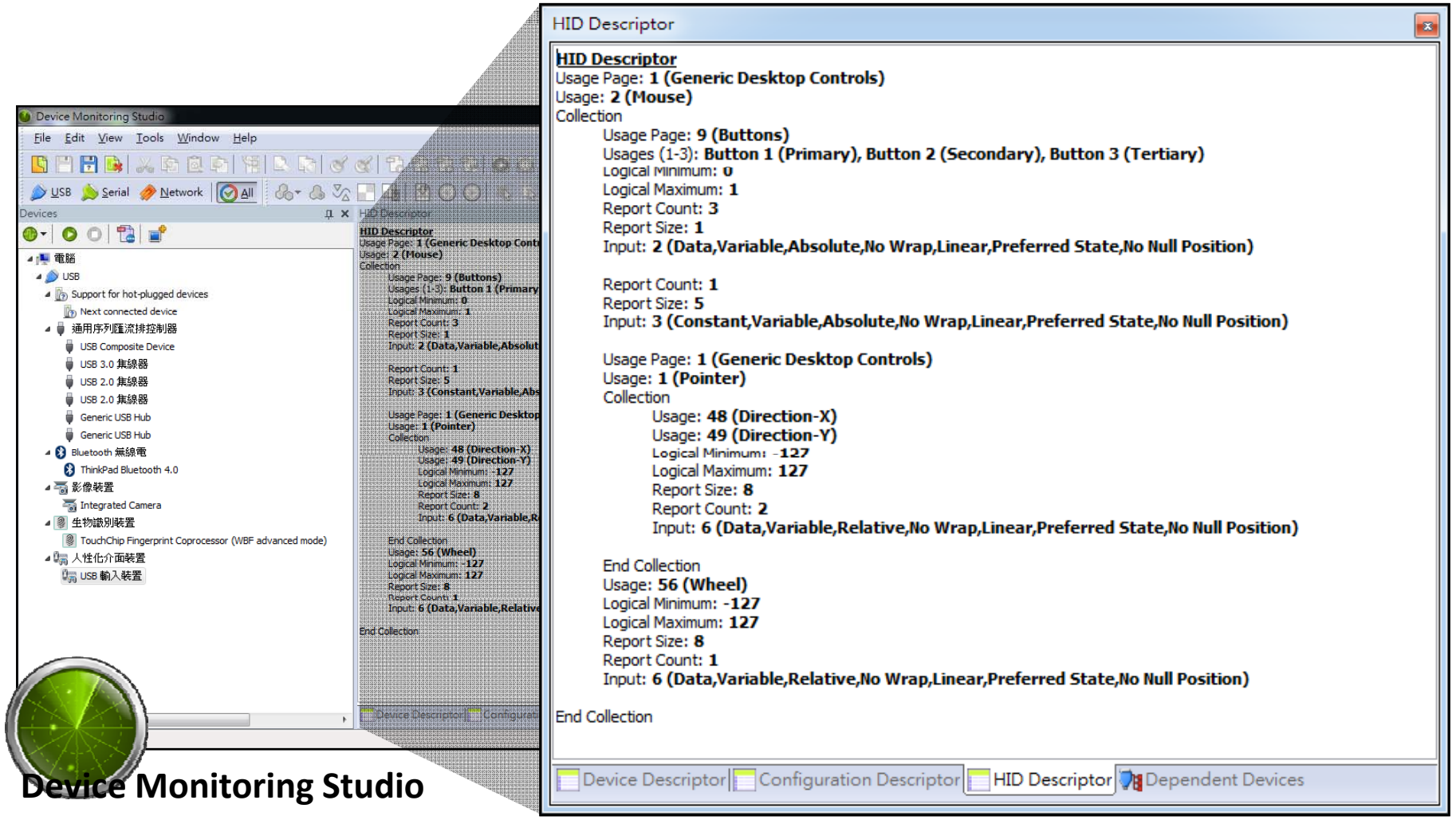


MICROCHIP

Regional Training Centers

Reference form other product

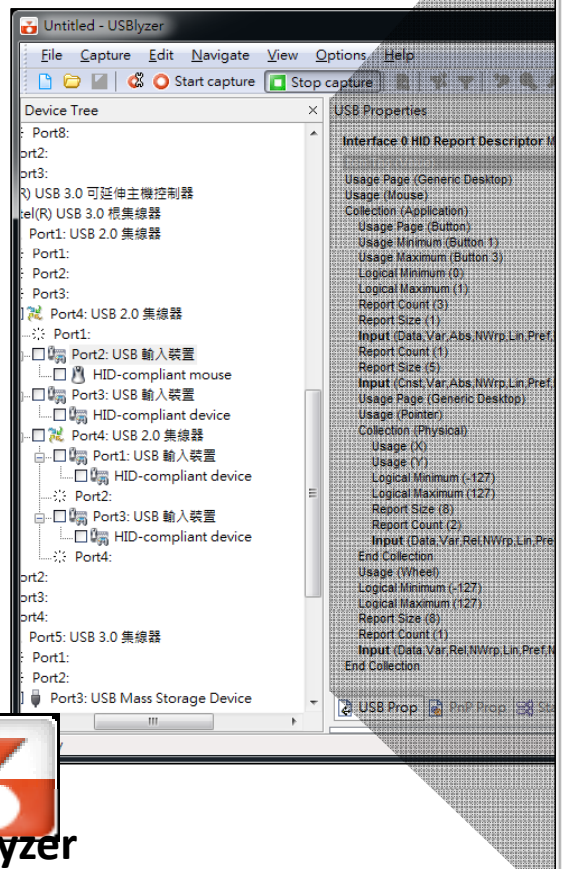
Example : Mouse



Device Monitoring Studio

Reference form other product

Example : Mouse



USBlyzer

USB Properties

Interface 0 HID Report Descriptor Mouse

Item Tag (Value)	Raw Data
Usage Page (Generic Desktop)	05 01
Usage (Mouse)	09 02
Collection (Application)	A1 01
Usage Page (Button)	05 09
Usage Minimum (Button 1)	19 01
Usage Maximum (Button 3)	29 03
Logical Minimum (0)	15 00
Logical Maximum (1)	25 01
Report Count (3)	95 03
Report Size (1)	75 01
Input (Data,Var,Abs,NWrp,Lin,Pref,NNul,Bit)	81 02
Report Count (1)	95 01
Report Size (5)	75 05
Input (Cnst,Var,Abs,NWrp,Lin,Pref,NNul,Bit)	81 03
Usage Page (Generic Desktop)	05 01
Usage (Pointer)	09 01
Collection (Physical)	A1 00
Usage (X)	09 30
Usage (Y)	09 31
Logical Minimum (-127)	15 81
Logical Maximum (127)	25 7F
Report Size (8)	75 08
Report Count (2)	95 02
Input (Data,Var,Rel,NWrp,Lin,Pref,NNul,Bit)	81 06
End Collection	C0
Usage (Wheel)	09 38
Logical Minimum (-127)	15 81
Logical Maximum (127)	25 7F
Report Size (8)	75 08
Report Count (1)	95 01
Input (Data,Var,Rel,NWrp,Lin,Pref,NNul,Bit)	81 06
End Collection	C0

MICROCHIP
Regional Training Centers

Quiz !

- **The HID protocol is defined at what level?**
  - Device
  - Configuration
  - Interface
  - Endpoint

- **Which type of endpoints are used in HID devices?**
  - Bulk
  - Interrupt
  - Control
  - Isochronous

MICROCHIP
Regional Training Centers

Answer !

- **The HID protocol is defined at what level?**
  - Device
  - Configuration
  - **Interface**
  - Endpoint

- **Which type of endpoints are used in HID devices?**
  - Bulk
  - **Interrupt**
  - **Control**
  - Isochronous

MICROCHIP

Regional Training Centers

The Custom HID Device

MICROCHIP
Regional Training Centers

Custom Class USB Devices

- **Why a custom class device?**
  - Sometimes devices need to transfer arbitrary data to a custom host application.
  - This data may not fit into a standard USB class such as Mass Storage, HID, or CDC.
- **This is what custom-class USB devices are for :**
  - Device Descriptor: bDeviceClass: 0x00 or 0xff
  - Interface Descriptor: bInterfaceClass 0xff: Vendor Defined
  - On Linux/Mac/FreeBSD, user-space programs have direct access to these types of devices using libusb.

MICROCHIP
Regional Training Centers

Custom Class USB Devices

- **Why not a custom class device?**
  - On Windows it's not so simple:

    Until recently, custom-class devices have required a driver to be written on Windows.
  - Currently, one can use WinUSB out of the box on Vista, 7, and 8. Windows XP requires a manual Install of WinUSB.
  - Additionally, on Windows XP (and on un-patched vista and 7), an INF file must be created for your device and "installed."
  - This INF file requires signing by Microsoft in order to avoid annoying popups when a user installs it.

MICROCHIP
Regional Training Centers

Custom Class USB Devices

- **Why not a custom class device?**
  - With Windows 8, Microsoft has introduced a WinUSB Descriptor which binds the WinUSB driver to your device automatically, with no INF file required.
  - The WinUSB descriptor goes in your USB device.
  - Non-standard extension
  - It's about time they at least made this possible.
  - A better solution would have been to allow WinUSB to talk to ANY device, much like Linux/BSD/Mac. This would have required a permissions model for devices.
  - Recently, WinUSB descriptor support has been pushed to
  - some older OS's through service packs. Consult Microsoft documentation for specifics.

MICROCHIP
Regional Training Centers

Custom HID Devices

- **Since WinUSB support is still difficult to deploy on some Windows versions, there is another option: Custom HID devices.**
- **Windows (for all currently supported versions) provides a user DLL which can send and receive raw data to and from HID devices.**
  - No driver required
  - No INF file required
- **HID devices can contain reports which only have user-defined data fields in them. These things together provide a way for normally-privileged user applications to communicate custom, vendor-defined data over a USB interface.**

MICROCHIP
Regional Training Centers

HID Device Class

Custom HID Device

- **For Custom HID Device**
  bInterfaceSubClass = 0
  bInterfaceProtocol = 0
  Usage Page at Report Descriptor = Vendor Defined
- Host **IN**/**OUT** the data base on interval at endpoint descriptor.

```
//Class specific descriptor - HID
ROM struct{BYTE report[HID_RPT01_SIZE];}hid_rpt01={
{
   0x06, 0x00, 0xFF, // Usage Page = 0xFF00 (Vendor Defined)
   0x09, 0x01,       // Usage (Vendor Usage 1)
   0xA1, 0x01,       // Collection (Application)
   0x19, 0x01,       //      Usage Minimum
   0x29, 0x40,       //      Usage Maximum      (64 input usages)
   0x15, 0x00,       //      Logical Minimum
   0x26, 0xff, 0x00, //      Logical Maximum 255 (16-bit field)
   0x75, 0x08,       //      Report Size: 8-bit field size
   0x95, 0x40,       //      Report Count: 64 of Report Size
   0x81, 0x00,       //      Input (Data, Array, Abs)
   0x19, 0x01,       //      Usage Minimum: 1
   0x29, 0x40,       //      Usage Maximum: (64 output usages)
   0x91, 0x00,       //      Output (Data, Array, Abs)
   0xC0}          // End Collection
};
```

MICROCHIP
Regional Training Centers

Lab Prepare

- Install MPLAB X IDE (v3.26) or MPLAB IDE(v8.92)
- Install MPLAB C18 Lite (v3.47)
- Install Microchip Application Libraries (v2013-06-15)
- Install Microsoft Visual C++ 2008/2010 Express

MICROCHIP
Regional Training Centers

Microsoft Visual Studio

- Download from http://www.visualstudio.com/

MICROCHIP

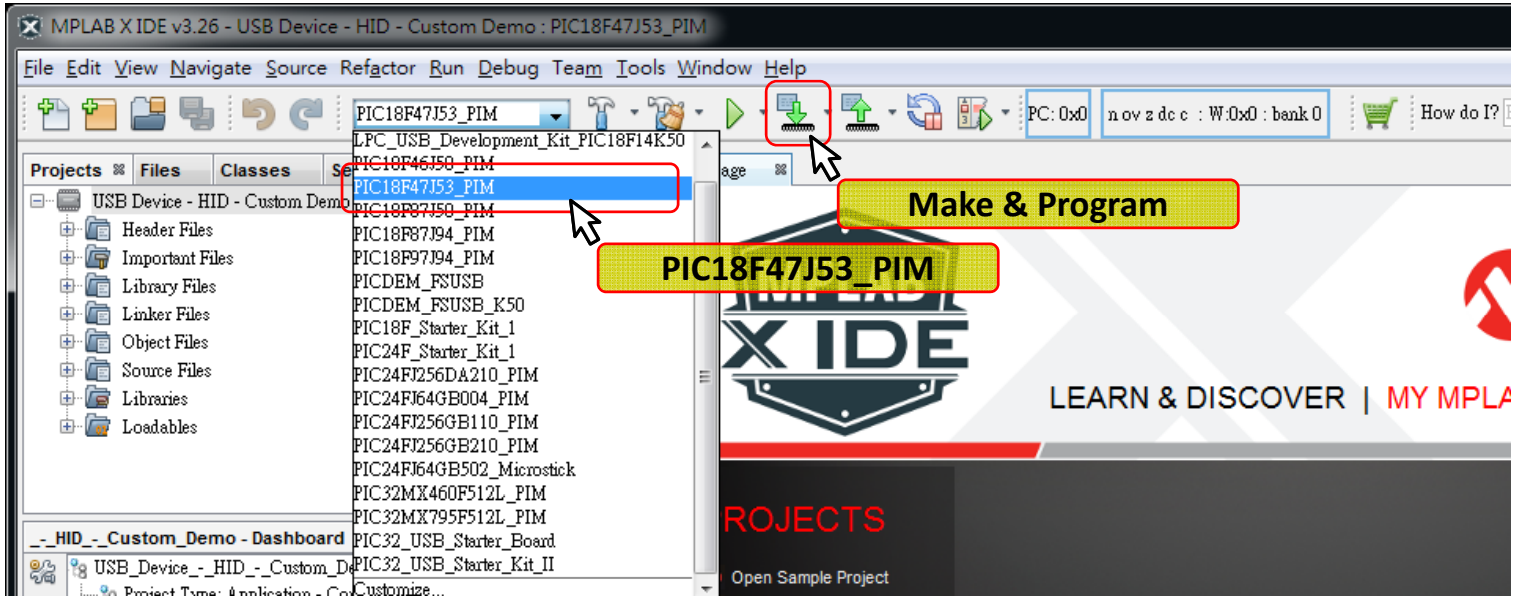
Regional Training Centers

Lab1
Case Study
Device HID Custom Demos

MICROCHIP
Regional Training Centers

Lab1 Exercise

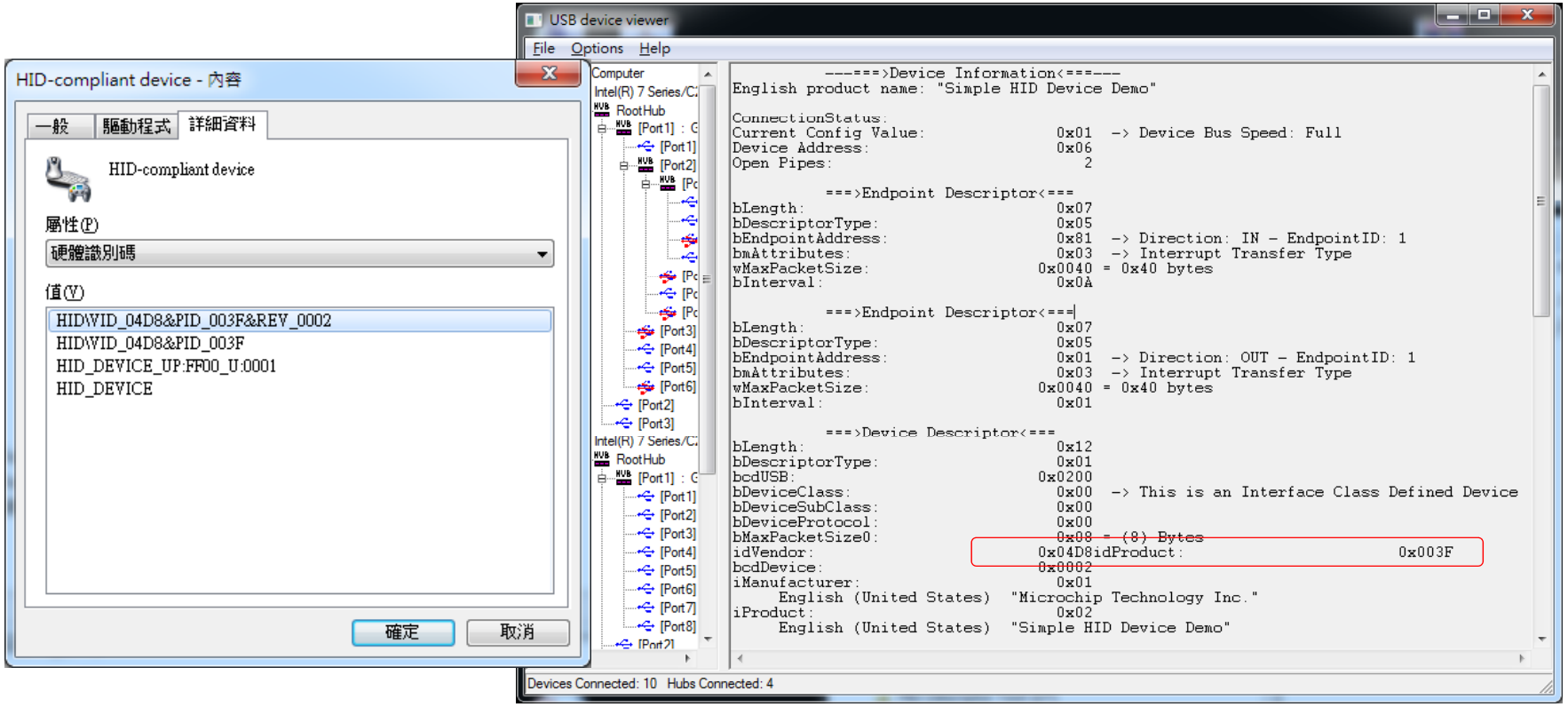
- Open Firmware Project
  - C:\microchip_solutions_v2013-06-15\USB\Device - HID - Custom Demos\Firmware\MPLAB.X
  - Configuration Select : PIC18F47J53_PIM
- Make & Program



MICROCHIP
Regional Training Centers

Lab1 Exercise

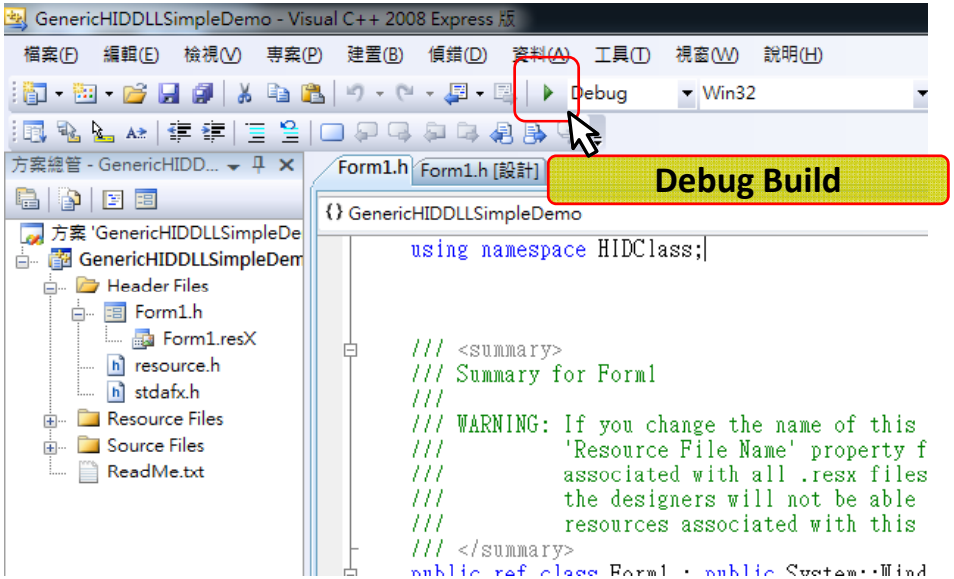
- Find The HID Device From Device Manager
- Check VID & PID (0x04d8, 0x003f)
- Use USB Device Viewer for detail



MICROCHIP
Regional Training Centers

Lab1 Exercise

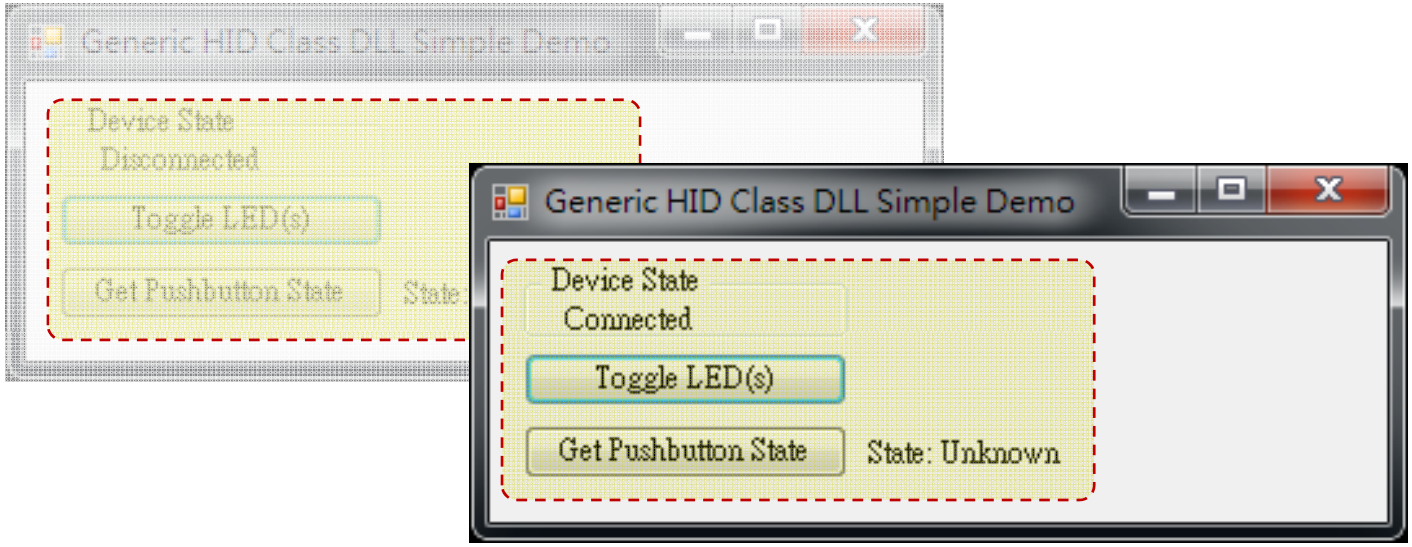
- Open PC Application Project
  - C:\microchip_solutions_v2013-06-15\USB\Device - HID - Custom Demos\HID DLL - PC Software\Microsoft Visual C++ 2008 Express\GenericHIDDLLSimpleDemo.vcproj
- Press Debug Build



MICROCHIP
Regional Training Centers

Lab1 Exercise

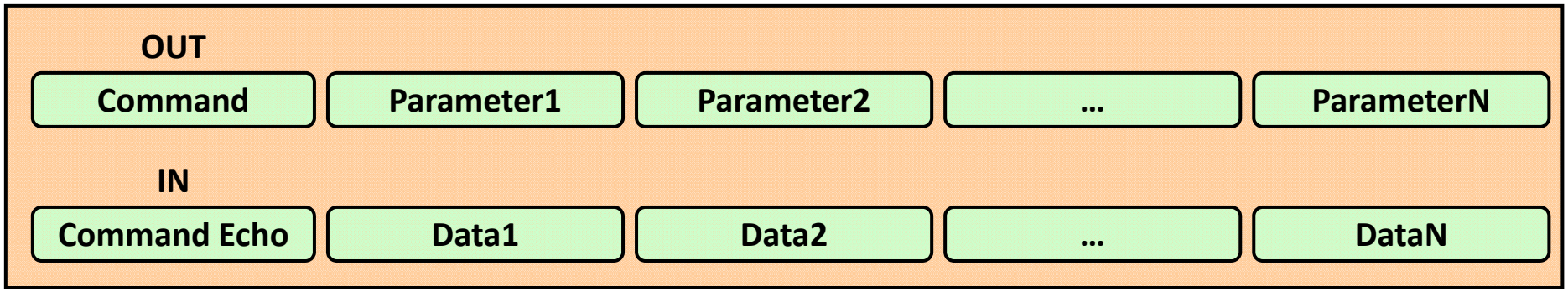
- Test App's Function & Observation
  - Device State – OK
  - Toggle LED(s) – OK
  - Get Push button State – OK, But Not Real Time !



MICROCHIP
Regional Training Centers

Command Response Base

- The example (Device - HID - Custom Demos) use command & response mode for data communication.
- Default command:
  - 0x80 : Toggle LEDs
  - 0x81 : Read Push Button Status
  - 0x37 : Read VR Value (Not Ready!)



Command Response Base

PIC Microcontroller

```
//Check if we have received an OUT data packet from the host
if(!HIDRxHandleBusy(USBOutHandle))
{
    //We just received a packet of data from the USB host.
    //Check the first byte of the packet to see what command the host
    //application software wants us to fulfill.
    switch(ReceivedDataBuffer[0])                //Look at the data the host sent, to see what kind of application specific command it sent.
    {
        case 0x80:  //Toggle LEDs command
                    blinkStatusValid = FALSE;          //Stop blinking the LEDs automatically, going to manually control them now.
            if(mGetLED_1() == mGetLED_2())
            {
                mLED_1_Toggle();
                mLED_2_Toggle();
            }
            else
            {
                if(mGetLED_1())
                {
                    mLED_2_On();
                }
                else
                {
                    mLED_2_Off();
                }
            }
            break;
```

PC Application

```
private: System::Void ToggleLED_btn_Click(System::Object^  sender, System::EventArgs^  e) {
         unsigned char OutputPacketBuffer[64];  //Allocate a memory buffer equal to our report size

         OutputPacketBuffer[0] = 0x80;      //0x80 is the "Toggle LED(s)" command in the firmware
         if (MCHPHIDClass::USBHIDWriteReport (OutputPacketBuffer, 1) == true)
         {
             lblDeviceState->Text = "Connected";
         }
         else
         {
             lblDeviceState->Text = "Disconnected";
         }
         }
```

MICROCHIP
Regional Training Centers

HID Class APIs & Functions

- **Windows Sides APIs** PC Application
  - void ***USBHIDClassInit*** (VID, PID, Size);
  - bool ***USBHIDWriteReport*** (buffer, len);
  - bool ***USBHIDReadReport*** (buffer);
  - Bool ***USBHIDIsConnected*** ( );
- **Firmware Functions** PIC Microcontroller
  - bool ***HIDTxHandleBusy*** (USB_HANDLE handle);
  - bool ***HIDRxHandleBusy*** (USB_HANDLE handle);
  - USB_HANDLE ***HIDTxPacket*** (BYTE ep , BYTE* data , WORD len);
  - USB_HANDLE ***HIDRxPacket*** (BYTE ep , BYTE* data , WORD len);

MICROCHIP
Regional Training Centers

USB HID Class DLL

- **Microchip provide 4 USB HID function for HID operation.**
  - .NET Assembly “HID class.dll”
  - Encapsulates the Win32 HID Data transfer APIs
  - Namespace HIDClass::MCHPHIDClass
- **Functions:**
  - void ***USBHIDClassInit*** (VID, PID, Size);
    Initial Device.
  - bool ***USBHIDWriteReport*** (buffer, len);
    Send a buffer to Device.
  - bool ***USBHIDReadReport*** (buffer);
    Read a packet from Device and return it to buffer.
  - Bool ***USBHIDIsConnected*** ( );
    Check with the OS to see if Device is connected.

MICROCHIP
Regional Training Centers

API Example

```
// Initial Device
MCHPHIDClass::USBHIDClassInit (0x4D8, 0x000A, 64);

// Connect Status
if (MCHPHIDClass::USBHIDIsConnected () == true)
{
      lblDeviceState->Text = “Connected“;
}

// Send OutBuffer to Device
if ( MCHPHIDClass::USBHIDIsConnected () == true)
      MCHPHIDClass::USBHIDWriteReport (OutBuffer, 5);

// Send OutBuffer to Device and and return data to InBuffer
if(MCHPHIDClass::USBHIDWriteReport (OutBuffer, 5) == true)
{
      MCHPHIDClass::USBHIDReadReport (InputBuffer);
}
```

MICROCHIP
Regional Training Centers

HID Receive Usage

```
/* Arm the OUT endpoint for Out packet */
USBOutHandle = HIDRxPacket(HID_EP, RxBuf, sizeof(RxBuf));

if (!HIDRxHandleBusy(USBOutHandle))
{
        /* Data has been received */
        Data = RxBuf[0];

        /* Re-arm the OUT endpoint for next packet */
        USBOutHandle = HIDRxPacket(HID_EP, RxBuf , sizeof(RxBuf));
}
```

MICROCHIP
Regional Training Centers

HID Transmit Usage

```
/* Pack data buffer */
TxBuf[0] = Data;

if (!HIDTxHandleBusy(USBOutHandle))
{
    /* Send data */
    USBInHandle = HIDTxPacket(HID_EP, TxBuf , sizeof(TxBuf));
}
```

Hardware Porting

HardwareProfile - PIC18F47J53 PIM.h

```
#define DEMO_BOARD          PIC18F47J53_PIM
#define PIC18F47J53_PIM
#define CLOCK_FREQ          48000000
#define GetSystemClock()    CLOCK_FREQ
…
#define mInitAllLEDs()      LATE &= 0xFC; TRISE &= 0xFC;
#define mLED_1              LATEbits.LATE0
#define mLED_2              LATEbits.LATE1
…
#define mInitSwitch2()      RISBbits.TRISB2=1;
#define mInitSwitch3()      mInitSwitch2();
#define mInitAllSwitches()  mInitSwitch2();
…
```

main.c

```
Void main( void )
{
   #if defined(PIC18F47J53_PIM)
      #pragma config OSC = HSPLL
      #pragma config PLLDIV = 3
      …
   #endif
  …
}
static void InitializeSystem(void)
{
   #if defined(PIC18F47J53_PIM)
      unsigned int pll_startup_counter = 600;
      OSCTUNEbits.PLLEN = 1;
      while(pll_startup_counter--);
      ....
   #endif
   …
}
```

MICROCHIP

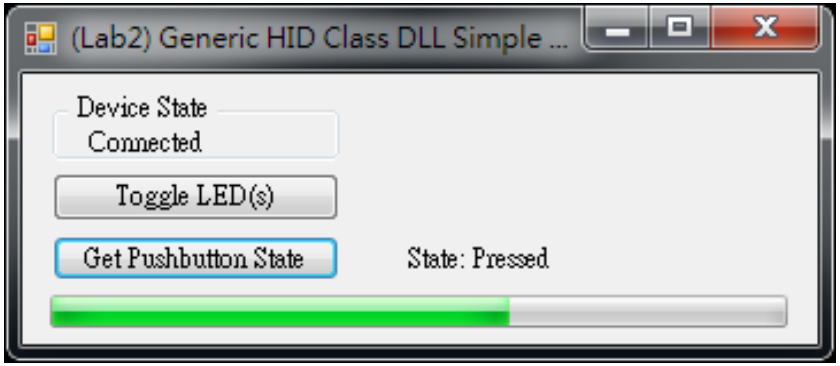
Regional Training Centers

Lab2
HID Custom Class Communication
Use Command Response Base

MICROCHIP
Regional Training Centers

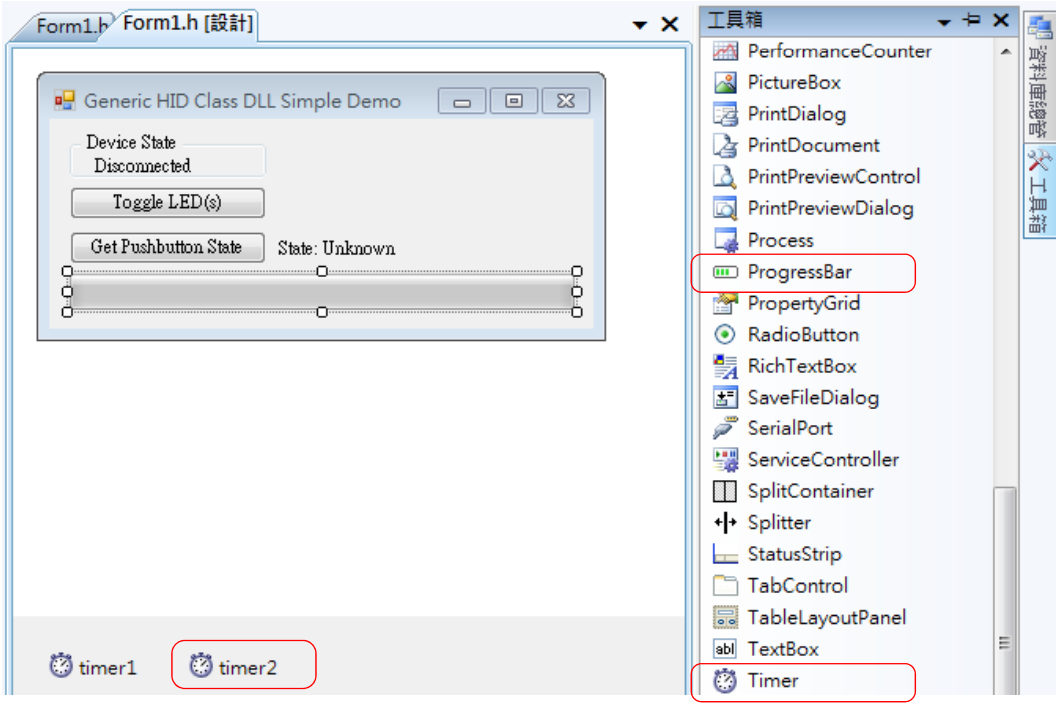
Lab2 Exercise

- Modify the project. Add new feature (Read VR Value) to the project.
  - 0x80 : Toggle LEDs
  - 0x81 : Read Push Button Status
  - 0x37 : Read VR Value
- Follow Command Response Base
- Use *ProgressBar* or *TrackBar* to Show VR Value
- Update the VR Value per 50mS (Add Timer)
- Firmware ready, Only need modify PC Application.



Lab2 Exercise

- Add Timer and *ProgressBar* control to Form1.h
- Set *Timer* : Interval (100), Enable(True).
- Set *ProgressBar* : Name(progressBar_VR), Min/Max Value(0/1023).



MICROCHIP
Regional Training Centers

Lab2 Exercise

- **Add some code to Timer2_Tick**

```
private: System::Void timer2_Tick(System::Object^  sender, System::EventArgs^  e)
{
        unsigned char OutputPacketBuffer[64];
        unsigned char InputPacketBuffer[64];

        OutputPacketBuffer[0] = ??;

        if (MCHPHIDClass::USBHIDWriteReport (OutputPacketBuffer, 1) == true)
        {


           /* Add Your Code Here * /


        }
}
```

MICROCHIP
Regional Training Centers

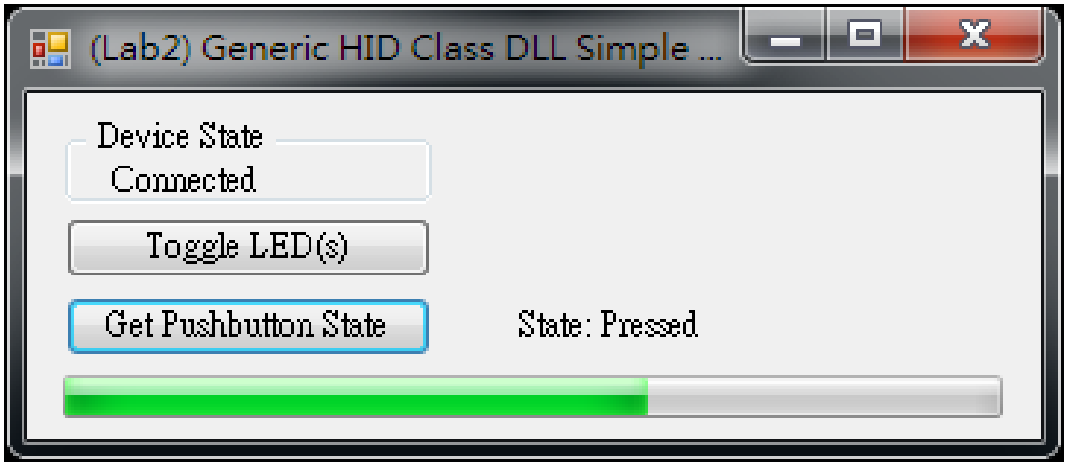
Lab2 Exercise

- add some code to Timer2_Tick

```
private: System::Void timer2_Tick(System::Object^  sender, System::EventArgs^  e)
{
        unsigned char OutputPacketBuffer[64];
        unsigned char InputPacketBuffer[64];
        unsigned long VRValue;

        OutputPacketBuffer[0] = 0x37;

        if (MCHPHIDClass::USBHIDWriteReport (OutputPacketBuffer, 1) == true)
        {
                MCHPHIDClass::USBHIDReadReport (InputPacketBuffer);
                VRValue = InputPacketBuffer[2] * 255 + InputPacketBuffer[1];
                if( VRValue >= 0 && VRValue <= 1023 )
                        progressBar_VR->Value = VRValue;
        }

        else
        {
                 lblDeviceState->Text = "Disconnected";
        }
}
```

Lab2 Exercise

- Test App's Function & Observation
  - Device State – OK
  - Toggle LED(s) – OK
  - Get Pushbutton State – OK, Not Real Time !
  - Get VR Value – OK, Real Time Update !



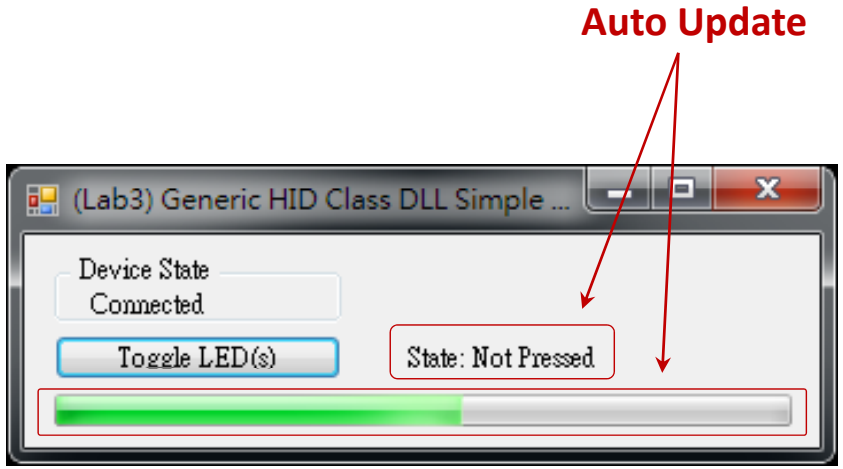
MICROCHIP

Regional Training Centers

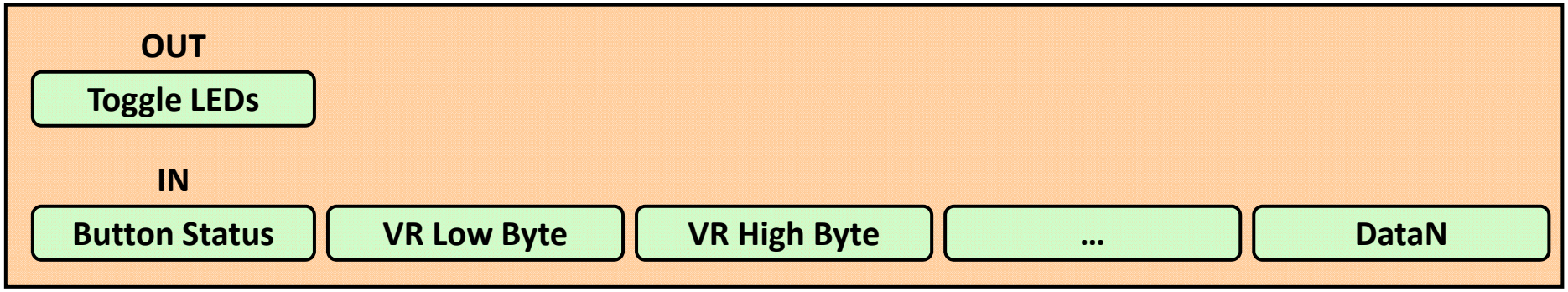
Lab3
HID Custom Class Communication
Use Auto Report

Lab3 Exercise

- Modify the project. (Command Response Base -> Auto Report)
  - HID Class have Interrupt transfer
  - Pack all data to Interrupt IN transfer
- Update the VR Value and Push Button Status per 50mS
- Both firmware and PC Application need to modify



Interrupt IN & OUT transfer Pack



MICROCHIP
Regional Training Centers

Lab3 Exercise

- add some code to Timer2_Tick

```
private: System::Void timer2_Tick(System::Object^  sender, System::EventArgs^  e)
{
        unsigned char InputPacketBuffer[64];
        unsigned long VRValue;

         if (MCHPHIDClass::USBHIDReadReport (InputPacketBuffer) == true)
         {

                /* Add Code Here * /


         }
}
```

MICROCHIP
Regional Training Centers

Lab3 Exercise

- add some code to ProcessIO( );

```
void ProcessIO(void)
{
    ...
    if(!HIDTxHandleBusy(USBInHandle))
    {

        /* Add Code Here * /

    }
    ...
}
```

MICROCHIP
Regional Training Centers

Lab3 Exercise

- add some code to Timer2_Tick

```
private: System::Void timer2_Tick(System::Object^  sender, System::EventArgs^  e)
{
        unsigned char InputPacketBuffer[64];
        unsigned long VRValue;

        if (MCHPHIDClass::USBHIDReadReport (InputPacketBuffer) == true)
        {
                lblDeviceState->Text = "Connected";

                if(InputPacketBuffer[0] == 0x01)
                        StateLabel->Text = "State: Not Pressed";
                else
                        StateLabel->Text = "State: Pressed";

                 VRValue = InputPacketBuffer[2] * 255 + InputPacketBuffer[1];
                if( VRValue >= 0 && VRValue <= 1023 )
                progressBar_VR->Value = VRValue;
        }
}
```

MICROCHIP
Regional Training Centers

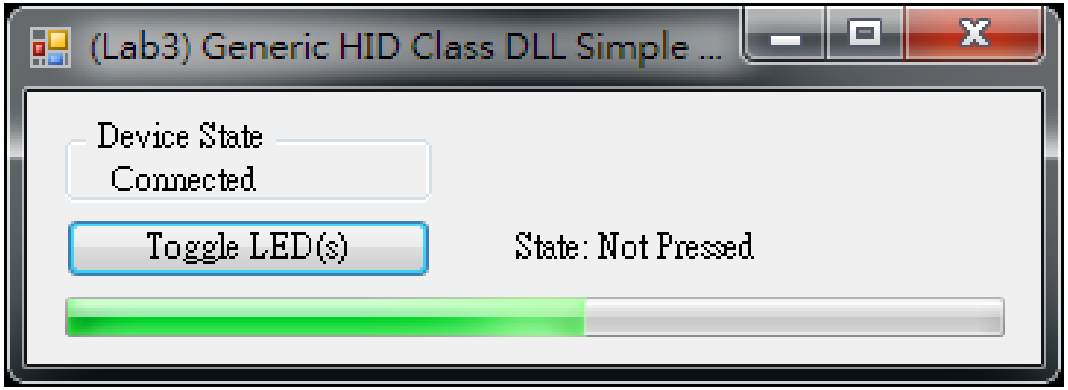
Lab3 Exercise

- add some code to ProcessIO( );

```
void ProcessIO(void)
{
    ...
    if(!HIDTxHandleBusy(USBInHandle))
    {
        WORD_VAL w;
        w = ReadPOT();

        ToSendDataBuffer[1] = w.v[0];
        ToSendDataBuffer[2] = w.v[1];

        if(sw3)
          ToSendDataBuffer[0] = 0x01;
        else
          ToSendDataBuffer[0] = 0x00;

        USBInHandle =
        HIDTxPacket(HID_EP,(BYTE*)&ToSendDataBuffer[0],64);
    }
    ...
}
```

Lab3 Exercise

- Test App's Function & Observation
  - Device State – OK
  - Toggle LED(s) – OK
  - Get Pushbutton State – OK, Real Time Update!
  - Get VR Value – OK, Real Time Update!



MICROCHIP
Regional Training Centers

Quiz !

- **It is safe to call HIDTxPacket() at any time.**
  - True
  - False

- **Calling HIDTxPacket() blocks until the packet is successfully sent.**
  - True
  - False

MICROCHIP
Regional Training Centers

Answer !

- **It is safe to call HIDTxPacket() at any time.**
  - True
  - **False**

False.
Only call HIDTxPacket( ) after determining that the endpoint is not busy by calling HIDTxHandleBusy().

- **Calling HIDTxPacket() blocks until the packet is successfully sent.**
  - True
  - **False**

False.
HIDTxPacket( ) schedules the packet to be sent the next time the host polls the endpoint.
The buffer passed in must remain valid until that time.

MICROCHIP
Regional Training Centers

Trademarks

- The Microchip name and logo, the Microchip logo, dsPIC, KeeLoq, KeeLoq logo, MPLAB, PIC, PICmicro, PICSTART, PIC32 logo, rfPIC and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.
- FilterLab, Hampshire, HI-TECH C, Linear Active Thermistor, MXDEV, MXLAB, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.
- Analog-for-the-Digital Age, Application Maestro, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, dsSPEAK, ECAN, ECONOMONITOR, FanSense, HI-TIDE, In-Circuit Serial Programming, ICSP, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, mTouch, Octopus, Omniscient Code Generation, PICC, PICC-18, PICDEM, PICDEM.net, PICkit, PICtail, REAL ICE, rfLAB, Select Mode, Total Endurance, TSHARC, UniWinDriver, WiperLock and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.
- SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.
- All other trademarks mentioned herein are property of their respective companies.
- © 2010, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

MICROCHIP

Regional Training Centers

Thank you !

MICROCHIP

Regional Training Centers

Development Tools

Microchip Full Speed USB 2.0 Demo Boards for other PIC18

Regional Training Centers

Silicon Product
+
Software Protocol
+
Dev Boards
=
Complete Solution

PIC18F46J50 USB Plug-in Module (PIM)

FS USB PIM
MA180024
$45
PIC18F46J50
Use Stand Alone or with PIC18 Explorer

Low Pin Count USB Development Kit

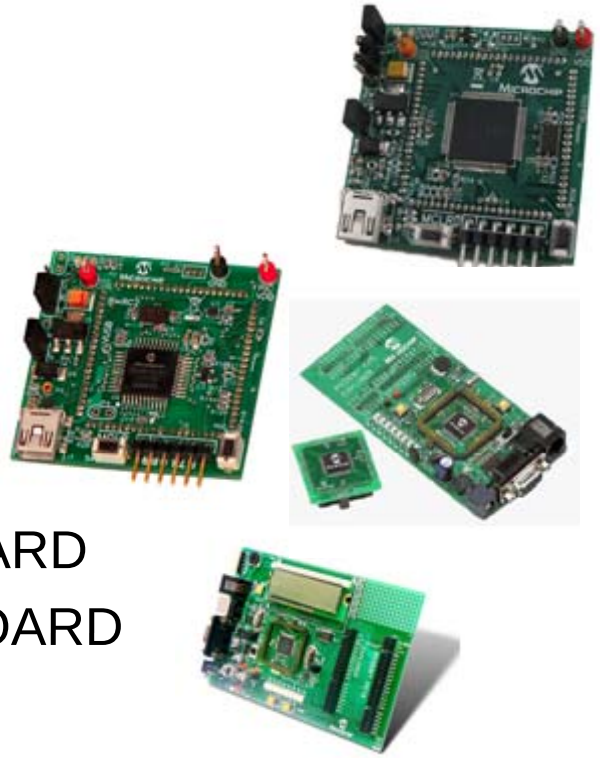
DV164126
$45
PIC18F13K50 / 14K50
Getting started with USB lab

PICDEM™ Full-Speed USB Demo Kit

- Contains everything you need to get started quickly
  - Use with any of the PIC18F4550 family microcontrollers
- Includes self-directed class and lab material
- The Demo Kit provides all of the hardware and software needed to demonstrate and develop a complete USB communication solution
- Priced from $59.99
- Part Numbers
  - DM163025
- Available Now

PIC18FXXJ50 Full-Speed USB Plug-In Module (PIM)

- Contains everything you need to get started quickly
  - Use with any of the PIC18F87J50 or PIC18F46J50 family microcontrollers
- Can be plugged into PICDEM™ HPC Explorer Board or PICDEM PIC18 Explorer Board
- Can be operated as a stand-alone board
- Priced from $40.00
- Part Numbers
  - MA180021 - PIC18F87J50 FS USB PIM
  - MA180024 - PIC18F46J50 FS USB PIM
  - DM183022 - PICDEM HPC EXPLORER BOARD
  - DM183032 - PICDEM PIC18 EXPLORER BOARD
- Available Now



PIC18 Starter Kit

- Functions as a USB mouse, joystick or mass storage device all using the on-board capacitive touch sense pads
- Includes a MicroSD™ memory card, potentiometer, acceleration sensor, and OLED display
- On-board debugger/programming
- Completely USB-powered
- Demonstrates PIC18 Family
  - USB communication
- Priced from $59.98
- Part Numbers
  - DM180021
- Available Now

Low Pin Count USB Development Kit

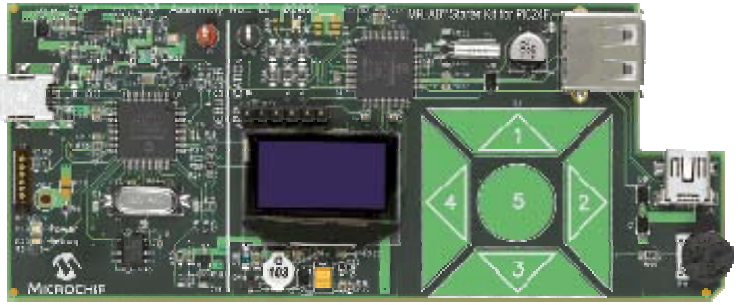
- Contains everything you need to get started quickly
  - Use with new 20-pin PIC18F USB microcontrollers – PIC18F13K50, PIC18F14K50
- Includes self-directed class and lab material
- Quickly implement common USB functions:
  - RS-232 to Serial
  - Keyboard/Mouse, etc…
- Priced from $39.99
- Part Numbers
  - DV164126 (w/PICkit™ 2)
  - DM164127
- Available Now

16-/32-bit USB Development Boards

Regional Training Centers

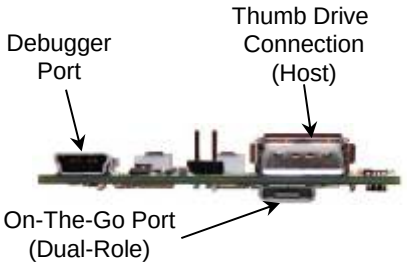
- PIC24F Starter Kit 1
  - Part #: DM240011
  - PIC24FJ256GB110

  Priced from $59.99



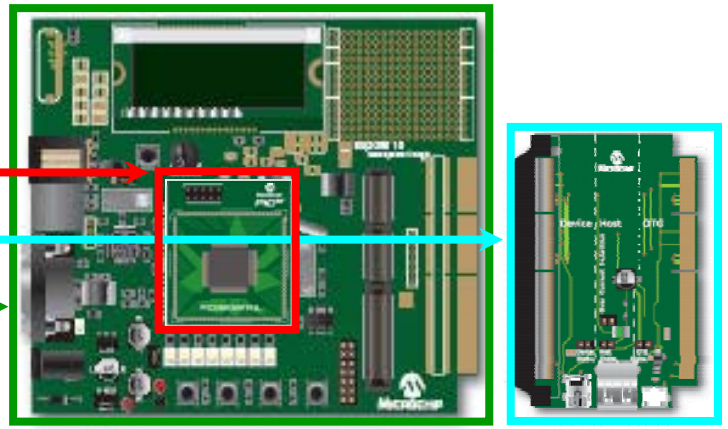
- PIC32 USB Starter Board II
  - Part #: DM320003-2
  - PIC32MX795F512L

  Priced from $55.00



- Explorer 16 + USB PICtail™ Plus Daughter Board + USB PIMs
  - Part #: MA320002/MA240014
  - Part #: AC164131
  - Part #: DM240001

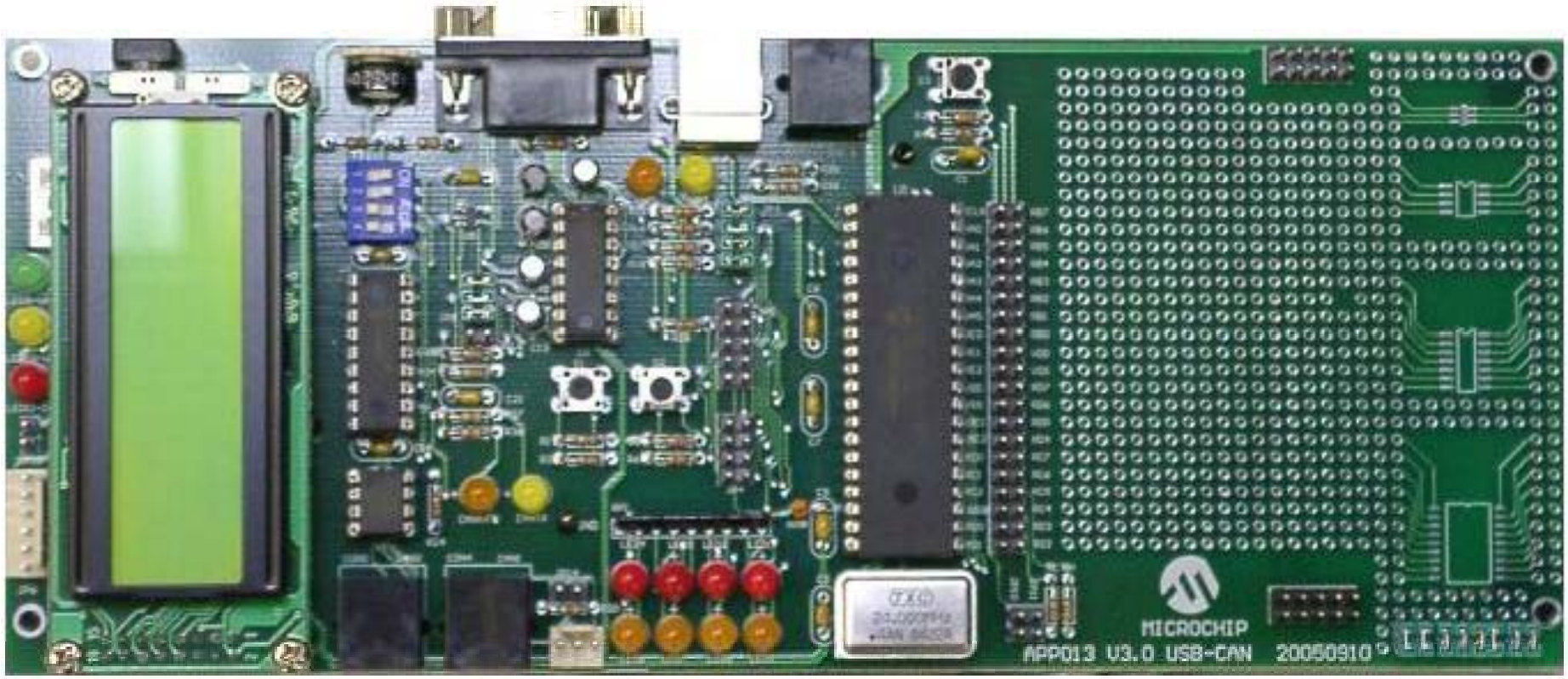
  Priced from $214.99



- All Available Now

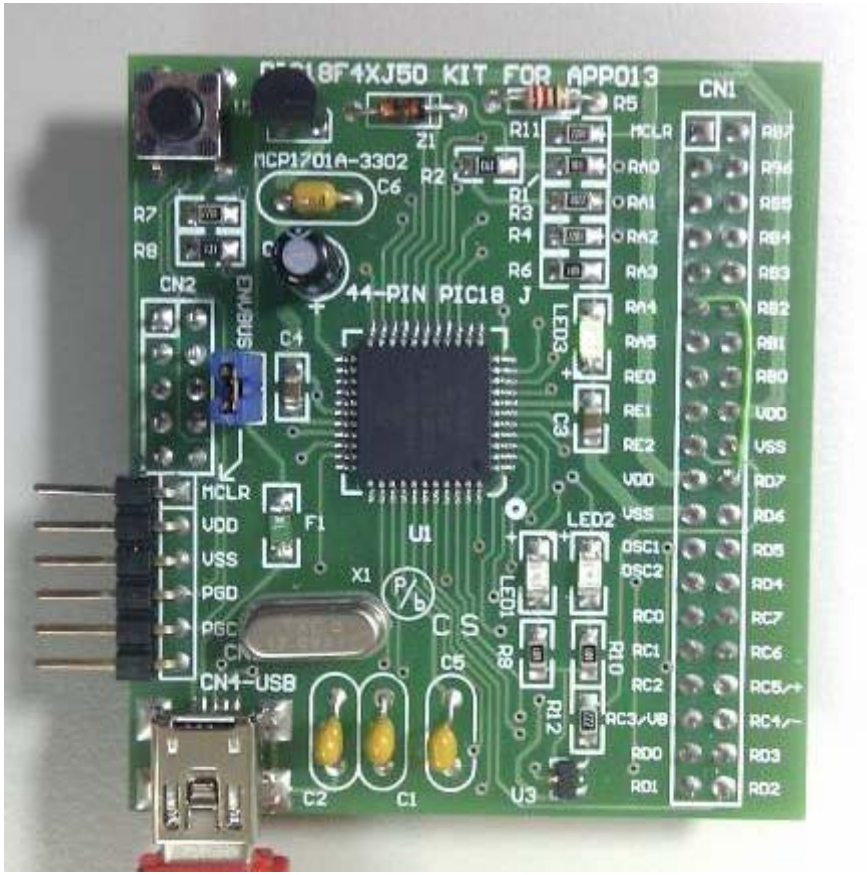
APP013

- APP013相容於DM163025, 但新增許多功能 (CAN, LCD Module and ICSP)



APP013 Hobby Kit(停售)

- Hobby Kit為APP013的子版, 可以透過子版跟新不同系列之MCU.



EDF09 & APP1618-SD

- **EDF09 與APP1618-SD**
  搭載
  PIC18F46J50(EDF09)
  PIC18F47J53(APP1618-SD)
  可透過**USB**供電, 獨立操作