Lab03 - PWM

Introduction to Embedded Systems - University of Nebraska

Dat Nguyen

Contents

1 Introduction

This lab provides hand-on activity and insight of PWM usages.

2 Program Description

2.1 Program 1 - Generate PWM Signal with 20ms Period

The fastest clock to select to achieve a 20ms PWM period is 2Mhz. This is selected by noticing that $\frac{2^{16}}{16Mhz} \simeq 4ms$ and $\frac{2^{16}}{2Mhz} \simeq 32ms$, where 2^{16} indicate the maximum top PWM range with 16Mhz and 2Mhz as the first two fastest clock selection. Notice that using 2Mhz clock will provide a range of period from 0ms to 32ms to tune.

Using above clock selection we can calculate and configure ICR1, COM1A, WGM1 and CS1 as follow

$$\frac{IRC1}{\frac{16Mhz}{prescaler}} = T = 20ms$$

$\Rightarrow IRC1 = 20ms * \frac{16Mhz}{8} = 40000$

$\Rightarrow CS1 = 010b \left(prescaler = 8 e.g \frac{CLK}{8} = 2Mhz\right)$

$\Rightarrow COM1A = 10b\,(clear\ OC1A\ on\ compare\ match\ and\ set\ at\ bottom\ mode)$

$\Rightarrow WGM1 = 1110b\,(select\ Fast\ PWM\ mode\ with\ top\ value\ as\ IRC1)$

Program is tuned with different OCR1 value to sanity check. Figure 1, 2, and 3 in the Appendix show the PWM signal with OCR1 value equals to 10000, 20000, and 30000 correspond to 25%, 50%, 75% duty cycle. Using above COM1A, WGM1, and CS1 setting, OCR1A is directly proportional to duty cycle.

Figure 1: 25% cycle PWM signal with 10000 ORC1 value

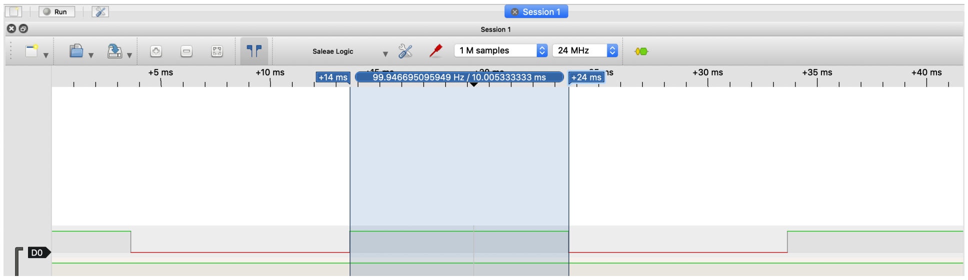

Figure 2: 50% cycle PWM signal with 20000 ORC1 valuel

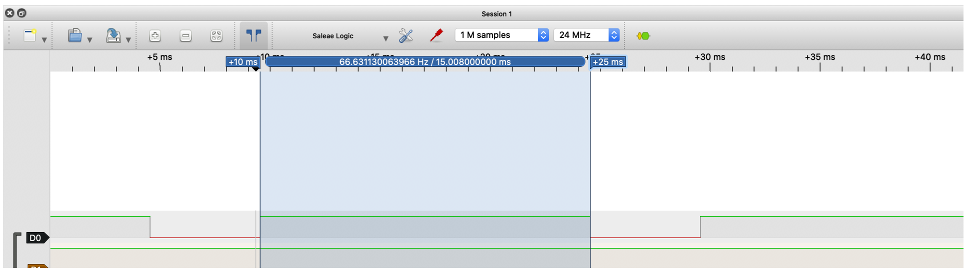

Figure 3: 75% cycle PWM signal with 30000 ORC1 value

2.2 Program 2 - Drive a Servo with PWM

For this experiment, the High Speed Continuous Parallax Servo was used. The servo is controller through pulse width modulation. Rotational speed direction are determined by the duration of high pulse, refreshed every 20ms(hence 20ms period). A 1.50ms control pulse make the servo stand still. As pulse width decrease from 1.52ms to 1.3ms, the servo rotate faster, clockwise. AS pulse width increases from 1.52ms to 1.7ms, the servo gradually rotate faster, counterclockwise. Figure 4 and 5 show the servo and its control pulse for different servo state.

Figure 4: Parallax High Speed Continuous Servo

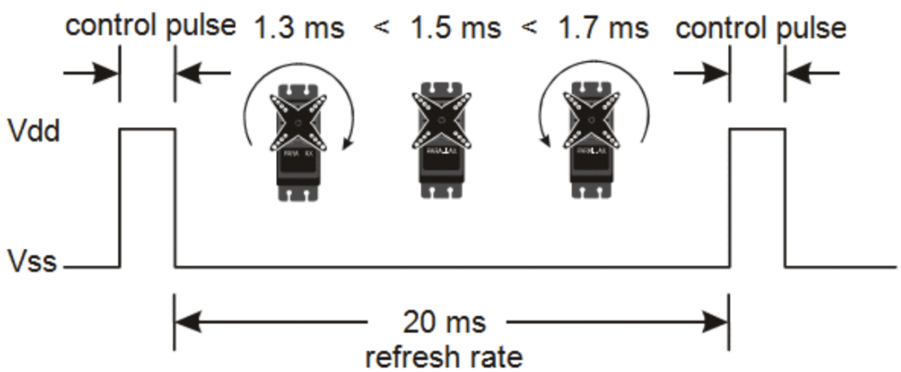

Figure 5: Servo Control Signal for Different State

We will use experiment 01 configuration for this with 40000 as top value and 2Mhz as clock cycle. Converting the above PWM signal from time to value between 0 and 40000 from equation below we have

$$\frac{high\ duration}{20ms} = \frac{OCR1}{40000}$$

OCR1 = (high $duration) * 40000/20ms$

(high $duration) = 1.5ms \Rightarrow OCR1 = 3000$

(high $duration) = 1.3ms \Rightarrow OCR1 = 2600$

(high $duration) = 1.7ms \Rightarrow OCR1 = 3400$

From computed OCR1 value range above, we configured 2800 to drive CW, 3000 to stop servo, and 3200 to drive CCW. The top value is fixed as 40000, we can derive the duty cycle and period as follow

State	OCR1	Period(ms)	Duty cycle (%)
Stop	3000	1.5	7.5
CW	2800	1.4	7
CCW	3200	1.6	8

Figure 6: Period and Duty Cycle for Different Servo State

Period = OCR1 * 20ms / 40000

Duty cycle = Period * 100 / (20ms)

Figure 7, 8, and 9 display PWM signal of stop, cw, ccw state respectively

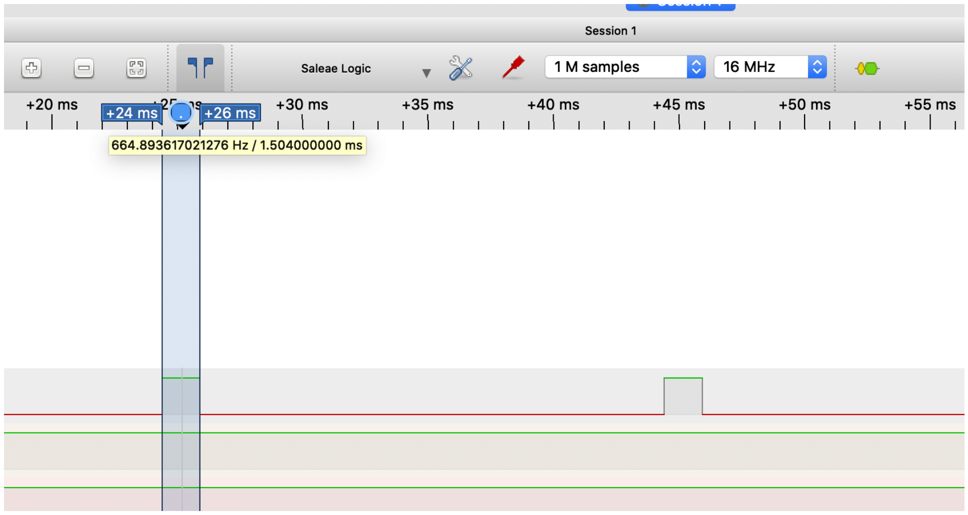

Figure 7: PWM Signal for STOP Servo State

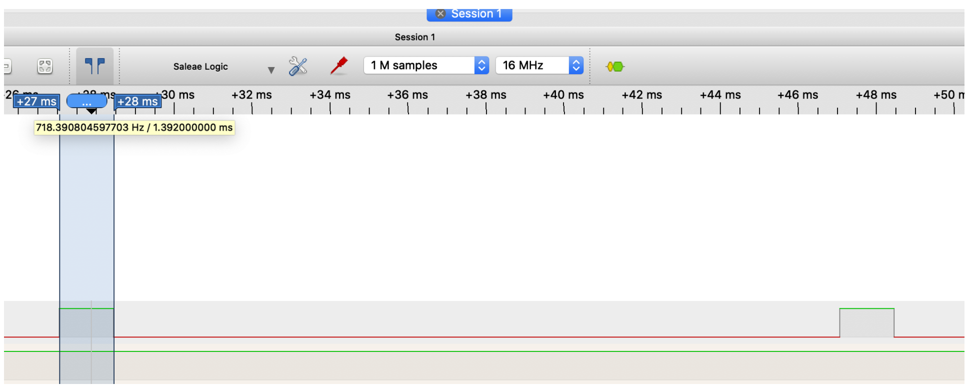

Figure 8: PWM Signal for CW Servo State

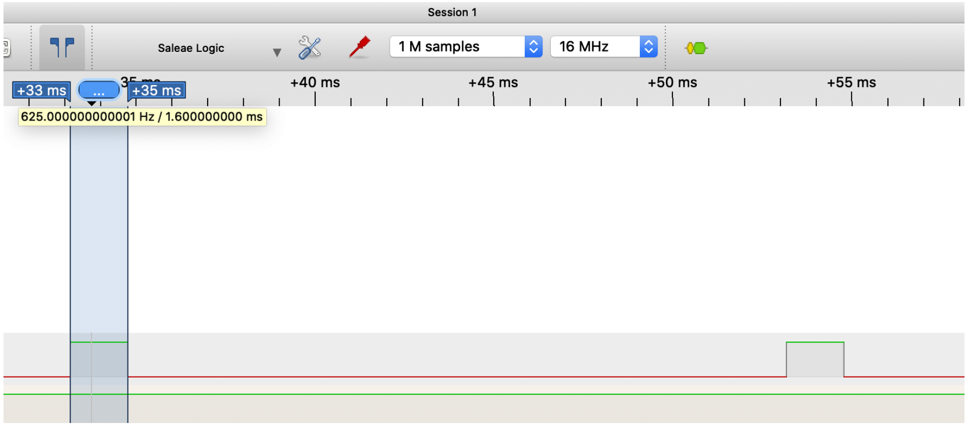

Figure 9: PWM Signal for CCW Servo State

Even though the PWM period is being produced to stop servo, the power consumption is negligibly small. Servo consumped large amount of current on the 5v line when it is driving CW and CCW. Figure 10 below display in detail the current and power consumption for each servo state.

State	Current(mA)	Voltage(V)	Power(mW)
STOP	12	5	60
CW	85	5	425
CCW	85	5	425

Figure 10: Servo states and Power Consumption

3 Summary

This lab introduced Timer/Counter 1 peripheral and pulse width modulation. Experiment 1 help demonstrate Counter 1 peripheral usage on configuring Fast PWM mode with TOP value controlled via ICR1.

4 Appendix

4.1 Main program

```
#include <stdint.h>;

#include "expriments.h";

int main(void) {
  init();
  Serial.begin(9600);
//  experiment01();
  experiment02();
  return 0;
```

```
}
```

4.2 Program 01

```
#include <Arduino.h>

/**********************************************/
/***********EXPERIMENT 01*********************/
/**********************************************/


volatile uint8_t *ioDDRB,
                 *ioPORTB, // used in experiment 02 only to config button
                 *ioPINB,  // used in experiment 02 only to config button
                 *pTCCR1A, // timer counter control register A
                 *pTCCR1B, // timer counter control register B . A and B help group
                     information and not nessary similar to I/O A and B.
                 *pOCR1AH, // output control register A high
                 *pOCR1AL, // output control register A low
                 *pICR1H,  // input capture register high
                 *pICR1L;  // input capture register low


void experiment01() {
  ioDDRB  = (uint8_t *) 0x24;

  pTCCR1A = (uint8_t *) 0x80;
  pTCCR1B = (uint8_t *) 0x81;
  pOCR1AH = (uint8_t *) 0x89;
  pOCR1AL = (uint8_t *) 0x88;
  pICR1H = (uint8_t *) 0x87;
  pICR1L = (uint8_t *) 0x86;

  /**
   * Write a program to generate a PWM signal with a period of 20 msec on OC1A pin(PB1 -
       pin 9) using the Timer/Counter 1 peripheral
   * Select Fast PWM mode with TOP value controlled via the ICR1 register
   * Try to create PWM signle with period 20ms on OC1A
   * using clock @ 2Mhz = 16Mhz/8
   *
   * IRC1/2Mhz = 20ms
   * IRC1 = 20ms * 2Mhz = 40,000 or 001110001000000b
   */
   //make PB1 an output
   *ioDDRB = 0x02; // DDRB[7:0] = 0000 0010
   uint16_t IRC1_val = 40000;
   uint16_t OCR1A_val = 30000; // value range [0, IRC1_val-1]
  *pTCCR1A = 0b10000010;  // C M1A = 10b clear  C1A  on compare match and set @ bottm
                          // WGM[1:0] = 10b (mode 14 - Fast PWM with ICR1 as TOP value)
  *pTCCR1B = 0b00011010;  // WGM[4:3] = 11b (mode 14 - Fast PWM with ICR1 as TOP value)
                          // CSI[2:0] = 010b (prescale as 8 e.g CLK/8 = 2Mghz)
  *pICR1H  = (IRC1_val >> 8);
  *pICR1L  = IRC1_val & 0xFF;

  *pOCR1AH = (OCR1A_val >> 8);
  *pOCR1AL = OCR1A_val & 0xFF;

  while(1);
}
```

4.3 Program 2

```
/**********************************************/
```

```
/************EXPERIMENT 02**********************/
/**********************************************/
// Experiment 02: Drive Servo with different state configure through button

volatile uint8_t *ioDDRB,
                 *ioPORTB, // used in experiment 02 only to config button
                 *ioPINB,  // used in experiment 02 only to config button
                 *pTCCR1A, // timer counter control register A
                 *pTCCR1B, // timer counter control register B . A and B help group
                     information and not nessary similar to I/O A and B.
                 *pOCR1AH, // output control register A high
                 *pOCR1AL, // output control register A low
                 *pICR1H,  // input capture register high
                 *pICR1L;  // input capture register low

#define STOP_VALUE   3000
#define CW_VALUE     2800
#define CCW_VALUE    3200
#define PRESSED      0
#define NOT_PRESSED 1

#define STATE_SERVO_STOP 0
#define STATE_SERVO_CW    1
#define STATE_SERVO_CCW   2

uint8_t state = 0;
uint8_t pinPB0State = 0;

void changeState() {
  /*
   * State 0: stop
   * State 1: rotate servo CW
   * State 2: rotate servo CCW
   */
  state = (state + 1) % 3;
}


void changeTopValue(uint16_t IRC1_val) {
  *pICR1H  = (IRC1_val >> 8);
  *pICR1L  = IRC1_val & 0xFF;
}

void changeDutyCycle(uint16_t OCR1A_val) { // value range [0, IRC1_val-1]
  *pOCR1AH = (OCR1A_val >> 8);
  *pOCR1AL = OCR1A_val & 0xFF;
}

uint8_t readInputPB0() {
  return ((*ioPINB) & 0x01) >> 0;
}

void myHardDelay(uint32_t ms) {
  volatile int16_t count;

  while (ms) {
    for (count = 0; count < 835; count++);
    ms -= 1;
  }
}

int debouncePB0(void) {
  uint8_t currentPB0Val = readInputPB0();

  if (currentPB0Val != pinPB0State) {
    // have a potential pin change!!
    myHardDelay(20); // wait for bounce to end
    currentPB0Val = readInputPB0();
```

```
    if (currentPB0Val != pinPB0State) { // if still diff, then it was a transtient and
        was an actual pin changed
      pinPB0State = currentPB0Val;
      return 1; // pin changed
    }
  }
  return 0; // pin didn't changed
}

void experiment02() { // measure ADC0 reading
  ioPINB  = (uint8_t *) 0x23;
  ioDDRB  = (uint8_t *) 0x24;
  ioPORTB = (uint8_t *) 0x25;

  pTCCR1A = (uint8_t *) 0x80;
  pTCCR1B = (uint8_t *) 0x81;
  pOCR1AH = (uint8_t *) 0x89;
  pOCR1AL = (uint8_t *) 0x88;
  pICR1H  = (uint8_t *) 0x87;
  pICR1L  = (uint8_t *) 0x86;

  /**
   * Write a program to generate a PWM signal with a period of 20 msec on OC1A pin(PB1 -
       pin 9) using the Timer/Counter 1 peripheral
   * Select Fast PWM mode with TOP value controlled via the ICR1 register
   * Try to create PWM signle with period 20ms on OC1A
   * using clock @ 2Mhz = 16Mhz/8
   *
   * IRC1/2Mhz = 20ms
   * IRC1 = 20ms * 2Mhz = 40,000 or 001110001000000b
   */
   //make PB1 an output and PB0 an input
   *ioDDRB = 0x02; // DDRB[7:0] = 0000 0010

   //Enable internal pull up for PB0
   *ioPORTB = 0x01;

   uint16_t IRC1_val = 40000;

  *pTCCR1A = 0b10000010; // C M1A = 10b clear  C1A  on compare match and set @ bottm
                          // WGM[1:0] = 10b (mode 14 - Fast PWM with ICR1 as TOP value)
  *pTCCR1B = 0b00011010;  // WGM[4:3] = 11b (mode 14 - Fast PWM with ICR1 as TOP value)
                          // CSI[2:0] = 010b (prescale as 8 e.g CLK/8 = 2Mghz)

  changeTopValue(IRC1_val);
  changeDutyCycle(STOP_VALUE);

  pinPB0State = NOT_PRESSED;

  while(1) {
    if (debouncePB0()) {
      if (pinPB0State == NOT_PRESSED) {
        Serial.println("PB0 not pressed");
      } else {
        Serial.println("PB0 pressed");
        changeState();
      }
    }
    Serial.print("Current state ");
    if (state == STATE_SERVO_STOP) {
      Serial.println("(0:STOP)");
      changeDutyCycle(STOP_VALUE);
    } else if (state == STATE_SERVO_CW) {
      Serial.println("(1:CW)");
      changeDutyCycle(CW_VALUE);
    } else if (state == STATE_SERVO_CCW) {
      Serial.println("(2:CCW)");
      changeDutyCycle(CCW_VALUE);
```

```
    } else {
      // why are you here?
    }

  }
}
```