CprE 288 – Homework Question Set 4

Question 1: ADC Successive Approximation

A 4-bit ADC (with M = 16 steps) uses the Successive Approximation implementation. The input voltage range of the ADC is 0 V – 20 V. If the input is 14 V, how does the ADC compute each bit of the digital encoding? Fill in the following table to show the steps (refer to examples in slides and the VYES book chapter on ADC). The first step is given.

Step	Output	DN_Mid	AV_Mid(V)	Input >= AV_Mid?
0	xxxx	1000	10	1 (yes)
1	1xxx			
2				
3				
4				

The final digital value is (in binary and decimal): ______________________________

Question 2: GPTM Timers and Input Capture

Timer 1B is being used to measure the time between events on an input. It is configured to count up and detect falling edges. It uses the system clock of 16 MHz.

a. Consider the following input signal connected to a Timer 1B CCP input pin. Event E1 is the first falling edge. Event E2 is the second falling edge.

Suppose the 16-bit timer count values are: E1 = 15,080 and E2 = 56,250. What is the elapsed time in seconds between events E1 and E2?

Answer: _______________

b. Write a line of code that configures the timer (Timer 1B) to detect falling edges.

c. Assume that capture mode event interrupts have been enabled using GPTM timer interrupt registers for Timer 1B. Event interrupts also need to be enabled in the interrupt controller. Determine **j** and **m** (see the register names) for the names of the NVIC enable and priority registers to be configured.

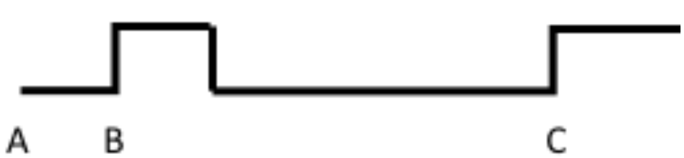
```
NVIC_ENj_R ... //assume appropriate assignment

NVIC_PRIm_R ... //assume appropriate assignment
```

What number is j? ________

What number is m? ________

Question 3: GPTM Timers and Input Capture

a. Consider the following input signal connected to the TM4C123 microcontroller Timer 3 input capture hardware via a GPIO pin. The TM4C123 system clock is 16 MHz.



At point A, let Timer 3 = 0. At event B, let Timer 3 = 500. Note that these are the count values in the timer register in decimal. It is also given that the time period between events B and C rising edges is 2.5 ms.

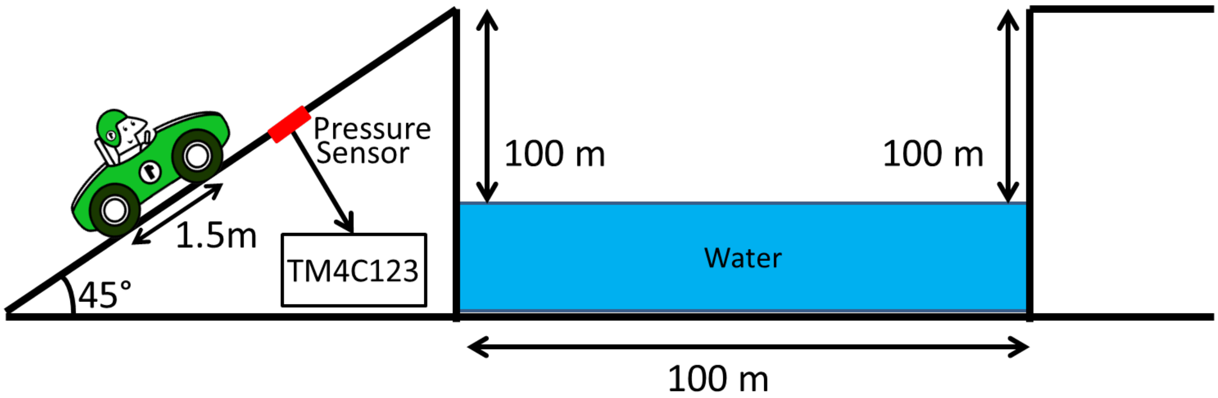
Based on known information and the figure, what will the value of Timer 3 be when event C occurs, i.e., the second rising-edge event? Write your answer in hex. Show your work.

b. Suppose Timer 2B is being used in edge-time mode with an interrupt priority of 3. It uses IRQ (Interrupt ReQuest) number 24. Complete the code to initialize interrupts for the timer.

```
Config_Timer2()
{
  // Assume the associated GPIO module has been initialized.
  // Assume Timer 2B has been configured for 16-bit, count up,
  // edge-time, capture mode to detect positive edges.

  // Set up Timer 2 interrupts
  // Clear event capture interrupt status

    TIMER2_ICR_R |= ____________________________;

  // Enable event capture interrupts

    TIMER2_IMR_R |= ____________________________;

  // Set up NVIC for Timer 2B interrupts
  // Put the correct PRI and EN register numbers in the blank.

    NVIC_PRI_____R = ____________________________;


    NVIC_EN_____R |= ____________________________;

 // Assume initialization finishes by setting up the interrupt
 // handler, re-enabling timer, and globally enabling interrupts
}
```

Question 4: Timer Input Capture Programming

A pressure sensor is embedded in the ramp below.



When pressure is applied to the sensor, a logic 1 is driven on the Timer 1A Input Capture input pin of the TM4C123, else a logic 0 is driven. You are to write a C program that will control the car's brakes to stop the car if its velocity is not fast enough to jump the gap shown. This program uses the Timer 1A Input Capture interrupt.

Assumptions:
1) The car is moving at a constant velocity while on the ramp.
2) The car is treated as a "point mass" for computing the physics of the problem. This results in a minimum velocity of about 31.2 m/s needed to jump the gap. (You may want to try to calculate this on your own just for fun. These resources might help: https://www.khanacademy.org/science/physics/two-dimensional-motion/two-dimensional-projectile-mot/a/what-are-velocity-components , http://www.physicsclassroom.com/class/vectors/Lesson-2/Initial-Velocity-Components .)

a. Initialize Timer 1A as a 24-bit timer, to detect positive (rising) edge events, and with the input capture interrupt enabled.

i. Select an appropriate TM4C GPIO pin to use for the input pulse from the pressure sensor. What GPIO port and pin did you select? Hint: Input capture pins are denoted by TnCCPm labels in the Alternate Function list in Table 23-5 in the datasheet, where n=1 for Timer 1 and m=0 for timer A.

ii. Briefly describe the initialization tasks at a high level (at a higher level than C code or comments). What features need to be initialized and for what purpose? Do not provide specific register macros, bitwise operations, or code.

b. Complete the initialization code below.

```
Config_Timer1A()
{
```

```
  // 1. Set up GPIO
  // A) Configure GPIO module associated with Timer 1A
    // i. Turn on clock for GPIO Port B and Timer 1
    // Note: Timer 1A can use Port B or F, this code uses Port B
    // Note: Port F would use different pins of its port.
    SYSCTL_RCGCGPIO_R |=
    SYSCTL_PRGPIO_R
    SYSCTL_RCGCTIMER_R |=
    SYSCTL_PRTIMER_R

    // ii. Enable alternate function and set peripheral functionality
    GPIO_PORTB_AFSEL_R |=
    GPIO_PORTB_PCTL_R

    // iii. Set digital mode
    GPIO_PORTB_DEN_R

  // 2. Set up Timer 1A
  // A) Configure Timer 1 mode
    //Disable Timer 1A device while we set it up
    TIMER1_CTL_R &=

    // Configure Timer 1A functionality
    TIMER1_CFG_R
    TIMER1_TAMR_R
    TIMER1_CTL_R

    TIMER1_TAPR_R = 0xFF; // Use prescaler extension to 24 bits
    TIMER1_TAILR_R = 0xFFFF // Load max 24-bit value

  // B) Set up Timer 1A interrupts
    TIMER1_ICR_R |=
    TIMER1_IM_R  |=

  // 3. NVIC setup
  // A) Configure NVIC to allow Timer 1A interrupts (use priority=1)
    NVIC_EN0_R |=
    NVIC_PRI5_R

 //  B) Bind Timer 1A interrupt requests to user's interrupt handler
    IntRegister(___________, TIMER1A_Handler);

 // Re-enable Timer 1A
    TIMER1_CTL_R |=

 // Globally enable CPU to service interrupts
    IntMasterEnable();
}
```

c. Suppose you are given the following main program and Stop_car function. Answer questions i. and ii.

```
// Global variables (additional variables may be used)
```

```
volatile unsigned int first_wheel_hit;  // 1st wheel hits sensor
volatile unsigned int second_wheel_hit; // 2nd wheel hits sensor
volatile int done_flag=0; // 1 after both first and
                          // second_wheel_hit have been stored
// Program to stop car if it is not fast enough to jump the gap.
main()
{
  int stop = 0;
  Config_Timer1A();
  while(1)
  {
    // Wait for sensor input events to be captured
    while(!done_flag)
    {
    }
    done_flag = 0;  // Clear flag
    // Check if car needs to stop
    stop = Stop_car();
    if(stop)
    {
      lcd_printf("Stopping car!");
    }
    else
    {
      lcd_printf("Car going to jump!!");
    }
  } // end while
}

// Return 0 if the car is fast enough to jump the gap
int Stop_car(void)
{
  float velocity_car;
  float time;

  // Compute car velocity based on captured times and distance
  // 16 MHz clock and counter tick rate => .0625us tick period
  time= (second_wheel_hit - first_wheel_hit) * .0625 // in usec
  time= time/1000000; // in seconds
  velocity_car = 1.5/time;  // in meter/s
  if(velocity_car < 31.2)
  {
    return 1;  // Not fast enough, stop
  }
  else
  {
    return 0;  // Fast enough, make jump
  }
}
```

i. Finish the Interrupt Service Routine code.

```
// Store timer values in first_wheel_hit and second_wheel_hit
void TIMER1A_Handler(void)
{
  static int state = 0; // 0/1: before/after first wheel hit

  // Check if an input edge-time capture interrupt occurred
  // Test the interrupt status bit
  if (TIMER1_MIS_R //finish this condition
  {
    // Clear the interrupt status bit by writing 1 to ICR bit
    TIMER1_ICR_R

    // Capture rising edge time for when 1st wheel hits sensor
    if(state == 0)
    {
      first_wheel_hit =
      state =
    }
    else // Capture rising edge time for when 2nd wheel hits sensor
    {
      second_wheel_hit =
      done_flag =
      state =
    }
  }
}
```

ii. Is it necessary for the `Stop_car()` function to check for timer overflow? Briefly explain why or why not?

Question 5: GPTM Timers and PWM

Suppose GPTM Timer 5A is configured in PWM mode. The system clock is 16 MHz. A variable for the match value is initialized as follows.

```
unsigned long match_value = 600,000;
```

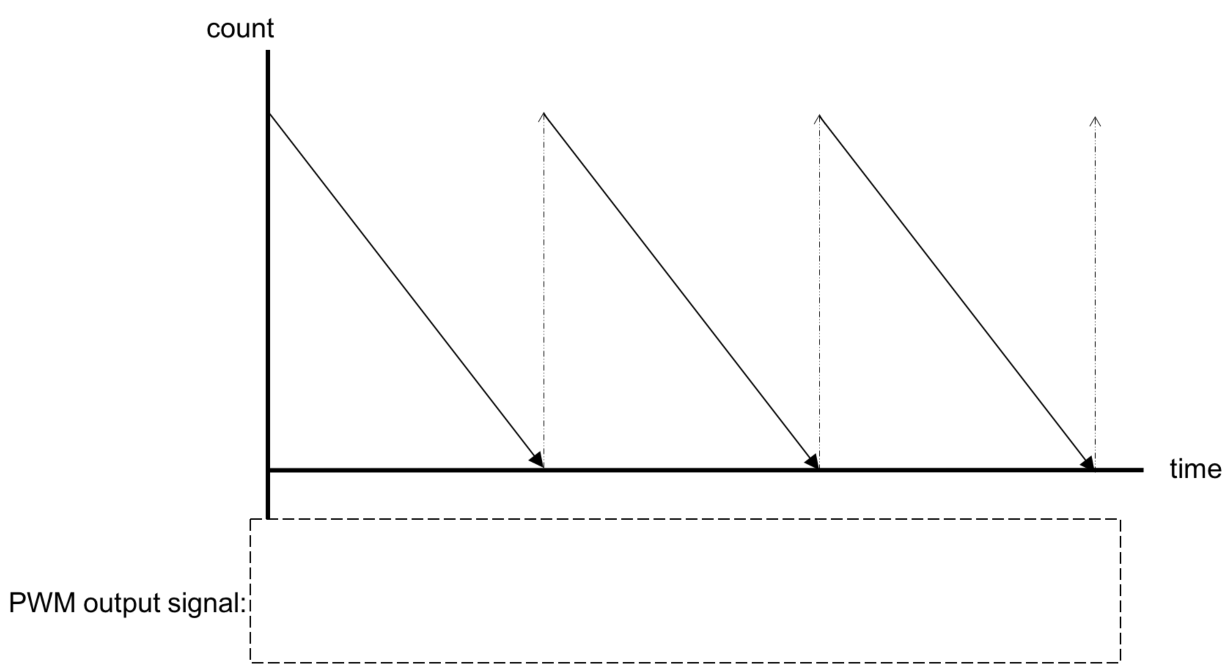
a. Counting down to 0 from the match value, what time in seconds would elapse for a `match_value` of 600,000?

b. What hex values would be put in these match registers during initialization?

GPTMTAPMR	GPTMTAMATCHR
0x	0x

c. For a start value of 750,000 and the match value of 600,000, what is the duty cycle of the PWM output signal? Note: the PWM output is high at the start (not inverted).

d. Similar to Figure 11-5 in the datasheet, show the start and match values in the diagram below on the y (count) axis, and sketch several cycles of the PWM output waveform in the dashed rectangle below the diagram. Use the values given in part c.



Question 6: GPTM Timers and PWM

a. Suppose Timer 2 is configured as follows. Assume that the GPIO module has been initialized appropriately.

```
//Timer 2 configuration

void init_TIMER2()
{
    SYSCTL_RCGCTIMER_R |= 0b00000100;
    while ((SYSCTL_PRTIMER_R & 0b00000100) {};
    TIMER2_CTL_R &= ~0x1; // Timer 2 Control
    TIMER2_CFG_R = 0x4;   // Timer 2 Configuration
    TIMER2_TAMR_R = 0xA; // Timer 2A Mode
    TIMER2_TAPR_R = 0;    // Timer 2A Prescaler
    TIMER2_TAILR_R = 32; // Timer 2A Interval Load
    TIMER2_TAPMR_R = 0; // Timer 2A Prescaler Match
    TIMER2_TAMATCHR_R = 8; // Timer 2A Match
    TIMER2_CTL_R |= 0x1; // Timer 2 Control
}
```

i) Describe the modes and configuration of the timer. Be as specific as possible.

ii) Which GPIO port and pin would be used with this configuration of the timer?

iii) What is the duty cycle of the output waveform?

b. Suppose you want to generate a waveform having a period of 5 ms and high pulse width of 1 ms. What values should be assigned in lines L6 through L9? Calculate numbers for the values, and write the numbers in hex in the blanks.

```
L6:    TIMER2_TAPR_R =          0x____________________;

L7:    TIMER2_TAILR_R =         0x____________________;

L8:    TIMER2_TAPMR_R =         0x____________________;

L9:    TIMER2_TAMATCHR_R =      0x____________________;
```

Question 7: PWM Programming

Generate a square wave (50% duty cycle) with a 12 ms period. Use GPTM Timer 1B in PWM mode. Assume the associated GPIO module has already been configured using another function. Your function should initialize and enable the timer.

```
void init_TIMER1()

{

    // Assume associated GPIO module already configured
```

Question 8: Timer Initialization

Suppose it's your job to design a microcontroller-based system that detects edges on an input signal waveform, calculates the period of the input signal, keeps a running average of the period, and generates an output signal having the same (average) period. The output signal should be a pulse train with a constant high pulse of 500 microseconds. The output signal period should vary with the average period of the input signal.

You have started your design and selected GPTM Timer 2 in 16-bit mode. You will use both the A and B timers from Timer 2. Timer 2A will detect rising edges of the input signal waveform (input CCP pin), and Timer 2B will generate the PWM output signal (output CCP pin). The CCP pins are alternate functions of GPIO pins.

a. Find the GPIO port(s) for the CCP pins for Timer 2. Write code that configures the GPIOAFSEL register so that CCP pins are set up as alternate functions. Don't worry about other GPIO registers; just initialize the alternate function of the pins for CCP. Preserve other bits.

b. Next, configure Timer 2 (and A and B) as needed using the following registers: RCGCTIMER, PRTIMER, GPTMCFG, GPTMTnMR, GPTMCTL. Don't worry about other registers that may need to be initialized. Write initialization code for this subset of registers only. Make assumptions as needed.

Question 9: General-Purpose Timers (GPTM)

a. Briefly describe each of the timer modes given in Table 9.1 of the Bai textbook.

b. For the GPTM Raw Interrupt Status Register (GPTMRIS), refer to information about bit 8, TBTORIS (Timer B Time-Out Raw Interrupt) time-out status flag. Under what condition(s) will the TBTORIS bit be set in this register? Hint: What does timeout mean?

c. For the GPTM Timer Mode Register (GPTMTnMR), refer to information about bit 10, TnMRSU (Match Register Update). This bit defaults to 0, and could be set to 1 by a program. Describe and/or sketch how the value of the MRSU bit affects when the assignment to the match register takes effect and hence the timing of the PWM output pulse in relation to the free-running counter.

   You used the GPTMTnMATCHR register in Lab 8 to generate an output pulse in PWM mode. Your program may have assigned a new match value when you wanted to change the pulse width and hence the position of the servo motor.

   Hint: Use Figure 11-5 (datasheet) as an example. Suppose a new value is assigned to the match register shortly after the count-down cycle begins. This new value could be any value in the count range. If MRSU = 0, when does the new match value take effect? If MRSU = 1, when does the new match value take effect? How does this affect when the PWM output signal goes low?

Question 10: Software-Implemented Input Capture

You have been using the input capture (edge time) mode of a GPTM timer. This means that input capture is implemented in the hardware of the timer module. You use the GPTM registers to set up and control the input capture hardware. What if the TM4C microcontroller did not have special input capture hardware? You would have to implement similar functionality in software.

a. Write a C program to save Timer 1A's count value (e.g., GPTMTAV) when a positive edge event occurs on GPIO Port D, pin 4 (PD4). It should store the timer value into variable `rising_edge_time`. Assume that the GPIO and GPTM modules have been properly configured in other code. Assume Timer 1A is configured and running in periodic mode.

```
int main(void)

{

  unsigned int rising_edge_time;


  // Assume GPIO Port D already configured properly

  // Assume GPTM Timer 1A already running in periodic mode


  while(1)

  {


  }

}
```

b. Describe two disadvantages of the software-implemented input capture as compared to input capture mode of the timer hardware.