dsPIC33E/PIC24E Bootloader ReadMe

INTRODUCTION

The Bootloader for dsPIC33E/PIC24E devices can be used to upgrade firmware on a target device without the need for a programmer or debugger. The following are three implementations of Bootloaders for dsPIC33E/PIC24E devices with the auxiliary Flash segment:

- UART
- USB device based on the HID USB class
- USB Host based on the mass storage USB class

The Bootloader consists of these applications:

- The Bootloader firmware, which is to be programmed into the target dsPIC33E/PIC24E device
- A demonstration application, which can be downloaded into the target dsPIC33E/PIC24E device using the Bootloader
- A PC host application (required for the UART and USB HID Bootloaders only) to communicate with the Bootloader firmware running inside the dsPIC33E/PIC24E device. This application is used to perform erase and programming operations.

PREREQUISITES

The prerequisites for running the Bootloader application are as follows:

- A target device with auxiliary Flash
- A PC with MPLAB® IDE v8.60 or higher version installed
- One of the following hardware resources:
  - dsPIC33E/PIC24E USB Starter Kit
  - Explorer 16 Development Board with a dsPIC33E/PIC24E Plug-in Module (PIM) for a device with auxiliary Flash
- A USB-to-serial port converter (if the COM port is not available on the PC) for the UART Bootloader
- A thumb drive for use with the USB Mass Storage Bootloader
- A traditional programming tool for initially writing the Bootloader firmware into the dsPIC33E/PIC24E device (such as the MPLAB® REAL ICE™ In-Circuit Emulator or the ICD 3 In-Circuit Debugger). Starter kits do not require any programming tools.

The user should be familiar with the following concepts:

- Device Configuration registers
- Compiling and programming the dsPIC33E/PIC24E device
- dsPIC33E/PIC24E linker scripts

CONCEPT

The Bootloader application is placed in the auxiliary Flash segment. This document applies only to dsPIC33E/PIC24E devices with an auxiliary Flash segment.

The Bootloader code begins executing upon a device Reset. If there is no valid application, or if there is an external trigger, the Bootloader application enters Firmware Upgrade mode.

If there is no external trigger, but there is a valid user application, the Bootloader code branches the control to the user application and begins executing the user application. The external trigger is a push button switch that the user must press during device boot-up. The Bootloader application tests the validity of the user application by reading its reset address content. The Bootloader application stays in Firmware Upgrade mode if this content is found to have been erased.

In Firmware Upgrade mode for the USB Mass Storage Bootloader, the Bootloader waits for storage media to be attached to the USB port. When the storage media is detected, it then checks for a properly named application firmware file. If the file is found, the application firmware is programmed into the program Flash. If programming is successful, a jump is made to the application's reset address.

In Firmware Upgrade mode for the UART and USB HID Bootloaders, the application continuously checks for commands from the PC host application. When a command is received from the PC host application, the Bootloader performs the appropriate action, such as erasing or programming the program Flash memory, and so on. The flowchart in Figure 2 illustrates the operation of the Bootloader application.

IMPLEMENTATION OVERVIEW (UART AND USB HID)

The Bootloader application is implemented using a framework. The Bootloader firmware communicates with the PC host application by using a predefined communication protocol. The Bootloader framework provides Application Programming Interface (API) functions to handle the protocol related frames from the PC application. For more information on the communication protocol, see **"Bootloader Communication Protocol (UART and USB HID)"**.

FRAMEWORK (UART AND USB HID)

The Bootloader framework provides several API functions, which can be called by the Bootloader application and the transport layer. The Bootloader framework helps the user to easily modify the Bootloader application to suit different requirements. The Bootloader architecture is illustrated in Figure 1.

The `Bootloader.c` file contains the Bootloader application code. This file includes the Bootloader functionality, and is illustrated in the form of a flowchart in Figure 2.

FIGURE 1: BOOTLOADER ARCHITECTURE (UART AND USB HID)

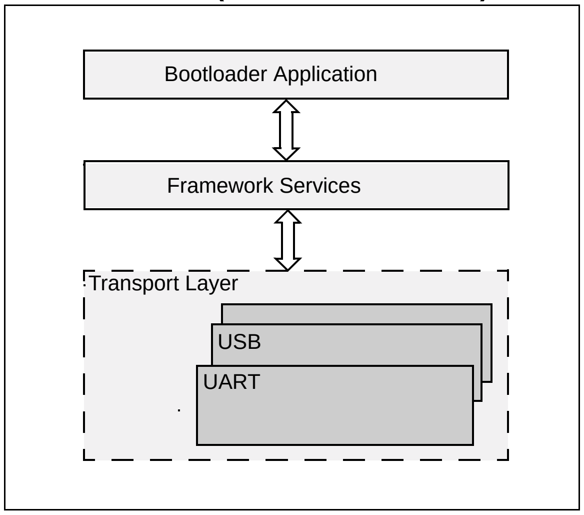
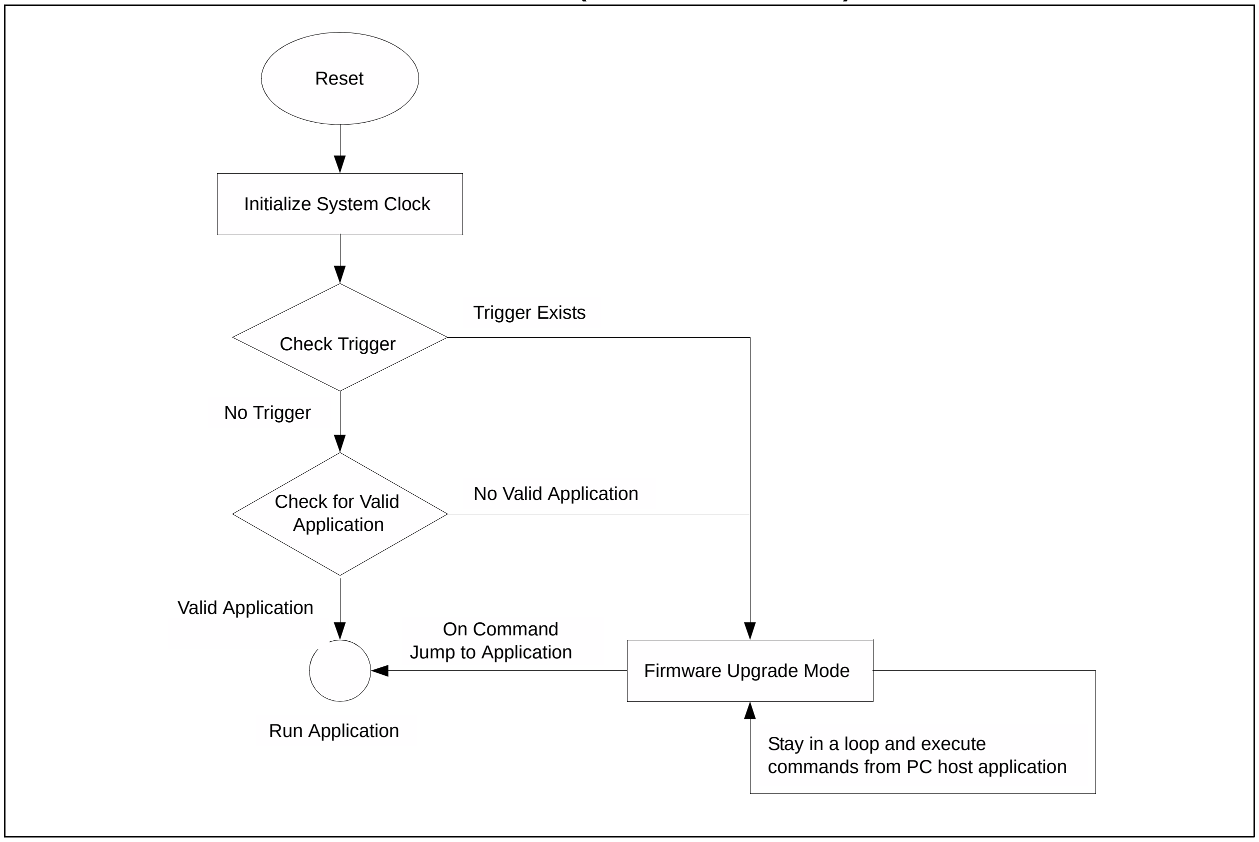


FIGURE 2: BOOTLOADER OPERATION (UART AND USB HID)



© 2011 Microchip Technology Inc.

The `Framework.c` file contains the framework functions. The framework handles the communication protocol frames and executes commands received from the PC host application.

The transport layer files, `Uart.c` and `Usb_HID_tasks.c`, include the functionality for transmitting and receiving the raw bytes to and from the PC host application.

The API functions provided by the framework are listed in Table 1.

TABLE 1: FRAMEWORK API DESCRIPTIONS (UART AND USB HID)

API	Description
`void FrameWorkTask(void)`	This function executes the command if there is a valid frame from the PC host application. It must be called periodically from the Bootloader application task. **Input Parameters:** None. **Return:** None.
`void BuildRxFrame(UINT8 *RxData, INT16 RxLen)`	The transport layer calls this function when it receives data from the PC host application. **Input Parameters:** `*RxData`: Pointer to the received data buffer. `RxLen`: Length of the received data bytes.
`UINT GetTransmitFrame(UINT8* TxData)`	Returns a non-zero value if there is a valid response frame from the framework. The transport layer calls this function to check if there is a frame to be transmitted to the PC host application. **Input Parameters:** `*TxData`: Pointer to a data buffer that will carry the response frame. **Return:** Length of the response frame. Zero indicates no response frame.
`BOOL ExitFirmwareUpgradeMode(void)`	Framework directs the Bootloader application to exit the Firmware Upgrade mode and execute the user application. This can happen when the PC host application issues the `run application` command. **Input Parameters:** None. **Return:** If value is true, exit Firmware Upgrade mode.

BOOTLOADER WORKSPACES

The `Bootloader` folder contains the Bootloader firmware source code. The available MPLAB workspaces are listed in Table 2.

Programming the Bootloader Firmware

Use the following procedure to select a suitable workspace depending on the hardware configuration and communication channel selected.

1. Using MPLAB IDE V8.60 or higher, open one of the Bootloader workspaces.
2. Select the correct device part number in IDE and rebuild the project.
3. After building the project, program the Bootloader into the device.
4. Use REAL ICE or ICD 3 to program the device on an Explorer 16 Development Board. The dsPIC33E/PIC24E USB Starter Kit does not require a programmer.

TABLE 2: AVAILABLE BOOTLOADER WORKSPACES

MPLAB® Workspace	Hardware Resources	Supported Devices	Communication Channel	Bootloader Mapping
`dsPIC33E_UART_Bootloader_Explorer16.mcp`	Explorer 16 Development Board and the dsPIC33E/PIC24E PIM	dsPIC33EP256MU806, dsPIC33EP256MU810, dsPIC33EP256MU814, dsPIC33EP512GP806, dsPIC33EP512MC806, dsPIC33EP512MU810, dsPIC33EP512MU814	UART	Entire Bootloader is mapped inside auxiliary Flash
`dsPIC33E_USB_HID_Bootloader_StarterKit.mcp`	dsPIC33E/PIC24E USB Starter Kit	dsPIC33EP256MU806, dsPIC33EP256MU810, dsPIC33EP256MU814, dsPIC33EP512MU810, dsPIC33EP512MU814	USB Device (HID Bootloader)	Entire Bootloader is mapped inside auxiliary Flash
`dsPIC33E_USB_MSD_Host_Bootloader_StarterKit.mcp`	dsPIC33E/PIC24E USB Starter Kit	dsPIC33EP256MU806, dsPIC33EP256MU810, dsPIC33EP256MU814, dsPIC33EP512MU810, dsPIC33EP512MU814	USB Host (Thumb Drive Bootloader)	Entire Bootloader is mapped inside auxiliary Flash
`PIC24E_UART_Bootloader_Explorer16.mcp`	Explorer 16 Development Board and the dsPIC33E/PIC24E PIM	PIC24EP256GU810, PIC24EP256GU814, PIC24EP512GP806, PIC24EP512GU810, PIC24EP512GU814	UART	Entire Bootloader is mapped inside auxiliary Flash
`PIC24E_USB_HID_Bootloader_StarterKit.mcp`	dsPIC33E/PIC24E USB Starter Kit	PIC24EP256GU810, PIC24EP256GU814, PIC24EP512GU810, PIC24EP512GU814	USB Device (HID Bootloader)	Entire Bootloader is mapped inside auxiliary Flash
`PIC24E_USB_MSD_Host_Bootloader_StarterKit.mcp`	dsPIC33E/PIC24E USB Starter Kit	PIC24EP256GU810, PIC24EP256GU814, PIC24EP512GU810, PIC24EP512GU814	USB Host (Thumb Drive Bootloader)	Entire Bootloader is mapped inside auxiliary Flash

© 2011 Microchip Technology Inc.

DEMONSTRATION APPLICATION

The `Demo_Application` folder contains a sample demonstration application that controls two LEDs, causing them to blink.

Table 3 lists the two workspaces that support the Explorer 16 Development Board (referred to as development board) and the dsPIC33E/PIC24E USB Starter Kit (referred to as starter kit).

The demonstration application uses the default linker script file provided by the C30 compiler tool suite, which maps the entire application in the user program Flash memory. The application reset vector address is located at 0x000000. The Bootloader passes the control to this application reset vector address when there is a condition to execute the user application.

Use the following procedure to configure and build the project:

1. In MPLAB, open the desired workspace project.
2. Choose the desired device by selecting *Select > Configure > Select Device*.
3. Build the project in Release mode by pressing Ctrl+F10. Make sure the project compiles and generates a hex file.

The resulting application hex file can be downloaded into the development board or the starter kit via the Bootloader. This demonstration application controls LED D9 and LED D10 on the development board or LED1 and LED2 on the starter kit, causing them to blink.

ENTERING THE FIRMWARE UPGRADE MODE

To enter the Bootloader firmware upgrade mode, press and hold switch S3 on the development board during power-up. On the starter kit, press and hold switch SW3 during power-up.

EXITING THE FIRMWARE UPGRADE MODE

For the USB HID or UART Bootloader, the firmware upgrade mode can be exited either by applying a hard Reset to the device or by sending a “Jump to Application” command from the PC. For the Thumb Drive Bootloader, the firmware upgrade mode can be exited by applying a hard Reset to the device.

RUNNING THE USB HOST THUMB DRIVE BOOTLOADER

Use the following procedure to run the USB Host Thumb Drive Bootloader:

1. Program the USB Host MSD Bootloader into the starter kit, making sure to use Release Build.
2. After programming the device, unplug the USB debug cable from the starter kit.
3. Make sure MPLAB IDE is open and **Starter Kit on Board** is selected as the programmer.
4. Reconnect the USB debug cable to the starter kit and connect the other end of the USB cable to the PC.
5. Press and hold switch SW3 on the starter kit until MPLAB IDE has detected the starter kit. Once the starter kit is detected, LED3 will start blinking, which indicates that the Bootloader is in Firmware Upgrade mode.
6. Copy the application hex file to a USB thumb drive and rename the hex file to `image.hex`.
7. Insert the USB thumb drive into the starter kit. The Bootloader starts programming the image into the user Program Flash memory of the device. If applicable, the LED on the thumb drive blinks during the programming operation.

Once programming has completed, the Bootloader exits and starts running the application.

Note: If the device doesn’t recognize the thumb drive (or the file on the thumb drive), try formatting the thumb drive in the FAT32 format.

TABLE 3: DEMONSTRATION APPLICATION WORKSPACES

MPLAB® Workspace	Hardware Resource	Comments
`Demo_App_Explorer16.mcp`	Explorer 16 Development Board	Blinks LED D9 and LED D10
`Demo_App_Starter_Kits.mcp`	dsPIC33E/PIC24E USB Starter Kit	Blinks LED1 and LED2

BOOTLOADER CONFIGURATIONS

The Bootloader code provides macros for configuring the settings. Table 4 lists these macros and their usage. Depending on user requirements, the values of these macros may need to be modified.

TABLE 4: BOOTLOADER CONFIGURATION MACROS

Macro Type	Macro Name	File Name	Usage
Generic	`MAJOR_VERSION`, `MINOR_VERSION`	`Bootloader.h`	User macros for Bootloader version control.
UART Bootloader	`DEFAULT_BAUDRATE`	`Uart.h`	Sets the required UART baud rate.
USB HID Bootloader	`USB_VENDOR_ID`	`usb_descriptors.c`	Sets the Vendor ID.
	`USB_PRODUCT_ID`	`usb_descriptors.c`	Sets the Product ID.

LINKER SCRIPTS

A special linker script is provided for the Bootloader, which maps it entirely into the auxiliary Flash memory segment. As a result, the entire user program Flash memory is available for the application and there is no need to modify the default C30 linker script to accommodate the Bootloader.

HANDLING DEVICE CONFIGURATION WORDS

The Bootloader code provided along with this document does not erase or write the device Configuration words. This is because the device Configuration words settings are shared by both the Bootloader and the user application, and any accidental erase of the device Configuration words may make the Bootloader non-functional. Therefore, it is highly recommended to have common device Configuration words settings for both the user application and the Bootloader.

USING THE PC APPLICATION (USB HID AND SERIAL PORT BOOTLOADER ONLY)

Program the Bootloader into the device using a debugger/programmer, making sure to use Release Build. If using the development board, disconnect the programmer after programming the device. If using the starter kit, unplug the USB debug cable from the starter kit after programming the device.

1. Run `PC_Loader.exe`, which is located in the `..\PC_Application\` directory
2. Depending on the Bootloader workspace you are using, enable either the USB or Serial port using the Communication Settings menu in the PC application. For the serial port Bootloader, the default baud rate is 115200. For the USB Bootloader, the default values of VID and PID are 0x4D8 and 0x03C, respectively.
3. To enter the firmware upgrade mode, use the appropriate procedure for the selected hardware resource:

   Explorer 16: Press switch S3 and simultaneously apply the device Reset by pressing the $\overline{\text{MCLR}}$ switch (S1) on the development board. If the device enters the firmware upgrade mode successfully, LED D5 will blink.

   Starter Kits: Make sure MPLAB IDE is open and **Starter Kit on Board** is selected as the programmer. Connect the USB debug cable to the starter kit and connect the other end of the USB cable to the PC. Press and hold switch SW3 on the starter kit until it is detected by MPLAB IDE. Once detected, LED3 begins blinking, which indicates that the Bootloader is in Firmware Upgrade mode.
4. Connect the serial cable or micro-USB cable to the development board depending on the type of Bootloader programmed.
5. Click **Connect**. The device connects and the Bootloader version information is read.
6. Click **Load Hex File** and browse to `Demo_App_Starter_Kits`.
7. Select the demonstration application file `Demo_App_Explorer16.hex`, which is in the `Firmware\Demo_Application` folder.
8. Click **Erase** to erase the device.
9. Program the previously loaded hex file into the device Flash by clicking **Program**.
10. Verify the Flash contents by clicking **Verify**. If the Flash is written correctly, the *Verification successful* message is displayed on the console.
11. Run the application by clicking **Run Application**. LED D9 and D10 LED on the development board, or LED1 and LED2 on the starter kit start blinking. This confirms that the application is programmed and was loaded successfully. Optionally, steps 8, 9, and 10 can be performed as a single action by clicking **Erase-Program-Verify**.
12. Once the device starts running the programmed application firmware, the PC application will not be able to further communicate with the device. To reconnect to the Bootloader, go back to step 4.

© 2011 Microchip Technology Inc.

BOOTLOADER COMMUNICATION PROTOCOL (UART AND USB HID)

The PC host application uses a communication protocol to interact with the Bootloader firmware. The PC host application acts as a master and issues commands to the Bootloader firmware to perform specific operations.

Frame Format

The communication protocol follows the frame format shown in Example 1. The frame format remains the same in both directions, that is, from the host application to the Bootloader and from the Bootloader to the host application.

EXAMPLE 1: FRAME FORMAT

```
[<SOH>…]<SOH>[<DATA>…]<CRCL><CRCH><EOT>
```

Where:

`<...>` – Represents a byte

`[...]` – Represents an optional or variable number of bytes

The frame starts with a control character, Start of Header (SOH), and ends with another control character, End of Transmission (EOT). The integrity of the frame is protected by two bytes of CRC-16, represented by CRCL (low-byte) and CRCH (high-byte).

Control Characters

TABLE 5: CONTROL CHARACTER DESCRIPTIONS

Control	Hex Value	Description
<SOH>	0x01	Marks the beginning of a frame
<EOT>	0x04	Marks the end of a frame
<DLE>	0x10	Data link escape

Some bytes in the data field may imitate the control characters, SOH and EOT. The Data Link Escape (DLE) character is used to escape such bytes that could be interpreted as control characters. The Bootloader always accepts the byte following a <DLE> as data, and always sends a <DLE> before any of the control characters.

Commands

The PC host application can issue the commands listed in Table 6 to the Bootloader. The first byte in the data field carries the command.

TABLE 6: COMMAND DESCRIPTION

Command Value in Hexadecimal	Description
0x01	Read the Bootloader version information
0x02	Erase Flash
0x03	Program Flash
0x04	Read CRC
0x05	Jump to application

READ BOOTLOADER VERSION INFORMATION

The PC host application request for version information to the Bootloader is shown in Example 2.

EXAMPLE 2: REQUEST

```
[<SOH>…]<SOH>[<0x01>]<CRCL><CRCH><EOT>
```

The Bootloader responds to the PC's request for version information in two bytes, as shown in Example 3.

EXAMPLE 3: RESPONSE

```
[<SOH>…]<SOH><0x01><MAJOR_VER><MINOR_VER><C
RCL><CRCH><EOT>
```

Where:

`MAJOR_VER` – Major version of the Bootloader

`MINOR_VER` – Minor version of the Bootloader

ERASE FLASH

On receiving the Erase Flash command from the PC host application, the Bootloader erases that part of the program Flash, which is allocated for the user application.

The request frame from the PC host application to the Bootloader is shown in Example 4.

EXAMPLE 4: REQUEST

```
[<SOH>…]<SOH><0x02><CRCL><CRCH><EOT>
```

The response frame from the Bootloader to the PC host application is shown in Example 5.

EXAMPLE 5: RESPONSE

```
[<SOH>…]<SOH><0x02><CRCL><CRCH><EOT>
```

PROGRAM FLASH

The PC host application sends one or multiple hex records in Intel Hex format along with the Program Flash command. The MPLAB C30 compiler generates the image in the Intel Hex format. Each line in the Intel hexadecimal file represents a hexadecimal record. Each hexadecimal record starts with a colon (:) and is in ASCII format. The PC host application discards the colon and converts the remaining data from ASCII to hexadecimal, and then sends the data to the Bootloader. The Bootloader extracts the destination address and data from the hex record, and writes the data into program Flash.

The request frame from the PC host application to the Bootloader is shown in Example 6.

EXAMPLE 6: REQUEST

```
[<SOH>…]<SOH><0x03>[<HEX_RECORD>…]<CRCL><CR
CH><EOT>
```

Where:

HEX_RECORD – Intel hex record in hex format

The response from the Bootloader to the PC host application is shown in Example 7.

EXAMPLE 7: RESPONSE

```
[<SOH>…]<SOH><0x03><CRCL><CRCH><EOT>
```

READ CRC

The Read CRC command is used to verify the content of the program Flash after programming.

The request frame from the PC host application to the Bootloader is shown in Example 8.

EXAMPLE 8: REQUEST

```
[<SOH>…]<SOH><0x04><ADRS_LB><ADRS_HB><ADRS_
UB><ADRS_MB><NUMBYTES_LB>
<NUMBYTES_HB><NUMBYTES_UB><NUMBYTES_MB><CRC
L><CRCH><EOT>
```

The response from the Bootloader to the PC host application is shown in Example 9.

EXAMPLE 9: RESPONSE

```
[<SOH>…]<SOH><0x04><FLASH_CRCL><FLASH_CRCH>
<CRCL><CRCH><EOT>
```

ADRS_LB, ADRS_HB, ADRS_UB and ADRS_MB, as shown in Example 8, represent the 32-bit Flash addresses from where the CRC calculation begins.

NUMBYTES_LB, NUMBYTES_HB, NUMBYTES_UB and NUMBYTES_MB, shown in Example 8, represent the total number of bytes in 32-bit format for which the CRC needs to be calculated.

JUMP TO APPLICATION

The **Jump to Application** command from the PC host application commands the Bootloader to execute the application. The request frame from the PC host application to the Bootloader is shown in Example 10.

EXAMPLE 10: REQUEST

```
[<SOH>…]<SOH><0x05><CRCL><CRCH><EOT>
```

There is no response to this command from the Bootloader because the Bootloader immediately exits Firmware Upgrade mode and begins executing the application.

© 2011 Microchip Technology Inc.