임베디드시스템 설계 및 실험

11 주차 결과보고서

2023.11.20

003 (수) 분반

8 조

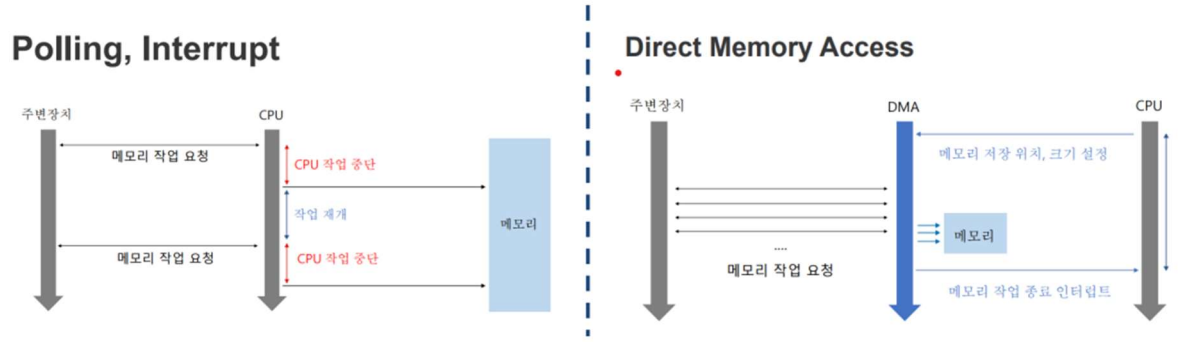
201924437 김윤하

202012142 이은진

202055604 조찬우

202155565 성가빈

1. DMA에 대한 이해



DMA는 주변장치들이 메모리에 직접 접근하여 읽거나 쓸 수 있도록 하는 기능으로, CPU 의 개입 없이 I/O 장치와 기억장치 데이터를 전송하는 접근 방식인 Interrupt 와 달리 별도의 중앙제어장치는 명령을 실행할 필요가 없고 메모리 처리 Interrupt 의 사이클 만큼 성능의 향상된다.

DMA Channel

모듈들은 DMA Controller의 DMA Channel을 통해서 메모리에 접근하는데, STM32보드의 DMA Channel은 12이다.

DMA Mode

DMA는 2가지 모드로 동작하는데(Normal Mode, Circular Mode) Normal 모드에서는 DMA Controller가 데이터를 전송할 때마다 DMA를 통해 전송할 총 용량인 NDT를 감소시킨다. NDT가 0이 되면 데이터 전송이 중단되므로 새로운 요청이 필요하다. CircularMode는 주기적인 값의 전송이 필요할때로, NDT가 0이 될경우 설정한 데이터 최대 크기로 자동으로 재설정된다.

DMA Controller

DMA Controller는 DMA request가 생기면 우선순위를 설정하고 request에 대한 서비스를 제공한다.

2. 실험 내용

1. DMA 및 ADC 를 사용하여 1 개의 조도센서 값을 받아오도록 구현

2. 읽은 조도센서 값을 TFT-LCD 에 출력

3. 평상시 TFT-LCD 배경색 WHITE, 조도센서에 스마트폰 플래시로 비출 때 TFT-LCD 배경색 GRAY

- 배경색 바꾸면 글자도 사라지므로 배경 바꾸고 조도 값 띄우기

4. 조도센서 값 threshold 는 각자 실험적으로 결정

3. 실험 보고서

```
#include "stm32f10x.h"
#include "core_cm3.h"
#include "misc.h"
#include "stm32f10x_gpio.h"
#include "stm32f10x_rcc.h"
#include "stm32f10x_usart.h"
#include "stm32f10x_adc.h"
#include "stm32f10x_dma.h"
#include "lcd.h"
#include "touch.h"
```

그림1. 실험에 필요한 라이브러리

이번 과제는 DMA를 사용해야하기 때문에 음 관련 라이브러리를 include 하였다.

```
int color[12] = {WHITE,CYAN,BLUE,RED,MAGENTA,LGRAY,GREEN,YELLOW,BROWN,BRRED,GRAY};
volatile uint32_t ADC_value[1];
```

그림2. 조도 센서, 색 목록

조도 센서에 들어온 측정치는 ADC_value로 받는데,전역변수로 선언한 ADC 값을 저장할 공간을 항상 참조하도록 volatile 키워드 이용하였다.

```
void RCC_Configure(){
 RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
 RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);
 RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1, ENABLE);
}
```

그림3. 사용한 RCC클락과 핀을 설정

DMA1, ADC1, 포트A에 대한 클럭을 configure 한다.

```
void GPIO_Configure(){
 GPIO_InitTypeDef GPIO_InitStructure;
 GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1;
 GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
 GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
 GPIO_Init(GPIOA, &GPIO_InitStructure);
}
```

그림4. GPIO를 설정

```
void ADC_Configure(){
 ADC_InitTypeDef ADC_InitStructure;

 ADC_DeInit(ADC1);
 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_1, 1, ADC_SampleTime_239Cycles5);
 ADC_DMACmd(ADC1, ENABLE);
 ADC_Cmd(ADC1, ENABLE);

 ADC_ResetCalibration(ADC1);
 while(ADC_GetResetCalibrationStatus(ADC1));
 ADC_StartCalibration(ADC1);
 while(ADC_GetCalibrationStatus(ADC1));
 ADC_SoftwareStartConvCmd(ADC1, ENABLE);
}
```

그림5. 조도 센서 설정

조도 센서가 받은 밝기를 디지털 값으로 받기위해서 ADC를 설정하는 함수이다. 이전에는 ADC_ITConfig를 사용했지만 이번에는 ADC_DMACmd 함수를 이용하여 설정한다.

```
void DMA_Configure() {
DMA_InitTypeDef DMA_InitStructure;

DMA_DeInit(DMA1_Channel1);
DMA_InitStructure.DMA_BufferSize = 1;
DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;
DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
DMA_InitStructure.DMA_MemoryBaseAddr = (uint32_t)&ADC_value[0];
DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_Word;
DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Disable;
DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
DMA_InitStructure.DMA_PeripheralBaseAddr = (uint32_t)&(ADC1->DR);
DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_Word;
DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
DMA_InitStructure.DMA_Priority = DMA_Priority_VeryHigh;

DMA_Init(DMA1_Channel1, &DMA_InitStructure);
DMA_Cmd(DMA1_Channel1, ENABLE);
}
```

그림5. DMA설정

사전에 정의되어있는 DMA함수에 접근하여 DMA를 새롭게 정의한다. 12개의 채널 중 DMA1에 속한 7개의 채널에서 DMA1_Channel1을 사용한다. 이를 이용해서 ADC_value를 DMA_MemoryBaseAddr를 사용하여 가져온다. DMA_Mode는 Circular로 하여 NDT 값이 0이 되면 자동으로 최대크기로 설정되게한다.

```
void delay() {
 for (int i=0; i<1000000; i++);
}
```

그림6. 딜레이

실험을 가시적으로 확인 할 수있게 딜레이 함수를 세팅해놓는다.

```
int main() {
 SystemInit();
 RCC_Configure();
 GPIO_Configure();
 ADC_Configure();
 DMA_Configure();
 // ------------------------------
 LCD_Init();
 Touch_Configuration();
 Touch_Adjust();
 LCD_Clear(WHITE);

 LCD_ShowNum(50, 20, (uint32_t)ADC_value[0], 5, BLACK, WHITE);

while(1){
if ((uint32_t)ADC_value[0] >= 3600) {
 LCD_Clear(WHITE);
 LCD_ShowNum(50, 20, (uint32_t)ADC_value[0], 5, BLACK, WHITE);
}
else {
 LCD_Clear(GRAY);
 LCD_ShowNum(50, 20, (uint32_t)ADC_value[0], 5, WHITE, GRAY);
}
 delay();
}


}
```

그림7. main함수

SystemInit()부터 DMA_Configure()까지 핀들에 대한 클락을 설정해놓고, LCD를 설정하기 위해서 LCD_Init(), Touch_Configuration(), Touch_Adjust()를 실행하고 흰색 바탕으로 화면을 설정하기 위해서 LCD_Clear(WHITE)를 실행한다. 이후 다음과 같이 코드를 설정하였다.

1. ADC_value가 3600이상이 되면 흰색 바탕에 검은색으로 표시하고,
2. ADC_value가 그보다 작다면 회색 바탕에 하얀색으로 표시되게 한다.
3. LCD_ShowNum으로 글시의 좌표, 색, 배경을 결정한다.

4. 실험 결과

그림7. 화면 전환 중, 조도 센서가 높은 경우, 조도센서 값이 낮은 경우

5. 느낀 점

이번 실험은 Interrupt를 DMA로 바꾸고 이미 했었던 LCD 조작, 조도센서 납땜을 실행하는 것이기 때문에 어려운 점은 없었다. 가끔씩 LCD가 오류가 났었던 점을 제외하면 빠르게 끝났었다.