



Microchip USB Device Firmware Framework User's Guide

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights.

Trademarks

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

FilterLab, Linear Active Thermistor, 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, dsSPEAK, ECAN, ECONOMONITOR, FanSense, In-Circuit Serial Programming, ICSP, ICEPIC, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, mTouch, PICkit, PICDEM, PICDEM.net, PICTail, PIC³² logo, PowerCal, PowerInfo, PowerMate, PowerTool, REAL ICE, rLAB, Select Mode, 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.

© 2008, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
== ISO/TS 16949:2002 ==

Microchip received ISO/TS-16949:2002 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.



MICROCHIP USB DEVICE FIRMWARE FRAMEWORK USER'S GUIDE

Table of Contents

Preface	1
Chapter 1. Using the USB Device Firmware Framework	
1.1 Highlights	5
1.2 Overview of the Framework	5
1.3 USB Device Firmware in the Framework	8
Index	13
Worldwide Sales and Service	14

Microchip USB Device Firmware Framework User's Guide

NOTES:



MICROCHIP USB DEVICE FIRMWARE FRAMEWORK USER'S GUIDE

Preface

NOTICE TO CUSTOMERS

All documentation becomes dated, and this manual is no exception. Microchip tools and documentation are constantly evolving to meet customer needs, so some actual dialogs and/or tool descriptions may differ from those in this document. Please refer to our web site (www.microchip.com) to obtain the latest documentation available.

Documents are identified with a “DS” number. This number is located on the bottom of each page, in front of the page number. The numbering convention for the DS number is “DSXXXXA”, where “XXXX” is the document number and “A” is the revision level of the document.

For the most up-to-date information on development tools, see the MPLAB® IDE on-line help. Select the Help menu, and then Topics to open a list of available on-line help files.

INTRODUCTION

This document discusses the technical details and functionality of the Microchip USB Device Firmware Framework. It assumes that the programmer already is familiar with the following:

- How to write C programs
- How to use the MPLAB Integrated Development Environment (IDE)
- The microcontroller data sheet for which the code is being written
- Basic USB concepts, such as those covered in chapters 5, 8 and 9 in the official USB 2.0 specifications

Items discussed in this chapter include:

- Conventions Used in this Guide
- Recommended Reading
- The Microchip Web Site
- Development Systems Customer Change Notification Service
- Customer Support
- Document Revision History

CONVENTIONS USED IN THIS GUIDE

This manual uses the following documentation conventions:

DOCUMENTATION CONVENTIONS

Description	Represents	Examples
Arial font:		
Italic characters	Referenced books	<i>MPLAB[®] IDE User's Guide</i>
	Emphasized text	...is the <i>only</i> compiler...
Initial caps	A window	the Output window
	A dialog	the Settings dialog
	A menu selection	select Enable Programmer
Quotes	A field name in a window or dialog	"Save project before build"
Underlined, italic text with right angle bracket	A menu path	<u><i>File>Save</i></u>
Bold characters	A dialog button	Click OK
	A tab	Click the Power tab
Text in angle brackets < >	A key on the keyboard	Press <Enter>, <F1>
Courier New font:		
Plain Courier New	Sample source code	<code>#define START</code>
	Filenames	<code>autoexec.bat</code>
	File paths	<code>c:\mcc18\h</code>
	Keywords	<code>_asm, _endasm, static</code>
	Command-line options	<code>-Opa+, -Opa-</code>
	Bit values	<code>0, 1</code>
	Constants	<code>0xFF, 'A'</code>
Italic Courier New	A variable argument	<i>file.o</i> , where <i>file</i> can be any valid filename
Square brackets []	Optional arguments	<code>mcc18 [options] file [options]</code>
Curly brackets and pipe character: { }	Choice of mutually exclusive arguments; an OR selection	<code>errorlevel {0 1}</code>
Ellipses...	Replaces repeated text	<code>var_name [, var_name...]</code>
	Represents code supplied by user	<code>void main (void) { ... }</code>

Microchip USB Device Firmware Framework User's Guide

RECOMMENDED READING

This user's guide describes how to use the Microchip USB Device Firmware Framework. Other documents which may be useful on this subject are listed below. The following documents are available and recommended as supplemental reference resources.

Microchip USB Device Firmware Framework Release Notes

As new versions of the USB Device Firmware Framework are developed, they will be distributed along with release notes that may cover specific and key items not necessarily covered in this User's Guide. When new versions are released, they will be posted at the Microchip USB design center:

<http://www.microchip.com/usb/> (click on the "Full-Speed USB Solutions" link)

PIC24FJ256GB110 PIM Manual (DS39908)

USB PICtail™ Plus Daughter Board Manual (DS39909)

PICDEM™ FS USB Demonstration Board User's Guide (DS51526)

PIC18F87J50 FS USB Plug-In Module User's Guide (DS51678)

These documents provide information on the hardware configuration of Microchip's USB development and demonstration kits.

Official USB 2.0 Specifications

Chapter 9 in the official USB 2.0 specifications covers the commands that all USB peripheral devices must support. This chapter is especially important as it strongly influences the requirements of USB firmware. Chapters 5 and 8 also provide useful information regarding how data moves across the USB.

The official USB specifications can be downloaded from the USB Implementers Forum web site:

<http://www.usb.org/>

THE MICROCHIP WEB SITE

Microchip provides online support via our web site at www.microchip.com. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

DEVELOPMENT SYSTEMS CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at www.microchip.com, click on Customer Change Notification and follow the registration instructions.

The Development Systems product group categories are:

- **Compilers** – The latest information on Microchip C compilers and other language tools. These include the MPLAB C18 and MPLAB C30 C compilers; MPASM™ and MPLAB ASM30 assemblers; MPLINK™ and MPLAB LINK30 object linkers; and MPLIB™ and MPLAB LIB30 object librarians.
- **Emulators** – The latest information on Microchip in-circuit emulators. This includes the MPLAB ICE 2000 and MPLAB ICE 4000.
- **In-Circuit Debuggers** – The latest information on the Microchip in-circuit debugger, MPLAB ICD 2.
- **MPLAB® IDE** – The latest information on Microchip MPLAB IDE, the Windows® Integrated Development Environment for development systems tools. This list is focused on the MPLAB IDE, MPLAB SIM simulator, MPLAB IDE Project Manager and general editing and debugging features.
- **Programmers** – The latest information on Microchip programmers. These include the MPLAB PM3 and PRO MATE® II device programmers and the PICSTART® Plus and PICKit™ 1 development programmers.

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or field application engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at: <http://support.microchip.com>

DOCUMENT REVISION HISTORY

Revision B (March 2008)

- Complete revision to support the Microchip USB Device Firmware framework.

Revision A (September 2007)

- Initial version of the document, released under the title "MCHPFSUSB Firmware User's Guide"; written to support the predecessor the Firmware Framework.

Chapter 1. Using the USB Device Firmware Framework

1.1 HIGHLIGHTS

The items discussed in this chapter are:

- Overview of the Framework
- USB Device Firmware in the Framework

1.2 OVERVIEW OF THE FRAMEWORK

The Microchip USB Device Firmware Framework is a library that can be used to create new USB applications. It can be thought of as a reference design project, containing the necessary firmware code for USB operation and providing a placeholder for the user's code. The whole code project is contained within one single root project directory, with many subdirectories for source code organization.

The USB Framework is based on the latest versions (as this is written) of Microchip's development tools. To provide the best level of USB application support, you should verify that you have at least these revisions of the following tools before starting:

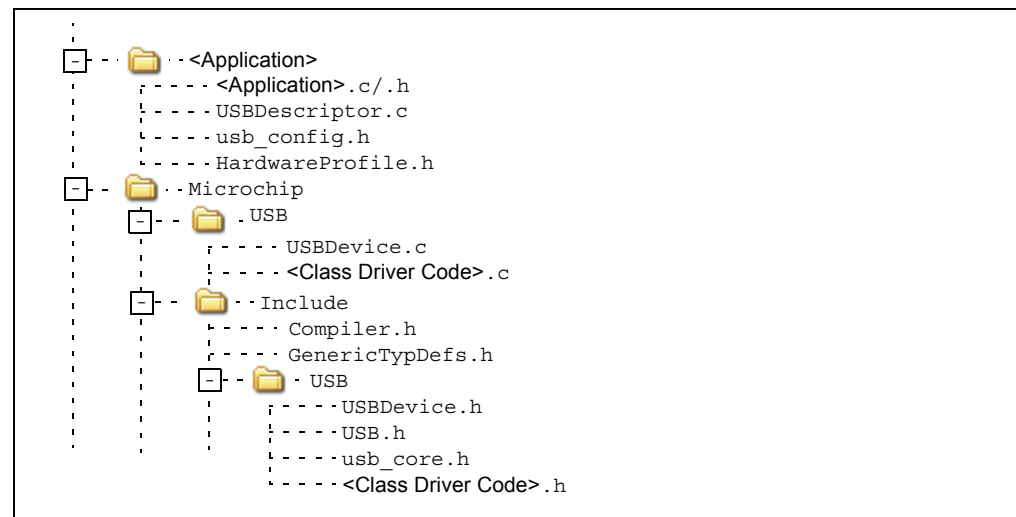
- MPLAB® IDE, v 8.02
- Microchip C30 C Compiler, v 3.02 or later; or Microchip C18 C Compiler, v3.10 or later

This section describes the importance in setting up the project paths and how the Framework is organized.

1.2.1 The Directory Structure

The Framework is designed in such a way that files located in the `Microchip` directory do not need modification. All files that need to be modified by the user are located in the `<Application>` folder. The directory structure of the USB Device Firmware Framework is the following:

FIGURE 1-1: FIRMWARE FRAMEWORK DIRECTORY STRUCTURE



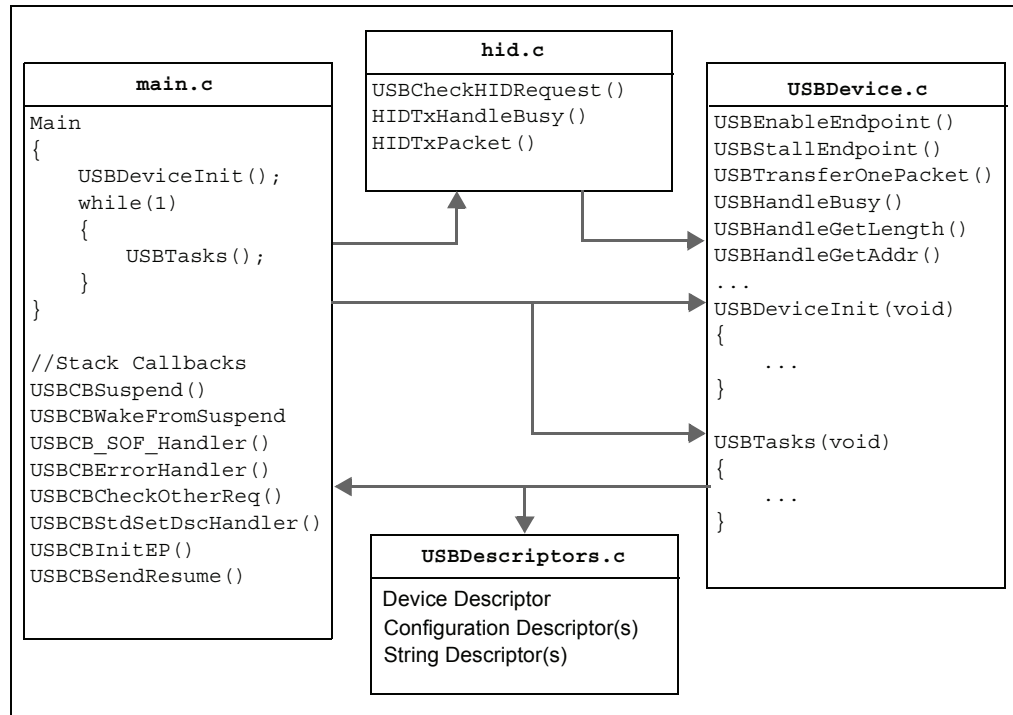
Using the USB Device Firmware Framework

1.2.2 The Logical Structure

The USB Firmware Framework provides a set of modular firmware interfaces that handle most of the work for implementing USB communications. Figure 1-2 shows a typical USB program flow. Each firmware reference project is written to have a cooperative multitasking environment; thus, no blocking functions should be implemented.

The `main()` function is an infinite loop that services different tasks. These can be logically thought of as either a USB task or a user task. USB tasks are handled by `USBTasks()`, which polls and services all USB hardware interrupts.

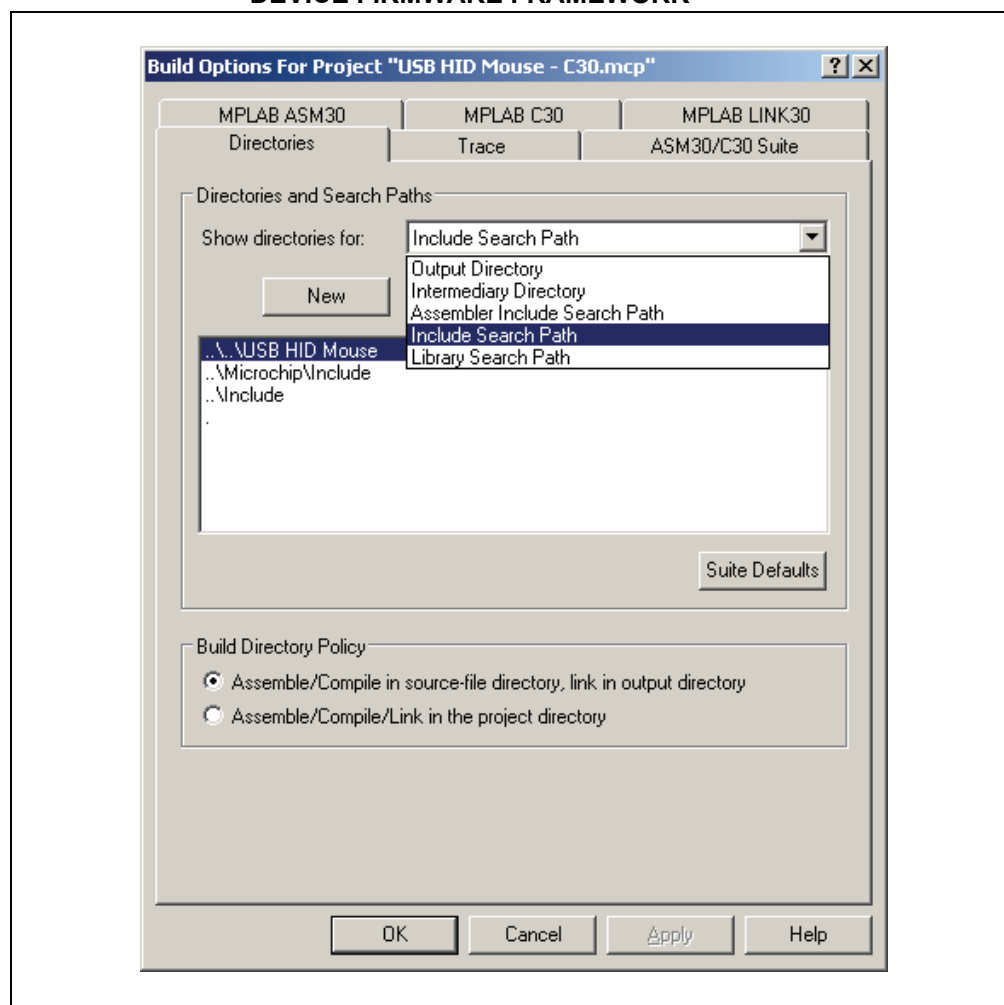
FIGURE 1-2: LOGICAL STRUCTURE OF THE FIRMWARE FRAMEWORK



1.2.3 Configuring MPLAB IDE for the Framework

1. Launch MPLAB IDE.
2. From the MPLAB IDE menu bar, select *File > Open Workspace...*
3. Navigate and select the workspace file. Please see the release notes associated with the stack for more information about the reference designs available.
4. Return to the menu bar and select *Project > Build Options... > Project*. The Build Options dialog appears (Figure 1-3).
5. Click on the **Directories** tab and verify the following:
 - The **Output Directory** and **Intermediary Directory** paths point to the directory where the output files should be placed.
 - The **Include Search Path** directory includes the following entries: “.”, “..\\Include”, “..\\Microchip\\Include”, and “..\\.<Application>” (see Figure 1-3 for an example). This allows the files within the Framework to locate the application-specific configuration files as well as the application-specific code to reach the Framework functions.
 - The **Library Search Path** points to the “lib” directory under the appropriate compilers installation directory.
 - The **Linker-Script Path** points to the directory of the project’s linker script file.

FIGURE 1-3: CONFIGURING MPLAB® IDE FOR THE MICROCHIP USB DEVICE FIRMWARE FRAMEWORK



Note: In some cases, the C18 compiler may fail to compile correctly, giving an error that certain header files from the `mcc18` library could not be found. In this case, add the path `<C18 Installation directory>\h` after the project root path in the **Include Path** field. This is often `C:\mcc18`.

1.2.4 Selecting the Hardware Configuration

The Framework is set up to automatically select the correct compile-time options for the standard demo boards available from Microchip. In order to get the code to compile and link correctly, there are still a few steps that may be required by the user.

To use these projects with other microcontrollers or circuit board platforms, the following changes will be needed:

1. Open the MPLAB IDE project.
2. From the menu bar, select the microcontroller which will be used (*Configure > Select Device*).
3. Replace the linker script in the project with the appropriate device-specific linker script. These can be found in the `\lkr` directory inside the installation directory for the C18 compiler or the `\support\gld` directory inside the installation directory for the C30 compiler.

Using the USB Device Firmware Framework

Although the USB-specific registers are similar between all of the different Microchip USB products, a few differences still exist. For example, when configured for a PLL-Based Oscillator mode, the PIC18F4550 will power-up by default with the PLL enabled. On the PIC18F87J50 family of devices, the PLL is initially disabled even when configured for a PLL-Based Oscillator mode. If the USB module clock is being derived from the PLL, user firmware must set the PLEN bit (OSCTUNE<6>) bit at least two milliseconds before enabling the USB module.

To accommodate these types of differences, the source code for the Framework uses `#if defined` statements based on the processor to change the code accordingly. The selection of the demo board is found in the `HardwareProfile.h` file of the example code.

If developing for a non-Microchip demo board platform, additional changes may also be needed. The example framework projects make use of the hardware features available on the demo boards, such as LEDs, push buttons, temperature sensor, RS-232 connector, etc. When developing for a different platform, these aspects of the projects will either need to be replaced or removed. The `HardwareProfile.h` file maps functions to pin names and will likely need modification. See **Section 1.3.1.3 “HardwareProfile.h”** for more details.

1.3 USB DEVICE FIRMWARE IN THE FRAMEWORK

As seen in the directory structure section above, there are three main folders associated with the USB Device Framework:

- The application folder, “<Application>”
- The USB Framework source folder, `Microchip\USB`
- The Framework header folder, `Microchip\Include\USB`

Both the application folder and the Framework folders will be discussed in detail.

1.3.1 Application Folder

The application folder contains the application-specific information. This includes the configuration information specific to this project, the application code itself, the USB descriptors and the MPLAB IDE associated files.

1.3.1.1 `usb_config.h`

This header file is a key element to the configuration of the USB stack. It defines various parameters inside the stack that determine how the stack will operate and which optional features to include. Changing these options can have implications of code size, RAM usage and data throughput. `USB_SUPPORT_DEVICE` is the definition that enabled the entire device/peripheral stack. Without this definition, the stack will not run in Device mode.

- **`EP0_BUFF_SIZE`** defines the buffer size for Endpoint 0. It can have a valid value of 8, 16, 32 or 64 bytes. This definition is used globally in the project for many things. It is used during project build to allocate appropriate buffer size for Endpoint 0. It is used in the USB descriptor to notify the USB host of the size of the Endpoint 0 buffer. It is also used during control transfer operation.

When defining this variable, note that a low-speed USB device can only use an 8-byte buffer, while a full-speed USB device can use an 8, 16, 32 or 64-byte buffer.

- **`MAX_NUM_INT`** defines the size of the array which keeps track of the active alternate setting for each interface, which can be switched during operation. Valid values are integers [0, 1, 2,...]. If a device has multiple configurations, the number of interfaces from the configuration with the highest number of interfaces should be used. For example, a device with two configurations has three interfaces in the first configuration and two interfaces in the second. In this case, `MAX_NUM_INT` should be three.

Microchip USB Device Firmware Framework User's Guide

- **USB_PING_PONG_MODE** defines the Ping-Pong Buffer mode to be used during run time. The function of each mode is explained in the USB chapter of the device data sheet. The options for this setting are:

```
USB_PING_PONG__NO_PING_PONG
USB_PING_PONG__EP0_OUT_ONLY
USB_PING_PONG__FULL_PING_PONG
USB_PING_PONG__ALL_BUT_EP0
```

Not all of these settings may be available for every device. Please check with the appropriate device data sheet.

- **USB_USE_CLASS** is used to indicate which USB classes should be included in the code project. The options for this setting are the USB classes with class-specific header files implemented in the Framework, including (among others):

```
USB_USE_CDC
USB_USE_GEN
USB_USE_HID
USB_USE_MSD
```

When one or more of these are defined, it tells the USB global header file, `usb.h`, which class-specific header files to include. The `usb.h` header is used globally as the necessary include file when using the USB library. If the HID class is used, then `hid.c` and `hid.h` should also be added to the MPLAB IDE project. If the CDC class is used, then `cdc.c` and `cdc.h` should also be added to the MPLAB IDE project.

- **USE_USB_BUS_SENSE_IO** indicates that the firmware will use the pin defined in `HardwareProfile.h` to determine when to enable the USB module. If the target board design does not use an I/O pin to detect the presence of the USB bus, this definition must be commented out.

When `USE_USB_BUS_SENSE_IO` is undefined, the USB module will always be enabled. Using this feature helps to improve the power efficiency of the system because the USB module is only enabled when the bus is present. Additionally, in order for the device to pass USB compliance certification, all self-powered devices are required to support a bus sense feature. Self-powered devices which do not implement this feature will fail the back drive voltage tests. The USB specifications require that devices should not source current on D+ or D- (and never VBUS) unless the host is actively powering the VBUS line. A self-powered device will not know when the host is actively powering VBUS (and when it is acceptable to enable the D+ or D- pull-up resistor) unless a bus sense feature is implemented. Purely bus-powered devices do not need to implement this feature and this feature may be commented out.

- **USE_SELF_POWER_SENSE_IO** indicates that the microcontroller is sensing the presence of on-board power through an I/O pin. If the target board design does not use an I/O pin to detect the presence of self-power, this definition must be commented out.
- **USB_MAX_EP_NUMBER** must equal the highest endpoint number used in the project. For example, if the highest endpoint number used is Endpoint 5, then `USB_MAX_EP_NUMBER` should equal five. This definition is used mainly in the `usbmmmap.c` to allocate the buffer descriptor registers.
- **USB_DEVICE_DESCRIPTOR** is the name of the ROM variable that contains the device descriptor information.
- **USB_CONFIG_DESCRIPTOR** is the name of the ROM variable that contains the configuration descriptor information.

In addition to the options listed above, there may be additional definitions required in `usb_config.h` in order to run certain USB class code with the Framework. Please refer to the documentation for each class in order to determine the definitions that are required.

Using the USB Device Firmware Framework

1.3.1.2 USBDescriptors.c

This file contains the USB descriptor information for the device. This information varies based on the application.

A typical configuration descriptor consists of these components:

- At least one configuration descriptor
- One or more interface descriptors
- One or more endpoint descriptors

In addition, there is usually a descriptor string that provides a plain text description of the device.

1.3.1.2.1 Configuration Descriptor

The `bmAttributes` object of the configuration descriptor can be modified to meet the needs to the specific application. The available options are shown in Table 1-1. The selected options should be ORed together to form the corresponding entry (e.g., a device with all of the options should have “`_DEFAULT | _SELF | _RWU`”).

For more details about the configuration descriptor, please see Table 9-10 of the USB specification.

TABLE 1-1: `bmAttributes` OPTIONS FOR CONFIGURATION DESCRIPTOR

Option	Description
<code>_DEFAULT</code>	Required definition that sets the reserved bits of this attribute as required by the specification.
<code>_SELF</code>	Designate this device as a self-powered device.
<code>_RWU</code>	Designate this device as having remote wake-up functionality.

1.3.1.2.2 Endpoint Descriptor

Both the `bEndpointAddress` and `bmAttributes` field of the endpoint descriptors have definitions available in the stack to select the various options. The options should be ORed together to form the desired set of options. The available options are shown in Table 1-2 and Table 1-3.

More information about the endpoint descriptors can be found in Table 9-13 of the USB specification.

TABLE 1-2: `bEndpointAddress` OPTIONS

Option	Description
<code>_EP_IN</code>	Defines the endpoint as an IN endpoint. This definition must be ORed with the endpoint number to form the required value (e.g., <code>MY_ENDPOINT_NUMBER _EP_IN</code>).
<code>_EP_OUT</code>	Defines the endpoint as an OUT endpoint. This definition must be ORed with the endpoint number to form the required value (e.g., <code>MY_ENDPOINT_NUMBER _EP_OUT</code>).
<code>_EPxx_OUT</code>	Defines the endpoint as an OUT endpoint where <code><xx></code> is the endpoint number. No other definitions are required.
<code>_EPxx_IN</code>	Defines the endpoint as an IN endpoint where <code><xx></code> is the endpoint number. No other definitions are required.

TABLE 1-3: bmAttributes OPTIONS FOR ENDPOINT DESCRIPTOR

Option	Description
<code>_CTRL</code>	This endpoint is used for control transfers.
<code>_ISO</code>	This endpoint is used for isochronous transfers.
<code>_BULK</code>	This endpoint is used for bulk transfers.
<code>_INT</code>	This endpoint is used for interrupt transfers.
<code>_NS</code>	If <code>_ISO</code> is defined, the endpoint is not synchronized.
<code>_AS</code>	If <code>_ISO</code> is defined, the endpoint is asynchronous.
<code>_AD</code>	If <code>_ISO</code> is defined, the endpoint is adaptive.
<code>_SY</code>	If <code>_ISO</code> is defined, the endpoint is synchronous.
<code>_DE</code>	If <code>_ISO</code> is defined, the endpoint is a data endpoint.
<code>_FE</code>	If <code>_ISO</code> is defined, the endpoint is a feedback endpoint.
<code>_IE</code>	If <code>_ISO</code> is defined, the endpoint is an implicit feedback data endpoint.

If `_ISO` is defined, then one synchronous/adaptive option (`_NS`, `_AS`, `_AD` or `_SY`) and one data/feedback option (`_DE`, `_FE` or `_IE`) must also be defined. The default options are `_NS` and `_DE` (`_ISO` | `_NS` | `_DE`).

1.3.1.2.3 String Descriptors

Rather than appearing as a simple string of text, the descriptor is formatted in a particular data structure. The string descriptor array takes the format:

```
ROM struct
{
    BYTE bLength;
    BYTE bDscType;
    WORD string[size];
}sdxxx=
{
    sizeof(sdxxx),
    USB_DESCRIPTOR_STRING,
    {<text>}
};
```

This structure provides a means for the C compiler to calculate the length of string descriptor, `sdxxx`, where “xxx” is the string index number. The first two bytes of the descriptor are the descriptor length and type. The remaining `<text>` are string texts which must be in Unicode format. This is achieved by declaring each character as a word type. The whole text string is declared as a word array with the number of characters equal to `<size>`, which must be calculated manually by counting characters and then entered into the array declaration.

For example, if the string is `.USB.`, then the string descriptor should be (using index 02):

```
ROM struct
{
    BYTE bLength;
    BYTE bDscType;
    WORD string[3];
}sd002=
{
    sizeof(sd002),
    USB_DESCRIPTOR_STRING,
    {'U','S','B'}
};
```

Using the USB Device Firmware Framework

A USB project may have multiple strings. The firmware supports the management of multiple strings through a look-up table, which is defined as:

```
ROM BYTE *ROM USB_SD_Ptr[]={&sd000,&sd001,&sd002};
```

The above example has 3 strings (`sd000`, `sd001` and `sd002`). Strings can be removed or added as needed. The strings are managed by the look-up table, `USB_SD_Ptr`; the index of the string must match the index position of the array (`&sd000` must be in position, `USB_SD_Ptr[0]`, and so on). The look-up table, `USB_SD_Ptr`, is used by the `USBStdGetDscHandler` function in `usb9.c`. The string descriptor, `sd000`, is a specialized descriptor that defines the language code, which is usually US English (0x0409).

1.3.1.2.4 Adding Configurations

A USB device may have more than one configuration descriptor (e.g., `configDescriptor2`, `configDescriptor3`, etc.). To add another configuration descriptor, implement a new set of structures, similar to `configDescriptor1`, to the `USBDescriptors.c` file. Once this is done, add the new configuration descriptor name (`configDescriptor2`, `configDescriptor3`) to the look-up table, `USB_CD_Ptr`, in the same method used for managing descriptor strings. `USB_CD_Ptr[0]` is a dummy place holder for configuration 0, the unconfigured state defined by the USB specification. The configuration handler, `USBStdSetCfgHandler`, must also be modified to support the additional configurations.

1.3.1.3 HardwareProfile.h

This file is used to map the various demo board hardware setups to a common definition for the code to use. This defines which port pins LEDs are located on, the clock speed of the demo board, etc. This file will also determine which demo board is currently being used by looking at the compiler that is compiling the project and the device selected in MPLAB IDE. If a custom board is required, or if a modification is made to one of the demo boards, then the `DEMO_BOARD` definition needs to be defined at the top of `HardwareProfile.h` to indicate to the compiler that it is a custom board. Custom boards may also require the addition of code for other features used by the framework.

1.3.2 Framework Folders

The Application Programming Interface (API) to the Framework is available in the documentation folder of the Framework distribution. This document provides a description and examples of the functions required in order to use the Framework

Index

A

Adding Configurations.....	12
Application Folder	8

B

bEndpointAddress Options	10
bmAttributes Options	
Configuration Descriptor	10
Endpoint Descriptor	11

C

Configuration Descriptor	10
Customer Notification Service.....	4
Customer Support.....	4

D

Documentation	
Conventions	2

E

Endpoint Descriptor	10
---------------------------	----

F

Firmware Framework	
Directory Structure	5
Firmware Framework	
Configuring MPLAB	6
Selecting Hardware Configuration	7
Firmware Framework	
Logical Structure	6
Framework Device Firmware	8
Framework Folders	12

H

HardwareProfile.h.....	12
------------------------	----

I

Internet Address.....	3
-----------------------	---

M

Microchip Internet Web Site	3
-----------------------------------	---

R

Reading, Recommended	3
----------------------------	---

U

usb_config.....	8
USBDescriptors.c	
Adding Configurations.....	12
String Descriptor Format.....	11
USBDescriptors.c.....	10

W

WWW Address.....	3
------------------	---



WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office

2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
<http://support.microchip.com>
Web Address:
www.microchip.com

Atlanta

Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Boston

Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago

Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas

Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit

Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

Kokomo

Kokomo, IN
Tel: 765-864-8360
Fax: 765-864-8387

Los Angeles

Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

Santa Clara

Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

Toronto

Mississauga, Ontario,
Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office

Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

Australia - Sydney

Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing

Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

China - Chengdu

Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Hong Kong SAR

Tel: 852-2401-1200
Fax: 852-2401-3431

China - Nanjing

Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao

Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai

Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang

Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen

Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

China - Wuhan

Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xiamen

Tel: 86-592-2388138
Fax: 86-592-2388130

China - Xian

Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

China - Zhuhai

Tel: 86-756-3210040
Fax: 86-756-3210049

ASIA/PACIFIC

India - Bangalore

Tel: 91-80-4182-8400
Fax: 91-80-4182-8422

India - New Delhi

Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune

Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

Japan - Yokohama

Tel: 81-45-471- 6166
Fax: 81-45-471-6122

Korea - Daegu

Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul

Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur

Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang

Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila

Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore

Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu

Tel: 886-3-572-9526
Fax: 886-3-572-6459

Taiwan - Kaohsiung

Tel: 886-7-536-4818
Fax: 886-7-536-4803

Taiwan - Taipei

Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

Thailand - Bangkok

Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels

Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen

Tel: 45-4450-2828
Fax: 45-4485-2829

France - Paris

Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Munich

Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan

Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands - Drunen

Tel: 31-416-690399
Fax: 31-416-690340

Spain - Madrid

Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

UK - Wokingham

Tel: 44-118-921-5869
Fax: 44-118-921-5820