CS 431 Lab #10
Introduction to μC/OS-II Embedded Operating System

Spring 2015

Demonstration due the week of Demonstration due the week of April 29, 2015, 2015.

1 Overview

In this lab you will expand on your lab #9 solution for PID control by re-implementing it on top of the μC/OS-II real-time embedded operating system. RTOS solutions offer many advantages in terms of task scheduling, synchronization primitives and inter-task communication. Your solution for this lab should be composed of three tasks: 1) A dislay task which updates the LEDs and LCD every second, 2) a PID control task which executes every 50ms, and 3) a touch screen sampling and filtering task which executes every 10ms. Points for this lab will be awarded as follows:

1. Succesful compilation of the the μC/OS-II operating system and implementation of the display task (40 points). The display task must perform the following functions:
   - Clearly display group number on the top line of the LCD.
   - Clearly display the seconds since last reset on the second line of the LCD.
   - Update the LED state to rotate from left to right (matching demo program).
2. Implement the PID controller task (30 points):
   - Add a PID controller task which reads the ball position, performs calculations and adjusts servo positions accordingly in order to move the ball in a circle. This task should execute every 50 milliseconds.
   - Update the display task to output the ball's X and Y position values to the LCD.
3. Implement the touch screen task (30 points):
   - Read the touch screen for the selected dimension.
   - Filter the touch screen readings using a Butterworth filter.
   - Change the selected dimension.

2 Procedure

1. Before getting started, review the μC/OS-II page on Wikipedia. Additional information can also be found on the official Micrium website at http://www.micrium.com/page/products/rtos/os-ii.

2. A demo program that demonstrates the features that you need to implement is provided in compiled form on the course website.

3. Download the *lab10.zip* file from the course website and extract to a working folder. Verify that you can compile and download the source code to the FLEX board.

4. IDEs such as MPLAB support task lists. Be sure to have the Task List open because the template project has many TODOs. See https://microchip.wikidot.com/mplabx:tasks-list for more information.

5. Update *app.c* such that it fulfills the requirements for the display task as outlined.

6. Update *app.c* to create the touch screen sampling and filtering task as outlined.

7. Update *app.c* to create the PID controller task as outlined.

8. Experiment with your program to find some "good" values for K_p , K_d , and K_i.

3 Notes

- Functions for turning on, turning off and toggling LEDs have been provided for you. These functions are *LED_On(ledId), LED_Off(ledId)* and *LED_Toggle(ledId)* respectively. Valid values for *ledId* are 1 through 5 corresponding to the LED to act upon. To control all LEDs at once, zero can be passed as a valid input.

- Functions for writing to the LCD display have been provided for you. These functions are *DispClrScr()* and *DispStr(...)* for clearing the screen and writing a line of text. Function prototypes can be found in *lcd.h*.

- Pay attention to comments in the *app.c* file. They can help guide you on where to add additional logic.

- Timer #4 is used by the operating system so be sure not to use it in your code.

4 Questions to Ponder

1. The MPLAB IDE contains a built-in "RTOS Viewer" utility that can be used to examine the state of the RTOS during execution. Launch the "RTOS Viewer" utility from the MPLAB IDE Tools menu. What tasks do you see running?

2. The μC/OS-II operating system provides semaphores for task synchronization and enforcement of critical sections within the code. Are these needed to synchronize access to global PID values between the control and display tasks? Why or why not?

3. Disregarding the need for a 10ms delay between switching touch screen dimensions, is there any advantage of the touch screen task running one dimension every 10ms instead of both dimensions every 20ms?