University of Texas at El Paso
Electrical and Computer Engineering Department

EE 3176 – Laboratory for Microprocessors I

Spring 2023

LAB 05

Pulse Width Modulation

Goals:

- Use Port Interrupts with PWM (Pulse Width Modulation) to sweep the single-bladed arm of the SG90 back and forth between 0 and 180 degrees.
- Use buttons to start and stop the sweep.
- Use the LCD display to indicate the state of the motor as STOP,START, and position.

Pre Lab Questions:

- What values of duty cycle must you use to move the arm to 0°, 90° and 180°?
- Give your answers in milliseconds and percentage.
- What happens with duty cycles below or above those mentioned in the specification sheet?

Pulse Width Modulation

Lab Guide

Timers of the MSP432 microcontroller support time-based measurements, input signal measurements, and output signal generation. For instance, a TimerA has a basic timer block and a number of channels to measure input signal parameters and/or generate output signals such as a pulse train or a square wave. The duty cycle and the frequency of a signal to generate can be controlled through output channel configuration. To generate a square wave, an 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. Mathematically, 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. In turn, controlling the signal duty cycle is referred to as Pulse Width Modulation (PWM).

TimerA

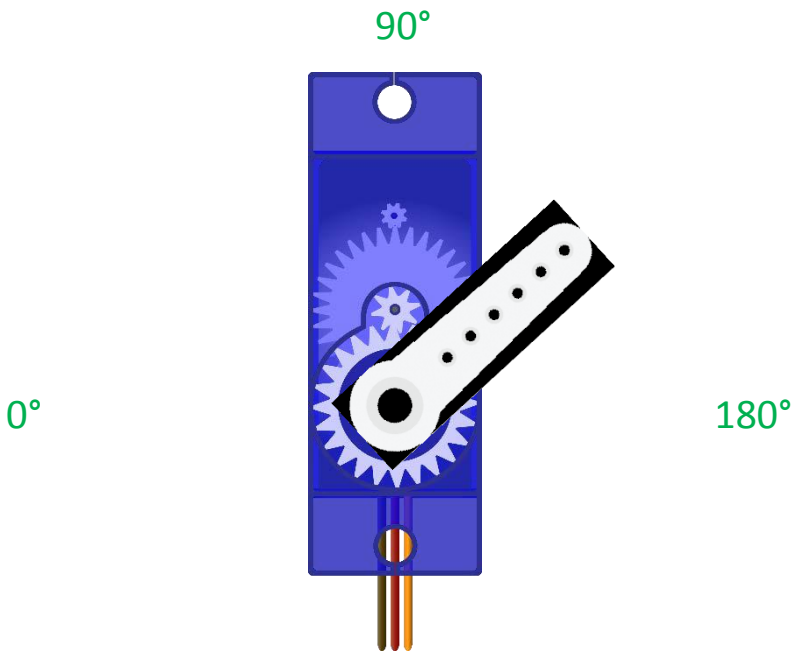
Notice that the number of TimerAs found in an MSP432 microcontroller depends on the model of MSP432 used. For instance, when there are two TimerA modules, they are referred to as **Timer A0** and **Timer A1**.
Registers that control TimerA peripherals include the following:

- 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. If it is 0 or 1, we have Timer A0 or Timer A1, and **y** is the Capture/Control Channel number: 0, 1, 2, etc. Each control register has status and control bits. Capture/Compare channels can produce interrupt signals for the processor such as Timer Overflows and Capture/Compare flags. Bit functions of each register are defined in the TimerA documentation.

SG90 Servo Motor

The SG90 is a simple servo motor with a 180° position capability. You can use 3 base positions to determine your motor exact range: far left, middle, and far right at -90°, 0° and 90° respectively. See the diagram below:



Its datasheet contains all the information you need. Connect the power wire to any VCC pin on the MSP432 and the Ground pin to a LaunchPad GND pin. The PWM wire must be connected to a PWM output Port Pin. For instance, use pins controlled by the TAx->CCR1. See the Interrupts Guide for more details. You can change the PWM duty cycle by setting this value. The cycle time of the PWM signal is controlled by the value of TAx->CCR0.

Notes:

- Each motor is slightly different, but they run best under 5V. However, the MSP432 puts out about 3.3V. This affects the pulse width's period, meaning that the duty cycle necessary for a given position will vary slightly from what is mentioned in the data sheet. A period of 20ms is recommended.
- The position of the motor may cause a momentary short, which will stop communication with the PC. This is not cause for concern.
- Your motor may have a range slightly greater or less than 90° degrees. Once the motor can no longer move, stop increasing the PWM duty cycle.

Pulse Width Modulation

System Design

The goal of the Lab is to use PWM to sweep back and forth through the whole 180° range of the motor, being able to start and stop using push buttons. Notice that depending on how you increment the pulse width duty cycle and set up timing, you will either get a slow choppy motion or a swift and smooth swipe.

- Run and analyze the included PWM Demo program.
- Write the demo program algorithm flow chart.
- Identify the PWM period register and duty cycle register. Their value is computed as follows: PWM frequency = Clock Source frequency / period and Duty cycle % = (100 * Time on)/cycle Time
- Modify the demo program to meet the following requirements.
- Use input port interrupts to control the servo motor position by setting PWM values according to the servo specs sheet.
- Use the LCD display to show current PWM parameters and servo motor position.
- Turn on a buzzer whenever the servo motor is moving.

```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

Frequently Asked Questions

What's inside a servo motor?

Inside the servo motor, we have a regular DC motor, a potentiometer, and a control circuit. The DC motor is connected to the control circuit through a series of gears and the potentiometer. As the DC motor rotates, the resistance increases and the motor determines where to stop depending on the pulse signal applied to the **PWM** wire.

Can I use the motor to push, pull or lower and raise something; I see '9g' on the side of the SG90; does that mean it can only lift 9 grams?

Each motor has a **Stall Torque** rating; a weight and distance that the motor can safely stall at, resisting any motion. For example, the SG90 has a Stall Torque rating of 16.7oz/in or 1.2kg/cm at 4.8V, meaning that the SG90 can, for instance, hold an item that weighs 16.7oz suspended by a string about 1 inch from the pivot of the attached arm or center of the motor. However, since we are interfacing with a 3.3V micro controller, those numbers will be smaller. It would still be close to a pound though. The '9g' on the side of the motor means that the motor itself weighs 9 grams!

My motor sometimes swings fast and other times it moves slow; how can I get it to move more consistently?

The speed of the motor is proportional to the current position and the desired position, meaning that the speed will be faster if the distance between the two positions is greater, and slower if the distance is smaller. This is known as **Proportional Control** and helps the motor be as efficient as possible, keeping the motor from doing more work than necessary. Unfortunately, this means precise speed control is not possible and so you must take this into account.