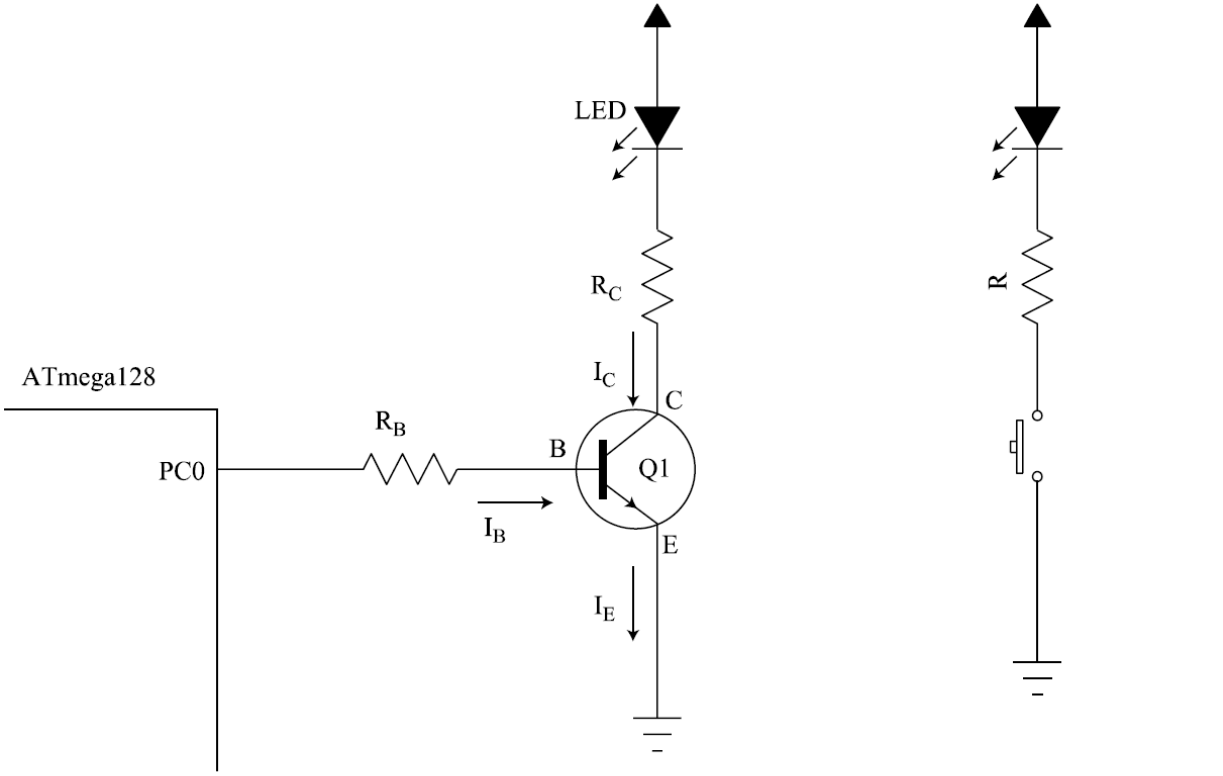
□ 스위치 ON/OFF의 전기적 특성

➢ 트랜지스터와 기계적인 스위치를 이용한 LED ON/OFF 동작 비교

- LED를 켜기 위한 회로([그림 5-6])



(a) 트랜지스터가 연결된 LED

(b) 기계적인 스위치가 연결된 LED

□ 트랜지스터의 직류 특성

- **활성**
  - 직류 상황에서 전류 I_C는 전류 I_B보다 h_{FE}배만큼 더 많은 전류가 흐름
    - » 베이스와 이미터가 순방향으로 바이어스
    - » 컬렉터와 베이스가 역방향으로 바이어스
- **차단**
  - 만일 I_B가 0[A]이면 컬렉터 전류 I_C는 거의 0[A], 트랜지스터는 차단
- **포화**
  - 베이스와 이미터, 컬렉터와 베이스가 모두 순방향
  - I_B 가 계속 증가하더라도 최대로 흐를 수 있는 이상 I_C는 증가하지 않음

□ 트랜지스터 포화/차단과 스위치 ON/OFF 관계

➢ 트랜지스터 포화/차단 전기적 특성

트랜지스터 상태	컬렉터 전류 I_C	컬렉터와 이미터 양단 전압차 V_{CE}	컬렉터에서 본 저항(V_{CE}/I_C)
포화	흐른다($\neq 0$[A]).	매우 작다(0.3[V] 이하).	매우 작다.
차단	거의 흐르지 않는다($\fallingdotseq 0$[A]).	있다($\neq 0$[V]).	고저항

➢ 기계적인 스위치 ON/OFF에 의한 전기적 특성

스위칭 상태	전류	스위치 양단 전압차	저항
ON	흐른다($\neq 0$[A]).	없다(0[V]).	거의 0[Ω]
OFF	흐르지 않는다(0[A]).	있다($\neq 0$[V]).	고저항

□ ATmega128 디지털 I/O 핀을 이용한 트랜지스터 스위치 ON/OFF

➢ **[그림 5-6(a)] 회로의 트랜지스터 스위치를 ON 동작**

- PC0 출력 핀에 HIGH 디지털 전압레벨 출력
- R_B가 작으면 트랜지스터는 포화 영역에 있게 됨
- 트랜지스터는 스위치 ON 동작과 같게 됨
- 트랜지스터의 I_C 정격 전류까지 흐르게 할 수 있음

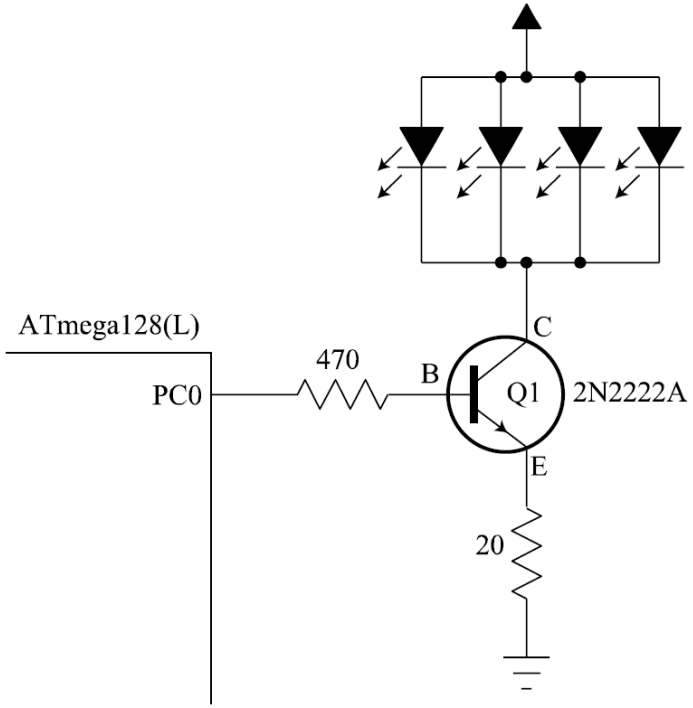
➢ **[그림 5-6(a)] 회로의 트랜지스터 스위치를 OFF 동작**

- PC0 출력 핀에 LOW 디지털 전압레벨을 출력
- 베이스 전류가 거의 0[A], 트랜지스터는 차단 영역에 있게 됨
- 트랜지스터는 스위치 OFF 동작을 하게 됨

□ NPN 트랜지스터를 이용한 구동 회로

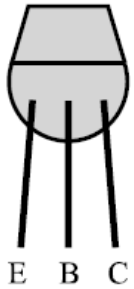
➢ **ATmega128 정격을 초과하는 전류 공급 사례**

- 고휘도 LED 4개를 구동
- PC0에 HIGH 디지털 전압레벨이 출력
  - » Q1의 스위칭 동작은 ON
  - » LED 켜기에 충분한 전류가 흐름
- LOW 디지털 전압레벨이 출력
  - » Q1의 스위칭 동작은 OFF
  - » LED 꺼짐



[NPN 트랜지스터를 이용한 정격 초과전류 공급 회로]

- 범용 NPN 트랜지스터 2N2222A의 포화, 차단 스위칭 특성(Philips datasheet 참조)



심볼	매개 변수		값	단위
	내용	테스트 조건		
I_{CBO}	컬렉터 차단 전류	V_{CE} = 60[V], $V_{EB(off)}$ = 3.0[V]	최대 10	[nA]
$V_{CE(sat)}$	컬렉터-이미터 포화 전압	I_C = 150[mA]	최대 0.3	[V]
$V_{BE(sat)}$	베이스-이미터 포화 전압	I_C = 150[mA]	0.6~1.2	[V]
I_C	컬렉터 정격 전류		1.0	[A]

- **I_C = 75[mA] 계산 값**
  - VCE(Sat)=0.3[V], LED 전압강하 3.2[V]

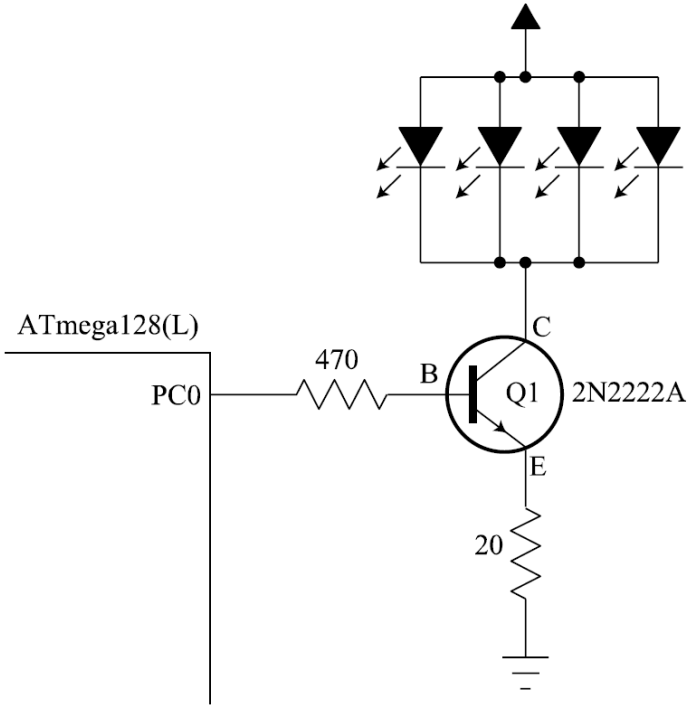
$$5[V] = 3.2[V] + 0.3[V] + 20\,I_C$$

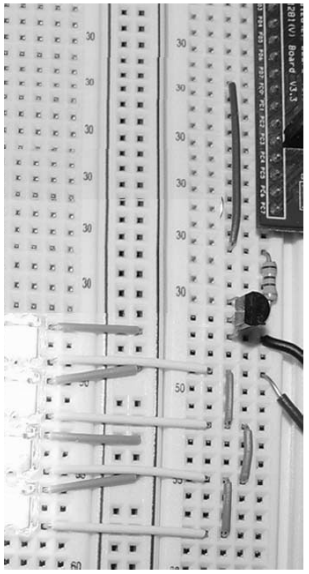
$$I_C = \frac{5[V] - 3.5[V]}{20} = 75[mA]$$

- **I_B = 6[mA] 계산 값, 출력 핀으로 충분히 공급할 수 있음**
  - 20[Ω] 양단의 전압강하는 5-3.2-0.3 = 1.5[V]

$$I_B = \frac{5 - 1.5 - 0.7}{470} = 6.0[mA]$$

- **측정값 77[mA]**

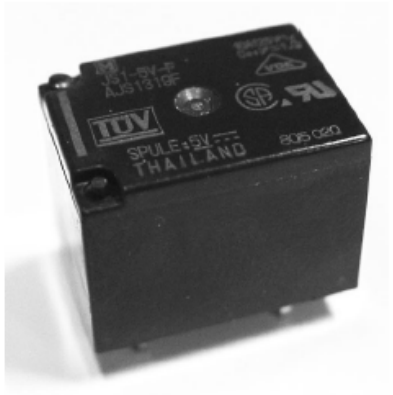




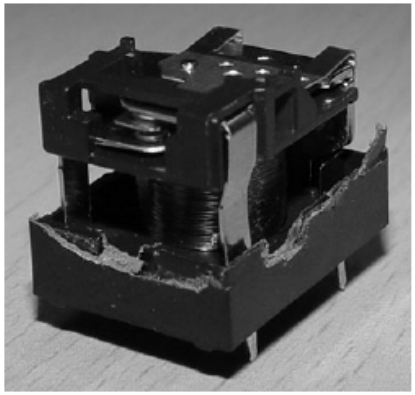
□ 사전 지식

➢ 릴레이 동작

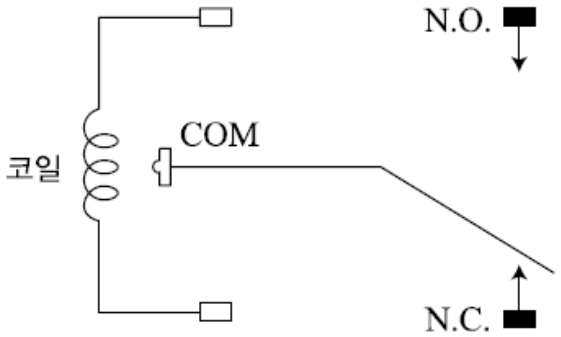
- 코일에 전류가 흐르면, 코일은 전자석이 되어 위의 판을 끌어당김
- COM과 연결된 접점의 접촉 위치가 변경
- 평상시 N.C.(Normal Close) 위치, 전자석이 되면 N.O.(Normal Open)과 연결



(a) 외관



(b) 내부 구조

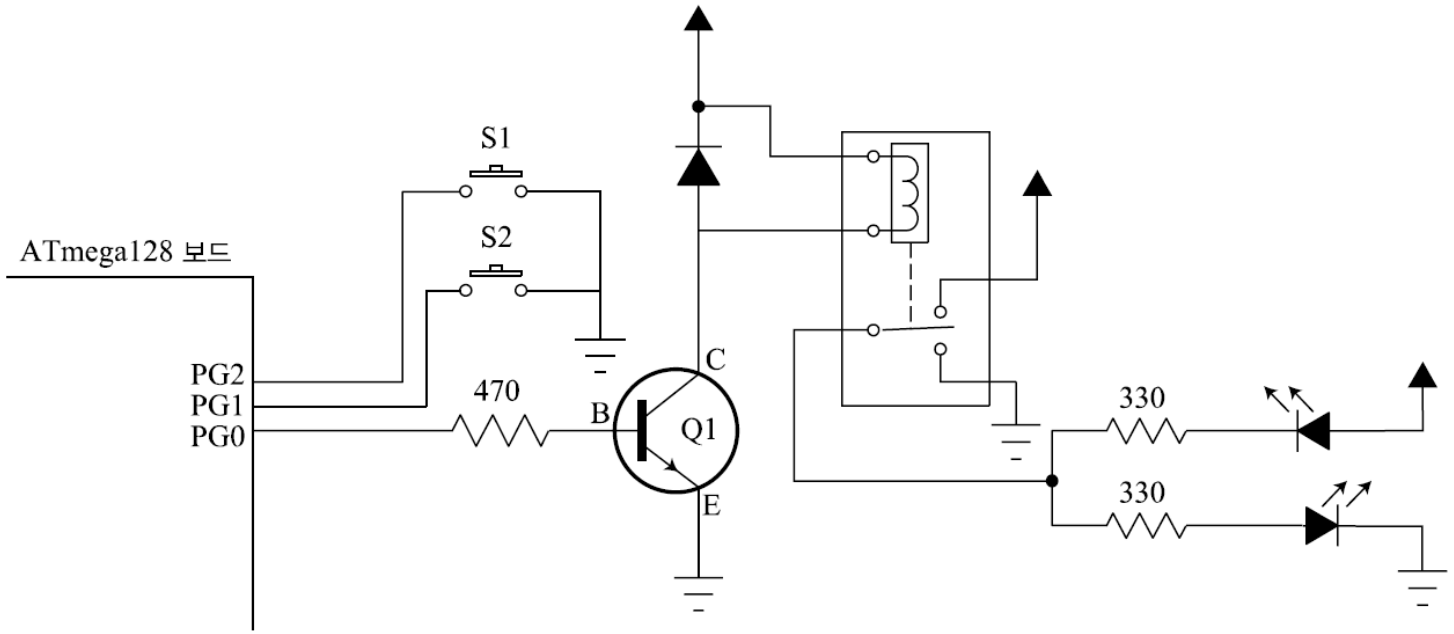


(c) 접점 구조

[소형 릴레이 JS1F-5V 구조 [출처 : NAIS JS Relay 데이터시트(c)]]

□ 소형 릴레이 JS1F-5V 사례

- 전자석이 되기 위해 5[V] 조건에서 72[mA]가 필요
- 디지털 I/O 핀으로 직접 구동이 어려움
- 출력핀으로 트랜지스터를 구동한 컬렉터 전류를 사용하여 간접 구동
  - 릴레이 구동을 위한 회로



- 릴레이 구동을 위한 프로그램

```
1   #include   <avr/io.h>
2
3   int main(void)
4   {
5       DDRG = (DDRG & ~(1<<PG1 | 1<<PG2)) | 1<<PG0;
6       PORTG = PORTG | 1<<PG1 | 1<<PG2;
7
8       while(1){
9           if( !(PING & 1<<PG1))
10              PORTG = PORTG | 1<<PG0;
11
12          if( !(PING & 1<<PG2))
13              PORTG = PORTG & ~(1<<PG0);
14      }
15
16      return 0;
17  }
```

핀 신호 방향 설정

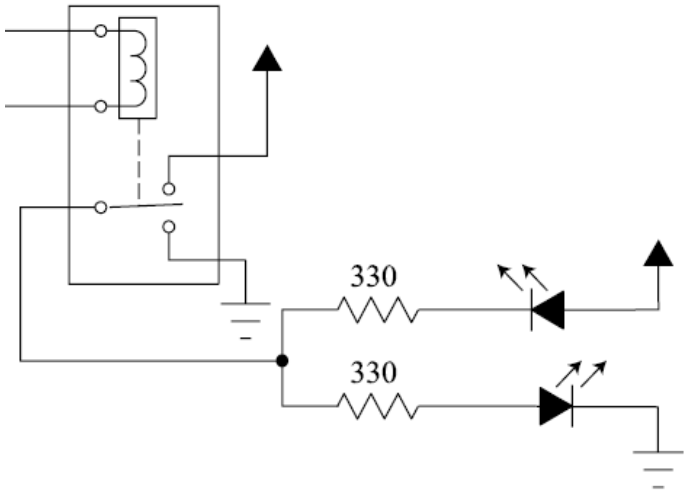
입력신호 내부 풀업

PG1 스위치 눌림을 폴링으로 감지

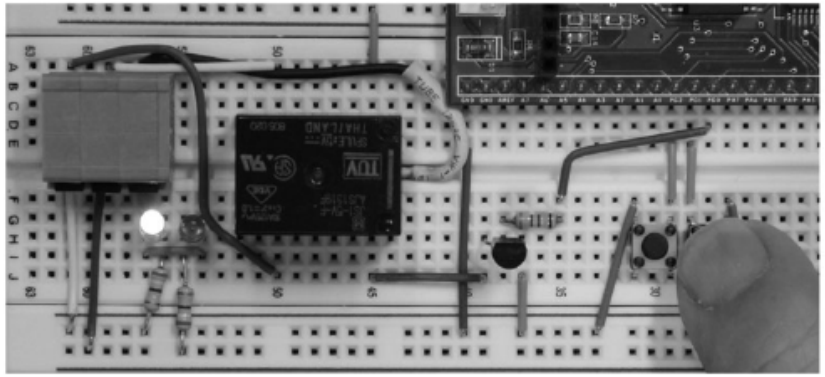
PG2 스위치 눌림을 폴링으로 감지

□ 릴레이 구동 회로 동작을 LED로 확인

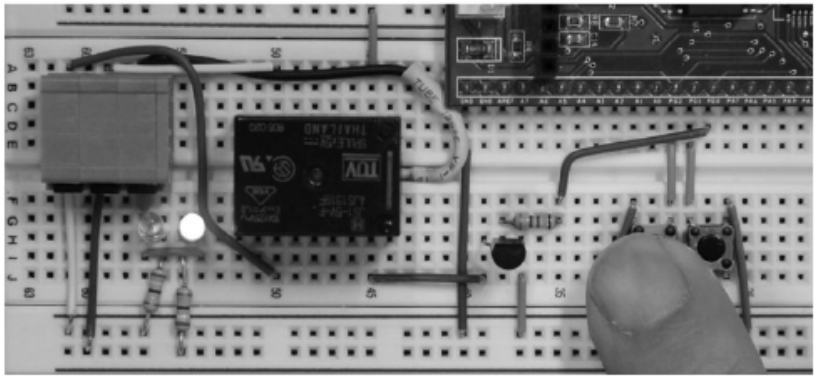
- N.C.가 연결될 때
- N.O.이 연결될 때
- 서로 다른 LED가 켜지게 구성된 회로



- 릴레이 ON/OFF 작동



(a) 릴레이 ON



(b) 릴레이 OFF

□ 실험 5-5 릴레이 구동 (video.zip 참고 / 5-5)

□ GNU gcc 컴파일러의 라이브러리 시간 지연 함수

- **CPU가 외부 장치에 명령을 보낸 후, 적절한 동작이 완성되기까지 기다려야 하는 경우가 많기 때문**

- **시간지연 라이브러리 함수**

```
void _delay_us(double __us)
void _delay_ms(double __ms)
```

  - <util/delay.h>를 include 해야 사용할 수 있음

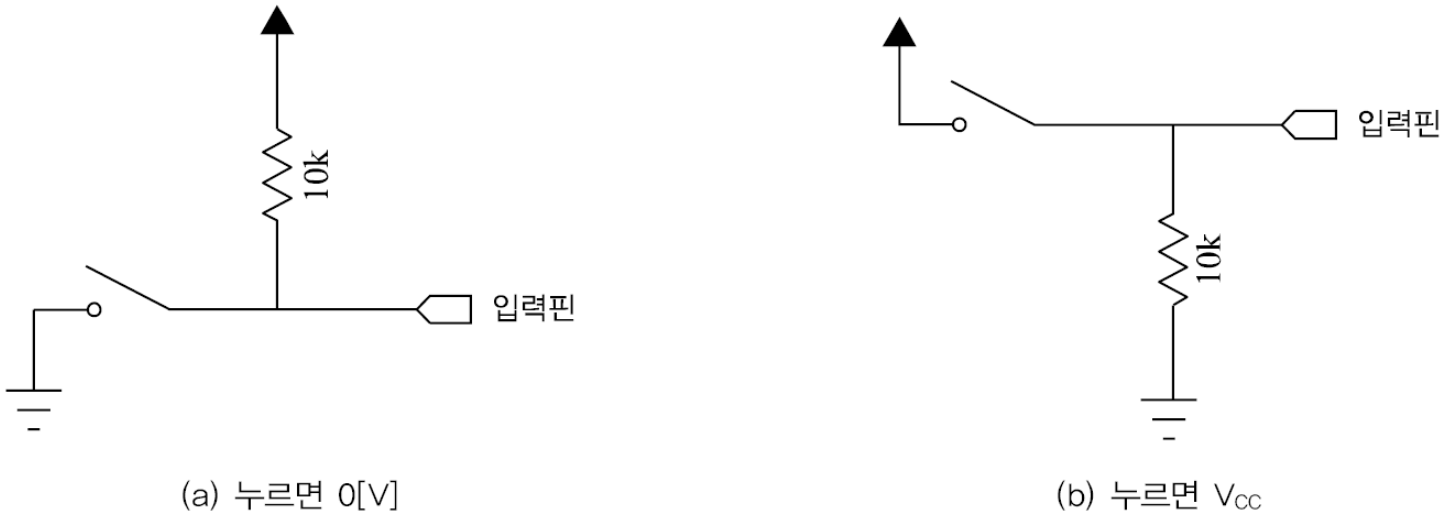
- **위 함수 사용을 위해 F_CPU가 정의하는 방법**
  - Project 메뉴→Configuration Options→General → Frequency 항 입력

  <혹은>

  - #define F_CPU 16000000UL로 정의, CPU 주파수가 16MHz인 경우

□ 스위치 디바운싱 현상과 소프트웨어적인 해결 방법

➢ 입력 핀에 인가되는 전압을 결정하는 회로



(a) 누르면 0[V]

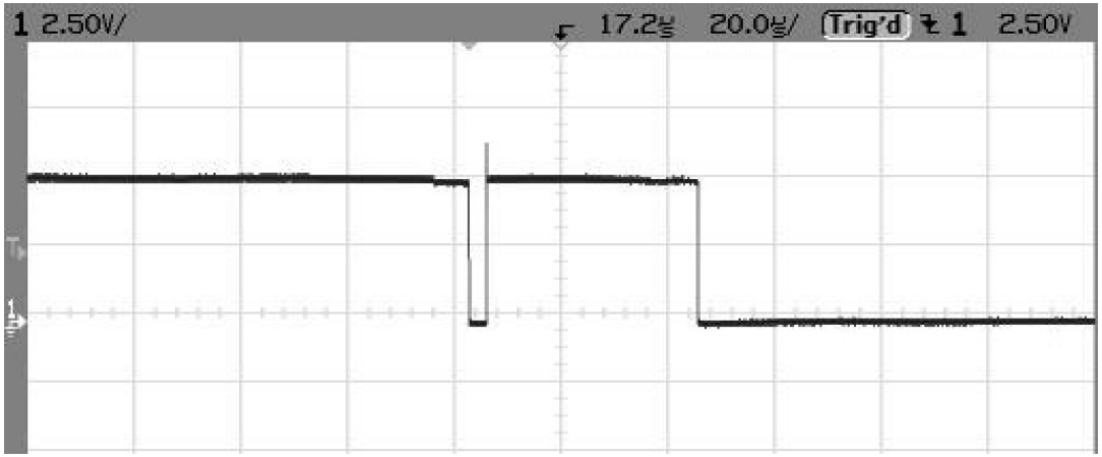
(b) 누르면 V_{CC}

[통상적인 푸시버튼 스위치 입력 회로]

➢ 스위치 바운싱 현상

- 스위치를 여러 번 누른 동작으로 오동작할 수 있음
- 스위치의 기계적인 특성으로 발생 → 전압 변화 유발
- 소프트웨어적으로 **스위치 디바운싱** 역할 수행

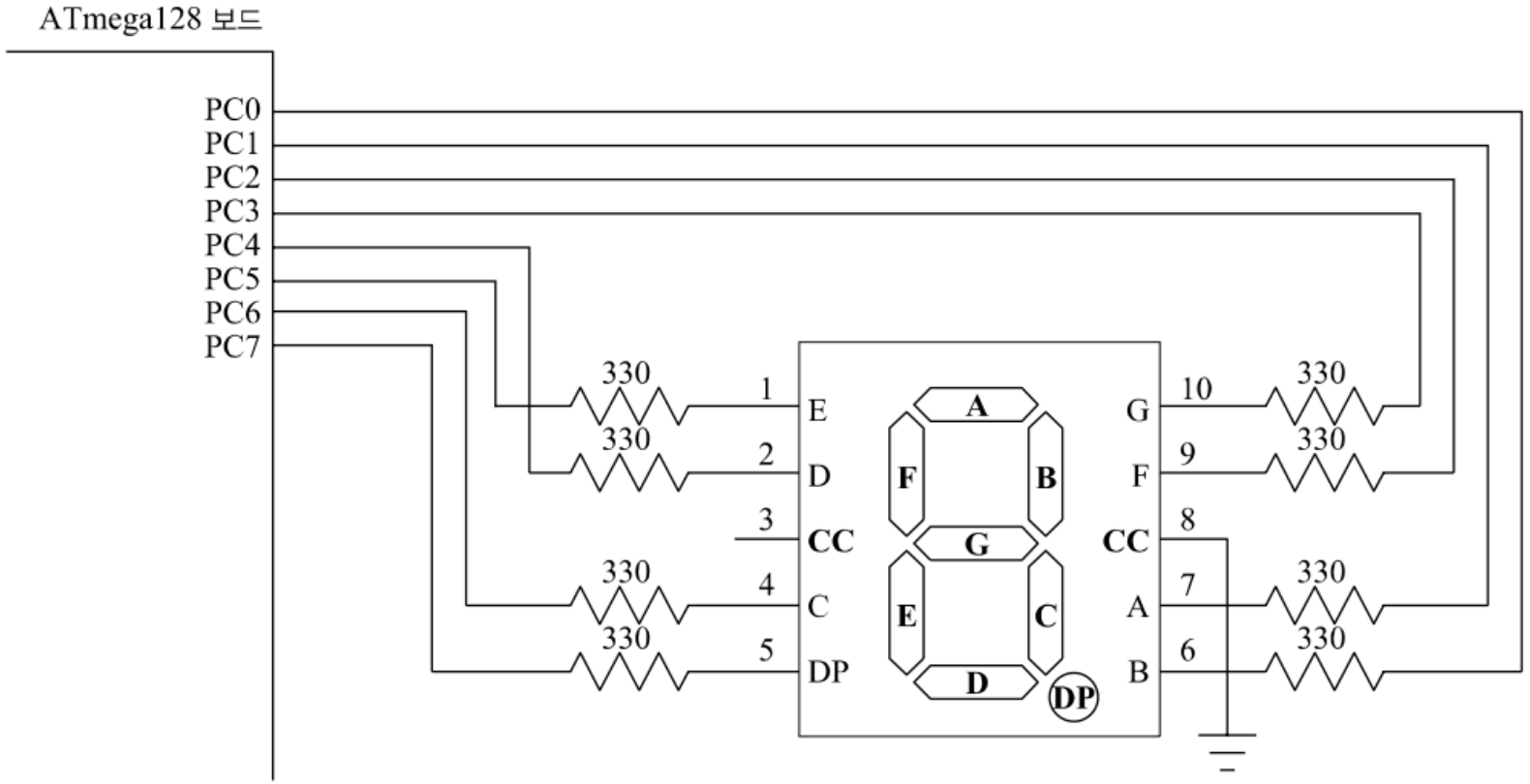
  » 스위치 변화를 처음 감지 후, 일정시간 동안 변화를 무시함



[스위치 바운싱에 의한 전압 변화]

□ 실험 목적

- 시간 지연 함수를 호출하여 1초마다 숫자가 증가되는 동작을 이해
- 회로



□ 실험 절차

- **5.2절에서 실험한 7-세그먼트 LED 디스플레이를 위한 배열 사용**

- **#include <util/delay.h>를 포함하여 _delay_ms( ) 함수 호출 가능**
  - _delay_ms(1000) 함수 호출은 1000[msec] 즉 1초 시간 지연

- **_delay_ms() 함수에서 F_CPU 매크로 레이블 정의 필요**
  - #define F_CPU 16000000UL에 의한 정의
  - 혹은, 개발 환경에서 Frequency 항목 기록에 의해 정의
  - 둘 다 사용하면 프로그램에서 경고 발생
    - » #define F_CPU 16000000UL 정의가 사용됨

- **sec % 10 연산은 정수 sec을 10으로 나눈 후, 나머지 값**
  - 0 ~ 9 까지만 디스플레이 됨

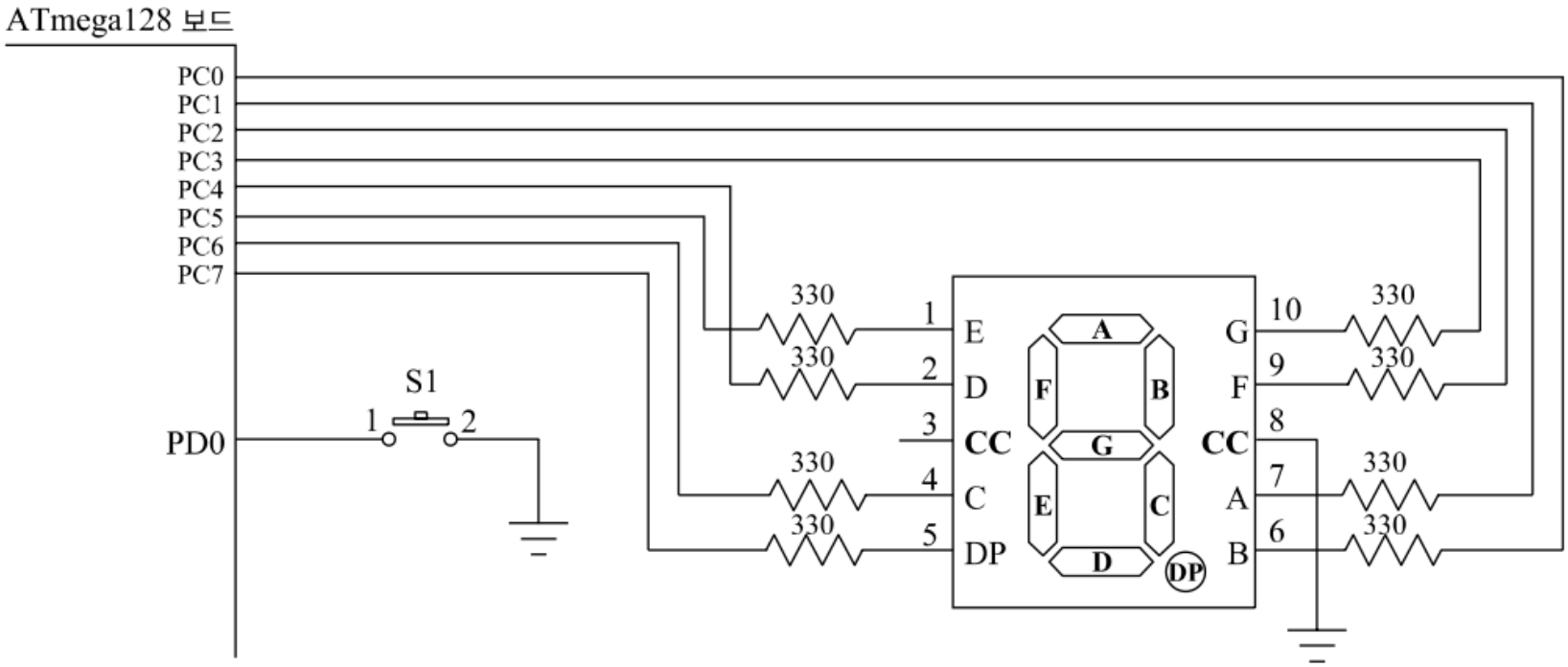
Section 07 시간 지연 함수를 이용한 1초마다 숫자 증가

- _delay_ms( ) 함수를 이용한 1초 지연 프로그램

```
1   #define F_CPU    16000000UL
2
3   #include   <avr/io.h>
4   #include   <util/delay.h>
5
6   unsigned char digit[] = {0x77, 0x41, 0x3B, 0x5B, 0x4D, 0x5E, 0x7C, 0x43, 0x7F, 0x4F};
7
8   void    display_7segled(unsigned char led[], unsigned int number)
9   {   PORTC = led[number];    }
10
11  int main(void)
12  {   int sec;
13
14      DDRC = 0xFF;
15
16      sec = 0;
17      while(1){
18          sec++;
19
20          display_7segled(digit, sec % 10);
21
22          _delay_ms(1000);
23      }
24
25      return 0;
26  }
```

□ 실험 목적

- 스위치 바운싱 현상에 의한 오동작을 확인
- 시간 지연을 이용한 스위치 디바운싱 현상을 확인
- 스위치 디바운싱을 위한 회로



□ 실험 절차

➢ 초기화 과정

```
DDRC = 0xFF;
DDRD = DDRD & ~(1<<PD0);          // PD0를 입력 핀으로 설정
PORTD = PORTD | 1<<PD0;           // 입력 핀 PD0를 내부 풀업저항으로 연결

number = 0;                       // 스위치 눌린 횟수
before = RELEASED;                // 초기 스위치는 개방된 상황으로 간주
```

➢ 스위치 눌림 감지는 before 값과 현재의 PD0 핀 논리값 검사

```
if( before == RELEASED && !(PIND & 1<<PD0) ){   // 전에 눌리지 않은 상태에서 처음 눌림
    number++;

    before = PRESSED;
}else if( before == PRESSED && (PIND & 1<<PD0) ){ // 전에 눌렸으나 처음으로 떨어짐

    before = RELEASED;
}
```

- 바운싱 현상을 제거하지 못한 프로그램

```
1   #include       <avr/io.h>
2
3   #define PRESSED     1
4   #define RELEASED    0
5
6   unsigned char digit[] = {0x77, 0x41, 0x3B, 0x5B, 0x4D, 0x5E, 0x7C, 0x43, 0x7F, 0x4F};
7
8   void    display_7segled(unsigned char led[], unsigned int number)
9   {   PORTC = led[number];    }
10
11  int main(void)
12  {
13      int number;
14      int before;
15
16      DDRC = 0xFF;
17      DDRD = DDRD & ~(1<<PD0);            // PD0를 입력 핀으로 설정
18      PORTD = PORTD | 1<<PD0;             // 입력 핀 PD0를 내부 풀업저항으로 연결
19
20      number = 0;
21      before = RELEASED;
22
23      while(1){
24          display_7segled(digit, number % 10);
25
26          if( before == RELEASED && !(PIND & 1<<PD0) ){  // 전에 눌리지 않은 상태에서 처음으로 눌림
27              number++;
28              before = PRESSED;
29          }else if( before == PRESSED && (PIND & 1<<PD0) ){  // 전에 눌린 상태에서 처음으로 떨어짐
30              before = RELEASED;
31          }
32      }
33      return 0;
34  }
```

소프트웨어적인 디바운싱

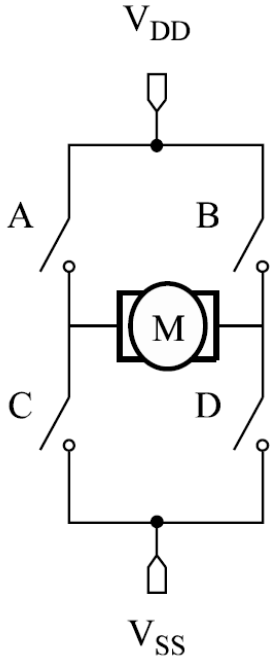
- 스위치 변화 후, 일정기간 강제 시간지연을 갖게 함

```
while(1){
    display_7segled(digit, number % 10);

    if( before == RELEASED && !(PIND & 1<<PD0) ){   // 전에 눌리지 않은 상태에서 처음 눌림
        number++;
        _delay_ms(DEBOUNCE_MS);
        before = PRESSED;
    }else if( before == PRESSED && (PIND & 1<<PD0) ){ // 전에 눌렸으나 처음으로 떨어짐
        _delay_ms(DEBOUNCE_MS);
        before = RELEASED;
    }
}
```

□ DC 모터의 정회전과 역회전을 위한 H-브리지 회로

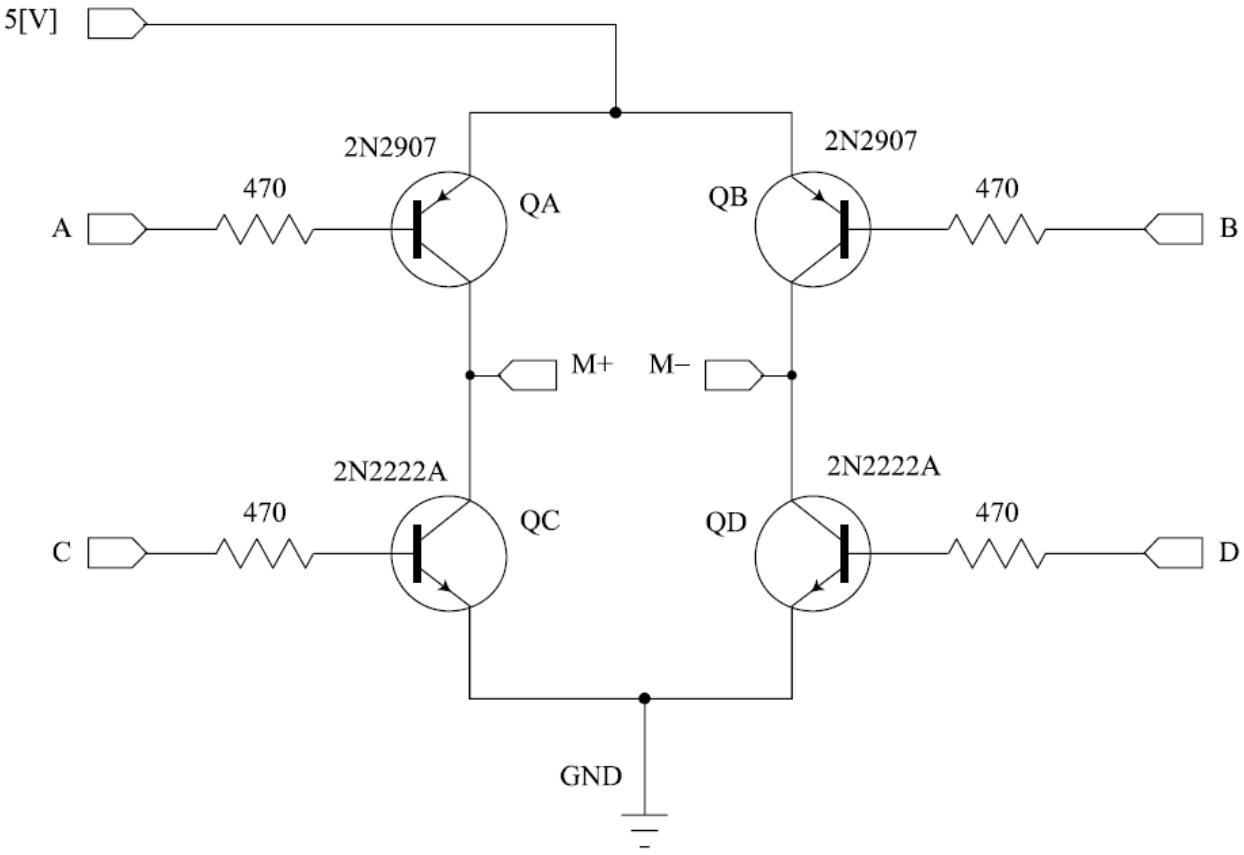
➢ H-브리지 회로

- 모터의 전류 흐름 : 좌 → 우
  - » A, D 단락
  - » B, C 개방
- 모터의 전류 흐름 : 우 → 좌
  - » A, D 개방
  - » B, C 단락
- 모터 전류 차단
  - » A, B, C, D 모두 개방
  - » (혹은) A, B 개방 혹은 C, D 개방



[H-브리지 회로와 스위치]

□ 소형 모터 구동을 위한 H-브리지 회로



[H-브리지 회로를 이용한 DC 모터 정·역방향 제어 회로]

➢ 베이스 저항값이 470[Ω]일 때

$$I_B = \frac{V_{CC} - V_{BE(sat)}}{R_B} = \frac{5 - 1.0}{470} = 8.5[\text{mA}]$$

- 직류 전류 이득을 50이라 가정해도
  - » 450[mA] 까지 구동할 수 있음

➢ DC 모터 양단에 걸리는 전압

$$V_{\text{모터양단}} = 5[\text{V}] - V_{CE(sat),\ 2N2907A} - V_{CE(sat)2N2222A} = 5 - 0.3 - 0.3 = 4.6[\text{V}]$$

- 4.6[V], 450[mA] 범위의 소형 또는 마이크로 DC 모터를 구동

➢ DC 모터 정회전, 역회전을 위한 디지털 전압레벨

- [그림 5-21]회로의 DC 모터 회전을 위한 A, B, C, D 입력 신호

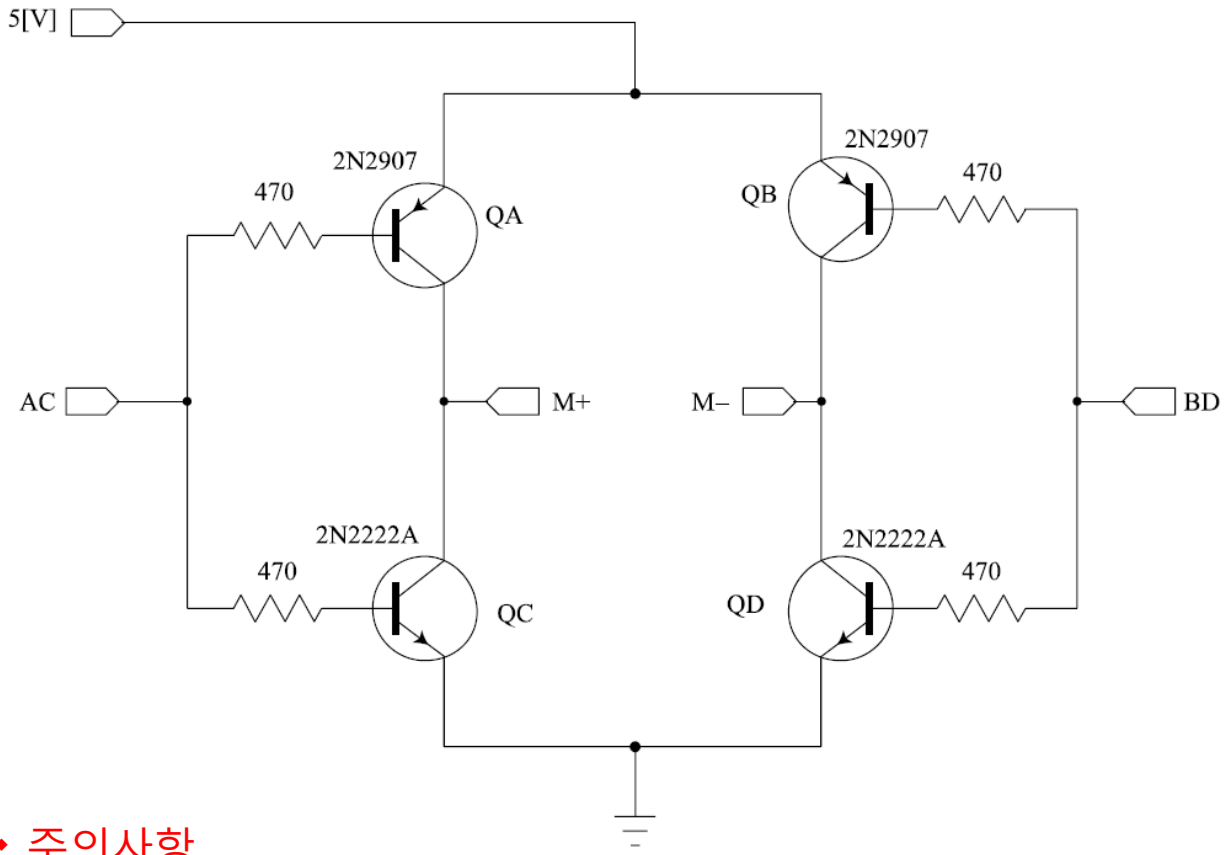
회전 방향	트랜지스터 스위칭				디지털 전압레벨			
	QA	QB	QC	QD	A	B	C	D
정방향	ON	OFF	OFF	ON	LOW	HIGH	LOW	HIGH
역방향	OFF	ON	ON	OFF	HIGH	LOW	HIGH	LOW
정지	OFF	OFF	–	–	HIGH	HIGH	–	–
	–	–	OFF	OFF	–	–	LOW	LOW

➢ AC, BD 디지털 전압레벨을 동일하게 하면 간소화된 H-브리지 회로

- [그림 5-22]회로의 DC 모터 회전을 위한 AC, BD 입력 신호

회전 방향	트랜지스터 스위칭				디지털 전압레벨	
	QA	QB	QC	QD	AC	BD
정방향	ON	OFF	OFF	ON	LOW	HIGH
역방향	OFF	ON	ON	OFF	HIGH	LOW
정지	OFF	OFF	ON	ON	HIGH	HIGH
	ON	ON	OFF	OFF	LOW	LOW

➢ 소형 모터를 위한 간소화된 회로



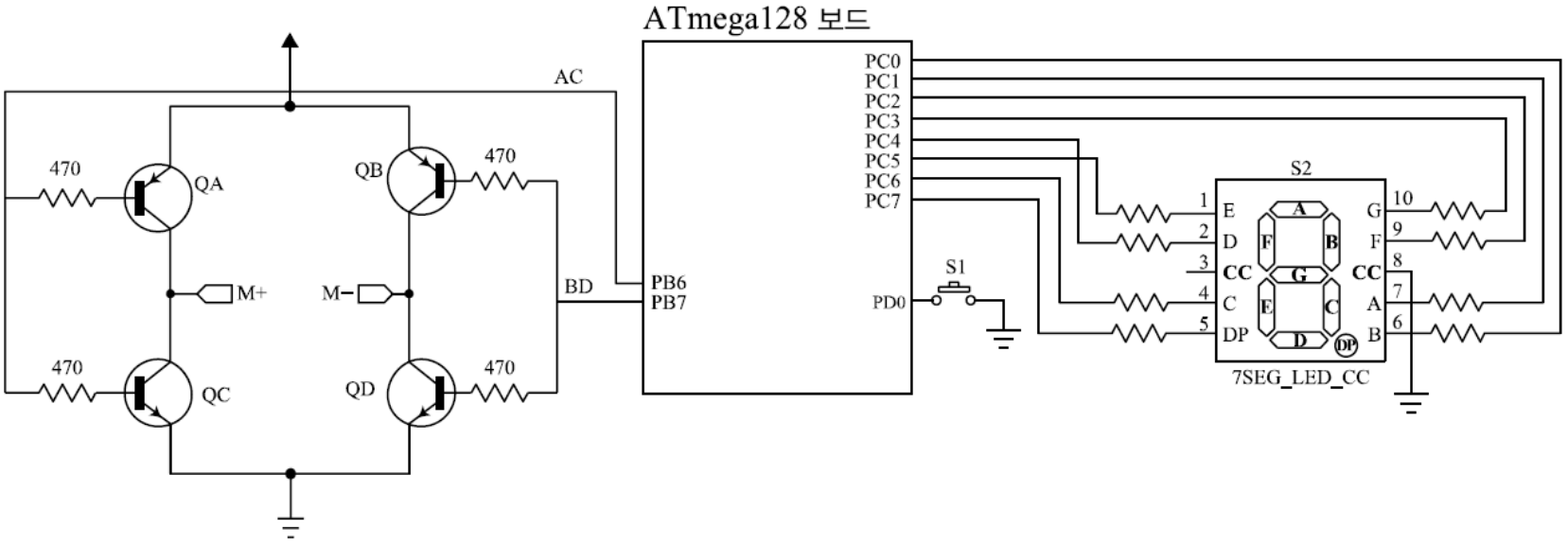
❖ 주의사항

전원을 연결하고 AC 혹은 BD 신호를 개방시키면 안 됨

→ 트랜지스터가 모두 ON 되어 쇼트된다. AC, BD는 HIGH 혹은 LOW가 항상 인가되어야 함

□ 실험 목적

- 트랜지스터를 이용한 H-브리지 회로 동작을 구현하여 확인
- 시간 지연 함수를 이용한 스위칭 디바운싱 기능
- H-브리지 회로의 정·역회전을 디바운싱 스위치로 제어하는 프로그래밍
- 한 개의 스위치로 정지, 정회전, 역회전 동작
- DC 모터 정·역회전, 공통 캐소드 7-세그먼트 LED 디스플레이 결합 회로([그림 5-21])



□ 실험 절차

➢ 모터 연결 신호 레이블

```
#define AC   PB6
#define BD   PB7
#define MOTOR_PORT  PORTB
#define MOTOR_DDR   DDRB
```

➢ 모터 동작 레이블

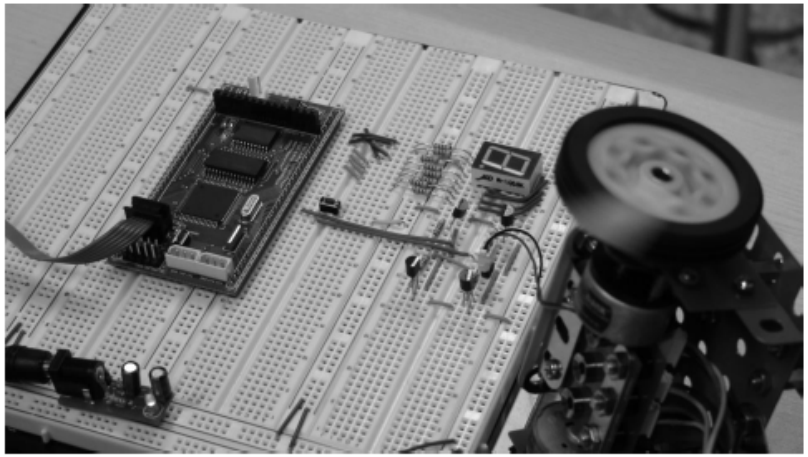
```
#define CLAER   (MOTOR_PORT & 0x3F)

#define MOTOR_STOP  (1<<AC | 1<<BD)
#define MOTOR_FORE  (1<<BD)
#define MOTOR_BACK  (1<<AC)
```

➢ 모터 정회전, 역회전, 정지 명령

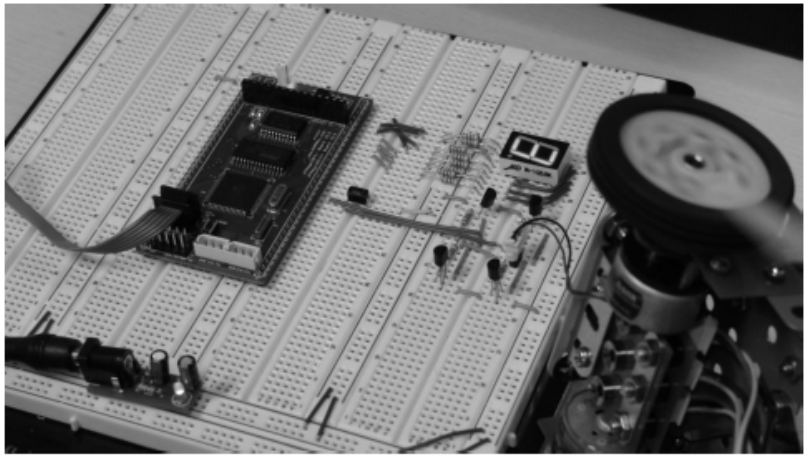
```
if( number % 10 == 1)       // number에 대한 1의 자리가 1일 때
    MOTOR_PORT = (MOTOR_PORT & CLAER) | MOTOR_FORE;
else if(number % 10 == 3)  // number에 대한 1의 자리가 3일 때
    MOTOR_PORT = (MOTOR_PORT & CLAER) | MOTOR_BACK;
else                                // number에 대한 1의 자리가 기타 값
    MOTOR_PORT = (MOTOR_PORT & CLAER) | MOTOR_STOP;
```

□ 7-세그먼트 LED 값과 모터의 회전

➢ H-브리지 회로를 이용한 DC 모터의 회전



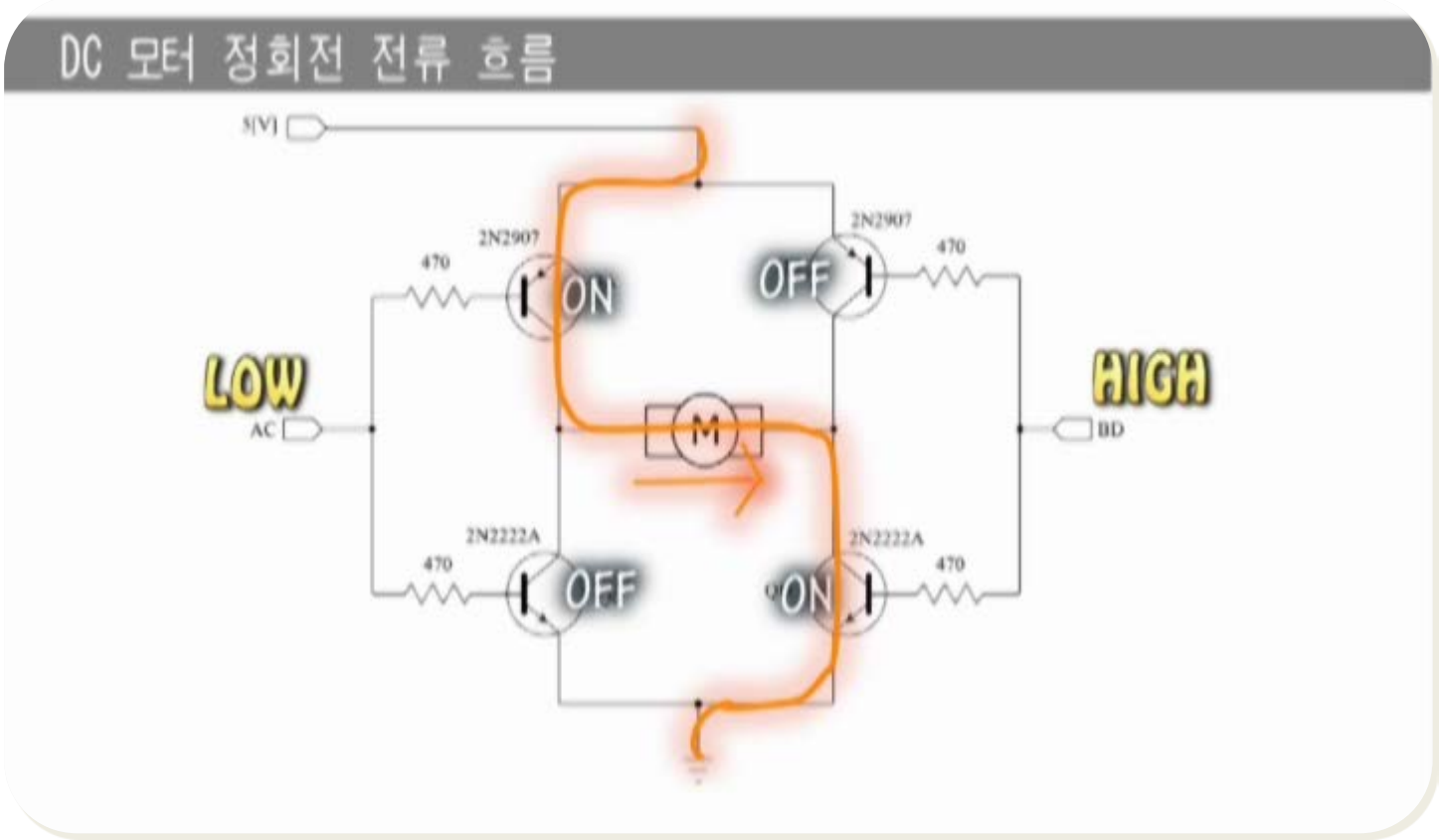
(a) DC 모터 정회전

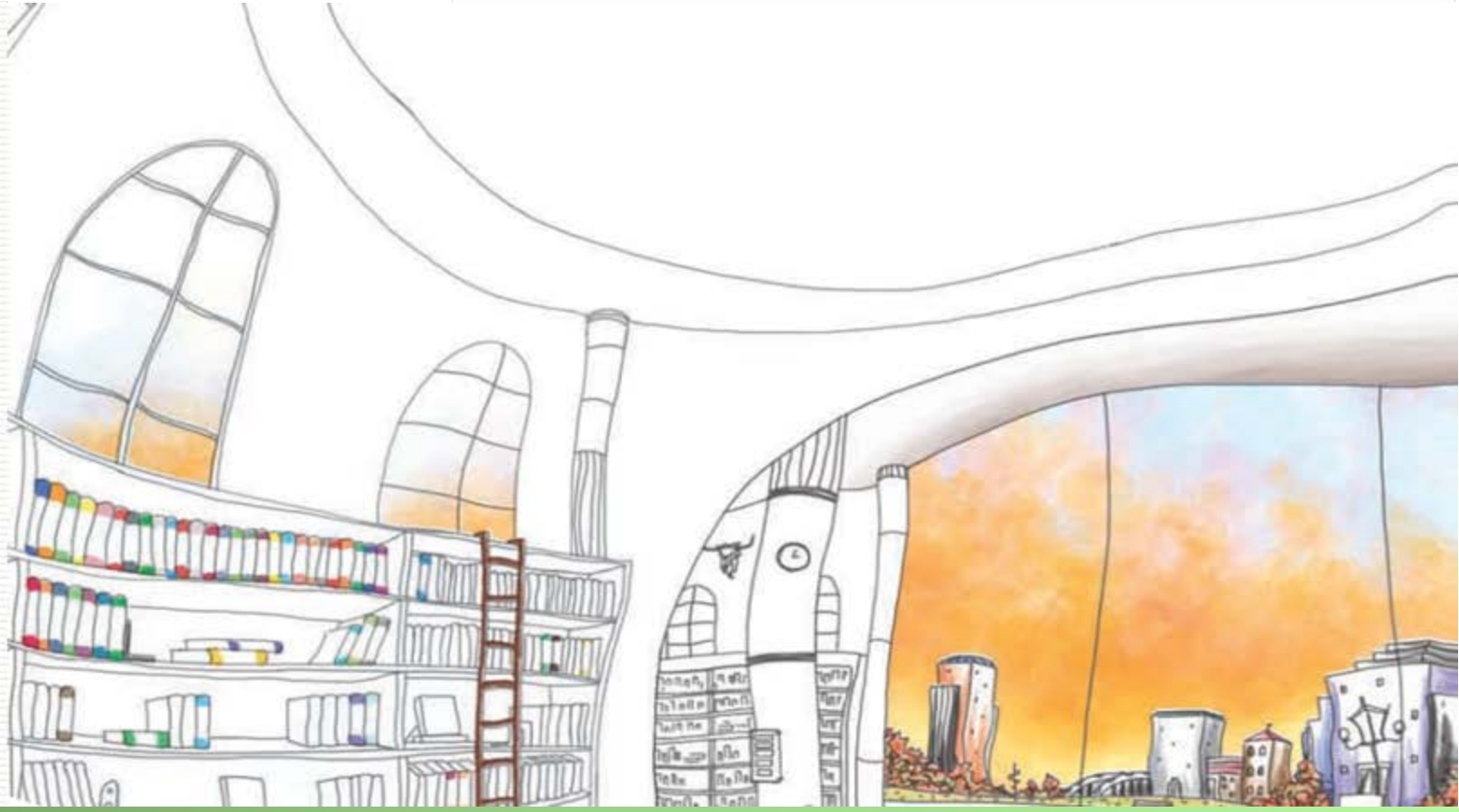


(b) DC 모터 역회전

□ 실험 5-10 H-브리지 회로를 이용한 소형 DC 모터 정•역회전

(video.zip 참고 / 5-10)





Thank You

IT CookBook, 마이크로컨트롤러 AVR ATmega128

한빛미디어 Hanbit Media, Inc.