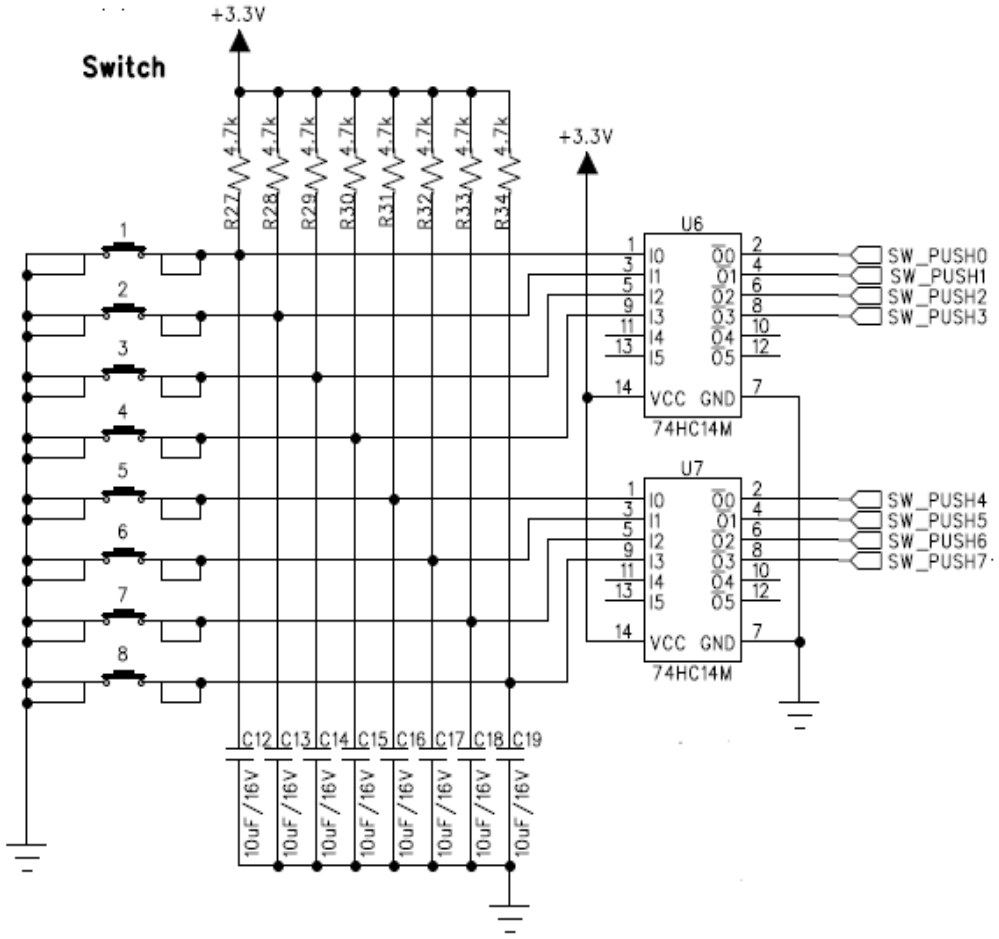
㈜한백전자 교육사업부

Chapter 4
스위치 모듈

4-가 푸쉬 버튼 스위치

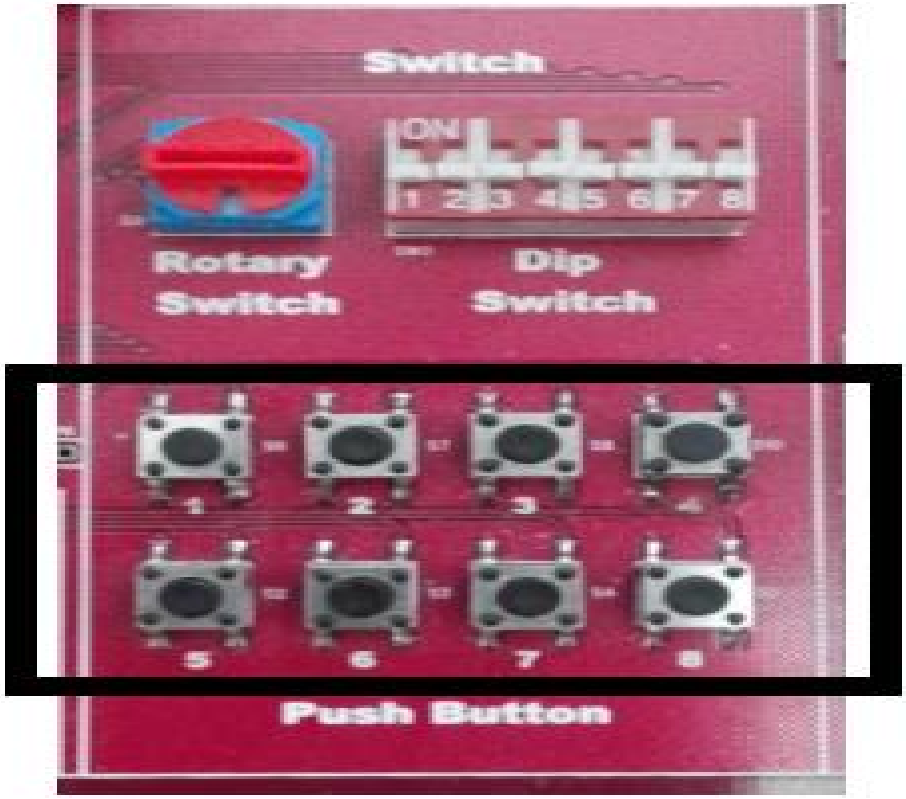
4-가-1 푸쉬 버튼 스위치 회로도와 모듈

- 푸쉬 버튼 스위치 회로도



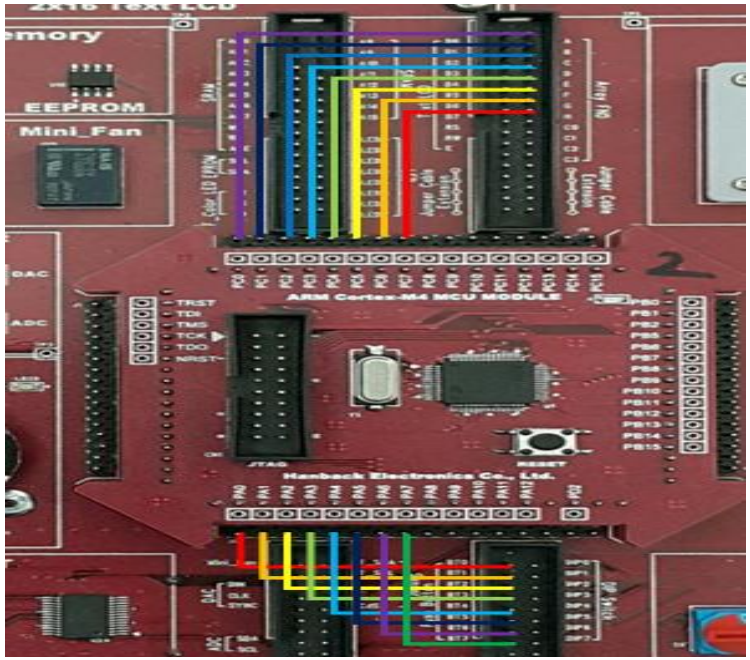
4-가-1 푸쉬 버튼 스위치 회로도와 모듈

- 푸쉬 버튼 모듈



4-가-실험1 : 스위치에 해당되는 LED 켜기

- 실험 내용 : 스위치를 누르면 해당 LED가 ON되는 프로그램을 작성
- 실험 방법
  - 포트 C의 PC0 ~ PC7를 LED 0 ~ 7까지로 연결해 준다.
  - 포트 A의 PA0 ~ PA7을 8핀 케이블로 푸쉬 버튼 신호인 BT0 ~ BT7까지로 연결
- FND 결선



4-가-실험1 : 스위치에 해당되는 LED 켜기

- 소스 파일

```
// stm32f4xx의 각 레지스터들을 정의한 헤더파일
#include "stm32f4xx.h"
int main()
{
GPIO_InitTypeDef GPIO_InitStructure;
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA|RCC_AHB1Periph_GPIOC, ENABLE);
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|GPIO_Pin_3|
GPIO_Pin_4|GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7;
// C 포트 하위 8비트를 출력으로 선언한다.
GPIO_Init(GPIOC, &GPIO_InitStructure);
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN; // A 포트 하위 8비트를 입력으로 선언한다.
GPIO_Init(GPIOA, & GPIO_InitStructure);
while(1)
{
// A 포트 하위 8비트에 입력된 값을 포트 C의 하위 8비트로 바로 출력한다.
GPIO_Write(GPIOC,GPIO_ReadInputData(GPIOA)&0x00FF);
}
}
```

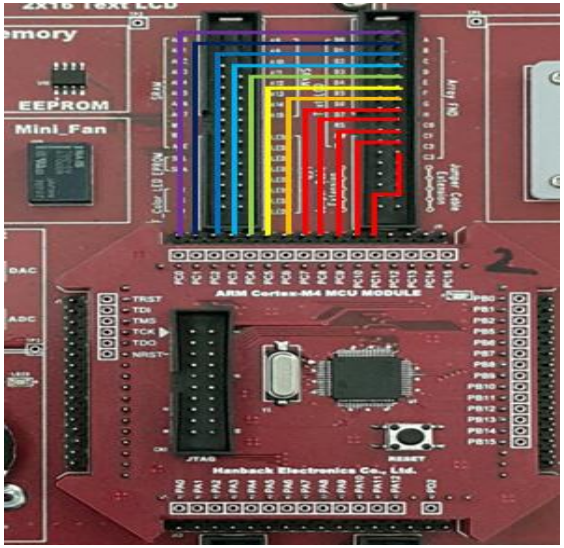
4-가-실험2 : 눌린 스위치 번호를 7-세그먼트에 표시하기

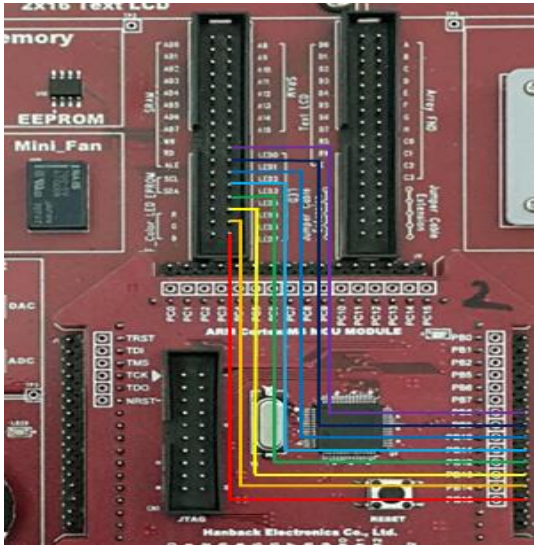
- 실험 내용 : 실험 4-1에서와 같이 스위치를 누르면 해당 LED가 ON 이 되면서 해당 스위치 번호가 7-세그먼트에 표시되는 프로그램을 작성
- 실험 방법
  - 포트 B의 PB8~PB15를 LED 0~7까지로 연결해 준다.
  - 포트 A의 PA0~PA7을 8핀 케이블로 푸쉬 버튼 신호인 BT0~BT7까지로 연결해 준다.
  - MCU 모듈 포트 PC0~PC7를 7-세그먼트 a ~ h 까지 연결해주고, PC8~PC11을 7-세그먼트 C0 ~ C3까지 연결
- 이번 실험을 하기 위해서는 [실험 4-1]에서 MCU 모듈과 LED를 연결한 것에서 실험 3-3과 같이 7-세그먼트 모듈도 추가적으로 연결 하여야 한다. 즉, 7-세그먼트를 MCU 모듈 포트 PC0~PC7, PC8~PC11까지 연결해 준다.

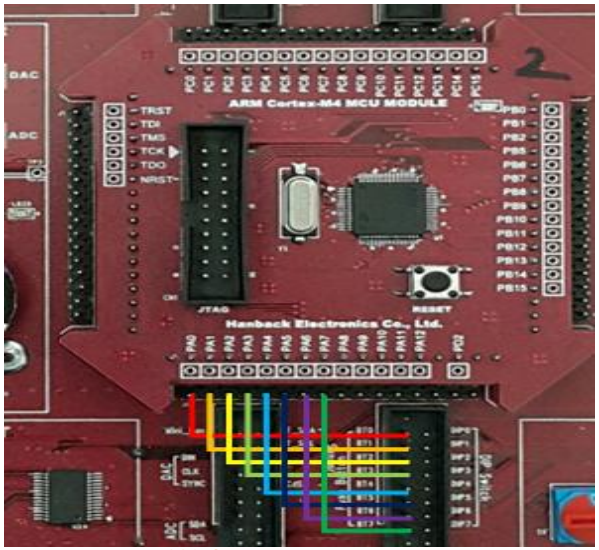
포트 연결:
1) ARM Cortex-M4 모듈의 포트C (PC0 ~ PC7)를 8핀 케이블을 이용해서 Array FND모듈의 SA_A ~ SA_H에 연결한다. (SA_A가 PC0로 연결돼야 한다.)
2) ARM Cortex-M4 모듈의 포트C (PC8 ~ PC11)를 4핀 케이블을 이용해서 Array FND모듈의 C0 ~ C3에 연결한다. (C0가 PC8로 연결돼야 한다.)
3) ARM Cortex-M4 모듈의 포트B (PB8 ~ PB15)를 8핀 케이블을 이용해서 LED모듈의 LED0 ~ LED7에 연결한다. (LED0가 PB8로 연결돼야 한다.)
4) ARM Cortex-M4 모듈의 포트A (PA0 ~ PA7)를 8핀 케이블을 이용해서 Switch모듈의 BT0 ~ BT7에 연결한다. (BT0가 PA0로 연결돼야 한다.)

4-가-실험2 : 눌린 스위치 번호를 7-세그먼트에 표시하기

- 결선 방법







4-가-실험2 : 눌린 스위치 번호를 7-세그먼트에 표시하기

- 소스 파일 – (1)

```
// stm32f4xx의 각 레지스터들을 정의한 헤더파일
#include "stm32f4xx.h"
// 7-세그먼트 폰트를 배열로 지정한다.
unsigned char Font[18] = {0x3F, 0X06, 0X5B, 0X4F,
0X66, 0X6D, 0X7C, 0X07,
0X7F, 0X67, 0X77, 0X7C,
0X39, 0X5E, 0X79, 0X71,
0X08, 0X80};
int main()
{
GPIO_InitTypeDef GPIO_InitStructure;
int ps_switch;
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA|RCC_AHB1Periph_GPIOB|RCC_AHB1Periph_
GPIOC, ENABLE);
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8|GPIO_Pin_9|GPIO_Pin_10|GPIO_Pin_11|
GPIO_Pin_12|GPIO_Pin_13|GPIO_Pin_14|GPIO_Pin_15;
GPIO_Init(GPIOB, &GPIO_InitStructure);
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|GPIO_Pin_3|
GPIO_Pin_4|GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7|
GPIO_Pin_8|GPIO_Pin_9|GPIO_Pin_10|GPIO_Pin_11;
GPIO_Init(GPIOC, &GPIO_InitStructure);
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|GPIO_Pin_3|
GPIO_Pin_4|GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7;
GPIO_Init(GPIOA, &GPIO_InitStructure);
// C포트의 8 ~ 11 핀을 모두 '0'으로 출력함으로써 7-세그먼트 4개를 다 켠다.
GPIO_ResetBits(GPIOC,GPIO_Pin_8|GPIO_Pin_9|GPIO_Pin_10|GPIO_Pin_11);
```

4-가-실험2 : 눌린 스위치 번호를 7-세그먼트에 표시하기

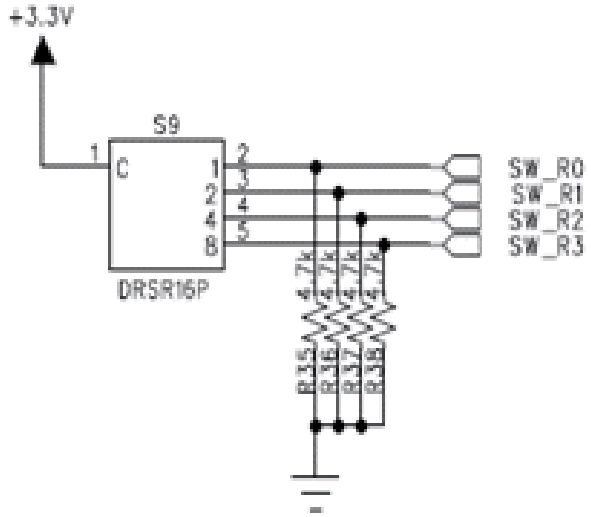
- 소스 파일 – (2)

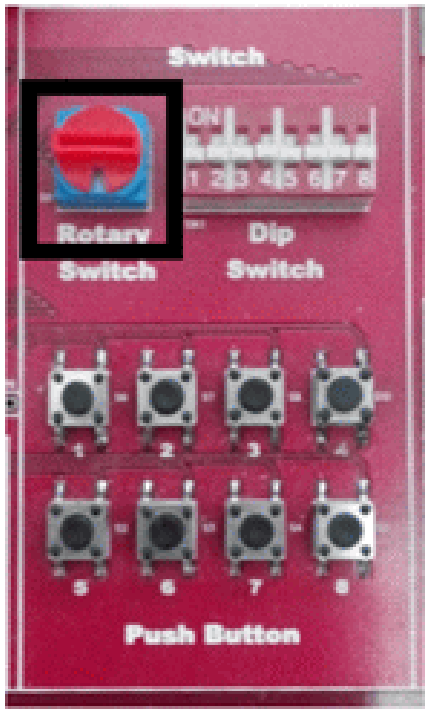
```
while(1)
{
//스위치 입력을 포트 A에서 읽어 온다.
ps_switch = GPIO_ReadInputData(GPIOA)&0x00FF ;
// 읽어온 스위치 입력을 LED로 출력해 준다.
GPIO_Write(GPIOB, ps_switch<<8);
// 스위치 입력이 없으면 세그먼트에 '0000'을 표시해 주고
// 스위치 '1'이 눌렸으면 세그먼트에 '1111'을 표시해주고
// 스위치 '2'가 눌렸으면 세그먼트에 '2222' 이런 식으로 표시해 준다.
// 스위치가 안 눌림
if(ps_switch == 0x00) GPIO_Write(GPIOC, Font[0]&0x00FF);
// 스위치 1이 눌림
else if(ps_switch == 0x01) GPIO_Write(GPIOC, Font[1]&0x00FF);
// 스위치 2가 눌림
else if( ps_switch == 0x02 ) GPIO_Write(GPIOC, Font[2]&0x00FF);
// 스위치 3이 눌림
else if( ps_switch == 0x04 ) GPIO_Write(GPIOC, Font[3]&0x00FF);
// 스위치 4가 눌림
else if( ps_switch == 0x08 ) GPIO_Write(GPIOC, Font[4]&0x00FF);
//스위치 5가 눌림
else if( ps_switch == 0x10 ) GPIO_Write(GPIOC, Font[5]&0x00FF);
// 스위치 6이 눌림
else if( ps_switch == 0x20 ) GPIO_Write(GPIOC, Font[6]&0x00FF);
//스위치 7이 눌림
else if( ps_switch == 0x40 ) GPIO_Write(GPIOC, Font[7]&0x00FF);
//스위치 8이 눌림
else if( ps_switch == 0x80 ) GPIO_Write(GPIOC, Font[8]&0x00FF);
}
}
```

4-나 로터리 스위치

4-나-1 로터리 스위치 회로도와 모듈

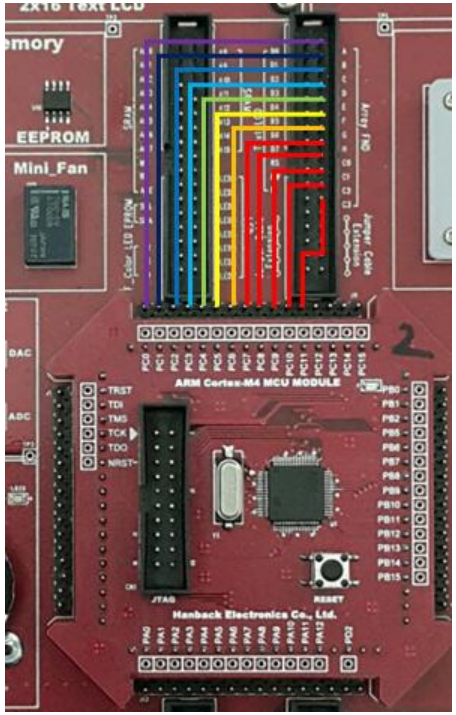
- HBE-MCU-Multi II에서 사용한 로터리 스위치는 헥사 코드 로타리 스위치로 0부터 F 까지 선택을 할 수 있고 그에 해당되는 출력이 SWR0 ~SWR3까지 4비트로 출력되게 되어 있다.

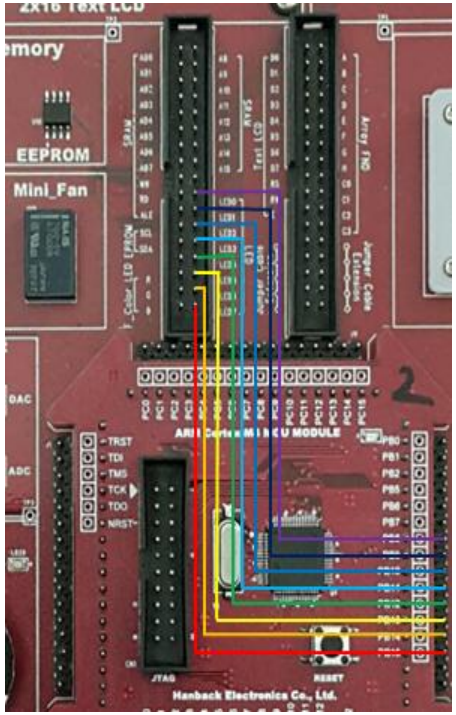


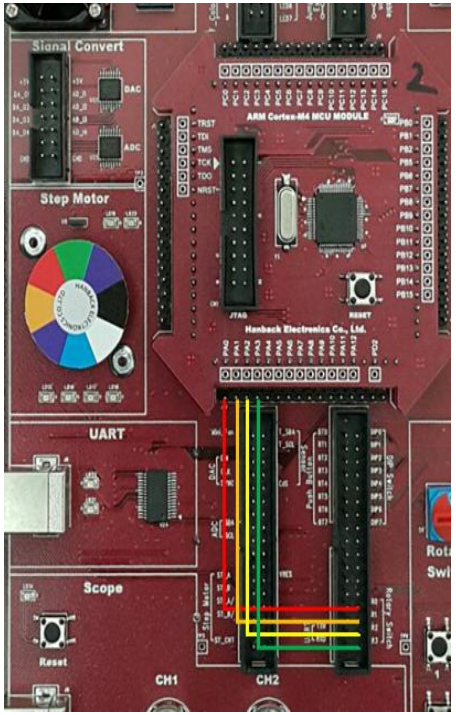


4-나-실험1 : 로터리 스위치 입력에 따라 7-세그먼트 표시

- 실험 내용 : 로터리 스위치에 해당 값을 선택하면 그 값이 7-세그먼트에 출력되는 프로그램을 작성
- 실험 방법
  - 포트 PC0 ~ PC7, PC8 ~ PC11에 7-세그먼트의 Array FND 부분을 연결
  - 포트 PB8 ~ PB15에 LED 0 ~ LED 7을 연결
  - 포트 PA0 ~ PA3에 Rotary Switch를 연결







4-나-실험1 : 로터리 스위치 입력에 따라 7-세그먼트 표시

- 소스 파일 – (1)

포트 연결:
1) ARM Cortex-M4 모듈의 포트C (PC0 ~ PC7)를 8핀 케이블을 이용해서 Array FND모듈의 SA_A ~ SA_H에 연결한다. (SA_A가 PC0로 연결돼야 한다.)
2) ARM Cortex-M4 모듈의 포트C (PC8 ~ PC11)를 4핀 케이블을 이용해서 Array FND모듈의 C0 ~ C3에 연결한다. (C0가 PC8로 연결돼야 한다.)
3) ARM Cortex-M4 모듈의 포트B (PB8 ~ PB15)를 8핀 케이블을 이용해서 LED모듈의 LED0 ~ LED7에 연결한다. (LED0가 PB8로 연결돼야 한다.)
4) ARM Cortex-M4 모듈의 포트A (PA0 ~ PA3)를 4핀 케이블을 이용해서 Rotary Switch모듈의 R0 ~ R3에 연결한다. (R0가 PA0로 연결돼야 한다.)

```
// stm32f4xx의 각 레지스터들을 정의한 헤더파일
#include "stm32f4xx.h"
// 7-세그먼트 폰트를 배열로 지정한다.
unsigned char Font[18] = {0x3F, 0X06, 0X5B, 0X4F,
0X66, 0X6D, 0X7C, 0X07,
0X7F, 0X67, 0X77, 0X7C,
0X39, 0X5E, 0X79, 0X71,
0X08, 0X80};
int main()
{
GPIO_InitTypeDef GPIO_InitStructure;
int R_sw;
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA|RCC_AHB1Periph_GPIOB|RCC_AHB1P
eriph_GPIOC, ENABLE);
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8|GPIO_Pin_9|GPIO_Pin_10|GPIO_Pin_11|
GPIO_Pin_12|GPIO_Pin_13|GPIO_Pin_14|GPIO_Pin_15;
GPIO_Init(GPIOB, &GPIO_InitStructure);
```

4-나-실험1 : 로터리 스위치 입력에 따라 7-세그먼트 표시

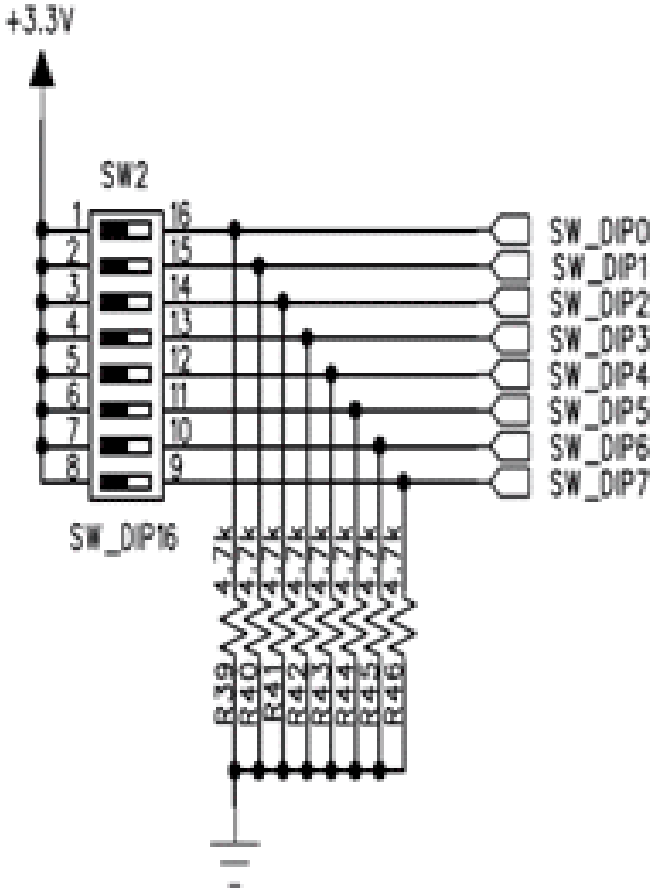
- 소스 파일 – (2)

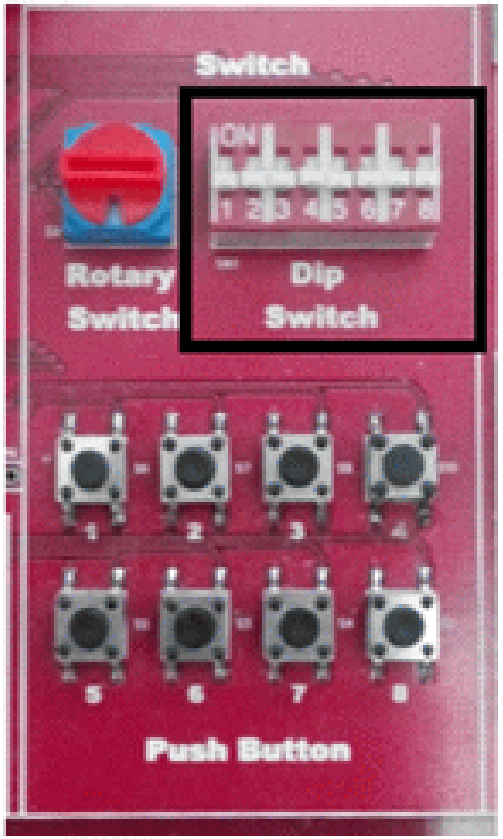
```
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|GPIO_Pin_3|
GPIO_Pin_4|GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7|
GPIO_Pin_8|GPIO_Pin_9|GPIO_Pin_10|GPIO_Pin_11;
GPIO_Init(GPIOC, &GPIO_InitStructure);
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|GPIO_Pin_3;
GPIO_Init(GPIOA, &GPIO_InitStructure);
// C포트의 8 ~ 11 핀을 모두 '0'으로 출력함으로써 7-세그먼트 4개를 다 켠다.
GPIO_ResetBits(GPIOC,GPIO_Pin_8|GPIO_Pin_9|GPIO_Pin_10|GPIO_Pin_11);
while(1)
{
//로타리 스위치 입력을 읽어 온다.
R_sw = GPIO_ReadInputData(GPIOA) ;
// A 포트의 15~4 비트를 확실히 '0'으로 해준다.
R_sw = R_sw & 0x000f ;
// 읽어온 스위치 입력을 LED로 출력해 준다.
GPIO_Write(GPIOB, R_sw<<8);
// 읽어온 로타리 스위치에 해당되는 값을 7-세그먼트에 출력한다.
GPIO_Write(GPIOC, Font[R_sw]&0x00FF);
}
}
```

4-다 DIP 스위치

4-다-1 Dip 스위치의 회로도와 모듈

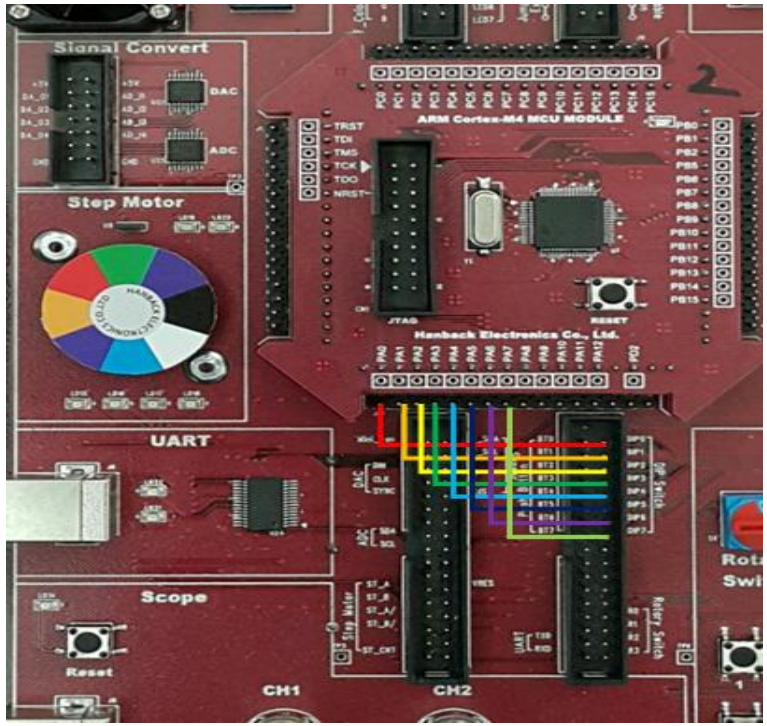
- Dip 스위치 회로도와 모듈





4-다-실험1 : Dip 스위치로 LED 켜기

- 실험 내용 : Dip 스위치 각 비트를 ON 시키면 그에 해당되는 LED가 켜지도록 프로그램을 작성
- 실험 방법
  - 포트 PC0 ~ PC7에 LED 0 ~ LED 7을 연결
  - 포트 PA0 ~ PA7에 DIP Switch를 연결



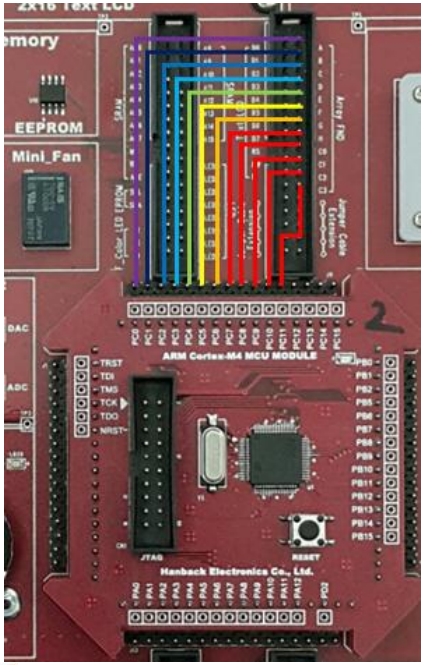
4-다-실험1 : Dip 스위치로 LED 켜기

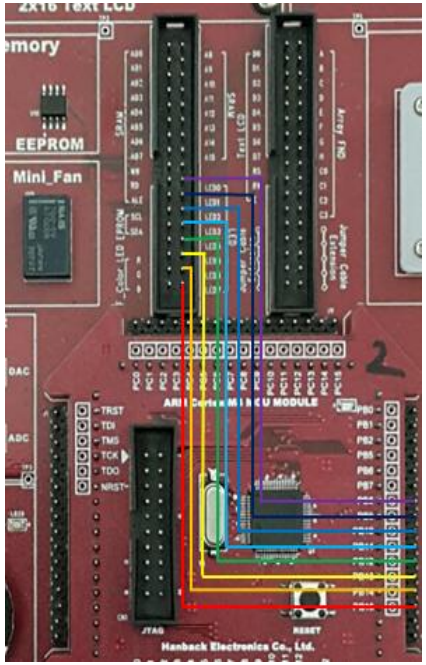
- 소스 파일 – (1)

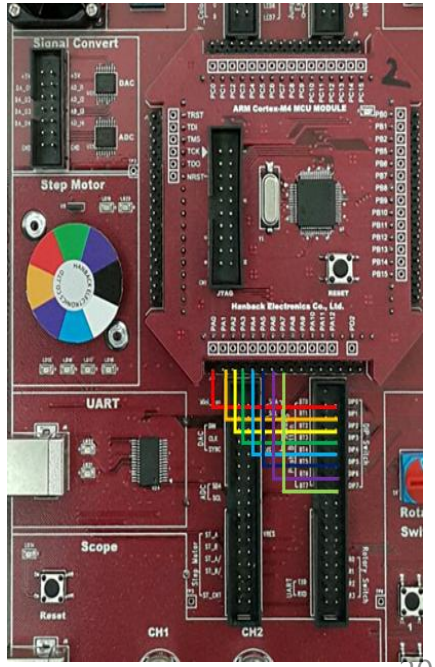
```
// stm32f4xx의 각 레지스터들을 정의한 헤더파일
#include "stm32f4xx.h"
int main()
{
GPIO_InitTypeDef GPIO_InitStructure;
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA|RCC_AHB1Periph_GPIOC, ENABLE);
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|GPIO_Pin_3|
GPIO_Pin_4|GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7;
// C 포트 하위 8비트를 출력으로 선언한다.
GPIO_Init(GPIOC, &GPIO_InitStructure);
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN; // A 포트 하위 8비트를 입력으로 선언한다.
GPIO_Init(GPIOA, &GPIO_InitStructure);
while(1)
{
// Dip 스위치 입력을 포트 E의 LED로 직접 출력한다.
GPIO_Write(GPIOC,GPIO_ReadInputData(GPIOA)&0x00FF);
}
}
```

4-다-실험2 : Dip 스위치로 7-세그먼트 켜기

- 실험 내용 : 실험 4-4와 같이 Dip 스위치 각 비트를 ON 시키면 해당되는 LED가 켜지고, 그에 해당되는 헥사(HEX) 값이 7-Segment의 왼쪽에서 세 번째 네 번째에 표시되도록 프로그램을 작성
- 실험 방법
  - 포트 PB8~PB15에 LED 0~LED 7을 연결
  - 포트 PA0~PA7에 DIP Switch를 연결
  - 포트 PC0~PC7, PC8~PC11에 7-세그먼트의 Array FND 부분을 연결







4-다-실험2 : Dip 스위치로 7-세그먼트 켜기

- 소스 파일 – (1)

포트 연결:
1) ARM Cortex-M4 모듈의 포트C (PC0 ~ PC7)를 8핀 케이블을 이용해서
Array FND모듈의 SA_A ~ SA_H에 연결한다. (SA_A가 PC0로 연결돼야 한다.)
2) ARM Cortex-M4 모듈의 포트C (PC8 ~ PC11)를 4핀 케이블을 이용해서
Array FND모듈의 C0 ~ C3에 연결한다. (C0가 PC8로 연결돼야 한다.)
3) ARM Cortex-M4 모듈의 포트B (PB8 ~ PB15)를 8핀 케이블을 이용해서
LED모듈의 LED0 ~ LED7에 연결한다. (LED0가 PB8로 연결돼야 한다.)
4) ARM Cortex-M4 모듈의 포트A (PA0 ~ PA7)를 8핀 케이블을 이용해서
Dip Switch모듈의 DIP0 ~ DIP73에 연결한다. (DIP0가 PA0로 연결돼야 한다.)

```
// stm32f4xx의 각 레지스터들을 정의한 헤더파일
#include "stm32f4xx.h"
// delay 함수
static void Delay(const uint32_t Count)
{
__IO uint32_t index = 0;
for(index = (16800 * Count); index != 0; index--);
}
// 7-세그먼트 폰트를 배열로 지정한다.
unsigned char Font[18] = {0x3F, 0X06, 0X5B, 0X4F,
0X66, 0X6D, 0X7C, 0X07,
0X7F, 0X67, 0X77, 0X7C,
0X39, 0X5E, 0X79, 0X71,
0X08, 0X80};
// Hex Segment 함수 선언 ==> 이 함수는 입력되는 값을 16진 HEX 값으로
// 7-세그먼트에 표시해 주는 함수이다
void HexSegment ( unsigned char N )
{
int i ;
unsigned char N10, N1 ;
```

4-다-실험2 : Dip 스위치로 7-세그먼트 켜기

- 소스 파일 – (2)

```
// 세그먼트에서 사용하는 첫 번째의 자리를 추출
N10 = N /16;
N1 = N % 16 ;
for( i = 0 ; i < 30; i++ )
{
// 왼쪽 세 번째 세그먼트를 ON하고, 첫 번째 숫자를 출력해 준다.
GPIO_Write(GPIOC, Font[N10]|0x0b00);
Delay(1);
// 왼쪽 네 번째 세그먼트를 ON하고, 두 번째 숫자를 출력해 준다.
GPIO_Write(GPIOC, Font[N1]|0x0700);
Delay(1);
}
}
int main()
{
GPIO_InitTypeDef GPIO_InitStructure;
int Dip_sw;
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA|RCC_AHB1Periph_GPIOB|RCC_AHB1Periph_GPIOC,
ENABLE);
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8|GPIO_Pin_9|GPIO_Pin_10|GPIO_Pin_11|
GPIO_Pin_12|GPIO_Pin_13|GPIO_Pin_14|GPIO_Pin_15;
GPIO_Init(GPIOB, &GPIO_InitStructure);
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|GPIO_Pin_3|
GPIO_Pin_4|GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7|
GPIO_Pin_8|GPIO_Pin_9|GPIO_Pin_10|GPIO_Pin_11;
```

4-다-실험2 : Dip 스위치로 7-세그먼트 켜기

- 소스 파일 – (3)

```
GPIO_Init(GPIOC, &GPIO_InitStructure);
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|GPIO_Pin_3;
GPIO_Init(GPIOA, &GPIO_InitStructure);
while(1)
{
//Dip 스위치 입력을 읽어 온다.
Dip_sw = GPIO_ReadInputData(GPIOA) ;
// A 포트의 15~8 비트를 확실히 '0'으로 해준다.
Dip_sw = Dip_sw & 0x00ff ;
// 읽어온 스위치 입력을 LED로 출력해 준다.
GPIO_Write(GPIOB, Dip_sw<<8);
// 읽어온 Dip 스위치에 해당되는 값을 7-세그먼트에 16진으로 표시한다.
HexSegment(Dip_sw);
}
}
```