Lab 3 TC and PWM

EE138

Colin Chen & Daniel Graham

3/19/2015

Introduction

In this lab we will be using the Timer/Counter (TC) peripheral of the SAMD20 microcontroller to perform two separate tasks.

- Create a sine wave of 500Hz using a lowpass filter and a PWM
- Control relative speed of a DC motor in an H-bridge configuration using PWM

To successfully complete this task, we will need to properly set up the TC register correctly so that it will output the correct PWM signals to create our sine wave. In addition, we will be incorporating concepts we have used in Labs 1-3 to successfully integrate the potentiometer into controlling the speed of the motor.

Procedure

Sine wave

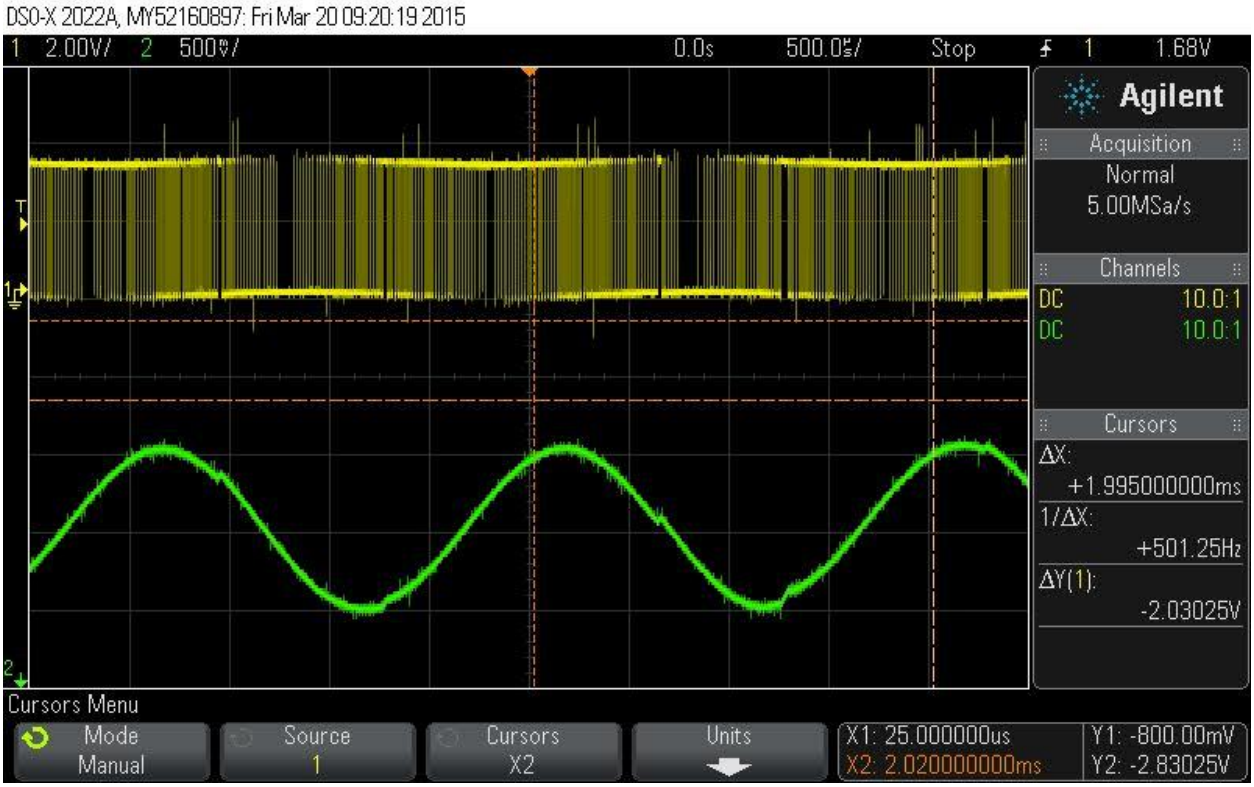
To construct a sine wave, we will be sending a variety of PWM signals into a lowpass filter. A sine wave has values going from -1 to 1 as a function of time. Through a lowpass filter, a duty cycle of a PWM will average the voltage allowing us to create a sine wave by modifying the duty cycle in respective to time. The way we figured out how to create a 500Hz sine wave, was we had to break down the sine wave into the respective PWM. For example, if we have 50 samples of a sine wave, we would need to have a PWM frequency of 500Hz*50. Using an external math tool, or using the sine fast_math function on the microcontroller, we can populate an array and use the array to adjust the duty cycles for the sine wave.

Motor Driving

The analog motor control began with implementing the ADC code from the previous lab. We found that we had to initialize the ADC control before we could initialize the TC due to a validation/verification error upon flashing the SAMD20 board. The ADC code from Lab 3 took an analog voltage across a potentiometer, converted it via the ADC, and the resulting data was displayed real-time on the 7-segment display. In order to control both the speed and the direction of the motor, we had to configure pins PORT_PA22 and PORT_PA23 to properly use the TC peripheral. Next, we initialized the TC clocks, correctly configuring the APBCMASK to work with the proper TC for the selected port output group. After we initialized the clock, we enabled an 8-bit TC in normal PWM mode. We found that if the duty cycle of the first counter was larger than the second counter, the motor would rotate clockwise when facing the shaft. We found that if the second counter was larger than the first, the motor would rotate counter clockwise. If both duty cycles were the same, the motor would not spin. We took the ADC output and created a system of equations to relate the ADC output to the duty cycles required to drive the motor. When the potentiometer rotates in either direction, one duty cycle increases while the other decreases relative to the direction the POT is turning. As the counter has an 8-bit upper limit, we chose a period of 125 which is within the limit. Once the duty cycles have been set, the TC is enable through CTRLA.

Results

The sine wave was generated correctly as shown in the figure below. We imported an array of sine values and imported static values in respective to the bit size we set in the PER register. The static values were imported into the function and set as the duty cycle value every time the TC register was accessed. We used 80 different samples along a sine wave to create the PWM duty cycle to produce our sine wave.



The main function of the program loops through endlessly, reading in the ADC value, bounding the result (to keep the adjusted duty cycles within period), displaying the ADC value, and then enabling the TC. We set a "dead" zone at roughly half of the maximum ADC output in order to ensure the motor would stop (as the voltage across the potentiometer can fluctuate, so can the ADC output and thus the motor duty cycles could fluctuate). We were able to control the clockwise and counterclockwise rotation, speed, and stopping position of the motor via the analog signal across the potentiometer by converting with the ADC, relating the result to duty cycles, and enabling the TC.