Calculating and Displaying Temperature

by

Loyda Yusufova

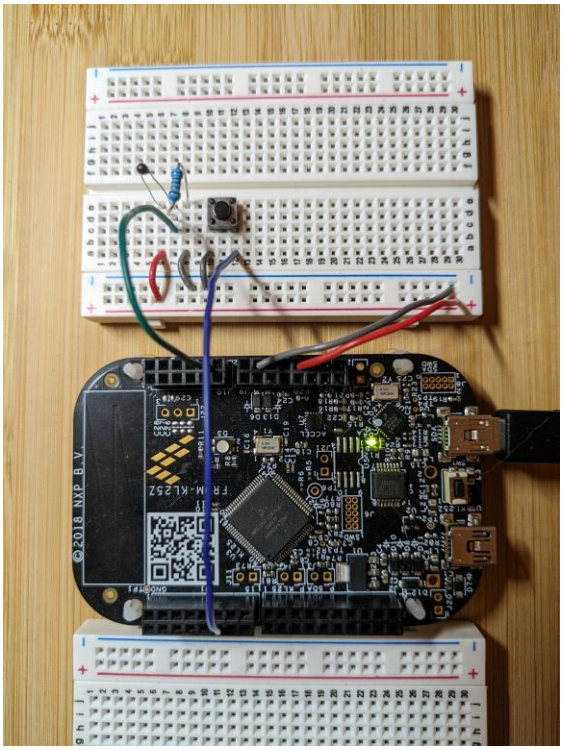


Figure 1: Image provided by author.

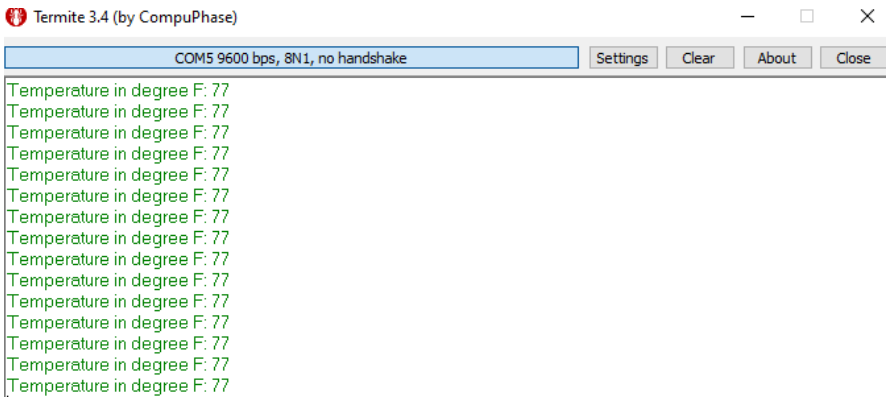


Figure 2: Screenshot of temperature readings from the Termite terminal window.

Table of Contents:

1. Introduction

This project uses a KL25Z microcontroller to show how to calculate ambient temperature using an NTC temperature sensor and displaying it through a serial monitor. Any communication protocol can be used to send the temperature information to a device that can display it through a monitor. In this case, the UART is configured to display the temperature.

The Periodic Interrupt Timer (PIT) module is configured to trigger an ADC conversion every 10 seconds. The ADC then proceeds to do the conversion. After the ADC has completed the conversion, the software calculates the temperature from the ADC conversion. Finally, the value is converted to ASCII characters and is sent to the UART to be displayed on a terminal such as Putty or Termite.

When the program starts running from system reset, nothing happens, the system is running in an idle state. A switch connected to a pin is configured to start or stop the timer that triggers the system. The switch must be pressed to start the PIT counter and thus start the whole system. At the same time, the button must be pressed to stop the timer from triggering the system. After the timer is stopped, the system remains idle.

In the upcoming sections, the hardware and software components that allow the system to calculate and display temperature will be discussed in detail.

2. Equipment

1. KL25Z Microcontroller

Figure 3: KL25Z Microcontroller (Image obtained from amazon.com).

2. NTC thermistor

Figure 4: 100K-Ohm NTC Thermistor (image obtained from digitkey.com).

3. Tactile Switch

Figure 5: Tactile Switch (Image obtained from amazon.com).

4. 100K Resistor

Figure 6: 100K-Ohm resistor (Image obtained from amazon.com).

5. Wires

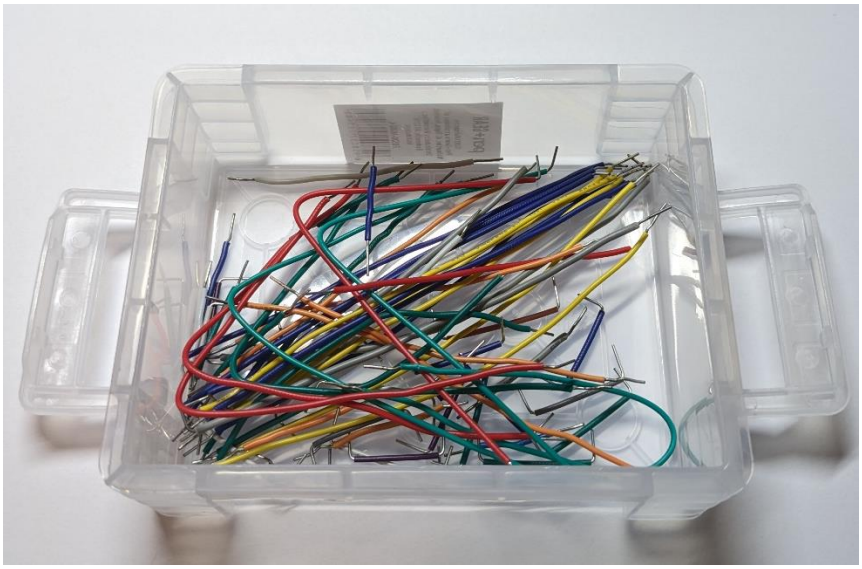


Figure 7: Wires (Image provided by author).

6. Breadboard

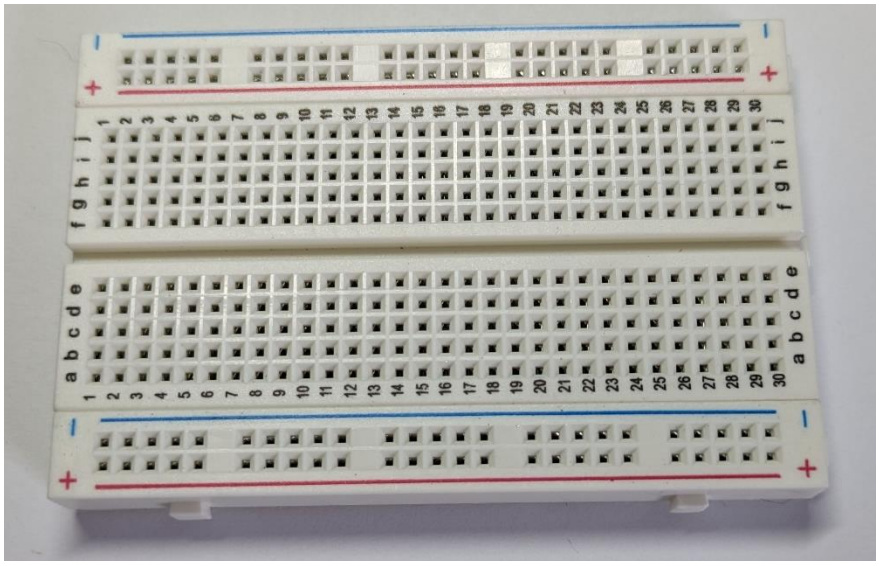


Figure 8: Breadboard (Image provided by author).

7. **Micro USB Cable**

Figure 9: Micro USB Cable (Image obtained from mbientlab.com).

3. Hardware

a. KL25Z Microcontroller

At the heart of this project is the KL25Z Microcontroller from NXP. This microcontroller has a 32-bit MCU, and an ARM Cortex-M0+ core processor. This microcontroller has many peripherals. The peripherals used in this project are the PORT, the GPIO, the Periodic Interrupt timer (PIT), the ADC, and the UART. All the mentioned peripherals and the external NTC thermistor work together to calculate the ambient temperature and display it on a PC terminal.

b. PORT Peripheral

The port modules used in this project are PORTA and PORTB. The specific pins configured for each module are discussed below:

I. PORTA pins 1 and 2 (PTA1 and PTA2):

These two pins are connected to the Microcontroller's Serial and Debug Adapter (OpenSDA). The Open SDA has many capabilities and functionalities. One of its functionalities is to provide a USB Communications Device Class (CDC) which serves as a bridge for serial communications between the USB host and this serial interface on the KL25Z.

So, in order to display messages on a PC terminal, PTA1 and PTA2 must be configured. PTA1 is connected to the UART0 peripheral's Receiver and PTA2 to the transmitter. So, PTA1 and PTA2 are configured as such.

II. PORTA pin 5 and 2 (PTA5):

This pin is connected to a press button. This button controls the PIT timer. When the button is pressed for the first time, the PIT timer starts and triggers the ADC converter. When the button is pressed a second time, it disables the PIT timer, and the ADC stops making temperature conversions. So, basically this pin is used to start and end the whole system.

Therefore, this pin is configured as a GPIO, input, interrupt pin. It triggers an interrupt on a falling edge since it is connected to a pull-up resistor.

III. PORTB pin 1 (PTB1):

Internally, this pin is connected to channel 9 of the ADC0 peripheral. So, the NTC thermistor is connected to it externally to serve as input to the ADC channel 9. Therefore, this pin is configured as an analog input.

IV. PORTB pin 18 (PTB18):

This pin is connected to the microcontroller's on-board red LED. The red LED is toggled every time a UART message has been sent to the terminal. So, this pin is configured as GPIO output to drive the red LED.

The PORT configuration function and all other PORT related functions are defined in the PORT.c file. PORT macros and function declarations are in the PORT.h file.

c. GPIO Peripheral

The GPIO is used to drive the on-board red LED connected to PTB18. So, this pin is configured as a GPIO output.

It is also used to trigger interrupts via the push button connected to PTA5. So, PTA5 is configured as a GPIO input that triggers interrupts on a falling edge.

The GPIO configuration function and all other GPIO related functions are defined in the GPIO.c file. PORT macros and function declarations are in the GPIO.h file.

d. PIT Timer

The Periodic Interrupt Timmer channel 0 (PIT0) of the KL25Z microcontroller is configured to trigger an interrupt every 10 seconds from the moment it is enabled. The interrupt service routine for this timer is configured to enable the ADC converter. So, the function of this timer is to trigger ADC conversions every 10 seconds.

The PIT configuration function and all other PIT related functions are defined in the PIT.c file. PIT macros and function declarations are in the PIT.h file.

e. ADC

The ADC0 is used to read an analog input from its channel number 9 connected to PORTB pin 1 (PTB1). This analog input is an NTC thermistor that is used to sense the ambient temperature.

The ADC is configured as follows:

- single-ended mode
- low power mode is used.
- the conversion rate is set to 1.5 MHz.
- Hardware trigger. The PIT channel 0 timer triggers it every 10 seconds.
- Its voltage reference is set to 3.3V
- Hardware average is configured to average 16 samples.
- The interrupt is enabled. So, after a conversion, the ADC interrupt is triggered.

In addition, the ADC Interrupt Service routine is configured to read the conversion result from the ADC channel 9 (connected to the NTC thermistor) and set an event flag to indicate that the ADC has finished conversion.

The ADC configuration function and all other ADC related functions are defined in the ADC.c file. ADC macros and function declarations are in the ADC.h file.

f. UART

The UART peripheral 0 is used to send the temperature value, that has been read and converted by the ADC, to the PC.

A queue buffer is created for data transmission. The software writes to this buffer and the UART0 takes characters from this buffer and transmits them. When the buffer is empty, the UART0 stops transmitting data.

The UART is configured as follows:

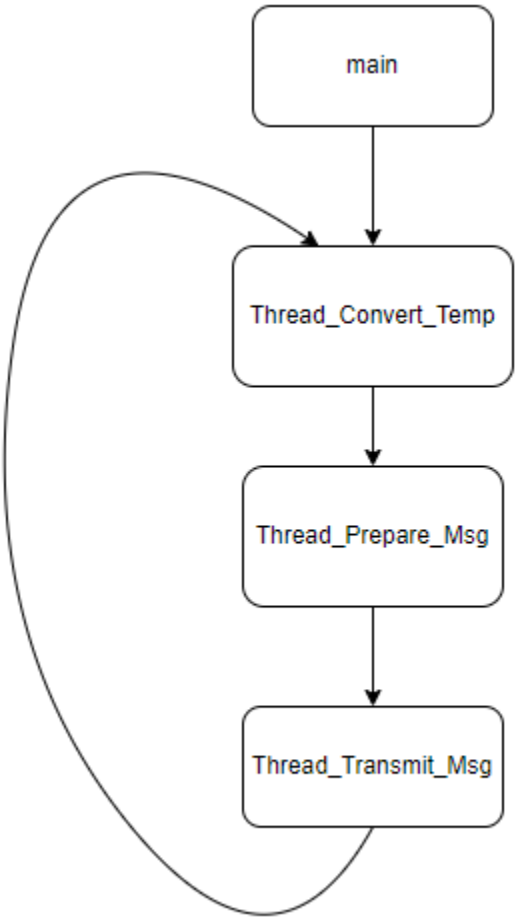
- Baud rate = 9600.
- One stop bit.
- No parity.
- 8 data bits.
- Interrupts for errors are enabled: overrun, noise, framing, and parity.
- Transmit interrupt is enabled to trigger an interrupt when the UART transmitter buffer is empty and ready to transmit another character.

In addition, the UART0 interrupt service routine is configured to clear the data register if the interrupt was caused by an error. If the interrupt was caused by the transmitter, then

data from the transmitter queue is dequeued and written into the data register of the UART0 to be displayed on the terminal. Once the buffer is empty, the UART0 transmitter interrupts are disabled.

The UART0 configuration function and all other UART0 related functions are defined in the UART0.c file. UART0 macros and function declarations are in the UART0.h file.

4. Software



The RTX5 RTOS is used to support threads and thread synchronization and communication. Besides the RTOS, there are four pieces of software that are very important in this project. These four pieces of software are discussed in detail below:

e. Main function

The main function's purpose is to initialize the system and all the peripherals that are used in the project. It also initializes the kernel, creates the threads and thread synchronization objects, and starts the kernel.

f. Thread_Convert_Temp

The function of this thread is to convert the ADC result into a temperature value in degree Fahrenheit. The thread blocks until the ADC has finished the conversion. At the end of the temperature calculation, the thread releases a semaphore to let the Thread_Prepare_Msg thread proceed with the message preparation.

g. Thread_Prepare_Msg

This thread prepares a message to be sent to the serial terminal through UART. But first, it blocks on a semaphore that must be released by Thread_Convert_Temp

The first part of the message is “"Temperature in degree F: ". This first part of the message is enqueued into the transmit (TX) queue.

For the second part of the message, the thread proceeds to convert the calculated temperature digits to ASCII characters. Once the conversion is completed, the thread enqueues the ASCII characters representing the temperature into the TX queue.

Finally, the thread releases a semaphore to indicate that a new message is ready to be transmitted. This semaphore allows the Thread_Transmit_Msg to transmit the message through UART.

h. Thread_Transmit_Msg

This thread transmits the message containing the temperature reading through UART.

First, the thread blocks until a new message is ready, which is indicated by the semaphore that is released by Thread_Prepare_Msg. Once the thread acquires the semaphore, it proceeds by enabling the UART transmitter and the transmitter interrupt. The thread then waits for the UART transmitter interrupt to clear the interrupt flag which indicates that the message has been sent. Finally, the thread proceeds to disabling the UART transmitter and enabling the PIT timer to trigger a new ADC conversion.

The whole process is repeated indefinitely until the user presses the button to disable the PIT timer and thus stops the process.

5. System diagram