

MSP430FRBoot porting guide

---Gary Gao

This chapter gives a general process how to porting the demo projects to a customized project. The mainly steps as below:

1. Choose a proper demo as template and build the custom app, boot and host's project.
2. Generate linker files based on your device with the linker generator script and add them to the boot and app's project.
3. Change correct communication interface, LED and button control pins in boot project.
4. Add application tasks code in the app project and generate "xxx.txt" file.
5. Convert the "xxx .txt" file to " xxx.c" file with the Perl script and put the" xxx.c" file into the host project.
6. Debug and test.

For this case will use the device MSP430FR2355 to show this porting process clearly. This case scenario: MSP430FR2355 is the boot side and the MSP430FR2433 is the host side and the host will send the LED toggle application code to the boot side by UART (Not use the dual image mode).

(1) First choose the proper demo as template. Due to the MSP430FR2355 and MSP430FR2433 are both MSP430FR2xx4xx family, choose the MSP430FR2433 UART demo code as the template with CCS. Because the host side is also MSP430FR2433, just create boot and app's project. Create two empty projects based on msp430fr2355 named "App1_MSPBoot_FR2355_UART" and "MSPBoot_FR2355_UART". Then cut the linker file "lnk_msp430fr2355.cmd" to a temporary folder because we will use it later (The other same one can be deleted directly).

1) For the App1_MSPBoot_FR2355_UART project, copy files "main.c"," TI_MSPBoot_Mgr_Vectors_FR24xx.c" and "TI_MSPBoot_Mgr_Vectors.h" from the template project to this project.

2) For the MSPBoot_FR2355_UART project, copy folders (include all the files in the folder) "AppMgr","Comm" , "MI" and files "boot.c", "main.c", "TI_MSPBoot_Common.h" and "TI_MSPBoot_Config.h" from the template project to this project. Delete the files named "TI_MSPBoot_AppMgrDualImg.c" in "AppMgr" and "TI_MSPBoot_MI_FRAMDualImg.c" in "MI", because those files are used in dual image mode. The final structure of the projects like figure 1

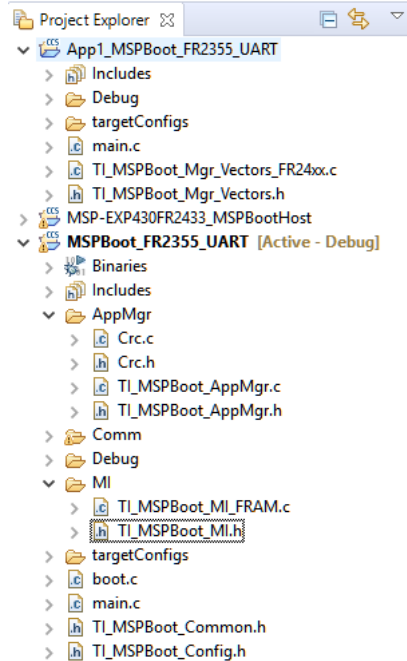


Figure 1 Projects structure in CCS

Then add the include options path "\${PROJECT_LOC}\AppMgr", "\${PROJECT_LOC}\Comm", "\${PROJECT_LOC}\MI", "\${PROJECT_LOC}" and predefined symbols "MSPBoot_20bit", "MSPBoot_BSL", "MSPBoot_CI_UART" like figure 2.

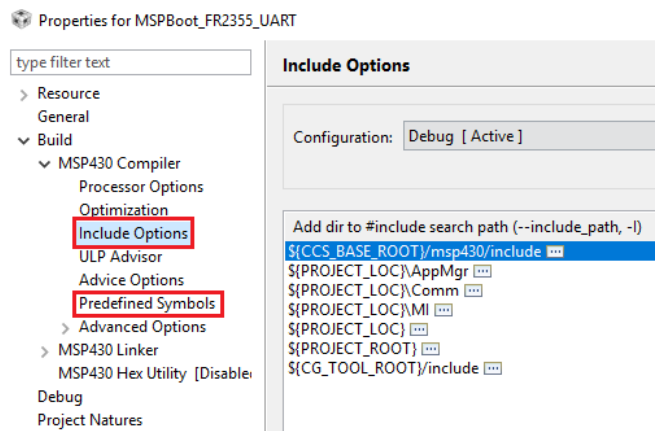
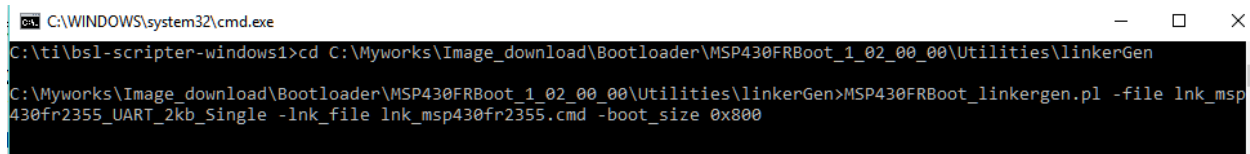


Figure 2 Configurations in properties

3) For the host side project, select BSLBased_16bit_UART (Not use the dual image mode) to be active and comment "sentBSLFlipFlop = !sentBSLFlipFlop;" (We don't need download App2). By the way, we should redefine the static values "CRC_Addr" to 0x8000 and "App_StartAddress" to 0x8003 based on MSP430FR2355's main memory start address.

(2) Generate the linker files for boot and application code by the Perl script named “MSPFRBootLinkerGen.pl”. Copy the linker file “lnk_msp430fr2355.cmd” from the temporary folder to the folder path “..\MSP430FRBoot_XXXXX\Utilities\linkerGen”. Open CMD command window in windows system and change the folder path to “..\MSP430FRBoot_XXXXX\Utilities\linkerGen”. Execute the command “MSP430FRBoot_linkergen.pl -file lnk_msp430fr2355_UART_2kb_Single -lnk_file lnk_msp430fr2355.cmd -boot_size 0x800” like figure 2(For more details about this command meaning you can refer to chapter 4.3.3).



```
C:\WINDOWS\system32\cmd.exe
C:\ti\bs1-scripiter-windows1>cd C:\Myworks\Image_download\Bootloader\MSP430FRBoot_1_02_00_00\Utilities\linkerGen
C:\Myworks\Image_download\Bootloader\MSP430FRBoot_1_02_00_00\Utilities\linkerGen>MSP430FRBoot_linkergen.pl -file lnk_msp430fr2355_UART_2kb_Single -lnk_file lnk_msp430fr2355.cmd -boot_size 0x800
```

Figure 2 CMD command window

The script will generate two CMD files that named “lnk_msp430fr2355_UART_2kb_Single_App” and “lnk_msp430fr2355_UART_2kb_Single_Boot” in the output folder. Then put the two CMD files into the application and the boot project which have no linker files yet.

(3) Change correct communication interface, LED and button control pins in boot project. The modification is based on your hardware. In this case, the hardware is MSP320FR2355 LaunchPad, so the LED control pins are P1.0 and P6.6 and the button control pins are P2.3 and P4.1. For this case, we will use P1.0 and P6.6 for LED control and P2.3 for button control. Modify the GPIO configuration codes in “main.c” and “TI_MSPBoot_Config.h”(The define of “HW_ENTRY_CONDITION”). For the UART interface, we can use the P1.6 and P1.7 and just configure the P1SELx to 01. The UART configuration is in the function “void TI_MSPBoot_CI_PHYDL_Init(t_CI_Callback * CI_Callback);” in file named “TI_MSPBoot_CI_PHYDL_USCI_UART.c”. What’s more, change the definition of “CONFIG_CI_PHYDL_UART_BAUDRATE” to 9600 in “TI_MSPBoot_Config.h”.

(4) Add the LED toggle task application code in the app project (Named “App1_MSPBoot_FR2355_UART”) and generate “xxx.txt” file. The application code is different for different project. For easy to understand, we just use LED toggle task for an example and the hardware is MSP430FR2355 LaunchPad. We can use the P1.0 to toggle LED and P2.3 to be the button control. In the application project, just keep P1.0 to toggle LED and modify the button pin number from P2.7 to P2.3 in the “main.c” file. Due to the MSP430FR2355 doesn’t have Timer_A and we can use the Timer_B in this case. Because MSP430FR2355 just has 32k memory size, it doesn’t need the symbol named “FLASH2” defined in the CMD file named “lnk_msp430fr2355_UART_2kb_Single_App”. Delete all the information about FLASH2 and keep the symbol FLASH. Then configure the project’s properties to generate hex format image file like figure 4.

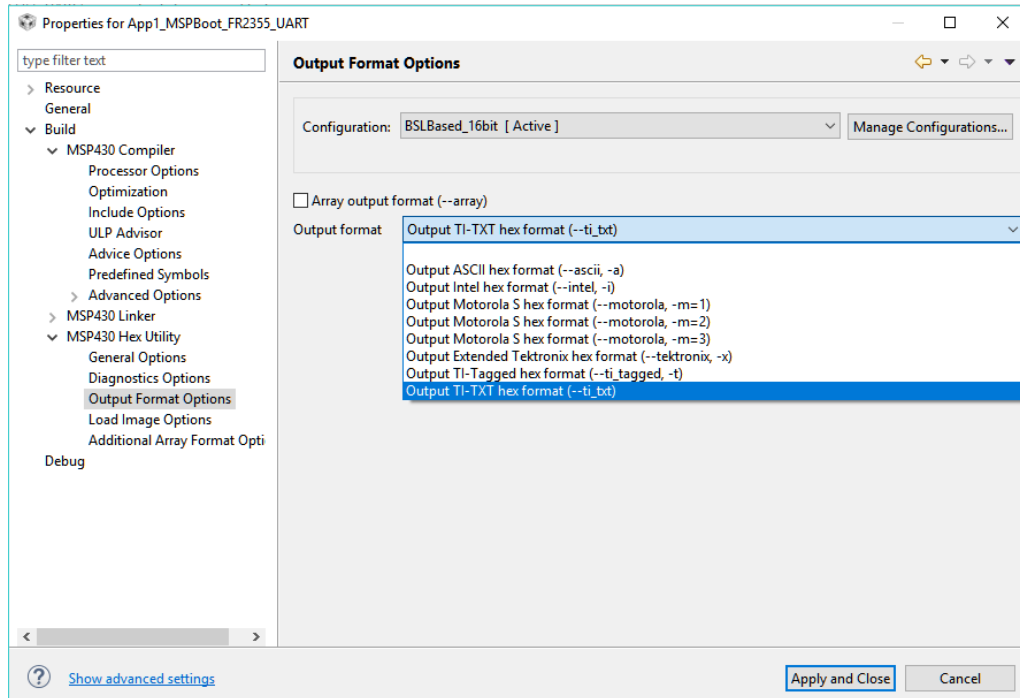


Figure 4 configure the project's properties to generate hex file

(5) Convert the "xxx .txt" file to "xxx.c" file with the Perl script and put the "xxx.c" file into the host project. Build the application project with no errors. Copy the generated .txt file to the folder "..\MSP430FRBoot_XXXXX\Utilities\430txt_converter". Due to MSP430FR2355's memory size is 32k that is larger than 16k, 430txt2C_20bit.pl can be used in this case. Open CMD command window in windows system and change the path to "..\MSP430FRBoot_XXXXX\Utilities\430txt_converter". Execute the command "430txt2C_20bit.pl xxx.txt xxx.c App1" like the example of figure 5

```
E:\works\MSP430\bootloader\MSP430FRBoot_1_02_00_00\Utilities\430txt_converter>430txt2C_20bit.pl App1_MSPBoot_FR2355_UART
.txt App1_FR2355_TEST.c App1

430txt2c.pl V1.2
Texas Instruments Inc 2013

*****File App1_FR2355_TEST.c generated OK! *****
```

Figure 5 Convert the .txt file to .c file example

Put the file "xxx.c" into the folder "..\MSP-EXP430FR2433\TargetApps" and replace the statement "#include "TargetApps\App1_FR2433_UART_BSLBased_16bit_FR2Host.c" with "#include "TargetApps\xxx.c" in file "main.c" of the host project.

(6) Debug and test. Finish the works before and download the code into the hardware to see if them works well.