**임베디드 시스템 설계 및 실험**

**12주차 실험 결과 보고서**

**조 : 8조**

**조원 : 서진욱, 이승민, 하연지, 하태훈**

**● 실험목표**

* **DMA에 대한 개념을 이해하고, 이를 활용하여 조도센서와 TFT-LCD를 사용하는 실습을 수행한다.**

**● 개념설명**

1. **TFT-LCD(10주차, 11주차 활용)**
   * 초 박막 액정 표시장치
   * 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 초박형 유리 기판 위에 반도체 막을 형성한 회로인 TFT 기판
   * Filter와 기판 사이에 주입된 액정과 광원인 Black light unit으로 구성

컴퓨터 구성 요소, 전자제품이(가) 표시된 사진

자동 생성된 설명

그림 . TFT-LCD

1. **조도센서(10주차 내용 참고)**
   * 주변의 밝기를 측정하는 센서
   * 빛의 양이 많아질수록 전도율이 높아져 저항이 낮아짐



그림 2. 조도 센서

1. **DMA(Direct Memory Access)**
   * 주변장치들이 메모리에 직접 접근하여 읽거나 쓸 수 있도록 하는 기능
   * CPU 의 개입 없이 I/O 장치와 기억장치 데이터를 전송하는 접근 방식
   * Interrupt 와 달리 별도의 중앙제어장치는 명령을 실행할 필요가 없음
   * 메모리 처리 Interrupt 의 사이클 만큼 성능의 향상

텍스트, 스크린샷, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명

그림 3. Direct Memory Access(DMA)

* + **DMA 방식의 메모리 접근**
    - RAM이 I/O 장치로부터 데이터가 필요해지면, CPU는 DMA 컨트롤러에게 신호(전송 크기, 주소 등등)를 보냄
    - DMA 컨트롤러가 RAM 주소로 데이터를 bus를 통해 주고 받음
    - 모든 데이터 전송이 끝나면, DMA Controller가 CPU에게 Interrupt 신호를 보냄

텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

그림 4. DMA 방식의 메모리 접근

* + **DMA Channel**
    - 모듈은 DMA Controller 의 DMA 채널을 통해 메모리 R/W
    - STM32 보드 DMA 채널은 총 12 개
      * DAM1 채널 7개, DMA2 채널 5개
    - 한 DMA의 여러 채널 사이 요청은 Priority에 따라 동작
      * 4 level: very high, high, medium, low
    - Peripheral-to-memory, memory-to-peripheral, and peripheral-to-peripheral 전송
  + **DMA Mode**
    - **Normal Mode**
      * DMA Controller 는 데이터를 전송할 때 마다 NDT 값을 감소시킴
      * NDT 는 DMA 를 통해 전송할 데이터의 총 용량을 의미하며 레지스터의 값이 0이 되면 데이터 전송 중단
      * 데이터 전송을 받고 싶을 때 마다 새롭게 요청이 필요
    - **Circular Mode**
      * 주기적인 값의 전송(업데이트)이 필요할 때 사용하는 모드
      * NDT 값이 0이 될 경우 설정한 데이터 최대 크기로 재설정됨
  + **DMA Controller**
    - 주변 장치의 Request Signal 의 발생
      * DMA Controller 에서 우선순위 설정 및 요청에 대한 서비스 제공
      * Request / ACK 방식을 통한 주변 장치와 DMA Controller 간 통신

**● 실험 진행**

* **코드 작성**
  + 전역변수 선언
    - (코드 1)ADC의 값을 저장하기 위한 배열 ADC\_Value를 volatile 전역변수로 선언해 준다.



**코드 1. volatile ADC\_Value**

* + RCC\_Configure
    - (코드 2)ADC1과 DMA1의 RCC를 ENABLE하는 코드로 조도센서 값을 변환하기 위해 ADC1을 사용
    - Channel 8번을 사용하기 위해 ABP2Peripheral의 ADC1과 Port B를 Alternate Function I/O로 설정하고 AHBPeripheral의 DMA1을 ENABLE

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

코드 2. RCC\_Configure

* + GPIO\_Configure
    - (코드 3)GPIO 구성에 대한 코드로 ADC1의 Channel8을 사용하고자 함
    - PB0를 ENABLE로 설정
    - 조도센서의 값을 읽어오는 것이므로 Analog Input모드로 설정

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

코드 3. GPIO\_Configure

* + ADC\_Configureation
    - (코드 4)ADC 구성에 관한 코드로 ADC1 한 개만 사용하며 Independent 모드로 설정하고, ADC1 한 개만 사용할 것이므로 ScanConvMode는 DISABLE
    - 조도센서에서 값을 지속적으로 불러올 것이므로 ContinuousConvMode는 ENABLE
    - 외부 트리거 없이 계속해서 값을 읽어 변환할 것이므로 ExternalTrigConv는 ADC\_ExternalTrigConv\_None으로 할당
    - 변환 결과를 쉽게 처리하기 위하여 DataAlign은 Right로 설정
    - ADC1의 Channel8만을 사용하므로 ADC\_NbrChannel은 1을 할당
    - RegularChannelConfig에서는 ADC1, Channel8, 채널 1개만을 사용하고, 변환한 값의 정확도를 높이기 위하여 샘플링 시간은 가장 긴 SampleTime인 239Cycles5(총 244Cycles)으로 설정
    - DMA로 조도센서의 값을 쓰고 읽을 것이므로 ADC\_DMACmd()를 이용하여 ADC1를 ENABLE하고 ACD\_Cmd()를 이용하여 ADC1을 ENABLE
    - 먼저 ADC\_ResetCalibration()으로 ADC1의 Calibration을 RESET하고 Calibration Reset이 완료되면 ADC\_StartCalibration()으로 ADC1의 Calibration을 시작
    - ADC\_GetCalibrationStatus()가 ADC1의 Calibration이 종료됨을 반환하면 ADC\_SoftwareStartConvCmd()로 ADC1을 ENABLE하여 Conversion 시작

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

코드 4. ADC\_Configure

* + DMA\_Configure
    - (코드 5)DMA 설정을 위한 코드
    - 사용할 DMA 채널인 DMA1번 채널을 초기화 한 후 ADC1의 레지스터 주소를 기본주소로 설정하여 값을 읽어올 수 있도록 설정
    - DMA\_DIR을 DMA\_DIR\_PeripheralSRC 로 설정해줌으로써 DMA가 ADC에서 데이터를 읽어와 메모리로 전송할 수 있도록 설정
      * 설정 후 전송할 데이터의 버퍼를 1로 설정
    - 사용할 주변장치가 조도센서 하나밖에 없으므로 Peripheral 의 주소를 계속 증가시키는 DMA\_InitStructure.DMA\_peripheralInc 를 DISABLE
    - 반면에 메모리 주소 레지스터는 효율을 위해 계속 증가시킬 수 있도록

DMA\_InitStructure. DMA\_MemoryInc는 ENABLE

* + - PeripheralData의 사이즈는 Halfword, 즉 2바이트로 설정
    - MemoryData의 사이즈는 word, 즉 4바이트로 설정
    - DMA\_InitStructure.DMA\_Mode를 DMA\_Mode\_Circular로 설정하여 DMA가 순환모드로 동작하여 데이터를 반복, 연속적으로 보낼 수 있도록 설정
    - DMA의 전송 우선 순위를 High로 설정
    - 메모리간 전송 기능인 DMA\_M2M를 disable로 설정
    - 그 후 DMA\_Init을 통해 초기화를 진행하고 DMA1\_Channel1을 활성화

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

코드 5. DMA\_Configure

* + main
    - 각 함수들을 불러오고 LCD화면을 WHITE으로 초기화
    - While 문 안에서는 LCD에 50, 50좌표에 ADC에서 읽어온 값을 계속 refresh 해주고, 조건문을 통해 전역변수로 선언해둔 ACD\_Value의 값을 사전에 정의해둔 threshold와 비교
      * threshold보다 더 작을 경우: LCD 배경을 GRAY로 변환하고, 글자색을 YELLOW로 변환
      * 그 외(반대 case): WHITE 배경에 BLACK 글자로 출력되도록 설정
    - Threshold 값은 최초에는 임의로 지정한 후, 보드에 포팅 후 직접 조도센서에 빛을 비추어 보면서 LCD 화면에 나타나는 값을 추출해내어 다시 설정
      * Threshold 값 : 3900 설정

텍스트, 스크린샷, 폰트, 문서이(가) 표시된 사진

자동 생성된 설명

코드 . main 실행 코드

**● 결과 및 결론**

* **실험 결과**

**1) 초기 및 평상 시 결과**

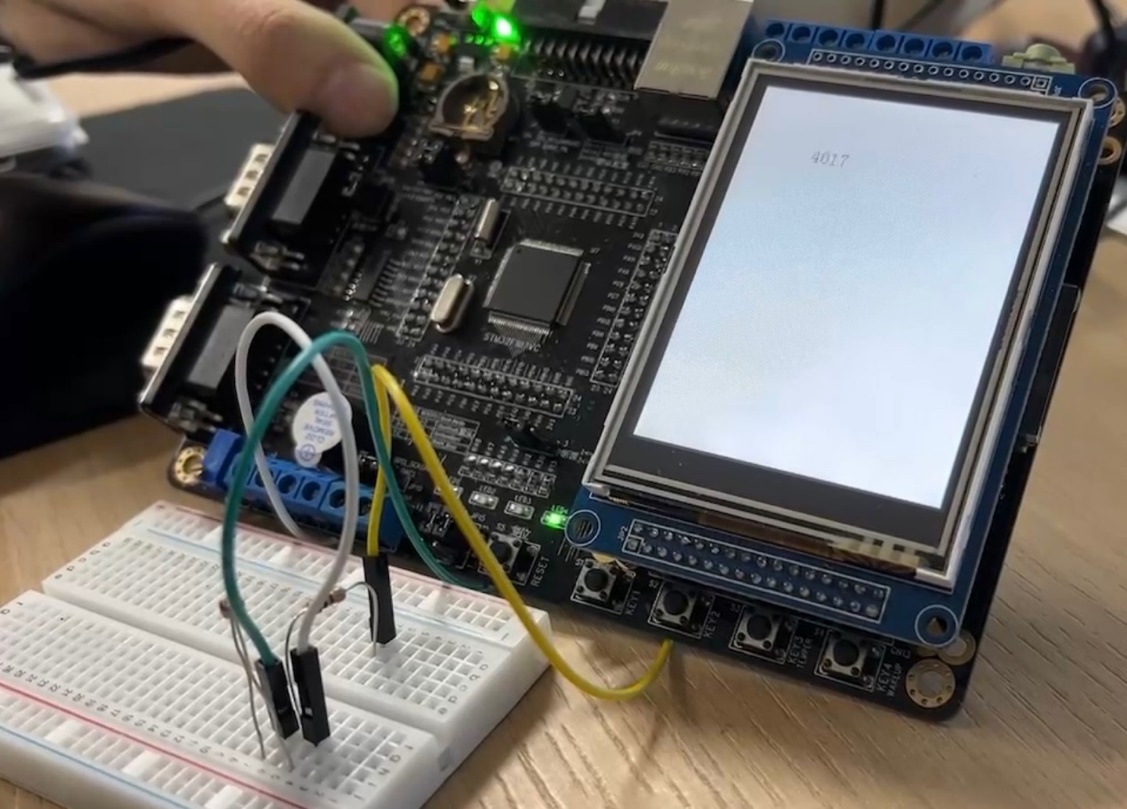


그림 5. 초기 작동(평상시) TFT-LCD 상태

* + 조도 센서의 값을 TFT-LCD에 출력
  + 배경 : WHITE 출력
  + 조도 센서 값(일반적 case) : 4000~4100

**2) 플래시를 비출 경우**

전자제품, 전자 공학, 컴퓨터 하드웨어, 전자 기기이(가) 표시된 사진

자동 생성된 설명

그림 6. 플래시를 비출 경우 TFT-LCD 상태

* + 플래시를 비춘 경우: 3800 내외의 값 확인
  + TFT-LCD 상태 변경 확인
    - 배경색 : WHITE → Gray
    - 글자색 : Yellow로 변경
* **결론**
  + DMA는 CPU에 인터럽트 신호를 보내지 않고도 데이터 통신을 가능하게 한다는 것을 알 수 있었다.
  + 조도센서 값의 임계치를 실험을 통해 결정하며 센서 데이터에 대한 적절한 처리방법을 학습할 수 있었다.