*Subsystem Design*

# *Thermistor Temperature Sensing*

## Design description

This subsystem uses a resistor in series with a positive temperature coefficient (PTC) thermistor (TMP61) to form a voltage divider, which has the effect of producing an output voltage that is linear over temperature. This external circuit is read by setting up the MSPM0 internal op-amp in a buffer configuration and sampling with the ADC. If an increase of temperature is measured, an RGB LED turns red; if temperature decreases, the LED turns blue; and if no significant change in temperature, the LED remains green. This document does not go into details of calculating a temperature value from the ADC readings as such calculations are dependent on the thermistor chosen. Download the code example here.

Figure 1-1 shows the functional diagram of this subsystem.

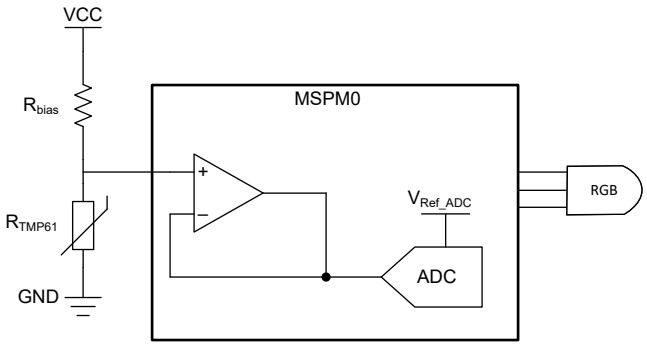


**Figure 1-1. Subsystem Functional Block Diagram**

## Required peripherals

This application requires an integrated OPA, ADC, Timer, and I/O pins.

**Table 1-1.**

| Sub-block functionality | Peripheral Used | Notes |
|---|---|---|
| Buffer amplifier | (1x) OPA | Called Thermistor_OPA_INST in code |
| Analog signal capture | (1x) ADC12 | Called ADC_INST in code |
| Timer for ADC sampling | (1x) TIMERx | Called Thermistor_TIMER_ADC in code |
| RGB LED Control | (3x) I/O Pins | Called RGB_RED_PIN, RGB_BLUE_PIN, and RGB_GREEN_PIN in code |

### Compatible devices

Based on the requirements in Table 1-1, this example is compatible with the devices in Table 1-2. The corresponding EVM may be used for prototyping.

**Table 1-2.**

| Compatible Devices | EVM |
|---|---|
| MSPM0L13xx | LP-MSPM0L1306 |
| MSPM0G35xx, MSPM0G15xx | LP-MSPM0G3507 |

### Design Steps

1. Determine $R_{bias}$. For the TMP61 thermistor used in this design, it is recommended for $R_{bias}$ to be 10 kΩ. Other configurations are available. See the TMP61 data sheet for details.
   a. Other thermistors can have different $R_{bias}$ recommendations or equations available to you for calculating $R_{bias}$. See the documentation for your chosen thermistor for details.
2. Setup OPA in SysConfig for buffer configuration with external input.
3. Setup ADC in SysConfig sample OPA output with chosen ADCMEMx.
4. Set ADC sample time in SysConfig to a minimum of $t_{Sample\_PGA}$ as given in the device data sheet.
5. Determine the temperature algorithm to use to convert ADC readings to temperature readings. This example uses raw ADC readings to calculate changes in temperature.

### Design considerations

1. Temperature calculation: Different thermistors will have various equations or lookup tables available in order to calculate temperature from the ADC readings and external circuit. Check your thermistor collateral for these resources that can be integrated into this design.
   a. Lookup tables take less compute time, but may not be valid for every situation and can possibly take up a large portion of memory.
   b. Equations take more compute time, but are more flexible to external variables. The complexity of the equations will depend on the accuracy or temperature range requirements.
2. OPA supply will be VCC of the MSPM0.
3. OPA GBW setting: A lower GBW setting for the OPA consumes less current, but responds slower; conversely, a higher GBW consumes more current, but has a larger slew rate and faster enable and settling times. See the device-specific data sheet for specification differences between the modes.
4. ADC Reference selection: MSPM0 devices can provide a reference voltage to the ADC from an internal reference generator (VREF), external source, or MCU VCC. Check the MSPM0 device data sheet for options available for the chosen device. For the configuration of this design, it is recommended to have the ADC reference to be equal to the bias voltage (VCC) of the external thermistor circuit.
5. ADC sampling: This example periodically samples the external circuit using a timer trigger. To adjust how often the circuit is sampled, adjust the timer parameters.
6. ADC results: The code example only stores the most current result captured in the global variable *gThermistorADCResult* . Full applications may want to store several readings in an array before performing actions on the data.
7. Race conditions on gCheckThermistor: This application clears gCheckThermistor as soon as possible. If the application waits too long to clear gCheckThermistor, the application can inadvertently miss new data.

### Software flowchart

Figure 1-2 shows the code flow diagram for this example and explains how the ADC samples the OPA output and the decision tree for LED illumination.

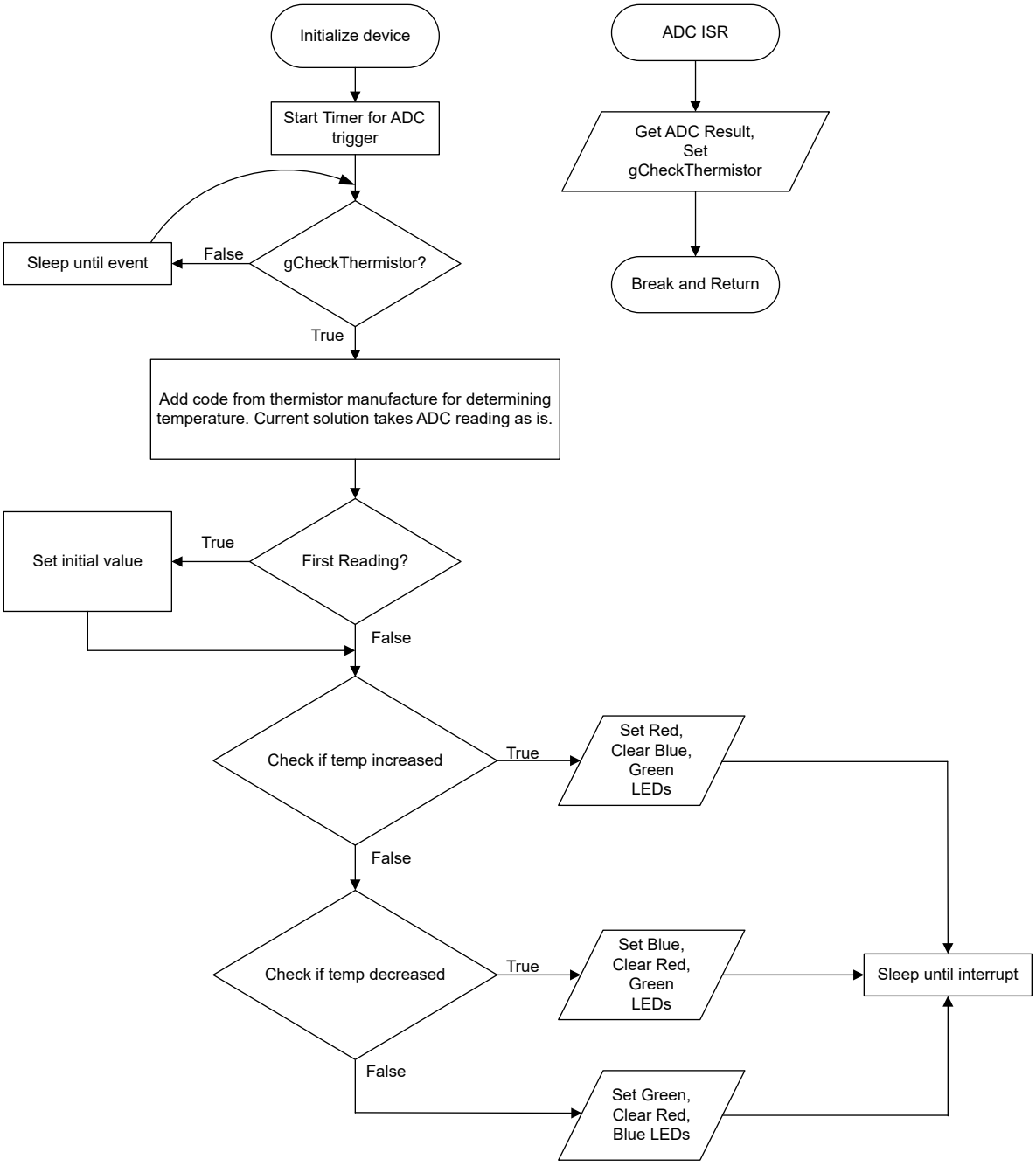


**Figure 1-2. Application Software Flowchart**

### Device configuration

This application makes use of TI System Configuration Tool (SysConfig) graphical interface to generate the configuration code of the device peripherals. Using a graphical interface to configure the device peripherals streamlines the application prototyping process.

The code for what is described in Figure 1-2 can be found in the beginning of *main()* in the *Thermistor_Example.c* file.

### Application code

This application does not compute temperature directly, but looks for a change in temperature. The following code snippet includes a value CHANGEFACTOR which is used to determine a minimal amount of ADC value change before recognizing a temperature change.

```
#include"ti_msp_dl_config.h"
#include<math.h>
#define CHANGEFACTOR 10
volatileuint16_tgThermistorADCResult = 0;
volatileboolgCheckThermistor = false;
```

The following code snippet shows where to add the temperature calculation method for the thermistor to compute actual temperature values. The current code takes an initial reading (gInitial_reading) at startup and compares the current reading (gCelcius_reading) with the CHANGEFACTOR adjustment to see if temperature has increased, decreased, or not changed enough. The RGB LED is then turned red (increase), blue (decrease), or green (no change) respective to the comparison result.

```
  while (1) {
        while (gCheckThermistor == false) {
                __WFE();
            }
        //Insert Thermistor Algorithm
            gCelcius_reading = gThermistorADCResult;
            if (first_reading) {
                gInitial_reading = gCelcius_reading;
                first_reading   = false;
            }
            /*
             * Change in LEDs is based on current sample compared to previous sample
             *
             * If the new sample is warmer than CHANGEFACTOR from initial temp, turn LED red
             * If the new sample is colder than CHANGEFACTOR  from initial temp, turn LED blue
             * Else, keep LED green
             * Variable gAlivecheck is utilized for debug window to confirm code is executing.
             * It is not needed in final applications.
             *
             */
            gAlivecheck++;
            if(gAlivecheck >= 0xFFF0){gAlivecheck =0;}
            if (gCelcius_reading - CHANGEFACTOR > gInitial_reading) {
                DL_GPIO_clearPins(
                    RGB_PORT, (RGB_GREEN_PIN | RGB_BLUE_PIN));
                DL_GPIO_setPins(RGB_PORT, RGB_RED_PIN);
            } else if (gCelcius_reading < gInitial_reading - CHANGEFACTOR) {
                DL_GPIO_clearPins(
                    RGB_PORT, (RGB_RED_PIN | RGB_BLUE_PIN));
                DL_GPIO_setPins(RGB_PORT, RGB_BLUE_PIN);
            } else {
                DL_GPIO_clearPins(
                    RGB_PORT, (RGB_RED_PIN | RGB_BLUE_PIN));
                DL_GPIO_setPins(RGB_PORT, RGB_GREEN_PIN);
            }
            gCheckThermistor = false;
        __WFI();
    }
```

### Additional resources

1. Download the MSPM0 SDK
2. Learn more about SysConfig
3. MSPM0L LaunchPad
4. MSPM0G LaunchPad
5. MSPM0 Timer academy
6. MSPM0 ADC academy
7. MSPM0 OPA academy

## Revision History

| DATE | REVISION | NOTES |
|---|---|---|
| February 2023 | * | Initial Release |

# IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265