<DTPML-SPI-151>

<DTPML-SPI-81>

- 비접촉 온도 측정
- 방사율 조절 가능
- IR refresh rate : 50Hz
- 원거리 온도 측정
- High Accuracy
- Digital Interface : SPI
- 레이저포인터 기본 장착
- AVR / 아두이노 UNO 예제코드 제공

▶ 제품 설명

- DTPML-SPI Series는 온도계산 프로세서를 내장하고 있어 정확한 온도 값을 출력합니다. (Master Controller에 온도계산 알고리즘이 필요하지 않습니다.)
- 방사율 조절이 가능합니다(초기값 0.97ε 으로 출하).
- DTPML-SPI Series는 디지털 통신(SPI)으로 온도값을 출력합니다.
- 센서 온도와 대상 온도를 동시에 측정합니다.
- 레이저 포인터를 장착하여 측정 방향을 쉽게 알 수 있습니다.
- Warning: This DTPML contains a class II laser device.(650nm)

▶ 특징

- 8:1 모델 측정 온도 구간 : -20℃ ~ 200℃
- 15:1 모델 측정 온도 구간 : -20℃ ~ 270℃
- 동작 온도 구간 : -20℃ ~ 70℃
- 동작 온도(레이저) : -10℃ ~ 40℃
- 분해능 : 0.1℃
- DTPML-SPI-81 DS ratio : 8:1
- DTPML-SPI-151 DS ratio : 15:1
- 정확도 : ±2%
- 입력 전압 : 3.3V
- 통신 인터페이스 : SPI

▶ 응용분야

- 과열방지 시스템
- 산업용 온도 측정 장치
- 체온 측정을 통한 인체 감지
- 가전기기

Absolute Maximum Ratings

- Supply voltage : 3.5V
- Operating Temperature Range : -20℃ ~ 70℃
- Storage Temperature Range : -40℃ ~ 85℃

위 조건을 넘어서게 되면 제품의 수명을 보장할 수 없습니다.
반드시 아래 Electrical Requirements 를 지켜주세요.

Electrical Requirements

Parameter	Symbol	Conditions	min	Typ	Max	Unit
공급전압	Vcc	Measured versus GND	3.1	3.3	3.5	V
방사율(Emission Coefficient)	ε		0.1		1	
소비 전류 (3.3V 기준)		Full ambient temp. range, Typical value, no output load		11.5	12	mA
		On Laser		37.89		mA
SPI Clock			0.1		1	MHz
SPI INPUT High Level			**3.1**		**3.5**	V
SPI INPUT Low Level					**0.9**	V
SPI OUTPUT High Level			**Vcc-0.3**		**Vcc**	V
SPI OUTPUT LOW Level			**Vss**		**Vss+0.3**	V

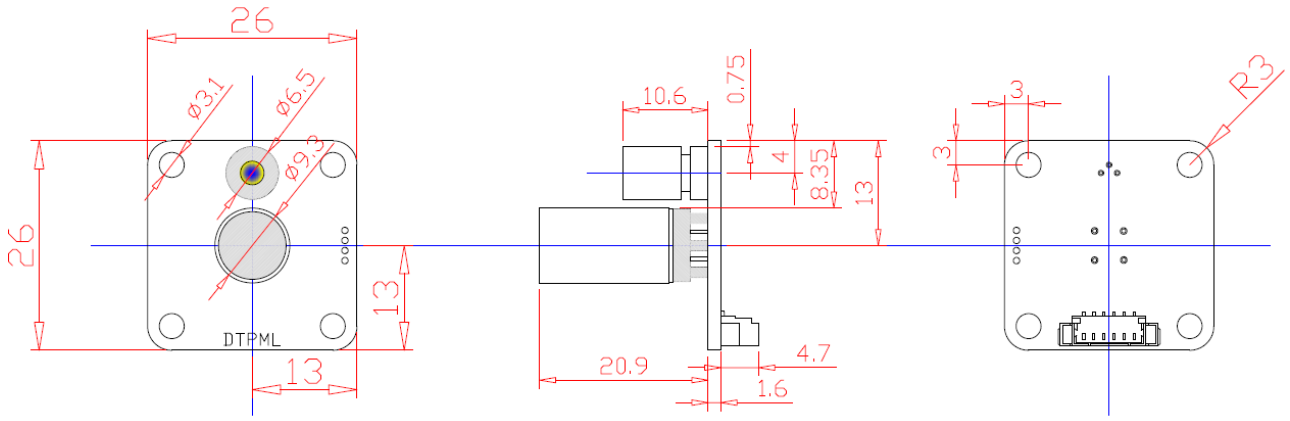
Operational Characteristics

- if not otherwise noted, 25℃ ambient temperature, 3.3V supply voltage and object with ε =0.97 were applied

Parameter	Symbol	Conditions	min	Typ	Max	Unit
DS ratio : DTPML-SPI-81 DTPML-SPI-151				8:1 15:1		
온도측정범위: DTPML-SPI-81 DTPML-SPI-151	Tobj		-20 -20		200 270	℃
동작온도(주변온도)	Tamb		-20		70	℃
동작온도(레이저동작)			-10		40	℃
온도측정 시간	Fout			20		msec
정확도	AccT			±2		%
Resolution Digital				0.1		℃
Standard Start-UP Time	tStart			1		sec
Stabilization Time	tStab			1		min

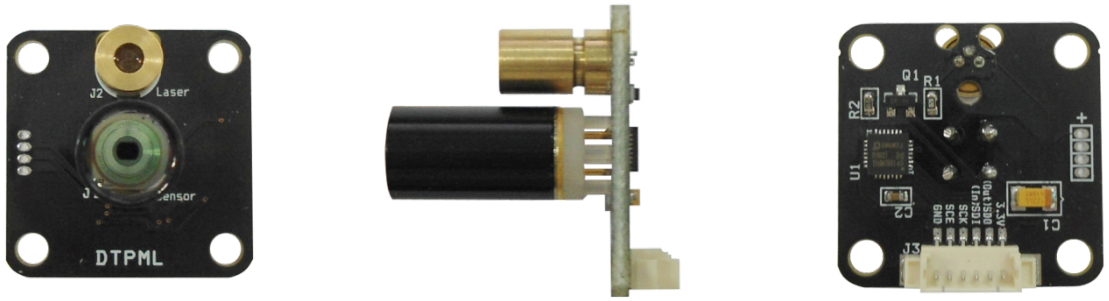
▶ Mechanical Dimensions (DTPML-SPI-151)

단위 : mm



< TOP View > < SIDE View > < BOTTOM View >

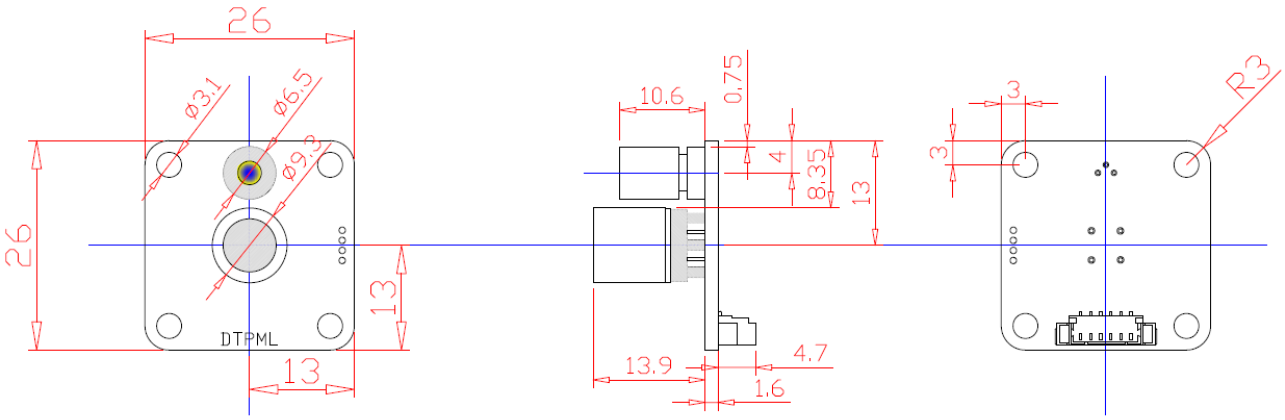
▶ 제품 사진 (DTPML-SPI-151)



< TOP View > < SIDE View > < BOTTOM View >

▶ Mechanical Dimensions (DTPML-SPI-81)

단위 : mm



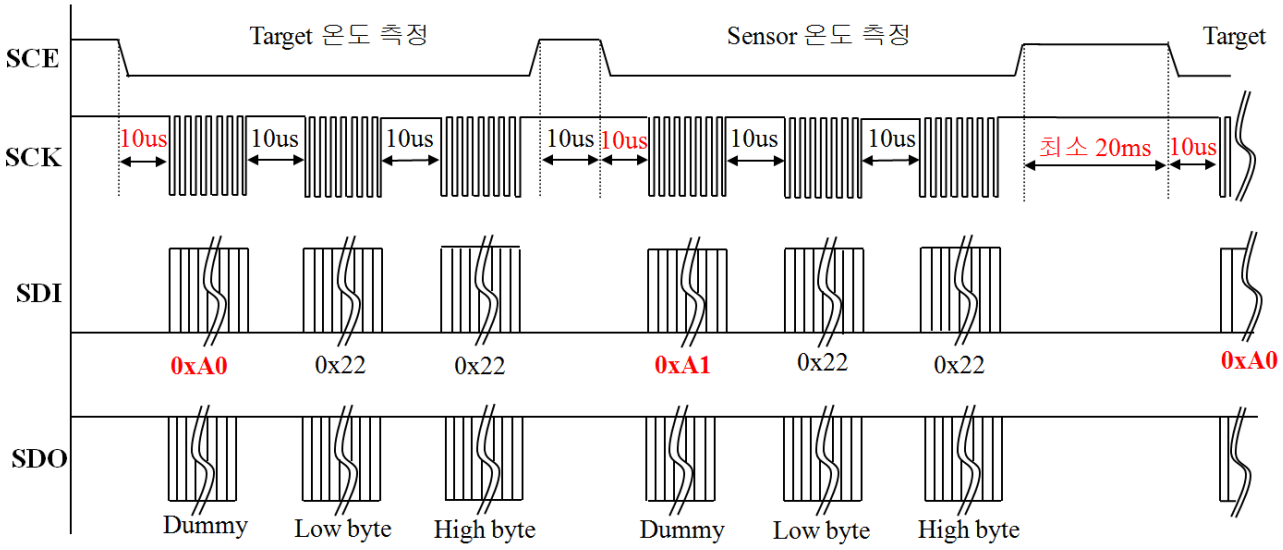
< TOP View > < SIDE View > < BOTTOM View >

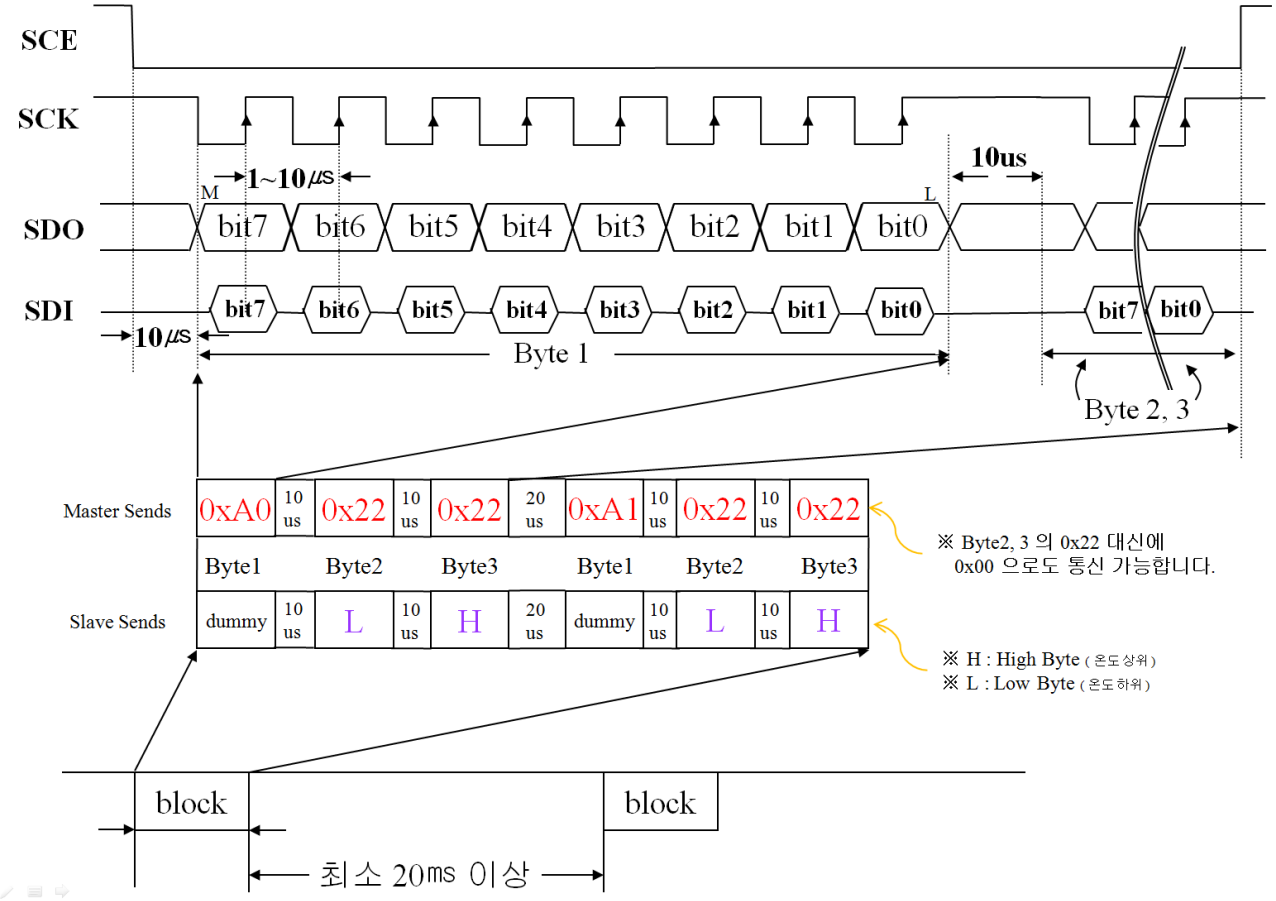
▶ 제품 사진 (DTPML-SPI-81)

< TOP View > < SIDE View > < BOTTOM View >

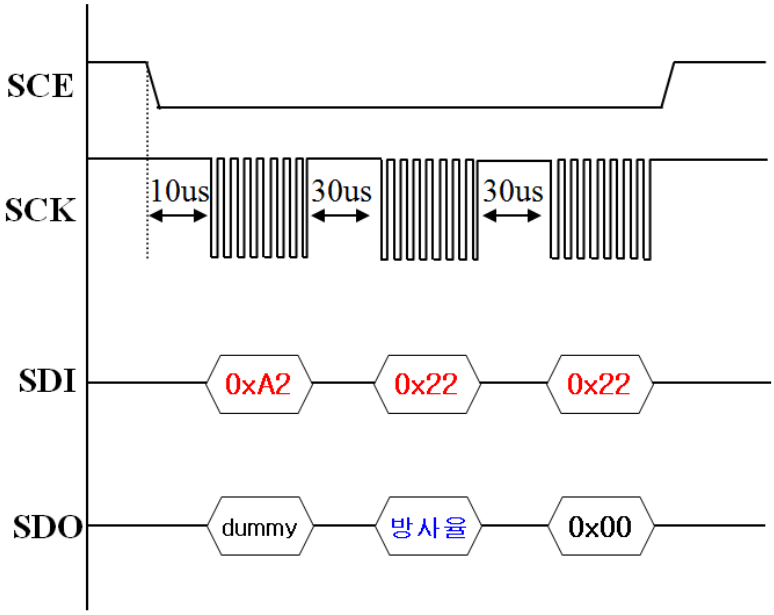
SPI Communication and Timings :

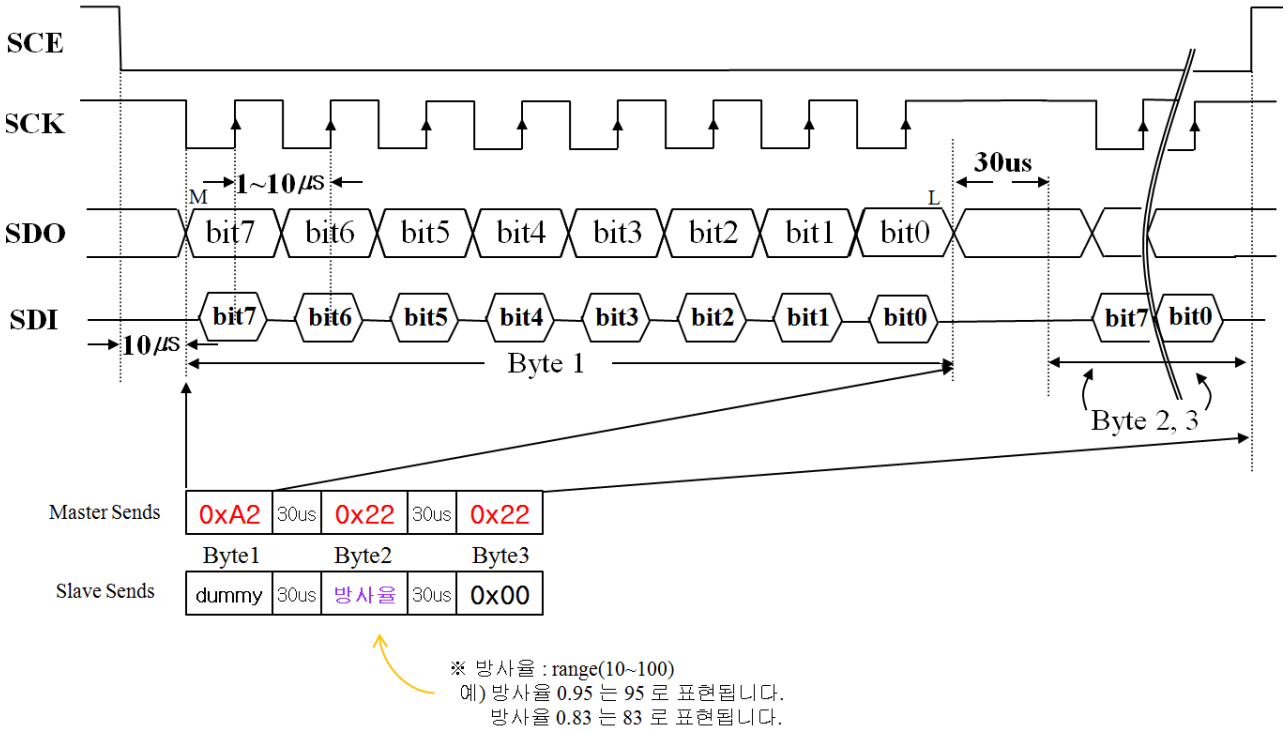
- **Target & Sensor 온도 읽기**





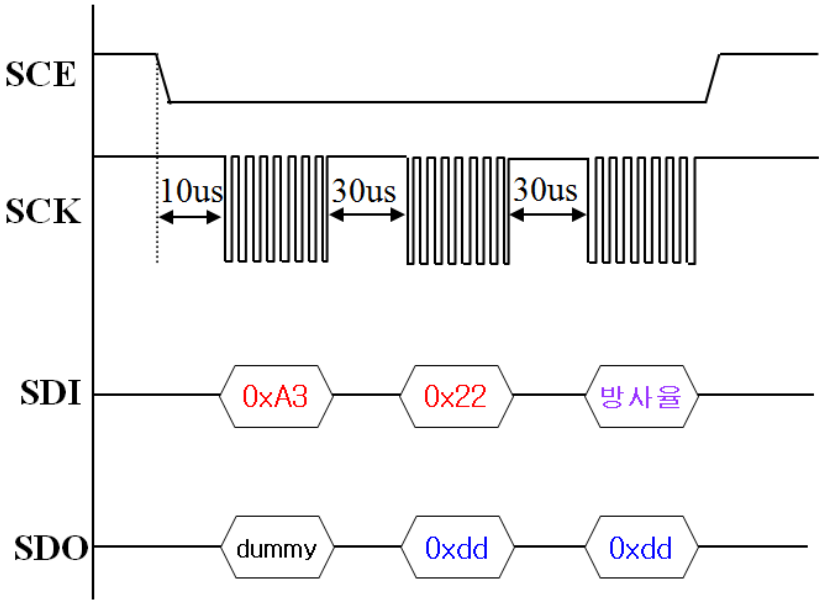
- **방사율 READ**

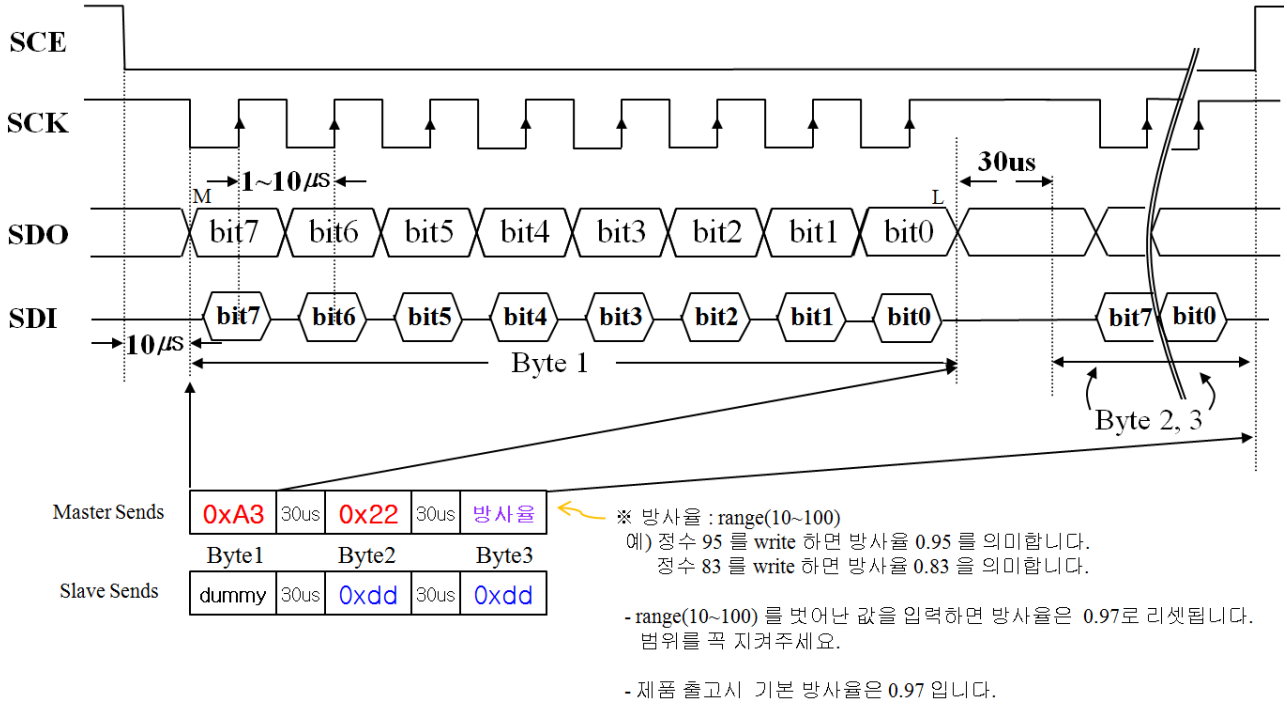




※ 방사율 : range(10~100)
예) 방사율 0.95 는 95 로 표현됩니다.
방사율 0.83 는 83 로 표현됩니다.

- **방사율 WRITE**



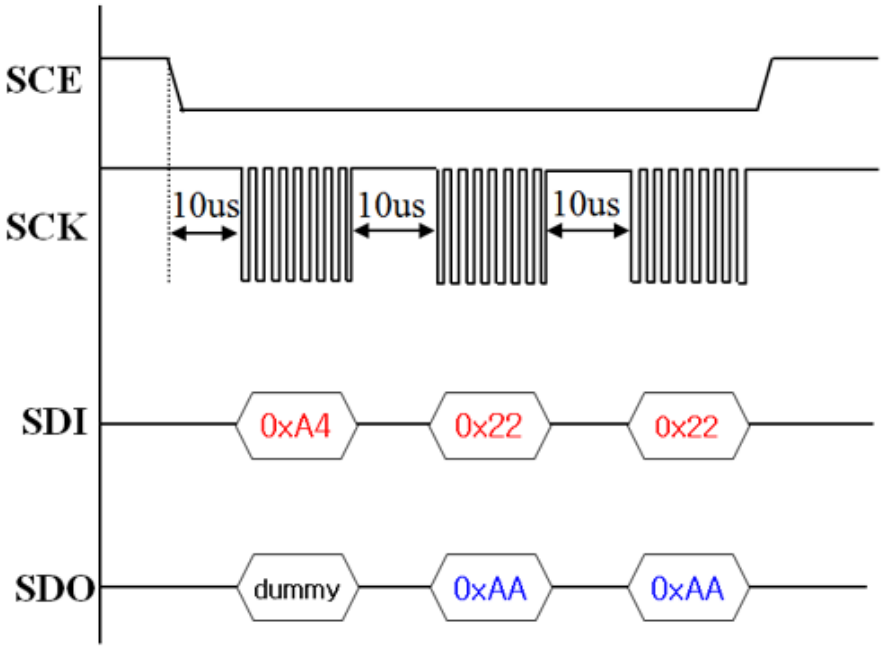


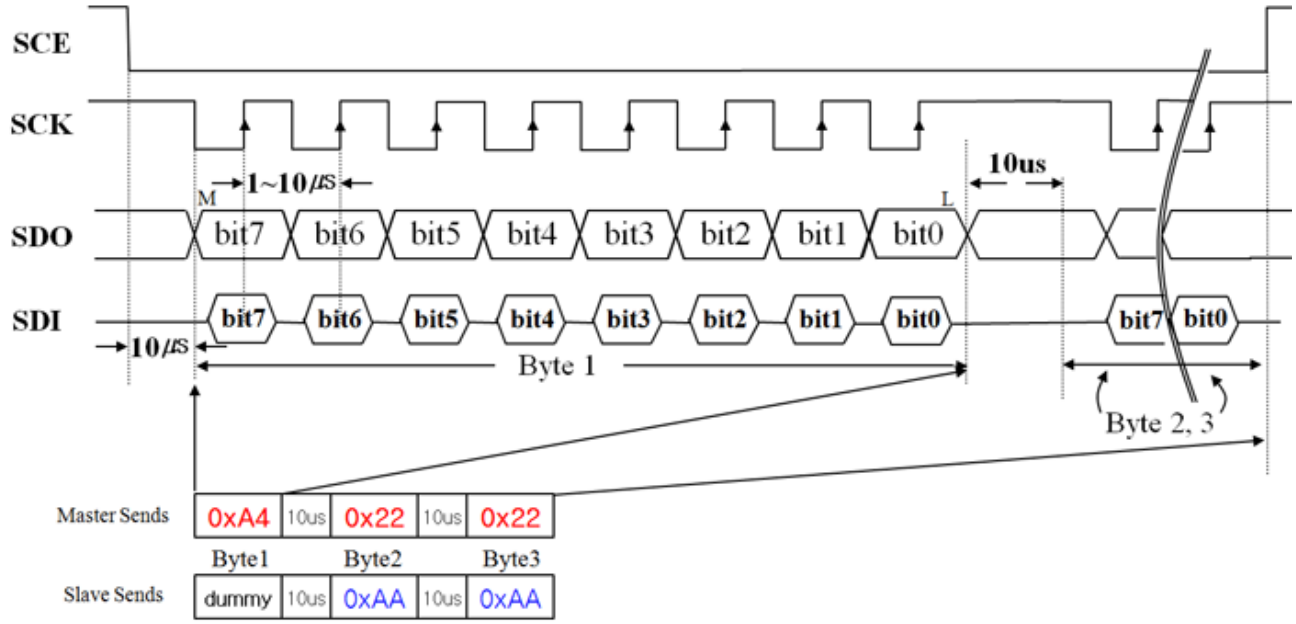
※ 방사율 : range(10~100)
예) 정수 95 를 write 하면 방사율 0.95 를 의미합니다.
정수 83 를 write 하면 방사율 0.83 을 의미합니다.

- range(10~100) 를 벗어난 값을 입력하면 방사율은 0.97로 리셋됩니다.
범위를 꼭 지켜주세요.

- 제품 출고시 기본 방사율은 0.97 입니다.

- **레이저 ON**





※ 레이저를 눈으로 바라보지 마십시오.

※ 레이저는 명령 후 약 15~20 초간 동작합니다. 레이저가 꺼지기 전에 반복 명령을 주면 마지막 Command 을 기준으로 15~20 초간 동작합니다.

※ 레이저 포인터는 1~2m 기준 중심에서 10cm 이내로 포인트 오차가 있을 수 있습니다. 가리키는 방향을 참고하시기 바랍니다.

▶ 온도 계산 방법

- **영상온도 계산(주의 : 결과값에 10을 나눠주면 됩니다.)**

Target	0x6D	0x01

Sensor	0xFA	0x00

*** 타겟 온도 계산 :** 상위 Byte(0x01) + 하위 Byte(0x6D) = 0x016D
=> 365(HEX → 10진수) 즉 36.5 도입니다.
*** 센서 온도 계산 :** 상위 Byte(0x00) + 하위 Byte(0xFA) = 0x00FA
= >250 (HEX → 10진수) 즉 25.0 도입니다.

- **영하온도 계산(영하(0도 미만)일 때는 2의 보수 값으로 전송됩니다.)**

Target	0xF1	0xFF

Sensor	0x9C	0xFF

*** 타겟 온도 계산 :** 상위 Byte(0xFF) + 하위 Byte(0xF1) = 0xFFF1
0xFFF1 = 1111 1111 1111 0001 (1의 보수 + 1 의 연산을 합니다)
0000 0000 0000 1110 ← (1의 보수)
0000 0000 0000 1111 = 0x000F ← (1의 보수 +1)
0x000F = 15 즉, -1.5도 입니다.
*** 센서 온도 계산 :** 상위 Byte(0xFF) + 하위 Byte(0x9C) = 0xFF9C => 즉, -10.0도입니다.

▶ 방사율 READ

- DTPML 모듈의 방사율 설정을 읽어옵니다. 수치는 방사율 x 100 입니다.
- 예) 읽어온 값이 hex : 0x61(dec : 97) 이면 방사율 0.97를 의미합니다.

- Request : 0xA2, 0x22(or 0x00), 0x22(or 0x00)
- Response : dummy, 방사율, 0x00
- 데이터 범위 : 1~100
- page21. 방사율표 참고

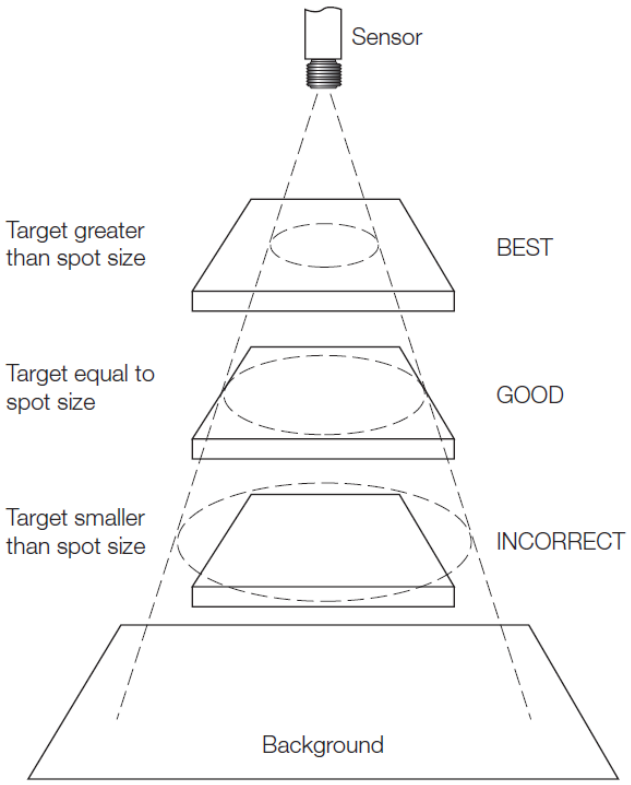
▶ 방사율 WRITE

- DTPML 모듈에 방사율을 저장합니다. 수치는 방사율 x 100 입니다.
- 예) 방사율 0.97를 쓰고자 하면 hex: 0x61 (dec:97) 값을 통신하면 됩니다.

- Request : 0xA3, 0x22(or 0x00), 방사율
- Response : dummy, 0xdd, 0xdd
- 데이터 범위 : 1~100
- page21. 방사율표 참고

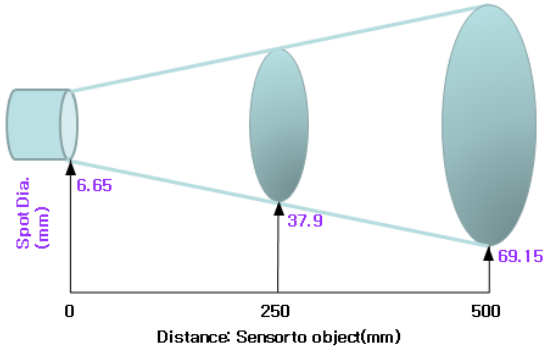
▶ DISTANCE AND SPOT SIZE

Spot Size는 아래 그림에서와 같이 측정하고자 하는 대상의 면적보다 반드시 작아야 합니다.

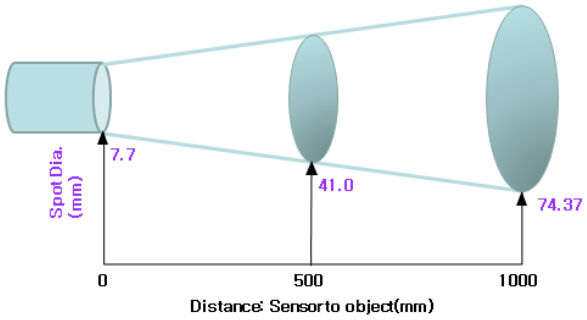


Optical field of view (FOV)

The optical chart below indicates the nominal target spot diameter at any given distance from the sensing head and assumes 50% energy.



< DTPML-SPI-81 >



< DTPML-SPI-151 >

Pin Assignment

number	Name	Description	Type
1	GND	Ground	Ground
2	SCE	ENABLE	Input
3	SCK	CLOCK	Input
4	SDI	Signal Input	Input
5	SDO	Signal Output	Output
6	VCC	Supply Voltage 3.3V	Supply

- Connector : (Header : molex 0533980671), (Female : molex 51021-0600)

Ordering Guide

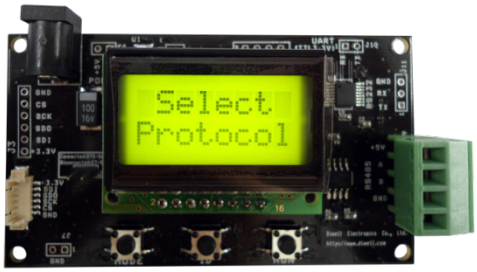
DTPML – △△△ – ◇◇◇

Laser		Protocol		DS ratio		측정 온도 범위
L	Laser 장착	485	Modbus 485 RTU	81	8:1	-20℃ ~ 200℃
		SPI	SPI	151	15:1	-20℃ ~ 270℃

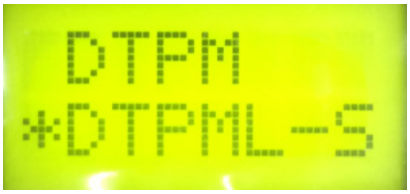
예) DTPML-SPI-151 - 레이저장착, 15:1 DS ratio, SPI 통신 모델을 의미합니다.

▶ CT-Testboard-Plus 테스트 보드(별도 구매)

- 손쉽게 측정하여 온도값을 다른 device(MCU, PC, embedded등) 로 전송(RS-232)할 수 있습니다.



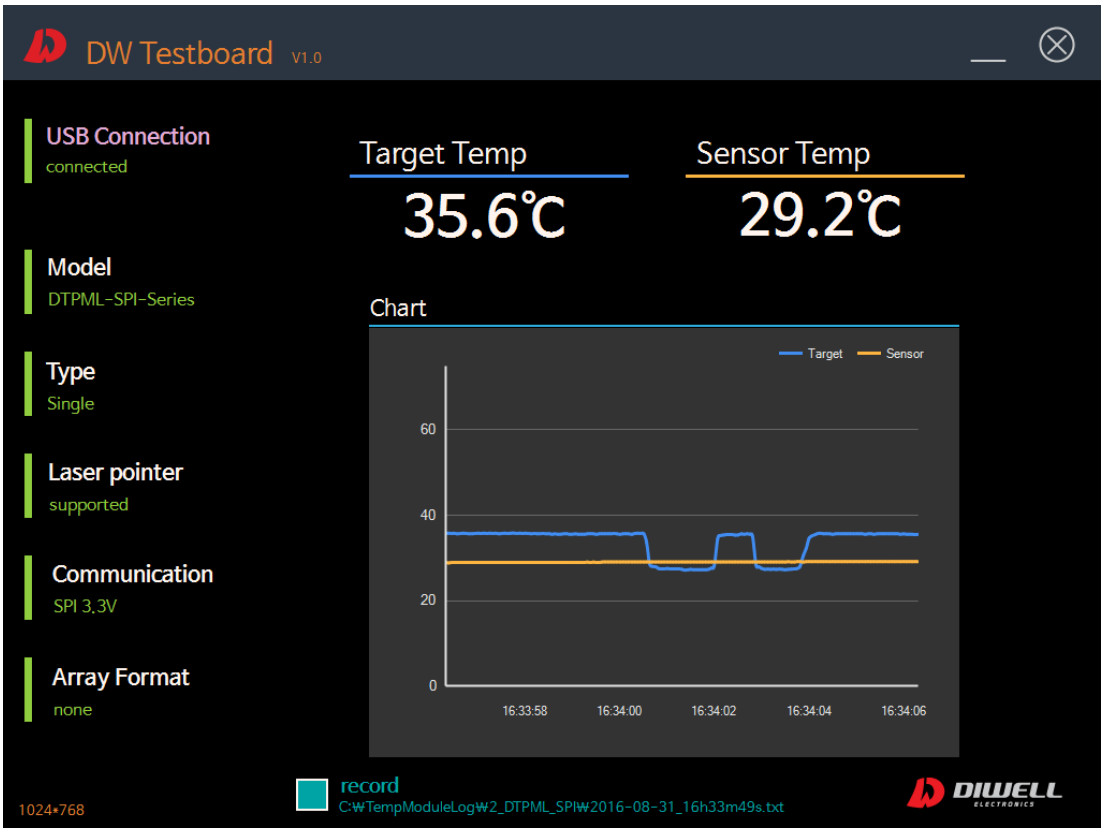
- 테스트 보드에서 레이저 켜는 법 : 온도 측정중에 "ID" 버튼을 누르면 레이저가 켜집니다.
  - 조건1. 연결된 측정모듈이 DTPML-SPI series
  - 조건2. 테스트보드 Protocol 선택 메뉴에서 "DTPML-S" 이 있는 버젼



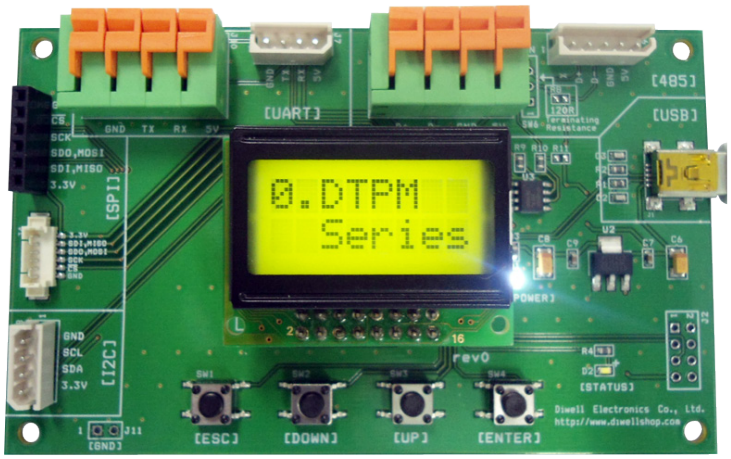
< DTPML 메뉴 화면 >

▶ DW Testboard (신제품 - 별도 구매)

- PC와 연결하여 온도 측정/ 실시간 기록이 가능한 테스트보드 신제품이 출시됐습니다.



< PC 화면 >



< DW Testboard - 신제품>

▶ Appendix - A (Example Code - ATMEGA128L 이용)

아래 소스코드는 DTPML 모듈과 통신을 위한 참고용 소스 코드 중 온도측정 부분입니다.
풀 코드는 아래 주소를 통해 프로젝트 다운로드가 가능합니다.
(컴파일 환경 : AVRSTUDIO 4.19, gcc 4.3.3, WinAVR-20100110)
Download : http://www.diwell.com/base_1/img/download/DTPML-SPI-MASTER_Atmega128L_8Mhz.zip

SPI 초기값 세팅 <AVR SPI MODE = MODE 3 에 해당 >
- Clock 주파수 최대 1Mhz
- Internal SPI Clock(Master Mode)
- SCK data transfer edge : Rising Edge
- MSB first data transfer
- SCK idle status : High

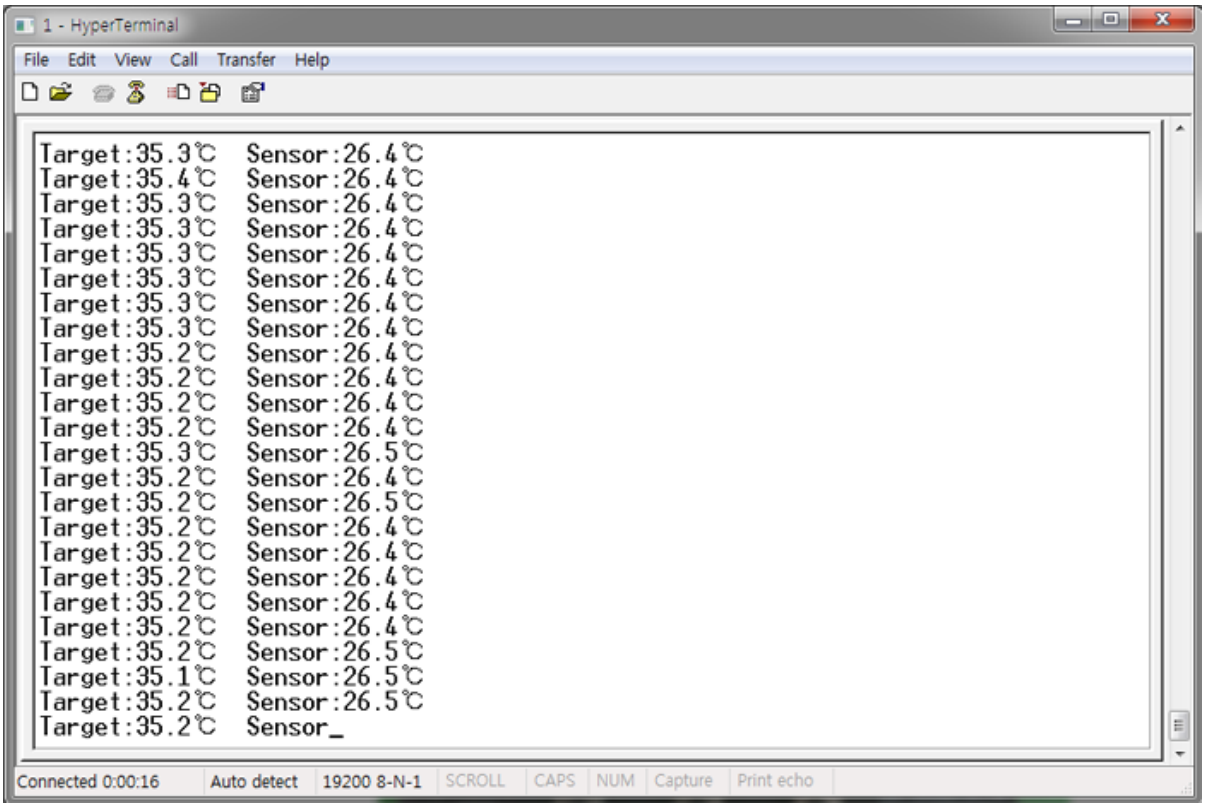
사용하는 MCU 환경에 따라 코드는 달라지므로 내용을 이해 하신 후 적용하고자 하는 MCU에 적용하시면 됩니다.

```
#include <avr/io.h>
#include <stdio.h>
#include <avr/interrupt.h>

#define CS_HIGH PORTB |=0x01;                    // SCE 1
#define CS_LOW  PORTB &=0xFE;                    // SCE 0
#define TARGET_CMD      0xA0                     // 대상 온도 커맨드
#define SENSOR_CMD      0xA1                     // 센서 온도 커맨드
#define LASER_CMD       0xA4                     // 레이저 커맨드

#define  ON       1

int         iTARGET, iSENSOR;                          // 부호 2byte 온도 저장 변수
unsigned char LASER_COMMAND=0;
unsigned char dummy;
```

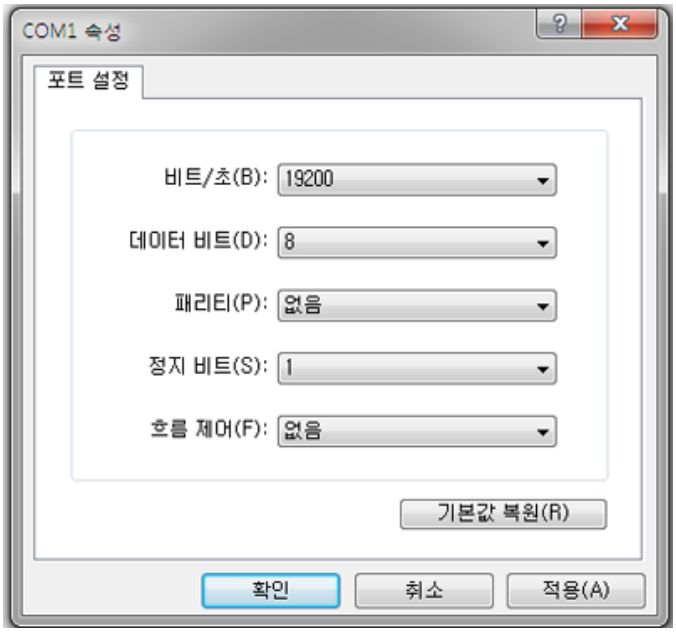
```
void SPI_MasterTransmit(char cData);
int SEND_COMMAND(unsigned char cCMD);
void DataOutput_UART0(void);

ISR(INT0_vect)                          // INT0 인터럽트 루틴 (PORT D.0 스위치 누르면 진입)
{
        EIMSK = 0x00;                   // INT0 인터럽트 Disable
        LASER_COMMAND = ON;
}

int main(void)
{
        Device_Init();                  // INT0, SPI 초기화

        CS_HIGH;                        // CS High Level
        _delay_ms(200);                 // 모듈 setup 시간 wait
        _delay_ms(200);
        _delay_ms(200);
        _delay_ms(200);
        EIFR |= 0x01;
        EIMSK = 0x01;                   // INT0 인터럽트 개별 허용
        sei();                          // 전체 인터럽트 Enable

        while(1)
        {
                iTARGET = SEND_COMMAND(TARGET_CMD);     // 대상 온도 Read
                _delay_10us();                          // 10us : 이 라인을 지우지 마세요
                iSENSOR = SEND_COMMAND(SENSOR_CMD);     // 센서 온도 Read
                _delay_ms(20);                          // 20ms : 이 라인을 지우지 마세요.

                //DataOutput_UART0();

                if(LASER_COMMAND == ON)                 // 측정중에 버튼을 눌렀으면
                {
                        LASER_COMMAND = 0;
```

```
                    SEND_COMMAND(0xA4);                    // 레이저 ON 명령
                    EIFR |= 0x01;              // 플래그 클리어
                    EIMSK = 0x01;              // INT0 인터럽트 Enable
                }
            }
            return 0;
}

void INT0_Init(void)
{
            PORTD = 0x01;                    // 내부 풀업
            DDRD = 0x00;
            EICRA = 0x02;                    // 하강 엣지
            EIMSK = 0x00;                    // INT0 인터럽트 불능
}
void SPI_MasterInit(void)
{
            PORTB = 0xff;
            DDRB = 0x07;    //MISO(PB3): INPUT // MOSI(PB2) : OUTPUT
                            //SCK(PB1) : OUTPUT // PB0(모듈 SCE 제어) : OUTPUT
            SPCR = 0x5d;
            SPSR = 0x01;
}
void Device_Init(void)
{
            INT0_Init();
            SPI_MasterInit();            // SPI 초기화
}

void SPI_MasterTransmit(char cData)
{
            /* Start transmission */
            SPDR = cData;
            /* Wait for transmission complete */
            while(!(SPSR & (1<<SPIF)));
}
```

```
int SEND_COMMAND(unsigned char cCMD)
{
        unsigned char Low_Byte, High_Byte;

        CS_LOW;                         // SCE LOW

        _delay_10us();                  // delay 10us (반드시 지켜주세요)

        SPI_MasterTransmit(cCMD);       // Send 1st Byte
        dummy = SPDR;                   // 첫 데이터 dummy 처리

        _delay_10us();                  // delay 10us (필수)

        SPI_MasterTransmit(0x00);       // Send 2nd Byte 0x22 or 0x00
        Low_Byte = SPDR;                // 하위 바이트 저장

        _delay_10us();                  // delay 10us (필수)

        SPI_MasterTransmit(0x00);       // Send 3rd Byte 0x22 or 0x00
        High_Byte = SPDR;               // 상위 바이트 저장

        CS_HIGH;                        // SCE HIGH

        return (High_Byte<<8 | Low_Byte); // 상위, 하위 바이트 연산
}
                                // ms 는 delay.h 파일참고하거나 다운받은 프로젝트를 참고하세요.
void _delay_10us(void)          // 8Mhz 클럭에서 10us 딜레이
{
        volatile unsigned char i;
        for(i=0;i<4;i++)
        {
                asm volatile(" NOP ");
        }
        asm volatile(" NOP ");
}
```



< 하이퍼 터미널을 통해 출력 >



< 통신 설정 >

※ COM 넘버는 다를 수 있습니다.

▶ Appendix - B (Example Code - I/O)

하단 코드는 컨트롤러에서 SPI 레지스터 설정이 아닌 I/O 포트 제어를 통해 통신하는 예제 코드입니다.
포트 설정/ 포트 상태 읽는 코드는 사용하고자 하는 MCU에 맞게끔 수정하셔야 합니다.(現, sonix MCU)
I/O 포트 전압 레벨이 3.3V 인지를 꼭 확인하세요.

- **SPI.H**

```
#define SCK_HIGH        FP16=1
#define SCK_LOW         FP16=0
#define SDO_HIGH        FP14=1
#define SDO_LOW         FP14=0
#define EN_HIGH          FP17=1
#define EN_LOW           FP17=0
long CHECK(unsigned char datum);
```

- **Main.C**

```
#include "SN8F27E65.h"
#include "delay.h"
#include "SPI.H"
Long Target_Value, Sensor_Value;
 // sonix 컴파일러는 long이 2byte 입니다. 해당하는 컴파일러에 맞게 2byte변수 선언하세요
void main(void)
{
    While(1)
    {
        Target_Value = CHECK(0xa0);   // 대상온도
        delay_us(10);
        Sensor_Value = CHECK(0xa1)   // 센서온도
        delay_ms(20);
        // LCD View CODE here
    }
}
```

- **SPI.C**

```
unsigned char buffer_Lo, buffer_Hi, p02; //1byte 선언

long CHECK(unsigned char datum)   //2 byte return 함수
{
      unsigned char i=0;
      buffer_Lo=0;
      buffer_Hi = 0;
      EN_LOW;
      delay_us(10);
      for(i=0; i<8; i++)
      {
            if(((0x80 >>i)&datum)==0){SDO_LOW;}
            else {SDO_HIGH;}
            SCK_LOW;
            delay_us(1);
            SCK_HIGH;
            delay_us(1);
      }
      SDO_LOW;                    // 0x22 or 0x00 전부 가능합니다만 코드간결을 위해 0x00 사용
      delay_us(10);

      for(i=0; i<8; i++)              // Low byte read
      {
            buffer_Lo = buffer_Lo <<1;
            SCK_LOW;
            delay_us(1);
            SCK_HIGH;
            delay_us(1);
            p02=FP02;      // 포트의 상태 읽는 문장
            if(p02 == 1){buffer_Lo = buffer_Lo|0x01;}
            else{buffer_Lo = buffer_Lo&0xFE;}
       }

         SDO_LOW;
         delay_us(10);

      for(i=0; i<8; i++)                // High byte read
       {
           buffer_Hi = buffer_Hi <<1;
           SCK_LOW;
           delay_us(1);
           SCK_HIGH;
           delay_us(1);
           p02=FP02;      // 포트의 상태 읽는 문장
           if(p02 == 1){buffer_Hi = buffer_Hi|0x01;}
           else{buffer_Hi = buffer_Hi&0xFE;}
       }
       EN_HIGH;
       return (buffer_Hi*256+buffer_Lo);
}
```

▶ Appendix - C (방사율표)

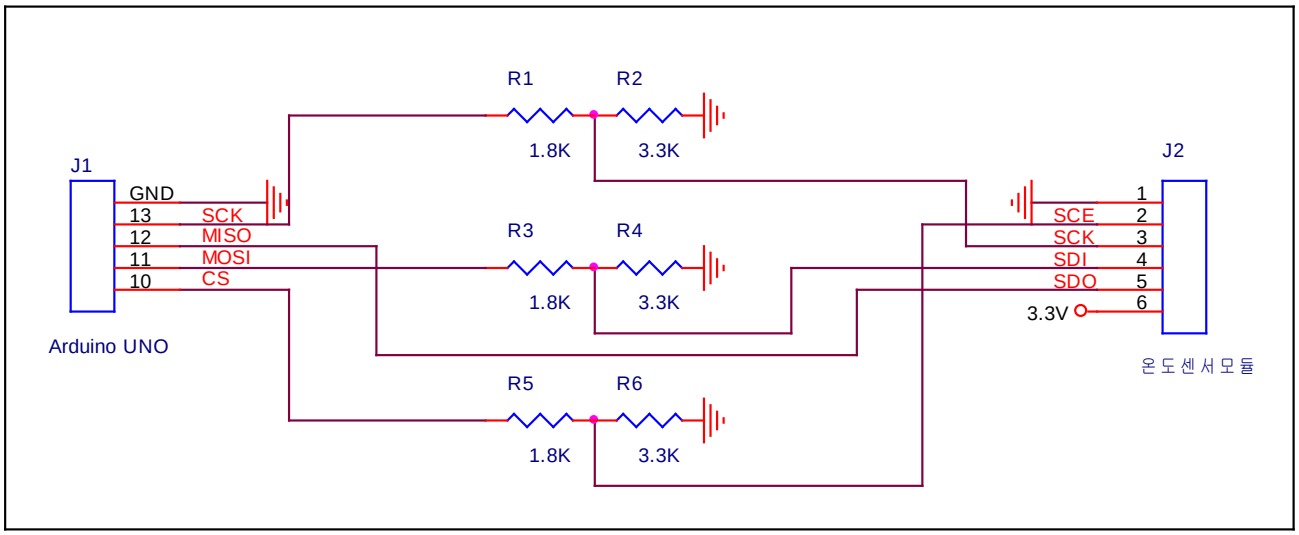
방사율이란 물체가 외부 적외선 에너지를 흡수, 투과 및 반사하는 비율을 말하는데,
이론적으로 외부에너지를 흡수만 하고 반사하지 않는 물체를 흑체라 하여 이때의 방사율은 "1"입니다.
하지만, 일반적으로 물체의 표면상태(광택, 거침, 산화여부 등) 에 따라 흡수, 반사하는 에너지량이 변합니다. 재질에 따른 방사율 값은 하단의 "방사율표" 를 참고하여 변경하면 됩니다. 단, "방사율표"의 값은 절대적인 값이 아니며 표면 상태와 그 외 복합적인 환경 요인에 따라 오차가 있을 수 있으니, 이점 참고하십시오.

대상	*방사율*	대상	*방사율*	대상	*방사율*
산화아연	**0.1**	에나멜	**0.9**	구리(연마된)	**0.5**
아연도금철	**0.3**	페인트	**0.95**	구리(산화된)	**0.8**
주석도금철	**0.1**	라 카	**0.9**	니켈(순수)	**0.1**
금(연마된)	**0.1**	고무(smooth)	**0.9**	니켈(산화된)	**0.4~0.5**
은(연마된)	**0.1**	고무(Rough)	**0.98**	니켈크롬	**0.7**
크롬(연마된)	**0.1**	플라스틱	**0.8~0.95**	니켈크롬(산화된)	**0.95**
붉은 벽돌	**0.75~0.9**	플라스틱필름	**0.5~0.95**	직물	**0.9**
흙	**0.92~0.96**	주철(연마)	**0.2**	피부	**0.98**
석면	**0.95**	Steel	**0.6**	가죽	**0.75~0.8**
콘크리트	**0.7**	산화 Steel	**0.9**	얼음	**0.96~0.98**
대리석	**0.9**	목재	**0.8~0.9**	모래	**0.9**
모르타르	**0.89~0.91**	스테인레스(연마된)	**0.1**	아스팔트	**0.9~0.98**
석고	**0.85**	스테인레스(기타)	**0.2~0.6**	유리	**0.8~0.9**
시멘트	**0.96**	알루미늄(연마된)	**0.1**	물	**0.8~0.9**
규토(정제된)	**0.4**	알루미늄(합금)	**0.1~0.25**	종이	**0.9**
세라믹	**0.90~0.94**	황동(연마된)	**0.1**	실리콘	**0.7**
석 영	**0.9**	황동(거친)	**0.2**	주철(부식된)	**0.95**
석 탄	**0.75**	황동(산화된)	**0.6**	Mild Steel	**0.3~0.5**
Fe(부식된)	**0.7~0.85**				

※ 측정하고자 하는 대상의 재질이 *코팅*이 돼 있거나 *반짝이는 재질*이라면 방사율을 수정 적용해도 온도 변화가 적을 수 있습니다. 이 때에는 측정 물체 표면에 무광의 락카 스프레이를 칠하면 됩니다.

▶ Appendix - D (5V MCU와 통신하기 위한 회로도)

※ 하단 회로도는 5V 의 MCU 와 연결하는 방법입니다. (예, 아두이노 우노(UNO)를 이용한 회로도)
Master MCU가 3.3V 일 경우는 저항 없이 직접 연결하면 됩니다.
아두이노 스케치 파일은 쇼핑몰의 DTPML 제품 상세페이지에서 다운로드 가능합니다.



▶ Appendix - E (Example Code - Arduino UNO)

하단 코드는 아두이노 UNO 코드 예제 입니다.

```
/**************************************************************************
*
* Copyright (C) 2016 Diwell Electronics Co.,Ltd.
* Project Name : (DTPML 시리즈) SPI Master Code
* Version : 1.1 (2016.05.09)
* SYSTEM CLOCK : 16Mhz
* BOARD : Arduino UNO. 5V operation
```

PORT Description

1. ChipSelectPin : 10
2. MOSI(Master Output) : 11
3. MISO(Master Input) : 12
4. SCK : 13

온도센서모듈 입력전원은 3.3V 로 하셔야 하며 포트 연결 방법은 회로도를 참고하십시오.
온도센서 통신포트의 논리 레벨은 3.3V 이기 때문에 반드시 회로도를 참고하시기 바랍니다.

Revision history.

1. 2016.5.4 : First version is released.
2. 2016.5.9 : 방사율 READ/WRITE 함수 추가(함수 내부 주의 사항 필독!)
3. 2016.5.9 : 레이저 ON 명령 함수 추가

**********/

```
#include<SPI.h>

#define TARGET_CMD      0xA0                    // 대상 온도 커맨드
#define SENSOR_CMD      0xA1                    // 센서 온도 커맨드
#define LASER_CMD         0xA4                    // 레이저 ON 커맨드

const int chipSelectPin   = 10;
unsigned char T_high_byte;
unsigned char T_low_byte;
unsigned char cEratio;   // 방사율 저장 변수
int   iTARGET, iSENSOR;    // 부호 2byte 온도 저장 변수
volatile unsigned char Laser_Flag=0;
```

```
void setup() {
  /* Initalize PORT */
  pinMode(MISO, INPUT);
  pinMode(chipSelectPin , OUTPUT);
  pinMode(MOSI, OUTPUT);
  pinMode(SCK, OUTPUT);
  Serial.begin(9600);

  /* Setting CS & SPI */
  digitalWrite(chipSelectPin , HIGH);      // CS High Level
  SPI.setDataMode(SPI_MODE3);                 // Setting SPI Mode
  SPI.setClockDivider(SPI_CLOCK_DIV32);   // 16MHz/16 = 1MHz
  SPI.setBitOrder(MSBFIRST);                  // MSB First
  SPI.begin();                                // Initialize SPI
  pinMode(2, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(2), LASER_ISR, FALLING);
  delay(500);                                   // wating for DTS setup time
}
void LASER_ISR(void)
{
  detachInterrupt(digitalPinToInterrupt(2));
  Laser_Flag =1;
}
int SEND_COMMAND(unsigned char cCMD)      // 온도 READ 함수
{
    digitalWrite(chipSelectPin , LOW);   // CS Low Level
    delayMicroseconds(10);                  // delay(10us)
    SPI.transfer(cCMD);                  // Send 1st Byte
    delayMicroseconds(10);                                  // delay(10us)
    T_low_byte = SPI.transfer(0x22);    // Send 2nd Byte
    delayMicroseconds(10);                                  //delay(10us)
    T_high_byte = SPI.transfer(0x22);   // Send 3rd Byte
    digitalWrite(chipSelectPin , HIGH); // CS High Level

    return (T_high_byte<<8 | T_low_byte);   // 상위, 하위 바이트 연산
}
```

```
void Ewrite_COMMAND(unsigned char cCMD)  // 방사율 WRITE 함수 (아래 주의 필독!!!)
{
  // 주의!!! : 방사율 WRITE 명령을 while 문 안에 넣어서 무한 반복으로 실행하면 안됩니다.
  // 방사율 WRITE 명령은 버튼이나 특정 조건에 의해 한번만 실행하도록 하세요.
  // 반복 플레시 작업시 제품 수명에 영향을 끼칩니다.
  // 한번 저장된 방사율은 전원 재 공급시에도 값을 유지합니다.
    unsigned char dummy;
    digitalWrite(chipSelectPin , LOW);   // CS Low Level
    delayMicroseconds(10);                  // delay(10us)
    SPI.transfer(0xA3);                  // Send 1st Byte
    delayMicroseconds(30);                  // delay(30us)
    dummy = SPI.transfer(0x22);          // Send 2nd Byte
    delayMicroseconds(30);                  //delay(30us)
    dummy = SPI.transfer(cCMD);          // Send 3rd Byte (Eratio)
    digitalWrite(chipSelectPin , HIGH); // CS High Level
    delay(500);                             // delay(500ms);
}

unsigned char Eread_COMMAND(void)   // 방사율 READ 함수
{
    unsigned char dummy, cDATA;
    digitalWrite(chipSelectPin , LOW);   // CS Low Level
    delayMicroseconds(10);                  // delay(10us)
    SPI.transfer(0xA2);                  // Send 1st Byte
    delayMicroseconds(30);                   // delay(30us)
    cDATA = SPI.transfer(0x22);          // Send 2nd Byte (Eratio)
    delayMicroseconds(30);;                 //delay(30us)
    dummy = SPI.transfer(0x22);          // Send 3rd Byte
    digitalWrite(chipSelectPin , HIGH); // CS High Level
    return cDATA;
}
```

```
void loop() {

  while(1) {

    iTARGET = SEND_COMMAND(TARGET_CMD);                    // 대상 온도 Read
    delayMicroseconds(20);                                 // 20us : 이 라인을 지우지 마세요

    iSENSOR = SEND_COMMAND(SENSOR_CMD);                    // 센서 온도 Read
    delayMicroseconds(20);                                 // 20us : 이 라인을 지우지 마세요

    cEratio = Eread_COMMAND();                             // 방사율 READ
    delay(20);                                             // 20ms : 이 라인을 지우지 마세요.

    if(Laser_Flag == 1)
    {
      Laser_Flag = 0;
      SEND_COMMAND(LASER_CMD);
      while(digitalRead(2)==LOW);
      attachInterrupt(digitalPinToInterrupt(2), LASER_ISR, FALLING);
    }

    Serial.print("Target Temp : ");
    Serial.print(float(iTARGET)/10);
    Serial.print("      Sensor Temp : ");
    Serial.print(float(iSENSOR)/10);
    Serial.print("      Eratio : ");
    Serial.println(float(cEratio)/100);
  }
}
```

Additional Information

- manufacturer : Diwell Electronics Co., Ltd. <(주)디웰전자>
- Homepage : www.diwell.com
- shopping mall : www.diwellshop.com
- Phone : +82-70-8235-0820
- Fax : +82-31-429-0821
- Technical support : expoeb2@diwell.com, dsjeong@diwell.com

DTPML Revision History

Version	Date	Description
1.0	2015-10-20	First version is released.
1.1	2015-12-22	기구도면 size 표기 오류 수정
1.2	2016-05-09	**방사율 R/W 프로토콜 delay 값 수정(10us→30us) <6, 7page>** **5V MCU와의 통신을 위한 회로도 추가 <21page>**
1.3	2016-08-31	**아두이노 UNO 예제 코드 추가** **DW Testboard 신제품 추가**