

Temperature Sensor Readout

Proposed Solution

Renato da Veiga Torres
renatoveigatorres@gmail.com

September 19, 2023

1 REQ01 - Analyze measuring circuit

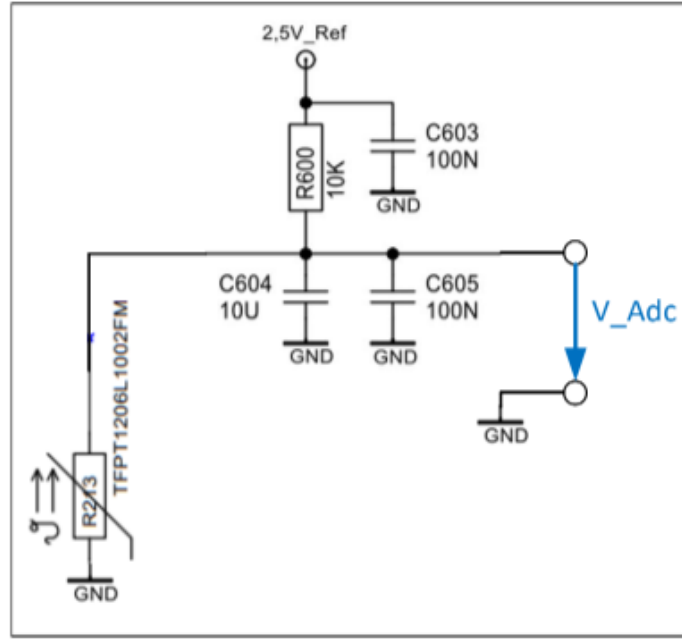


Figure 1: Temperature measurment circuit

For simplification, ADC input impedance, in this circuit (Figure 1), is considered high enough that is possible to neglect current leaking over ADC pin. In this way, the total current i on both resistors and V_{Adc} can be written as in (1):

$$i = \frac{V_{Ref}}{R_{600} + R_{213}^{RTC}} \quad (1)$$

As voltage divider, by construction, V_{Adc} can be calculated as described in (2):

$$V_{Adc} = \left[\frac{R_{213}^{RTC}}{R_{600} + R_{213}^{RTC}} \right] \cdot V_{Ref} \quad (2)$$

However, the purpose of the circuit is to 'read' V_{Adc} value and calculate R_{213}^{RTC} . As R_{213}^{RTC} is an RTC, its resistance/temperature are unequivocally correlated. Rewriting (2), R_{213}^{RTC} value and can be calculated as function of V_{Adc} as described in (3):

$$R_{213}^{RTC} = \left[\frac{V_{Adc}}{V_{Ref} - V_{Adc}} \right] \cdot R_{600} \quad (3)$$

As R_{600} is designed as same TFPT1206L1002FM (in 25° Celsius) - 10 kOhm, the value (R_{213}^{RTC}/R_{600}) is same as the average ratio (R_{Temp}/R_{25}) from datasheet ([tfpt-223737.pdf](#) - page 3):

$$ratio = \frac{R_{Temp}}{R_{25}} = \frac{R_{213}^{RTC}}{R_{600}} = \frac{V_{Adc}}{V_{Ref} - V_{Adc}} \quad (4)$$

Obtaining this ratio (R_{Temp}/R_{25}) - using (4) is possible to locate (as lookup table) the current temperature of the resistor in the datasheet ([tfpt-223737.pdf](#) - page 3).

2 REQ02 - Implement software that calculates temperature of NTC

```

1  #include "platform_types.h"
2  #include "RTC_TFPT1206L1002FM.h"
3
4  /**
5   * @brief Binary search to find the temperature based on datasheet values.
6   *
7   * This function performs a binary search on the TFPT_Vishay_Values array to find the
8   * temperature corresponding to a target resistance ratio.
9   *
10  * @param targetRatio The target resistance ratio for which to find the temperature.
11  * @return The temperature in degrees Celsius if a match or approximate match is
12  *         found,
13  *         or -273.0f (absolute zero) if the ratio is out of range.
14  */
15  static float getTemp_Datasheet(float targetRatio) {
16      int low = 0;
17      int high = sizeof(TFPT_Vishay_Values) / sizeof(TFPT_Vishay_Values[0]) - 1;
18      int resultIndex = -1;
19
20      while (low <= high) {
21          int mid = low + (high - low) / 2;
22
23          if (TFPT_Vishay_Values[mid].fRatio_R25 == targetRatio) {
24              resultIndex = mid;
25              break;
26          }
27
28          if (TFPT_Vishay_Values[mid].fRatio_R25 < targetRatio) {
29              low = mid + 1;
30          } else {
31              resultIndex = mid;
32              high = mid - 1;
33          }
34      }
35
36      if (resultIndex != -1) {
37          // Found the exact match or the next lower fRatio
38          if (resultIndex > 0) {
39              // Use linearization between 2 datasheet points to optimize accuracy
40              // fTx = fT1 + ((fRx - fR1)/(fR2 - fR1))*(T2 - T1);
41              // (T2 - T1) is always 1 oC
42              float fT1 = TFPT_Vishay_Values[resultIndex - 1].ftemperature;
43              float fT2 = TFPT_Vishay_Values[resultIndex].ftemperature;
44              float fR1 = TFPT_Vishay_Values[resultIndex - 1].fRatio_R25;
45              float fR2 = TFPT_Vishay_Values[resultIndex].fRatio_R25;
46              float fTx = fT1 + ((targetRatio - fR1) / (fR2 - fR1));
47              return fTx;
48          } else {
49              return TFPT_Vishay_Values[resultIndex].ftemperature;
50          }
51      } else {
52          // Return 'absolute zero' in case of ratio out of range
53          return -273.0f;
54      }
55  }

```

```
55
56  /**
57  * @brief Calculate the resistance ratio from an ADC value.
58  *
59  * This function calculates the resistance ratio based on a 12-bit ADC value and
60  * the characteristics of the TFPT1206L1002FM sensor.
61  *
62  * @param uiADCVal The ADC value obtained from the sensor.
63  * @return The resistance ratio.
64  */
65  static float getRatioFromADCValue(uint16_t uiADCVal) {
66      // 12-bit ADC => 2^12 = 4096
67      // Range:
68      // 0V - 0
69      // 2.5V - 4095
70
71      // R213 = [(uiADCVal) / (VREF - uiADCVal)] * R600
72      // However, R600 is 10K and R213 (TFPT1206L1002FM) is 10K on 25 oC.
73      // R213/R600 = datasheet_ratio itself
74
75      uint16_t VREF = 4095;
76      float fratio = (uiADCVal) / (VREF - uiADCVal);
77
78      return fratio;
79  }
80
81  /**
82  * @brief Get the temperature in degrees Celsius from an ADC value.
83  *
84  * This function calculates the temperature in degrees Celsius based on the ADC
85  * value obtained from the sensor and the datasheet values.
86  *
87  * @param uiADCVal The ADC value obtained from the sensor.
88  * @return The temperature in degrees Celsius if within the valid range,
89  *         or -273.0f (absolute zero) if out of range.
90  */
91  float getTemperature(uint16_t uiADCVal) {
92      return getTemp_Datasheet(getRatioFromADCValue(uiADCVal));
93  }
```

Listing 1: C Code for Temperature Sensor

3 REQ03 - Implement test-bench for software