임베디드 시스템 설계 및 실험 보고서
10 주차 실험_TFT LCD 와 타이머

분반 : 001 분반
교수님 : 정 상화 교수님
조교님 : 유 동화 조교님
실험일 : 2019-11-04
제출일 : 2019-11-11

00. 목차

7 조
장 수현
박 창조
임 다영
이 힘찬

01. 실험 목적

- TFT LCD 의 이해 및 제어
- 타이머의 이해 및 제어
- 오실로스코프를 사용하여 정확한 타이밍이 출력되는지 확인

02. 실험 과제

- TIMER 2 를 사용하여 10ms 마다 LED 1 번 깜빡이게 하기
- TIMER 2 로 인가되는 CLOCK 설정
- LCD 화면에 분, 초, 1/10 초, 1/100 초 단위 표시
- Start, Stop, Reset 버튼 LCD 에 구현 및 이용하여 동작

03. 실험 준비

3.1 실험에 필요한 기초지식

3.1.1 TFT-LCD

- 초박막 액정표시장치(Thin Film Transistor Liquid Crystal Display)의 약자
- 액체와 고체의 중간 특성을 가진 액정의 상태 변화와 편광판의 편광 성질 이용
- 통과하는 빛의 양을 조절함으로써 정보를 표시하는 첨단 디지털 디스플레이
- TFT-LCD 는 Color Filter 와 TFT 가 형성된 두 장의 유리기판과 그 사이에 주입된 액정(Liquid Crystal), 광원(Back Light Unit)으로 구성됨.

3.1.2 TIMER 의 종류

- SysTick Timer: 모든 Cortex-M Core 가 포함하고있는 타이머로 일정시간마다 인터럽트를 발생시켜 Delay 함수에 사용하거나 코드의 수행 시간을 측정하는 용도로 사용하는 타이머.

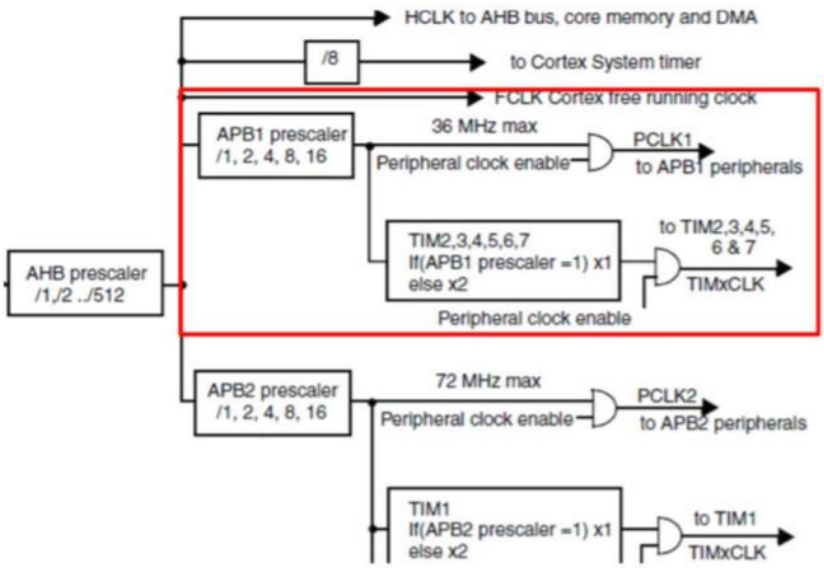
-IWDG, WWDG: 일반적인 WDG 는 주기적인 타이머로 동작되며 완료되기 전에 갱신하지 않으면 시스템을 리셋 하는 기능을 갖고 있다. 즉, 시스템이 일시적인 결함으로 정상적으로 수행이 되지 않을 경우 WDG 를 갱신하 는 루틴을 수행하지 못하고 타이머 의 주기에 도달하 게 되면 시스템을 리셋하여 시스템 초기의 정상적인 수행이 가능한 상태로 복구한다.

- Basic Timer : 단순한 형태의 16bit 타이머로서 Input / Output 핀 없이 순수 time base generator 동작 수행. (Tim6,7)

- General purpose timer: 16bit Auto-reload up/down 카운터와 16 비트 Prescal 를 기반으로하며, 4 개의 채널을 가지고 있음. Input Capture 기능을 통해 신호의 타이밍을 계측 할 수 있으며, 이를 통해 신호의 주기를 파악할 수 있다.(Tim2,3,4,5)

- Advanced control timer: 기본적으로 General Purpose 타이머의 기능은 모두 제공하며, 모터 제어나 Digital Power Conversion 에 적합한 Complementary signal, Dead time insertion, Emergency shut-down input 등의 기능을 제공함. (Tim1,8)

3.1.3 Timer 회로 및 동작 방법



- APB1 버스에 연결되어 있음
-72MHz 의 Clock 이 공급됨

3.1.4 분주 계산 공식

$$\frac{1}{f_{CLK}} \times prescaler \times period \quad \Rightarrow \quad \frac{1}{24MHz} \times 24 \times 1000 = 0.001[s]$$

3.1.5 오실로스코프 사용법

-측정

(1). [Meas] 버튼을 눌러 측정 메뉴가 표시되도록 한다.

(2). [소스] 소프트 키를 눌러 채널을 선택한다 (선을 연결한 채널).

(3). [유형] 소프트 키를 눌러 측정하고 싶은 단위를 선택한다.

-Trigger

(1). 전면 패널의 [Trigger] 버튼을 눌러 Trigger 설정 메뉴가 표시되도록 한다.

(2). [소스] 소프트 키로 채널을 선택한 후 [트리거] 소프트 키를 눌러 “에지” 를 선택한다.

(3). [슬로프] 소프트 키를 눌러 상승, 하강, 교대 중 원하는 에지를 선택한다.

3.2 실험 진행 시 주의사항

- SystemInit() 사용
- 매 1/100 초가 갱신 될 때 마다 LCD_ShowString() 수행
- ISR 에서 상태 변수를 바꾸고 main 에서 수행하면 Delay 발생
- 예외적으로 ISR 안에서 각 초들의 숫자 부분 갱신을 수행

3.3 환경 설정

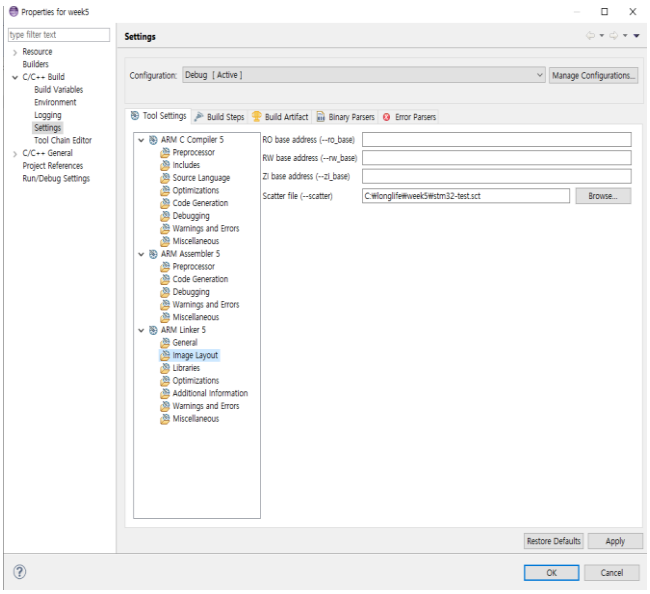
실험을 진행하기 전 아래의 프로젝트 생성 및 기본 환경 설정을 해주어야 한다.

3.3.1 프로젝트 생성

C 언어로 소스코드를 작성하기 때문에 C++프로젝트가 아닌 C 프로젝트를 생성해준다. Project type 은 Bare-metal Executable -〉 Empty Project 로, Toolchains 은 Arm Compiler 5 로 설정한다.

3.3.2 프로젝트 Properties-Settings

4 주차 실험과 동일하게 스캐터 파일을 이용한다.



3.3.3 보드 연결

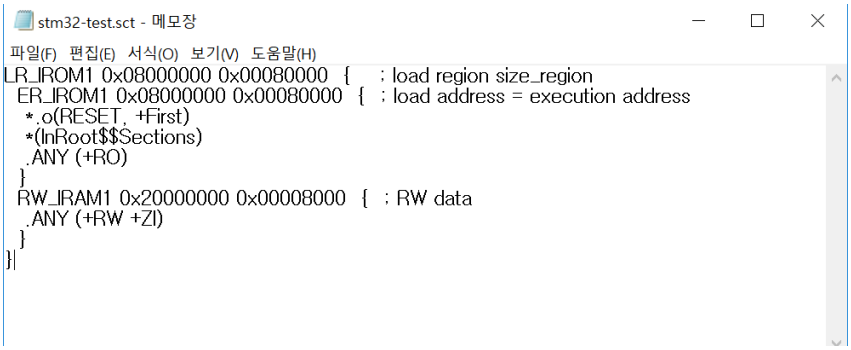
보드 연결 시에는 반드시 연결 순서를 지키고, 규격에 맞는 전원선을 사용해야한다.

연결 순서 : 보드와 Dstream JTAG 연결 -〉 보드전원선 연결(보드 전원은 OFF) -〉 Dstream 전원 연결 및 ON -〉 Dstream Status LED 점등 확인 후 보드 전원 ON -〉 Dstream Target LED 점등 확인 후 DS-5 에서 'connect target'

3.3.4 데이터 베이스 설정

Dstream 를 USB 로 컴퓨터에 연결하고 Debug Hardware config 에서 보드를 connect 한다. 보드 연결 후에는 Auto Configure 를 클릭하여 설정을 진행하고, rvc 파일로 저장한다. rvc 파일 저장 경로에 한글이 들어가지 않도록 주의한다.

3.3.5 Scatter file 수정



스캐터 파일의 메모리 범위를 위와 같이 수정한다.

3.3.6 Optimization level 설정

실험에서 사용하는 라이브러리의 용량이 크기 때문에 최적화 레벨을 high 로 설정한다. (O2 로 설정하되 필요시 O3 설정 가능)

Project -〉 Option -〉 Settings -〉 Optimization level high

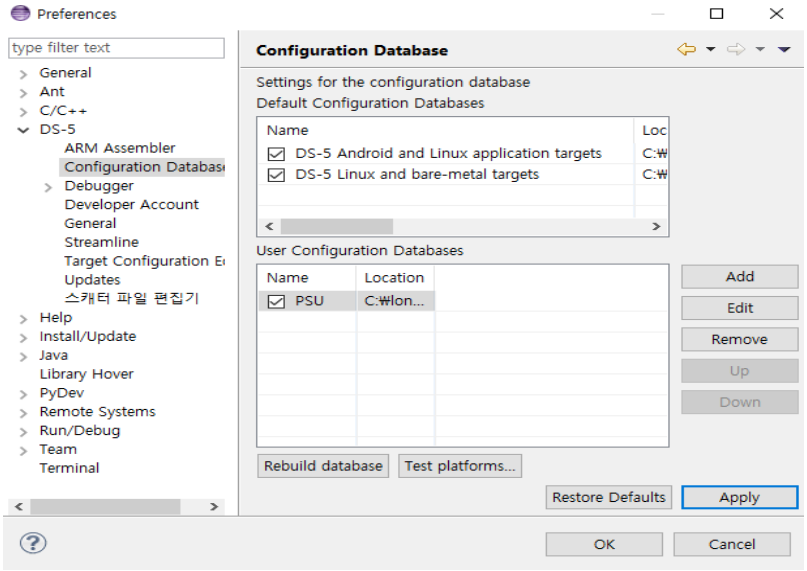
3.3.7 cdbimporter

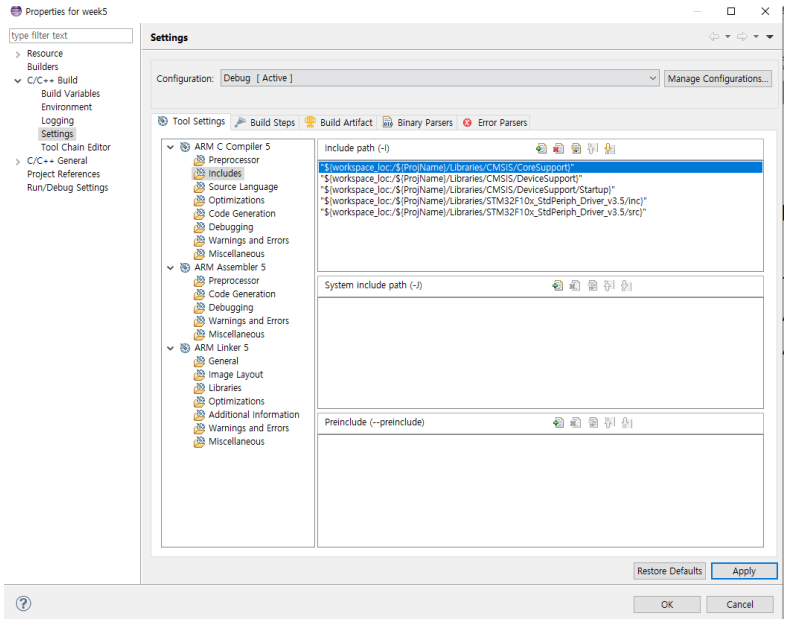
DS-5 Command Prompt 에서 RVC 파일이 있는 폴더로 이동 후 아래와 같이 명령문을 작성한다.

3.3.8 데이터 베이스 등록 및 라이브러리 추가

아래와 같이 데이터 베이스 등록 후 프로젝트 폴더에 라이브러리를 추가한다.

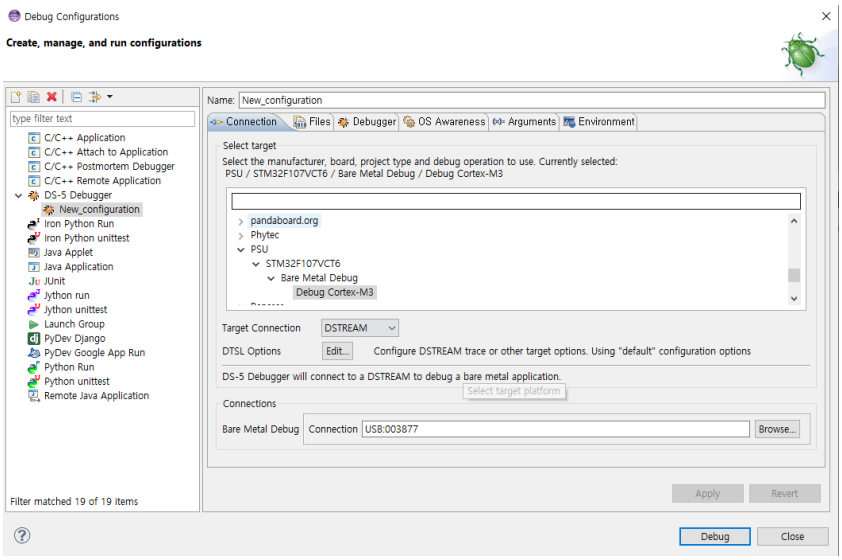
프로젝트 파일이 있는 경로에 font.h, lcd.c, lcd.h, touch.c, touch.h 를 추가한다.

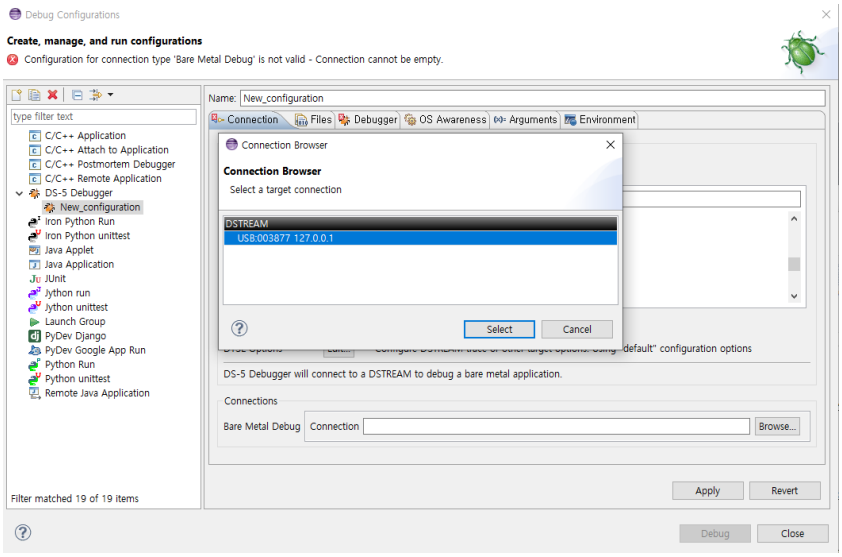


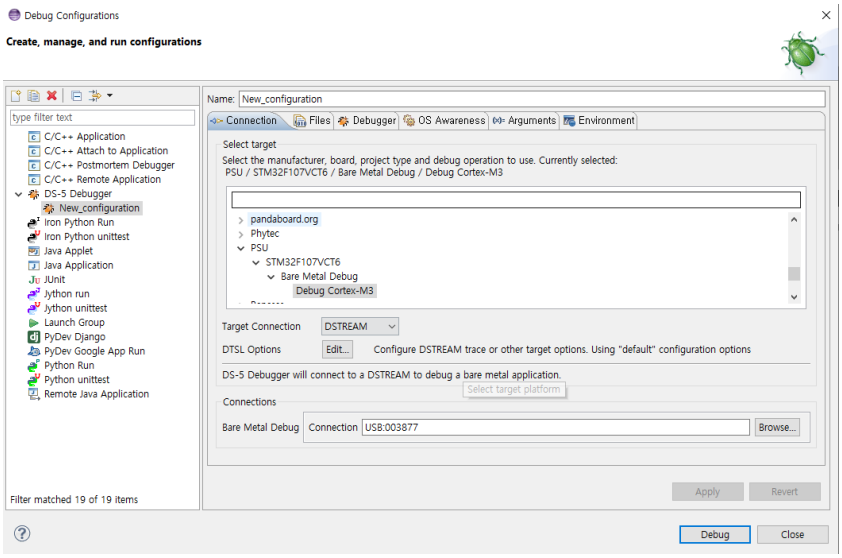


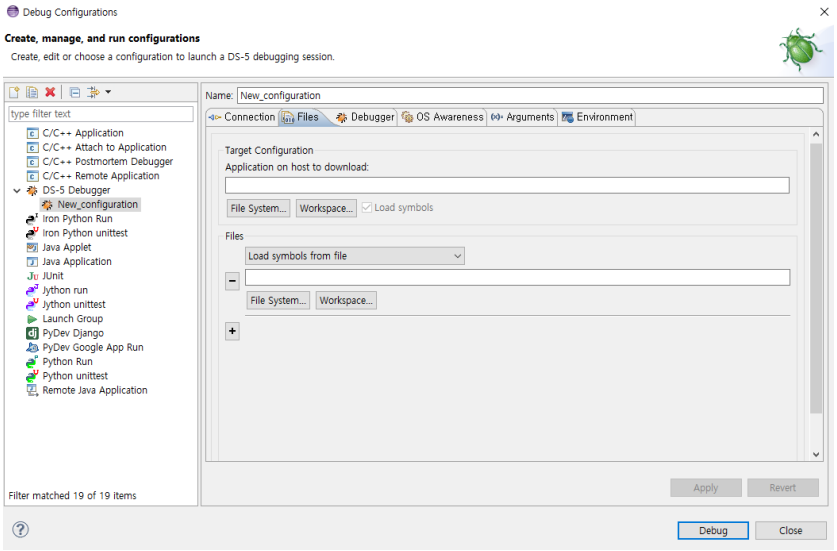
3.3.9 디버그 설정

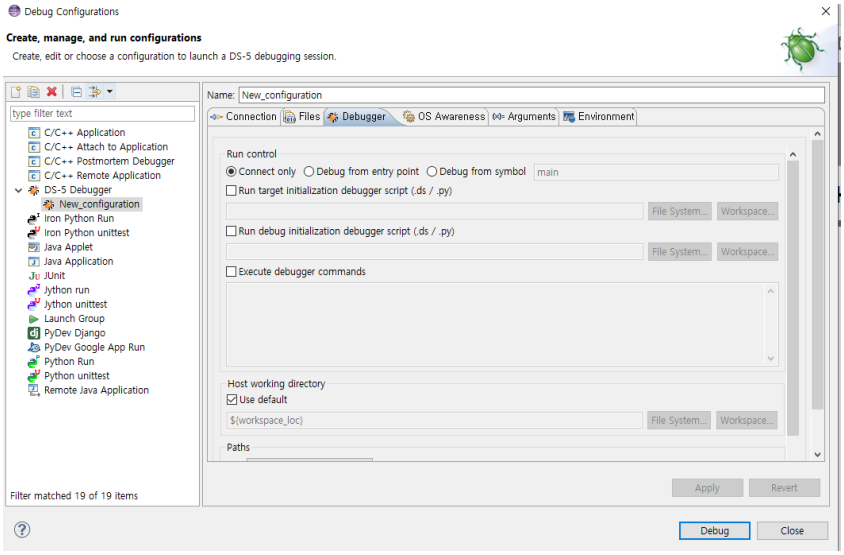
C 언어 소스파일을 만들어 빌드하고 디버그 설정을 한다. 4 주차와 동일한 환경으로 진행한다.





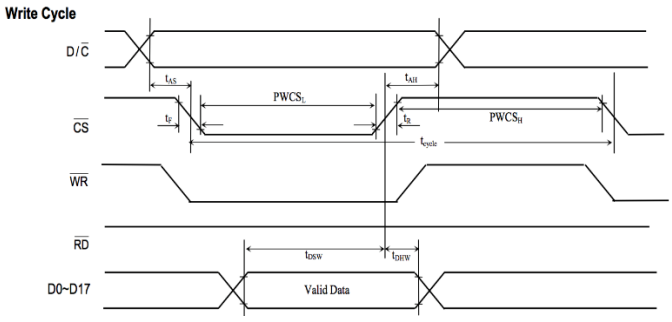

Debug Configurations
Create, manage, and run configurations
type filter text
C/C++ Application
C/C++ Attach to Application
C/C++ Postmortem Debugger
C/C++ Remote Application
DS-5 Debugger
New_configuration
Iron Python Run
Iron Python unittest
Java Applet
Java Application
JUnit
Jython run
Jython unittest
Launch Group
PyDev Django
PyDev Google App Run
Python Run
Python unittest
Remote Java Application
Filter matched 19 of 19 items
Name: New_configuration
Connection
Files
Debugger
OS Awareness
Arguments
Environment
Select target
Select the manufacturer, board, project type and debug operation to use. Currently selected:
PSU / STM32F107VCT6 / Bare Metal Debug / Debug Cortex-M3
pandaboard.org
Phytec
PSU
STM32F107VCT6
Bare Metal Debug
Debug Cortex-M3
Target Connection
DSTREAM
DTSL Options
Edit...
Configure DSTREAM trace or other target options. Using "default" configuration options
DS-5 Debugger will connect to a DSTREAM to debug a bare metal application.
Select target platform
Connections
Bare Metal Debug
Connection
USB:003877
Browse...
Apply
Revert
Debug
Close


Debug Configurations
Create, manage, and run configurations
Create, edit or choose a configuration to launch a DS-5 debugging session.
type filter text
C/C++ Application
C/C++ Attach to Application
C/C++ Postmortem Debugger
C/C++ Remote Application
DS-5 Debugger
New_configuration
Iron Python Run
Iron Python unittest
Java Applet
Java Application
JUnit
Jython run
Jython unittest
Launch Group
PyDev Django
PyDev Google App Run
Python Run
Python unittest
Remote Java Application
Filter matched 19 of 19 items
Name: New_configuration
Connection
Files
Debugger
OS Awareness
Arguments
Environment
Target Configuration
Application on host to download:
File System...
Workspace...
Load symbols
Files
Load symbols from file
File System...
Workspace...
Apply
Revert
Debug
Close

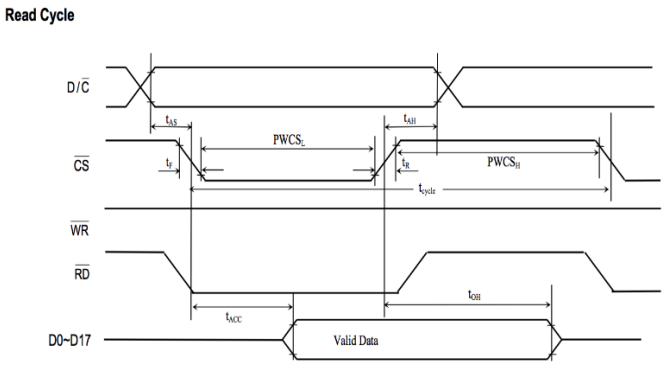

Debug Configurations
Create, manage, and run configurations
Create, edit or choose a configuration to launch a DS-5 debugging session.
type filter text
C/C++ Application
C/C++ Attach to Application
C/C++ Postmortem Debugger
C/C++ Remote Application
DS-5 Debugger
New_configuration
Iron Python Run
Iron Python unittest
Java Applet
Java Application
JUnit
Jython run
Jython unittest
Launch Group
PyDev Django
PyDev Google App Run
Python Run
Python unittest
Remote Java Application
Filter matched 19 of 19 items
Name: New_configuration
Connection
Files
Debugger
OS Awareness
Arguments
Environment
Run control
Connect only
Debug from entry point
Debug from symbol
main
Run target initialization debugger script (.ds / .py)
Run debug initialization debugger script (.ds / .py)
Execute debugger commands
Host working directory
Use default
${workspace_loc}
Paths
File System...
Workspace...
Apply
Revert
Debug
Close

04. 실험 및 과제 해결

4.1 LCD Library 수정



1. CS 를 low 로 하여 chip 동작시킨다.
2. RS(=D/C)가 low 라면 command, high 라면 display data 가 register 에 write 된다.
3. Write cycle 이므로 RD 는 항상 high 이다.
4. WR 이 low->high 일 때 display RAM 에 data/ register 에 command 를 write 한다.



1. CS 를 low 로 하여 chip 동작시킨다.
2. RS 를 high 로 하여 display RAM 에 저장된 display data 를 읽어오도록 한다.
3. Read cycle 이므로 WR 는 high 를 유지한다.
4. RD 가 low->high 일 때 display data 를 읽는다.

위의 Timing diagram 을 참고하여 lcd.c 파일을 아래와 같이 수정한다.

```
static void LCD_WR_REG(uint16_t LCD_Reg)
{
    LCD_CS(0);
    LCD_RS(0);
    LCD_WR(0);
    GPIO_Write(GPIOE, LCD_Reg);
    LCD_WR(1);
    LCD_CS(1);
}
static void LCD_WR_DATA(uint16_t LCD_Data)
{
    LCD_CS(0);
    LCD_RS(1);
    LCD_WR(0);
```

```
    GPIO_Write(GPIOE, LCD_Data);
    LCD_WR(1);
    LCD_CS(1);
}
static uint16_t LCD_ReadReg(uint16_t LCD_Reg)
{
      uint16_t temp;
      GPIO_InitTypeDef GPIO_InitStructure;
      LCD_WR_REG(LCD_Reg);
      // To Read from Data Line
      GPIO_InitStructure.GPIO_Pin = GPIO_Pin_All;
      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
      GPIO_Init(GPIOE, &GPIO_InitStructure);
      LCD_CS(0);
      LCD_RS(1);
      LCD_RD(0);
      temp = GPIO_ReadInputData(GPIOE);
      LCD_RD(1);
      LCD_CS(1);
      // Read Done, Reset
      GPIO_InitStructure.GPIO_Pin = GPIO_Pin_All;
      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
      GPIO_Init(GPIOE, &GPIO_InitStructure);
      return temp;
}
```

4.2 RCC & GPIO Configure.

실험에 필요한 포트에 클럭을 인가하고 아래와 같이 GPIO 를 설정하였다.

```
void RCC_Configure() {
         RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);
         RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);
         RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOC |
RCC_APB2Periph_GPIOD | RCC_APB2Periph_GPIOE | RCC_APB2Periph_GPIOB | RCC_APB2Periph_ADC1,
ENABLE);
}

void GPIO_Configure() {
         GPIO_InitTypeDef GPIO_InitStructure;

         GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
         GPIO_InitStructure.GPIO_Pin = (GPIO_Pin_2);
         GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
         GPIO_Init(GPIOD, &GPIO_InitStructure);
}
```

4.3 TIM_Configure

Timer 2 를 사용하기 위한 TIM configure 은 아래와 같다. 분주 공식을 이용해 1/100 초를 맞추기 위해 Period 에 120, Prescaler 에 6000 의 값을 주었다.

```
void TIM_Configure() {
         TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;

         TIM_TimeBaseStructure.TIM_Period = 120;
         TIM_TimeBaseStructure.TIM_Prescaler = 6000;
         TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;
         TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Down;
         TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);

         TIM_ARRPreloadConfig(TIM2, ENABLE);
```

```
        TIM_Cmd(TIM2, ENABLE);
        TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE);
}
```

```
typedef struct
{
  uint16_t TIM_Prescaler;         /*!< Specifies the prescaler value used to divide the TIM clock.
                                       This parameter can be a number between 0x0000 and 0xFFFF */

  uint16_t TIM_CounterMode;       /*!< Specifies the counter mode.
                                       This parameter can be a value of @ref TIM_Counter_Mode */

  uint16_t TIM_Period;            /*!< Specifies the period value to be loaded into the active
                                       Auto-Reload Register at the next update event.
                                       This parameter must be a number between 0x0000 and 0xFFFF.  */

  uint16_t TIM_ClockDivision;     /*!< Specifies the clock division.
                                      This parameter can be a value of @ref TIM_Clock_Division_CKD */

  uint8_t TIM_RepetitionCounter;  /*!< Specifies the repetition counter value. Each time the RCR downcounter
                                       reaches zero, an update event is generated and counting restarts
                                       from the RCR value (N).
                                       This means in PWM mode that (N+1) corresponds to:
                                          - the number of PWM periods in edge-aligned mode
                                          - the number of half PWM period in center-aligned mode
                                       This parameter must be a number between 0x00 and 0xFF.
                                       @note This parameter is valid only for TIM1 and TIM8. */
} TIM_TimeBaseInitTypeDef;
```

4.4 NVIC_Configure

Timer Interrupt 를 사용하기 위한 NVIC configure 은 아래와 같다.

```
void NVIC_Configure() {
        NVIC_InitTypeDef NVIC_InitStructure;

        NVIC_InitStructure.NVIC_IRQChannel = TIM2_IRQn;
        NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
        NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
        NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
        NVIC_Init(&NVIC_InitStructure);
}
```

4.5 TIM2_IRQHandler

Timer Handler 는 아래와 같다. 1/100 초마다 LED 가 1 번 깜빡인다. Timer 의 숫자가 1/100 초부터 증가한다. LCD_ShowNum 을 이용해 LCD 에 증가하는 타이머 숫자를 나타낸다.

```
void TIM2_IRQHandler() {
        if(TIM_GetITStatus(TIM2, TIM_IT_Update)!=RESET) {
                if(flag==0) {
                        if(ledflag==0) {
                                GPIO_SetBits(GPIOD, GPIO_Pin_2);
                                ledflag=1;
                        }
                        else {
                                GPIO_ResetBits(GPIOD, GPIO_Pin_2);
                                ledflag=0;
                        }
                        t1++;
                        if(t1 == 10) {
                                t2++;
                                t1 = 0;
                        }
                        if(t2 == 10) {
                                t3++;
                                t2 = 0;
                        }
                        if(t3 == 60) {
                                t4++;
                                t3 = 0;
```

```
                        }
                }
                else if(flag == 1) {
                        ;
                }
                else if(flag == 2) {
                        t1 = 0, t2 = 0, t3 = 0, t4 = 0;
                }
                LCD_ShowNum(40, 130, t4, 2, BLACK, WHITE);
                LCD_ShowNum(80, 130, t3, 2, BLACK, WHITE);
                LCD_ShowNum(130, 130, t2, 2, BLACK, WHITE);
                LCD_ShowNum(180, 130, t1, 2, BLACK, WHITE);
        }
        TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
}
```

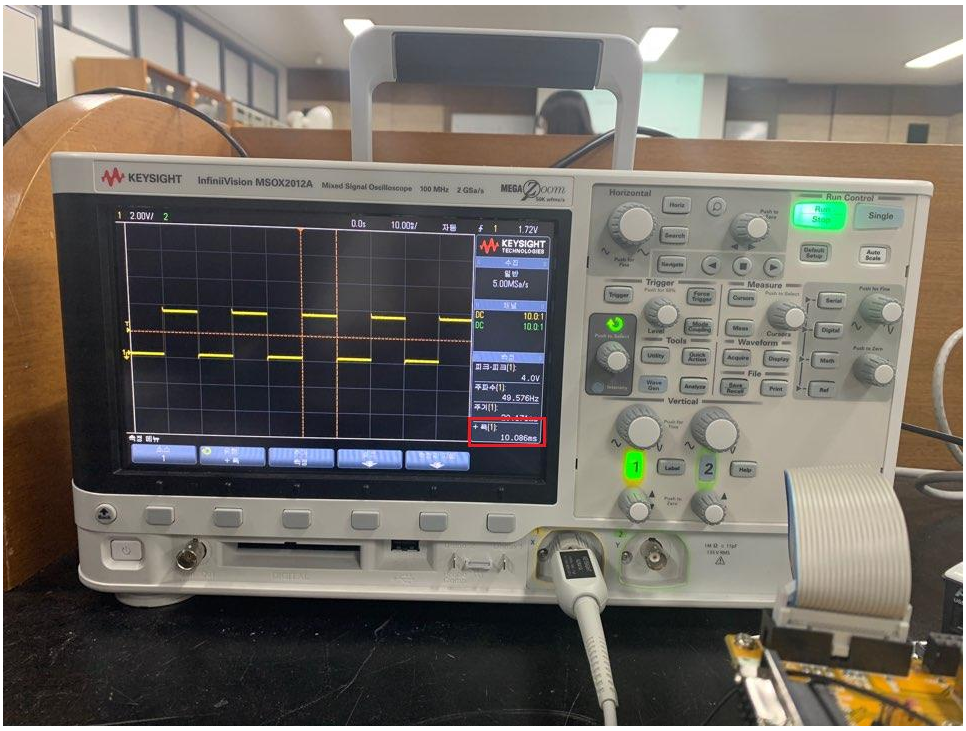
4.6 main

LCD_ShowString 을 이용하여 LCD 화면에 분, 초, 1/10 초, 1/100 초 단위를 표시한다. LCD_ShowString 와 LCD_DrawRectangle 을 이용하여 LCD 화면에 Start, Stop, Reset 버튼을 표시한다. while 문 안에서 Touch 좌표와 Flag 값을 이용하여 Start(flag=0), Stop(flag=1), Reset(flag=2) 을 구분해준다.

```
int main(void) {
        SystemInit();
        LCD_Init();
        Touch_Configuration();
        Touch_Adjust();
        LCD_Clear(WHITE);

        RCC_Configure();
        GPIO_Configure();
        TIM_Configure();
        NVIC_Configure();

        LCD_ShowString(80,80,"Mon_Team07", BLACK, WHITE);

        LCD_ShowString(20,210,"start", BLACK, WHITE);
        LCD_ShowString(95,210,"stop", BLACK, WHITE);
        LCD_ShowString(165,210,"reset", BLACK, WHITE);

        LCD_DrawRectangle(10, 200, 70, 230);
        LCD_DrawRectangle(80, 200, 140, 230);
        LCD_DrawRectangle(150, 200, 210, 230);

        LCD_ShowString(30,170,"min", BLACK, WHITE);
        LCD_ShowString(70,170,"sec", BLACK, WHITE);
        LCD_ShowString(110,170,"1/10", BLACK, WHITE);
        LCD_ShowString(160,170,"1/100", BLACK, WHITE);

        while(1) {

                Touch_GetXY(&pos_x, &pos_y, 1);

                Convert_Pos(pos_x, pos_y, &con_x, &con_y);

                if( con_x > 10 && con_x < 70 && con_y > 200 && con_y < 230 ) {
                        flag = 0;
                }
                else if( con_x > 80 && con_x < 140 && con_y > 200 && con_y < 230 ) {
                        flag = 1;
                }
                else if( con_x > 150 && con_x < 210 && con_y > 200 && con_y < 230 ) {
                        flag = 2;
                }
        }
}
```

4.7 오실로스코프 확인

오실로스코프를 사용하여 확인한 타이밍은 다음과 같다. LED 신호에 연결한 오실로스코프를 보게 되면 1/10000 초 정도의 오차가 나오는 것을 확인 할 수 있다.



05. 실험 결과

5.1 소스코드

위의 내용을 바탕으로 작성한 전체 소스코드는 아래와 같다.

```
#include <misc.h>
#include <stm32f10x.h>
#include <stm32f10x_exti.h>
#include <stm32f10x_gpio.h>
#include <stm32f10x_rcc.h>
#include <stm32f10x_usart.h>
#include <stm32f10x_tim.h>
#include <stm32f10x_adc.h>
#include "lcd.h"
#include "touch.h"

uint16_t pos_x, pos_y, con_x, con_y;
int t1, t2, t3, t4;
int flag = 1, ledflag=0;

void delay(void) {
	int i;
	for(i=0; i<10000; ++i) ;
}

void RCC_Configure() {
	RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);
	RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);
	RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOC |
RCC_APB2Periph_GPIOD | RCC_APB2Periph_GPIOE | RCC_APB2Periph_GPIOB | RCC_APB2Periph_ADC1,
ENABLE);
}
```

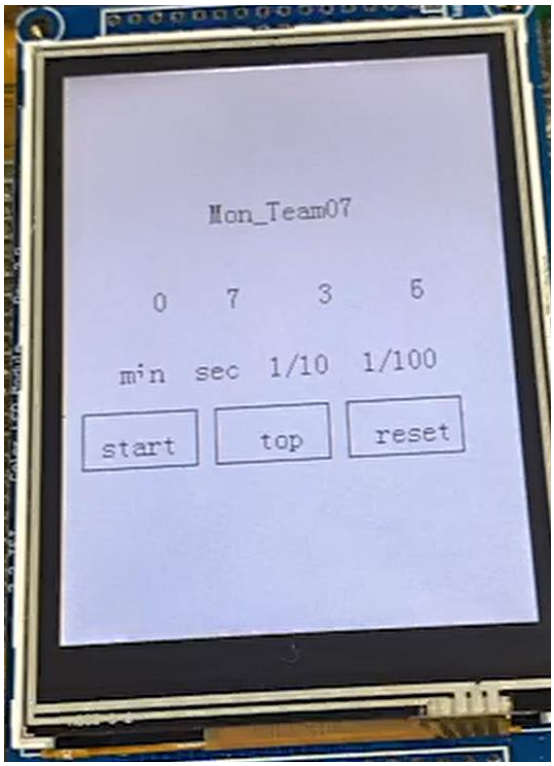
```
void GPIO_Configure() {
	GPIO_InitTypeDef GPIO_InitStructure;

	GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
	GPIO_InitStructure.GPIO_Pin = (GPIO_Pin_2);
	GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
	GPIO_Init(GPIOD, &GPIO_InitStructure);
}

void TIM_Configure() {
	TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;

	TIM_TimeBaseStructure.TIM_Period = 120;
	TIM_TimeBaseStructure.TIM_Prescaler = 6000;
	TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;
	TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Down;
	TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);

	TIM_ARRPreloadConfig(TIM2, ENABLE);
	TIM_Cmd(TIM2, ENABLE);
	TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE);
}

void NVIC_Configure() {
	NVIC_InitTypeDef NVIC_InitStructure;

	NVIC_InitStructure.NVIC_IRQChannel = TIM2_IRQn;
	NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
	NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
	NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
	NVIC_Init(&NVIC_InitStructure);
}

void TIM2_IRQHandler() {
	if(TIM_GetITStatus(TIM2, TIM_IT_Update)!=RESET) {
		if(flag==0) {
			if(ledflag==0) {
				GPIO_SetBits(GPIOD, GPIO_Pin_2);
				ledflag=1;
			}
			else {
				GPIO_ResetBits(GPIOD, GPIO_Pin_2);
				ledflag=0;
			}
			t1++;
			if(t1 == 10) {
				t2++;
				t1 = 0;
			}
			if(t2 == 10) {
				t3++;
				t2 = 0;
			}
			if(t3 == 60) {
				t4++;
				t3 = 0;
			}
		}
		else if(flag == 1) {
			;
		}
		else if(flag == 2) {
			t1 = 0, t2 = 0, t3 = 0, t4 = 0;
		}
		LCD_ShowNum(40, 130, t4, 2, BLACK, WHITE);
		LCD_ShowNum(80, 130, t3, 2, BLACK, WHITE);
		LCD_ShowNum(130, 130, t2, 2, BLACK, WHITE);
```

```
                    LCD_ShowNum(180, 130, t1, 2, BLACK, WHITE);
          }
          TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
}

int main(void) {
          SystemInit();
          LCD_Init();
          Touch_Configuration();
          Touch_Adjust();
          LCD_Clear(WHITE);

          RCC_Configure();
          GPIO_Configure();
          TIM_Configure();
          NVIC_Configure();

          LCD_ShowString(80,80,"Mon_Team07", BLACK, WHITE);

          LCD_ShowString(20,210,"start", BLACK, WHITE);
          LCD_ShowString(95,210,"stop", BLACK, WHITE);
          LCD_ShowString(165,210,"reset", BLACK, WHITE);

          LCD_DrawRectangle(10, 200, 70, 230);
          LCD_DrawRectangle(80, 200, 140, 230);
          LCD_DrawRectangle(150, 200, 210, 230);

          LCD_ShowString(30,170,"min", BLACK, WHITE);
          LCD_ShowString(70,170,"sec", BLACK, WHITE);
          LCD_ShowString(110,170,"1/10", BLACK, WHITE);
          LCD_ShowString(160,170,"1/100", BLACK, WHITE);

          while(1) {

                    Touch_GetXY(&pos_x, &pos_y, 1);

                    Convert_Pos(pos_x, pos_y, &con_x, &con_y);

                    if( con_x > 10 && con_x < 70 && con_y > 200 && con_y < 230 ) {
                              flag = 0;
                    }
                    else if( con_x > 80 && con_x < 140 && con_y > 200 && con_y < 230 ) {
                              flag = 1;
                    }
                    else if( con_x > 150 && con_x < 210 && con_y > 200 && con_y < 230 ) {
                              flag = 2;
                    }
          }
}
```

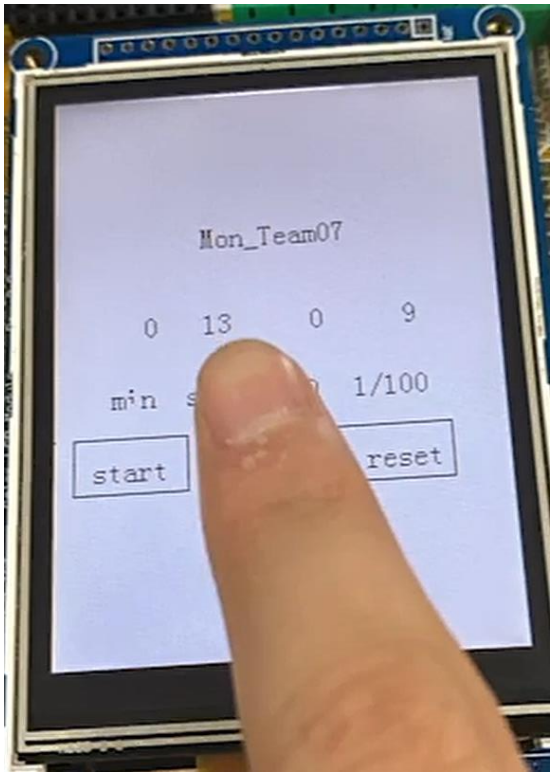
5.2 결과 확인 및 사진

소스 코드의 동작을 다음과 같이 확인했다.

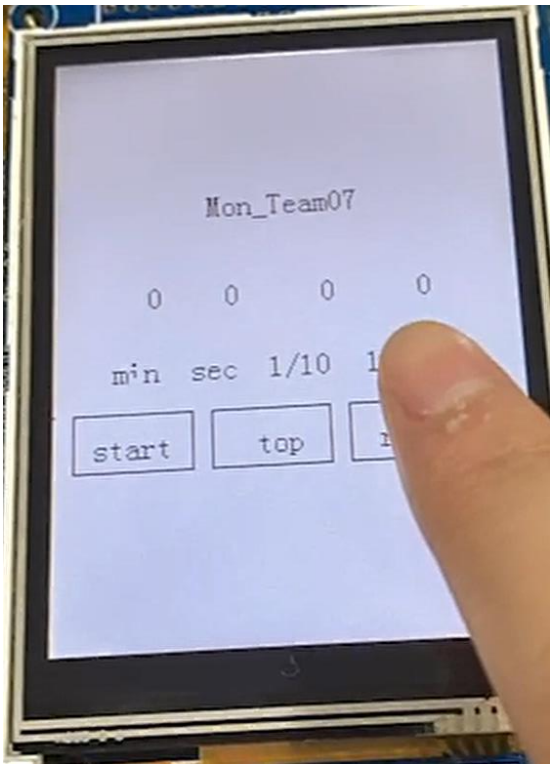
① Start 버튼 터치 시 작동



② Stop 버튼 터치 시 작동 멈춤



③ Reset 버튼 터치 시 모든 값 0 으로 초기화



06. 결론

지난 실험에서 사용한 TFT LCD 를 이용하여 Timer 를 구현 해보았다. TIM_Configure 와 NVIC_Configure 를 이용하여 Timer 와 interrupt 를 설정 할 수 있었다. 이번 실험을 통해 텀 프로젝트에서 타이머를 사용 할 일이 있으면 도움이 될 것 같다. 코드를 짤 땐 정확하다고 생각했는데 오실로스코프에 연결해 확인했을 때 1/10000 초 정도의 오차가 나서 아쉽긴 했지만 오차를 제외하면 만족스러운 실험이었다.