MICROCHIP

Regional Training Centers

COM 3101T v1.0

Introduction to Full-Speed USB

Authors: *Giacomo Colombo*
Dennis Cecic
Modify: *Adam Syu*
Microchip Technology Inc.

MICROCHIP

Regional Training Centers

許育財 Adam Syu

Microchip CAE

MICROCHIP
Regional Training Centers

Class Objectives

- When you finish this class you will be able to:
  - Describe the basics of USB, and how to apply them in an embedded application
  - Identify Microchip's USB MCUs, development boards, and USB software frameworks relevant to your project
  - Analyze the capabilities and limitations of the CDC device class, and implement basic communications using the CDC device framework on PIC18/24/32 MCUs

MICROCHIP
Regional Training Centers

Agenda

- **Part 1:**

  Introduction to Full-Speed USB

- **Part 2:**

  Introduction to Microchip's USB Device Frameworks

- **Part 3:**

  Using the Microchip CDC Class Device Framework (RS-232 Replacement)

Regional Training Centers

Class Folders

MCHPFSUSB Framework is also installed in C:\Microchip Solutions v20xx-xx-xx

COM3101T

\Exercise\

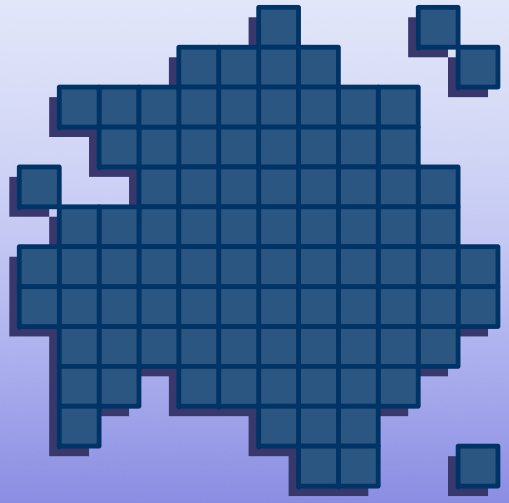
Lab1a, Lab1b, .., Lab5

Microchip (Framework)

\USB Precompiled Demos (*.hex)

\USB Tools

Presentation Files



Part 1.

Introduction to Full-Speed USB

MICROCHIP
Regional Training Centers

Agenda – Part 1

- USB Fundamentals – The serious & important stuff
  - Basics/Speeds
  - Topology/Physical Connection
  - Architecture/Programmer's Model
  - USB Transactions
  - USB Transfers
  - Device Classes
  - Enumeration
  - Descriptors
  - Power Planning
  - VID/PID & USB Compliance
- PIC18/24/32 USB Microcontrollers
- Microchip Demo/Development Solutions

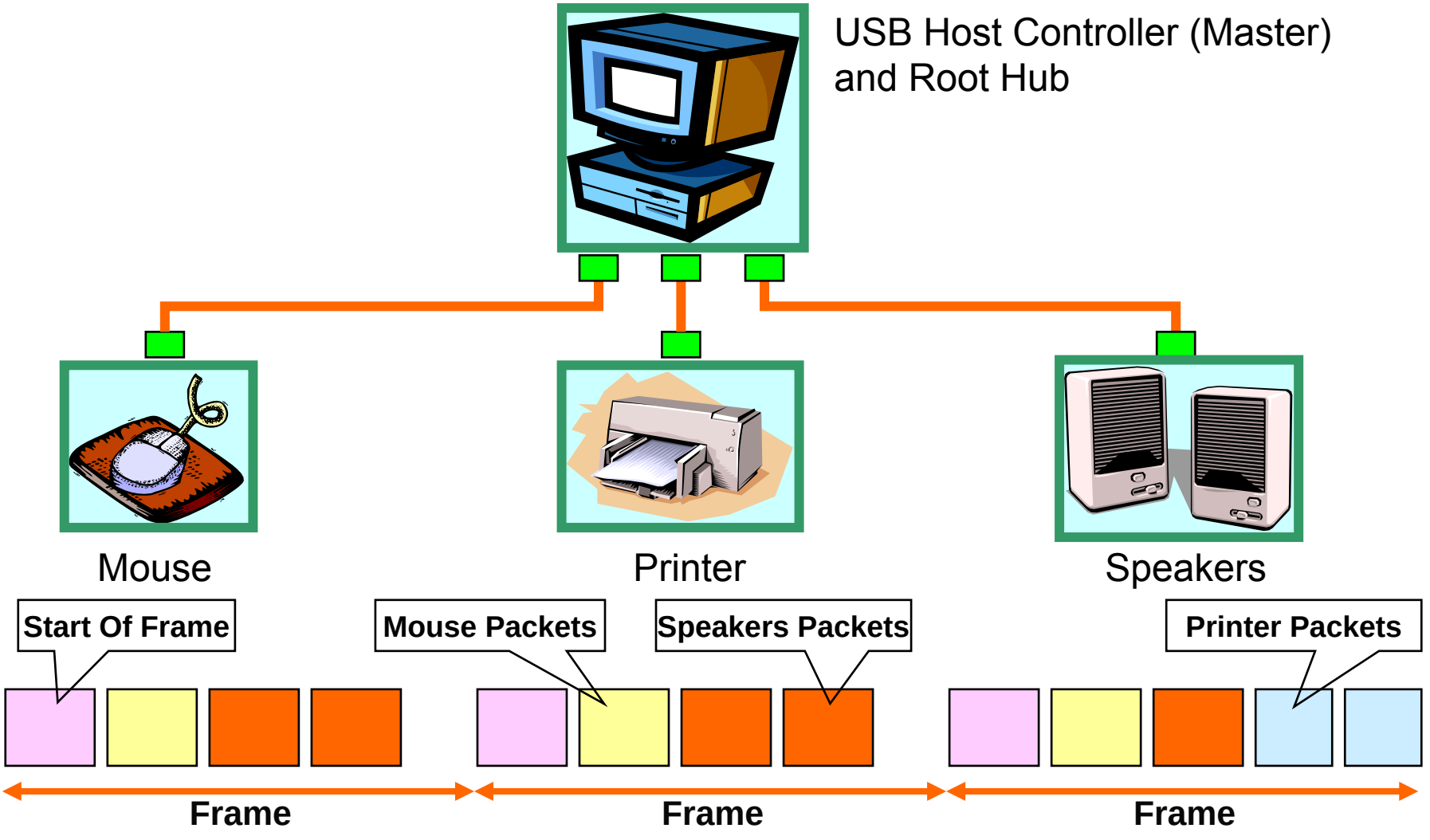
MICROCHIP
Regional Training Centers

A little history...

- 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
- Sep., 1998 – USB 1.1
- Apr., 2000 – USB 2.0
- 2003 – On-the-Go supplement to USB 2.0 (v1.0a)

USB Basics

- USB is a “Single Master + Multiple Slaves” **Polled Bus**



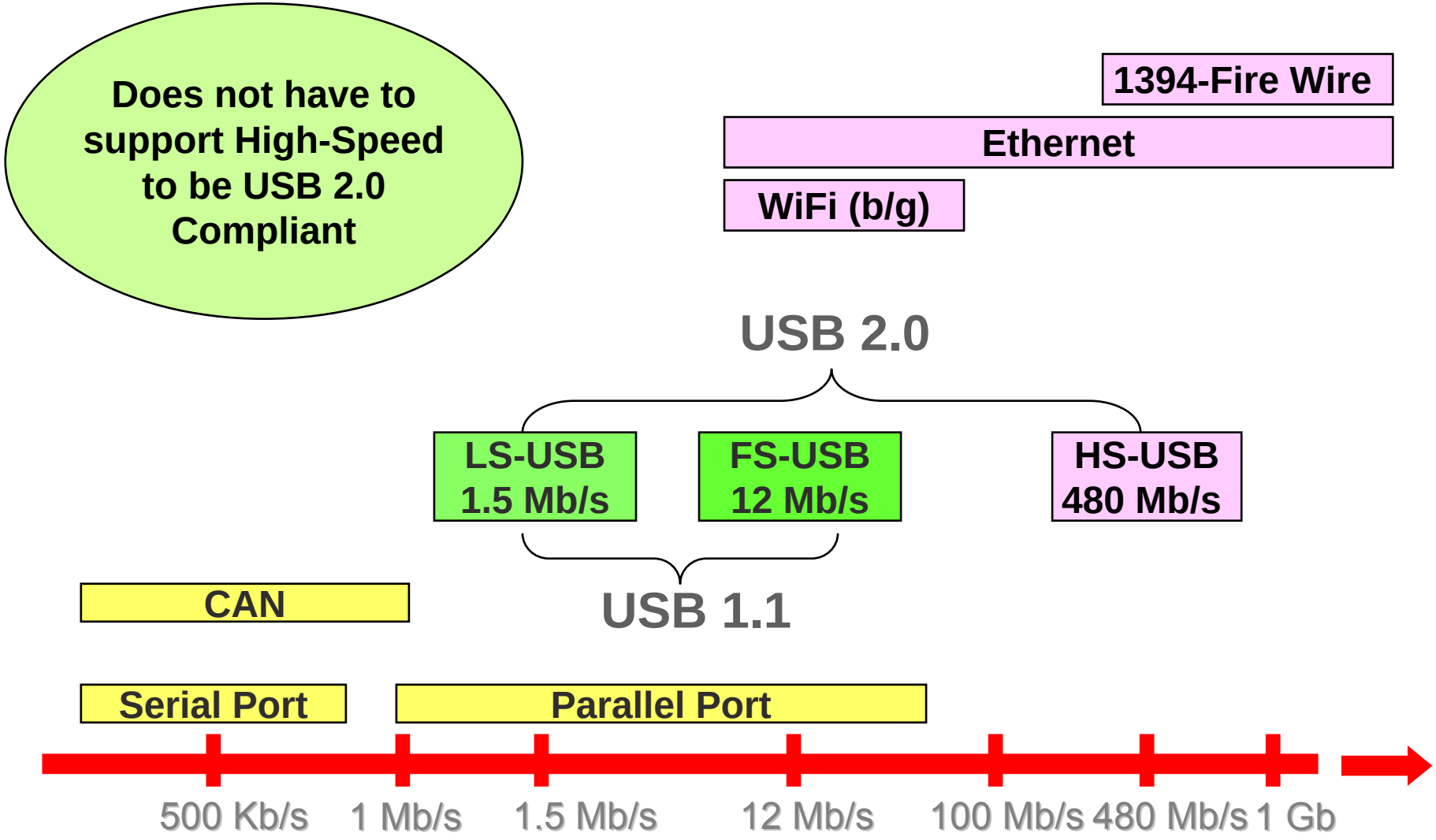
MICROCHIP
Regional Training Centers

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

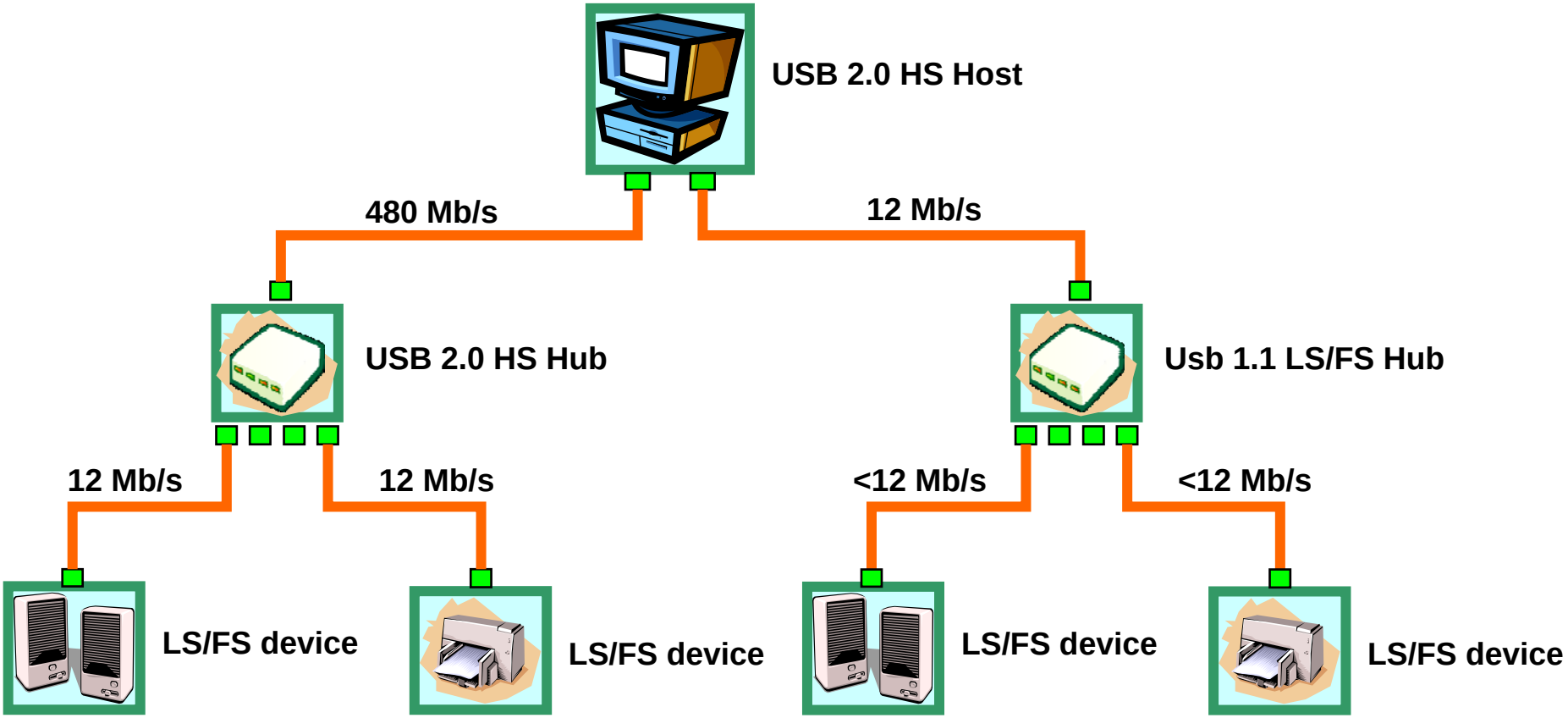
Buses & Speeds Comparison



MICROCHIP
Regional Training Centers

TIP

- **Connect Full-Speed Peripherals to an High-Speed HUB**

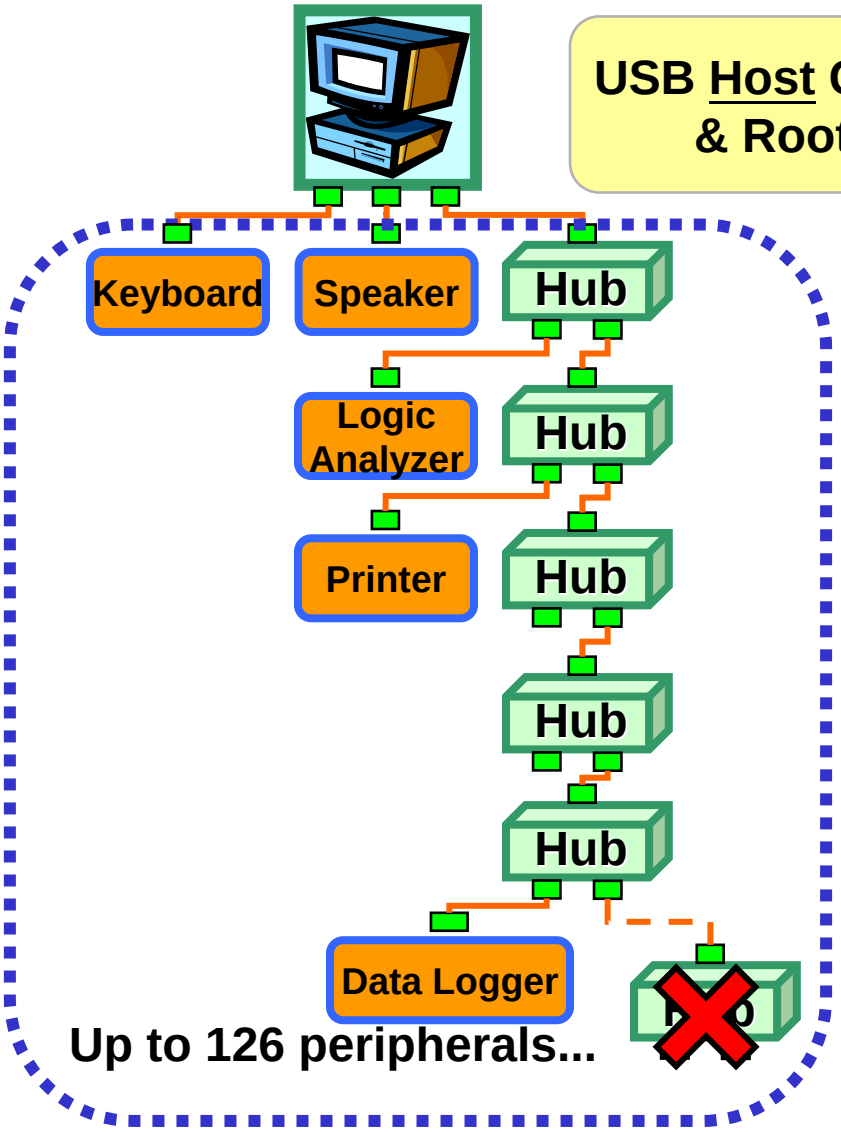


MICROCHIP
Regional Training Centers

Agenda – Part 1

- USB Fundamentals – The serious & important stuff
  - Basics/Speeds
  - Topology/Physical Connection
  - Architecture/Programmer's Model
  - USB Transactions
  - USB Transfers
  - Device Classes
  - Enumeration
  - Descriptors
  - Power Planning
  - VID/PID & USB Compliance
- PIC18/24/32 USB Microcontrollers
- Microchip Demo/Development Solutions

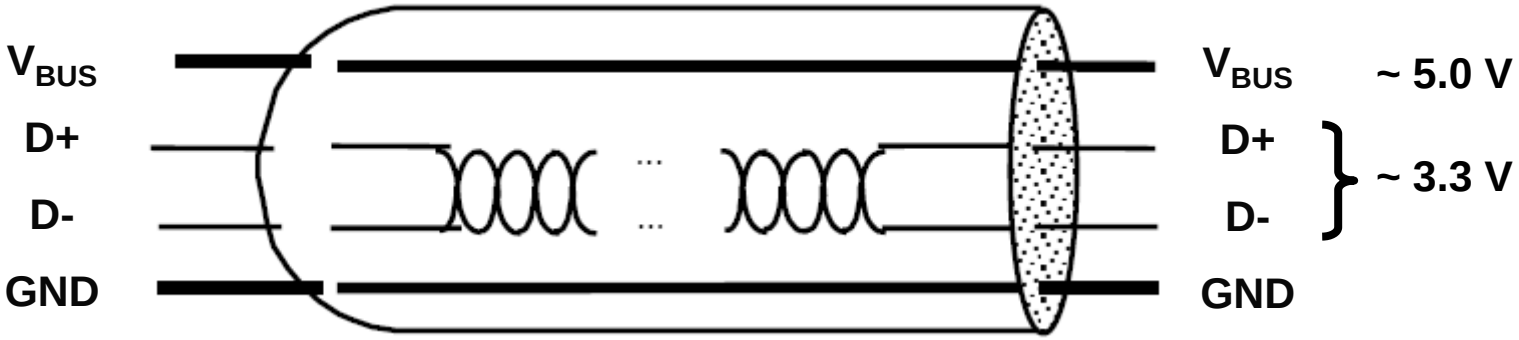
Physical Bus Topology



Hub: Max Chaining = 5

PIC18 USB devices are designed to be peripherals. PIC24/PIC32 can function as either embedded host or peripheral.

Physical Interface



- **Half Duplex with NRZI (No Return to Zero Invert) Data Encoding**
- **Bus Power to each device:**
  - 4.40-5.25V
  - Guaranteed 100 mA
  - 500 mA maximum through negotiation

Must use external power if more is required

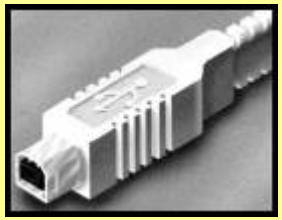
Standard Connectors

- USB 2.0 Specification -

"A"

USB Host

"B"
FS, HS
Device



"mini-B"
FS, HS
Device

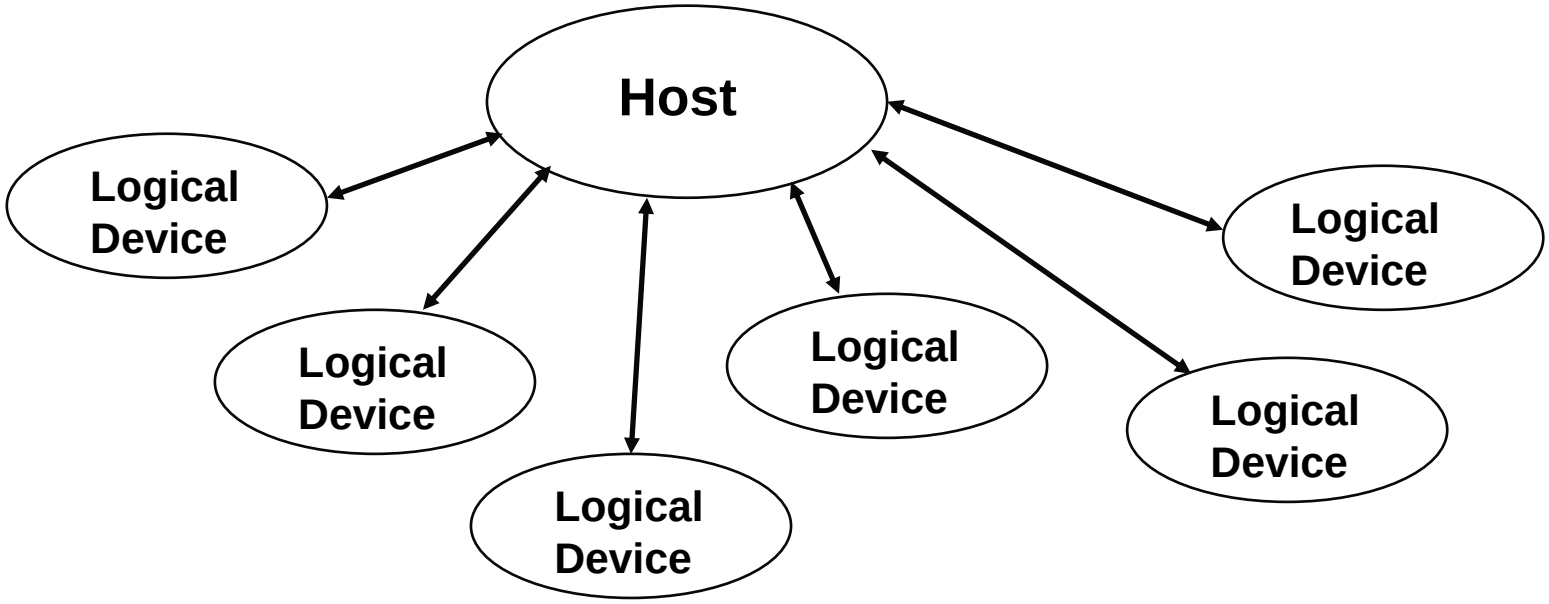
MICROCHIP
Regional Training Centers

Agenda – Part 1

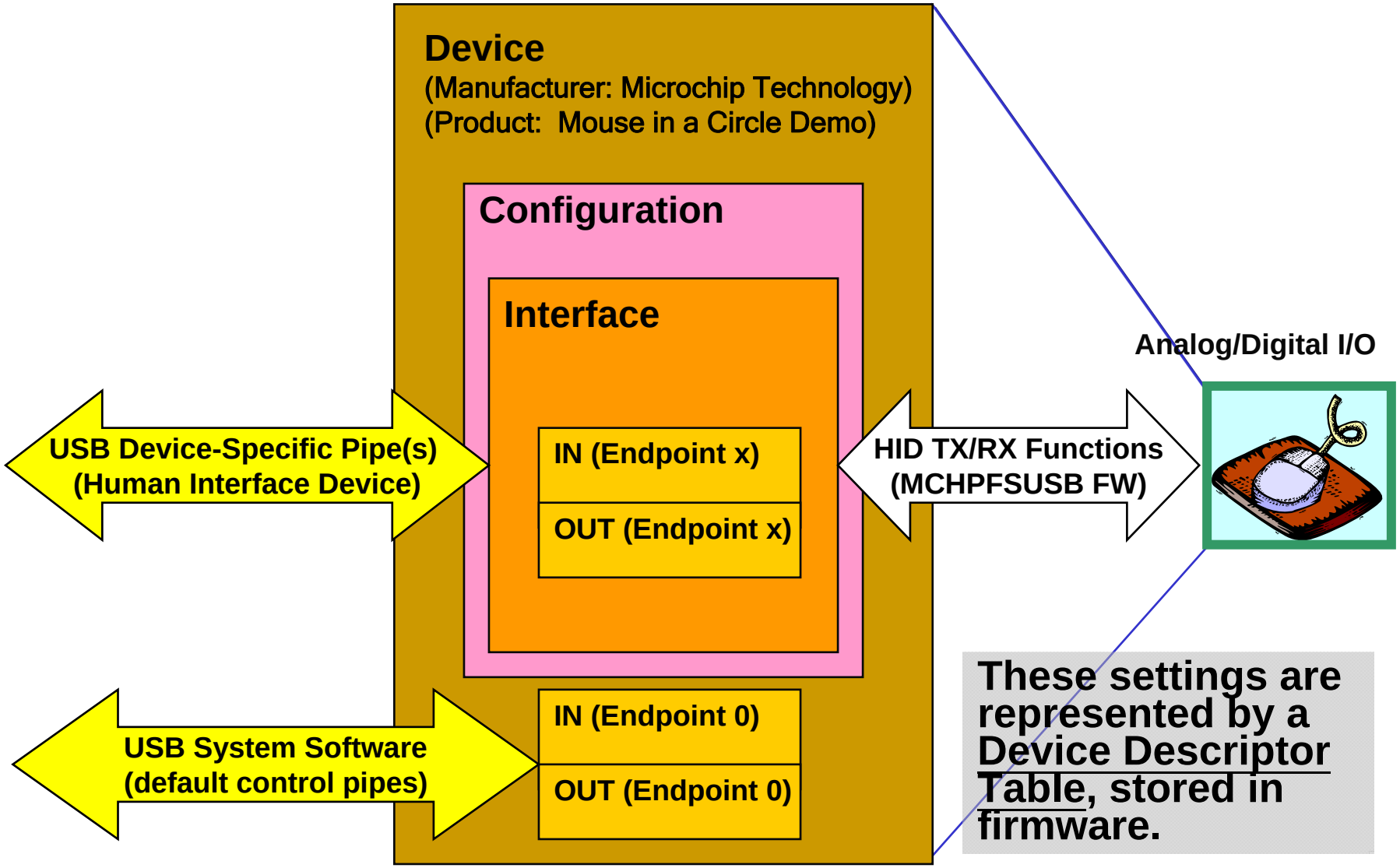
- USB Fundamentals – The serious & important stuff
  - Basics/Speeds
  - Topology/Physical Connection
  - Architecture/Programmer's Model
  - USB Transactions
  - USB Transfers
  - Device Classes
  - Enumeration
  - Descriptors
  - Power Planning
  - VID/PID & USB Compliance
- PIC18/24/32 USB Microcontrollers
- Microchip Demo/Development Solutions

Logical Bus Topology

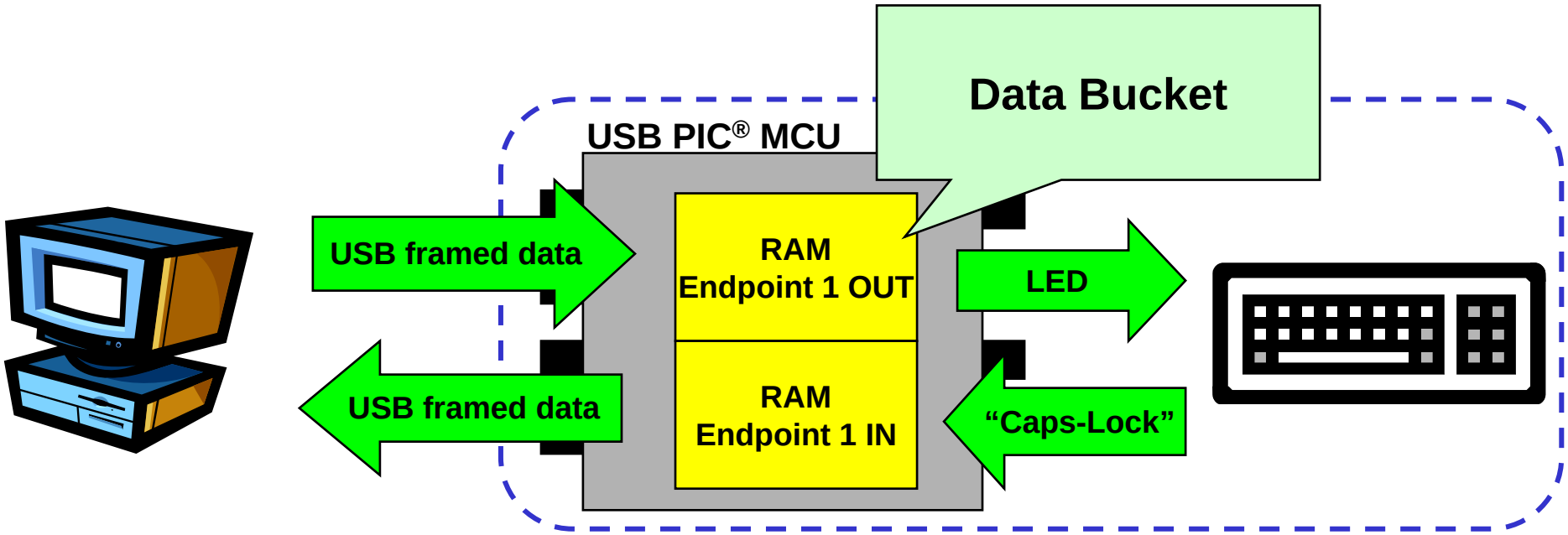
- Not a tiered-star!
- Host software communicates to each “logical” device as if it were directly connected to the root hub



The “Logical” Device



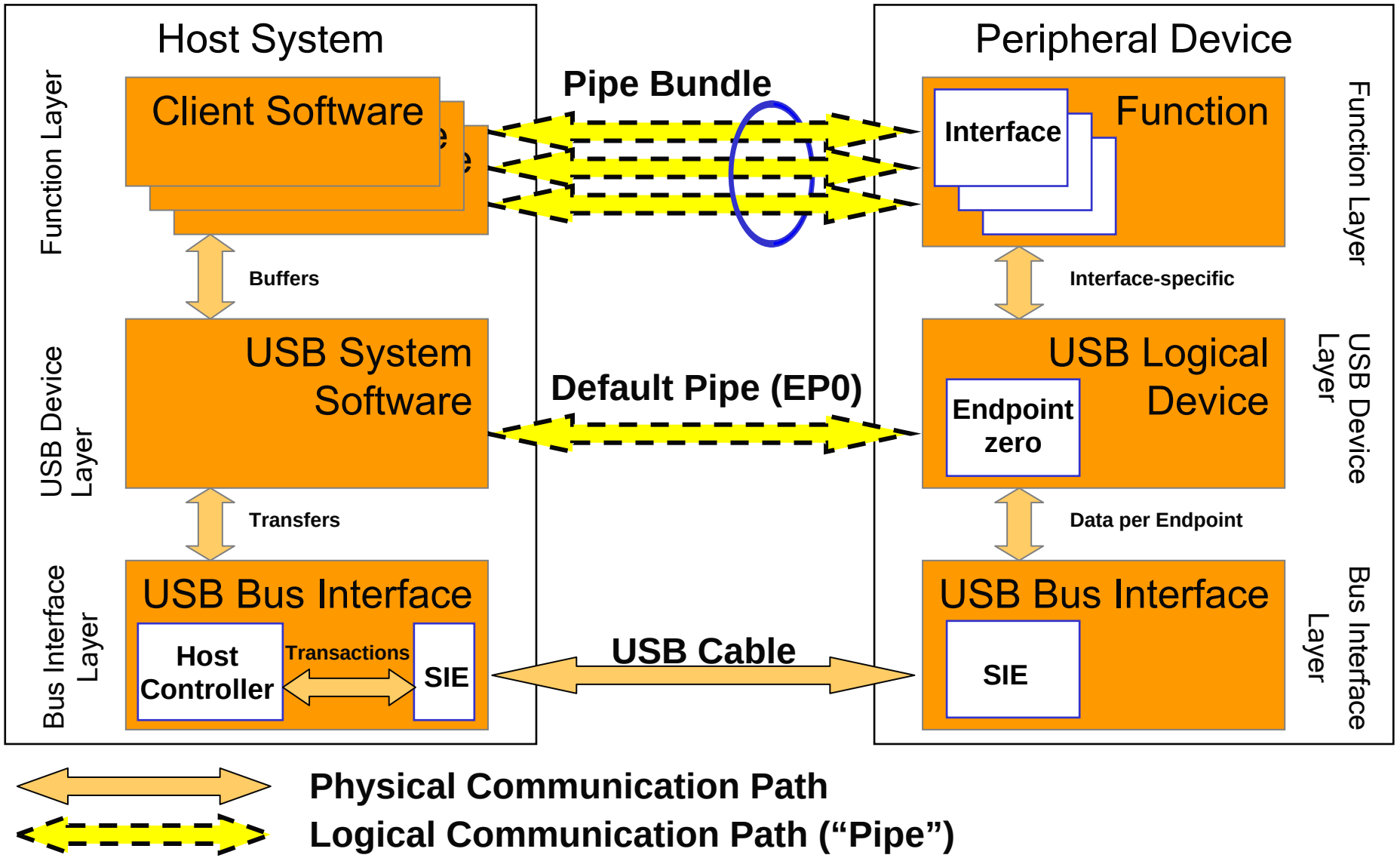
Endpoints: Source/Destination of USB Data in a Peripheral



- Maximum number of endpoints per device specified by USB specification:
  - 16 OUT endpoints + 16 IN endpoints = 32 endpoints
  - PIC18F87J50, PIC18F4550, PIC24F, PIC32MX supports up to 32 endpoints
  - PIC18F14K50 supports up to 16 endpoints
- EP0 = Default Communication Pipe

USB Device Framework

- Software View of Hardware -



MICROCHIP
Regional Training Centers

Sending/Receiving on the PC

- Only High Level Access
- Four Basic Function Types
- Example:
  - Microchip General Purpose USB Device Driver

```
MPUSBOpen(…);
MPUSBRead(…);
MPUSBWrite(…);
MPUSBClose(…);
```

  - Microsoft® WinUSB driver

```
WinUSB_Initialize(…);
WinUsb_ReadPipe(…);
WinUsb_WritePipe(…);
WinUSB_Free(…);
```

Sending/Receiving on the Device

- Only high level access
- Example: CDC Class RS-232 Emulation

```
BOOL USBUSARTIsTxTrfReady(void);
void putUSBUSART(char *data, BYTE Length);
BYTE getsUSBUSART(char *buffer, BYTE len);
```

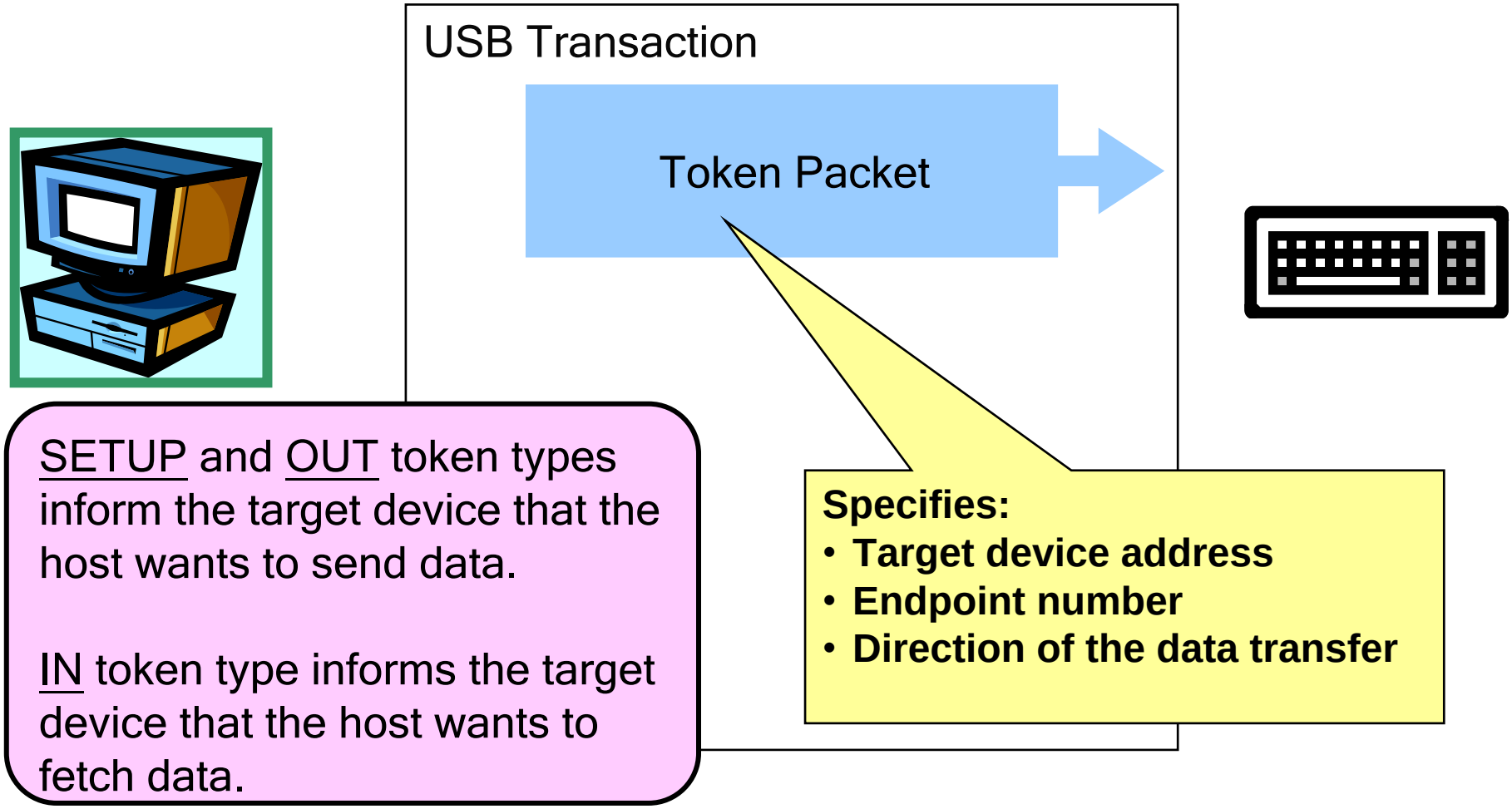
You are not directly reading/writing to the peripheral SFRs anymore!

MICROCHIP
Regional Training Centers

Agenda – Part 1

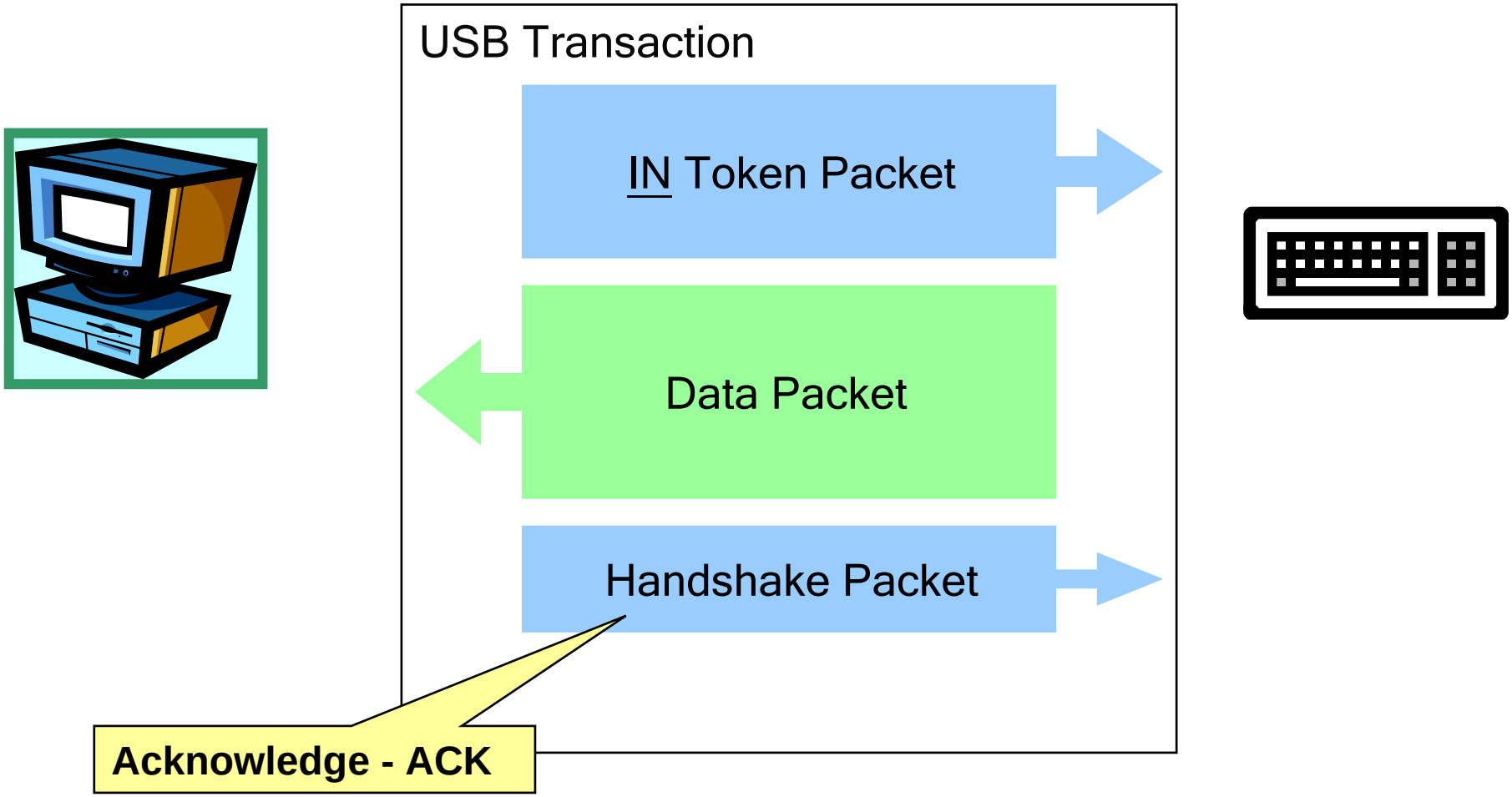
- USB Fundamentals – The serious & important stuff
  - Basics/Speeds
  - Topology/Physical Connection
  - Architecture/Programmer's Model
  - USB Transactions
  - USB Transfers
  - Device Classes
  - Enumeration
  - Descriptors
  - Power Planning
  - VID/PID & USB Compliance
- PIC18/24/32 USB Microcontrollers
- Microchip Demo/Development Solutions

USB Transaction

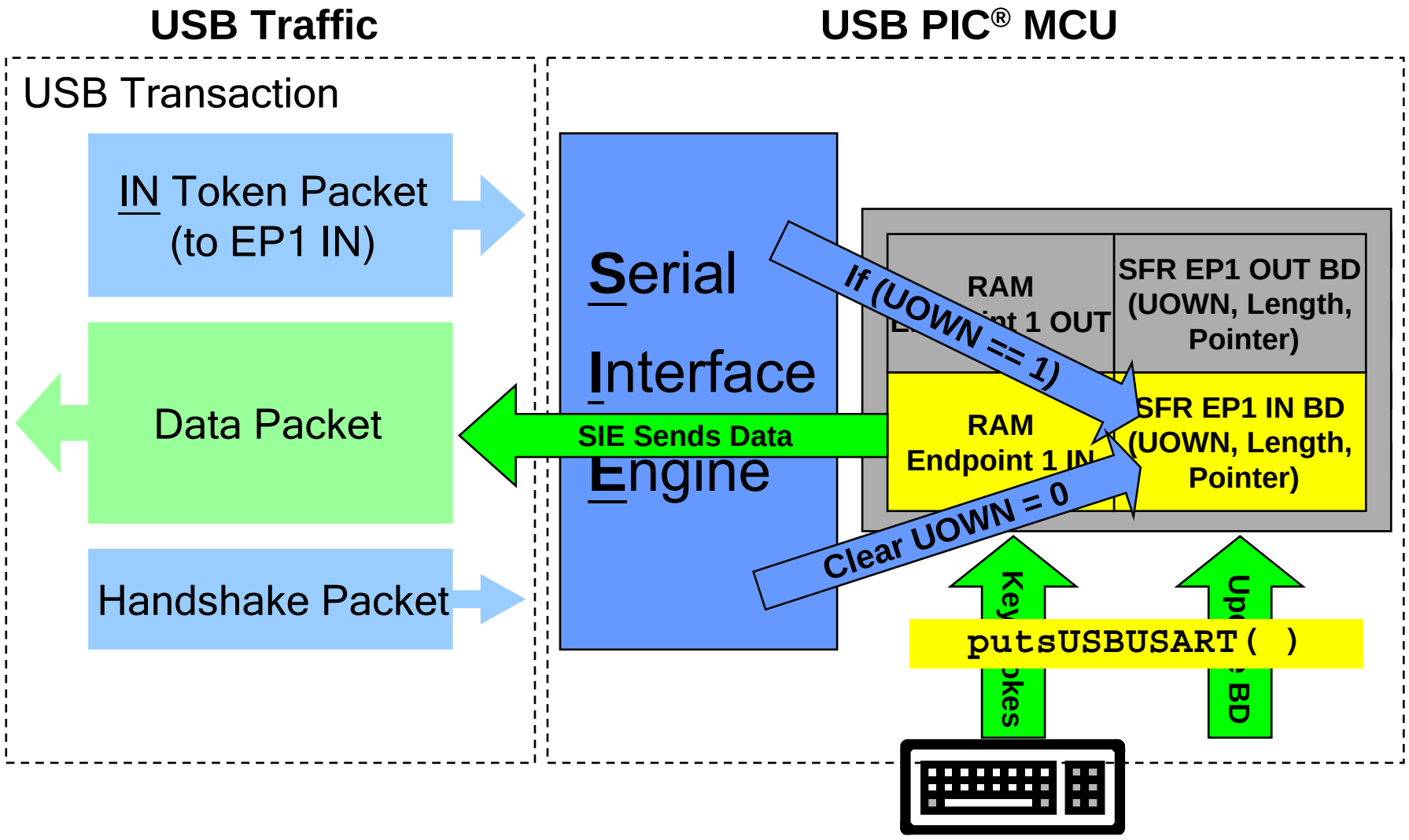


Microchip Regional Training Centers

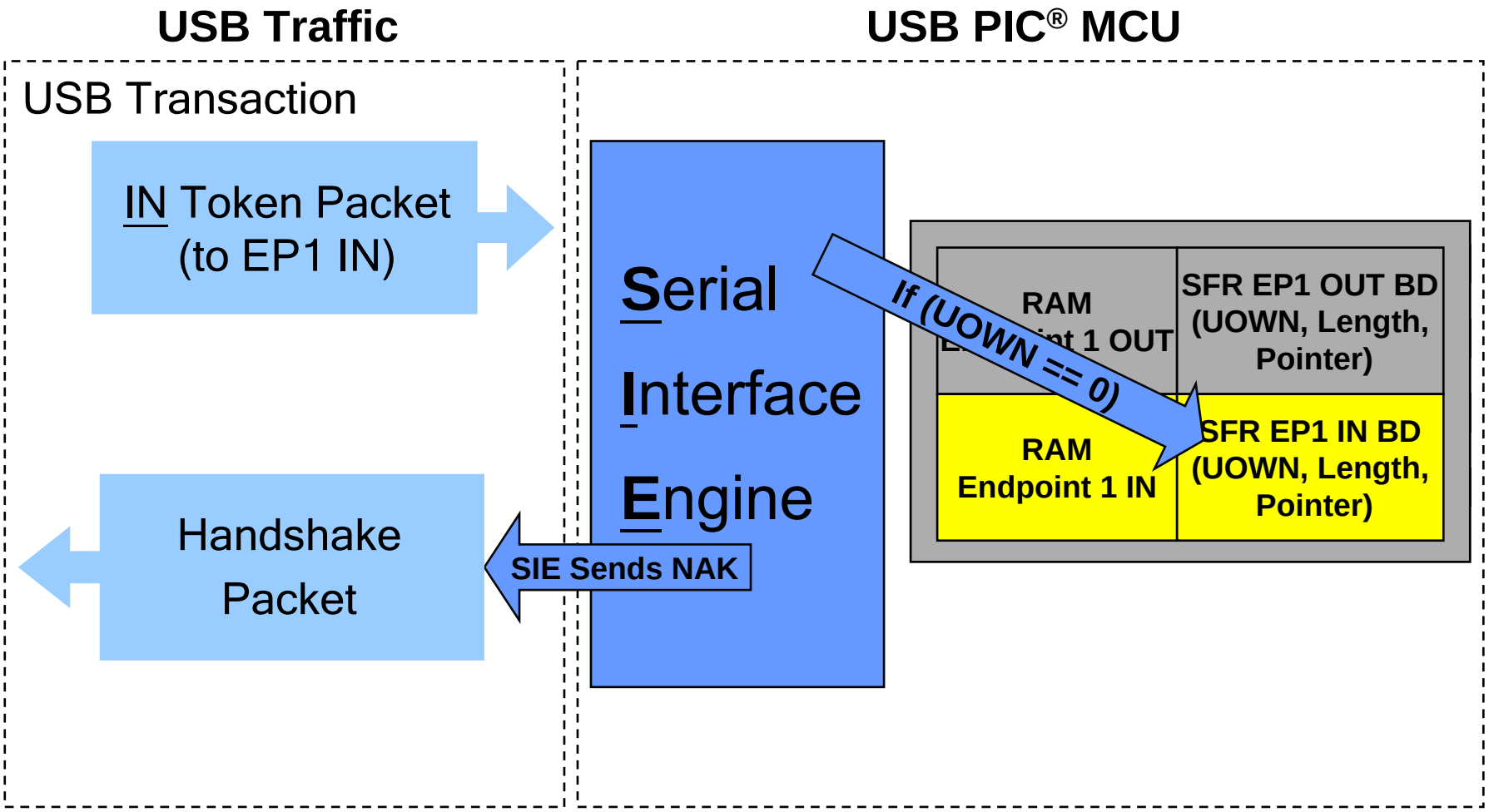
USB Transaction – IN



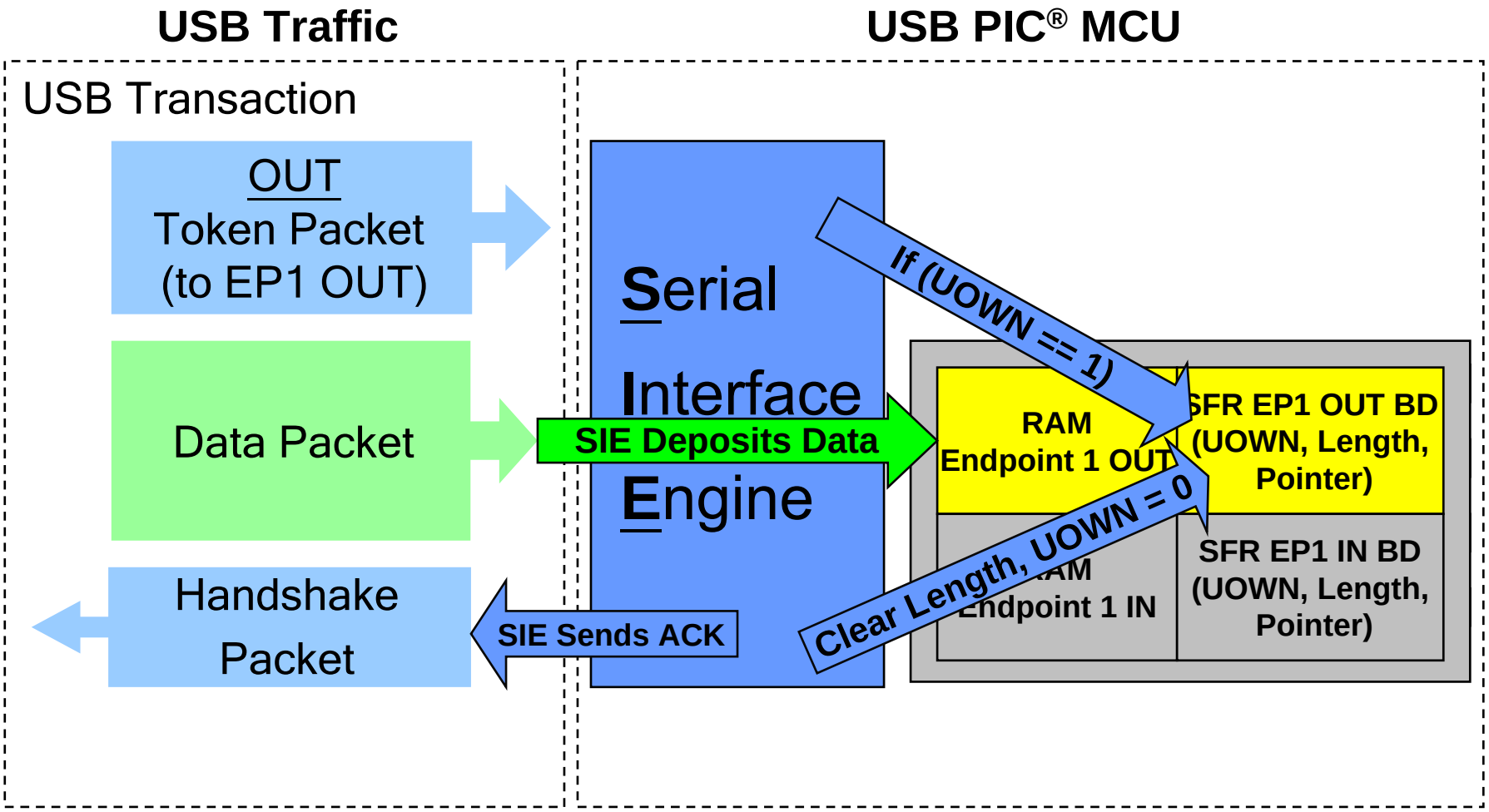
IN Transaction ACK



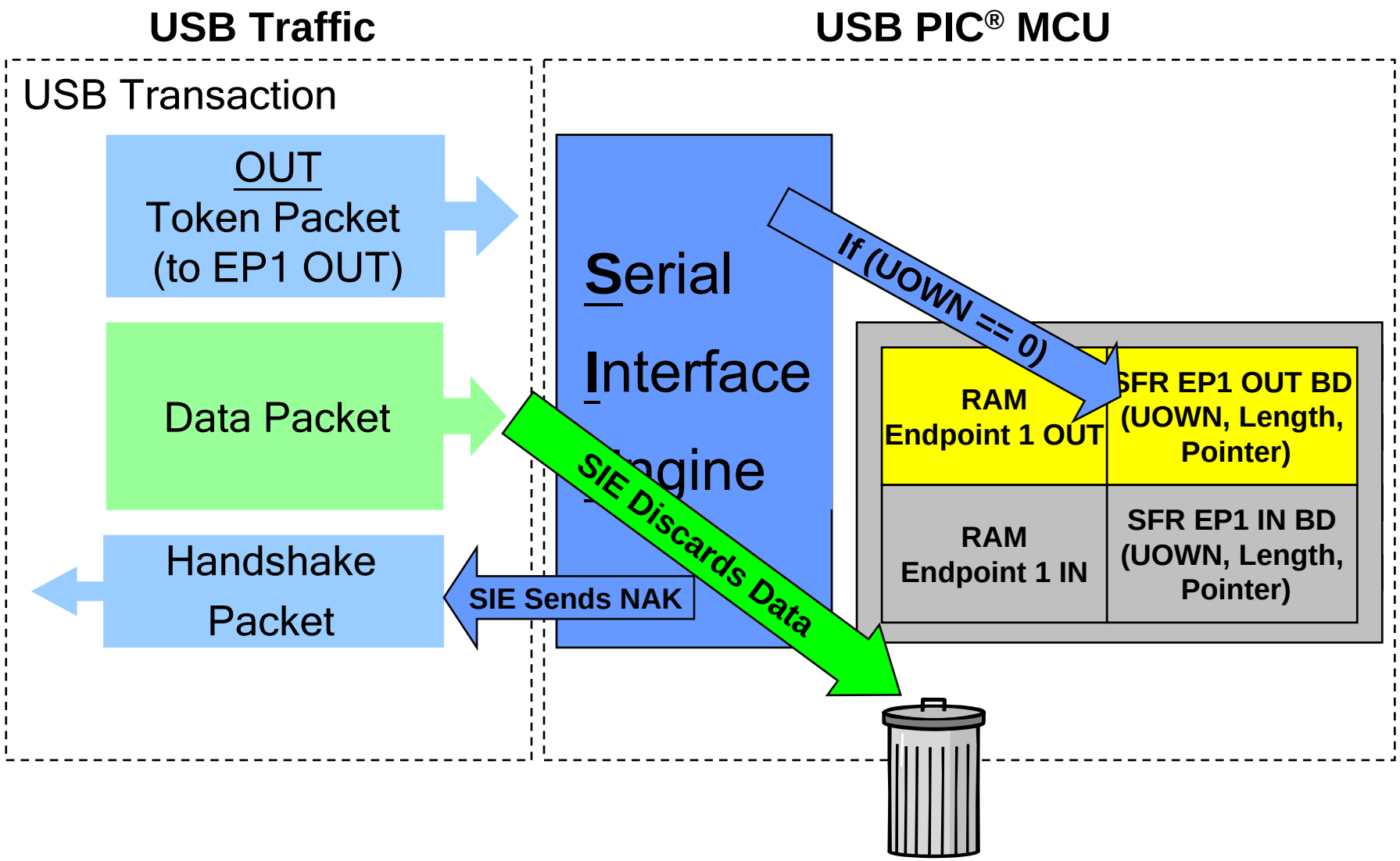
IN Transaction NAK



OUT Transaction ACK



OUT Transaction NAK



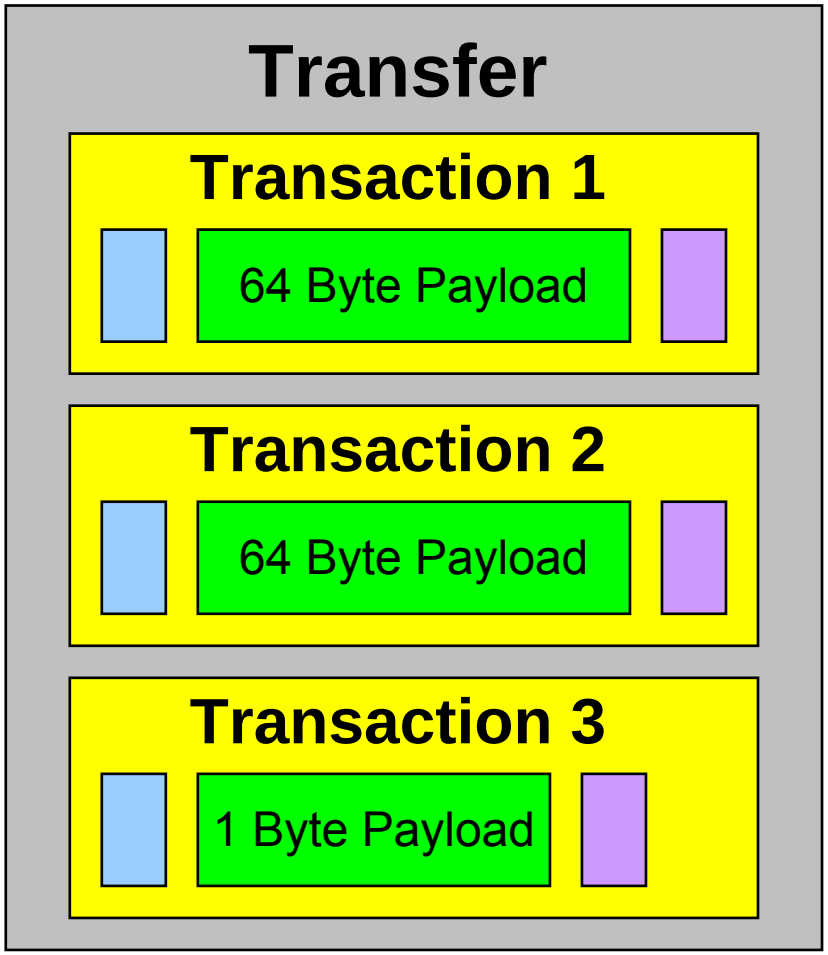
MICROCHIP
Regional Training Centers

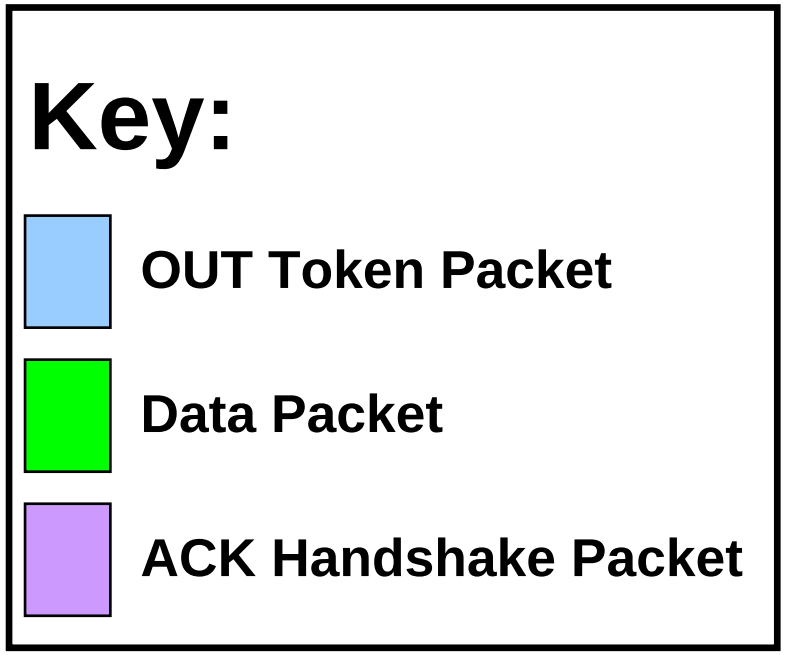
Agenda – Part 1

- USB Fundamentals – The serious & important stuff
  - Basics/Speeds
  - Topology/Physical Connection
  - Architecture/Programmer's Model
  - USB Transactions
  - USB Transfers
  - Device Classes
  - Enumeration
  - Descriptors
  - Power Planning
  - VID/PID & USB Compliance
- PIC18/24/32 USB Microcontrollers
- Microchip Demo/Development Solutions

Transfer: A Group of Related Transactions

```
MPUSBWrite(EP7, Pointer, Size = 129, Timeout)
```





Data Transfer Types
- Full Speed USB -

Control-> Signal and setup

Interrupt -> Guaranteed and timely

Isochronous -> Timely but *may* be corrupted (or missed)

Bulk -> Guaranteed but can be untimely

MICROCHIP
Regional Training Centers

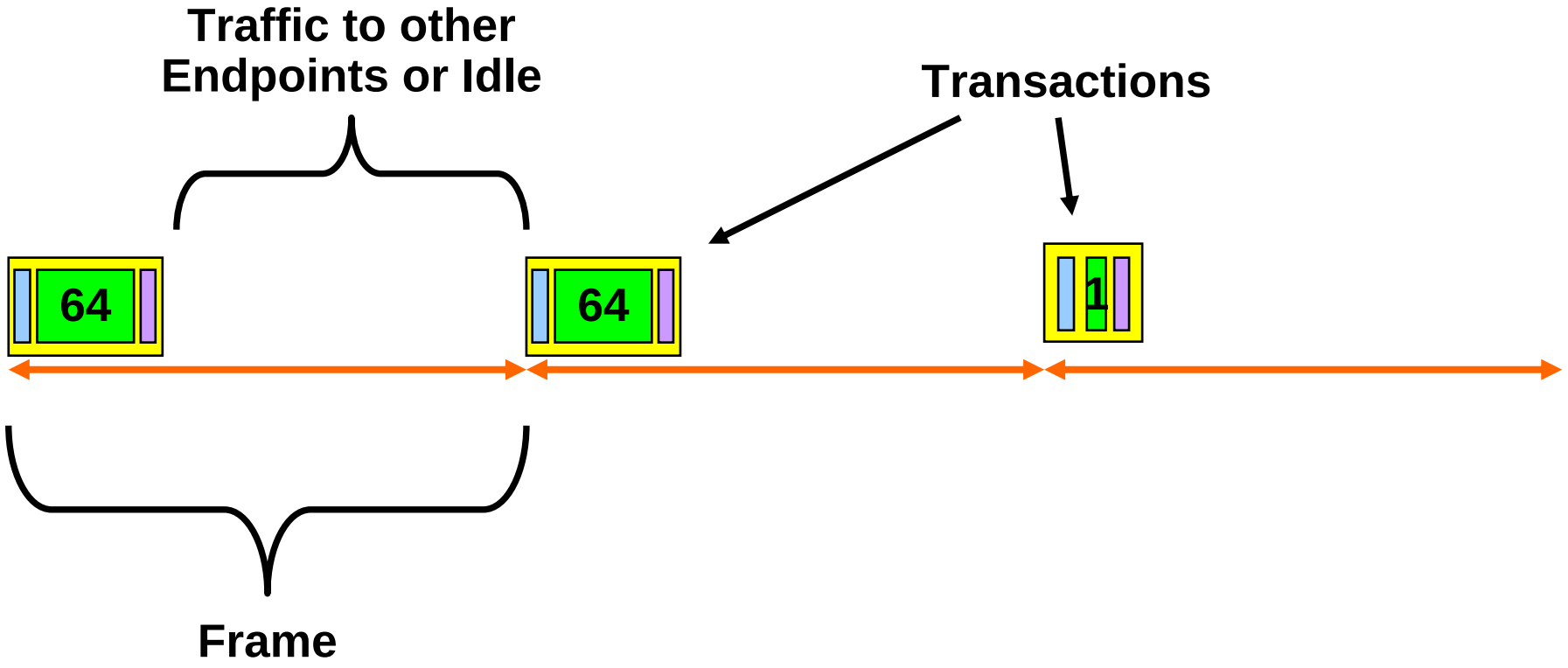
Summary - Data Transfer Types

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

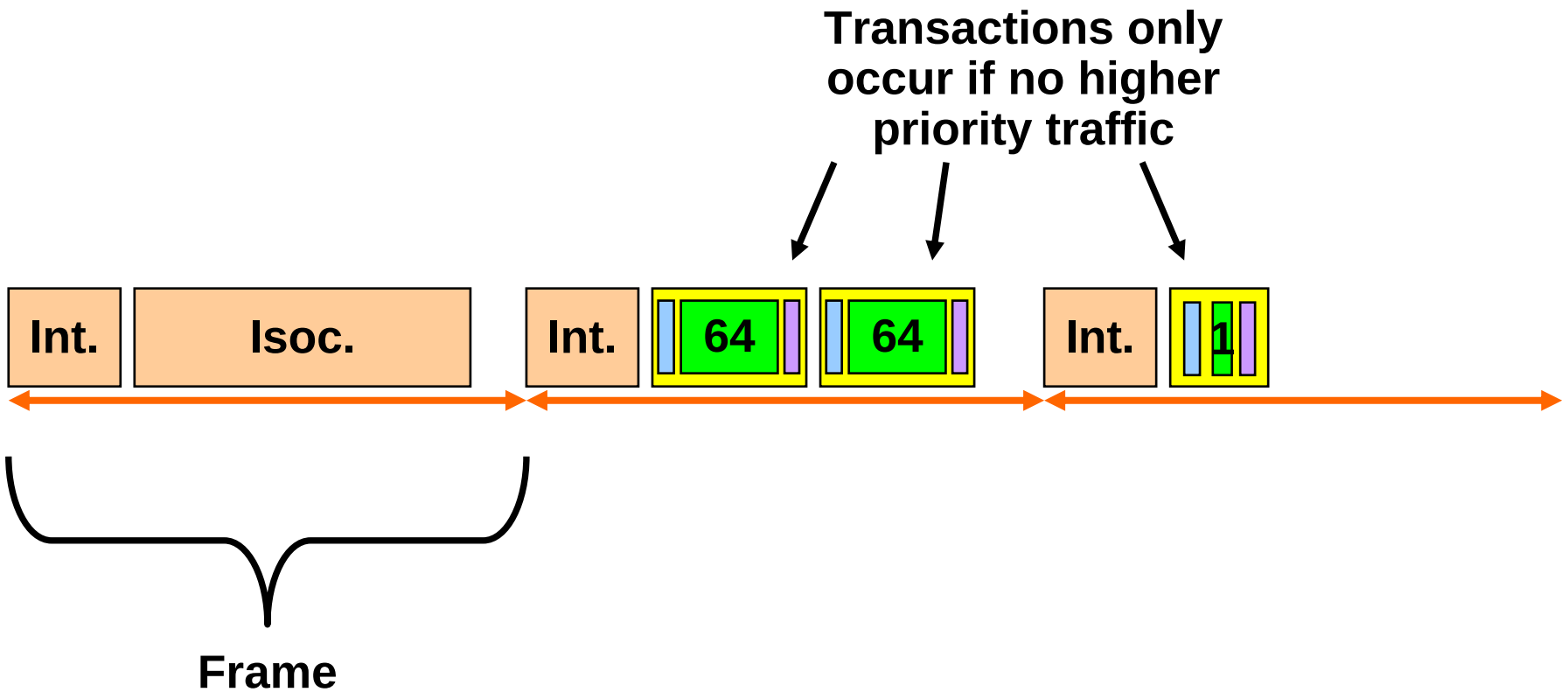
Interrupt Transfer Example

```
MPUSBWrite(EP7, Pointer, Size = 129, Timeout)
```

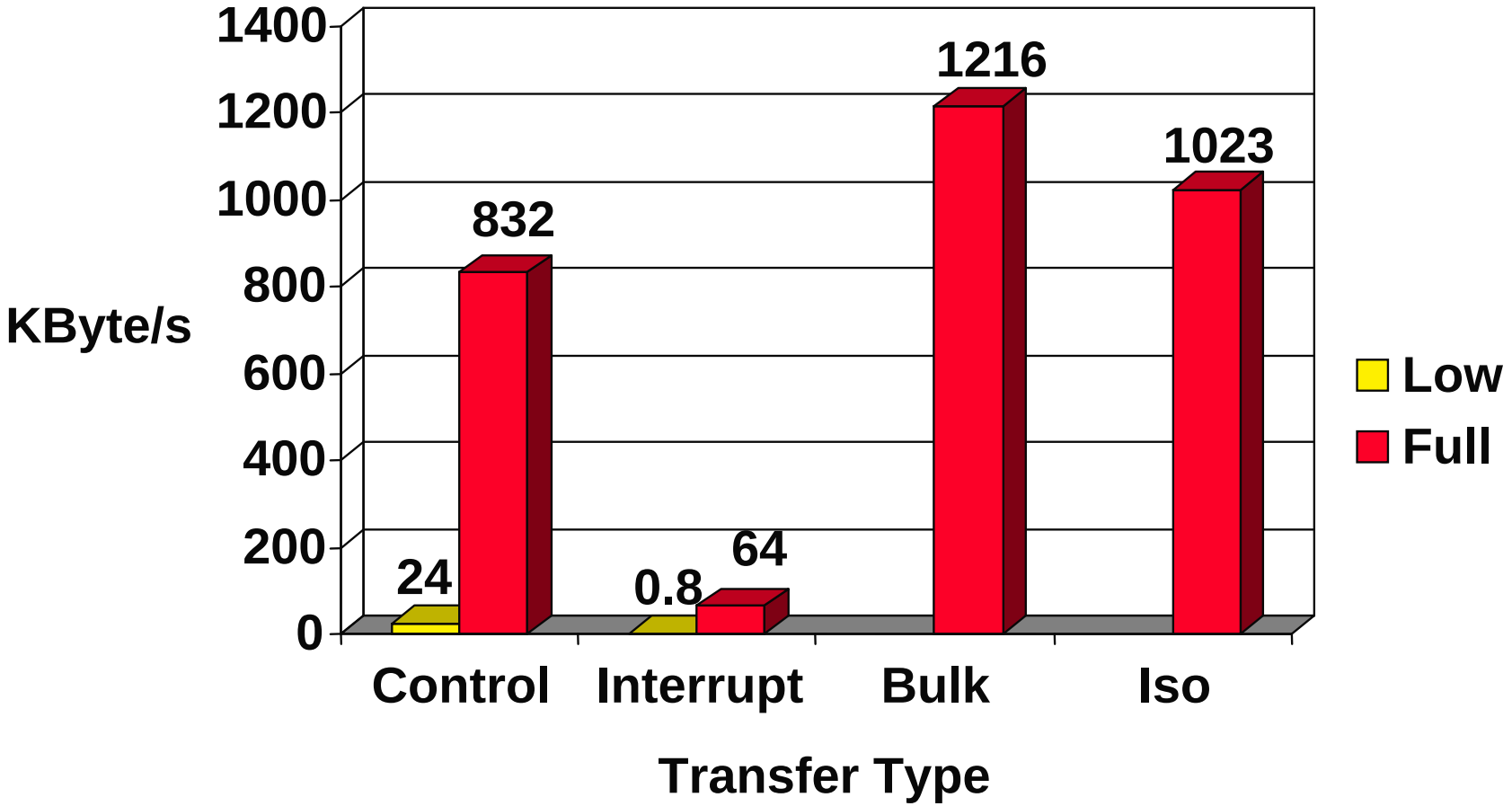


Bulk Transfer Example

```
MPUSBWrite(EP7, Pointer, Size = 129, Timeout)
```

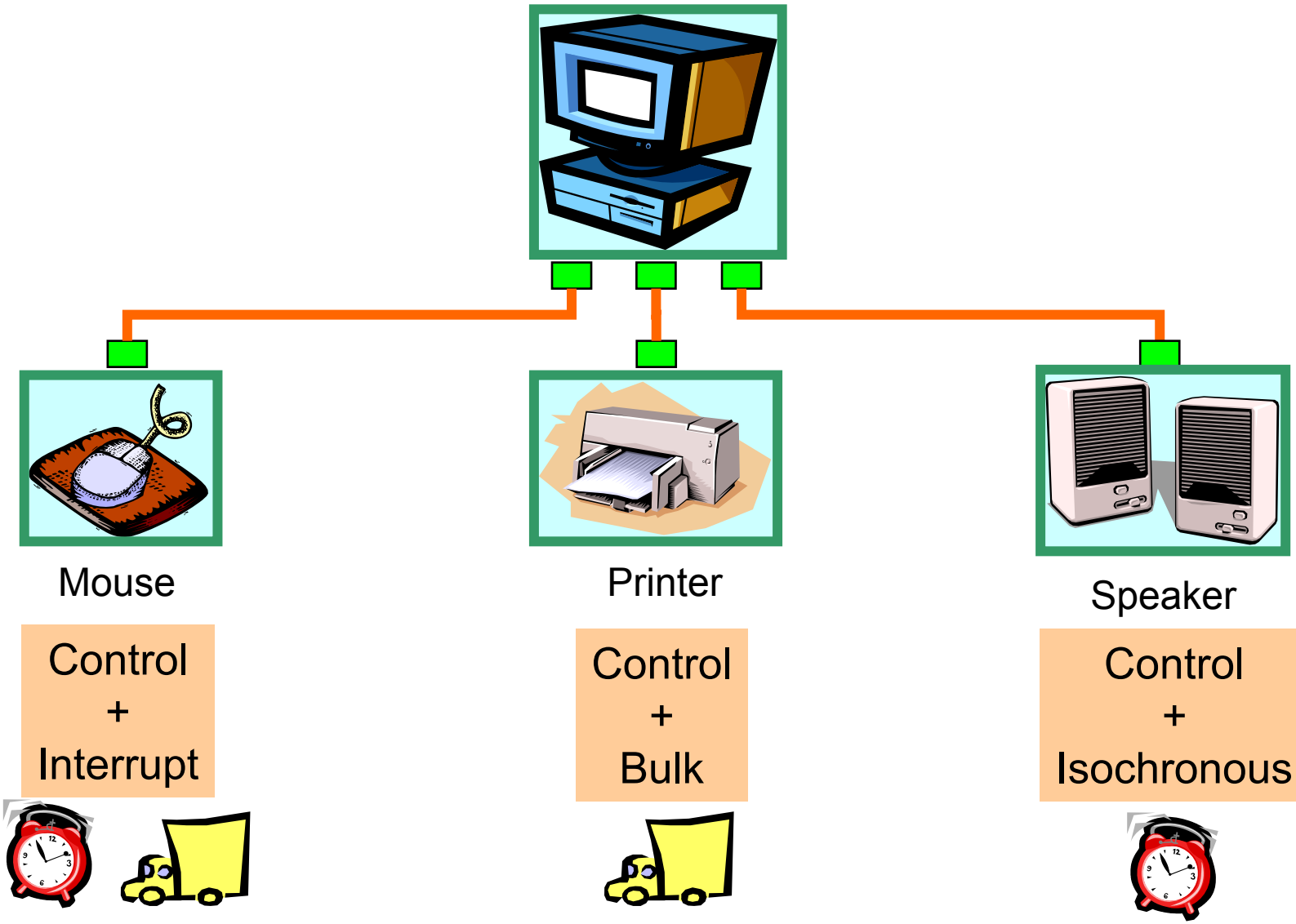


Theoretical Maximum Transfer Rate Per Endpoint



MICROCHIP
Regional Training Centers

Transfer Types – Examples



MICROCHIP
Regional Training Centers

Agenda – Part 1

- USB Fundamentals – The serious & important stuff
  - Basics/Speeds
  - Topology/Physical Connection
  - Architecture/Programmer's Model
  - USB Transactions
  - USB Transfers
  - Device Classes
  - Enumeration
  - Descriptors
  - Power Planning
  - VID/PID & USB Compliance
- PIC18/24/32 USB Microcontrollers
- Microchip Demo/Development Solutions

MICROCHIP
Regional Training Centers

USB Device Classes

Data Glove
Mouse
PICkit™ 3 Starter Kit
Joystick
Keyboard
Human Interface Device Class (HID)

Floppy Drive
External Hard Drive
Mass Storage Device Class (MSD)

Ethernet Adapter
Modem
Communication Device Class (CDC)

MPLAB® REAL ICE™ in-circuit emulator
Custom Class (Vendor Class)

Many more classes….

MICROCHIP
Regional Training Centers

USB Driver Choices

- Windows® PC Host -

Features	HID	CDC	MCHPUSB	WinUSB	LibUSB
Driver support built into Windows	Yes	Need .inf	No	Need .inf	No
64-bit PC Support	Yes	Yes	Yes	Yes	Yes
XP Ready	Yes	Yes	Yes	Yes	Yes
Vista Ready	Yes	Yes	Yes	Yes	32 bit
Transfer Types for user's data					
Control	No	No	Yes	Yes	Yes
Interrupt	Yes	No	Yes	Yes	Yes
Isochronous	No	No	Yes	No	Yes
Bulk	No	Yes	Yes	Yes	Yes
Max Speed	64 KB/s	~80 kB/s	~1.0 MB/s	~1.0 MB/s	~1.0 MB/s

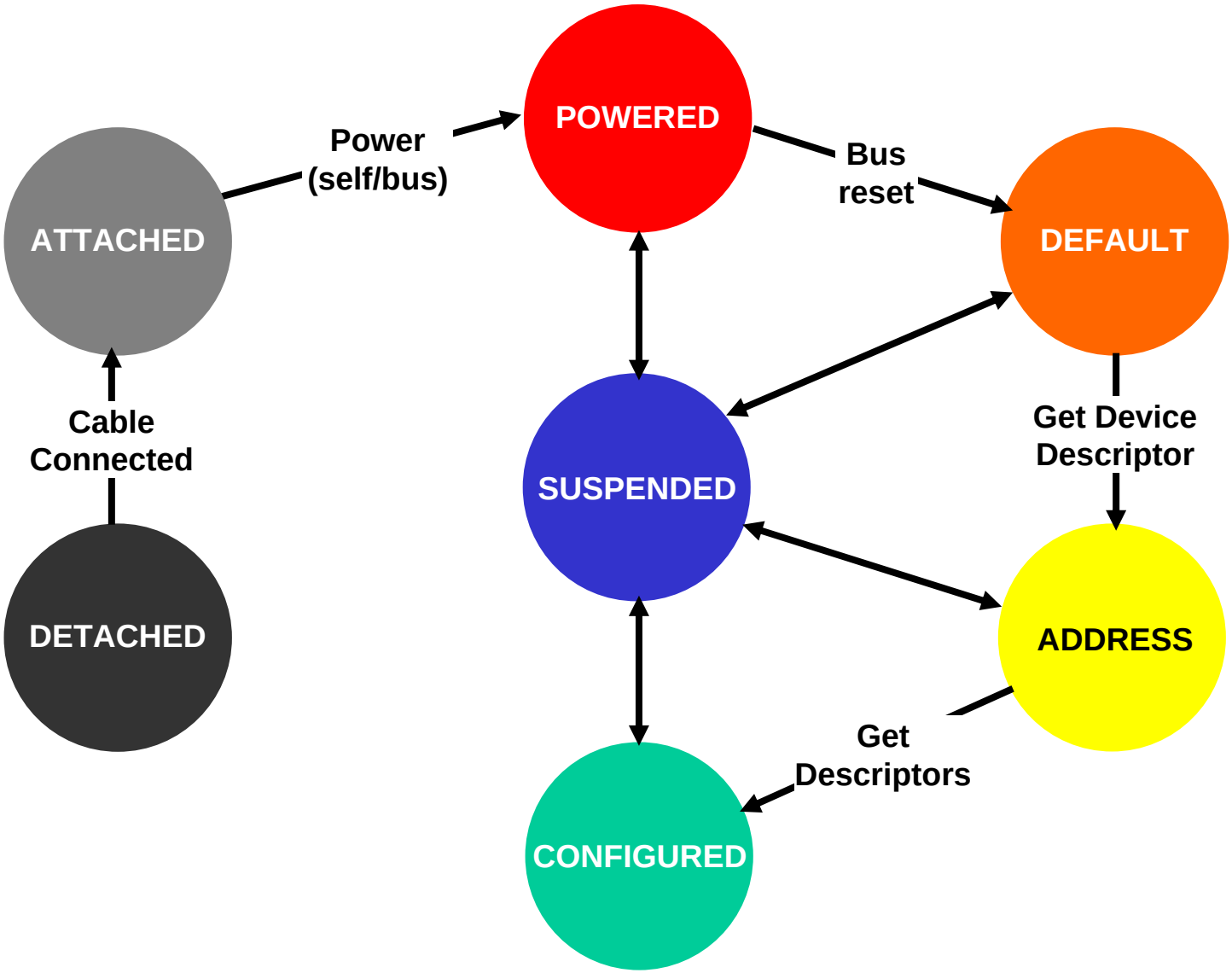
MICROCHIP
Regional Training Centers

Agenda – Part 1

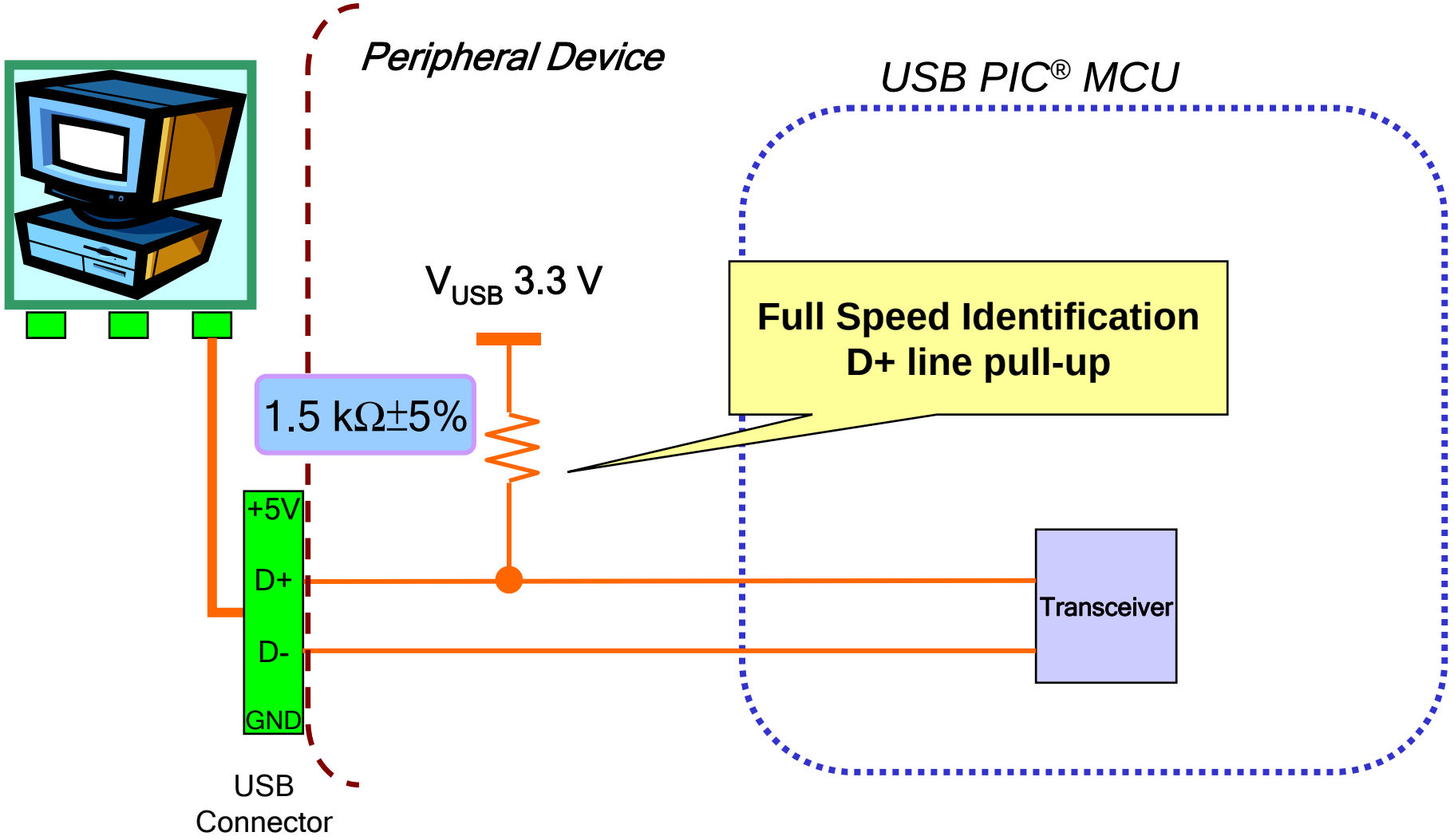
- USB Fundamentals – The serious & important stuff
  - Basics/Speeds
  - Topology/Physical Connection
  - Architecture/Programmer's Model
  - USB Transactions
  - USB Transfers
  - Device Classes
  - Enumeration
  - Descriptors
  - Power Planning
  - VID/PID & USB Compliance
- PIC18/24/32 USB Microcontrollers
- Microchip Demo/Development Solutions

MICROCHIP
Regional Training Centers

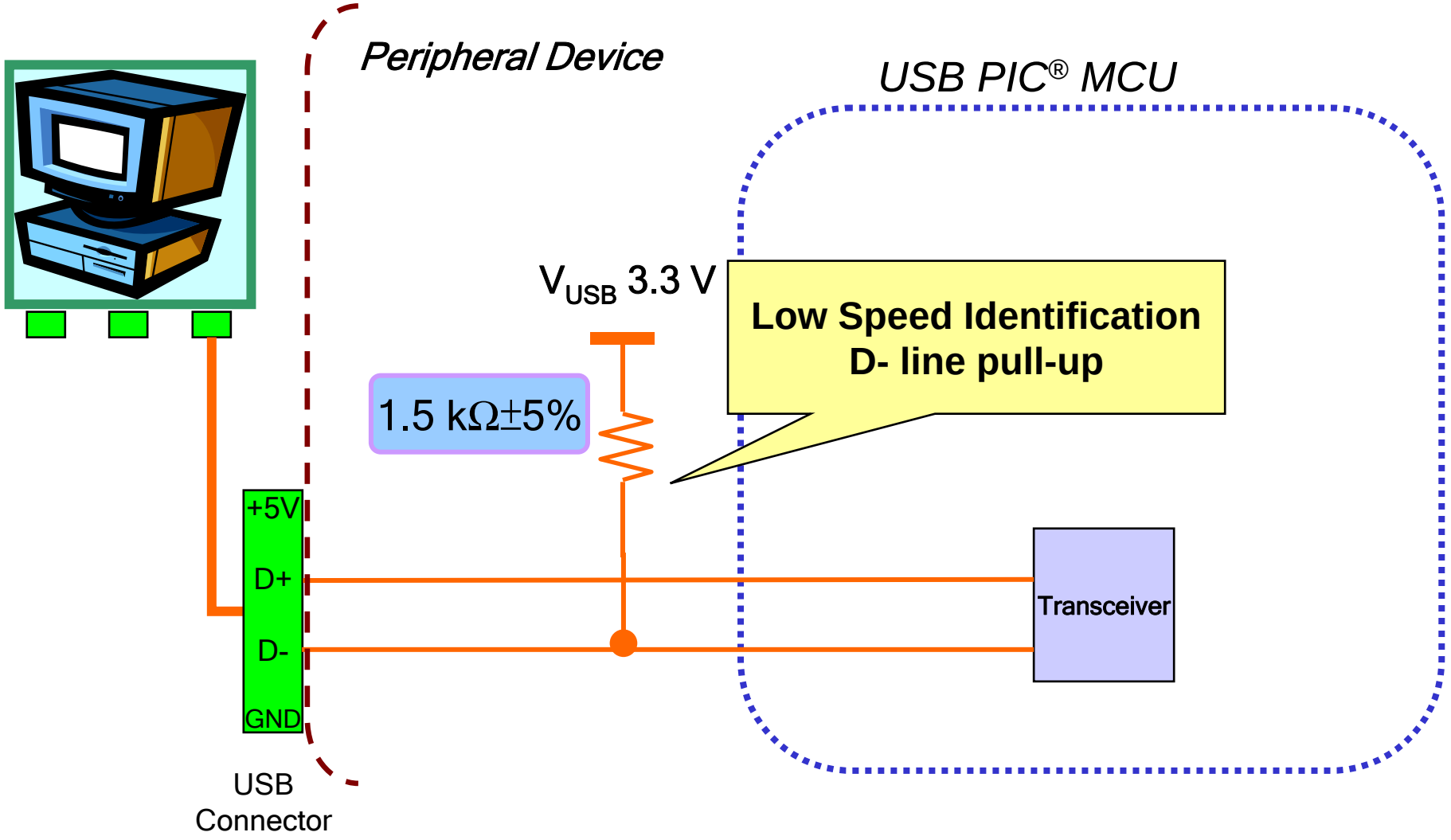
The Enumeration Process



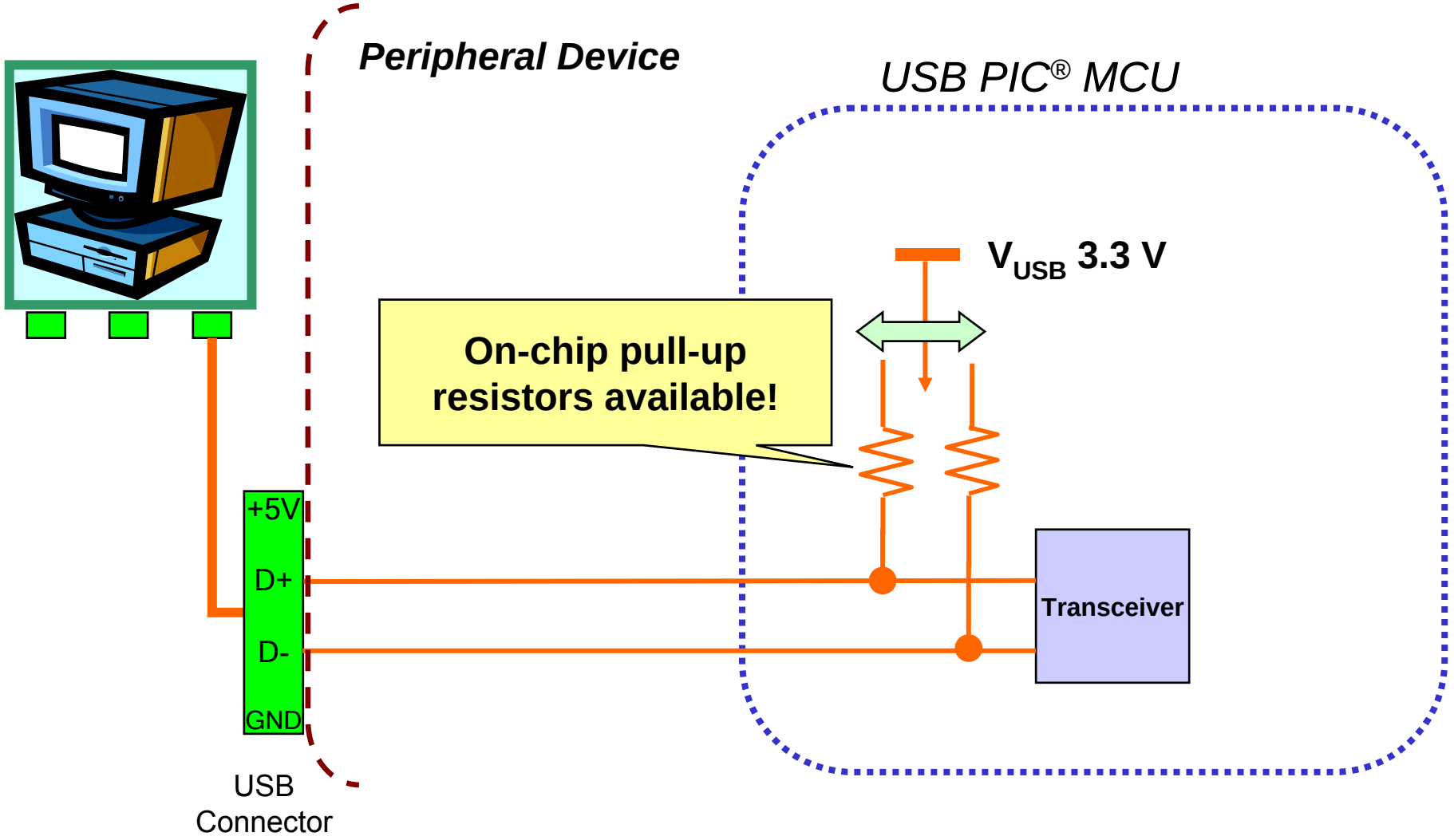
Auto-Detection: Full-Speed



Auto-Detection: Low-Speed

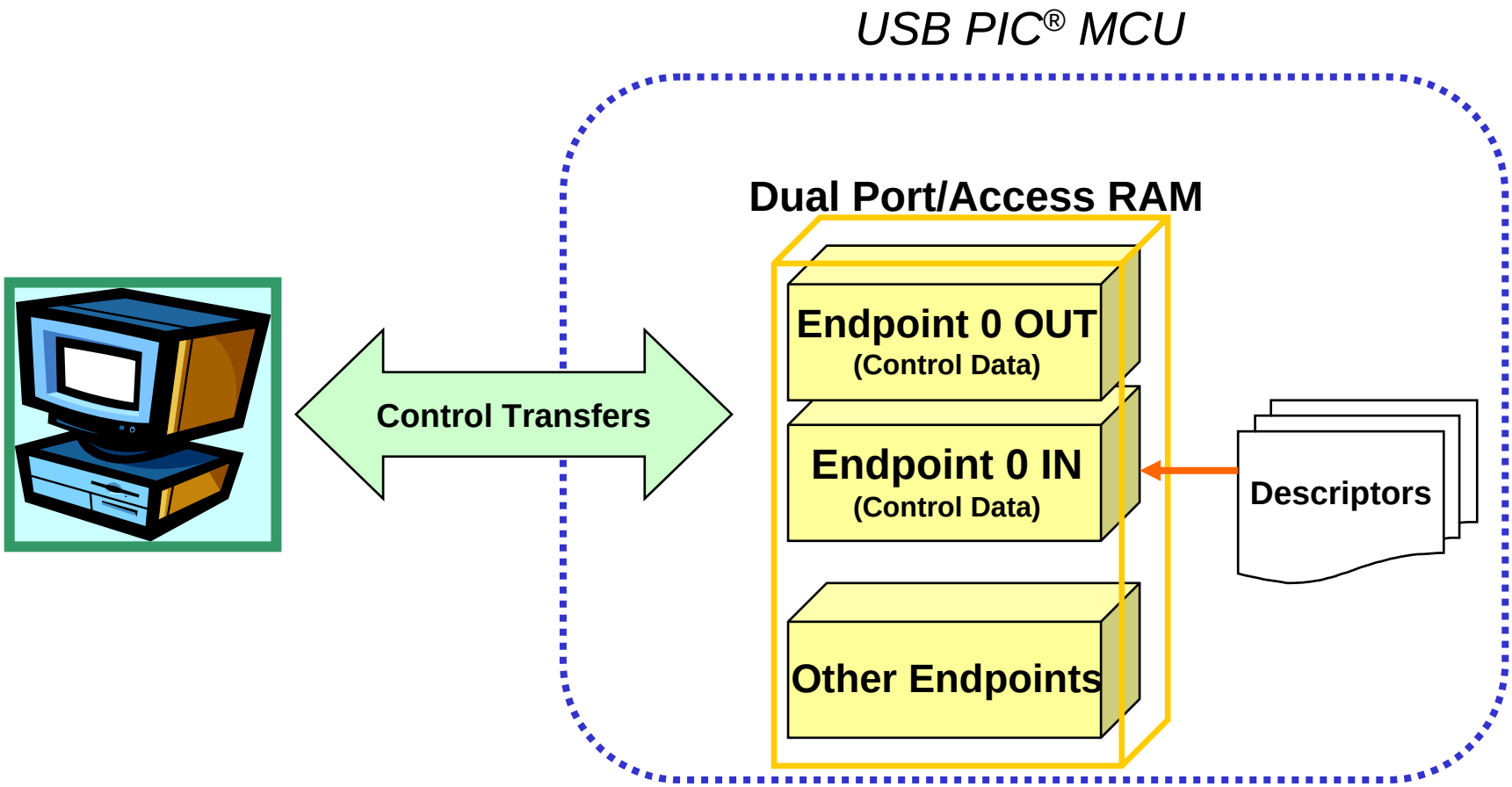


On-chip Pull-up Resistors



Address and Configuration: EP0

- See Chapter 9 in USB 2.0 Spec for more info.

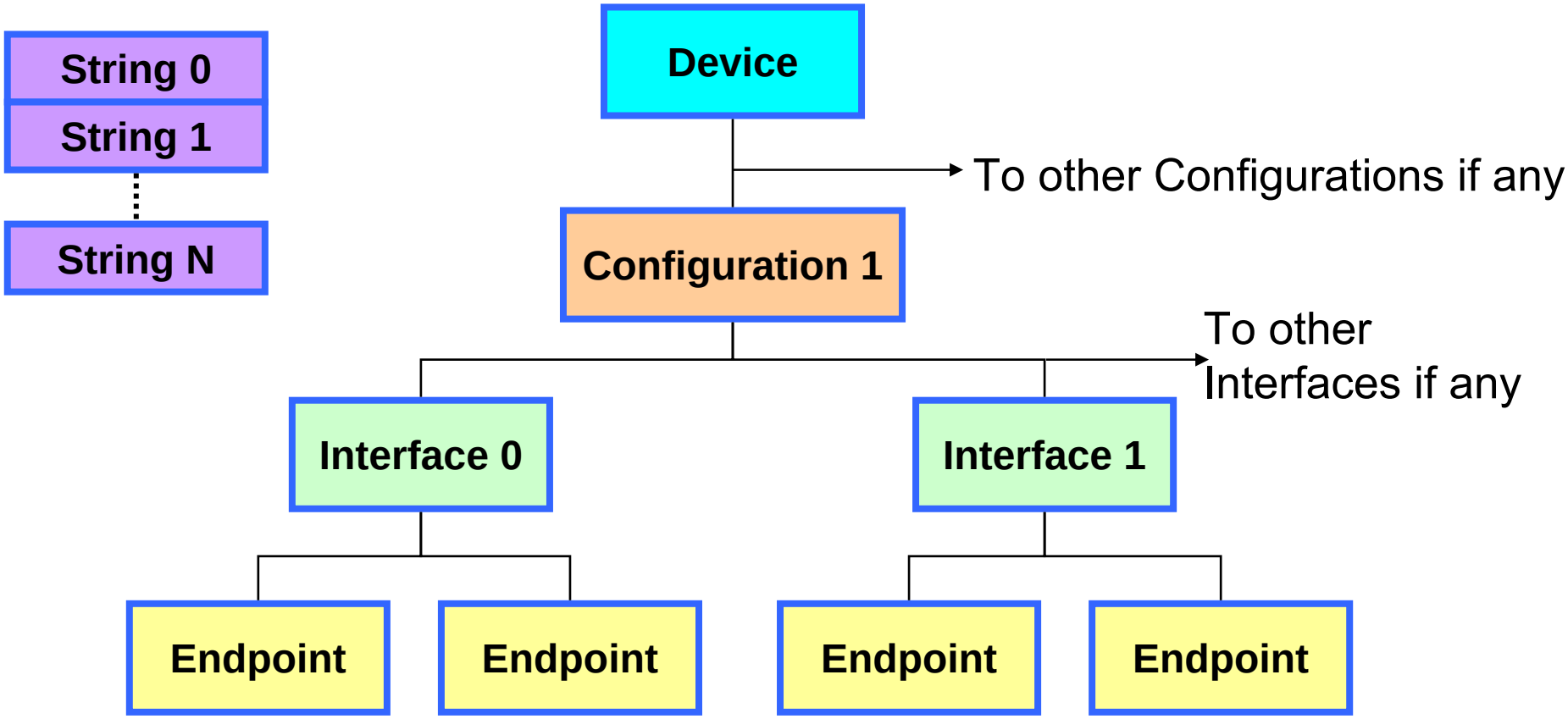


MICROCHIP
Regional Training Centers

Agenda – Part 1

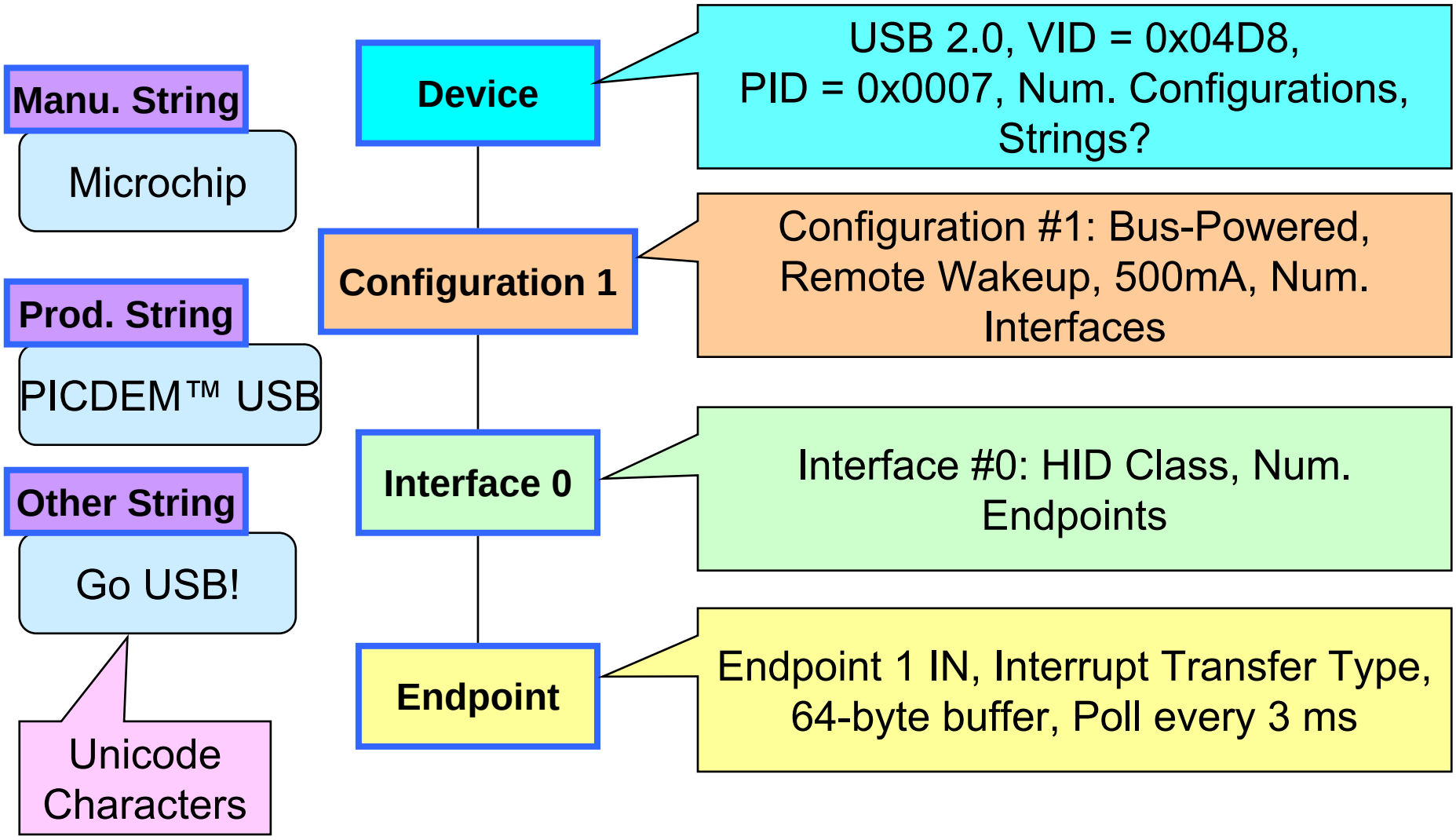
- USB Fundamentals – The serious & important stuff
  - Basics/Speeds
  - Topology/Physical Connection
  - Architecture/Programmer's Model
  - USB Transactions
  - USB Transfers
  - Device Classes
  - Enumeration
  - Descriptors
  - Power Planning
  - VID/PID & USB Compliance
- PIC18/24/32 USB Microcontrollers
- Microchip Demo/Development Solutions

Descriptors



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

Descriptors - Example



MCHPFSUSB Software Framework

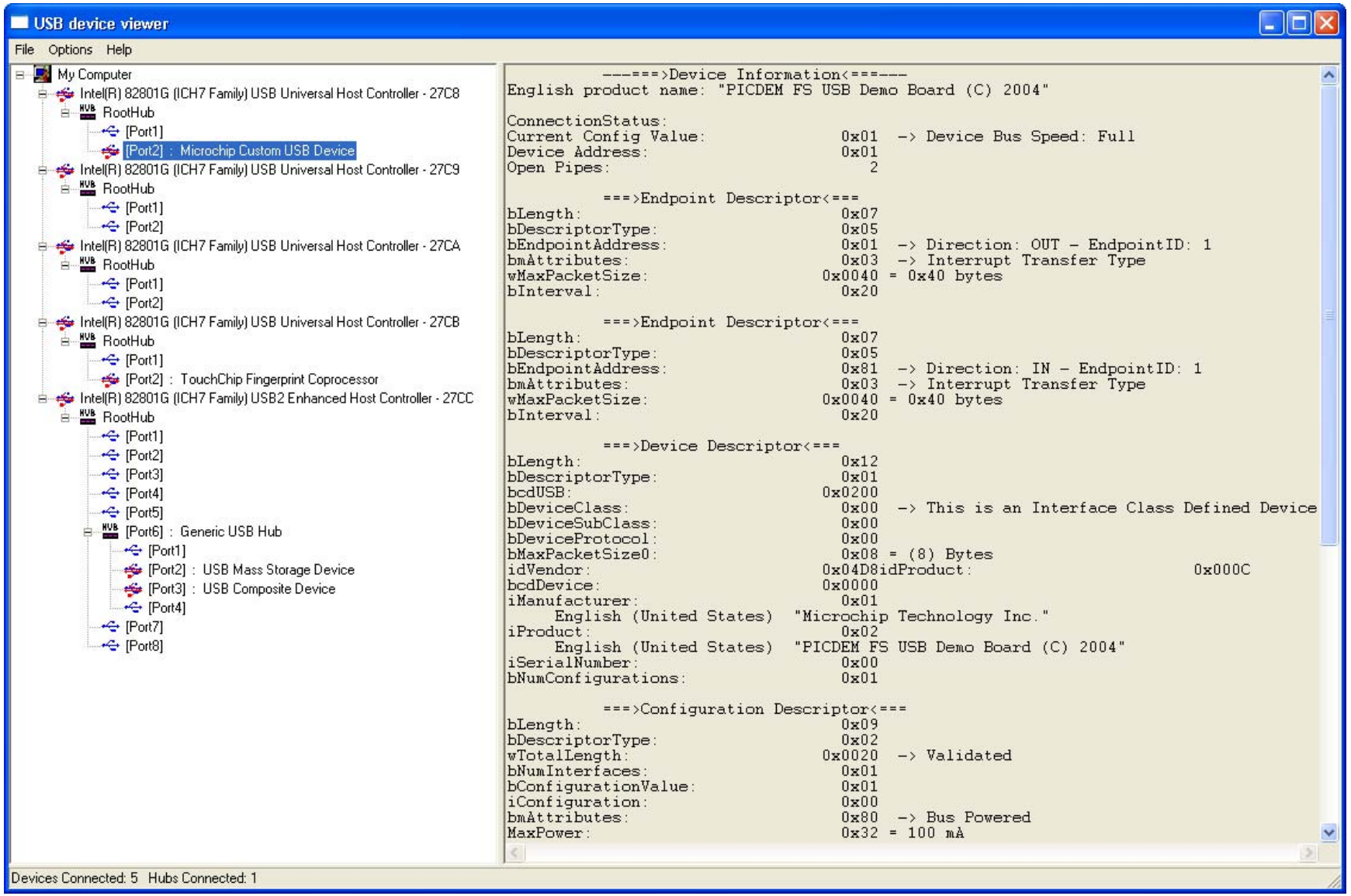
- Device Descriptor Table -

- usb_descriptors.c
  - Descriptors
    - VID & PID
    - Class Specific

```
/* Device Descriptor */
ROM USB_DEVICE_DESCRIPTOR device_dsc=
{    0x12,    // Size of this descriptor in bytes
     USB_DESCRIPTOR_DEVICE,  // DEVICE descriptor type
     0x0200,                 // USB Spec Release Number
     CDC_DEVICE,             // Class Code
     0x00,                   // Subclass code
     0x00,                   // Protocol code
     EP0_BUFF_SIZE,          // Max packet size for EP0,
     0x04D8,                 // Vendor ID
     0x000C,                 // Product ID
```

Demo UVCView.exe

- Viewing Descriptor Information -



MICROCHIP
Regional Training Centers

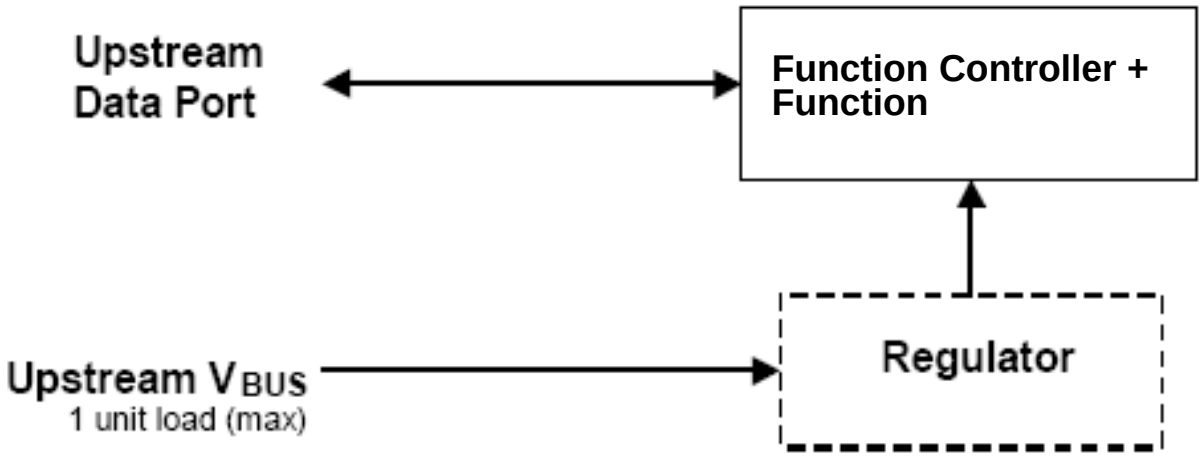
Agenda – Part 1

- USB Fundamentals – The serious & important stuff
  - Basics/Speeds
  - Topology/Physical Connection
  - Architecture/Programmer's Model
  - USB Transactions
  - USB Transfers
  - Device Classes
  - Enumeration
  - Descriptors
  - Power Planning
  - VID/PID & USB Compliance
- PIC18/24/32 USB Microcontrollers
- Microchip Demo/Development Solutions

Power Planning

- Architecture -

- **"Low Power" bus-powered function**
  - Draws up to 100 mA (1 'unit load') from the bus

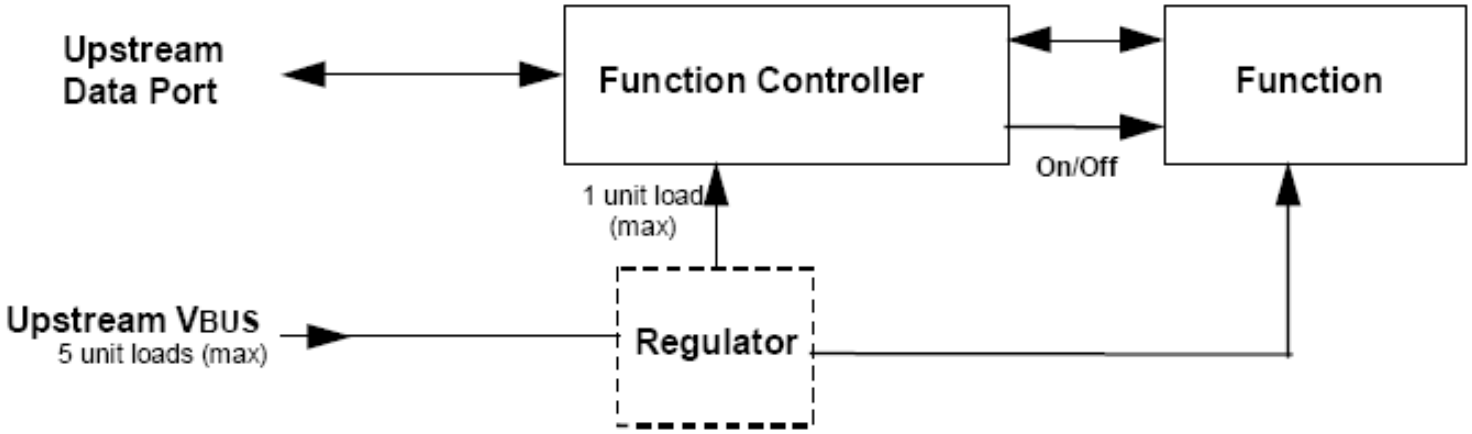


MICROCHIP
Regional Training Centers

Power Planning

- Architecture -

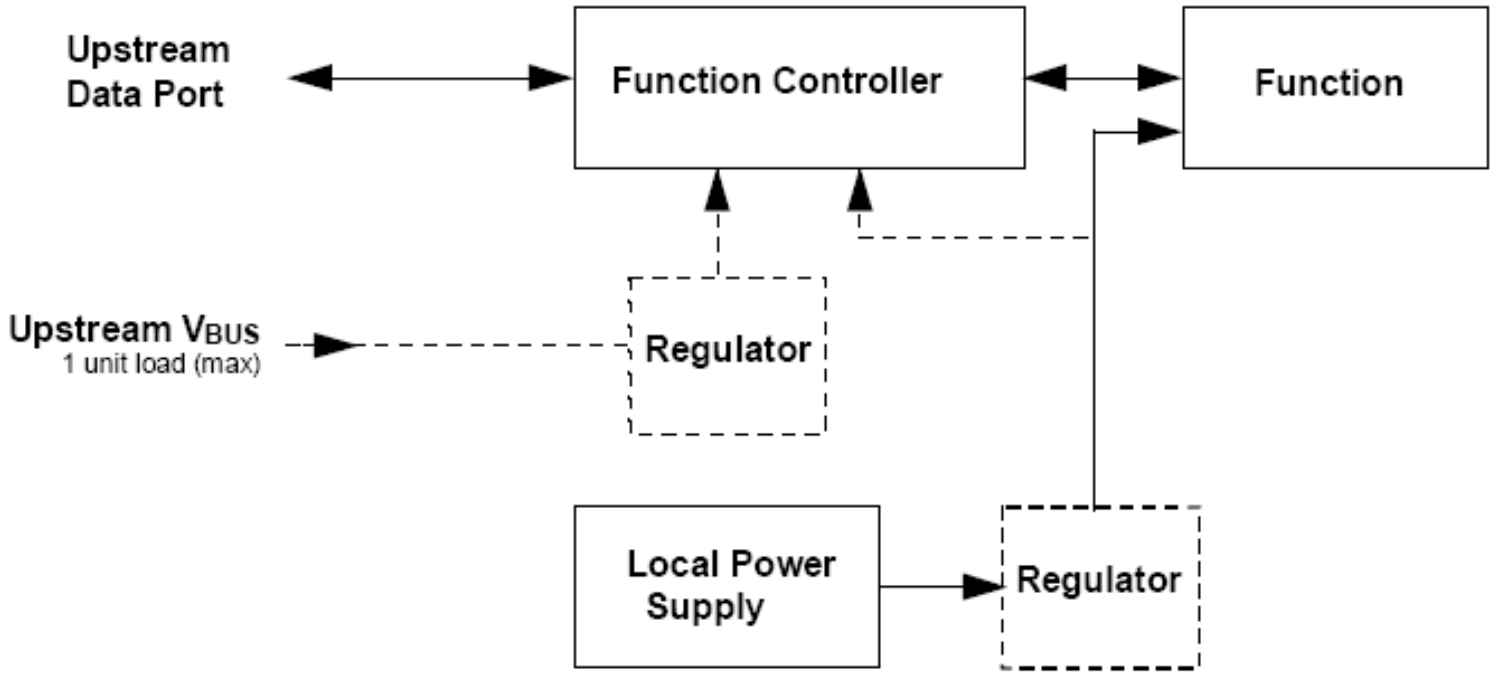
- **"High Power" bus-powered function**
  - Draws 100-500 mA from the bus
  - Must be able to enumerate at low power (100 mA)
    - **Device requests `bMaxPower`**
    - **Host enables configuration with appropriate `Set_Configuration` request**



Power Planning

- Architecture -

- **"Self Powered" device**
  - Can optionally draw up to 100 mA from the bus (if unsuspended) + as much as is available from its own supply



Power Planning

- Do I Need Self-Power? -

- **Device will need to provide self power if:**
  - It needs to function when not attached to the bus
  - It needs more than 500 mA
  - It needs to function when connected to battery powered PCs, or bus-powered hubs
    - **i.e. needs to function during Suspend mode**

MICROCHIP
Regional Training Centers

Power Planning

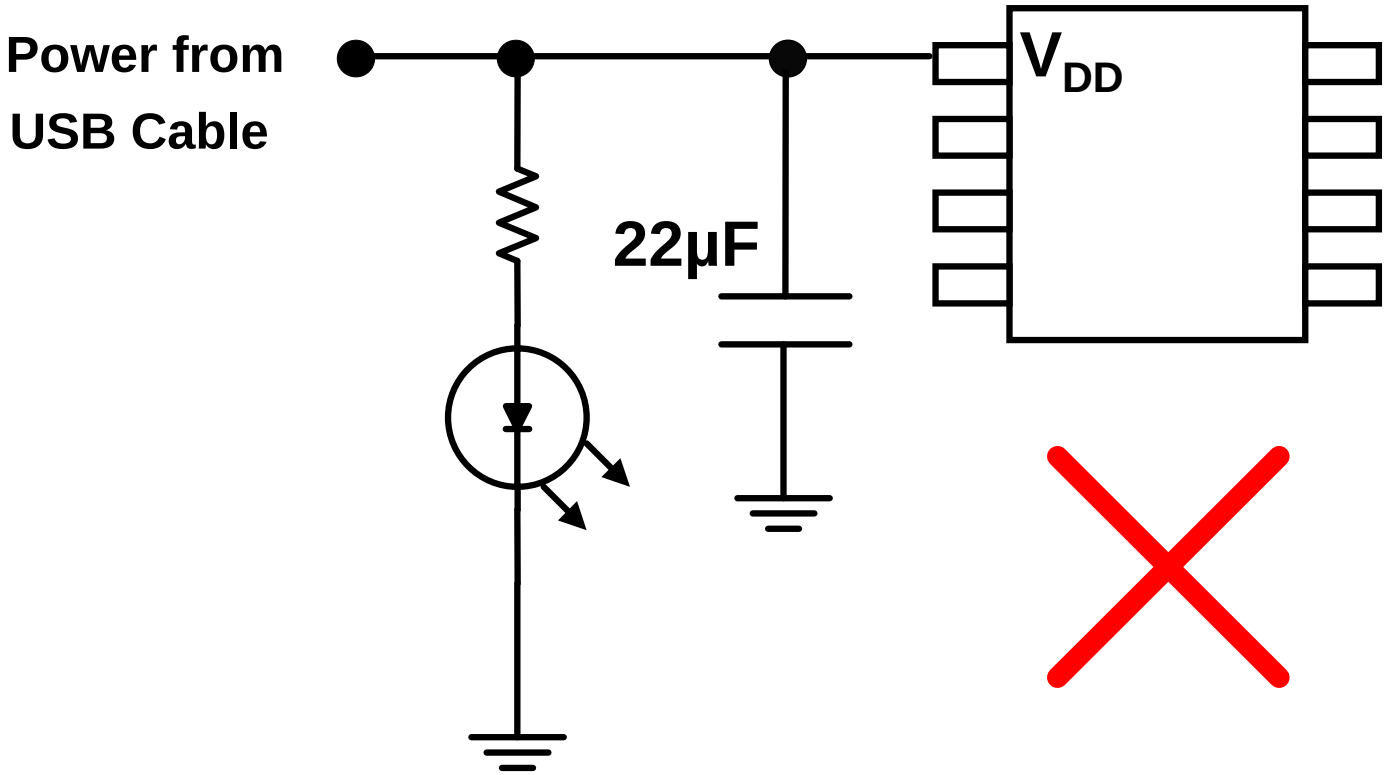
- Suspend Mode -

- **Bus activity may cease due to a host entering a "Suspend" mode to conserve battery life**
- **All devices must suspend if bus activity has not been observed for 3 ms**
  - And must operate with reduced current…

Power Planning

- Suspend Mode -

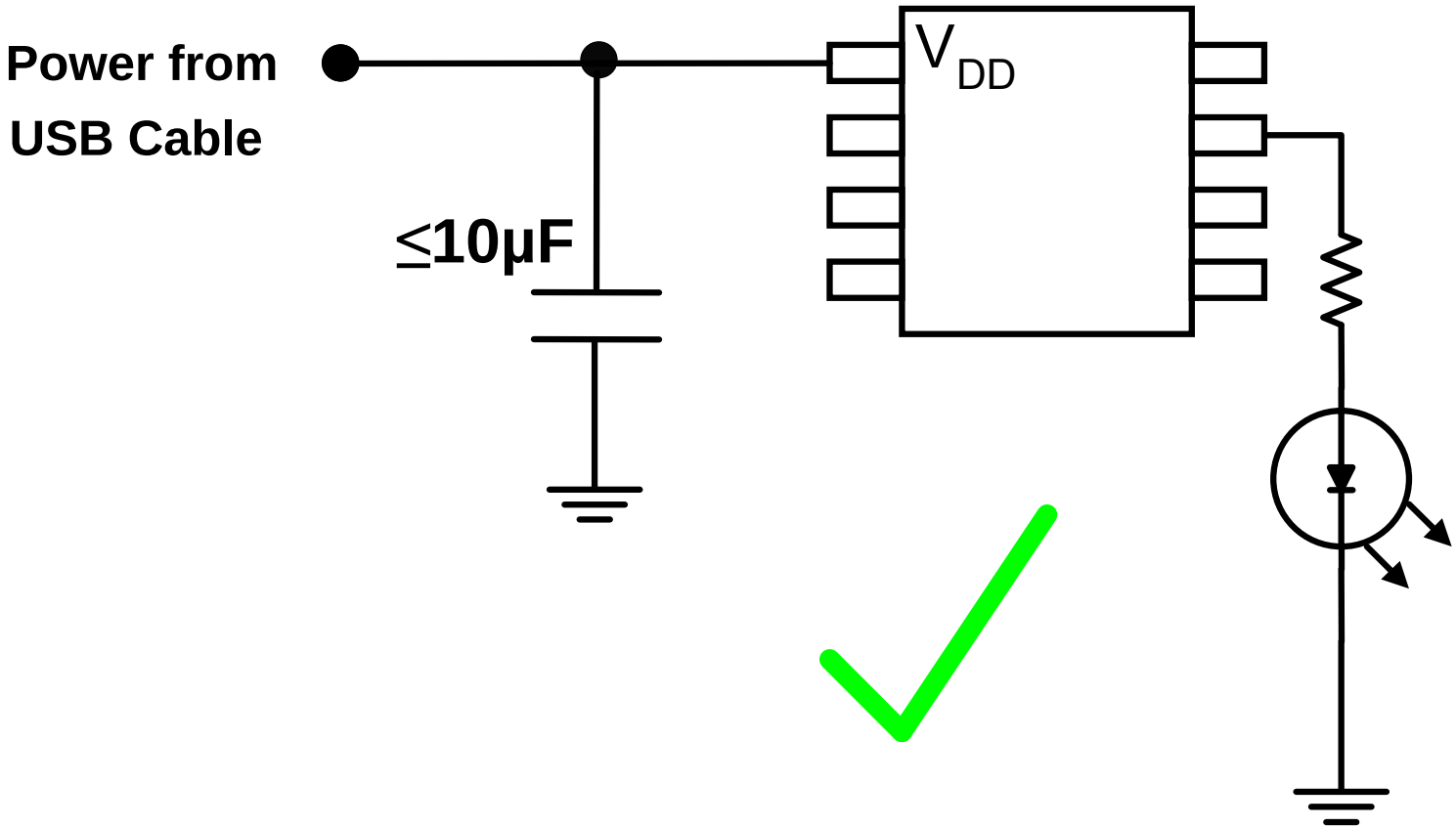
- **Maximum USB suspend current:**
  - 2.5 mA
- **Don't:**



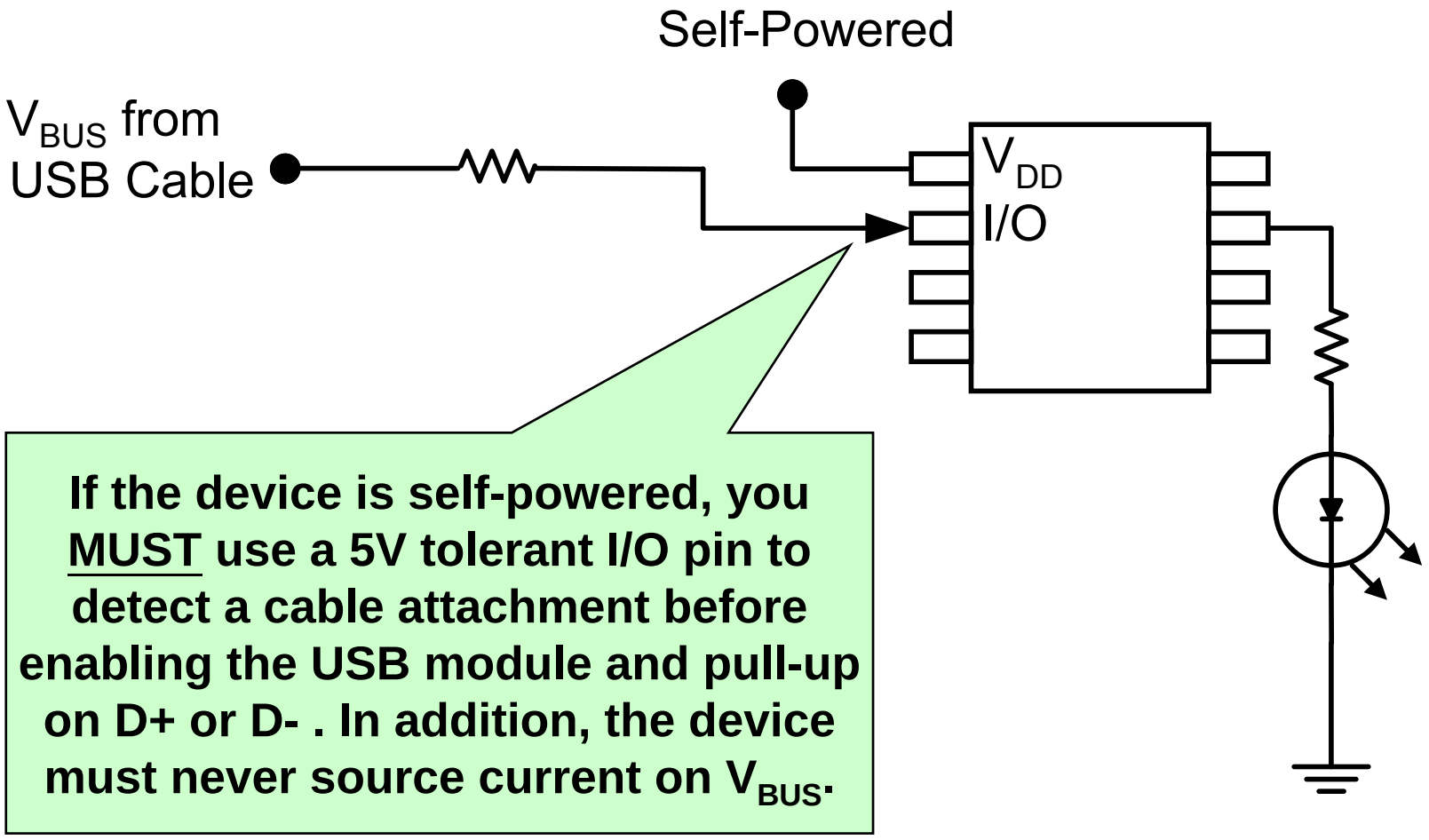
Power Planning

- Suspend Mode -

- **Maximum USB suspend current:**
  - 2.5 mA
- **Do:**



Self-Powered Devices
- Detecting a USB Attachment -



If the device is self-powered, you MUST use a 5V tolerant I/O pin to detect a cable attachment before enabling the USB module and pull-up on D+ or D- . In addition, the device must never source current on V_{BUS}.

MICROCHIP
Regional Training Centers

Agenda – Part 1

- USB Fundamentals – The serious & important stuff
  - Basics/Speeds
  - Topology/Physical Connection
  - Architecture/Programmer's Model
  - USB Transactions
  - USB Transfers
  - Device Classes
  - Power Planning
  - Enumeration
  - Descriptors
  - VID/PID & USB Compliance
- PIC18/24/32 USB Microcontrollers
- Microchip Demo/Development Solutions

MICROCHIP
Regional Training Centers

VID & PID

- **Vendor ID (VID):** ***16-bit number***
  - Required to market your product
  - http://www.usb.org/developers/vendor
  - USD $2,000
  - Technical & Legal trouble if not using an approved VID
- **Product ID (PID):** ***16-bit number***
  - Microchip's Sub-licensing Program
- **Every product line is required to have a unique combination of VID and PID**

MICROCHIP
Regional Training Centers

USB Compliance

- **Compliance Testing**
  - Must pass to use USB logo
  - Test fee: USD ~$1,500

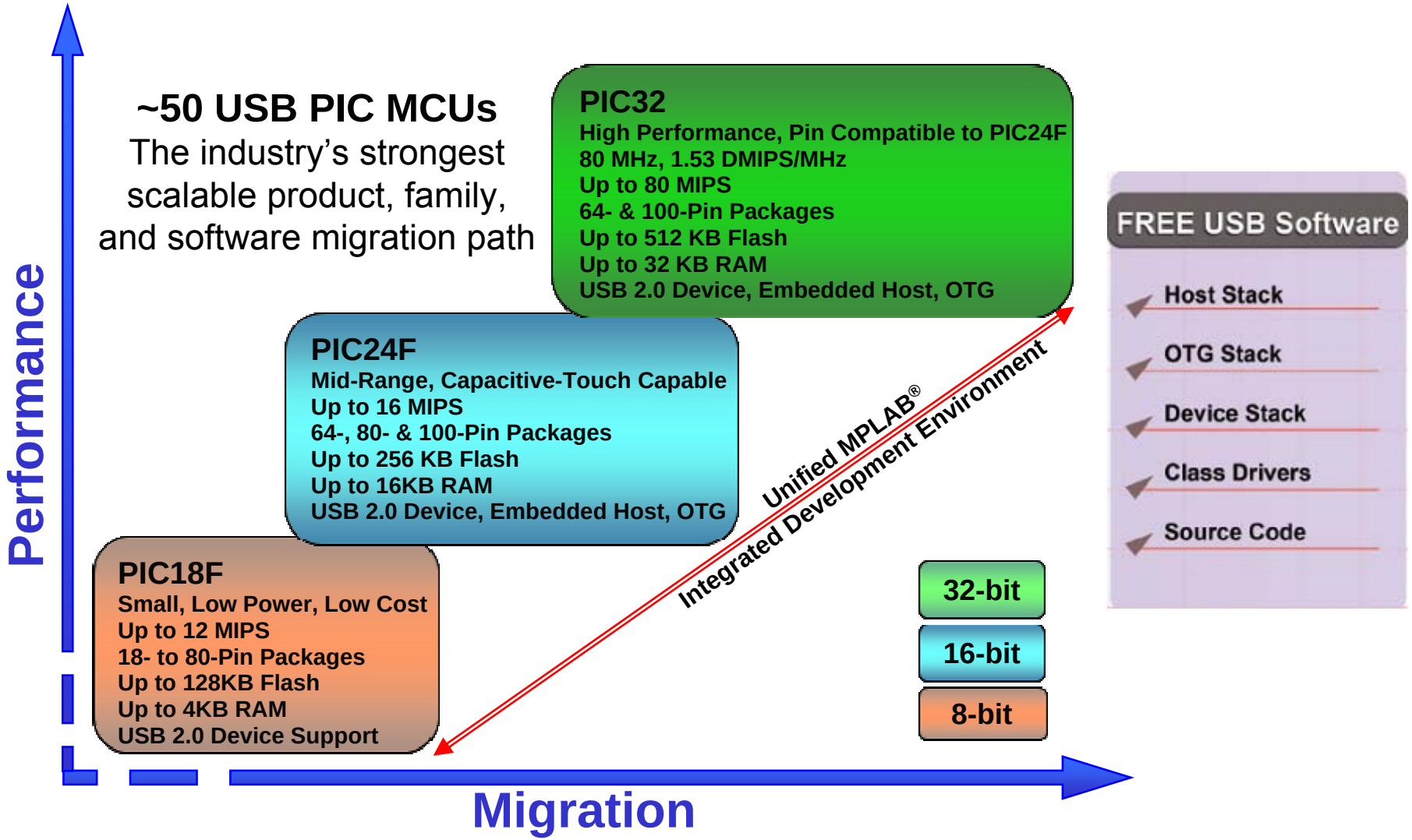
- **Tests device for conformance to USB Device Framework and Class standard control requests**
  - USB Protocol Analyzer
  - “USBCV” USB Command Verifier
  - **www.usb.org/developers/tools**
- **Electrical Signal Quality**
- **Power Management**

MICROCHIP
Regional Training Centers

Agenda – Part 1

- Brief history of USB
- USB Fundamentals – The serious & important stuff
  - Basics/Speeds
  - Topology/Physical Connection
  - Architecture/Programmer's Model
  - USB Transactions
  - USB Transfers
  - Device Classes
  - Enumeration
  - Descriptors
  - Power Planning
  - VID/PID & USB Compliance
- PIC18/24/32 USB Microcontrollers
- Microchip Demo/Development Solutions

Scalable USB PIC® MCU Portfolio



MICROCHIP
Regional Training Centers

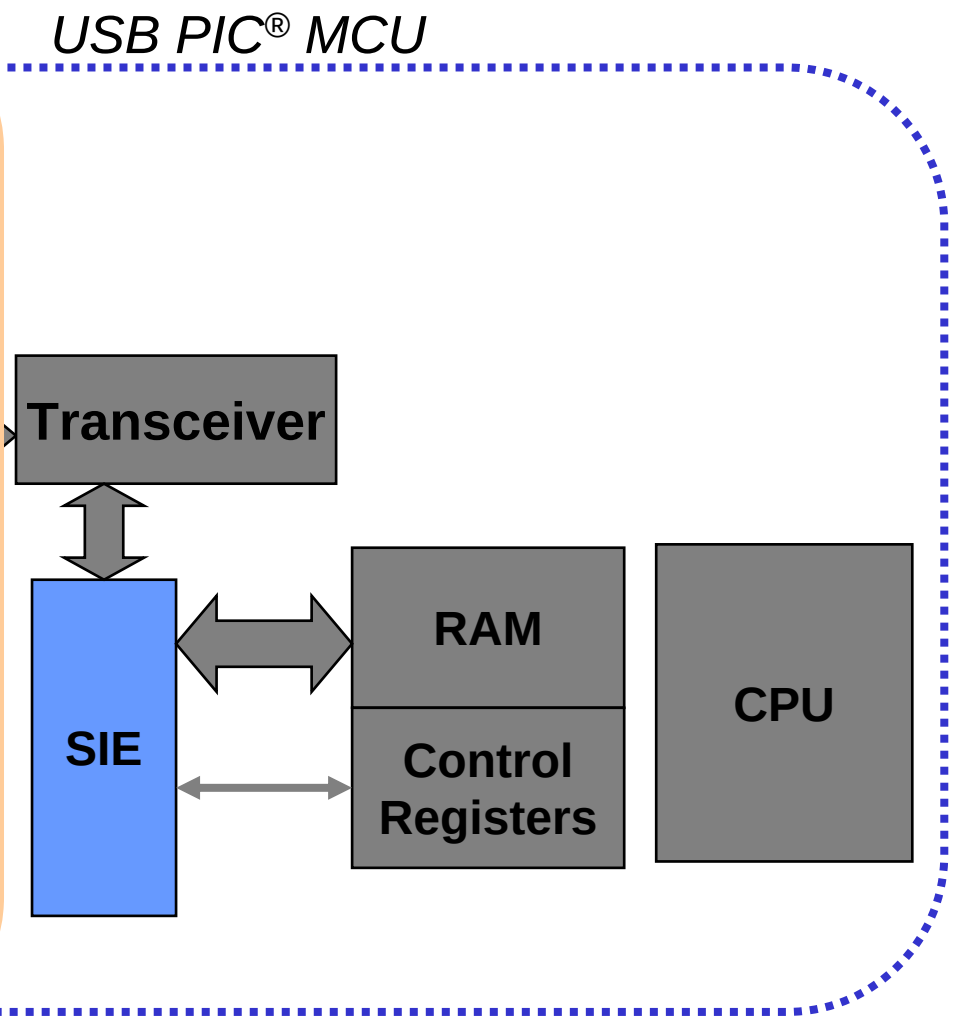
USB Microcontroller Portfolio

	PIC18F14K50	PIC18F4450 PIC18F4550 PIC18F4553	PIC18F87J50	PIC24FJ256GB1XX	PIC32MX4XXF512
Core	8-bit	8-bit	8-bit	16-bit	32-bit
USB	USB 2.0 device	USB 2.0 device	USB 2.0 device	USB 2.0 device, embedded host, dual role, OTG	USB 2.0 device, embedded host, dual role, OTG
Flash	16K bytes	up to 32K bytes	128K bytes	256K bytes	512K bytes
RAM	768 bytes	up to 2048 bytes	3904 bytes	16K Bytes	32K Bytes
mTouch™ support	yes	yes, external	yes, external	yes CTMU	yes, external
UARTs	1	1	2	4	2
SPI	1	1	1	3	2
I^2C™	1	1	1	3	2
Peripheral Pin Select	no	no	no	yes	no
ADC	10 bit, 9 channel	10 bit, 10 and 13 ch 12 bit, 10 and 13 ch	10 bit, 8 and 12 channel	10 bit, 16 channel	10 bit, 16 channel
RTCC	software	software	software	yes	yes
Parallel Master Port	no	no	yes	yes	yes
Analog comparators	2	2	2	3	2
Free SW Stacks	yes	yes	yes	yes	yes
Free Class Drivers	yes	yes	yes	yes	yes
Scalable Development Environment	yes	yes	yes	yes	yes
Packages	20-pin	28, 40, 44-pin	60, 80-pin	64, 80, 100-pin	64, 100-pin

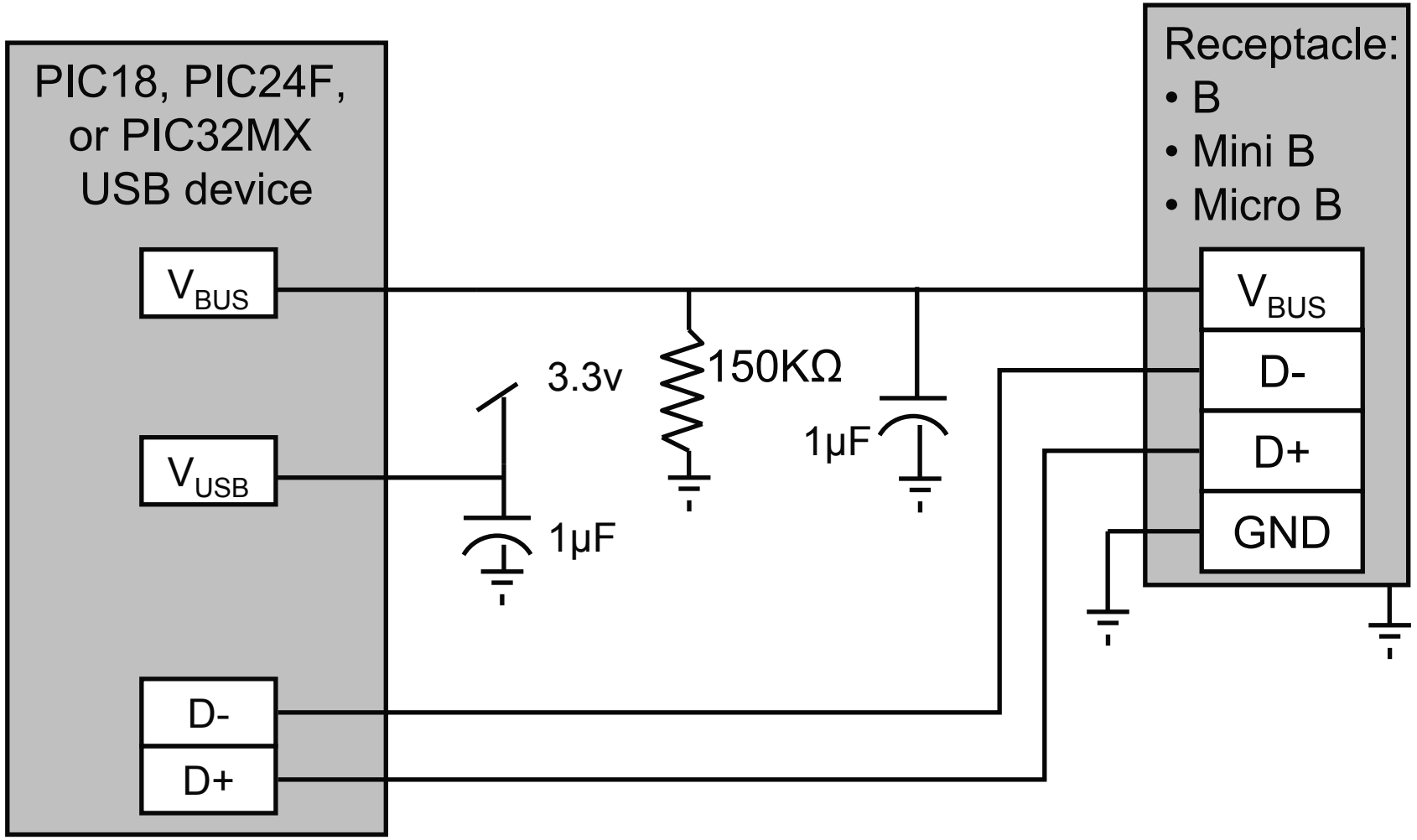
Serial Interface Engine

SIE ...

- Serializes and deserializes USB data
- Encodes and decodes NRZI data
- Handles bit stuffing
- Checks CRC to validate data packet
- Detects bus signaling events and notifies the CPU through interrupts
- Handles USB transactions
- Handles handshaking protocol



USB Device (Peripheral) Example Circuit



MICROCHIP
Regional Training Centers

Agenda – Part 1

- Brief history of USB
- USB Fundamentals – The serious & important stuff
  - Basics/Speeds
  - Topology/Physical Connection
  - Architecture/Programmer's Model
  - USB Transactions
  - USB Transfers
  - Device Classes
  - Enumeration
  - Descriptors
  - Power Planning
  - VID/PID & USB Compliance
- PIC18/24/32 USB Microcontrollers
- Microchip Demo/Development Solutions

MICROCHIP
Regional Training Centers

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 PIC18F46J50
  - USB communication
- Priced from $59.98
- Part Numbers
  - DM180021
- Available Now

MICROCHIP
Regional Training Centers

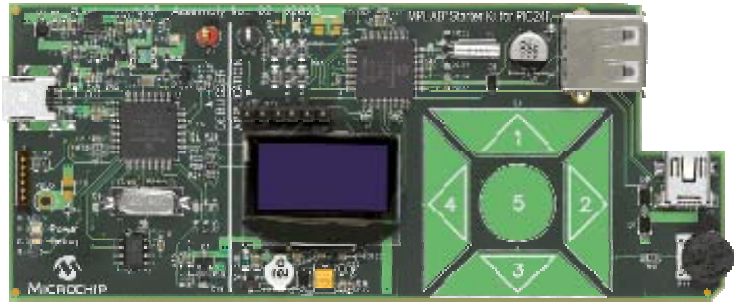
Low Pin Count USB Development Kit

- **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

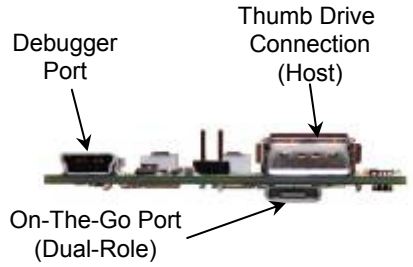
- **PIC24F Starter Kit 1**
  - Part #: DM240011
  - PIC24FJ256GB110

  Priced from $59.99



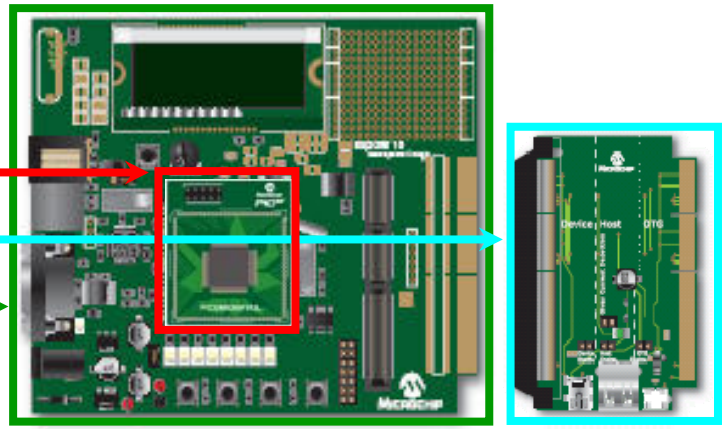
- **PIC32 USB Starter Board II**
  - Part #: DM320003-2
  - PIC32MX795F512L

  Priced from $55.00



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

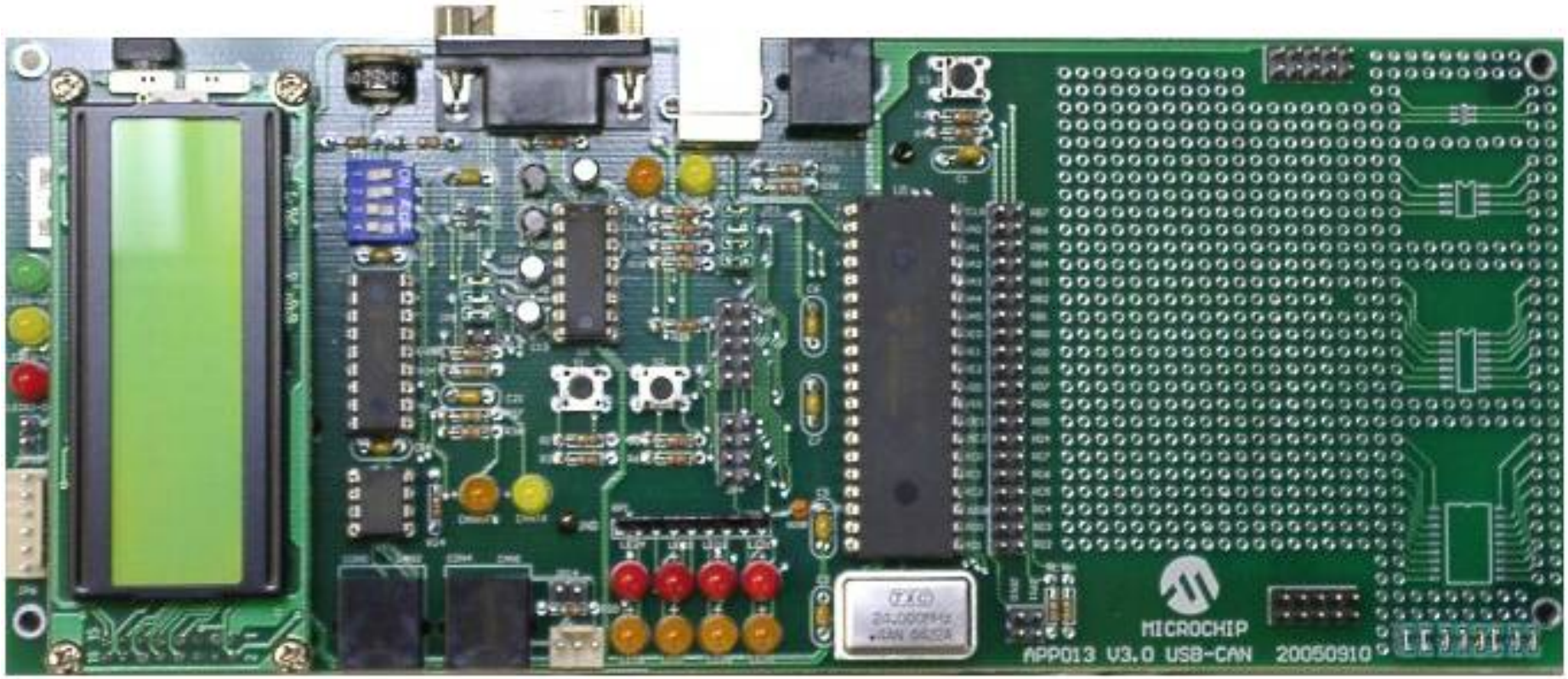
  Priced from $214.99



- **All Available Now**

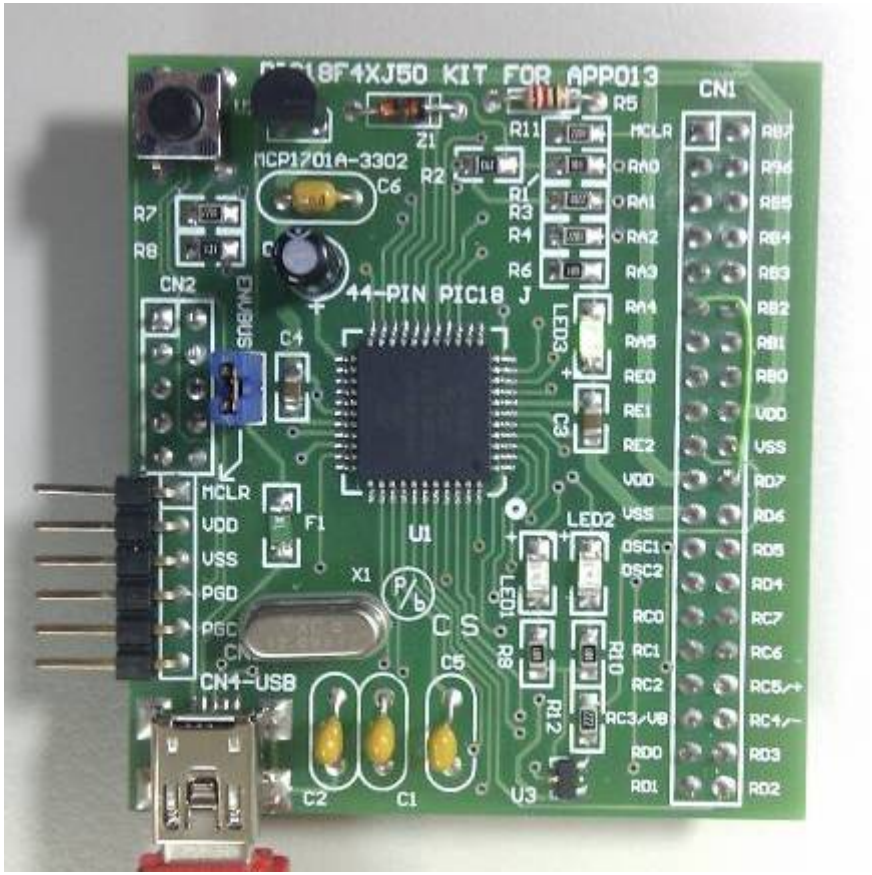
APP013

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



APP013 Hobby Kit

- **Hobby Kit爲APP013的子版, 可以透過子版更新不同系列之MCU.**



Microchip USB Frameworks

- www.microchip.com/usb -

- **MCHPFSUSB Framework**
  - PIC18F & PIC24F & PIC32 USB MCUs
  - C18/C30/C32 Compatible
  - MPLAB® IDE Project Centric
- **Device Stacks**
  - Audio, HID, CDC, MSD, Custom
  - Polling or Interrupt driven
- **Embedded Host Stack**
  - PIC24F & PIC32 USB MCUs
  - Polling or Event-driven Scheme
  - Client drivers for CDC, Charger, Custom, HID, MSD, Printer
- **On-The-Go (OTG) Support**
  - PIC24F & PIC32 USB MCUs

MICROCHIP
Regional Training Centers

www.microchip.com/usb

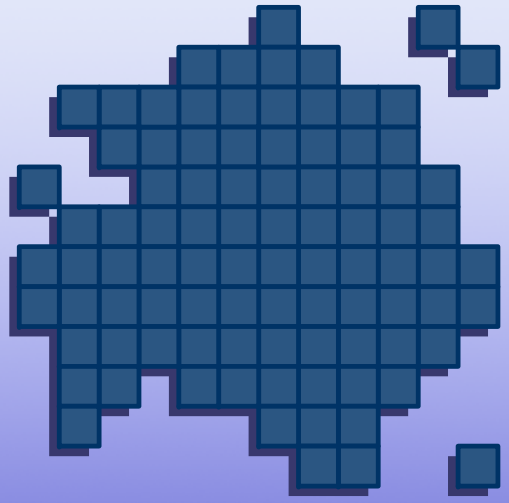
- Microchip USB Design Center

- Buy USB Development Boards and Kits from microchipDIRECT
  www.microchipdirect.com
- Detailed Product Information at:
  - www.microchip.com/pic32
  - www.microchip.com/pic24
  - www.microchip.com/pic18

MICROCHIP
Regional Training Centers

Quiz!

- Maximum number of devices USB can support?
- Number of pins in a standard USB connector?
- What is FS USB data rate?
- Which direction is Data-IN?
- USB data transfer types?
- What is USB Enumeration?



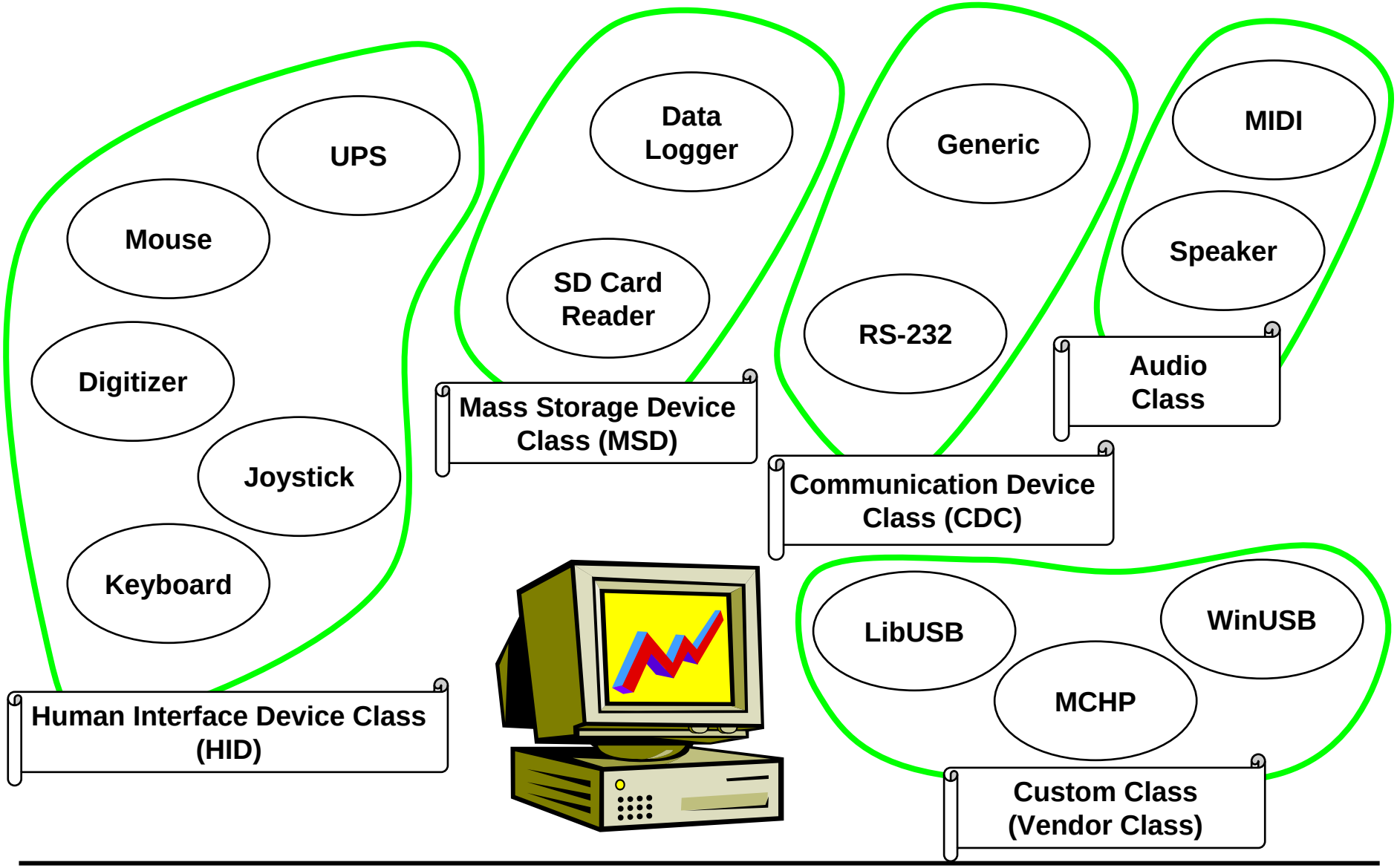
Part 2.

Introduction to Microchip's USB Device Frameworks

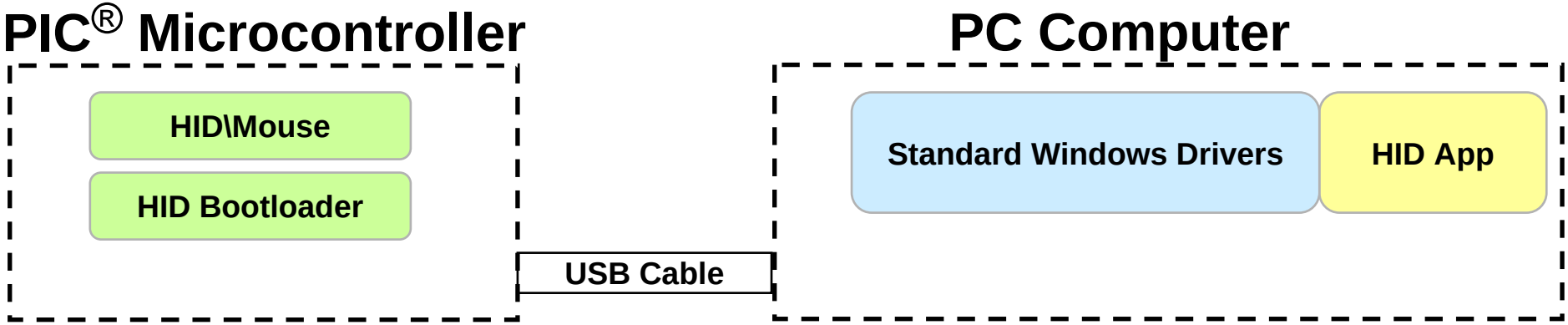
Agenda – Part 2

- Design Considerations
- The MCHPFSUSB Device Framework
  - Microchip Application Libraries
  - MCHPFSUSB PC Tools
  - Basic USB Device Project Structure (Polled vs. Interrupt-Driven)
    - Lab 1a – Create an USB template project (Polling)
    - Lab 1b – Create an USB template project (Interrupts)
  - Editable Files Description
    - usb_descriptors.c
    - hardware_profile.h
    - usb_config.h
  - Event Handler Function

MCHPFSUSB Device Side Class Frameworks Available



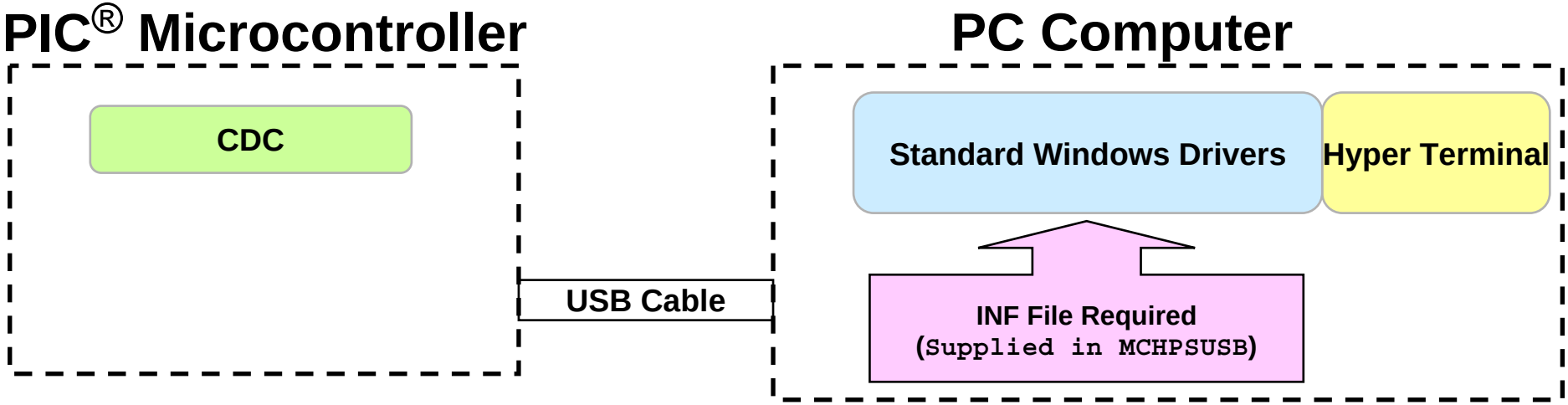
Human Interface Device (HID)



Design Considerations:

- 64 KB/s max
- Interrupt Transfer Type
- Standard Windows driver
- Custom PC application can access HID data through Win32 APIs

CDC – RS-232 Emulation

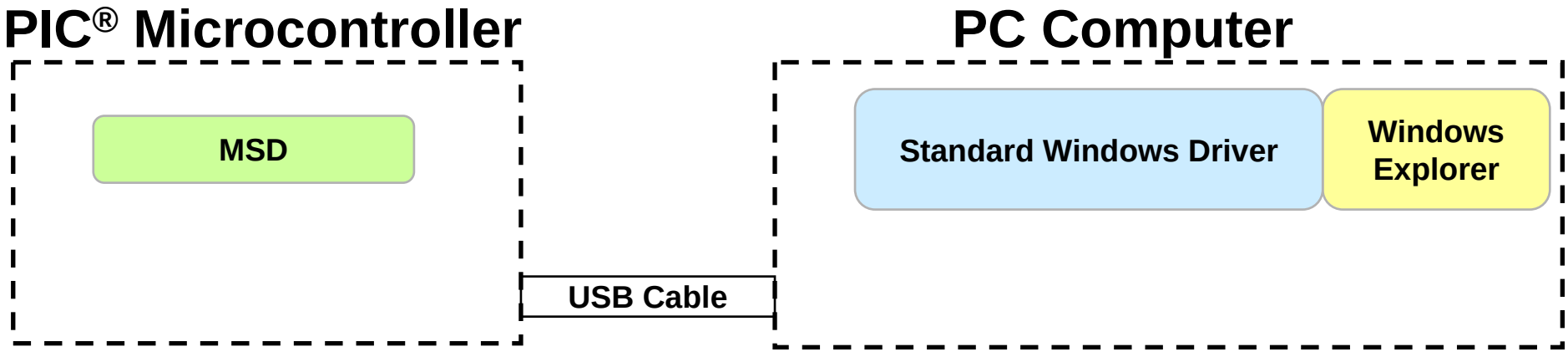


Design Considerations:

- ~80 KB/s max
- Bulk Transfers
- PC applications can access the device as though it is connected to a serial COM port

Mass Storage Device (MSD)

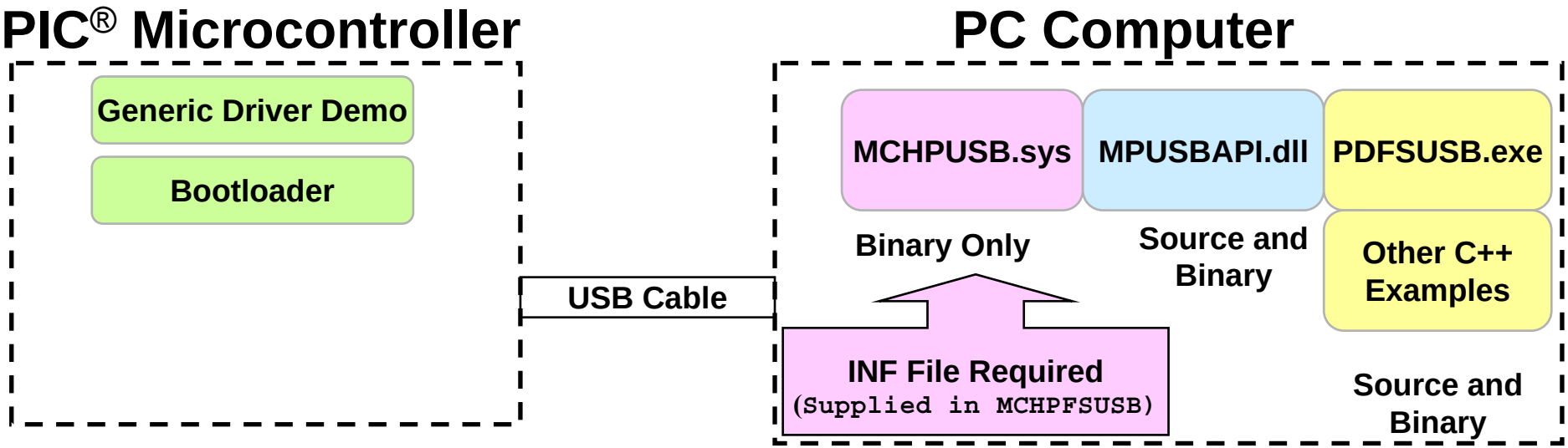
See AN1189: Implementing a Mass Storage Device Using the Microchip USB Device Firmware Framework



Design Considerations:

- Acts like a disk drive
- Fast data transfer over USB
- Speed is limited depending on the physical media interface
- For usefulness, a File System should also be implemented in firmware

Custom Class Driver



Design Considerations:

- ~1,088 KB/s max
- Very flexible (Control, Bulk, Int, Iso Transfers are possible)
- Not a standard Windows driver
- PC programming is required

MICROCHIP
Regional Training Centers

Agenda – Part 2

- Design Considerations
- The MCHPFSUSB Device Framework
  - Microchip Application Libraries
  - MCHPFSUSB PC Tools
  - Basic USB Device Project Structure (Polled vs. Interrupt-Driven)
    - Lab 1a – Create an USB template project (Polling)
    - Lab 1b – Create an USB template project (Interrupts)
  - Editable Files Description
    - usb_descriptors.c
    - hardware_profile.h
    - usb_config.h
  - Event Handler Function

Microchip Application Libraries

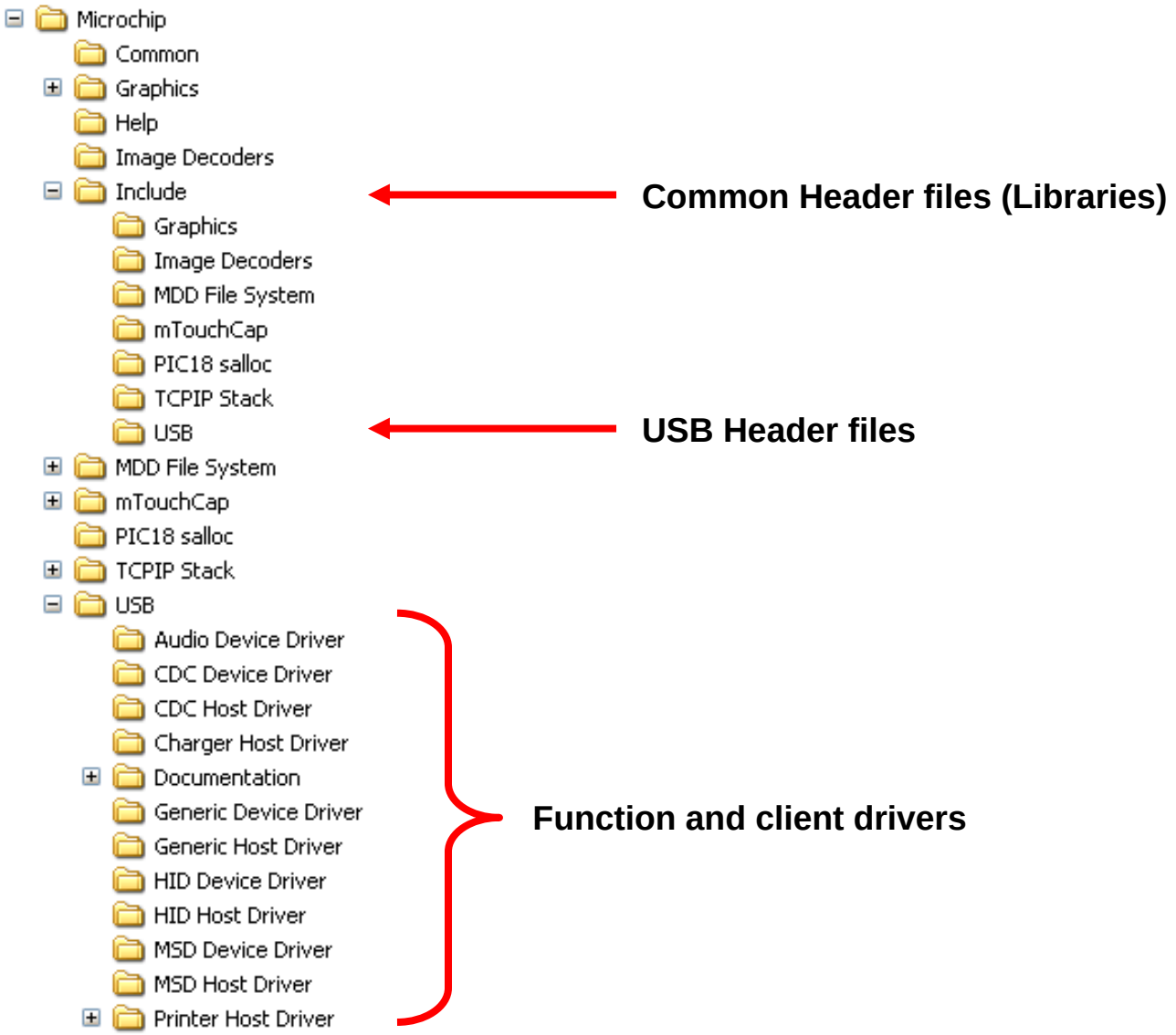
- Download from www.microchip.com/MAL
- Contains different stacks (USB, TCP/IP, Graphics, mTouch™, etc.)
- Default installation path:

  C:\Microchip Solutions v20xx-xx-xx\
  For Example:
  C:\Microchip Solutions v2011-10-19

- Contains libraries source code and several example firmware projects based on Microchip's Development Tools

Libraries Source Files Directory

- C:\Microchip Solutions\Microchip -



MCHPFSUSB Framework

- Supported Platforms -

- Low Pin Count USB Development Kit (PIC18F14K50 Family)
- PICDEM™ Full Speed USB (PIC18F4550 family)
- MPLAB® Starter Kit for PIC18F MCU (PIC18F46J50 family)
- PIC18F46J50 FS USB Demo Board (+ HPC Explorer Board)
- PIC18F87J50 FS USB Demo Board (+ HPC Explorer Board)
- MPLAB Starter Kit for PIC24F (PIC24FJ256GB110 family)
- PIC24F USB PIM (+ Explorer 16 + USB PICtail™ Plus)
- PIC32 USB PIM (+ Explorer 16 + USB PICtail Plus)
- PIC32 USB Starter Board (PIC32MX460F512L family)
- PIC32 USB Starter Board II (PIC32MX795F512L family)

MICROCHIP
Regional Training Centers

Agenda – Part 2

- Design Considerations
- The MCHPFSUSB Device Framework
  - Microchip Application Libraries
  - MCHPFSUSB PC Tools
  - Basic USB Device Project Structure (Polled vs. Interrupt-Driven)
    - Lab 1a – Create an USB template project (Polling)
    - Lab 1b – Create an USB template project (Interrupts)
  - Editable Files Description
    - usb_descriptors.c
    - hardware_profile.h
    - usb_config.h
  - Event Handler Function

MICROCHIP
Regional Training Centers

MCHPFSUSB PC Tools

- **Microchip General Purpose USB Driver**

  A general purpose Windows® driver which can be used by Windows applications to interface with a custom class USB device

- **PICDEM™ FS USB Demo Tool**

  A computer program demonstrates basic USB communication using the Microchip Custom class driver with a Windows GUI based application

- **USB CDC Serial Demo**

  A simple .inf file that needs to be supplied to Windows the first time you connect a CDC device to your PC

- **Microchip USB OTG Configuration Tool**

  A simple to use interface to help generate the configuration files required by the USB stack

MICROCHIP
Regional Training Centers

mchpcdc.inf

- .inf file tells Windows® how to configure and use a device
- Device Manager reads the VID and PID from the target device
- Windows uses .inf file to:
  - Associates driver to use with VID & PID
  - Specifies device identification strings
  - Specifies source and destination of device files
  - Sets registry keys

mchpcdc.inf

- Editable Sections -

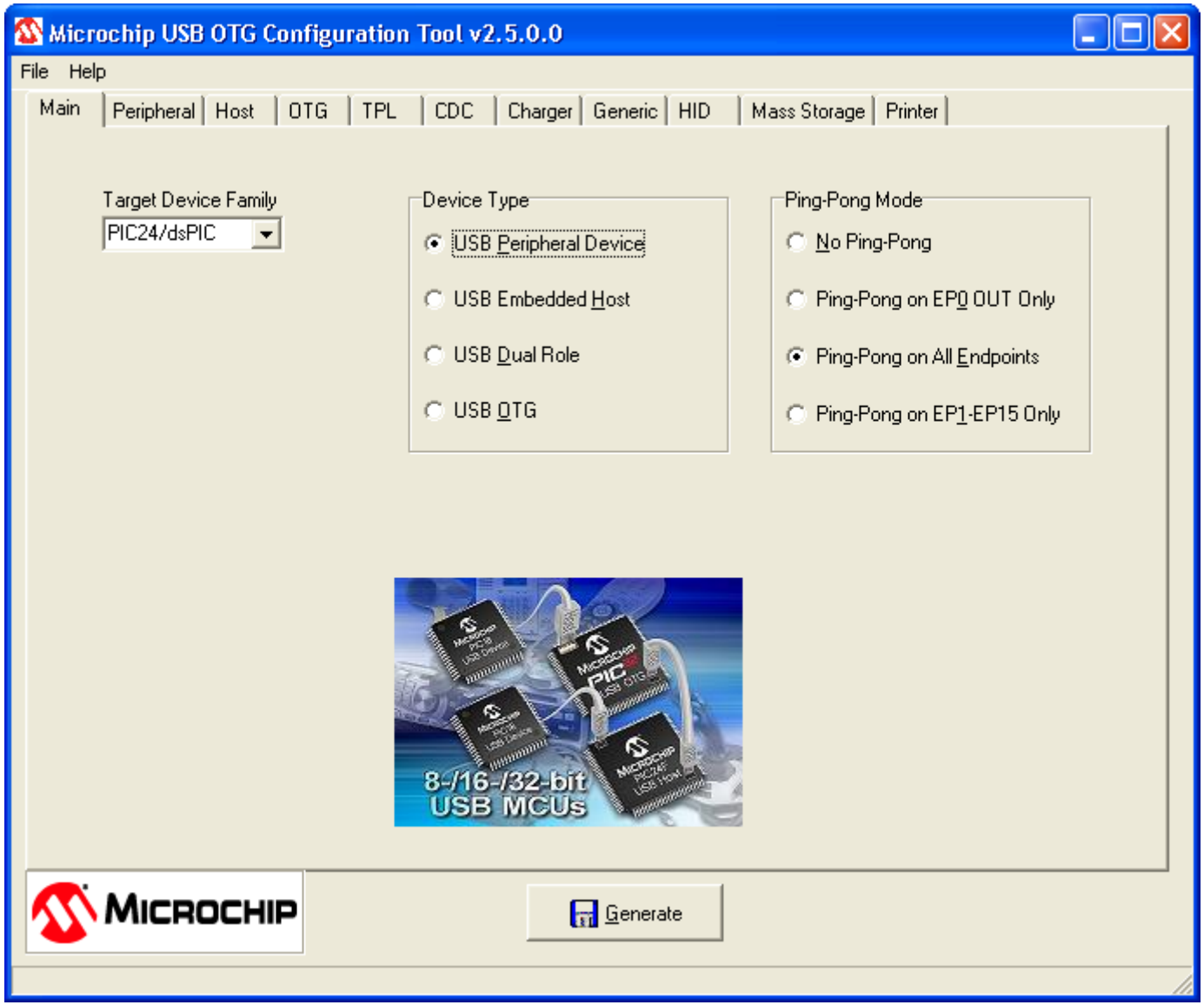
```
;------------------------------------------------------------------------------
;  Vendor and Product ID Definitions
;------------------------------------------------------------------------------
; Note: One INF file can be used for multiple devices with different VID and PIDs.
; For each supported device, append ",USB\VID_xxxx&PID_yyyy" to the end of the line.
;------------------------------------------------------------------------------
[SourceDisksFiles]
[SourceDisksNames]
[DeviceList]
%DESCRIPTION%=DriverInstall, USB\VID_04D8&PID_000A

[DeviceList.NTamd64]
%DESCRIPTION%=DriverInstall, USB\VID_04D8&PID_000A
;------------------------------------------------------------------------------
;  String Definitions
;------------------------------------------------------------------------------
;Modify these strings to customize your device
;------------------------------------------------------------------------------
[Strings]
MFGFILENAME="mchpcdc"
DRIVERFILENAME ="usbser"
MFGNAME="Microchip Technology, Inc."
INSTDISK="Microchip Technology, Inc. Installation Disc"
DESCRIPTION="Communications Port"
SERVICE="USB RS-232 Emulation Driver"
```

USB Stack Configuration Tool

Main:

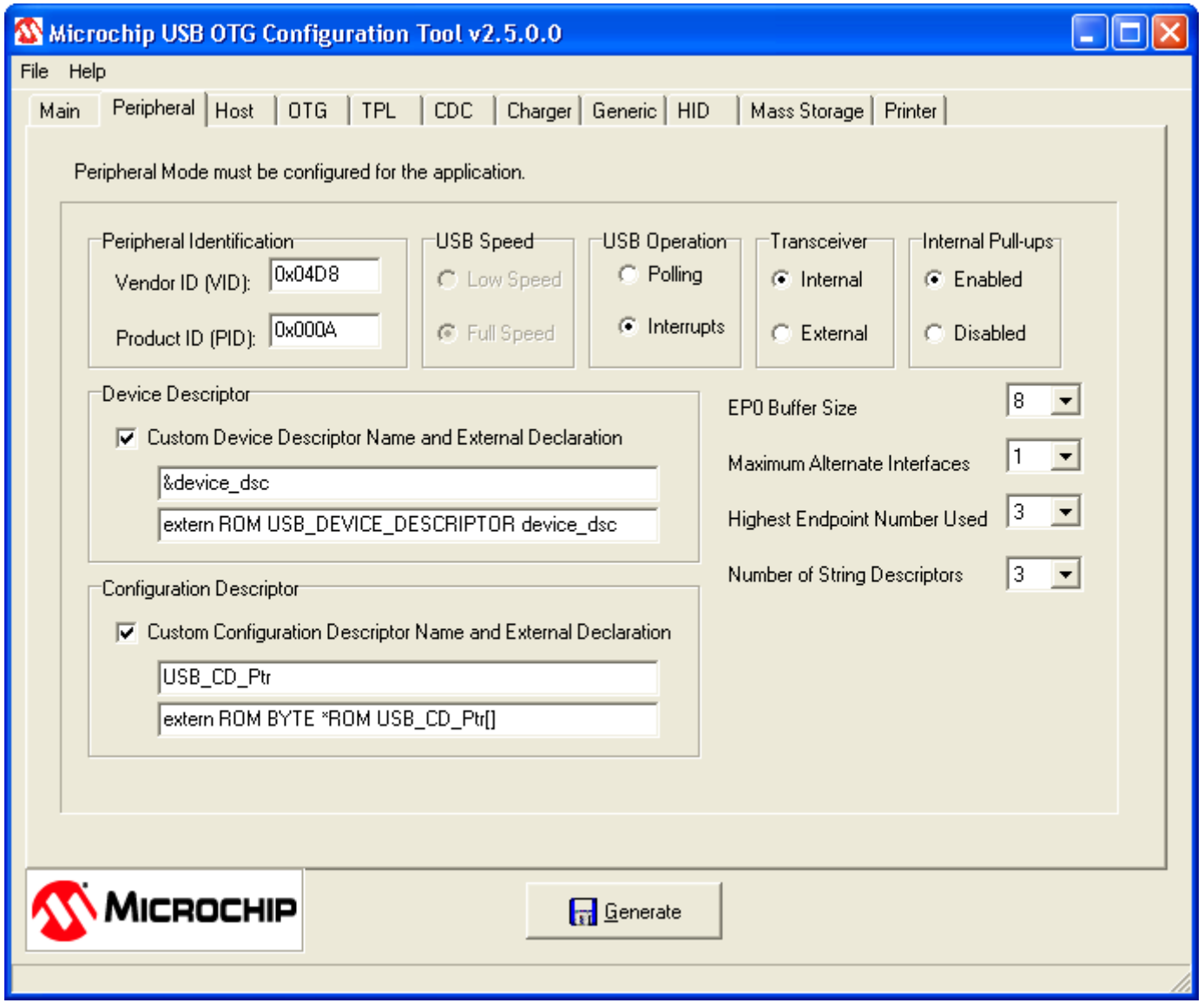
- Target Device
- Device Type
- Ping-Pong Mode



USB Stack Configuration Tool

Peripheral:

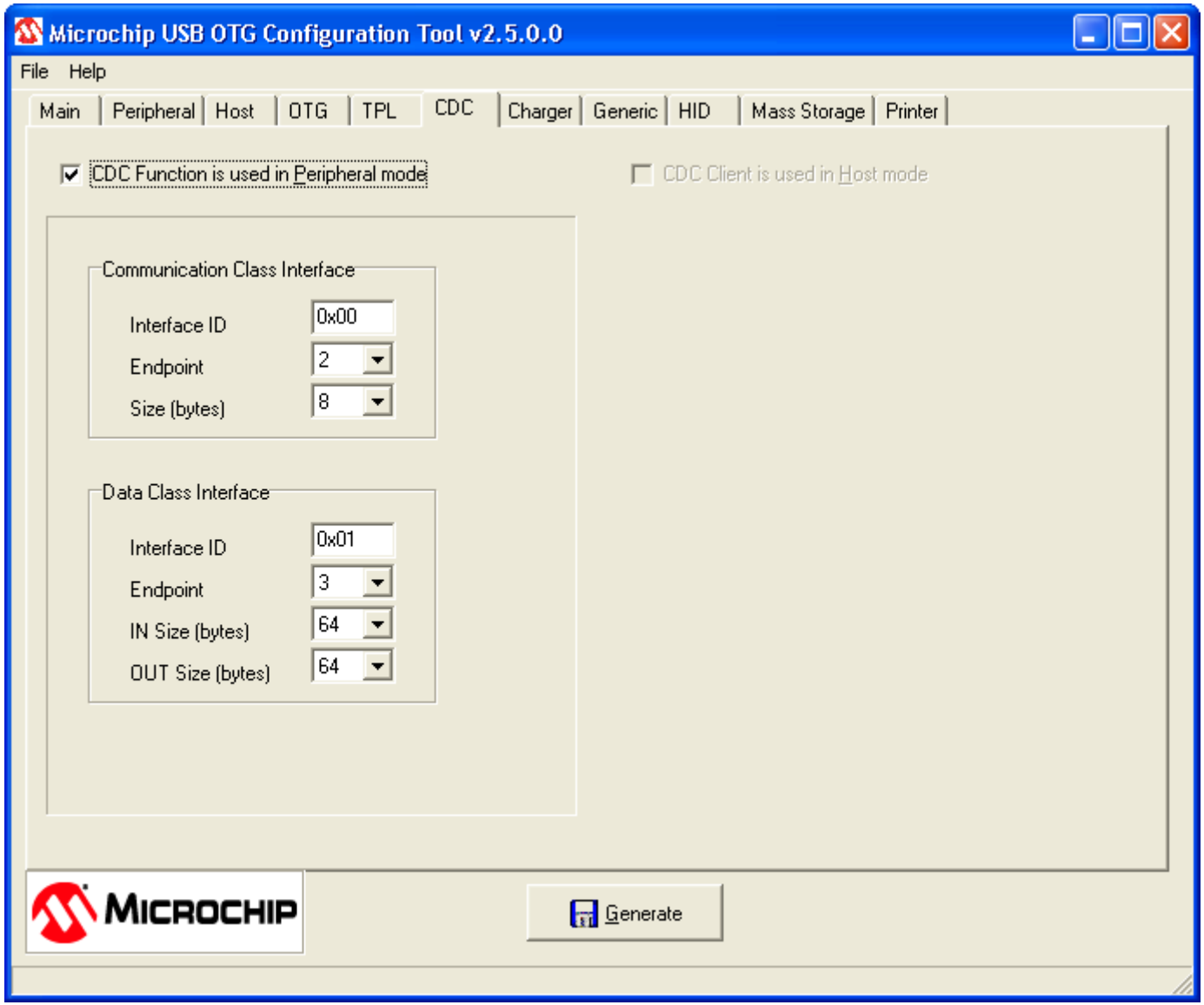
- VID &PID
- Speed
- USB operation
- Transceiver options
- Device and configuration descriptors pointers
- Endpoint 0 Buffer Size
- # interfaces
- # strings



USB Stack Configuration Tool

Function:

- Class specific interface options
- Endpoint used
- Endpoint configuration



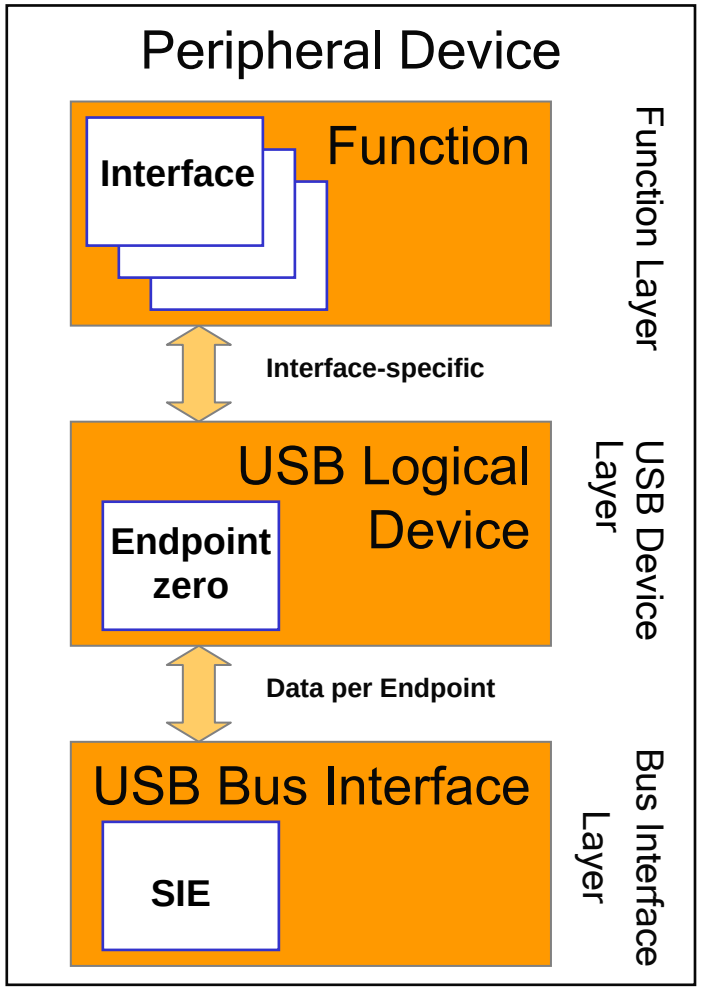
MICROCHIP
Regional Training Centers

Agenda – Part 2

- Design Considerations
- The MCHPFSUSB Device Framework
  - Microchip Application Libraries
  - MCHPFSUSB PC Tools
  - Basic USB Device Project Structure (Polled vs. Interrupt-Driven)
    - Lab 1a – Create an USB template project (Polling)
    - Lab 1b – Create an USB template project (Interrupts)
  - Editable Files Description
    - usb_descriptors.c
    - hardware_profile.h
    - usb_config.h
  - Event Handler Function

USB Device Framework

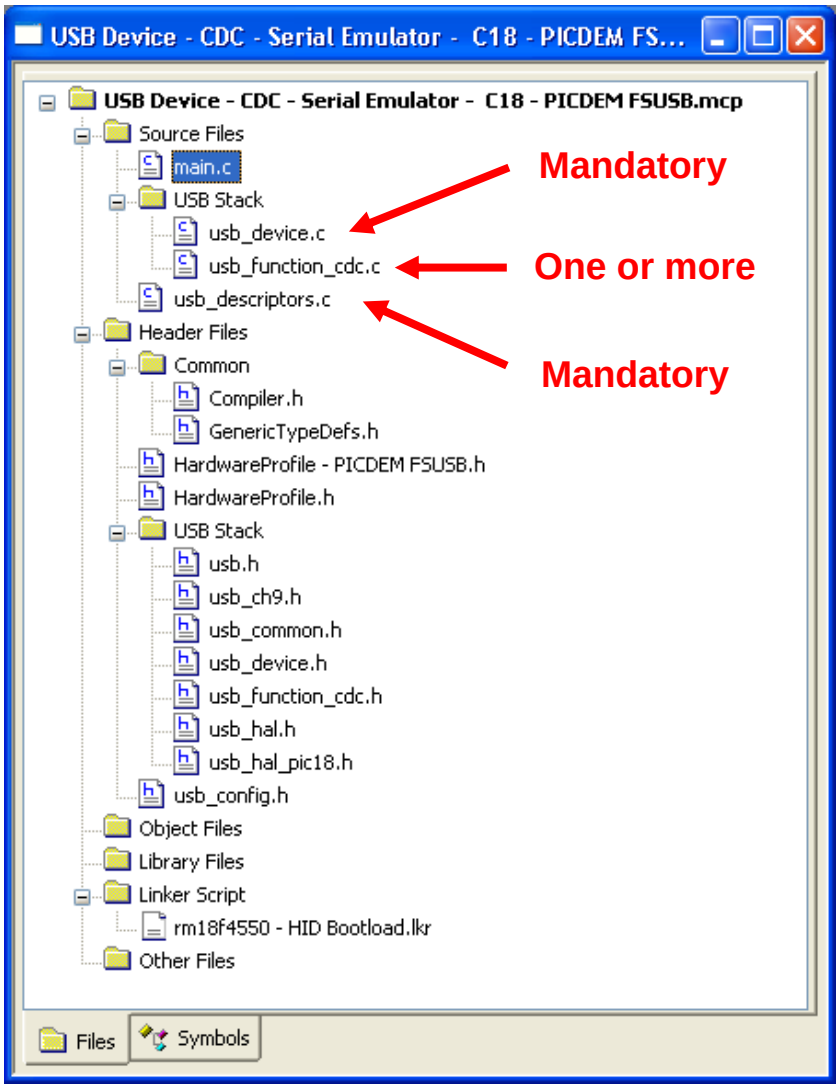
- Software View of Hardware -



- **Function:**
  `usb_function_audio.c`
  `usb_function_cdc.c`
  `usb_function_generic.c`
  `usb_function_hid.c`
  `usb_function_msd.c`

- **USB Logical Device:**
  `usb_device.c`

Generic USB Device Project



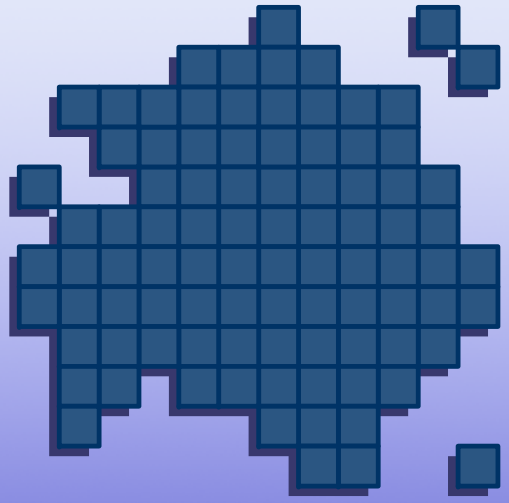
- Must have `main()` function (`main.c`)
- Must include:
  `usb_device.c`
  `usb_descriptors.c`
- Can include one or more function drivers (composite devices)
- Must have the path "`<user>\microchip\include`" into Include Search Path of building option

MICROCHIP
Regional Training Centers

Lab Prepare

- Install MPALC IDE (v8.76)
- Install MPLAB C18 (v3.40)
- Install Microchip Application Libraries (v2010-10-19)
- Copy Microchip USB Stack to exercise directory.

  Copy
  C:\Microchip Solutions v2010-10-19\Microchip
  to
  COM3101T\Exercise\



Lab Exercise 1a

Create an USB template project (Polling)

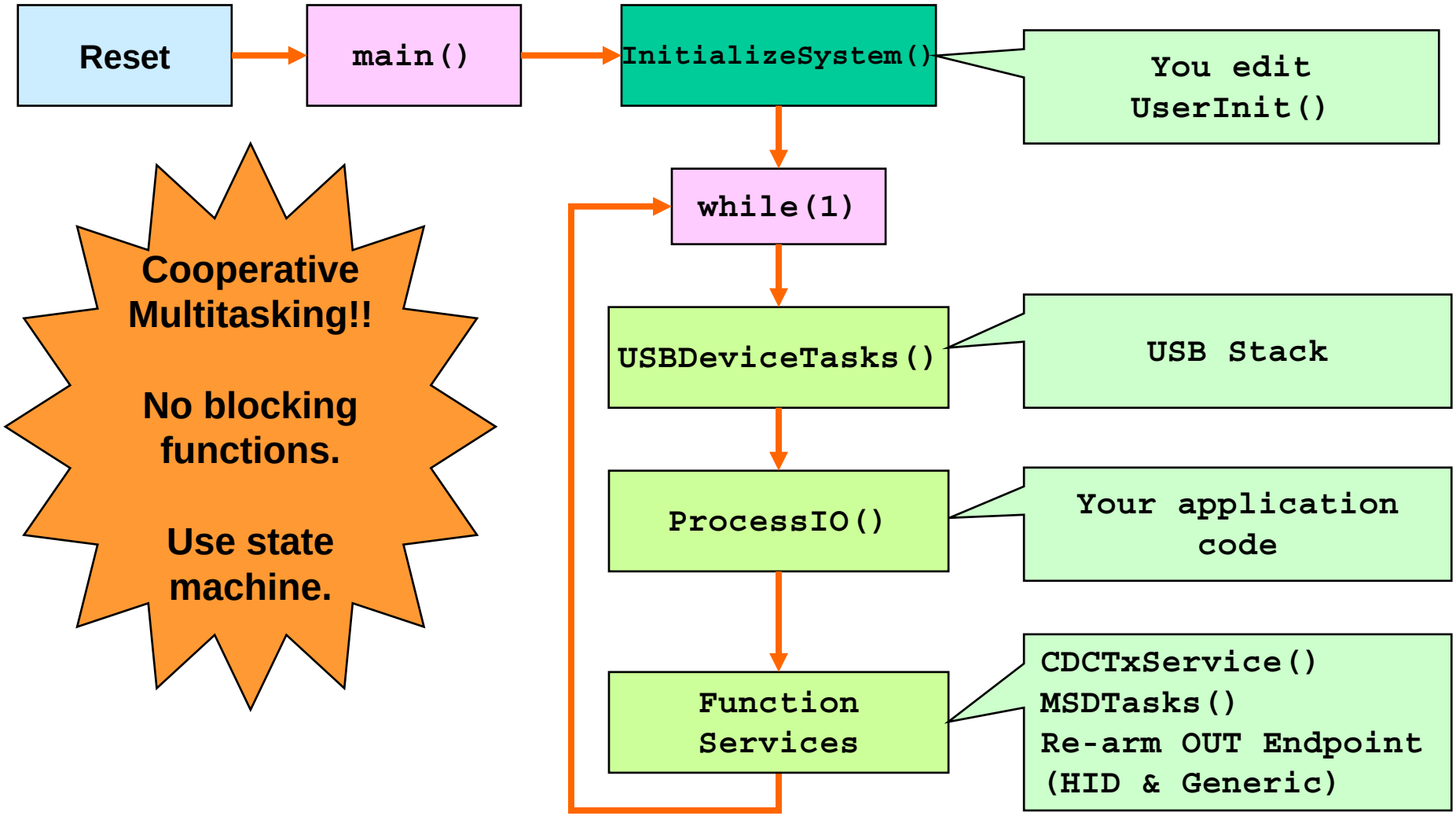
MICROCHIP
Regional Training Centers

Lab Exercise 1a

- Objectives
  - Use USB Configuration Tool to configure the MCHPFSUSB Device Framework
  - Create a new CDC Device project from scratch
  - Put your name as Product string
  - Complete the project with missing code as described in the Lab Manual

MCHPFSUSB Framework

- Polled Program Flow -



MICROCHIP
Regional Training Centers

Code Example

Main.c

```
#include "Compiler.h"
#include "USB\usb.h"
#include "USB\usb_function_cdc.h"
#include "HardwareProfile.h"

void UserInit(void){
    …
    …
}

void ProcessIO(void){
    if((USBDeviceState < CONFIGURED_STATE)||(USBSuspendControl==1)) return;
    …
    …
    CDCTxService();
}

static void InitializeSystem(void){
    #if define …
    #endif
    UserInit();
    USBDeviceInit();
}

int main(void){
    InitializeSystem();
    while(1){
        USBDeviceTasks();
        ProcessIO();
    }
}
```

Needed (`usb_config.h` is called by `usb.h`)

Put your initialization code here

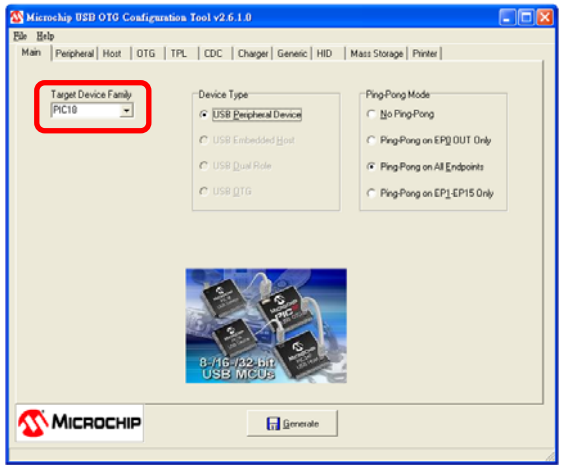
Put your application code (state machine) here

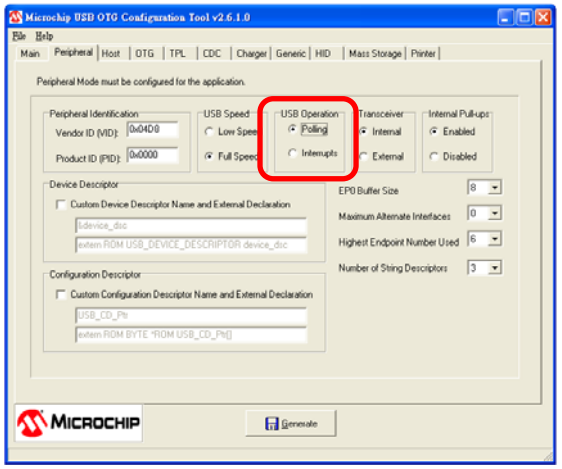
Conditional compiling (no need to change)

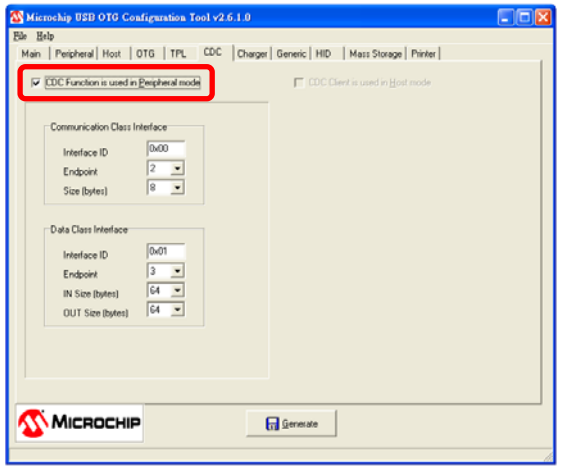
No need to change

Lab Exercise 1a-1

- **Use USB Configuration Tool to create:**
  "usb_config.c" and "usb_config.h"
  Page Main:PIC18
  Page Peripheral:Polling
  Page CDC:Enable CDC Function is ….
- Click Generate







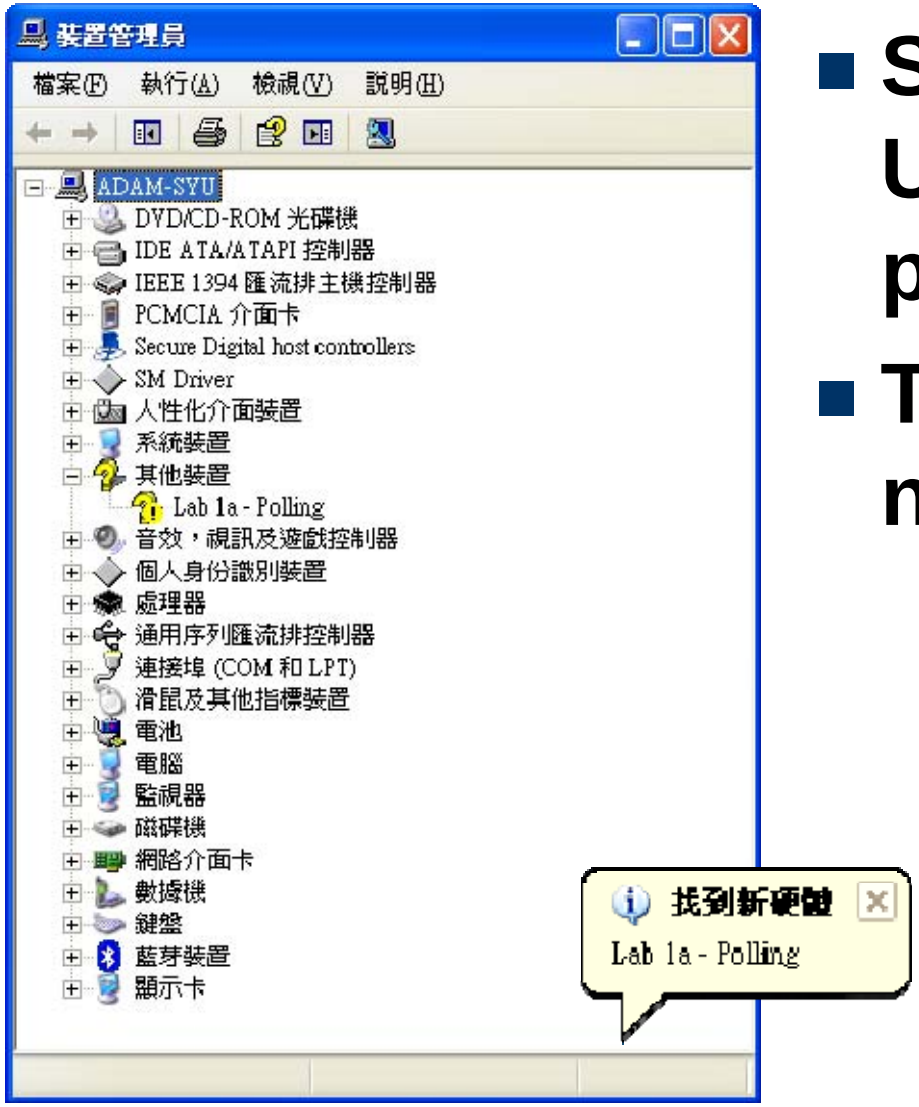
MICROCHIP
Regional Training Centers

Lab Exercise 1a-2

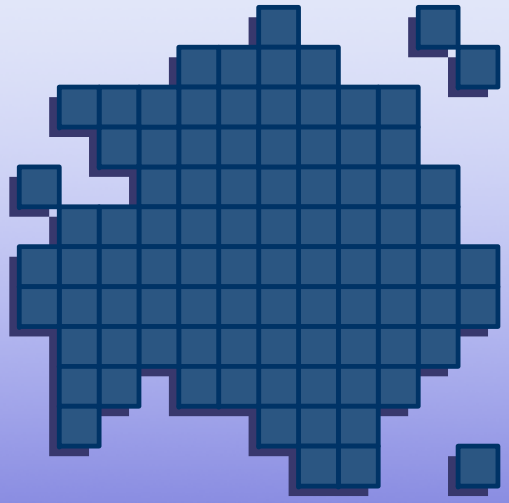
- **Use MPLAB IDE project wizard to create a project.**
  MCU:PIC18F4xJ5x, Compiler MPLAB C18.
- **Add 2 “include file path” to *Build Options***
  “.” and “..\Microchip\Include\”
- **Add 5 source code file (.c) to Project.**
  “main.c”, “usb_config.c”, “usb_descriptors.c”, Usb_device.c”, “Usb_function_cdc.c” **(Location :USB stack)**
- **Add 5 #include to main.c**
  #include "Compiler.h“
  #include "HardwareProfile.h“
  #include “usb_config.h“
  #include "USB\usb.h“
  #include "USB\usb_function_cdc.h“
- **Modify the product string descriptor to put your name.**
  (sd002, usb_descriptor.c)
- **Build & Program …**

MICROCHIP
Regional Training Centers

Lab 1a Results



- System can find a new USB device. When plug into USB port.
- The device product name is your name.



Lab Exercise 1b

Create an USB template project (Interrupts)

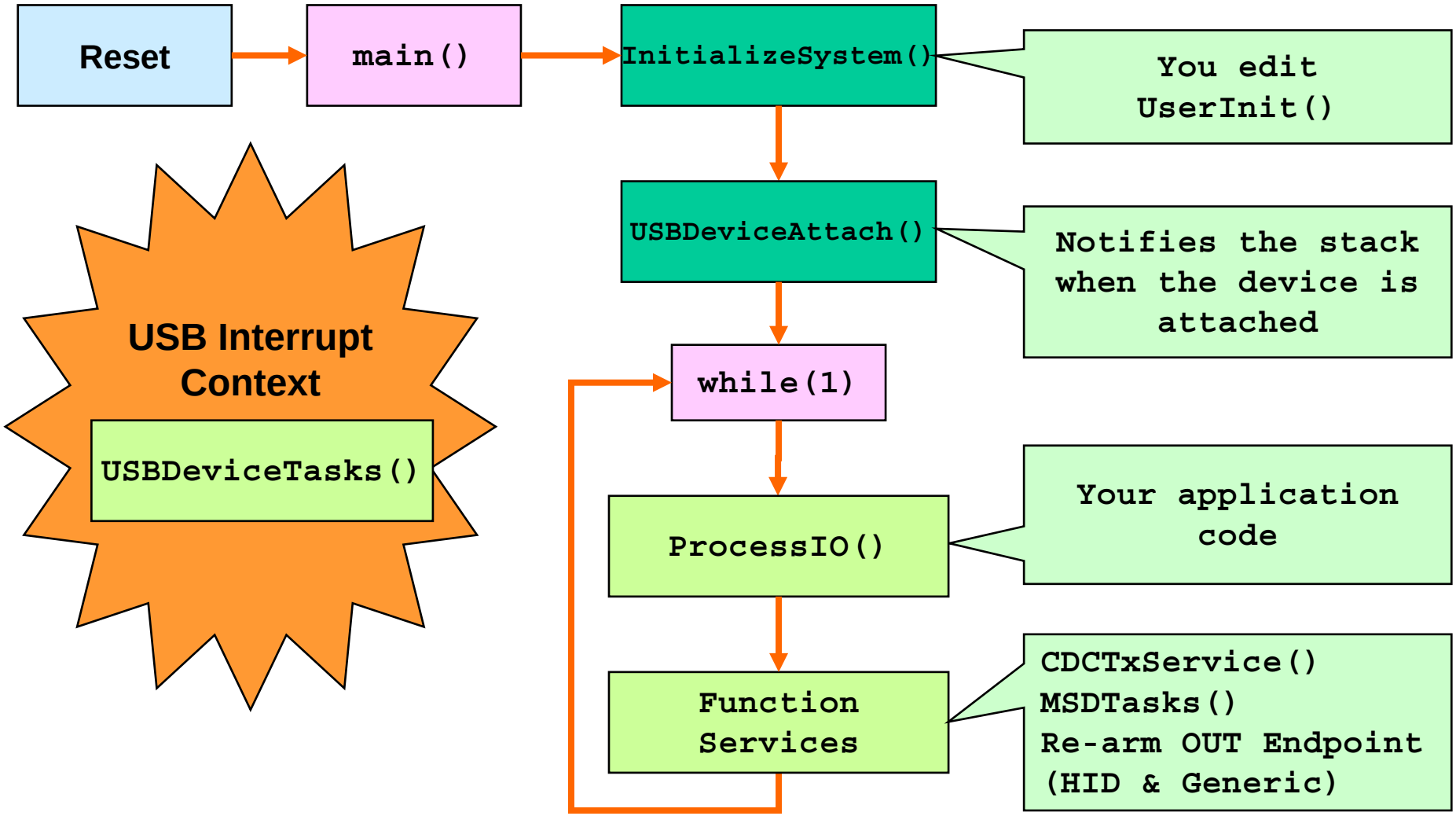
MICROCHIP
Regional Training Centers

Lab Exercise 1b

- **Objective**
  - **Modify the solution of Lab 1a to implement interrupts USB operations**

MCHPFSUSB Framework

- Interrupt Program Flow -



MICROCHIP
Regional Training Centers

Code Example

Main.c

```
#include "Compiler.h"
#include "USB\usb.h"
#include "USB\usb_function_cdc.h"
#include "HardwareProfile.h"

void UserInit(void){
    …
    …
}

void ProcessIO(void){
    if((USBDeviceState < CONFIGURED_STATE)||(USBSuspendControl==1)) return;
    …
    …
    CDCTxService();
}

static void InitializeSystem(void){
    #if define …
    #endif
    UserInit();
    USBDeviceInit();
}

int main(void){
    InitializeSystem();
    USBDeviceAttach();
    while(1){
        ProcessIO();
    }
}
```

Needed (`usb_config.h` is called by `usb.h`)

Put your initialization code here

Put your application code (state machine) here

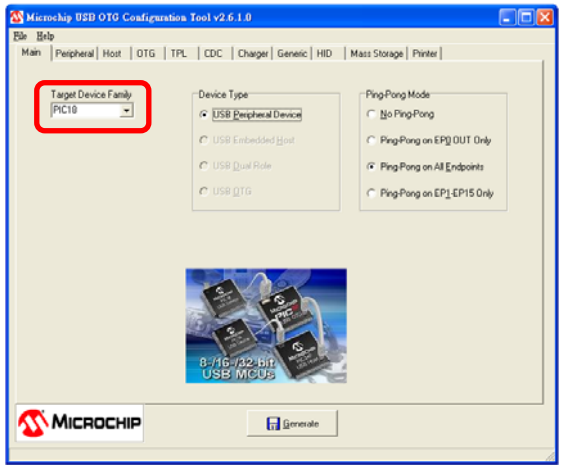
Conditional compiling (no need to change)

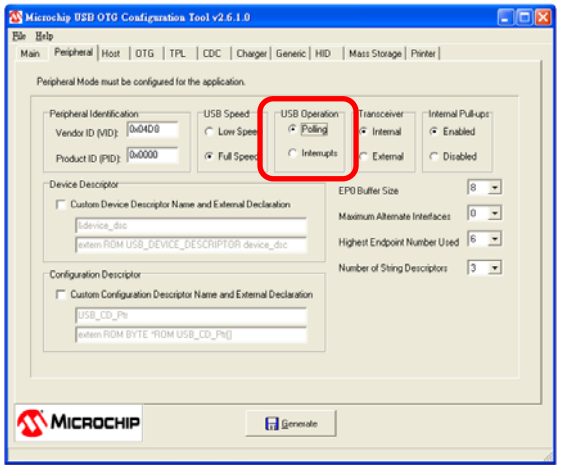
No need to change

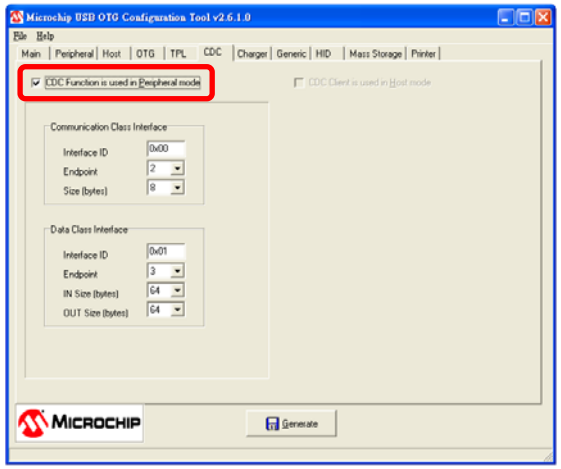
`USBDeviceTasks()` is executed in an ISR (High Priority PIC18, `_USB1Interrupt()` PIC24 & PIC32)

Lab Exercise 1b-1

- **Use USB Configuration Tool to create:**
  "usb_config.c" and "usb_config.h"
  Page Main:PIC18
  Page Peripheral:PID->0x000A, Interrupt
  Page CDC:Enable CDC Function is ….
- Click Generate







Lab Exercise 1b-2

- Use MPLAB IDE Open the project
- Modify and add some code to main.c .

```
void main( )
{
    ...
#if defined(USB_INTERRUPT)
        if(USB_BUS_SENSE && (USBGetDeviceState() ==
                DETACHED_STATE))
            USBDeviceAttach();
#endif
    ...
    #if defined(USB_POLLING)
    USBDeviceTasks();
    ...
}
```

- Modify the product string descriptor to put your name.
  (sd002, usb_descriptor.c)

Lab Exercise 1b-3

- **Modify and add some code to main.c .**

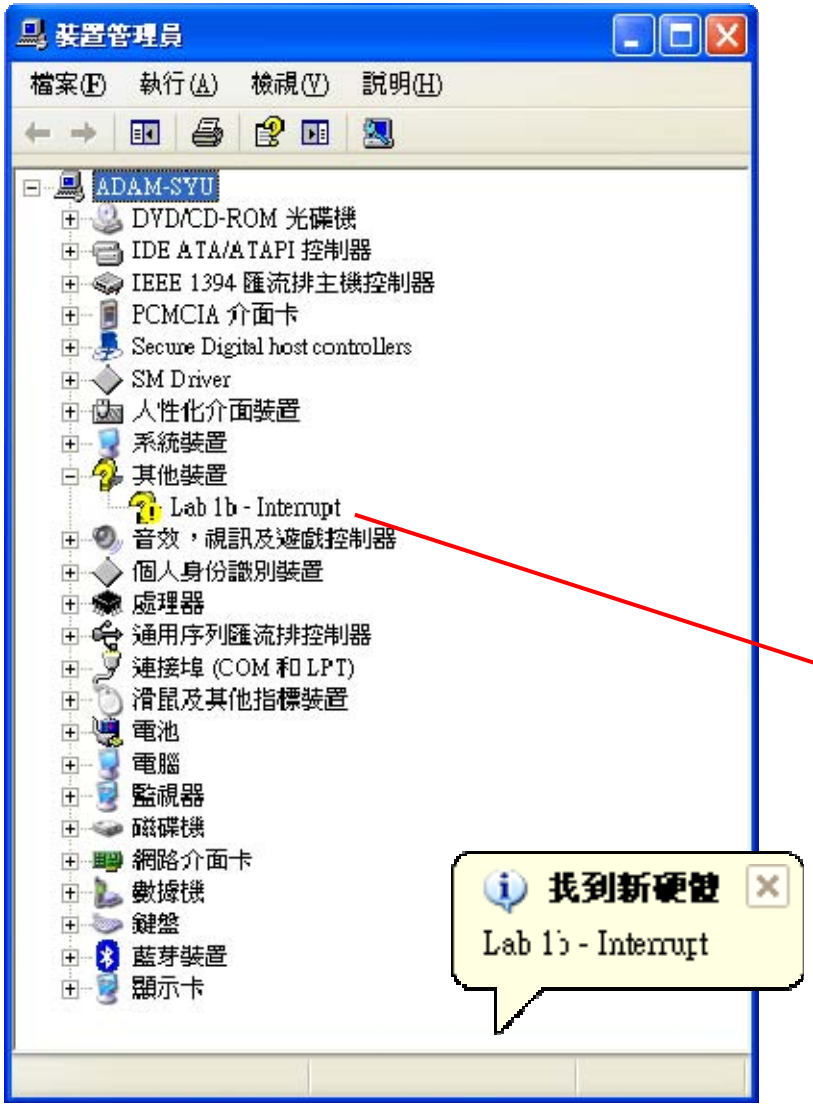
```
void ProcessIO( )
{
        if((USBDeviceState < CONFIGURED_STATE)||(USBSuspendControl==1))
                return;

        switch (mLED_1)
        {
                case 1
                        if(sw) mLED_1=0; break;
                case 0:
                        if(!sw) mLED_1=1;break;
                default:
                        mLED_1=0;
        }

        CDCTxService();
}
void UserInit( )
{
        mInitAllLEDs();
        mInitAllSwitches();
}
```

- **Build & Program …**

Lab 1b Results



- System can find a new USB device. When plug into USB port.
- The device product name is your name.
- Intstall driver

  path:\USB Tools\USB CDC Serial Demo\inf\mchpcdc.inf

- Try to press switch !!
  LED is turn On and Off.

MICROCHIP
Regional Training Centers

Agenda – Part 2

- Design Considerations
- The MCHPFSUSB Device Framework
  - Microchip Application Libraries
  - MCHPFSUSB PC Tools
  - Basic USB Device Project Structure (Polled vs. Interrupt-Driven)
    - Lab 1a – Create an USB template project (Polling)
    - Lab 1b – Create an USB template project (Interrupts)
  - Editable Files Description
    - usb_descriptors.c
    - hardware_profile.h
    - usb_config.h
  - Event Handler Function

Generic USB Project

- **General structure**

```
.\Your application
        main.c
        usb_descriptors.c
        HardwareProfile.h
        usb_config.h
.\Microchip
        \Include
        \USB
        \Common
        \…
```

Editable files

MCHPFSUSB Framework

- Editable Files -

- **usb_descriptors.c**
  - Define your device descriptors
    - VID & PID
    - Class Specific (may not need to change)
    - Strings

```
/* Device Descriptor */
ROM USB_DEVICE_DESCRIPTOR device_dsc=
{   0x12,                       // Size of this descriptor (byte)
    USB_DESCRIPTOR_DEVICE,      // DEVICE descriptor type
    0x0200,                     // USB Spec Release Number
    My_Class_code,              // Class code
    My_Subclass_code,           // Subclass code
    My_Protocol_code,           // Protocol code
    EP0_BUFF_SIZE,              // Max packet size for EP0
    My_VID,                     // Vendor ID
    My_PID,                     // Product ID
```

…

MCHPFSUSB Framework

- Editable Files -

- **HardwareProfile.h**
  - Define your board & hardware initialization routines

```
// #define USE_SELF_POWER_SENSE_IO
#define tris_self_power         TRISAbits.TRISA2    // Input
#if defined(USE_SELF_POWER_SENSE_IO)
   #define self_power           PORTAbits.RA2
#else
   #define self_power           1
#endif
//#define USE_USB_BUS_SENSE_IO
#define tris_usb_bus_sense      TRISAbits.TRISA1    // Input
#if defined(USE_USB_BUS_SENSE_IO)
   #define USB_BUS_SENSE        PORTAbits.RA1
#else
   #define USB_BUS_SENSE        1
#endif
...
```

MCHPFSUSB Framework

- Editable Files -

- **usb_config.h**
  - Define framework options (use configuration tool)
    - USB Definitions
    - Device Class Usage
    - Endpoint Allocation

```
/** DEFINITIONS ****************************************/
#define USB_EP0_BUFF_SIZE  8          // Valid Options: 8,
                                      // 16, 32, or 64 bytes.
//#define USB_POLLING
#define USB_INTERRUPT
//#define USB_SPEED_OPTION USB_LOW_SPEED //(not valid option
                                       // for PIC24F devices)
#define USB_SPEED_OPTION   USB_FULL_SPEED
#define USB_SUPPORT_DEVICE
…
/** DEVICE CLASS USAGE *********************************/
#define USB_USE_CDC
…
```

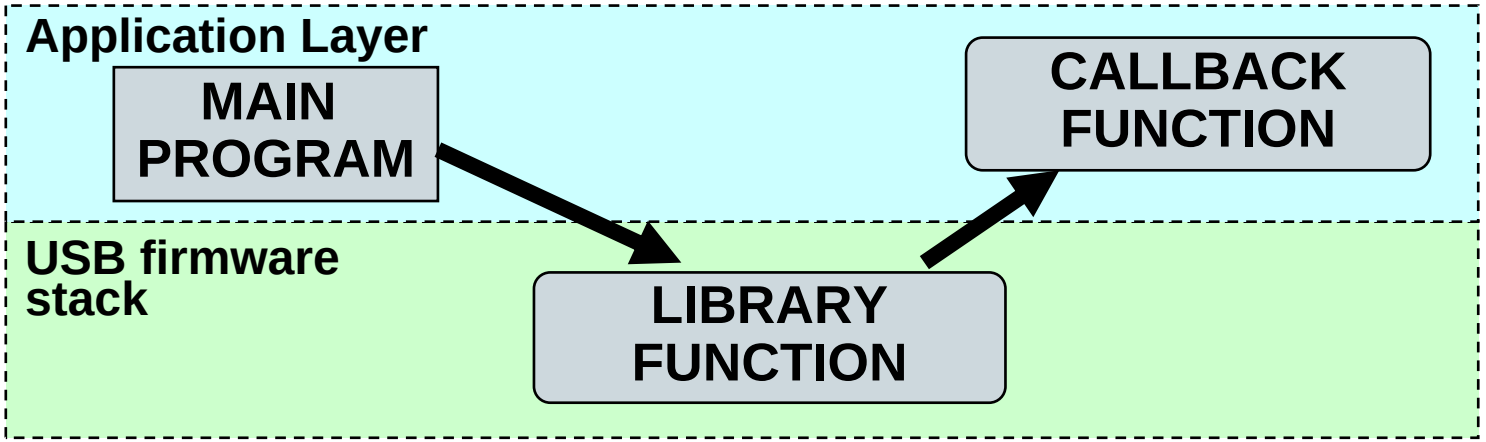
MICROCHIP
Regional Training Centers

Agenda – Part 2

- Design Considerations
- The MCHPFSUSB Device Framework
  - Microchip Application Libraries
  - MCHPFSUSB PC Tools
  - Basic USB Device Project Structure (Polled vs. Interrupt-Driven)
    - Lab 1a – Create an USB template project (Polling)
    - Lab 1b – Create an USB template project (Interrupts)
  - Editable Files Description
    - usb_descriptors.c
    - hardware_profile.h
    - usb_config.h
  - Event Handler Function

Call Back Function

- The USB firmware stack will call a callback function in response to certain USB related events



- You can modify that callback function to take appropriate actions for each of these conditions

Event Handler Function

- `BOOL USER_USB_CALLBACK_EVENT_HANDLER (USB_EVENT event, void *pdata, WORD size)`

Parameters:

- `USB_EVENT event`
  The type of event
- `void *pdata`
  Pointer to the event data
- `WORD size`
  Size of the event data

Output:

- The function should return TRUE (not used)

USB Device related events

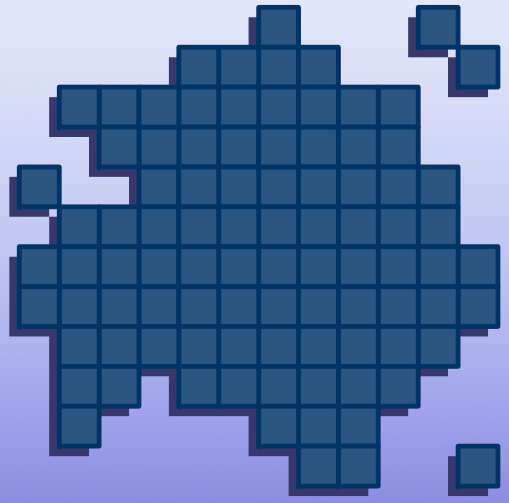
EVENT_NONE	No event occurred (NULL event).
EVENT_TRANSFER	A USB transfer has completed.
EVENT_SOF	A USB Start of Frame token has been received.
EVENT_RESUME	Device mode resume received.
EVENT_SUSPEND	Device mode suspend/idle event received.
EVENT_RESET	Device mode bus reset received.
EVENT_STALL	A stall has occurred.
EVENT_SETUP	A setup packet received (data: SETUP_PKT).
EVENT_CONFIGURED	Notification that a SET_CONFIGURATION() command was received.
EVENT_SET_DESCRIPTOR	A SET_DESCRIPTOR request was received.
EVENT_EP0_REQUEST	An endpoint 0 request was received that the stack did not know how to handle. This is most often a request for one of the class drivers. Refer to the class driver documentation for information related to what to do if this request is received.
EVENT_BUS_ERROR	There was a transfer error on the USB.

Microchip Regional Training Centers

Example

- CDC device -

```
BOOL USER_USB_CALLBACK_EVENT_HANDLER
        (USB_EVENT event, void *pdata, WORD size)
{
    switch (event)
    {
        case EVENT_CONFIGURED:
            CDCInitEP();
            break;
        case EVENT_EP0_REQUEST:
            USBCheckCDCRequest();
            break;
        default:
            break;
    }
    return TRUE;
}
```



Part 3.

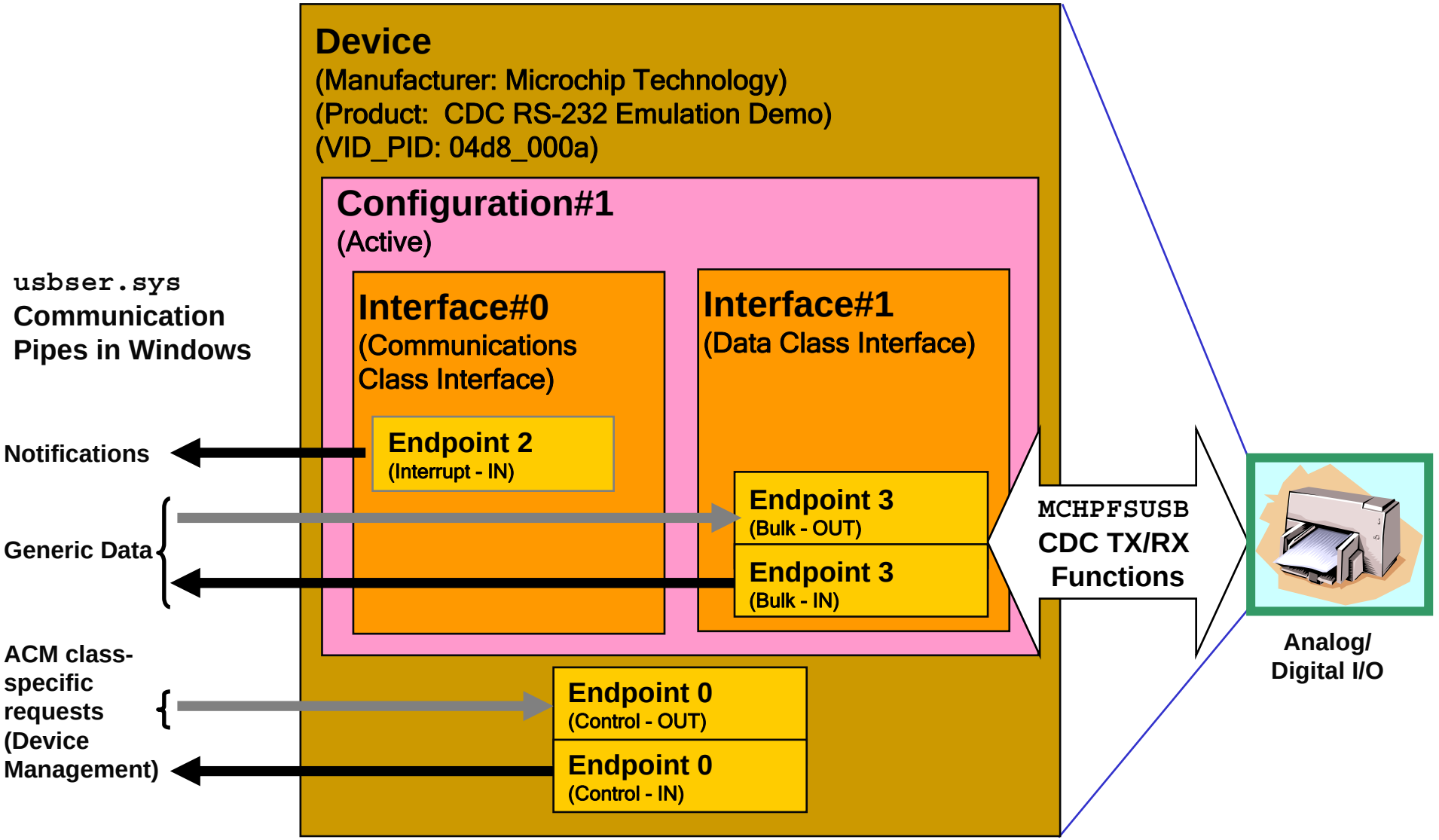
Using the Microchip CDC Class Device Framework

MICROCHIP
Regional Training Centers

Agenda – Part 3

- Overview of USB 2.0 CDC Class
- Abstract Control Model
- Microchip CDC-Class APIs
  - Lab 2 – Sending and Receiving strings
  - Lab 3 – Sending and Receiving raw data
- RS-232 Replacement (Optional)
  - Lab 4 – Migrating Applications to USB from RS-232 UART
  - Lab 5 – Adding a Serial Number String Descriptor

CDC Pipes



MICROCHIP
Regional Training Centers

Agenda – Part 3

- Overview of USB 2.0 CDC Class
- Microchip CDC-Class APIs
  - Lab 2 – Sending and Receiving strings
  - Lab 3 – Sending and Receiving raw data
- RS-232 Replacement(Optional)
  - Lab 4 – Migrating Applications to USB from RS-232 UART
  - Lab 5 – Adding a Serial Number String Descriptor

CDC RS-232 Emulation APIs

- Public API Members:

```
void putrsUSBUSART(const ROM char *data);
void putsUSBUSART(char *data);
void putUSBUSART(char *data, BYTE Length);
BYTE getsUSBUSART(char *buffer, BYTE len);
void CDCTxService(void);
void CDCInitEP(void);
```

Refer to MCHPFSUSB Library Help.chm
\Microchip Solutions\Microchip\Help\

MICROCHIP
Regional Training Centers

Importance of Checking State

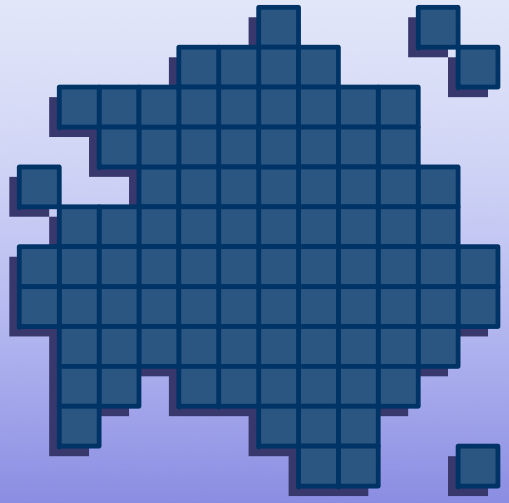
- **When calling an API that sends data to the host, must check that CDC state machine is ready to transmit, using the macro:**

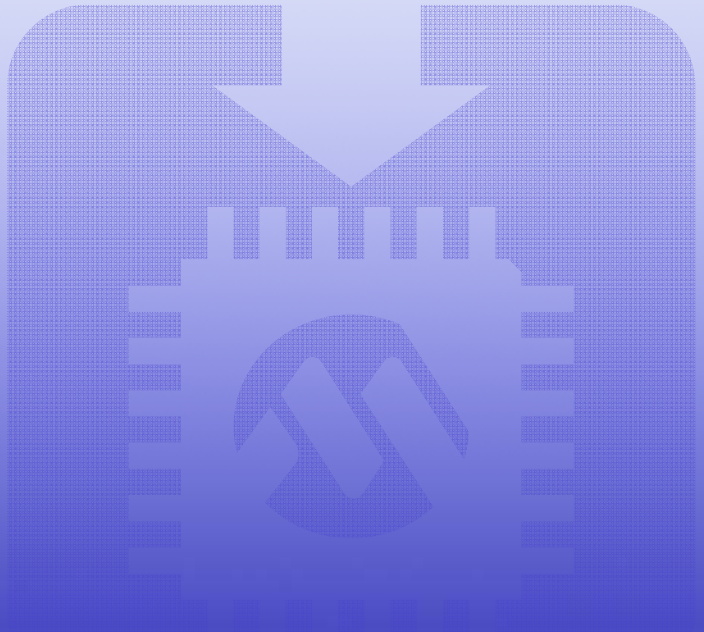
```
BOOL USBUSARTIsTxTrfReady(void);
```

Returns '1' when CDC state is `CDC_TX_READY`

- **Example:**

```
if(USBUSARTIsTxTrfReady())
{
  putrsUSBUSART("Hello World\r\n");
}
```





Lab Exercise 2

Sending and Receiving strings

Lab 2

- Sending and Receiving strings -

- **Objectives 1**
  - Write code that sends a literal null-terminated string of text (“This is Microchip USB CDC Demo!`\r\n`”) to the PC when switch is pressed
  - lab2a( )

```
{
        if (SwitchIsPressed())
        {
            … // Add your code
        }
}
```

Lab 2

- Sending and Receiving strings -

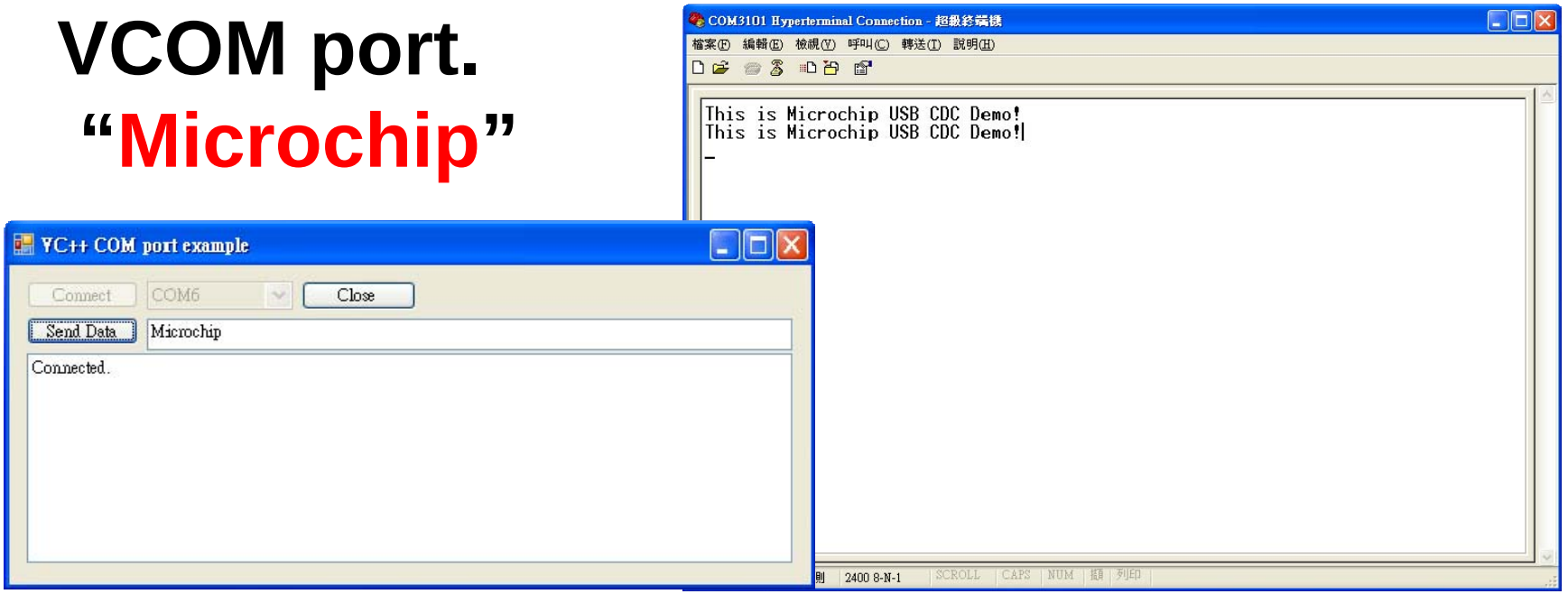
- **Objectives 2**
  - Write code that reads data from USB bus and toggles LED 1 when the char received equals ‘A’ or the string received equals “Microchip”
  - lab2b( )

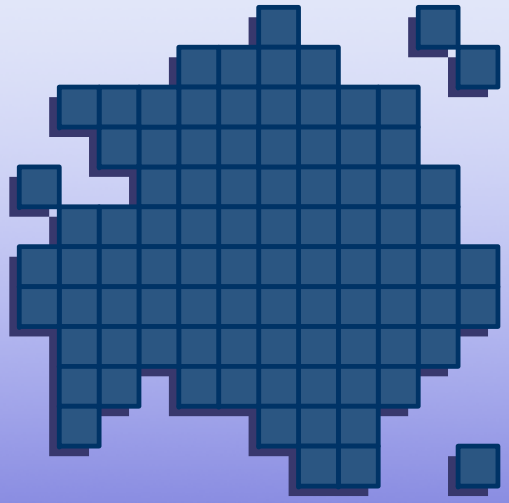
```
{
        … // Add your code
}
```

Lab 2 Results

- Hyper terminal received a string, when switch is pressed.
- LED1 is toggle, when type ‘A’ or ‘a’.
- LED1 is toggle, when send the string to VCOM port.

“Microchip”





Lab Exercise 3

Sending and Receiving raw data

MICROCHIP
Regional Training Centers

Lab 3

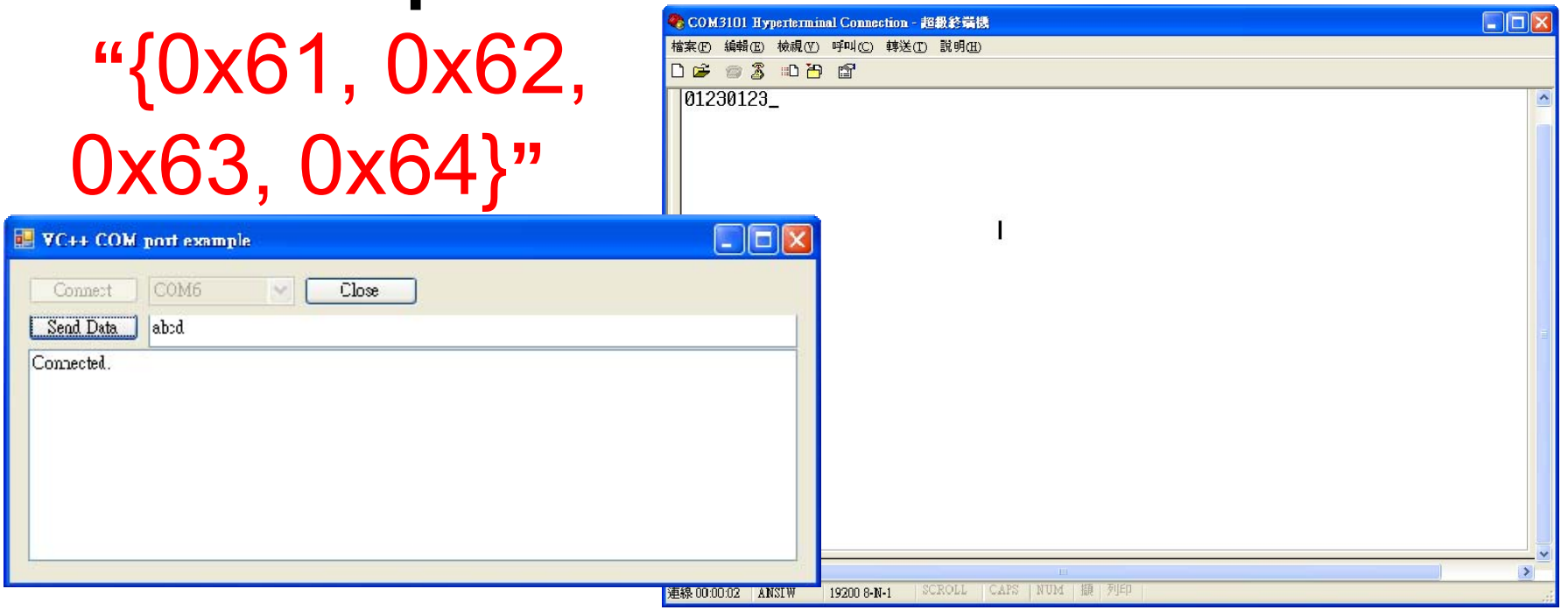
- Sending and Receiving raw data -

- **Objectives 1**
  Write code that sends the following 4 bytes of data {0x30, 0x31, 0x32, 0x33} when switch is pressed
- lab3a( )
- **Objectives 2**
  Write code that reads data from USB bus and toggles LED 1 when the data received equals {0x61, 0x62, 0x63, 0x64}
- lab3b( )

Lab 3 Results

- Hyper terminal received a raw data, when switch is pressed.
- LED1 is toggle, when send the raw data to VCOM port.

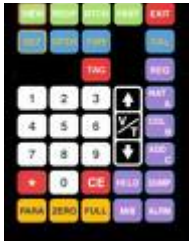
"{0x61, 0x62, 0x63, 0x64}"

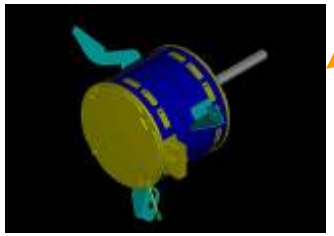


MICROCHIP
Regional Training Centers

Typical PIC® MCU Application

PIC MCU based Application





A/D, PWM, SPI, I²C™, I/O, etc.

New Communications via USB

Old Communications via RS232

MICROCHIP
Regional Training Centers

Trademarks

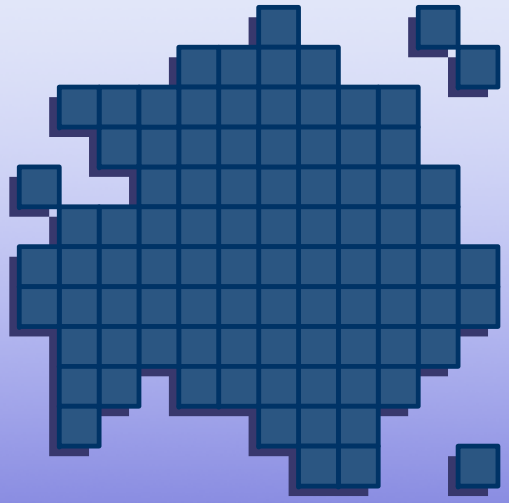
The Microchip name and logo, the Microchip logo, dsPIC, KeeLoq, KeeLoq logo, MPLAB, PIC, PICmicro, PICSTART, PIC^{32} logo, rfPIC and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

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.

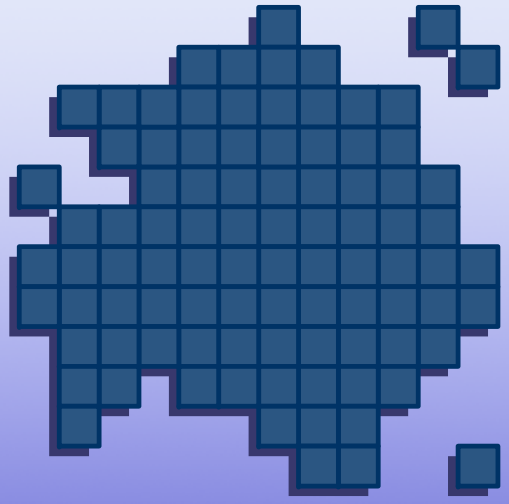
Analog-for-the-Digital Age, Application Maestro, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, dsSPEAK, ECAN, ECONOMONITOR, FanSense, HI-TIDE, In-Circuit Serial Programming, ICSP, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, mTouch, Octopus, Omniscient Code Generation, PICC, PICC-18, PICDEM, 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.

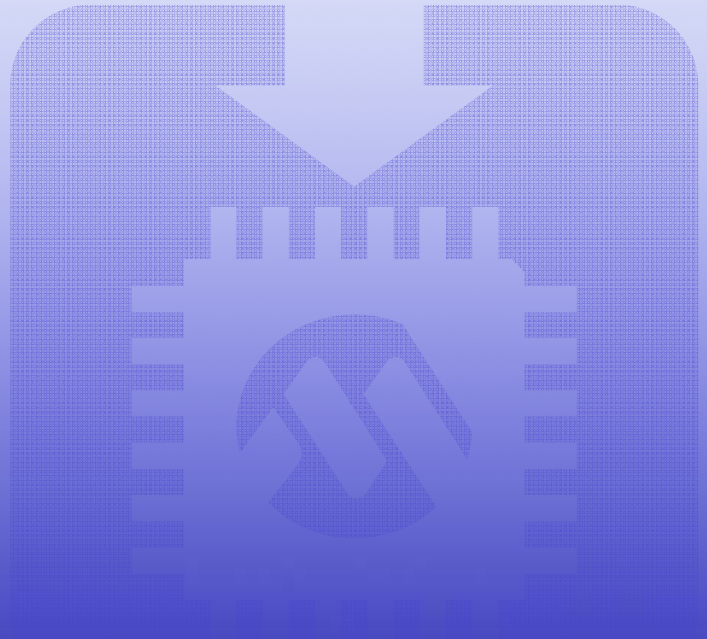
SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

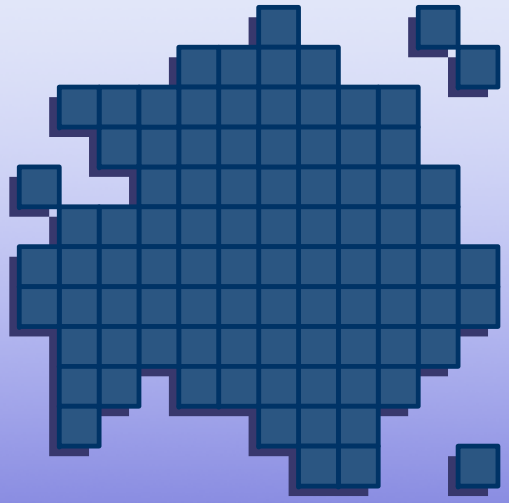


Thank You.





Reference Lab and Exerxise



Lab Exercise 4

Migrating Applications to USB from RS-232 UART

Lab 4

- Migrating Applications to USB from RS-232 UART -

- **Objective**
  - Apply the CDC Class RS-232 Replacement APIs to migrate an existing RS-232 UART application to use USB
- **Application**
  - Perform a temperature measurement and send the reading (in ASCII) to Tera Term or HyperTerminal when switch sw is pressed on the board.

Lab 4

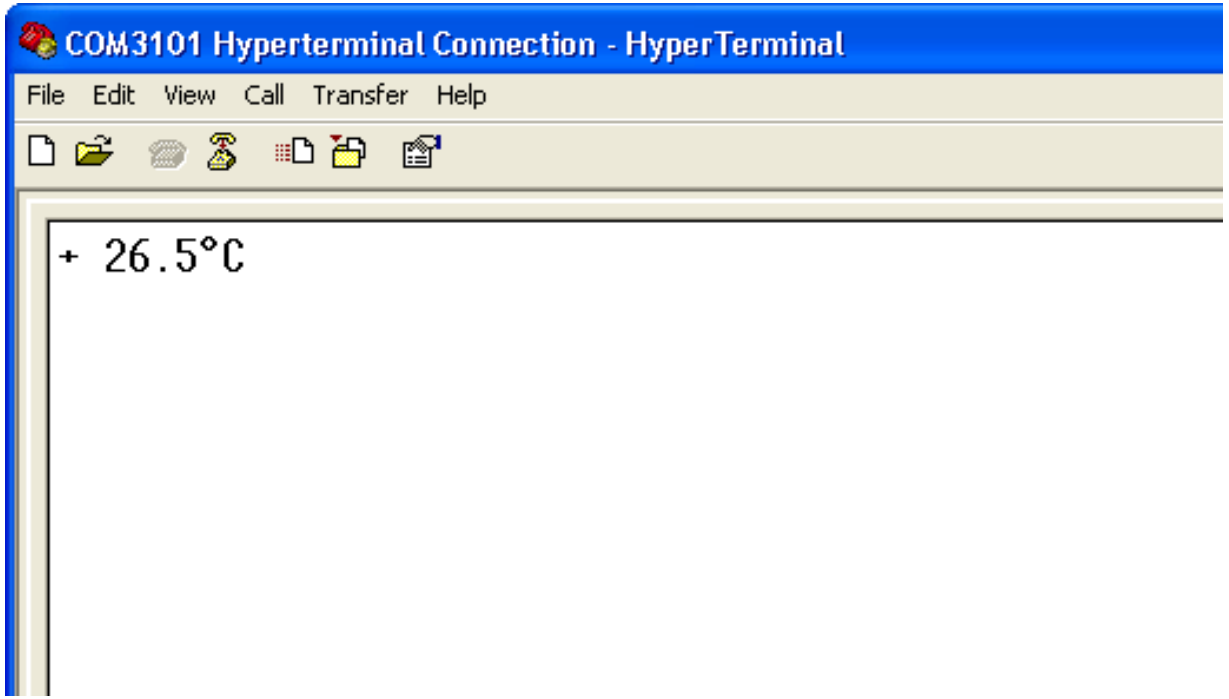
- Migrating Applications to USB from RS-232 UART -

- **Temperature Sensor ICs Used:**
  - TC72 Digital Temperature Sensor PICtail™ Demo Board
    - **TC72 (U1) – Digital Output (SPI)**
  - PICDEM FS USB
    - **TC77 (U4) – Digital Output (SPI)**
  - Explorer 16
    - **TC1047 (U4) – Analog Output**

Lab 4

- Expected Result -

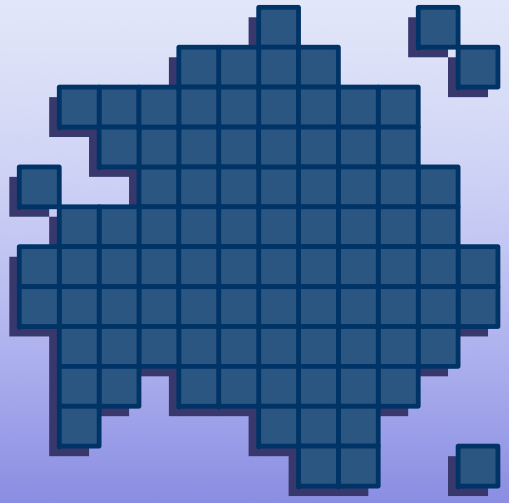
- **Pressing switch sw on the demo board, you should see the message below…**



MICROCHIP
Regional Training Centers

CDC & COM Port Numbering

- **Prevent unwanted "COM-Port proliferation"**
  - A device that contains a serial number string retains the assigned Virtual COM Port number even if moved to a different USB Port on the system



Lab Exercise 5

Adding a Serial Number String Descriptor

Lab 5

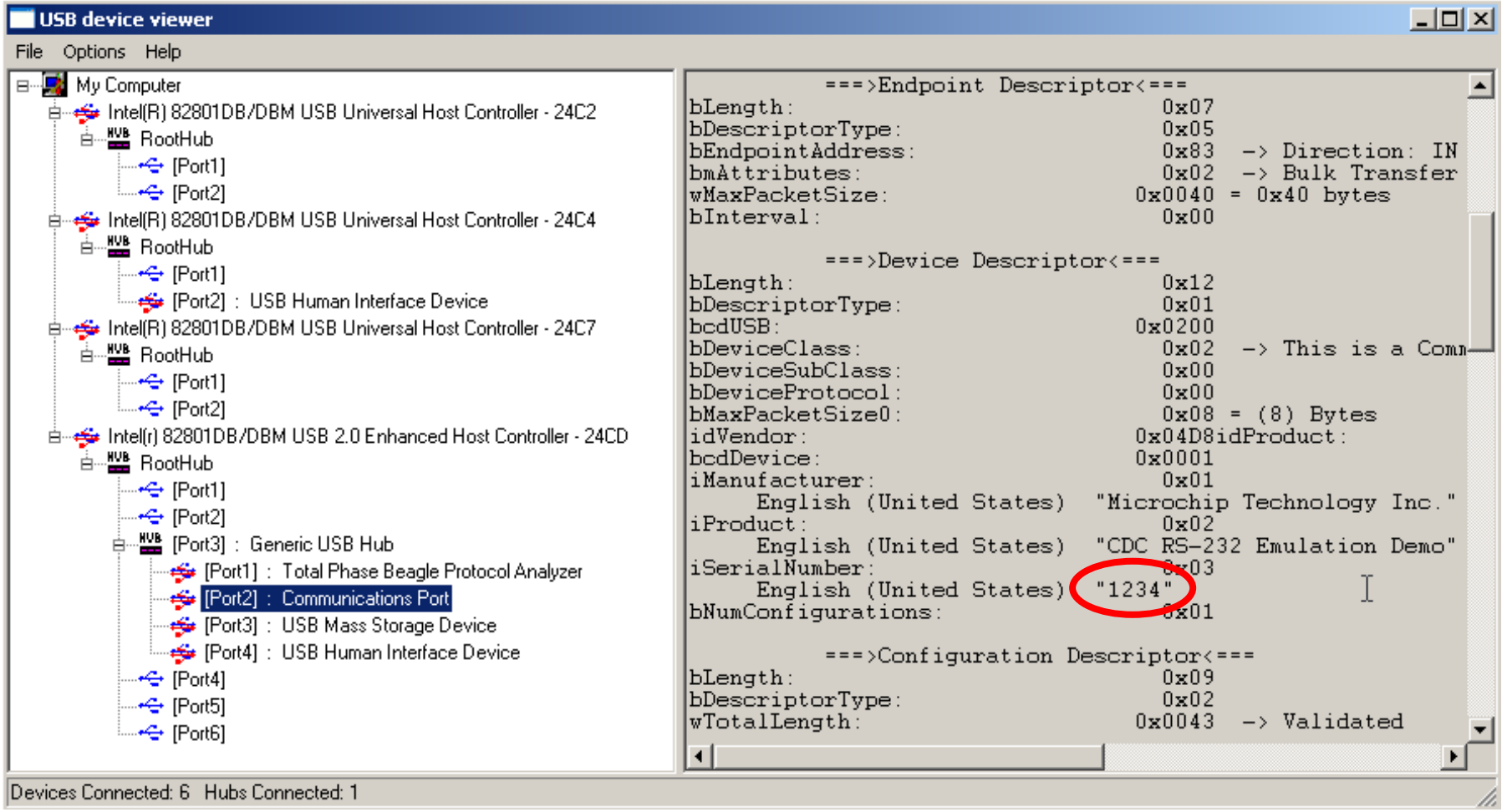
- Adding a Serial Number String Descriptor -

- **Objectives**
  - Add a serial number string descriptor to the USB device descriptor table
  - Observe the effect (no more COM port proliferation)

MICROCHIP
Regional Training Centers

Lab 5

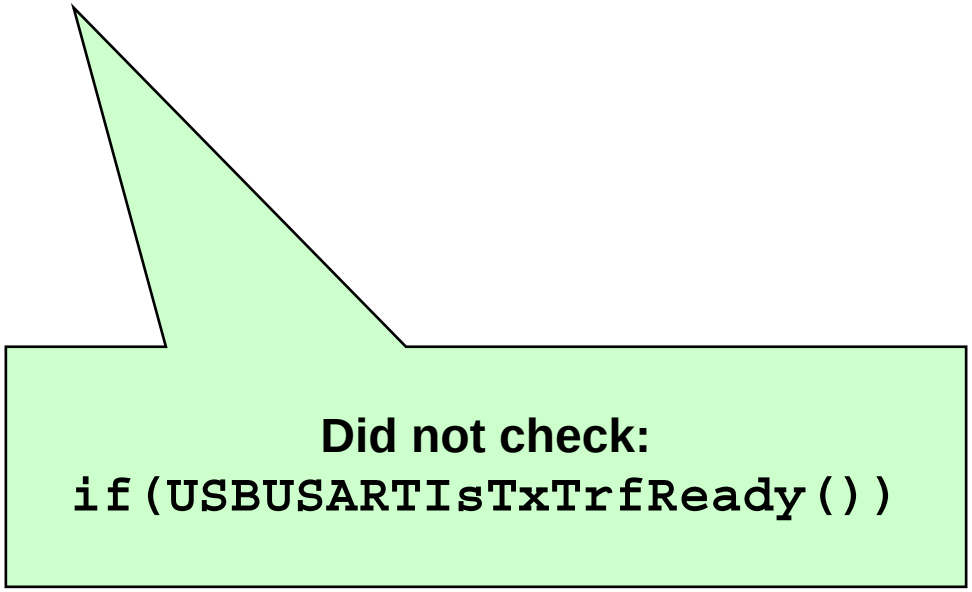
- Expected Result -



MICROCHIP
Regional Training Centers

What is Wrong with This Code?

```
void Exercise_XX(void)
{
   putrsUSBUSART("Hello World!");
}
```



MICROCHIP
Regional Training Centers

What is Wrong with This Code?

```
void Exercise_XX(void)
{
   while(!USBUSARTIsTxTrfReady());

   putrsUSBUSART("Hello World!");
}
```

Answer:
Blocking function! Not good for cooperative multitasking.

CDCTxService() in USBTasks() will never get called, and cdc_trf_state will never be updated.

Program will just be stuck in a loop.

Remember, use a state machine!

CDC Labs

- Post Exercise Analysis -

- **Benefits of Using CDC Class:**
  - Reuse existing OS driver (Windows®, Linux, MAC)
  - Almost plug-n-play in Windows – need .inf file
- **Speed:**
  - 640 Kbits/s = 80 Kbytes/s
  - Faster than RS-232 (UART)
  - Full-Speed USB device (low-speed USB device does not have bulk endpoint)
- **No hardware handshakes**

CDC Labs

- Post Exercise Analysis -

- **Program Memory Usage: ~ 6 KB**
- **Application Note**
  - AN956: Migrating Applications to USB from RS-232 UART with Minimal Impact on PC Software
- **What to do if need faster data transfer rate?**
  - Consider using the custom class Windows® drivers available and device framework…