Assigned Date: Wednesday, March 5, 2014
Recommended Due Date: Wednesday, March 26, 2014 by midnight
Points Possible: 135 pts – demo is required on Monday, March 31 or Tuesday, April 1 between 5:00 – 7:00 pm; setup demo time with Daniel I.

You may work with a partner for this project!!!

Background and setup:

Before starting this project I recommend that you familiarize yourself more with FreeRTOS by reading through http://www.freertos.org/. You will also need to download the FreeRTOS kernel from: http://sourceforge.net/projects/freertos/files/latest/download?source=files

Once you have installed FreeRTOS on your computer, you should have access to "Demo", "Source", and "License" folders. You will also find a "readme.txt." file in the parent directory. As indicated in the "readme.txt" file, the easiest way to get familiar with FreeRTOS is to start with a demo application project. You may modify the demo application project to fit the needs of the project described below.

Before getting started with one of the demo projects, I recommend that you delete all files and folders in the "Demo" folder aside from the "Common" and "PIC32MX_MPLAB". I also recommend that you delete all files and folders in the "Source" -> "portable" folder aside from the "MemMang" and "MPLAB" folders. Within the "MPLAB" folder you may delete all folders except for the "PIC32MX" folder.

For convenience you may want to copy the FreeRTOSConfig.h file to the FreeRTOS\Source\include. You may find a version of this file in the demo project. This file may need to be modified for this project. Ensure:

```
#define configCHECK_FOR_STACK_OVERFLOW 0
```

You will also need to add the relevant FreeRTOS files to the MPLAB project workspace (see page 210 of the "Using the FreeRTOS Real Time Kernel - A Practical Guide.pdf"). The source files include:

- FreeRTOSV\Source\list.c
- FreeRTOSV\Source\queue.c
- FreeRTOSV\Source\tasks.c
- FreeRTOSV\Source\timers.c
- FreeRTOSV\portable\MemMang\heap_n.c
- FreeRTOSV\portable\MPLAB\PIC32MX\port.c
- FreeRTOSV\portable\MPLAB\PIC32MX\port_asm.s

The header files include:

- FreeRTOSV\Source\include\FreeRTOS.h
- FreeRTOS\Source\include\FreeRTOSConfig.h

- FreeRTOS\Source\include\list.h
- FreeRTOS\Source\include\queue.h
- FreeRTOS\Source\include\task.h
- FreeRTOS\Source\portable\MPLAB\PIC32MX\portmacro.h
- FreeRTOS\Source\include\portable.h
- FreeRTOS\Source\include\semphr.h

These do not need to be copied locally. However, you must include the appropriate directories for the header files to the build path of MPLAB.

Overview of Project:

You may work with a partner for this project!!!

In this project you are required to port your solution from Project 1 into a single project which runs with FreeRTOS. You are required to create tasks which implement the rigid requirements and deadlines established in that project. However, instead of using the on-board LEDs, you must use the PmodCLS. You may choose any serial interface to communicate with the PmodCLS. Your counter must also count from 1 → 65,535. The counter value must be displayed in decimal and hexadecimal form every second. Lastly, you must poll the buttons every 2 ms. You do not need to resubmit a state diagram for this project, but you must provide an execution pattern diagram for each task you implement.

Software Required:

- 1 copy of FreeRTOS kernel (download from: http://sourceforge.net/projects/freertos/files/latest/download?source=files)
- 1 copy of MPLAB X as usual

Hardware Required:

- 1 - Digilent Cerebot MX4cK Embedded Controller Board
- 1 – USB A -> micro B programming cable
- 1 – Digilent Character LCD w/ Serial Interface Module (PmodCLS)
- 1 – 6-pin cable connector w/ 6-pin header

Submission via ANGEL (https://lms.wsu.edu):

A .zip file, named *your_last_names_project4.zip*, with the following content:

- Your well commented C source code implementing the application described above

- All MPLAB project files – you do not need to include all of the FreeRTOS files, except for the FreeRTOSConfig.h file and any that you may have modified

Grading- 135 pts Total:

- 100 pts for well commented and correct C source code satisfying the project requirements
  - 10 pts for configuring serial interface with PmodCLS
  - 20 pts for displaying the counter value in decimal and hexadecimal every 1 second on PmodCLS
  - 20 pts for polling the buttons every 2 ms
  - 10 pts for correct up counting
  - 10 pts for correct down counting
  - 10 pts for stopping the counter
  - 10 pts for clearing the counter
  - 10 pts for appropriate task design
- 10 pts for execution pattern diagram(s)
- 25 pts for demo
- Bonus?