**12주차 결과보고서**상징, 로고, 폰트, 원이(가) 표시된 사진

자동 생성된 설명

10조 (001분반)

과목명 : 임베디드시스템설계및실험

담당 교수 : 정상화 교수님

담당 조교 : 최호진 조교님

조원 : 201924603 하규승(조장)

201727102 강준혁

201924525 이광훈

202023139 박지원

제출 날짜 : 2023.11.26

**목차**

1. 실험 목표

2. 세부 목표

3. 실험 기구

4. 실험 과정

5. 실험 결과

6. 결과에 대한 논의

**1. 실험목표**

**1-1. DMA(Direct Memory Access)**

텍스트, 스크린샷, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명

그림 1-1. DMA 동작

주변장치들이 메모리에 직접 접근하여 읽거나 쓸 수 있도록 하는 기능이다. CPU의 개입 없이 I/O 장치와 기억장치 데이터를 전송하는 접근방식이다. DMA 방식을 통해 프로그램 수행 중 인터럽트의 발생을 최소화하고 시스템의 효율성을 높인다. 이때 CPU는 DMA 컨트롤러와 상태정보 및 제어정보만을 전달한다.

**1-2. DMA vs Interrupt**

텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

1. DMA

* RAM이 I/O 장치로부터 데이터가 필요해지면, CPU는 DMA 컨트롤러에게 신호(전송 크기, 주소 등)를 보낸다.
* DMA 컨트롤러가 RAM 주소로 데이터를 bus를 통해 주고받는다.
* 모든 데이터 전송이 끝나면, DMA Controller가 CPU에게 Interrupt 신호를 보낸다.
* 텍스트, 스크린샷, 도표, 라인이(가) 표시된 사진

  자동 생성된 설명

1. Interrupt

* 모든 I/O 로의 접근은 CPU를 통해서 수행한다.
* Data를 전달할 때마다 CPU가 관여한다.
* 오버헤드가 발생한다.

**1-3. DMA Channel**

DMA 채널을 통해 메모리를 R/W 한다. DAM1 채널 7개, DMA2 채널 5개로 구성되어 있으며, 한 DMA의 여러 채널 사이 요청은 Priority에 따라 동작한다.

텍스트, 번호, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

그림 1-3. ADC의 각 과정 DMA1 Channel

실험에서는 ADC1으로 동작을 수행하므로 [그림2.4]와 같이 DMA1의 Channel1을 사용한다.

**1-4. DMA Mode and DMA Controller**

a. Normal Mode

- DMA Controller 는 데이터를 전송할 때마다 NDT 값을 감소시킴

- NDT는 DMA를 통해 전송할 데이터의 총량을 의미하며 레지스터의 값이 0이 되면 데이터 전송을 중단한다.

- 데이터 전송을 받고 싶을 때 마다 새롭게 요청이 필요

b. Circular Mode

- 주기적인 값의 전송(업데이트)이 필요할 때 사용하는 모드

- NDT 값이 0이 될 경우 설정한 데이터 최대 크기로 재설정됨

c. DMA Controller

주변 장치의 Request Signal이 발생하면, DMA Controller 에서 우선순위 설정 및 요청에 대한 서비스 제공을 수행한다. Request / ACK 방식을 통해 주변 장치와 DMA Controller 간 통신이 이루어진다.

**1-5. DMA 동작순서**

텍스트, 도표, 평면도, 개략도이(가) 표시된 사진

자동 생성된 설명

그림 1-5. DMA Block Diagram]

a. 이벤트 발생

b. Peripheral은 DMA Controller에게 Request Signal을 보냄

c. DMA Controller는 채널 우선순위에 따라 요청 처리

d. DMA Controller가 Peripheral로 접근하고 Peripheral로 Ack 신호를 보냄

e. Ack 신호를 받은 Peripheral은 해당 요청에 따라 전송

**2. 세부 목표**

2-1. DMA 및 ADC를 사용하여 1개의 조도센서 값을 LCD에 출력

2-2. threshold에 따라 LCD의 배경색을 변경

**3. 실험 기구**

* STM32F107 보드
* TFT LCD
* 조도센서
* 납땜 기구

**4. 실험 과정**

**4-1. 프로젝트 세팅**

회로, 축적 모형, 실내이(가) 표시된 사진

자동 생성된 설명

**4-1-1.** LCD 기판을 보드의 핀과 정확히 맞도록 연결한다.

텍스트, 도표, 라인, 스크린샷이(가) 표시된 사진

자동 생성된 설명

**4-1-2.** 조도 센서와 보드를 연결한다. 연결 회로는 위 회로도를 참고한다. 해당 실험에서 보드의 핀은 A0 핀을 사용한다. 회로에 저항을 사용하지 않을 경우 조도 센서 값이 제대로 표시되지 않을 수 있음에 유의한다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**4-1-3.** LCD 사용에 필요한 소스 및 헤더파일을 10주차 파일을 참고하여 프로젝트 디렉토리 내의 라이브러리 폴더에 넣는다. 경로는 아래와 같다.

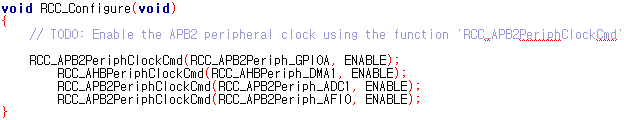


**4-2. main.c 구현**

****

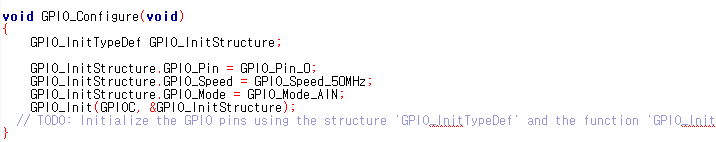
**4-2-1. Global Variable ADC\_Value[1];**

ADC\_Value의 값을 저장할 전역변수를 선언한다.

****

**4-2-2. RCC\_Configure method**

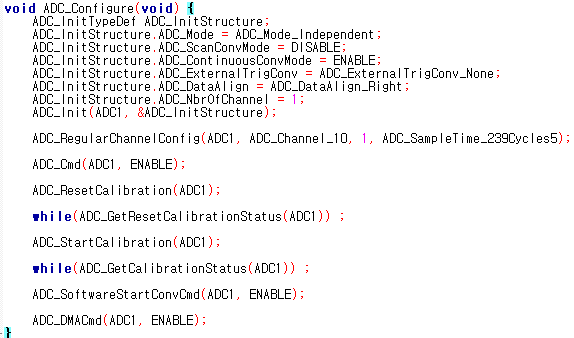
조도센서를 연결할 포트와 ADC 레지스터, 그리고 DMA레지스터와 AFIO의 RCC를 활성화한다.

****

**4-2-3. GPIO\_Configure method**

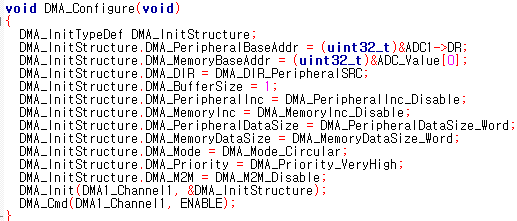
조도 센서와 연결된 핀(C0)의 GPIO를 활성화한다.

이때 조도 센서의 입력 값은 아날로그이기 때문에, GPIO 모드를 analog input으로 설정한다.

****

**4-2-4. ADC\_Configure method**

10주차의 ADC설정을 참고하여 ADC를 활성화한다. 이후 ADC의 값을 읽어오는 동작을 구현한다.

****

**4-2-5. DMA\_Configure method**

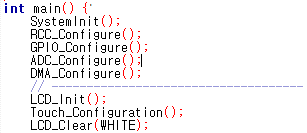
DMA 레지스터를 사용하기 위해 DMA\_Init을 이용해 설정한다. 이후, DMA1 채널을 DMA\_Cmd를 이용해 활성화 한다. DMA 구조체의 변수 및 설정 값에 대해서는 아래를 참조한다.

|  |  |  |
| --- | --- | --- |
| 구조체 변수 | 설정값 | 설명 |
| DMA\_PeripheralBaseAddr | (uint32\_t)&ADC1->DR | 사용한 DMA의 Channel을 지정 |
| DMA\_MemoryBaseAddr | (uint32\_t)&ADC\_Value[0] | 지정한 DMA를 위한 메모리 주소 지정 |
| DMA\_DIR | DMA\_DIR\_PeripheralSRC | Peripheral이 Source가 될지  Destination이 될지 지정 |
| DMA\_BufferSize | 1 | DMA의 버퍼 사이즈 지정 |
| DMA\_PeripheralInc | DMA\_PeripheralInc\_Disable | DMA갱신시 Peripheral주소 증가 여부 |
| DMA\_MemoryInc | DMA\_MemoryInc\_Disable | DMA갱신시 Memory주소 증가 여부 |
| DMA\_PeripheralDataSize | DMA\_PeripheralDataSize\_Word | Peripheral data의 크기 |
| DMA\_MemoryDataSize | DMA\_MemoryDataSize\_Word | Memory data의 크기 |
| DMA\_Mode | DMA\_Mode\_Circular | DMA의 동작 모드를 지정 |
| DMA\_Priority | DMA\_Priority\_VeryHigh | DMA의 동작 우선도를 지정 |
| DMA\_M2M | DMA\_M2M\_Disable | DMA채널이 Memory-to-Memory 전송으로 실행될 지 지정 |

****

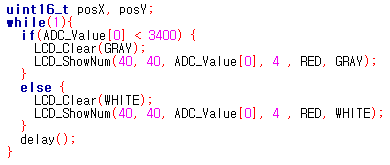
**4-2-6. delay method**

화면이 과도할 정도로 갱신되는 것을 막기 위해 for문을 이용하여 delay동작을 구현한다.



**4-2-7. main/Configure init**

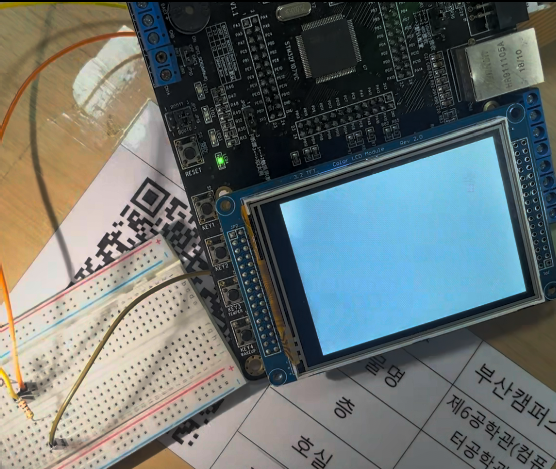
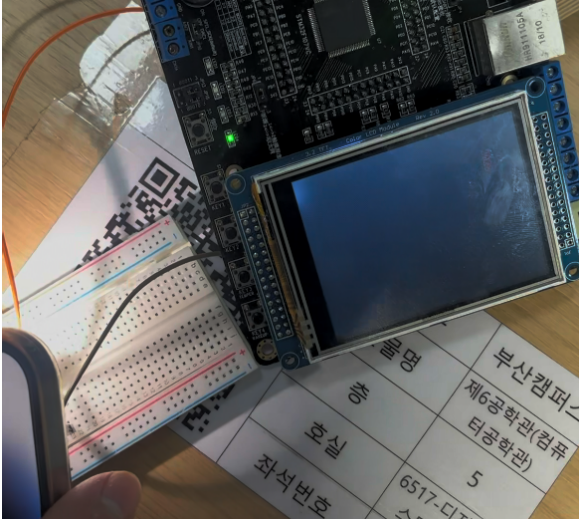
선언된 configure function들을 호출한다.



**4-2-8. main/while 문 작성**

LCD에 표시할 정보들을 입력한다.

**5. 결과**

LCD에 정상적으로 조도센서의 값이 출력되었다. 조도센서의 밝기 변화에 따라 화면이 회색과 흰색으로 바뀌며, 조도센서의 값 또한 바뀜을 확인할 수 있었다.

**6. 결과에 대한 논의**

보드에 전원을 인가했을 때 TFT-LCD에 조도센서의 값과 배경화면의 색상이 정상적으로 출력되는 것으로 보아 미리 정의된 함수들을 통해 각 포트들이 잘 설정되었음을 알 수 있다.

조도센서의 경우엔 10주차와 같이 값의 편차가 3000 ~ 4000 정도로 작은 편이긴 했으나, 조명을 비추었을 때 값이 낮게 출력되고 배경색이 검정색으로 변하고, 조명을 치웠을 때 값이 높게 출력되며 배경색이 흰색으로 잘 변하는 것을 보아 ADC와 조도센서가 정상적으로 동작함을 알 수 있다.

**7. 결론**

이번 주차에서는 TFT-LCD에 조도센서 입력 값 출력 및 입력 값에 따른 배경색 변화 동작을 수행하게 하였다. 10주차에도 느꼈지만, 조도 센서는 생각보다 값이 잘 변하지 않음을 느꼈고, 이를 통해 해당 센서를 다룰 때는 값을 좀 더 세밀하게 조정하여 사용하는 게 좋다는 것을 알 수 있었다.

그리고 DMA를 통한 정보 갱신주기는 굉장히 짧다는 것을 알 수 있었고, 이를 통해 DMA로 받아온 정보를 갱신 될 때 마다 표시할 때는 갱신 직후에 표시하는 것이 아니라 delay를 구현하여 일정 주기마다 표시하게 하는 것이 좀 더 좋다는 것을 알 수 있었다.