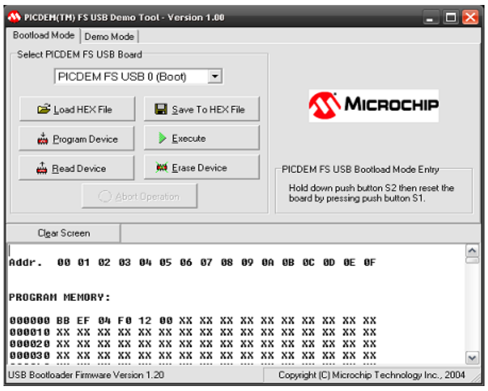
USB Bootloader

Introduction



Hold down push button S2 then reset the board by pressing push button S1.

```
Addr.   00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

PROGRAM MEMORY:

000000 BB EF 04 F0 12 00 XX XX XX XX XX XX XX XX XX XX
000010 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
000020 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
000030 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
```

USB Bootloader Firmware Version 1.20 | Copyright (C) Microchip Technology Inc., 2004

In this article I will show how to use and/or modify Microchip's MCHPUSB Bootloader Firmware in your own projects. The bootloader can be used for the all USB PIC devices (PIC18F4550, PIC18F4455, PIC18F2550, PIC18F2455, PIC18F4553, PIC18F4458, PIC18F2553, PIC18F2458).

The information provided here is based on Microchip-USB-Framework version 2.7.

USB-Bootloader from Microchip

Microchip provides a free USB-Bootloader in their USB framework which is part of Microchip's Applications Library. You can get it here: http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2680&dDocName=en547784.

The bootloader firmware (incl. source code) can be found in the installation directory in the following subdirectory: **USB Device - Bootloaders\Vendor Class - MCHPUSB Bootloader\Bootloader - Firmware for PIC18F4550 Family Devices**.
This bootloader was designed to be used with the PICDEM FS USB DEMONSTRATION BOARD from Microchip (PIC18F4550).

The bootloader starts after Power-On or Reset. The bootloader checks if pin RB4 is low or high:

- RB4 = low (0V): PIC starts in bootloader mode
- RB4 = high (5V): PIC starts user applicatiob

The current status of the bootloader is displayed via 4 LEDs which are connected to RD0-RD3.

How to compile the USB-Bootloader from Microchip

Since the bootloader is designed for the Microchip demo board, the configuration of the bootloader possibly will not fit for your project. E.g. you want to use another pin for bootloader mode entry or you do not want to spend 4 pins/LED's for displaying the current status.

Fortunately, the source code of the bootloader firmware is available. It is written in C language. It can be compiled with Microchip's C18-Compiler, even with the free version of the compiler.

In the directory **USB Device - Bootloaders\Vendor Class - MCHPUSB Bootloader\Bootloader - Firmware for PIC18F4550 Family Devices** you find the project file MCHPUSB.mcp which has to be opened with MPLAB. In the project file the path settings point to the default C18 compiler directory c:\mcc18. If you have installed the compiler in a different directroy, you have to modify the path settings in MPLAB. To do this, select "Project - Build options -

Project" from the MPLAB menu after opening the project file MCHPUSB.mcp. In this dialog go into the "Directories" tab and change the modify the

- Include Search Path
- Library Search Path
- Linker-Script Search Path

according to your C18 installation path.

Now you can build the project via "Project - Build All". If you use the free version of the C18 compiler, the linker will fail since the generated code exceeds the boot block size.

To fit the bootloader into the bootblock, we can stirp unnecessary functions from the source code, like code for showing the status of the bootloader via 4 different LEDs. Here the code parts which can be removed without loosing bootloader functionality:

boot.c

```
...
/** P R I V A T E  P R O T O T Y P E S ***************************************/
---> remove prototype
//void BlinkUSBStatus(void);
...
...
...
void BootService(void)
{
---> remove call to BlinkUSBStatus
//BlinkUSBStatus();
if((usb_device_state < CONFIGURED_STATE)||(UCONbits.SUSPND==1)) return;
...
...
...
case UPDATE_LED:
---> remove the following block
/*
if(dataPacket.led_num == 3)
{
mLED_3 = dataPacket.led_status;
counter = 0x01;
}//end if
if(dataPacket.led_num == 4)
{
mLED_4 = dataPacket.led_status;
counter = 0x01;
}//end if
*/
---> add the following line
counter = 0x01;
break;
...
...
...
---> remove BlinkUSBStatus function
/*
void BlinkUSBStatus(void)
{
//static word led_count=0;  //declared globablly instead, to save code space
if(led_count == 0)led_count = 20000U;
led_count--;
#define mLED_Both_Off()         {mLED_1_Off();mLED_2_Off();}
#define mLED_Both_On()          {mLED_1_On();mLED_2_On();}
#define mLED_Only_1_On()        {mLED_1_On();mLED_2_Off();}
#define mLED_Only_2_On()        {mLED_1_Off();mLED_2_On();}
if(UCONbits.SUSPND == 1)
{
if(led_count==0)
{
mLED_1_Toggle();
mLED_2 = mLED_1;        // Both blink at the same time
}//end if
}
else
{
switch(usb_device_state)
{
case DETACHED_STATE:
mLED_Both_Off();
break;
case ATTACHED_STATE:
mLED_Both_On();
break;
case ADDRESS_STATE:
if(led_count == 0)
{
mLED_1_Toggle();
mLED_2_Off();
}//end if
break;
case POWERED_STATE:
mLED_Only_1_On();
break;
case DEFAULT_STATE:
mLED_Only_2_On();
break;
case CONFIGURED_STATE:
if(led_count==0)
{
mLED_1_Toggle();
mLED_2 = !mLED_1;       // Alternate blink
}//end if
break;
default:                //For POWERED_STATE and DEFAULT_STATE
mLED_Both_On();
break;
}
}//end else of if(UCONbits.SUSPND...)
}//end BlinkUSBStatus
*/
...
```

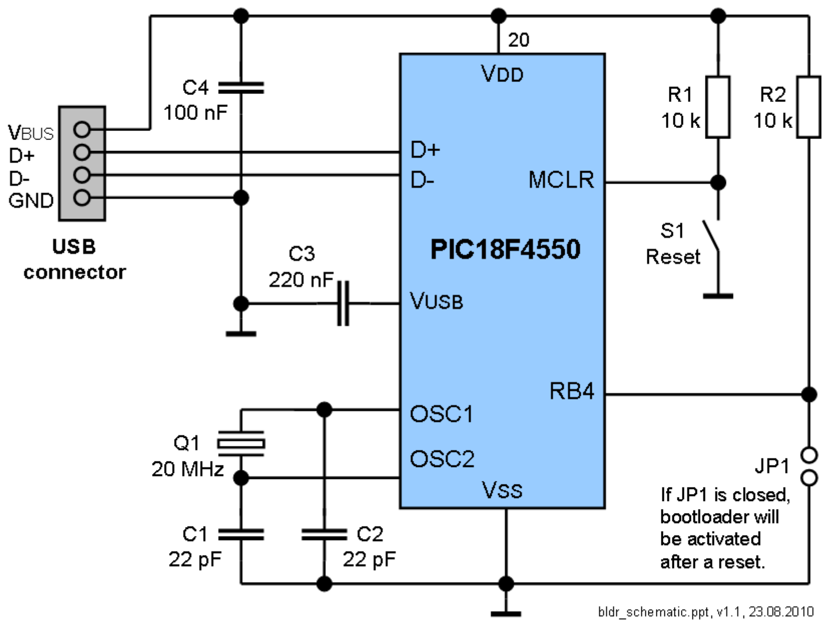
main.c

```
...
// Note: Some of the below configuration bits are commented out
// to prevent build errors with some of the above listed devices.
// For example, on the PIC18F4458 CP3, WRT3, and EBTR3 don't exist.
// adjust PLLDIV configuration to your oscillator frequency
#pragma config PLLDIV   = 5       // (20 MHz input)
...
...
...
// adjust pin to be checked for bootloader mode entry to your hardware
//Check Bootload Mode Entry Condition
---> adjust pin for bootloader entry if it is not RB4 pin for your hardware
if(PORTBbits.RB4 == 1)      // If not pressed, User Mode
{
ADCON1 = temp;          // Restore reset value
_asm goto RM_RESET_VECTOR _endasm
}//end if
//Bootload Mode
---> remove LED init
//mInitAllLEDs();
...
```

Now again try to build the project via "Project - Build All" as described earlier.

Basic USB Bootloader Circuit

Here the basic USB bootloader circuit. JP1 controls the bootloader entry. If JP1 is closed, the PIC will start in bootloader mode. Please note that the status of JP1 is only checked after the PIC has been reset. I.e. to enter the bootloader mode, first JP1 has to be closed. After that the PIC has to be reset.



Flash the USB-Bootloader into the PIC

If all the above steps were successful, you now have your bootloader HEX file MCHPUSB.hex which can be flashed into the PIC. Make sure that the config bits are set properly and match your hardware, e.g. PLLDIV fits to your oscillator frequency.

Now, just load the HEX file into PICPgm and flash it.

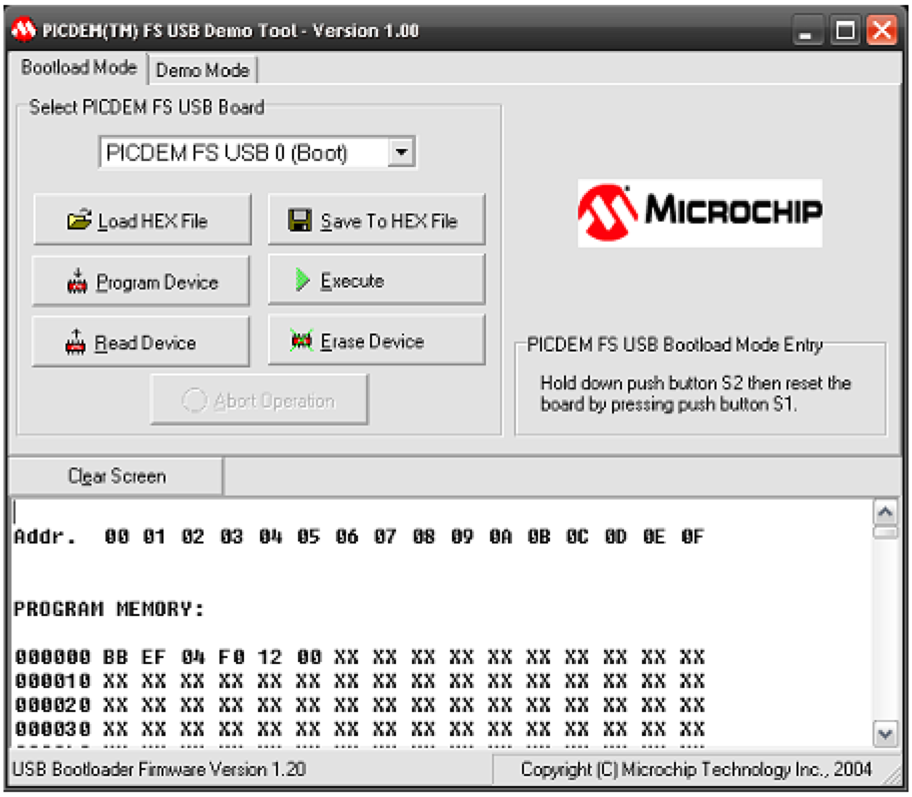
USB driver installation for the bootloader

After successfull programming, connect your device to the PC's USB port (if not already connected). Please make sure that the bootloader entry condition (e.g. RB4 = 0 in our example) is fullfilled!

Now Windows should detect a new USB device. If it askes for the driver, choose the driver location manually and navigate to the driver directory which is located in the Microchip's Applications Library installation directory in the subdirectory **USB Tools\MCHPUSB Custom Driver\MCHPUSB Driver\Release**.

Bootloader PC Software - PDFSUSB.exe

The bootloader PC software PDFSUSB.exe is located in the directory **USB Tools\Pdfsusb**. The tool itself is selfexplaining.

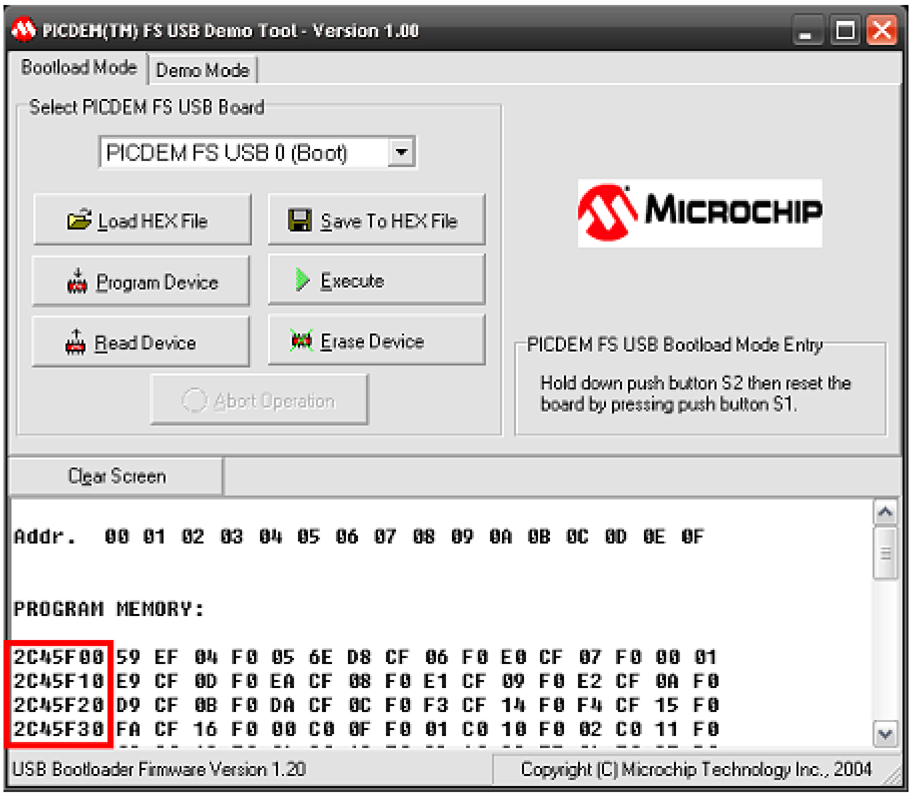

PICDEM(TM) FS USB Demo Tool - Version 1.00
Bootload Mode
Demo Mode
Select PICDEM FS USB Board
PICDEM FS USB 0 (Boot)
Load HEX File
Save To HEX File
Program Device
Execute
Read Device
Erase Device
Abort Operation
PICDEM FS USB Bootload Mode Entry
Hold down push button S2 then reset the board by pressing push button S1.
Clear Screen
USB Bootloader Firmware Version 1.20
Copyright (C) Microchip Technology Inc., 2004

```
Addr.   00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

PROGRAM MEMORY:

000000 BB EF 04 F0 12 00 XX XX XX XX XX XX XX XX XX XX
000010 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
000020 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
000030 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
```

Unfortunately, I had some problems with this tool when trying to load HEX files generated with the CCS compiler. The start address of the firmware shown in PDFSUSB.exe was some ugly address like 0x2C44340 instead of 0x800 which is the firmware start address.


PICDEM(TM) FS USB Demo Tool - Version 1.00
Bootload Mode
Demo Mode
Select PICDEM FS USB Board
PICDEM FS USB 0 (Boot)
Load HEX File
Save To HEX File
Program Device
Execute
Read Device
Erase Device
Abort Operation
PICDEM FS USB Bootload Mode Entry
Hold down push button S2 then reset the board by pressing push button S1.
Clear Screen
USB Bootloader Firmware Version 1.20
Copyright (C) Microchip Technology Inc., 2004

```
Addr.   00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

PROGRAM MEMORY:

2C45F00 59 EF 04 F0 05 6E D8 CF 06 F0 E0 CF 07 F0 00 01
2C45F10 E9 CF 0D F0 EA CF 08 F0 E1 CF 09 F0 E2 CF 0A F0
2C45F20 D9 CF 0B F0 DA CF 0C F0 F3 CF 14 F0 F4 CF 15 F0
2C45F30 FA CF 16 F0 00 C0 0F F0 01 C0 10 F0 02 C0 11 F0
```

After some investigations I found that PDFSUSB.exe has a problem if the following line in the first line in the HEX file is missing (initialization of Upper Linear Base Address):

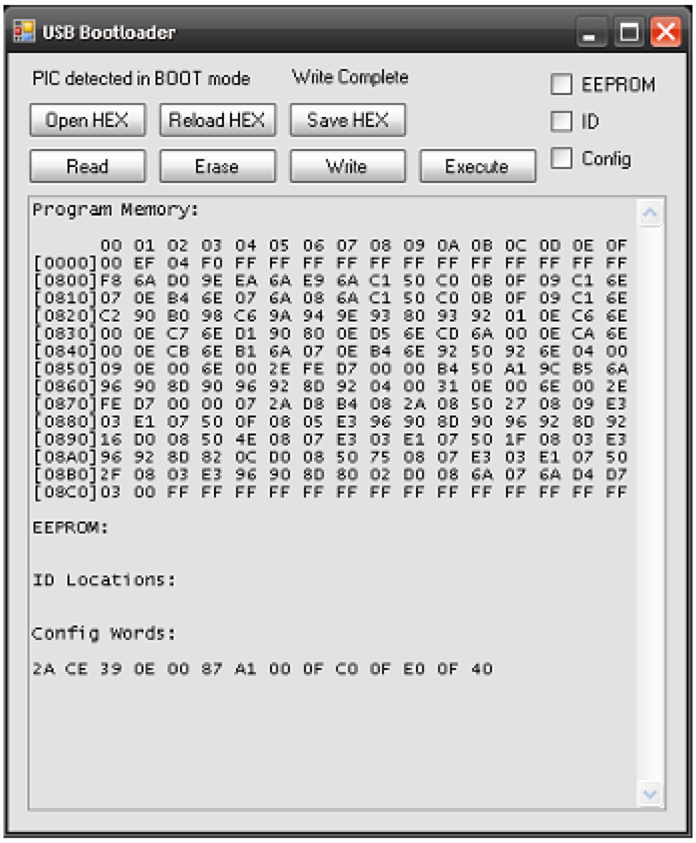
:020000040000FA

I.e. to solve the problem insert **:020000040000FA** before the first line in the HEX file.

Bootloader PC Software - USB Bootloader

But I also found an interesting alternative tool which is simply called USB Bootloader. You can get it here:http://eegeek.net/content/view/27/32/1/3/.

I have also put a copy of the application onto my webspace, so if the above link is not working, you can get the bootloader PC Software here.



```
Program Memory:

       00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
[0000]00 EF 04 F0 FF FF FF FF FF FF FF FF FF FF FF FF
[0800]F8 6A D0 9E EA 6A E9 6A C1 50 C0 0B 0F 09 C1 6E
[0810]07 0E B4 6E 07 6A 08 6A C1 50 C0 0B 0F 09 C1 6E
[0820]C2 90 B0 98 C6 9A 94 9E 93 80 93 92 01 0E C6 6E
[0830]00 0E C7 6E D1 90 80 0E D5 6E CD 6A 00 0E CA 6E
[0840]00 0E CB 6E B1 6A 07 0E B4 6E 92 50 92 6E 04 00
[0850]09 0E 00 6E 00 2E FE D7 00 00 B4 50 A1 9C B5 6A
[0860]96 90 8D 90 96 92 8D 92 04 00 31 0E 00 6E 00 2E
[0870]FE D7 00 00 07 2A D8 B4 08 2A 08 50 27 08 09 E3
[0880]03 E1 07 50 0F 08 05 E3 96 90 8D 90 96 92 8D 92
[0890]16 D0 08 50 4E 08 07 E3 03 E1 07 50 1F 08 03 E3
[08A0]96 92 8D 82 0C D0 08 50 75 08 07 E3 03 E1 07 50
[08B0]2F 08 03 E3 96 90 8D 80 02 D0 08 6A 07 6A D4 D7
[08C0]03 00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF

EEPROM:


ID Locations:


Config Words:

2A CE 39 0E 00 87 A1 00 0F C0 0F E0 0F 40
```

Application Software for usage of the Bootloader

Programs which shall be flashed with the bootloader need to be adapted since the bootloader uses the memory area 0x000 to 0x7FF. This area is usually used by the application software itself. So the following changes are required:

- reserve boot block area (0x000-0x7FF)
- map reset vector from 0x000 to 0x800
- map interrupt vector from 0x008/0x018 to 0x808/0x818

How this can be done is depending on the compiler which is used for generation of the HEX file.

MPLAB C18 compiler

For the MPLAB C18 compiler the following changes need to be done:

```
...
extern void _startup (void);        // See c018i.c in your C18 compiler dir

#pragma code _RESET_INTERRUPT_VECTOR = 0x000800
void _reset (void)
{
_asm goto _startup _endasm
}
#pragma code
#pragma code _HIGH_INTERRUPT_VECTOR = 0x000808
void _high_ISR (void)
```

```
{
;
}
#pragma code _LOW_INTERRUPT_VECTOR = 0x000818
void _low_ISR (void)
{
;
}
/* This pragma forces the code below this line to be put into the code */
/* section (memory address >= 0x82A). See linker script for details. */
#pragma code
...
```

Further, a linker script is required to make the application software start at 0x800. You can get the linker scripts here:usb_bootloader_linkerscripts.zip.

I have written a LED blinking demo application which can be loaded into the PIC with the bootloader. You can download it here: 18f4550_app4bootloader_mplab.zip.

CCS compiler

For the CCS compiler it is a little bit simpler:

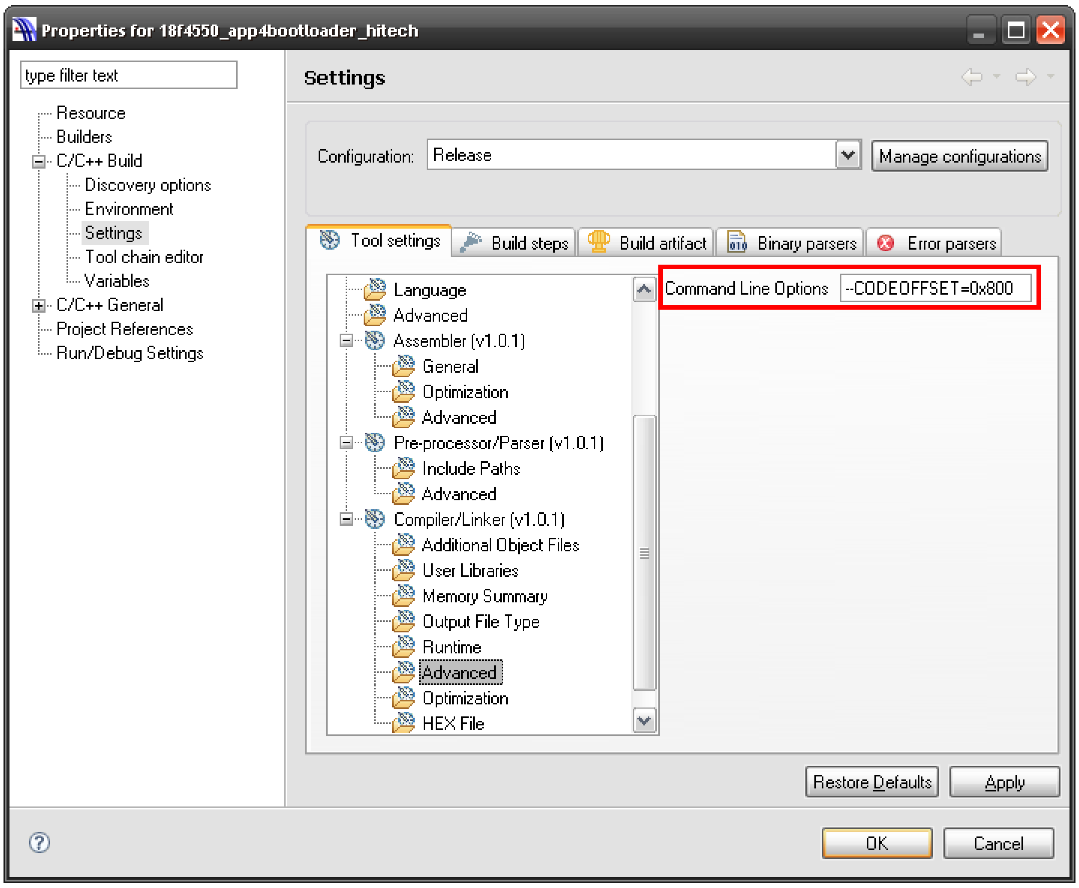
```
...
/* ------------------------------------------------------------------------- */
/* map reset vector and interrupt vector                                     */
/* 0x000-0x7FF is used by the bootloader. The bootloader maps the original   */
/* reset vector (0x000) to 0x800 and the interrupt vector (0x008) to 0x808.  */
/* ------------------------------------------------------------------------- */
#build (reset=0x800, interrupt=0x808)
/* ------------------------------------------------------------------------- */
/* reserve boot block area                                                   */
/* This memory range is used by the bootloader, so the application must not  */
/* use this area.                                                            */
/* ------------------------------------------------------------------------- */
#org 0, 0x7FF {}
...
```

LED blinking demo application: 18f4550_app4bootloader_ccs.zip.

HI-TECH C Compiler

For the HI-TECH C compiler the reset vector can be changed via the compiler option **--CODEOFFSET**. For the MCHPUSB bootloader we have to set the option to **--CODEOFFSET=0x0800**

If you use the HI-TIDE IDE, the option can be specified in the project properties (Project => Properties => C/C++ Build => Settings => Advanced => Command Line Options):



Configuration: Release

Command Line Options --CODEOFFSET=0x800

LED blinking demo application: 18f4550_app4bootloader_hitech.zip.