**Project 3: Kernel and Threads**

**Names: Abbi Fair and Steven Demers**

**Date: 7/29/15**

**Instructor: Dr. Parikh**

**Course: EGR 424**

**Scheduler.c**

void yield (void)

This function makes an svc call in assembly in order to change the privilege level.

void scheduler(void)

This function handles the scheduling of all threads by saving the state of threads, rotating through threads 0-3 and then restoring the state of the threads. The saving and restoring of the threads is handled by the threads.s file.

**Proj3.c**

void initializeSysTickTimer(void)

This function specifies the appropriate values for the SysTickTimer to generate a 1 millisecond interrupt. This is used to control the pre-emptive scheduling. If a thread is taking longer than 1 millisecond, it is interrupted and the next thread is called.

void initializeOLED(void)

This function initializes the OLED display and writes “Project 3” to the top of the screen temporarily.

void initializeLED(void)

This function initializes the GPIO port that controls the LED.

void initializeUART(void)

This function initializes the UART peripheral so that the threads can display through serial communication using the iprintf function.

void main(void)

This function initializes the clock and all the peripherals used by the threads. It then initializes the threads themselves, along with the SysTickTimer and enables interrupts. The yield function is called in order to give control of the program to the scheduler.

**Threads.s**

void createThread(int \*state, char \*stack)

This function is called by the initializeThreads function in scheduler.c after the necessary memory is allocated for the thread. It saves the current values of registers r4-r11 in the stack and then uses those registers to manipulate values stored into the state array. The state array will now contain default values for r0-r3 and r12-r15. When finished, this function will return to the threadStarter function.

void saveThreadState(int \*state)

Saves values from the stack into the state array. This function is called for every thread that is invoked by the scheduler function just in case it is interrupted by the next thread.

void restoreThreadState(int \*state)

Loads values from the stack and places them back into the state array. Then fakes a return so that thread mode and the process stack can be used.

**Context Switch Timing**

The context switch timing was measured by configuring GPIO Port F Pin 1 to be an output pin. Within the scheduler function, the pin was brought low and then high prior to any other operations. The pin was once again brought low prior to the call of the restoreThreadState function. An oscilloscope was hooked up to GPIO Port F Pin 1 and GND in order to measure the context switch time. The results are displayed in Figure 1 below. The time was measured using the vertical cursors, the context switch time was found to be 6.8 microseconds and this value can be seen in the screenshot taken from the oscilloscope.

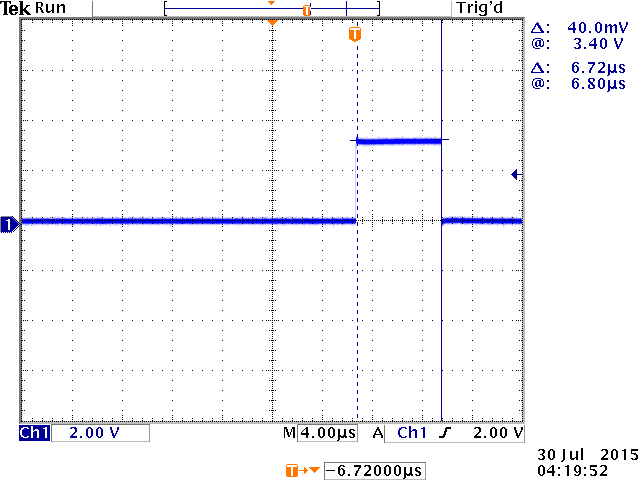


Figure 1. Context Switch Timing Measurement from Oscilloscope