Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

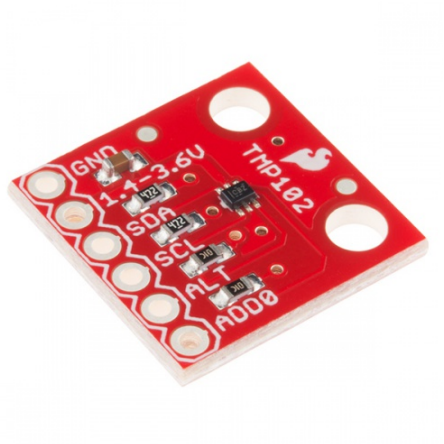

## Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

# TMP102 Digital Temperature Sensor Hookup Guide

## Introduction

The TMP102 is an easy-to-use digital temperature sensor from Texas Instruments. While some temperature sensors use an analog voltage to represent the temperature, the TMP102 uses the I²C bus of the Arduino to communicate the temperature.



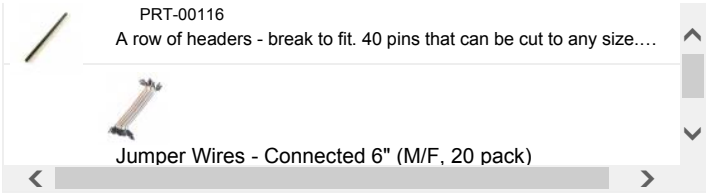

### SparkFun Digital Temperature Sensor Breakout - TMP102

SEN-13314

## Required Materials

To follow along with this hookup guide, you will need the following:

| **TMP102 Hookuo Guide** SparkFun Wish List | |
|---|---|
|  | SparkFun Digital Temperature Sensor Breakout - TMP102<br>SEN-11931<br>This is a breakout board for the incredibly small TMP102 digital temp… |
|  | SparkFun RedBoard - Programmed with Arduino<br>DEV-12757<br>At SparkFun we use many Arduinos and we're always looking for the… |
| | Break Away Headers - Straight |



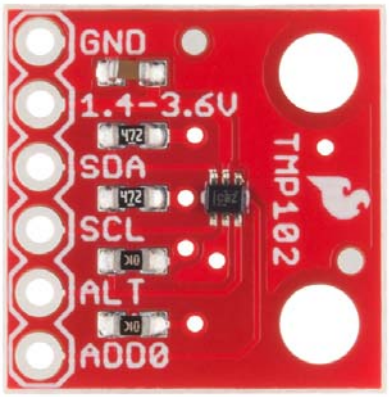

## Suggested Reading

Before getting started, you may find the following links useful:

- I²C Protocol
- Logic Levels
- Installing an Arduino Library
- What are Pull-up Resistors?
- How to use a Breadbaord

# Board Overview

Let's go over the TMP102 Breakout in detail.




### TMP102 Details:

- Uses the I²C interface
- 12-bit, 0.0625°C resolution
- Typical temperature accuracy of ±0.5°C
- 3.3V sensor - use inline logic level converters or 10 kΩ resistors to limit 5V signals
- Supports up to four TMP102 sensors on the I²C bus at a time

## Pull-up Resistors

This breakout board has built-in 4.7 kΩ pull up resistors for I²C communications. If you're hooking up multiple I²C devices on the same bus, you may want to disable/enable the pull-up resistors for one or more boards. On the TMP102, the pull-ups are enabled by default. To disable them, simply use a hobby knife to cut the traces connecting the left and right pads of the jumper labeled **I2C PU** on the back of the board. This will disconnect the resistors from VCC and from the I²C bus.

# Hardware Connections

## Connecting the TMP102 to an Arduino

Wiring the TMP102 is very easy! We recommend soldering six male headers to the breakout board. You can also solder wires to fit your application's needs.

## Power

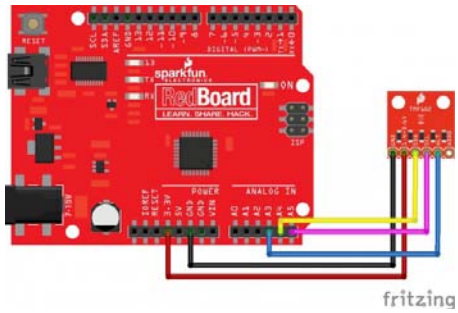
This board runs from **1.4V to 3.6V**. Be sure to power the board from the 3.3V pin! I²C uses an open drain signaling, so there is no need to use level shifting; the 3.3V signal will work to communicate with the Arduino and will not exceed the maximum voltage rating of the pins on the TMP102.

## Connections to the Arduino

The TMP102 breakout board has six pins, however we'll only be using five of the pins in today's example. We'll be connecting VCC and GND to the normal power pins, two data lines for I²C communication, and one digital pin to see if there is an alert. If you're using a newer board that has SDA and SCL broken out, you can connect the SDA and SCL pins directly to those pins. If you're using an older board, SDA and SCL are pins A4 and A5 respectively.

- VCC → 3.3V
- GND → GND
- SDA → SDA/A4
- SCL → SCL/A5
- ALT → A3

This would looks something like this:




The only pin that we aren't using is **ADD0**, this pin is used to change the address of the TMP102. If you're using multiple TMP102s or another device that uses that address, you'll want to use this pin to change the address. The **default address is 0x48**. You can change the address by cutting the ADD0 jumper on the back of the board and connecting an external jumper wire to the following pins:

- VCC → 0x49
- SDA → 0x4A
- SCL → 0x4B

# TMP102 Library and Example Code

> **Note:**This example assumes you are using the latest version of the Arduino IDE on your desktop. If this is your first time using Arduino, please review ou tutorial on installing the Arduino IDE. If you have not previously installed an Arduino library, please check out our installation guide.

To get started immediately, use the example code and library files below.

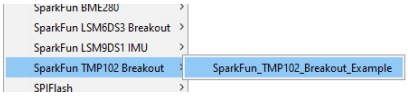
```
/*************************************************************
****************
TMP102_example.ino
Example for the TMP102 I2C Temperature Sensor
Alex Wende @ SparkFun Electronics
April 29th 2016
~

This sketch configures the TMP102 temperature sensor and print
s the
temperature and alert state (both from the physical pin, as we
ll as by
reading from the configuration register.

Resources:
Wire.h (included with Arduino IDE)
SparkFunTMP102.h

Development environment specifics:
Arduino 1.0+
Hardware Version 13

This code is beerware; if you see me (or any other SparkFun em
ployee) at
the local, and you've found our code helpful, please buy us a
round!

Distributed as-is; no warranty is given.
*************************************************************
***************/

#include <Wire.h> // Used to establied serial communication o
n the I2C bus
#include "SparkFunTMP102.h" // Used to send and recieve specif
ic information from our sensor

// Connections
// VCC = 3.3V
// GND = GND
// SDA = A4
// SCL = A5
const int ALERT_PIN = A3;

TMP102 sensor0(0x48); // Initialize sensor at I2C address 0x48
// Sensor address can be changed with an external jumper to:
// ADD0 - Address
//  VCC - 0x49
//  SDA - 0x4A
//  SCL - 0x4B

void setup() {
  Serial.begin(9600); // Start serial communication at 9600 ba
ud
  pinMode(ALERT_PIN,INPUT);  // Declare alertPin as an input
  sensor0.begin();  // Join I2C bus

  // Initialize sensor0 settings
  // These settings are saved in the sensor, even if it loses
power

  // set the number of consecutive faults before triggering al
arm.
  // 0-3: 0:1 fault, 1:2 faults, 2:4 faults, 3:6 faults.
  sensor0.setFault(0);  // Trigger alarm immediately
```

```
  // set the polarity of the Alarm. (0:Active LOW, 1:Active HI
GH).
  sensor0.setAlertPolarity(1); // Active HIGH

  // set the sensor in Comparator Mode (0) or Interrupt Mode
(1).
  sensor0.setAlertMode(0); // Comparator Mode.

  // set the Conversion Rate (how quickly the sensor gets a ne
w reading)
  //0-3: 0:0.25Hz, 1:1Hz, 2:4Hz, 3:8Hz
  sensor0.setConversionRate(2);

  //set Extended Mode.
  //0:12-bit Temperature(-55C to +128C) 1:13-bit Temperature
(-55C to +150C)
  sensor0.setExtendedMode(0);

  //set T_HIGH, the upper limit to trigger the alert on
  sensor0.setHighTempF(85.0);  // set T_HIGH in F
  //sensor0.setHighTempC(29.4); // set T_HIGH in C

  //set T_LOW, the lower limit to shut turn off the alert
  sensor0.setLowTempF(84.0);  // set T_LOW in F
  //sensor0.setLowTempC(26.67); // set T_LOW in C
}

void loop()
{
  float temperature;
  boolean alertPinState, alertRegisterState;

  // Turn sensor on to start temperature measurement.
  // Current consumtion typically ~10uA.
  sensor0.wakeup();

  // read temperature data
  temperature = sensor0.readTempF();
  //temperature = sensor0.readTempC();

  // Check for Alert
  alertPinState = digitalRead(ALERT_PIN); // read the Alert fr
om pin
  alertRegisterState = sensor0.alert();   // read the Alert fr
om register

  // Place sensor in sleep mode to save power.
  // Current consumtion typically <0.5uA.
  sensor0.sleep();

  // Print temperature and alarm state
  Serial.print("Temperature: ");
  Serial.print(temperature);

  Serial.print("\tAlert Pin: ");
  Serial.print(alertPinState);

  Serial.print("\tAlert Register: ");
  Serial.println(alertRegisterState);

  delay(1000);  // Wait 1000ms
}
```

You can download the library from the link below.

TMP102 ARDUINO LIBRARY

Once the library is installed, open Arduino, and expand the examples menu. You should see the TMP102 example.




## TMP102 Functions

### Main functions

These are functions used to read settings and temperatures from the sensor.

`TMP102::readTempC()` - Returns the current temperature in Celsius.

`TMP102::readTempF()` - Returns the current temperature in Fahrenheit.

`TMP102::readLowTempC(float temperature)` - Reads T_LOW register in Celsius.

`TMP102::readHighTempC(float temperature)` - Reads T_HIGH register in Celsius.

`TMP102::readLowTempF(float temperature)` - Reads T_LOW register in Fahrenheit.

`TMP102::readHighTempF(float temperature)` - Reads T_HIGH register in Fahrenheit.

`TMP102::sleep()` - Put TMP102 in low power mode (<0.5 uA).

`TMP102::wakeup()` - Return to normal power mode (~10 uA). When the sensor powers up, it is automatically running in normal power mode, and only needs to be used after `TMP102::sleep()` is used.

`TMP102::alert()` - Returns the state of the Alert register. The state of the register is the **same as the ALT pin**.

### Nonvolatile Functions

These are settings that are saved in the sensor, even after power is removed.

`TMP102::setLowTempC(float temperature)` - Sets T_LOW (in Celsius) alert threshold.

`TMP102::setHighTempC(float temperature)` - Sets T_HIGH (in Celsius) alert threshold.

`TMP102::setLowTempF(float temperature)` - Sets T_LOW (in Fahrenheit) alert threshold.

`TMP102::setHighTempF(float temperature)` - Sets T_HIGH (in Fahrenheit) alert threshold.

`TMP102::setConversionRate(byte rate)` - Sets the temperature reading conversion rate. 0: 0.25Hz, 1: 1Hz, 2: 4Hz (default), 3: 8Hz.

`TMP102::setExtendedMode(byte mode)` - Enable or disable extended mode. 0: disabled (-55C to +128C), 1: enabled (-55C to +150C).

`TMP102::setAlertPolarity(bool polarity)` - Sets the polarity of the alert. 0: active LOW, 1: active HIGH

`TMP102::setFault(byte faultSetting)` - Sets the number of consecutive faults before triggering alert. 0: 1 fault, 1: 2 faults, 2: 4 faults, 3: 6 faults.

`TMP102::setAlertMode(bool mode)` - Sets the type of alert. 0: Comparator Mode (Active from when temperature > T_HIGH until temperature < T_LOW), 1: Thermostat mode (Active from when temperature > T_HIGH until any read operation occurs.

## Resources and Going Further

For more information about the TMP102 Breakout, check out the links below.

- TMP102 datasheet
- TMP102 Breakout Board Schematic
- TMP102 Breakout Board Eagle Files
- Github repo containing the latest and greatest files and code.

For more sensor fun, check out these other great SparkFun tutorials.



### ITG-3200 Hookup Guide

Learn how to interact with the ITG-3200 Triple Axis Gyroscope.



### MAG3110 Magnetometer Hookup Guide

Get started with the MAG3110 3-Axis Magnetometer and learn how to make your own digital compass that senses the Earth's magnetic fields.



### FLIR Lepton Hookup Guide

See the invisible world of infrared radiation using the FLIR Dev Kit and Raspberry Pi.



### MPU-9250 Hookup Guide

Get up and running with the MPU-9250 9-axis MEMS sensor.