ELECTRONIC SYSTEMS ENGINEERING **ENEL 351**
FACULTY OF ENGINEERING and APPLIED SCIENCE
UNIVERSITY OF REGINA

Lab 5

Pulse Width Modulation (PWM using a Timer System)

Purpose:

This lab will familiarize the student with the configuration and operation of the PWM system on the STM32F10x.

Equipment/Software:

STM32F10x development Board
USB Cable
PC with Keil UVision
STM32F1XX Reference Manual
STM32F1XX Data Sheet

Background:

ESE/SSE students should by now be familiar with the concept of variable duty cycle pulse waveforms. The first encounter with a pulse waveform would have been the square wave, a pulse waveform of known frequency with a duty cycle of 50%. Duty cycle refers to the relationship of "on-time" vs. "off-time" for the waveform. Students in the ENEL 341/442 (previously known as ENEL 390/393) series of courses will know that we can also create pulse waveforms with a known frequency but with duty cycles of more or less than 50%.

One reason for our interest in the fixed frequency, variable duty cycle waveform lies in the average voltage provided to a load by such waveforms. If we apply a 0 to 5 volt square wave to a load, the average voltage delivered is 2.5 volts. If we increase the duty cycle to 75%, the average voltage rises to 3.75 volts. Conversely, a decrease in duty cycle to 25% results in an average load voltage of 1.25 volts. This property is especially useful if our load is a first order system, such as a small DC motor. The load acts as an integrator, providing an output speed proportional to the average of the applied voltage. If we use an appropriate motor driver interface, such as a BJT or FET H-bridge, we can control the speed of a motor directly from a 0 to 5 volt, fixed frequency, variable duty cycle pulse train. This type of signal is known as Pulse Width Modulation or PWM. You may have been exposed to uses for PWM in either ENEL 383 or ENEL 384.

A second application for PWM is the creation of command waveforms for Radio Control (R/C) servos. Most servos require a pulse train input with a period of around 20 mS and a duty cycle between 1 and 2 mS. This timing varies between manufacturers but will be sufficient for our purposes in this lab.

The STM32F10x is capable of generating PWM signals using the Timer system. The configuration options for PWM using a Timer system are detailed in the device Reference Manual.

Nucleo-64 Users: Use Timer3 Channel 1 (TIM3)

Procedure:

1) Develop a program to generate a PWM signal with a period of 10 mS and a duty cycle of 50%. All timing calculations must be based on a 24MHz clock.

2) Debug and test your program. Use the ADALM oscilloscope function to verify the timing of the output.

3) Modify the program so that you can increase or decrease the duty cycle while keeping the period constant. The duty cycle must stay between 1 and 99 %. You'll need to read some type of input in order to decide what the duty cycle should be at any point in time. Some possible inputs would be:

a) read the USER pushbutton state and use its value to determine the duty cycle **OR**

b) apply an analog voltage between 0 and 3.3 volts to PA0 and use its value to determine the duty cycle.

4) Debug and test the modified program. Again, use an oscilloscope to verify the result.

2023, Dave Duguid, Engineering Licensee, AScT

Deliverables:

Shoot and post a short (60 seconds max) video of your system in operation. Display the waveform being generated on your PWM output and demonstrate your ability to change the duty cycle with an external input. Provide the link to the video via the assignment link posted on URCourses.

Registers Used:

		TIM1	TIM3
TIMx_CR1		✓	✓
TIMx_CR2		✓	
TIMx_EGR		✓	✓
TIMx_CCMR1		✓	✓
TIMx_CCER		✓	✓
TIMx_PSC		✓	✓
TIMx_ARR		✓	✓
TIMx_CCR1		✓	✓
TIMx_BDTR		✓	

For Timer 1, Channel 1:

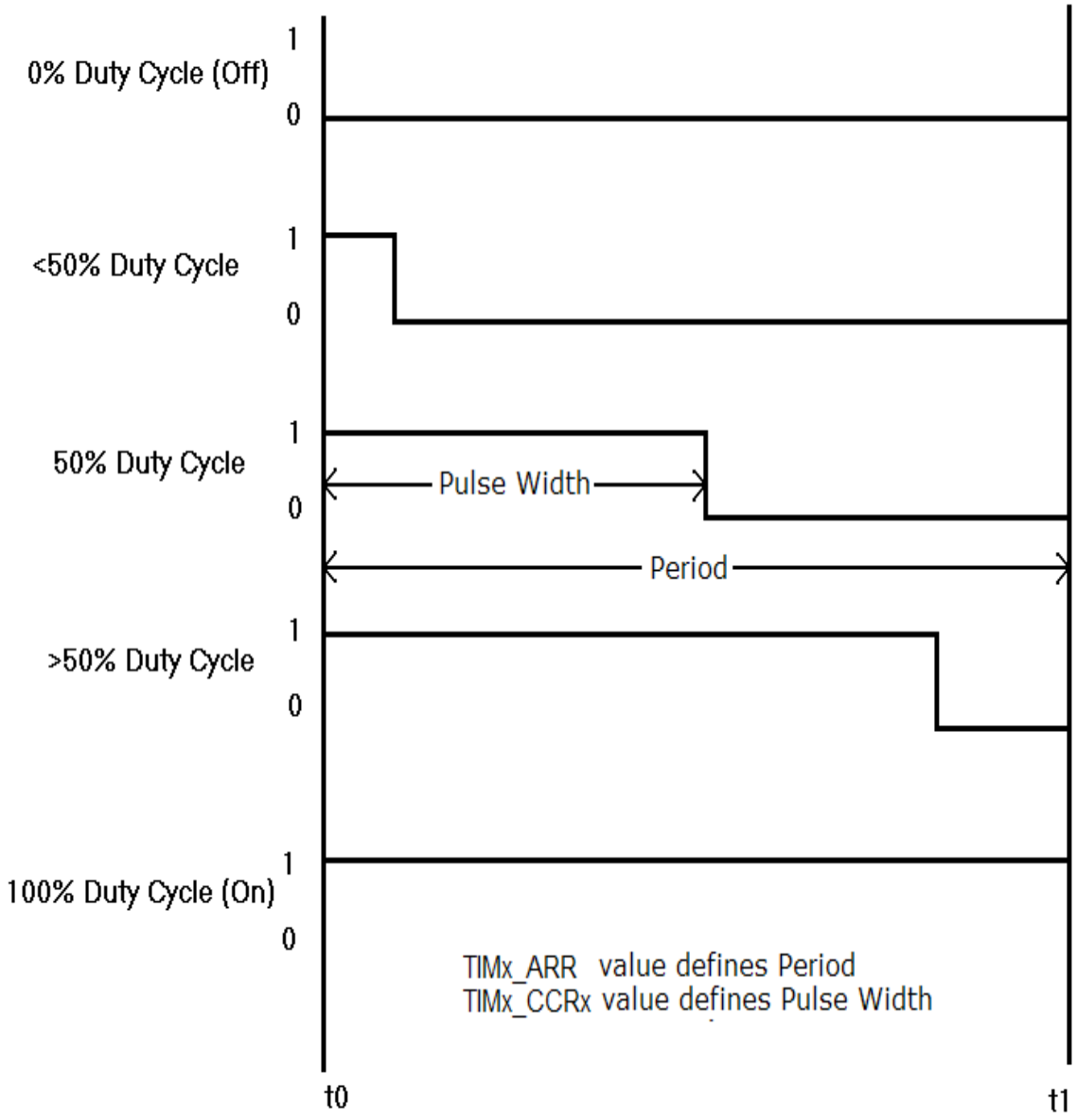
```
tim1GpioSetup();                                                    // Configure clocks and IO pins for TIM1 CH1
TIM1->CR1     |= TIM_CR1_CEN;                                       // Enable Timer1
TIM1->CR2     |= TIM_CR2_OIS1;                                      // Output Idle State for Channel 1 OC1=1 when MOE=0
TIM1->EGR     |= TIM_EGR_UG;                                        // Reinitialize the counter
TIM1->CCMR1   |= TIM_CCMR1_OC1M_2 | TIM_CCMR1_OC1M_1;               // PWM mode 1,
TIM1->CCMR1   |= TIM_CCMR1_OC1PE | TIM_CCMR1_OC1FE;                 // Preload Enable, Fast Enable
TIM1->CCER    |= TIM_CCER_CC1E;                                     //Enable CH1
TIM1->PSC     = 0x095F;                  // Divide 24 MHz by 2400 (PSC+1), PSC_CLK= 10000 Hz, 1 count = 0.1 ms
TIM1->ARR     = 100;                                                // 100 counts = 10 ms or 100 Hz
TIM1->CCR1    = 50;                                                 // 50 counts = 5 ms = 50% duty cycle
TIM1->BDTR    |= TIM_BDTR_MOE | TIM_BDTR_OSSI;                      // Main Output Enable, Force Idle Level First
TIM1->CR1     |= TIM_CR1_ARPE | TIM_CR1_CEN;                        // Enable Timer1
```

For Timer 3, Channel 1:

```
tim3GpioSetup();                                                    // Configure clocks and IO pins for TIM3 CH1
TIM3→CR1      |= TIM_CR1_CEN;                                       // Enable Timer3
TIM3->EGR     |= TIM_EGR_UG;                                        // Reinitialize the counter
TIM3->CCMR1   |= TIM_CCMR1_OC1M_2 | TIM_CCMR1_OC1M_1;               // PWM mode 1
TIM3->CCMR1   |= TIM_CCMR1_OC1PE | TIM_CCMR1_OC1FE;                 // Preload Enable, Fast Enable
TIM3→CCER     |= TIM_CCER_CC1E;                                     // Enable CH1
TIM3->PSC     =  0x095F;                 // Divide 24 MHz by 2400 (PSC+1), PSC_CLK= 10000 Hz, 1 count = 0.1 ms
TIM3→ARR      = 100;                                                // 100 counts = 10 ms or 100 Hz
TIM3→CCR1     = 50;                                                 // 50 counts = 5 ms = 50% duty cycle
TIM3→CR1      |= TIM_CR1_ARPE | TIM_CR1_CEN;                        // Enable Timer3
```

2023, Dave Duguid, Engineering Licensee, AScT

Figure 1: Typical PWM Waveforms



2023, Dave Duguid, Engineering Licensee, AScT