임베디드시스템 설계 및 실험 보고서

[002 분반 - 2 조 - 9 주차]

조원	202055531 김후겸 202055584 이태경 202155540 김채현 202255535 김진우

실험날짜	2024-11-13

1. 실험 주제

- **DMA(Direct Memory Access) 이해 및 구현**: DMA 의 동작 방식을 이해하고 이를 활용하여 CPU 의 개입 없이 데이터 전송을 처리하는 방법을 익힌다.
- **ADC 와 DMA 의 연계 실습**: 조도 센서를 이용하여 ADC 값을 DMA 로 읽고, 이를 활용해 TFT-LCD 디스플레이에 데이터와 상태를 출력한다.

2. 실험 목적

- CPU 개입을 최소화하면서 메모리와 주변 장치 간 데이터를 효율적으로 전송하는 방법을 실습한다.
- DMA 와 ADC 를 결합하여 데이터 수집 및 디스플레이 제어를 구현한다.
- 실시간 데이터에 따라 동적인 UI 변경을 구현한다.

3. 세부 실험 목적

1. ADC 와 DMA 를 이용하여 조도 센서 데이터를 읽어온다.
2. 읽은 데이터를 TFT-LCD 에 출력한다.
3. 조도 값에 따라 배경색과 글자 색을 동적으로 변경하여 실시간 반응을 확인한다.
4. 코드의 효율성을 유지하며 CPU 리소스를 최소화한다.

4. 실험 장비

- STM32 보드
- TFT-LCD 디스플레이
- 조도 센서

5. 실험 과정

1. 주어진 DMA_Template 파일을 참고하여 main.c 를 작성한다.

   1) AdcInit 함수

```
ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
ADC_InitStructure.ADC_NbrOfChannel = 1;
ADC_InitStructure.ADC_ScanConvMode = DISABLE;
ADC_Init(ADC1, &ADC_InitStructure);
```

AdcInit 함수는 ADC 를 초기화하여 조도 센서 데이터를 읽어오는 기능을 수행한다.

•independent mode 설정: 강의자료 p9 에 따라, 각 ADC 가 서로 간섭 없이 독립적으로 동작하도록 설정

•Continuous Conversion Mode 활성화: 강의자료 p8 의 DMA circular mode 에 따라, ADC 변환이 연속적으로 수행되도록 설정

•Trigger 설정: 외부 신호에 의해 트리거되지 않도록 설정하여 내부적으로 연속 변환이 진행되도록

•Channel 수: 조도 센서 데이터를 읽기 위한 최소 설정

2) DMA_Configure 함수

```
DMA_DeInit(DMA1_Channel1);
DMA_StructInit(&DMA_Instructure);

DMA_Instructure.DMA_PeripheralBaseAddr = (uint32_t)&ADC1->DR;
DMA_Instructure.DMA_MemoryBaseAddr = (uint32_t)&ADC_Value;
DMA_Instructure.DMA_BufferSize = 1;
DMA_Instructure.DMA_MemoryInc = DMA_MemoryInc_Disable;
DMA_Instructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_Word;
DMA_Instructure.DMA_MemoryDataSize = DMA_MemoryDataSize_Word;
DMA_Instructure.DMA_Mode = DMA_Mode_Circular;
DMA_Instructure.DMA_Priority = DMA_Priority_High;

DMA_Init(DMA1_Channel1, &DMA_Instructure);
```

DMA_Configure 함수는 DMA_InitTypeDef 구조체를 사용하여 DMA(Direct Memory Access)를 초기화하여 ADC 데이터를 CPU 개입 없이 메모리에 저장하도록 설정한다.

•Peripheral Base Address 설정: DMA 가 데이터를 가져올 ADC 데이터 레지스터 주소를 설정

•Memory Base Address 설정: ADC 데이터가 저장될 메모리 주소를 지정

→ 이 전역 변수는 volatile 로 선언하여 DMA 에 의한 메모리 변화를 보장

•Increment Mode 설정: 메모리 주소 증가를 비활성화하여 ADC 값이 항상 동일한 메모리 위치에 저장되도록

•Data Size 설정: 주변 장치(ADC) 데이터 크기를 Word(32-bit)로 설정

•DMA Mode: 강의자료 p8 에 따라, 데이터 전송이 끝나면 시작 주소로 돌아가 반복적으로 데이터 갱신하도록 circular mode 설정

•DMA 우선순위: DMA 채널의 우선순위를 높게 설정하여 데이터 전송이 빠르게 이루어지도록

•DMA Enable: DMA1 채널 1 을 활성화하여 ADC 데이터를 읽을 준비를 완료

3) main 함수

```
int flag = 0;
    LCD_ShowString(LCD_TEAM_NAME_X+50, LCD_TEAM_NAME_Y, "WED_Team02", WHITE, GRAY);
    LCD_ShowNum(LCD_TEAM_NAME_X+50, LCD_TEAM_NAME_Y+50, ADC_Value, 4, WHITE, GRAY);

    while(1){
       if(ADC_Value <= 3400){
         if(flag == 1){
           LCD_Clear(GRAY);
           flag = 0;
         }
         LCD_ShowString(LCD_TEAM_NAME_X+50, LCD_TEAM_NAME_Y, "WED_Team02", WHITE, GRAY);
         LCD_ShowNum(LCD_TEAM_NAME_X+50, LCD_TEAM_NAME_Y+50, ADC_Value, 4, WHITE, GRAY);
       }

       else{
         if(flag == 0){
           LCD_Clear(WHITE);
           flag = 1;
         }
         LCD_ShowString(LCD_TEAM_NAME_X+50, LCD_TEAM_NAME_Y, "WED_Team02", GRAY, WHITE);
         LCD_ShowNum(LCD_TEAM_NAME_X+50, LCD_TEAM_NAME_Y+50, ADC_Value, 4, GRAY, WHITE);
       }

       Delay();
    }
```

Main 함수는 조도 센서 값(ADC_Value)에 따라 LCD 배경색과 글자색을 동적으로 변경한다.

- TFT-LCD 에 "WED_Team02"과 ADC 값을 출력

•ADC_Value <= 3400: 배경색 GRAY, 글자색 WHITE

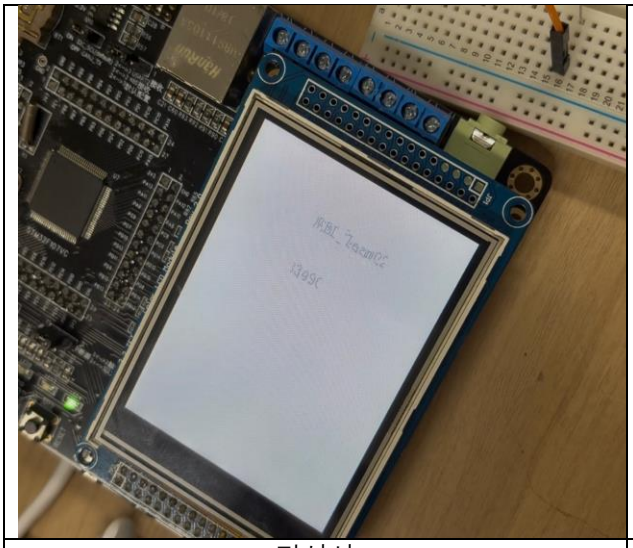
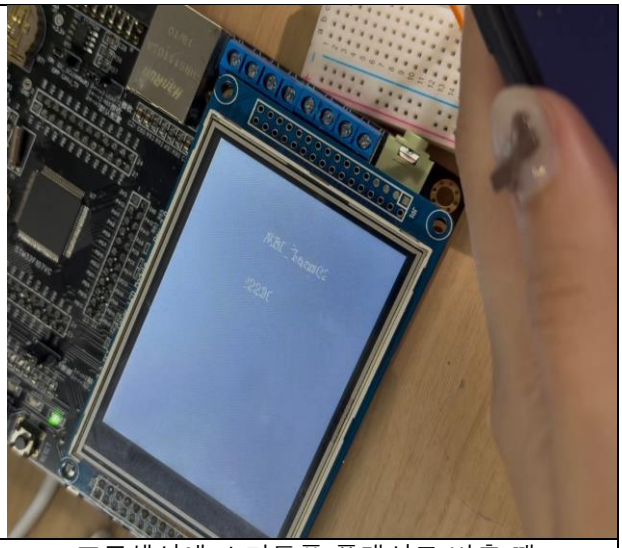
•ADC_Value > 3400: 배경색 WHITE, 글자색 GRAY

2. 회로연결

*10 주차 강의 자료 p13*에 있는 회로와 동일하게 TFT-LCD 를 연결한다.

*10 주차 강의 자료 p17*에 있는 회로와 동일하게 조도센서를 연결한다.

6. 실험 결과

	
평상시	조도센서에 스마트폰 플래시로 비출 때

7. 분석 및 결론

이번 실험을 통해 DMA 와 ADC 를 활용한 데이터 처리 및 TFT-LCD 제어에 대해 알아보았다. DMA 를 이용해 ADC 데이터 전송 과정에서 CPU 의 개입을 최소화하여 시스템 성능을 최적화했으며, Circular Mode 를 활용해 지속적으로 조도 데이터를 갱신함으로써 실시간 데이터 처리 환경을 구축하였다. 또한, 조도 센서 데이터를 기반으로 LCD 의 배경색과 글자 색상을 실시간으로 변경하였다. 이러한 과정은 DMA 와 ADC 의 연계성과 실시간 데이터 처리의

중요성을 실감하게 했으며, 임베디드 시스템 설계에서 자원의 효율적인 활용 방안을 학습하는 계기가 되었다.