Bootloader for BaseStation

1. INTRODUCTION

The bootloader for ImageProc2 is used to load and run your application on the target device without MPLAB debuggers such as MPLAB ICD2. The bootloader consists of two applications:

- Target side bootloader application which must be programmed into dsPIC33F program memory prior to bootloader operation. It is written in C.
- Host PC bootloader application which communicates with the target side bootloader. It is written in Python.

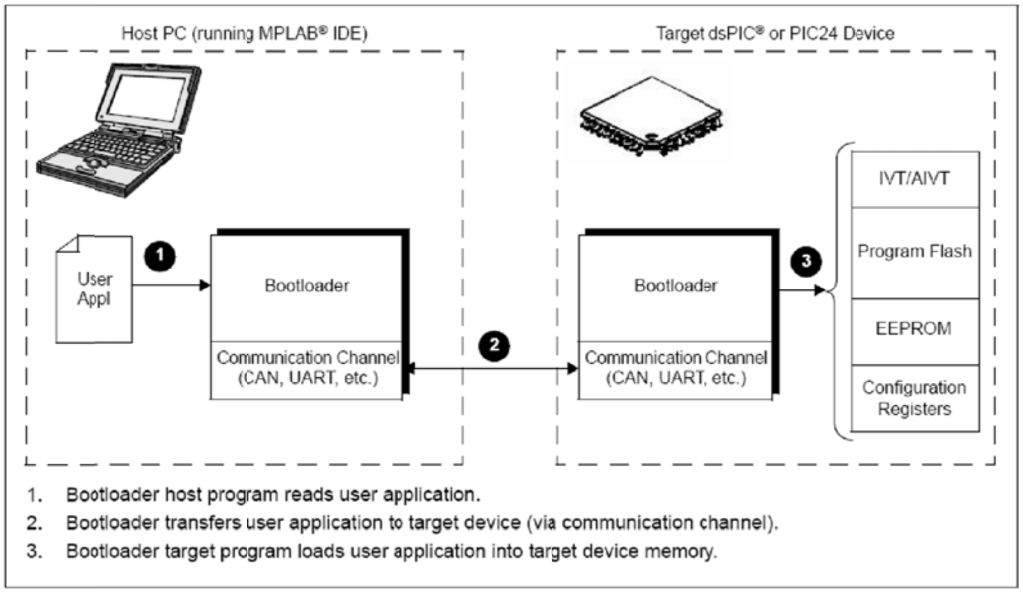
The bootloader parses the program HEX file and then copies it into the appropriate program memory on the target device via USB/UART communication channel. Figure 1 illustrates this process.

2. SYSTEM CONCEPT

The bootloader target application is located in the dedicated program memory region, starting at address 0x1000. On start-up, the bootloader reads program memory address 0x2000, which contai contains a bootloader delay value. If the bootloader fails to detect incoming radio activity within the time period specified by the delay value, it suspends itself and transfers execution to the user application located at program memory address 0x2002. On the other hand, if the bootloader detects incoming radio activity before it suspends itself, it programs program memories with the data it receives from the bootloader host application via serial interface.

The bootloader host application parses the HEX file containing the user application (generated by MPLAB® IDE) and sends this data to the bootloader target application. The bootloader host application also supports additional features, such as read of program memories.

FIGURE 1: BOOTLOADER PROCESS


Host PC (running MPLAB® IDE)
Target dsPIC® or PIC24 Device
User Appl
1
Bootloader
Communication Channel (CAN, UART, etc.)
2
Bootloader
Communication Channel (CAN, UART, etc.)
3
IVT/AIVT
Program Flash
EEPROM
Configuration Registers

1. Bootloader host program reads user application.
2. Bootloader transfers user application to target device (via communication channel).
3. Bootloader target program loads user application into target device memory.

3. DEVICE MEMORY USAGE

Figure 2 illustrates memory organization for dsPIC33F targets. The interrupt tables (IVT/AIVT) use memory space up to address 0x1FE. The bootloader cannot be placed immediately following this address because erasing the first Flash memory page also erases the bootloader. Therefore, the bootloader must start at address 0x400 or after, which leaves a “hole” of unused memory from 0x200 through 0x3FE (or 0xFFE for our case since the bootloader is located in 0x1000 – 0x1FFE). However, this available memory can be used for user applications. Also, because of this Flash memory page restriction, the user application cannot be placed immediately after the bootloader. It must be pushed to the beginning of the next Flash memory page (address 0x2000). Starting at that address, the application specifies the bootloader delay value, followed by the actual application code at address 0x2002.

FIGURE 2: MEMORY ORGANIZATION FOR dsPIC33F

0x0000 – 0x0003	Reset	Page 0: 512 Instructions
0x0004 – 0x01FF	IVT/AIVT	
0x0200 – 0x03FF	Available	
0x0400 – 0x0FFF	Available	Page 1 - 3: 512 * 3 Instructions
0x1000 – 0x1FFF	Bootloader	Page 4 - 7: 512 * 4 Instructions
0x2000	Delay	Page 8 -
0x2002 -	User Application	

4. DEVICE PERIPHERAL USAGE

The target side bootloader application uses program memory from address 0x1000 to 0x2000 (inclusive) on dsPIC33F devices. It also uses Reset vectors from the Interrupt Vector Table (IVT). The target side bootloader application uses these peripherals:

- UART1 or UART2
- Timer 2 & 3

These peripherals can be overridden by user application. The target side bootloader application uses no interrupt.

5. FILES

The bootloader application is organized into two subdirectories:

- Target Side: ...\PICDevel\trunk\bootloader2\target
- Host Side: ...\PICDevel\trunk\bootloader2\host

The target side bootloader application is developed with MPLAB IDE tools and consists of the following files:

- Project Files: ImageProc2.mcp & ImageProc2.mcw
- Main Program (performs all main tasks, such as initialization and communication): main.c & bootloader.c
- Support File (contains memory routines, such as erase and write): memory.s
- Test Application Files (located in the ...\PICDevel\trunk\bootloader2\test directory)

The bootloader host application is developed with Python and consists of the following files:
bootloader2.py, payload.py, and scanf.py

6. BUILDING AND LOADING THE TARGET SIDE BOOTLOADER

Before building and loading the target side bootloader, the bootloader's linker script (.GLD file) must be modified to specify the program address for the bootloader. For dsPIC33F devices, the .GLD file is modified to place the bootloader at address 0x1000 as shown in Table 1.

TABLE 1: GLD EXAMPLE FOR BOOTLOADER

An example of Original GLD file	An example of Modified GLD file

MEMORY
{
data (a!xr) : ORIGIN = 0x800, LENGTH = 0x4000
reset : ORIGIN = 0x0, LENGTH = 0x4
ivt : ORIGIN = 0x4, LENGTH = 0xFC
aivt : ORIGIN = 0x104, LENGTH = 0xFC
program (xr) : ORIGIN = 0x0200, LENGTH = 0x15600
.....
.....
}
 	
...
 	
__DMA_END = 0x47FF;
__CODE_BASE = 0x0200;
	MEMORY
	{
	data (a!xr) : ORIGIN = 0x800, LENGTH = 0x4000
	reset : ORIGIN = 0x0, LENGTH = 0x4
	ivt : ORIGIN = 0x4, LENGTH = 0xFC
	aivt : ORIGIN = 0x104, LENGTH = 0xFC
	program (xr) : ORIGIN = 0x1000, LENGTH = 0x14600
	.....
	.....
	}
	
	...
	
	__DMA_END = 0x47FF;
	__CODE_BASE = 0x1000;

To build and load the target side bootloader:

1. Open the project file: ImageProc2.mcp
2. Connect the target board to the host computer via MPLAB ICD 2.
3. From the **Programmer** menu, choose **Select Tool**, and then click on **ICD2**.
4. From the **Programmer** menu, select **Program**.
5. Reset the target board.

At this point, the bootloader reads the delay value of 0xFF (since Flash was erased by the MPLAB IDE tools), and waits for radio activity.

TABLE 2: VALID DELAY VALUES

Delay (seconds)	Results
0	Suspend bootloader and transfer execution to the user application.
1-254	Wait specified number of seconds for HEX file transfer. If no radio communication is detected before the delay time has expired, suspend bootloader and transfer execution to the user application
255	Wait forever for HEX file transfer.

7. REQUIREMENTS FOR A USER APPLICATION

The following requirements apply to any application intended to be loaded by the bootloader:

- Application cannot place code into memory space reserved by the bootloader.
- Bootloader delay must be specified for subsequent bootloader executions.

To satisfy these requirements, the corresponding user application's linker script (.GLD file) must be modified to specify the application address and designate the bootloader delay period. For dsPIC33F devices, the .GLD file is modified to place the user's application at address 0x2002 and provide a time-out value for the bootloader.

TABLE 3: GLD EXAMPLE

An example of Original GLD file	An example of Modified GLD file
MEMORY	MEMORY
{	{
data (a!xr) : ORIGIN = 0x800, LENGTH = 0x4000	data (a!xr) : ORIGIN = 0x800, LENGTH = 0x4000
reset : ORIGIN = 0x0, LENGTH = 0x4	reset : ORIGIN = 0x0, LENGTH = 0x4
ivt : ORIGIN = 0x4, LENGTH = 0xFC	ivt : ORIGIN = 0x4, LENGTH = 0xFC
aivt : ORIGIN = 0x104, LENGTH = 0xFC	aivt : ORIGIN = 0x104, LENGTH = 0xFC
program (xr) : ORIGIN = 0x0200, LENGTH = 0x15600	program (xr) : ORIGIN = 0x2000, LENGTH = 0x13600
.....	.....
.....	.....
}	}
…	…
__DMA_END = 0x47FF;	__DMA_END = 0x47FF;
__CODE_BASE = 0x0200;	__CODE_BASE = 0x2000;
….	….
.text :	.text :
{	{
*(.init);	SHORT(0x05); /* Bootloader timeout in sec */
*(.user_init);	*(.init);

*(.handle);
*(.libc) *(.libm) *(.libdsp); /* keep together in this order */
(.lib);
} >program
	*(.user_init);
	*(.handle);
	*(.libc) *(.libm) *(.libdsp); /* keep together in this order */
	(.lib);
	} >program

Although configuration bits in user applications are ignored and not written to Flash memory, it is recommended not to add configuration bits in user applications. The following configuration bits are already included in the target device's bootloader.

```
_FOSCSEL(FNOSC_PRIPLL  &  IESO_ON);
_FOSC(POSCMD_EC  &  FCKSM_CSDCMD  &  OSCIOFNC_OFF);
_FWDT(FWDTEN_OFF);  //  Watchdog  Timer  Disabled
```

Configuration bits in user applications are redundant and they may cause serious problems. Clock setup is not necessary since the following lines are already added in the target device.

```
_PLLDIV = 6; // M = 8
_PLLPRE = 0; // N1 = 2
_PLLPOST = 0; // N2 = 2
```

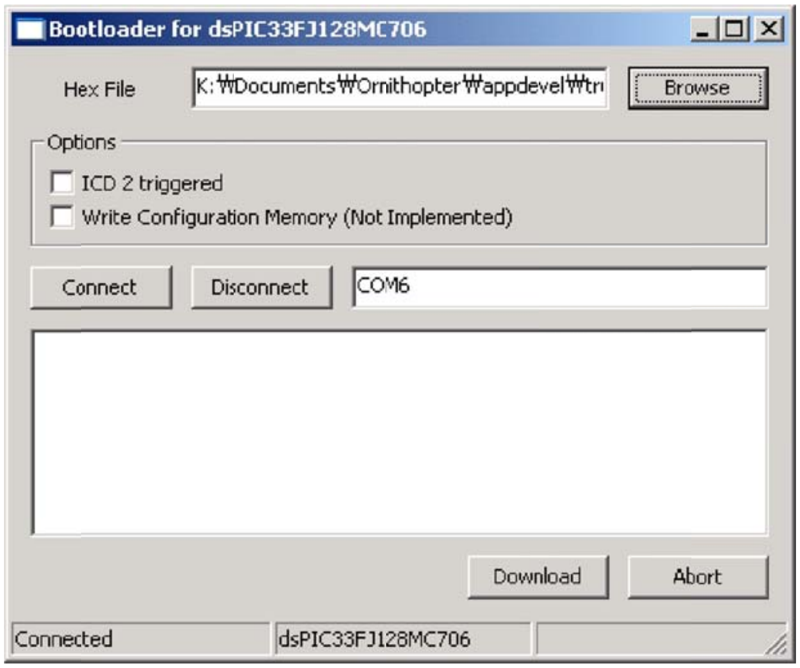
Clock switching is already in the target device. Do not add it in user applications.

8. LOADING USER APPLICATIONS WITH THE BOOTLOADER

Use one of the following methods to load user applications into the target device:
1. Run bootloader.py with arguments (**%run bootloader COM5 230400 ../ImageProc2/test.hex** in iPython).
2. Run bootloader.py with default settings (**%run bootloader default** in iPython). The default values must be set in the bootloader.py file.
3. Run bootloader.py with GUI (**%run bootloader** in iPython). Figure 3 illustrates host PC bootloader. Note:

Here is the procedure to download user applications with bootloader.py with GUI.
1. Run bootloader.py (**%run bootloader** in iPython).
2. Click on **Browse** and select the .hex file you want to download.
3. Power on the target hardware. All three LEDs will blink twice.
4. Click on **Connect** for the communication through wireless between the PC host and the target hardware. If successful, the device name (dsPIC33FJ128MC706) will be displayed on the status bar as shown in Figure 3. If your target MCU is dsPIC33FJ128MC706A, it will still display dsPIC33FJ128MC706.
5. Click on **Download** to start writing user application onto the target device.
6. If loading is successful, the standard output window will display results similar to Figure 4, and the user application will be running.

FIGURE 3: HOST PC BOOTLOADER



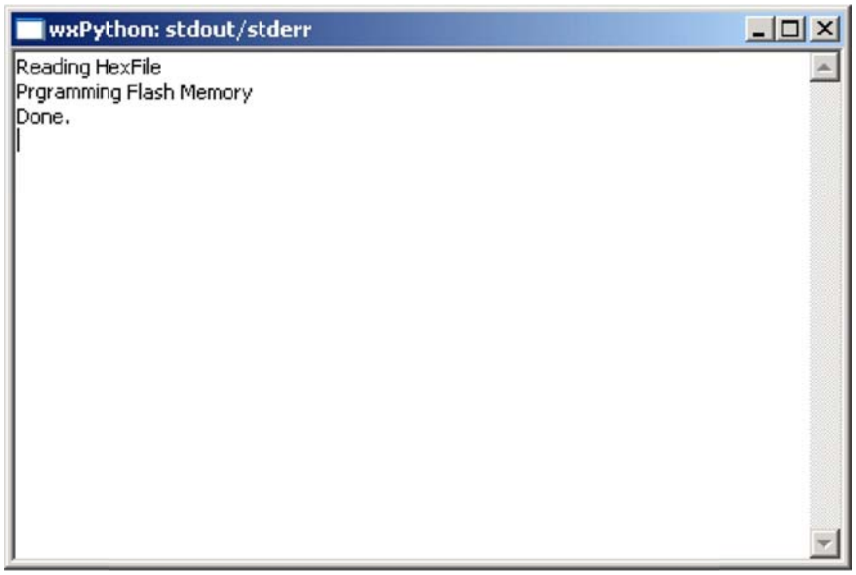
Hex File: K:₩Documents₩Ornithopter₩appdevel₩tr

☐ ICD 2 triggered
☐ Write Configuration Memory (Not Implemented)

COM6

Connected | dsPIC33FJ128MC706

FIGURE 4: STANDARD OUTPUT DISPLAYING RESULTS



```
Reading HexFile
Prgramming Flash Memory
Done.
```

9. RUNNING USER APPLICATION

User application can be started by one of the following two ways:

- Power on the target hardware and wait for the time specified at the address 0x2000.
- Power on the target hardware and send 0x00 to the target device via wireless communication link.

Note: This document is based on AN1094 (DS01094A, Microchip), “Bootloader for dsPIC30F/33F and PIC24F/24H Devices.”