LAB: Timer & PWM

Embedded Controller Lab Report

LAB: Timer & PWM

Date: 2023.10.19

Author: Hanmin Kim

Github:https://github.com/ansterdaz/HKim/tree/main

Demo video:

RC motor: https://youtube.com/shorts/VEphpQGaS0I

DC motor: https://youtube.com/shorts/Ya6Zn1ocMfU

Introduction

Create a simple program that control a sevo motor and a DC motor with PWM output.

Requirement

Hardware

MCU

- NUCLEO-F411RE

Actuator/Sensor/Others:

- RC Servo Motor (SG90)
- DC motor (5V)
- DC motor driver(LS9110s)
- breadboard

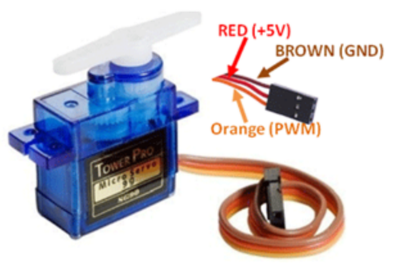
Software

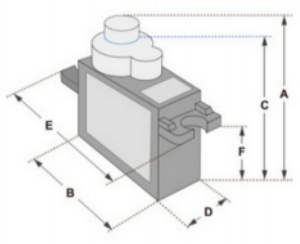
Keil uVision, CMSIS, EC_HAL library

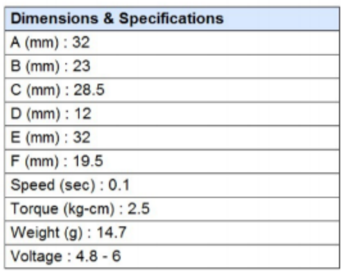
Problem 1: RC servo motor

Procedure

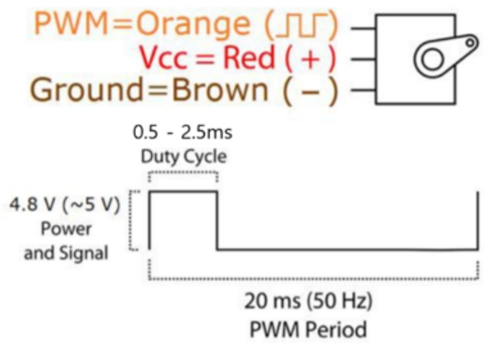
An RC servo motor is a tiny and light weight motor with high output power. It is used to control rotation angles, approximately 180 degrees (90 degrees in each direction) and commonly applied in RC car, and Small-scaled robots. The angle of the motor can be controlled by the pulse width (duty ratio) of PWM signal. The PWM period should be set at **20ms or 50Hz**. Refer to the datasheet of the RC servo motor for detailed specifications.

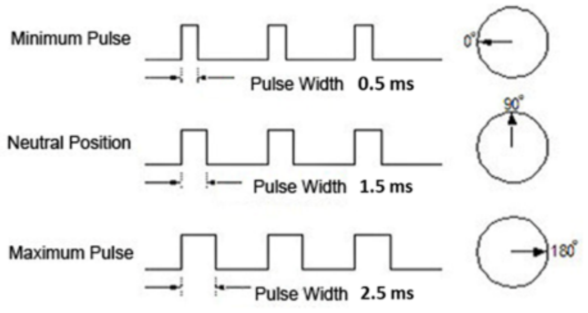






Dimensions & Specifications
A (mm) : 32
B (mm) : 23
C (mm) : 28.5
D (mm) : 12
E (mm) : 32
F (mm) : 19.5
Speed (sec) : 0.1
Torque (kg-cm) : 2.5
Weight (g) : 14.7
Voltage : 4.8 - 6





1-1. Create HAL library

Download files:

- ecPinNames.h ecPinNames.c
- ecTIM_student.h, ecTIM_student.c
- ecPWM_student.h, ecPWM_student.c

Then, change the library files as

- ecTIM.h, ecTIM.c
- ecPWM.h, ecPWM.c

Declare and define the following functions in your library. You must update your header files located in the directory `EC \lib\`.

ecTIM.h

```c
// Timer Period setup
void TIM_init(TIM_TypeDef *TIMx, uint32_t msec);
void TIM_period(TIM_TypeDef* TIMx, uint32_t msec);
void TIM_period_ms(TIM_TypeDef* TIMx, uint32_t msec);
void TIM_period_us(TIM_TypeDef* TIMx, uint32_t usec);

// Timer Interrupt setup
void TIM_UI_init(TIM_TypeDef* TIMx, uint32_t msec);
void TIM_UI_enable(TIM_TypeDef* TIMx);
void TIM_UI_disable(TIM_TypeDef* TIMx);


// Timer Interrupt Flag
uint32_t is_UIF(TIM_TypeDef *TIMx);
void clear_UIF(TIM_TypeDef *TIMx);
```

ecPWM.h

```
/* PWM Configuration using PinName_t Structure */

/* PWM initialization */
// Default: 84MHz PLL, 1MHz CK_CNT, 50% duty ratio, 1msec period
void PWM_init(PinName_t pinName);
void PWM_pinmap(PinName_t pinName, TIM_TypeDef **TIMx, int *chN);


/* PWM PERIOD SETUP */
// allowable range for msec:  1~2,000
void PWM_period(PinName_t pinName,  uint32_t msec);
void PWM_period_ms(PinName_t pinName,  uint32_t msec);  // same as PWM_period(
// allowable range for usec:  1~1,000
void PWM_period_us(PinName_t pinName, uint32_t usec);


/* DUTY RATIO SETUP */
// High Pulse width in msec
void PWM_pulsewidth(PinName_t pinName, uint32_t pulse_width_ms);
void PWM_pulsewidth_ms(PinName_t pinName, uint32_t pulse_width_ms);  // same a
// Duty ratio 0~1.0
void PWM_duty(PinName_t pinName, float duty);
```

Procedure

Make a simple program that changes the angle of the RC servo motor that rotates back and forth from 0 deg to 180 degree within a given period of time.

Reset to '0' degree by pressing the push button (PC13).

- Button input has to be an External Interrupt
- Use Port A Pin 1 as PWM output pin for TIM2_CH2.
- Use Timer interrupt of period 500msec.
- Angle of RC servo motor should rotate from 0° to 180° and back 0° at a step of 10° at the rate of 500msec.

1. Create a new project under the directory `\repos\EC\LAB\LAB_PWM`

- The project name is “**LAB_PWM”.**
- Create a new source file named as “**LAB_PWM_RCmotor.c”**

2. Include your updated library in `\repos\EC\lib\` to your project.

- **ecPinNames.h ecPinNames.c**
- **ecGPIO.h, ecGPIO.c**
- **ecRCC.h, ecRCC.c**
- **ecEXTI.h, ecEXTI.c**
- **ecTIM.h**, **ecTIM.c**

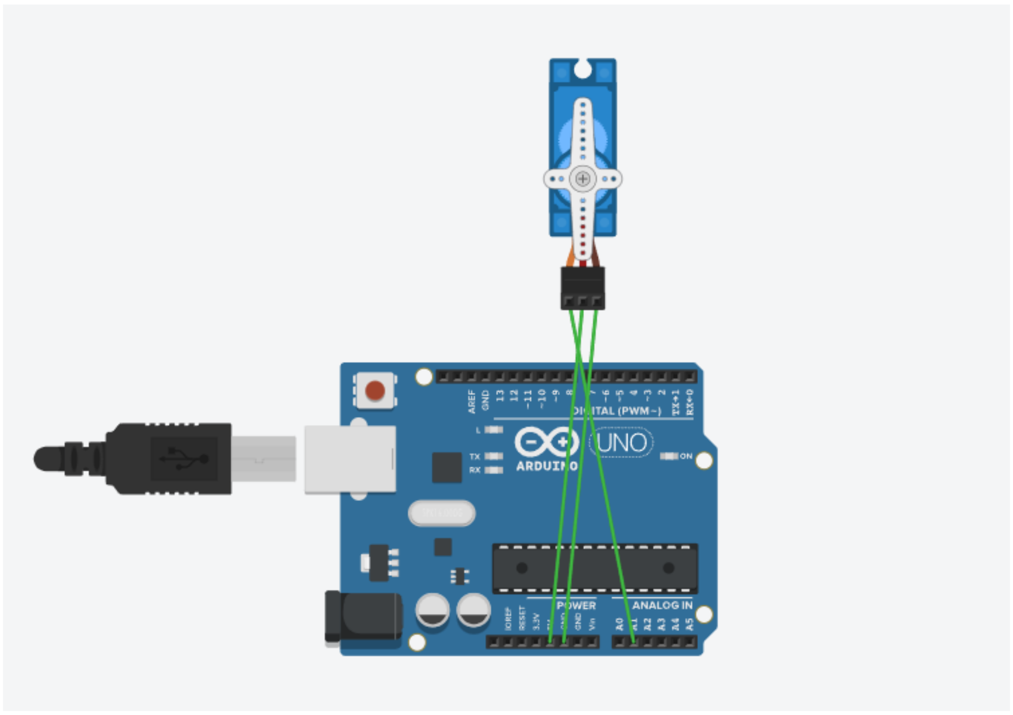
- **ecPWM.h ecPWM.h**

3. Connect the RC servo motor to MCU pin (PA1) , VCC and GND
4. Increase the angle of RC servo motor from 0° to 180° with a step of 10° every 500msec. After reaching 180°, decrease the angle back to 0°. Use timer interrupt IRQ.
5. When the button is pressed, it should reset to the angle 0° and start over. Use EXT interrupt.

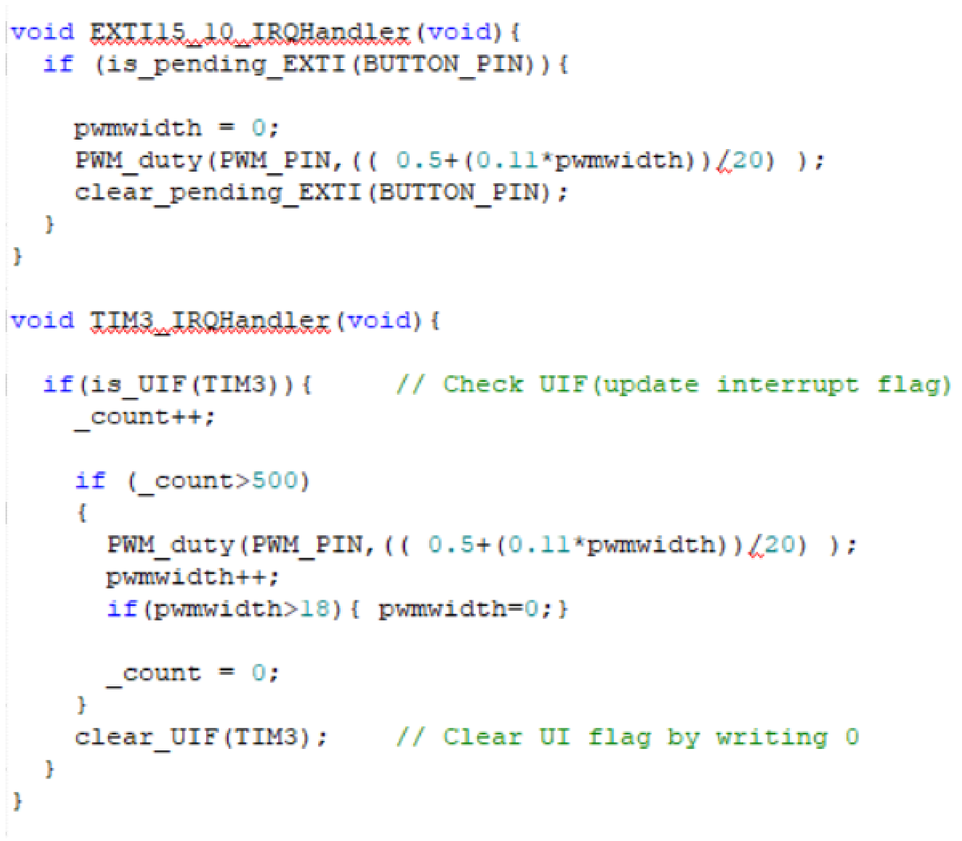
Configuration

Type	Port - Pin	Configuration
Button	Digital In (PC13)	Pull-Up
PWM Pin	AF (PA1)	Push-Pull, Pull-Up, Fast
PWM Timer	TIM2_CH2 (PA1)	TIM2 (PWM) period: 20msec, Duty ratio: 0.5~2.5msec
Timer Interrupt	TIM3	TIM3 Period: 1msec, Timer Interrupt of 500 msec

Circuit Diagram



Code



```
void EXTI15_10_IRQHandler(void){
  if (is_pending_EXTI(BUTTON_PIN)){

    pwmwidth = 0;
    PWM_duty(PWM_PIN,(( 0.5+(0.11*pwmwidth))/20) );
    clear_pending_EXTI(BUTTON_PIN);
  }
}

void TIM3_IRQHandler(void){

  if(is_UIF(TIM3)){      // Check UIF(update interrupt flag)
    _count++;

    if (_count>500)
    {
      PWM_duty(PWM_PIN,(( 0.5+(0.11*pwmwidth))/20) );
      pwmwidth++;
      if(pwmwidth>18){ pwmwidth=0;}

      _count = 0;
    }
    clear_UIF(TIM3);     // Clear UI flag by writing 0
  }
}
```

No code was written in the while statement of main. Using EXTI, the angle was designed to go to 0 degrees when the button pin input came in.

Using the TIM3 IRQHandler, the duty was controlled every 500 msec with a count of 500 to change the angle of the motor, and when the motor angle exceeded 180, it was initialized to 0 degrees.Result

Result

The RC motor moves by 10 degrees every 0.5 seconds, but when it exceeds 180 degrees, it is reset to 0 degrees. During this process, when a button pin input is received, the motor angle is initialized to 0 degrees.

Demo video: https://youtube.com/shorts/VEphpQGaS0I

Discussion

1. Derive a simple logic to calculate CRR and ARR values to generate x[Hz] and y[%] duty ratio of PWM. How can you read the values of input clock frequency and PSC?

   Duty is the same value as CCR(ARR+1). The CCR value can be obtained by (ARR+1)/2. The input clock frequency can be obtained by multiplying the count frequency by (PSC+1).

2. What is the smallest and highest PWM frequency that can be generated for Q1?

   CCR value, the smallest frequency when PSC is maximum, the highest frequency is output when CCR and PSC are minimum.

problem 2: DC motor

Procedure

Make a simple program that rotates a DC motor that changes the duty ratio from 25% -->75%--> 25% --> and so on.

The rotating speed level changes every 2 seconds.

By pressing the push button (PC13), toggle from Running and stopping the DC motor

1. Use the same project.

- Create a new source file named “**LAB_PWM_DCmotor.c”**
- You need to eliminate the other source file that contains `main()` from the project
  - e.g. Eliminate "“**LAB_PWM_RCmotor.c”** from the project

2. Connect DC motor and DC motor driver.

- PA_0 for the DC motor PWM
- PC_2 for Direction Pin

3. Change DC motor from LOW Speed to HIGH Speed for every 2 seconds

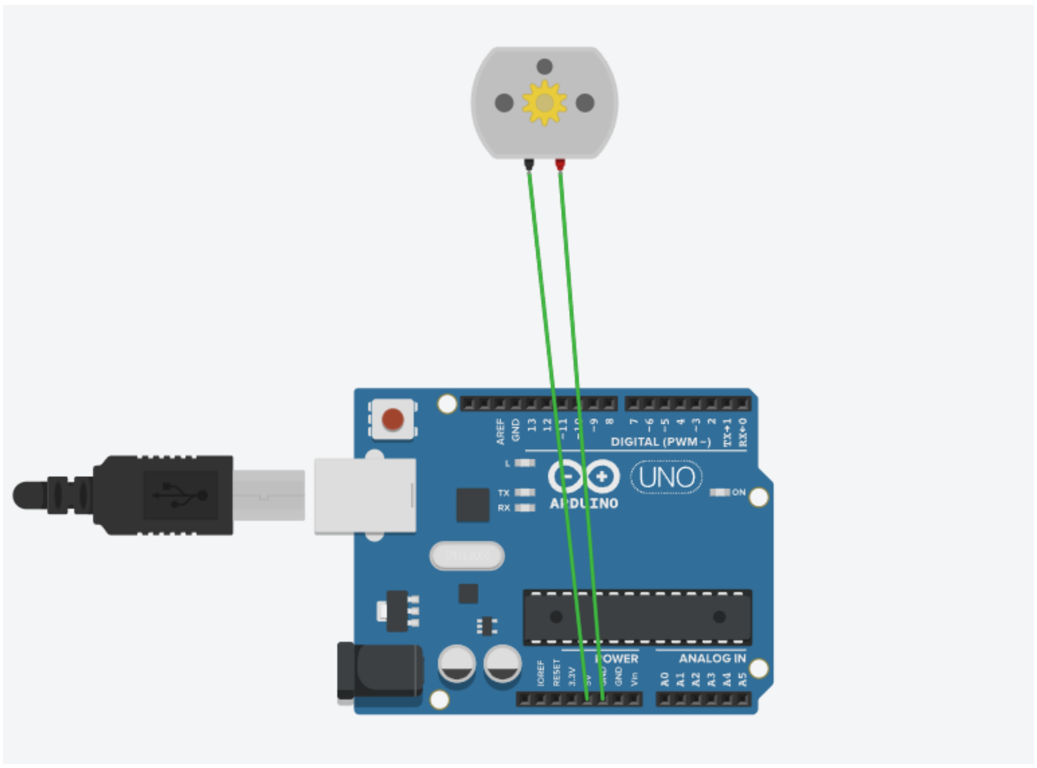
- e.g. 25% -->75%--> 25% --> and so on.

4. When Button is pressed, it should PAUSE or CONTINUE motor run'

Configuration

Function	Port - Pin	Configuration
Button	Digital In (PC13)	Pull-Up
Direction Pin	Digital Out (PC2)	Push-Pull
PWM Pin	AF (PA0)	Push-Pull, Pull-Up, Fast
PWM Timer	TIM2_CH1 (PA0)	TIM2 (PWM) period: **1msec (1kHz)**
Timer Interrupt	TIM3	TIM3 Period: 1msec, Timer Interrupt of 500 msec

Circuit Diagram



Code

```
uint32_t  count = 0;
uint32_t pwmwidth = 0;
uint32_t flag = 1;
```

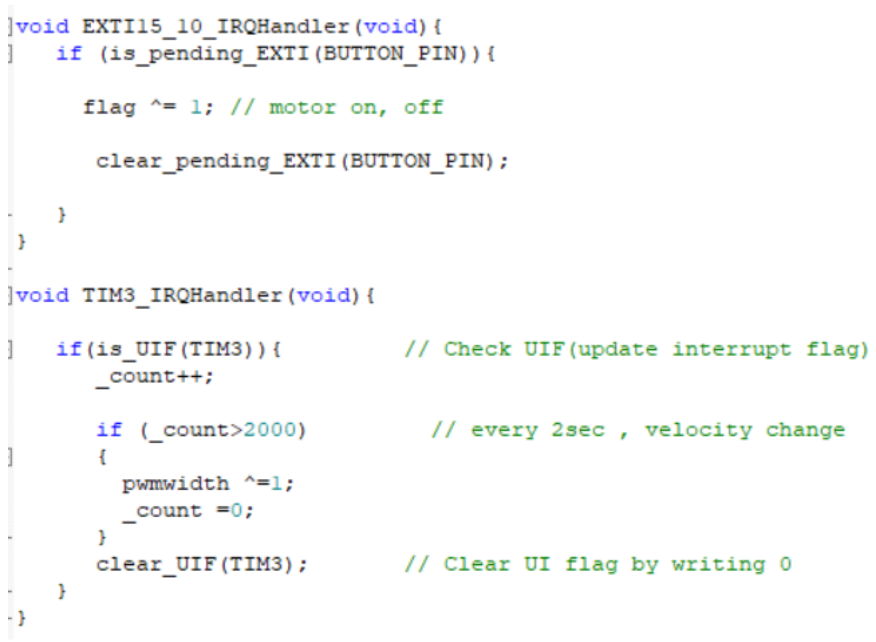
A flag was created to stop the motor when a button input is received.

```
int main(void) {
    // Initialization --------------------------------------------------
    setup();

    // Infinite Loop ---------------------------------------------------
    while(1){

        PWM_duty(PWM_PIN, (0.25+0.5*pwmwidth)*flag);
    }
}
```

In the while statement of main, speed control was implemented by multiplying the pwm duty by pwm, and the on and off of the motor was controlled by multiplying the flag.



```
void EXTI15_10_IRQHandler(void){
   if (is_pending_EXTI(BUTTON_PIN)){

     flag ^= 1; // motor on, off

      clear_pending_EXTI(BUTTON_PIN);

   }
}

void TIM3_IRQHandler(void){

   if(is_UIF(TIM3)){          // Check UIF(update interrupt flag)
      _count++;

      if (_count>2000)          // every 2sec , velocity change
      {
        pwmwidth ^=1;
        _count =0;
      }
      clear_UIF(TIM3);        // Clear UI flag by writing 0
   }
}
```

Using EXTI, when a button input is received, the flag is toggled to turn the motor on and off.

The speed of the motor was controlled to change every 2 seconds using TIM3_IRQHandler.

Result

The speed of the DC motor changes from 25% to 75% every 2 seconds. During this process, when a button pin input is received, the motor remains stopped, and when a button pin input is received once more, the motor starts operating again.

Demo Video: https://youtube.com/shorts/Ya6Zn1ocMfU

Reference

https://github.com/ykkimhgu

https://ykkim.gitbook.io/ec/ec-course/lab/lab-timer-and-pwm

https://ykkim.gitbook.io/ec/ec-course/tutorial/tutorial-dcmotor-motor-driver-connection