Department of Engineering

EE 4710 Lab 4

Title: Time Delays

Objective: The student should become acquainted with the concept of time delays in real-time operating systems and how they can be used to implement periodic tasks

Parts: 1-C8051FX20-TB Evaluation Board
1-USB Debug Adapter
1-DB-9 Serial cable (USB adapter cable is also ok)

Software: Silicon Laboratories IDE version 3.50.00 or greater. Keil compiler.

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.

Modify the beginning of the timer interrupt routine from Lab 3 so that when timer2 overflows, the t_delay member of each task descriptor is decremented (unless it is already 0). You may use the code below as a guide.

```
timer2_int:
     push psw
     push acc
     jnb  tf2,no_timer_tick
     ; decrement each task's delay counter
     mov  dptr,#taskdesc
 decrement_loop:
     movx a,@dptr     ; most significant byte (big endian)
     inc  dptr
     jnz  dec_timer   ; MSB nonzero, decrement for sure
     movx a,@dptr
     jz   decrement_done
 dec_timer:
     movx a,@dptr
     add  a,#-1       ; dec LSB, clear CY if MSB should dec
     movx @dptr,a
     jc   decrement_done
     dec  dpl         ; go back and decrement MSB
     movx a,@dptr
     dec  a
     movx @dptr,a
     inc  dptr
 decrement_done:
     mov  a,dpl
     add  a,#taskdesc_size-1
```

```
    mov  dpl,a
    cjne a,#low(taskdesc+num_taskdesc*taskdesc_size),decrement_loop
no_timer_tick:
    anl  T2CON, #3FH ; clear interrupt flags
```

Add the sleep function, below, to your C code from lab 3. This function puts the current task to sleep until the specified number of ticks have elapsed.

```
void sleep(unsigned int delay)
{
  EA = 0;
  current_task->t_delay = delay;
  yield();
  EA = 1;
}
```

Modify the scheduler for this exercise. If task 1 or 2 has t_delay = 0, you should schedule one of those. If not, you should schedule task 0 (the background task that runs aperiodic tasks when it can).

Study the sleep function, your scheduler and the interrupt service routine until you understand them, then explain how it all works in your lab book. Particularly explain why it is necessary to disable and enable interrupts in the sleep() function. (There are 2 reasons.)

Write code to test your sleep function by removing the cyclic scheduler and adding two other tasks. The first should alternately send the strings “no you can’t” and “yes I can” to the serial port (or any other two phrases, as long as they are funny). Messages should be sent every 500ms (50 clock ticks). The other should blink the light once every 2 seconds. (Alternatively, you may wire up your prototype board and do something interesting with it instead.) The code below may be helpful.

```
void put_string(char code *s)
{
  while ( *s )
  {
    TI0 = 0;
    SBUF0 = *s;
    while ( !TI0 ) yield();
     s++;
  }
}
```

Procedure: Use a DB-9 serial cable to connect your 8051 board to a computer running a terminal emulator such as puTTY. Configure the terminal emulator for 8 data bits, 1 stop bit, no parity. Compile, link, download

and run your code. Verify that messages alternate on the terminal emulator at a rate of 2 per second, and that the LED blinks once every 2 seconds (or whatever interesting thing you planned for your prototype board happens). It should still be that whenever a ‘0’ is pressed on the keyboard, the LED goes off and that whenever ‘1’ is pressed the LED goes on. 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.