EE 3171 HW4

Due: 27 November 2017

No code for this assignment needs to be tested unless you feel it is necessary.

1. Use the Tiva Reference Manual to answer the following questions:
   (a) (2) What GPIO pin does `M1PWM1` use?
   Pin `PD1`
   (b) (2) What PWM module/generator controls `M0PWM5`?
   Module 0, PWM Generator 2
   (c) (2) What PWM signal uses pin `PF1`?
   `M1PWM5`

2. (10) Write all of the code (the GPIO configuration as well as the PWM configuration) to create a 100 kHz PWM signal with a 45% duty cycle on `M0PWM4`.

Listing 1: PWM Configuration

```
SysCtlClockSet(SYSCTL_SYSDIV_1 | SYSCTL_USE_OSC |
               SYSCTL_XTAL_16MHZ | SYSCTL_OSC_MAIN);

SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);
SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM0);

SysCtlDelay(3);

uint32_t ulPeriod = SysCtlClockGet() / 100000;

GPIOPinConfigure(GPIO_PE4_M0PWM4);
GPIOPinTypePWM(GPIO_PORTE_BASE, GPIO_PIN_4);

SysCtlPWMClockSet(SYSCTL_PWMDIV_1);

PWMGenConfigure(PWM0_BASE, PWM_GEN_2, PWM_GEN_MODE_DOWN | PWM_GEN_MODE_NO_SYNC);
PWMGenPeriodSet(PWM0_BASE, PWM_GEN_2, ulPeriod);
PWMPulseWidthSet(PWM0_BASE, PWM_OUT_4, ulPeriod*0.45);
PWMGenEnable(PWM0_BASE, PWM_GEN_2);
PWMOutputState(PWM0_BASE, PWM_OUT_4_BIT, true);
```

3. (3) Describe the three basic operations that a microcontroller timer performs.
   Timestamping inputs, scheduling events for the future, counting external events.

4. (5) Based on your answer to the question above, describe which kind (or kinds) of timer would be used in each application described below.

   (a) Turning the magnetron on and off when reheating food in a microwave
   Scheduling events for the future

   (b) Measuring an individual's heart rate
   Timestamping inputs *and* counting external events.

   (c) Calculating how long it takes to swing a baseball bat
   Timestamping inputs

   (d) Blink an LED
   Scheduling events for the future

   (e) Deciding when to pulse the anti-lock brakes in a vehicle *and* the actual pulsing of the brakes
   Timestamping inputs and scheduling events for the future

5. (5) Exercise 10.3a
   There are 1,600,000 ticks of the clock in 100 ms. The value in countdown mode should be 1,600,000 or `0x186A00`. If we are to be completely pedantic, the value in count *up* mode should be 1,599,999 (or `0x1869FF`, as a counter value of 0 is worth 1/16000000 s.

6. (4) Exercise 10.4 (Describe how the timer would be configured.)
   There are a variety of ways to get this behavior, but if we're again being really pedantic and want one single interrupt on the 5th interrupt, we should configure the timer in event counting mode and put the value of 5 in the `MATCH` register. If we then enable match interrupts, we should get a single interrupt on the 5th event. Another, albeit sketchy, method would be to put the timer in event counting mode, then keep polling the timer register until it contains 4. Then, and only then, we enable interrupts for external events. That means the interrupt occurs only on the 5th event.

7. (20) Exercise 10.7

Listing 2: Pedestrian Signal Program

```
bool interruptFired = false;
int main()
{
 // Configure (but do not start) the timer
 uint32_t clockFreq = SysCtlClockGet();
 SysCtlPeripheralEnable(SYSCTL_PERIPH_WTIMER0);
 TimerConfigure(WTIMER0_BASE, TIMER_CFG_ONE_SHOT);
 TimerLoadSet(WTIMER0_BASE, TIMER_A, clockFreq * 15);
 TimerIntEnable(WTIMER0_BASE, TIMER_TIMA_TIMEOUT);
 IntEnable(INT_WTIMER0A);
 IntMasterEnable();

 while (1)
 {
  // Poll for the button
```

```
    while (0x08 != (GPIO_PORTB_DATA_R & 0x80)) ;

    // Switch the signals
    GPIO_PORTB_DATA_R ^= 0x30;

   // Start the timer
   TimerEnable(WTIMER0_BASE, TIMER_A);

   // Wait for the interrupt
   while (!interruptFired) ;
   interruptFired = false;

   // Loop back to the top
   }
}
// Here's the ISR
void TimerISR(void)
{
   // Clear the interrupt
   TimerIntClear(WTIMER0_BASE, TIMER_TIMA_TIMEOUT);

   // Switch the signals
   GPIO_PORTB_DATA_R ^= 0x30;

   // Set the flag for the main program
   interruptFired = true;

   // All done.
   return;
}
```

8. (5) Exercise 10.13
   2^{32} = 4294967296. Divide that by 1M and get 4294.967296 s.