마이크로컨트롤러 기능

제 3장 GPIO 입출력제어

GPIO 입출력제어

1. HBE-MUC-Multi II Elec 구동
2. 마이크로컨트롤러와 GPIO
3. AVR 마이크로컨트롤러의 입출력 포트
4. GPIO를 이용하여 LED 켜기
5. GPIO를 이용하여 푸쉬 버튼을 눌러 LED 켜기
6. GPIO를 이용하여 FND LED 켜기

HBE-MUC-Multi II Elec 구동

- HBE-MUC-Multi II Elec

HBE-MUC-Multi II Elec 구동

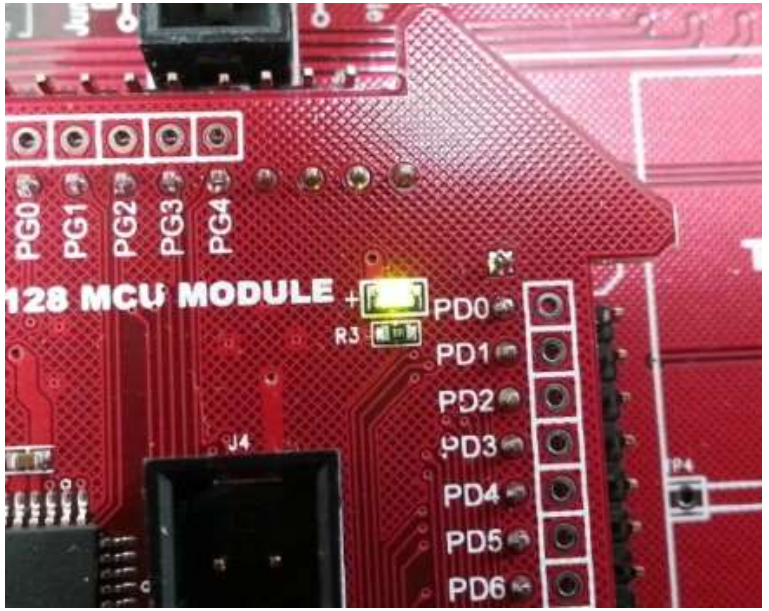
- HBE-MUC-Multi II Elec 장비의 모듈 구성
  - MCU Module
  - On-Board 디바이스
    - Text LCD, Array FND & LED, SRAM Flash, Mini Fan & Relay, Full Color LED, A/D & D/A Converter, Step Motor, 온/습도 센서, 조도 센서 & 가변저항, UART, 오실로스코프, 스위치 소자
  - Module 디바이스
    - Power, BLDC 모터, AC Load, 부스트 컨버터, Line Sensing, 벅 컨버터, 인버터
  - 케이블 수납함
    - 제품에 사용되는 각종 케이블 수납

HBE-MUC-Multi II Elec 구동

- HBE-MUC-Multi II Elec 실습전 모습

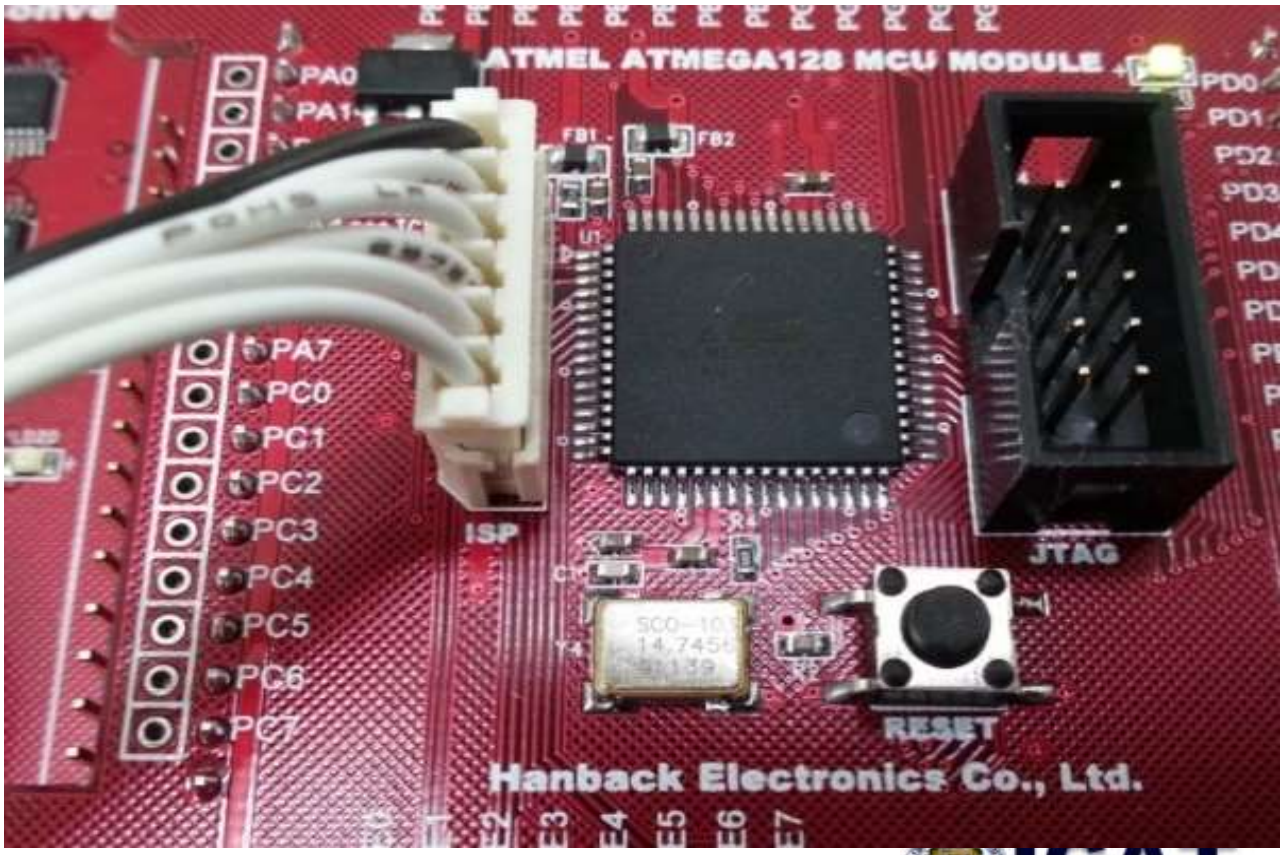
HBE-MUC-Multi II Elec 구동

❑HBE-MUC-Multi II Elec 전원케이블 연결



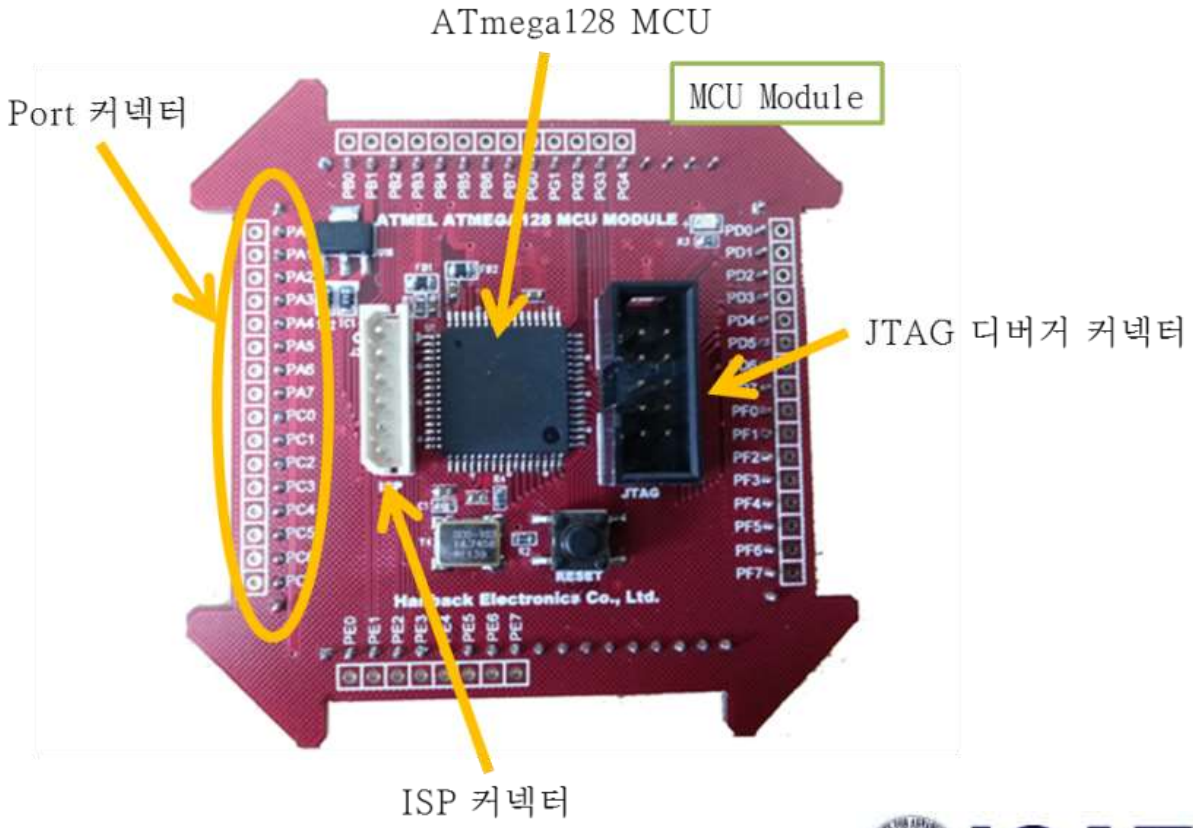
HBE-MUC-Multi II Elec 구동

- HBE-MUC-Multi II Elec과 AVR-ISP를 케이블로 연결



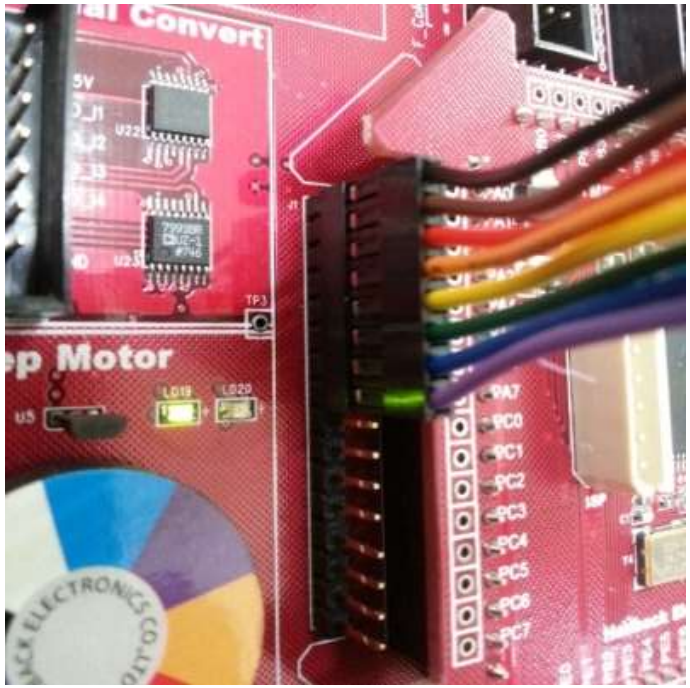
HBE-MUC-Multi II Elec 구동

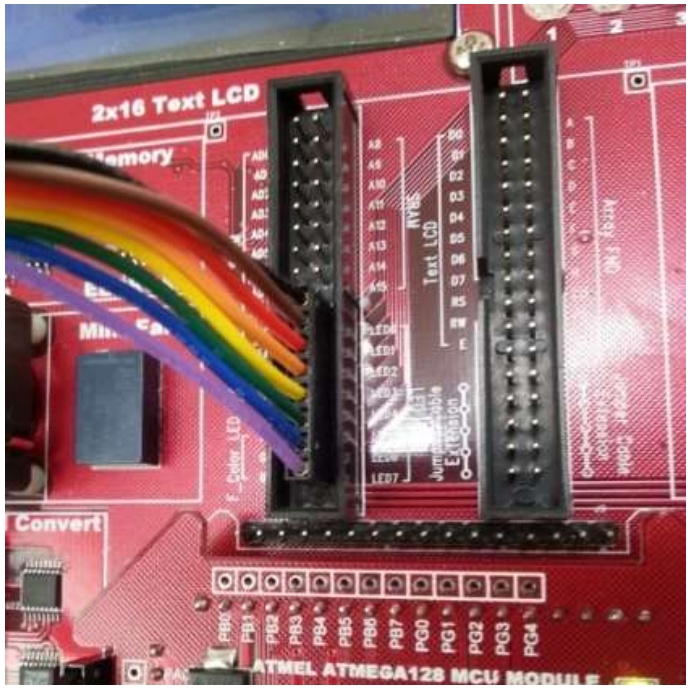
❑AVR MCU 모듈 구성



HBE-MUC-Multi II Elec 구동

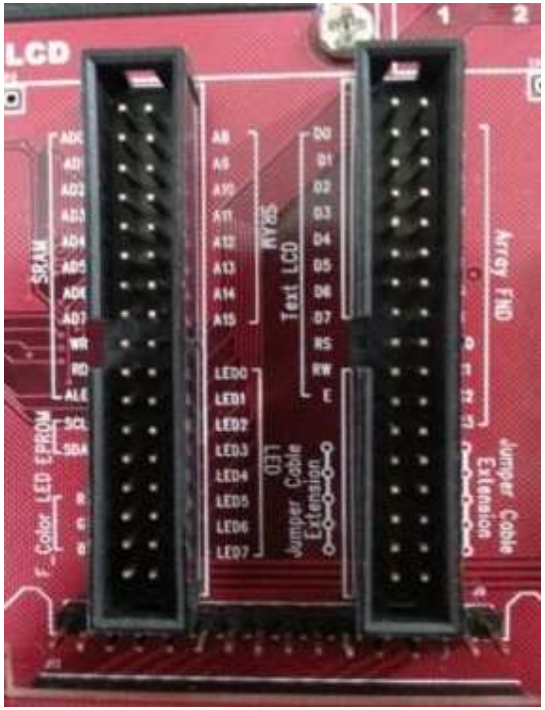
❑HBE-MUC-Multi II Elec 신호선 연결





HBE-MUC-Multi II Elec 구동

- HBE-MUC-Multi II Elec 기능 모듈(LED 모듈) 구성



마이크로컨트롤러와 GPIO

- GPIO(General Purpose Input Output)
  - 범용으로 사용되는 입출력 포트 : 설계자가 마음대로 변형하면서 제어할 수 있도록 제공해 주는 I/O(입출력) 포트
  - 입력과 출력을 마음대로 선택할 수 있고, 0과 1의 출력 신호를 임의로 만들어줄 수 있는 구조를 가짐
  - 입력으로 사용할 때는 외부 인터럽트를 처리할 수 있도록 하는 경우가 많음.
  - 입출력 방향 전환용 레지스터와 출력용/입력용 데이터 레지스터등이 필요
  - 마이크로컨트롤러에서는 대부분의 핀들을 GPIO로 설정하는 경우가 많음.

AVR 마이크로컨트롤러의 입출력 포트

- AVR 마이크로컨트롤러 입출력 포트
  - 6개의 8비트 I/O포트와 1개의 5비트 I/O포트로 구성.
  - 출력포트의 버퍼는 많은 유입전류와 유출전류를 사용(최대 40mA)할 수 있음.
  - 모든 포트 핀은 개별적으로 내부 풀업 저항을 사용할 수 있음.
  - 모든 I/O핀은 VCC와 GND사이에 다이오드를 접속하여 포트를 보호.
  - Read-Modify-Write기능을 가지고 있어, 비트 단위의 포트 설정이 가능.
  - 각 포트에 대한 데이터 출력용 레지스터(PORTx)와 데이터 입출력 방향 지정용 레지스터(Data Direction Register: DDRx), 그리고 데이터 입력용 레지스터(PINx)를 보유.

AVR 마이크로컨트롤러의 입출력 포트

- 입출력 포트 제어용 레지스터
  - DDRx 레지스터
    - 입출력의 방향설정을 하기 위한 레지스터.
    - DDRA~DDRG레지스터의 해당 비트에 '1'을 쓰면 출력, '0'을 쓰면 입력으로 설정.
  - PORTx 레지스터
    - 데이터를 출력하기 위한 레지스터이다
    - 출력을 원하는 데이터 값을 PORTx 레지스터에 넣어주면 된다
  - PINx 레지스터
    - 데이터 입력용 레지스터이다
    - PINx 레지스터에 해당하는 값을 읽으면 해당 핀의 값이 읽어진다.
  - SFIOR 레지스터
    - Special Function IO Register.
    - AVR 입출력 포트의 특수 기능을 제어하기 위한 레지스터
    - SFIOR의 비트2(PUD: Pull-Up Disable)를 '1'로 세트하면 풀업 저항을 비활성화시킨다

AVR 마이크로컨트롤러의 입출력 포트

❑ATMega128A의 범용 입출력 포트 : A 포트(PA7~PA0 : 핀44-51)

- ❖ 내부 풀업 저항이 있는 8비트 양방향 입출력 단자
- ❖ 외부메모리를 둘 경우에는 주소버스(A7-A0)와 데이터버스(D7-D0)로 사용

포트 핀	부가기능
PA7	AD7(외부 메모리 인터페이스 주소와 데이터 비트7)
PA6	AD6(외부 메모리 인터페이스 주소와 데이터 비트6)
PA5	AD5(외부 메모리 인터페이스 주소와 데이터 비트5)
PA4	AD4(외부 메모리 인터페이스 주소와 데이터 비트4)
PA3	AD3(외부 메모리 인터페이스 주소와 데이터 비트3)
PA2	AD2(외부 메모리 인터페이스 주소와 데이트 비트2)
PA1	AD1(외부 메모리 인터페이스 주소와 데이터 비트1)
PA0	AD0(외부 메모리 인터페이스 주소와 데이터 비트0)

AVR 마이크로컨트롤러의 입출력 포트

❑ATMega128A의 범용 입출력 포트 : B 포트(PB7~PB0 : 핀10-17)

- ❖ 내부 풀업 저항이 있는 8비트 양방향 입출력 단자
- ❖ 타이머/카운터나 SPI용 단자 혹은 PWM 단자로도 사용

포트 핀	부가기능
PB7	OC2/OC1C(출력비교 또는 타이머/카운터2의 PWM출력, 또는 출력비교와 타이머/카운터1의 PWM출력 C)
PB6	OC1B(출력비교 또는 타이머/카운터1의 PWM출력 B)
PB5	OC1A(출력비교 또는 타이머/카운터1의 PWM출력 A)
PB4	OC0(출력비교 또는 타이머/카운터0의 PWM출력)
PB3	MISO(SPI 버스 마스터 입력/종속 출력)
PB2	MOSI(SPI 버스 마스터 출력/종속 입력)
PB1	SCK(SPI 버스 직렬 클럭)
PB0	/SS(SPI 종속 선택 입력)

AVR 마이크로컨트롤러의 입출력 포트

❑ATMega128A의 범용 입출력 포트 : C 포트(PC7~PC0 : 핀35-42)

- ❖ 내부 풀업 저항이 있는 8비트 양방향 입출력 단자
- ❖ 외부메모리를 둘 경우에는 주소버스(A15-A8)로 사용

포트 핀	부가기능
PC7	AD7(외부 메모리 인터페이스 주소 비트15)
PC6	AD6(외부 메모리 인터페이스 주소 비트14)
PC5	AD5(외부 메모리 인터페이스 주소 비트13)
PC4	AD4(외부 메모리 인터페이스 주소 비트12)
PC3	AD3(외부 메모리 인터페이스 주소 비트11)
PC2	AD2(외부 메모리 인터페이스 주소 비트10)
PC1	AD1(외부 메모리 인터페이스 주소 비트 9)
PC0	AD0(외부 메모리 인터페이스 주소 비트8)

AVR 마이크로컨트롤러의 입출력 포트

❑ATMega128A의 범용 입출력 포트 : D 포트(PD7~PD0 : 핀25-32)

- ❖ 내부 풀업 저항이 있는 8비트 양방향 입출력 단자
- ❖ 타이머용 단자 혹은 외부인터럽트용 단자로도 사용.

포트 핀	부가기능
PD7	T2(타이머/카운터2 클럭 입력)
PD6	T1(타이머/카운터1 클럭 입력)
PD5	XCK1(USART1 외부 클럭 입/출력)
PD4	IC1(타이머/카운터1 입력 캡쳐 트리거)
PD3	INT3/TXD1(외부 인터럽트3 입력 또는 USART1 전송 핀)
PD2	INT2/RXD1(외부 인터럽트2 입력 또는 USART1 수신 핀)
PD1	INTI/SDA(외부 인터럽트1 입력 또는 TWI 직렬 데이터)
PD0	INT0/SCL(외부 인터럽트0 입력 또는 TWI 직렬 클럭)

AVR 마이크로컨트롤러의 입출력 포트

❑ATMega128A의 범용 입출력 포트 : E 포트(PE7~PE0 : 핀2-9)

- ❖ 내부 풀업 저항이 있는 8비트 양방향 입출력 단자
- ❖ 타이머용 단자, 외부인터럽트, 아날로그 비교기, USART용 단자로도 사용.

포트핀	부가기능
PE7	INT7/IC3(외부 인터럽트 7 입력 또는 타이머/카운터3 입력 캡쳐 트리거)
PE6	INT6/T3(외부 인터럽트 6 입력 또는 타이머/카운터3 클럭입력)
PE5	INT5/OC3C(외부 인터럽트 5 입력 또는 타이머/카운터3의 출력 캡쳐와 PWM 출력 C)
PE4	INT4/OC3B(외부 인터럽트 4 입력 또는 타이머/카운터3의 출력 캡쳐와 PWM 출력 B)
PE3	AIN1/OC3A(아날로그 비교 반대입력 또는 타이머/카운터3의 출력 비교와 PWM 출력A)
PE2	AIN0/XCK0(아날로그 비교 입력 또는 USART0 외부 클럭 입/출력)
PE1	PDO/TXD0(프로그램 데이터 출력 또는 UART0 전송 핀)
PE0	PDI/RXD0(프로그램 데이터 입력 또는 UART0 수신 핀)

AVR 마이크로컨트롤러의 입출력 포트

❑ATMega128A의 범용 입출력 포트 : F 포트(PF7~PF0 : 핀54-61)

- ❖ 내부 풀업 저항이 있는 8비트 양방향 입출력 단자
- ❖ AD변환기 혹은 JTAG 인터페이스용 단자로도 사용.

포트핀	부가기능
PF7	ADC7/TDI(ADC 입력 채널 7 또는 JTAG Test Data Input)
PF6	ADC6/TDO(ADC 입력 채널 6 또는 JTAG Test Data Output)
PF5	ADC5/TMS(ADC 입력 채널 5 또는 JTAG Test Mode Select)
PF4	ADC4/TCK(ADC 입력 채널 4 또는 JTAG Test Clock)
PF3	ADC3 (ADC 입력 채널 3)
PF2	ADC2 (ADC 입력 채널 2)
PF1	ADC1 (ADC 입력 채널 1)
PF0	ADC0 (ADC 입력 채널 0)

AVR 마이크로컨트롤러의 입출력 포트

❑ATMega128A의 범용 입출력 포트 : G 포트(PG4~PG0 : 핀19, 18, 43, 34, 33)

- ❖ 내부 풀업 저항이 있는 8비트 양방향 입출력 단자
- ❖ 외부 메모리 접속을 위한 스트로브 신호용, RTC(Real Time Counter) 타이머용 발진기 단자로도 사용.

포트 핀	부가기능
PG4	TOSC1(타이머/카운터0의 카운터로 동작할 때 클럭 입력)
PG3	TOSC2(타이머/카운터0의 32KHz 클럭 접속 입력)
PG2	ALE(외부메모리에 주소 래치 인에이블)
PG1	RD(외부메모리에 스트로브 읽기)
PG0	WR(외부메모리에 스트로브 쓰기)

실습 1 : GPIO를 이용하여 LED 켜기

❑실습 개요

- ❖ ATmega128A 마이크로컨트롤러의 GPIO를 이용하여 LED를 켜는 가장 단순한 실습
- ❖ 입출력 포트를 출력으로 설정하고, 그 포트를 이용하여 LED에 신호를 보내 점등
- ❖ 프로그램이 시작하면 1초 마다 LED 에 불이 점등.

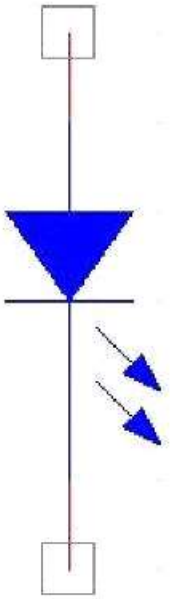
❑실습 목표

- ❖ GPIO 입출력 포트의 방향 제어 및 출력 제어 방법 습득
- ❖ LED 동작 원리 습득
- ❖ 프로그램에서 시간지연 방법 습득

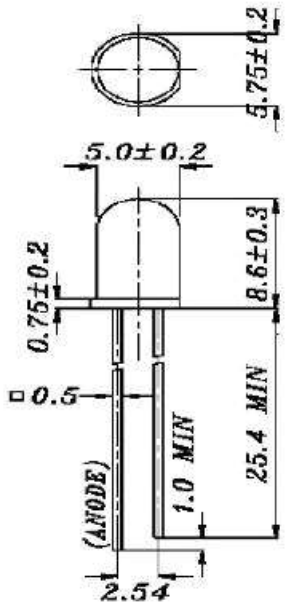
실습 1 : GPIO를 이용하여 LED 켜기

❑LED 구조

- ❖ LED(Light-emitting diode) : 빛을 발산하는 반도체 소자(발광 다이오드)
- ❖ 순방향에 전류를 흘리는 것에 따라 전자와 정공이 재결합하여 발광
- ❖ 다리가 긴 부분이 양극 (Anode), 짧은 부분이 음극 (Cathode)



심볼

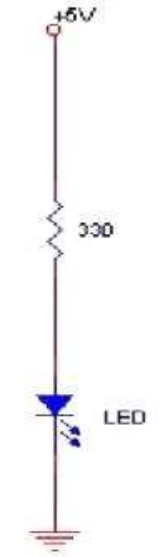


패키지 형상

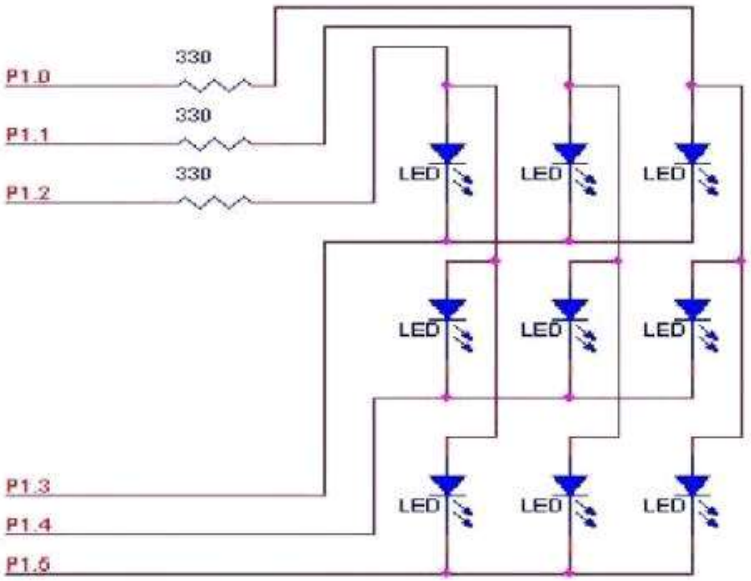
실습 1 : GPIO를 이용하여 LED 켜기

❑LED 구동방법

- ❖ 정적 구동방식 : 각각의 LED를 독립해서 구동
- ❖ 동적 구동방식 : 여러 개의 LED를 매트릭스 구조로 엮어서 함께 구동



정적구동

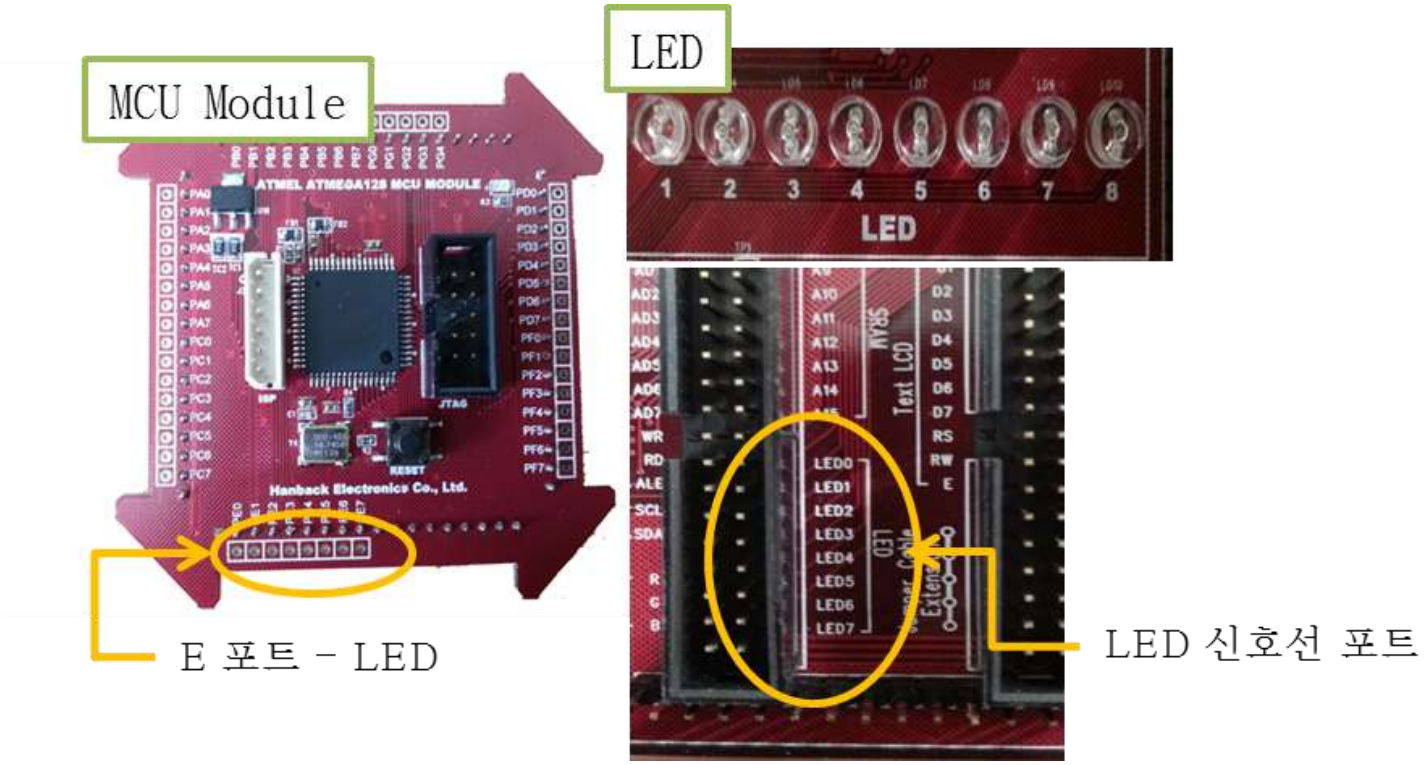


동적구동

실습 1 : GPIO를 이용하여 LED 켜기

❑사용 모듈

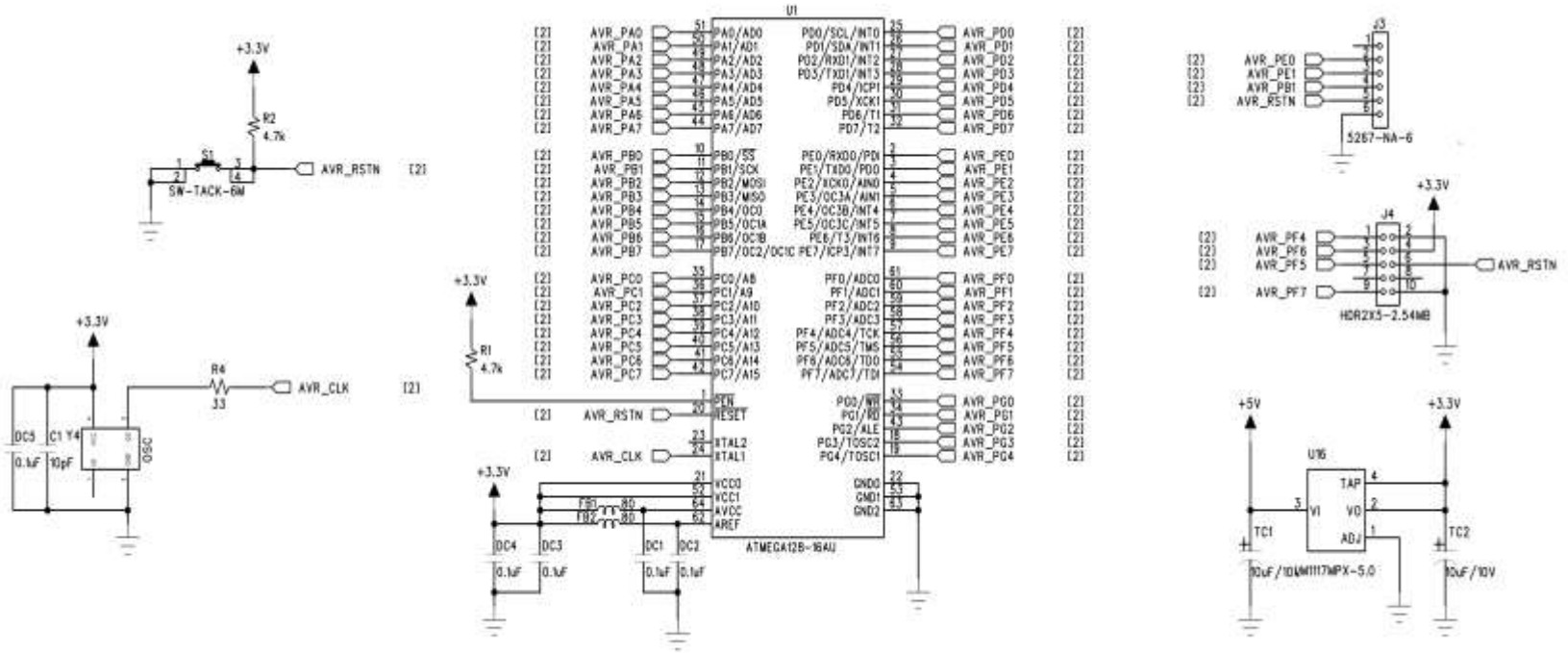
❖ MCU 모듈, LED 모듈



실습 1 : GPIO를 이용하여 LED 켜기

❑사용 모듈의 회로

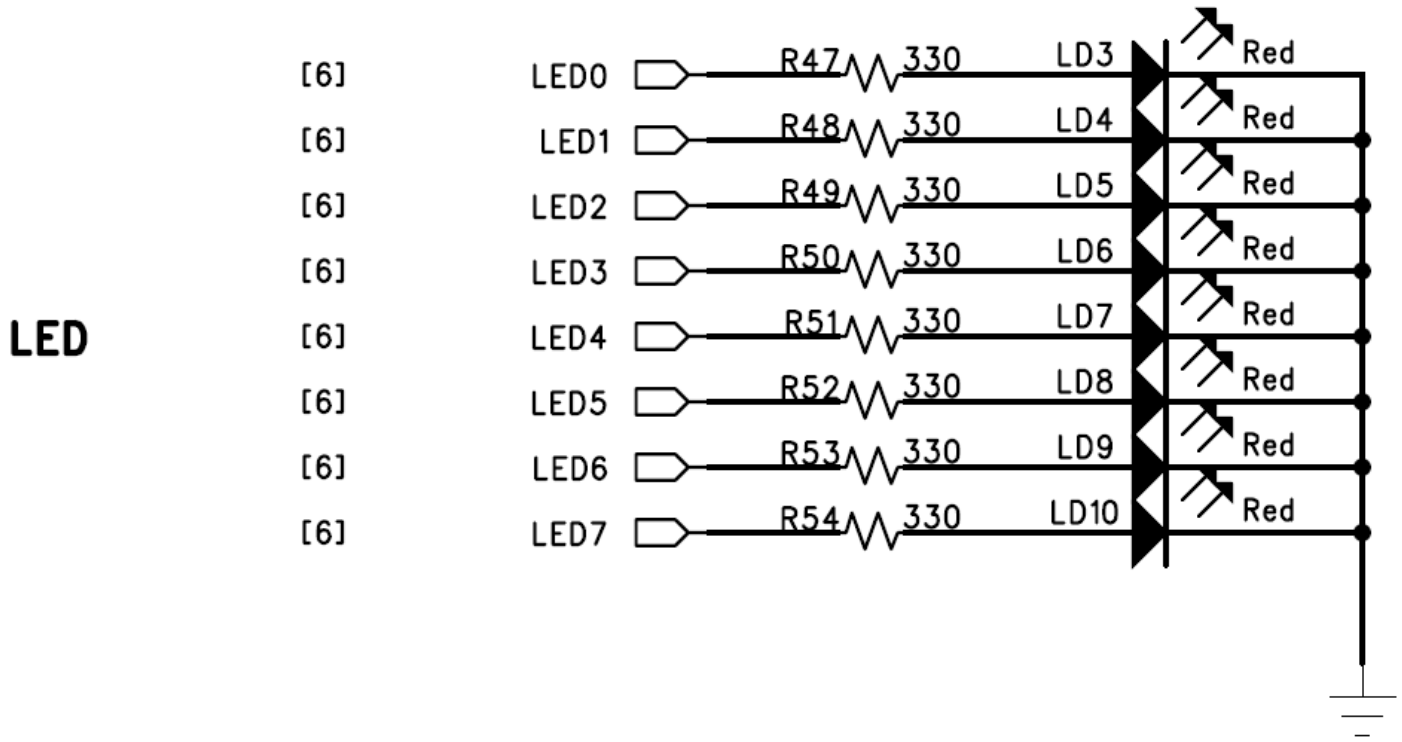
❖ MCU 모듈



실습 1 : GPIO를 이용하여 LED 켜기

❑사용 모듈의 회로

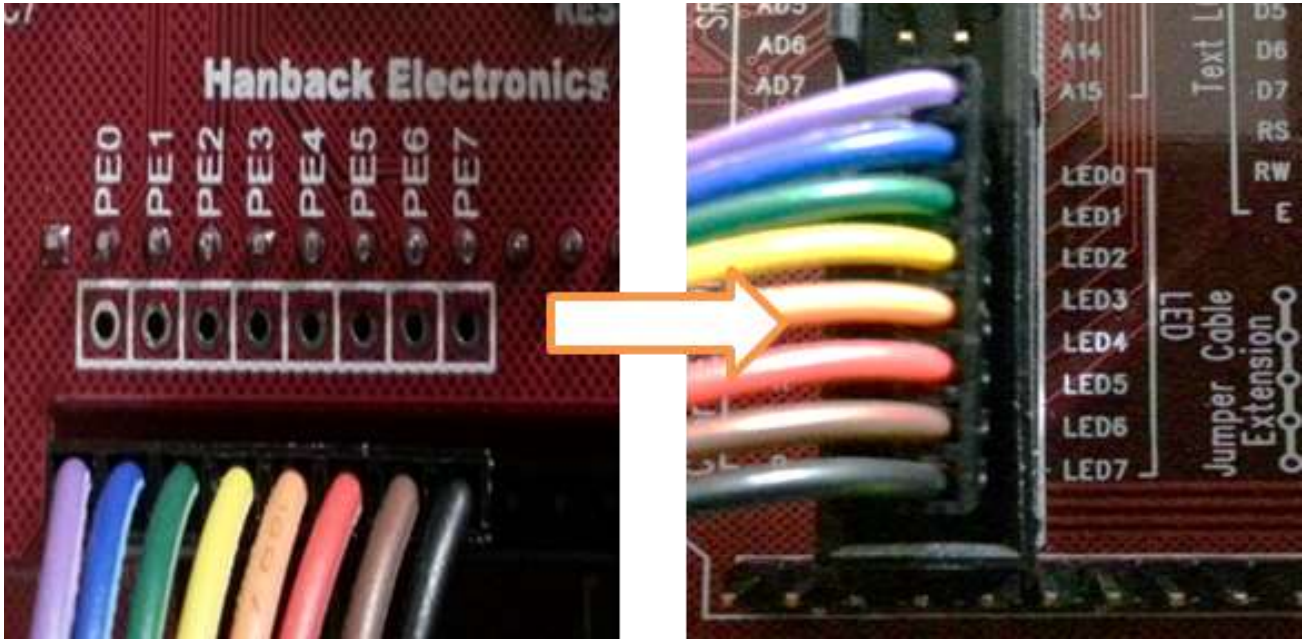
❖ LED 모듈



실습 1 : GPIO를 이용하여 LED 켜기

❑모듈 결선 방법

❖ MCU 모듈 포트 E의 PE0~PE7을 LED 모듈의 LED0~LED7핀에 연결



실습 1 : GPIO를 이용하여 LED 켜기

- 구동 프로그램 : 사전 지식
  - LED를 점등하기 위해서는 LED 신호에‘1’을 인가해야 함.
    즉, MCU E포트에서‘1’을 출력하도록 해야 함.
  - MCU E 포트에‘1’을 출력하려면
    - 입출력 포트 E의 GPIO 방향을 출력으로 만들어야 함.
    - 입출력 포트를 출력으로 선언하려면 DDRx 레지스터(여기서는 E 포트를 사용하므로 DDRE 레지스터)에‘1’을 적어 주어야 함.
    - PORTx 레지스터(여기서는 PORTE 레지스터)에‘1’을 적어주어야 함.

실습 1 : GPIO를 이용하여 LED 켜기

❑AVR 시스템 헤더 파일

헤더 파일명	설명
<avr/interrupt.h>	ATmega128A의 인터럽트에 관련된 내용을 정의
<avr/signal.h>	ATmega128A에서 발생되는 신호에 관련된 내용을 정의
<avr/pgmspace.h>	ATmega128A의 프로그램 공간에 관련된 내용을 정의
<avr/eeprom.h>	ATmega128A의 EEPROM에 관련된 내용을 정의
<avr/wdt.h>	ATmega128A의 워치독 타이머에 관련된 내용을 정의

실습 1 : GPIO를 이용하여 LED 켜기

- 마이크로컨트롤러 구동시 시간지연 방법
  - 반복문에 의한 시간 지연
    - for-loop나 while-loop를 사용하여 시간을 지연.

```
void delay(unsigned char i){
while(i--); }
혹은
void delay(unsigned char i){
int k;
for(k=0;k<=i;k++) ;
}
void main(void){
delay(0x0100);
}
```

    - 매우 부정확한 방법임(MCU상태, 클럭속도에 따라 달라짐)
    - 그러나 가장 손쉬운 방법.

실습 1 : GPIO를 이용하여 LED 켜기

❑마이크로컨트롤러 구동시 시간지연 방법

❖ 시스템 제공함수를 이용하는 시간 지연

- 시스템에서 소프트웨어적으로 제공하는 라이브러리 함수를 이용하여 시간지연을 하는 방법.
- AVR 개발환경에서 제공하는 시간지연용 함수들은 delay.h라는 헤더 화일에 정의되어 있음.
- _delay_ms(unsigned int i), _delay_us(unsigned int i)
- 비교적 정확한 시간지연을 얻을 수 있음.
- 인터럽트 등에 의해 지연 발생이 가능함.

❖ 하드웨어에 의한 시간 지연

- 마이크로컨트롤러에서 하드웨어로 제공하는 내부 타이머/카운터를 사용하는 방법.
- 가장 정확한 방법.

실습 1 : GPIO를 이용하여 LED 켜기

❑실행 결과

실습 1 : GPIO를 이용하여 LED 켜기

❑구동 프로그램 : 소스 분석

❖ led.c

1)	#include<avr/io.h>
	#include<util/delay.h>
	
	int main(){
	unsigned char LED_Data = 0x00;

2)	DDRE = 0xFF; //포트 E를(0~7비트까지 모두) 출력 포트로 사용

3)	while(1){
	PORTE = LED_Data; //포트E를 LED_Data로 두고, LED_Data를 하나씩 늘인다.
	LED_Data++;
	_delay_ms(1000); //ms단위의 딜레이함수
	}
	return 0;
	}

실습 2 : GPIO를 이용하여 푸쉬 버튼을 눌러 LED 켜기

❑실습 개요

- ❖ 단순 출력이 아니고, GPIO 포트를 통해 신호를 입력하여 그 신호에 따라 LED의 불을 켜는 실습
- ❖ 스위치 모듈의 스위치를 누르면 해당되는 LED 모듈의 LED가 점등되도록 함.
- ❖ 입출력 포트를 스위치쪽은 입력으로 LED쪽은 출력으로 설정하도록 함.

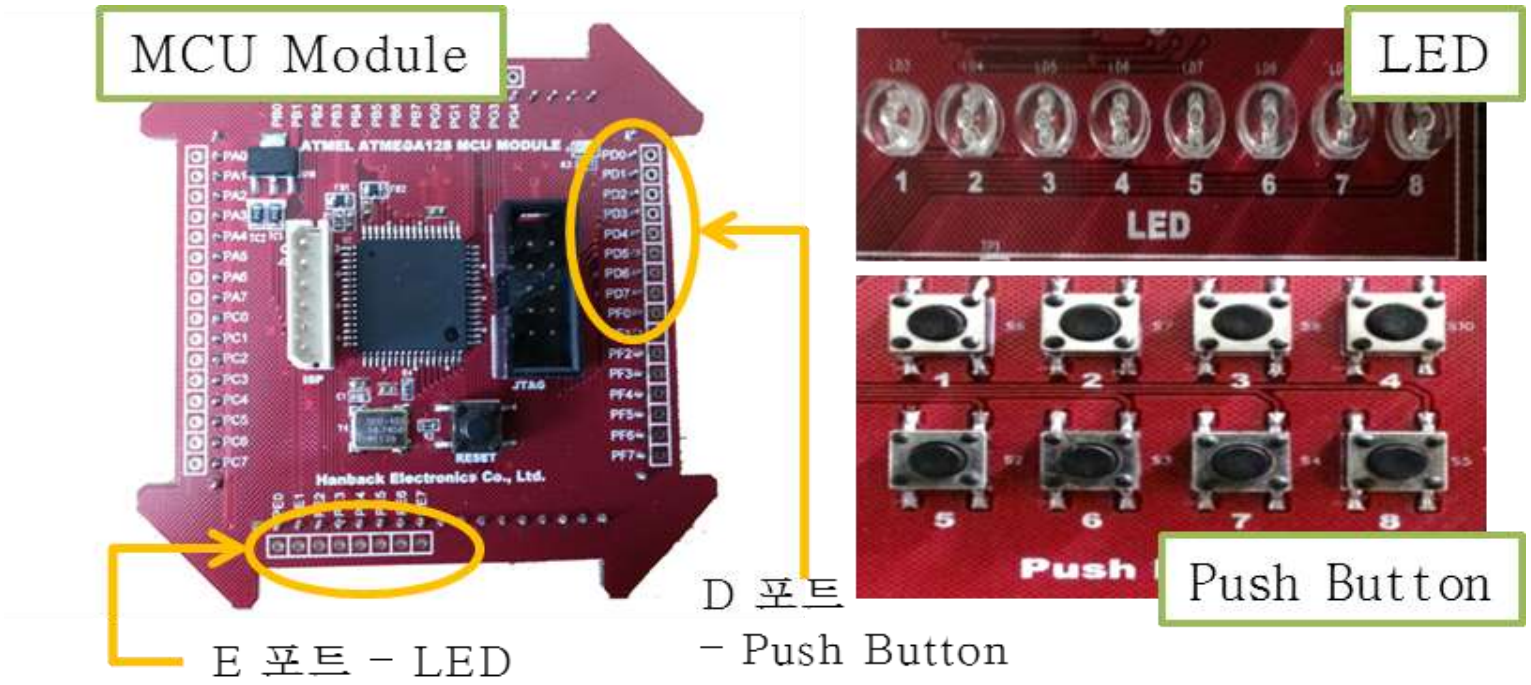
❑실습 목표

- ❖ GPIO 입출력 포트의 방향 제어 및 입력 제어 방법 습득
- ❖ 스위치 동작원리 습득

실습 2 : GPIO를 이용하여 푸쉬 버튼을 눌러 LED 켜기

❑사용 모듈

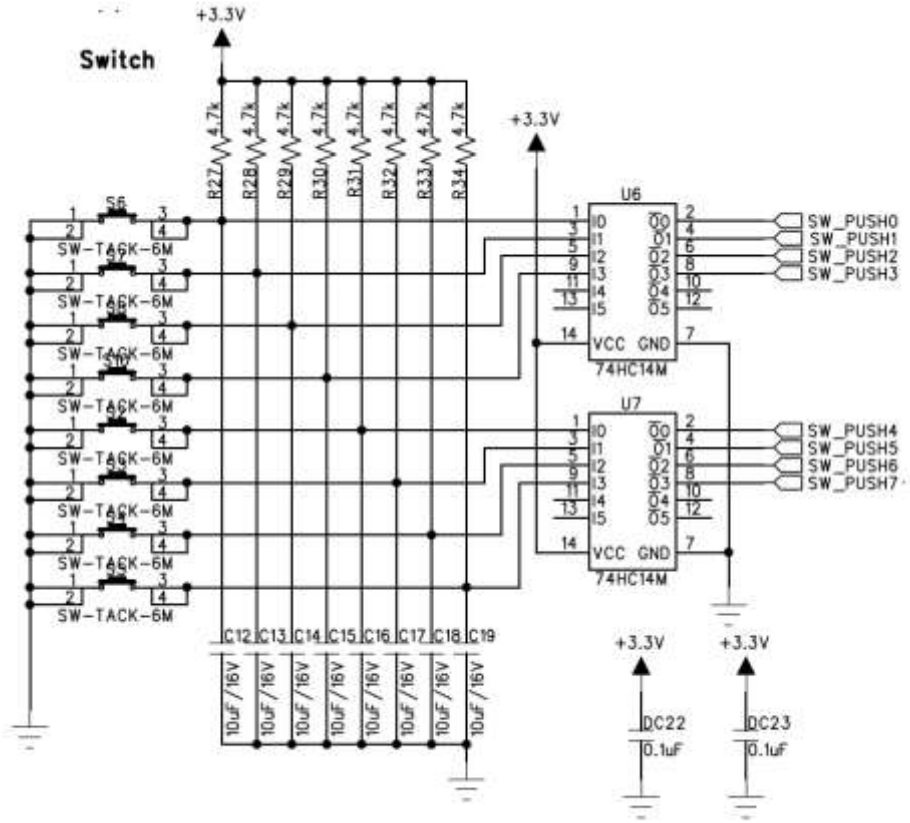
❖ MCU 모듈, Push Button 모듈, LED 모듈



실습 2 : GPIO를 이용하여 푸쉬 버튼을 눌러 LED 켜기

❑사용 모듈의 회로

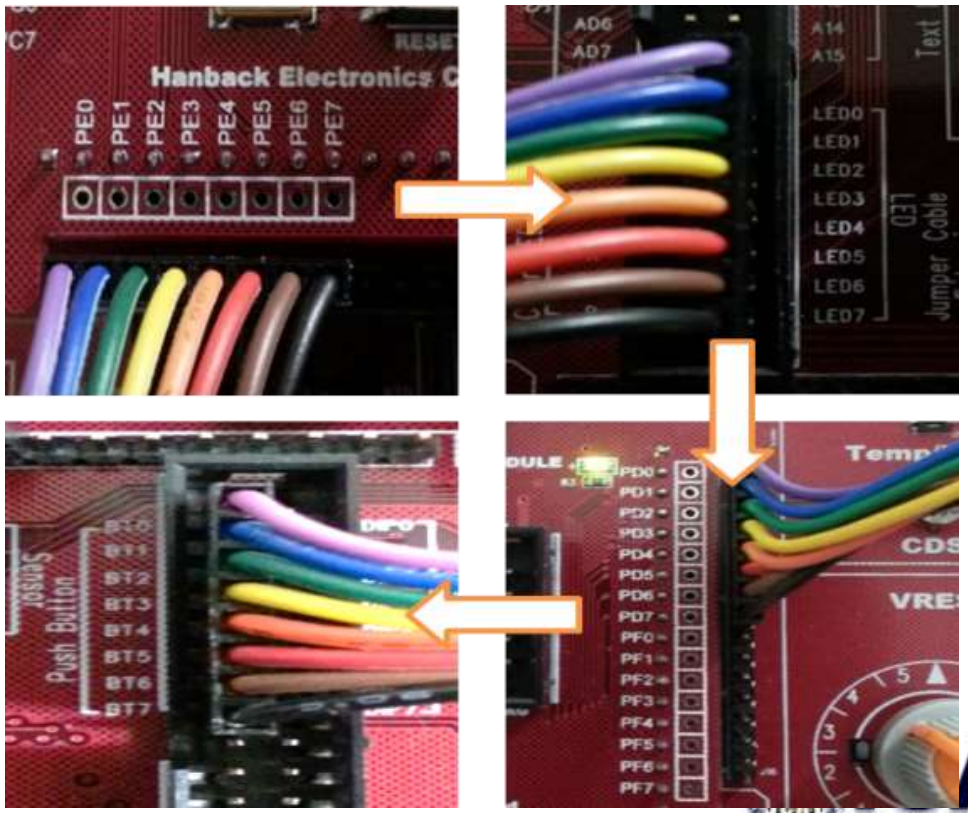
❖ Push Button 모듈



실습 2 : GPIO를 이용하여 푸쉬 버튼을 눌러 LED 켜기

❑모듈 결선 방법

- ❖ MCU 모듈의 포트 E의 PE0~PE7을 LED 모듈의 LED0~LED7까지 연결
- ❖ MCU 모듈의 포트 D의 PD0~PD7을 푸쉬 스위치 모듈의 BT0~BT7까지 연결



실습 2 : GPIO를 이용하여 푸쉬 버튼을 눌러 LED 켜기

❑구동 프로그램 : 사전 지식

- ❖ 스위치를 누르면“1”신호가 나오고 놓으면“0”신호가 나옴.
- ❖ 이 신호를 입력 받기 위해서는
  - MCU의 입출력 포트를 입력으로 선언해야 함. 즉, 입력으로 사용하기로 한 MCU D 포트를 입력으로 선언해야 함.
  - 입출력 포트를 입력으로 선언하려면 DDRx 레지스터(여기서는 D 포트를 사용하므로 DDRD 레지스터)에‘0’을 적어 주어야 함.
  - 스위치 모듈의 버튼을 누른다면 PINx 레지스터(여기서는 D 포트를 사용하므로 PIND 레지스터)에 ‘1’이라는 값이 입력되어 들어옴.
- ❖ LED 출력 방법은 앞의 예제와 동일

실습 2 : GPIO를 이용하여 푸쉬 버튼을 눌러 LED 켜기

❑실행결과

❖ Switch모듈의 눌러진 버튼과 같은 LED의 불이 점등한다.

1 버튼을 누른다.

2 Push Button을 누르면 해당 LED가 켜진다.

실습 2 : GPIO를 이용하여 푸쉬 버튼을 눌러 LED 켜기

❑구동 프로그램 : 소스분석

❖ switch.c

1)	#include<avr/io.h> int main(){

2)	DDRE = 0xFF; //포트E를 출력포트로 사용 (0~7비트까지 모두사용)
	DDRD = 0x00; //포트D를 입력포트로 사용 (0~7비트까지 모두사용)

3)	while(1){
	PORTE = PIND; /* 포트 E를 포트 D의 핀으로 둠
	(PORT는 R/W모두 가능하지만, PIN은 R만 가능) */
	}
	return 0;
	}

실습 3 : GPIO를 이용하여 FND LED 켜기

❑실습 개요

- ❖ 단순 LED가 아닌 FND(Flexible Numeric Display: 7-Segment LED)를 이용하여 숫자를 표시하는 실습
- ❖ 마이크로컨트롤러의 포트를 출력으로 선언하고, 이 포트를 Array FND 모듈의 신호선 포트에 연결함.
- ❖ 일정 시간마다 클럭에 의해 Array FND에 숫자와 문자가 디스플레이 되도록 함.

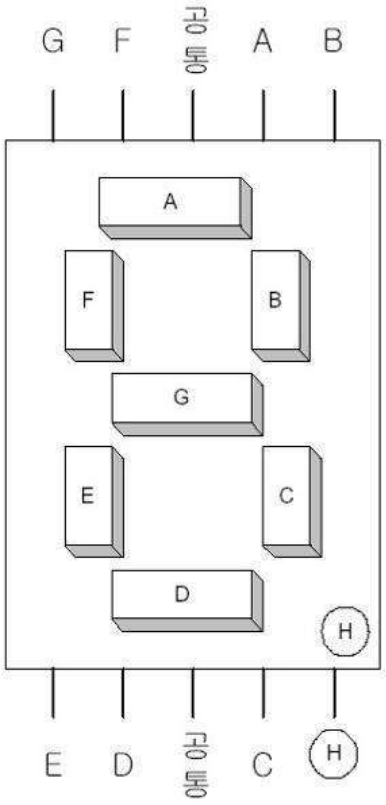
❑실습 목표

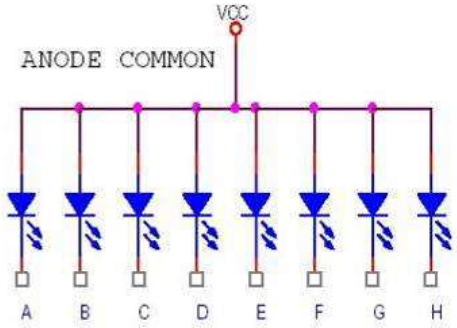
- ❖ GPIO 입출력 포트의 방향 제어 및 출력 제어 방법 습득
- ❖ FND LED 동작원리 습득

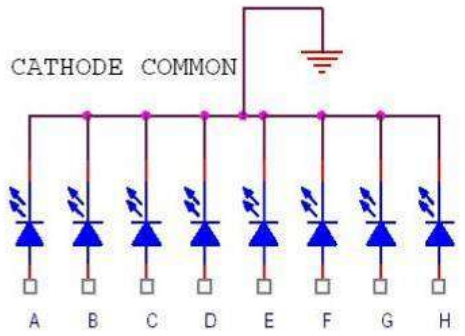
실습 3 : GPIO를 이용하여 FND LED 켜기

❑FND(7-Segment LED) 구조

- ❖ 7-세그먼트는 LED 8개를 그림과 같이 배열
- ❖ 숫자나 간단한 기호 표현에 많이 사용됨.







공통(Common)단자에 인가되는 전원에 따라서

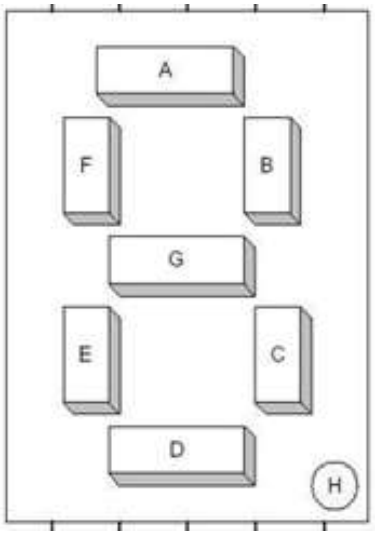
Common Anode(+공통)과 Common Cathode(-공통)으로 분류

실습 3 : GPIO를 이용하여 FND LED 켜기

❑FND(7-Segment LED) 구동방법

❖ Common-Cathode방식 : 각 단자에 '1'이 입력되면 해당 LED가 켜짐

7-Segment 에서 16진수 표시방법

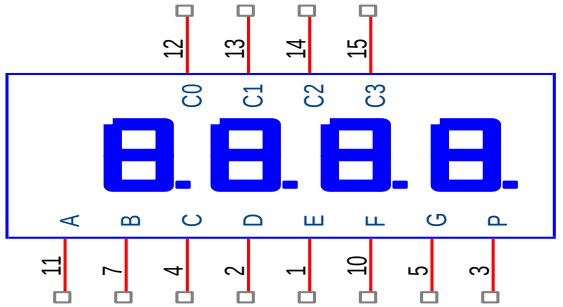


16 진수	7-세그먼트의 비트값								데이터 값 (HEX)
	H	G	F	E	D	C	B	A	
0	0	0	1	1	1	1	1	1	0X3F
1	0	0	0	0	0	1	1	0	0X06
2	0	1	0	1	1	0	1	1	0X5B
3	0	1	0	0	1	1	1	1	0X4F
4	0	1	1	0	0	1	1	0	0X66
5	0	1	1	0	1	1	0	1	0X6D
6	0	1	1	1	1	1	0	1	0X7D
7	0	0	1	0	0	1	1	1	0X27
8	0	1	1	1	1	1	1	1	0X7F
9	0	1	1	0	1	1	1	1	0X6F
A	0	1	1	1	0	1	1	1	0X77
B	0	1	1	1	1	1	0	0	0X7C
C	0	0	1	1	1	0	0	1	0X39
D	0	1	0	1	1	1	1	0	0X5E
E	0	1	1	1	1	0	0	1	0X79
F	0	1	1	1	0	0	0	1	0X71

실습 3 : GPIO를 이용하여 FND LED 켜기

❑Array FND(7-Segment LED) 구조와 구동 방법

- ❖ 눈의 착시현상을 이용하여 숫자를 표시
- ❖ Data선(A~P)을 서로 공통으로 연결
- ❖ C0를 제외한 다른 com단자(C1~C3)에 대해 GND(0전위)가 안 되도록 한다. 그러면 1개의 7-Segment만 숫자가 표시될 것이다.
- ❖ 이 방법을 눈이 알아차리기 전에 매우 빠른 속도로 번갈아 가면서 반복 동작



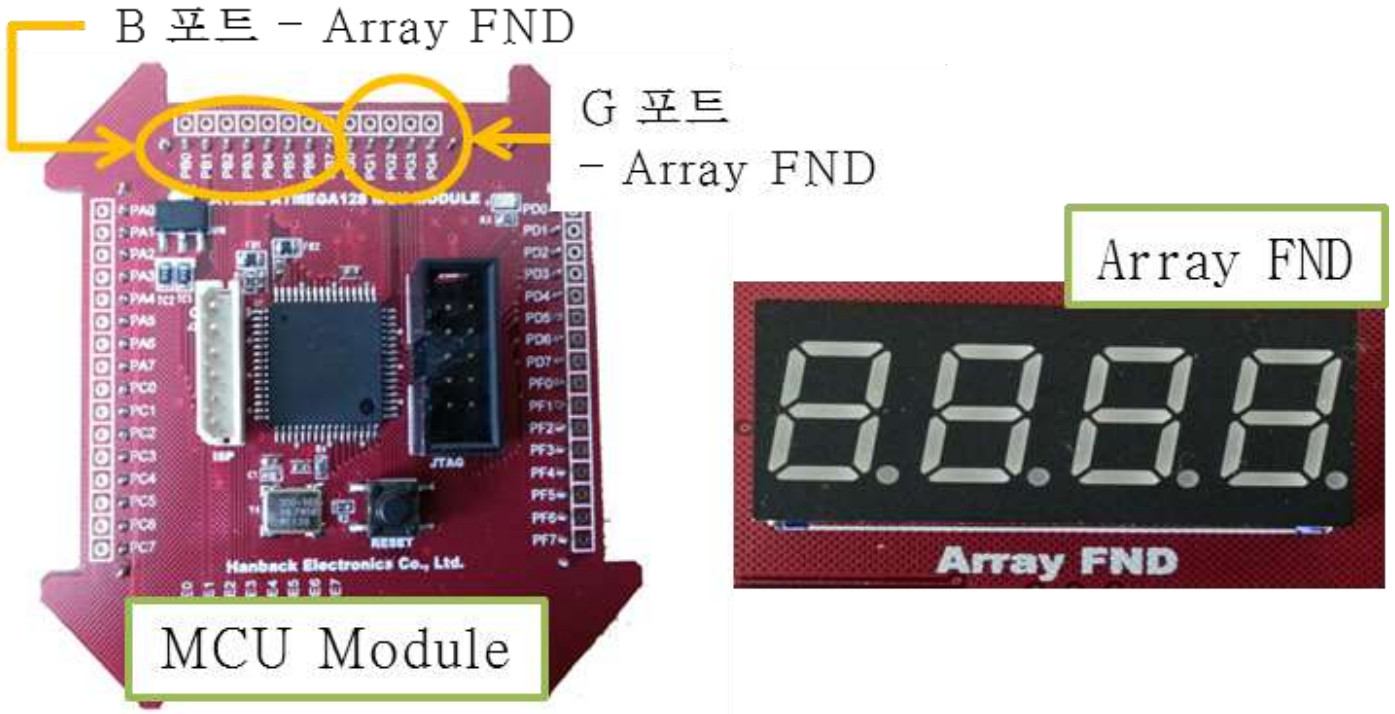
❑특징

- ❖ 전력소모가 적다.
- ❖ data선을 최소로 줄일 수 있다.
- ❖ 깜박거림에 의해 눈이 피로해 질 수도 있다.
- ❖ 프로그램이 다소 어렵다.

실습 3 : GPIO를 이용하여 FND LED 켜기

❑사용 모듈

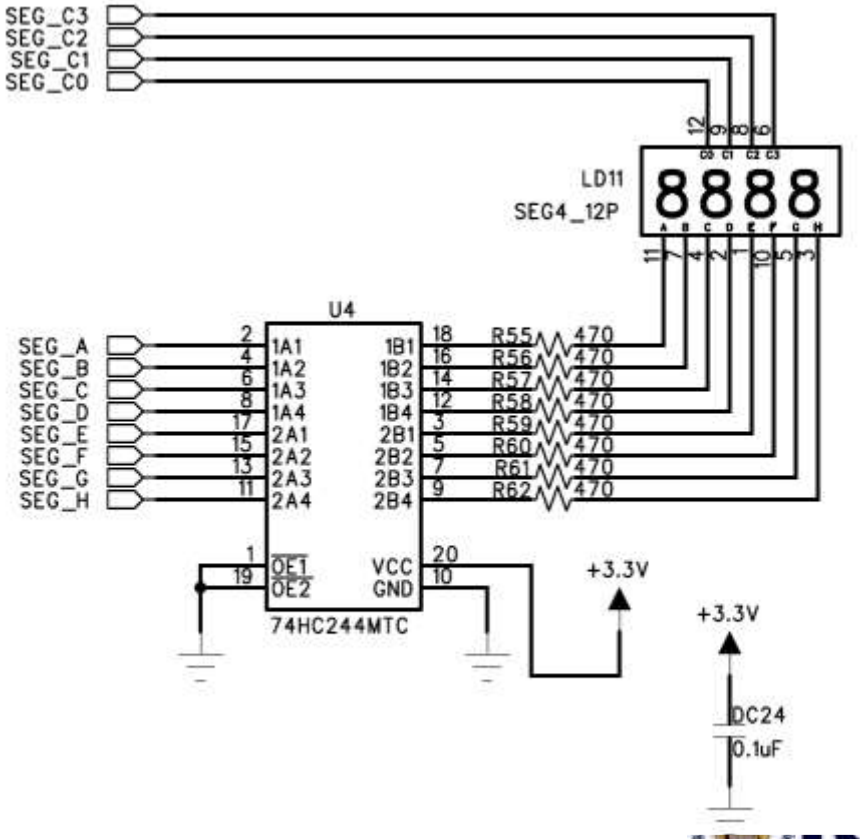
❖ MCU 모듈, FND 모듈



실습 3 : GPIO를 이용하여 FND LED 켜기

❑사용 모듈의 회로

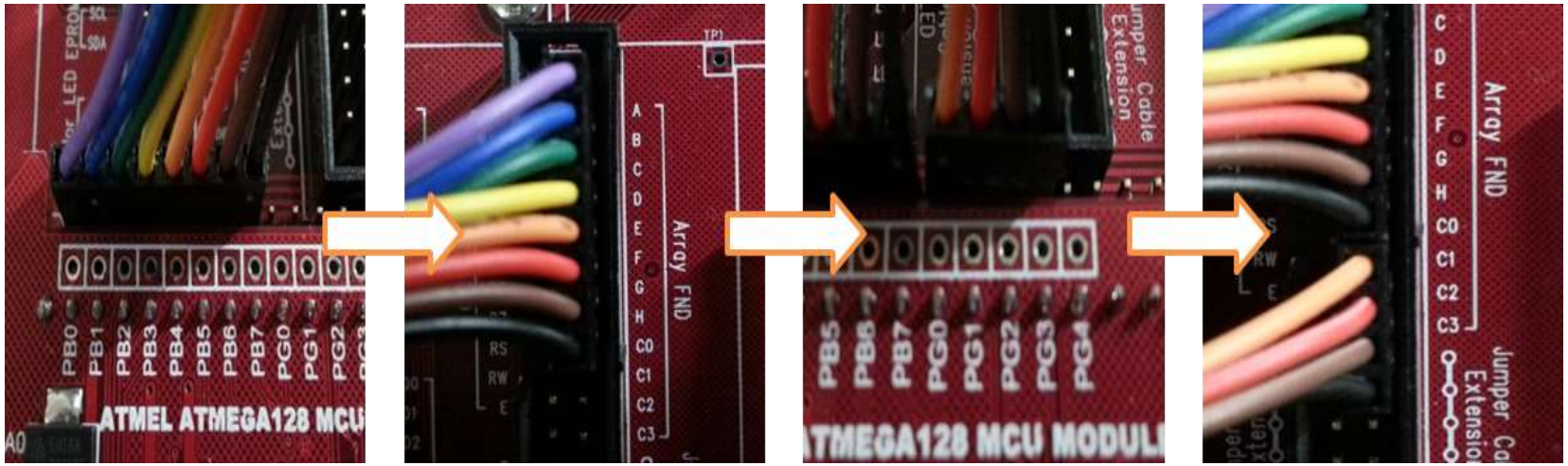
- FND 모듈(Common-Cathode)



실습 3 : GPIO를 이용하여 FND LED 켜기

❑모듈 결선방법

- ❖ MCU 모듈 포트 B의 PB0~PB7을 Array FND 모듈의 A~H 로 연결
- ❖ 포트 G의 PG0~PG3을 Array FND 모듈의 C0~C3까지 연결



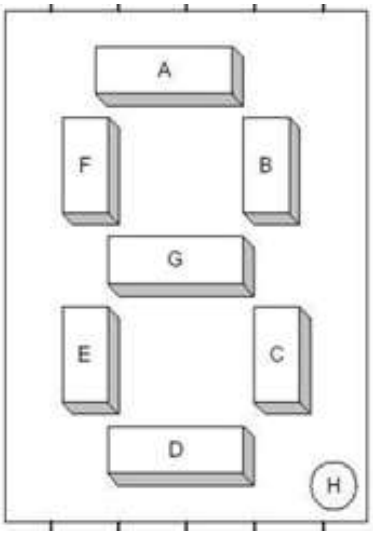
실습 3 : GPIO를 이용하여 FND LED 켜기

❑구동 프로그램 : 사전지식

❖ MCU 모듈의 B포트를 FND의 불을 켜기 위한 출력 포트로 설정.

- DDRx 레지스터(여기서는 DDRB 레지스터)에 '1'을 적어줌.

❖ 표를 참조하여 PORTx(여기서는 PORTB 레지스터)에 '1'을 출력.



16 진수	7-세그먼트의 비트값								데이터 값 (HEX)
	H	G	F	E	D	C	B	A	
0	0	0	1	1	1	1	1	1	0X3F
1	0	0	0	0	0	1	1	0	0X06
2	0	1	0	1	1	0	1	1	0X5B
3	0	1	0	0	1	1	1	1	0X4F
4	0	1	1	0	0	1	1	0	0X66
5	0	1	1	0	1	1	0	1	0X6D
6	0	1	1	1	1	1	0	0	0X7C
7	0	0	0	0	0	1	1	1	0X07
8	0	1	1	1	1	1	1	1	0X7F
9	0	1	1	0	0	1	1	1	0X67
A	0	1	1	1	0	1	1	1	0X77
B	0	1	1	1	1	1	0	0	0X7C
C	0	0	1	1	1	0	0	1	0X39
D	0	1	0	1	1	1	1	0	0X5E
E	0	1	1	1	1	0	0	1	0X79
F	0	1	1	1	0	0	0	1	0X71

실습 3 : GPIO를 이용하여 FND LED 켜기

❑실행 결과

❖ 프로그램이 시작하면 500ms 마다 FND 에 0부터 9, A ~ F 그리고 '_' , '.' 을 순차적으로 출력한다.

실습 3 : GPIO를 이용하여 FND LED 켜기

❑구동 프로그램 : 소스분석

❖ FND.C

1)

```
#include<avr/io.h>
#include<util/delay.h>
int main(){
```

AVR 입출력에 대한 헤더 파일과 delay 함수사용을 위한 헤더파일을 선언한다

2)

```
unsigned char FND_DATA_TBL [] = {0x3F, 0X06, 0X5B, 0X4F, 0X66, 0X6D, 0X7C,
0X07, 0X7F, 0X67, 0X77, 0X7C, 0X39, 0X5E, 0X79, 0X71, 0X08, 0X80};
unsigned char cnt=0;
```

3)

```
DDRG = 0x0F;          //포트 G의 하위 4비트를 출력포트로 사용(FND 제어핀)
DDRB = 0xFF;          //포트 B를 출력포트로 사용(0~7 비트까지 모두사용)
```

4)

```
while(1){
PORTB = FND_DATA_TBL[cnt];
cnt++;
if(cnt>17) cnt=0;          //테이블 크기를 초과하는 경우 방지.
_delay_ms(500);
}
return 0;
}
```

FND_DATA_TBL [] : 7-Segment에 표시할 글자의 입력 데이터를 저장

출력되는 데이터는
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, _, .}