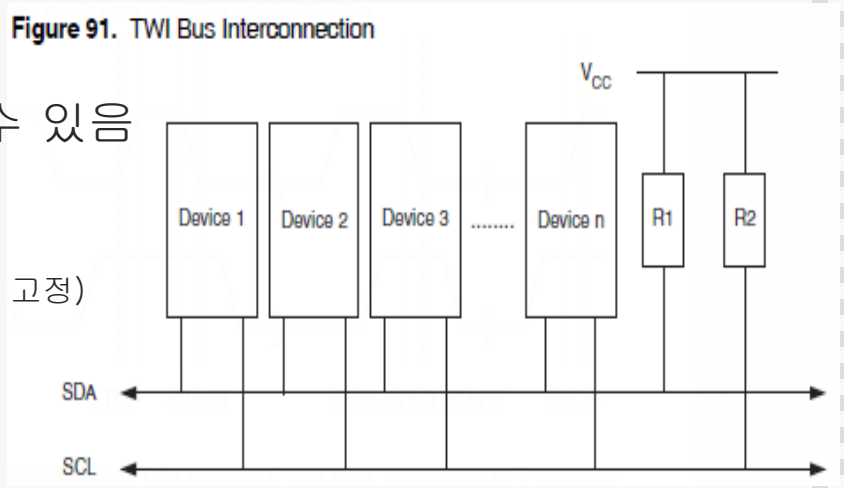
Project #14(Pjt14_I2C_TCN75)

I2C Interface

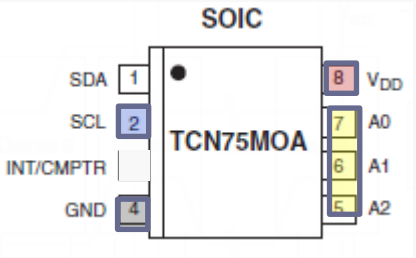
- 2-wire half-duplex serial communication
- master가 동일 버스에 연결된 여러 개의 slave(센서) 중 하나를 7bit 주소로 지정하여 통신
- 주소와 읽기/쓰기 명령으로 구성된 패킷(SLA+R/SLA+W)을 보낸 후,
  . SLA(Slave Address)는 "1001+주소(000~111)" 형태의 상위 7bit. ex)1001000r/w
  . SLA+R(1)인 경우 master가 slave로부터 1 바이트 데이터를 수신
  . SLA+W(0)인 경우 master가 slave에게 1 바이트 데이터를 송신
  . ex) 0x91 ➔ SLA: 1001000(=0x48), R(=1)
  . ex) 0x90 ➔ SLA: 1001000(=0x48), W(=0)
- 1 바이트 데이터 수신 후 ack bit를 보낼 수 있음
- Pin
  . SCL : PORTD 0번
  . SDA : PORTD 1번
  . A0 : PORTH 1번
  . A1 : PORTH 3번
  . A2 : PORTH 5번

임의 선택(TWI의 경우 SCL, SDA 고정)

000 이므로 하나로 결합사용 가능



Project #14(Pjt14_I2C_T



TCN75 센서

- slave 주소는 1001xxx(xxx의 기본값은 000)
- 5 개의 레지스터

TABLE 3-1: TCN75 SLAVE ADDRESS

1	0	0	1	A2	A1	A0
MSB						LSBS

. Pointer 레지스터

D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]
Must Be Set To Zero						Pointer	

. Config 레지스터

D [7]	D [6]	D [5]	D [4]	D [3]	D [2]	D [1]	D [0]
Must Be Set To Zero			Fault Queue		INT/ CMPTR, Polarity	COM P/INT	Shut-down

. Temp 레지스터
. Thyst 레지스터
. Tset 레지스터

D[15]	D[14]	D[13]	D[12]	D[11]	D[10]	D[9]	D[8]	D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]
MSB	D7	D6	D5	D4	D3	D2	D1	LSB	X	X	X	X	X	X	X

- Pointer 레지스터는 기본으로 접근할 레지스터 번호를 저장

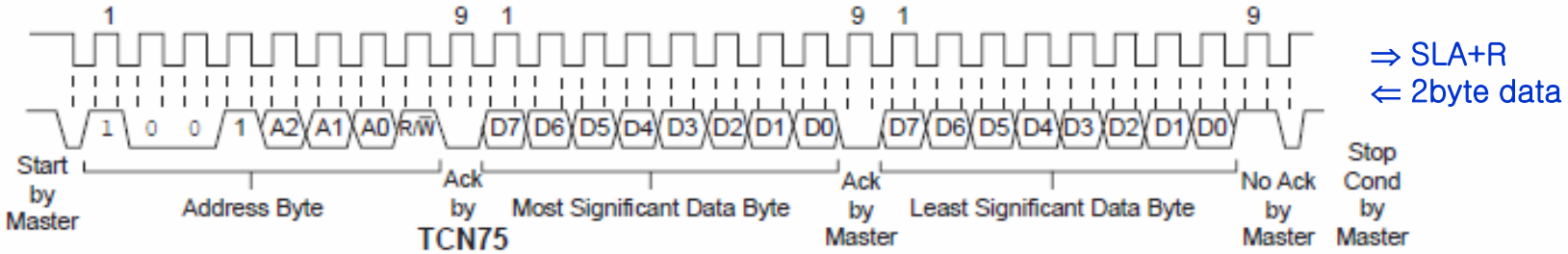
D1	D0	Register Selection
0	0	TEMP
0	1	CONFIG
1	0	T_{HYST}
1	1	T_{SET}

Temperature	Binary Value	HEX Value
+125°C	0 11111010	0FA
+25°C	0 00110010	032
+0.5°C	0 00000001	001
0°C	0 00000000	00
0.5°C	1 11111111	1FF
-25°C	1 11001110	1CE
-40°C	1 10110000	1B0
-55°C	1 10010010	192

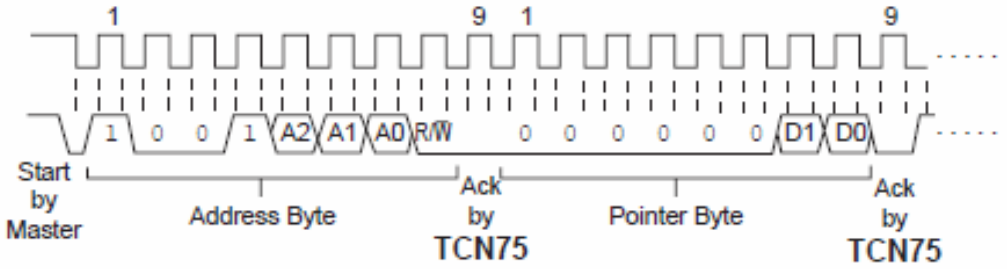
Project #14(Pjt14_I2C_TCN75)

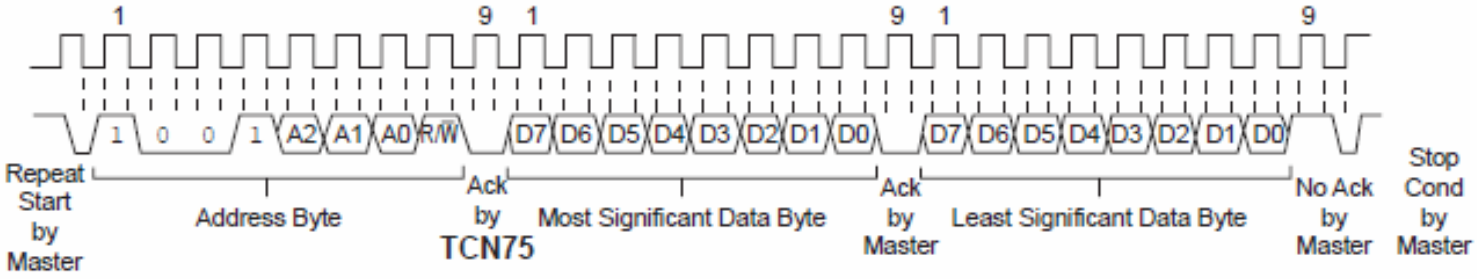
I2C Protocol(TCN75)

- START신호로 시작하고, STOP신호로 종료
- 바이트 단위 수신 및 송신 사이에 수신자는 ack 비트를 보냄
- SLA+R 패킷을 수신하면 Pointer에 설정된 레지스터(기본으로 Temp)의 값을 보냄
- SLA+W 패킷 후 포인터 번호를 수신하면 Pointer 레지스터 재설정
- SLA+W 패킷 후 포인터 번호, Start 신호, SLA+R 수패킷을 수신하면 채널을 종료하지 않고 해당 레지스터 값을 이어서 전송

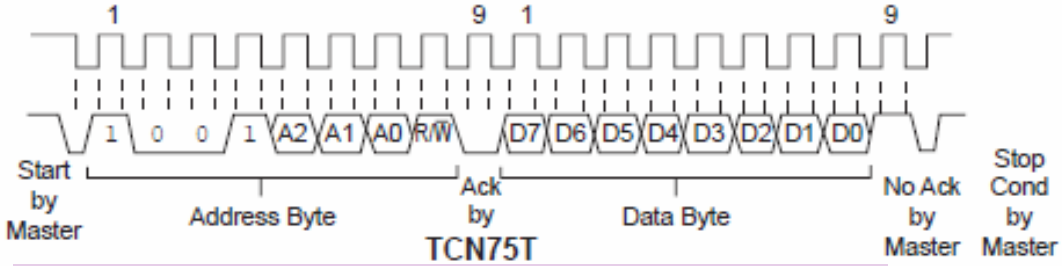


(a) Typical 2-Byte Read From Preset Pointer Location Such as Temp, T_{OS}, T_{HYST}





(b) Typical Pointer Set Followed by Immediate Read for 2-Byte Register Such as Temp, T_{OS}, T_{HYST}



(c) Typical 1-Byte Read From Configuration Register with Preset Pointer

%1byte, 2byte data는 ack bit로 식별

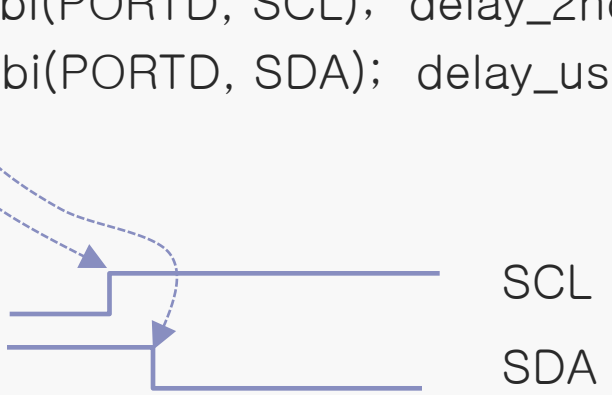
Project #14(Pjt14_I2C_TCN75)

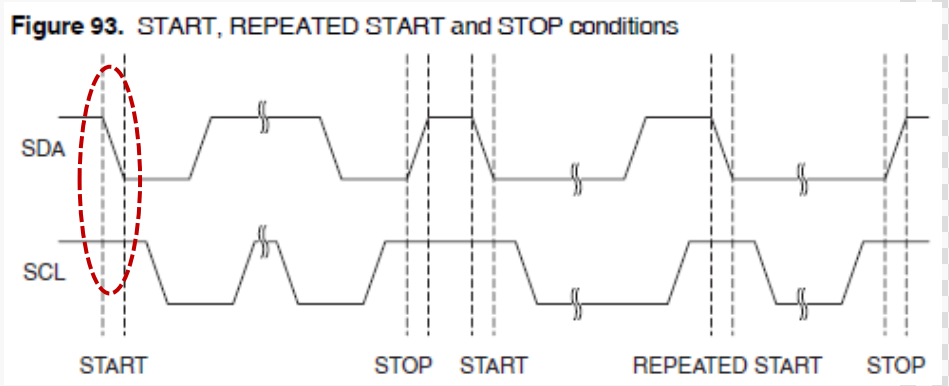
- I2C Interface Application : TCN75 Driver

```
#define delay_2nop()  asm volatile("nop"::); asm volatile("nop"::);
#define delay_us        _delay_us
#define SCL  0
#define SDA  1

void  tcn75_i2c_init()
{
     sbi(DDRD,   SDA); sbi(PORTD, SDA); // output, hi
     sbi(DDRD,   SCL); sbi(PORTD, SCL);  // output, hi        ➔ idle state

     sbi(DDRH, 1);  cbi(PORTH, 1);        // output , A0 ← 0
     sbi(DDRH, 3);  cbi(PORTH, 3);        // output , A1 ← 0
     sbi(DDRH, 5);  cbi(PORTH, 5);        // output , A2 ← 0  ➔ address bit
 }
```

Project #14(Pjt14_I2C_TCN75)

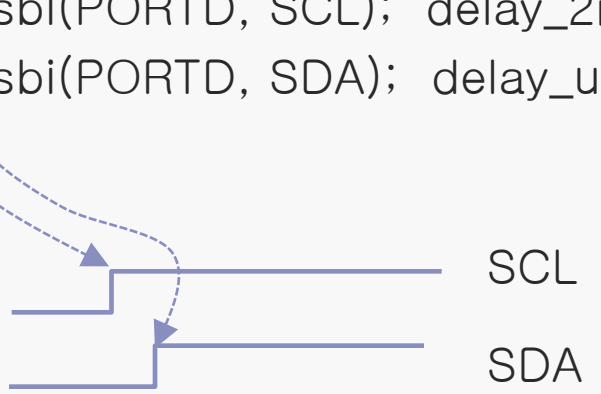
```
inline void  i2c_tcn75_trans_start()
{
      cbi(PORTD, SCL);                          // SCL ← L
      sbi(PORTD, SDA);  delay_2nop();     // SDA ← H
      sbi(PORTD, SCL);  delay_2nop();      // SCL ← H
      cbi(PORTD, SDA);  delay_us(1);        // SDA ← L
 }
```

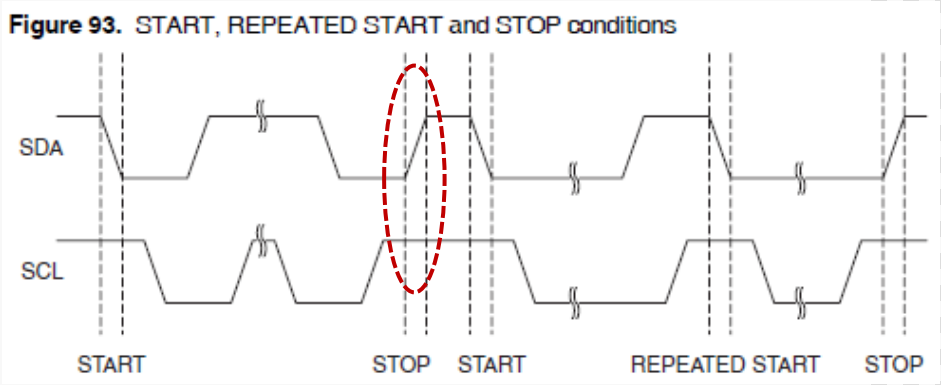




Project #14(Pjt14_I2C_TCN75)

```
inline void  i2c_tcn75_trans_stop()
{
      cbi(PORTD, SCL);                          // SCL ← L
      cbi(PORTD, SDA);  delay_2nop();   // SDA ← L
      sbi(PORTD, SCL);  delay_2nop();   // SCL ← H
      sbi(PORTD, SDA);  delay_us(1);      // SDA ← H
 }
```





Project #14(Pjt14_I2C_TCN75)

```
inline uint8_t  i2c_tcn75_write_one_byte(uint8_t data)
{
    uint8_t    mask, ack;

    sbi(DDRD, SDA);                                  // SDA output(write)

    for (mask = 0x80; mask; mask = mask >> 1) {
        cbi(PORTD, SCL);  delay_2nop();              // SCL ← L
        if (mask & data) sbi(PORTD, SDA);            // SDA ← 1
        else             cbi(PORTD, SDA);            // SDA ← 0
        delay_us(1);
        sbi(PORTD, SCL); delay_us(1);                // SCL ← H
     }
     cbi(DDRD,  SDA);                                // SDA input(read)

     cbi(PORTD, SCL); delay_us(1);                   // SCL ← L
     sbi(PORTD, SCL); delay_us(1);                   // SCL ← H

     ack = PIND & (1 << SDA);                        // ack ← SDA } read  ack of TC75
                                                                   } must be 0
    return(ack);
}
```

Project #14(Pjt14_I2C_TCN75)

```
inline uint8_t  i2c_tcn75_read_one_byte(uint8_t ack)
{
      uint8_t   i, data = 0;

      cbi(DDRD, SDA);                                          // SDA input

      for (i = 0; i < 8; i++) {
            cbi(PORTD, SCL); delay_us(1);                      // SCL ← L
            sbi(PORTD, SCL); delay_us(1);                      // SCL ← H
            data = (data << 1);
            if (PIND & (1 << SDA)) data = data | 0x01;   // data ← SDA
      }
      sbi(DDRD, SDA);                                          // SDA output, for sending ack
      cbi(PORTD, SCL); delay_2nop();                           // SCL ← L

      if (ack) sbi(PORTD, SDA); else cbi(PORTD, SDA); // SDA ← 0 or 1 (Ack or Nck)
      delay_us(1);
      sbi(PORTD, SCL); delay_us(1);                            // SCL ← H

      return(data);
}
```

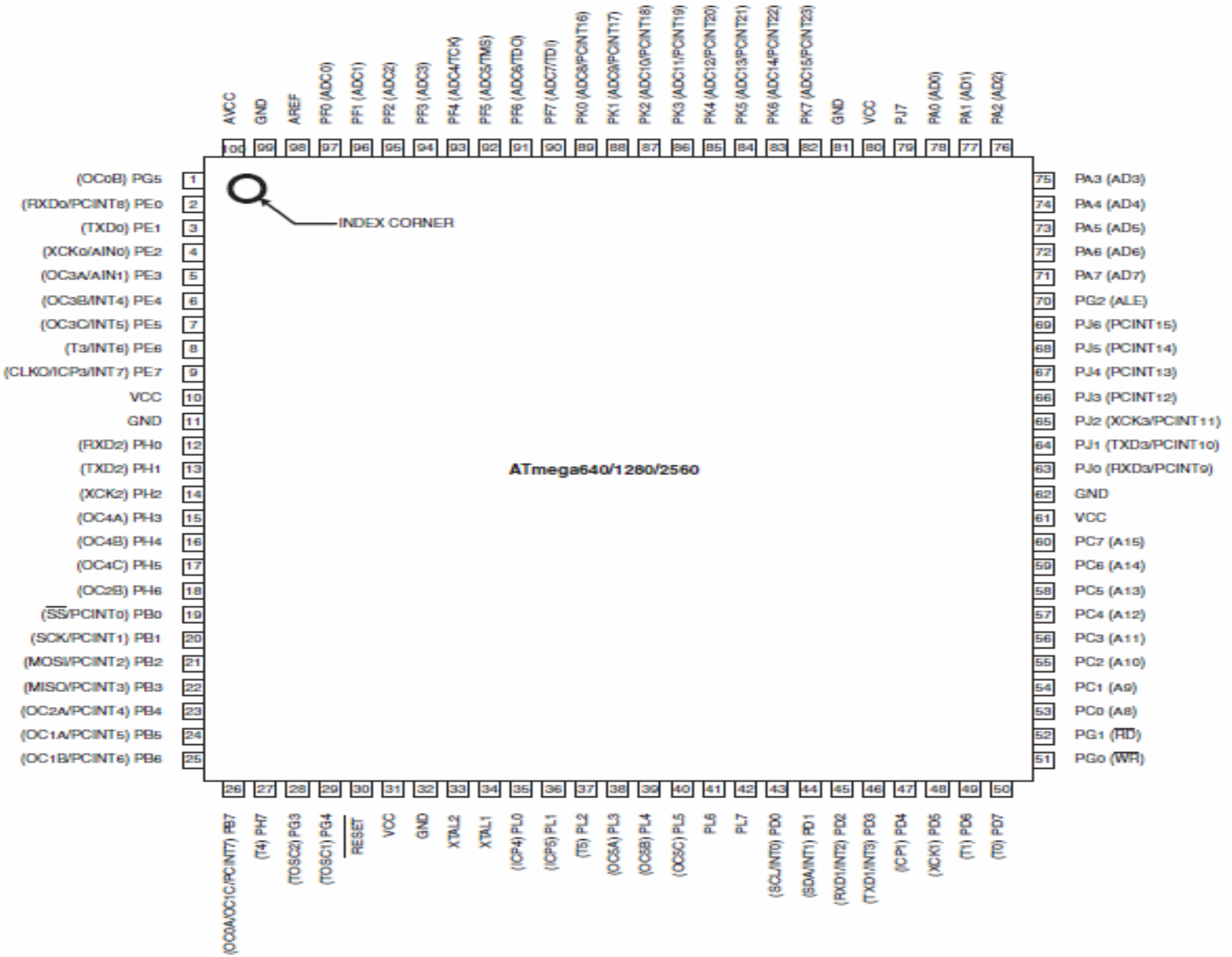
Project #14(Pjt14_I2C_TCN75)

- I2C Interface Sensor Application : task_tcn75

```
void  task_tcn75_i2c(voidr *arg)
{
      uint16_t  value;
      i2c_tcn75_trans_start();
      if (i2c_tcn75_write_one_byte(0x90) != 0) {     // address + write_operation
            i2c_tcn75_trans_stop();  printf("task_tcn75_i2c() : SLA+W write fail…₩n");
            return;
      }
      if (i2c_tcn75_write_one_byte(0x00) != 0) {     // pointer(TEMP)
            i2c_tcn75_trans_stop();  printf("task_tcn75_i2c() : pointer write fail…₩n");
            return;
      }
      i2c_tcn75_trans_start();                              // Repeat Start
      if (i2c_tcn75_write_one_byte(0x91) != 0) {      // address + read_operation
            i2c_tcn75_trans_stop();  printf("task_tcn75_i2c() : SLA+R write fail…₩n");
            return;
      }                                                                                  // read TEMP register
      value  = ((i2c_tcn75_read_one_byte(0) << 8) | i2c_tcn75_read_one_byte(1)) >> 7;
      i2c_tcn75_trans_stop();
      value = value >> 1;                              // value = value * 0.5
      printf("task_tcn75_i2c() : current_temperature ➔ %d degree.₩n", value);
}
```

Project #14(Pjt14_I2C_TCN75)

```
void task_cmd(void *arg)
 {
        …
        …
        else if (!strcmp(cp0, “tcn75_i2c”))
                task_tcn75_i2c(“”);
        …
        …
 }
```


ATmega640/1280/2560
INDEX CORNER
1 (OC0B) PG5
2 (RXD0/PCINT8) PE0
3 (TXD0) PE1
4 (XCK0/AIN0) PE2
5 (OC3A/AIN1) PE3
6 (OC3B/INT4) PE4
7 (OC3C/INT5) PE5
8 (T3/INT6) PE6
9 (CLKO/ICP3/INT7) PE7
10 VCC
11 GND
12 (RXD2) PH0
13 (TXD2) PH1
14 (XCK2) PH2
15 (OC4A) PH3
16 (OC4B) PH4
17 (OC4C) PH5
18 (OC2B) PH6
19 (SS/PCINT0) PB0
20 (SCK/PCINT1) PB1
21 (MOSI/PCINT2) PB2
22 (MISO/PCINT3) PB3
23 (OC2A/PCINT4) PB4
24 (OC1A/PCINT5) PB5
25 (OC1B/PCINT6) PB6
26 (OC0A/OC1C/PCINT7) PB7
27 (T4) PH7
28 (TOSC2) PG3
29 (TOSC1) PG4
30 RESET
31 VCC
32 GND
33 XTAL2
34 XTAL1
35 (ICP4) PL0
36 (ICP5) PL1
37 (T5) PL2
38 (OC5A) PL3
39 (OC5B) PL4
40 (OC5C) PL5
41 PL6
42 PL7
43 (SCL/INT0) PD0
44 (SDA/INT1) PD1
45 (RXD1/INT2) PD2
46 (TXD1/INT3) PD3
47 (ICP1) PD4
48 (XCK1) PD5
49 (T1) PD6
50 (T0) PD7
51 PG0 (WR)
52 PG1 (RD)
53 PC0 (A8)
54 PC1 (A9)
55 PC2 (A10)
56 PC3 (A11)
57 PC4 (A12)
58 PC5 (A13)
59 PC6 (A14)
60 PC7 (A15)
61 VCC
62 GND
63 PJ0 (RXD3/PCINT9)
64 PJ1 (TXD3/PCINT10)
65 PJ2 (XCK3/PCINT11)
66 PJ3 (PCINT12)
67 PJ4 (PCINT13)
68 PJ5 (PCINT14)
69 PJ6 (PCINT15)
70 PG2 (ALE)
71 PA7 (AD7)
72 PA6 (AD6)
73 PA5 (AD5)
74 PA4 (AD4)
75 PA3 (AD3)
76 PA2 (AD2)
77 PA1 (AD1)
78 PA0 (AD0)
79 PJ7
80 VCC
81 GND
82 PK7 (ADC15/PCINT23)
83 PK6 (ADC14/PCINT22)
84 PK5 (ADC13/PCINT21)
85 PK4 (ADC12/PCINT20)
86 PK3 (ADC11/PCINT19)
87 PK2 (ADC10/PCINT18)
88 PK1 (ADC9/PCINT17)
89 PK0 (ADC8/PCINT16)
90 PF7 (ADC7/TDI)
91 PF6 (ADC6/TDO)
92 PF5 (ADC5/TMS)
93 PF4 (ADC4/TCK)
94 PF3 (ADC3)
95 PF2 (ADC2)
96 PF1 (ADC1)
97 PF0 (ADC0)
98 AREF
99 GND
100 AVCC

Project #14(Pjt14_I2C_TCN75)

- PIN Layout