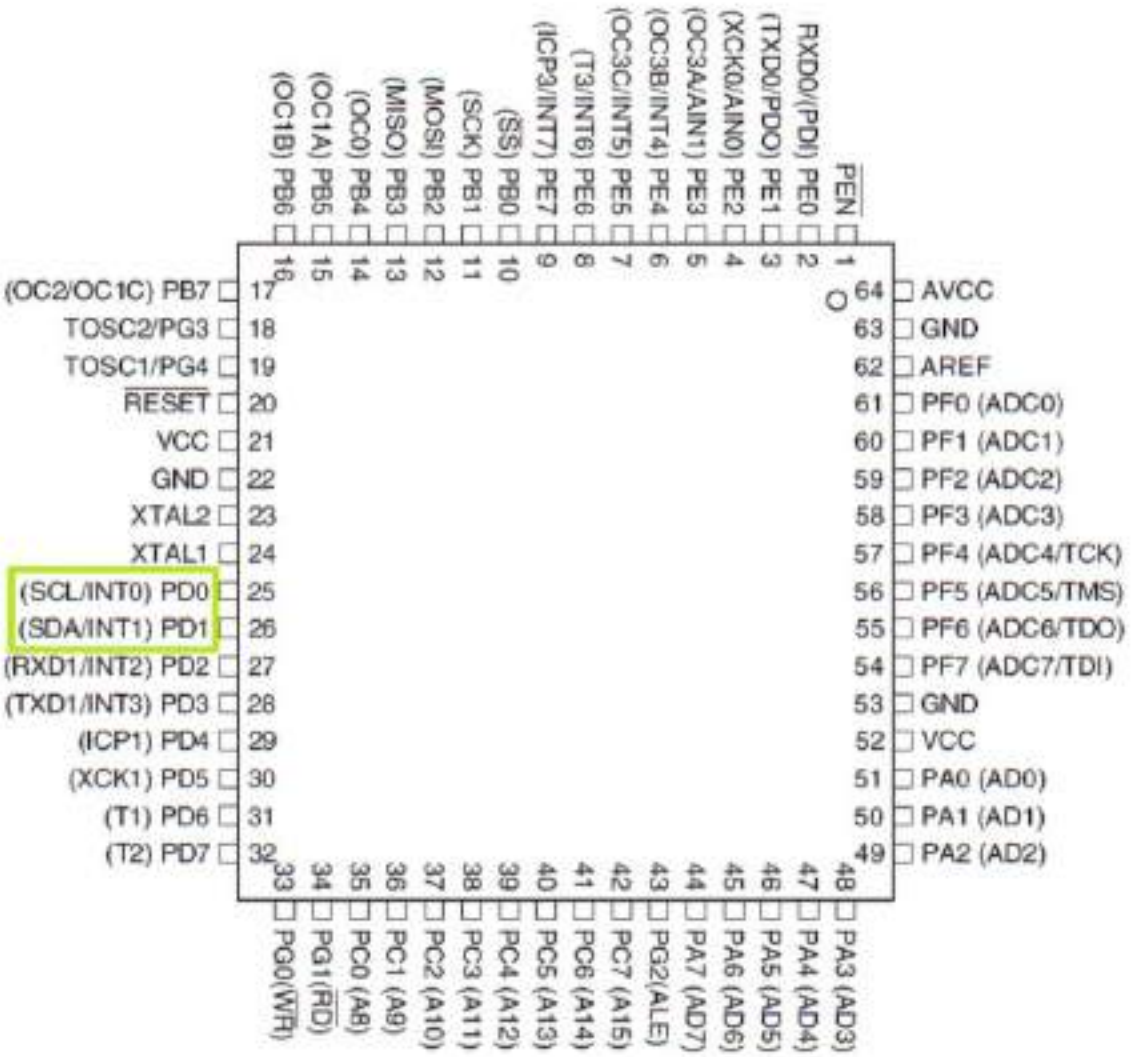

PEN 1
RXD0/(PDI) PE0 2
(TXD0/PDO) PE1 3
(XCK0/AIN0) PE2 4
(OC3A/AIN1) PE3 5
(OC3B/INT4) PE4 6
(OC3C/INT5) PE5 7
(T3/INT6) PE6 8
(ICP3/INT7) PE7 9
(SS) PB0 10
(SCK) PB1 11
(MOSI) PB2 12
(MISO) PB3 13
(OC0) PB4 14
(OC1A) PB5 15
(OC1B) PB6 16
(OC2/OC1C) PB7 17
TOSC2/PG3 18
TOSC1/PG4 19
RESET 20
VCC 21
GND 22
XTAL2 23
XTAL1 24
(SCL/INT0) PD0 25
(SDA/INT1) PD1 26
(RXD1/INT2) PD2 27
(TXD1/INT3) PD3 28
(ICP1) PD4 29
(XCK1) PD5 30
(T1) PD6 31
(T2) PD7 32
64 AVCC
63 GND
62 AREF
61 PF0 (ADC0)
60 PF1 (ADC1)
59 PF2 (ADC2)
58 PF3 (ADC3)
57 PF4 (ADC4/TCK)
56 PF5 (ADC5/TMS)
55 PF6 (ADC6/TDO)
54 PF7 (ADC7/TDI)
53 GND
52 VCC
51 PA0 (AD0)
50 PA1 (AD1)
49 PA2 (AD2)
48 PA3 (AD3)
47 PA4 (AD4)
46 PA5 (AD5)
45 PA6 (AD6)
44 PA7 (AD7)
43 PG2(ALE)
42 PC7 (A15)
41 PC6 (A14)
40 PC5 (A13)
39 PC4 (A12)
38 PC3 (A11)
37 PC2 (A10)
36 PC1 (A9)
35 PC0 (A8)
34 PG1(RD)
33 PG0(WR)

출처: https://m.blog.naver.com/xisaturn/220768837160

I^2C 특징

a. 다수의 장치간의 연결이 가능.

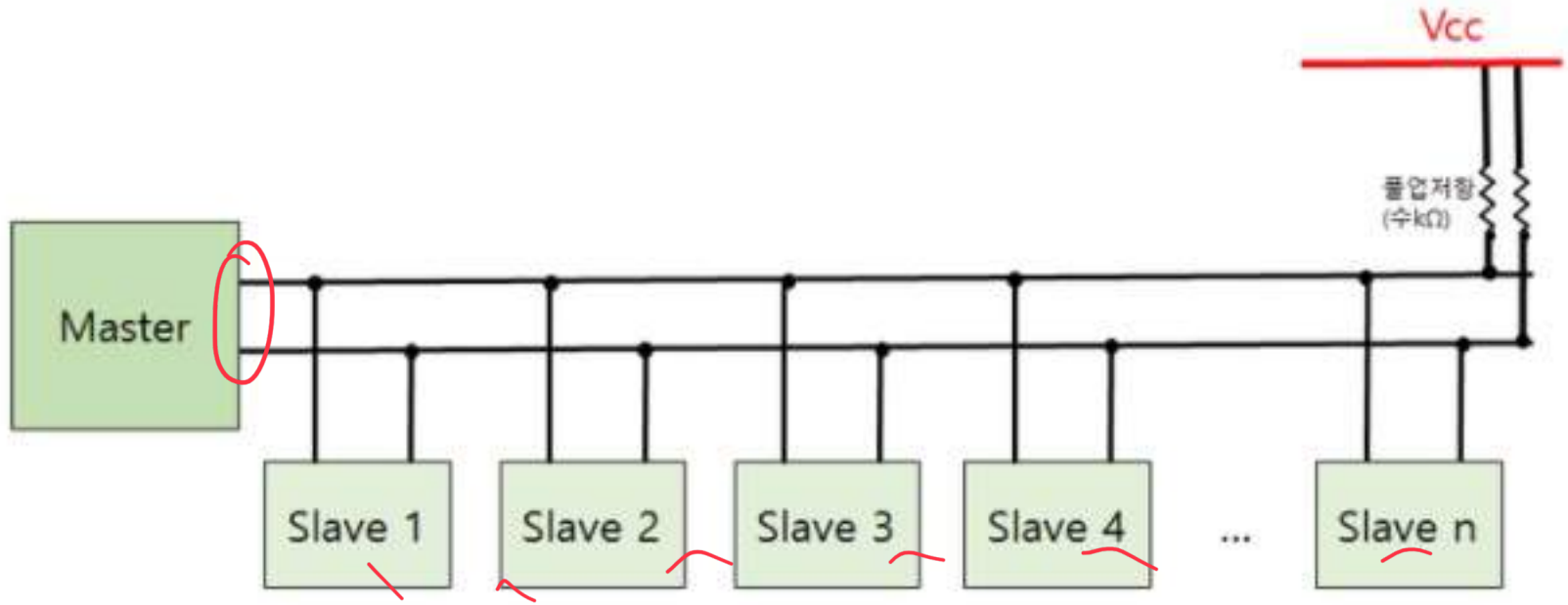
- 버스의 용량(capacitor)이 400pF을 넘지 않는 한에서 장치를 다량 연결 가능
- 다수의 마스터가 버스에 연결되어 있어도 장치간 충돌 검출 가능
- 버스의 연결된 장치들이 고유의 주소값을 가지고 master/slave 관계를 가진다.
- 각 (슬레이브)장치는 고유의 주소값(7bit)을 가진다.

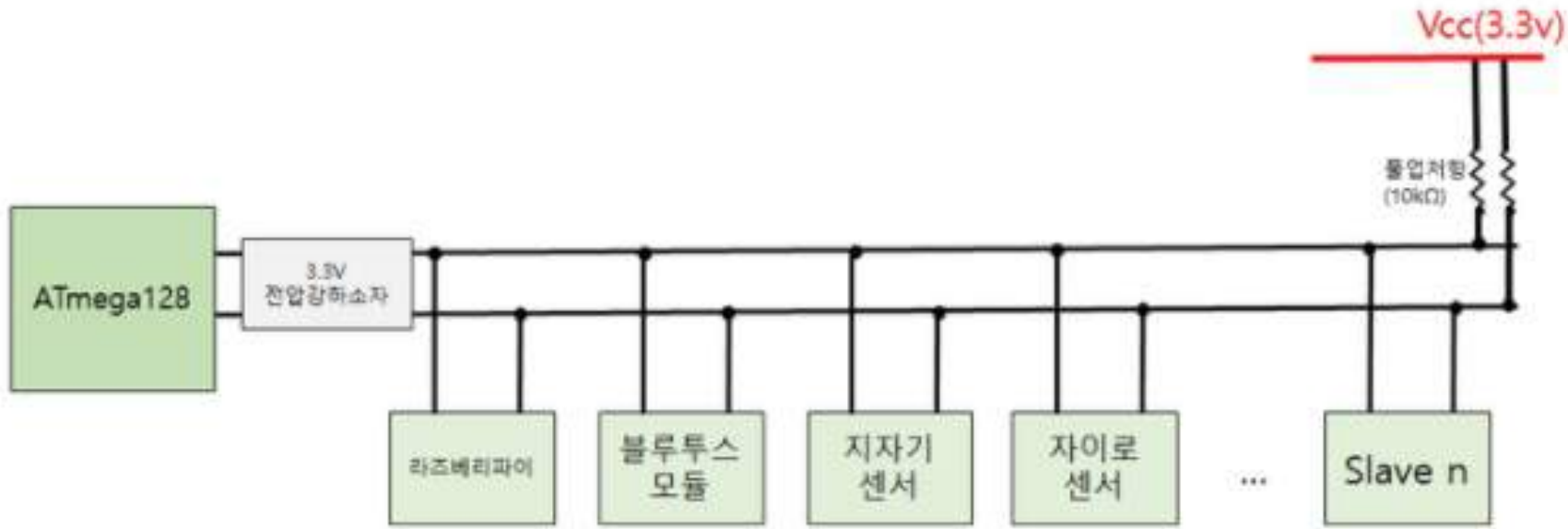
b. 세 가지 전송 속도를 지원.

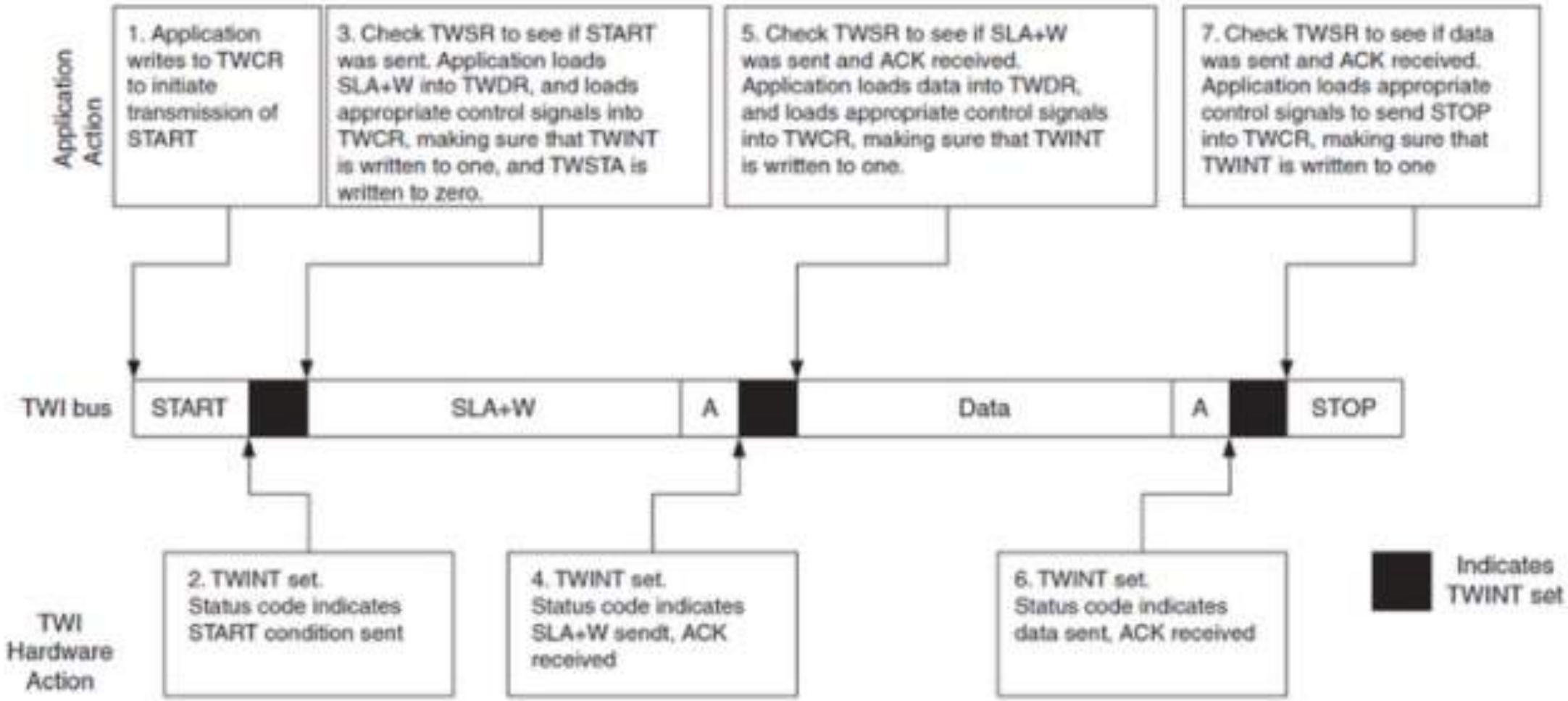
- 표준 모드(Standard Mode) : 100 Kbps
- 고속 모드(Fast Mode) : 400 Kbps
- 초고속 모드(High Speed Mode) : 3.4Mbps

c. 통신 방법

- '마스터'가 통신의 시작 및 통신의 종료를 선언하여 통신이 이루어진다.
- '마스터'가 특정 주소를 가진 슬레이브와 연결하여 송수신을 수행한다.
- '마스터'가 클록신호 SCL을 발생하여 통신을 중계한다.

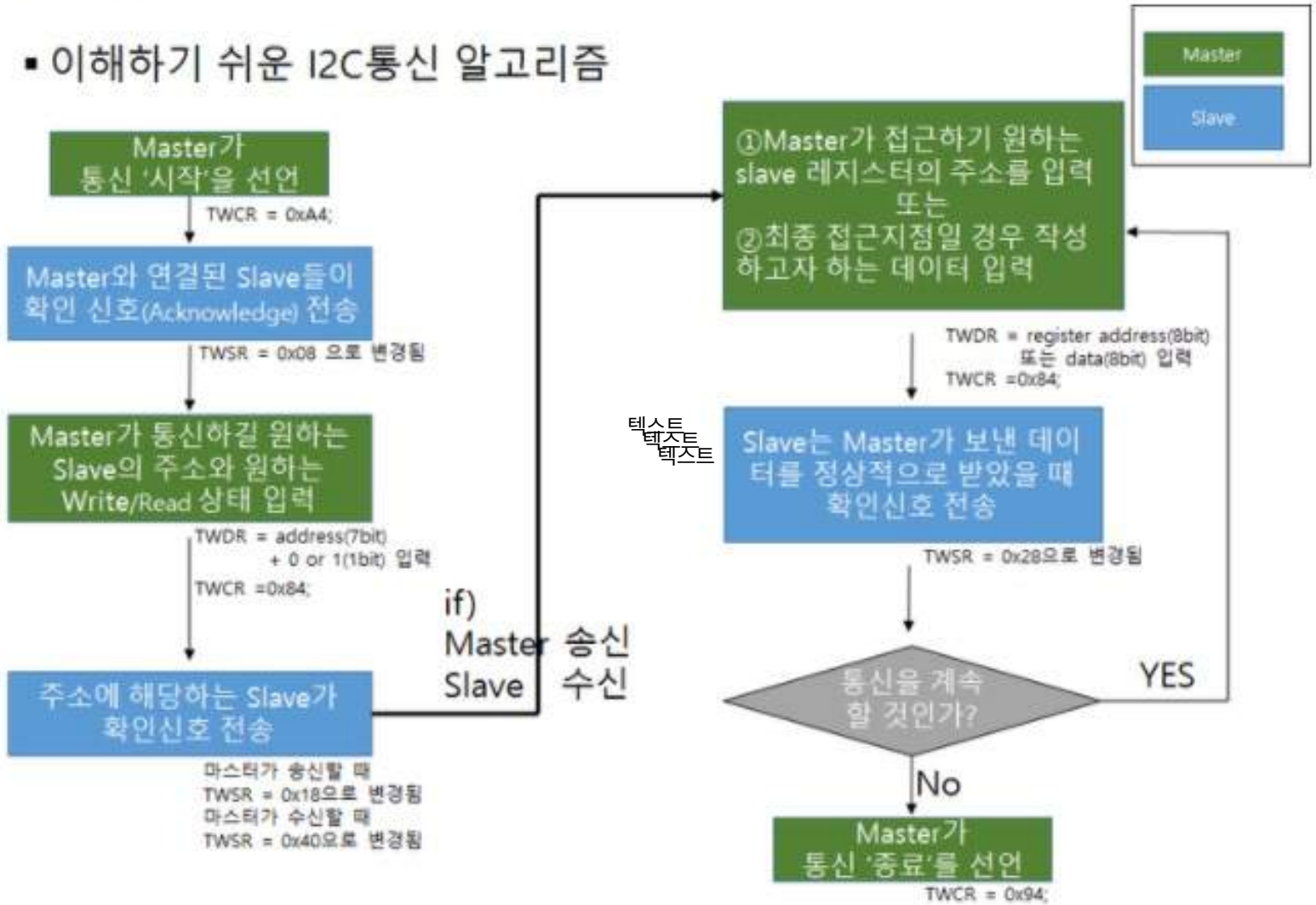

Vcc
풀업저항
(수kΩ)
Master
Slave 1
Slave 2
Slave 3
Slave 4
...
Slave n


Vcc(3.3v)
풀업저항
(10kΩ)
ATmega128
3.3V
전압강하소자
라즈베리파이
블루투스
모듈
지자기
센서
자이로
센서
...
Slave n


Application Action
1. Application writes to TWCR to initiate transmission of START
3. Check TWSR to see if START was sent. Application loads SLA+W into TWDR, and loads appropriate control signals into TWCR, making sure that TWINT is written to one, and TWSTA is written to zero.
5. Check TWSR to see if SLA+W was sent and ACK received. Application loads data into TWDR, and loads appropriate control signals into TWCR, making sure that TWINT is written to one.
7. Check TWSR to see if data was sent and ACK received. Application loads appropriate control signals to send STOP into TWCR, making sure that TWINT is written to one
TWI bus
START
SLA+W
A
Data
A
STOP
TWI Hardware Action
2. TWINT set. Status code indicates START condition sent
4. TWINT set. Status code indicates SLA+W sendt, ACK received
6. TWINT set. Status code indicates data sent, ACK received
Indicates TWINT set

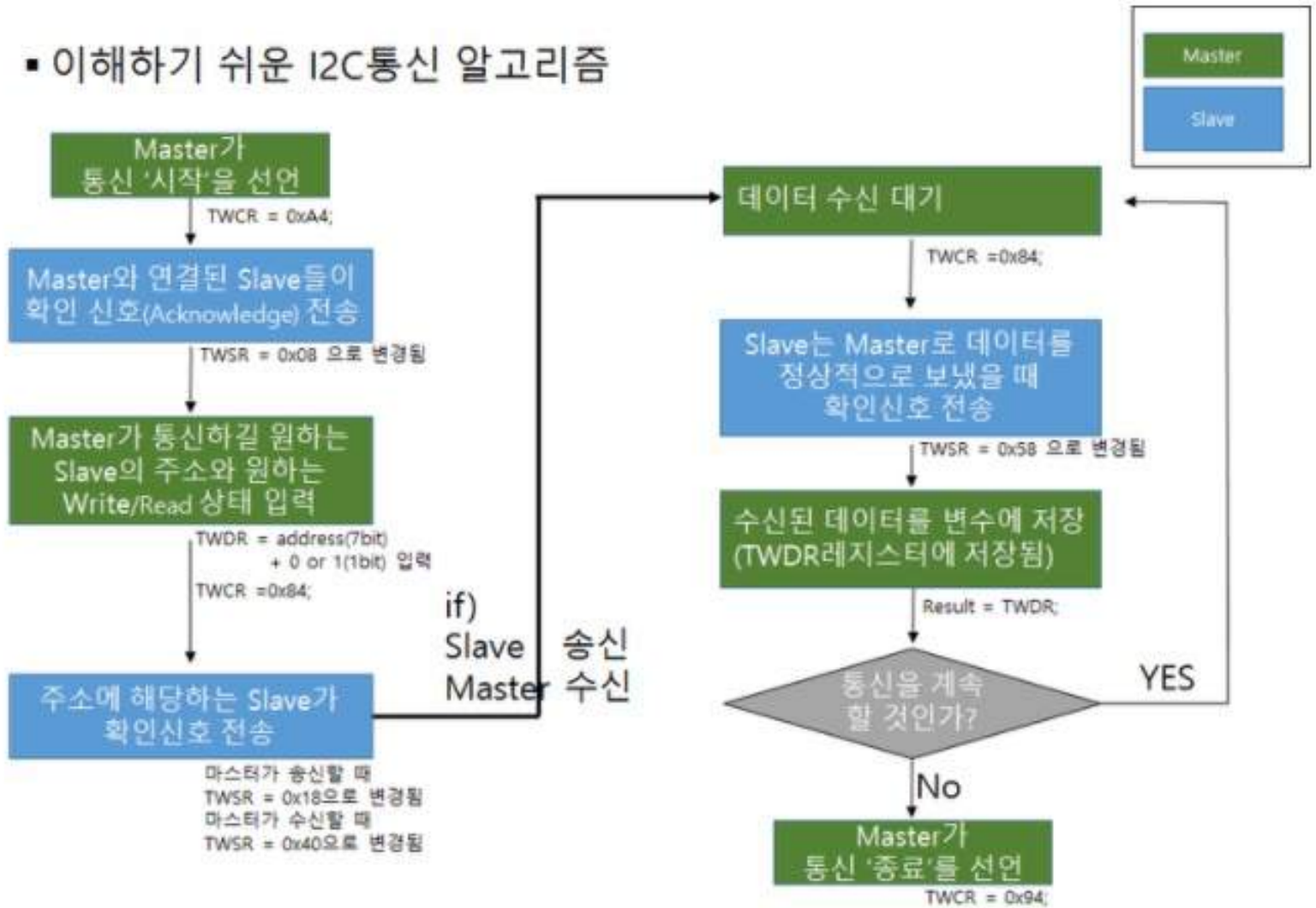
a. 마스터송신

▪ 이해하기 쉬운 I2C통신 알고리즘



b. 마스터수신

▪ 이해하기 쉬운 I2C통신 알고리즘



```
void I2C_init(void)
{
        TWBR = 12;          // I2C 전송속도 400KHz로 세팅 (시스템클럭이 16MHz인 경우)
        TWSR = 0x00;
}

void I2C_start(void)
{
        TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN);
}

void I2C_stop(void)
{
        TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTO);
}
```

```
void I2Cwrite(unsigned char address, unsigned char data)
{
        I2C_start()                     // START condition 송신 (TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN);)
        // START condition 신호가 전송완료까지 대기
        while(((TWCR & 0x80) == 0x00) || (TWSR & 0xF8) != 0x08);
        TWDR = 0xd0 | 0x00;                                     // 부품의 I2C 주소(datasheet: DS1307은 0xd0 )
        TWCR = 0x84;                                            // 마스터 송신 모드
        // 전송 완료 시 슬레이브에서 ACK 대기
        while(((TWCR & 0x80) == 0x00) || (TWSR & 0xF8) != 0x18);
        TWDR = address;                                         // Write address (datasheet참조해야 함)
        TWCR = 0x84;                                            // 마스터 송신 모드
        // 전송 완료 시 슬레이브에서 ACK 대기
        while(((TWCR & 0x80) == 0x00) || (TWSR & 0xF8) != 0x28);
        TWDR = data;                                            // write data
        TWCR = 0x84;                                            // 마스터 송신 모드
        // 전송 완료 시 슬레이브에서 ACK 대기
        while(((TWCR & 0x80) == 0x00) || (TWSR & 0xF8) != 0x28);
        I2C_stop();                     // STOP condition (TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTO);)
        delay(1);
}
```

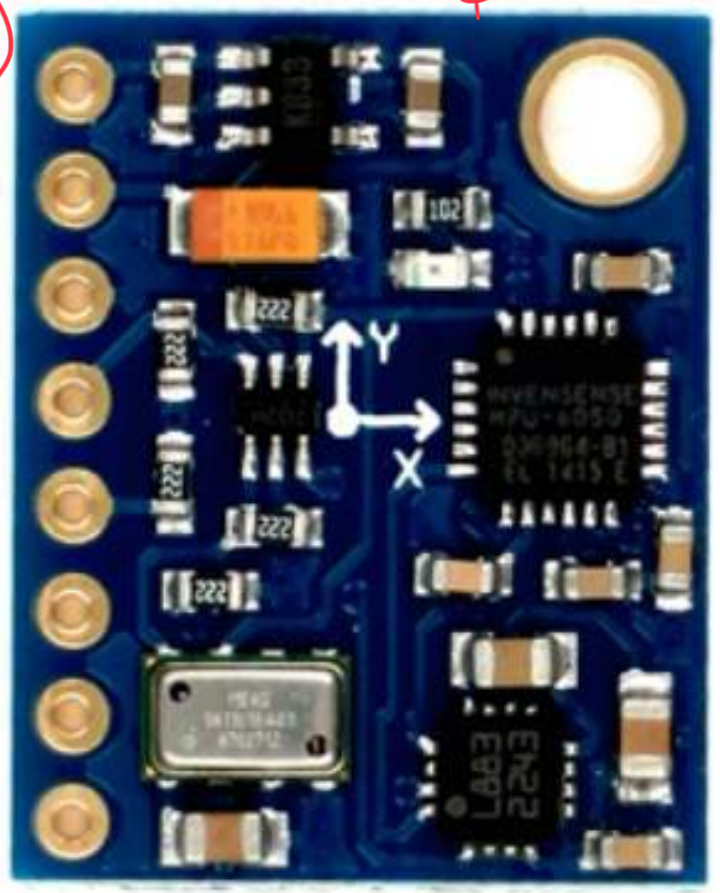
```
unsigned char I2Cread(unsigned char address)
{
        unsigned char data = 0;

        I2C_start();  // START condition 송신
        // START condition 신호가 전송완료까지 대기
        while(((TWCR & 0x80) == 0x00) || (TWSR & 0xF8) != 0x08);
        TWDR = DS1307|WMODE;            // DS1307 address & write mode
        TWCR = 0x84;                    // 마스터 송신 모드
        // 전송 완료시 슬레이브에서 ACK 대기
        while(((TWCR & 0x80) == 0x00) || (TWSR & 0xF8) != 0x18);
        TWDR = address;         // Write address (읽고자 하는 주소)
        TWCR = 0x84;            // 마스터 송신 모드
        // 전송 완료시 슬레이브에서 ACK 대기
        while(((TWCR & 0x80) == 0x00) || (TWSR & 0xF8) != 0x28);
        TWCR = 0xA4;            // RESTART
        // 상태코드가 0x10 확인
        while(((TWCR & 0x80) == 0x00) || (TWSR & 0xF8) != 0x10);
        TWDR = DS1307|RMODE;            // DS1307 address & read mode
        TWCR = 0x84;                    // 마스터 송신 모드
        // 전송 완료시 슬레이브에서 ACK 대기
        while(((TWCR & 0x80) == 0x00) || (TWSR & 0xF8) != 0x40);
        TWCR = 0x84;                    // 마스터 수신대기 모드
        // data 수신 대기
        while(((TWCR & 0x80) == 0x00) || (TWSR & 0xF8) != 0x58);
        data = TWDR;                    // data 수신
        TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTO);     // STOP condition
        return data;
}
```

예제: GY-86 (출처: https://blog.komastar.kr/13)

- I2C(TWI)통신으로 GY-86모듈 내에 있는 MPU6050의 가속도 XYZ 출력(0x3B - 0x40)과 자이로 XYZ 출력(0x43 - 0x48) 레지스터를 읽기

Description

10DOF modules (3축 자이로 + 3축 가속도 + 3축 지자계 + 대기압력)
Sensor : MPU6050 + HMC5883L + MS5611
gyroscope range: + 250 500 1000 2000 °/ s
Acceleration range: ± 2 ± 4 ± 8 ± 16 g
HMC5883L: Measuring range: ± 1.3-8 Gauss.
Power supply :3-5v
(fully compatible with 3v-5V systems, including LLC circuit)
Interface : I2C
Weight: 1.46g
Size: 22 x 17mm



DATASHEET 다운로드 !!!
(MPU6050 + HMC5883L + MS5611)

MPU6050 datasheet의 register map (7페이지)를 보면 아래와 같음 (원하는 데이터의 주소를 확인할 것)

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
38	56	INT_ENABLE	R/W	FF_EN	MOT_EN	ZMOT_EN	FIFO _OFLOW _EN	I2C_MST _INT_EN	-	-	DATA _RDY_EN
3A	58	INT_STATUS	R	FF_INT	MOT_INT	ZMOT _INT	FIFO _OFLOW _INT	I2C_MST _INT	-	-	DATA _RDY_INT
3B	59	ACCEL_XOUT_H	R	ACCEL_XOUT[15:8]							
3C	60	ACCEL_XOUT_L	R	ACCEL_XOUT[7:0]							
3D	61	ACCEL_YOUT_H	R	ACCEL_YOUT[15:8]							
3E	62	ACCEL_YOUT_L	R	ACCEL_YOUT[7:0]							
3F	63	ACCEL_ZOUT_H	R	ACCEL_ZOUT[15:8]							
40	64	ACCEL_ZOUT_L	R	ACCEL_ZOUT[7:0]							
41	65	TEMP_OUT_H	R	TEMP_OUT[15:8]							
42	66	TEMP_OUT_L	R	TEMP_OUT[7:0]							
43	67	GYRO_XOUT_H	R	GYRO_XOUT[15:8]							
44	68	GYRO_XOUT_L	R	GYRO_XOUT[7:0]							
45	69	GYRO_YOUT_H	R	GYRO_YOUT[15:8]							
46	70	GYRO_YOUT_L	R	GYRO_YOUT[7:0]							
47	71	GYRO_ZOUT_H	R	GYRO_ZOUT[15:8]							
48	72	GYRO_ZOUT_L	R	GYRO_ZOUT[7:0]							

```
#include <stdlib.h>
#include <delay.h>
#include <mega128.h>
#include <math.h>
#include <stdio.h>
#include <string.h>
typedef unsigned char byte;

byte MPU6050_read(byte addr);
void MPU6050_write(byte addr, char data);
void getRawData();

unsigned int gx = 0, gy = 0, gz = 0, ax = 0, ay = 0, az = 0;
byte buffer[12];

void main(void)
{
        TWSR = 0x00;
        TWBR = 0x12;

        MPU6050_write(0x6B, 0x00);
        MPU6050_write(0x6C, 0x00);

        getRawData();

        while(1)
        {
                getRawData();
        }
}
```

```
byte MPU6050_read(byte addr)
{
        byte dat;

        TWCR = 0xA4;
        while(((TWCR & 0x80) == 0x00 || ((TWSR & 0xF8) != 0x08)));

        TWDR = 0xD0;
        TWCR = 0x84;

        while(((TWCR & 0x80) == 0x00 || ((TWSR & 0xF8) != 0x18)));
        TWDR = addr;
        TWCR = 0x84;

        while(((TWCR & 0x80) == 0x00 || ((TWSR & 0xF8) != 0x28)));

        TWCR = 0xA4;

        while(((TWCR & 0x80) == 0x00 || ((TWSR & 0xF8) != 0x10)));

        TWDR = 0xD1;
        TWCR = 0x84;

        while(((TWCR & 0x80) == 0x00 || ((TWSR & 0xF8) != 0x40)));
        TWCR = 0x84;
        while(((TWCR & 0x80) == 0x00 || ((TWSR & 0xF8) != 0x58)));
        dat = TWDR;
        TWCR = 0x94;
        return dat;
}
```

```
void MPU6050_write(byte addr, char data)
{
        TWCR = 0xA4;
        while(((TWCR & 0x80) == 0x00 || ((TWSR & 0xF8) != 0x08)));

        TWDR = 0xD0;
        TWCR = 0x84;

        while(((TWCR & 0x80) == 0x00 || ((TWSR & 0xF8) != 0x18)));
        TWDR = addr; // addr = 0x43
        TWCR = 0x84;

        while(((TWCR & 0x80) == 0x00 || ((TWSR & 0xF8) != 0x28)));
        TWDR = data;
        TWCR = 0x84;

        while(((TWCR & 0x80) == 0x00 || ((TWSR & 0xF8) != 0x28)));
        TWCR = 0x94;

        delay_us(50);
}
```

```
void getRawData()
{
        buffer[0] = MPU6050_read(0x3B);
        buffer[1] = MPU6050_read(0x3C);
        buffer[2] = MPU6050_read(0x3D);
        buffer[3] = MPU6050_read(0x3E);
        buffer[4] = MPU6050_read(0x3F);
        buffer[5] = MPU6050_read(0x40);
        buffer[6] = MPU6050_read(0x43);
        buffer[7] = MPU6050_read(0x44);
        buffer[8] = MPU6050_read(0x45);
        buffer[9] = MPU6050_read(0x46);
        buffer[10] = MPU6050_read(0x47);
        buffer[11] = MPU6050_read(0x48);

        ax = (int)buffer[0] << 8 | (int)buffer[1];
        ay = (int)buffer[2] << 8 | (int)buffer[3];
        az = (int)buffer[4] << 8 | (int)buffer[5];

        gx = (int)buffer[6] << 8 | (int)buffer[7];
        gy = (int)buffer[8] << 8 | (int)buffer[9];
        gz = (int)buffer[10] << 8 | (int)buffer[11];
}
```