11 주차 결과보고서

9조 팀장 김주송

팀원 강선호

박형주

손봉국

1. 실험 목적

A. 타이머의 종류와 특징을 이해하고 분주를 계산한다.

B. PWM을 이해하고 펄스를 조절한다.

2. 배경 지식

가. 타이머

주기적 시간 처리에 사용하는 디지털 가운터 회로 모듈

펄스폭 계측, 주기적인 interrupt 발생 등에 사용

주파수가 높기 때문에 우선 prescaler를 사용하여 주파수를 낮춘 후 낮아진 주파수로 8, 16비트 등의 카운터 회로를 사용하여 주기를 얻는다.

SYM32 타이머 종류

- SysTick Timer
- Watchdog Timer
- Advanced-control Timer(TIM1, TIM8)
- General-purpose Timer(TIM2~TIM5)
- Basic Timer(TIM6, TIM7)

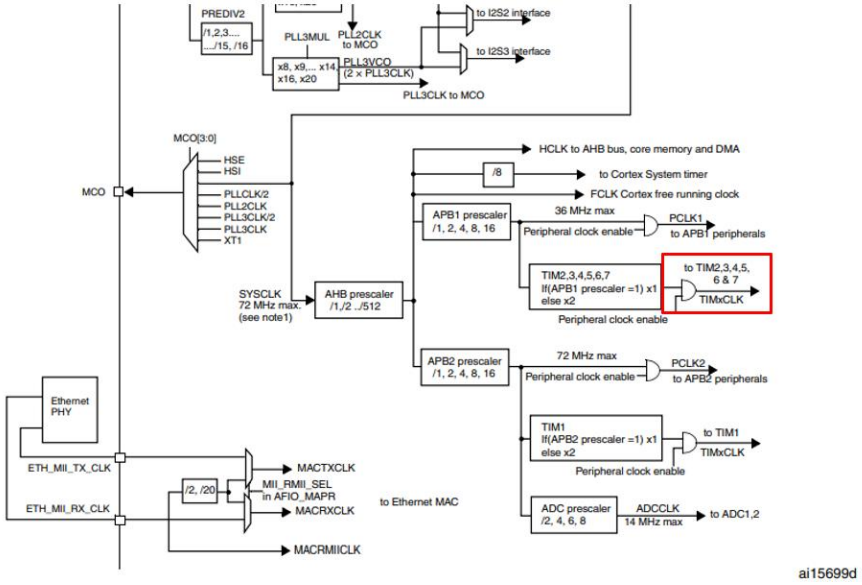
나. 분주 계산

분주란 MCU에서 제공하는 Frequency를 우리가 사용하기 쉬운 값으로 바꾸어 주는 것을 말한다. Counter clock frequency를 1~65536의 값으로 나누기 위해 16-bit programmable prescaler를 사용한다. Period로 몇 번 count 하는지 설정한다.

$$f_{clk} * \frac{1}{prescaler} * \frac{1}{period} = \text{주파수}[Hz]$$

다. Clock frequency

라이브러리에서 설정된 timer clock frequency를 확인한다.

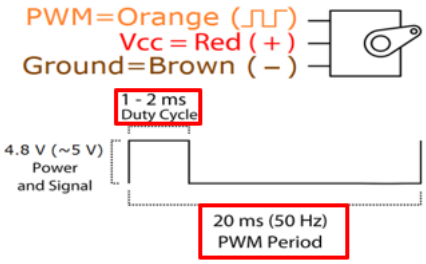


라. PWM(pulse width modulation)

일정한 주기 내에서 Duty ratio를 변화시켜서 평균 전압을 제어하는 방법이다.

Ex) 0~5V의 전력 범위에서 2.5V 전압을 가하고 싶다면, 50% 듀티 사이클을 적용한다.

대부분의 서보모터는 50~1000Hz의 주파수를 요구한다.



Position "0" (1.5 ms pulse) is middle, "90" (~2 ms pulse) is all the way to the right, "-90" (~1 ms pulse) is all the way to the left.

3. 실험 내용

A. TFTLCD에 team 이름, LED 토글 ON/OFF 상태, LED ON/OFF 버튼 생성

B. LED ON 버튼 터치 시 TIM2 interrupt, TIM3 PWM을 활용하여 LED 2개와 서보모터 제어

i. LED: 1초 마다 LED1 Toggle, 5초 마다 LED2 Toggle

ii. 서보모터 : 1초 마다 한쪽 방향으로 조금씩(100) 이동

C. LED OFF 버튼 터치 시 LED Toggle 동작 해제 및 서보모터 동작 반전

i. 서보모터 : 1초 마다 반대쪽 방향으로 조금씩(100) 이동

4. 실험 과정

코드 작성

```
void RCC_Configure(void)
{

    /* Alternate Function IO clock enable */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);

    /* ADC */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);

    /* ADC */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);

    /* LED */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD, ENABLE);

    /* PWM */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);
}
```

먼저 사용할 포트에 clock을 인가한다. 이번 실험에서 led, 서보모터, 타이머를 사용하므로 APB2의 D, B, APB1의 TIM2, TIM3에 clock을 enable한다.

```
void GPIO_Configure(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;
    // PWM
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_Init(GPIOB, &GPIO_InitStructure);
```

```
    // LED 1, LED 2
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2 | GPIO_Pin_3;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_Init(GPIOD, &GPIO_InitStructure);

}
```

사용할 GPIO핀 초기화 한다. Led1,2를 사용하므로 port D의 2,3 pin을 output모드, 50MHz로 설정한다. 서보모터는 port B의 0번 pin을 50MHz, output모드로 초기화한다.

```
void NVIC_Configure(void) {

    NVIC_InitTypeDef NVIC_InitStructure;
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_0);

    /* Enable TIM2 Global Interrupt */
    NVIC_InitStructure.NVIC_IRQChannel = TIM2_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
}
```

타이머를 interrupt 방식으로 제어할 것이므로 TIM2를 우선순위를 지정해 벡터에 저장한다.

```
void TIM2_Configure(void) {
    TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
    TIM_TimeBaseStructure.TIM_Period = 10000;
    TIM_TimeBaseStructure.TIM_Prescaler = 7200;
    TIM_TimeBaseStructure.TIM_ClockDivision = 0;
    TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Down;

    TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);
    TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE);
    TIM_OC3PreloadConfig(TIM2, TIM_OCPreload_Disable);
    TIM_ARRPreloadConfig(TIM2, ENABLE);
    TIM_Cmd(TIM2, ENABLE);
}
```

타이머를 초기화한다. 분주 식에 따라 period와 prescaler를 설정하여, 72MHz system clock에서 1초에 한번 count할 수 있도록 설정한다. TIM_TimeBaseInit으로 TIM2를 초기화

한다. 또한 TIM_ITConfig로 사용할 interrupt를 enable한고, 나머지 함수들로 TIM2를 enable한다.

```
void TIM3_Configure(void) {

    TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
    TIM_OCInitTypeDef TIM_OCInitStructure;
    uint16_t prescale = (uint16_t) (SystemCoreClock / 1000000); // = 72

    TIM_TimeBaseStructure.TIM_Period = 20000;
    TIM_TimeBaseStructure.TIM_Prescaler = prescale;
    TIM_TimeBaseStructure.TIM_ClockDivision = 0;
    TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Down;

    TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
    TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
    TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
    TIM_OCInitStructure.TIM_Pulse = 10000;
    TIM_OC3Init(TIM3, &TIM_OCInitStructure);

    TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);
    TIM_OC3PreloadConfig(TIM3, TIM_OCPreload_Disable);
    TIM_ARRPreloadConfig(TIM3, ENABLE);
    TIM_Cmd(TIM3, ENABLE);
}
```

TIM2와 비슷한 방식으로 TIM3를 초기화하는데 여기에 PWM으로 사용하는 설정을 추가해준다. TIM_OCInitStructure를 이용하여 초기화한다. PWM은 일정한 주기 내에서 Duty ratio를 변화시켜 평균 전압을 제어하는 방법이다. 해당 서보모터는 50Hz주파수를 사용하므로 period와 prescaler를 설정해준다. Prescaler를 72로 설정하였으므로 period = 20ms를 만들어 줘야한다. 그래서 period를 20000으로 설정해준다.

```
void TIM2_IRQHandler(void) {
    if(TIM_GetITStatus(TIM2,TIM_IT_Update) != RESET) {
        if (!LED_ON) {
            GPIO_ResetBits(GPIOD, GPIO_Pin_2);
            GPIO_ResetBits(GPIOD, GPIO_Pin_3);
        }
        else {
            if (LED_1_STATUS) GPIO_ResetBits(GPIOD, GPIO_Pin_2);
            else GPIO_SetBits(GPIOD, GPIO_Pin_2);
            LED_1_STATUS = !LED_1_STATUS;
```

```
            if (LED_2_STATUS) {
                if (LED_2_CNT > 4) {
                    GPIO_ResetBits(GPIOD, GPIO_Pin_3);
                    LED_2_CNT = 0;
                    LED_2_STATUS = !LED_2_STATUS;
                }
            }
            else {
                if (LED_2_CNT > 4) {
                    GPIO_SetBits(GPIOD, GPIO_Pin_3);
                    LED_2_CNT = 0;
                    LED_2_STATUS = !LED_2_STATUS;
                }
            }
            LED_2_CNT++;
        }
        TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
    }
}
```

TIM2의 타이머를 이용해 1초마다 count를 하는 함수를 만들었다. 이 함수는 led1이 1초마다 깜빡이고, led2가 5초마다 깜빡이도록 1초마다 interrupt를 실행하여 led를 키고 끈다.

```
void moveMotor(uint16_t var){
    TIM_OCInitTypeDef TIM_OCInitStructure;
    TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
    TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
    TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
    TIM_OCInitStructure.TIM_Pulse = var;
    TIM_OC3Init(TIM3, &TIM_OCInitStructure);
}
```

모터를 움직이기 위한 함수를 만들었다 pulse를 변수로 주어 pulse에 따라 모터의 각도가 변하도록 만들었다. TIM3 초기화된 값을 참고하여 50Hz에서 각도 변화를 계산하면 아래와 같다.

-90도 : 20000*3.5/100 = 700

0도 : 20000*7.5/100 = 1500

90도 : 20000*11.5/100 = 2300

```c
void printTeamName(void) {
    char * str = "TEAM09\0";
    for (int i = 0; i < 6; i++) {
        LCD_ShowChar(0x50 + i * 10, 0x35, str[i], 16, BLACK, WHITE);
    }
}

void drawLEDStatus(int * LED_ON) {
    if (*LED_ON) LCD_ShowString(0x50, 0x55, "ON   ", RED, WHITE);
    else LCD_ShowString(0x50, 0x55, "OFF", RED, WHITE);
}

void drawLEDButton() {
    LCD_ShowString(0x60, 0x80, "BUT", RED, WHITE);
    LCD_DrawRectangle(0x50, 0x65, 0x80, 0x95);
}

int checkLCDButtonTouch(int x, int y) {
    if (x >= 0x50 && x <= 0x80) {
        if (y >= 0x65 && y <= 0x95) {
            return 1;
        }
    }
    return 0;
}
```

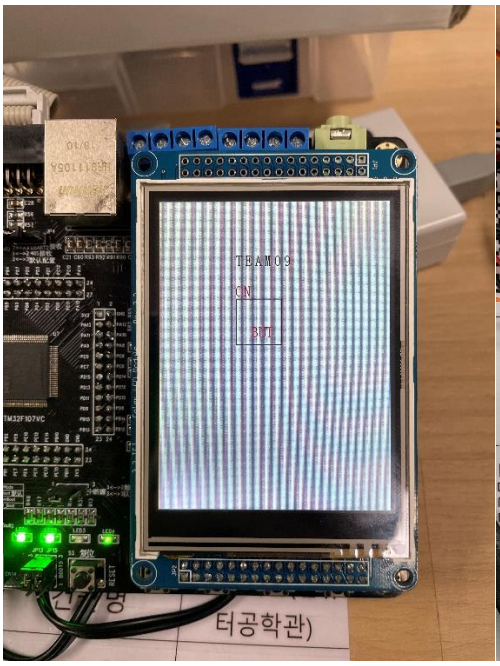
10주차 수업에서 LCD화면에 버튼을 만들고 버튼의 터치 여부에 따라 값이 변경되는 함수들을 만들었다.

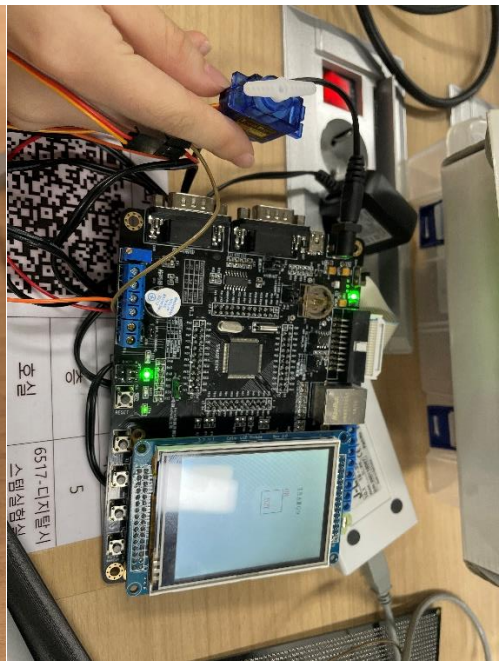
```c
int main(void)
{
    SystemInit();
    RCC_Configure();
    GPIO_Configure();
    NVIC_Configure();
    TIM2_Configure();
    TIM3_Configure();

    LCD_Init();
    Touch_Configuration();
    Touch_Adjust();
    LCD_Clear(WHITE);

    uint16_t x;
```

```
    uint16_t y;
    uint16_t convert_x;
    uint16_t convert_y;
    uint16_t val[17] = {700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500,
1600, 1700, 1800, 1900, 2000, 2100, 2200, 2300};
    int idx = 0;

    while(1) {
      moveMotor(val[idx]);
      if (motorflag){
        idx++;
        idx %= 17;}
      else{
        idx--;
        if (idx < 0) idx = 16;}

        printTeamName();
        drawLEDButton();
        drawLEDStatus(&LED_ON);
        Touch_GetXY(&x, &y, 0);
        Convert_Pos(x, y, &convert_x, &convert_y);
        if (T_INT != 1) {
            if (checkLCDButtonTouch(convert_x, convert_y)) {
                LED_ON = !LED_ON;
                motorflag = !motorflag;
            }
        }
        Delay();
    }
    return 0;
}
```

메인 함수를 작성한다. Val array를 이용하여 700~2300의 값을 100단위로 하여 이동할 수 있도록 저장했다. 또한 moterflag를 이용하여 모터의 방향을 결정해 버튼의 ON/OFF모드에 따라 모터의 방향이 결정될 수 있게 만들었다.

5. 실험 결과





이번 실험을 통해 STM32보드의 타이머를 사용한 interrupt와 PWM 신호를 이용하여 서보모터를 제어했다. 타이머에서 clock을 통해 시간을 측정하고 interrupt하는 것은 앞선 수업들을 참고하여 어렵지 않게 수행했다. 하지만 PWM에서 pulse가 무엇인지 pulse를 통해 각도를 정하는 건 어떤 식을 이용하는 것인지 몰라 어려움을 겪었다. 이후 pulse는 period*duty cycle이라는 것을 알고 각도를 결정해 모터를 작동할 수 있었다.