EARTH PEOPLE TECHNOLOGY

EPT-200TMP-TS-U2 Temperature Sensor Docking Board User Manual

The EPT-200TMP-TS-U2 is a temperature sensor mounted on a docking board. The board is designed to fit onto the Arduino Uno platform. It is compatible with the +5V Arduino's and requires no external hook wires. Just plug the EPT-200TMP-TS-U2 into the Arduino and load the code. The TMP102 temperature sensor power and communications are provided by the docking board connectors from the Arduino.

The EPT-200TMP-TS-U2 is designed for applications that require a robust connection between the sensor and the Arduino platform. These are applications where a bread board and hook up wires could fail. Applications where the boards are subject to vibrations such as robots or industrial environments. The EPT-200TMP-TS-U2 sensor and docking board provides a tight coupling of the board to the Arduino platform.

TMP102 Temperature Sensor

This docking board is based on the TMP102 Temperature Sensor chip from Texas Instruments. It can measure the ambient temperature between -25℃ to +85℃. The temperature is measured with an accuracy of ± 0.5℃ across the temperature range. The TMP102 is capable of reading temperatures to a resolution of 0.0625°C

The EPT-200TMP-TS-U2 is a combination of the TMP102 sensor, daughter board and docking board. The docking board provides a convenient method to connect the TMP102 to an array of

Arduino boards. It is compatible with both +3.3V and +5V Arduinos. There is a power indicator Green LED, and a user Green LED. It has stackable Headers that allow the board to plug into an Arduino and allow other boards to stack on top of it.

Hardware Features:

- Uses the I2C interface
- 12-bit, 0.0625°C resolution
- Typical temperature accuracy of ±0.5°C
- +3.3V sensor
- Compatible with +3.3V or +5V interface

Digital Temperature Output

The digital output from each temperature measurement is stored in the read-only temperature register. The temperature register of the TMP102 device is configured as a 12-bit, read-only register (configuration register EM bit = 0, see the Extended Mode (EM) section), or as a 13-bit, read-only register (configuration register EM bit = 1) that stores the output of the most recent conversion. Two bytes must be read to obtain data. Byte 1 is the most significant byte (MSB), followed by byte 2, the least significant byte (LSB). The first 12 bits (13 bits in extended mode) are used to indicate temperature. The least significant byte does not have to be read if that information is not needed. The data format for temperature is summarized below. One LSB equals 0.0625°C. Negative numbers are represented in binary twos-complement format. Following power-up or reset, the temperature register reads 0°C until the first conversion is complete. Bit D0 of byte 2 indicates normal mode (EM bit = 0) or extended mode (EM bit = 1) , and can be used to distinguish between the two temperature register data formats. The unused bits in the temperature register always read 0.

TEMPERATURE (°C)	DIGITAL OUTPUT (BINARY)	HEX
128	0111 1111 1111	7FF
127.9375	0111 1111 1111	7FF
100	0110 0100 0000	640
80	0101 0000 0000	500
75	0100 1011 0000	4B0
50	0011 0010 0000	320
25	0001 1001 0000	190
0.25	0000 0000 0100	004
0	0000 0000 0000	000
-0.25	1111 1111 1100	FFC
-25	1110 0111 0000	E70
-55	1100 1001 0000	C90

To convert positive temperatures to a digital data format:

1. Divide the temperature by the resolution

2. Convert the result to binary code with a 12-bit, left-justified format, and MSB = 0 to denote a positive sign.

Example: (50°C) / (0.0625°C / LSB) = 800 = 320h = 0011 0010 0000

To convert a positive digital data format to temperature:

1. Convert the 12-bit, left-justified binary temperature result, with the MSB = 0 to denote a positive sign, to a

decimal number.

2. Multiply the decimal number by the resolution to obtain the positive temperature.

Example: 0011 0010 0000 = 320h = 800 × (0.0625°C / LSB) = 50°C

To convert negative temperatures to a digital data format:

1. Divide the absolute value of the temperature by the resolution, and convert the result to binary code with a

12-bit, left-justified format.

2. Generate the twos complement of the result by complementing the binary number and adding one. Denote a

negative number with MSB = 1.

Example: (|–25°C|) / (0.0625°C / LSB) = 400 = 190h = 0001 1001 0000

Two's complement format: 1110 0110 1111 + 1 = 1110 0111 0000

To convert a negative digital data format to temperature:

1. Generate the twos compliment of the 12-bit, left-justified binary number of the temperature result (with MSB

= 1, denoting negative temperature result) by complementing the binary number and adding one. This

represents the binary number of the absolute value of the temperature.

2. Convert to decimal number and multiply by the resolution to get the absolute temperature, then multiply by

–1 for the negative sign.

Example: 1110 0111 0000 has twos compliment of 0001 1001 0000 = 0001 1000 1111 + 1

Convert to temperature: 0001 1001 0000 = 190h = 400; 400 × (0.0625°C / LSB) = 25°C = (|–25°C|);

(|–25°C|) × (–1) = –25°C

TMP102 Temperature Sensor and Arduino Programming

The Arduino IDE makes coding the TMP102 sensor quite easy. Everything needed to communicate with the sensor is in the “Wire” library. Just include the “Wire.h” file in your Arduino sketch. Connect EPT-200TMP-TS-U2 to the Arduino and plug the USB cable into a port on the PC.

A simple code example to display the temperature to the Serial Monitor:

```
#include "Wire.h"
#define TMP102_I2C_ADDRESS 72 /* This is the I2C address for our chip.
This value is correct if you tie the ADD0 pin to ground. See the datasheet for some other values. */
```

```


void setup() {
  Wire.begin(); // start the I2C library
  Serial.begin(115200); //Start serial communication at 115200 baud
}


void getTemp102(){
  byte firstbyte, secondbyte; //these are the bytes we read from the TMP102 temperature registers
  int val; /* an int is capable of storing two bytes, this is where we "chuck" the two bytes together. */
  float convertedtemp; /* We then need to multiply our two bytes by a scaling factor, mentioned in the datasheet. */
  float correctedtemp;
  // The sensor overreads (?)


  /* Reset the register pointer (by default it is ready to read temperatures)
You can alter it to a writeable register and alter some of the configuration -
the sensor is capable of alerting you if the temperature is above or below a specified threshold. */

  Wire.beginTransmission(TMP102_I2C_ADDRESS); //Say hi to the sensor.
  Wire.send(0x00);
  Wire.endTransmission();
  Wire.requestFrom(TMP102_I2C_ADDRESS, 2);
  Wire.endTransmission();


  firstbyte      = (Wire.receive());
/*read the TMP102 datasheet - here we read one byte from
 each of the temperature registers on the TMP102*/
  secondbyte     = (Wire.receive());
/*The first byte contains the most significant bits, and
 the second the less significant */
    val = ((firstbyte) << 4);
 /* MSB */
    val |= (secondbyte >> 4);
/* LSB is ORed into the second 4 bits of our byte.
Bitwise maths is a bit funky, but there's a good tutorial on the playground*/
    convertedtemp = val*0.0625;
    correctedtemp = convertedtemp - 5;
    /* See the above note on overreading */


  Serial.print("firstbyte is ");
  Serial.print("\t");
  Serial.println(firstbyte, BIN);
  Serial.print("secondbyte is ");
```

```
  Serial.print("\t");
  Serial.println(secondbyte, BIN);
  Serial.print("Concatenated byte is ");
  Serial.print("\t");
  Serial.println(val, BIN);
  Serial.print("Converted temp is ");
  Serial.print("\t");
  Serial.println(val*0.0625);
  Serial.print("Corrected temp is ");
  Serial.print("\t");
  Serial.println(correctedtemp);
  Serial.println();
}

void loop() {
  getTemp102();
  delay(5000); //wait 5 seconds before printing our next set of readings.
}
```