


Slide 1



Welcome to “Getting Started with Microchip’s Low Pin Count USB Solutions”. This self-directed course is intended to provide the user with a quick overview of the USB, introduce the new Low Pin Count USB Development kit, and Microchip’s Full-Speed USB Firmware Framework to ease the development of your own USB applications quickly.

## Slide 2




### Class Prerequisites

- Attendees should have a the following:
  - A general knowledge of the Universal Serial Bus (USB)
  - A working knowledge of the C programming language
  - Familiarity with Microchip's High Performance PIC18 Microcontrollers

Getting Started with Microchip's Low Pin Count USB Solutions Slide 2

In order to fully benefit from this self-directed course the user should have a very basic knowledge of the USB, have programmed in C, and be familiar with Microchip's High Performance PIC18 Microcontrollers. Once completed, the user should complete the Project Labs listed in the Low Pin-Count USB Development kit user's guide.

## Slide 3



### Agenda

- High-level overview of the USB and how it relates to the PIC18F1XK50 Device
  - Physical and Logical Topologies
  - “Plug and Play”
  - Communication
- Overview of Microchip’s Low Pin Count USB Solutions
  - Low Pin-Count USB Development Kit
  - Microchip’s Full Speed USB Firmware Framework

Getting Started with Microchip’s Low Pin Count USB Solutions Slide 3


This class will begin with an overview, albeit moderately high-level, of the USB. This is a complex protocol. Therefore, you should not feel discouraged if you don’t understand everything the first time through this class. There is a lot of information to cover. However, the good news is that with Microchip’s Low Pin-Count USB Solutions, you don’t have to be an expert to get a USB application up and running quickly. Firmware development is made easier when using Microchip’s Full Speed USB Firmware Framework as it abstracts much of the protocol’s lower level function. Use the first part of this class as general familiarization with protocol. This is all made possible by the feature rich USB peripheral found on the PIC18F1XK50 used in this class. Once you have read through the class, it is recommended that you move on to the Low Pin-Count USB Development Kit User’s Guide. The guide contains a number

of project labs that will help reinforce concepts discussed here and get you programming your first USB applications.

Slide 4




## Slide 5



### Universal Serial Bus

- Auto detection & configuration (Plug-&-Play)
- Easy expansion using hubs
- Can be Bus powered
- Data CRC protected
- Three speeds:
  - Low- 1.5 Megabits / second
  - Full- 12 Megabits / second
  - High- 480 Megabits / second



Getting Started with Microchip's Low Pin Count USB Solutions Slide 5

In recent years, the USB has nearly completely replaced the serial and parallel ports on PCs. The speed flexibility and plug & play convenience have won the hearts and minds of users. Accessory vendors have converted over and it's nearly impossible to find a device with a serial or parallel port.

USB has several advantages from the user's standpoint.

It supports hot plugging. The host detects new devices and configures them for use.

Data integrity is ensured, Data is CRC protected and bad packets are automatically resent. Thus an application can rely on the quality of the data. Older standards made no such guarantee which put the integrity of the data at risk.

The USB cable can provide power for the device, guaranteed 100mA and can negotiate up to 500mA. This reduces the number of wall transformers needed on the desk.

Data to or from a device is protected by CRC and any bad packets are automatically resent. So applications can rely on the integrity of their data from end to end.

USB 2.0 now supports 3 speeds.


Low speed is designed for “human input devices” – keyboard, mouse, POS terminals. People are slow compared to computers so Low Speed devices relax timing requirements so cheaper components such as the clock source and cable may be used.

Full speed was designed for moving bulk data – cameras, thumb drives, data loggers, external disk drives. The clocking runs at 12MB/sec but with overhead and traffic, maximum throughput is around 1MB/sec.

High speed was designed to compete with FireWire with high volume data, particularly video. Clocking runs at 480MB/second. Now of course many of the cameras and thumb drives and removable disks are using High speed USB.

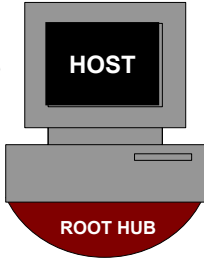
All of microchip’s current flash USB offerings support the 2.0 Full speed standard.

## Slide 6



### USB Host and Root Hub

- Coordinating entity for the USB
- Initiates data transfers
- Configures attached devices
- **ONE HOST/Bus**
- Commonly a PC




The diagram shows a computer monitor labeled 'HOST' sitting on a base labeled 'ROOT HUB'. The base is a semi-circle with a red gradient.

Getting Started with Microchip's Low Pin Count USB Solutions Slide 6

The USB is Host centric. The coordinating entity for the USB, the host initiates all data transfers through a polling mechanism. Each device is polled or interrogated as to whether it is ready to receive data from the host or if it has new data to transmit to the host. In this class, the PIC MCU used will be used as a device and the Host will be the PC. Note that only one Host is allowed on a single USB.

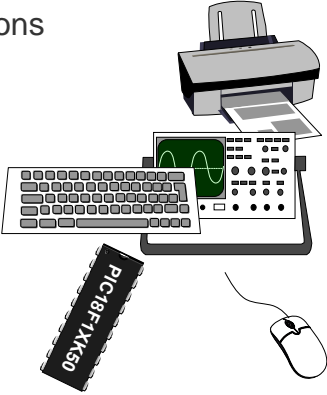


## Slide 7



### Device

- Provides USB functions
- Bus or Self-Powered
- Carries and reports configuration information



Getting Started with Microchip's Low Pin Count USB Solutions Slide 7

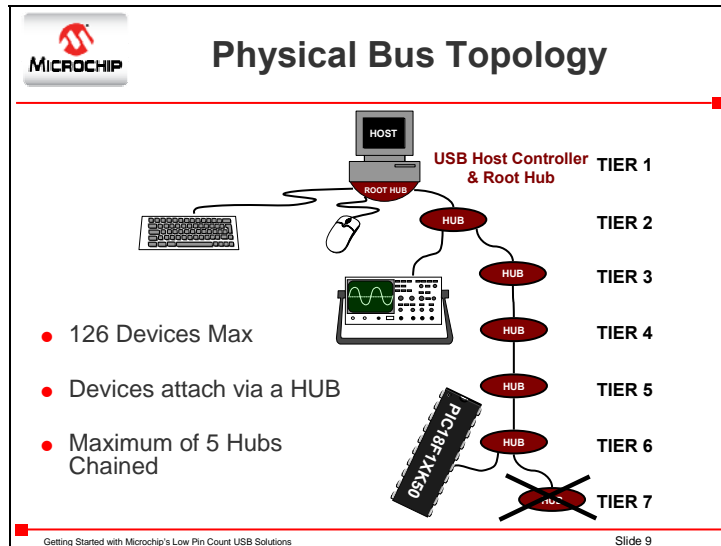
Devices connect to the Host via the USB. The device will provide the Host pertinent information about itself so that the Host will configure it accordingly. The device can obtain power from the USB, within limits, or be self-powered. This information is relayed to the Host at the time of connection to the USB.

Slide 8



Referring to the USB 2.0 specification, two important topologies must be considered.

Slide 9



The first topology is the physical interconnect between Host and Device via a USB cable connected to a port or hub. A number of hubs can be connected together.

The USB physical interconnect is a tiered star topology, with a Hub at the center of each star.

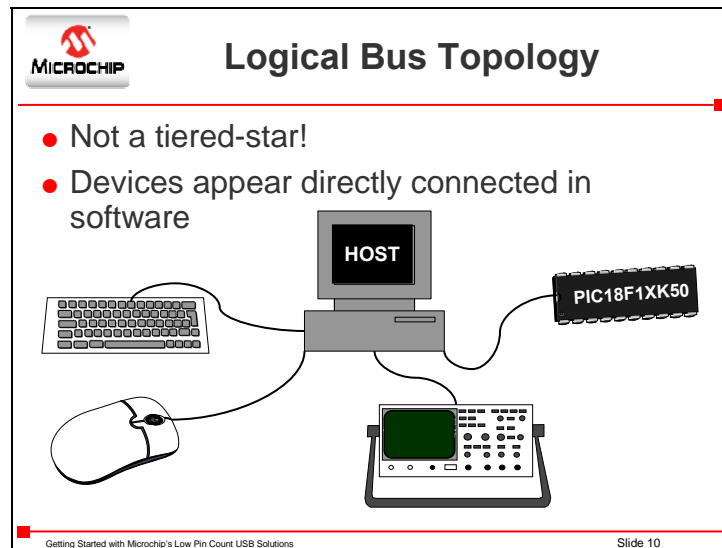
Hub: allows the connection of additional USB devices. A maximum chaining of 5 hub levels to the root hub. Any more and timing requirements would be exceeded negatively impacting data transmission.

Typical Ports per hub: 4

-----  
The PIC18F1XK50 is designed to be a peripheral. This means that it cannot be a hub or host.

Cable Segments can be as long as 5 meters. So, max distance from PC using 5 hubs is 30 meters.

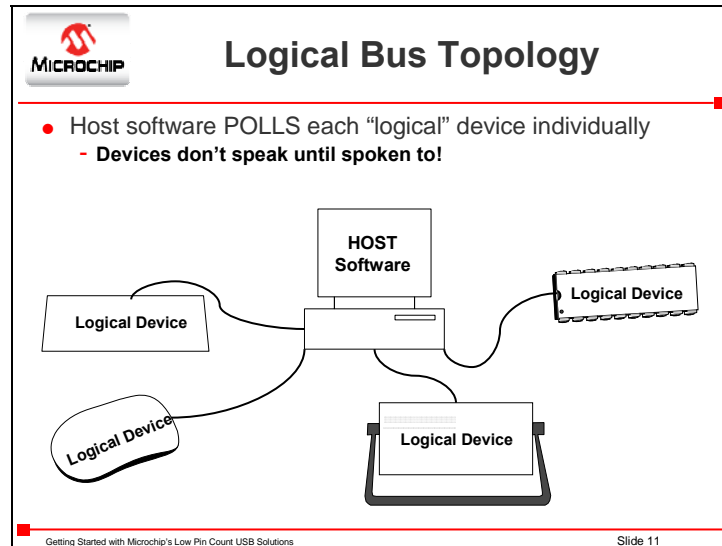
## Slide 10



The second topology describes how the software on the Host will actually perceive connected devices. Host software sees each device as if it were directly connected to the root hub with hub levels ignored.

\*Note: Even though most communication flows maintain this perspective, the host still maintains an awareness of the physical topology to support processing the removal/insertion of Hubs.

Slide 11



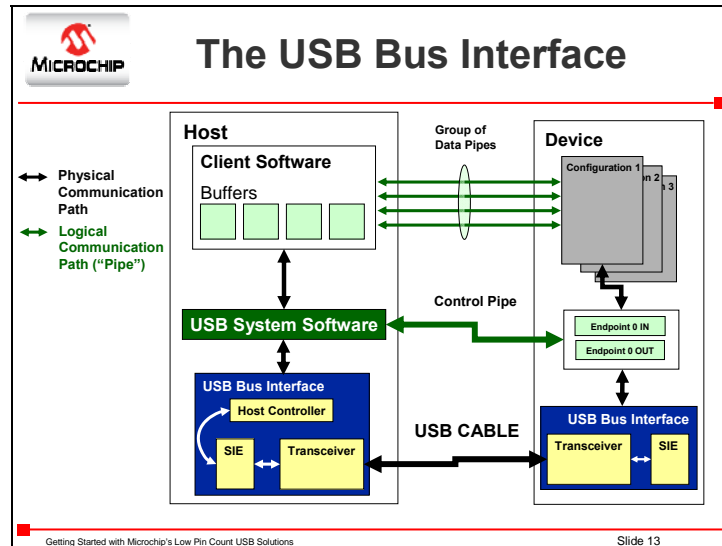
As mentioned before, the Host Software will poll each logical device. The device is interrogated as to whether it is ready to receive or transmit any new data. Data transmission on the bus will only occur if the Host allows it.

Slide 12



Let's expand on the Logical Bus Topology by looking closer at the software view of the hardware.

Slide 13



The USB 2.0 specification document describes the USB interface as shown in the slide. Notice that this diagram indicates two types of communication paths:

Logical communication represents how the software on the Host “sees” the data path to the Software on the device using what are know as Pipes. The second communication path represents the actual physical connection of the device to the Host through the USB cable.

It is important to remember that all communication takes place over the USB cable. As this course proceeds and/or referencing the USB specification, these concepts will prove helpful.

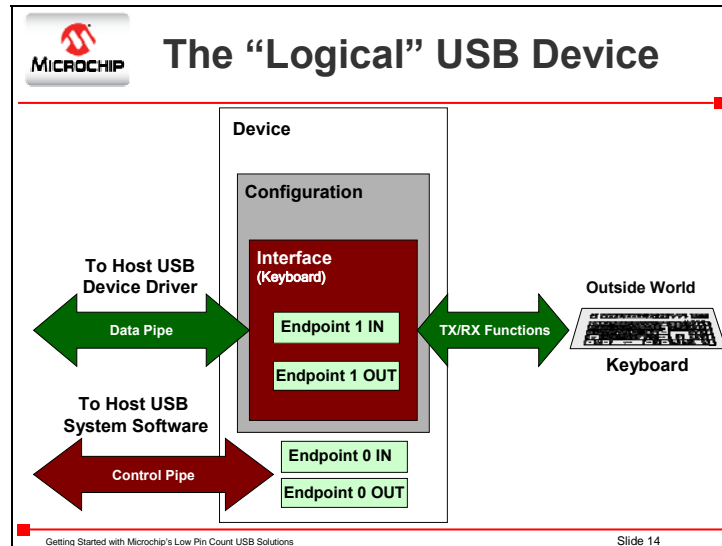


Referring to the Logical communication path, notice that a Host component called the Client Software communicates via groups of data pipes to components on the device known as configurations. It is within the Client Software that Drivers are found. Each configuration on a device will require a driver. This will be discussed in greater detail later in this class.

The second logical communication path is the control pipe which interfaces the USB system software to a special buffer on the device known as endpoint0. It is here that the Host tells the device which of its configurations to use. More on this in the next slide.

The USB cable connection is represented by the Physical Communication path. This path interfaces components on the Host to components on the device necessary to enable communication between the two. Later, we will step through each of these physical hardware components and show how these relate back to the PIC18F1XK50 MCU device.

Slide 14



USB Devices are complex and made up of several logical units. The relationship between these units can be described as follows:

Each device must contain a special buffer called Endpoint 0, and hardware/software in the device that responds to standard requests made by the USB Host System Software to/from this endpoint. "Standard requests" are covered in Ch9 of the USB Spec.

Devices must consist of 1 or more "Configurations"

Only 1-configuration can only be enabled at a time

Configurations must consist of 1 or more "Interfaces"

Each Interface represents a basic device “function” (mouse, keyboard etc), and so, this is what the host USB device driver communicates with. Specifically, the host device driver (Human Interface Device “HID” for Keyboard) communicates to the “Keyboard” in the device via the “Keyboard” interface.

Interfaces must consist of 1 or more “Endpoints”

An Endpoint is a buffer through which data is transferred by the host & device

An Endpoint is uniquely addressed by: a number (1-15) and direction (IN/OUT) i.e EP1-IN

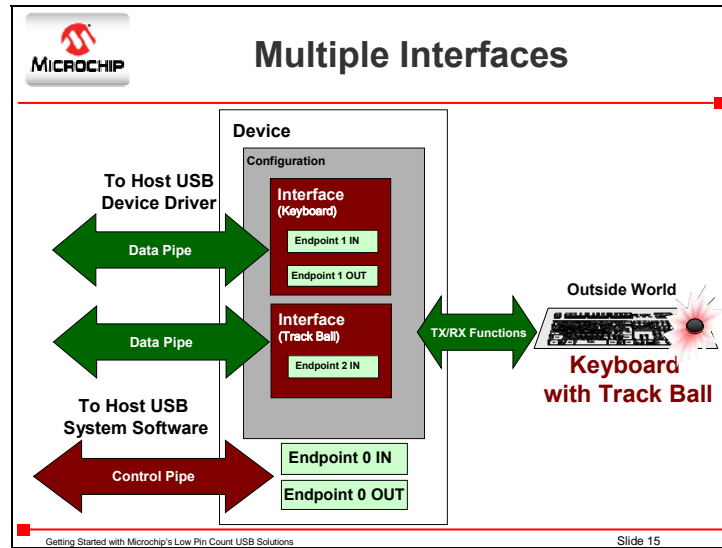
An Endpoint can only carry data in one direction, either from the host to the device (called an OUT endpoint) or from the device to the host (called an IN endpoint). Endpoints can be thought of as unidirectional pipes.

Each Endpoint must have a USB transfer type associated with it (CONTROL, INTERRUPT, BULK, ISOCHRONOUS)

Ultimately, device firmware must implement the function by communicating with the real/world device (keyboard) and send/receive buffers to/from the endpoints using class-specific TX/RX functions.

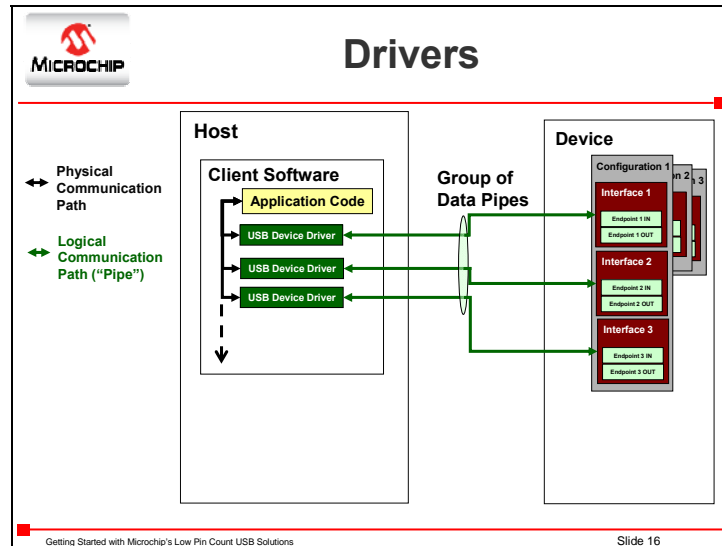
Device+Configuration+Interface+Endpoint settings are stored in ROM data structures in the device called “Descriptors”. The USB System Software reads these descriptors during enumeration to understand the requirements of the device, and set up the physical & logical connections to the device. More on this later...

Slide 15




Here is an example of a device with multiple interfaces in a single configuration.

Slide 16

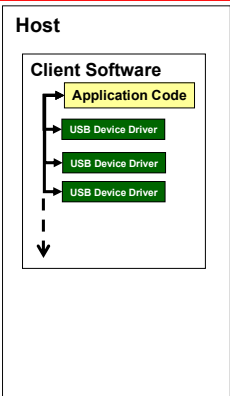


As was mentioned previously, the client software contains specific drivers used to enable communication with a specific device interface through the Data Pipes.

## Slide 17



### Drivers



- Many Windows® drivers already exist for common devices
- USB specification groups drivers into “classes”
  - Devices that share common attributes

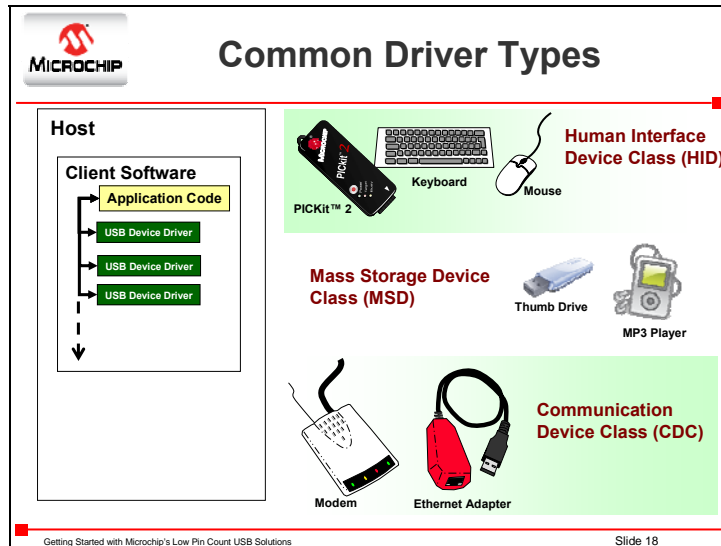
Getting Started with Microchip's Low Pin Count USB Solutions Slide 17

The diagram illustrates the software stack on a host. It shows a 'Host' container with a 'Client Software' sub-container. Inside 'Client Software', there is a yellow box for 'Application Code' at the top, followed by three green boxes for 'USB Device Driver'. Arrows point from the 'Application Code' to the first 'USB Device Driver', and from each 'USB Device Driver' to the next one below it. A dashed arrow points downwards from the bottom 'USB Device Driver' towards the bottom of the 'Host' container.

A Windows based PC already comes equipped with many drivers for common devices available. This frees the developer from having to write a driver for their specific device. However, the developer needs to make their device application compatible with the available drivers.

The USB specification groups these specific drivers into what are called classes. Devices in a specific class share similar attributes as shown on the next slide.

Slide 18



Here are some examples of common devices and the classes in which they belong. Human Interface Device Class or HID is probably the most common. Devices in the class enable the user to manually transmit data along the USB.

The Mass Storage Device Class or MSD includes devices that store large amounts of data such as thumb drives or even an MP3 player.

The Communication Device Class or CDC enables communication via such things as a modem or convert from older protocols such as an RS-232 to USB converter.

In the projects found at the end of the Low-Pin Count USB Development Kit User's Guide, you will implement two HID (mouse, keyboard) and one CDC (RS-232 to USB converter) applications.

One other class that should be mentioned is the Vendor Specific Class. In this case, the developer is required to write a driver for a device that is not defined in any other class. This could be a arduous task to say the least. Suffice to say, developing a device that fits into one of the available classes is the easier way to go.



Slide 19



In order for the Host to assign the appropriate driver to the device it requires information on such things as:

The class that this device conforms to

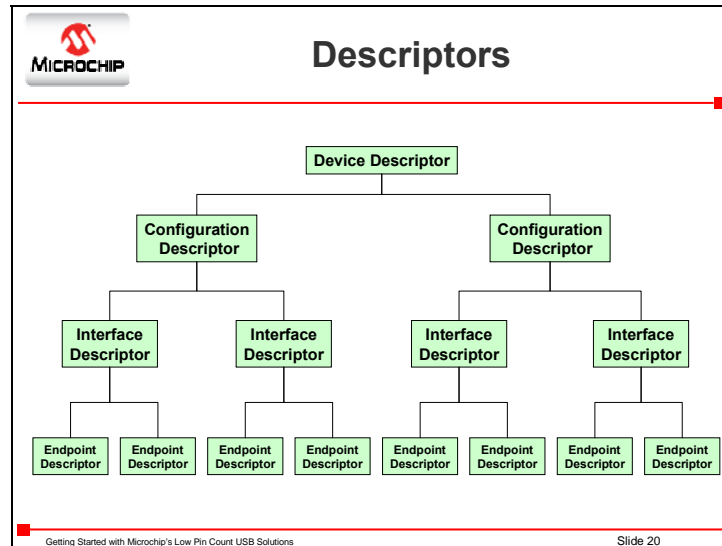
How many possible configurations does this device have

How many interfaces does each configuration have

What is the size and number of endpoints that are available on this device

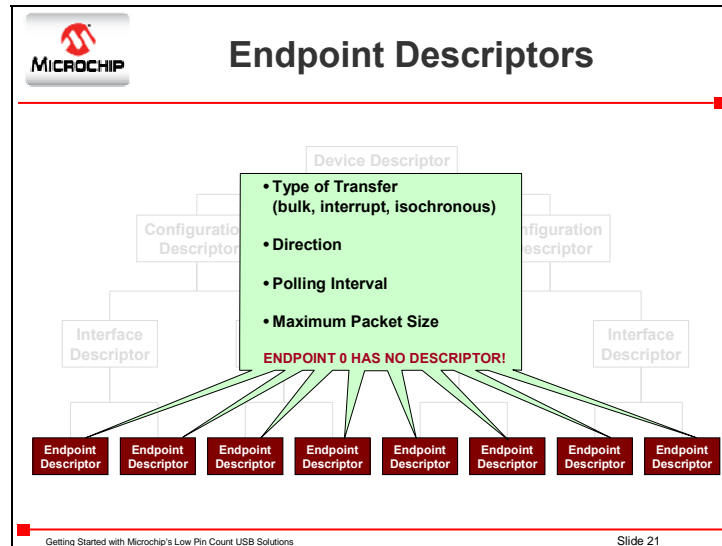
The device passes this information to the Host at connection in the form of descriptors. As we will see later in the course, Microchip's Full Speed USB Firmware Framework provides the means of easily defining these descriptors.

Slide 20



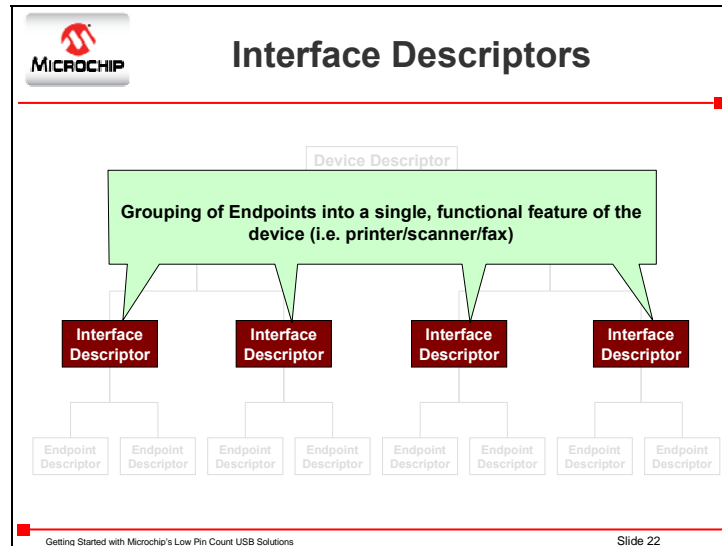
This slides shows five of the most important types of standard descriptors that must be defined within the software on the device.

Slide 21



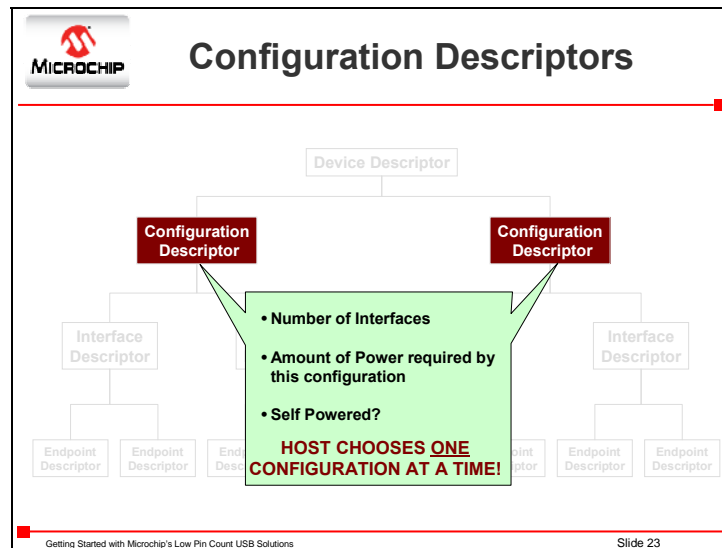
The endpoint descriptors identify the type of endpoints, the direction (IN or OUT) of each endpoint, and other endpoint specific information.

Slide 22



Interface descriptors detail the number of endpoints that are used in specific interface as well as what class that interface will support.

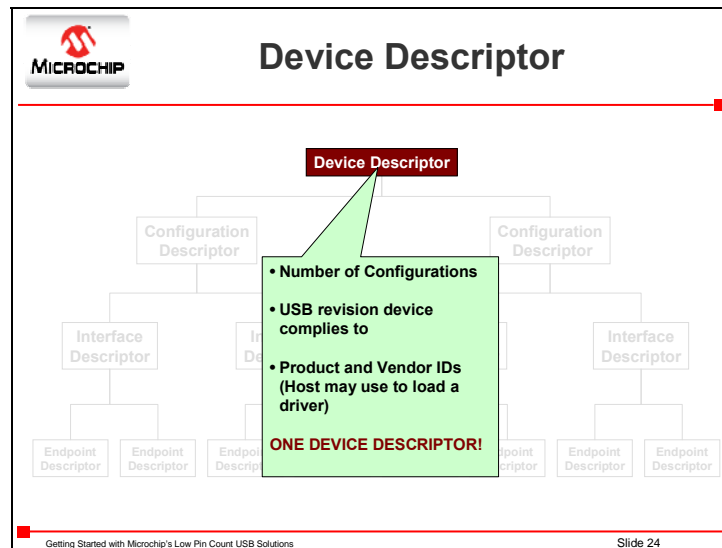
Slide 23



Configuration descriptors provide information on such things as device power requirements and how many different interfaces are supported when a particular configuration is in use by the Host.

Note that only one configuration is selected by the Host at any given time.

## Slide 24



The device descriptor provides general information such as:

Manufacturer (identified by the Vendor ID)

Product Number (Product ID)

Serial number

The class of the device

Number of configurations

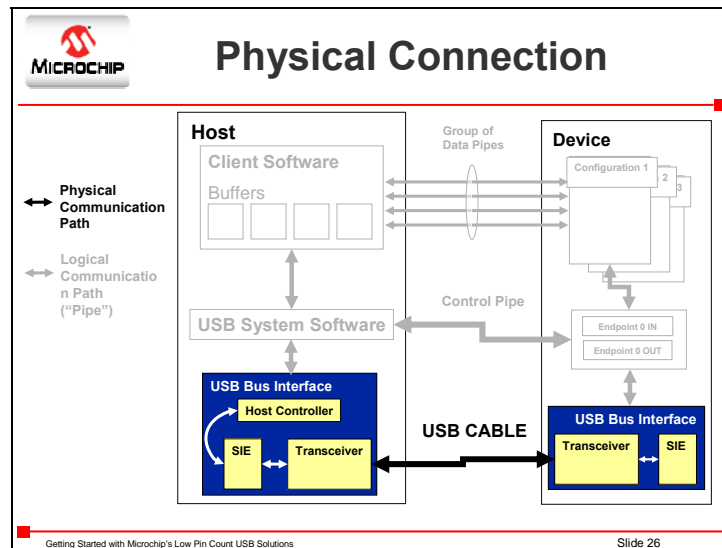
There is only one device descriptor per device.

Slide 25



We have just looked at the Logical or Software topology of the USB. Let's now take a closer look at the actual physical connection.

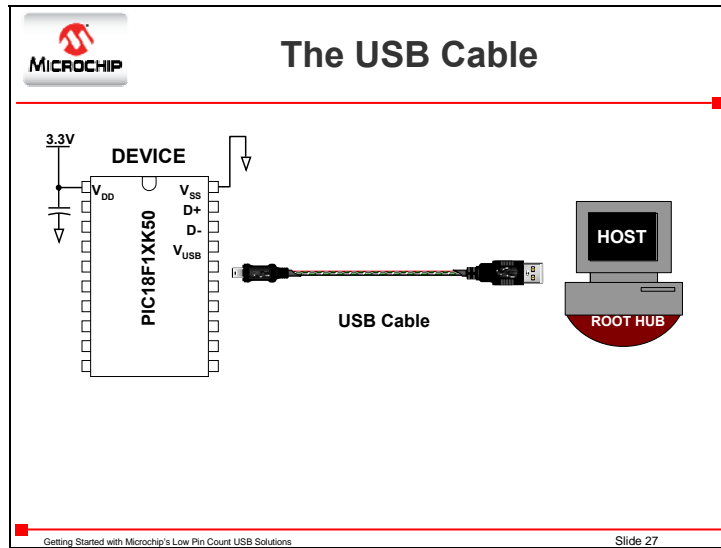
Slide 26



Referring once again to representation of the USB system, the physical connection has a number of components that makes communication possible.

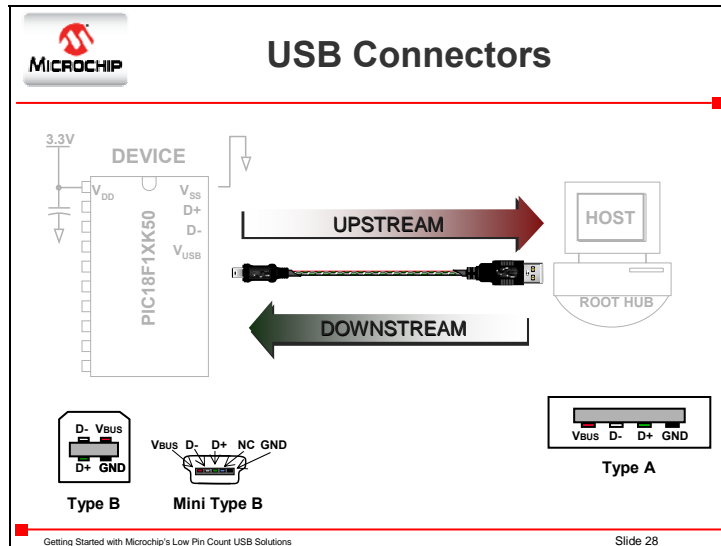


Slide 27



First, let's take a look at the USB cable itself.

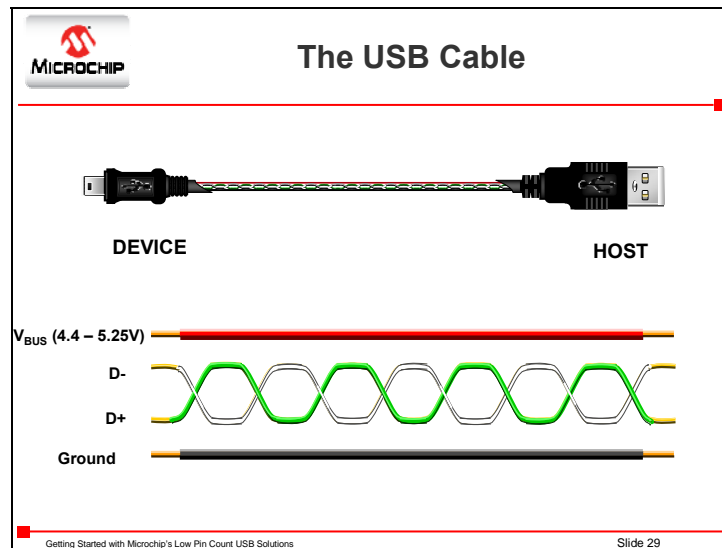
Slide 28



The USB cable implements a Keyed Connector which minimizes the chance of an improper connection. The Host side utilizes the A type connector/plug while the Device side utilizes the Type B or Mini Type B connectors/plugs.

Note the terminology in the slide used to denote data direction. Towards the Host is identified as Upstream, while data flow towards the Device is identified as Downstream.

Slide 29




The cable itself is constructed of four conductors:

VBUS and Ground for power conduction

Two signaling lines (D+ and D-) in a twisted pair configuration excellent for common mode noise rejection.

Data signaling will be discussed further later in this presentation.

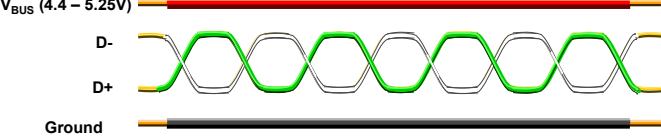
Slide 30



### Bus Power

- **Bus Power to each device:**
  - 4.40 - 5.25 V
  - Guaranteed 100 mA
  - 500 mA maximum through negotiation

Any more than this and device needs to be self-powered



Getting Started with Microchip's Low Pin Count USB Solutions

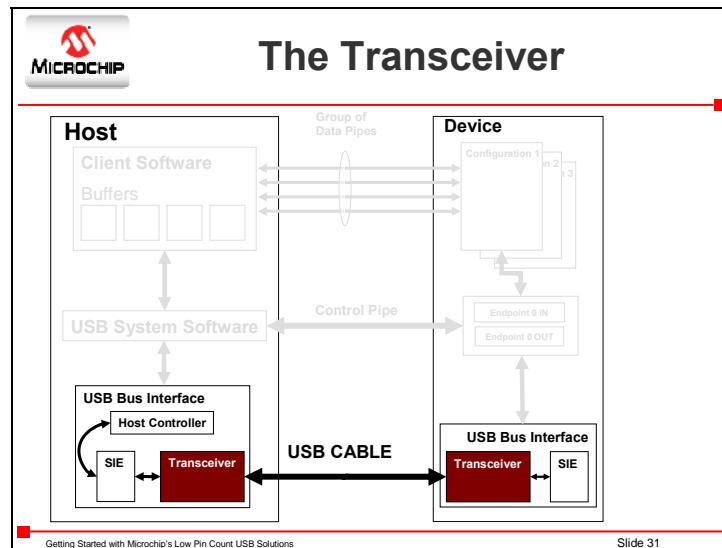
Slide 30

The VBUS line can be used to supply power from the Host to the Device. Typically 4.4V to 5.25V maximum with a guaranteed 100mA current source. Any power requirements above 500mA will necessitate the Device be supplied by an alternate power source.

Note:

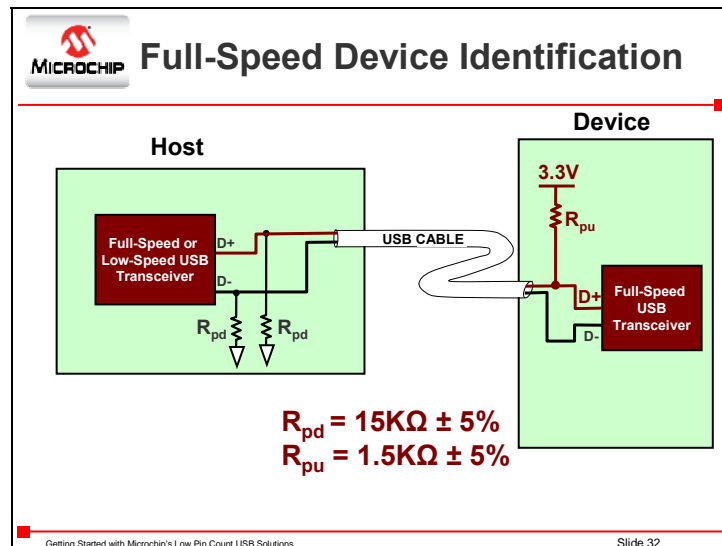
Each USB port on a typical computer can provide 100-500mA. 500mA to a Bus-Powered Hub can provide each end devices with 100mA. Most Self-Powered hubs have 4-ports using 400mA. Therefore there is 100mA left for the hub itself.

Slide 31



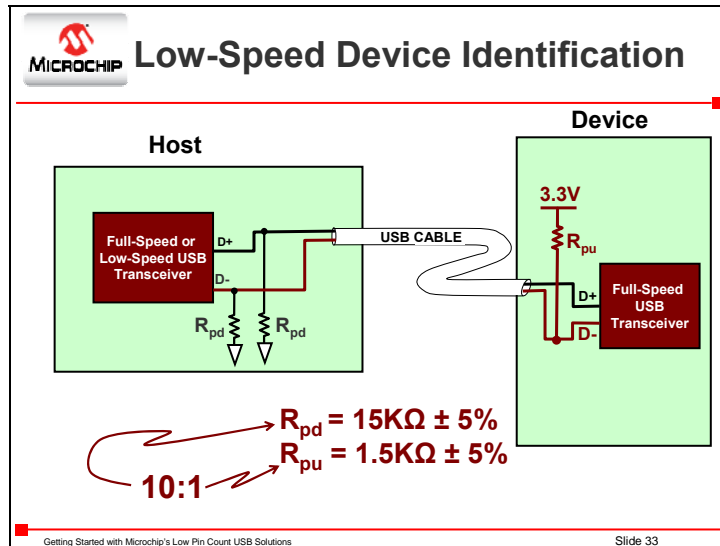
The USB cable connects to the Host and Device ports which in turn connect to a data link on each end called the transceiver. The transceiver recognizes data signaling which is passed on to the rest of the Host/Device system.

Slide 32



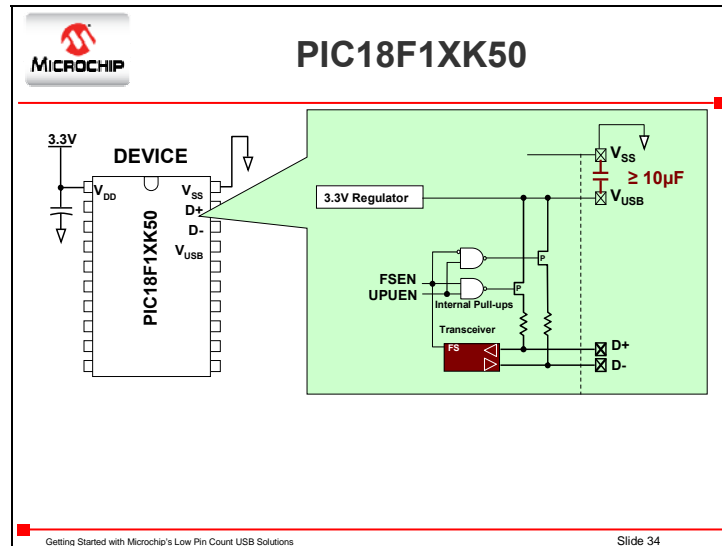
When a Device is initially connected, the Host recognizes immediately if the Device is a Full or Low speed device. This is accomplished by pulling either the D+ line HIGH (3.3V derived from VBUS) for Full Speed devices...

Slide 33



And by pulling the D- line High for low speed devices. On the Host side, pull-down resistors decrease the chance of noise on the data line and to identify to the Host the absence of a Device connected to that port. Note that pull-up resistor values are specified at 1.5kohm while pull-down at 15kohms( 10:1 ratio.).

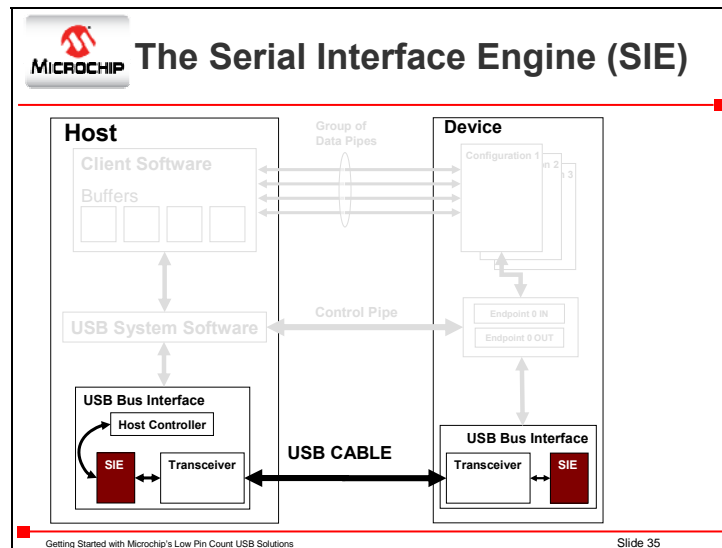
Slide 34



The USB peripheral on the PIC18F1XK50 has a built-in, USB 2.0, full-speed and low-speed capable transceiver. A 3.3V regulator provides power to the transceiver and on-chip pull-up resistors for low-cost, single chip applications. In order to meet USB signaling level specifications, the V<sub>USB</sub> line output of the regulator must be supplied with a voltage source between 3 and 3.6V. The best electrical signal quality is obtained when a 3.3V supply is used and locally bypassed with a high quality ceramic capacitor ( $\geq 10\mu\text{F}$ ) placed as close to the V<sub>USB</sub> and V<sub>SS</sub> pins found on the same edge of the package.

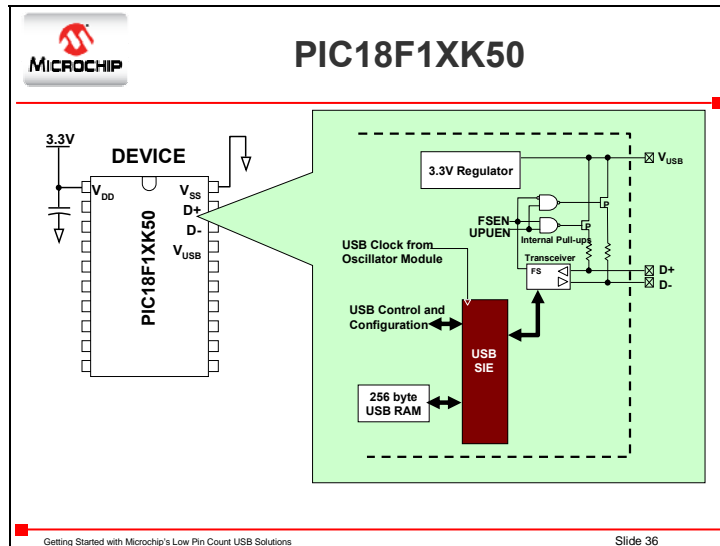


Slide 35



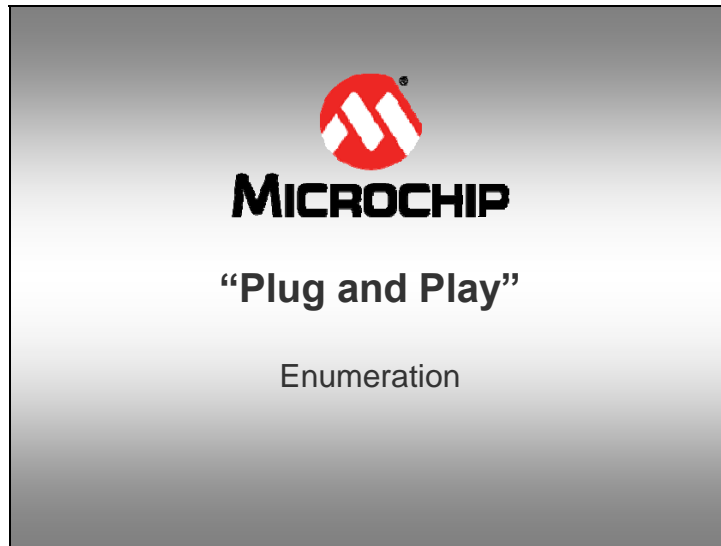
The next component in our physical connection is the Serial Interface Engine (SIE). This dedicated hardware component carries out low-level functions such as time critical communication portions of the USB protocol. These will be briefly discussed in the communication section of the presentation.

Slide 36




The PIC18F1XK50 contains a full-speed and low-speed compatible USB SIE enabling fast communication between any USB Host and the PIC microcontroller. The SIE interfaces directly with USB through the internal transceiver. As will be demonstrated in the Project Labs contained in the Low Pin-Count USB Development Kit User's Guide, the SIE will be checked before any data is transmitted or received. Since the SIE handles both transmission and reception of data, there will be times when the SIE is busy. Luckily, this is all taken care of in the Microchip Full-Speed USB Firmware Framework files.

Slide 37

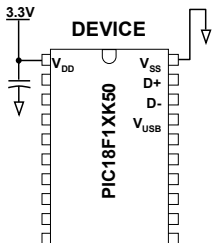


## Slide 38

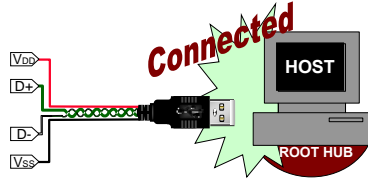


### Enumeration

- What happens when a Device is connected to the HOST:



**DEVICE**  
PIC18F1XK50



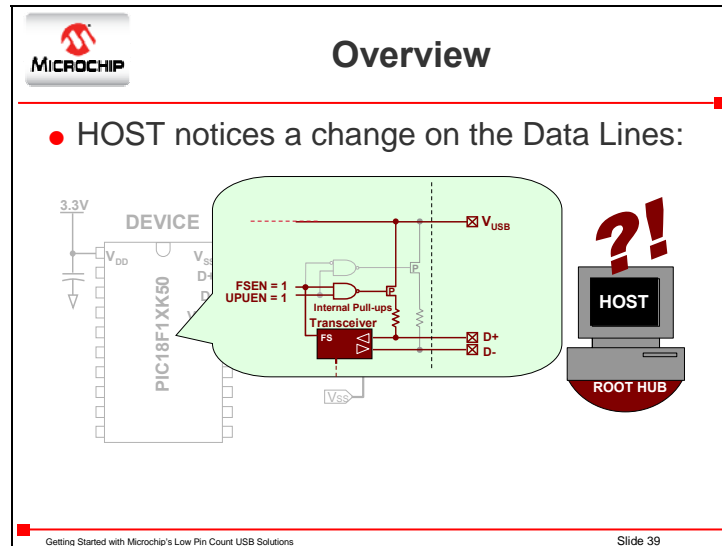
**Connected**

**HOST**  
ROOT HUB

Getting Started with Microchip's Low Pin Count USB SolutionsSlide 38

When a Device is plugged into a Host for the very first time, a process called Enumeration occurs. The Host interrogates a Device's descriptors so that an appropriate driver can be assigned and the Device configured accordingly. Let's step through this process.

Slide 39



The Host becomes aware of the Device due to pull-up resistors on one of its data lines. In this case, the PIC18F1XK50 is configured as full-speed Device with D+ pulled HIGH.

Slide 40

**MICROCHIP**

## Overview

- Host waits 100 mS to allow power on device to stabilize :


The diagram illustrates the power-up sequence of a PIC microcontroller connected to a host. On the left, a PIC microcontroller is shown with a 3.3V power supply connected to its VDD pin. A callout box highlights the VDD pin's behavior over time, showing a curve that rises from 0V and levels off at 3.3V. The host, labeled 'HOST', is connected to the PIC via a USB root hub, which is labeled 'ROOT HUB'. The host's power supply is labeled 'VSS'.

Getting Started with Microchip's Low Pin Count USB Solutions

Slide 40

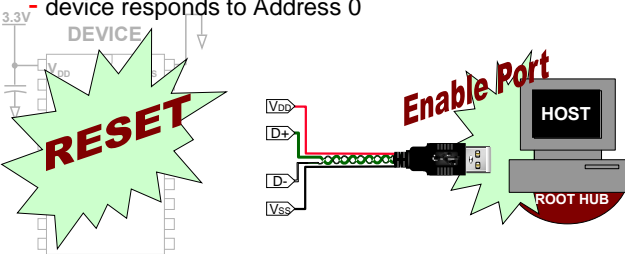
To wait for the power on the Device to stabilize to a functional level, the Host waits for 100mS before moving to the next step of the enumeration process.

## Slide 41



### Overview

- Host enables port and resets the device
  - ensures device is in a "known" state
  - device responds to Address 0



3.3V  
V<sub>DD</sub>  
D+  
D-  
V<sub>SS</sub>

Enable Port

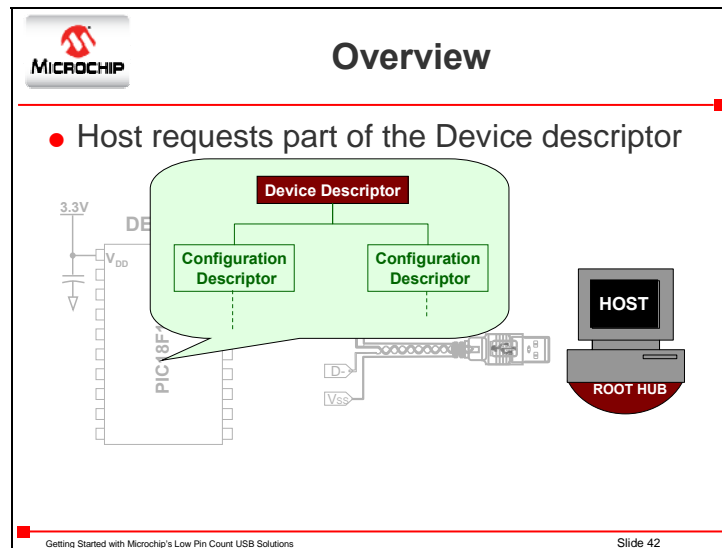
HOST  
ROOT HUB

Getting Started with Microchip's Low Pin Count USB Solutions Slide 41

The Host then signals a USB Reset so that the Device is in a known, non-configured state. At this point, the Device responds to Address 0 (reserved address for enumeration purposes).

Note: A USB Reset is achieved when the Host pulls both data signaling lines LOW for 10mS or more.


Slide 42



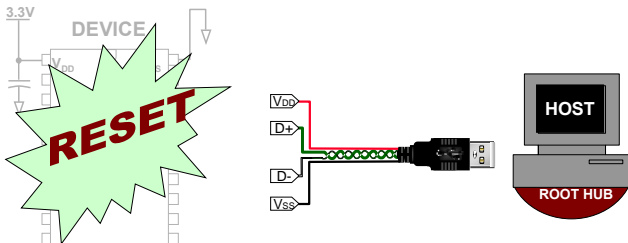
The Host then request a portion of the Device Descriptor. Specifically, what is the size of the endpoint 0 (Remember, this is the control endpoint) so that the Host knows the maximum size of the subsequent data packets it can send.



Slide 43

 **Overview**

- Typically, HOST will again RESET the device




The diagram illustrates a USB connection between a device and a host. On the left, a schematic of a device shows a 3.3V supply connected to a VDD pin. A green starburst labeled 'RESET' is superimposed over the device. On the right, a host is shown with a 'ROOT HUB' and a 'HOST' label. A USB cable connects the two, with pins labeled VDD, D+, D-, and VSS. The VDD pin is connected to the host's VDD pin, and the D+ and D- pins are connected to the host's D+ and D- pins respectively. The VSS pin is connected to the host's VSS pin.

Getting Started with Microchip's Low Pin Count USB Solutions Slide 43

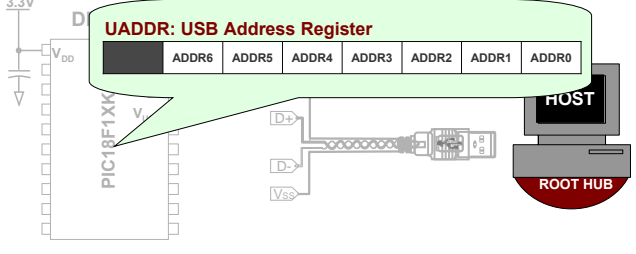
Typically, another USB Reset is issued...

## Slide 44



### Overview

- HOST assigns the device a unique Address
- HOST free to RESET other devices



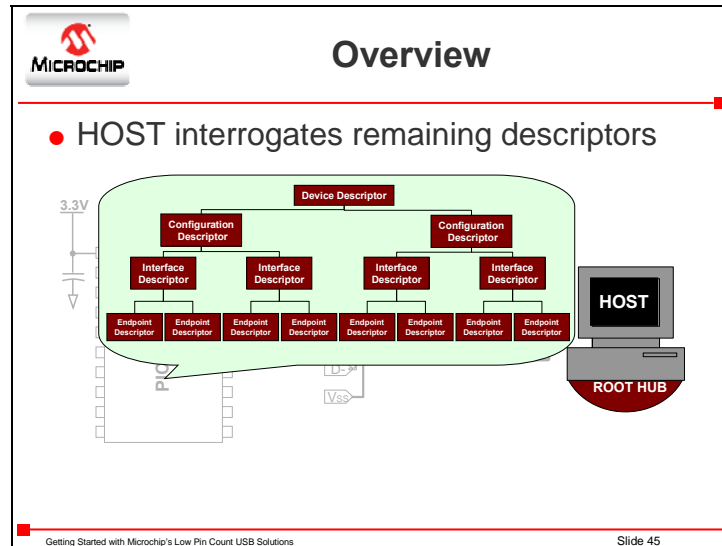
**UADDR: USB Address Register**

	ADDR6	ADDR5	ADDR4	ADDR3	ADDR2	ADDR1	ADDR0
--	-------	-------	-------	-------	-------	-------	-------

Getting Started with Microchip's Low Pin Count USB SolutionsSlide 44

The Host then assigns a unique address to the device other than zero. This enables the Host to now Reset other devices. Note that on the PIC18F1XK50 this unique address is stored in the USB Address Register UADDR. This is a 7-bit register since only 126 devices can be connected to a single Host at any given time.

Slide 45



The Host now interrogates the Device further requesting all other descriptors.

Slide 46

**MICROCHIP**

## Overview

- HOST assigns a suitable driver
- Driver selects a configuration for the device

The diagram illustrates the USB configuration process. On the left, a 'DEVICE' is shown with a 3.3V supply and a green starburst labeled 'Configured'. On the right, a 'HOST' is shown with a 'ROOT HUB' and a USB cable. The cable is labeled 'Assign Driver' and shows connections for VDD, D+, D-, and VSS.

Getting Started with Microchip's Low Pin Count USB Solutions Slide 46

Now that the Host fully understands the intended purpose of the Device, it assigns a Driver (or requests that one be loaded for a Vendor Specific Class Device) and the Device is now configured accordingly.

Slide 47




So now that the Device is connected and configured with the appropriate driver on the Host, let's take a look at how data is actually passed over the USB.

Note:

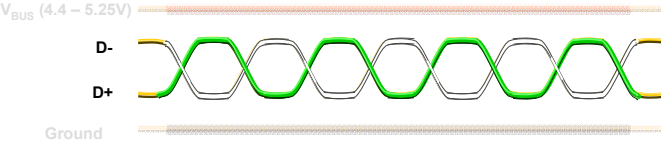
Since the Low-Level functions of the USB communication are taken care of when using Microchip's Full-Speed USB Firmware Framework, a very high level overview is discussed here. To learn more about USB communication, packets and transactions refer to the resources listed at the end of this class.

## Slide 48



### Data Communication

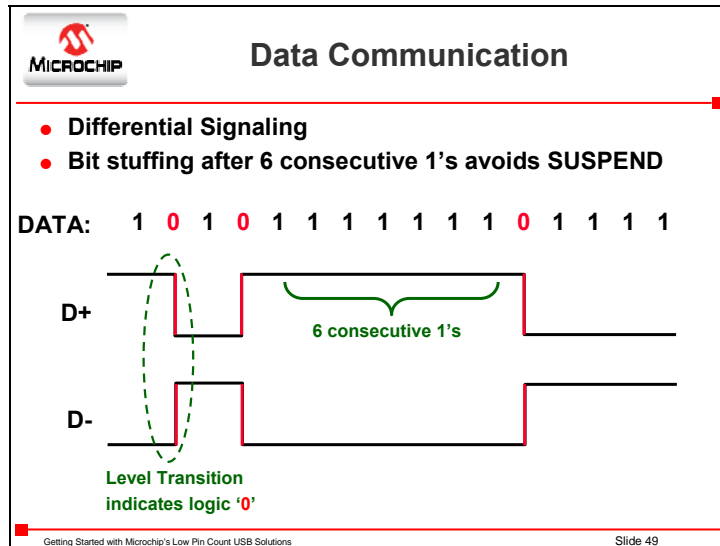
- **Half Duplex with Non Return to Zero Invert (NRZI) Data Encoding**
- **Data lines implemented as a twisted pair**
  - minimizes crosstalk
  - minimizes EMI



Getting Started with Microchip's Low Pin Count USB Solutions Slide 48


The USB implements a Non Return Zero, Half Duplex data encoding scheme- data transmission can go in only one direction at a time. As mentioned previously the data signaling lines are in a twisted pair configuration in order to minimize any crosstalk and interference from external devices that may impede data transfer.

Slide 49



Implementing differential signaling reduces any interference from common mode noise by effectively cancelling it out. The D- line is a mirror image of the D+ line. Logic '1' is recognized by a constant signal level, either HIGH or LOW, for a given period of time. Logic '0' is recognized by a HIGH-to-LOW or LOW-to-HIGH transition. If a Device doesn't receive any data (no transitions noted on the differential pair) for 3mS or more, the Device enters a Low-Powered SUSPEND mode (Hint: think about SLEEP on the PIC MCU) to conserve power on the Bus. In order to avoid this condition, a periodic level transition is required. Therefore, for long periods of constant signal level (i.e. 6 consecutive '1's) a level transition is issued. This is called bit stuffing.

## Slide 50



### Packets

- Every transfer of data consists of packets
  - **Token Packet**
    - IN: Host wants information from Device
    - OUT: Host has information for Device
    - SETUP: Starts Control transfers
  - **Data Packet**
    - Contains the data
  - **Status or Handshake Packet**
    - **ACK**nowledges data transaction
    - Used in error checking

Getting Started with Microchip's Low Pin Count USB Solutions Slide 50

Every transfer of data is accomplished using packets. There are three common types of packets:

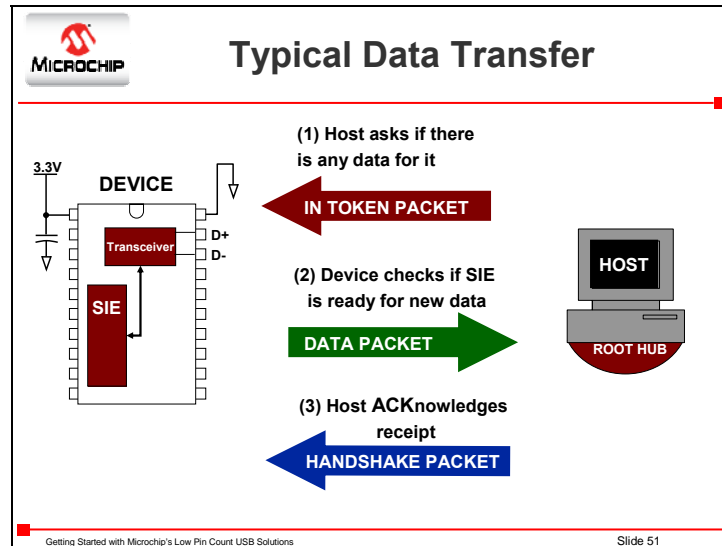
**Token Packet:** This type of packet is used to inform the Device that the Host wishes to send information to the Device, receive information from the Device, or is sending a control transfer that will change how the Device behaves.

**Data Packet:** This packet contains the data that the Host wishes to send/receive.

**Status or Handshake Packet:** As with most communication protocols, a mechanism is needed to inform either the Device or Host that data was received accurately. The USB implements Cyclic Redundancy Checking to ensure accurate transmission.



Slide 51



This slide demonstrates a typical transfer of data from the Device to the Host.


The Host begins by sending an IN Token packet informing the Device that data is being requested from it.

The Device responds by transmitting the requested data to the Host

Finally, the Host transmits an ACKnowledge Handshake Packet confirming data reception.



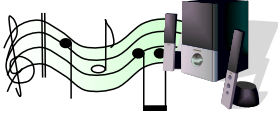
The reader is encouraged to read more about the various types of data transfer found in the resources referenced at the end of this class.

Slide 52



### Four Basic Data Flow Types

- Control
- Isochronous
- Bulk
- Interrupt



Getting Started with Microchip's Low Pin Count USB Solutions

Slide 52

This sequence of three packets (Token, Data, Handshake) comprises a transfer of data that can be categorized into one of four common data flow types.


Control data flow types are used by the Host to interface with endpoint 0 on a device to change a Device's operational characteristics.

Isochronous data flow types are used when a stream of data is needed to be transmitted for such applications as video or audio. There is no error checking in this type data flow as the occasional flicker of a screen or pop in the audio is of no consequence. The idea here is to guarantee bandwidth.

Bulk data flow types transfer large amounts of data with error free delivery. This type of data transfer does not guarantee bandwidth. In fact, this type of data flow receives the lowest priority on the USB. The Host schedules this type of data flow transfers only once all other transfer types have been serviced. Applications for this type include printers and Mass Storage Devices.

Interrupt data flow types are used when small amounts of data are used to keep the Host updated on the status of a Device. Applications such as Keyboards or a Mouse use this type of data flow type. These are not to be confused with interrupts. This data flow type does not interrupt the USB. The name was given as this type is used with applications that would have traditionally used interrupts on older communication protocols.

Slide 53



### Data Transaction Types

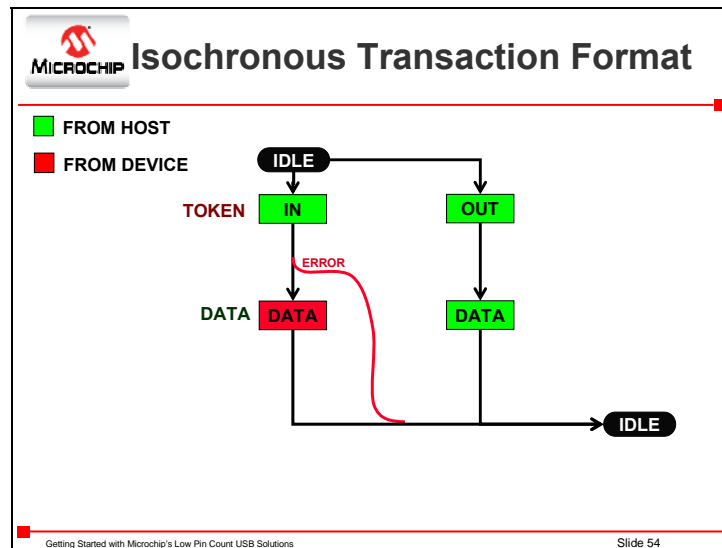
Data Flow / Endpoint Type	Polling Interval	Priority	Guarantees
Interrupt	Fixed, Periodic	High	64 Bytes/Period, Data Integrity
Isochronous	Fixed, Periodic	High	1023 Bytes/Period
Bulk	Variable, Uses Free Bandwidth	Low	Data Integrity
Control	Variable	Medium	Some Bandwidth, Data Integrity

Getting Started with Microchip's Low Pin Count USB Solutions

Slide 53

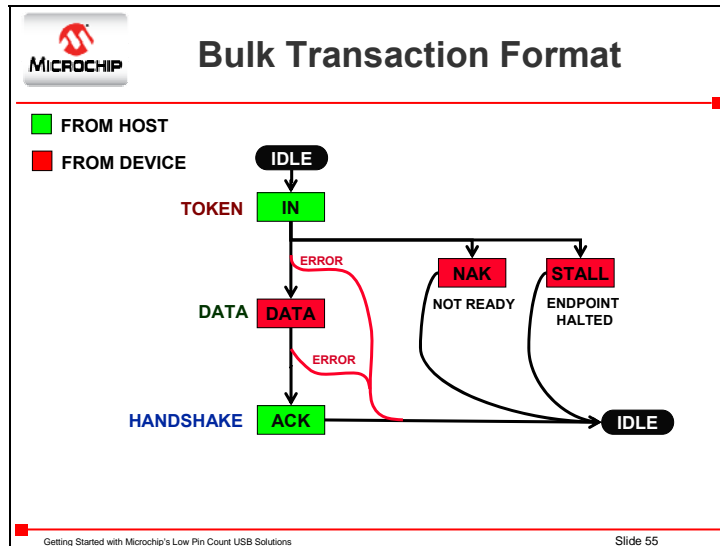
The table in this slides highlights the main characteristics of each type.

Slide 54

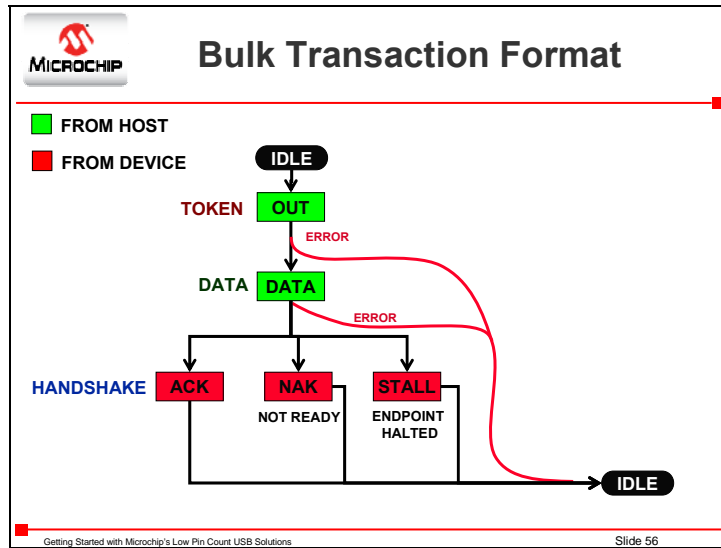


The next few slides demonstrate the format for each data flow type. The good news here is that you really don't need to bother learning these. Using Microchip's Full-Speed Firmware Framework takes care of all this in the background leaving you free to work on the application.

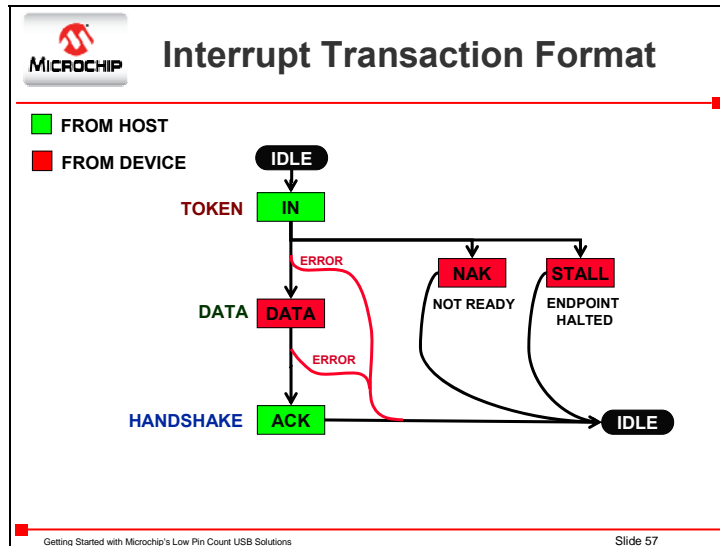
Slide 55



Slide 56

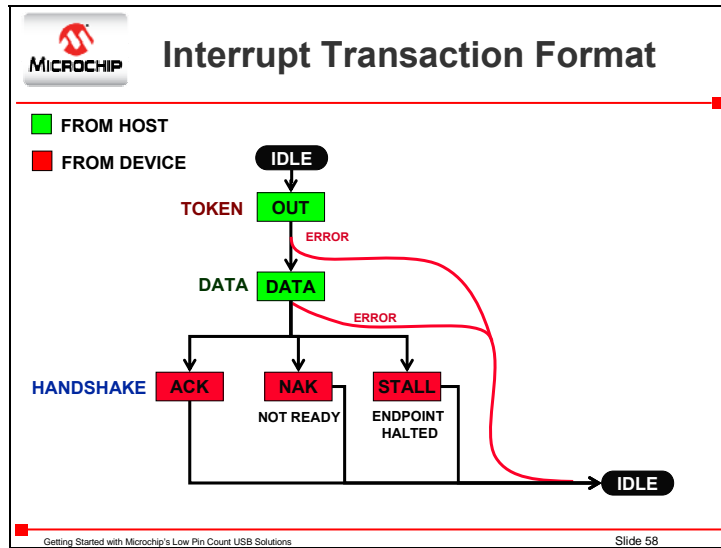


Slide 57





Slide 58




Slide 59





The good news here is that you really don't need to bother learning these various data flow formats. Using Microchip's Low-Pin Count USB Solutions take care of these low-level USB functions letting you concentrate on the application and not the protocol.

Slide 60



## Microchip's Low Pin Count USB solutions

- Simplifies the development
  - Hardware Development using the Low Pin-Count USB Development Kit
  - Software Development using Microchip's Full-Speed USB Framework



Getting Started with Microchip's Low Pin Count USB Solutions Slide 60

Microchip's Low Pin-Count USB solution includes the following:

The Low Pin-Count USB Development Kit: Provides a platform from which to easily test and develop hardware for your USB application.

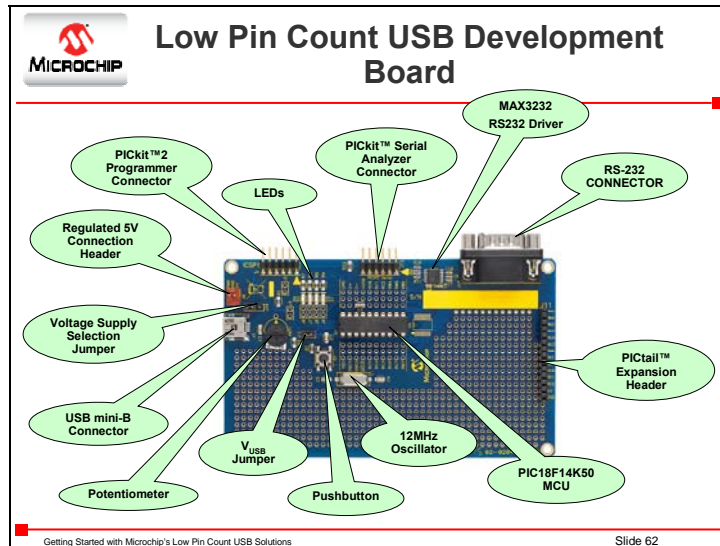
Microchip's Full-Speed USB Framework provides a library of firmware resources that take care of low-level USB functions allowing you to concentrate on the application and not the protocol

Slide 61




Let's take a closer look at the Low-Pin Count USB Development Board.

Slide 62



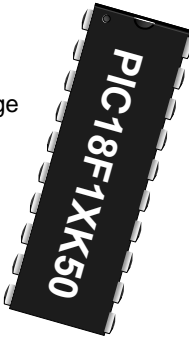
This slide highlights the main components of the development board.

## Slide 63



### PIC18F1XK50 USB Peripheral

- Power Managed Modes
- Up to 48MHz Operation
- 1.8V to 5.5V operating voltage range
- CCP/ECCP
- I<sup>2</sup>C/SPI
- EUSART
- 10-bit ADC
- Dual Comparators



Getting Started with Microchip's Low Pin Count USB Solutions Slide 63

The Low Pin-Count USB Development comes populated with a PIC18F14K50 MCU loaded with the peripherals listed.

Slide 64

**MICROCHIP** **PIC18F1XK50 USB Peripheral**

- USB V2.0 Compliant SIE
- Low Speed (1.5 Mb/s) and Full Speed (12 Mb/s)
- Supports Control, Interrupt, Isochronous and Bulk Transfers
- 16 Endpoints (8 bidirectional)
- Input-change interrupt on D+/D- for detecting connection to HOST

**Simplified USB Peripheral Block Diagram**

Getting Started with Microchip's Low Pin Count USB Solutions Slide 64

The USB peripheral is loaded with features that will ease the development of your USB application. Many of these features have already been discussed. As always, refer to the device datasheet for a more in-depth discussion this or any other peripheral's features and electrical characteristics.


Slide 65



MCHPFSUSB is a distribution package containing a variety of USB related PIC18 and PIC24F firmware projects, along with other USB related drivers and resources intended for use on the PC. The USB embedded host stack is API compatible with the USB Device and Embedded Host Stack for PIC32. All release notes are included in the .zip file bundle.



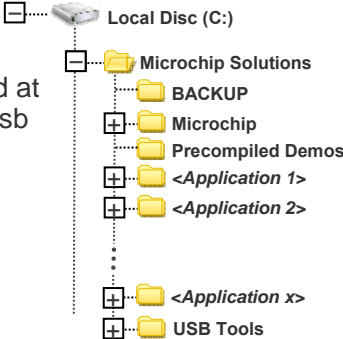
Slide 66

 **Overview of Framework**

---

- A library of resources to create USB applications:

Available for download at [www.microchip.com/usb](http://www.microchip.com/usb) and follow the links




```
graph TD; C[Local Disc (C:)] --> MS[Microchip Solutions]; MS --> B[BACKUP]; MS --> M[Microchip]; MS --> PD[Precompiled Demos]; MS --> A1[<Application 1>]; MS --> A2[<Application 2>]; MS --> A3[...]; MS --> AX[<Application x>]; MS --> UT[USB Tools];
```

Getting Started with Microchip's Low Pin Count USB Solutions Slide 66

The MCHPFSUSB is available for download from [www.microchip.com/usb](http://www.microchip.com/usb). Once downloaded, the installer will store the framework resource files in the C directory folder “Microchip Solutions”.

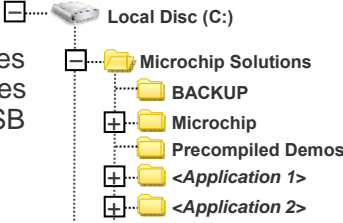
Slide 67



## Overview of Framework

---

- A library of resources to create USB applications:
  - Organized directories of firmware that takes care of low level USB operation




```
Local Disc (C:)
├── Microchip Solutions
│   ├── BACKUP
│   ├── Microchip
│   ├── Precompiled Demos
│   ├── <Application 1>
│   └── <Application 2>
```

**Cooperative Multitasking**  
**No Blocking Code!**  
**i.e.**  
`while((HIDTxHandleBusy(lastTransmission) != 0))`

Getting Started with Microchip's Low Pin Count USB SolutionsSlide 67

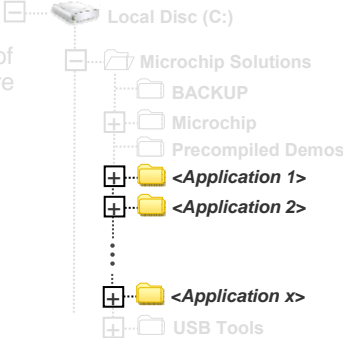
In this folder are a variety of firmware source and header files that will take care of much of the low-level functions of the USB protocol. The MCHPFSUSB provides a set of modular firmware interfaces that handle the work for implementing USB communications. This is a cooperative multitasking environment. Therefore no blocking functions should be implemented.

Slide 68

 **Application Examples**

---

- A library of resources to create USB applications:
  - Organized directories of firmware that takes care of low level USB operation
  - Application examples organized by class




Local Disc (C:)

- Microchip Solutions
  - BACKUP
  - Microchip
  - Precompiled Demos
  - <Application 1>
  - <Application 2>
  - ...
  - <Application x>
  - USB Tools

Getting Started with Microchip's Low Pin Count USB Solutions Slide 68

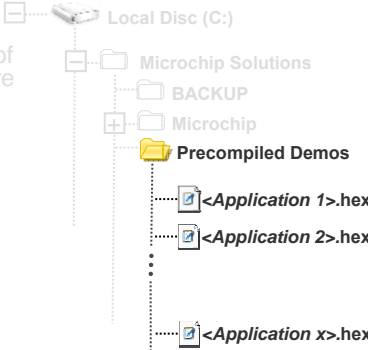
A number of application examples organized by class are provided as a reference to kick-start the learning process. All firmware is completely accessible should the developer wish to investigate how various aspects of the USB protocol are accomplished.

## Slide 69



### Precompiled Demos

- A library of resources to create USB applications:
  - Organized directories of firmware that takes care of low level USB operation
  - Application examples organized by class
  - Precompiled demos of application examples



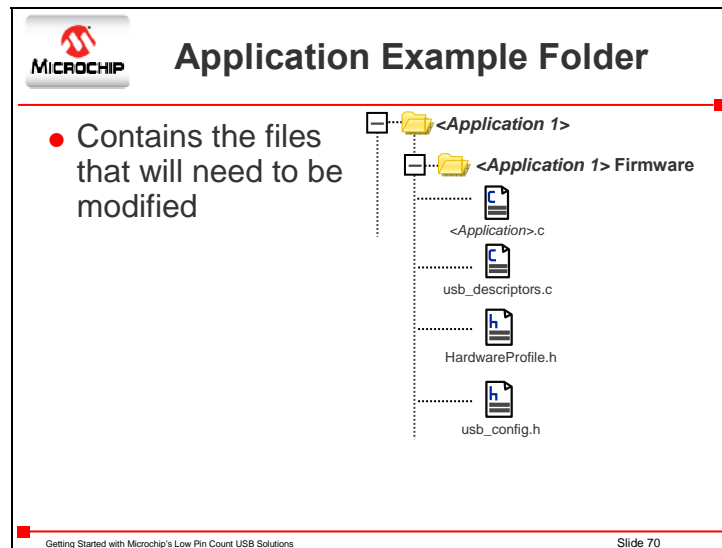
Local Disc (C:)

- Microchip Solutions
- BACKUP
- Microchip
  - Precompiled Demos
    - <Application 1>.hex
    - <Application 2>.hex
    - ...
    - <Application x>.hex

Getting Started with Microchip's Low Pin Count USB Solutions Slide 69

A number of precompiled demos are provided. Simply download the .hex file onto the PIC18F1XK50 for a quick demonstration of how the application firmware works.

## Slide 70



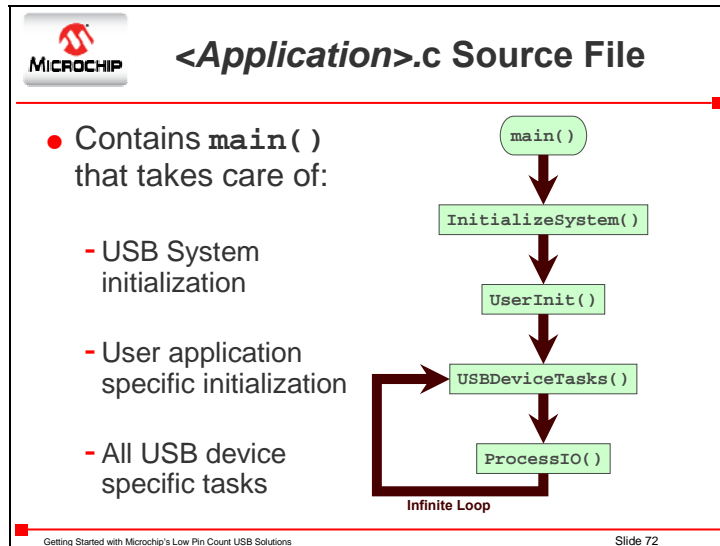
The user will notice that there are four main firmware source and header files common to all application folders. Only these files need to be altered to accommodate an application that related to class the example conforms to. Let's take a closer look.

Slide 71



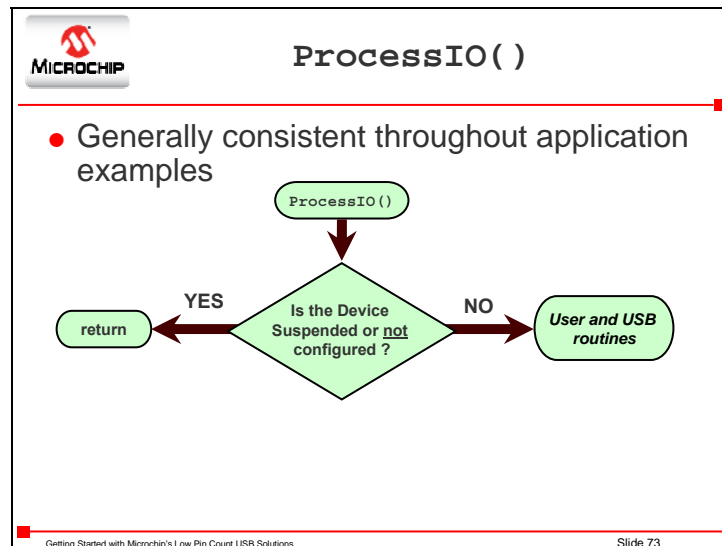
The application.c file contains the main code that relates directly to the application.

Slide 72



A software control loop is implemented within the main function that services different tasks. These tasks can be thought of as either USB tasks or user tasks. Function calls initialize the USB system, initialize peripherals and variables specific to your application, and calls the `USBTasks()` which handles all USB tasks.

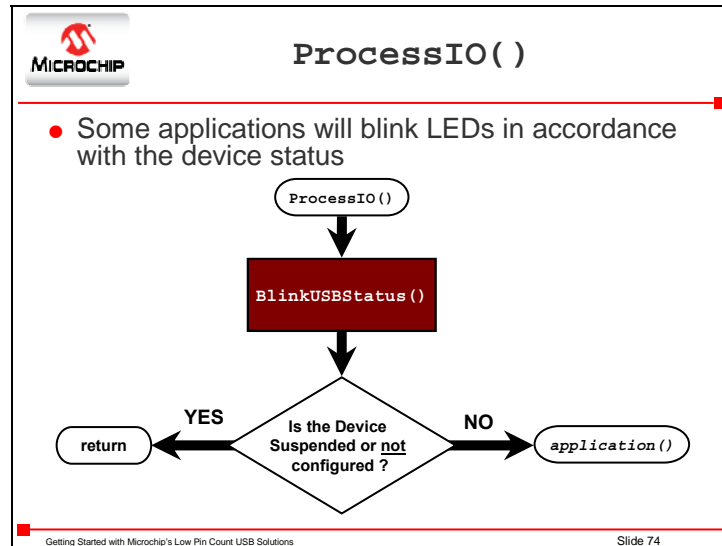
Slide 73



The ProcessIO() is the function that will contain the code specific to your application. The function begins by checking to see if the Device is configured and that it is not in the SUSPEND mode. Function calls specific to your application should be placed immediately following this Device status check.

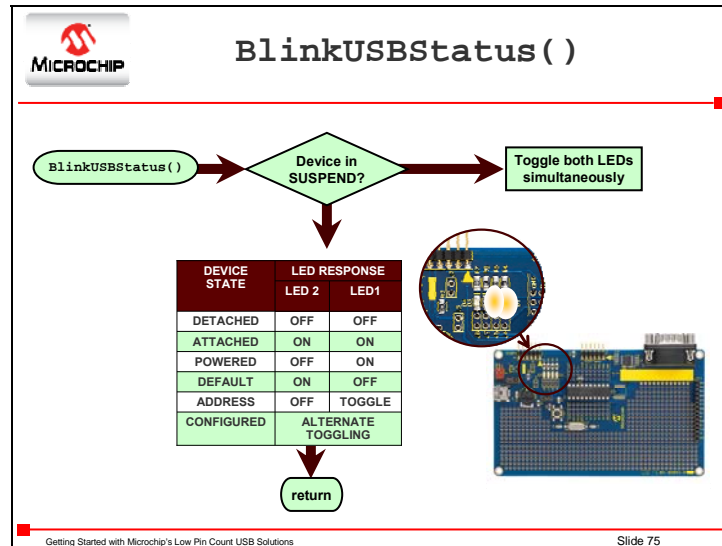


Slide 74



Some applications will also call the BlinkUSBStatus() before checking Device Status.

Slide 75




This function will flash the LEDs which populate the Low-Pin Count USB Development board according to the status of the Device. Again, all the background code is already written for you.

Slide 76



The next source file important that will need to be altered is the `usb_descriptors.c` file.

## Slide 77



# usb\_descriptors.c Overview

---

- Defines descriptors that will be used during enumeration
  - as per Chapter 9 of the USB 2.0 Specification

**Device Descriptor Example**

```
ROM USB_DEVICE_DESCRIPTOR device_dsc=
{
  0x12, // Size of this descriptor in bytes
  USB_DESCRIPTOR_DEVICE, // DEVICE descriptor type
  0x0200, // USB Spec. Release Number in BCD format
  0x00, // Class Code
  0x00, // Subclass code
  0x00, // Protocol code
  EP0_BUFF_SIZE, // Max. packet size for EP0, see usbcfg.h
  0x04D8, // Vendor ID
  0x0015, // Product ID: Business card app
  0x0001, // Device release number in BCD format
  0x01, // Manufacturer string index
  0x02, // Product string index
  0x00, // Device serial number string index
  0x01 // Number of possible configurations
};
```

Getting Started with Microchip's Low Pin Count USB SolutionsSlide 77


It is within this file that all descriptors are defined. Each line of code is commented to allow easy reference to the USB 2.0 specification Chapter 9 descriptor definitions. The first Project Lab in the Low Pin-Count USB Development Kit User's Guide will walk you through setting up the descriptors so that the Device is enumerated successfully.

Slide 78



Finally, the Hardware\_Profile.h file will require alteration when you are ready to move your application from the Low-Pin Count USB Development Board to your own design.

## Slide 79



### HardwareProfile.h

---

- Maps various demo board components
  - Which PORT pins LEDs connect to
  - Development Board clock speed
  
- **DEMO\_BOARD** definition
  - Must be altered when using a custom board

---

Getting Started with Microchip's Low Pin Count USB Solutions Slide 79


This file maps the various development board hardware setups to a common definition for use in firmware. Such things as which port pins are connected to LEDs, pushbuttons, potentiometers etc... are described here. This file has been setup to allow the firmware to recognize which of Microchip's Development Board's are being used based off of information provided by the compiler.

Modification to an existing board or the use of a custom board requires that the **DEMO\_BOARD** definition at the top of HardwareProfile.h be altered accordingly. This will indicate to the compiler that a custom board is being used.

Slide 80



## Slide 81



### Microchip's VID/PID Sublicense Program

---

- Why?
  - Enables fast development of original applications without using the USB Developers Forum
  
- How?
  - Fill out the application form from [www.microchip.com/usb](http://www.microchip.com/usb)
  
- Provides VID and PID for small volume applications (10,000 units or less)

---

Getting Started with Microchip's Low Pin Count USB Solutions Slide 81

Microchip will sublicense its Vendor ID (VID) number and assign a Product ID (PID) number to assist customers with internal development and small volume (10000 units or less) applications. The application for sublicense is available from [www.microchip.com/usb](http://www.microchip.com/usb). Once completed, the application is returned to the address listed on the application form. This program is designed to be an interim solution and it is assumed that the user will eventually become a member of the USB implementers' forum and apply for their own Vendor ID.

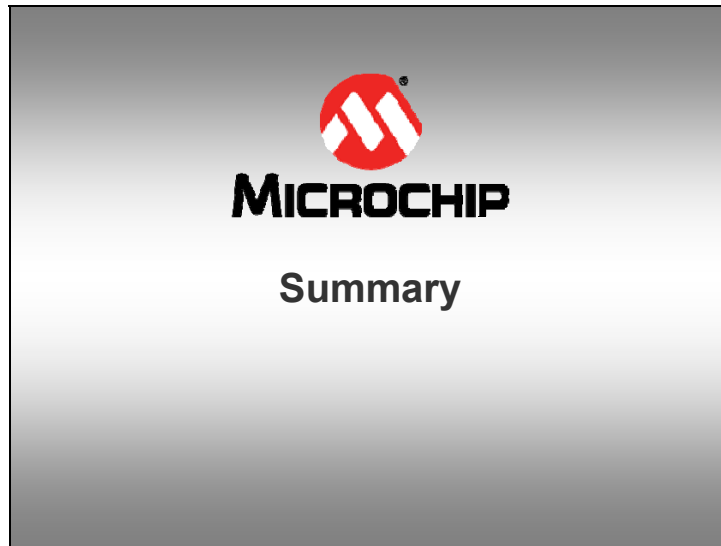
The Sublicense shall terminate on the earlier of:

- The date in which you have exceeded the limit of 10,000 units
- Upon notification of termination by either party




- Automatic termination due to violation of the license terms. You must discontinue use of the VID and PID immediately upon termination.

Slide 82



## Slide 83




### Summary

- USB provides an easy user interface
  - This means it's a complex protocol
- Microchip's Low Pin-Count USB Solutions
  - Microchip's Full-Speed USB Firmware Framework
    - abstracts the developer from the complexity
  - Low Pin-Count USB Development Kit
    - user friendly development environment

Getting Started with Microchip's Low Pin Count USB Solutions Slide 83

This class has introduced the USB protocol on a moderately high-level. This communication protocol is rapidly becoming the standard due to its ease of use. However, this comes at a price for the developer. “Easy for the user” means a complex protocol. Microchip's Low Pin-Count USB solutions provide the user with the tools that will ease the development of their own USB designs. The Full-Speed USB Firmware Framework abstracts the low-level USB functional complexities allowing the developer to concentrate on the application and not the protocol. The Low Pin-Count USB Development Kit provides a user friendly platform from which to bench test applications and ease the transition to custom designs.

## Slide 84




### References

- MCHPFSUSB Firmware User's Guide (DS51679B)
- "Windows System Programming, 3/E" – Johnson Hart
- "Microsoft Windows Internals, 4/E" – Mark Russinovich
- "Operating System Concepts, 7/E" – Silbershatz
- "Modern Operating Systems, 3/E" – Andrew Tanenbaum

Getting Started with Microchip's Low Pin Count USB Solutions Slide 84

## Slide 85



### Other Resources

- Microchip USB Design Center: Firmware, Custom Driver, App. Notes, other USB specific resources
  - <http://www.microchip.com/usb>
- USB 2.0 Specifications and Device Class Specifications
  - <http://www.usb.org>
- Developers Discussion Forum
  - <http://www.usb.org/phpbb/>
  - <http://forum.microchip.com/tt.aspx?forumid=102>
- Application for Sublicense to Microchip Universal Serial Bus Vendor ID
  - [http://www.microchip.com/stellent/groups/sitecomm\\_sg/documents/market\\_communication/en025058.pdf](http://www.microchip.com/stellent/groups/sitecomm_sg/documents/market_communication/en025058.pdf)

Getting Started with Microchip's Low Pin Count USB Solutions Slide 85

Slide 86




**MICROCHIP**

**Thank you !**

**Next step...complete the Project Labs  
at the end of the Low Pin-Count USB  
Development Kit User's Guide**

## Slide 87



# Trademarks

---

The Microchip name and logo, the Microchip logo, Accuron, dsPIC, KeeLog, KeeLog logo, microID, MPLAB, PIC, PICmicro, PICSTART, PRO MATE, rPIC and SmartShunt are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AmpLab, FilterLab, Linear Active Thermistor, Migratable Memory, MXDEV, MXLAB, SEEVAL, SmartSensor 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, ECAN, ECONOMICMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, PICKit, PICDEM, PICDEM.net, PICLAB, PICtail, PowerCal, PowerInfo, PowerMate, PowerTool, REALICE, rLAB, Select Mode, Smart Serial, SmartTel, Total Endurance, UNI/O, 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.

---

Getting Started with Microchip's Low Pin Count USB SolutionsSlide 87