DSTH01
Digital Temperature Humidity Sensor Module

Features:
- I2C host interface
- Temperature range: -40°C →+85°C
- Temperature accuracy: ±0.5°C (typical)
  ±1°C (Max. @0~70°C)
- RH operating range: 0~100%
- RH accuracy: ±3% (typical)
- Wake up time: 10ms
- Operating voltage range: 2.1~3.6V
- Integrated on-chip heater
- Low power consumption
- Excellent long term stability
- Factory calibrated

Applications
- Industrial HVAC/R
- Thermostats / humidistats
- Respiratory therapy
- Automotive climate control
- Asset and goods tracking

DESCRIPTION
DSTH01 is a type of digital relative humidity and temperature sensor module which integrates
temperature and humidity sensor elements, an analog-to-digital converter, signal processing,
calibration data and an I2C host interface. Both the temperature and humidity sensors are
factory-calibrated and the calibration data is stored in the on-chip non-volatile memory which
ensures the DSTH01 modules are fully interchangeable and no recalibration or software changes
are required.

Necessary components are integrated on the DSTH01 modules so users can get a quick start with
the microcontroller with extra design. The sensor top is covered with silicon gel which can protect
the sensor from the dust and other particles. The module works at 2.1~3.6V. It consumes about
240uA during RH conversion and 1.5mA in normal work mode. The DSTH01 module offers an
accurate and factory-calibrated solution for embedded applications ranging from HVAC/R system
to consumer electronic products.
**PIN FUNCTIONS**

![DSTH01 Pin Layout](image)

Figure 1: DSTH01 Pin Layout

<table>
<thead>
<tr>
<th>PIN</th>
<th>DIP</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>GND</td>
<td>Ground (0V)</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>SCK</td>
<td>Input I2C serial port, clock pin</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>SDA</td>
<td>Input/output I2C serial port, data pin</td>
</tr>
<tr>
<td>4</td>
<td>/CS</td>
<td>Input</td>
<td>Chip selection, low effective</td>
</tr>
<tr>
<td>5</td>
<td>VCC</td>
<td>Power</td>
<td>Power supply</td>
</tr>
</tbody>
</table>

Table 1: DSTH01 Pin functions

**ELECTRICAL SPECIFICATIONS**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter (condition)</th>
<th>Min.</th>
<th>Typ.</th>
<th>Max.</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCC</td>
<td>Supply Voltage</td>
<td>2.1</td>
<td>3.3</td>
<td>3.6</td>
<td>V</td>
</tr>
<tr>
<td>Temp</td>
<td>Temperature operating range</td>
<td>-40</td>
<td>25</td>
<td>85</td>
<td>°C</td>
</tr>
<tr>
<td>TR(1)</td>
<td>Temperature resolution</td>
<td>14</td>
<td>bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TA(2)</td>
<td>Temperature accuracy: typical @25°C</td>
<td>±0.5</td>
<td>±1</td>
<td>°C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Maximum</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRN</td>
<td>Temperature repeatability-noise</td>
<td>0.1</td>
<td>°C</td>
<td>RMS</td>
<td></td>
</tr>
<tr>
<td>TRT(3)</td>
<td>Response time to reach 63% of final value</td>
<td>1.5</td>
<td>s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TLS</td>
<td>Temperature long term stability</td>
<td>&lt;0.05</td>
<td>°C/yr</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RH(4)</td>
<td>Relative humidity operating range</td>
<td>0</td>
<td>100</td>
<td>%</td>
<td></td>
</tr>
<tr>
<td>Hr(5)</td>
<td>Humidity resolution</td>
<td>12</td>
<td>bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HA(6)</td>
<td>Humidity accuracy: 20~80%RH</td>
<td>±3</td>
<td>±4.5</td>
<td>%RH</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0~100%RH</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hrn</td>
<td>Humidity repeatability-noise</td>
<td>0.05</td>
<td>%RH</td>
<td>RMS</td>
<td></td>
</tr>
</tbody>
</table>

(1) Temperature resolution
(2) Temperature accuracy: typical @25°C Maximum
(3) Response time to reach 63% of final value
(4) Relative humidity operating range
(5) Humidity resolution
(6) Humidity accuracy: 20~80%RH 0~100%RH
Table 2: DSTH01 Electrical Specifications

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value 1</th>
<th>Value 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>H&lt;sub&gt;RT&lt;/sub&gt; @ 1 m/s airflow</td>
<td>8</td>
<td>s</td>
</tr>
<tr>
<td>H&lt;sub&gt;H&lt;/sub&gt;</td>
<td>±1</td>
<td>%RH</td>
</tr>
<tr>
<td>H&lt;sub&gt;LS&lt;/sub&gt; Humidity LS</td>
<td>≤0.25</td>
<td>%RH</td>
</tr>
<tr>
<td>I&lt;sub&gt;DD&lt;/sub&gt; Current. @ RH conversion in progress</td>
<td>240</td>
<td>560 uA</td>
</tr>
<tr>
<td>@ Temp conversion in progress</td>
<td>320</td>
<td>565 uA</td>
</tr>
<tr>
<td>@ Heater enabled, on conversion in progress</td>
<td>24</td>
<td>31 mA</td>
</tr>
<tr>
<td>T&lt;sub&gt;CON&lt;/sub&gt; Conversion time. @ 14-bit temp, 12-bit RH (fast=0)</td>
<td>35</td>
<td>40 ms</td>
</tr>
<tr>
<td>@ 13-bit temp, 11-bit RH (fast=1)</td>
<td>18</td>
<td>21 ms</td>
</tr>
<tr>
<td>T&lt;sub&gt;PU&lt;/sub&gt; Power up time</td>
<td>10</td>
<td>15 ms</td>
</tr>
</tbody>
</table>

Notes:

(1). The DSTH01 module has a nominal output of 32 codes °C, with 0000=-50°C

(2). Temperature sensor accuracy is for VDD = 2.3 to 3.6 V.

(3). Actual response times will vary dependent on system thermal mass and air-flow.

(4). Recommended humidity operating range is 20 to 80%RH (non-condensing) over 0 to 60°C. Prolonged operation beyond these ranges may result in a shift of sensor reading, with slow recovery time.

(5). The DSTH01 module has a nominal output of 16 codes per %RH, with 0h0000=-24%RH.

(6). Excludes hysteresis, long-term drift, and certain other factors and is applicable to non-condensing environments only.

(7). Time for sensor output to reach 63% of its final value after a step change.

Figure 2: DSTH01 RH Accuracy at 30°C
Figure 3: DSTH01 Temperature Accuracy

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter (condition)</th>
<th>Min.</th>
<th>Typ.</th>
<th>Max.</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>V_{HYS}</td>
<td>Hysteresis</td>
<td>0.05*VCC</td>
<td></td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>F_{SCLK}</td>
<td>SCLK frequency</td>
<td></td>
<td>400 kHz</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T_{SKH}</td>
<td>SCL high time</td>
<td>0.6</td>
<td></td>
<td></td>
<td>us</td>
</tr>
<tr>
<td>T_{SKL}</td>
<td>SCL low time</td>
<td>1.3</td>
<td></td>
<td></td>
<td>us</td>
</tr>
<tr>
<td>T_{SHT}</td>
<td>Start hold time</td>
<td>0.6</td>
<td></td>
<td></td>
<td>us</td>
</tr>
<tr>
<td>T_{STS}</td>
<td>Start setup time</td>
<td>0.6</td>
<td></td>
<td></td>
<td>us</td>
</tr>
<tr>
<td>T_{SPS}</td>
<td>Stop setup time</td>
<td>0.6</td>
<td></td>
<td></td>
<td>us</td>
</tr>
<tr>
<td>T_{BUS}</td>
<td>Bus free time between stop and start</td>
<td>1.3</td>
<td></td>
<td></td>
<td>us</td>
</tr>
<tr>
<td>T_{DS}</td>
<td>SDA setup time</td>
<td>100</td>
<td></td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T_{DH}</td>
<td>SDA hold time</td>
<td>100</td>
<td></td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T_{VD}</td>
<td>SDA valid time</td>
<td>0.9</td>
<td></td>
<td></td>
<td>us</td>
</tr>
<tr>
<td>T_{VT}</td>
<td>SDA acknowledge time</td>
<td>0.9</td>
<td></td>
<td></td>
<td>us</td>
</tr>
</tbody>
</table>

Table 3: I²C Interface Specifications
ABSOLUTE MAXIMUM RATINGS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min.</th>
<th>Max.</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_D$</td>
<td>Voltage on VCC with respect to GND</td>
<td>-0.3</td>
<td>4.2</td>
<td>V</td>
</tr>
<tr>
<td>$V_{i2c}$</td>
<td>Voltage on SDA or SCL pin with respect to GND</td>
<td>-0.3</td>
<td>3.9</td>
<td>V</td>
</tr>
<tr>
<td>$T_{AT(1)}$</td>
<td>Ambient temperature under bias</td>
<td>-55</td>
<td>125</td>
<td>°C</td>
</tr>
<tr>
<td>$T_{ST}$</td>
<td>Storage temperature</td>
<td>-55</td>
<td>150</td>
<td>°C</td>
</tr>
</tbody>
</table>

Table 4: DSTH01 Maximum Ratings

Notes: For best accuracy, the DSTH01 module should be stored in climate controlled conditions (10 to 35°C, 20 to 60%RH). Exposure to high temperature and/or high humidity environments can cause a small upwards shift in RH readings.

HOST INTERFACE

1. I2C Interface

The DSTH01 sensor module has an I2C serial interface with a 7-bit address of 0x40. It is a slave device supporting data transfer with rates up to 400 kHz. The table 5 shows the register summary of the DSTH01 modules. Users can configure related values to obtain corresponding temperature and humidity parameters.
Please note that any register address which is not listed here is reserved and must not be written. The Reserved Register Bits (RSVD) must always written as zero; the result of a read operation on these bits is undefined.

2. Performing a Relative Humidity Measurement

The following steps must be executed in sequence in order to take a relative humidity measurement.

1. Set START bit (bit0) and clear TEMP bit (bit4) in CONFIG register (0x03) to begin a new conversion, i.e., write CONFIG (0x03) with value 0x01

2. Poll RDY(D0) in STATUS register (0x00) until it is low (=0).

3. Read the upper and lower bytes of the RH value from DATAh and DATAl registers (0x01 & 0x02) respectively.

<table>
<thead>
<tr>
<th>DATAh</th>
<th>DATAl</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>D6</td>
</tr>
<tr>
<td>D5</td>
<td>D4</td>
</tr>
<tr>
<td>D3</td>
<td>D2</td>
</tr>
<tr>
<td>D1</td>
<td>D0</td>
</tr>
<tr>
<td></td>
<td>D7</td>
</tr>
<tr>
<td>D6</td>
<td>D5</td>
</tr>
<tr>
<td>D4</td>
<td>D3</td>
</tr>
<tr>
<td>D2</td>
<td>D1</td>
</tr>
<tr>
<td>D0</td>
<td></td>
</tr>
</tbody>
</table>

12-bit relative humidity code 0 0 0 0 0

Table 6: 12-bit Relative Humidity Result in Registers DATAh and DATAl

(4). Convert the RH value to %RH with the equation: \( \%RH = \frac{RH}{16} - 24 \), which the RH is the measured value returned in DATAh & DATAl.

(5). Apply temperature compensation and/or linearization which will be discussed in the following section.

The table below shows the values that corresponding to various measured RH levels.

<table>
<thead>
<tr>
<th>%RH</th>
<th>12 Bit Code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dec</td>
</tr>
<tr>
<td>0</td>
<td>384</td>
</tr>
<tr>
<td>10</td>
<td>544</td>
</tr>
<tr>
<td>20</td>
<td>704</td>
</tr>
<tr>
<td>30</td>
<td>864</td>
</tr>
<tr>
<td>40</td>
<td>1024</td>
</tr>
<tr>
<td>50</td>
<td>1184</td>
</tr>
<tr>
<td>60</td>
<td>1344</td>
</tr>
<tr>
<td>70</td>
<td>1504</td>
</tr>
<tr>
<td>80</td>
<td>1664</td>
</tr>
<tr>
<td>90</td>
<td>1824</td>
</tr>
<tr>
<td>100</td>
<td>1984</td>
</tr>
</tbody>
</table>

Table 7: Typical %RH Measurement Codes for 0 to 100% RH Range

The above sequence assumes to be in normal mode, Tconv=35ms (typical). Conversion also can be performed in fast mode.
3. Performing a Temperature Measurement

(1). Set START bit (bit0) and TEMP bit (bit4) in CONFIG register (0x03) to begin a new conversion, i.e.:
   write CONFIG (0x03) with value 0x11

(2). Poll RDY(D0) in STATUS register (0x00) until it is low (=0).

(3). Read the upper and lower bytes of the RH value from DATAh and DATAl registers (0x01 & 0x02) respectively.

<table>
<thead>
<tr>
<th>DATAh</th>
<th>DATAl</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>D6</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>12-bit relative humidity code</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 8: 14-bit Temperature Result in Registers DATAh and DATAl

(4). Convert the value to temperature with the equation: \( \text{Temperature}({}^\circ\text{C}) = \frac{\text{TEMP}}{32} - 50 \), which the TEMP is the measured value returned in DATAh & DATAl.

<table>
<thead>
<tr>
<th>TEMP(°C)</th>
<th>14 Bit Code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dec</td>
</tr>
<tr>
<td>-40</td>
<td>320</td>
</tr>
<tr>
<td>-30</td>
<td>640</td>
</tr>
<tr>
<td>-20</td>
<td>960</td>
</tr>
<tr>
<td>-10</td>
<td>1280</td>
</tr>
<tr>
<td>0</td>
<td>1600</td>
</tr>
<tr>
<td>10</td>
<td>1920</td>
</tr>
<tr>
<td>20</td>
<td>2240</td>
</tr>
<tr>
<td>30</td>
<td>2560</td>
</tr>
<tr>
<td>40</td>
<td>2880</td>
</tr>
<tr>
<td>50</td>
<td>3200</td>
</tr>
<tr>
<td>60</td>
<td>3520</td>
</tr>
<tr>
<td>70</td>
<td>3840</td>
</tr>
<tr>
<td>80</td>
<td>4160</td>
</tr>
<tr>
<td>90</td>
<td>4480</td>
</tr>
<tr>
<td>100</td>
<td>4800</td>
</tr>
</tbody>
</table>

Table 9: Typical Temperature Measurement Codes for -40°C to 100°C Range

4. Normal Conversion Mode and Fast Conversion Mode

The switch between two modes is realized by setting the value of FAST (bit5) in CONFIG register (0x03). Fast=0 is normal mode and Fast=1 is fast mode.

<table>
<thead>
<tr>
<th>Mode</th>
<th>( T_{\text{CON}} ) (Typical)</th>
<th>Temperature Resolution</th>
<th>Humidity resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal Mode</td>
<td>35ms</td>
<td>14-bit</td>
<td>12-bit</td>
</tr>
<tr>
<td>Fast Mode</td>
<td>18ms</td>
<td>13-bit</td>
<td>11-bit</td>
</tr>
</tbody>
</table>

Table 10: Normal Conversion Mode vs Fast Conversion Mode
5. **Heater**

The sensor chip on DSTH01 module integrates a resistive heating element which may be used to raise the temperature of the humidity sensor. This element can be used to drive off condensation or to implement dew-point measurement when the module is used in conjunction with a separate temperature sensor such as another DSTH01 module.

The heater can be activated by setting HEAT (D1) in CONFIG (register 0x03). Turning on the heater will reduce the tendency of the humidity sensor to accumulate an offset due to “memory” of sustained high humidity conditions. When the heater is enabled, the reading of the on-chip temperature sensor will be affected (increased).

6. **I\(^2\)C Operation**

If the DSTH01 module shares the I\(^2\)C bus with other slave devices, it should be powered down when the master controller is communicating with the other slave devices, which can be realized either by setting /CS to logic high or setting the VCC pin to 0V. User can power off the module by using GPIO pin to control the VCC of DSTH01 module. Please note that users must consider the driving current of GPIO when using the heater function of the module, in which function the current of the enabled heater might consumes the current more than 30mA (seeing the table of ELECTRIC SPECIFICATIONS).

<table>
<thead>
<tr>
<th>A6</th>
<th>A5</th>
<th>A4</th>
<th>A3</th>
<th>A2</th>
<th>A1</th>
<th>A0</th>
<th>R/W</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1/0</td>
</tr>
</tbody>
</table>

**Table 11: I\(^2\)C Slave Address Byte**

(1). **I\(^2\)C write operation**

To write a register on DSTH01 module, the master should issue a start command (S) followed by the slave address-0x40. The slave address should be followed by a 0 to indicate that the operation is a write. Upon recognizing its slave address, the DSTH01 issues an acknowledge (A) by pulling the SDA line low for the high duration of the ninth SCL cycle. The next byte which the master places on the bus is the register address pointer, selecting the register on the DSTH01 to which the data should be transferred. After the DSTH01 acknowledges this byte, the master places a data byte on the bus. This byte will be written to the register selected by the address pointer. The DSTH01 module will acknowledge the data byte, after which the master issues a Stop command (P).

**Table 12: Identification for Master and Slave Data**

<table>
<thead>
<tr>
<th>S</th>
<th>Slave Address</th>
<th>W</th>
<th>A</th>
<th>Address Pointer</th>
<th>A</th>
<th>Register Data</th>
<th>A</th>
<th>P</th>
</tr>
</thead>
</table>

**Table 13: Sequence to Write a Register**
(2). I²C read operation

To read a register on the DSTH01 module, the master must first set the address pointer to indicate the register from which the data is to be transferred. The master should issue a start command (S) followed by the slave address---0x40. The slave address is followed by a 0 to indicate that the operation is a write. Upon recognizing its slave address, the DSTH01 will issue an acknowledge (A) by pulling the SDA line low for the high duration of the ninth SCL cycle. The next byte the master places on the bus is the register address pointer selecting the register on the DSTH01 from which the data should be transferred. After the DSTH01 acknowledges this byte, the master issues a repeated start command (Sr) indicating that a new transfer is to take place. The DSTH01 is addressed once again with the R/W bit set to 1, indicating a read operation. The DSTH01 will acknowledge its slave address and output data from the previously-selected register onto the data bus under the control of the SCL signal, the master should not acknowledge (A) the data byte and issue a stop (P) command. However, if a RH or Temperature conversion result (two bytes) is to be read, the master should acknowledge (A) the first data byte and continue to activate the SCL signal. The DSTH01 will automatically output the second data byte.

Upon receiving the second byte, the master should issue a not Acknowledge (A) followed by a stop command.
APPLICATION CODES

This section shows the basic communication between DTH01 and STM8 microcontroller through I2C interface. Reading temperature and relative humidity parameters are demonstrated in the codes.

```
#include "stm8110x.h"

#define SDA_H    GPIO_SetBits(GPIOC, GPIO_Pin_0)
#define SDA_L    GPIO_ResetBits(GPIOC, GPIO_Pin_0)
#define SCK_H    GPIO_SetBits(GPIOC, GPIO_Pin_1)
#define SCK_L    GPIO_ResetBits(GPIOC, GPIO_Pin_1)
#define SDA      GPIO_ReadInputDataBit(GPIOC,GPIO_Pin_0)
#define SlaveAddress  0x40
#define RegisterAddress0  0x00
#define RegisterAddress1  0x01
#define RegisterAddress2  0x02
#define RegisterAddress3  0x03
#define RegisterAddress11  0x11

void CLK_INT(void);
void GPIO_INT(void);
void DELAY(uint16_t n);
void I2C_START(void);
void I2C_STOP(void);
void I2C_WRITE(uint8_t Data);
uint8_t I2C_READ(void);
void I2C_ACK(uint8_t a);
uint8_t I2C_SEND(uint8_t SlaveAdd,uint8_t RegisterAdd,uint8_t *s);
uint8_t I2C_RECEIVE(uint8_t SlaveAdd,uint8_t RegisterAdd,uint8_t *s);
void Temperature_READ(uint8_t *s);
void RelativeHumidity_READ(uint8_t *s);

uint8_t Ack;
uint8_t id;
uint8_t RelativeHumidity;
uint8_t Temperature;
uint8_t Start1=0x01;
uint8_t Start2=0x11;
```
void main()
{
    CLK_INT();
    GPIO_INT();
    I2C_RECEIVE(SlaveAddress,RegisterAddress11,&id);

    while (1)
    {
        RelativeHumidity_READ(&RelativeHumidity);
        Temperature_READ(&Temperature);
    }
}

void CLK_INT()
{
    CLK_DeInit();
    DELAY(500);
    CLK_CCOCmd(ENABLE);
    CLK_MasterPrescalerConfig(CLK_MasterPrescaler_HSIDiv1);
}

void GPIO_INT()
{
    GPIO_DeInit(GPIOC);
    GPIO_Init(GPIOC,GPIO_Pin_1,GPIO_Mode_Out_PP_Low_Slow);
    GPIO_Init(GPIOC,GPIO_Pin_0,GPIO_Mode_Out_OD_Low_Slow);
}

void DELAY(uint16_t n)
{
    uint8_t i;
    while(n--)
    {
        for(i=0;i<16;i++);
    }
}

void I2C_START()
{
    SDA_H;
    SCK_H;
    SDA_L;
    SCK_L;
void I2C_STOP()
{
    SDA_L;
    SCK_H;
    SDA_H;
}

void I2C_WRITE(uint8_t Data)
{
    uint8_t i;
    for(i=0;i<8;i++)
    {
        if(Data & 0x80)
            SDA_H;
        else
            SDA_L;
        SCK_H;
        SCK_L;
        Data<<=1;
    }
    SDA_H;
    SCK_H;
    if(SDA==1)
        Ack=0;
    else
        Ack=1;
    SCK_L;
}

uint8_t I2C_READ()
{
    uint8_t Data=0;
    uint8_t i;
    SDA_H;
    for(i=0;i<8;i++)
    {
        SCK_L;
        SCK_H;
        Data<<=1;
        if(SDA==1)
            Data=Data+1;
    }
void I2C_ACK(uint8_t a)
{
    if(a==0)
        SDA_L;
    else
        SDA_H;
    SCK_H;
    SCK_L;
}

uint8_t I2C_SEND(uint8_t SlaveAdd,uint8_t RegisterAdd,uint8_t *s)
{
    SlaveAdd=SlaveAdd<<1;
    I2C_START();
    I2C_WRITE(SlaveAdd);
    if(Ack==0)
        return 0;
    I2C_WRITE(RegisterAdd);
    if(Ack==0)
        return 0;
    I2C_WRITE(*s);
    if(Ack==0)
        return 0;
    I2C_STOP();
    return 1;
}

uint8_t I2C_RECEIVE(uint8_t SlaveAdd,uint8_t RegisterAdd,uint8_t *s)
{
    SlaveAdd=SlaveAdd<<1;
    I2C_START();
    I2C_WRITE(SlaveAdd);
    if(Ack==0)
        return 0;
    I2C_WRITE(RegisterAdd);
    if(Ack==0)
        return 0;
    I2C_START();
I2C_WRITE(SlaveAdd+1);
    if(Ack==0)
        return 0;
    *s=I2C_READ();
    I2C_ACK(1);
    I2C_STOP();
    return 1;
}

void RelativeHumidity_READ(uint8_t *s)
{
    uint8_t Status=1;
    uint8_t RelativeHumidityH;
    uint8_t RelativeHumidityL;
    uint16_t RelHum;
    I2C_SEND(SlaveAddress,RegisterAddress3,&Start1);
    while(Status==1)
    {
        I2C_RECEIVE(SlaveAddress,RegisterAddress0,&Status);
    }
    I2C_RECEIVE(SlaveAddress,RegisterAddress1,&RelativeHumidityH);
    I2C_RECEIVE(SlaveAddress,RegisterAddress2,&RelativeHumidityL);
    RelHum=RelativeHumidityH;
    RelHum=RelHum<<8;
    RelHum+=RelativeHumidityL;
    RelHum=RelHum>>4;
    *s=RelHum/16-24;
}

void Temperature_READ(uint8_t *s)
{
    uint8_t Status=1;
    uint8_t TemperatureH;
    uint8_t TemperatureL;
    uint16_t Temp;
    I2C_SEND(SlaveAddress,RegisterAddress3,&Start2);
    while(Status==1)
    {
        I2C_RECEIVE(SlaveAddress,RegisterAddress0,&Status);
    }
    I2C_RECEIVE(SlaveAddress,RegisterAddress1,&TemperatureH);
    I2C_RECEIVE(SlaveAddress,RegisterAddress2,&TemperatureL);
    Temp=TemperatureH;
Temp<<8;
Temp+=TemperatureL;
Temp>>=2;
*s=Temp/32-50;

MECHANICAL DATA
Unit: mm

Figure 5: DSTH01 Dimensions
<table>
<thead>
<tr>
<th>Dorji Industrial Group Co., Ltd reserves the right to make corrections, modifications, improvements and other changes to its products and services at any time and to discontinue any product or service without notice. Customers are expected to visit websites for getting newest product information before placing orders.</th>
</tr>
</thead>
<tbody>
<tr>
<td>These products are not designed for use in life support appliances, devices or other products where malfunction of these products might result in personal injury. Customers using these products in such applications do so at their own risk and agree to fully indemnify Dorji Industrial Group for any damages resulting from improper use.</td>
</tr>
</tbody>
</table>

Dorji Applied Technologies
A division of **Dorji Industrial Group Co., Ltd**

Add.: Xinchenhuayuan 2, Dalangnanlu, Longhua, Baoan district, Shenzhen, China 518109
Tel: 0086-755-28156122
Fax.: 0086-755-28156133
Email: sales@dorji.com
Web: http://www.dorji.com