Lecture 02~03

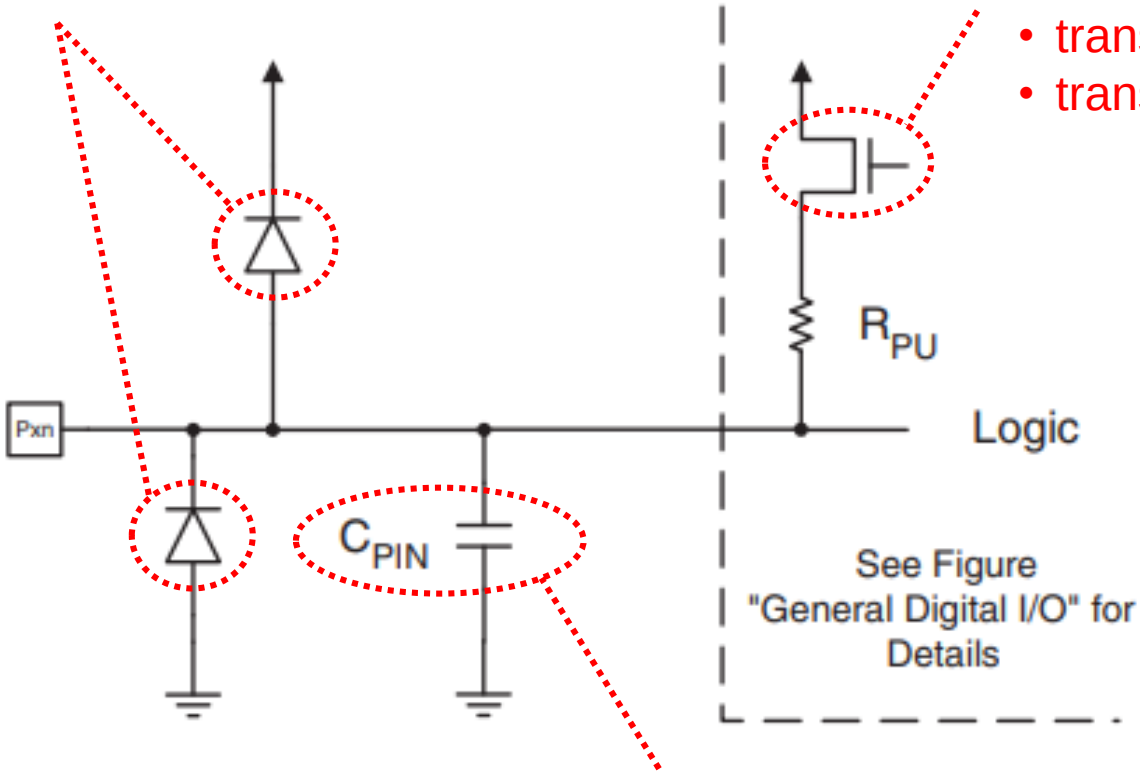
I/O 포트 제어

구조 및 기능

포트 보호

출력 모드

- transistor 켜짐: high(1) 상태
- transistor 꺼짐: low(0) 상태



입력 모드: 외부의 전압레벨을 읽어 들임

ATmega128 포트

- 종 7개의 포트
  - 8-bit 포트: A, B, C, D, E, F
  - 5-bit 포트: G
- 포트 제어 관련 register

Register	기능	초기값
DDRx	포트 모드 설정 (1: 출력, 0: 입력)	0000_0000
PORTx	출력 전압 레벨 설정	0000_0000
PINx	외부의 전압 레벨을 일어 들임	high impedance (HiZ)

ATmega128 포트

- 종 64핀
  - I/O 포트 53핀
  - 나머지 11핀

→ I/O 핀과 겹쳐 있게 설계됨

포트	겹쳐 있는 기능
A, C	외부 메모리 연결
B	SPI 통신, 타이머/카운터의 PWM 출력 핀
D	외부 인터럽트 입력 핀, 타이머/카운터 입출력 핀
E	ISP(In System Programmer), EEPROM 프로그래밍
F	ADC, JTAG 기능
G	RTC(Real Time Clock) 연결, 외부 메모리 인터럽트

DDRx

- 데이터 방향 설정
  - Tri-state 버퍼 통해 입출력 방향 설정 가능
  - 설정 값
    - 해당 비트를 0로 설정하면 해당 핀은 입력 핀이 됨
    - 해당 비트를 1로 설정하면 해당 핀은 출력 핀이 됨

비트	7	6	5	4	3	2	1	0
	DDRx7	DDRx6	DDRx5	DDRx4	DDRx3	DDRx2	DDRx1	DDRx0
읽기/쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
초기값	0	0	0	0	0	0	0	0

PORTx

- 출력 전압레벨 설정
  - PORTx 설정 시 먼저 DDRx로 포트를 출력으로 설정해 야 함
  - 설정 값
    - 해당 비트를 0로 설정하면 해당 핀은 전압레벨 0(GND)이 됨
    - 해당 비트를 1로 설정하면 해당 핀은 전압레벨 1(VCC)이 됨

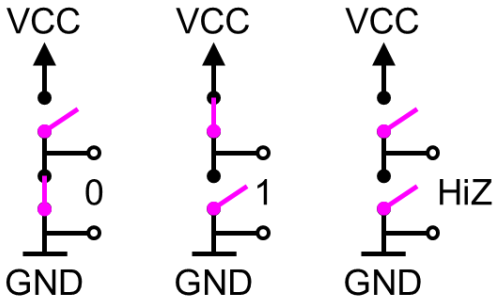
비트	7	6	5	4	3	2	1	0
	PORTx7	PORTx6	PORTx5	PORTx4	PORTx3	PORTx2	PORTx1	PORTx0
읽기/쓰기	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
초기값	0	0	0	0	0	0	0	0

PINx

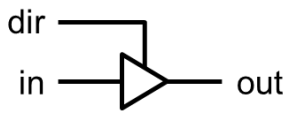
- 외부 전압레벨을 일어 들임
  - PINx 사용 시 먼저 DDRx로 포트를 입력으로 설정해 야 함
  - PINx는 read-only라서 PINx에 데이터를 쓸 수 없음

비트	7	6	5	4	3	2	1	0
	PINx7	PINx6	PINx5	PINx4	PINx3	PINx2	PINx1	PINx0
읽기/쓰기	R	R	R	R	R	R	R	R
초기값	NA	NA	NA	NA	NA	NA	NA	NA

High impedance

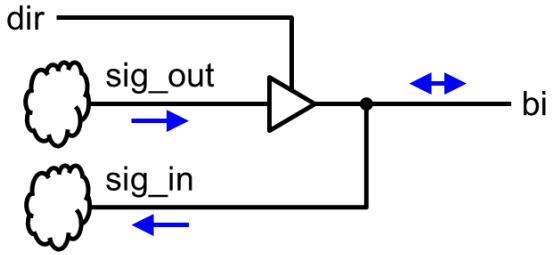


Tri-state 버퍼



dir	out
0	z
1	in

Bi-directional 핀



PINx

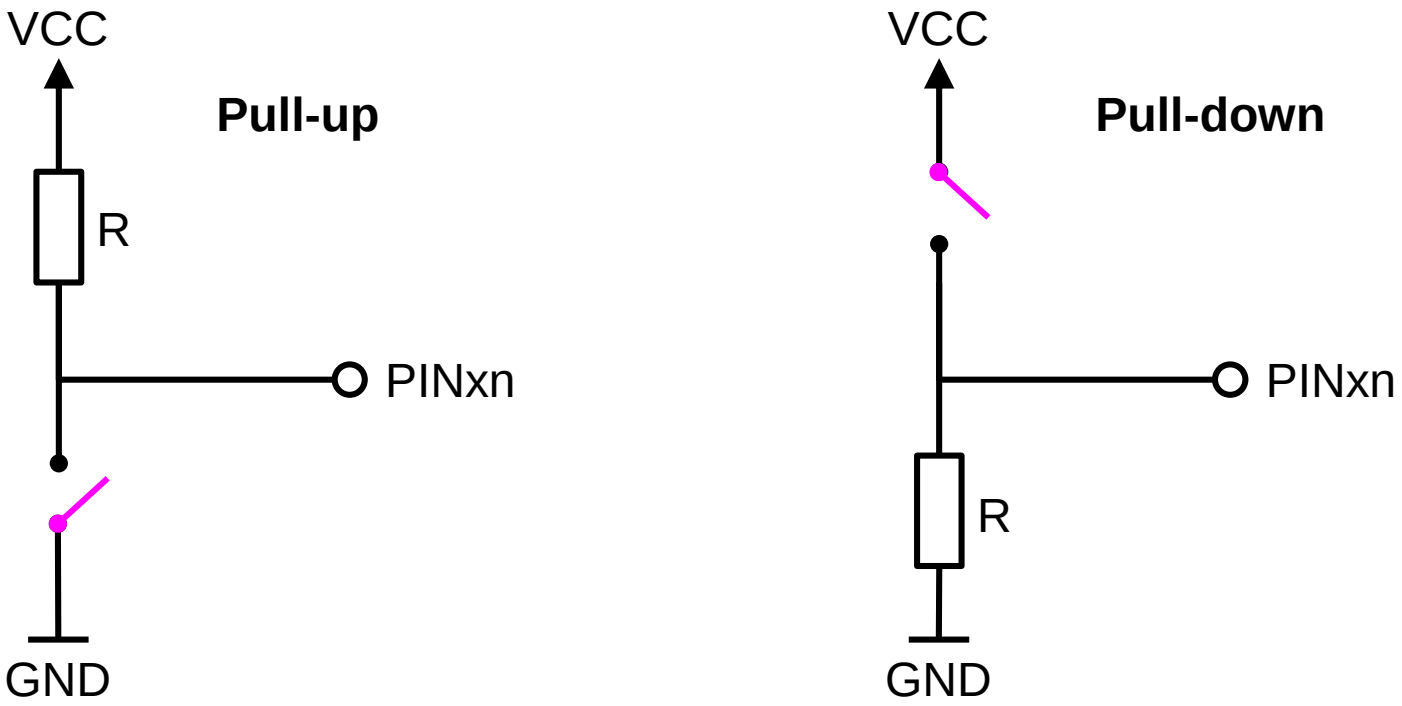
- Pull-up 및 pull-down

x = A, …, G
n = 0, …, 7

DDRxn	PORTxn	PUD (in SFIOR)	I/O	Pull-up	설명
0	0	x	입력	No	HiZ
0	1	0	입력	Yes	외부의 전압변환 없으면 PINxn = 1, 외부의 전압변환 있으면(예, 스위치 누름) PINxn = 0
0	1	1	입력	No	HiZ
1	0	x	출력	No	출력 전압레벨 0
1	1	x	출력	No	출력 전압레벨 1

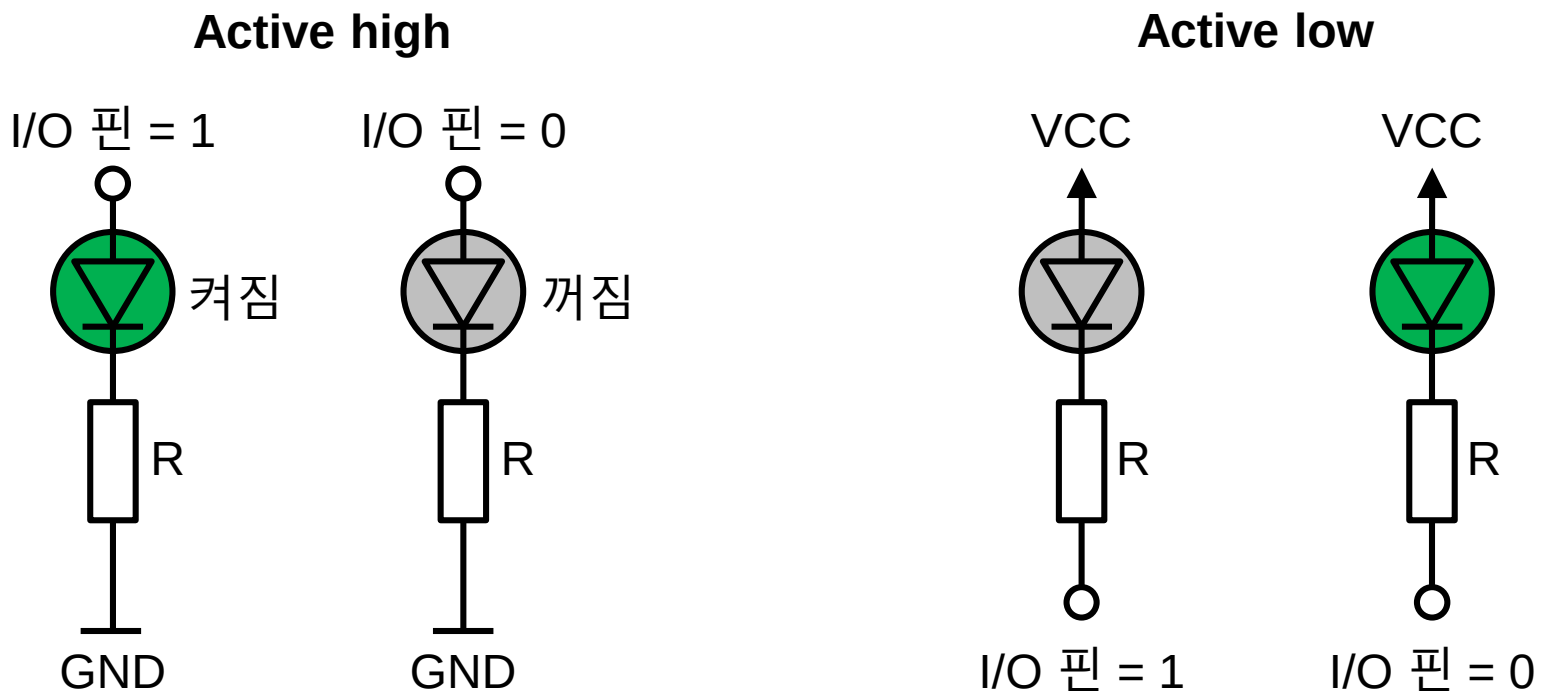
PINx

- Pull-up 및 pull-down



LED 제어

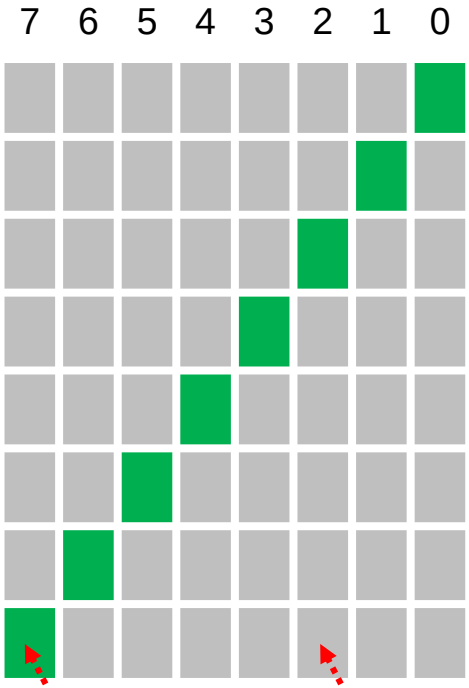
- LED 제어 방법



LED8

- LED의 8개 제어

LED 상태



PORTx 값 (2진수)

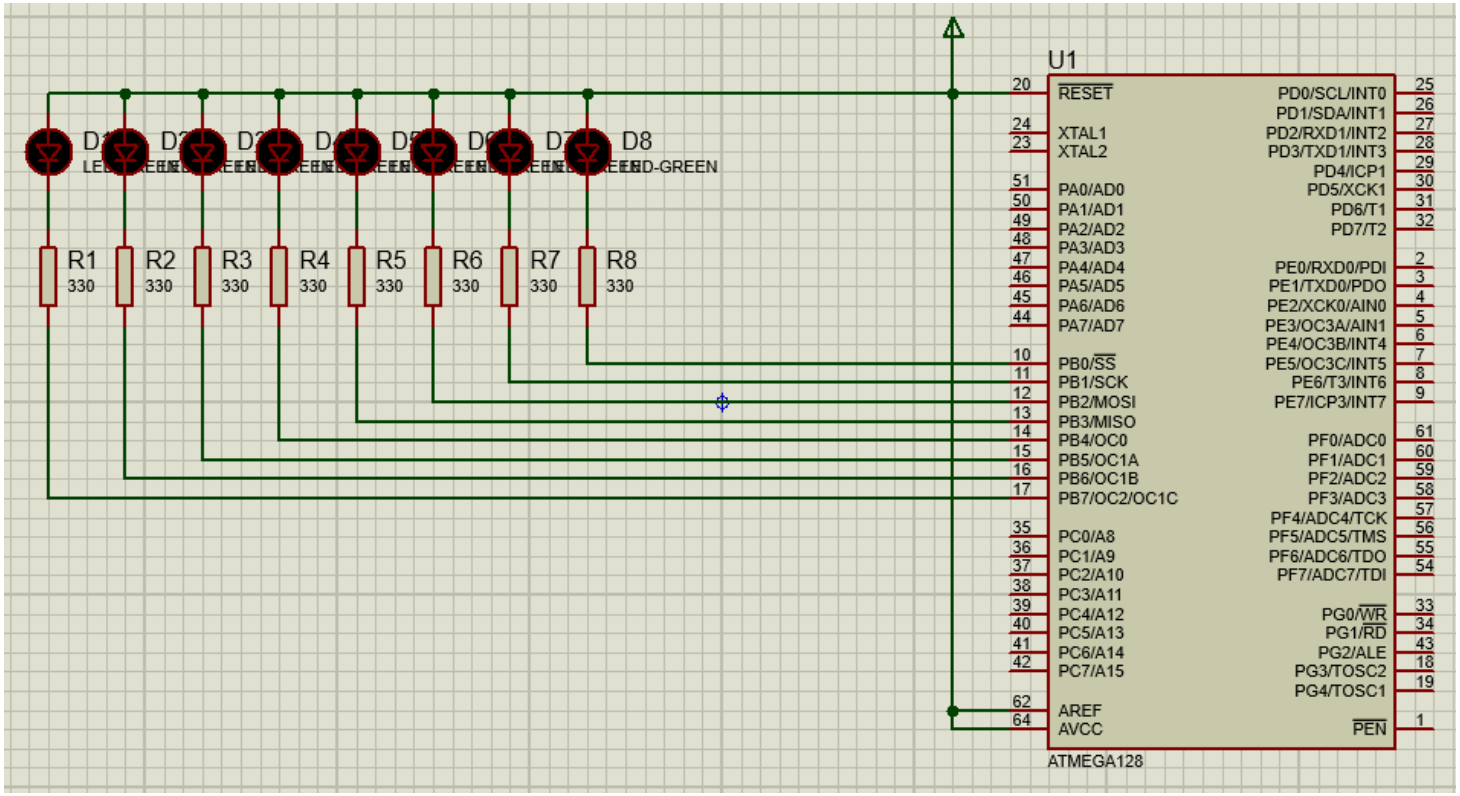
7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	0
1	1	1	1	1	1	0	1
1	1	1	1	1	0	1	1
1	1	1	1	0	1	1	1
1	1	1	0	1	1	1	1
1	1	0	1	1	1	1	1
1	0	1	1	1	1	1	1
0	1	1	1	1	1	1	1

PORTx 값 (16진수)

0xfe
0xfd
0xfb
0xf7
0xef
0xdf
0xbf
0x7f

LED8

- Proteus



LED8

- 코드

```
#include <xc.h>
#include <avr/io.h>
#include <avr/delay.h>

#define DELAY_TIME 1000

void delay(int);

int main(void) {
    ...
}


void delay(int d) {
    int i;
    for (i=0; i<d; i++) _delay_ms(1);
}
```

```
int main(void) {
    DDRB = 0xff; // PORTB 출력 모드
    while(1) {
        PORTB = 0xfe; delay(DELAY_TIME);
        PORTB = 0xfd; delay(DELAY_TIME);
        PORTB = 0xfb; delay(DELAY_TIME);
        PORTB = 0xf7; delay(DELAY_TIME);
        PORTB = 0xef; delay(DELAY_TIME);
        PORTB = 0xdf; delay(DELAY_TIME);
        PORTB = 0xbf; delay(DELAY_TIME);
        PORTB = 0x7f; delay(DELAY_TIME);
    }
}
```

LED8

- 코드 정리

PORTx	1	1	1	1	1	1	1	1	
mask	0	0	0	0	0	0	0	1	xor
PORTx	1	1	1	1	1	1	1	0	
PORTx	1	1	1	1	1	1	1	1	
mask<<1	0	0	0	0	0	0	1	0	xor
PORTx	1	1	1	1	1	1	0	1	
PORTx	1	1	1	1	1	1	1	1	
mask<<2	0	0	0	0	0	1	0	0	xor
PORTx	1	1	1	1	1	0	1	1	

Pseudo 코드

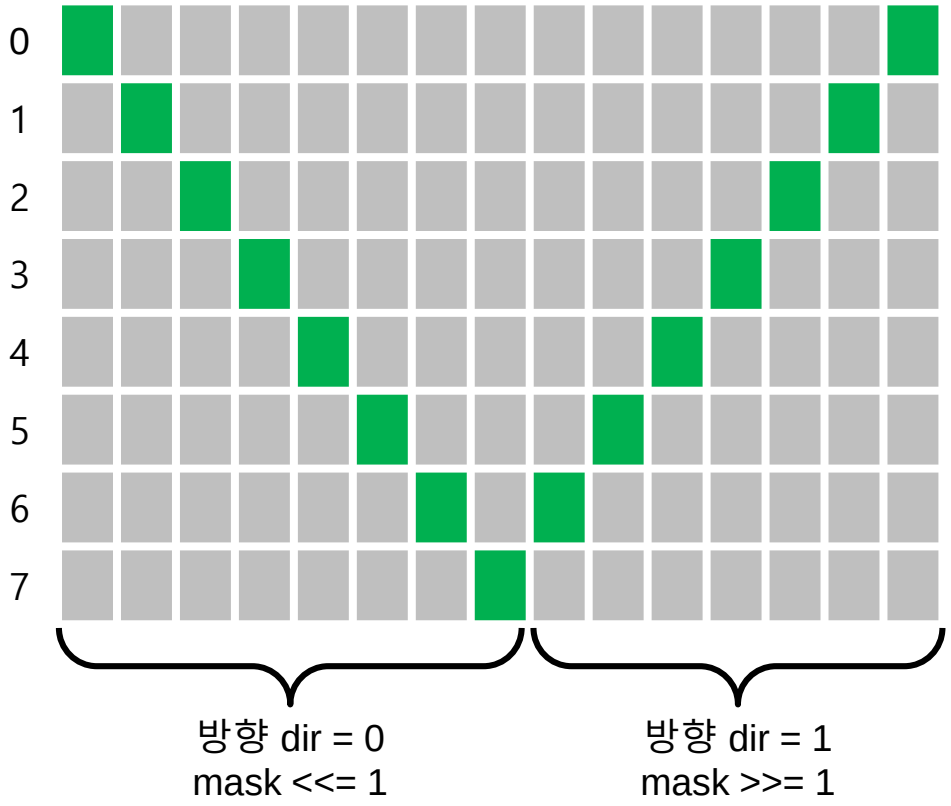
```
mask = 0x01
while (1)
    PORTx = 0xff^mask
    delay
    mask = mask<<1
```

결국 mask = 0
→ PORTx 안 바꿈

```
int main(void) {
    unsigned char mask=0x01;
    DDRB = 0xff; // PORTB 출력 모드
    while(1) {
        if (mask==0x00) mask = 0x01;
        PORTB = 0xff^mask;
        delay(DELAY_TIME);
        mask = mask<<1;
    }
}
```

LED8

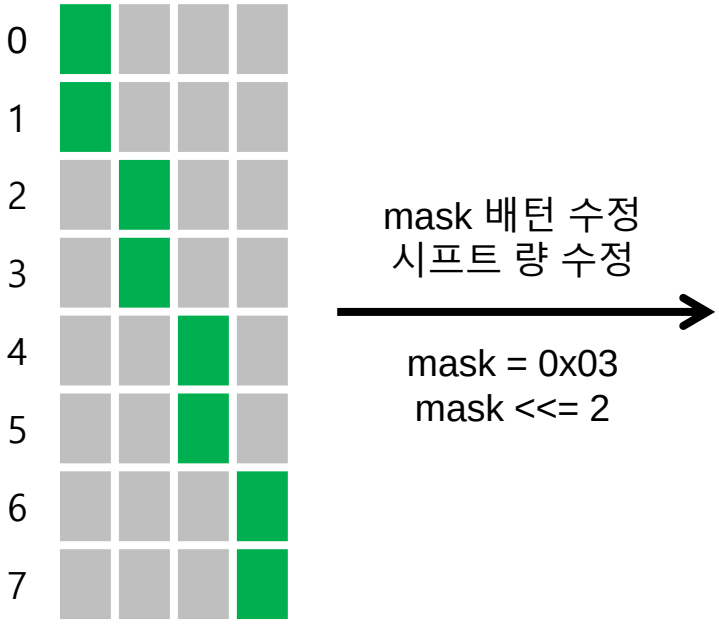
- 다른 LED 켜짐 패턴




```
int main(void) {
    unsigned char mask=0x01, dir=0;
    DDRB = 0xff; // PORTB 출력 모드
    while(1) {
        PORTB = 0xff^mask;
        delay(DELAY_TIME);
        if (dir==0) {
            mask = mask<<1;
            if (mask==0x00) {
                mask = 0x40;
                dir = 1;
            }
        } else {
            mask = mask>>1;
            if (mask==0x00) {
                mask = 0x02;
                dir = 0;
            }
        }
    }
}
```

LED8

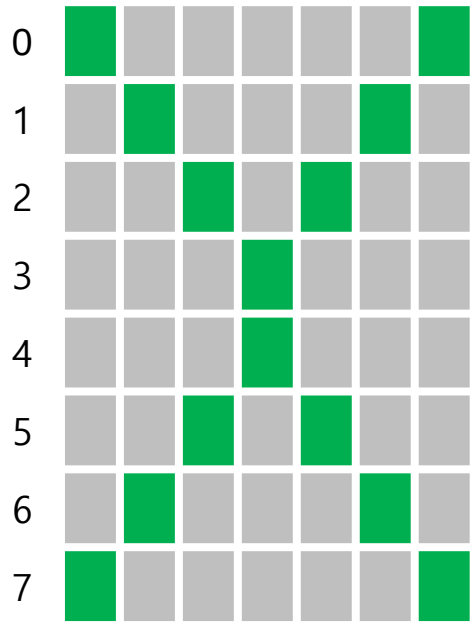
- 다른 LED 켜짐 패턴




```
int main(void) {
    unsigned char mask=0x03;
    DDRB = 0xff; // PORTB 출력 모드
    while(1) {
        if (mask==0x00) mask = 0x03;
        PORTB = 0xff^mask;
        delay(DELAY_TIME);
        mask = mask<<2;
    }
}
```

LED8

- 다른 LED 켜짐 패턴

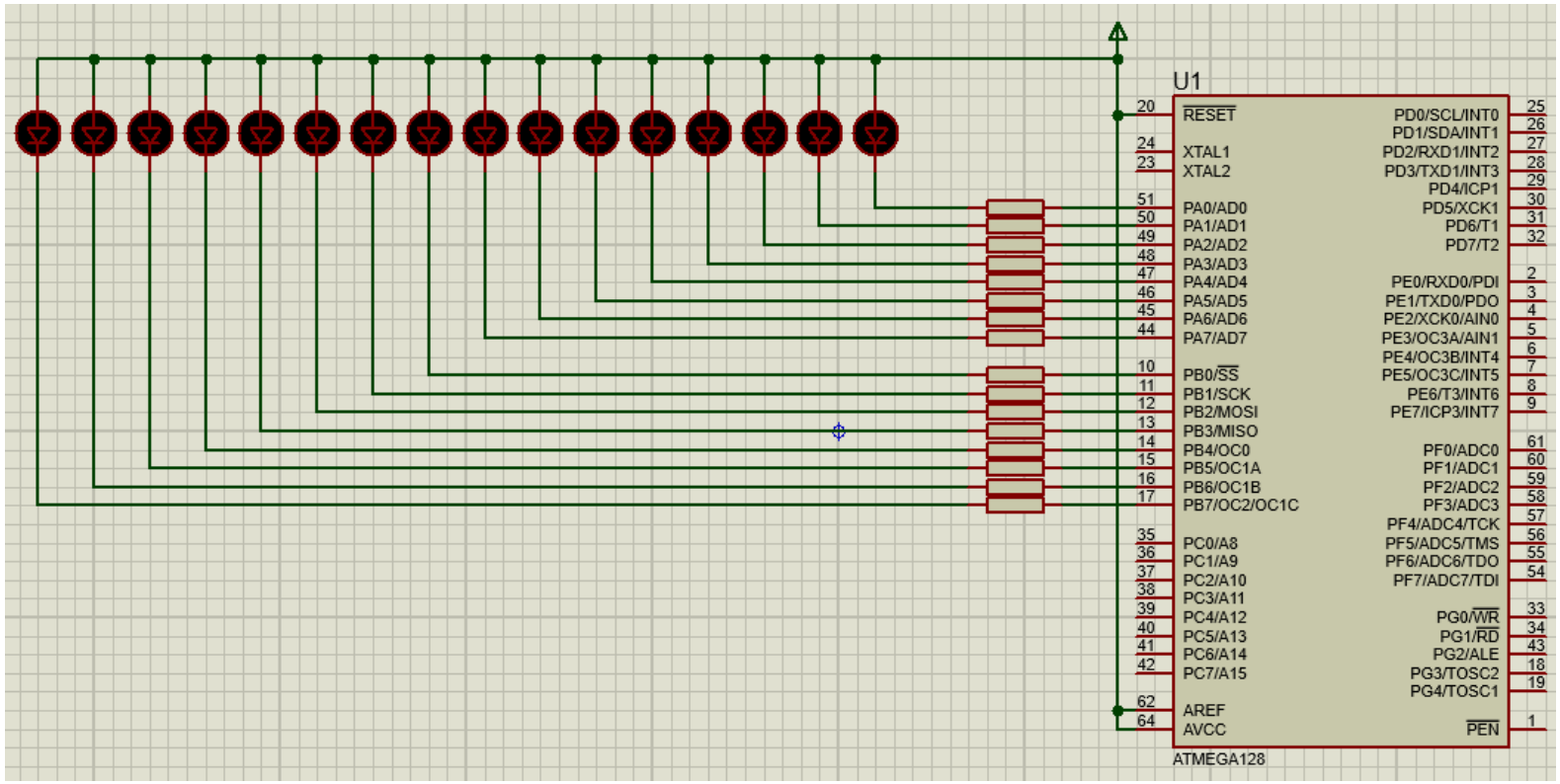


mask 2개 사용
방향 변수 추가

```
int main(void) {
   unsigned char mask1=0x01, mask2=0x80, dir=0;
   DDRB = 0xff; // PORTB 출력 모드
   while(1) {
      PORTB = 0xff^(mask1|mask2);
      delay(DELAY_TIME);
      if (dir==0) {
         mask1 = mask1<<1;
         mask2 = mask2>>1;
         if (mask1==0x10) {
            mask1 = 0x04;
            mask2 = 0x20;
            dir = 1;
         }
      } else {
         mask1 = mask1>>1;
         mask2 = mask2<<1;
         if (mask1==0x00) {
            mask1 = 0x02;
            mask2 = 0x40;
            dir = 0;
         }
      }
   }
}
```

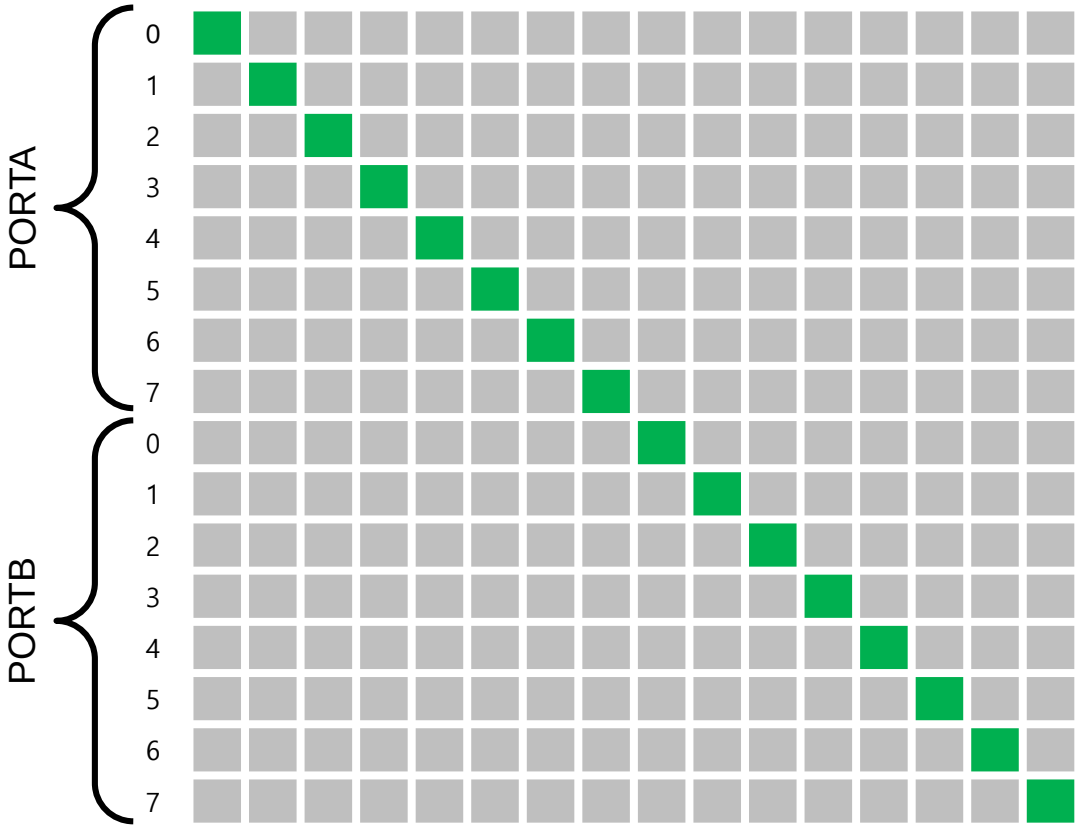
LED16

- Proteus



LED16

- LED의 16개 제어



LED8

8-bit mask와 시프트 사용

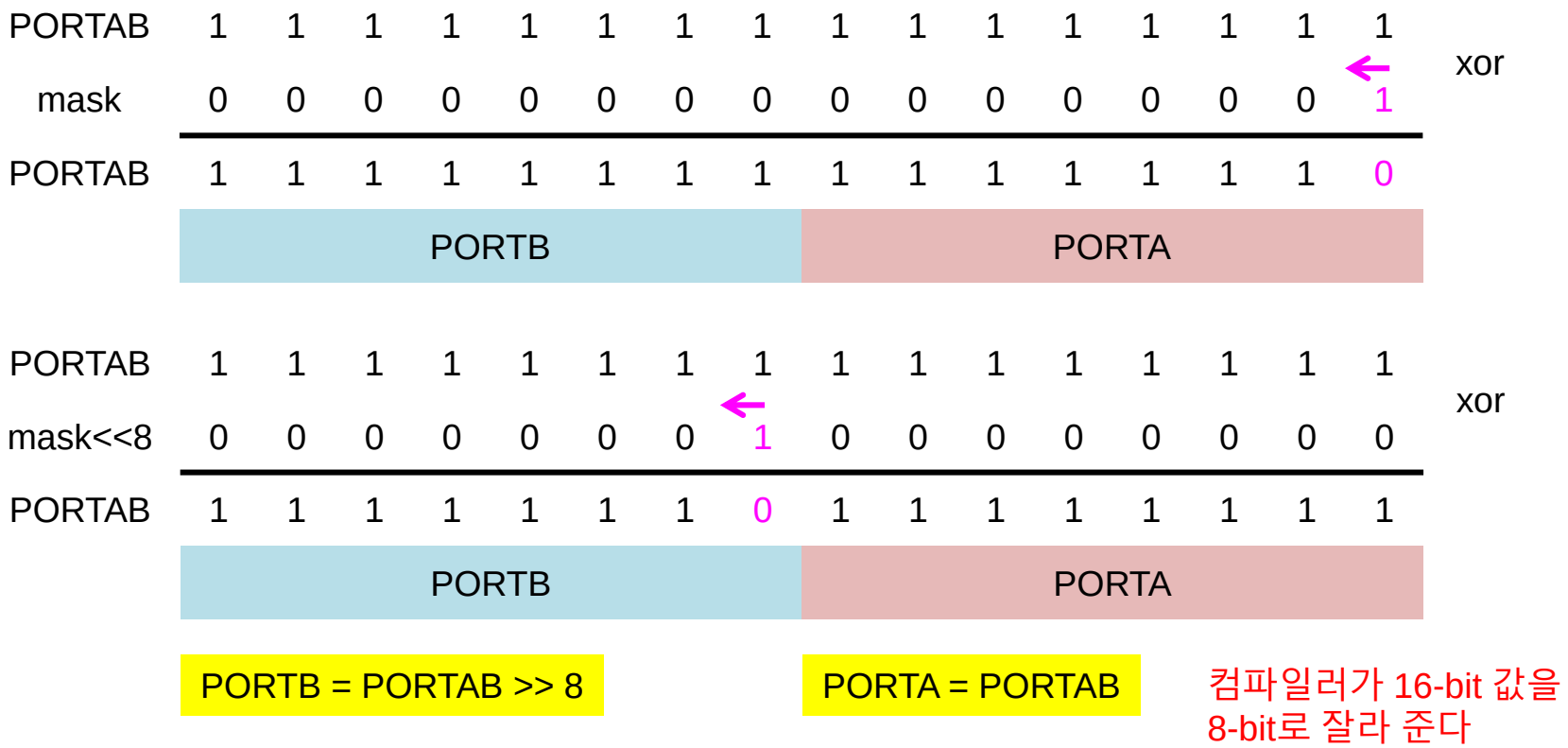
16-bit로 수정

LED16

16-bit mask와 시프트 사용

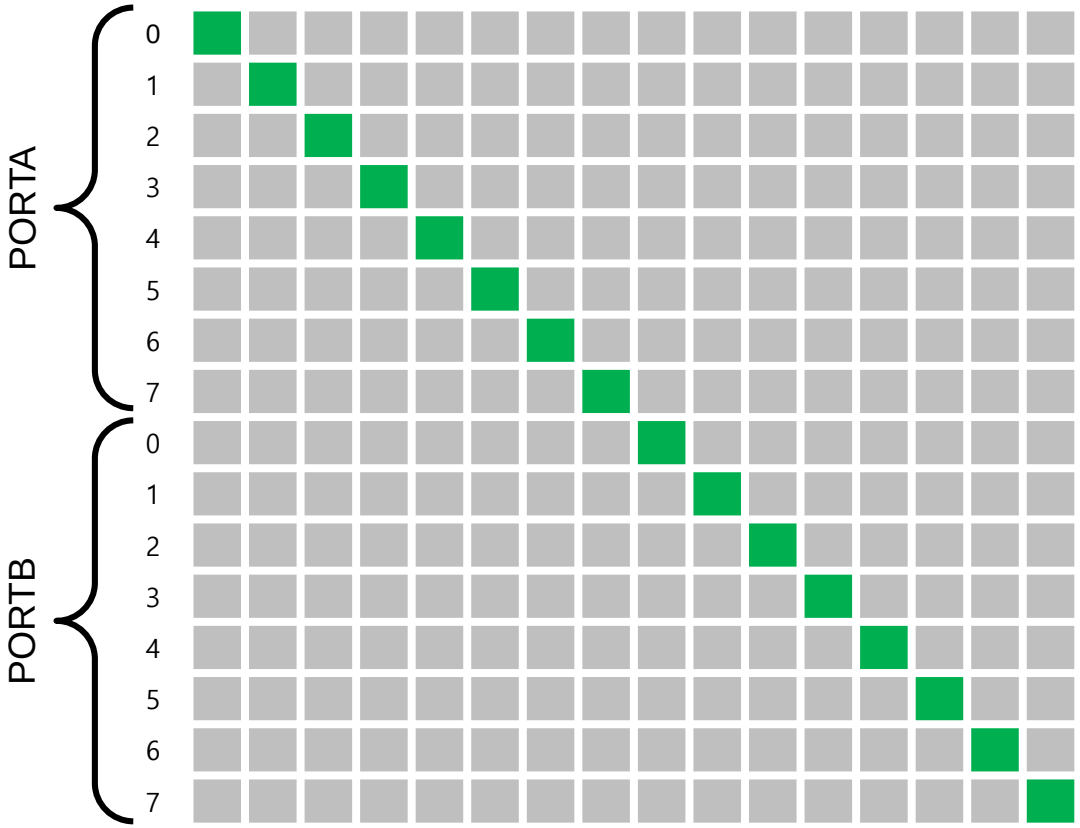
LED16

- LED의 16개 제어



LED16

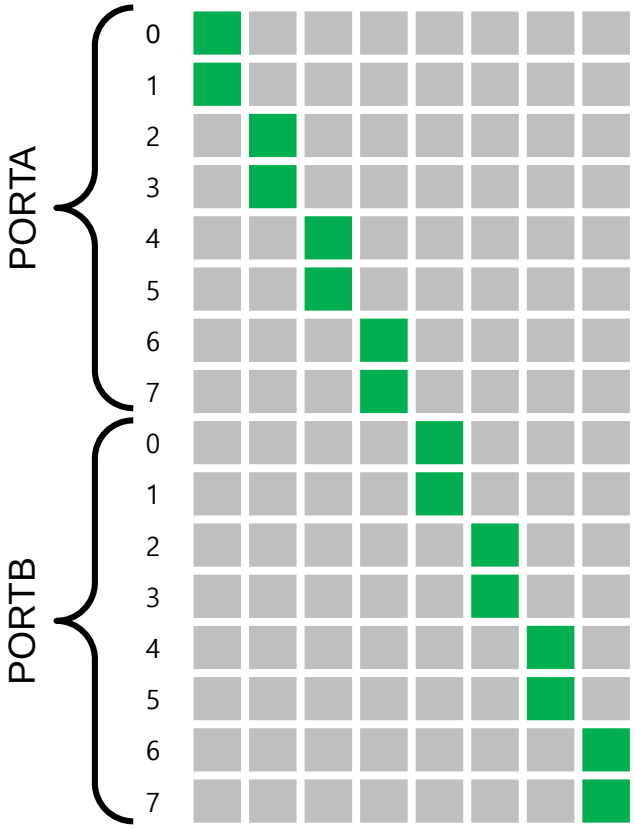
- LED의 16개 제어




```
int main(void) {
    unsigned int mask=0x0001;
    DDRA = 0xff; // PORTA 출력 모드
    DDRB = 0xff; // PORTB 출력 모드
    while(1) {
        if (mask==0x0000) mask = 0x0001;
        PORTA = 0xffff^mask;
        PORTB = (0xffff^mask)>>8;
        delay(DELAY_TIME);
        mask = mask<<1;
    }
}
```

LED16

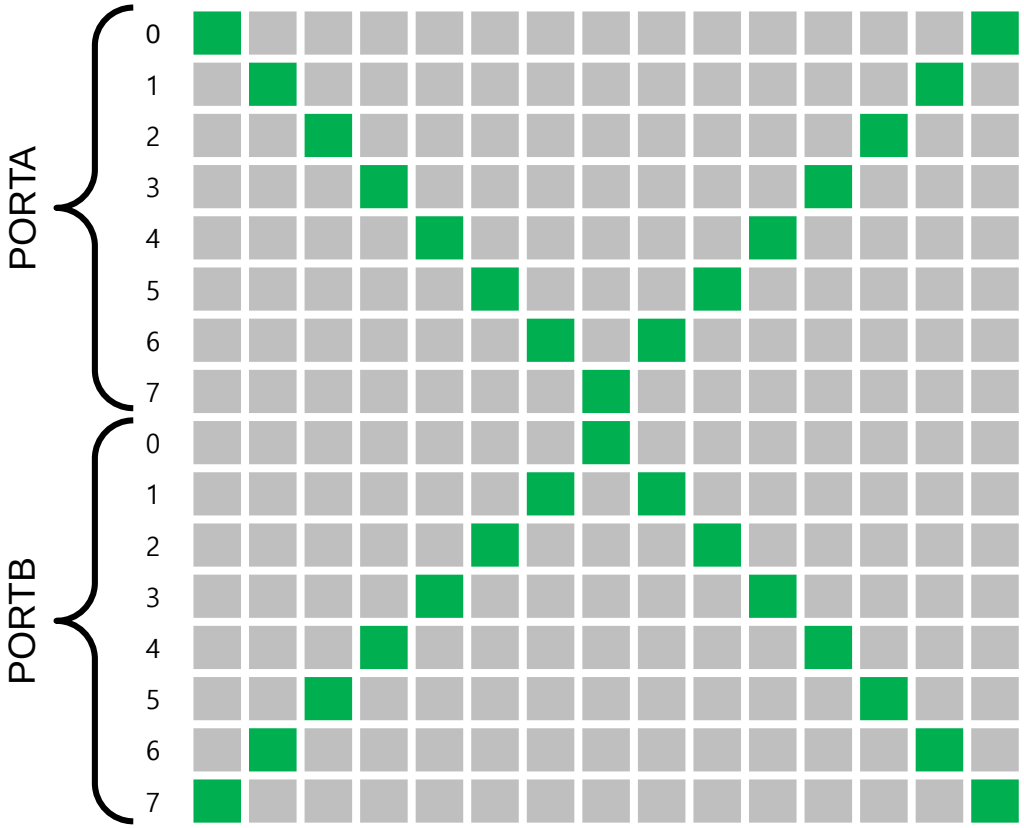
- LED의 16개 제어




```
int main(void) {
    unsigned int mask=0x0003;
    DDRA = 0xff; // PORTA 출력 모드
    DDRB = 0xff; // PORTB 출력 모드
    while(1) {
        if (mask==0x0000) mask = 0x0003;
        PORTA = 0xffff^mask;
        PORTB = (0xffff^mask)>>8;
        delay(DELAY_TIME);
        mask = mask<<2;
    }
}
```

LED16

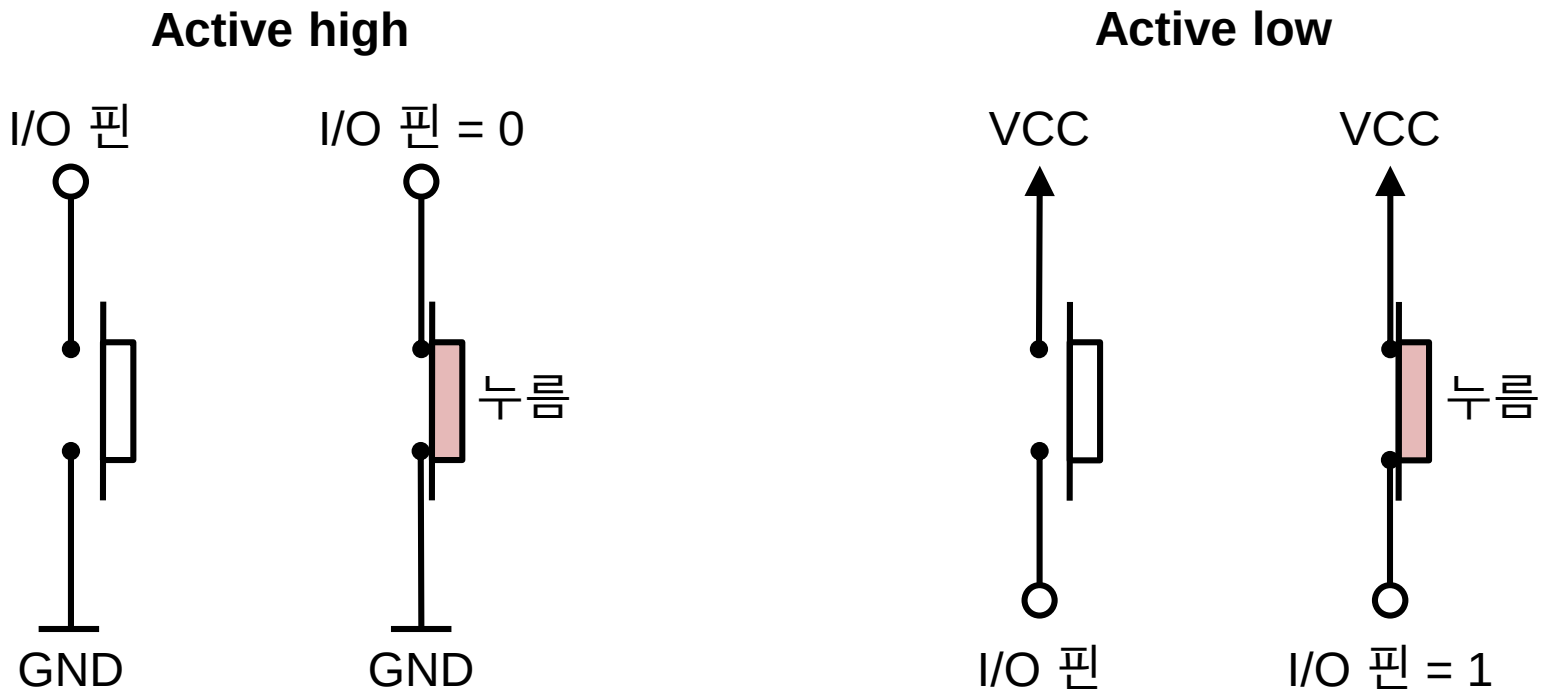
- LED의 16개 제어




```
int main(void) {
    unsigned int mask1=0x0001, mask2=0x8000;
    unsigned char dir=0;
    DDRA = 0xff; // PORTA 출력 모드
    DDRB = 0xff; // PORTB 출력 모드
    while(1) {
        PORTA = 0xffff^(mask1|mask2);
        PORTB = (0xffff^(mask1|mask2))>>8;
        delay(DELAY_TIME);
        if (dir==0) {
            mask1 = mask1<<1;
            mask2 = mask2>>1;
            if (mask1==0x0100) {
                mask1 = 0x0040;
                mask2 = 0x0200;
                dir = 1;
            }
        } else {
            mask1 = mask1>>1;
            mask2 = mask2<<1;
            if (mask1==0x0000) {
                mask1 = 0x0002;
                mask2 = 0x4000;
                dir = 0;
            }
        }
    }
}
```

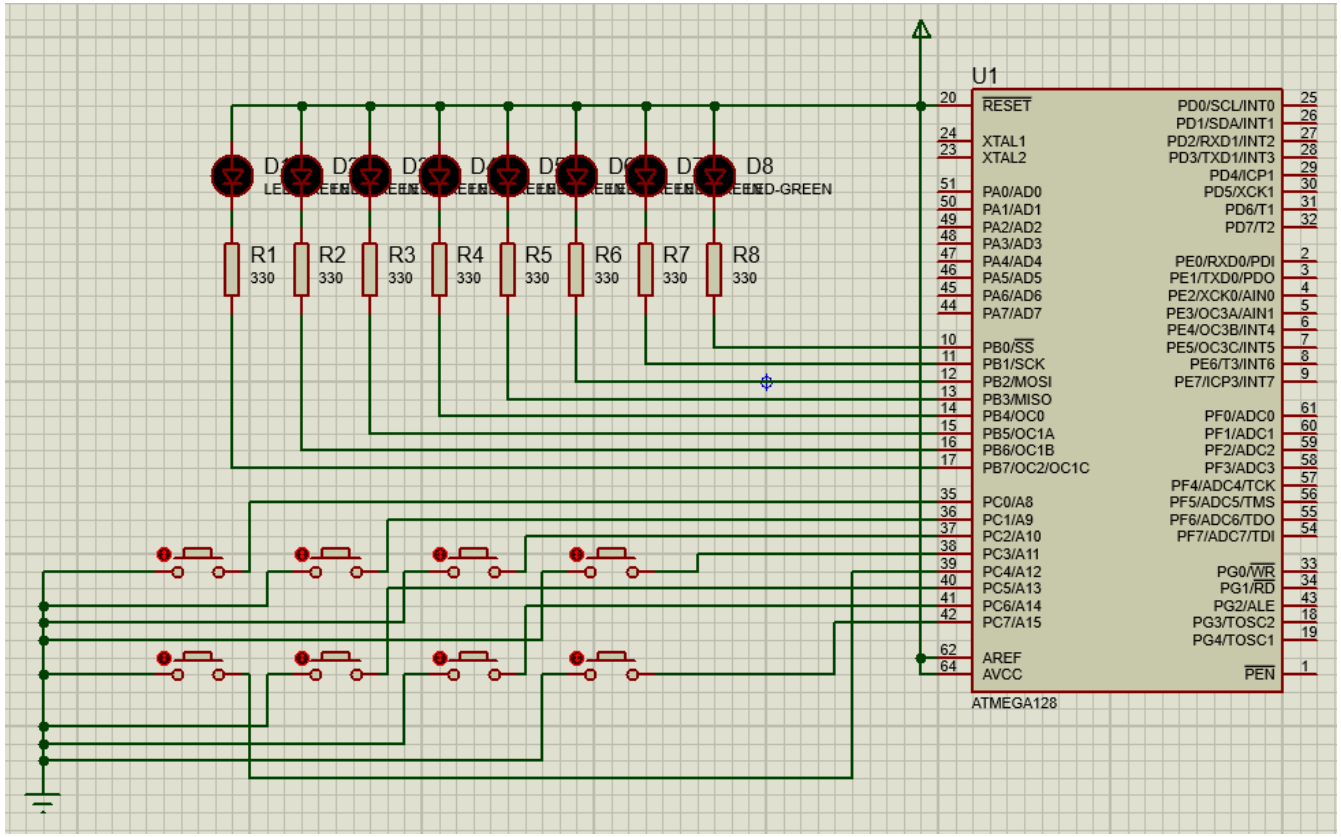
버턴 제어

- 버턴 제어 방법



버턴 제어

- Proteus



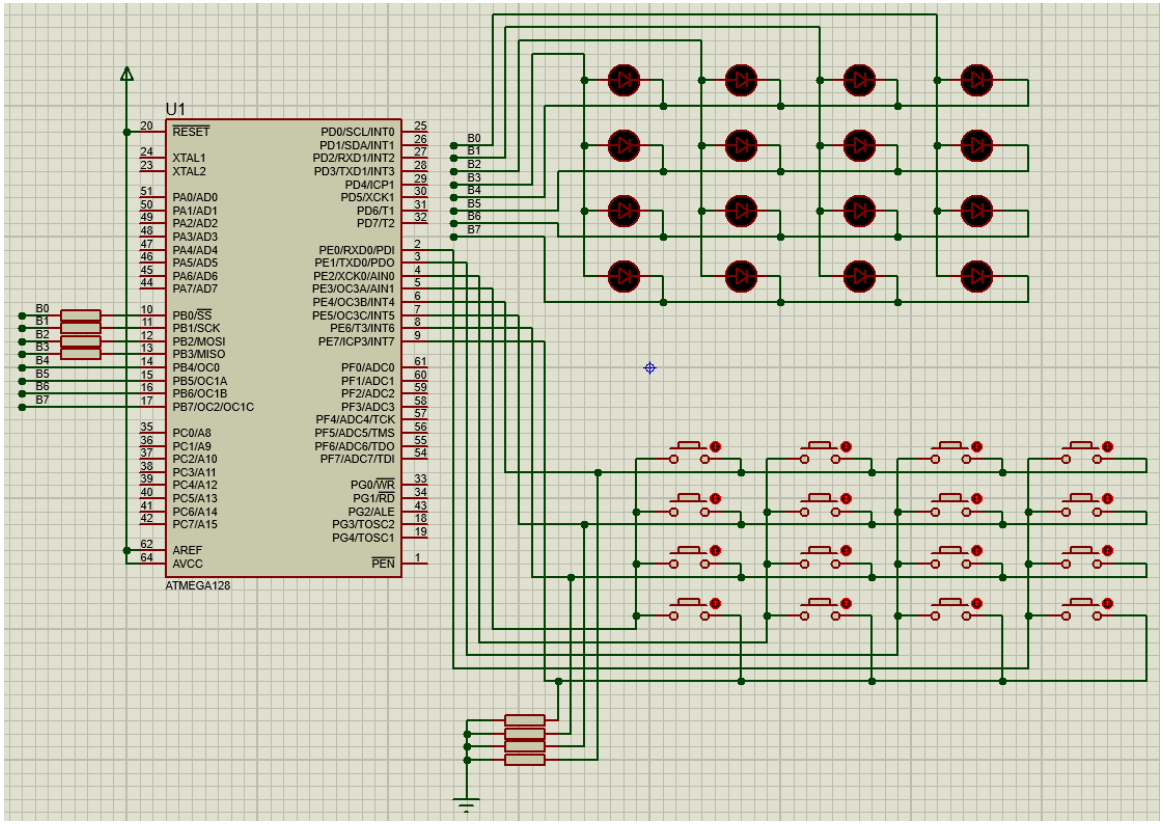
버턴 제어

- 코드

```
int main(void) {
   unsigned char button;
   DDRB = 0xff; // PORTB 출력 모드
   DDRC = 0x00; // PORTC 입력 모드
   PORTC = 0xff; // PORTC pull-up, 버턴 누름 -> PINCn = 0
   while(1) {
      button = PINC;
      PORTB = button;
      delay(DELAY_TIME);
   }
}
```

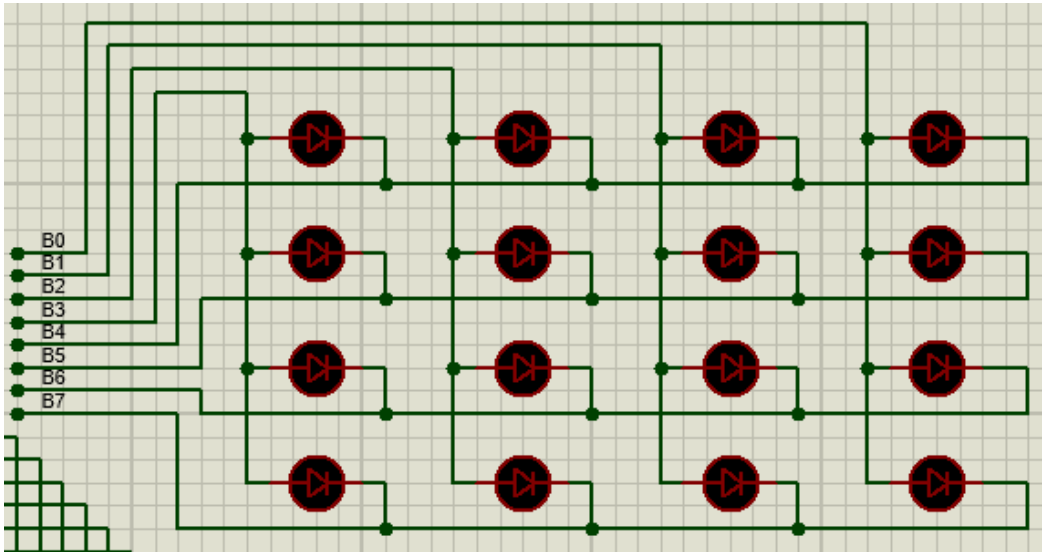
버턴 매트릭스

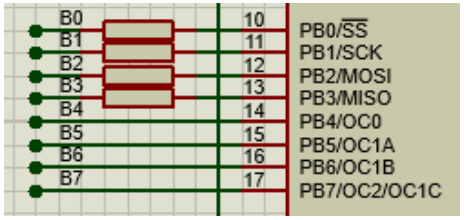
- Proteus

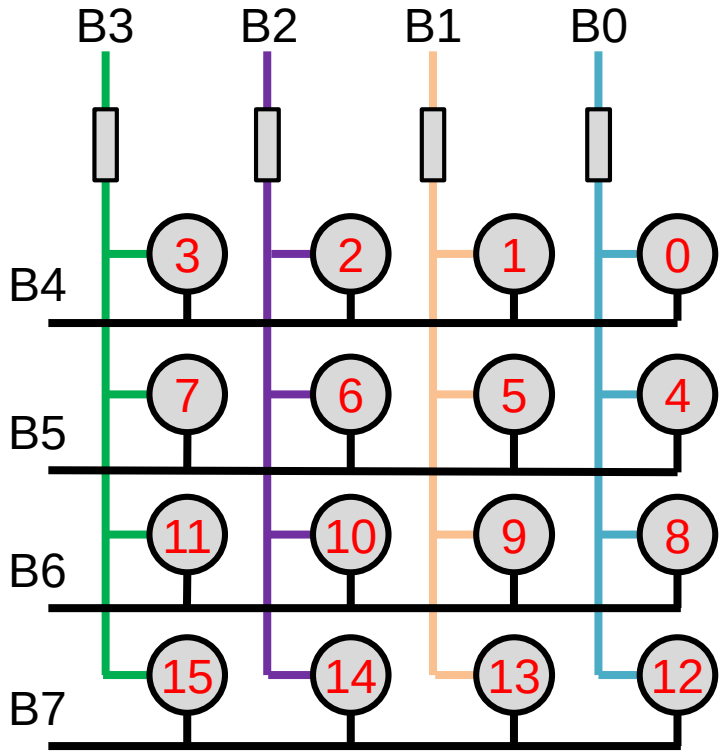


버턴 매트릭스

- 4×4 LED 매트릭스



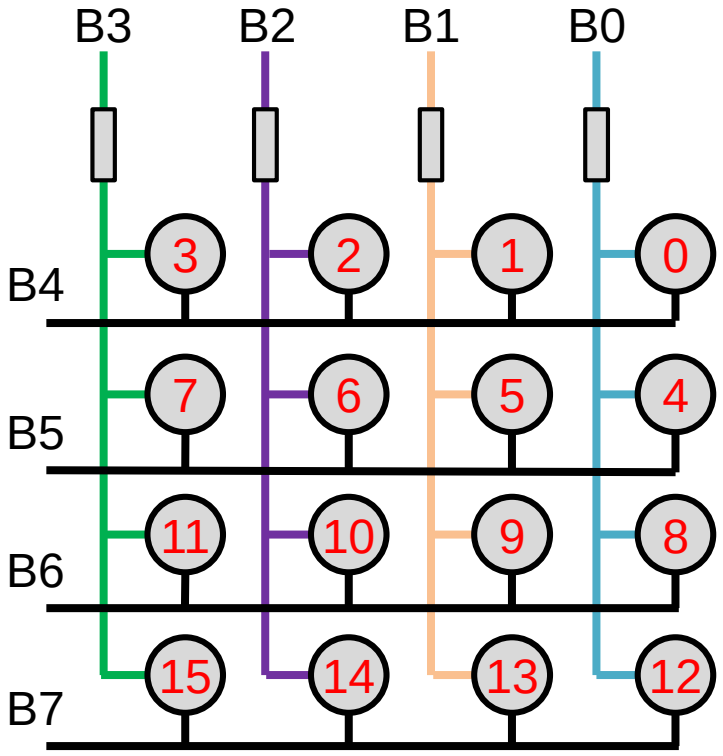




버턴 매트릭스

- 4×4 LED 매트릭스 제어
  - PORTB = B7…B0
  - B3…B0 = 1111, B7…B4 = 0000 → 모두 LED 켜짐
  - B3…B0 = 0000, B7…B4 = 1111 → 모두 LED 꺼짐

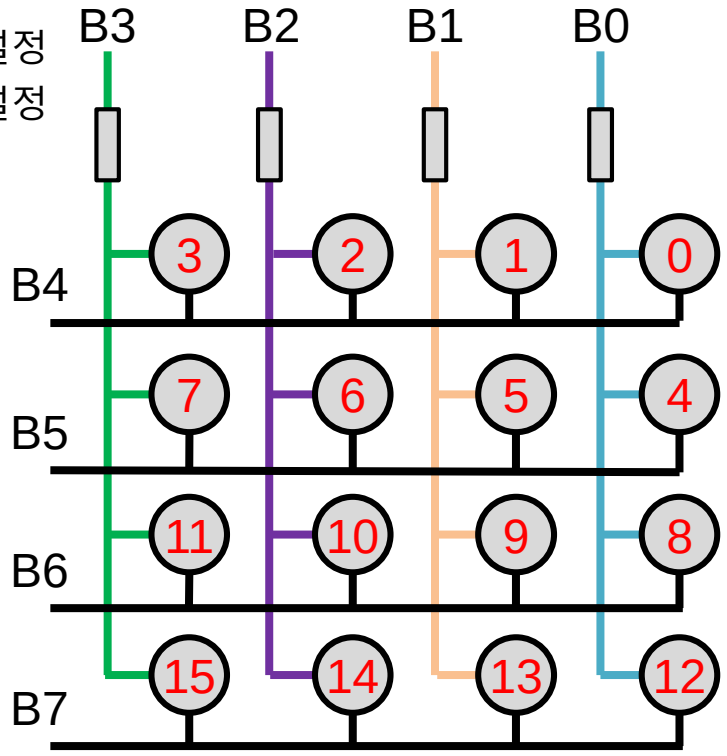
3	2	1	0
1110_1000 (4, 3)	1110_0100 (4, 2)	1110_0010 (4, 1)	1110_0001 (4, 0)
7	6	5	4
1101_1000 (5, 3)	1101_0100 (5, 2)	1101_0010 (5, 1)	1101_0001 (5, 0)
11	10	9	8
1011_1000 (6, 3)	1011_0100 (6, 2)	1011_0010 (6, 1)	1011_0001 (6, 0)
15	14	13	12
0111_1000 (7, 3)	0111_0100 (7, 2)	0111_0010 (7, 1)	0111_0001 (7, 0)



버턴 매트릭스

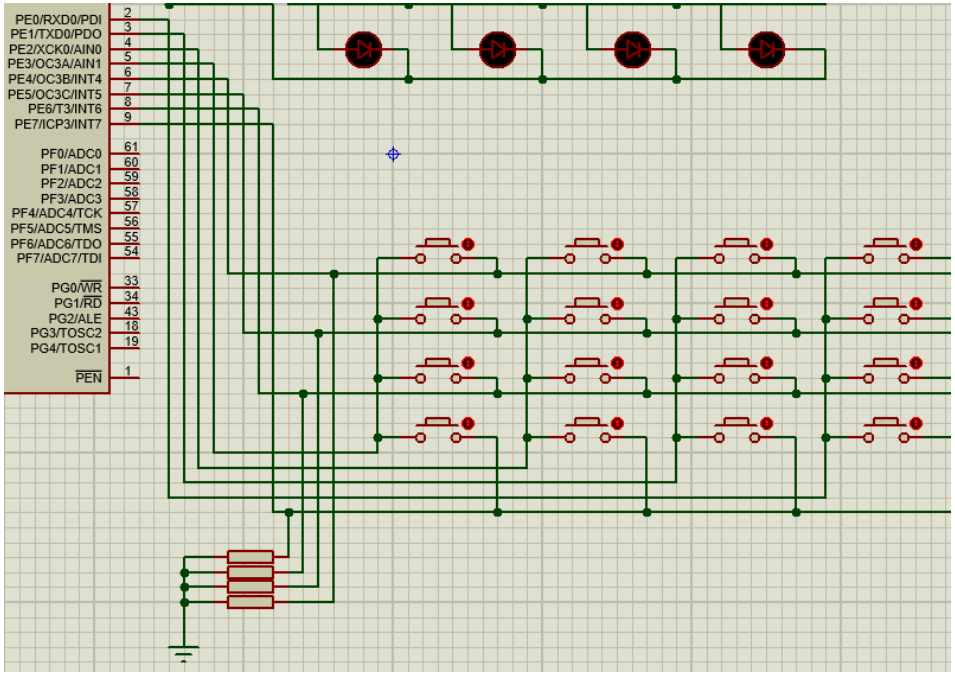
- 4×4 LED 매트릭스 제어 규칙
  - no = 0, 1, 2, 3, …, 15
  - LED 켜짐: 비트 (no/4 + 4)를 0로, 비트 (no%4)를 1로 설정
  - LED 꺼짐: 비트 (no%4)를 0로, 비트 (no/4 + 4)를 1로 설정

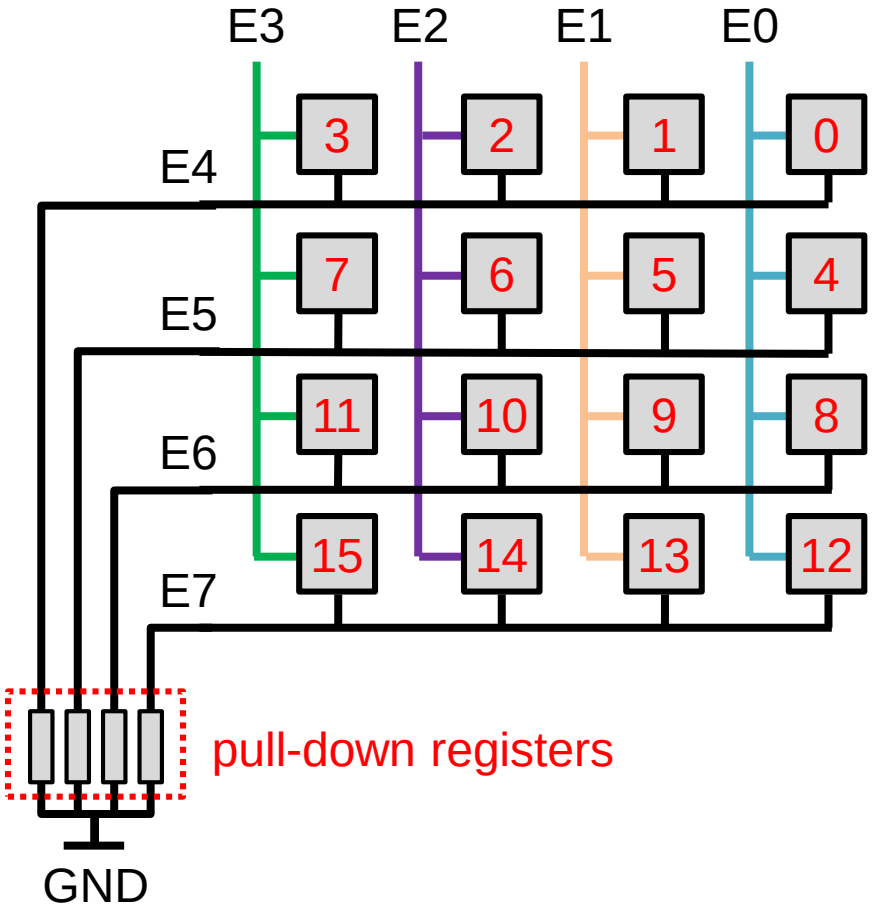
3	2	1	0
1110_1000 (4, 3)	1110_0100 (4, 2)	1110_0010 (4, 1)	1110_0001 (4, 0)
7	6	5	4
1101_1000 (5, 3)	1101_0100 (5, 2)	1101_0010 (5, 1)	1101_0001 (5, 0)
11	10	9	8
1011_1000 (6, 3)	1011_0100 (6, 2)	1011_0010 (6, 1)	1011_0001 (6, 0)
15	14	13	12
0111_1000 (7, 3)	0111_0100 (7, 2)	0111_0010 (7, 1)	0111_0001 (7, 0)



버턴 매트릭스

- 4×4 버턴 매트릭스



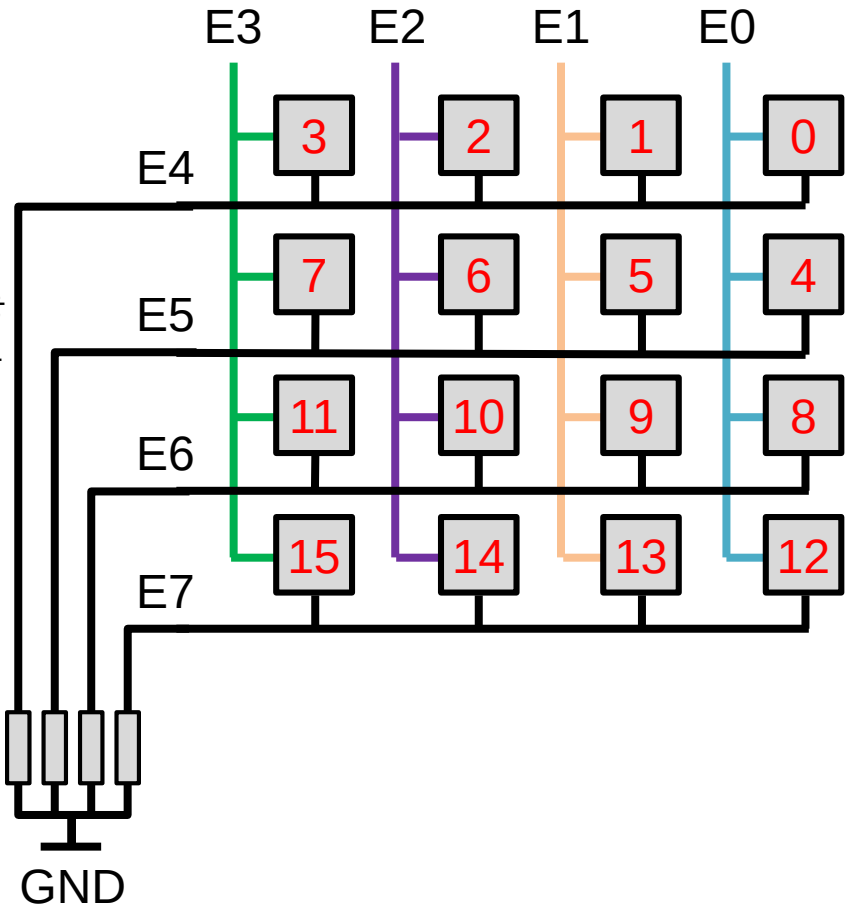


버튼 매트릭스

- 4×4 버튼 매트릭스 제어
  - PORTE = E7…E0
  - E0 = 1 → 버튼 0, 4, 8, 12를 누린 지 인식 가능
  - E1 = 1 → 버튼 1, 5, 9, 13를 누린 지 인식 가능
  - E2 = 1 → 버튼 2, 6, 10, 14를 누린 지 인식 가능
  - E3 = 1 → 버튼 3, 7, 11, 15를 누린 지 인식 가능
  - 예: E0 = 1, 버튼 0을 누림 → E4 = 1

  → E3…E0을 출력, E7…E4를 입력으로 설정

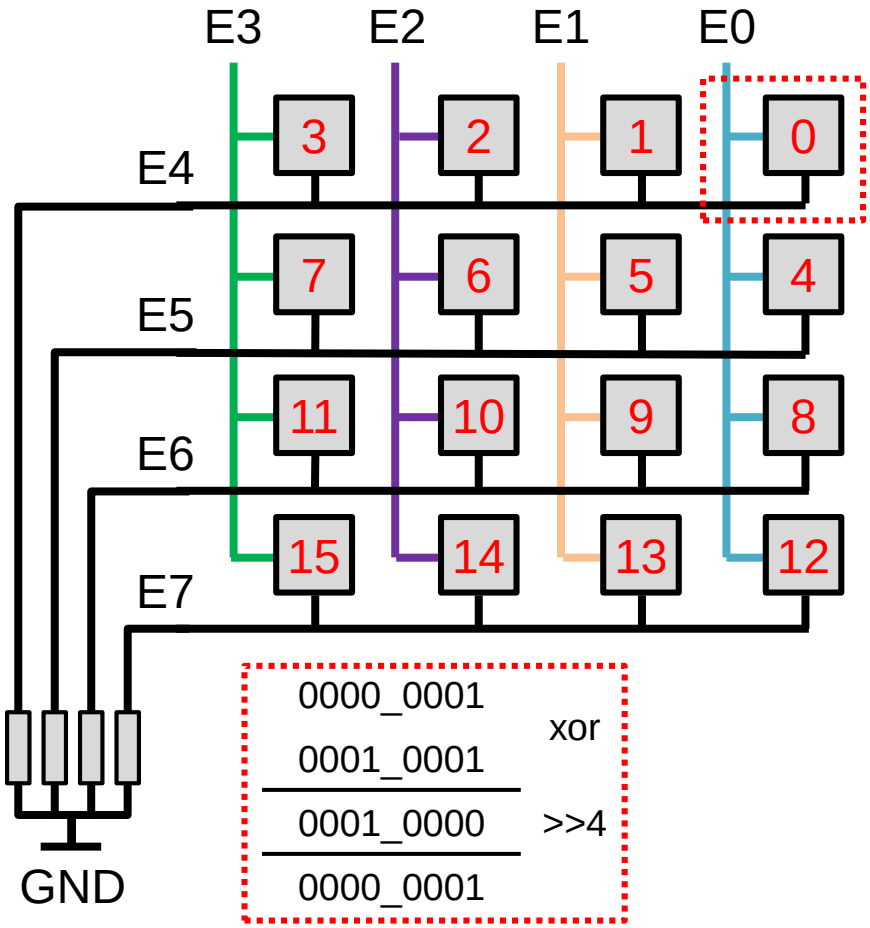
  → E0, E1, E2, E3을 순서대로 1로 설정되면 4×4 버튼 매트릭스 중 어떤 버튼을 누린 지 인식 가능



버턴 매트릭스

- 4×4 버턴 매트릭스 제어 규칙

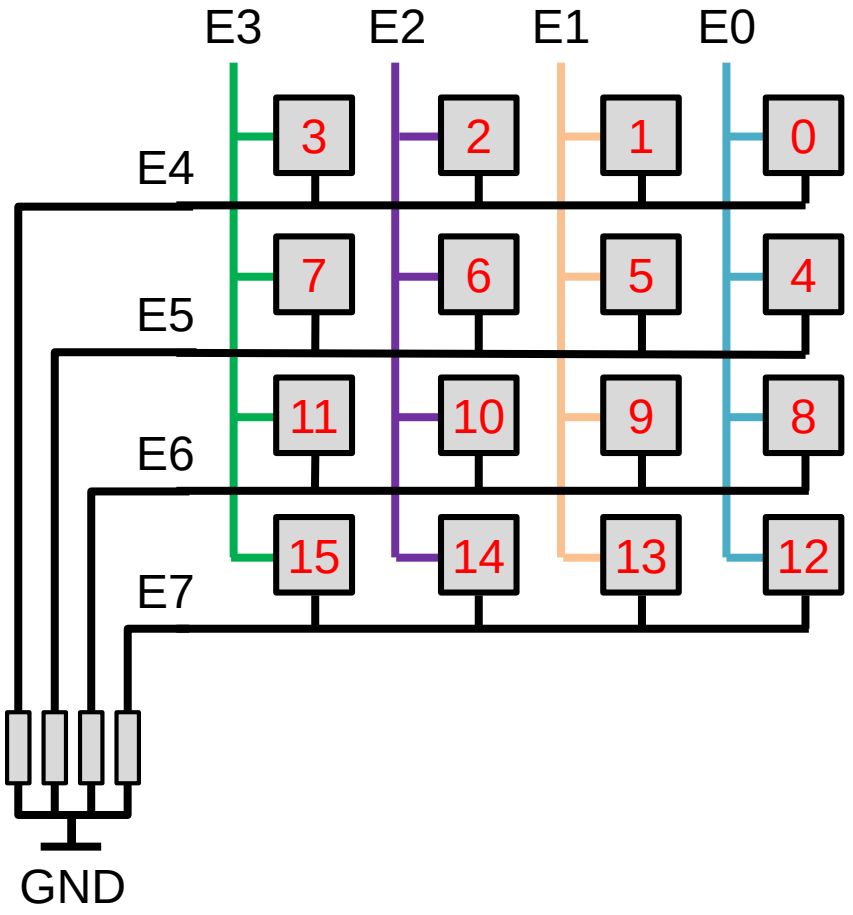
버턴번호	3	2	1	0
출력값	0000_1000	0000_0100	0000_0010	0000_0001
입력값	0001_1000	0001_0100	0001_0010	0001_0001
버턴번호	7	6	5	4
출력값	0000_1000	0000_0100	0000_0010	0000_0001
입력값	0010_1000	0010_0100	0010_0010	0010_0001
버턴번호	11	10	9	8
출력값	0000_1000	0000_0100	0000_0010	0000_0001
입력값	0100_1000	0100_0100	0100_0010	0100_0001
버턴번호	15	14	13	12
출력값	0000_1000	0000_0100	0000_0010	0000_0001
입력값	1000_1000	1000_0100	1000_0010	1000_0001



버턴 매트릭스

- 4×4 버턴 매트릭스 제어 규칙

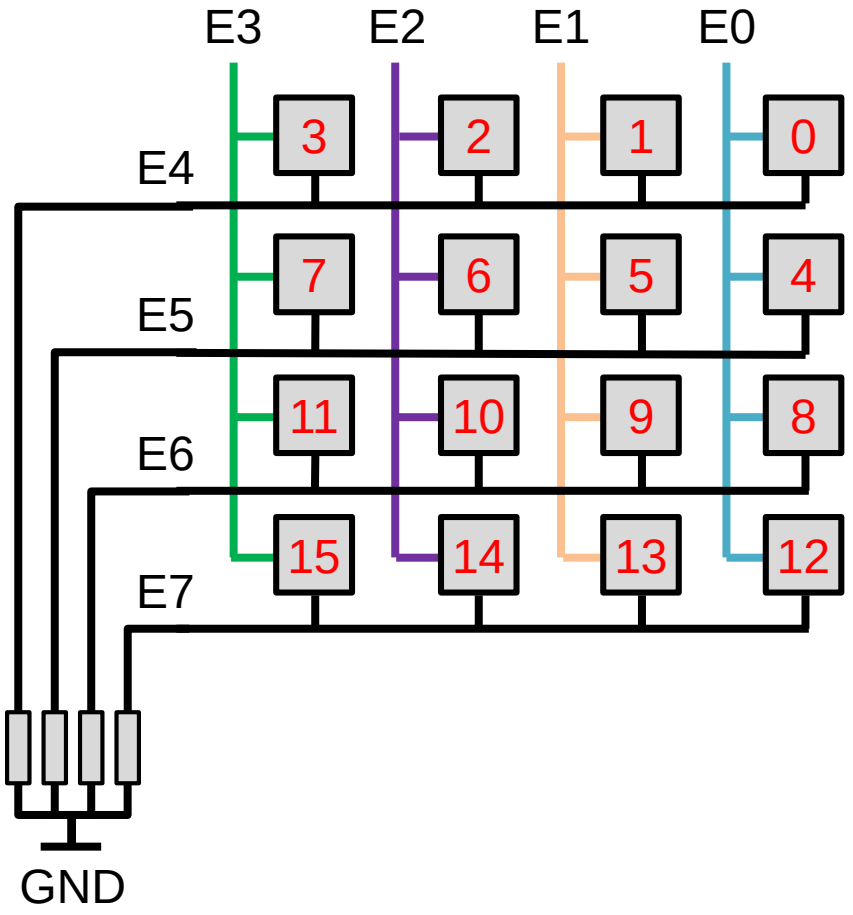
버턴번호	3	2	1	0
출력값	8	4	2	1
입력값	1	1	1	1
버턴번호	7	6	5	4
출력값	8	4	2	1
입력값	2	2	2	2
버턴번호	11	10	9	8
출력값	8	4	2	1
입력값	4	4	4	4
버턴번호	15	14	13	12
출력값	8	4	2	1
입력값	8	8	8	8



버튼 매트릭스

- 4×4 버튼 매트릭스 제어 규칙

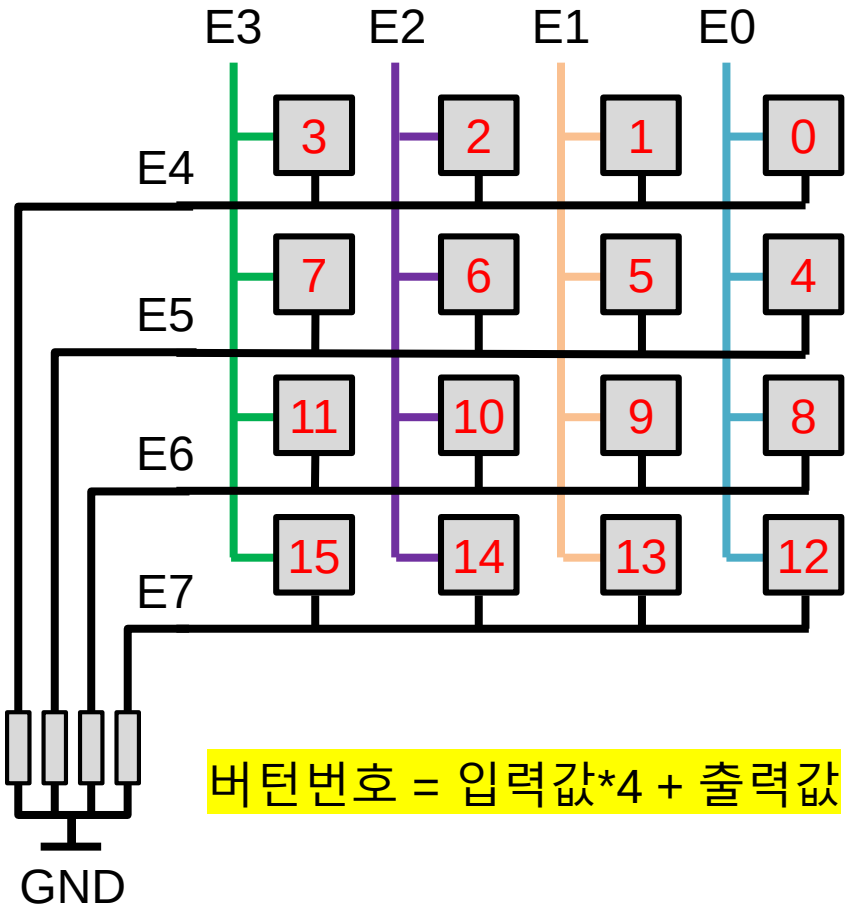
버튼번호	3	2	1	0
출력값	4	2	1	0
입력값	0	0	0	0
버튼번호	7	6	5	4
출력값	4	2	1	0
입력값	1	1	1	1
버튼번호	11	10	9	8
출력값	4	2	1	0
입력값	2	2	2	2
버튼번호	15	14	13	12
출력값	4	2	1	0
입력값	4	4	4	4



버튼 매트릭스

- 4×4 버튼 매트릭스 제어 규칙

버튼번호	3	2	1	0
출력값	3	2	1	0
입력값	0	0	0	0
버튼번호	7	6	5	4
출력값	3	2	1	0
입력값	1	1	1	1
버튼번호	11	10	9	8
출력값	3	2	1	0
입력값	2	2	2	2
버튼번호	15	14	13	12
출력값	3	2	1	0
입력값	3	3	3	3



버턴 매트릭스

- 코드

```
#include <xc.h>
#include <avr/io.h>
#include <util/delay.h>

#define DELAY_TIME 1000
#define cbi(REG8, BITNUM) REG8&=~(_BV(BITNUM))
#define sbi(REG8, BITNUM) REG8|=(_BV(BITNUM))

void getKeyNo(unsigned char);
void number(unsigned char);
void delay(int);
int main(void) {
   unsigned char outE = 0x01;
   DDRB = 0xff; PORTB = 0xf0;
   DDRE = 0x0f; PORTE = outE;
   while(1) {
      getKeyNo(outE);
      outE = outE<<1;
      if (outE==0x10) outE = 0x01;
      PORTE = outE;
   }
}
```

```
void getKeyNo(unsigned char outE) {
   char poE, a, b, no=0;
   poE = PINE;
   poE ^= outE;
   poE >>= 4;
   if (poE) {
      a = poE>>1; // divide by 2
      if (a>3) a = 3;
      b = outE>>1;
      if (b>3) b = 3;
      no = (a<<2)+b; // a*4+b
      number(no);
   }
}

void number(unsigned char no) {
   sbi(PORTB, no%4);
   cbi(PORTB, (no>>2)+4); // no/4+4
   delay(DELAY_TIME);
   cbi(PORTB, no%4);
   sbi(PORTB, (no>>2)+4);
}
```