Educational Objective:

The educational objectives of this laboratory assignment is to investigate the use of the timer modules in the MSP432 to create delays and Pulse Width Modulated (PWM) waveforms. The objective will be met by creating functions that control the movement of the robot's motor drivers using PWM signals. The functions will be called from a software state machines whose loop timing is determined by an additional function that creates a delay.

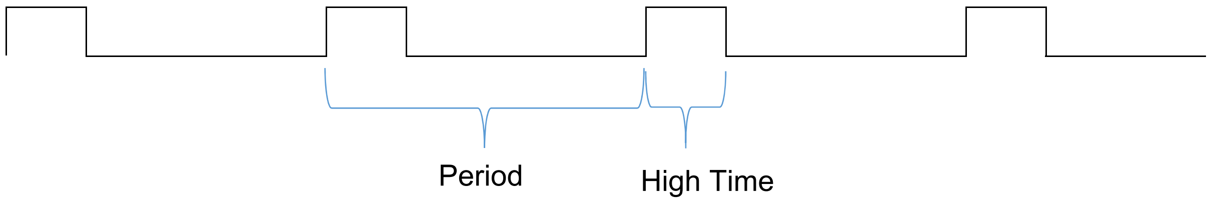
Background:

MSP432 Timer Module:

In lab 3, we used SysTick to create the PWM waveforms. However, the MSP432 has timers that are separate and distinct from SysTick. The timer module has much more versatility, runs in the background and can automatically create output waveforms. This lab will use the timer module for two purposes: PWM waves and delay. The timer has several clock options and we will be using SMCLK, which is 12MHz. Because the largest delay or waveform period that can be achieved with a 12MHz clock is 5.5ms (1/12Mhz * 2^{16}), clock dividers are needed to achieve the timing required for this lab. Clock division is set in the TAxCTL and TAxEX0 registers. In order to successfully set up the timer for this lab, you will need to reference the lecture notes and the MSP432 datasheet.

PWM:

A PWM signal is a wave that has a fixed frequency and a "high-time" whose duration determines the duty cycle.



$$Duty\ Cycle = \frac{High\ Time}{Period} \times 100$$

Figure 1. PWM signal with duty cycle calculation

As learned in lab 3, the EN signal on the DRV8838 motor driver needs to be driven by a PWM signal to move the wheels. The duty cycle controls the speed of the motors; the larger the duty cycle, the faster the wheels turn. For this lab, the period of the PWM signal will be fixed at 10ms and the duty cycle will be programmable through an input to a function call.

One of the advantages of using the microcontroller timer is that it can generate output waveforms on specific general purpose I/O (GPIO) pins that have been configured as outputs. Table 1 below shows the output pins, along with the SEL0 and SEL1 settings, for each of the timers.

Port Pin	Timer Output	(PxSEL1, PxSEL0)	Port Pin	Timer Output	(PxSEL1, PxSEL0)
P7.3	TA0.0	0, 1	P8.1	TA2.0	1, 0
P2.4	TA0.1	0, 1	P5.6	TA2.1	0, 1
P2.5	TA0.2	0, 1	P5.7	TA2.2	0, 1
P2.6	TA0.3	0, 1	P6.6	TA2.3	0, 1
P2.7	TA0.4	0, 1	P6.7	TA2.4	0, 1
P8.0	TA1.0	1, 0	P10.4	TA3.0	0, 1
P7.7	TA1.1	0, 1	P10.5	TA3.1	0, 1
P7.6	TA1.2	0, 1	P8.2	TA3.2	0, 1
P7.5	TA1.3	0, 1	P9.2	TA3.3	0, 1
P7.4	TA1.4	0, 1	P9.3	TA3.4	0, 1

Table 1. Timer Output Pins
(TAx.y = Timer x, Compare Block y)

Pin	TI-RSLK chassis board	DRV8838 Motor Driver	Description
P5.5	DIRR	PH	Right Motor Direction
P3.6	nSLPR	nSLEEP	Right Motor Sleep
P2.6	PWMR	EN	Right Motor PWM
P5.4	DIRL	PH	Left Motor Direction
P3.7	nSLPL	nSLEEP	Left Motor Sleep
P2.7	PWML	EN	Left Motor PWM

Table 2: Motor Signal Pinouts

Since the left and right motor pins are hardwired to P2.6 and P2.7, timer0, compare blocks 3 and 4 have to be used for the PWM signals on the TI-RSLK robot. When using the timer output pin, you have to initialize the pin as an output and set the PxSEL0 and PxSEL1 registers to select a function instead of GPIO pins.

Time Delay in Software State Machine:

In Lab 3, the amount of time that was spent in each state was determined by the period of the PWM multiplied by the stateTimer variable. That method cannot be used when the PWM signal is being created by the timer because it is continuous. Instead, we need to have a delay at the end of the while(1) loop. While the timing is not exact, for now it is close enough. A stateTimer variable that gets incremented each time a state is visited is used with a **delay()** function at the end of the loop to determine how long the state machine stays in a given state. For example, if the delay is 1 second, each state is held for 'stateTimer' seconds. For this lab you will create a function with a fixed 100ms delay length.

Pre-Laboratory:

1. Read and understand the requirements for this lab.
2. Create a new project in Code Composer and start a new main.c.
   a. Add the constants from lab 3 to the code. Add the **MotorInit()** function from lab 3 to the code. Be sure to add the register writes to choose the P2.6 and P2.7 pins as timer PWM outputs.
   b. Using the function template in Appendix A, write the **TimerInit()** function.
   c. Rewrite the four motion functions from last week to use the timer to create the PWM wave. The functions will set the period to a fixed 10ms. The duty cycles for the right and left motor drivers will be determined by two function inputs. You may have noticed in lab 3 that the motors are not perfectly calibrated and one is "faster" than the other, which prevents the robot from driving in a straight line. As such, you may have to drive the right and left motors with PWM signals that have different duty cycles. The two inputs to the function are the right motor duty cycle and the left motor duty cycle. The inputs should be in terms of microseconds. For example, an input of 2500 would create a PWM with a 25% duty cycle. Use the timer guide posted in MyCourses to help with setting the timer registers in these functions.
   d. Create a function that delays for 10ms. You can choose any timer available for this function. Use the timer guide posted in MyCourses to help with setting the registers for this function.
3. Review the coding structure for a software state machine and be prepared to write the code in lab.
4. Submit your main.c to the dropbox in MyCourses prior to your lab section. Also, put your code on a flash drive if you do not have a laptop to bring to lab.

Procedure:

1. Choose one of the direction functions (they should be very similar) and use an oscilloscope to verify you are creating a PWM wave with a 10ms period and a duty cycle of 25%. Do not turn the motors on when you are doing this.
2. Inside the while(1) loop of main(), call the **delay()** function and toggle an output pin each time you call it. Use an oscilloscope to verify that the output pin toggles every 10ms. **Demonstrate the PWM signals and delay signal on the oscilloscope for a signoff.**
3. Using the functions that your wrote in prelab, write the code, using a software state machine, that controls the robot to drive in pattern. It can be a square or a more creative pattern of your choosing. You will have to experiment with the calibration of the left and right motors so that the robot can move in a straight line.
4. **Demonstrate your robot pattern for a signoff.**

Post Lab:

Take the Post lab quiz in MyCourses before your next lab section.

Appendix A:

```
void TimerInit(void)
{
   //First initialize TimerA0 for PWM
   //Since the motors are connected to P2.6 and P2.7,use TimerA0,compare blocks 3 & 4
   //stop the timer
   //choose SMCLK for the clock source
   //choose clock divider of 2
   //Outmode 7: reset/set

   //Now initialize TimerAx for the delay function
   //stop the timer
   //choose SMCLK for the clock source
   //choose clock divider of 4 : ID = 10
   //choose second clock divider in TAxEX0 of 5, total divide is 20
}

void main(void)
{
   WDT_A->CTL = WDT_A_CTL_PW | WDT_A_CTL_HOLD;     // stop watchdog timer
   Clock_Init48MHz();  // makes bus clock 48 MHz
   MotorInit();
   TimerInit();

   //declare enumerated states, declare starting state, declare previous state, declare state timer
   //declare boolean to know if state has switched
   while(1)
   {
     switch (state) {
        case

        break;
        case

       break;
     } //switch
     Delay();
   } //while(1)
} //main()
```

Signoffs and Grade:

Name:____________________________________

Component	Signoff	Date
O’scope Waveforms		
Program Execution		

===

Component	Received	Possible
Prelab		20
Signoffs		60
Post-lab Quiz		20
Penalties • after the start of lab session • no signoffs after lab session	-	
Total		100