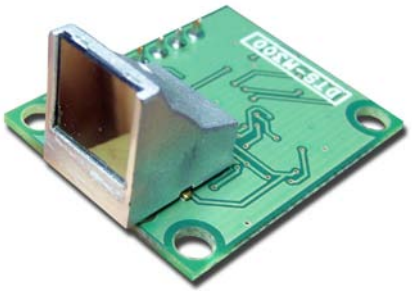


- 비접촉 온도측정
- 근거리 온도측정
- **Small Size**
- **High Accuracy**
- 디지털 인터페이스(SPI 프로토콜)

▶ 제품설명

- DTS-M300 은 적외선온도센서를 기반으로 한 접촉하지 않고 원하는 대상에 온도를 정확히 측정할 수 있는 온도센서 모듈입니다.
- DTS-M300 은 접촉을 하지 않고 원하는 물체 표면에 온도를 1 초 이내에 정확하게 측정할 수 있는 온도센서 모듈입니다.
- DTS-M300 은 온도계산 프로세서를 내장하고 있어 정확한 온도 값을 출력합니다.
  (Master MCU 에 온도계산알고리즘이 필요하지 않습니다.)
- DTS-M300 은 SPI 포트가 내장되어 있어 디지털 통신으로 온도 값을 출력합니다.
- 주변 온도와 타겟 온도를 동시에 측정.

▶ 특 징

- DS-ratio = 8:1
- 근거리 물체온도 측정 가능
- -30~300℃ 타켓온도 측정
- -20~80℃ 주변온도 측정
- 0.01℃ 분해능
- 1mA 저전류 소비
- Small Size(25mm*33mm)

▶ 응용분야

- 적외선 온도계.
- 과열방지 시스템.
- 인체온도를 측정하는 체온계.
- 산업용 온도측정장치.
- 체온측정을 통한 인체감지.
- 전자레인지, 에어컨, 토스터기 및 기타 가전기기.
- 자동차내 온도제어장치.

▶ ABSOLUTE MAXIMUM RATINGS

Absolute maximum rating 값을 초과하는 조건에서 DTS-M300 을 동작시킬 경우 DTS-M300 에 치명적인 손상을 가할 수 있습니다.

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Supply Voltage	Vcc	Measured versus GND	-0.2		4.0	V
Storage temperature	Tstor		-40		85	°C
Operating temperature	Top		-20		80	°C

▶ ELECTRICAL REQUIREMENTS

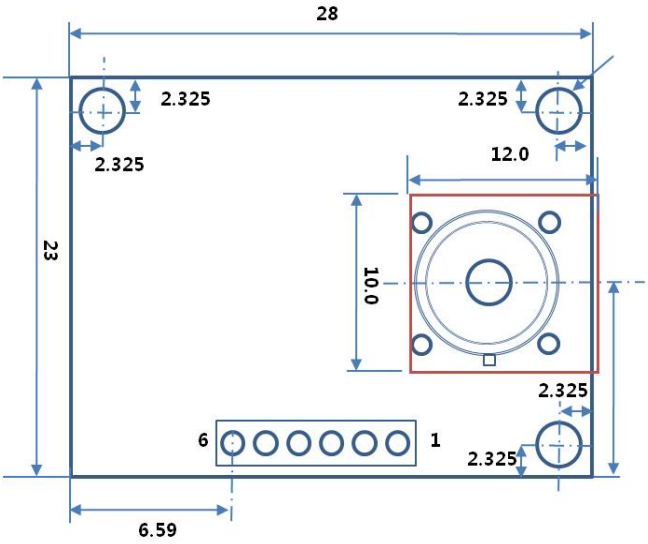
Parameter	Symbol	Conditions	Min	Typ	Max	Unit
공급전압	Vcc	Measured versus GND	2.4		3.6	V
방사율(Emission Coefficient)	ε			0.95		
공급전류		Full ambient temp. range, typical value, no output load		1		mA
SPI Clock					1	MHz

▶ OPERATIONAL CHARACTERISTICS

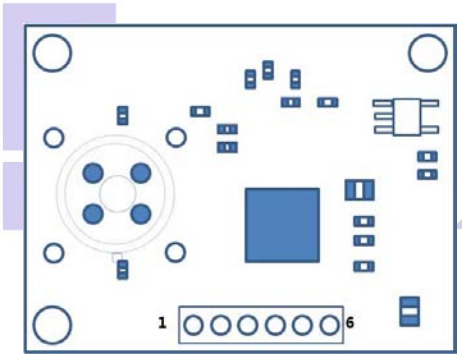
If not otherwise noted, 25°C ambient temperature, 3.3V supply voltage and object with ε =0.95 were applied.

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
DS ratio				8:1		
온도측정범위(타겟온도범위)	Tobj		-30		300	°C
동작온도(주변온도)	Tamb		-20		80	°C
온도측정 시간	Fout				1	sec
정확도	AccT			±2		%
온도분해능(Resolution Digital)				0.01		°C

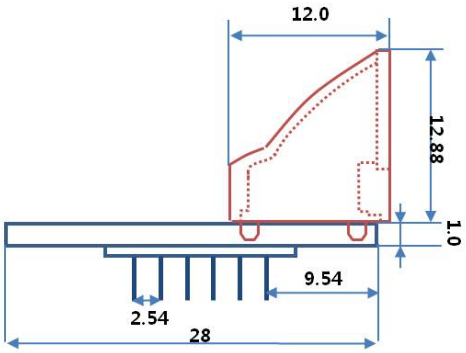
▶ MECHANICAL DIMENSIONS



< TOP VIEW >



< BOTTOM VIEW >



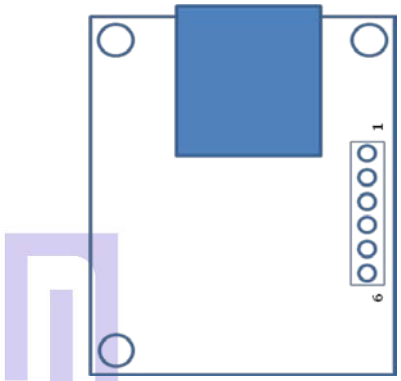
< SIDE VIEW > All units mm

Figure 1: Mechanical dimensions of DTS-M300

▶ TERMINALS

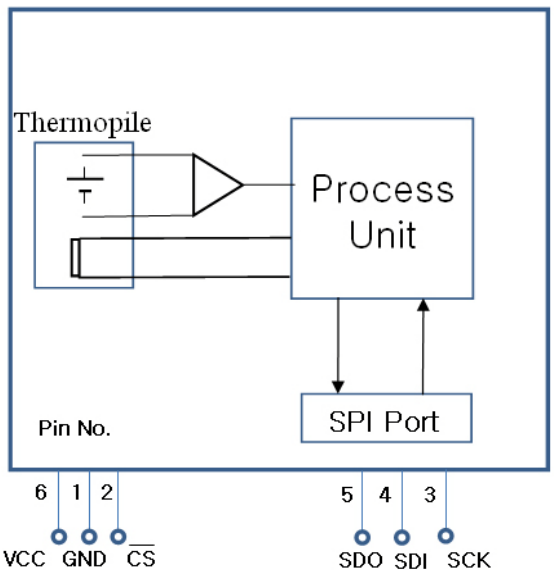
Connector : 2.54mm pitch

Pin	Name	Description	Type
1	GND	Ground	Ground
2	/CS	Enable	Input
3	SCK	Clock	Input
4	SDI	Signal Input	Input
5	SDO	Signal Output	Output
6	VCC	Supply Voltage	Supply



< TOP VIEW >

▶ BLOCK DIAGRAM



▶ SPI INTERFACE

- General Description

- DTS-M300 은 **SPI Slave mode** 로 동작합니다.
- 500ms(약 2Hz) 주기로 온도를 측정하여 data 를 업데이트 합니다.
- **전송방식 : MSB, SCK 주파수 : 최대 1MHz**

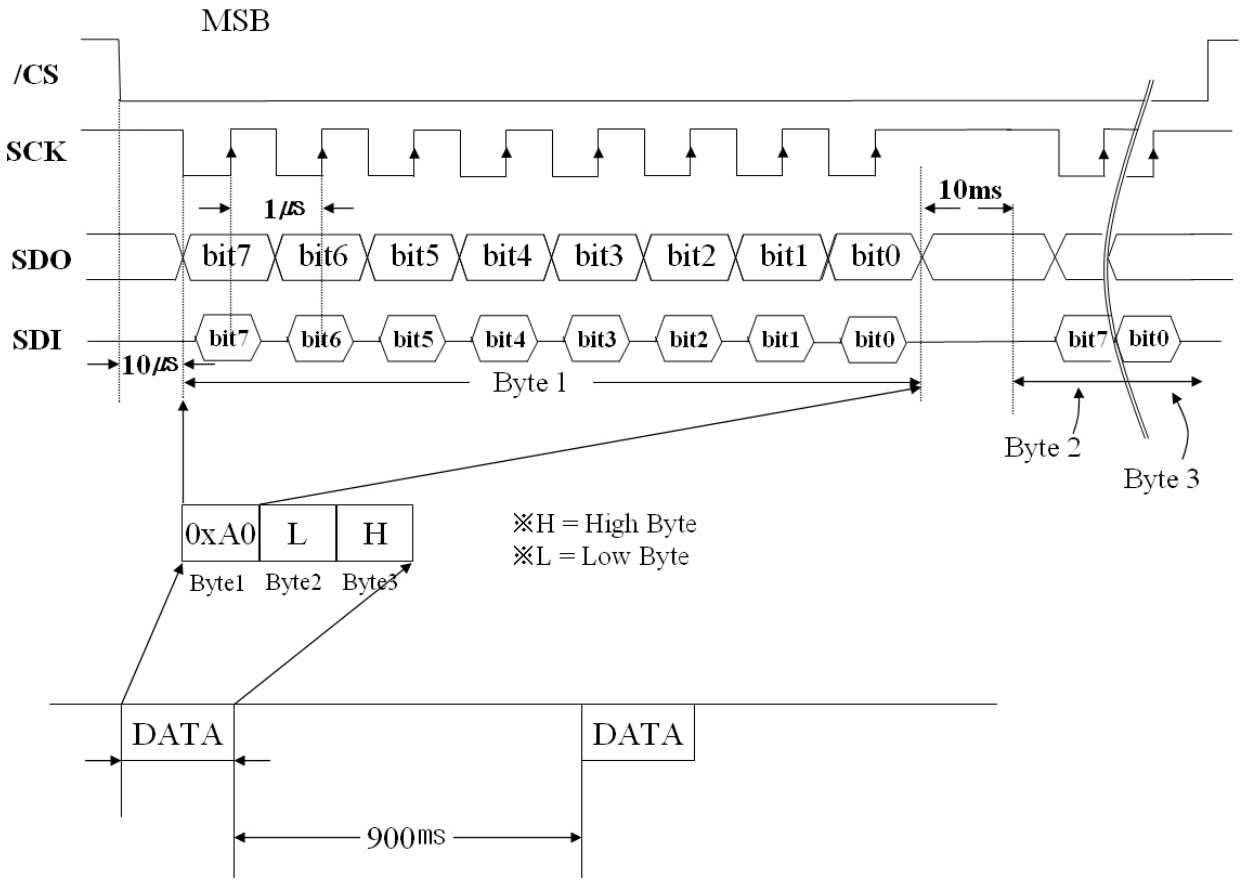
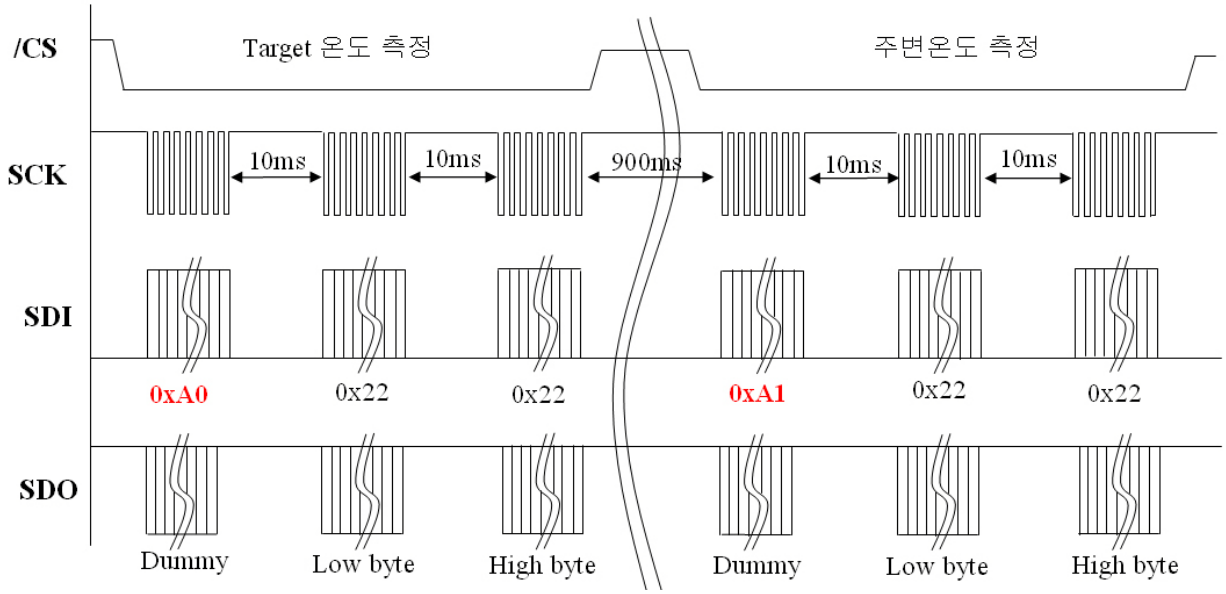


Figure. 2 Timing of SPI

- 온도 측정 프로토콜



- 온도계산방법(Temperature calculation)

● 영상온도 계산

0xA0	0x42	0x0E	0xA1	0x C4	0x09

* 타겟 온도 계산 : 상위 Byte(0x0E) + 하위 Byte(0x42) = 0x0E42

=> 3650(HEX → 10 진수) 즉 36.50 도입니다.

*주변 온도 계산 : 상위 Byte(0x09) + 하위 Byte(0xC4) = 0x09C4

= >2500 (HEX → 10 진수) 즉 25.00 도입니다.

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

0xA0	0x6A	0xFF	0xA1	0x7A	0xFF

*타겟 온도 계산 : 상위 Byte(0xFF) + 하위 Byte(0x6A) = 0xFF6A = 350

0xFF6A = 1111 1111 0110 1010 (1 의 보수 값 + 1 의 연산을 합니다)

0000 0000 1001 0101 → 1 의 보수값

0000 0000 1001 0110 → +1 = 0x0096

0x0096 = 150 즉, -1.50 도 입니다.

*주변 온도 계산 : 상위 Byte(0xFF) + 하위 Byte(0x7A) = 0xFF7A = -1.34 도입니다.

- **Sleep Mode 셋팅 방법**(삭제) - 더 이상 Sleep mode 기능은 지원하지 않습니다.

▶ 주의사항

- 본 제품은 비접촉 적외선 온도센서 모듈입니다.
- 공급전원은 2.4~3.6V 입니다. 전원을 3.6V 이상 공급하면 제품에 손상이 갈 수 있습니다.
- 각 핀에 연결은 제대로 되었는지 확인 하세요. 확인 시에는 전원을 분리하여 주세요.
- 각 핀에 연결방법은 아래 그림과 같습니다.

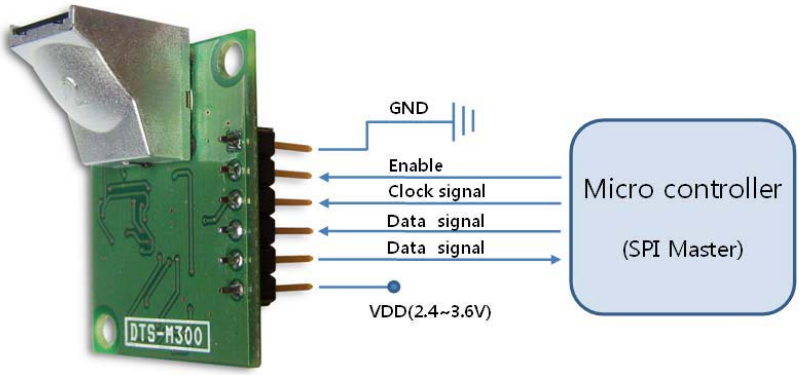


Figure 3: Pin connection Diagram

- 제품에 심한 전기적 쇼크나 충격을 가하지 않도록 하십시요. 오작동의 원인이 됩니다.
- 본제품의 DS ratio 는 8:1 입니다.
- 테스트보드를 사용하시면 보다 편리하게 온도측정을 할 수 있습니다.

Figure 5: Test Board

Additional Information

제조회사 ㈜디웰전자(DIWELL Electronics Co., Ltd.)
경기도 군포시 당정동 358 군포창업보육센터 202 호
202, Kunpo Business Incubator Center, 358, Dangjung-Dong, Gunpo-City, Gyeonggi-Do, South Korea

Phone 070-8235-0820 (+82-70-8235-0820)
Fax 031-429-0821 (+82-31-429-0821)

기술문의 email : expoeb2@diwell.com,
dsjeong@diwell.com

▶ 부록 1 (Sonix MCU spi 레지스터를 이용한 예제 코드)

아래 소스코드는 **DTS Series** 통신을 위한 참고용 가이드 소스 코드로 반드시 지켜야 할 사항이 아닙니다.
코드를 참고하여 사용자 환경에 맞게끔 레지스터 설정/ 응용하시길 바랍니다.
컴파일러 마다 레지스터 설정 명령이 전부 다릅니다. 하지만 설정하는 내용은 같습니다.
아래 설정값으로 해당 컴파일러에 맞게 응용 적용하셔야 합니다.

SPI 초기값 세팅

- **Clock 주파수 최대 1Mhz**
- **Internal SPI Clock(Master Mode)**
- **SCK data transfer edge : Rising Edge**
- **MSB first data transfer**
- **SCK idle status : High**

```
long Check_Temp(unsigned char datum)   //  Sonix 컴파일러에는 Long 이 2byte 입니다.
{
        long temp_bank=0;
        SIOB = datum;   // Buffer 레지스터에 저장
        NOP(1);
        EN_LOW;         // Enable Low
        delay_us(10);   // 10us delay
        FSTART = 1;      // SPI 전송 시작
        while(FSTART==1); // 전송 완료 flag check

        buffer2=SIOB;     // dummy
        SIOB = 0x22;      // 0x22 or 0x00 가능합니다.
        delay_ms(10);     // 10ms delay
```

```
        FSTART = 1;       // SPI 전송 시작
        while(FSTART==1); // 전송 완료 flag check

        buffer=SIOB;      // 하위 Byte 저장
        SIOB = 0x22;      // 0x22 or 0x00 가능합니다./
        delay_ms(10);     //  10ms delay
        FSTART = 1;       // SPI 전송 시작
        while(FSTART==1); // 전송 완료 flag check

        buffer1=SIOB;     // 상위 Byte 저장
        EN_HIGH;          // Enable High

        temp_bank=(buffer1*256)+buffer;   // 상위, 하위 Byte 온도 계산식.
                                          //  temp_bank : 최종 온도
        return temp_bank;
}

void main(void)
{
    Long Target_Value, Ambient_Value; //부호 있는 2byte 선언
    Port_init();    // PORT 초기화
    LCD_init();     // LCD  초기화
    EN_HIGH;        // CS  idle High
    SPI_init();     // SPI 초기화

    while(1)
     {
          Target_Value   = Check_Temp(0xa0);  // SPI 통신(타겟온도)
          delay_ms(100);
          Ambient_Value = Check_Temp(0xa1);  // SPI 통신(주변온도)
          LCD_view();             // 온도값 LCD 표시
          delay_ms(900);          // 900ms delay(필수)
                                  // 온도값 read 명령 후 다음 명령까지 900ms delay
     }
}
```

▶ 부록 2. (IO 를 이용한 SPI 예제 코드)

아래 소스코드는 IO 를 이용해 SPI 통신을 하는 함수입니다. 참고하시어 적용하십시오.

단, 포트 표현 방법이 컴파일러마다 다르므로 해당 컴파일러에 맞게 수정하시고 흐름을 참고하십시오.

SPI.H

```
#ifndef _SPI_
#define _SPI_
#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);

#endif
```

SPI.C

```
#include "SN8F27E65.h"
#include "delay.h"
#include "SPI.h"

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_ms(10);

      //Low byte read
      for(i=0; i<8; i++)
      {
          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_ms(10);

      //High byte read
          for(i=0; i<8; i++)
```

```
        {
          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);
}
```

```
void main(void)
{
   Long Target_Value, Ambient_Value;  // sonix 컴파일러는 long 이 2byte 입니다. 해당하는 컴파일러에 맞게 2byte 변수 선언하세요

    // spi 해당 포트 초기화 //클럭포트는 Idle =HIGH 입니다.
    // 기타 LCD 등 사용자 포트 초기화
    while(1)
   {
       Target_Value =  CHECK(0xa0); // 대상온도
       delay_ms(100);
       Ambient_Value = CHECK(0xa1) // 주변온도
       delay_ms(900);
       // LCD View CODE here
   }
}
```