임베디드시스템 설계 및 실험

10주차 결과 보고서

2023.11.07.

2조

202155592 이지수

202155553 박은재

201924515 유승훈

201524410 고상현

201924402 강민수

1. 실험 목표

- TFT-LCD의 원리와 동작 방법에 대한 이해
- TFT-LCD 라이브러리 작성과 이해
- TFT-LCD Touch 동작 제어
- ADC 개념 이해
- 조도 센서 사용 방법 학습

2. 실험 이론 및 원리

1) TFT-LCD

- 초 박막 액정표시장치
- 액체와 고체의 중간 특성을 가진 액정의 상태 변화와 편광판의 편광 성질을 이용하여 통과하는 빛의 양을 조절함으로써 정보를 표시
- RGB 픽셀이 유리판에 코딩 되어 컬러 영상을 구현하는 Color Filter
- 액정을 제어하기 위해 조박형 유리 기판 위에 반도체 막을 형성한 회로인 TFT 기판
- Filter와 기판 사이에 주입된 액정과 광원인 Black light unit으로 구성

2) Timing Diagram

- 각 신호들이 시간별로 처리되는 과정을 그림으로 나타냄
- Low에서 High로 올라가는 구간을 Rising Edge
- High에서 Low로 떨어지는 구간을 Falling Edge

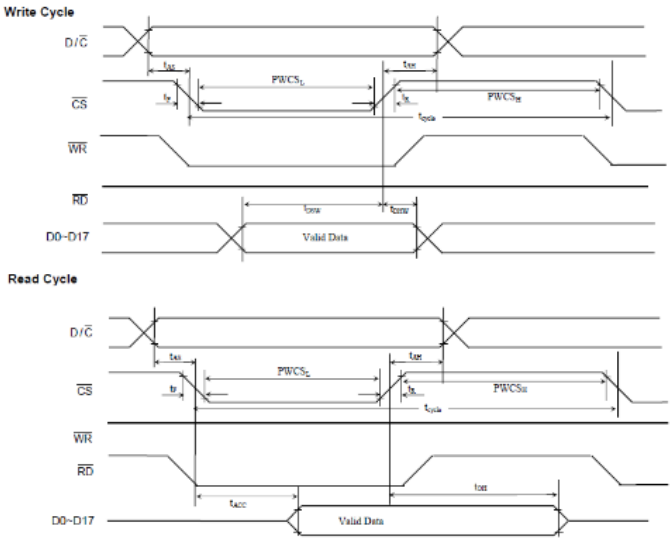
- $\boldsymbol{CS}$: Chip Select (Chip Enable)
- High 에서 Low로 Falling Edge 일 때 LCD Chip 을 사용

- $\boldsymbol{D/C}$: Data / Command (핀맵에서 RS)
- LCD는 Data 와 명령어 레지스터를 함께 사용

- High 로 두고 Data를 전송, Low 로 두고 Command를 전송

- ***WR***, ***RD*** : Write / Read

- High 에서 Low로 Falling Edge 일 때, Data를 display에 Write / Read 한다



3) ADC

- 아날로그 신호를 디지털로 변환하는 것

- 아날로그 신호가 들어오면 이를 표본화, 양자화를 거쳐 부호화 한다.

- 표본화: 일정한 간격으로 아날로그 신호의 값을 추출

- 양자화: 추출한 표본 샘플 신호의 레벨을 단계를 나누어 나타내는 과정

- 부호화: 양자화로 나눈 레벨에 속한 값을 이진수로 변환

4) 조도센서

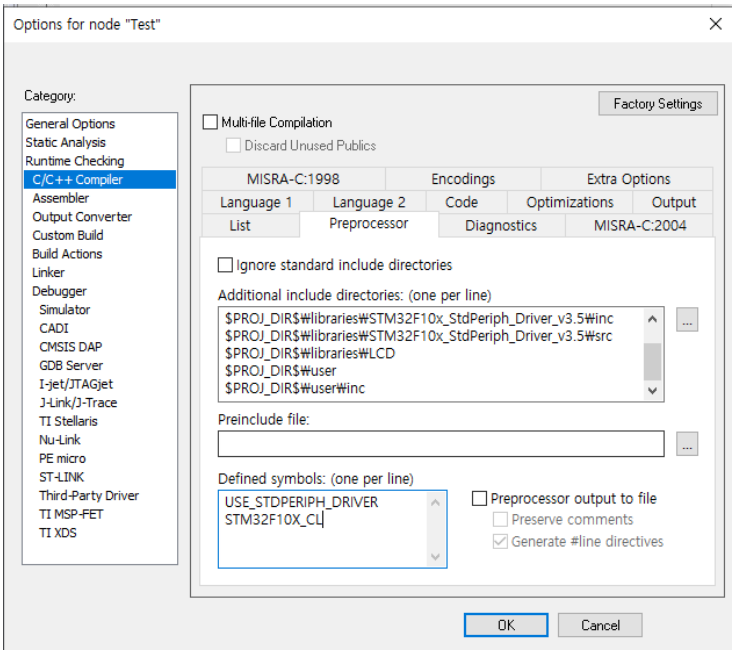
- 주변의 밝기를 측정하는 센서

- 빛의 양이 많아질수록 전도율이 높아져 저항이 낮아짐

3. 실험 과정

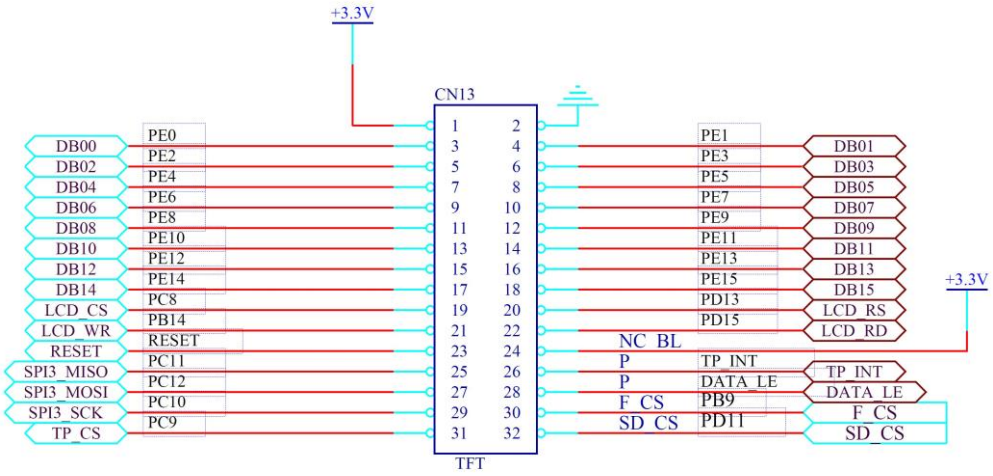
1) LCD 라이브러리 등록하기

Libraries 폴더 밑에 LCD 폴더 생성한 후 font.h, lcd.c, lcd.h, touch.c, touch.h 총 5개의 라이브러리 파일을 추가한다.



2) lcd.c 코드 작성하기

본 보고서의 '실험 이론 및 원리'의 'Timing Diagram' 설명을 참고하여 코드를 작성한다.



TFT-LCD의 schematic을 보면 LCD_CS는 PC8, LCD_WR은 PB14, LCD_RS는 PD13, LCD_RD는 PD15이다.

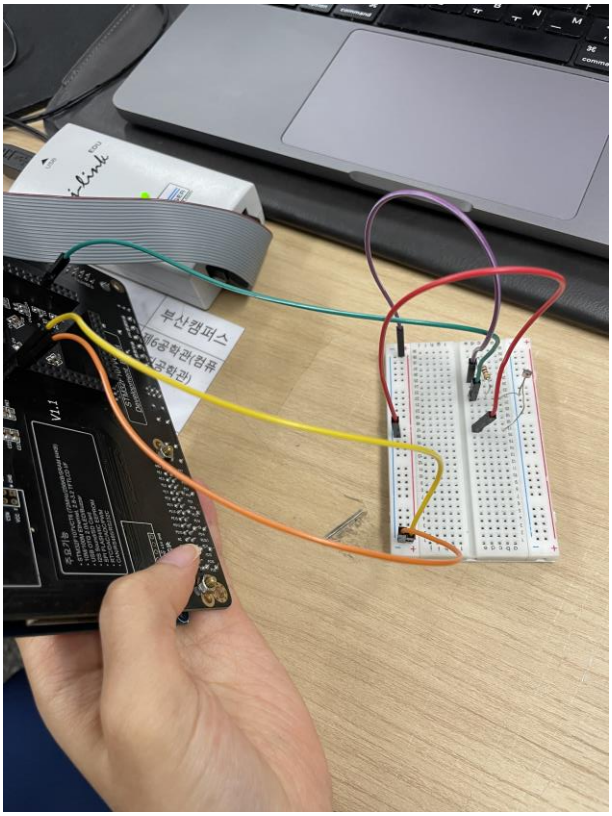
```
static void LCD_WR_REG(uint16_t LCD_Reg)
{
    // TODO implement using GPIO_ResetBits/GPIO_SetBits
    GPIO_ResetBits(GPIOD, GPIO_Pin_13);
    GPIO_ResetBits(GPIOC, GPIO_Pin_8);
    GPIO_ResetBits(GPIOB, GPIO_Pin_14);
    GPIO_Write(GPIOE, LCD_Reg);
    // TODO implement using GPIO_ResetBits/GPIO_SetBits
    GPIO_SetBits(GPIOC, GPIO_Pin_8);
    GPIO_SetBits(GPIOB, GPIO_Pin_14);

}
```

LCD_WR_REG에서는 RS, CS, WR을 reset하여 전송한 후에 CS,WR을 set한다.

```
static void LCD_WR_DATA(uint16_t LCD_Data)
{
    // TODO implement using GPIO_ResetBits/GPIO_SetBits
    GPIO_SetBits(GPIOD, GPIO_Pin_13);
    GPIO_ResetBits(GPIOC, GPIO_Pin_8);
    GPIO_ResetBits(GPIOB, GPIO_Pin_14);

    GPIO_Write(GPIOE, LCD_Data);
    // TODO implement using GPIO_ResetBits/GPIO_SetBits
    GPIO_SetBits(GPIOC, GPIO_Pin_8);
    GPIO_SetBits(GPIOB, GPIO_Pin_14);
}
```

LCD_WR_DATA에서는 RS는 set하고 CS와 WR을 reset하여 전송한 후에 CS, WR을 set한다.

3) 보드와 TFT-LCD 연결하고 조도센서 회로 구성하기



4) main.c 코드 작성하기

```
void RCC_Configure(void)
{
  // TODO: Enable the APB2 peripheral clock using the function 'RCC_APB2PeriphClockCmd'
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC,ENABLE);
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1,ENABLE);
}

void GPIO_Configure(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2 ;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOC, &GPIO_InitStructure);

    // TODO: Initialize the GPIO pins using the structure 'GPIO_InitTypeDef' and the fur


}
```

사용할 핀에 대해 RCC Clock과 핀 설정을 한다. RCC_APB2Periph로 GPIOC와 ADC1에 Clock을 준다. 그리고 ADC_Channel_12를 사용할 것이므로 PC2에 대해 Analog IN모드로 설정한다.

```
void ADC_Configure(void) {
    ADC_InitTypeDef ADC_12;
    ADC_12.ADC_ContinuousConvMode = ENABLE;
    ADC_12.ADC_DataAlign = ADC_DataAlign_Right;
    ADC_12.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
    ADC_12.ADC_Mode = ADC_Mode_Independent;
    ADC_12.ADC_NbrOfChannel = 1;
    ADC_12.ADC_ScanConvMode = DISABLE;

    ADC_Init(ADC1, &ADC_12);
    ADC_RegularChannelConfig(ADC1, ADC_Channel_12, 1, ADC_SampleTime_239Cycles5);
    ADC_ITConfig(ADC1,ADC_IT_EOC, ENABLE);
    ADC_Cmd(ADC1, ENABLE);
    ADC_ResetCalibration(ADC1);
    while(ADC_GetResetCalibrationStatus(ADC1));
    ADC_StartCalibration(ADC1);
    while(ADC_GetCalibrationStatus(ADC1));
    ADC_SoftwareStartConvCmd(ADC1, ENABLE);
}
```

조도센서의 Analog 신호를 Digital 신호로 변환하기 위한 ADC 설정함수이다. ADC 구조체를 선언하여 설정한다. ADC의 모드를 Independent로 설정하고 변환을 단일 채널로 수행하기 때문에 ADC_ScanConvMode를 DISABLE로 설정한다. 그리고 연속적인 변환이 필요하므로 ADC_ContinousConvMode를 ENABLE로 설정하고 외부 trigger를 None으로 설정한다. 마지막으로 신호를 왼쪽부터 읽을지, 오른쪽부터 읽을지 설정하는 ADC_DataAlign을 Right로 설정해주고 사용할 채널의 개수를 1로 설정한다.

```
void NVIC_Configure(void) {

    NVIC_InitTypeDef NVIC_InitStructure;

    // TODO: fill the arg you want
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1); // need modif

    // TODO: Initialize the NVIC using the structure 'NVIC_InitTy
    NVIC_InitStructure.NVIC_IRQChannel = ADC1_2_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x00;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x00;
    NVIC_Init(&NVIC_InitStructure);

}

void ADC1_2_IRQHandler() {
  if(ADC_GetITStatus(ADC1,ADC_IT_EOC) != RESET) {
    value = ADC_GetConversionValue(ADC1);
  }
  ADC_ClearITPendingBit(ADC1, ADC_IT_EOC);
}
```

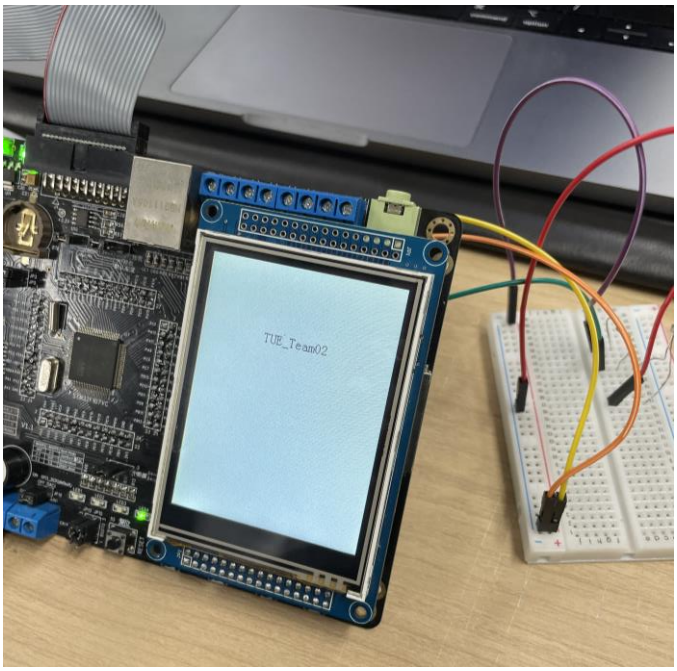
ADC1_2에 대한 인터럽트 핸들러를 구현한다. 인터럽트 핸들러가 하나밖에 없기 때문에 우선순위를 임의로 지정한다. ADC_GetItStatus로 체크하고 ADC1로 얻은 값을 value에 넣는다.

```
int main(void)
{

    SystemInit();

    RCC_Configure();

    GPIO_Configure();

    ADC_Configure();

    NVIC_Configure();

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


    LCD_ShowString(80,80,"TUE_Team02",BLACK,WHITE);

    while (1) {
      Touch_GetXY(&x,&y,1);
      Convert_Pos(x,y,&x,&y);
        if(x !=0 || y !=0) {
          LCD_DrawCircle(x,y,5);
          LCD_ShowNum(80,120,value,4,BLACK,WHITE);
          LCD_ShowNum(80,140,x,3,BLACK,WHITE);
          LCD_ShowNum(80,160,y,3,BLACK,WHITE);
        }

    }
    return 0;
}
```

LCD_Show 함수로 LCD에 원하는 값을 출력할 수 있는데 이때 x, y 좌표도 함께 설정해 주어야 한다. 여기서는 LCD_ShowString으로 "TUE_Team02"를 출력했고, Touch_GetXY와 Convert_Pos로 터치한 지점의 x, y 좌표를 받아왔으며 LCD_DrawCircle로 원을 그렸다. 또한, ADC를 통해 받아온 조도센서의 값과 터치한 지점의 x, y 좌표를 LCD_ShowNum으로 출력한다.

4. 실험 결과

1) TFT-LCD에 “TUE_Team02” 출력



2) LCD 터치 시 해당 위치에 작은 원을 그리고 좌표(X, Y) 및 전역변수에 저장했던 조도 센서 값 출력