MICROCHIP

Regional Training Centers

COM 3101T v1.1.1

Introduction to Full-Speed USB

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

Class Objectives

MICROCHIP
Regional Training Centers

- ## 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

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)

COM 3101

Class Folders

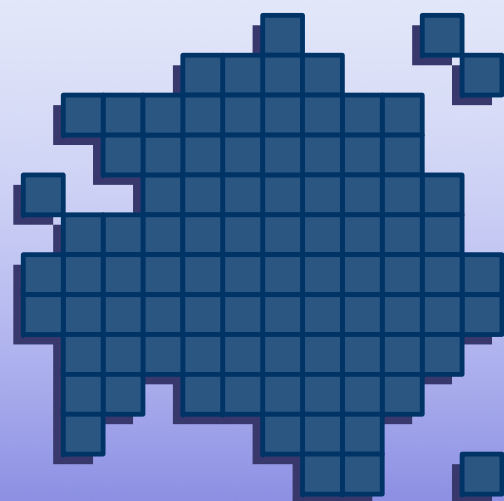MCHPFSUSB Framework is installed in
C:\Microchip_Solutions_v2013-xx-xx

COM3101T

\Exercise

Lab1a, Lab1b, ...,
Lab Serial Emulator

\USB Precompiled Demos (*.hex)

\USB Tools

Presentation Files

Part 1.

Introduction to Full-Speed USB

Agenda – Part 1

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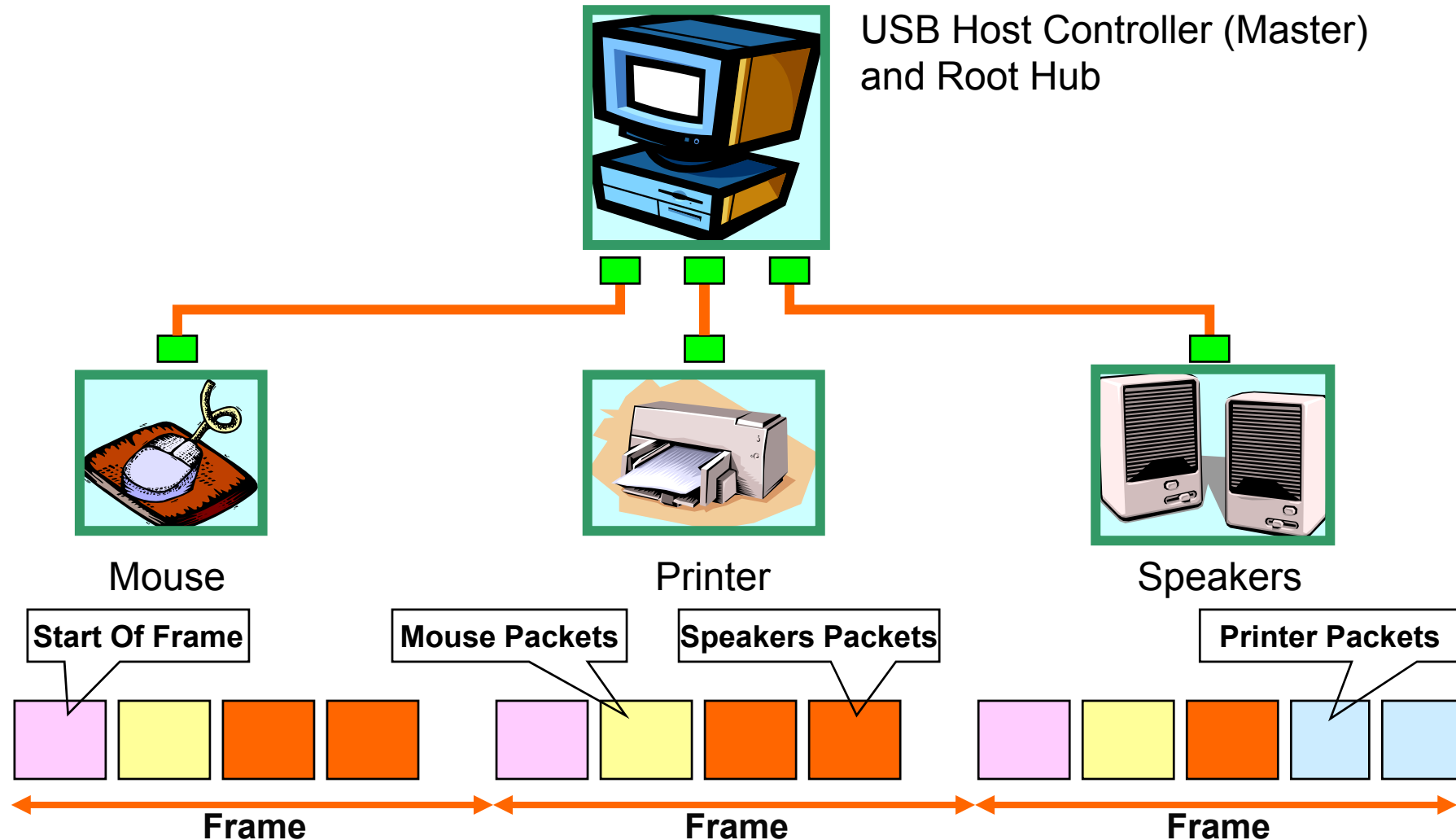
- **USB Fundamentals – The serious & important stuff**
  - **Basics/Speeds**
  - **Topology/Physical Connection**
  - **Architecture/Programmer's Model**
  - **USB Transactions**
  - **USB Transfers**
  - **Device Classes**
  - **Enumeration**
  - **Descriptors**
  - **Power Planning**
  - **VID/PID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

COM 3101

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)**
- **2008 – USB 3.0**

COM 3101

USB Basics

■ **USB is a "Single Master + Multiple Slaves" <u>polled bus</u>**

USB Host Controller (Master) and Root Hub

Mouse

Printer

Speakers

| Start Of Frame | | | | Mouse Packets | | Speakers Packets | | | | Printer Packets | |

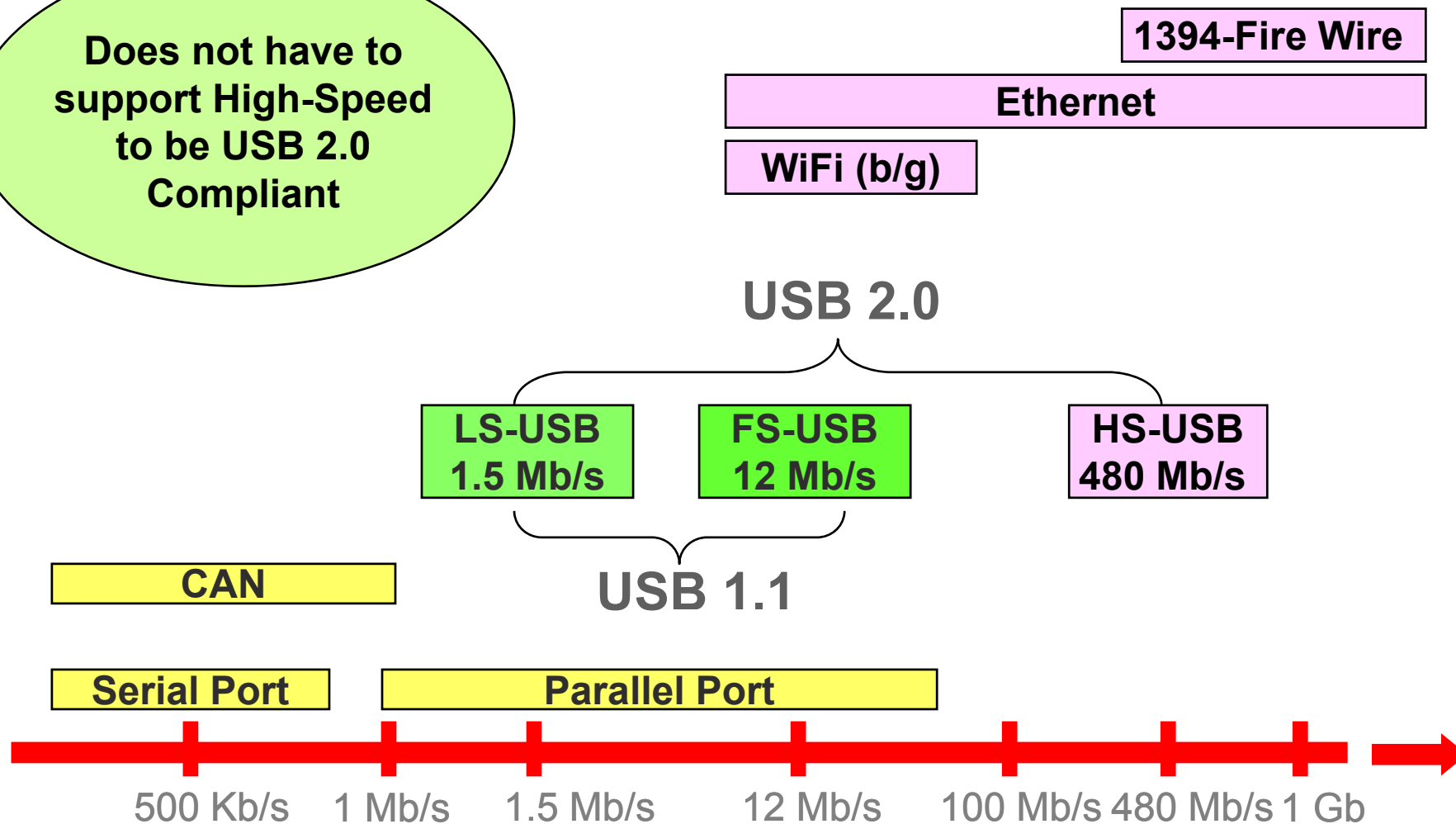**Frame**   **Frame**   **Frame**

    COM 3101

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**
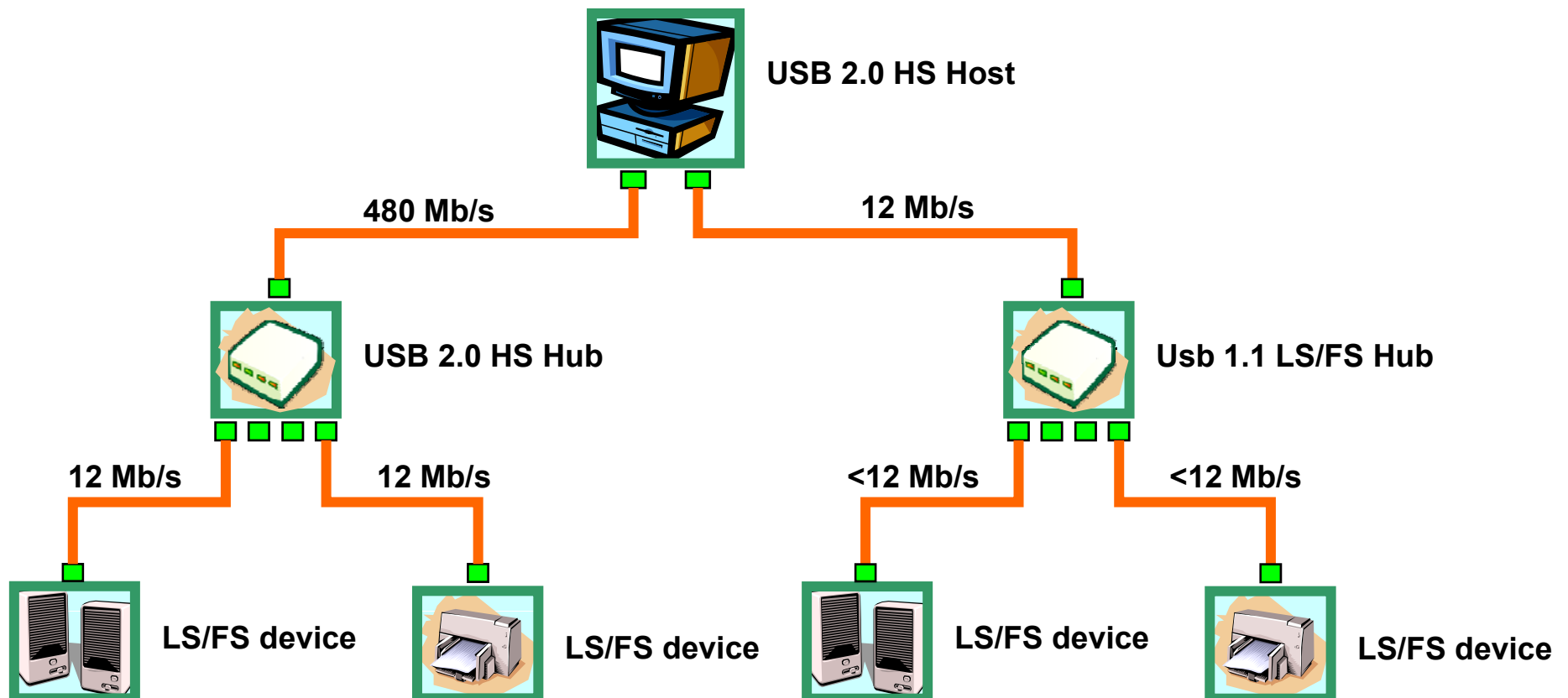   - **i.e. video camera (both audio & video interfaces active)**

COM 3101

TIP

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

USB 2.0 HS Host

480 Mb/s — 12 Mb/s

USB 2.0 HS Hub — Usb 1.1 LS/FS Hub

12 Mb/s — 12 Mb/s — <12 Mb/s — <12 Mb/s

LS/FS device — LS/FS device — LS/FS device — LS/FS device

COM 3101

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**
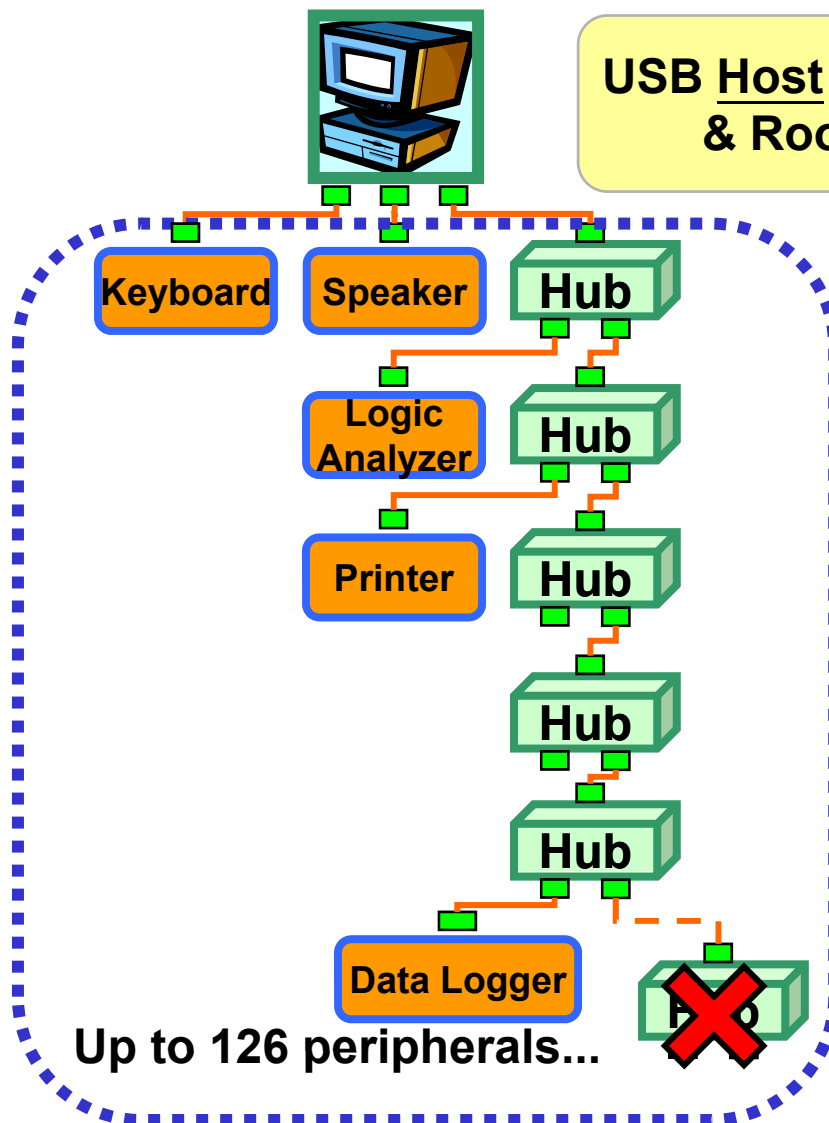
Physical Bus Topology

Host (Tier 1)

USB Host Controller & Root Hub

Tier 2 — Keyboard | Speaker | **Hub**

Hub: Max Chaining = 5

Tier 3 — Logic Analyzer | **Hub**

Tier 4 — Printer | **Hub**

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

Tier 5 — **Hub**

Tier 6 — **Hub**

Tier 7 — Data Logger | Hub ✕

Up to 126 peripherals...

MICROCHIP
Regional Training Centers

Physical Interface

V_{BUS} — V_{BUS}   ~ 5.0 V

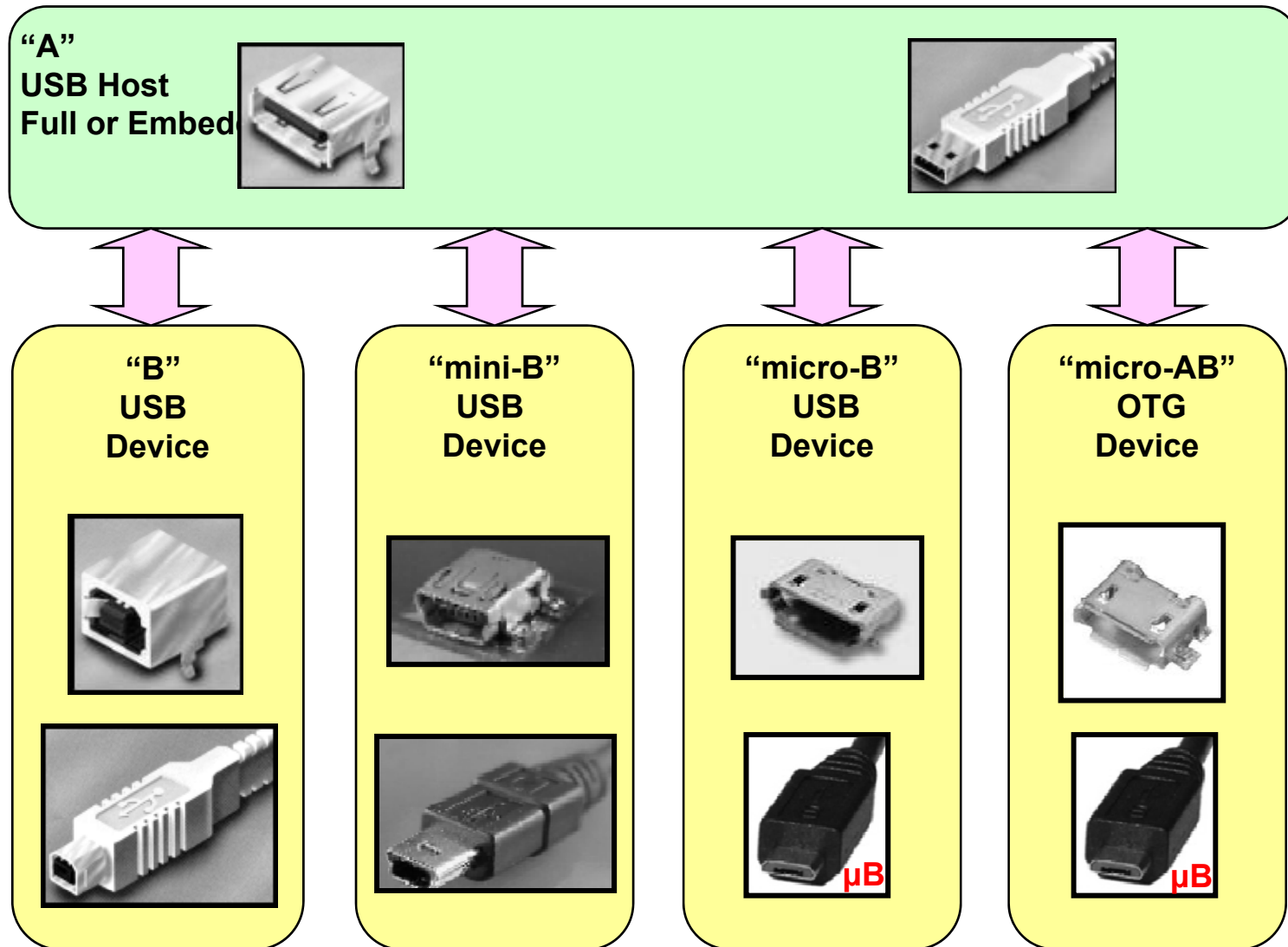D+ — D+   } ~ 3.3 V

D- — D-

GND — GND
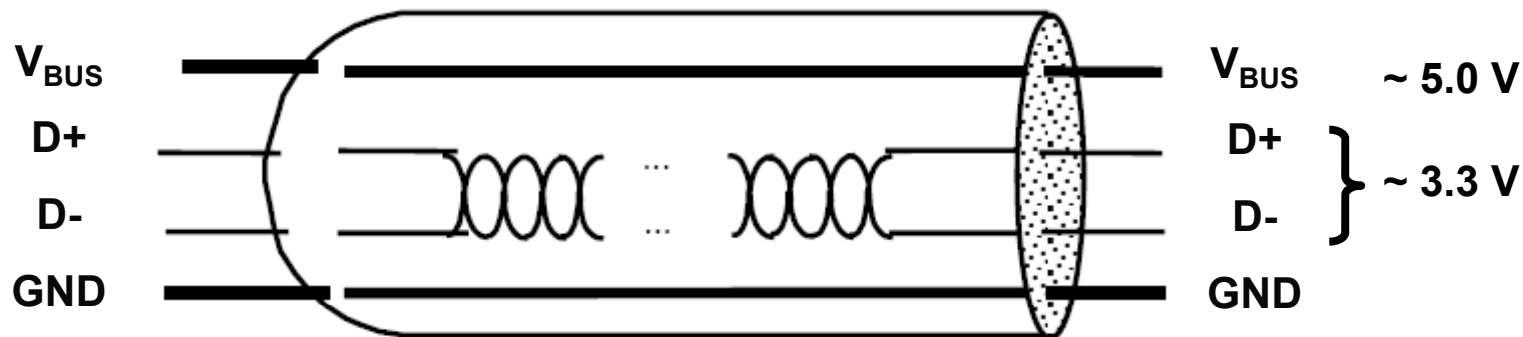
- **Half Duplex with NRZI (No Return to Zero Invert) Data Encoding**
- **Bus Power to each device:**
  - 4.40-5.25V
  - Guaranteed 100 mA
  - 500 mA maximum through negotiation
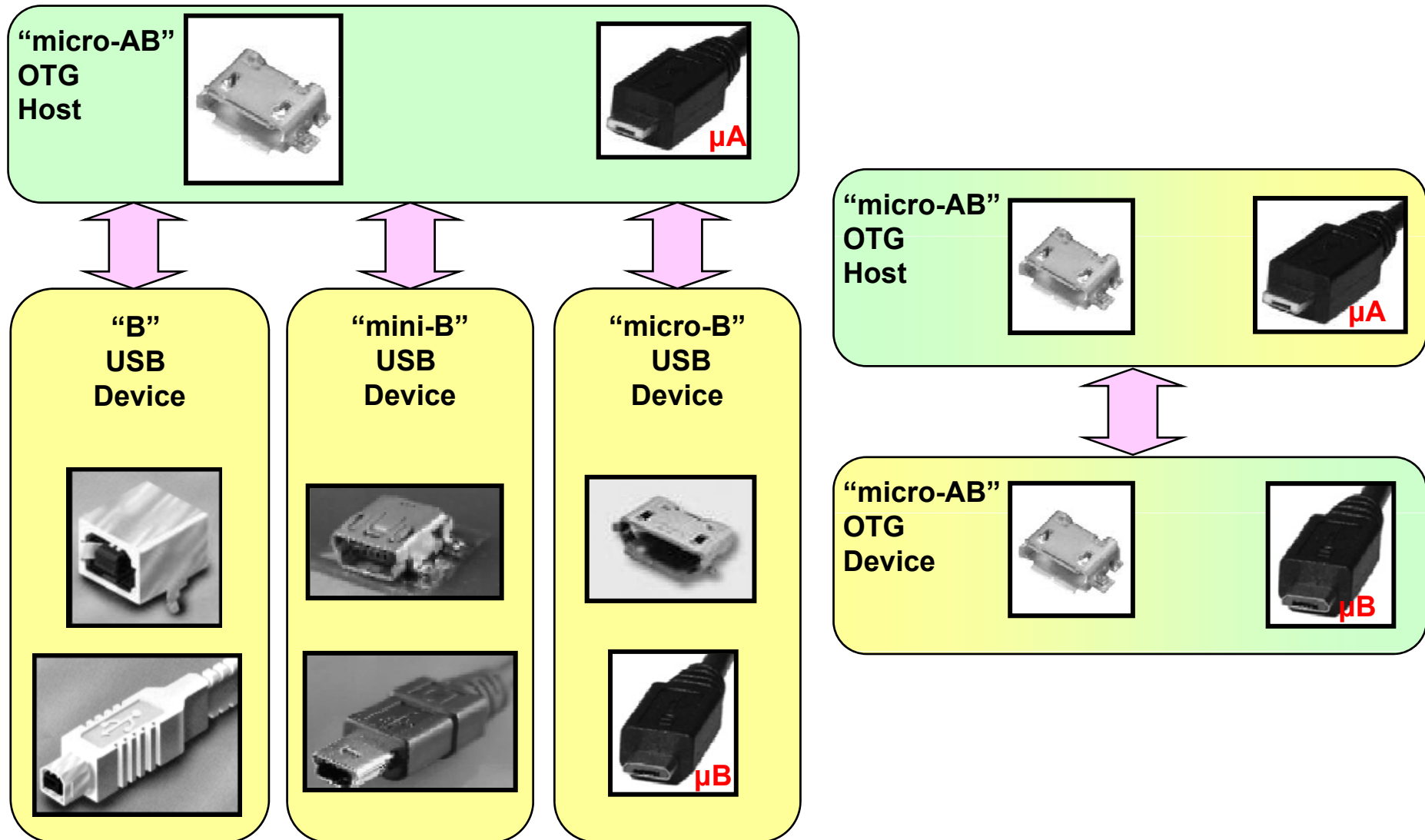
Must use external power if more is required

Standard Connectors
- USB 2.0 Specification -

"A"
USB Host
Full or Embed

"B"
USB
Device

"mini-B"
USB
Device

"micro-B"
USB
Device

"micro-AB"
OTG
Device

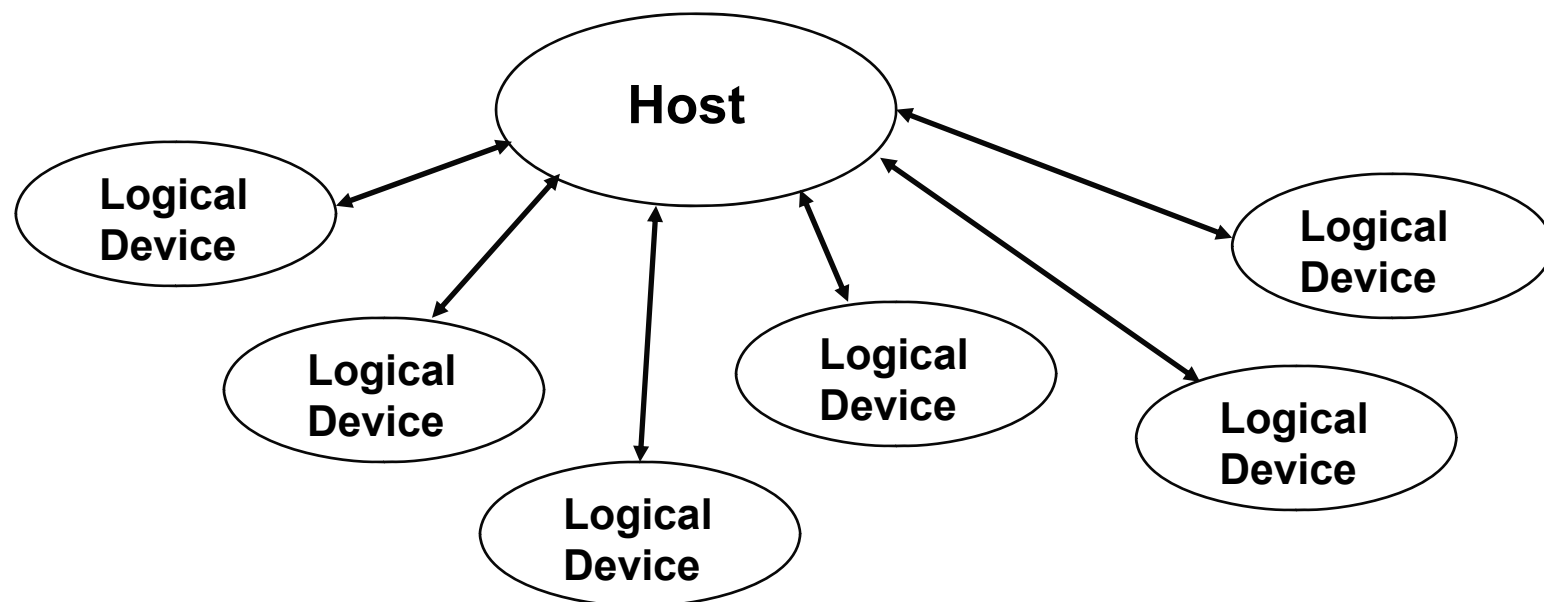# Standard Connectors
- USB 2.0 Specification -
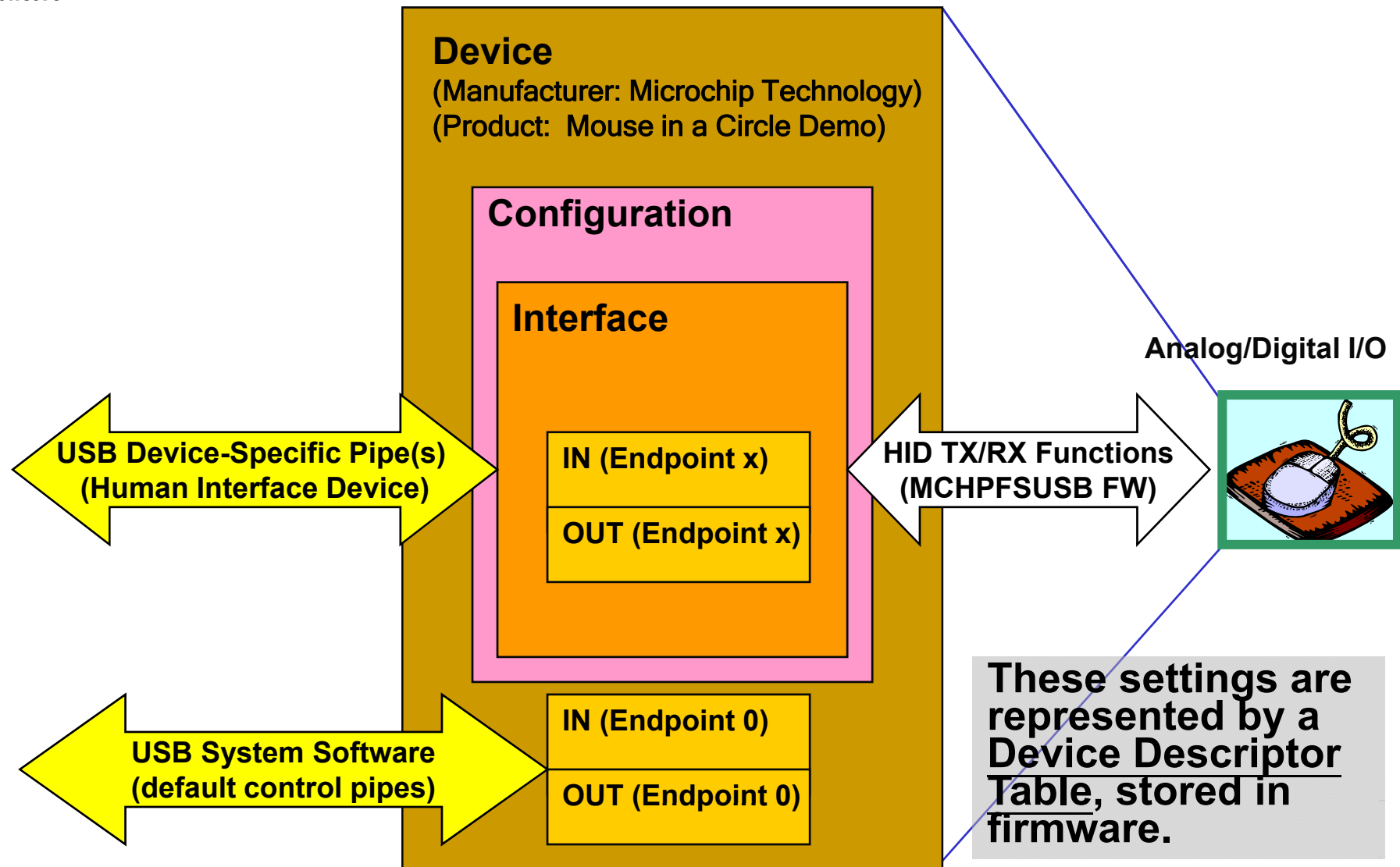
Agenda – Part 1

- **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**
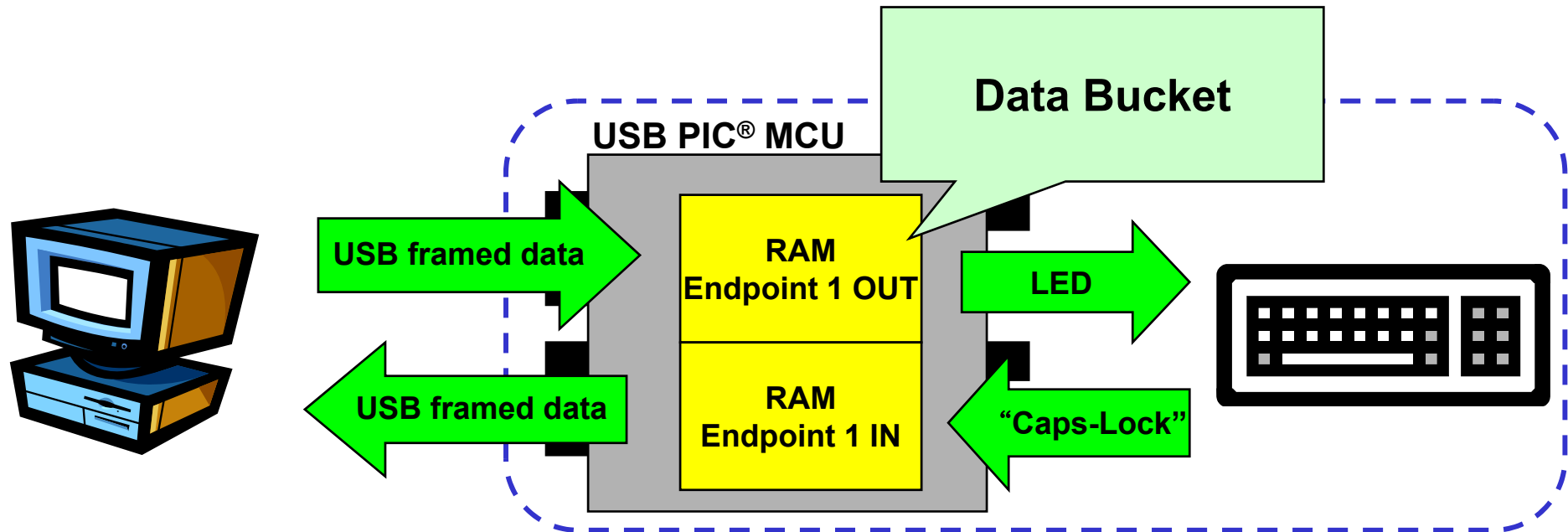- **Microchip Demo/Development Solutions**

Logical Bus Topology

- ## Not a tiered-star!

- ## Host software communicates to each "logical" device as if it were directly connected to the root hub
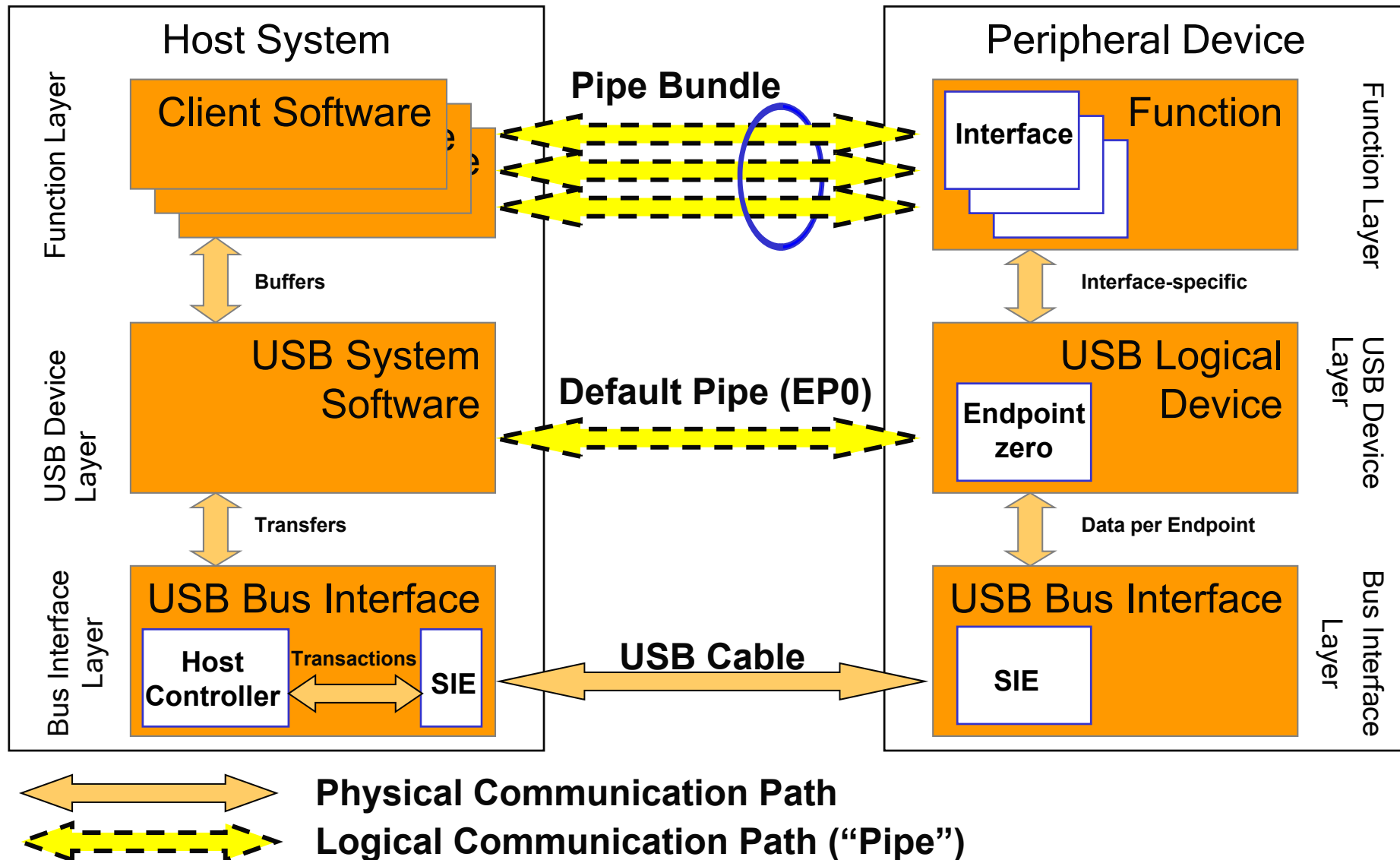
The "Logical" Device

Device
(Manufacturer: Microchip Technology)
(Product: Mouse in a Circle Demo)

Configuration

Interface

IN (Endpoint x)

OUT (Endpoint x)

IN (Endpoint 0)

OUT (Endpoint 0)

USB Device-Specific Pipe(s)
(Human Interface Device)

USB System Software
(default control pipes)

Analog/Digital I/O

HID TX/RX Functions
(MCHPFSUSB FW)

These settings are represented by a Device Descriptor Table, stored in firmware.

MICROCHIP
Regional Training Centers

Endpoints: Source/Destination of USB Data in a Peripheral

Data Bucket

USB PIC® MCU

USB framed data → RAM Endpoint 1 OUT → LED

USB framed data ← RAM Endpoint 1 IN ← "Caps-Lock"

- **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 -

Host System

Peripheral Device

Function Layer

Pipe Bundle

Client Software

Interface | Function

Buffers

Interface-specific

USB Device Layer

USB System Software

Default Pipe (EP0)

USB Logical Device

Endpoint zero

Transfers

Data per Endpoint

Bus Interface Layer

USB Bus Interface

Host Controller — Transactions — SIE

USB Cable

USB Bus Interface

SIE

Function Layer

USB Device Layer

Bus Interface Layer

Physical Communication Path

Logical Communication Path ("Pipe")

COM 3101 Slide 21

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!

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

USB Transaction

Token Packet
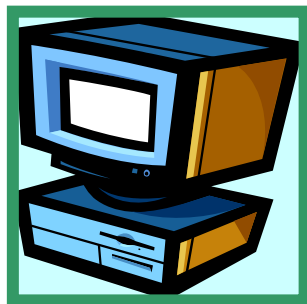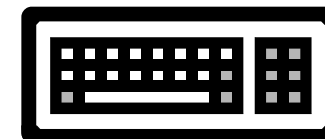
SETUP and OUT token types inform the target device that the host wants to send data.

IN token type informs the target device that the host wants to fetch data.

Specifies:
- **Target device address**
- **Endpoint number**
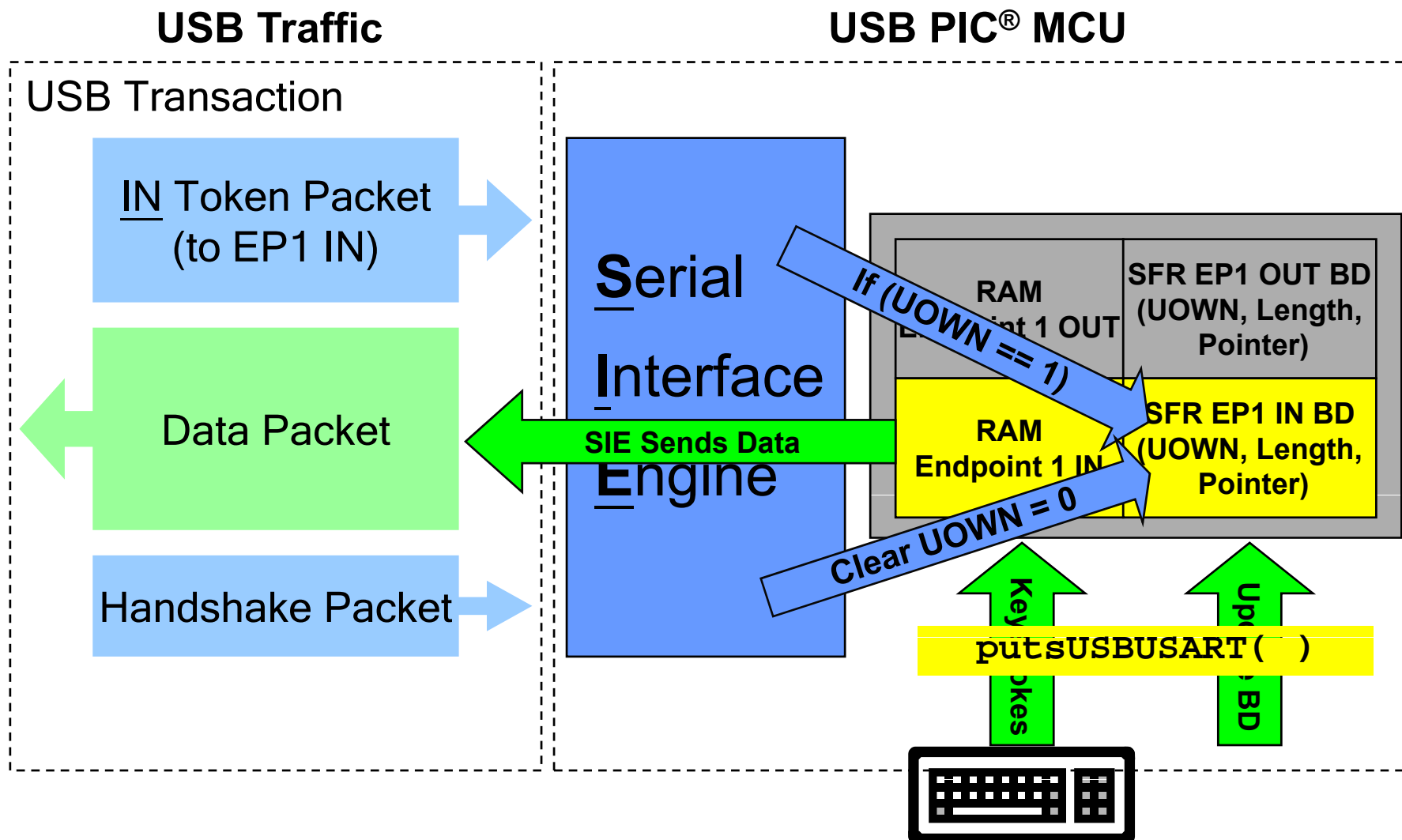- **Direction of the data transfer**

USB Transaction – IN

USB Transaction

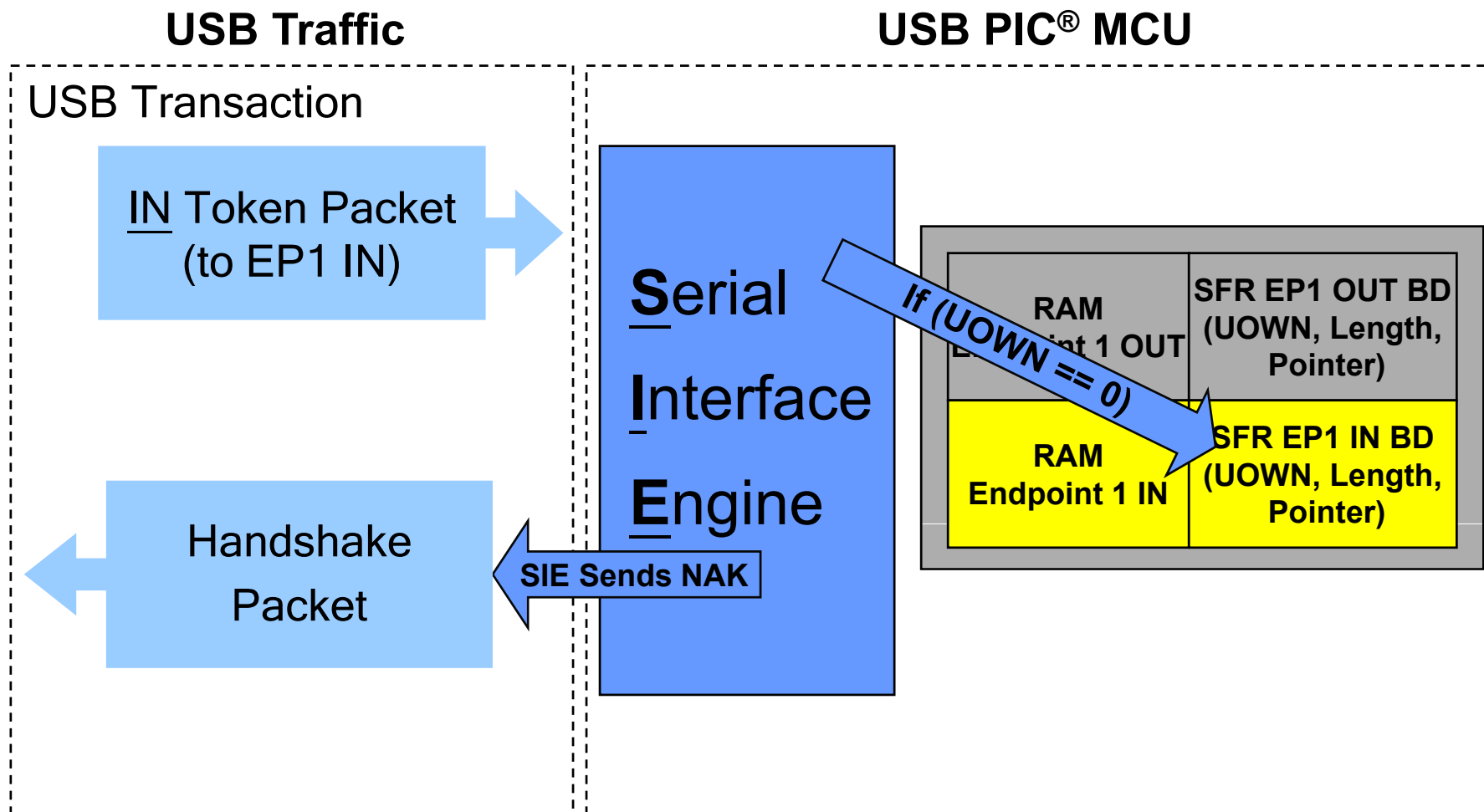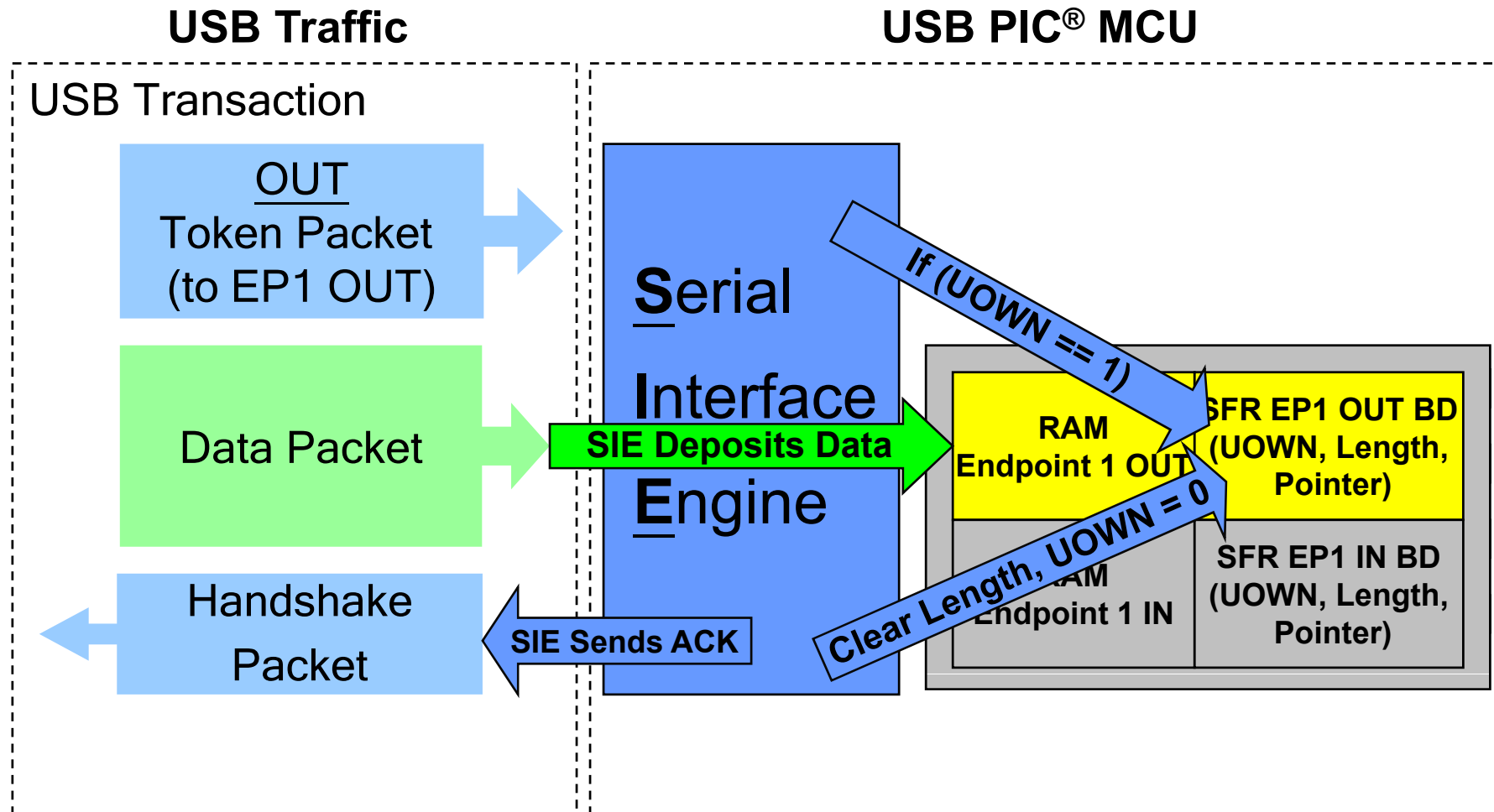IN Token Packet

Data Packet

Handshake Packet
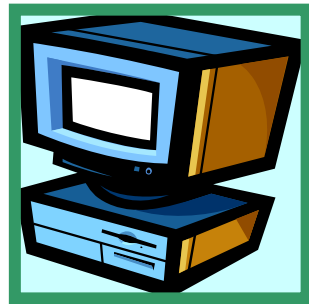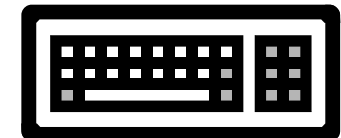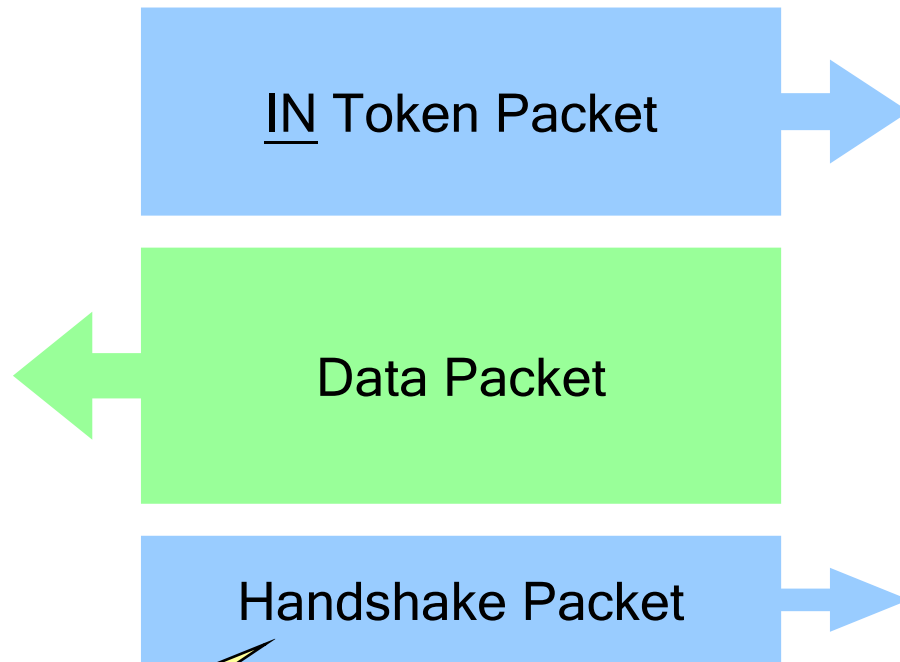
Acknowledge - ACK

IN Transaction ACK

USB Traffic

USB PIC® MCU

USB Transaction

IN Token Packet
(to EP1 IN)

Data Packet

Handshake Packet

Serial **I**nterface **E**ngine

SIE Sends Data

RAM Endpoint 1 OUT

RAM Endpoint 1 IN

SFR EP1 OUT BD
(UOWN, Length, Pointer)

SFR EP1 IN BD
(UOWN, Length, Pointer)

If (UOWN == 1)

Clear UOWN = 0

putsUSBUSART( )

IN Transaction NAK

USB Traffic

USB PIC® MCU

USB Transaction

IN Token Packet
(to EP1 IN)

Serial
Interface
Engine

If (UOWN == 0)

RAM Endpoint 1 OUT

SFR EP1 OUT BD
(UOWN, Length, Pointer)

RAM
Endpoint 1 IN

SFR EP1 IN BD
(UOWN, Length, Pointer)

Handshake
Packet

SIE Sends NAK

COM 3101

OUT Transaction ACK

USB Traffic

USB PIC® MCU

USB Transaction

OUT
Token Packet
(to EP1 OUT)

Serial

Interface

Engine

SIE Deposits Data

Data Packet

Handshake Packet

SIE Sends ACK

If (UOWN == 1)

RAM Endpoint 1 OUT

SFR EP1 OUT BD
(UOWN, Length, Pointer)

Clear Length, UOWN = 0

RAM Endpoint 1 IN

SFR EP1 IN BD
(UOWN, Length, Pointer)

COM 3101

OUT Transaction NAK

USB Traffic

USB PIC® MCU

USB Transaction

OUT
Token Packet
(to EP1 OUT)

Data Packet

Handshake Packet

Serial **I**nterface **E**ngine

SIE Sends NAK

SIE Discards Data

If (UOWN == 0)

RAM Endpoint 1 OUT	SFR EP1 OUT BD (UOWN, Length, Pointer)
RAM Endpoint 1 IN	SFR EP1 IN BD (UOWN, Length, Pointer)

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

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)`

Traffic to other Endpoints or Idle

Transactions

64

64

1

Frame

Bulk Transfer Example

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

Transactions only occur if no higher priority traffic

| Int. | Isoc. | | Int. | 64 | 64 | | Int. | 1 |

Frame
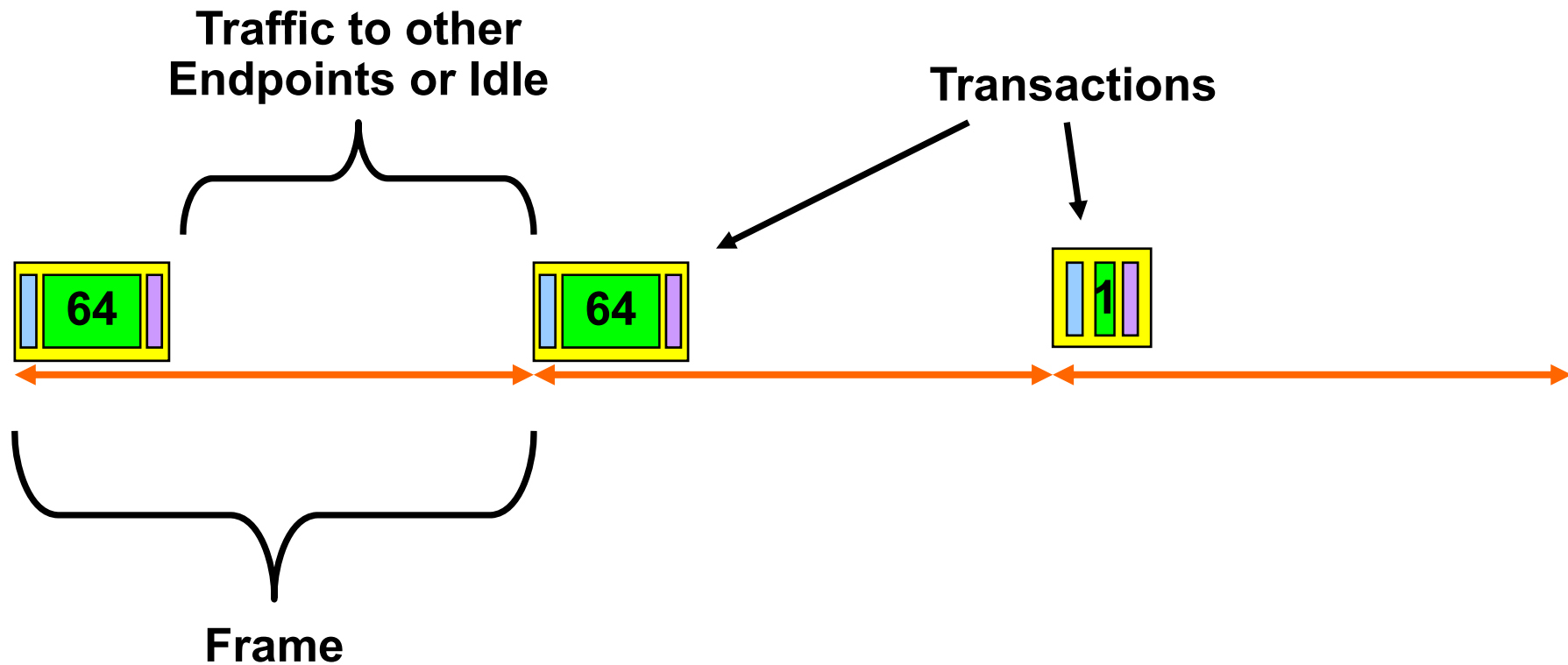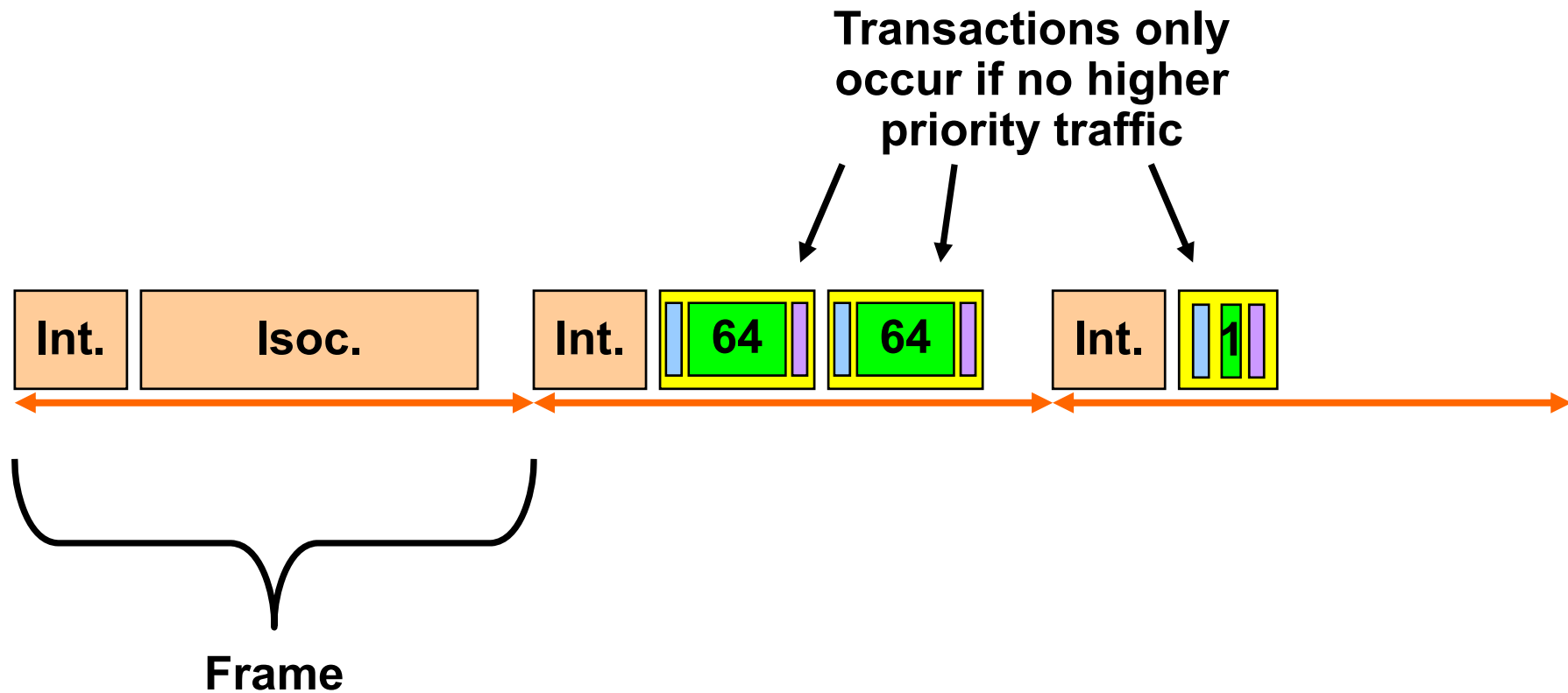
Theoretical Maximum Transfer Rate Per Endpoint

KByte/s — Transfer Type

- Low
- Full

Transfer Type	Low	Full
Control	24	832
Interrupt	0.8	64
Bulk		1216
Iso		1023

COM 3101
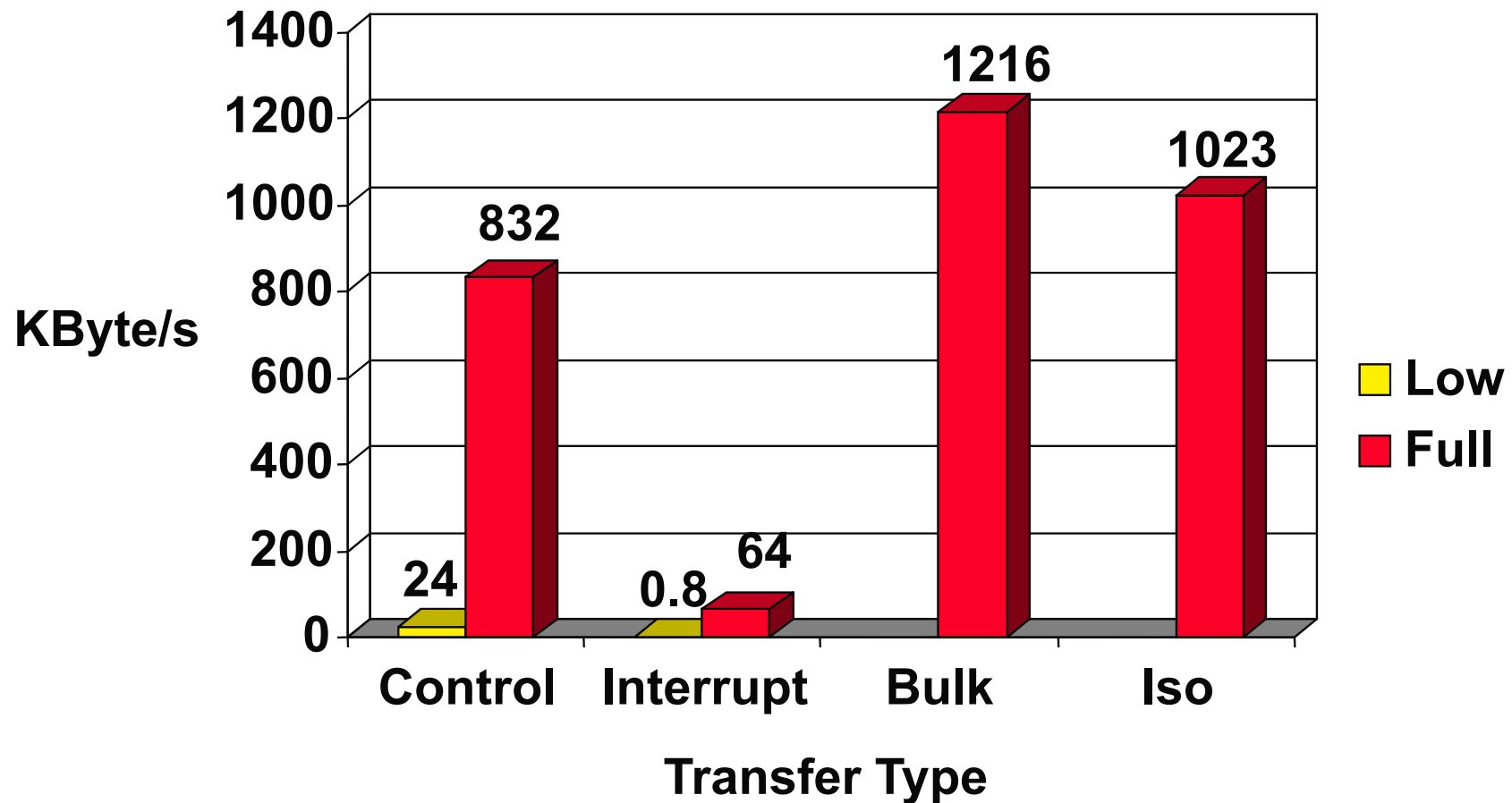
Transfer Types – Examples
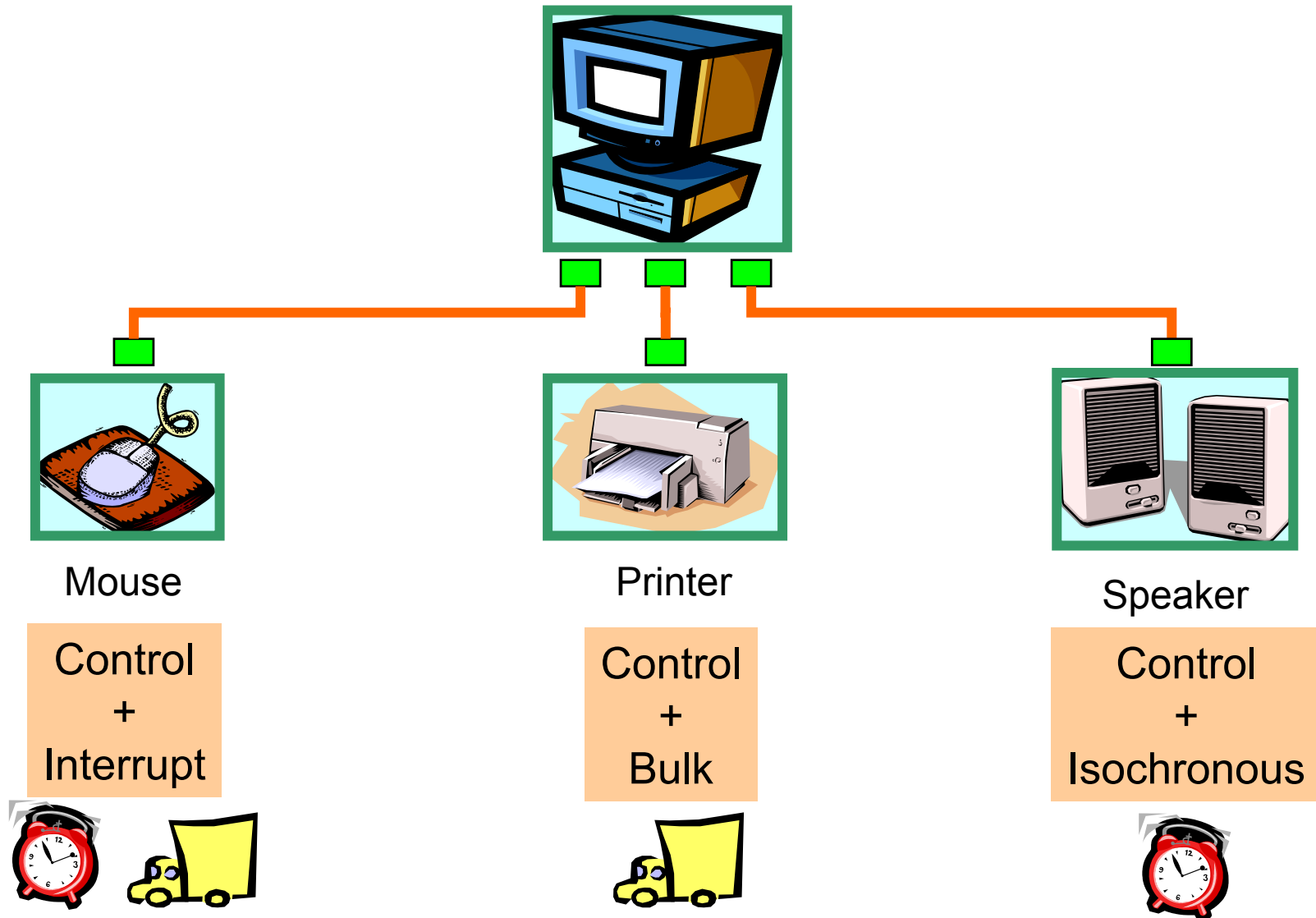
Mouse

Control
+
Interrupt

Printer

Control
+
Bulk

Speaker
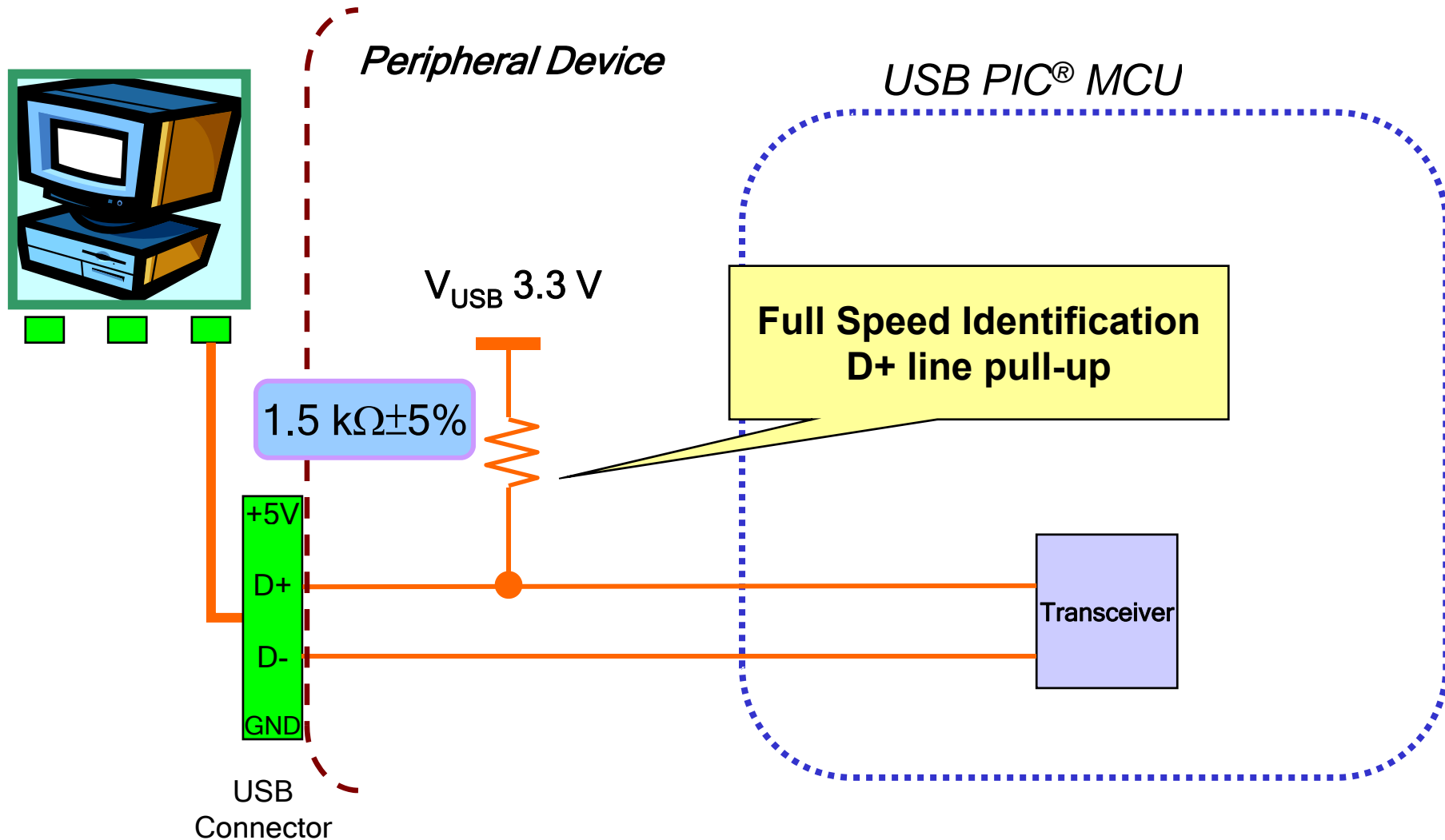
Control
+
Isochronous

COM 3101

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**
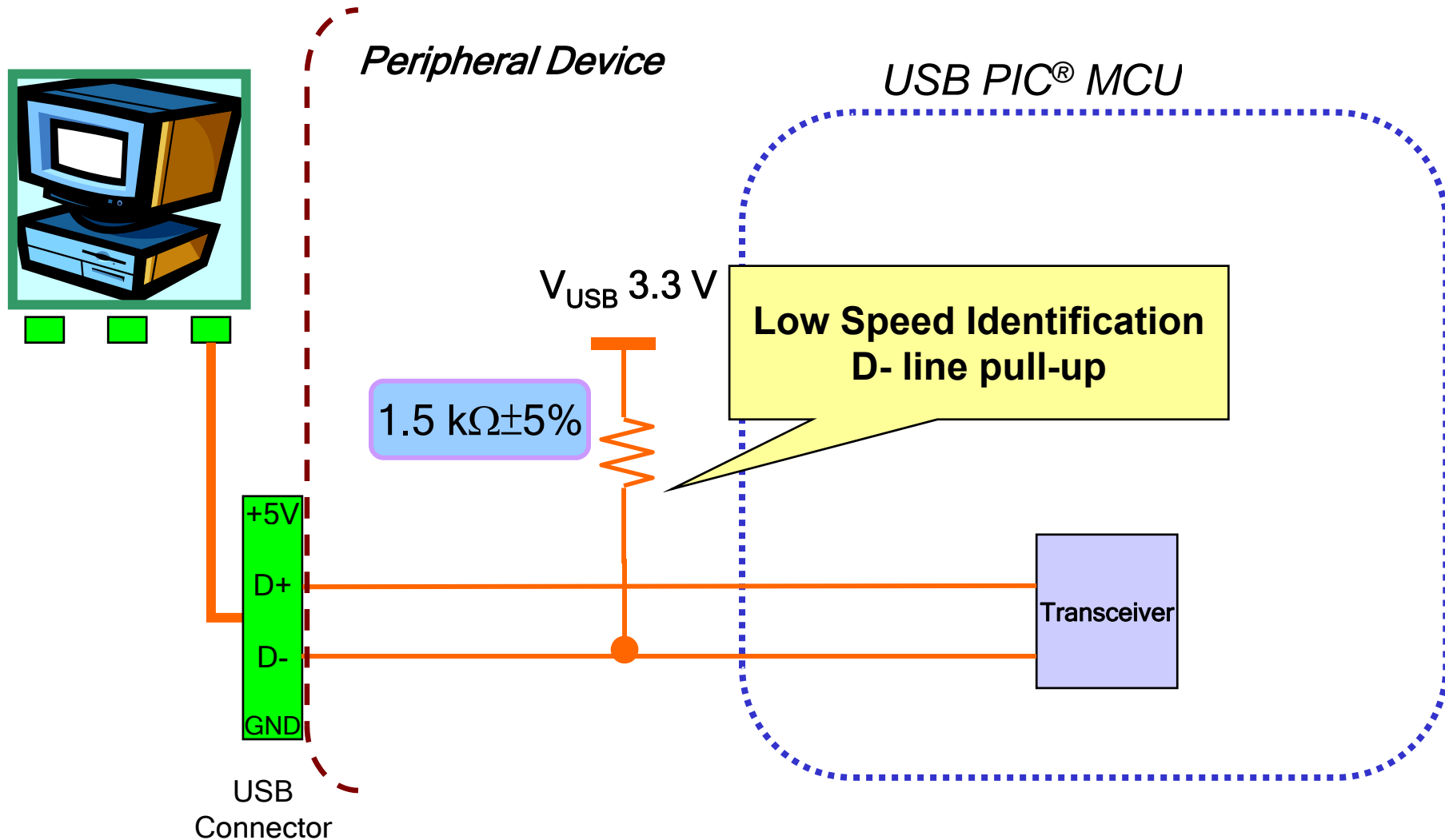- **Microchip Demo/Development Solutions**

USB Device Classes
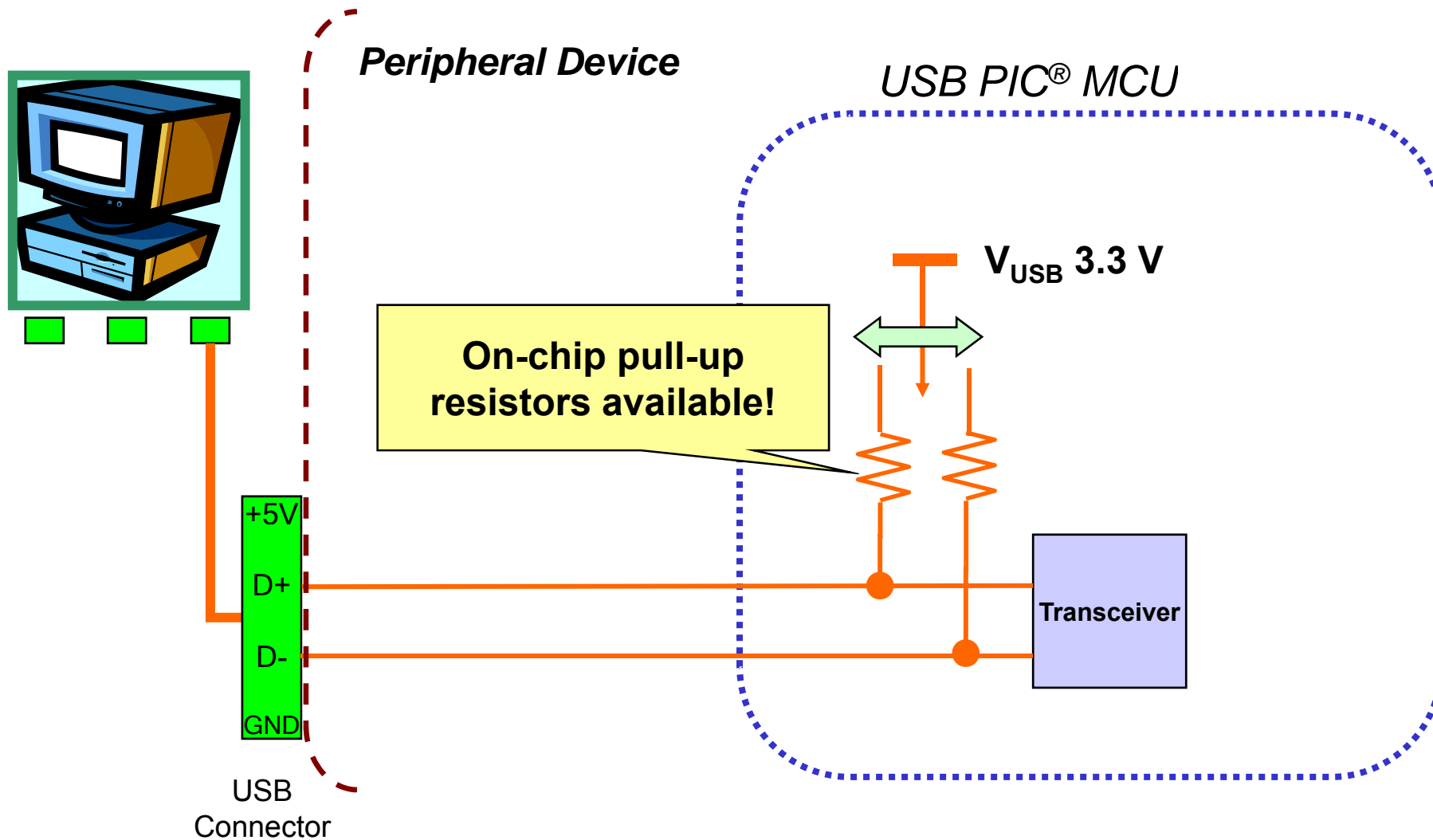
Data Glove

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….
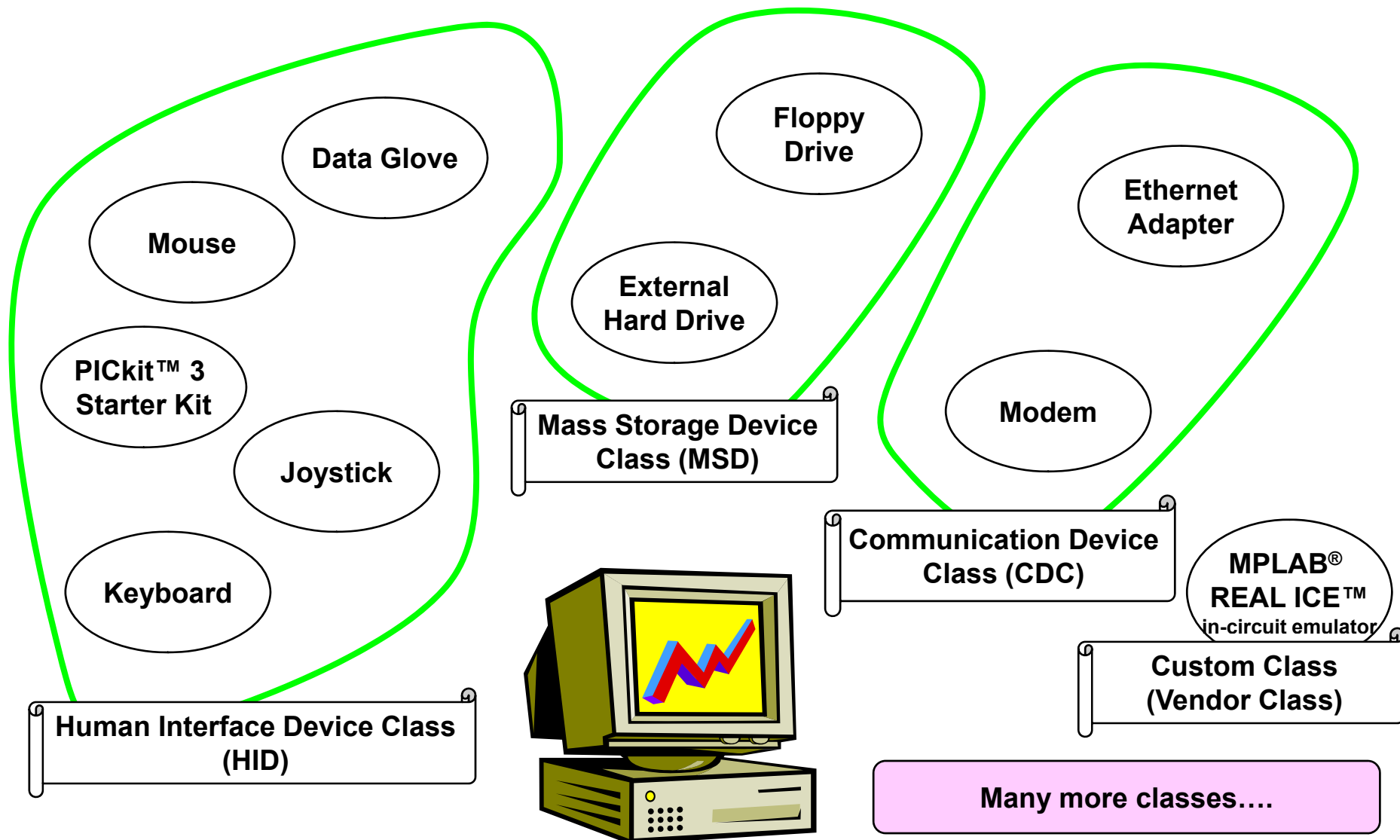
USB Driver Choices
- Windows® PC Host -

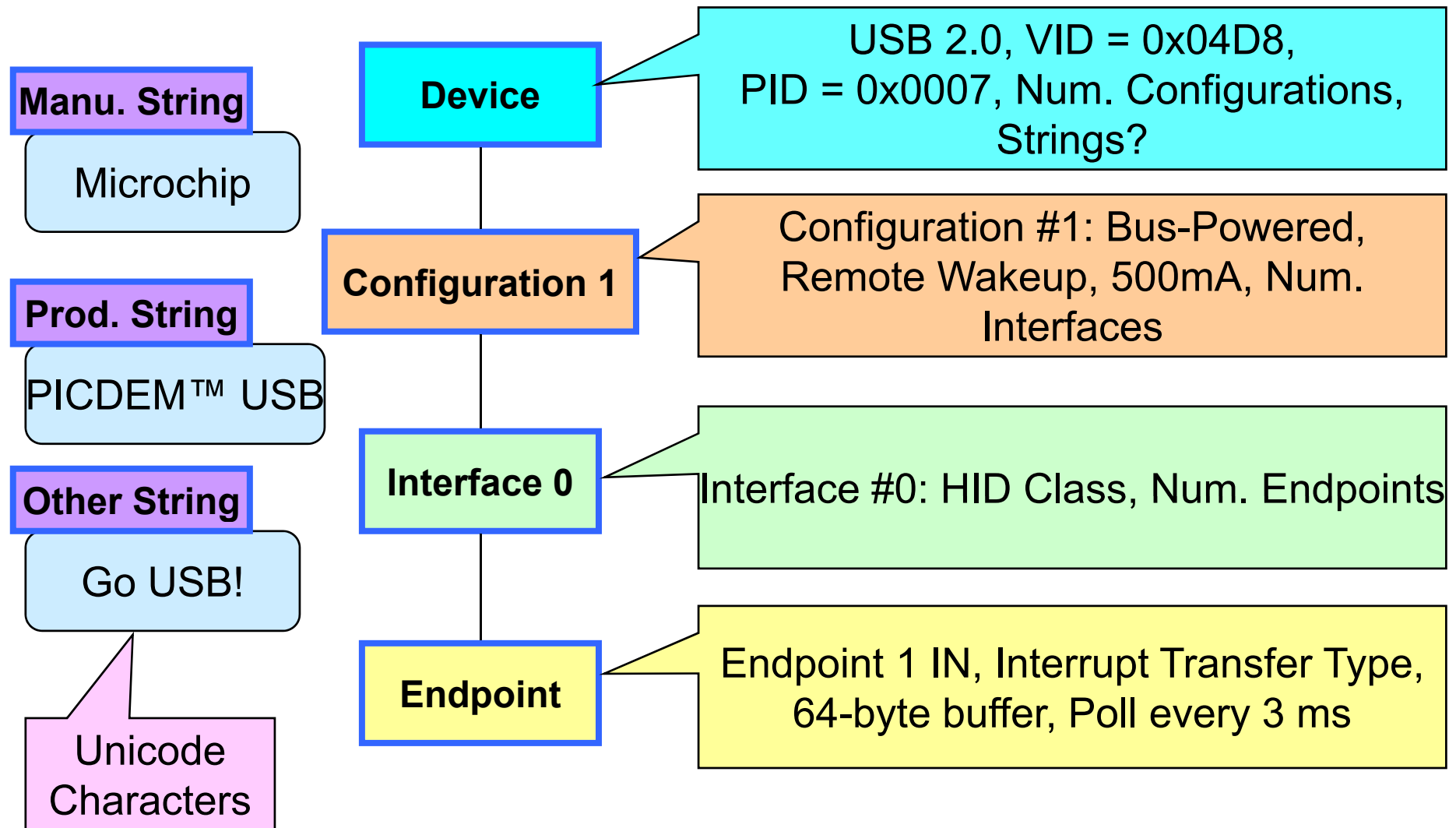| Features	HID	CDC	MCHPUSB	WinUSB	LibUSB
Driver support built into Windows	Yes	Need .inf	No	Need .inf	No
64-bit PC Support	Yes	Yes	Yes	Yes	Yes
XP Ready	Yes	Yes	Yes	Yes	Yes
Vista Ready	Yes	Yes	Yes	Yes	32 bit
Transfer Types for user's data					
Control	No	No	Yes	Yes	Yes
Interrupt	Yes	No	Yes	Yes	Yes
Isochronous	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

COM 3101

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**

The Enumeration Process
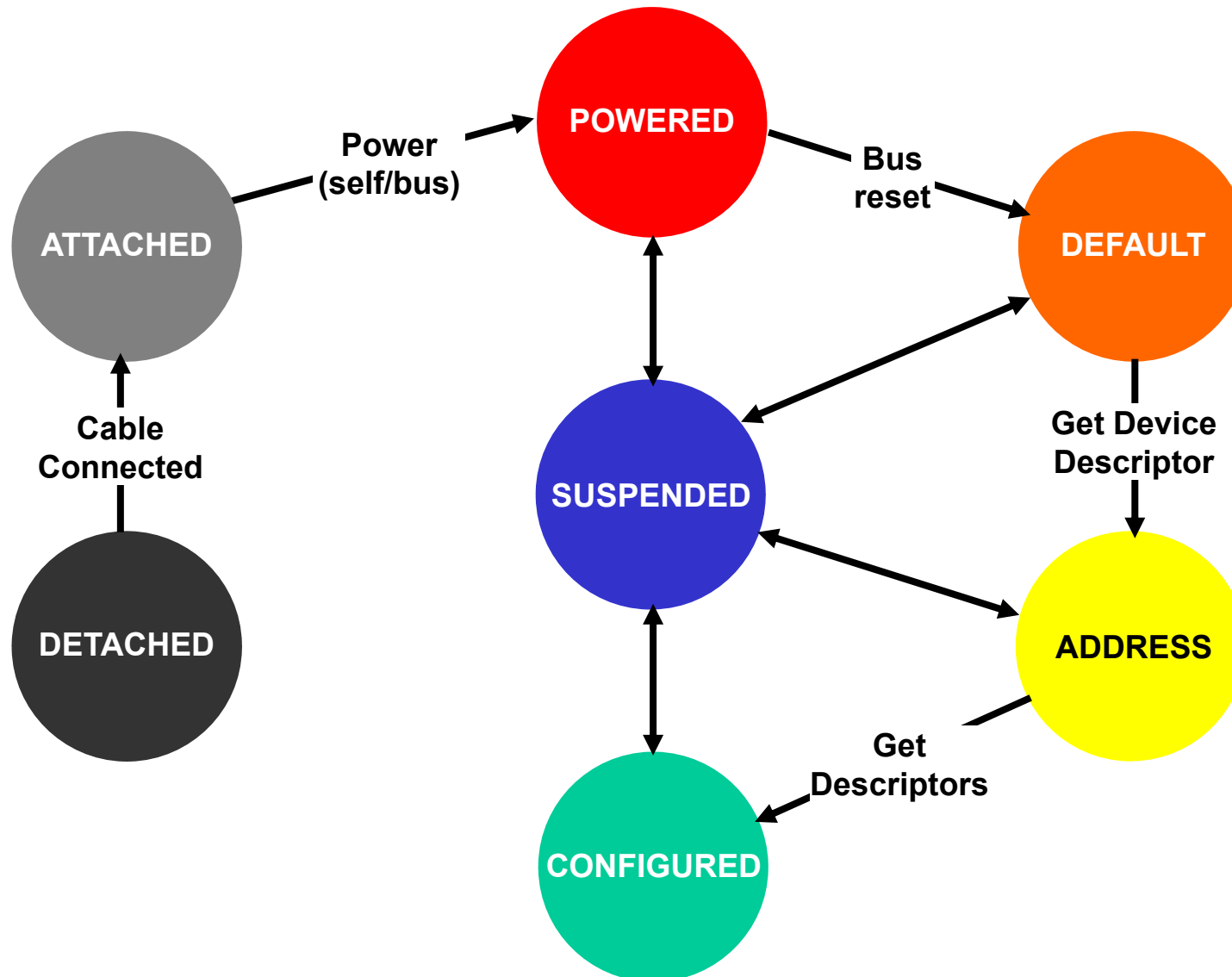
Auto-Detection: Full-Speed

Peripheral Device

USB PIC® MCU

V_{USB} 3.3 V

**Full Speed Identification
D+ line pull-up**

1.5 kΩ±5%

+5V

D+

D-

GND

Transceiver

USB
Connector

Auto-Detection: Low-Speed

Peripheral Device

USB PIC® MCU

V_{USB} 3.3 V

Low Speed Identification D- line pull-up

1.5 k$\Omega\pm$5%

+5V

D+

D-

GND

Transceiver

USB Connector

COM 3101

On-chip Pull-up Resistors

Peripheral Device

USB PIC® MCU

V_{USB} 3.3 V

On-chip pull-up resistors available!

+5V

D+

D-

GND

Transceiver

USB Connector

Address and Configuration: EP0

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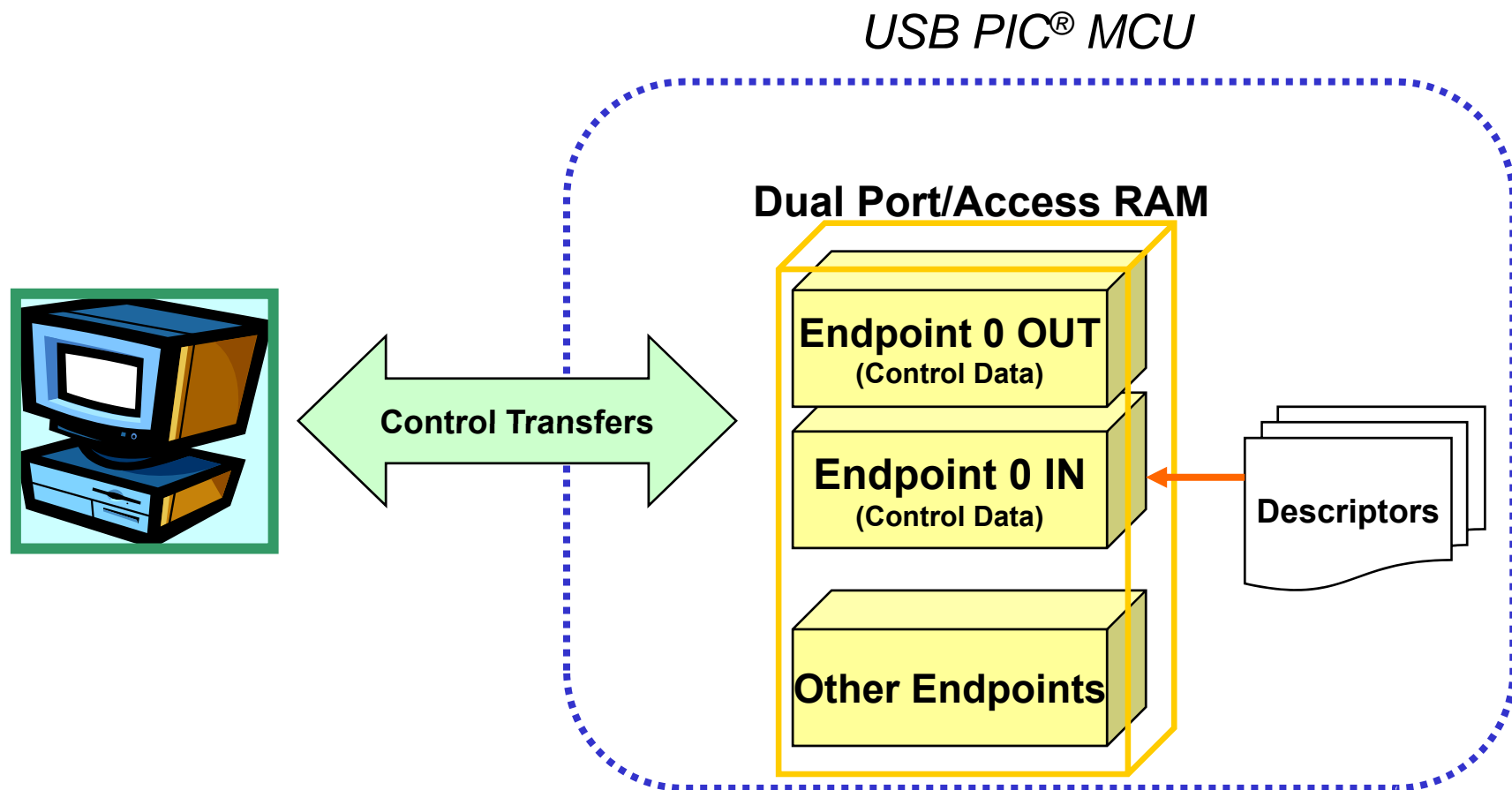
USB PIC® MCU

Control Transfers

Dual Port/Access RAM

Endpoint 0 OUT
(Control Data)

Endpoint 0 IN
(Control Data)

Other Endpoints

Descriptors

Agenda – Part 1

Regional Training Centers

- **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

String 0

String 1

String N

Device

To other Configurations if any

Configuration 1

To other Interfaces if any

Interface 0

Interface 1

Endpoint

Endpoint

Endpoint

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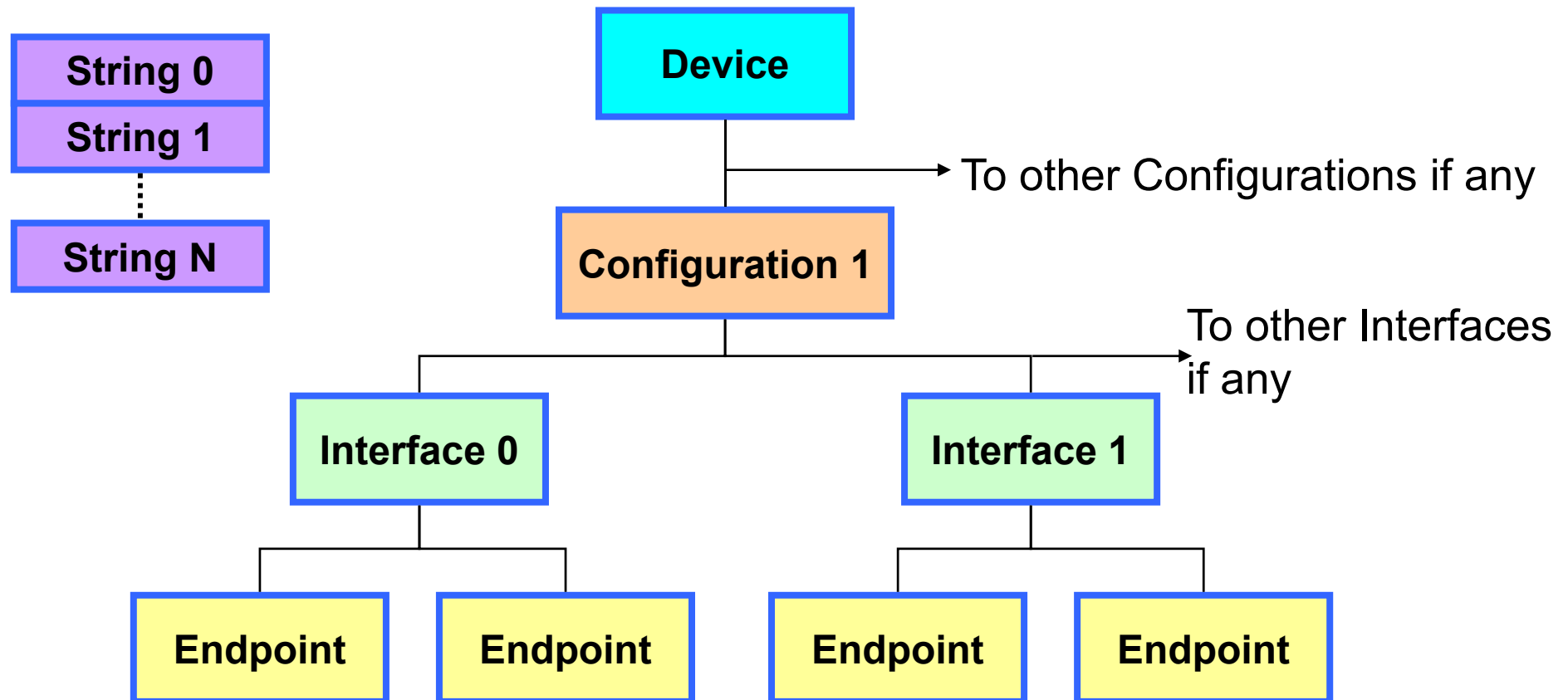
- **Descriptors are typically stored in non-volatile/Flash memory**

COM 3101

Descriptors - Example

Manu. String

Microchip

Prod. String

PICDEM™ USB

Other String

Go USB!

Unicode Characters

Device → USB 2.0, VID = 0x04D8, PID = 0x0007, Num. Configurations, Strings?

Configuration 1 → Configuration #1: Bus-Powered, Remote Wakeup, 500mA, Num. Interfaces

Interface 0 → Interface #0: HID Class, Num. Endpoints

Endpoint → Endpoint 1 IN, Interrupt Transfer Type, 64-byte buffer, Poll every 3 ms

MCHPFSUSB Software Framework
- Device Descriptor Table -

- ## usb descriptors.c
  - ## Descriptors
    - ### VID & PID
    - ### Class Specific
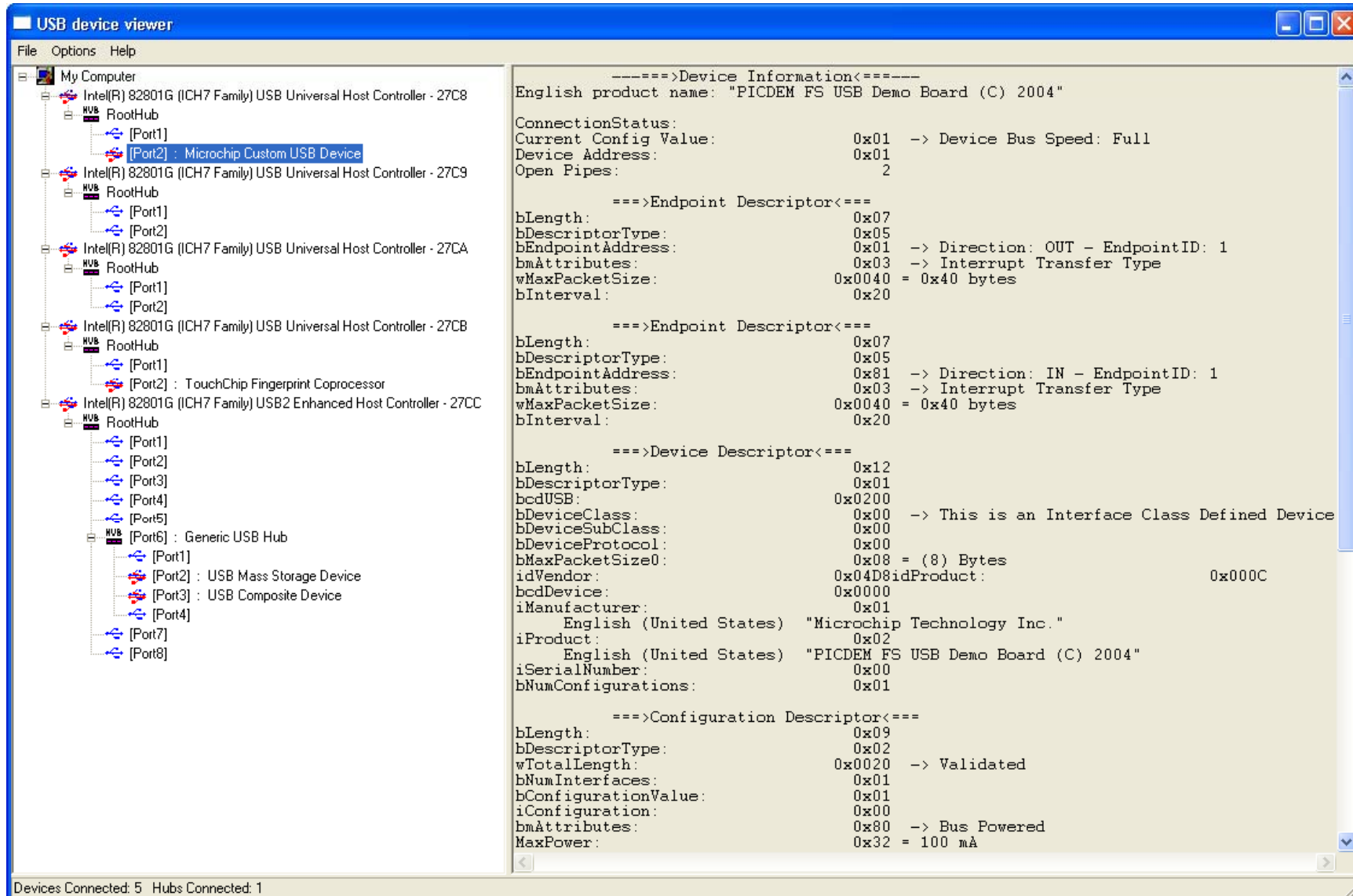
```
/* Device Descriptor */
ROM USB_DEVICE_DESCRIPTOR device_dsc=
{   0x12,       // Size of this descriptor in bytes
    USB_DESCRIPTOR_DEVICE,   // DEVICE descriptor type
    0x0200,                  // USB Spec Release Number
    CDC_DEVICE,              // Class Code
    0x00,                    // Subclass code
    0x00,                    // Protocol code
    EP0_BUFF_SIZE,           // Max packet size for EP0,
    0x04D8,                  // Vendor ID
    0x000C,                  // Product ID
…
```

Demo
- Viewing Descriptor Information -
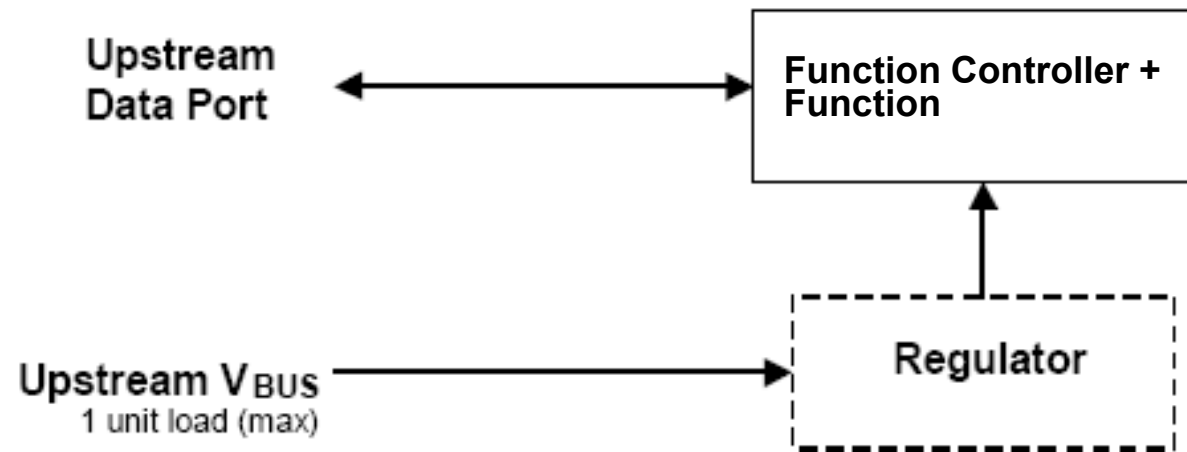
Agenda – Part 1

Regional Training Centers

- **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

Upstream Data Port

Function Controller + Function

Upstream V$_{BUS}$
1 unit load (max)

Regulator

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**

COM 3101
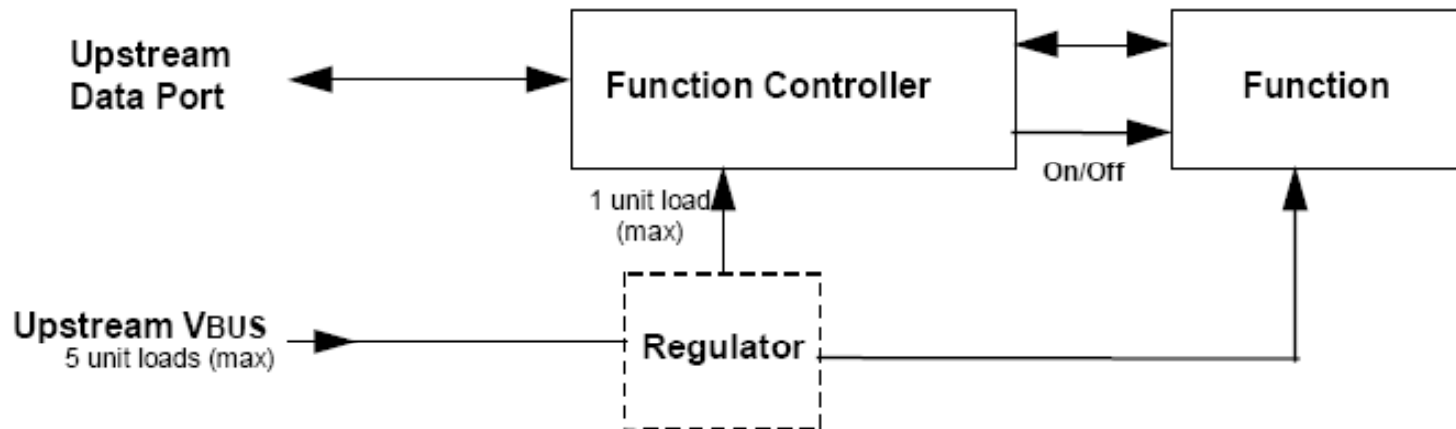
Power Planning
- Architecture -

- ## "Self Powered" device
  - Can optionally draw up to 100 mA from the bus (if unsuspended) + as much as is available from its own supply
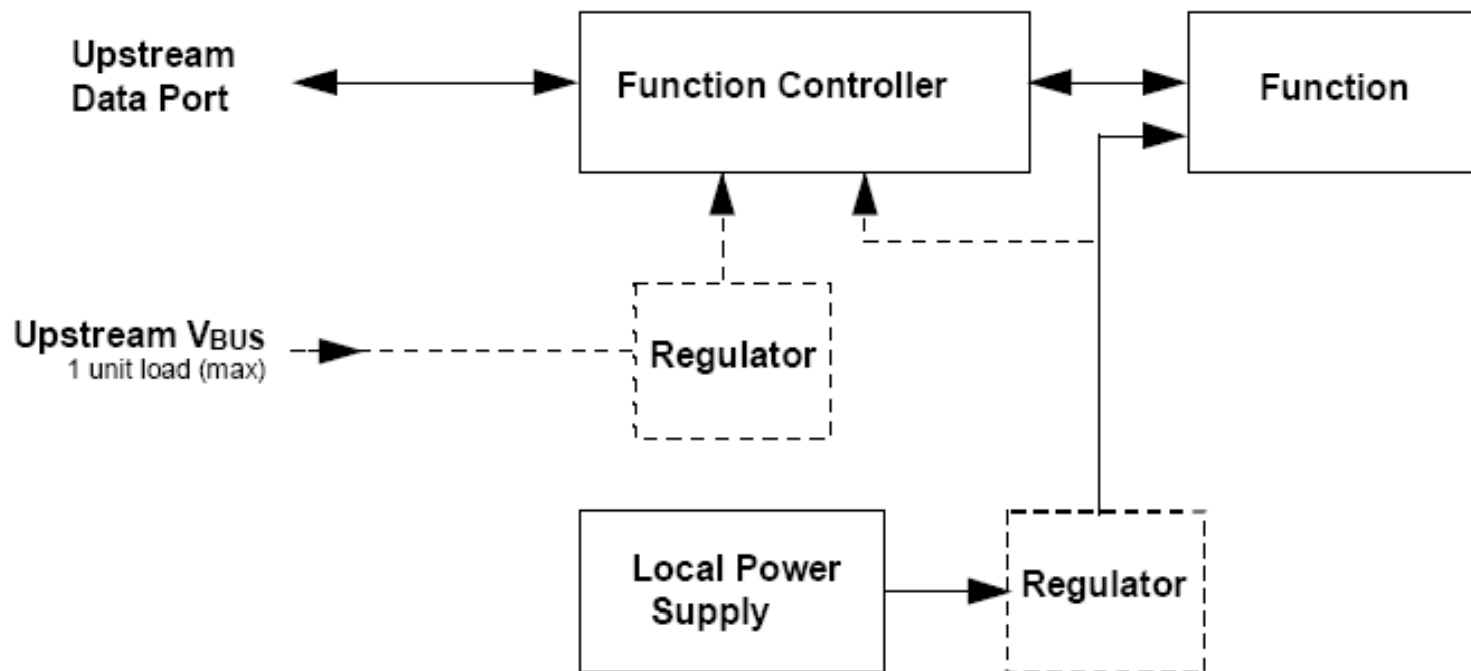
Power Planning
- Do I Need Self-Power? -

- **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**
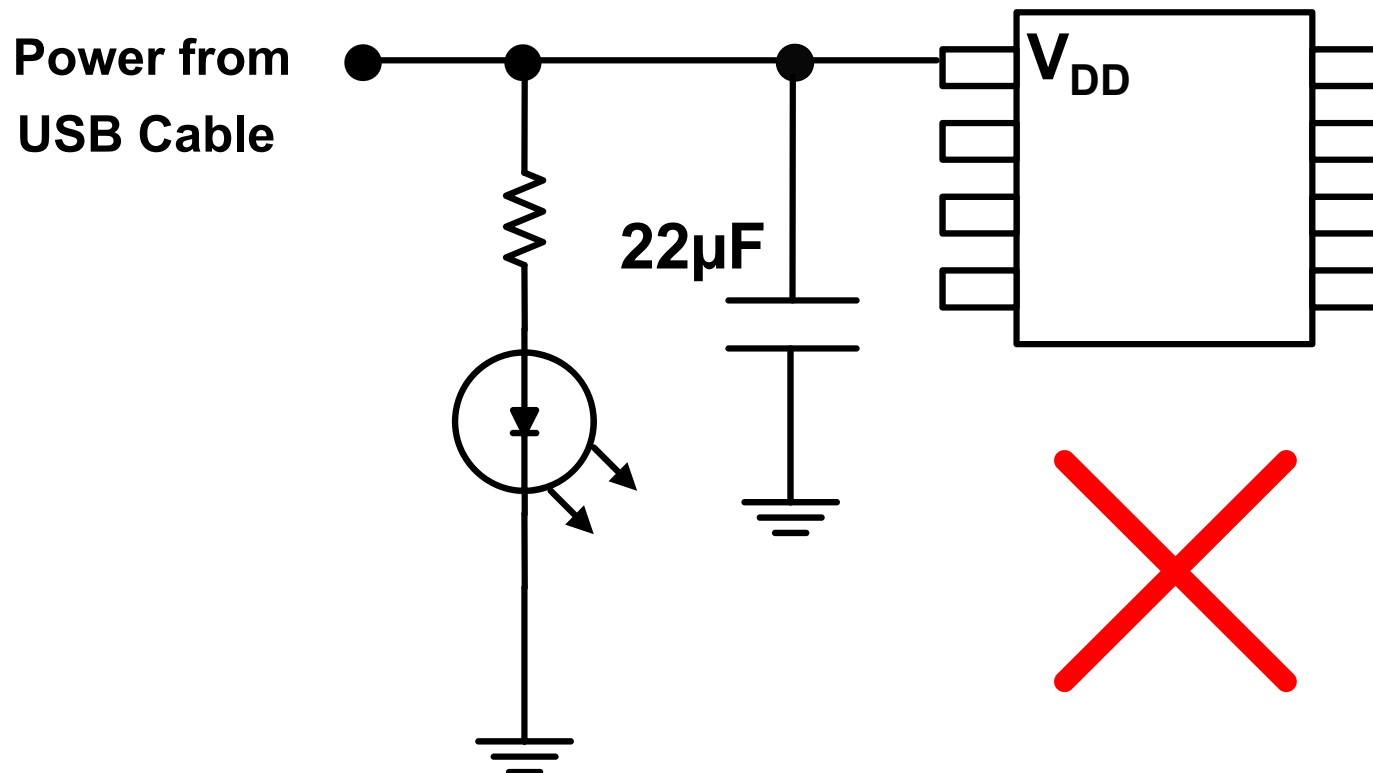
Power Planning

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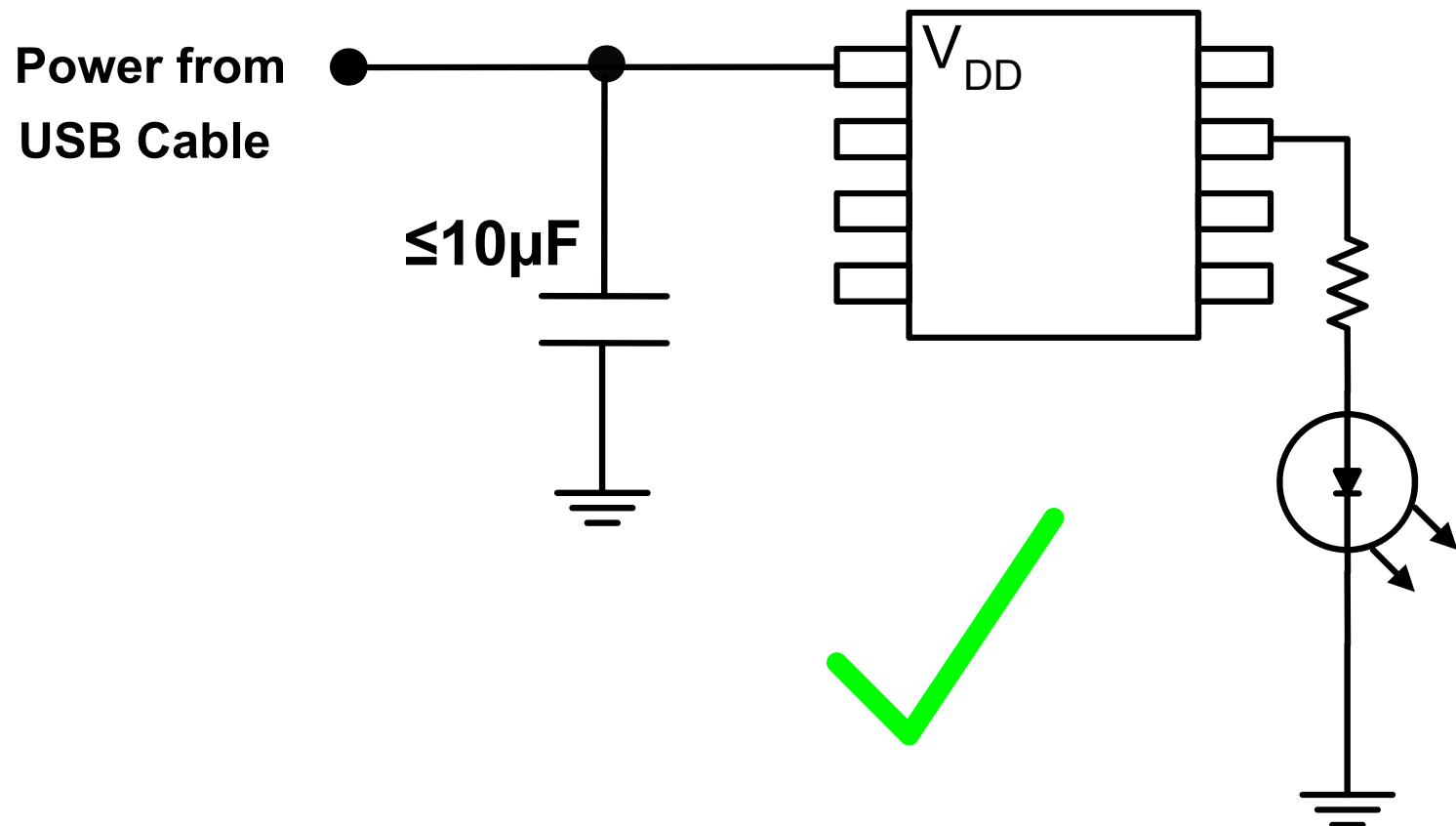
- **Maximum USB suspend current:**
  - 2.5 mA
- **Don't:**

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

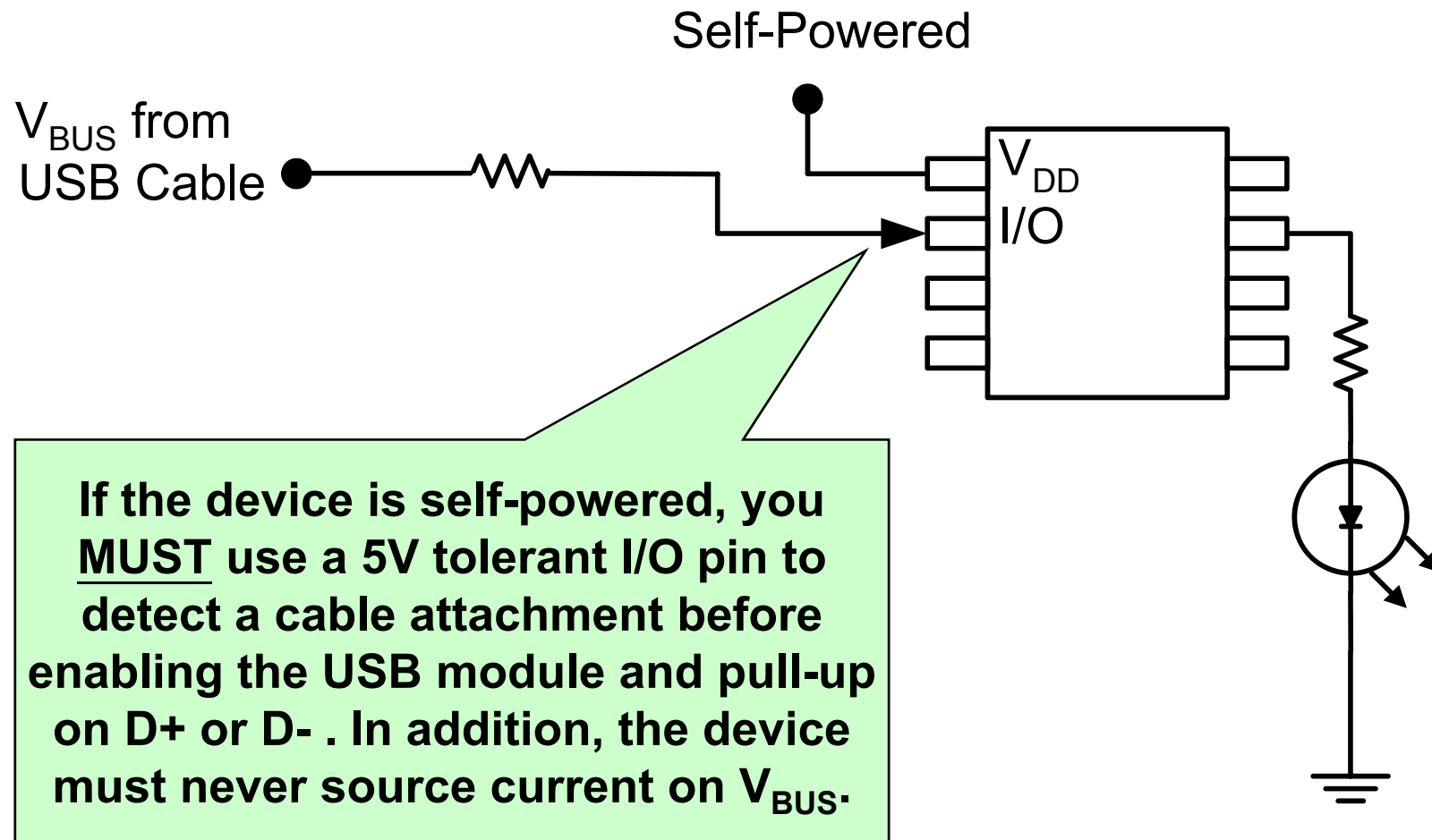**Power from**

USB Cable

V_{DD}

$\leq 10\mu F$

Self-Powered

V_{BUS} from
USB Cable

V_{DD}

I/O

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}.

COM 3101

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**

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

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
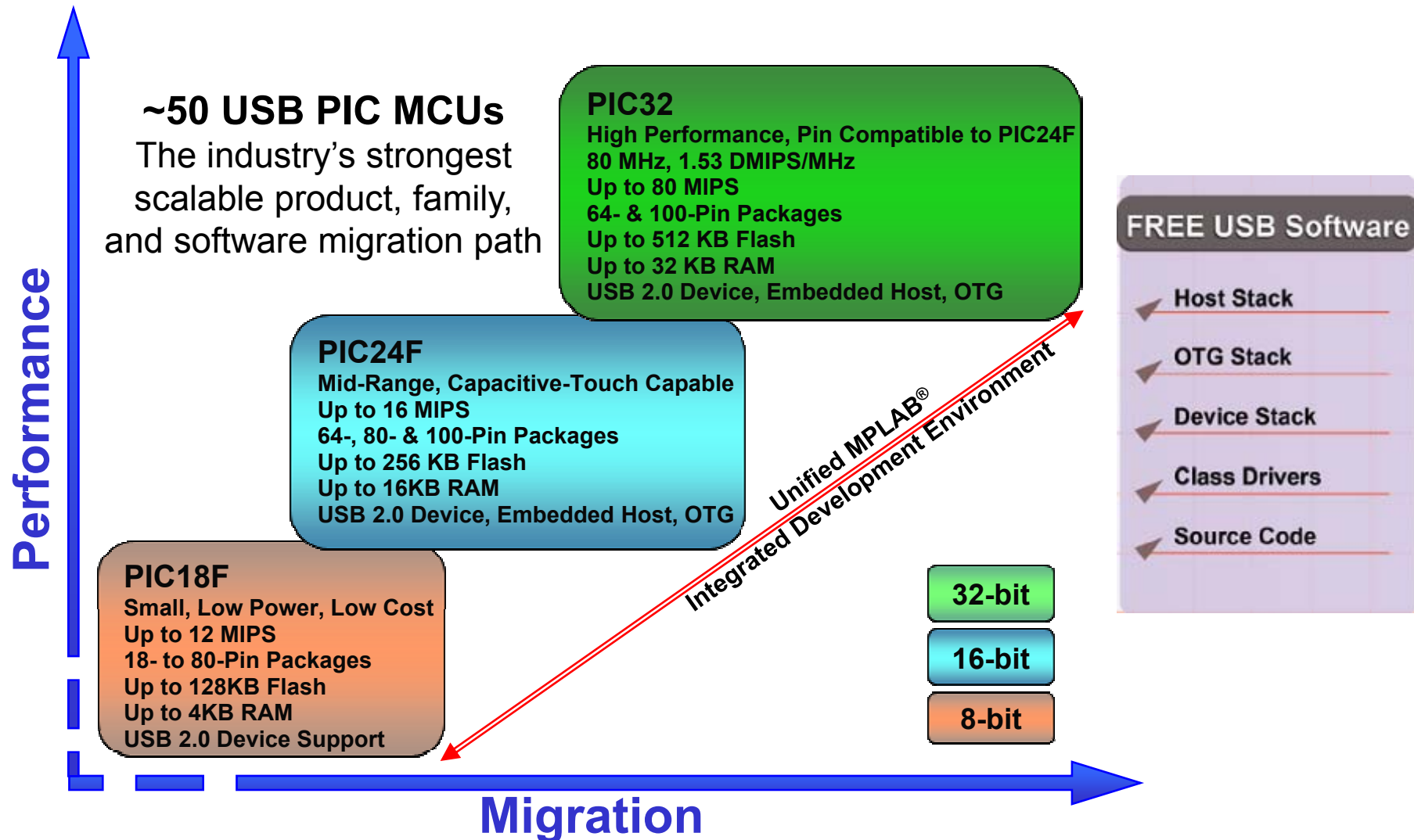
  - **www.usb.org/developers/tools**

- **Electrical Signal Quality**

- **Power Management**

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**

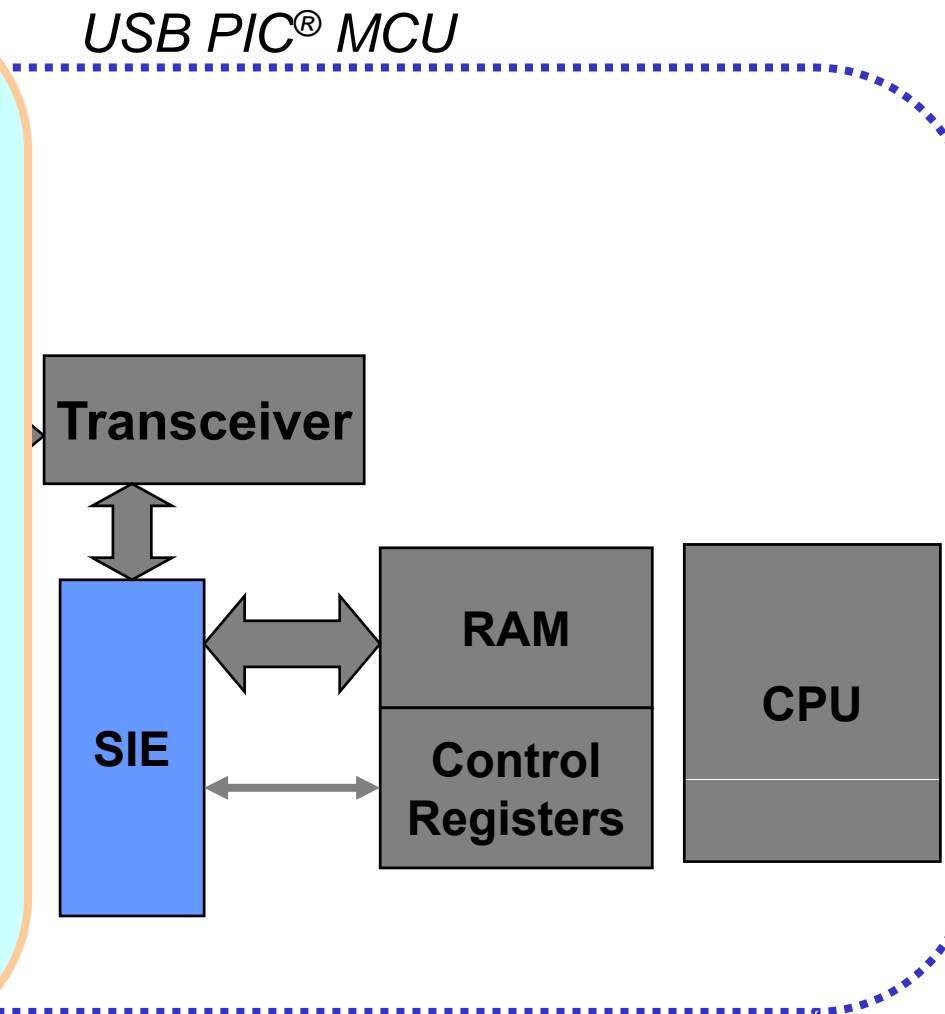- **Microchip Demo/Development Solutions**

Scalable USB PIC® MCU Portfolio

Performance ↑

~50 USB PIC MCUs
The industry's strongest scalable product, family, and software migration path

PIC32
High Performance, Pin Compatible to PIC24F
80 MHz, 1.53 DMIPS/MHz
Up to 80 MIPS
64- & 100-Pin Packages
Up to 512 KB Flash
Up to 32 KB RAM
USB 2.0 Device, Embedded Host, OTG

PIC24F
Mid-Range, Capacitive-Touch Capable
Up to 16 MIPS
64-, 80- & 100-Pin Packages
Up to 256 KB Flash
Up to 16KB RAM
USB 2.0 Device, Embedded Host, OTG

PIC18F
Small, Low Power, Low Cost
Up to 12 MIPS
18- to 80-Pin Packages
Up to 128KB Flash
Up to 4KB RAM
USB 2.0 Device Support

Unified MPLAB®
Integrated Development Environment

FREE USB Software
- Host Stack
- OTG Stack
- Device Stack
- Class Drivers
- Source Code

32-bit
16-bit
8-bit

Migration →

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^{2}C™	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

COM 3101

Serial Interface Engine

USB PIC® MCU

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

Transceiver

SIE

RAM

Control Registers

CPU

USB Device (Peripheral) Example Circuit
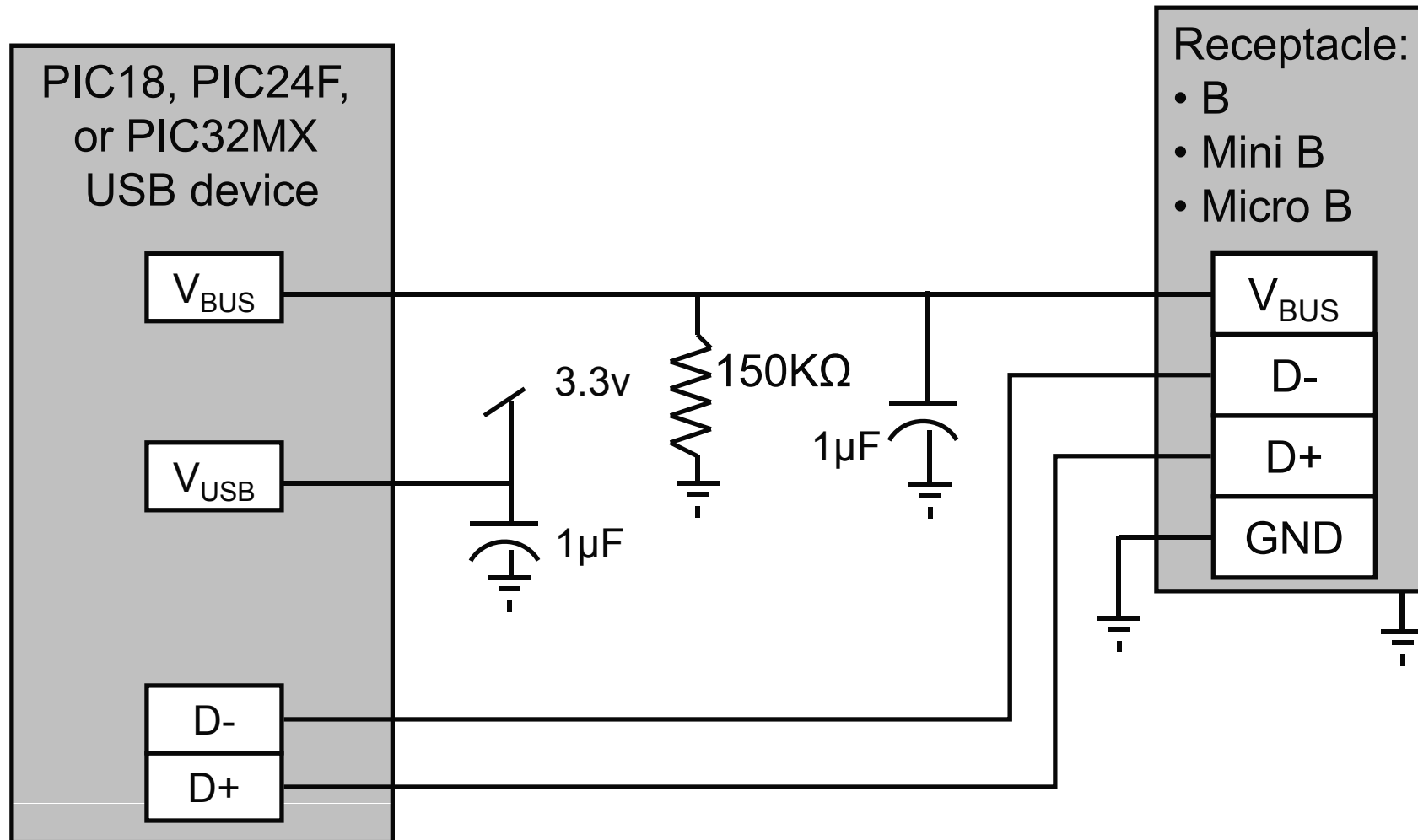
PIC18, PIC24F, or PIC32MX USB device

- V_{BUS}
- V_{USB} — 3.3v — 1µF
- D-
- D+

150KΩ

1µF

Receptacle:
- B
- Mini B
- Micro B

- V_{BUS}
- D-
- D+
- GND

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**

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**

COM 3101

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**

COM 3101

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**

COM 3101

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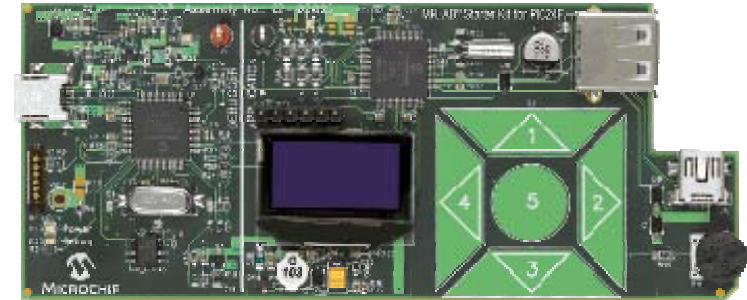
- **PIC32 USB Starter Board II**
  - Part #: DM320003-2
  - PIC32MX795F512L

  Priced from $55.00
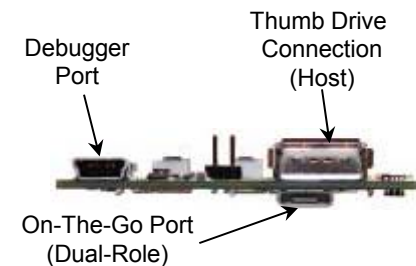
- **Explorer 16 + USB PICtail™ Plus Daughter Board + USB PIMs**
  - Part #: MA320002/MA240014
  - Part #: AC164131
  - Part #: DM240001

  Priced from $214.99

- **All Available Now**

Debugger Port

Thumb Drive Connection (Host)

On-The-Go Port (Dual-Role)
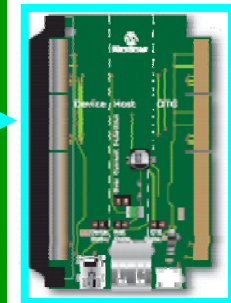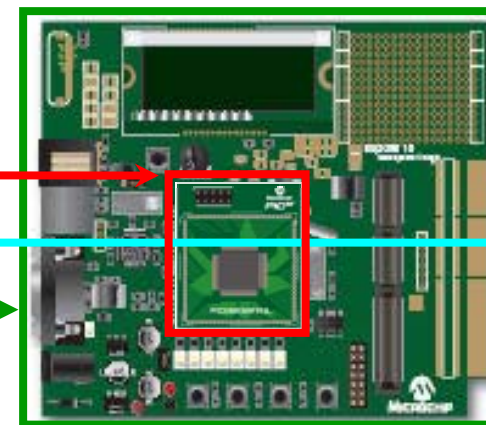
APP013

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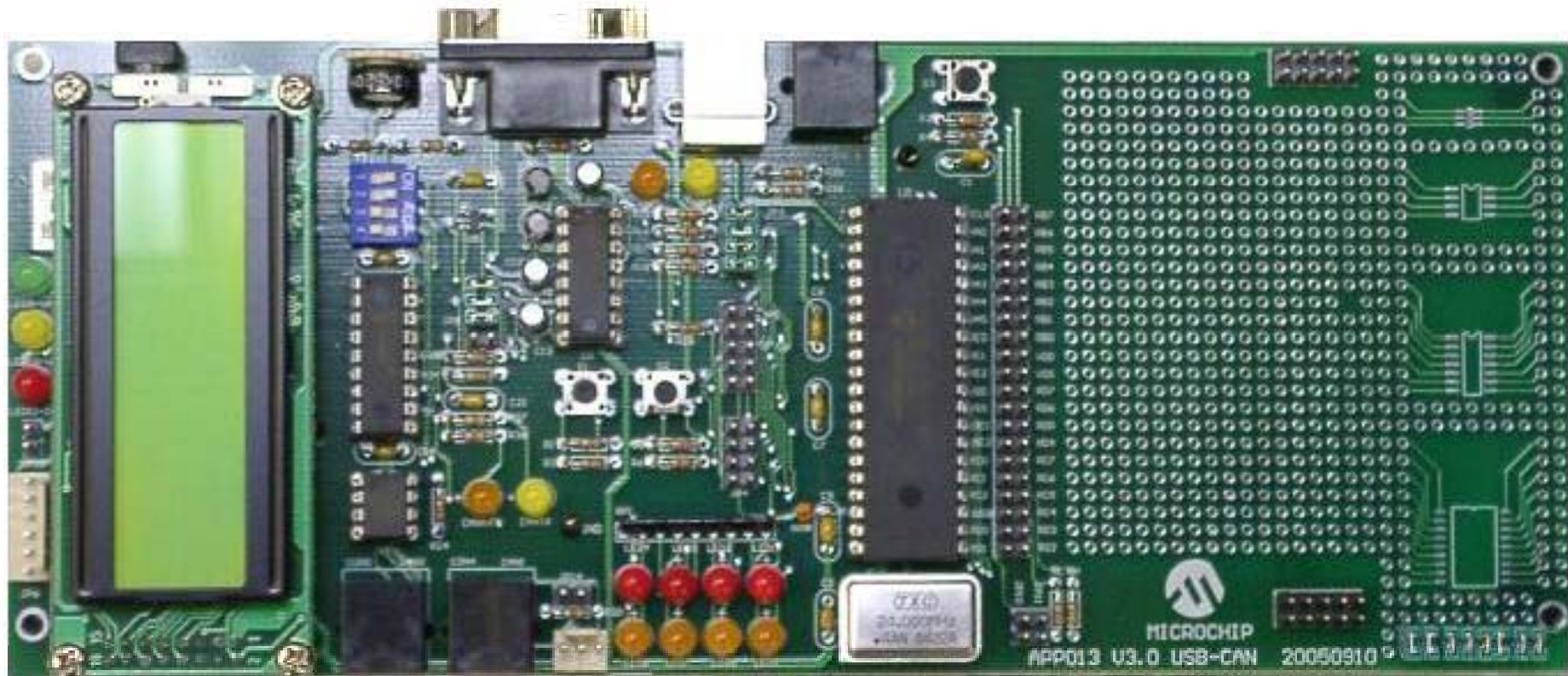
APP013 Hobby Kit(停售)

- **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

MAL

- Microchip Application Library包含所有Microchip的應用範例及程式庫,包括**USB Stack、TCP/IP Stack、Bluetooth Stack、mTouchTM、Graphics、Audio**等。

- Microchip Application Library中,可至專屬網頁(http://www.microchip.com/mal)下載,檔案安裝後,會在C磁碟機下產生Microchip_Solutions_v2013_xx_xx資料夾

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

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**

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?**
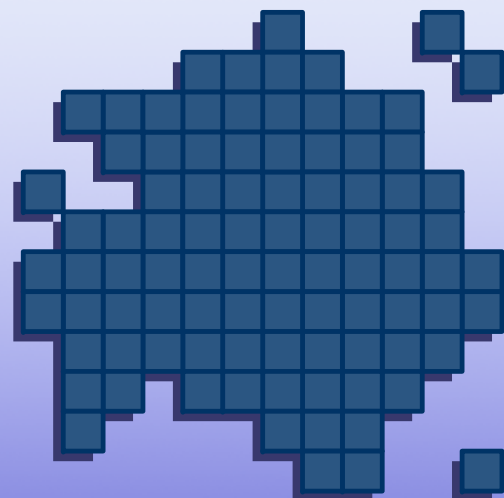
Part 2.

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
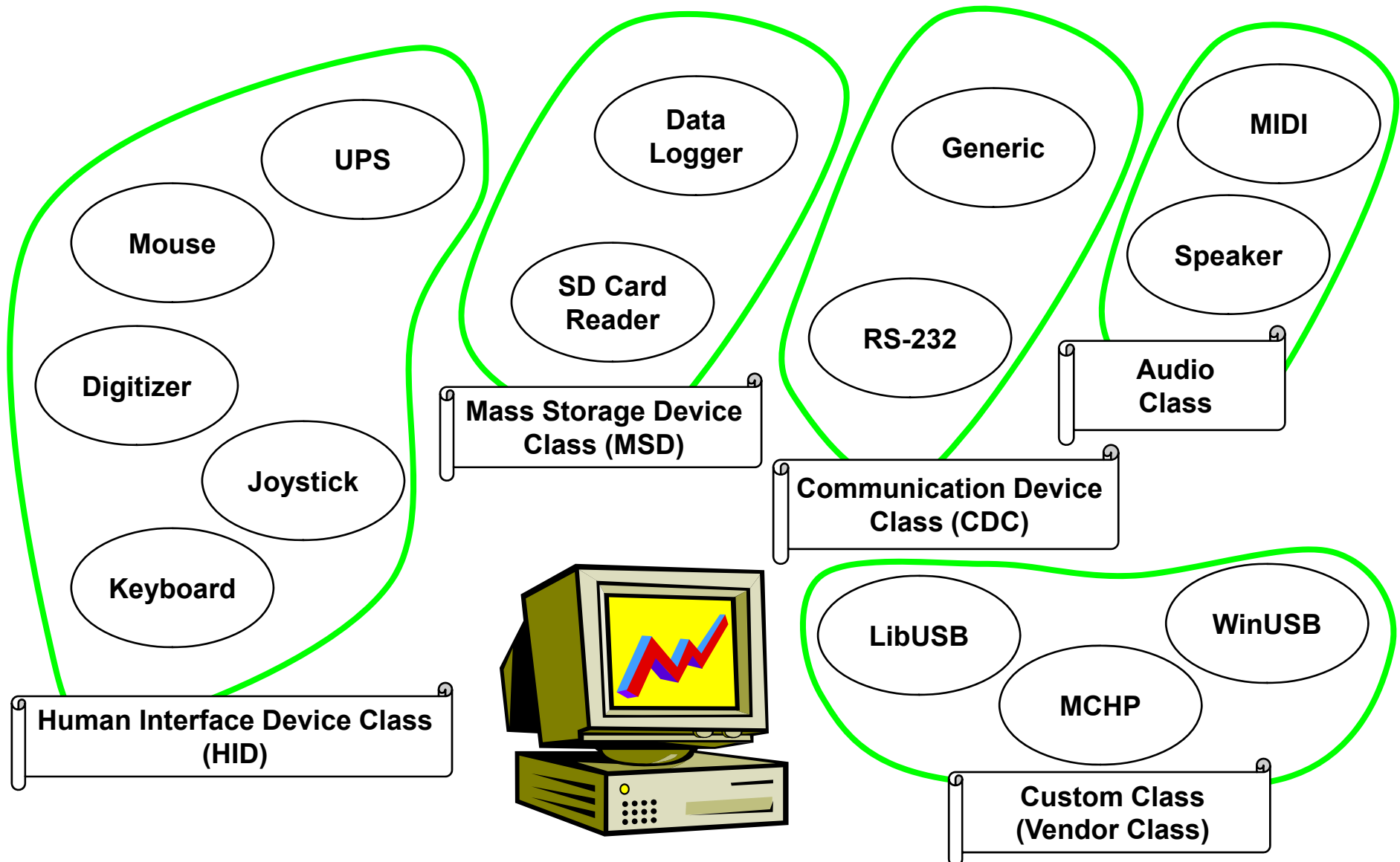
Human Interface Device (HID)

PIC® Microcontroller

HID\Mouse

HID Bootloader

PC Computer

Standard Windows Drivers

HID App
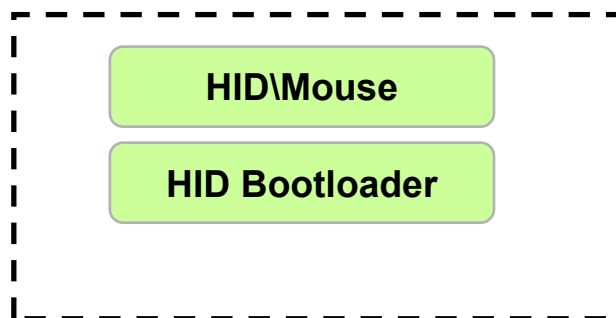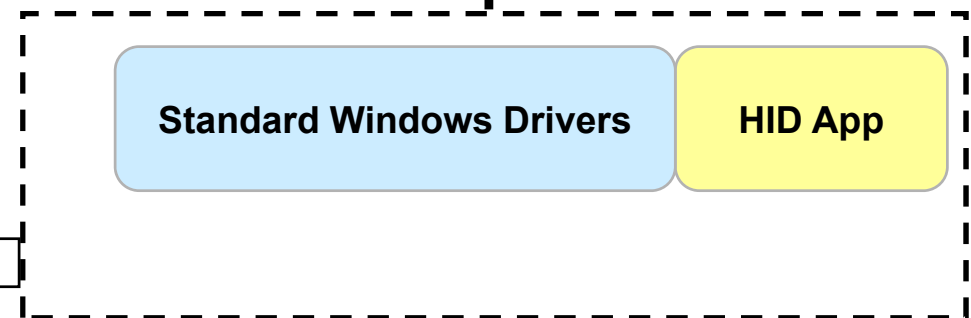
USB Cable

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

PIC® Microcontroller

CDC

PC Computer

Standard Windows Drivers	Hyper Terminal

USB Cable

INF File Required
(Supplied in MCHPSUSB)
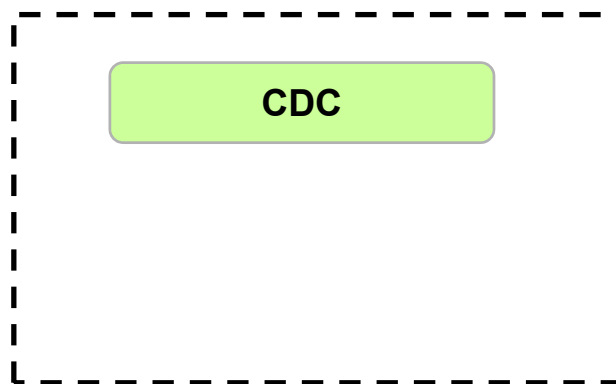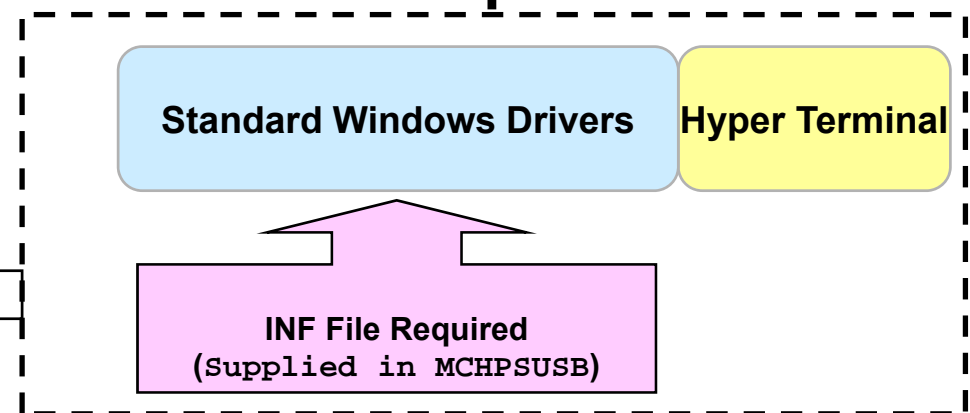
Design Considerations:
- ~80 KB/s max
- Bulk Transfers
- PC applications can access the device as though it is connected to a serial COM port

COM 3101

Mass Storage Device (MSD)

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

PIC® Microcontroller

MSD

PC Computer

Standard Windows Driver
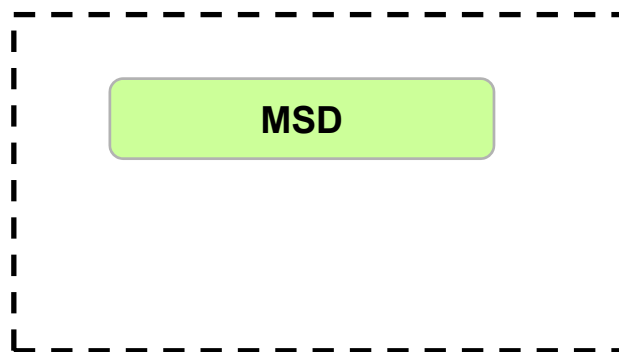
Windows Explorer
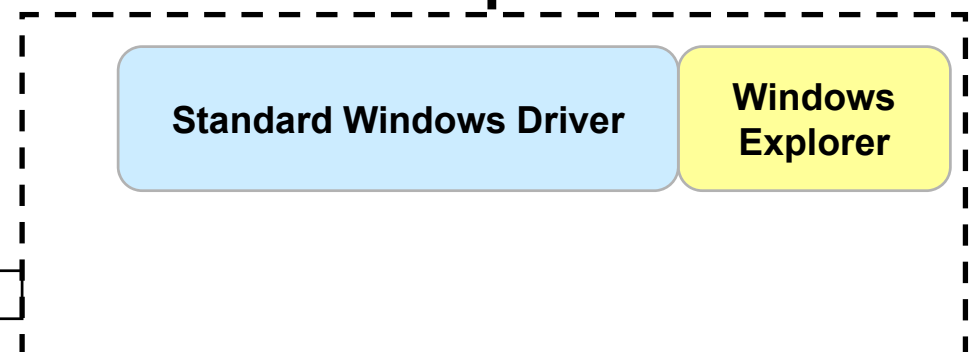
USB Cable

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

COM 3101

Custom Class Driver

PIC® Microcontroller

Generic Driver Demo

Bootloader

USB Cable

PC Computer

MCHPUSB.sys | MPUSBAPI.dll | PDFSUSB.exe

Binary Only

Source and Binary

Other C++ Examples

INF File Required
(Supplied in MCHPFSUSB)

Source and Binary
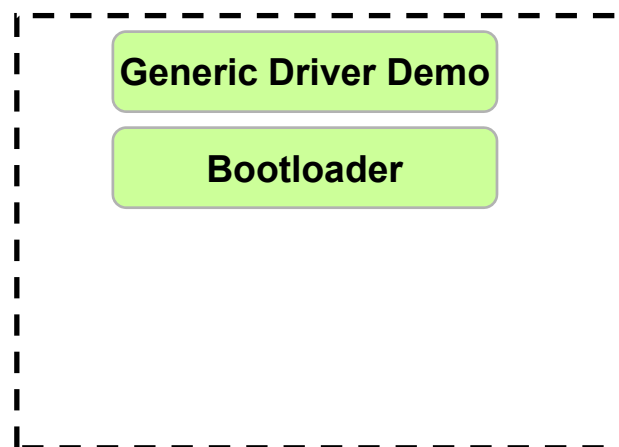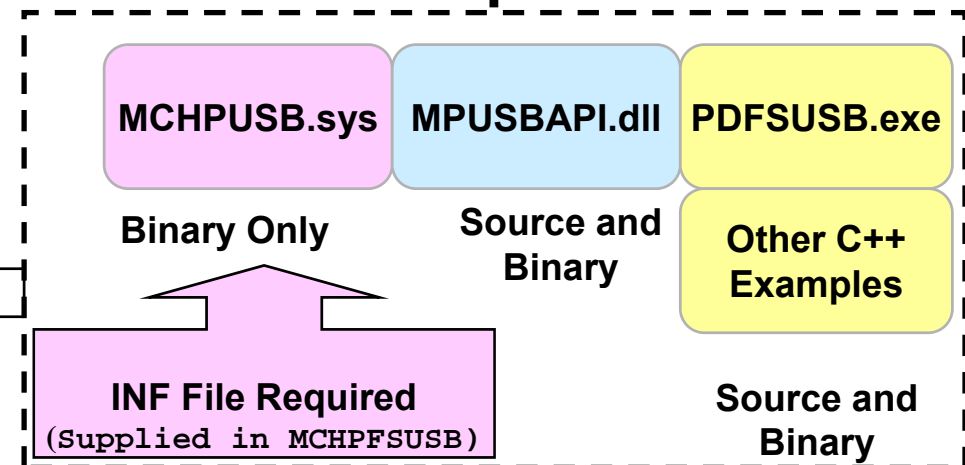
Design Considerations:

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

COM 3101

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**

Libraries Source Files Directory
C:\microchip_solutions_v2013-xx-xx\Microchip

- Microchip
  - Common
  - Graphics
  - Help
  - Image Decoders
  - Include ← **Common Header files (Libraries)**
    - Graphics
    - Image Decoders
    - MDD File System
    - mTouchCap
    - PIC18 salloc
    - TCPIP Stack
    - USB ← **USB Header files**
  - MDD File System
  - mTouchCap
  - PIC18 salloc
  - TCPIP Stack
  - USB
    - Audio Device Driver
    - CDC Device Driver
    - CDC Host Driver
    - Charger Host Driver
    - Documentation
    - Generic Device Driver
    - Generic Host Driver
    - HID Device Driver
    - HID Host Driver
    - MSD Device Driver
    - MSD Host Driver
    - Printer Host Driver

Function and client drivers

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)**

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 PC Tools

- **Microchip USB OTG Configuration Tool**

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

- **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

USB Stack Configuration Tool

Main:
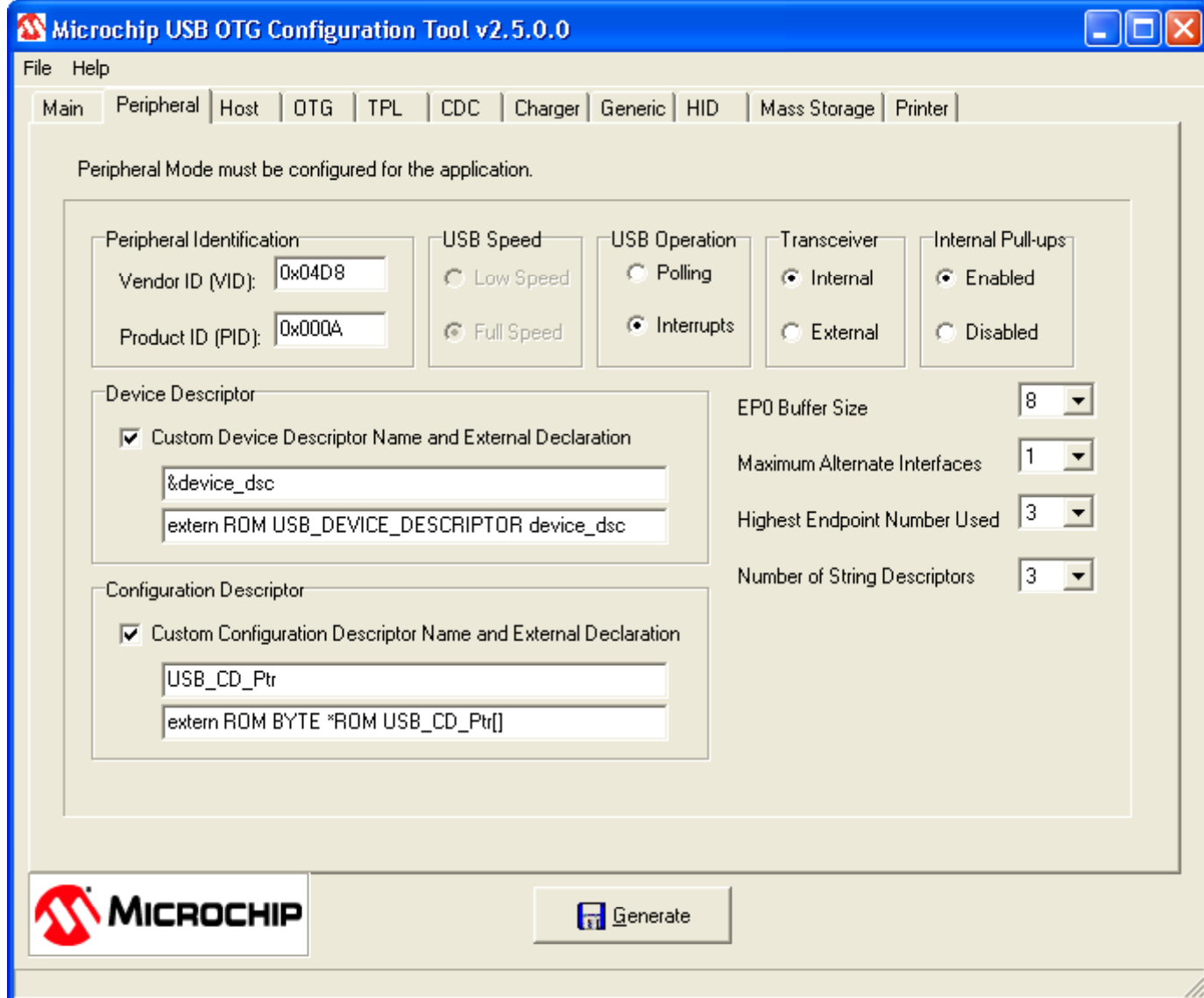
- **Target Device**
- **Device Type**
- **Ping-Pong Mode**
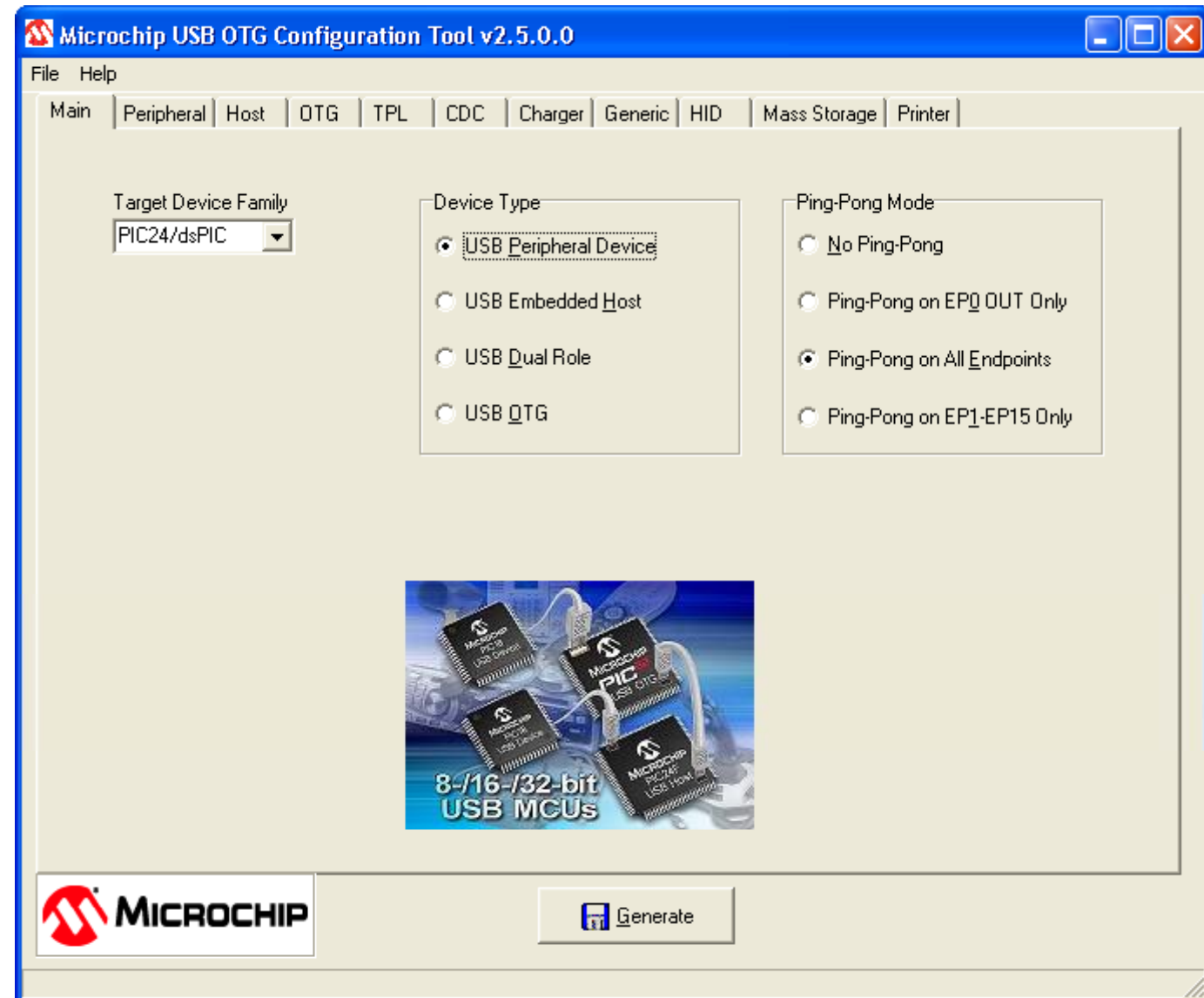
USB Stack Configuration Tool

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**
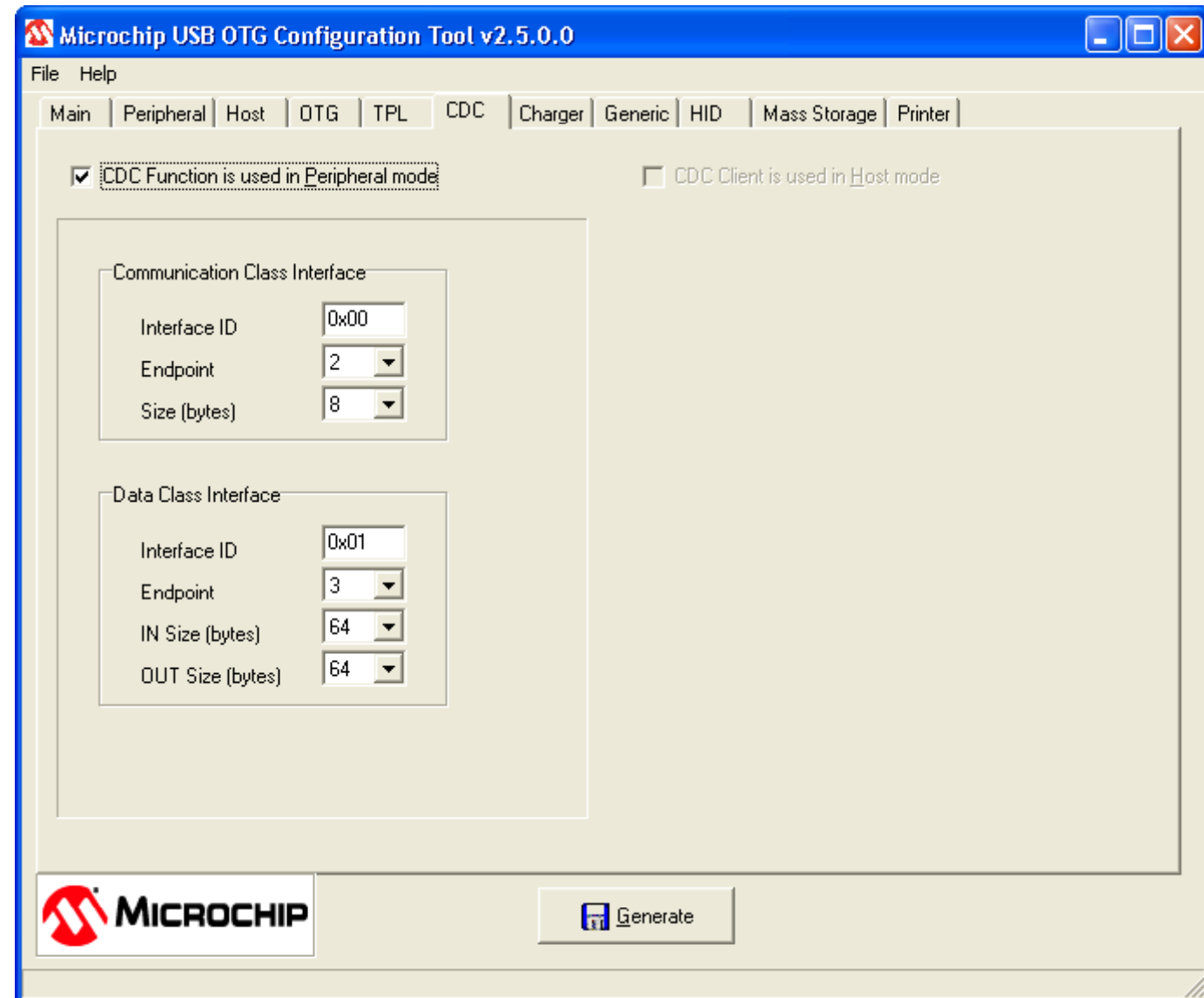
`mchpcdc.inf`

- **`.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"
```

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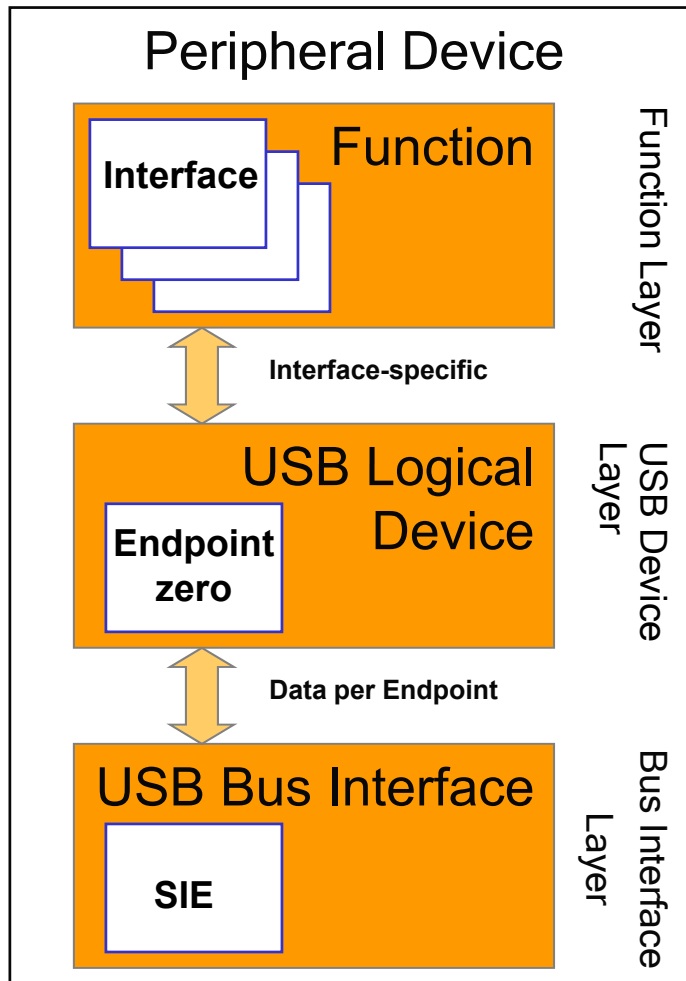
- **Function**:
  `usb_function_audio.c`
  `usb_function_cdc.c`
  `usb_function_generic.c`
  `usb_function_hid.c`
  `usb_function_msd.c`

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

Generic USB Device Project
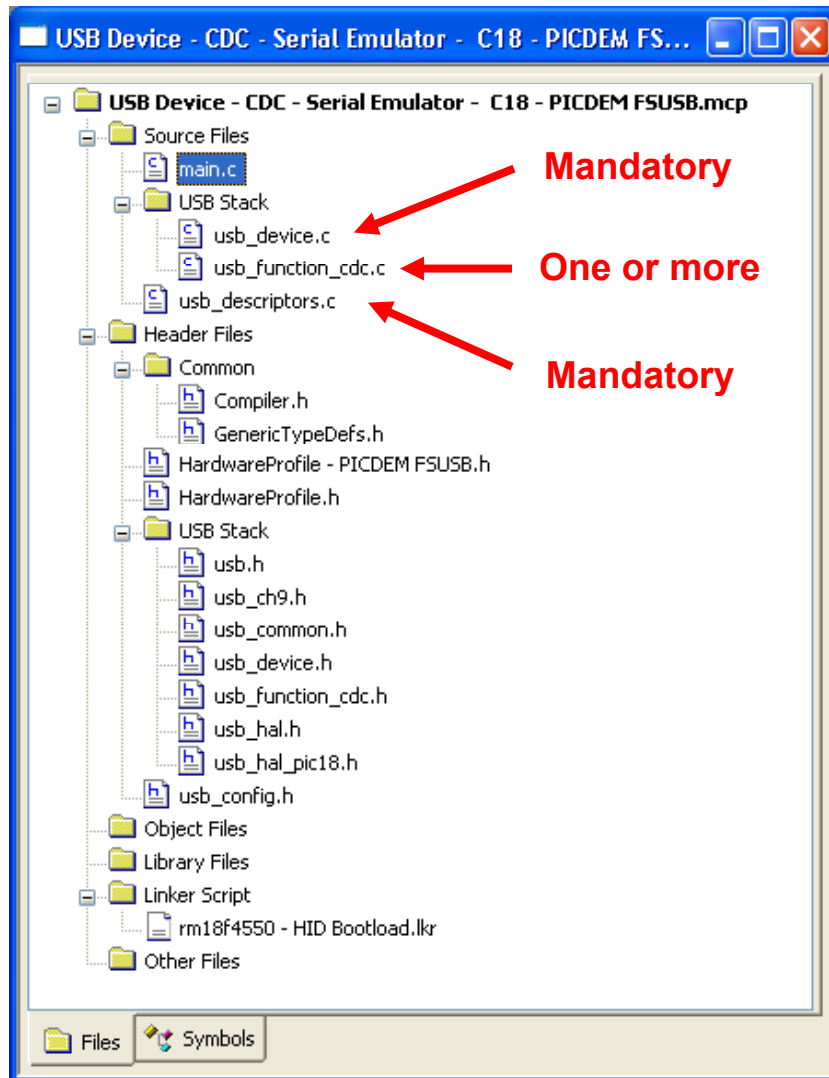
- **Must have `main()` function (`main.c`)**

- **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**

Lab Prepare

- **Install MPLAB IDE (v8.92)**

- **Install MPLAB C18 Lite (v3.45)**

- **Install Microchip Application Libraries (v2013-xx-xx)**

COM 3101
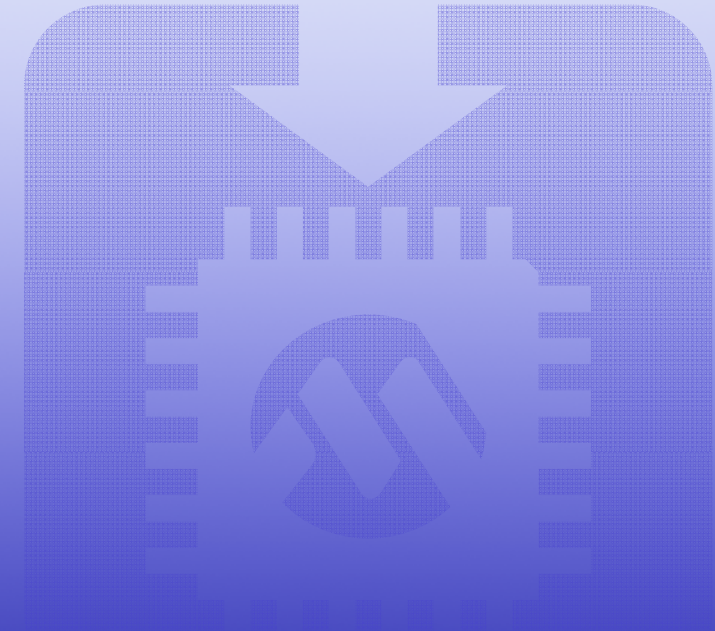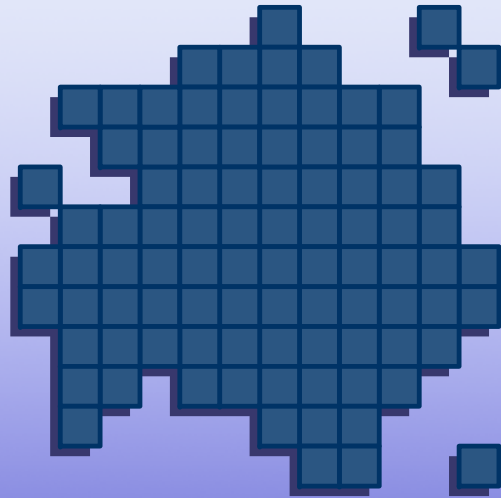
Lab Exercise 1a

Create an USB template project (Polling)

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 -

Reset → main() → InitializeSystem() ← You edit UserInit()

InitializeSystem() → while(1)

while(1) → USBDeviceTasks() ← USB Stack

USBDeviceTasks() → ProcessIO() ← Your application code

ProcessIO() → Function Services ← CDCTxService()
MSDTasks()
Re-arm OUT Endpoint
(HID & Generic)

Cooperative Multitasking!!

No blocking functions.

Use state machine.
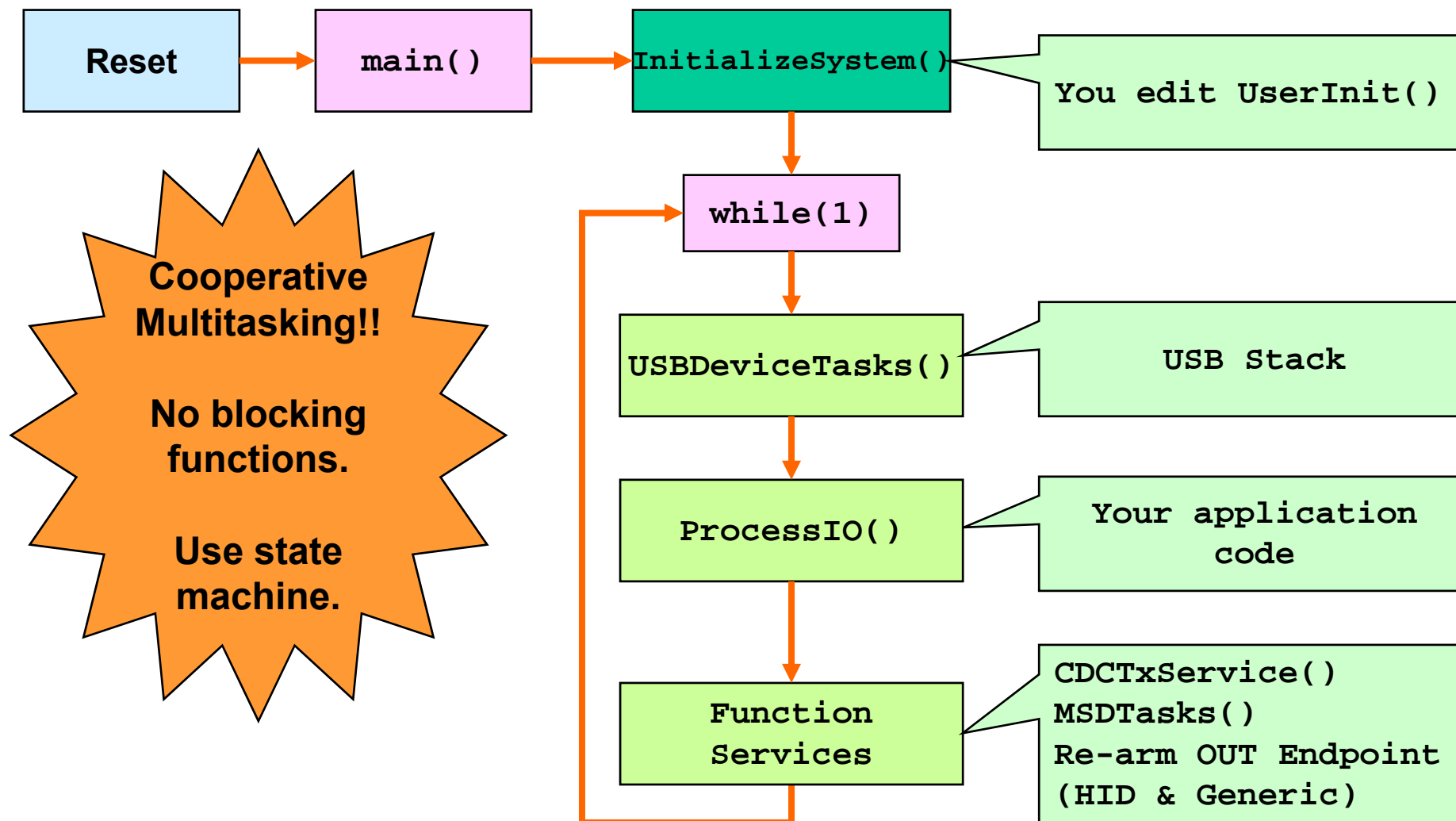
Code Example

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
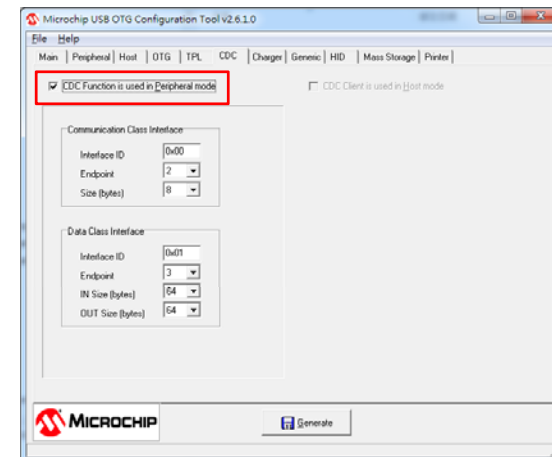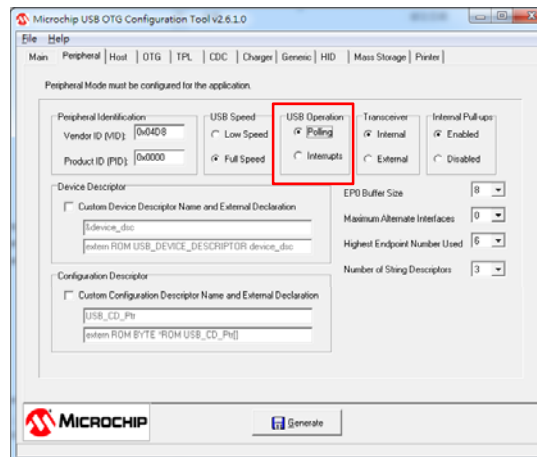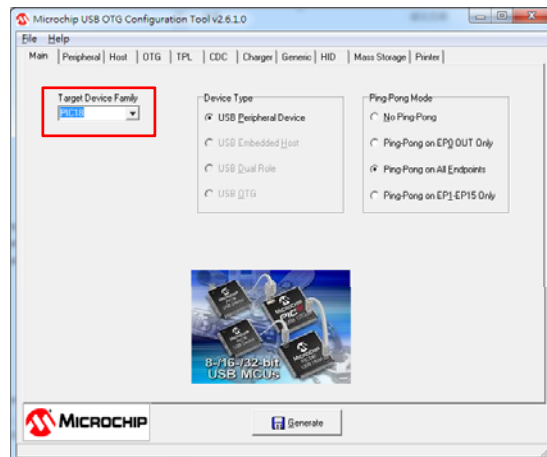
Lab Exercise 1a-2

- **Use MPLAB IDE project wizard to create a project.**
  MCU:PIC18F47J53, Compiler MPLAB C18.

- **Add 2 "include file path" to *Build Options***
  "\Exercise\Lab1a\" and
  "C:\Microchip_Solutions_v2013-xx-xx\Microchip\Include\"

- **Add 1 "Preprocessor macro" to *Build Options***
  "APP1618_SD_18F47J53"

- **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 ...**

Lab 1a Results

- **System can find a new USB device. When plug into USB port.**

- **The device product name is your name.**
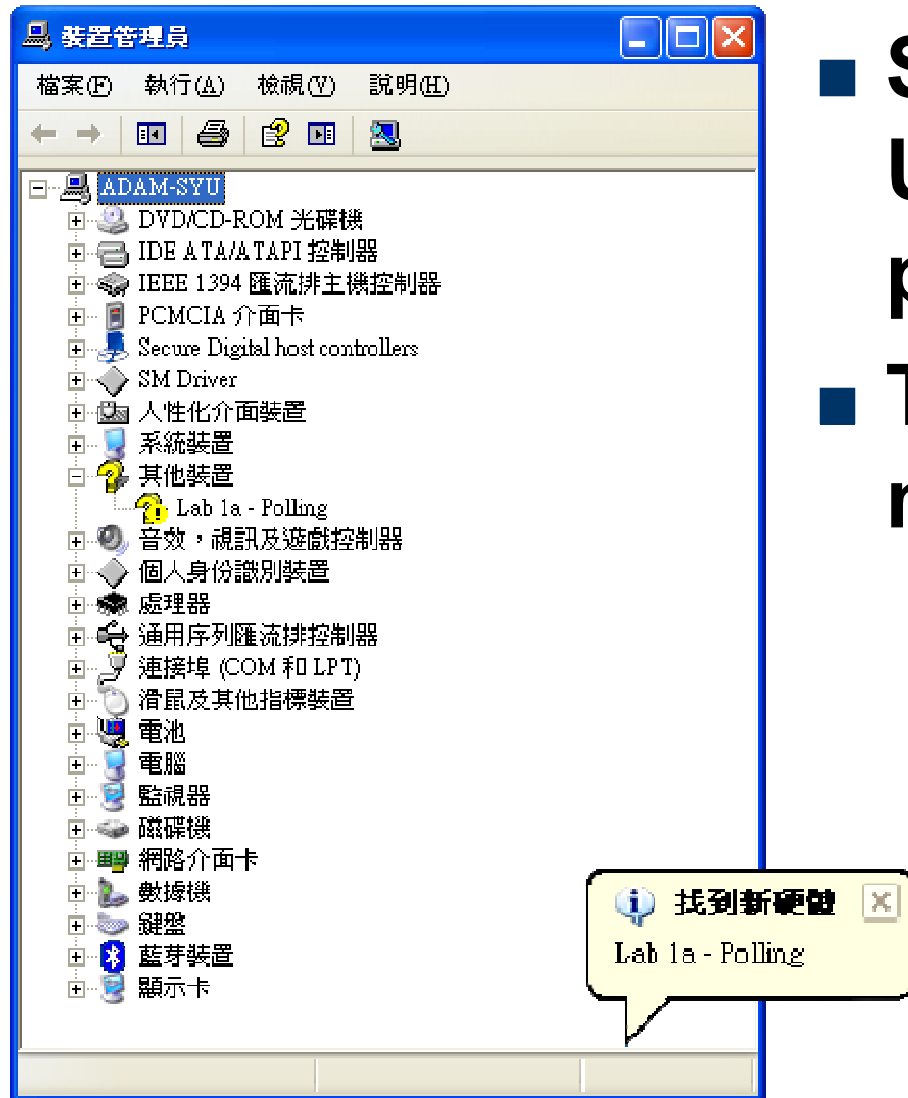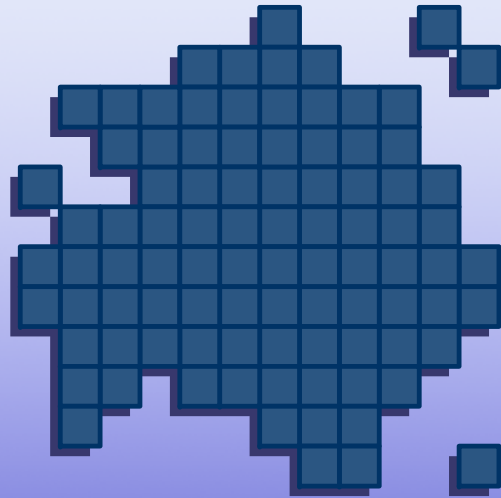
Lab Exercise 1b
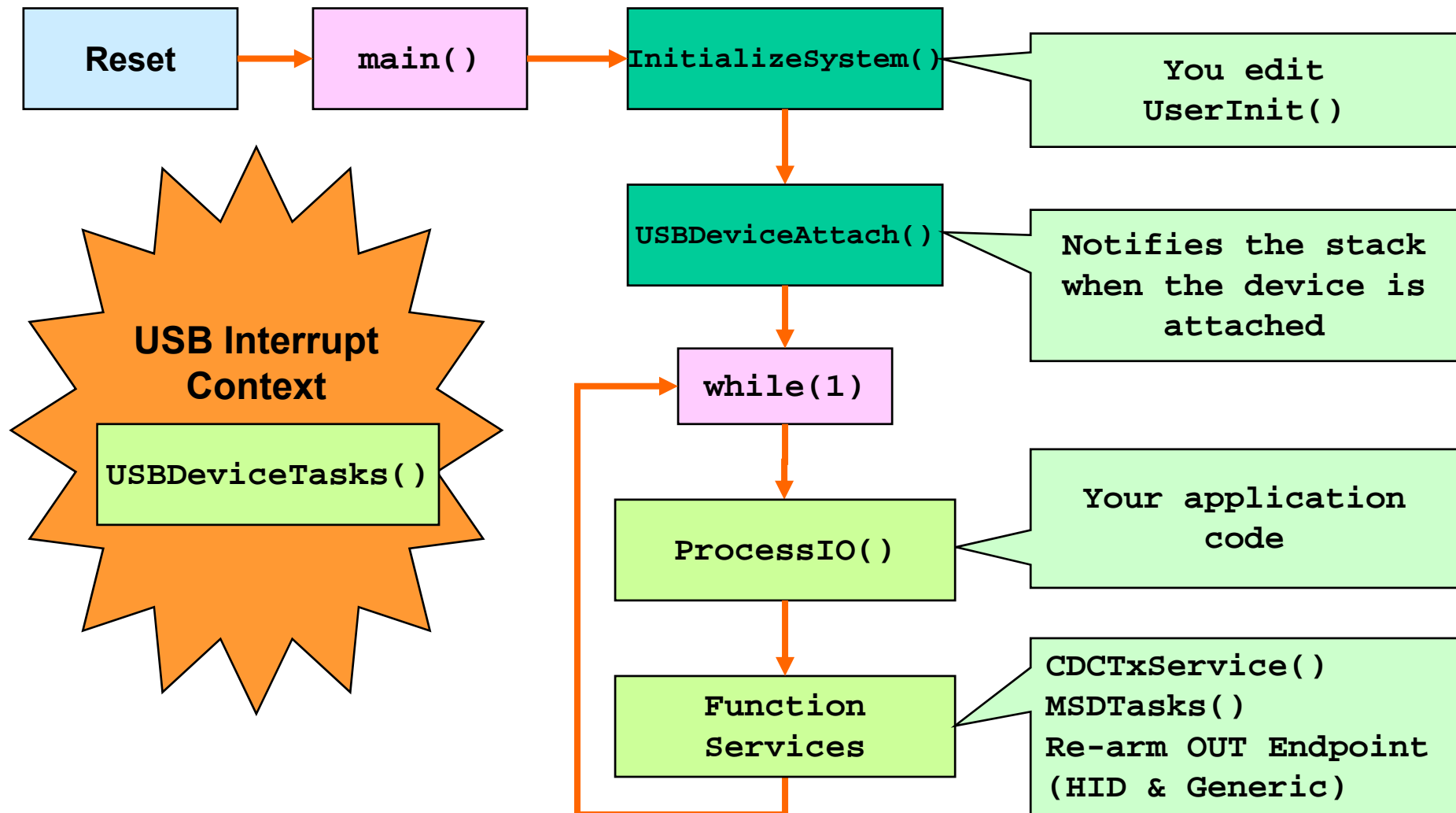
Create an USB template project (Interrupts)

Lab Exercise 1b

- ## Objective

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

COM 3101

MCHPFSUSB Framework
- Interrupt Program Flow -

Reset → main() → InitializeSystem()

InitializeSystem() → You edit UserInit()

USBDeviceAttach() → Notifies the stack when the device is attached

InitializeSystem() → USBDeviceAttach() → while(1) → ProcessIO() → Function Services → (loops back to while(1))

ProcessIO() → Your application code

Function Services → CDCTxService() MSDTasks() Re-arm OUT Endpoint (HID & Generic)

USB Interrupt Context

USBDeviceTasks()

Code Example

Main.c

```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
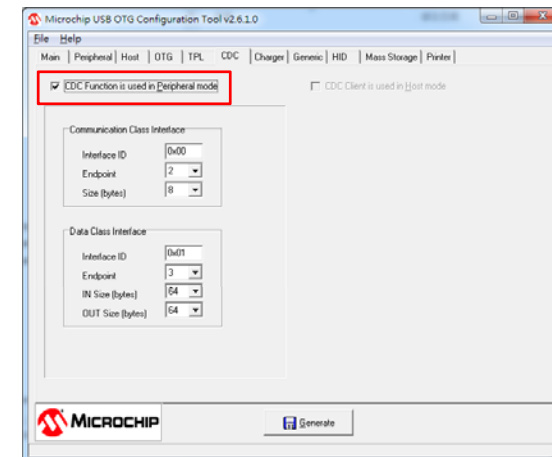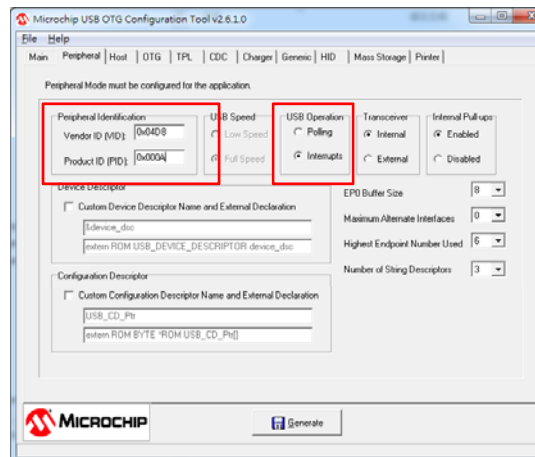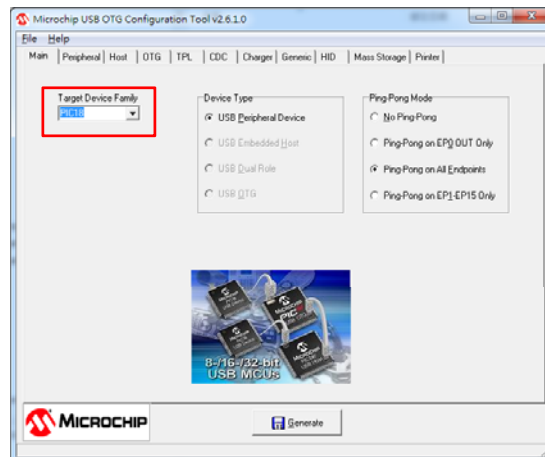
Lab Exercise 1b-2

- **Use MPLAB IDE Open the project**
- **Modify and add some code to main.c .**
  void main( )
  {
     while(1)
     {

```
#if defined(USB_INTERRUPT)
        if(USB_BUS_SENSE && (USBGetDeviceState() ==
                DETACHED_STATE))
          USBDeviceAttach();
#endif
```

  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_On(); break;
                case 0:
                        if(!sw) mLED_1_Off() ;break;
                default:
                        mLED_1=0;
        }

        CDCTxService();
}
void UserInit( )
{
        mInitAllLEDs();
        mInitAllSwitches();
}
```
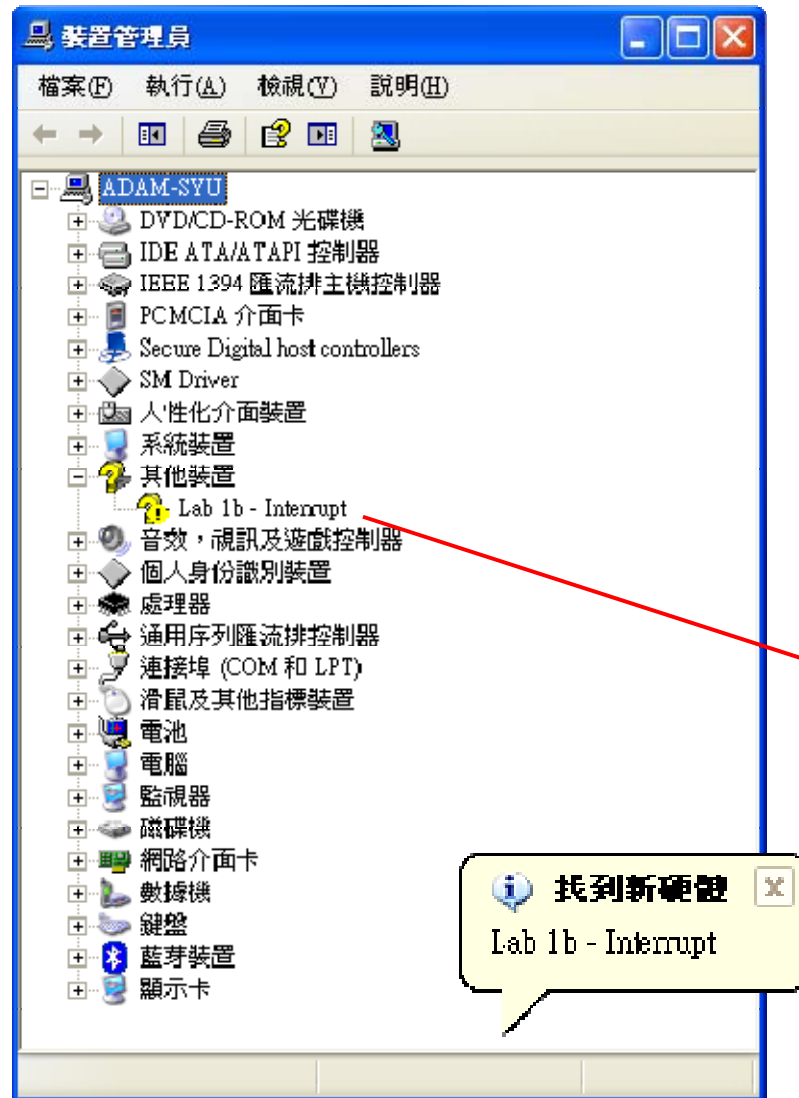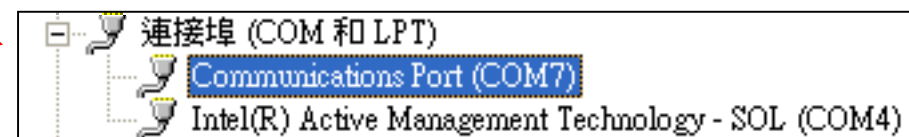
- **Build & Program …**

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.**

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

```c
/* 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
…
```

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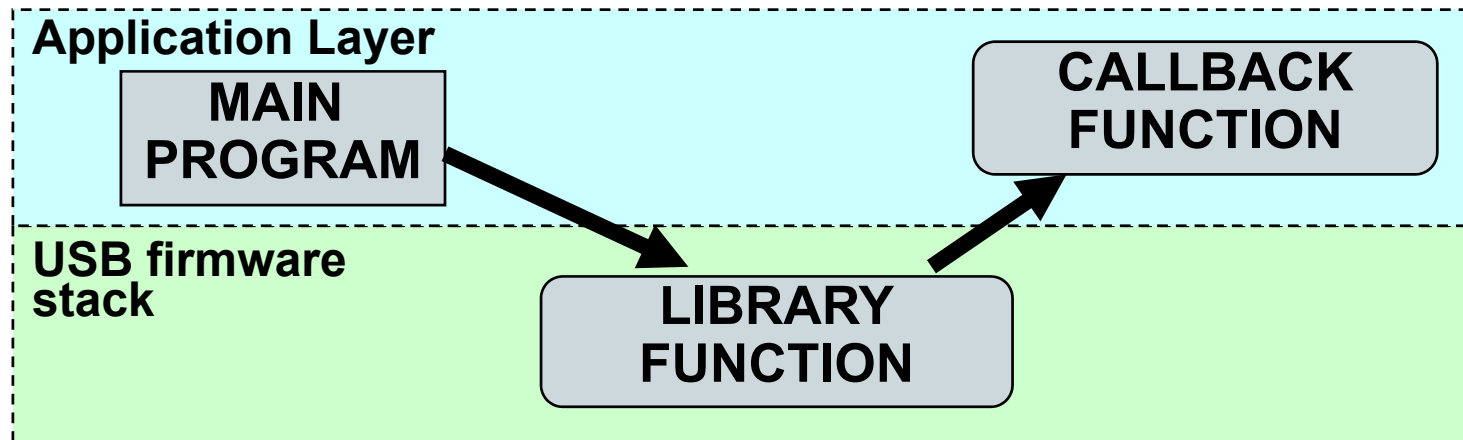
- **You can modify that callback function to take appropriate actions for each of these conditions**

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.

Example
- CDC device -

```c
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;
}
```
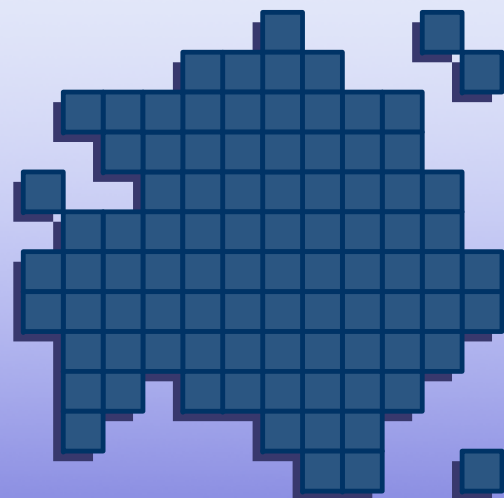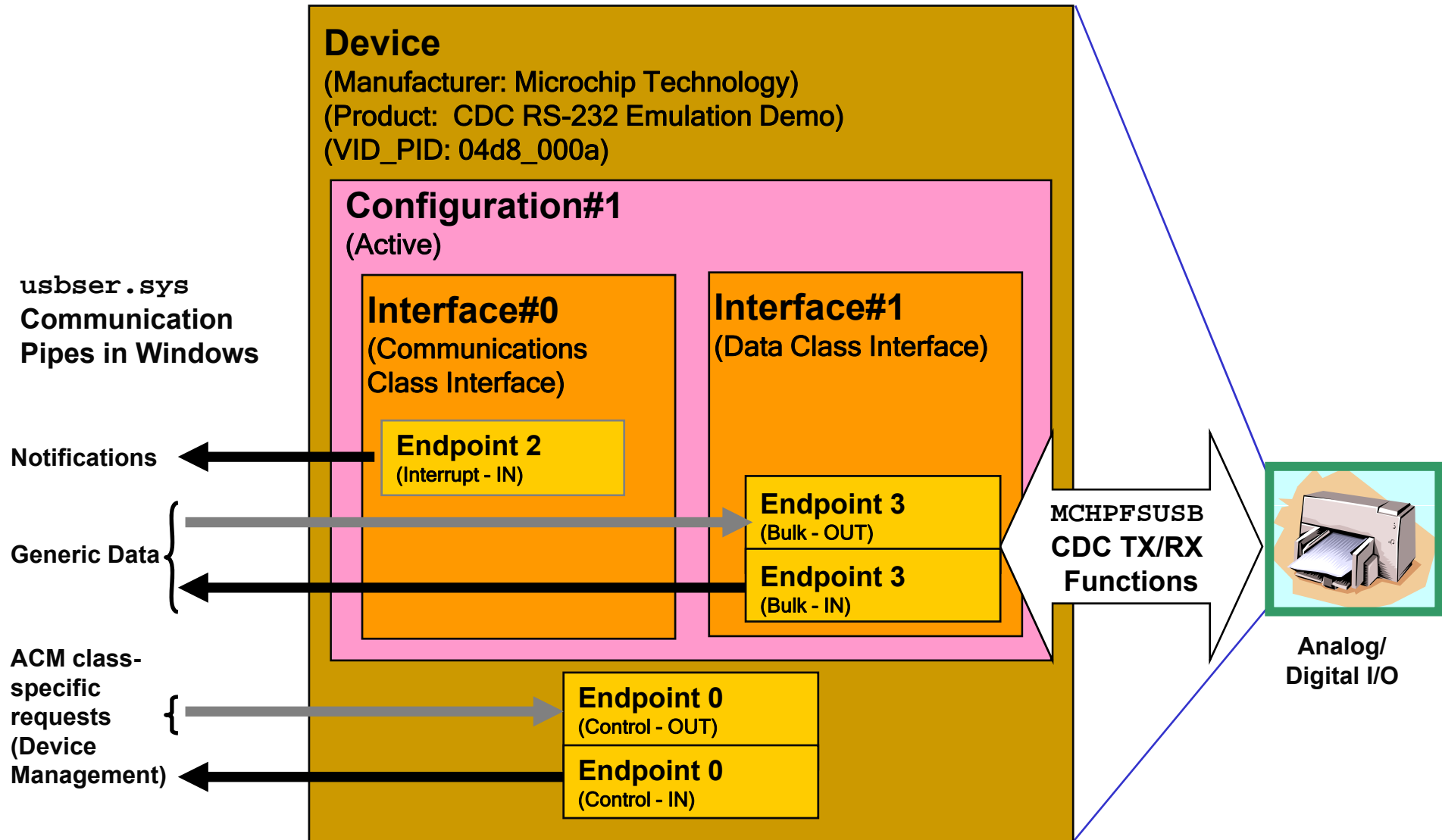
Part 3.

Using the Microchip CDC
Class Device Framework

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**

- **Serial Emulator (Demo)**
  - **Lab Serial Emulator**

CDC Elements

■ **There are three classes that make up the definition for communications devices:**

- ■ **Communications Device Class**
  - ■ **Communications Interface Class**
  - ■ **Data Interface Class**

CDC-ACM Pipes

Device
(Manufacturer: Microchip Technology)
(Product: CDC RS-232 Emulation Demo)
(VID_PID: 04d8_000a)

Configuration#1
(Active)

`usbser.sys`
Communication Pipes in Windows

Interface#0
(Communications Class Interface)

Interface#1
(Data Class Interface)

Endpoint 2
(Interrupt - IN)

Notifications ←

Endpoint 3
(Bulk - OUT)

Endpoint 3
(Bulk - IN)

Generic Data {

Endpoint 0
(Control - OUT)

Endpoint 0
(Control - IN)

ACM class-specific requests (Device Management) {

`MCHPFSUSB`
CDC TX/RX Functions

Analog/ Digital I/O

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**
- **Serial Emulator (Demo)**
    - **Lab Serial Emulator**

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

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");
  }
  ```
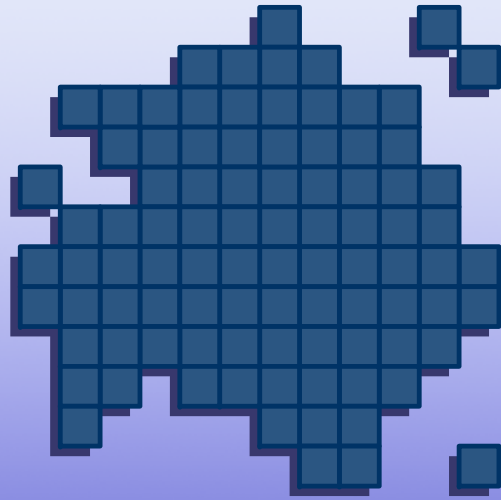
Lab Exercise 2

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 -

MICROCHIP

Regional Training Centers

- **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
}
```

COM 3101

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"
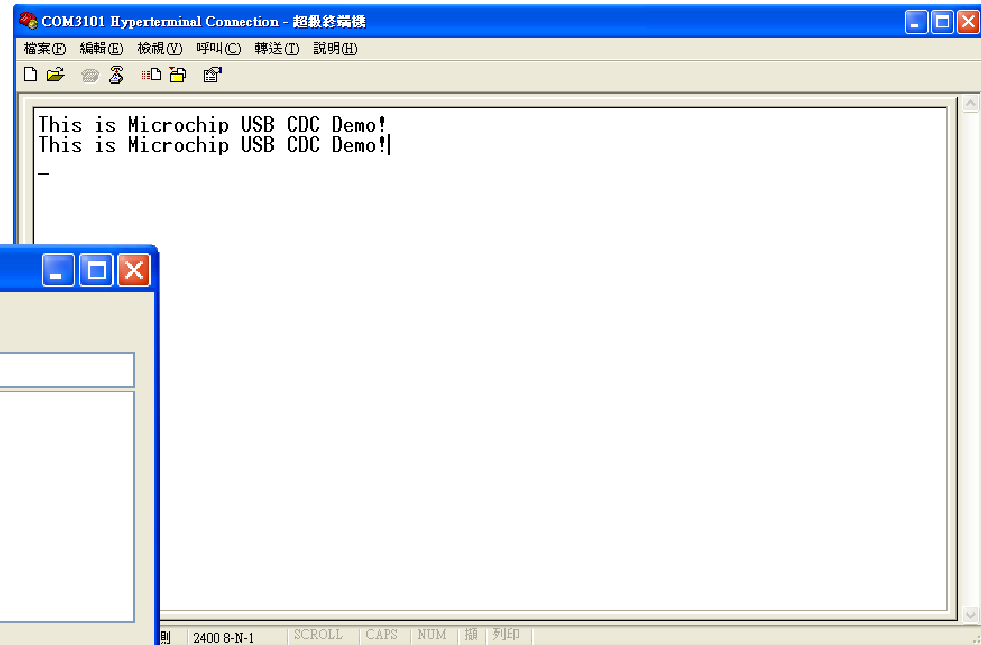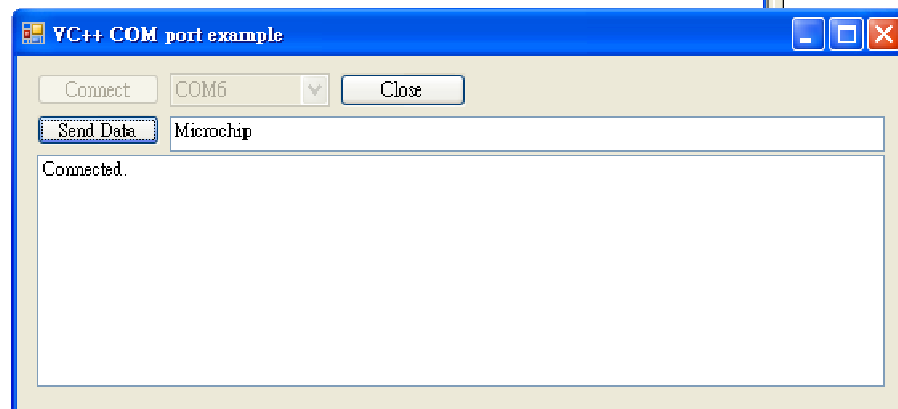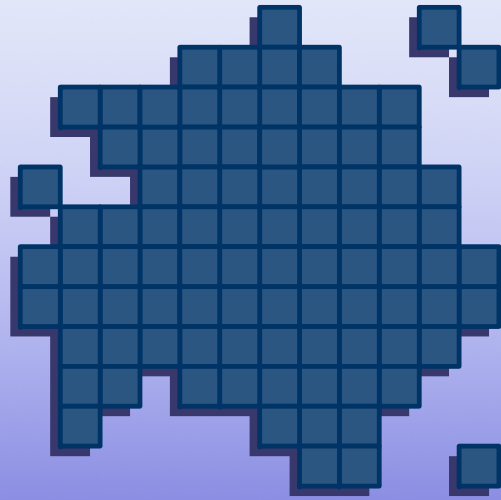
Lab Exercise 3

Sending and Receiving raw data

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}"
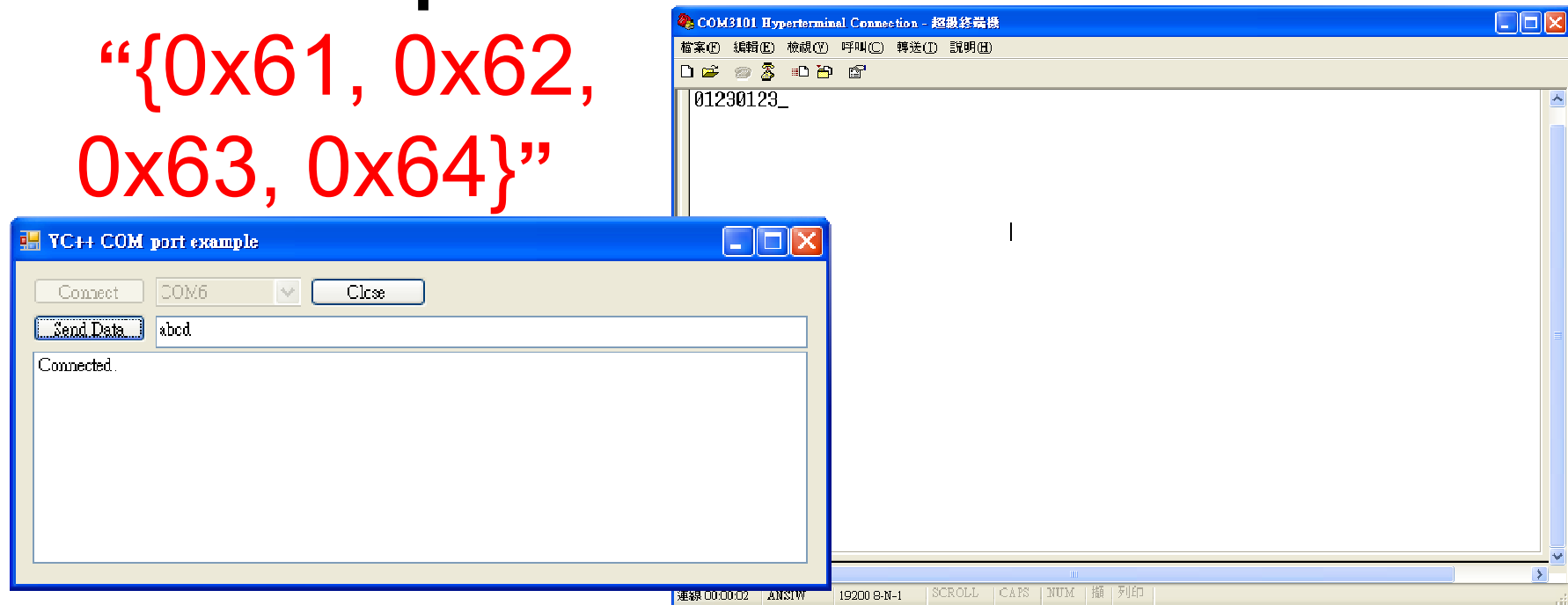
Typical PIC® MCU Application

PIC MCU based Application
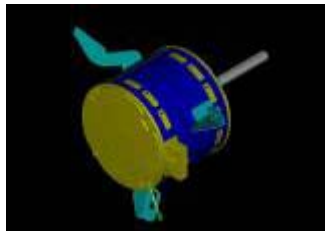
New Communications via USB

A/D, PWM, SPI, I²C™, I/O, etc.

Old Communications via RS232

Trademarks

Thank You.