2021년 2학기
임베디드 시스템설계 및 실험
10주차 실험보고서

201712159 조현우
201724566 전승윤
201724611 최호진
201924650 박지호

목차

1. 실험 목적

TFT LCD 제어 및 ADC 사용

2. 실험 과정

1) TFT-LCD 보드에 올바르게 결착

2) lcd.c에서 write 관련 코드 작성

3) TFT-LCD에 Text(팀명) 출력

4) 조도 센서 ADC 설정

5) ADC channel과 인터럽트를 사용하여 조도 센서 값을 전역 변수에 저장

6) LCD 터치 시(main에서 폴링 방식) 해당 위치에 작은 원을 그리고 좌표(X, Y), 전역변수에 저장했던 조도 센서 값 출력

3. 실험 세부 내용

1) TFT-LCD 보드에 올바르게 결착

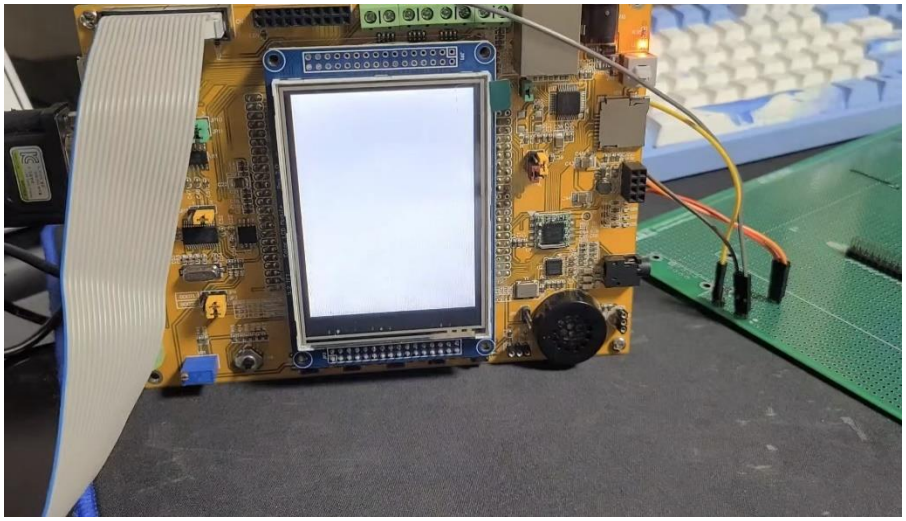


그림 1 - LCD 부착

2) lcd.c에서 write 관련 코드 작성

Lcd.c 에서는 Timediagram을 참고하여 LCD의 Write관련 함수인 LCD_WR_REG, LCD_WR_DATA를 작성한다.

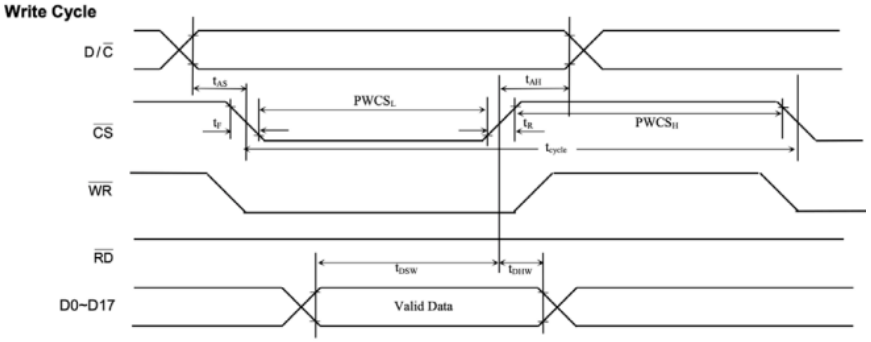


그림 2 - Timediagram

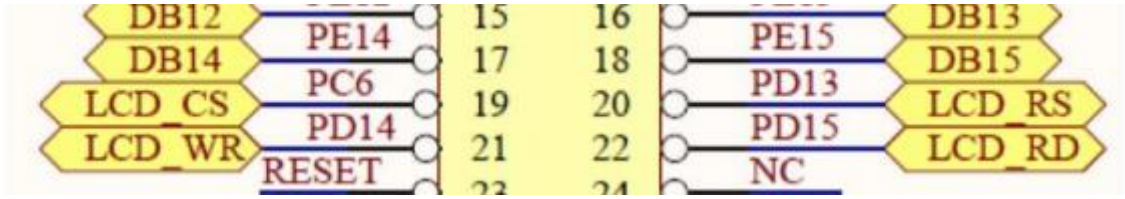


그림 3 – TFT-LCD pin맵

Timediagram의 D/C는 Data와 Command를 의미한다. 이를 참고하여 사용해야할 bit를 설정하여 사용해야할 Port와 Pin번호를 설정해준다. 이때 같이 설정해주어야 할 CS(RS), WR RD의 Port와 Pin번호는 그림 3의 Pin맵을 참고하였다.

```
static void LCD_WR_REG(uint16_t LCD_Reg)
{
        // TODO implement using GPIO_ResetBits/GPIO_SetBits
        GPIO_ResetBits(GPIOD, GPIO_Pin_13);   // LCD_RS(0);
        GPIO_ResetBits(GPIOC, GPIO_Pin_6);  // LCD_CS(0);
        GPIO_ResetBits(GPIOD, GPIO_Pin_14); // LCD_WR(0);

        GPIO_Write(GPIOE, LCD_Reg);

        // TODO implement using GPIO_ResetBits/GPIO_SetBits
        GPIO_SetBits(GPIOC, GPIO_Pin_6);  // LCD_CS(1);
        GPIO_SetBits(GPIOD, GPIO_Pin_14);  // LCD_WR(1);

}
```

그림 4 - REG(Command) 코드

```
static void LCD_WR_DATA(uint16_t LCD_Data)
{
        // TODO implement using GPIO_ResetBits/GPIO_SetBits
        GPIO_SetBits(GPIOD, GPIO_Pin_13);   // LCD_RS(1);
        GPIO_ResetBits(GPIOC, GPIO_Pin_6);  // LCD_CS(0);
        GPIO_ResetBits(GPIOD, GPIO_Pin_14); // LCD_WR(0);

        GPIO_Write(GPIOE, LCD_Data);

        // TODO implement using GPIO_ResetBits/GPIO_SetBits
        GPIO_SetBits(GPIOC, GPIO_Pin_6);  // LCD_CS(1);
        GPIO_SetBits(GPIOD, GPIO_Pin_14);  // LCD_WR(1);

}
```

그림 5 - Data 코드

- *COMMAND*
  - $D/\overline{C}$를 Low, $\overline{CS}$를 Low, $\overline{WR}$를 Low로 두고 Command를 전송
  - $\overline{CS}$를 High, $\overline{WR}$를 High로 다시 돌려놓기

- *DATA*
  - $D/\overline{C}$를 High, $\overline{CS}$를 Low, $\overline{WR}$를 Low로 두고 Data를 Display에 전송
  - $\overline{CS}$를 High, $\overline{WR}$를 High로 다시 돌려놓기

그림 6 – command, data 강의자료

위 강의 자료를 참고해 Command와 Data 부분의 코드를 구성할 수 있었다.

그림 4의 REG코드에서는 D/C는 Low, C/S를 Low, W/R를 Low로 두고 Command를 전송한다. 이후 C/S 와 W/R을 High로 돌려놓는다. 그림 5의 DATA코드에서는 D/C는 High, C/S를 Low, W/R를 Low로 두고 Command를 전송한다. 이후 C/S 와 W/R을 High로 돌려놓는다.

3) TFT-LCD에 Text(팀명) 출력

```
char msg[] = "Team03";
LCD_ShowString(0, 0, msg, color[11], color[0] );
```

그림 7 – 팀 명 출력

그림 7과 같이 팀 명 출력은 LCD_ShowString 함수를 통해 출력한다. 인자로는 시작지점, 출력 팀 명, 출력할 문자열의 색깔과 LCD보드 배경색을 전달한다.

4) 조도 센서 ADC 설정

```
void ADC_Configure() {
  ADC_InitTypeDef ADC_InitStructure;

  ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
  ADC_InitStructure.ADC_ScanConvMode = DISABLE;
  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_Init(ADC1, &ADC_InitStructure);

  ADC_RegularChannelConfig(ADC1, ADC_Channel_8, 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) ;

}
```

그림 8 – ADC_Configure 과정

GPIO_Configure과 비슷하게 ADC 포트를 활성화 시키기 위해서 Configure 과정이 필요하다. 그림 8의 코드는 10주차 강의 자료에 나온 ADC 구조체 관련 코드를 참고해 설정했다.

5) ADC channel과 인터럽트를 사용하여 조도 센서 값 을 전역 변수에 저장

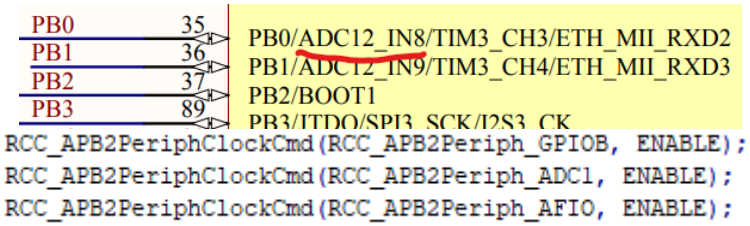


그림 9 - RCC Enable

그림 9는 RCC Enable을 해주기 위한 코드이다. 사용해야할 PortB와 ADC, AFIO를 Enable 해준다. ADC port로 ADC12_IN8을 사용했다.

```
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
GPIO_Init(GPIOB, &GPIO_InitStructure);
```

그림 10- GPIO setting

그림 10은 GPIO 설정 코드이다. 미리 정의된 GPIO_InitStructure 변수를 통해서 사용할 Port의 핀 번호와 GPIO_Mode를 부여해 GPIO_Init()의 인자로 넣어 원하는 설정을 한다.

```
NVIC_EnableIRQ(ADC1_2_IRQn);
NVIC_InitStructure.NVIC_IRQChannel = ADC1_2_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0; // TODO
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0; // TODO
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure);
```

그림 11 - NVIC 인터럽트 setting

그림 11은 NVIC를 이용한 GPIO 인터럽트 세팅이다. 동작이 동시에 일어나지 않으므로 모두 같은 PreemptionPriority와 Subpriority를 설정해 주었다.

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

    ADC_ClearITPendingBit(ADC1, ADC_IT_EOC);
  }
}
```

그림 12 – ADC 인터럽트 Handler

조도 센서의 값을 받아오는 인터럽트 Handler이다. 조도 센서가 아날로그 값을 보드에 전송하려고 할 때, ADC_GetConversionValue 함수를 통해서 해당 값을 저장한다. 이 때 value는 전역 변수로 설정되어 있다. 그리고 ADC의 값을 Clear 해준다.

6) LCD 터치 시(main에서 폴링 방식) 해당 위치에 작은 원을 그리고 좌표(X, Y), 전역변수에 저장했던 조도 센서 값 출력

```
SystemInit();
RCC_Configure();
GPIO_Configure();
ADC_Configure();
NVIC_Configure();
```

그림 13 – ADC 포트, GPIO 포트, 인터럽트 초기 설정 및 활성화

RCC_Configure을 통해서 조도 센서가 사용할 B포트와 ADC에 필요한 ADC 포트의 Clock을 활성화 해준다. GPIO_Configure을 통해서 B포트를 활성화 한다. ADC_Configure을 통해서 조도 센서의 아날로그 값이 디지털 값으로 올바르게 변환될 수 있도록 모드등을 설정한다. NVIC_Configure을 통해서 인터럽트에 사용될 Handler, 채널, 우선순위를 설정해준다.

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

그림 14 – LCD 초기화

LCD를 초기화 해주고 touch를 조정하는 코드를 실행해준다. LCD가 켜지면 가장자리를 정확히 반복 터치하는 방식으로 LCD의 touch를 조정한다.

```
uint16_t x, y;

while (1) {
  Touch_GetXY(&x, &y, 1);
  Convert_Pos(x, y, &x, &y);
  LCD_ShowNum(0, 32, value, 10,color[11], color[0]);
  LCD_ShowNum(0, 64, x, 10,color[11], color[0]);
  LCD_ShowNum(0, 90, y, 10,color[11], color[0]);
  LCD_DrawCircle(x, y, 6);
}
```

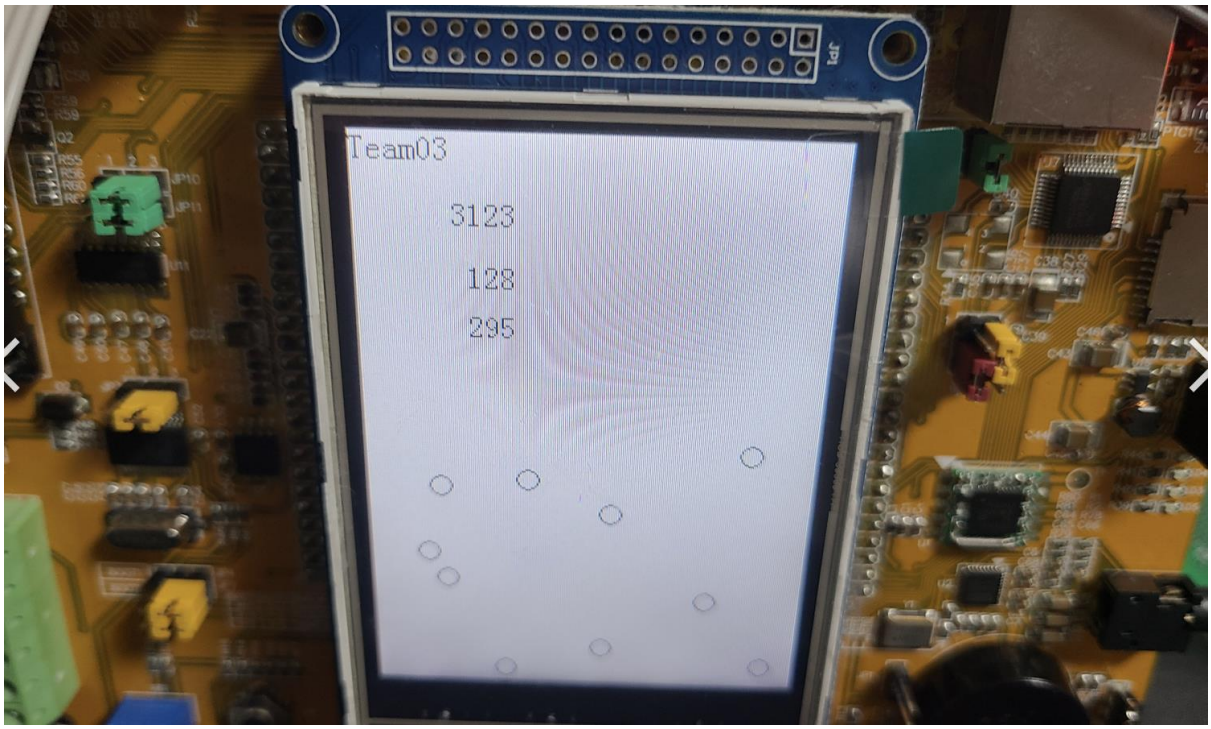
그림 15 – 주요 실행 코드

Touch_GetXY 함수를 통해서 LCD에서 touch를 감지한 부분의 x, y 좌표를 x, y 변수에 저장한다. x와 y가 저장된 순서를 올바르게 바꾸기 위해서 둘의 값을 Convert 해준다.

조도 센서가 읽어 들인 값인 *value*를 color[11] (gray) 색으로 화면에 표시한다. 글자의 배경색은 color[0](white)이다. 이어서 touch가 감지된 x와 y의 좌표를 출력한다. 출력되는 value, x, y 값은 모두 겹치지 않도록 (0, 32), (0, 64), (0, 90) 을 시작점으로 하여 출력된다.

마지막으로 터치 된 부분에 반지름이 6픽셀인 원이 그려진다.

4. 실험 결과



LCD를 통해서 문자열을 출력하고 현재 조도 센서의 값을 ADC를 통해서 숫자로 출력할 수 있게 하였다. 그리고 LCD를 터치한 부분을 동그라미로 표시하고 해당 좌표를 출력할 수 있게 하였다. 이번 실험을 통해서 ADC 의 설정 및 LCD 라이브러리 사용 방법에 대해 익힐 수 있었다.