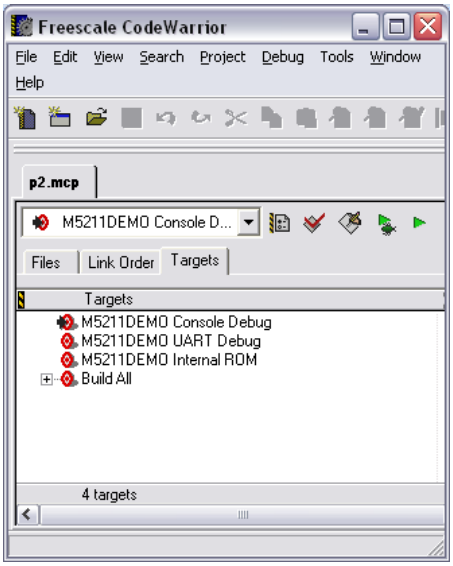
Assignment 2 Program Load and Link, Startup Code, and Memory Allocation – CSE 325, Fall 2010

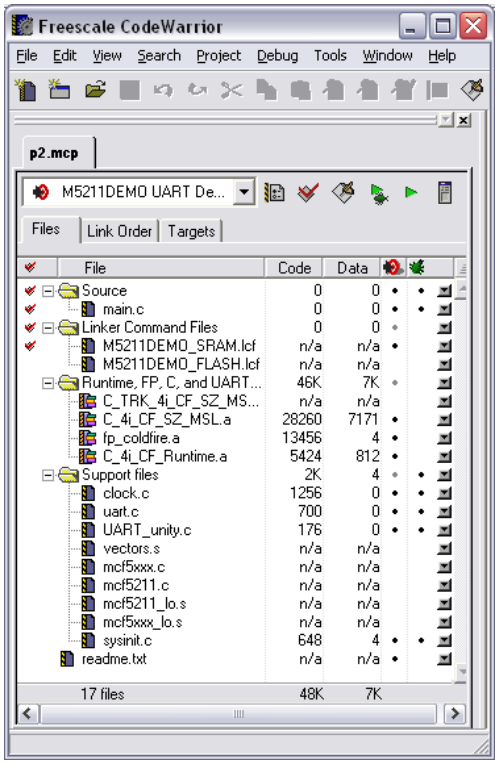
Assignment Objectives

1. To learn the startup code for the M5211DEMO board.
2. To learn how to use Freescale CodeWarrior IDE to build executable code for different memory configurations.
3. To learn target board initialization and application program execution with a romized program in flash memory
4. To utilize routines for clock initialization, delay function, and interrupt configuration.

In many embedded systems, the application software is running on bare hardware. After a hardware reset, the microcontroller will be initialized by system initialization code, and then the application program will begin executing Therefore, the application software and the system initialization code should be linked together as a complete program which can be stored in a non-volatile memory (ROM, EPROM, EEPROM, or flash). Even to run a C program, a system initialization code will occur before main() is called. In this exercise we are going to learn what it means of building a target executable code and how program execution is initiated after a hardware reset.

We will take the LED program (main.c and delay.c) from Assignment 1 to create a new project Assign02. Once the project is created, you can see there are 3 build targets in project window, i.e, M5211DEMO Console Debug, M5211DEMO UART Debug, and M5211DEMO Internal ROM. A project consisting of multiple files (added to the project) and can have multiple makefiles for different build targets.





For instance, when the target “M5211DEMO UART Debug” and the child window “File” are selected (as shown above), we can see all files included in the project and the ones (with a dot in the target column) are included in the build. You can simple click the row position in the target column to remove a file from or add a file into a build.

Here are a set of questions related to the three build targets. To answer the questions, you may need to look into *.xmap, *.elf, *.lcf files, CodeWarrior Development Studio for ColdFire Architectures v6.3 Targeting Manual, and some source code in the support file of the project.

Questions to be answered in your report:

1. In the new project Assign02, you can find 3 build targets: M5211DEMO Console Debug, M5211DEMO UART Debug, and M5211DEMO Internal ROM. List all files included in each build target. Compare the code sizes (the length of text segment) of the build targets and their entry points. The memory allocation information is stored in *.xmap file. Discuss why the code sizes are different.
2. What is the purpose of lcf files? Draw the memory layouts for the build targets of M5211DEMO UART Debug and M5211DEMO Internal ROM.
3. You may use flash programmer tool (such as CFFlasher from Freescale) to program flash memory. Assume that you write executable code of the target “M5211DEMO Internal ROM” to flash memory. After a hardware reset, the system will be initialized and enters an entry of the application software. Give a short description of the routines to be executed after a hardware reset for the target “M5211DEMO Internal ROM”. Note that

your description should include the routines in the application source files, but can skip the API of C_4i_CF_Runtime library, such as *printf* and ___*call_static_initializers*.

4. Add the following interrupt service routine before "void main()".

```
__interrupt__ void int_handler()
{
    kk++;
}
```

Then, add the following two statements before "while(1)" in main.c.

```
mcf5xxx_set_handler(10, int_handler);
asm_set_ipl(5);
```

When you run the program for the build targets M5211DEMO Console Debug, what are the changes made by the two statements? Note that you should resolve all compilation errors that you may encounter after the additions.

5. Continue with the modified main.c in question 4 and add the following statement before "while(1)" in main.c.

```
clock_pll(REF_CLK_KHZ, SYS_CLK_KHZ, 0) / clock_lpd(1);
```

Then, run the program you make for the build target M5211DEMO Console Debug. (Again, you may need to include additional support files to make this change work.) Do you see any differences on the LED display? If yes, explain what causes the difference.

Now, remove delay.c from the project and replace the call "delay_loop(delay1, delay2);" statement in main.c with a call to "cpu_pause(500000);". After including necessary support files, you can re-make and run main.c. Describe what you see on the LED display.

6. Compare the code size of the modified executable from question 5 with the one of the same build target of question 1. You may notice the significant increase of code size. Describe the reason(s) for the increase and suggest modification(s) of the support files to reduce the code size.