TI TMP102: A Brief Guide

Version 1.3

By Murtadha AlSaeedi

روابـي القابضة
Rawabi Holding

Revision History

Date	Rev	Details
February 10, 2014	1.2	Release
March 4, 2014	1.3	Added: Address assignment section, addresses in hex. Interface section is corrected regarding the ALT signal functionality

Table of Contents

1. Description

The TMP102 is a temperature sensor that supports I2C bus and SMBus. It requires no external components, and capable of reading temperatures to a resolution of 0.0625 C. TMP102 is an ideal sensor for extended temperature measurement in environmental, industrial, and instrumentation applications. Up to four TMP102 sensors can be connected to one I2C bus.

2. Specifications

Accuracy: 0.5 to 2 C error (-25 C to +85 C), 1 to 3 C error (-40 C to +125 C)
Operating Temperature: -55 to 150 C
Supply Range: 1.4 V to 3.6 V
Resolution: 12 bits
Digital output: Two-Wire serial interface
Supported I2C modes supported: Standard-mode, Fast-mode, High-speed mode.

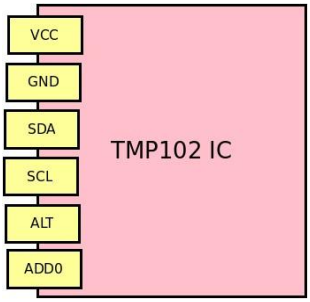

Figure 1: Pin Configuration

3. Interface

The TMP102 has 6 pins. Vcc and GND are used to supply voltage. The SDA pin should be connected to the SDA line of the I2C bus, and SCL should be connected to the SCL line. ALT is the pin used for the alert signal, it can be used if desired, otherwise keep it floating. A signal is triggered on the ALT pin when the temperature reaches a high limit specified in a special register on the TMP102. Pin ADD0 is used to connect more than one TMP102 sensor to the bus, by connecting it to a voltage source or ground.

4. Electrical Characteristics

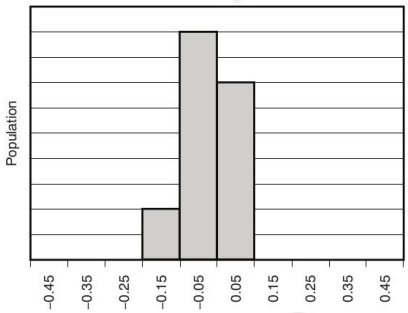

Figure 2: Temperature Error at +25 C

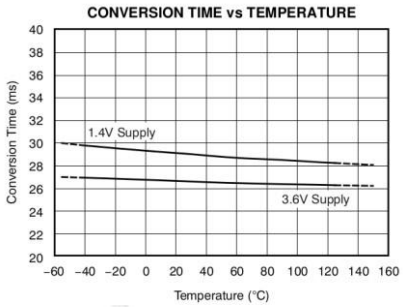

Figure 3: Conversion Time vs Temperature

Figure 2 shows that when reading temperatures above 25 Celsius, the error fraction is between -0.15 Volts and 0.05 Volts. Figure 3 depicts that the conversion time decreases as the temperature increases.

5. Application Information

The TMP102 has four 16-bit data registers: temperature register, configuration register, high temperature register, and low temperature register. Each of these registers can be accessed individually through a pointer register, table 1 shows each register's address. Issuing a read operation to the TMP102, will allow it to send the value stored in the data register selected by the pointer register.

Register Name	Address
Temperature Register	00000000 (0x0)
Configuration Register	00000001 (0x1)
High Temperature Register	00000010 (0x2)
Low Temperature Register	00000011 (0x3)

Table 1: Reserved Addresses

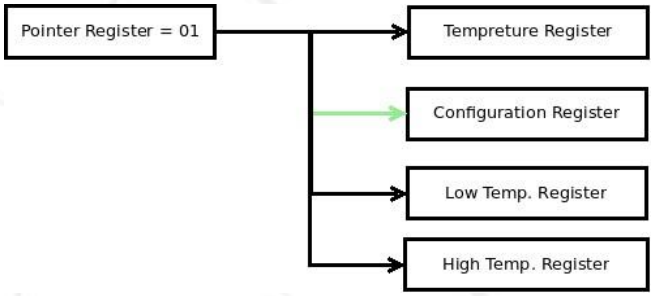

Figure 4: Registers Structure

5.1. Mechanism

5.1.1. Address Assignment

The address of the TMP102 can be configured by connecting pin ADD0 to ground, V_{DD}, SCL or SDA. Table 2 shows all the possible addresses for the device.

Pin ADD0 Connection		Device Address
Ground		1001000 (0x48)
VDD		1001001 (0x49)
SDA		1001010 (0x4A)
SCL		1001011 (0x4B)

Table 2: Address Assignment

5.1.2. Reading from a Register

When performing a read operation on the TMP102, the last value written to the pointer register is used to determine which register is read by a read operation. On power-up the default value of the pointer register is 0, which points to the temperature register. To read from a different register, you must first issue a write command to the TMP102 (send the address of the device followed by a 0), and then send the address of the register desired

to the bus. After that, you can issue a read command to the TMP102, to get values from the desired register directly. There is no need to write to the pointer register again if you want to read from the same register.

5.1.3. Writing to a Register

Every write operation requires a value to the pointer register. To write to a specific register, you must first issue a write command to the device. Then you send the address of the desired register to overwrite the value on the pointer register. After that, you send two bytes of data, to overwrite the desired 16-bit register.

5.2. Registers Description

5.2.1. Pointer Register

The pointer register is used to address one of the four data registers on the device. It is an 8-bit register, however, only the least significant two bits are used to point to one of the four data registers. The default value of the pointer register on power-up is 0, which points to the temperature register. To read from or write to a different register, you must overwrite the value on the pointer register. For example, if you want to write a value to the high temperature register, you must first write its address ($0000\ 0011_2$) on the pointer register.

5.2.2. Temperature Register

The temperature register is a read only register that stores the most recent conversion (current temperature). The temperature value stored on this register is 12 bits, the most significant 8 bits are the decimal part, while the next four bits are the fraction part.

5.2.3. Configuration Register

The configuration register is used to configure the operational modes of the TMP102. Table 2 shows the default values of the configuration bits, only some of them will be described in this document. The EM bit (Extended Mode) is used to specify the resolution of the values in temperature registers. The bits CR0 and CR1 are used to specify the conversion rate (how many temperature readings/conversion per second).

Configuration Register															
BYTE 1								BYTE 2							
OS	R1	R0	F1	F0	POL	TM	SD	CR1	CR0	AL	EM	-	-	-	-
0	1	1	0	0	0	0	0	1	0	1	0	0	0	0	0

Table 3: Default Configuration Register Setting

EX Bit	Temperature Resolution
0	12 bits (default)
1	13 bits

Table 4: Extended Mode (EX) Settings

CR1	CR0	Conversion Rate
0	0	0.25 Hz
0	1	1 Hz
1	0	4 Hz (default)
1	1	8 Hz

Table 5: Conversion Rate Settings

5.2.4. High and Low-Limit Registers

When the temperature exceeds the value stored in the high temperature register, an alert signal is generated on the ALT pin. The ALT pin is always 1 by default when there is no alert signal, pulses of 0 appear on it when the temperature reaches or exceeds the high limit. The alert signal keeps pulsing until the temperature falls below the low limit stored in the low temperature register.

5.3. TMP102 Propeller Example

```
/** Company: Rawabi United Safety Services
Engineer: Murtadha Al-Saeedi

Description: This code is written for the TMP102. The code makes use
of the alert functionality and performs continues reading. The code
first write the high temperature limit to the h-limit register. After that it
performs continuous temperature
reading, and trigger a buzzer when the temperature reaches the high
limit.
**/
```

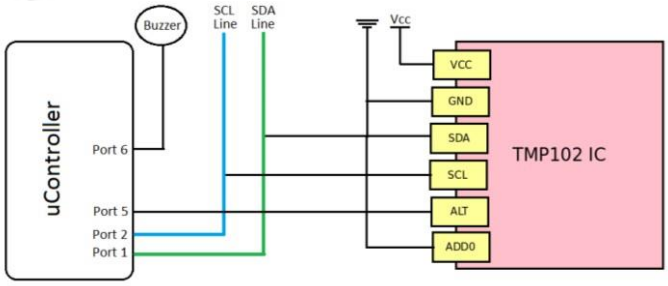

Figure 5: TMP102 Controlled by a Micro-Controller

```
#include "simpletools.h" // Include simple tools
#include "i2c.h" // Include the i2c library
unsigned int stack[(160 + (50 * 4)) / 4];

void alert(void *par); int main() // Main function
{
  int cog = cogstart(&alert, NULL, stack, sizeof(stack)); //assigning the alert function to a separate cog    i2c* bus;
  pause(1000); //to make sure we can see the readings printed on the terminal screen    bus = i2c_open
(bus, 2, 1, 0); //establising i2c lines on port 2 (SCL) and port 1 (SDA)    i2c_start(bus); //sending the start
bit to the bus
  if(i2c_writeByte(bus, 0x90) == 0) //sending the address of the TMP102 (1001_000) and the write bit (0)    printf("ack\n");
  if(i2c_writeByte(bus, 0x3) == 0) //sending the address of the high-limit register (0000_0010)    printf("ack\n");
  if(i2c_writeByte(bus, 0x1E) == 0) //sending the high limit we want to store (30 C)
printf("ack\n");    i2c_stop(bus); //sending the stop bit to the bus    i2c_start(bus); //sending the
start bit to the bus
  if(i2c_writeByte(bus, 0x90) == 0) //sending the address of the TMP102 (1001_000) and the write bit (0)    printf("ack\n");
  if(i2c_writeByte(bus, 0x0) == 0) //sending the address of the temperature register (0000_0000)    printf("ack\n");
  i2c_stop(bus); //sending the stop bit to the bus    while(1)
  { i2c_start(bus); //sending the start bit to the bus
            if(i2c_writeByte(bus, 0x91) == 0); //sending the address of the TMP102 (1001_000) and the read bit (0) printf("ack\n");
            printf("Temperature in Celsius: %d", i2c_readByte(bus, 0)); //getting the first byte printf(".%d",
            i2c_readByte(bus, 0)); //getting the second byte i2c_stop(bus); //sending the stop bit to the
            bus printf("\1");
            pause(300);
  }
  return 0;
}

void alert(void *par) //the alert function
{
  while(1)
  { if(input(5) == 0) high(6); else
            if(input(5) == 1) low(6);
  }
}
```