2016년 2학기
컴퓨터하드웨어 실험 보고서
7조 11주차

한성희 이호욱 한경수

Table Of Contents

layout: post author: 한성희, 이호욱, 한경수 date: 2016-11-03 17:48 +0900 tags:

comments: true

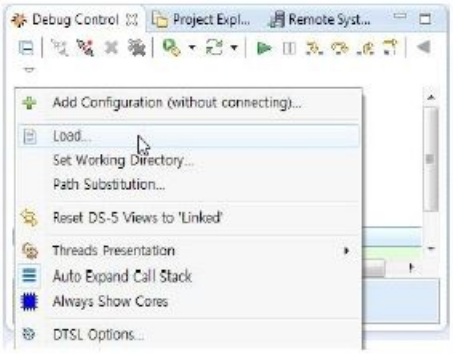
개요

Timer 의 원리와 동작 방식을 이하고 Timer 의 주기에 따라 Interrupt 를 발생시켜 LED를 제어한다.

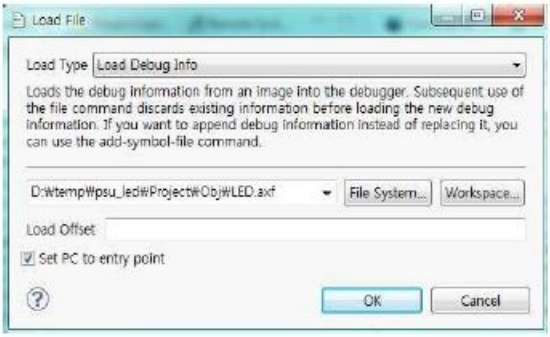
기본 개념 (배경 지식)

Flash Debugging

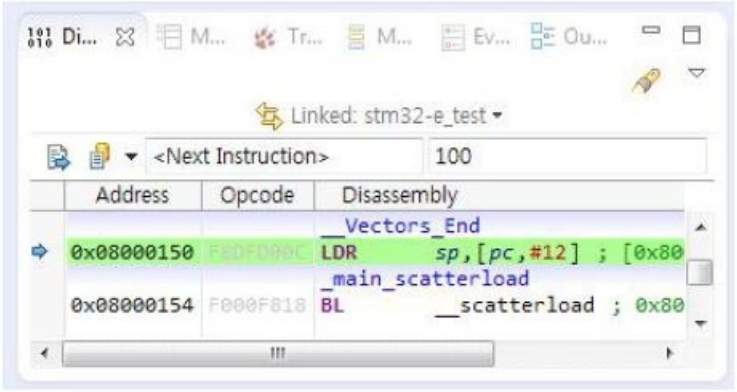
flash 에 load 되어있는 프로그래밍 디버깅



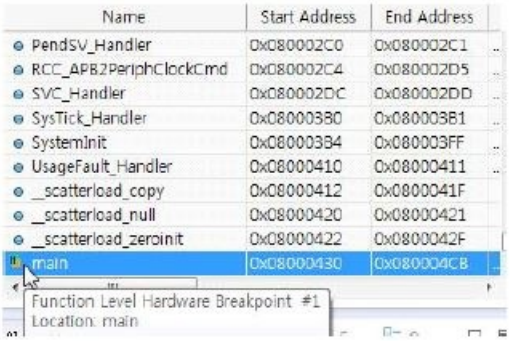
- connet only 로 board 와 연결
- Target 실행을 멈춘다. (F9, Interrupt 아이콘 클릭)
- Debugg control view -> Load 메뉴 선택



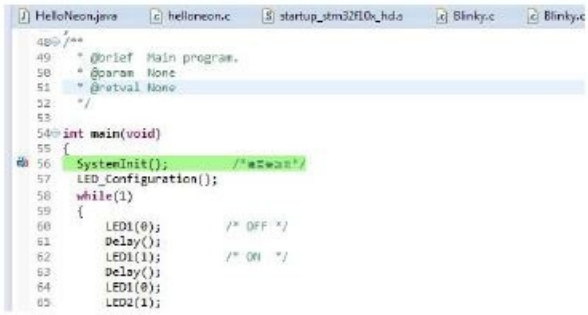
- Load file 창에서 load type 을 'load debug info' 로 설정
- Symbol 정보를 가지고 있는 실행 파일(.axf file)로 지정



다음과 같이 코드의 시작 위치를 볼 수 있다.



사용자의 main 함수까지 실행하기 위해 function view 에서 main 함수를 찾아 breakpoint 를 설정하고 run 시킨다. 단, 코드가 ROM 타입의 메모리에서 실행 중인 경우 H/W breakpoint 를 설정할 필요가 있다.



실행버튼 클릭시, 다음과 같이 target 이 main 함수에서 멈춘 것을 확인할 수 있다.

Timer

Timer 란,

Timer 는 일련의 사건이나 process를 제어하는 데 사용된다. Timer 는 스톱워치와 달리 특정한 시간 간격으로부터 숫자를 내려 센다.
Timer 는 기계적, 전자기계적, 전기적, S/W 적 방식을 취하기도 하며 현대의 모든 컴퓨터들은 하나 이상의 digital Timer 를 포함하기도 한다.

우리가 쓰는 STM32F10x board 는 SysTick Timer 1개, Watchdog Timer 2개, Advanced-control Timer 1개, General-purpose Timer 4개가 있다.

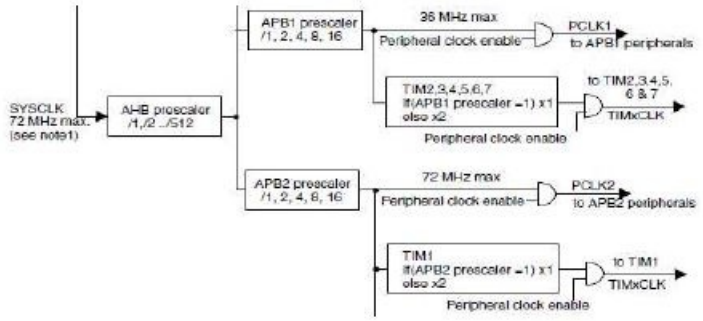
```
- TIM1 : Advanced-control Timer
- TIM2 ~ TIM5 : General-purpose Timer
- TIM6, TIM7 : Basic Timer
```

Table 4. Timer feature comparison

Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/compare channels	Complementary outputs
TIM1	16-bit	Up, down, up/down	Any integer between 1 and 65536	Yes	4	Yes
TIMx (TIM2, TIM3, TIM4, TIM5)	16-bit	Up, down, up/down	Any integer between 1 and 65536	Yes	4	No
TIM6, TIM7	16-bit	Up	Any integer between 1 and 65536	Yes	0	No

- SysTick : 일반 Timer 와 달리 오직 정주기를 만드는 용도로 사용되며 Cortex-M3 core 에 내장되어있다.
- Watchdog : 컴퓨터의 오작동을 감지하고 복구하기 위해 사용된다. 정상작동 중인 컴퓨터는 시간이 경과하거나 Time out 되는 것을 막기 위해, 정기적으로 Watchdog Timer 를 재가동시킨다.
- General-purpose : 기본적으로 16 bits auto-reload up/down 카운터로 외부 event 를 카운트하고 시간을 재기 위해 사용된다.
- Advanced-control : General-purpose Timer 와 거의 유사하나 complementary output 기능을 지원하여 PWM(Pulse Width Modulation : 펄스 폭 변조)의 기능을 좀 더 보완한 것이다.
- Basic : 주로 DAC trigger generation 에 사용된다.

Timer Clock



TIM1 은 APB2 bus 로 부터 clock 을 받는다. 이때 APB2 prescaler 가 1이 아니면 clock 에 2를 곱한 값을 받고 1이면 그대로 받는다.

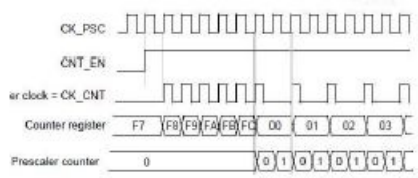
TIM2 ~ TIM7 은 APB1 bus 로부터 clock 을 받는다. TIM1 과 마찬가지로 APB1 prescalar 가 1이 아니면 clock 에 2를 곱한 값을 받고 1이면 그대로 받는다.

Timer Cycle

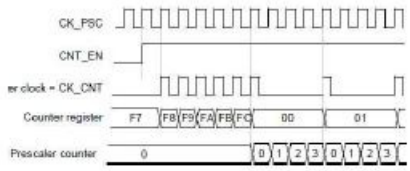
$$\frac{1}{f_{CLK}} \times prescaler \times period$$

Timer 주기를 구하는 식은 다음과 같다.

- prescaler 값이 1일 때



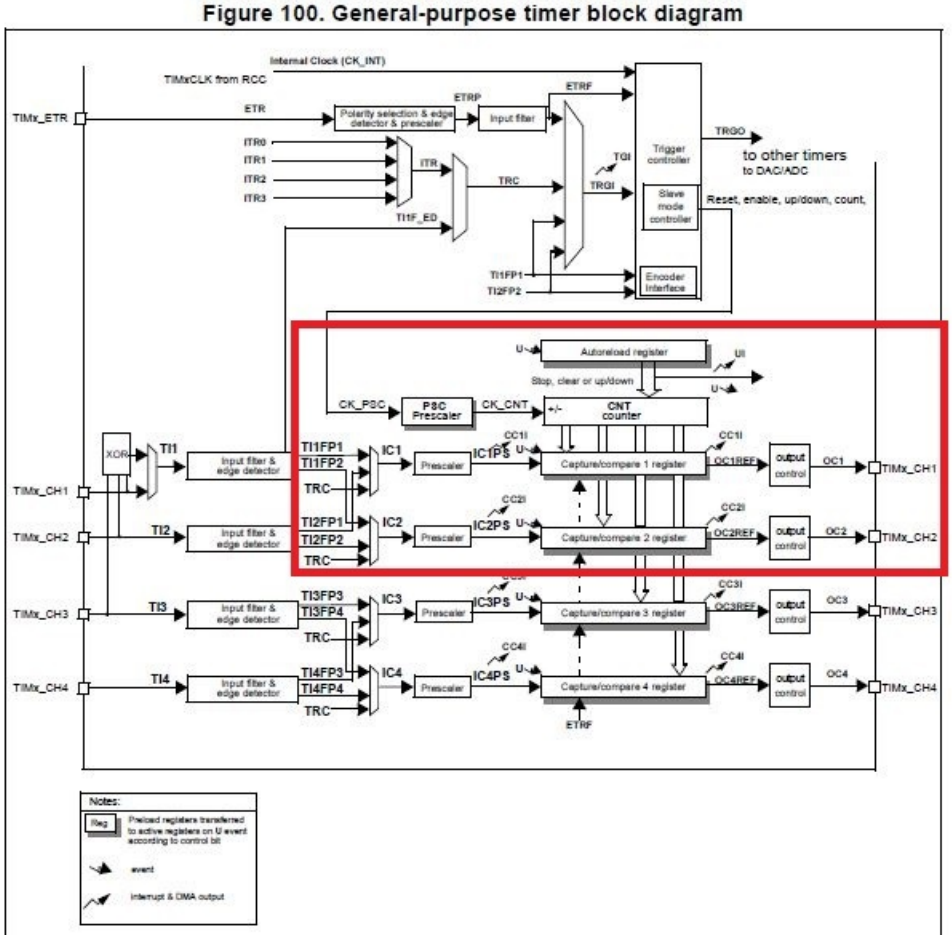
- prescaler 값이 3일 때



prescaler 와 period 의 값은 0 부터 시작하므로 주고자 하는 값에 -1을 해줘야 한다.

Timer Register

Figure 100. General-purpose timer block diagram



General-purpose timer block diagram 은 다음과 같다.

- TIMx_CNT (TIMx counter) : counter 값이 들어있는 register.

- TIMx_PSC (TIMx prescaler) : clock 을 분주하기 위한 값. clock 을 (PSC register 값 + 1) 로 나눈다.

- TIMx_ARR (TIMx auto-reload register) : counter 가 카운트할 최대 값. 최대 값까지 카운트 한 뒤 0으로 돌아간다.

Timer clock 이 PSC 의 값에 의해 분주되어 CNT 값이 증가하며 CNT 값이 ARR 값과 같아지면 CNT 값이 reload 된다.

실험 방법

미션은 3가지다.

1. 초시계를 1초 간격으로 LCD 에 출력
2. 3초 동안의 버튼 입력 횟수를 3초 간격으로 LCD 에 출력.
3. 5초 동안의 조도센서 최댓값을 5초 간격으로 LCD 에 출력.

10주차와 마찬가지로 TFT-LCD 를 보드에 결합시킨다.

그리고 1초에 한번씩 호출되는 타이머를 써야하는데 이를 세팅하는 코드는 다음과 같다. (초 조절하는 방법은 6주차 참고)

```
void init_Timer2() {
  NVIC_InitTypeDef NVIC_InitStructure; // for interreupt
  TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure; // timerbase...

  /* TIM2 Clock Enable */
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);

  /* 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);

  /* TIM2 Initialize */
  TIM_TimeBaseStructure.TIM_Period = 1200;
  TIM_TimeBaseStructure.TIM_Prescaler = 60000;
  //계산방법 : 1/72mhz * 1200 * 60000
  TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
  TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);
```

```
22
23    /* TIM2 Enale */
24    TIM_ARRPreloadConfig(TIM2, ENABLE);
25    TIM_Cmd(TIM2, ENABLE);
26    TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE); // interrupt enable
27  }
28
```

위 코드를 통해 1초에 한번씩 호출되는 함수가 TIM2_IRQ 로 설정이 된다. (1/72Mhz) * 1200 * 60000 = 1

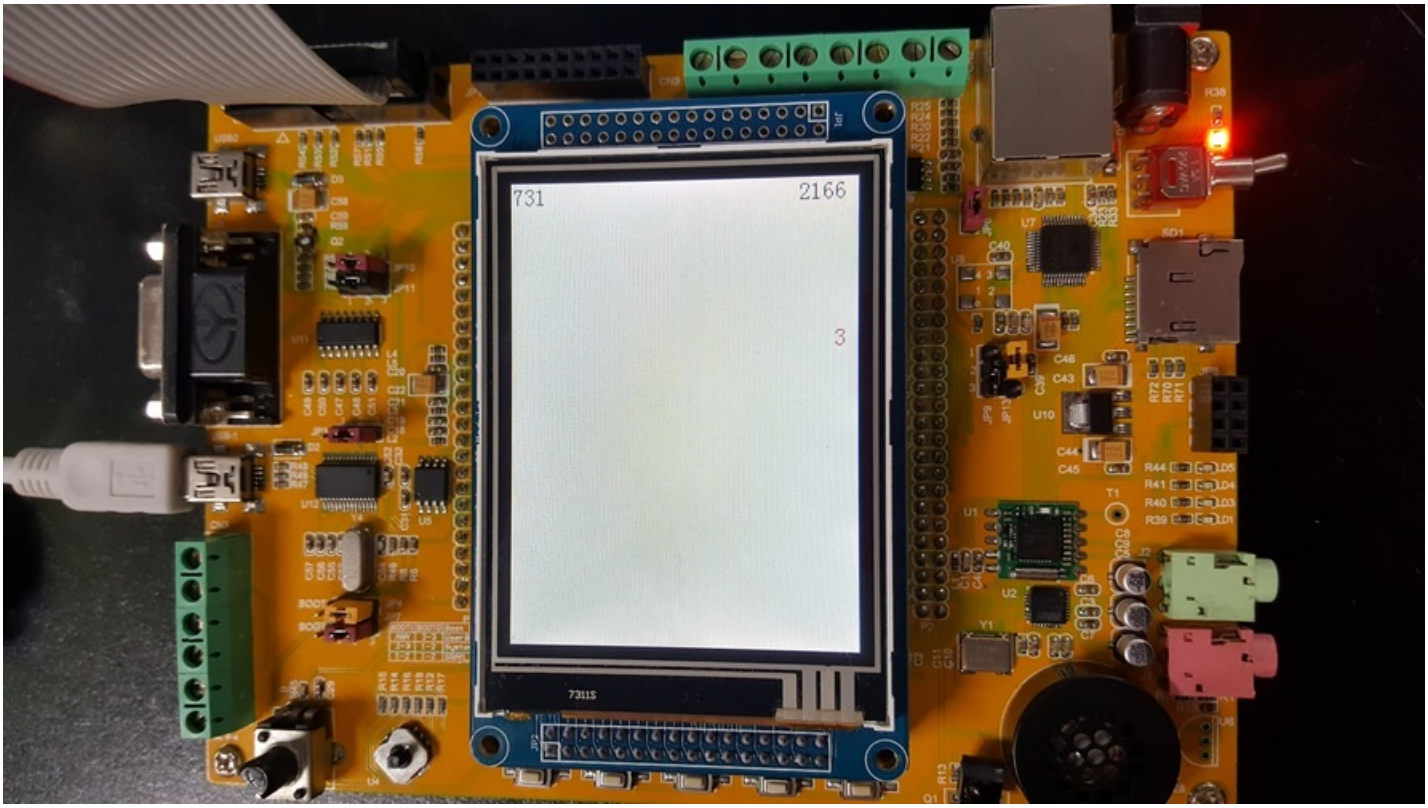
```
1   void TIM2_IRQHandler(void) {
2     t1++;
3     t2++;
4     t3++;
5     if (button_adder >= 4)
6       button_adder = 4;
7     button_count += button_adder;
8     button_adder = 0;
9     TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
10    //Clears the TIMx's interrupt pending bits.
11  }
```

인터럽트에서는 오래걸리는 일을 하면 안되기 때문에, 단순히 초를 카운트하는 기능만 넣는다. 그리고 버튼 자체의 문제점때문에 여러번 입력이 되는 경우엔 1초에 최대 4번만 누르게 한다.

main 에서의 while(1) 를 통해 인터럽트에서 카운팅한 초 값을 이용해서 TFT-LCD 에 우리가 얻은 값을 표시한다. 코드는 아래와 같다.

```
1   while (1) {
2     sprintf(sec, "%d", t1);
3     LCD_ShowString(1, 1, sec, BLACK, WHITE);
4     if (t2 % 4 == 0) {
5       col++;
6       LCD_ShowNum(200, 100, button_count, 4, col % 2 == 0 ? BLUE : RED,
7           WHITE);
8       button_count = 0;
9       t2 = 1;
10    }
11
12    if (t3 % 6 == 0) {
13
14      LCD_ShowNum(200, 1, jodo_max, 4, BLACK, WHITE);
15      jodo_max = 0;
16      t3 = 1;
17    }
18  }
```

실험 결과



그림과 같이 왼쪽 상단에는 초시계가 카운팅 되고, 오른쪽 상단엔 조도센서의 값을 가져와서 값을 출력하게 되었다.

그리고 3초기준으로 초기화는 버튼클릭 카운터가 작동되는 모습을 볼 수 있다.

결론

stm32 보드의 내부 타이머를 이용하는 방법을 알게 되었다. clk 수와 period 및 prescaler 를 이용하여 우리가 원하는 타이머의 주기를 쓸 수 있게 됐다.

처음엔 인터럽트 내부에서 오래걸리는 (I/O 등)작업을 하면서 문제를 해결했는데, 이럴 경우 인터럽트 순서등이 꼬일 수 있다는 것을 경험했고, 그로 인해 interrupt 함수는 간결하게 큰일을 하지 않도록 짜야했다.

전체 코드

```

#include "stm32f10x.h"
#include "stm32f10x_gpio.h"
#include "stm32f10x_rcc.h" // apb2를 제어하기 위함...
#include "stm32f10x_tim.h" // timer을 제어하기 위한 함수가 모여잇음
#include "misc.h"
#include "stm32f10x_it.h"
#include <lcd.h>

uint16_t jodo_max, button_count, button_adder;
int t1, t2 = 1, t3 = 1;

void TIM2_IRQHandler(void) {
  t1++;
  t2++;
  t3++;
  if (button_adder >= 4)
    button_adder = 4;
  button_count += button_adder;
  button_adder = 0;
  TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
  //Clears the TIMx's interrupt pending bits.
}

void EXTI15_10_IRQHandler(void) {
  if (EXTI_GetITStatus(EXTI_Line11) != RESET)
    button_adder++;
  EXTI_ClearITPendingBit(EXTI_Line11);
}

void init_Timer2() {
  NVIC_InitTypeDef NVIC_InitStructure; // for interreupt
```

```
  TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure; // timerbase...

  /* TIM2 Clock Enable */
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);

  /* 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);

  /* TIM2 Initialize */
  TIM_TimeBaseStructure.TIM_Period = 1200;
  TIM_TimeBaseStructure.TIM_Prescaler = 60000;
  //계산방법 : 1/72mhz * 1200 * 60000
  TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
  TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);

  /* TIM2 Enale */
  TIM_ARRPreloadConfig(TIM2, ENABLE);
  TIM_Cmd(TIM2, ENABLE);
  TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE); // interrupt enable
}

void delay(int i) {
  int j;
  for (j = 0; j <= i * 100000; j++)
    ;
}

void set_ENABLE(void) {
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);         // interrupt
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOE, ENABLE);     // RCC GPIO E
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);     // RCC GPIO C
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD, ENABLE);     // RCC GPIO D
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);         // ADC1
}

void set_PC1(void) {
  GPIO_InitTypeDef GPIO_InitStructure;
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_10MHz;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
  GPIO_Init(GPIOC, &GPIO_InitStructure);
}

void set_ADC(void) {
  ADC_InitTypeDef ADC_InitStructure;
  ADC_DeInit(ADC1);
  ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
  ADC_InitStructure.ADC_ScanConvMode = ENABLE;
  ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
  ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
  ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
  ADC_InitStructure.ADC_NbrOfChannel = 1;
  ADC_RegularChannelConfig(ADC1, ADC_Channel_11, 1,
      ADC_SampleTime_239Cycles5);
  ADC_Init(ADC1, &ADC_InitStructure);
}

void set_NVIC(void) {
  NVIC_InitTypeDef NVIC_InitStructure;
  NVIC_InitStructure.NVIC_IRQChannel = ADC1_2_IRQn;
  NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
  NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStructure);
  ADC_ITConfig(ADC1, ADC_IT_EOC, ENABLE);
  ADC_Cmd(ADC1, ENABLE);
}

void ADC_start(void) {
  ADC_ResetCalibration(ADC1);
  while (ADC_GetResetCalibrationStatus(ADC1))
    ;
  ADC_StartCalibration(ADC1);
  while (ADC_GetCalibrationStatus(ADC1))
    ;
  ADC_SoftwareStartConvCmd(ADC1, ENABLE);
}

void ADC1_2_IRQHandler(void) {
  uint16_t input;
  uint16_t jodo;

```

```
  input = ADC_GetConversionValue(ADC1);
  jodo = (double) input;

  if (jodo_max < jodo) {
    jodo_max = jodo;
  }
  ADC_ClearITPendingBit(ADC1, ADC_IT_EOC);
}

void EXTI11_Config(void) {
  GPIO_InitTypeDef GPIO_InitStructure;
  EXTI_InitTypeDef EXTI_InitStructure;
  NVIC_InitTypeDef NVIC_InitStructure;
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD, ENABLE);

  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPD;
  GPIO_Init(GPIOD, &GPIO_InitStructure);

  RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);

  GPIO_EXTILineConfig(GPIO_PortSourceGPIOD, GPIO_PinSource11);

  EXTI_InitStructure.EXTI_Line = EXTI_Line11;
  EXTI_InitStructure.EXTI_LineCmd = ENABLE;
  EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
  EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;
  EXTI_Init(&EXTI_InitStructure);

  NVIC_InitStructure.NVIC_IRQChannel = EXTI15_10_IRQn;
  NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x00;
  NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x01;
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStructure);
}

int main() {
  char sec[100];
  int col = 0;
  SystemInit();
  set_ENABLE();
  set_PC1();
  set_ADC();
  set_NVIC();
  LCD_Init();
  init_Timer2();
  LCD_Clear(WHITE);
  ADC_start();
  EXTI11_Config();

  while (1) {
    sprintf(sec, "%d", t1);
    LCD_ShowString(1, 1, sec, BLACK, WHITE);
    if (t2 % 4 == 0) {
      col++;
      LCD_ShowNum(200, 100, button_count, 4, col % 2 == 0 ? BLUE : RED,
          WHITE);
      button_count = 0;
      t2 = 1;
    }

    if (t3 % 6 == 0) {

      LCD_ShowNum(200, 1, jodo_max, 4, BLACK, WHITE);
      jodo_max = 0;
      t3 = 1;
    }
  }
}

/*
   flash load C:\Users\USER\Desktop\week11\Debug\week11.axf
 *
```