University of Texas at El Paso

Electrical and Computer Engineering Department

EE 3176 – Laboratory for Microprocessors I

Spring 2022

LAB 06

Pulse Width Modulation II

Goals:

- Use Port Interrupts with PWM (Pulse Width Modulation) to control the brightness pattern of an LED array.
- Use buttons to select the output pattern.
- Use the LCD display to indicate the output pattern.

Pre Lab Questions:

- What values of duty cycle must you use to control brightness in a perceivable range?
- Give your answers in milliseconds and percentage.
- What happens with duty cycles below or above the perceivable range?

Pulse Width Modulation II
Lab Guide

The microcontroller has a number of timers that support time-based measurements, input signal measurement, and output signal generation. For example, the TimerA is a peripheral with a basic timer block and a number of channels to measure input functions and generate output functions. In general, these channel operations are referred to as input capture and output compare, respectively. Thus, the channels may be referred to as capture compare channels. Notice that here, capture means input and compare means output. A pulse train or a square wave are types of signals you can produce, i.e., output, with the microcontroller.

When the signal is being generated, its frequency and duty cycle can be controlled through configuration of the used output channel. When a square wave is generated, the output channel can be configured to control the time the signal is a logical one vs. the time it is a logical zero. This is referred to as the signal duty cycle. Duty cycle is equal to the time the signal is logical one over the length of the signal cycle multiplied by 100 to express it as a percentage. Controlling the signal duty cycle is referred to as Pulse Width Modulation (PWM). The signal duty cycle is equal to the average output voltage and the actual voltage produced is equal to the duty cycle times the voltage value or a logical one. That is, the voltage difference between logical one and logical zero multiplied by the duty cycle plus the lowest output voltage. In combination with driver circuitry, the PWM average voltage can be used to control actuators such as LEDs, heaters, and DC motors.

TimerA

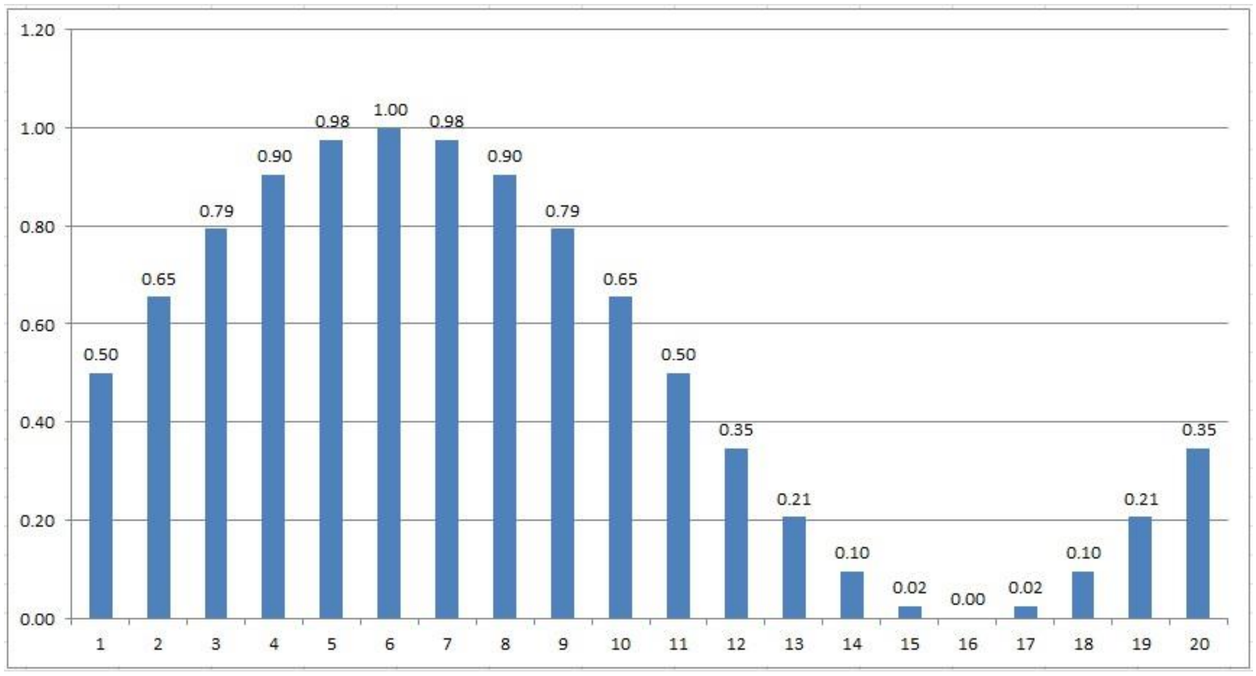
Timers found in a microcontroller depend on its chip number or actual model. The name of a timer is a combination of its function and how many like it are included in the microcontroller chip. For instance, when there are two TimerA modules, they are referred to as **Timer A0** and **Timer A1**. The registers to control a TimerA include the following:

• TAx->CTL	Control Register
• TAx->CCTLy	Capture/Compare Control Register
• TAx->CCRy	Capture/Compare Value Register
• TAx->TAR	Count Register
• TAx ->TAIV	Interrupt Vector Register

Where **x** is the module number. We can have Timer A0, Timer A1, etc. **y** is the Capture/Compare channel number: 0, 1, 2, etc. A register can have status, control, or data bits. Channels can interrupt the processor to indicate timer overflows or other events. Bit functions of each register are defined in the TimerA documentation.

Pulse Width Modulation II
System Design

The goal of the Lab is to use PWM to control an array of five LEDs to show three brightness patterns. The first pattern must be a five-sample window into a brightness sine wave that is constantly shifting left. Each LED must represent a sample and the signal period must be at least twenty samples. Here is a 20-sample example.



The second pattern must assign random brightness to each LED every 1.5 seconds. Finally, the third pattern must be a noisy version of the first pattern, i.e., the sine wave must have a small random increase or decrease in brightness with respect to the exact sine wave in the first pattern.

- Run and analyze the included PWM Demo program.
- Modify the demo program to output five PWM signals.
- Use input port interrupts to control the output patterns.
- Use the LCD display to show the name of the current output pattern.
- By default LEDs must be blinking at the same time with a slow frequency.

```c
#include "msp.h"
/******************************************************************************/
//  MSP432P401 Demo - TimerA, PWM, Up Mode, DCO SMCLK
//
//  Description: This program generates two PWM outputs on P2.4, P2.5 using
//  TimerA0.1 configured for up mode. The value in CCR0, 1000-1, defines the PWM
//  period and the values in CCR1 and CCR2 the PWM duty cycles. Using ~3MHz
//  SMCLK as TACLK, the timer period is ~1ms with a 75% duty cycle on one output
//  and 25% on the other.
//  ACLK = n/a, SMCLK = MCLK = TACLK = 3MHz
//
//           MSP432P401
//         ---------------
//     /|\|               |
//      | |               |
//      --|RST            |
//        |     P2.4/TA0.1|--> CCR1 - 75% PWM
//        |     P2.5/TA0.2|--> CCR2 - 25% PWM
//
//   William Goh
//   Texas Instruments Inc.
//   June 2016 (updated) | June 2014 (created)
//   Built with CCSv6.1, IAR, Keil, GCC
/******************************************************************************/
int main(void)
{
    WDT_A->CTL = WDT_A_CTL_PW | WDT_A_CTL_HOLD;     // stop watchdog timer

    // Configure GPIO
    P2->DIR |= BIT4 | BIT5;                  // P2.4~5 set TA0.1~2
    P2->SEL0 |= BIT4 | BIT5;
    P2->SEL1 &= ~(BIT4 | BIT5);

    TIMER_A0->CCR[0] = 1000 - 1;                 // PWM Period
    TIMER_A0->CCTL[1] = TIMER_A_CCTLN_OUTMOD_7; // CCR1 reset/set
    TIMER_A0->CCR[1] = 750;                      // CCR1 PWM duty cycle
    TIMER_A0->CCTL[2] = TIMER_A_CCTLN_OUTMOD_7; // CCR2 reset/set
    TIMER_A0->CCR[2] = 250;                      // CCR2 PWM duty cycle
    TIMER_A0->CTL = TIMER_A_CTL_SSEL__SMCLK |    // SMCLK
            TIMER_A_CTL_MC__UP |                 // Up mode
            TIMER_A_CTL_CLR;                     // Clear TAR

    // Enter LPM0
    __sleep();
    __no_operation();                        // For debugger
}
```

Pulse Width Modulation II
Post Lab

- Submit answers to the following questions together with your post-lab work.
- Are perceivable differences between LED brightness values linearly related? If not, what is the relationship between power and brightness?
- For each pattern, measure the output voltage of a channel and list the values that you read. Do they correspond to what you expected based on the used duty cycles and digital output voltage values? What do you think is happening?
- If you drive the kit servo with the output signals that you generated, what would be the effect on the motor? How do you know this?