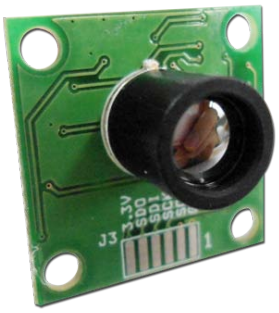
DTS-SIL300-B Specification



- 비접촉 온도 측정
- 근거리 온도 측정
- High Accuracy
- Digital Interface : SPI

▶ 제품 설명

- DTS-SIL300-B는 접촉을 하지 않고 원하는 물체 표면에 온도를 500ms 이내에 정확하게 측정할 수 있는 온도센서모듈입니다.
- DTS-SIL300-B는 온도계산 프로세서를 내장하고 있어 정확한 온도 값을 출력합니다. (Master Controller에 온도계산 알고리즘이 필요하지 않습니다.
- DTS-SIL300-B는 디지털 통신(SPI)으로 온도값을 출력합니다.
- 주변 온도와 대상 온도를 동시에 측정합니다.

DIWELL

▶ 특징

- 측정 온도 구간 : -30 ℃ ~ 300℃
- 동작 온도 구간 : -20 ℃ ~ 70℃
- 분해능 : 0.01℃
- DS ratio : 8:1
- 정확도 : ±2%
- 입력 전압 : 2.4V ~ 3.6V
- 통신 인터페이스 : SPI

▶ 응용분야

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

Absolute Maximum Ratings

- Absolute Maximum Rating 값을 초과하는 조건에서 DTS-SIL300-B을 동작시킬 경우 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	℃
Operating temperature	Top		-20		70	℃

Electrical Requirementse

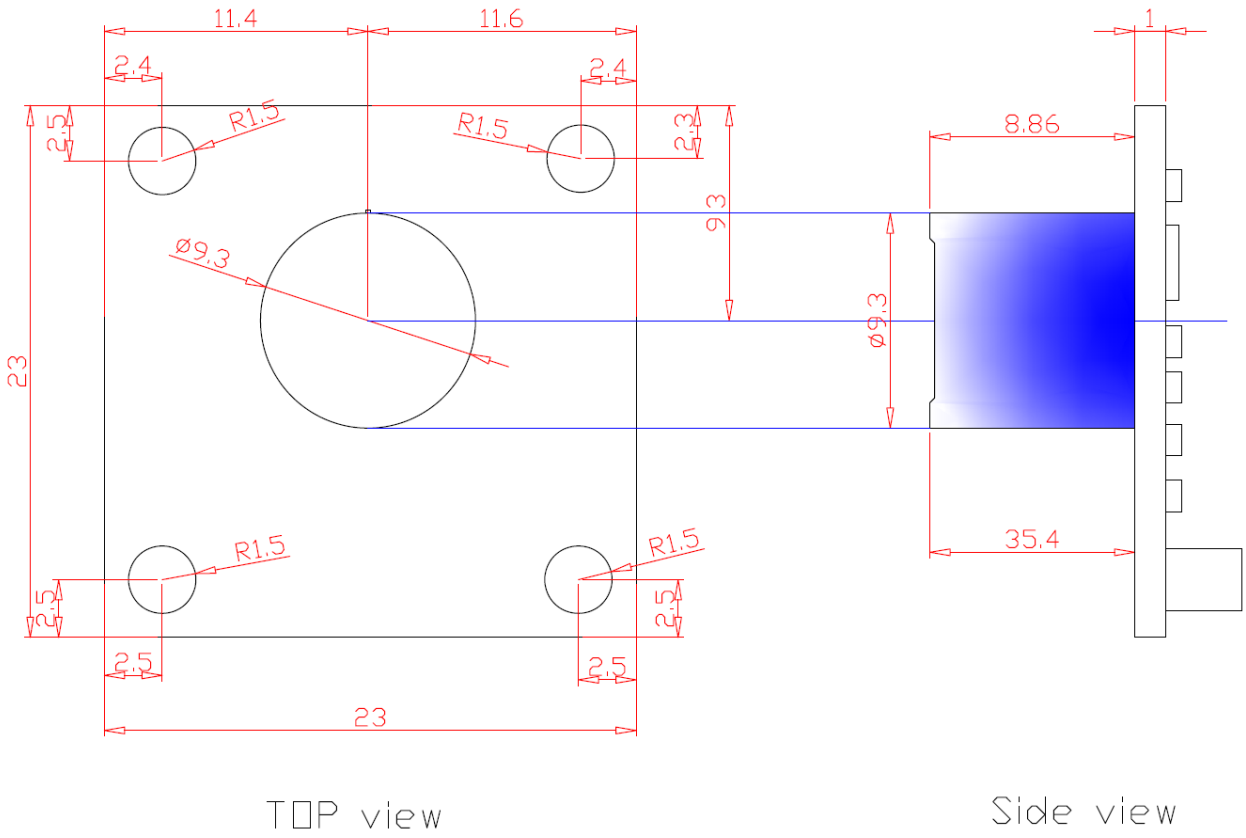
Parameter	Symbol	Conditions	min	Typ	Max	Unit
공급전압	Vcc	Measured versus GND	2.4	3.3	3.6	V
방사율(Emission Coefficient)	ε			0.95		
공급 전류		Full ambient temp. range, Typical value, no output load		1.25		mA
SPI Clock					1	MHz
INPUT High Level			**2.1**		**3.6**	V
INPUT Low Level					**0.9**	V
OUTPUT High Level			**VDD-0.3**		**VDD**	V
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 =0.95 were applied

Parameter	Symbol	Conditions	min	Typ	Max	Unit
DS ratio				8:1		
온도측정범위(타겟온도범위)	Tobj		-30		300	℃
동작온도(주변온도)	Tamb		-20		70	℃
온도측정 시간	Fout			0.5		sec
정확도	AccT			±2		%
Resolution Digital				0.01		℃
Standard Start-UP Time	tStart			3		sec
Stabilization Time	tStab			1		min

Mechanical Dimensions

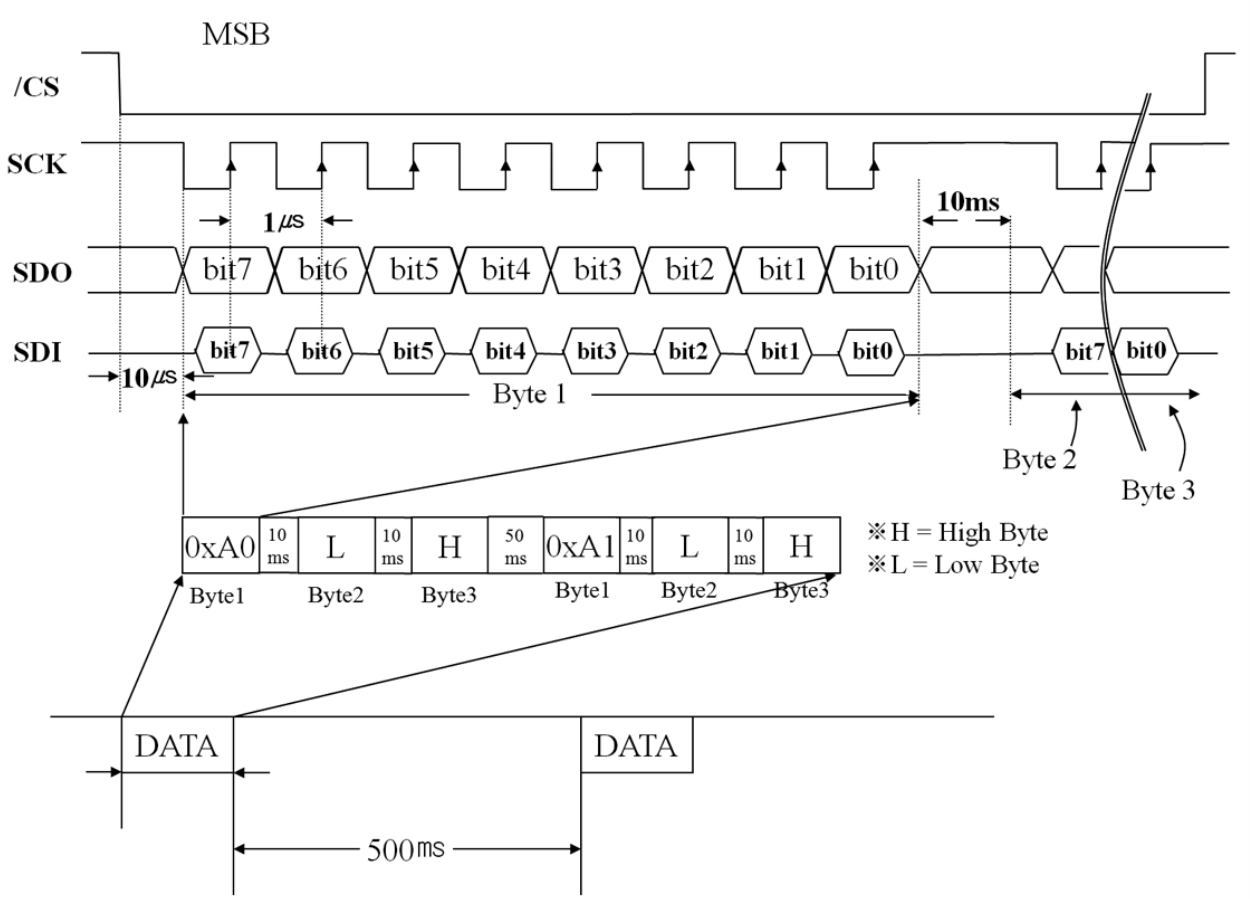


Pin Assignment

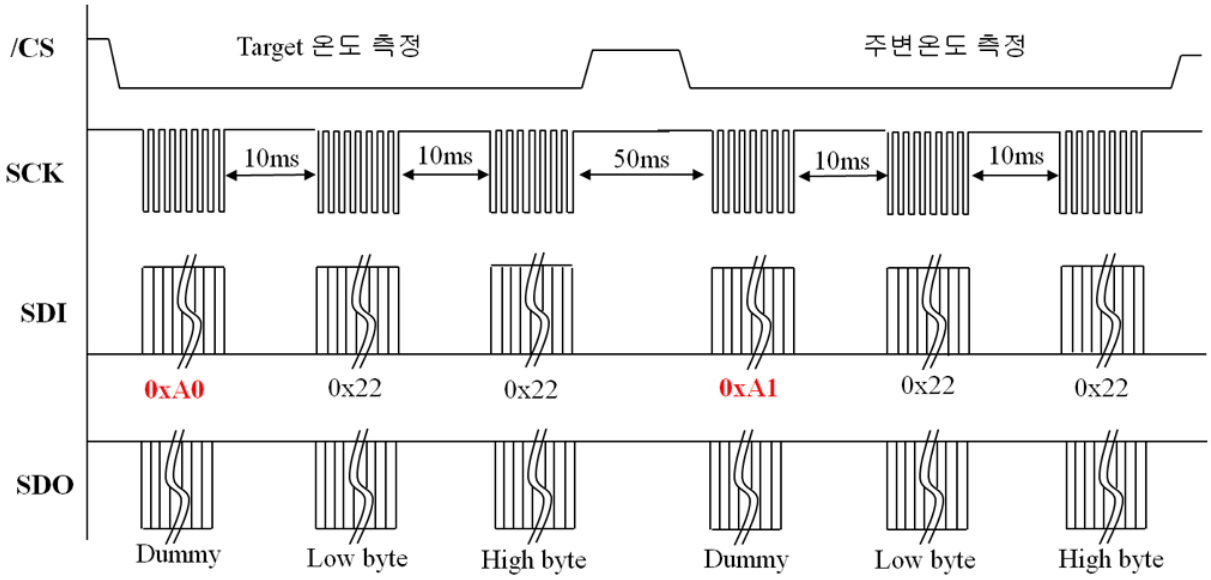
number	Name	Description	Type
1	GND	Ground	Ground
2	SCE	ENABLE	Input
3	SCK	CLOCK	Input
4	SDI	Signal Input	Input
5	SDO	Signal Output	Output
6	VCC	Supply Voltage	Supply

- Connector : Molex 053398-0671(반대편 Connector : Molex 51021-0600)

SPI - interface(모듈은 SPI Slave Mode로 동작합니다.)



- 온도 측정 프로토콜



▶ 온도 계산 방법

- **영상온도 계산**

0xA0	0x42	0x0E

0xA1	0xC4	0x09

*** 타겟 온도 계산 :** 상위 Byte(0x0E) + 하위 Byte(0x42) = 0x0E42
=> 3650(HEX → 10진수) 즉 36.50 도입니다.
*** 주변 온도 계산 :** 상위 Byte(0x09) + 하위 Byte(0xC4) = 0x09C4
= >2500 (HEX → 10진수) 즉 25.00 도입니다.

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

0xA0	0x6A	0xFF

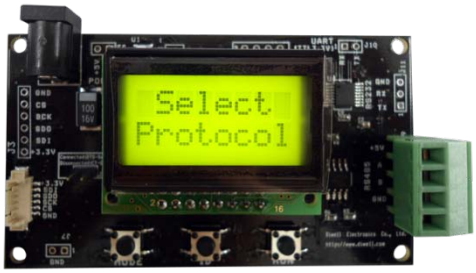
0xA1	0x7A	0xFF

***타겟 온도 계산 :** 상위 Byte(0xFF) + 하위 Byte(0x6A) = 0xFF6A = 350
0xFF6A = 1111 1111 0110 1010 (1의 보수 값 + 1 의 연산을 합니다)
0000 0000 1001 0101 □ 1의 보수값
0000 0000 1001 0110 □ +1 = 0x0096
0x0096 = 150 즉, -1.50도 입니다.
***주변 온도 계산 :** 상위 Byte(0xFF) + 하위 Byte(0x7A) = 0xFF7A = -1.34도입니다.

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

▶ 테스트 보드(별도 구매) - CT-Testboard

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



▶ Additional Information

- manufacturer : Diwell Electronics Co., Ltd. <(주)디웰전자>
- Address : 202, Kunpo Business Incubator Center, 358, Dangjung-Dong, Gunpo-City, Gyeonggi-Do, South Korea <경기도 군포시 당정동 358 군포창업보육센터 202호>
- Phone : +82-70-8235-0820
- Fax : +82-31-429-0821
- Technical support : expoeb2@diwell.com, dsjeong@diwell.com

▶ Appendix - A (Example Code - Sonix SPI 레지스터 이용)

아래 소스코드는 DTS-SIL-300 Series 통신을 위한 참고용 소스 코드 입니다.

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

사용하시는 환경에 따라 다르므로 내용을 이해 하신 후 적용하고자 하는 컴파일러에서 레지스터 설정해 주시면 됩니다. 레지스터 설정이 힘들다면 Appendix - B 의 IO 제어 코드를 참고하시길 바랍니다.

```
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(50);
            Ambient_Value = Check_Temp(0xa1);   // SPI 통신(주변온도)
```

```
            LCD_view();                    // 온도값 LCD 표시
            delay_ms(500);                 // 500 delay(필수)
                                           // 온도값 read 명령 후 다음 명령까지 500ms delay
        }
}
```

▶ Appendix - B (Example Code - I/O)

하단 코드는 컨트롤러에서 SPI 레지스터 설정이 아닌 I/O 포트 제어를 통해 통신하는 예제 코드입니다. 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, Ambient_Value;
 // sonix 컴파일러는 long이 2byte 입니다. 해당하는 컴파일러에 맞게 2byte변수 선언하세요
void main(void)
{
    While(1)
    {
        Target_Value =   CHECK(0xa0); // 대상온도
        delay_ms(50);
        Ambient_Value = CHECK(0xa1) // 주변온도
        delay_ms(500);
        // 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_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);
}
```

DTS-SIL300-B Revision History

Version	Date	Description
1.0	2013-06-24	**First version is released.**
1.1	2013-09-06	**Modify Example C code(page 6~9)**
1.2	2013-12-10	**Update Physical Interface Parameters.(page 2)**
1.3	2014-03-17	**Sleep mode 삭제, 방사율 수정**

DIWELL