ELECTRICAL ENGINEERING PROGRAM

California Polytechnic State University

By Brian Mealy 2011, Modified by Bridget Benson 2013

CPE 439 Experiment 4

Learning Objectives

1. FreeRTOS
   - To become more familiar with some basic objects and services in the FreeRTOS kernel including the tick hook function.
   - To become familiar with tasks, semaphores, and timing services and interrupts in an RTOS environment.

2. Serial Peripheral Interface
   - To gain more practice with the SPI bus and an SPI-controlled hardware
   - To write a device driver for an SPI-based 7-segment display

Reference Documents

The following documents are available on the CPE 439 PolyLearn site:

- FreeRTOS Real Time Kernel: A Practical Guide
- FreeRTOS Reference Manual: API Functions and Configuration Options
- Sparkfun Serial 7-Segment Display Datasheet

Introduction and Overview

This experiment is seemingly straight-forward, but you will run into some issues along the way. As with all goodly designed systems, be sure to make sure all data and resources are safely shared among the users of those items. One of the funny things you can do with this experiment is to "unprotect" your resources and watch how relatively quickly the system starts spewing crap.

And for the just to say you did it category, you'll also be using the tick hook function in this experiment. The tick hook is a callback function that is provided as a service of sorts by the FreeRTOS. The tick hook is actually part of the tick interrupt so there are some issues with its usage that you'll want to check out the details in the FreeRTOS manual.

Lastly, the experiment uses a special 7-segment display that has both a SPI and standard serial interface. You'll be using the SPI port so you'll need to hook up a few wires. The part is sold by Sparkfun (www.sparkfun.com) and you can find a datasheet there as well as posted on PolyLearn. Probably the first thing you want to do in this experiment is to write something that proves to you that the 7-segment display device is actually working. I've provided a spi_sseg header file (on PolyLearn) complete with macros and function prototypes. Your mission is to write the associated source file.

Programming Assignment:

Create an RTOS-based application that will:

- Blink LED0 at 5 Hz
  - Count the number of state transitions of LED0 and display that number on the two left-most digits on the 7-segment display
    - The count should automatically roll over at 99

- Blink LED2 at 10 Hz
  - Count the number of state transitions of LED2 and display that number on the two right-most digits on the 7-segment display
    - The count should automatically roll over at 99

- Blink the 7-segment display colon at a rate of approximately 1 Hz using the tick hook callback.

- Write a device driver source file for the provided header file (spi_sseg.h). You can add to the source file where necessary but you can't take stuff away from it. You must implement all the functions appearing in the file. Feel free to add more functions if you need to; try to make helper functions static if at all possible.

Constraints and Comments:

- Unless told otherwise, feel free to do the standard poll on the interrupt flag for the SPI transmission completed interrupt.

- Shared data and more importantly, shared hardware resources, should be properly protected.

- You'll definitely still need to look at the datasheet for the 7-segment device as well as the ATMegaxx datasheet regarding the operation of the SPI peripheral. Be sure to communicate with the 7-segment display using the SPI port.

- You'll thus need to the following line in your FreeRTOSConfig.h file:

```
#define configUSE_TICK_HOOK   1
```

Questions

1. Remove one shared data protection you put in place in your code and see if your program still works correctly. Try to explain why you observe the behavior you observe.

Deliverables

For this experiment, please provide the following.

1. Standard Lab Write Up. Your perfectly commented and structured c and h files. Answers to the questions and individual conclusions.

2. Demonstrate your program to me.