임베디드 시스템 설계 및 실험

3주차

7조
201424470 서민영
201424421 김시은
201424533 정종진

목차

1. 실험목표

1) 디버깅 툴(DS-5, DSTREAM) 사용방법 숙지
2) 레지스터와 주소 제어를 통한 임베디드 펌웨어 개발 (조이스틱 조작을 통한 LED 제어)

2. 배경지식

1) 데이터베이스 기반의 연결의 장점과 이유에 대해 조사
   - 데이터베이스 기반의 연결의 장점
     - 다수의 요구 처리, 응답
     - 지속적인 최신화
     - 논리적, 물리적 독립성
     - 쉬운 데이터 액세스
   - 데이터베이스 기반의 연결의 이유
     - 제한적인 하드웨어 자원으로 인해 물리적 독립성을 갖기 위함
     - 통합 데이터를 지원하여 빠른 응답 보장

2) 멀티미터 사용법
   - 검정색 프로브를 검정색 단자에 연결하고, 측정하고 싶은 것과 Range에 따라 빨간색 프로브를 빨간색 단자에 연결하여 측정

3) PC vs Imbedded System
   - PC
     - 키보드 또는 마우스와 같은 입력장치를 이용해 사용자의 명령을 받음
     - 32bit 또는 64bit와 같은 고성능 CPU 사용, 여러가지 프로그램 실행
     - 모니터 또는 스피커와 같은 정형화된 주변장치를 이용해 출력
   - Imbedded System
     - 다양한 센서를 통해 정보를 입력 받음
     - 저전력, 저비용 CPU 사용, 특정 기능만 담당하는 펌웨어 하나 실행
     - 다양한 종류의 구동 장치를 통해 출력

4) Cortex 계열 특징
   - Cortex는 ARM Architecture의 어플리케이션 프로세서
   - Cortex-M은 마이크로 컨트롤러, AISC, ASSP, FPGA 및 SoC용으로 설계된 ARM 마이크로 프로세서 코어, 32비트 연산 기능을 활용하고 ARM7 및 ARM9와 같은 기존 레거시 ARM 코어를 대체하는 어플리케이션에서 8비트 칩으로 널리 보급
   - Cortex-A32를 제외한 32비트 ARM Cortex-A 코어는 ARMv7 아키텍처의 ARMv7-A 프로필을 구현, ARMv7-A만 MMU를 포함

5) JTAG 스펙 및 사용법
   - JATG(Joint Test Action Group)는 디지털 회로에서 특정 노드의 디지털 입출력을 위해

직렬 통신 방식으로 출력 데이터를 전송하거나 입력데이터를 수신하는 방식

- 임베디드 시스템을 개발하기 위해 통합한 회로로 사용되며, CPU의 기계어 코드 실행없이 MCU 내부의 플래시 메모리나 임베디드 장치에서 CPU의 외부 플래시 메모리에 코드를 쓰거나 읽을 수 있다.
- TDI(데이터 입력), TDO(데이터 출력), TCK(클럭), TMS(모드), TRST(리셋) 핀으로 구성
- CPU의 상태와는 상관없이 디바이스의 모든 외부 핀을 구동 시키거나 값을 읽어 들일 수 있는 기능을 제공

6) DSTREAM 스펙 및 사용법

- DSTREAM은 모든 ARM Architecture Target에서 강력한 소프트웨어 디버깅 및 최적화를 가능하게 해주는 장비
- JTAG 또는 SWD를 통해 RealView Debugger와 DS-5 Debugger, 그리고 써드파티 Debugger와 연동이 가능
- JTAG 클럭 최대 60MHz 지원
- 300MHz DDR 16bit 와이드 트레이스 캡쳐
- JTAG을 통한 가상 이더넷 링크 지원

7) ARM9 과 Cortex-M3의 차이 및 Cortex-M3의 장점

- ARM9은 PC에서 사용되는 구형 32비트 RISC ARM 프로세서 코어 그룹이고, Cortex-M3는 마이크로 컨트롤러, ASIC, ASSP, FPGA 및 Soc용으로 설계된 어플리케이션에서 사용되는 ARM 프로세서 코어 그룹
- Code Bus, Data Bus, System Bus가 각기 따로 존재하여, 이전에 보다 데이터 읽기/쓰기의 대기시간을 줄임
- 3-stage pipeline with branch prediction
- Cortex-M3는 Thumb2 명령어만 사용할 수 있음
- 정수형 나눗셈이 하드웨어적으로 지원됨

3. 실험과정

1) 보드 연결

<그림 1 : Coretex M3/JTAG/DSTREAM을 연결한 모습>

다음과 같은 순서로 보드를 연결한다. 이 때 연결 및 분리 순서를 제대로 지키지 않으면 장비가 망가질 수 있으니 주의해야한다.

표 1 : 보드 연결 순서

① 보드와 DSTREAM JTAG 연결
② 보드 전원선만 연결
(보드의 전원은 OFF 상태)
③ DSTREAM 전원 연결 및 ON
④ DSTREAM Status LED 점등 확인
⑤ 보드 전원 ON
⑥ DSTREAM Target LED 점등 확인
⑦ DS-5에서 'connect target'

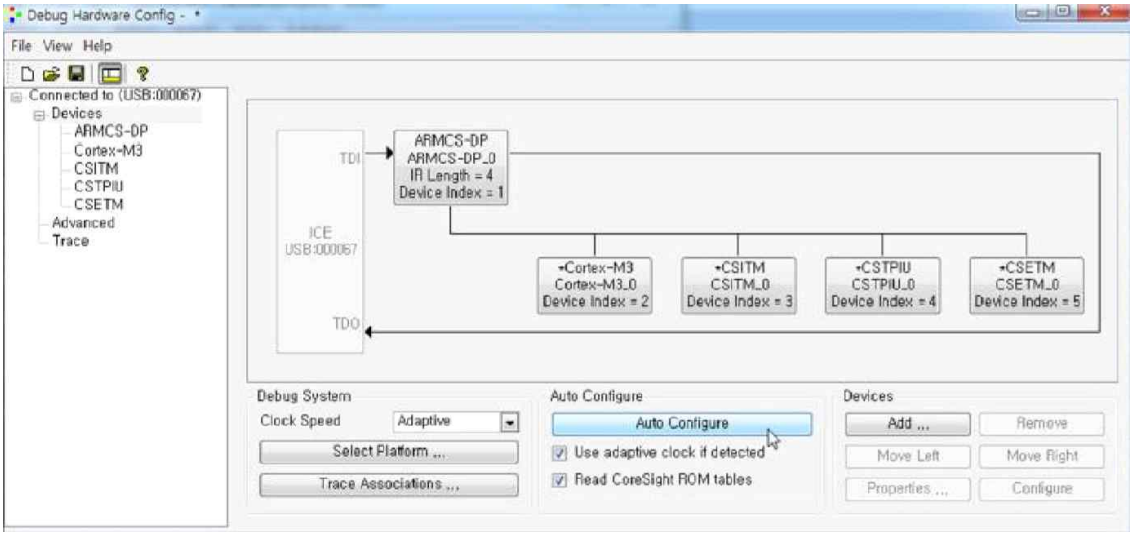
2) DS-5 디바이스 데이터베이스 추가

2-1) 사용자 데이터베이스를 생성 및 저장

① ARM DS-5 프로그램의 Debug Hardware → Debug Hardware Configuration 선택

② Auto Configuration 클릭

③ 사용자 데이터베이스를 저장경로 설정 및 저장(Save as)



<그림 2 : 사용자 데이터베이스를 생성 및 저장한 모습>

2-2) 생성한 DB 연동

① DS-5 폴더 내 bin 폴더(C:\Program Files\DS-5\bin)로 이동한 후 bin 폴더 내에 있는 'cdbimporter.exe'를 실행

② 다음과 같은 명령어를 입력

cdbimporter -t <타겟DB주소> <rvc파일명.rvc>

(-c <메인타겟DB경로>는 디폴트로 설정되어 있기 때문에 생략해도 됨.)

```
C:\Program Files\DS-5\bin>cdbimporter
DS-5 Config Database Import Utility v1.2
Copyright 2011-2014 ARM Ltd


No RVC files specified

usage: cdbimporter -?
usage: cdbimporter [-c source_db] -l
usage: cdbimporter [-c source_db] [-t destination_db] rvc_file
```

<그림 3 : DS-5 prompt 창을 통해 생성한 DB를 연동>

2-3) Eclipse에 데이터베이스 추가

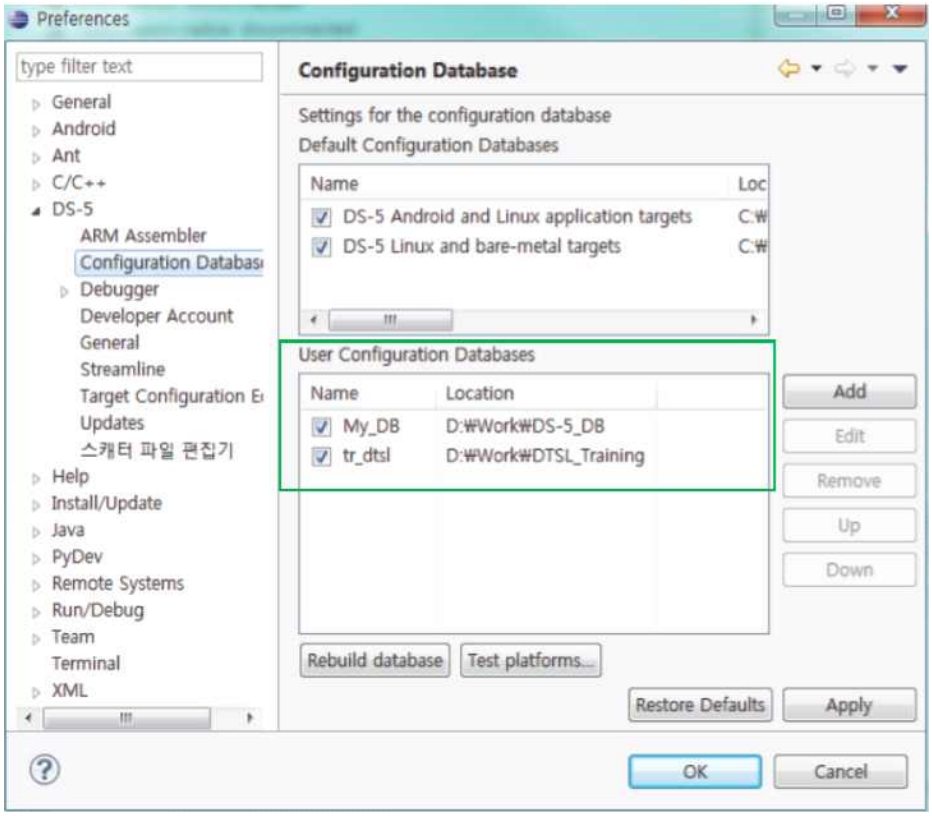
① 시작 → 모든 프로그램 → ARM DS-5 → Eclipse for DS-5 메뉴를 선택

② Windows → Preferences

③ DS-5 → Configuration Database 항목을 선택

④ Add 버튼을 클릭하여 사용자 데이터베이스의 디렉터리를 지정

⑤ Rebuild database 버튼을 클릭하여 데이터베이스 추가를 완료



<그림 4 : Eclipse에 데이터베이스 추가>

3) C Project 생성 및 환경설정

3-1) C Project 생성

① New Project → C project → Excutable → Empty Project를 선택해주고 Toolchains는 ARM Cmpiler 5(DS-5 built-in)로 선택 후 프로젝트 생성

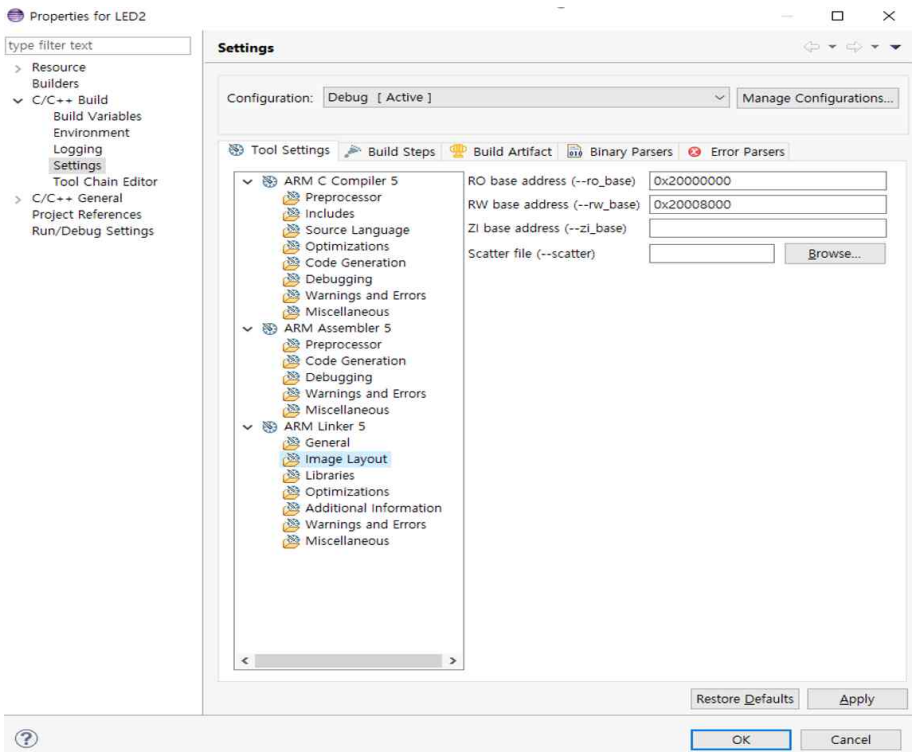
3-2) C Project Properties 설정

① C project 우클릭 후 Properties 선택 → C/C++ Build → Settings → ARM Linker 5 → Image Layout RO base address와 RW base address 값 설정 (0x20000000 ~ 0x2000FFFF 중에서 적절한 값으로 설정)

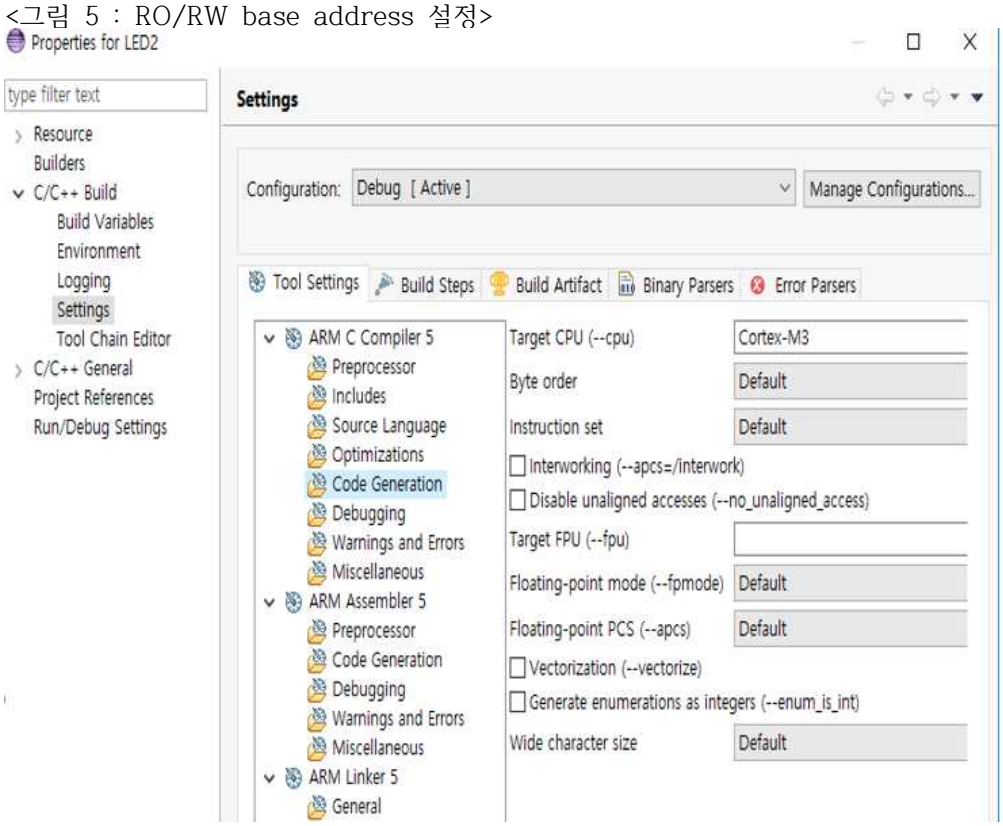
② C project 우클릭 후 Properties 선택 → C/C++ Build → Settings → Code Generation과 General의 Target CPU를 Cortex M3로 설정

3-3) C Project Build

① 위의 과정이 끝났으면, C Project를 Build한다. Build 시 .axf이 생성된다.



<그림 5 : RO/RW base address 설정>

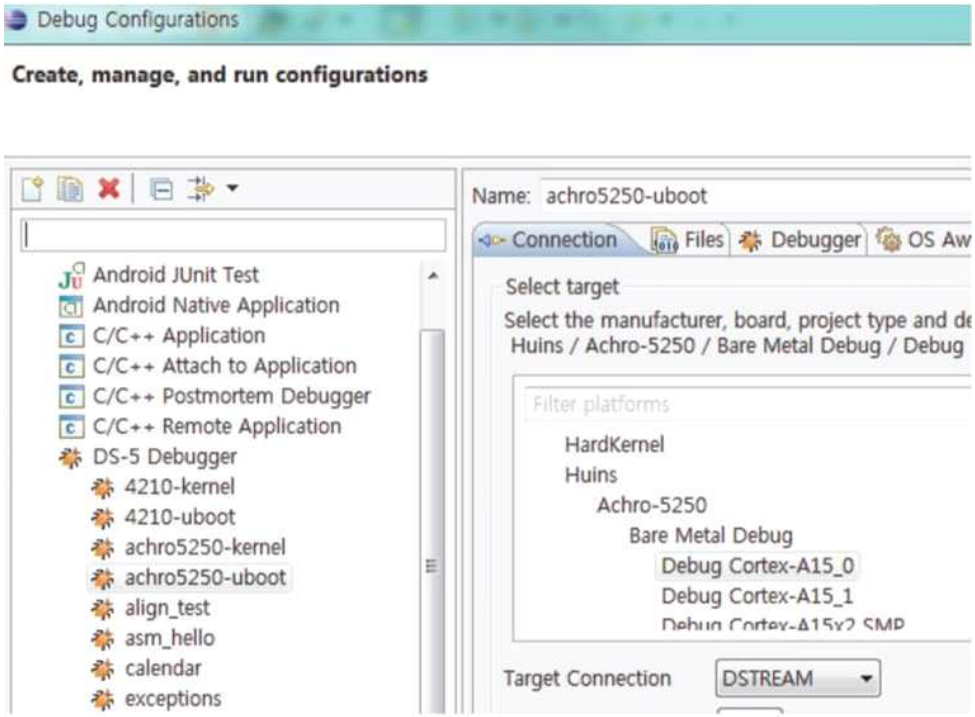


<그림 6 : Target CPU 설정>

4) DS-5 Debugger 연결

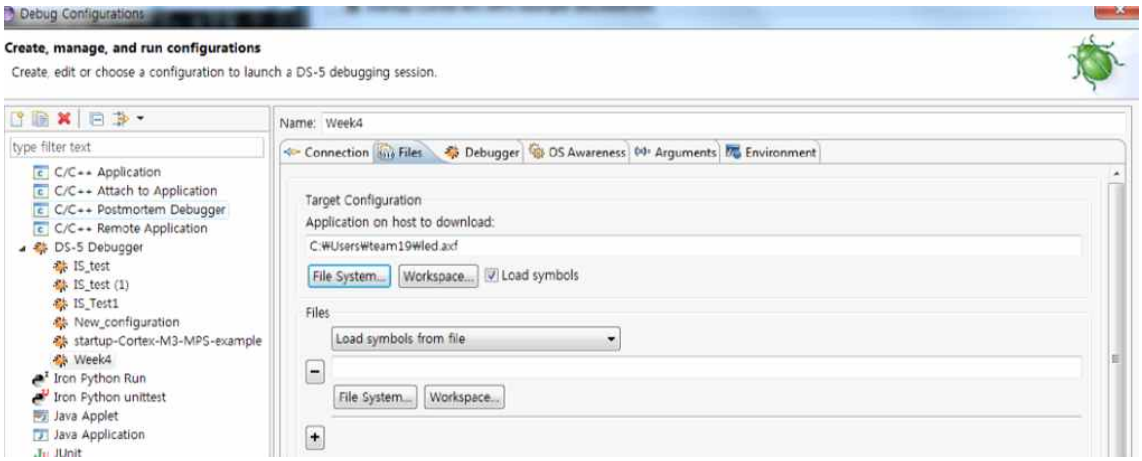
4-1) Debug Configuration 설정

① Run → Debug Configuration 메뉴를 선택
② DS-5 Debugger 더블 클릭하여 새로운 하위 오브젝트 생성
③ Name, Platform 등 Debug 환경설정을 변경
④ Browse 버튼을 클릭하여 DSTREAM 장비를 detection

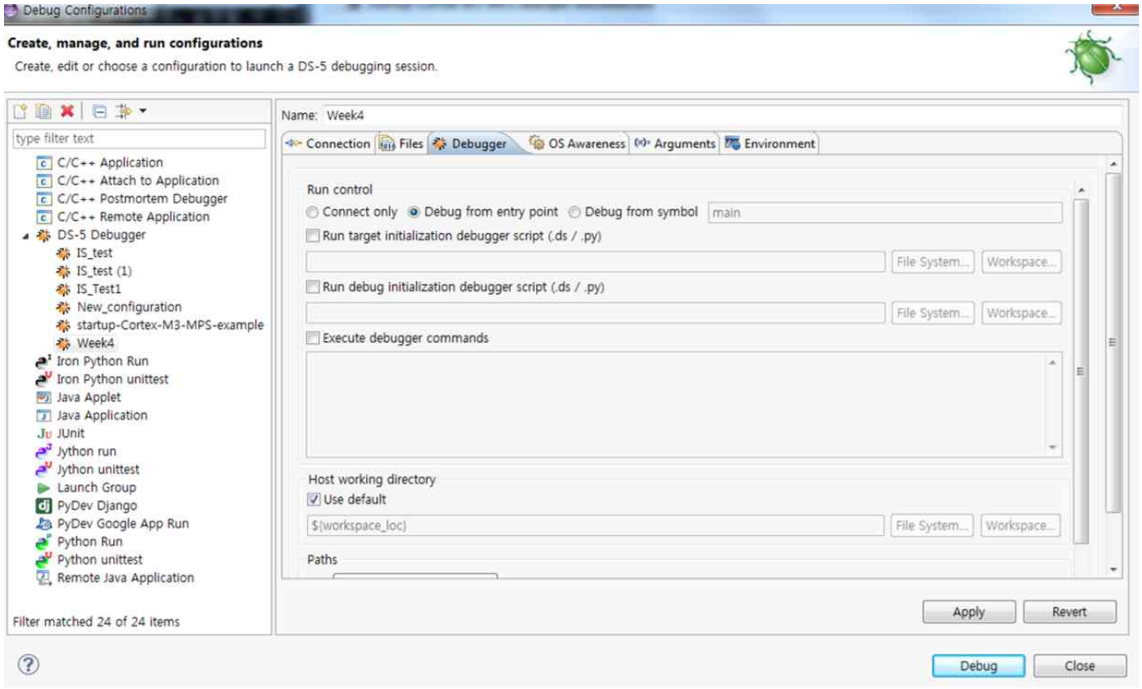


<그림 7 : Debug configuration 설정 및 connection>

4-2) .axf 파일 업로드 및 Debug
① File 탭에서 Workspace를 통해 .axf파일 경로를 설정
② Debugger 탭에서 Debug from entry point 체크
③ Apply → Debug.



<그림 8 : .axf 파일 경로설정>



<그림 9 : Debug 설정 및 Debug>

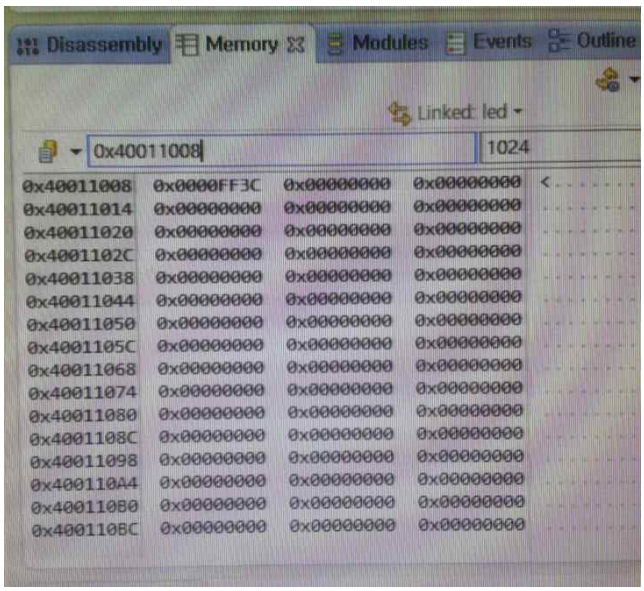
5) C source code 작성

5-1) 해당 Port를 Enable 및 초기화

① LED를 켜기 위한 Port D, 조이스틱을 동작시키기 위한 Port B,C를 enable 시키고 적절한 값으로 초기화를 해줌.(stm32_ReferenceManual과 stm32_DataSheet를 참고) 필요에 따라 Eclipse에서 제공하는 debugging 툴을 이용하여 주소값을 찾음

5-2) 주어진 조건에 맞는 적절한 함수를 작성

① 조이스틱 조작에 맞게 특정 패턴으로 LED가 켜지고 꺼지도록 적절한 함수를 작성



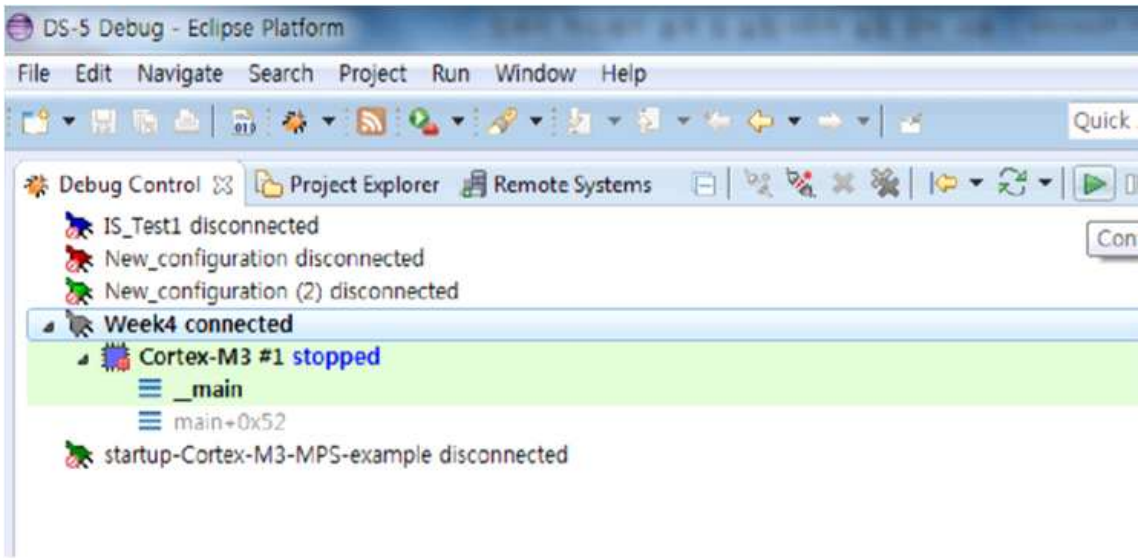
<그림 10 : debugging 툴을 이용>

6) LED 동작 유무 확인

6-1) Debug As 및 LED 확인

① C Project 우클릭 → Debug → Debug As

② stop 상태를 countinue로 바꾸어 LED 동작을 확인한다.



<그림 11 : stopped된 상태. continue로 바꾸면 LED동작을 확인할 수 있다>

7) 보드 연결 해체

앞서 보드 연결과 마찬가지로, 보드 연결 해체 시에도 순서를 제대로 지키지 않으면 보드가 망가질 수 있으므로 유의해야한다. 보드 연결 해체 순서는 다음과 같다.

표 2 : 보드 연결 해체 순서

① DS-5에서 'disconnect target'
② 보드 전원 OFF
③ DSTREAM 전원 해제 및 OFF
④ 보드 전원선 분리
⑤ DSTREAM과 보드 JTAG 분리

4. 작성한 소스코드 및 실험결과

1) 작성한 소스코드

전체 소스코드

```
#include <stdio.h>

const int ALE1= 0x04;
const int ALE2= 0x08;
const int ALE3= 0x10;
const int ALE4= 0x80;
```

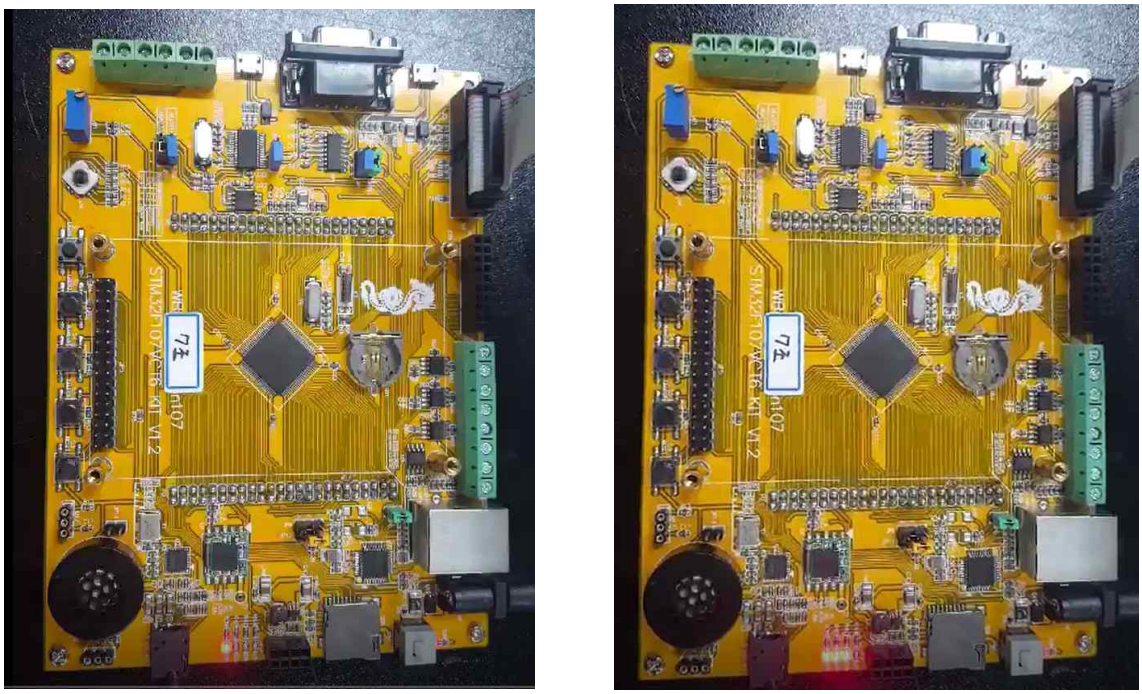
```
const int UP = 0x20;
const int DOWN = 0x04;
const int LEFT = 0x08;
const int RIGHT = 0x10;
const int SELECT = 0x100;

int PREV = 0;

void delay(num)
{
    int i;
    for(i=0;i<num*10;++i)
        ;
}

void onAll()
{
    (*(volatile unsigned *)0x40011410) = ALE1 | ALE2 | ALE3 | ALE4;
}

void offAll()
{
    (*(volatile unsigned *)0x40011414) = ALE1 | ALE2 | ALE3 | ALE4;

}

void upWave()
{
    (*(volatile unsigned *)0x40011410) = ALE1;
    delay(100000);
    (*(volatile unsigned *)0x40011410) |= ALE2;
    delay(100000);
    (*(volatile unsigned *)0x40011410) |= ALE3;
    delay(100000);
    (*(volatile unsigned *)0x40011410) |= ALE4;
    delay(100000);
    (*(volatile unsigned *)0x40011414) = ALE4;
    delay(100000);
    (*(volatile unsigned *)0x40011414) |= ALE3;
    delay(100000);
    (*(volatile unsigned *)0x40011414) |= ALE2;
```

```
    delay(100000);
    (*(volatile unsigned *)0x40011414) |= ALE1;
    delay(100000);
}

void downWave()
{
    (*(volatile unsigned *)0x40011410) = ALE4;
    delay(100000);
    (*(volatile unsigned *)0x40011410) |= ALE3;
    delay(100000);
    (*(volatile unsigned *)0x40011410) |= ALE2;
    delay(100000);
    (*(volatile unsigned *)0x40011410) |= ALE1;
    delay(100000);
    (*(volatile unsigned *)0x40011414) = ALE1;
    delay(100000);
    (*(volatile unsigned *)0x40011414) |= ALE2;
    delay(100000);
    (*(volatile unsigned *)0x40011414) |= ALE3;
    delay(100000);
    (*(volatile unsigned *)0x40011414) |= ALE4;
    delay(100000);
}

void blink(int num)
{
    (*(volatile unsigned *)0x40011410) = num;
    delay(100000);
    (*(volatile unsigned *)0x40011414) |= num;
    delay(100000);
}

int main(void) {
  (*(volatile unsigned *)0x40021018) |= 0x38;
  (*(volatile unsigned *)0x40011400) = 0x44444444;
  (*(volatile unsigned *)0x40011400) = 0x10011100;

  (*(volatile unsigned *)0x40011000) = 0x00888800;
  (*(volatile unsigned *)0x40010C04) = 0x8;
  (*(volatile unsigned *)0x40011008) = 0x00000000;
```

```
  while(1) {
      if(PREV == UP || (~(*(volatile unsigned *)0x40011008) & UP)) {
          upWave();
          PREV = UP;
      }
      if(PREV == DOWN || (~(*(volatile unsigned *)0x40011008) & DOWN)) {
          downWave();
          PREV = DOWN;
      }
      if(PREV == LEFT || (~(*(volatile unsigned *)0x40011008) & LEFT)) {
          blink(ALE4);
          PREV = LEFT;
      }
      if(PREV == RIGHT || (~(*(volatile unsigned *)0x40011008) & RIGHT)) {
          blink(ALE1);
          PREV = RIGHT;
      }
      if(PREV == SELECT || (~(*(volatile unsigned *)0x40010C08) & SELECT)) {
          offAll();
          PREV = SELECT;
      }
  }
  return 0;
}
```

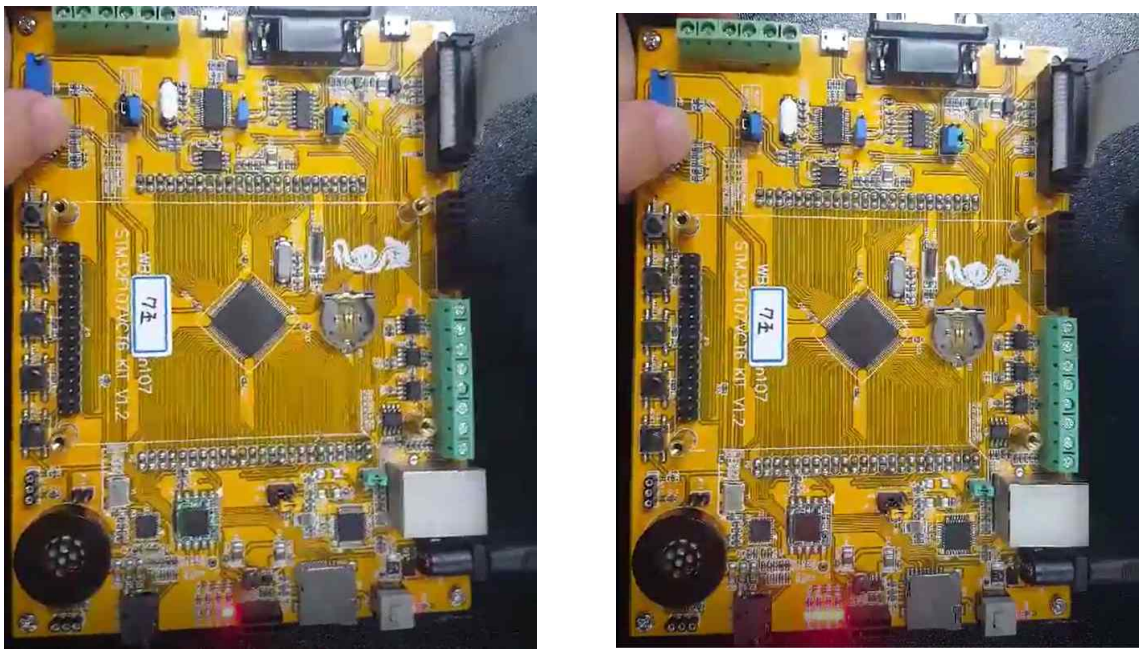
2) 실험결과

2-1) Up : 위로 물결



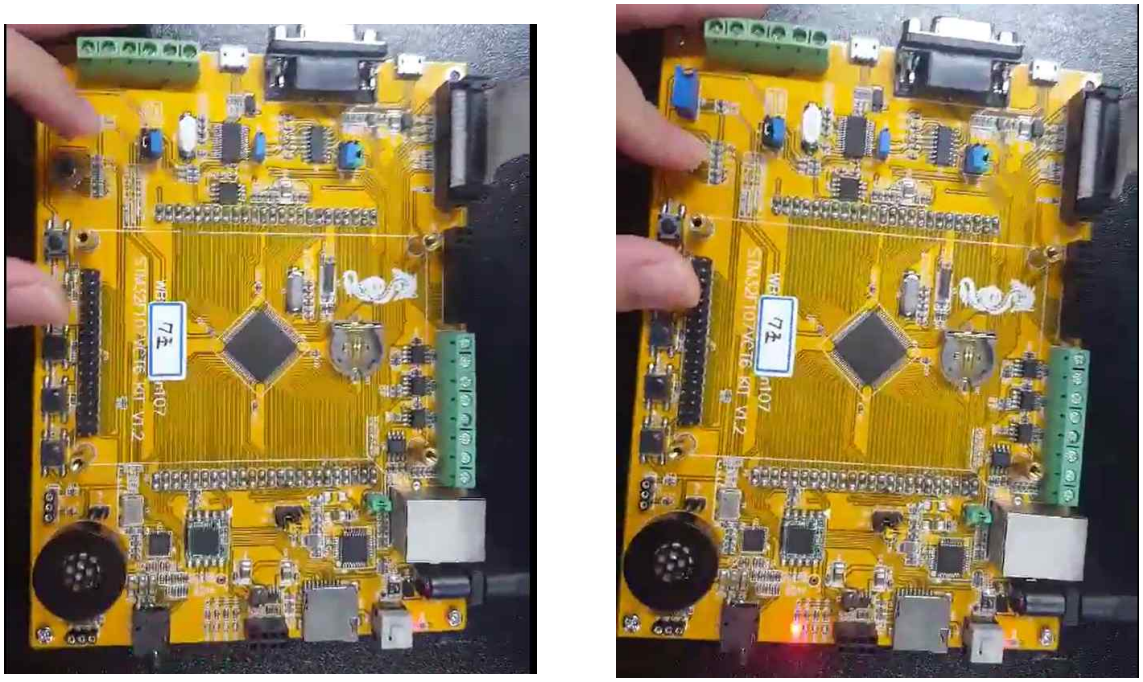
<그림 12 : 조이스틱 Up & 위로 물결>

2-2) Down : 아래로 물결



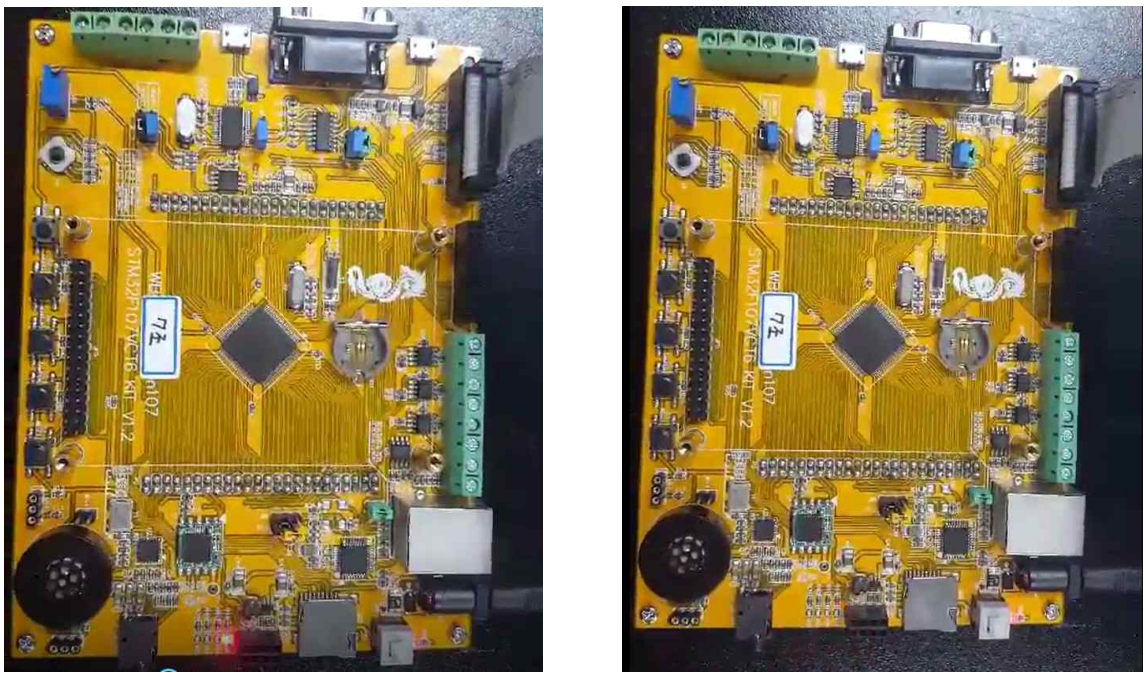
<그림 13 : 조이스틱 Donw & 아래로 물결>

2-3) Left : 4번 LED 점멸



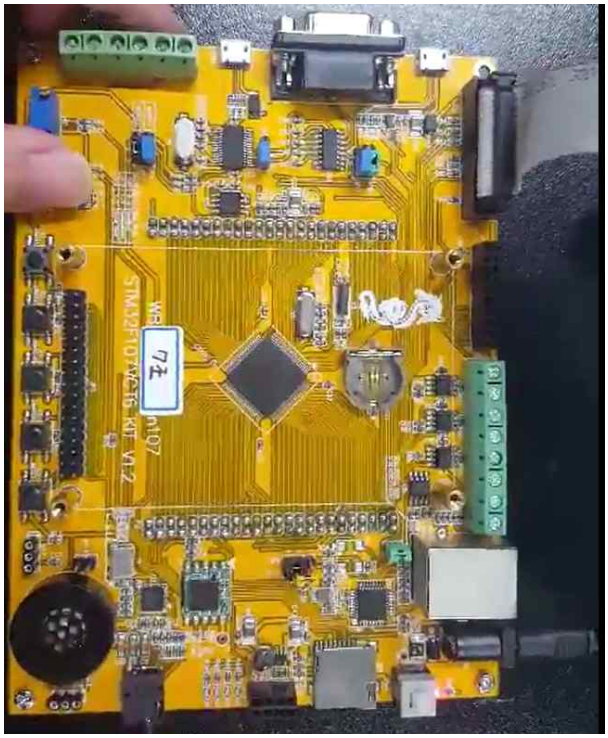
<그림 14 : 조이스틱 Left & 4번 LED 점멸>

2-4) Right : 1번 LED 점멸



<그림 15 : 조이스틱 Right & 1번 LED 점멸>

2-5) Select : 모든 LED OFF



<그림 16 : 조이스틱 Select & off>

5. 결론 및 느낀점

이번 실험을 통해 DSTREAM과 DS-5의 연결 방법 및 사용법을 익히고, 조이스틱과 LED를 입출력으로 사용하여 코드를 구현하고, 보드를 통해 결과를 확인함으로써 전체적인 동작과정을 이해하였다.

Memory Map Debugging을 통해 특정 주소의 register에 어떤 값이 저장되어있는지 확인하고 저장값이 변하는 것을 확인하였다.

이번 실험에서 가장 시간이 많이 걸리고 이해하기 어려웠던 부분은 조이스틱과 LED를 사용함에 있어서 GPIO를 사용해 제어하는 부분이었는데, LED의 경우 D port의 2, 3, 4, 7을 사용하기 때문에 Port configuration register low(GPIOx_CRL)을 사용하여 제어하였고, 조이스틱의 Select의 경우 B port의 8번을 사용하기 때문에 Port configuration register high(GPIOx_CRH)를 사용하여 제어하였다.

특히 LED를 제어할 때 Port bit set/reset register(GPIOx_BSRR)에 넣은 값을 Port bit reset register(GPIOx_BRR)에 넣으면 LED가 꺼지는 것을 이용하여 제어하였는데, 이 부분은 꼭 기억해둬야 할 것 같다.

실험에 있어 힘들었던 부분은 DS-5 디버거 설정에서 Connection 부분에서 인식이 잘 안돼서 시간이 오래 걸렸었는데, 전원을 껐다 켜야 해결되는 문제였다. 이유를 알 수는 없지만 이런 부분이 실험시간이 길어지는 이유 중 하나인 것 같다.

보드를 통해 동작시키는 것 보다 각각의 코드가 무엇을 의미하고 왜 이런 동작을 하는지 이해하는 부분이 더 중요하고 많은 시간이 걸린다는 점이 느꼈고, 앞으로 실험할 때는 이 부분에 더 신경을 써서 실험하게 될 것 같다.