LAB: PWM Out – Servo Motor

I. Introduction

In this lab, you are required to create a simple program that control a sevo motor with PWM output. Create HAL drivers for Timer and PWM control and use these APIs for the lab.

Hardware

NUCLEO -F411RE

LEDs x 3, Resistor 330 ohm x 3, breadboard, RC Servo Motor (SG90), DC motor

Software

Keil uVision IDE, CMSIS, EC_HAL

II. Procedure

A. Create EC_HAL functions

Download the source code template

Specific for given Output Pins

Include File	Function	Description
ecGPIO.h, c	// modify functions or add new functions to allow AF mode for TIMx	

ecTIM.h, c	`void TIM_init(TIM_TypeDef *timerx, uint32_t msec);`
	`void TIM_period_us(TIM_TypeDef* timx, uint32_t usec);`
	`void TIM_period_ms(TIM_TypeDef* timx, uint32_t msec);`
	
	`void TIM_INT_init(TIM_TypeDef* timerx, uint32_t msec);`
	`void TIM_INT_enable(TIM_TypeDef* timx);`
	`void TIM_INT_disable(TIM_TypeDef* timx);`
	`uint32_t is_UIF(TIM_TypeDef *TIMx);`
	`void clear_UIF(TIM_TypeDef *TIMx);`
		Initialize timer counter period of usec. For Timerx= TIM1, TIM2, …
		
		Update Interrupt

ecPWM.h,c

```
typedef struct {
  GPIO_TypeDef *port;
  int pin;
  TIM_TypeDef *timer;
  int ch;
} PWM_t;
```

PWM_t is a structure type for initializing GPIO port, pin and the number of timer, channel. You can use this variable as a handler of PWM signal.

```
void PWM_init(PWM_t *pwm, GPIO_TypeDef *port, int pin);
```

Timer Initialization and enable.

Default: 84MHz source clk, 1MHz counter clock, 50% duty, 1msec period

```
void PWM_period_ms(PWM_t *pwm,  uint32_t msec);
```

msec =1~2,000

```
void PWM_period_us(PWM_t *PWM_pin, uint32_t usec);
```

usec=1~1000

```
void PWM_pulsewidth_ms(PWM_t *pwm, float pulse_width_ms);
```

pulsewidth_ms=1~20000

```
void PWM_duty(PWM_t *pwm, float duty);
```

float duty: 0.0~1.0

You can refer to [example code using mbedOS]

B. RC Servo motor: RC Servo Motor (SG90)

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 data sheet of the RC servo motor for detailed specifications.

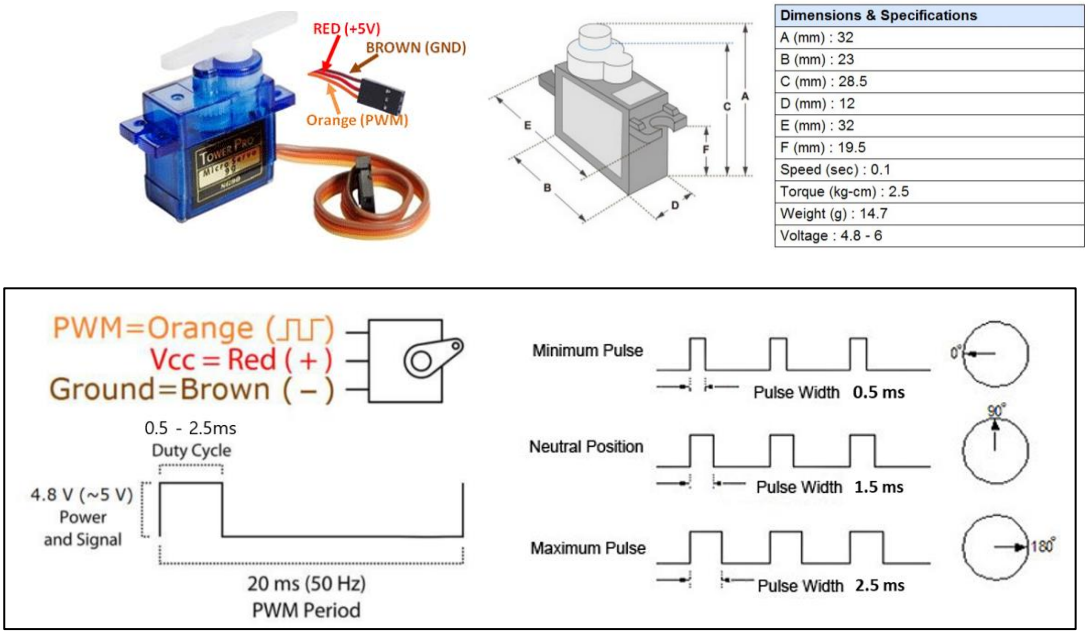

Figure 1. Operation of Servo Motor

Discussion

C. Configuration

Create a new project named as “**LAB_PWM_Servo”.**

Name the source file as “**LAB_PWM_Servo.c”**

You MUST write your name in the top of the source file, inside the comment section.

Configure Input and Output pins

Digital In: Button	Digital Out:
GPIOC, Pin 13 Digital Input Set PULL-UP	PA1 AF output Push-Pull No Pull-up Pull-down Fast
TIMER	**PWM**
TIM2: Counter Period 1kHz	TIM2_CH2: GPIO A, Pin 1 PWM period: 20ms PWM duty ratio: 0.5ms to 2.5ms

D. RC servo motor control

Make a simple program that changes the angle of the RC servo motor by pressing the push button (PC13).

- The button input has to be External Interrupt
- Use Port A Pin 1 as PWM output pin, for TIM2_Ch2.

Increase the angle of RC servor motor from 0° to 180° each time you push the button. After reaching 180°, decrease the angle back to 0°.

- Divide 180° into 10 intervals.

You need to observe how the PWM signal output is generated as input button is pushed, using an oscilloscope. You need to captute the Oscilloscope output in the report.

Embedded Controller

TIMER_interupt_example

```c
#include "stm32f411xe.h"
#include "ecGPIO.h"
#include "ecRCC.h"
#include "ecTIM.h"

uint32_t _count=0;

#define LED_PIN   5

void setup(void);

int main(void) {
  // Initialiization --------------------------------------------------------
  setup();

  // Inifinite Loop ---------------------------------------------------------
  while(1){}
}

// Initialiization
void setup(void)
{
  RCC_PLL_init();                          // System Clock = 84MHz
  GPIO_init(GPIOA, LED_PIN, OUTPUT);    // calls RCC_GPIOA_enable()
  TIM_INT_init(TIM2,1000);   // usec >=100
  TIM_INT_enable(TIM2);
}
void TIM2_IRQHandler(void){
  if(is_pending_TIM(TIM2)){// update interrupt flag
    _count++;
    if (_count >1000) {
      LED_toggle();
      _count=0;}
    clear_pending_TIM(TIM2);// clear by writing 0
  }

}


```

TIMER_PWM_example

```c
#include "stm32f411xe.h"
#include "ecGPIO.h"
#include "ecSysTick.h"
#include "ecRCC.h"
#include "ecTIM.h"
#include "ecPWM.h"

#define LED_PIN   5

PWM_t pwm;
// Initialiization
void setup(void)
{
  RCC_PLL_init();                          // System Clock = 84MHz
  SysTick_init();

  GPIO_init(GPIOA, LED_PIN, EC_ALTE);    // calls RCC_GPIOA_enable()
  GPIO_ospeed(GPIOA, 1, EC_HIGH);
  GPIO_pudr(GPIOA, 1, EC_NONE);

  PWM_init(&pwm, GPIOA, 5);
  PWM_period_ms(&pwm, 1);

}

int main(void) {
  // Initialiization --------------------------------------------------------
  setup();

  // Inifinite Loop ---------------------------------------------------------
  while(1){
    for(int i =0; i<3;i++){
    PWM_duty(&pwm,0.5*i);
    delay_ms(100);
    }
  }
}

```

Discussion

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

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

3) What is the major difference of advanced timer and general purpose timer?

III. Report

You are required to write a consice lab report and submit the program files.

Lab Report: See sample report.

- Write Lab Title, Date, Your name, Introduction
- For each Part show only main() source file. Also, need to include the external circuit diagram if necessary.
- Show your whole code **in the appendix,**
- Answer **Discussion questions**
- You can write Troubleshooting section
- Submit in both PDF and original file (*.docx etc)
- No need to print out. Only the On-Line submission.

Source Code:

- Write description of your functions in github.
- Upload the final version of your library in github.
- Zip all the necessary source files(main.c, ecRCC.h, ecGPIO.h etc...).
- Only the source code files. Do not submit project files etc.