Department of Engineering

EE 4710 Lab 2

Title: Clock-driven and priority driven scheduling

Objective: The student should become acquainted with clock and priority driven systems, and use a real-time operating systems to successfully schedule a set of tasks.

Parts: 1-C8051FX20-TB Evaluation Board
1-USB Debug Adapter

Software: Silicon Laboratories IDE version 3.50.00 or greater, Salvo RTOS.

Preparation: Write the title and a short description of this lab in your lab book. Make sure the page is numbered and make an entry in the table of contents for this lab.

Create a new project using the Silicon Labs IDE as you did in Lab 1 (include mem.c, etc. but this time include library sfc51sdaa.lib). Download the lab2 test file from the course website and store it in a file called lab_test.c. Add that file to your project along with a new file that you create called main.c. Change salvocfg to the following:

```
#define OSUSE_LIBRARY           TRUE
#define OSLIBRARY_TYPE          OSF // use the free library
#define OSLIBRARY_GLOBALS       OSD // globals in the data segment
#define OSLIBRARY_CONFIG        OSA // with delays and events
#define OSLIBRARY_VARIANT       OSA // OS functions called from anywhere
#define OSEVENTS                3   // 1 events
#define OSEVENT_FLAGS           0   // 1 flags
#define OSMESSAGE_QUEUES        0   // 0 message queues
#define OSTASKS                 3   // 2 tasks
```

lab2_test.c contains code, that once initialized continually releases three tasks. Time is broken down into (approximately) 1ms slices, and the tasks have the following properties (all times are number of time slices.

Task	Period	Relative Deadline	Execution Time
A	3	3	1.5
B	4	2	1.5
C	12	12	1

All tasks are initially released at time 0.

Part 1: Clock Driven Schedule

You are to write a function, main() that initially calls OSInit() then BoardInit() (from lab2_test.c) then enters a loop to perform jobs. The loop should start with a call to WaitForNextTimeSlice(), followed by at most two calls to DoJobX() (where X is A, B or C, depending upon which tasks you schedule.) Repeat calls to WaitForNextTimeSlice() and DoJobX() until you have the whole hyperperiod scheduled.

Note that DoJobX() is a function that consumes one half of a time slice, and that it needs to be called enough times before the deadline of task X to satisfy its execution time. (i.e. DoJobC must be called 2 times before 12 time slices have elapsed.)

Part 2. Priority Driven Schedule

Determine the static priority assignments necessary for these three tasks so that a priority driven scheduler will produce a feasible schedule.

Procedure: Compile, link, download and run your clock-driven scheduling code. If it is working properly, the green LED on the board will light and remain lit. Demonstrate this to your lab instructor.

Now, write a new main.c that, after calling OSInit and BoardInit() creates 3 tasks with the priorities you determined earlier. Then, for each task, write code similar to:

```
_OSLabel(TaskLabelA)

void TaskA( void )
{
  static int i;
  for (;;)
  {
    WAIT_FOR_RELEASE_A();
    for (i=0; i<3; ++i )
    {  DoJobA();
       OS_Yield(TaskLabelA);
    }
  }
}
```

Compile, link, download and run your clock-driven scheduling code. If it is working properly, the green LED on the board will light and remain lit. Demonstrate this to your lab instructor.

Affix all your source code to your lab book then write a summary or conclusion. Remember to sign or initial then date each page.