Laboratory 3

Creating Device Drivers

Due Date: Beginning of Week 5 Lab Period

Files Needed:
Week3Lab.zip and 22100d.pdf from Blackboard

Concept:

Develop a device driver to be used as a service in an embedded system.

Introduction:

The code in a real-time operating system belongs to one of three categories. We are using MicroC/OS-II for our kernel code. Other than changing some configurations, there is little work do be done by us for the kernel. The application code contains the tasks, and we will be completely responsible for generating that code once we have some more background. In the middle sits services. Some services are provided along with kernel code, like semaphores and mutexes. Other services, like device drivers, may need to be written by the application writer. To explore this process, this lab will involve writing a device driver for an SPI interface on the Dragon12-plus evaluation board.

Assignment:

In the project Week3Lab, there are spi.c and spi.h. You will complete the subroutines in these files to provide the driver interface for the Microchip 23K256 RAM. The data sheet for this device is available in the project folder and on Blackboard, and you will need to use this to complete your code. You will see that two static subroutines have already been written in spi.c to simplify the process, spi0put and spi0get. These should be called from your subroutines (and can only be called from within spi.c due to the "static" modifier) to access the chip. Note that the subroutines do not account for the SPI's chip select (CS) signal. The CS signal going to the 23K256 is Port S pin 7 (Hint: Use the header files to determine what labels have already been defined for that pin.) You must control this pin manually in your subroutines.

Write the five subroutines described below.

- static void spi_RAM_config(char options);
    - The argument sets the status register in the 23K256 to control the read/write options mode and should accept values for page, sequential, or byte mode.
    - This subroutine is called by your other four subroutines within spi.c.

- void spi_RAM_write(int address, char data);
    - The char should be written to the address specified. You may assume the address is valid.

- char spi_RAM_read(int address);
    - The char returned is from the address specified. You may assume the address is valid.
- void spi_RAM_write_string(int address, char* string);
    - Starting at the address given, writes chars in the string until a 0x00 is found, and the 0x00 should be the last char sent to the RAM. (i.e. this sub stores a null-terminated string.)

- void spi_RAM_read_string(int address, char* string);
    - Starting at the address given, writes chars into the string until a 0x00 is found, and the 0x00 should be the last char stored into the string. However, if a 0x00 is not found after 159 chars, the subroutine should stop accessing memory and store 0x00 as the final value. (i.e. returns a null-terminated string up to 160 bytes, including the null character.)

Deliverables:

- Your CodeWarrior project in a zip file.