**EGR 226: Microcontroller Programming and Applications**

**Winter 2021**

Instructor: Professor Trevor Ekins

Lab 8: Creating a PWM Pseudo Analog Voltage Using Timer Peripherals of a Microcontroller

Sam Wieneke

March 17, 2021

**Contents**

1. **Objectives**
2. **Equipment**
3. **Introduction**
4. **Procedure**
5. **Results/ Discussion**
6. **Conclusion and Future Work**

**Appendices**

1. **Source Code: Part 1**

**Source Code: Part 2**

**Source Code: Part 3**

1. **Objectives**

The objectives of this experiment were to develop a program for the MSP432 that interfaces with a DC motor and controls the speed using PWM, to use SysTick to develop a PWM signal, to use the Timer A configured to provide a PWM signal as an output with connection to the I/O pin, and to interface the keypad to enter a value for setting the speed level.

1. **Equipment**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Part | Description | Model | Measured Value | Notes |
| Code Composer Studios | Texas Instruments programming environment | Version 9.3.0 | N/A | N/A |
| GitLab | Remote Repository for code maintenance | N/A | N/A | Makes collaboration on team projects and code very convenient. |
|  |  |  |  |  |

1. **Introduction**

**Part 1: Controlling the DC motor using the SysTick Timer**

Part 1 involved controlling a DC motor using the SysTick timer. The code had to be set up to be able to change the speed of the motor by manipulating the duty cycle within the CCS debugger window (meaning the speed was not hardcoded).

**Part 2: Controlling the Speed of a DC motor using Timer A in PWM Mode**

Part 2 involved controlling the DC motor using Timer A. It was very similar to part 1 but instead it used Timer A instead of SysTick. This part also was not allowed to be hardcoded and had to have a variable that could be changed within the CCS window.

**Part 3: Generating a PWM signal by reading the Duty Cycle from a Keypad**

Part 3 was part 2 but with the keypad implemented into it. The user would enter 2 numbers and that would be the duty cycle and the motor would change based off of the user’s input.

1. **Procedure**

**Part 1**

Part 1 was setup by setting up variables for duty cycle, the times on and off, and the period. Duty cycle was created as a volatile float so it could be changed within the CCS window without it being hardcoded.

**Part 2**

Part 1 was very similar to part 2, but Timer A was used instead of Systick. The Timer A code that was given in lecture was given and edited to fit this assignment.

TIMER\_A0->CCR[1] = (50000 \* (duty\_cycle/100));

This line of code was placed within a while(1) function because that allowed the duty cycle to be changed. The PWM period was set to 50000 and how the code above worked was it would take the duty cycle entered and entered into a formula to find the percentage of the period that was desired.

**Part 3**

Part 3 was similar to part 2, but more complex because the keypad was added. For this section, this was the main part of the code:

**while**(1)

{

pressed = Read\_Keypad();

**if**(pressed)

{

**if**(num != 10 && num !=12) //if the number is not \* or #

{

**if** (i==2)

{

pinArray[0] = pinArray[1]; //takes the last 2 numbers pressed

pinArray[1] = num;

}

**if** (i<2)

{

pinArray[i] = num;

i++;

}

}

**if**(num == 12 && i==2) //once # is pressed and there are 2 numbers it will change the value of duty cycle

{

duty\_cycle = ((10 \* pinArray[0]) + pinArray[1]);

DC\_Motor(duty\_cycle);

}

**if**(num == 12 && i==1)

{

duty\_cycle = pinArray[0];

DC\_Motor(duty\_cycle);

}

}

}

As seen above, for the keypad, when the keypad was pressed it met certain conditions it would either be saved into an array or disregarded completely. If a correct amount was entered and the pound key was pressed, it would then run through the DC\_Motor function and change the value of duty cycle in the same manner as part 2. The only difference is the DC motor function now has an integer being implemented instead of just being void.

1. **Results/ Discussion**

**Lab Results**

Questions:

1. How does the program use SysTick to create the desired delay for PWM?

The program used SysTick to delay a certain number of milliseconds based on the functions for timeON and timeOFF which was the duty cycle/100 times the period and the period minus timeON. This delay was what determined how fast the motor turned and my manipulating the wave it was possible to change the speed.

1. How does the timer configuration create the desired PWM?

The timer configuration creates the desired PWM almost exactly like SysTick does. It manipulates the wave by changing Ton to change the speed of the DC motor.

**Prelab:**

The prelab involved providing a description of how PWM changes the speed of the motor. It changes the speed because there is a wave and when it is high (Ton) the MSP will do something and when it is low (Toff) it will not. The length of Ton can be changed to do things faster or slower by manipulating the duty cycle. The prelab also involved drawing a circuit diagram of how the dc motor was to be setup with the optocoupler and the MSP.

A picture containing text, whiteboard

Description automatically generated

1. **Conclusion/ Future Work**

This lab was the first use of the TimerA and the DC motor. Some challenges that were faced during the lab were using Timer A because we haven’t used that outside of class until this point, and getting the CCS debugger to work correctly for changing the value of duty cycle while the code was running. The biggest problem that was faced during this lab and resulted in it being late was the fact that I had duty\_cycle /100 and not /100.0 which caused hours of looking over the code and trying to find what was wrong with it. Future work that could be implemented into this lab would be to learn Timer A and how it works better and practicing more with changing variables while the code is ran within CCS without using something like a keypad.

**Appendix A**

Part 1 Source Code:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Author: Samuel Wieneke

\* Course: EGR 226 - 902

\* Date: 3/17/2021

\* Project: Lab 8

\* File:

\* Description: This program will change the duty cycle of the DC motor within CCS

\* \*\* Uses SysTick

\*

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**#include** "msp.h"

**#include** <stdio.h>

**void** **SysTick\_Init** (**void**);

**void** **SysTick\_delay** (uint16\_t delay);

**void** **DC\_Motor** (**void**);

**volatile** **float** duty\_cycle;

**void** **main**(**void**)

{

WDT\_A->CTL = WDT\_A\_CTL\_PW | WDT\_A\_CTL\_HOLD; // stop watchdog timer

DC\_Motor();

SysTick\_Init();

**float** timeON, timeOFF, period;

//float duty\_cycle =10;

// int on, off;

duty\_cycle = 10;

period = 100;

**while**(1)

{

timeON = ((duty\_cycle/100)\* period);

timeOFF = period - timeON;

//on = (int)timeON;

//off = (int)timeOFF;

P2->OUT |= BIT4;

SysTick\_delay(timeON);

P2->OUT &= ~BIT4;

SysTick\_delay(timeOFF);

}

}

/\*\*\*\*| DC\_Motor | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Brief: This function is to initialize the pin

\* connected to the MSP

\* param:

\* (void) data: N/A

\* return:

\* N/A

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **DC\_Motor** (**void**)

{

//P2.4 is connected to DC Motor

P2->SEL0 &= ~BIT4; // configure P2.4 as simple I/O

P2->SEL1 &= ~BIT4;

P2->DIR |= BIT4; // P2.4 set as output pin \*/

P2->OUT &= ~BIT4;

}

/\*\*\*\*| SysTick\_Init | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Brief: This function is used for the initialization

\* of the SysTick timer

\* param:

\* (void) data: N/A

\* return:

\* N/A

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **SysTick\_Init** (**void**)

{ //initialization of systic timer

SysTick->CTRL = 0; // disable SysTick During step

SysTick->LOAD = 0x00FFFFFF; // max reload value

SysTick->VAL = 0; // any write to current clears it

SysTick->CTRL = 0x00000005; // enable systic, 3MHz, No Interrupts

}

/\*\*\*\*| SysTick\_delay | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Brief: This function sets the SysTick timer to be used as a

\* delay when called

\* param:

\* (uint16\_t) data: one integer worth of data input that

\* determines the duration of the delay

\* return:

\* N/A

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **SysTick\_delay** (uint16\_t delay)

{ // Systick delay function

SysTick->LOAD = ((delay \* 3000) - 1); //delay for 1 msecond per delay value

SysTick->VAL = 0; // any write to CVR clears it

**while** ( (SysTick->CTRL & 0x00010000) == 0); // wait for flag to be SET

}

Part 2 Source Code:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Author: Samuel Wieneke

\* Course: EGR 226 - 902

\* Date: 3/17/2021

\* Project: Lab 8

\* File:

\* Description: This program will change the duty cycle of the DC motor within CCS

\* \*\*Uses TimerA

\*

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**#include** "msp.h"

**#include** <stdio.h>

**void** **DC\_Motor** (**void**);

**volatile** **float** duty\_cycle=20;

**void** **main**(**void**)

{

WDT\_A->CTL = WDT\_A\_CTL\_PW | WDT\_A\_CTL\_HOLD; // stop watchdog timer

// duty\_cycle=20;

//uint16\_t duty\_cycle = 20;

DC\_Motor();

**while**(1)

{

//DC\_Motor(duty\_cycle);

TIMER\_A0->CCR[1] = (50000 \* (duty\_cycle/100));

}

}

/\*\*\*\*| DC\_Motor | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Brief: This function is to initialize Timer A and the Pin

\* connected to the MSP

\* param:

\* (void) data: N/A

\* return:

\* N/A

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **DC\_Motor** (**void**)

{

//P2.4 is connected to DC Motor

// Configure GPIO for PWM output

P2->SEL0 |= BIT4;

P2->SEL1 &= ~(BIT4);

P2->DIR |= BIT4; // P2.4 set TA0.1

//duty\_cycle = 80;

TIMER\_A0->CCR[0] = 50000 - 1; // PWM Period (# cycles of clock)

TIMER\_A0->CCTL[1] = TIMER\_A\_CCTLN\_OUTMOD\_7; // CCR1 reset/set mode 7

TIMER\_A0->CCR[1] = (50000 \* (duty\_cycle/100)); // CCR1 PWM duty cycle in 10ths of percent

TIMER\_A0->CTL = TIMER\_A\_CTL\_SSEL\_\_SMCLK | // use SMCLK

TIMER\_A\_CTL\_MC\_\_UP | // in Up mode

TIMER\_A\_CTL\_CLR; // Clear TAR to start

}

Part 3 Source Code:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Author: Samuel Wieneke

\* Course: EGR 226 - 902

\* Date: 3/17/2021

\* Project: Lab 8

\* File:

\* Description: This program will change the duty cycle of the DC motor by using keypad

\* \*\*Uses TimerA

\*

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**#include** "msp.h"

**#include** <stdio.h>

**void** **keypad\_init** (**void**);

uint8\_t **Read\_Keypad**(**void**);

**void** **SysTick\_Init** (**void**);

**void** **SysTick\_delay** (uint16\_t delay);

**void** **DC\_Motor** (**int** duty\_cycle);

uint8\_t pressed, num;

**int** duty\_cycle=50;

**void** **main**(**void**)

{

WDT\_A->CTL = WDT\_A\_CTL\_PW | WDT\_A\_CTL\_HOLD; // stop watchdog timer

keypad\_init();

SysTick\_Init();

DC\_Motor(duty\_cycle);

**int** i=0;

uint8\_t pinArray[2];

**while**(1)

{

pressed = Read\_Keypad();

**if**(pressed)

{

**if**(num != 10 && num !=12) //if the number is not \* or #

{

**if** (i==2)

{

pinArray[0] = pinArray[1]; //takes the last 2 numbers pressed

pinArray[1] = num;

}

**if** (i<2)

{

pinArray[i] = num;

i++;

}

}

**if**(num == 12 && i==2) //once # is pressed and there are 2 numbers it will change the value of duty cycle

{

duty\_cycle = ((10 \* pinArray[0]) + pinArray[1]);

DC\_Motor(duty\_cycle);

}

**if**(num == 12 && i==1)

{

duty\_cycle = pinArray[0];

DC\_Motor(duty\_cycle);

}

}

}

}

/\*\*\*\*| DC\_Motor | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Brief: This function is to initialize Timer A and the Pin

\* connected to the MSP

\* param: int duty\_cycle

\*

\* return:

\* N/A

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **DC\_Motor** (**int** duty\_cycle)

{

//P2.4 is connected to DC Motor

// Configure GPIO for PWM output

P2->SEL0 |= BIT4;

P2->SEL1 &= ~(BIT4);

P2->DIR |= BIT4; // P2.4 set TA0.1

//duty\_cycle = 80;

TIMER\_A0->CCR[0] = 50000 - 1; // PWM Period (# cycles of clock)

TIMER\_A0->CCTL[1] = TIMER\_A\_CCTLN\_OUTMOD\_7; // CCR1 reset/set mode 7

TIMER\_A0->CCR[1] = (50000 \* (duty\_cycle/100.0)); // CCR1 PWM duty cycle in 10ths of percent

TIMER\_A0->CTL = TIMER\_A\_CTL\_SSEL\_\_SMCLK | // use SMCLK

TIMER\_A\_CTL\_MC\_\_UP | // in Up mode

TIMER\_A\_CTL\_CLR; // Clear TAR to start

}

/\*\*\*\*| SysTick\_Init | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Brief: This function is used for the initialization

\* of the SysTick timer

\* param:

\* (void) data: N/A

\* return:

\* N/A

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **SysTick\_Init** (**void**)

{ //initialization of systic timer

SysTick->CTRL = 0; // disable SysTick During step

SysTick->LOAD = 0x00FFFFFF; // max reload value

SysTick->VAL = 0; // any write to current clears it

SysTick->CTRL = 0x00000005; // enable systic, 3MHz, No Interrupts

}

/\*\*\*\*| SysTick\_delay | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Brief: This function sets the SysTick timer to be used as a

\* delay when called

\* param:

\* (uint16\_t) data: one integer worth of data input that

\* determines the duration of the delay

\* return:

\* N/A

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **SysTick\_delay** (uint16\_t delay)

{ // Systick delay function

SysTick->LOAD = ((delay \* 3000) - 1); //delay for 1 msecond per delay value

SysTick->VAL = 0; // any write to CVR clears it

**while** ( (SysTick->CTRL & 0x00010000) == 0); // wait for flag to be SET

}

/\*\*\*\*| keypad\_init | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Brief: This function is to initialize all of the pins used in the

\* keypad

\* param:

\* (void) data: N/A

\* return:

\* N/A

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **keypad\_init** (**void**)

{

P4->SEL1 &= ~BIT0; // configure P4.0 as simple I/O

P4->SEL0 &= ~BIT0;

P4->DIR &= ~BIT0; // P4.0 set as output pin \*/

P4->REN |= BIT0;

P4->OUT |= BIT0;

P4->SEL1 &= ~BIT1; // configure P4.1 as simple I/O

P4->SEL0 &= ~BIT1;

P4->DIR &= ~BIT1; // P4.1 set as output pin \*/

P4->REN |= BIT1;

P4->OUT |= BIT1;

P4->SEL1 &= ~BIT2; // configure P4.2 as simple I/O

P4->SEL0 &= ~BIT2;

P4->DIR &= ~BIT2; // P4.2 set as output pin \*/

P4->REN |= BIT2;

P4->OUT |= BIT2;

P4->SEL1 &= ~BIT3; // configure P4.3 as simple I/O

P4->SEL0 &= ~BIT3;

P4->DIR &= ~BIT3; // P4.3 set as output pin \*/

P4->REN |= BIT3;

P4->OUT |= BIT3;

P4->SEL1 &= ~BIT4; // configure P4.4 as simple I/O

P4->SEL0 &= ~BIT4;

P4->DIR &= ~BIT4; // P4.4 set as output pin \*/

P4->SEL1 &= ~BIT5; // configure P4.5 as simple I/O

P4->SEL0 &= ~BIT5;

P4->DIR &= ~BIT5; // P4.5 set as output pin \*/

P4->SEL1 &= ~BIT6; // configure P4.6 as simple I/O

P4->SEL0 &= ~BIT6;

P4->DIR &= ~BIT6; // P4.6 set as output pin \*/

}

/\*\*\*\*| Read\_Keypad | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Brief: This function is to read the keypad and assign a number based off the one

\* pressed to the global variable num.

\* param:

\* (void) data: N/A

\* return:

\* 1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uint8\_t **Read\_Keypad**(**void**)

{

uint8\_t col, row;

**for** ( col = 0; col < 3; col++ )

{

P4->DIR = 0x00; // Set Columns to inputs

P4->DIR |= BIT ( 4+col ); // Set column 3 to output

P4->OUT &= ~BIT ( 4+col ); // Set column 3 to LOW

SysTick\_delay (10); // Delay the while loop

row = P4->IN & 0x0F; // read all rows

**while** ( !(P4IN & BIT0) | !(P4IN & BIT1) | !( P4IN & BIT2) | !( P4IN & BIT3) );

**if** (row != 0x0F) **break**; // if one of the input is low, some key is pressed.

}

P4->DIR = 0x00; // Set Columns to inputs

**if** ( col == 3)

**return** 0;

**if** (row == 0x0E) num = col + 1; // key in row 0

**if** (row == 0x0D) num = 3 + col +1; // key in row 1

**if** (row == 0x0B) num = 6 + col +1; // key in row 2

**if** (row == 0x07) num = 9 + col+1; // key in row 3

**return** 1;

}