

Application Note

Infrared Thermopile Module TSEM01-L

Version 1.0

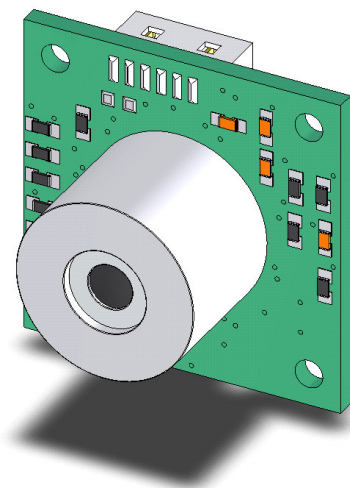


Table of Content

1	History	4
2	General Information	4
3	Device Overview	4
3.1	Thermopile	4
3.2	Applications	4
4	Setting up Connection	5
4.1	Terminals	5
4.2	I ² C Bus	6
4.2.1	Low Level I ² C Bus	6
4.2.2	High Level I ² C Bus	6
4.2.3	Physical Interface Parameters	7
4.2.4	Timing Parameters	7
5	Voltage Requirements	9
5.1	Maximum Ratings	9
5.2	Operating Conditions	9
6	Setting up Microcontroller for I ² C Operation	9
6.1	Configuration of Slave Address	9
7	I ² C Commands	9
7.1	Ambient and Object Temperature Measurement	9
7.2	Byte Interpretation	9
7.3	Temperature Value Interpretation	10
7.4	Example of Reading Temperatures	11
7.4.1	Send Command	11
7.4.2	Receiving Ambient and Object Temperatures	11
7.5	Out of Range Indication	12
8	Software examples for I ² C communication	12
8.1	Send one Byte to I ² C Slave	13
8.2	Read one Byte from I ² C Slave	14
8.3	Read Object and Ambient Temperature from TSEM01-L	15
9	Infrared Temperature Measurement Guide	16
9.1	Testing I ² C Communication	16

9.2	Measurement of Ambient (Sensor) Temperature	16
9.3	Measurement of Object Temperature	17
9.3.1	Field of View	17
9.3.2	Direct Sunlight	17
9.3.3	Emissivity	18
9.3.4	Touching the Sensors Cap	18
10	Additional Information	18

1 History

Ver.	Document name	Date	Purpose	Author
1.0	TSEM01-L_App_Note_V1_0_20070213.doc	13.02.2007	Creation	M. Basel

2 General Information

TSEM01-L is a contact less temperature measuring system for OEM use based on the detection of infrared radiation.

TSEM01L is equipped with an infrared sensor (Thermopile) in front. In this version a single Element Thermopile Sensor is used. It has to be pointed at the target object.

The basic working principle is:

- Collection of infrared radiation by an optics
- Detection of collected infrared radiation with a Thermopile sensor
- Further analogue signal processing
- Calculation of ambient and object temperature using a microcontroller
- Providing the ambient and objects temperature at digital output (I2C-Slave Operation)

The main fields of applications are temperature measuring in industrial applications i.e. at moving or inaccessible parts.

3 Device Overview

3.1 Thermopile

Any object emits infrared radiation. The radiation power is increasing with growing surface temperatures. Based on this relation, thermopiles measure the emitted power and determine the object's temperature precisely.

Thermopiles are based on the Seebeck effect, which is used since a long time for conventional thermocouples. The application of micromechanics and thin film technology allows the production of miniaturized and cost effective sensor elements.

3.2 Applications

The TSEM01-L is suitable for a wide range of application where non-contact temperature measurement and high accuracy are required.

For example:

- Climate control
- Industrial applications

4 Setting up Connection

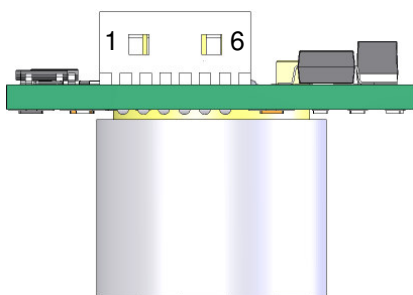
This chapter describes how to connect the TSEM01-L temperature module to a customer microcontroller system.

4.1 Terminals

Following connector has to be used:

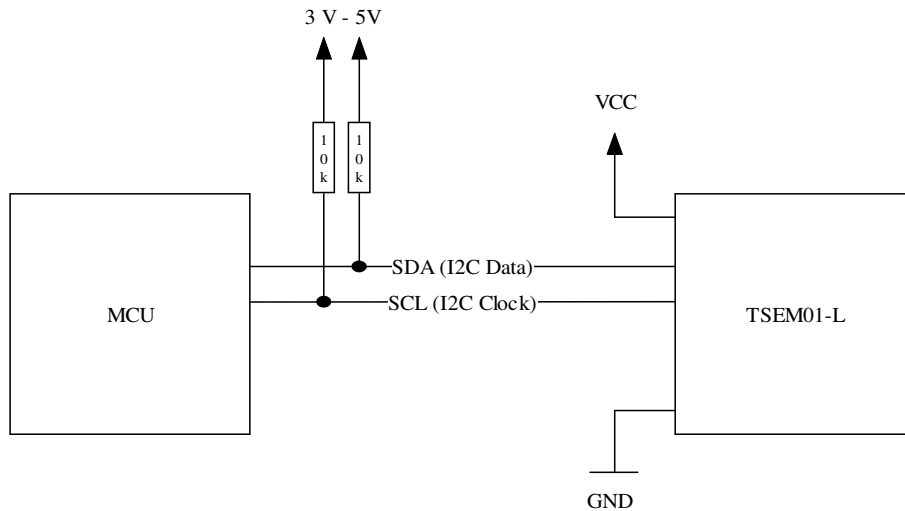
Molex 51021-0600 (Farnell-In-One: 1012261, Digikey: WM1724-ND)

Pin	Name	Description	Type
1	+VS	Supply Voltage	Supply
2	GND	Ground potential	Supply
3	NC		
4	SCL	I ² C like Clock	Input
5	NC		
6	SDA	I ² C like Data	Input / Output



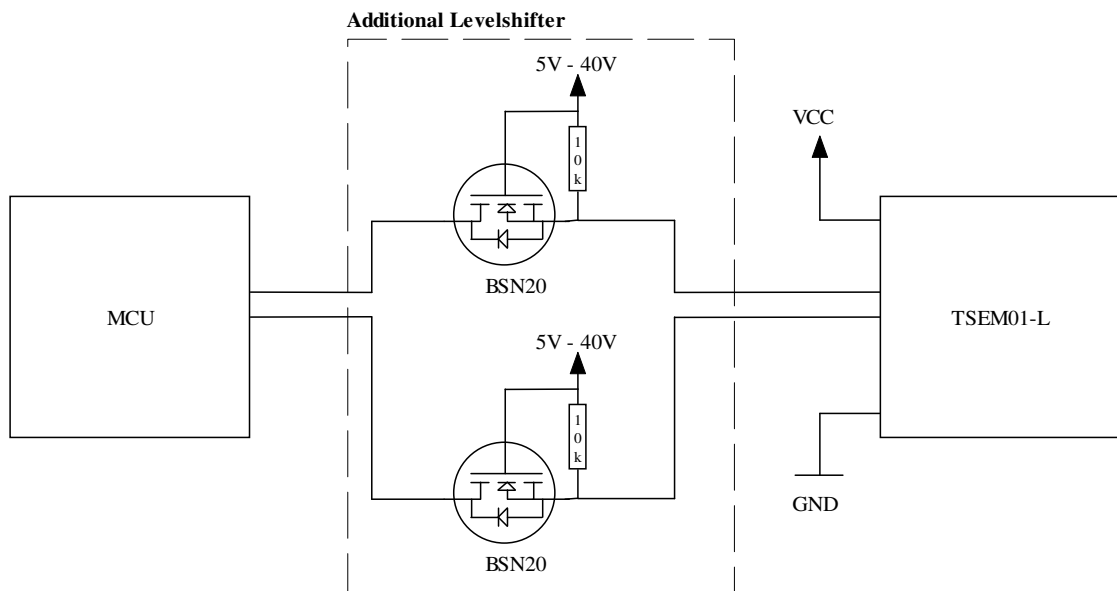
4.2 I²C Bus

4.2.1 Low Level I²C Bus



4.2.2 High Level I²C Bus

An additional (external) bi-directional level shifter may be used to interconnect two devices of an I²C-bus system, each device with a different supply voltage and different logic levels.



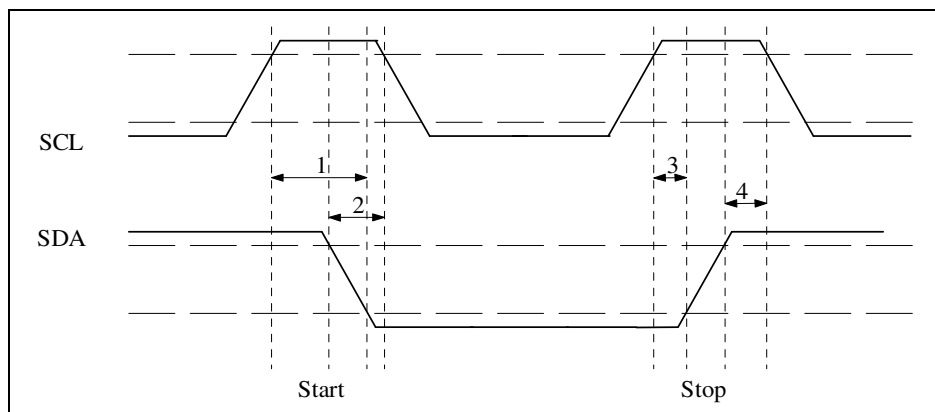
4.2.3 Physical Interface Parameters

Parameter	Min	Typical	Max	Unit
Baudrate	10	100	300	kBit/s
Address length	---	7	---	Bit
Address (standard)	---	160	---	---
Input High Level	0,7 VDD	---	---	V
Input Low Level	---	---	0,3 VDD	V
Output High Level	VDD – 0,7	---	---	V
Output Low Level	---	---	0,6	V

4.2.4 Timing Parameters

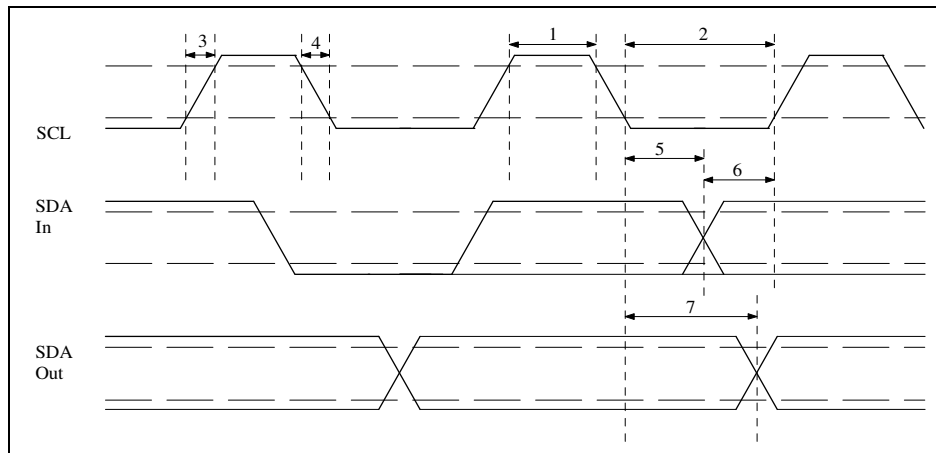
No.	Parameter	Description	Min	Typ	Max	Unit
1	TSU:STA	Start Setup Time	4700	---	---	ns
2	THD:STA	Start Hold Time	4000	---	---	ns
3	TSU:STO	Stop Setup Time	4700	---	---	ns
4	THD:STO	Stop Hold Time	4000	---	---	ns

Start/Stop



Data

No	Parameter	Description	Min	Max	Unit
1	THIGH	Clock High Time	4.0	---	μs
2	TLOW	Clock Low Time	4.7	---	μs
3	TR	SDA & SCL Rise Time	---	1000	ns
4	TF	SDA & SCL Fall Time	---	300	ns
5	THD:DAT	Data Input Hold Time	0	---	ns
6	TSU:DAT	Data Input Setup Time	250	---	ns
7	TAA	Output Valid From Clock	---	3500	ns
8	TBUF	Bus Free Time	4.7	---	μs



5 Voltage Requirements

5.1 Maximum Ratings

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Supply Voltage	V _{cc}	Measured versus GND	-0.3		5.5	V

5.2 Operating Conditions

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Supply voltage	V _{cc}	Measured versus GND	4.75		5.25	V

6 Setting up Microcontroller for I²C Operation

6.1 Configuration of Slave Address

TSEM01-L is always operating in pure slave modus of a two wire interface similar to I²C. The typical baud rate of this device is 100kBit/s. The supported address length is seven bits. Addresses with a length of 10 bits are not supported.

7-Bit Slave Address							Direction Bit R/W	Byte Transfer to Slave
MSB							LSB	
1	0	1	0	0	0	0	0	160 dec, 0xA0 (WRITE)
1	0	1	0	0	0	0	1	161 dec, 0xA1. (READ)

Every transfer has to be initiated by a start sequence and terminated by a stop sequence. The master device can only send one data byte during one transmission.

7 I²C Commands

7.1 Ambient and Object Temperature Measurement

Please refer following table for I²C commands to read object temperature and ambient temperature. Both values are transmitted in tenth of degrees.

Command	Description	Reply	Bytes
0x07	Read ambient and object temperature	Ambient and Object Temperature in tenth of degree.	4

7.2 Byte Interpretation

Data is always transmitted and received in below order.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MSB							LSB

7.3 Temperature Value Interpretation

Transferred data representing ambient and object temperature are always 16Bit signed integer values. Due to I²C-specification both values are transferred in two 8 byte fractions of the 16 byte value.

Sequence of 8 byte transfer:

First byte: High byte

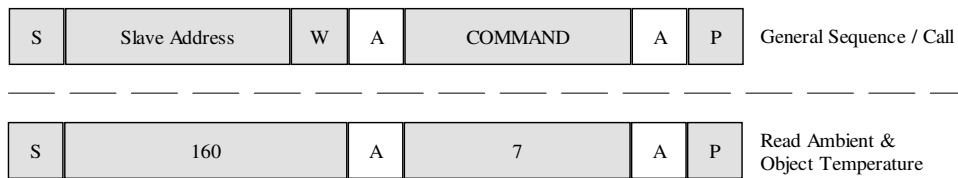
Second byte: Low byte

High byte							
Bit 15	14	13	12	11	10	9	8

Low byte							
7	6	5	4	3	2	1	0

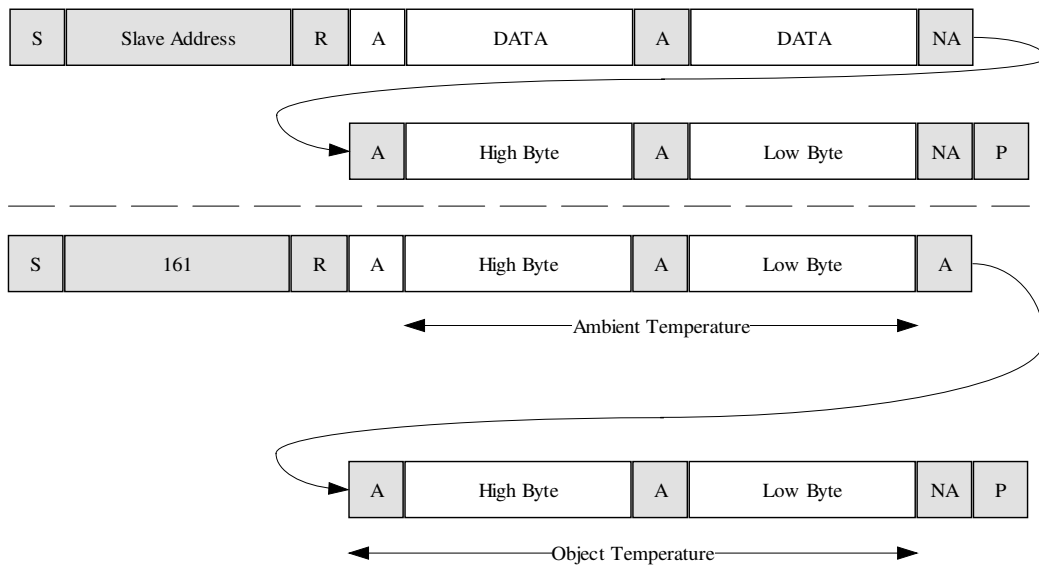
7.4 Example of Reading Temperatures

7.4.1 Send Command



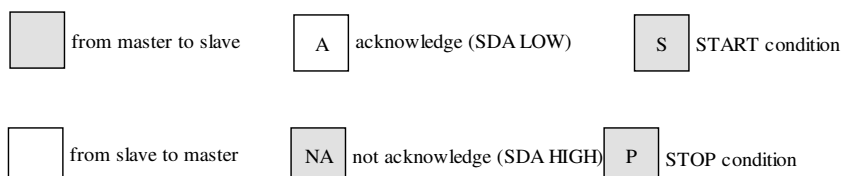
Following a START condition the master sends the I²C-Slave-Address with R/W bit reset to "0". The slave acknowledges the address match automatically and waits for the command. After receipt of the command the slave device again responds with an acknowledge. The master terminates the transfer by generating a STOP condition.

7.4.2 Receiving Ambient and Object Temperatures



Following a START condition the master sends the I²C-Slave-Address with R/W bit set to "1". The slave acknowledges this and transmits the first data byte. The master issues an acknowledge and the slave transmit the next sequentially data byte. After the last received byte the master does not acknowledge the byte output and terminates the transfer with a STOP condition.

The temperature value is always given as a signed integer value (16Bit) in tenth of degree Celsius.



e.g. : integer value 185 => be equivalent to 18.5 °C

For reading object temperature send: 0x07

Return values i.e.: Byte(0) = 0x01, Byte(1) = 0x0E

Object temperature

$T_{obj} = (256 * \text{Byte}(0) + \text{Byte}(1)) / 10 = (256 * 1 + 14) / 10 = 27.0\text{°C}$

7.5 Out of Range Indication

In case of ambient or object temperature over exceeding specified temperature ranges, temperature outputs representing following data:

Description	Reply
Ambient temperature out of range	0x1FF0
Object Temperature out of range	0x1FF3

8 Software examples for I²C communication

Subsequent are given the routines to communicate with TSEM01-L. The shown code is written for Microchip PIC devices but can easily be adapted to any μ -Controller with I²C interface.

Code contains two fundamental functions:

- Read a byte from I²C-Slave
- Write a byte to I²C-Slave

Further code take advantage of fundamental function to realize:

- Reading measured ambient and object temperature

8.1 Send one Byte to I²C Slave

```

/*-----
Author:      M. Basel
Purpose:     Write a byte to I2C-Slave
Return:     ---
Variables:   cSlaveAddress:  I2C-Slave-Adress
cData:      Data (1 Byte)
bSendAddress:  Send I2C-Slave-Adress (true/false)
bStart:      Send Start-Bit (true/false)
bStop:      Send Stop-Bit (true/false)
-----*/

void I2C_Write(char cSlaveAddress,
char cData,
BOOLEAN bSendAddress,
BOOLEAN bStart,
BOOLEAN bStop)
{
    // If Start-Bit has to be send
    if (bStart == TRUE)
    {
        SEN = 1;                               // Send Start-Condition
        while(SEN == 1);                         // Wait until Start-Condition
        transfered
    }

    SSPIF = 0;                                 // Reset I2C-Interrupt

    // If I2C-Slave-Adress has to be send
    if (bSendAddress == TRUE)
    {
        SSPBUF = cSlaveAddress | 0x00;          // Send I2C-Slave-Adress and Write-Bit
        (0)
        while(SSPIF == 0) {}                    // Wait until I2C-Interrupt set
        while(ACKSTAT == 1) {}                 // Wait until I2C-Slave send
        Acknowledge
    }

    SSPIF = 0;                                 // Reset I2C-Interrupt

    SSPBUF = cData;                             // Write Data to be transferred in
    // transfer-buffer. Data will be transferred

    while(SSPIF == 0) {}                       // Wait until I2C-Interrupt set
    while(ACKSTAT == 1) {}                     // Wait until I2C-Slave send Acknowledge

    // If Stop-Bit has to be send
    if (bStop == TRUE)
    {
        PEN = 1;                               // Send Stop-Condition
        while(PEN == 1) {}                     // Wait until Stop-Condition
        transfered
    }
}

```

8.2 Read one Byte from I²C Slave

```

/*-----
Author:      M. Basel
Purpose:     Read a byte from I2C-Slave
Return:      Received byte
Variables:   cSlaveAddress:  I2C-Slave-Adress
              bSendAddress:   Send I2C-Slave-Adress (true/false)
              bStart:         Send Start-Bit (true/false)
              bStop:          Send Start-Bit (true/false)
              bAck:           Send Acknowledge (true/false)
-----*/

char I2C_Read(char cSlaveAddress,
              BOOLEAN bSendAddress,
              BOOLEAN bStart,
              BOOLEAN bStop,
              BOOLEAN bAck)
{
    char cResult;

    // If Start-Bit has to be send
    if (bStart == TRUE)
    {
        SEN = 1; // Send Start-Condition
        while(SEN == 1); // Wait until Start-Condition transfered
    }

    // If I2C-Slave-Adress has to be send
    if (bSendAddress == TRUE)
    {
        SSPBUF = cSlaveAddress | 0x01; // Send I2C-Slave-Adress and ReadBit (1)
        while(SSPIF == 0) {} // Wait until I2C-Interrupt set
        while(ACKSTAT == 1) {} // Wait until I2C-Slave send Acknowledge
    }

    RCEN = 1; // Enable receive mode. Data will be
              // received

    while(SSPIF == 0) {} // Wait until I2C-Interrupt set

    // Send Acknowledge or NotAcknowledge
    if (bAck == TRUE) ACKDT = 0; // Set ACK-Bit to Not Acknowledge
    else ACKDT = 1; // Set ACK-Bit to Acknowledge
    ACKEN = 1; // Send ACK-Bit

    while (ACKEN == 1) {} // Wait for transfer completed

    // If Stop-Bit has to be send
    if (bStop == TRUE)
    {
        PEN = 1; // Send Stop-Condition
        while(PEN == 1) {} // Wait until Stop-Condition transfered
        RCEN = 0; // Disable receive mode
    }

    while (BF == 0) {} // Wait for receive-buffer full

    cResult = SSPBUF; // Save content of receive-buffer

    return cResult; // Return received data
}

```

8.3 Read Object and Ambient Temperature from TSEM01-L

```

/*-----
Author:      M. Basel
Purpose:     Read measured ambient and object temperature
              from I2C-Slave
Return:     State of success
Variables:   uiAmbTemp: Pointer for result of ambient
              temperature
              uiObjTemp Pointer for result of object
              temperature
Comment:    Ambient and object temperature are unsigned
              integer values multiplied by hundred
-----*/
BOOLEAN Read_Temperatures(unsigned int* uiAmbTemp, unsigned int* uiObjTemp)
{
    unsigned char cLoByte;
    char cSlaveAddress;
    char cCommand;
    char cReceivedByte;

    cSlaveAddress = 160;          // Set I2C-Slave-Address to 160d
    cCommand = 7;                // Command: Read temperatures (7)

    // Write command to I2C-Slave
    // -----
    // cSlaveAddress:    I2C-Slave-Address (160)
    // cCommand:         Command to read I2C-Slave-EEPROM (7)
    // bSendAddress = TRUE: Send I2C-Slave-Address
    // bStart = TRUE:    Send Start-Sequence
    // bStop = TRUE:     Send Stop-Sequence
    I2C_Write(cSlaveAddress, cCommand, TRUE, TRUE, TRUE);

    // Receive High-Byte of ambient temperature from I2C-Slave
    // -----
    // cSlaveAddress:    I2C-Slave-Address (160)
    // bSendAddress = TRUE: Send I2C-Slave-Address
    // bStart = TRUE:    Send Start-Sequence
    // bStop = FALSE:   Don't send Stop-Sequence
    // bAck = TRUE:     Send Acknowledge
    cReceivedByte = I2C_Read(cSlaveAddress, TRUE, TRUE, FALSE, TRUE);

    // Save High-Byte of ambient temperature
    uiAmbTemp = cReceivedByte << 8;

    // Receive Low-Byte of ambient temperature from I2C-Slave
    // -----
    // cSlaveAddress:    I2C-Slave-Address (160)
    // bSendAddress = FALSE: Don't send I2C-Slave-Address
    // bStart = FALSE:   Don't send Start-Sequence
    // bStop = FALSE:   Don't send Stop-Sequence
    // bAck = TRUE:     Send Acknowledge
    cReceivedByte = I2C_Read(cSlaveAddress, FALSE, FALSE, FALSE, TRUE);

    // Save Low-Byte of ambient temperature
    uiAmbTemp = uiAmbTemp | cReceivedByte;

    // Receive High-Byte of object temperature from I2C-Slave
    // -----
    // cSlaveAddress:    I2C-Slave-Address (160)
    // bSendAddress = FALSE: Don't send I2C-Slave-Address
    // bStart = FALSE:   Don't send Start-Sequence
    // bStop = FALSE:   Don't send Stop-Sequence
    // bAck = TRUE:     Send Acknowledge
    cReceivedByte = I2C_Read(cSlaveAddress, FALSE, FALSE, FALSE, TRUE);

    // Save High-Byte of object temperature
    uiObjTemp = cReceivedByte << 8;

    // Receive Low-Byte of object temperature from I2C-Slave
    // and finish transfer
    // -----
    // cSlaveAddress:    I2C-Slave-Address (160)

```

```
// bSendAdress = FALSE: Don't send I2C-Slave-Adress
// bStart = FALSE:   Don't send Start-Sequence
// bStop = TRUE:     Send Stop-Sequence
// bAck = FALSE:     Send NotAcknowledge
cReceivedByte = I2C_Read(cSlaveAdress, FALSE, FALSE, TRUE, FALSE);

// Save Low-Byte of object temperature
uiObjTemp = uiObjTemp | cReceivedByte;

// Return from function
return TRUE;
}
```

9 Infrared Temperature Measurement Guide

9.1 Testing I²C Communication

Before performing of measurement, user may check valid I²C communication between Master system and TSEM01-L.

Therefore request ambient temperature measurement and check if the transmitted value correspond to the actual room temperature.

Next step is to test object temperature measurement. Point the thermopile sensor to any constant temperature surface (i.e. room wall). The measured object temperature has to be close to the ambient temperature measured before.

9.2 Measurement of Ambient (Sensor) Temperature

A thermistor is integrated in the thermopile sensor due to the need of ambient temperature compensation. This thermistor is measuring the sensors temperature. Therefore, after some settling time, the sensor temperature matches the actual room temperature.

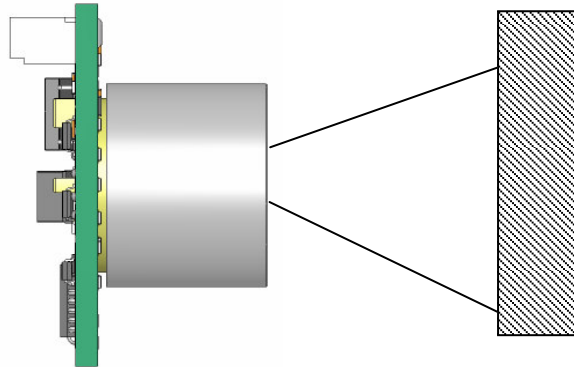
9.3 Measurement of Object Temperature

To achieve reliable and accurate object temperature measurements, user has to comply to some advice when it comes to handle with thermopile sensors.

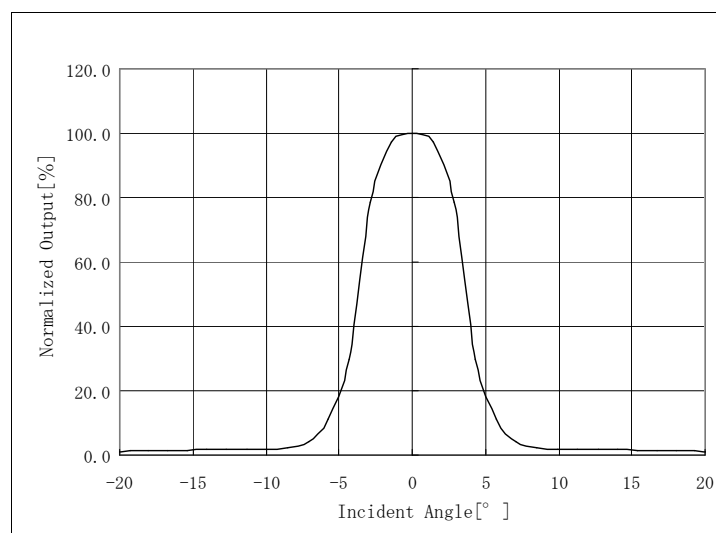
- Field of View
- Valid results are available at 1 seconds after pointing at object of interest.
- Avoid scratches or contamination of window surface.
- Avoid touching the cap of the sensor

9.3.1 Field of View

The thermopiles field of view has to be directed to the object surface of interest. The distance to the surface or the surface diameter has to be adjusted to insure that the complete sensors field of view is covered by the object.



Please refer to the sensors field of view as shown below.



9.3.2 Direct Sunlight

Sun light radiation which is transmitted through a glass window may influence the measurement accuracy. To avoid this, the thermopile sensor is equipped with a long wavelength filter. Due to not ideal filter characteristics a small portion of radiation will

be added to the radiation of the object. In case of direct sunlight exposure this error can be up to +0.2 °C.

9.3.3 Emissivity

Every object is transmitting infrared energy in dependence to its temperature. The emissivity is the ratio of the radiated power by an object to the radiation of an ideal black body. Common materials like liquids, clothes, human skin, foods have emissivity factors >0.90 and therefore they can be measured very accurately without adopting the sensors specification.



The internal emissivity correction algorithm may be adjusted to customer application while calibration process.

9.3.4 Touching the Sensors Cap

User should avoid touching the sensors cap. TSEM01-L is equipped with a special mechanism to decrease impacts of heating or cooling on measurement accuracy. Even so there will still be a measurement deviation for some seconds after changing the sensors temperature rapidly.

10 Additional Information

Company:	HL-Planartechnik GmbH Hauert 13 D-44227 Dortmund
Phone	+49 (0)231/ 9740-0
Fax	+49 (0)231/ 9740-20
URL	www.hlplanar.com
Mail	mailto:service@hlplanar.de