Lab 2. Data Processing Using C

(Fall 2019)

Preparation

Read Sections 1.12, 2.1, 2.2, 2.3, 2.5, and 2.7 of the book.
Find the starter project Lab2_EE319K (EE319K installer, or Canvas directions).
This lab will be performed individually.

Purpose

The general purpose of this laboratory is to familiarize you with the software development in the C programming language. We choose a problem that exercises problem-solving skills that allow you to devise a solution (algorithm). However, you will code the solution in C instead of assembly.

Requirements

The objective of this lab is to write three support routines in **Lab2.c** that are called by a controller in **main.c** to perform data analysis. The controller is collecting temperature sensor data periodically. Your task is to write three data analysis routines so the controller can call them as part of its control algorithm. The data exists as an array within the **main.c** program (which you are not allowed to edit), and a pointer to the16-bit signed array is passed to your functions as an input parameter (e.g., int16_t Readings[]). The size of the array is passed as a value (e.g., N). You may assume the size of the array is less than 50. When const is used like this in the parameter declaration, it means the parameter is read-only (your routine can not change its values).

The specific routines are:

1. int16_t Find_Mean(const int16_t Readings[],const int32_t N) - Computes the mean (average) of the temperature sensor data collected in the Readings array. The return result is an integer, therefore any fractional component of the computed mean must be truncated. For this function you may assume there is at least one point.
2. int16_t FtoC(const int16_t TinF) - Convert the temperature in Fahrenheit to the temperature in Centigrade. Assume the input varies from -459 F to 1000 F. Therefore, the output will range from -273 to 538 C. The correct solution uses integer math. You may either truncate or round your result. For example, 307 F should convert to 153 C (152.778). However, in integer math, ((307-32)*5)/9 will only be 152.

3. int IsMonotonic(const int16_t Readings[],const int32_t N) - Checks whether the recorded readings are a non-decreasing monotonic series. The controller performs some remedial operation and the desired effect of the operation is to lower the temperature of the sensed system. If all the values are equal, the result should be true.
4.

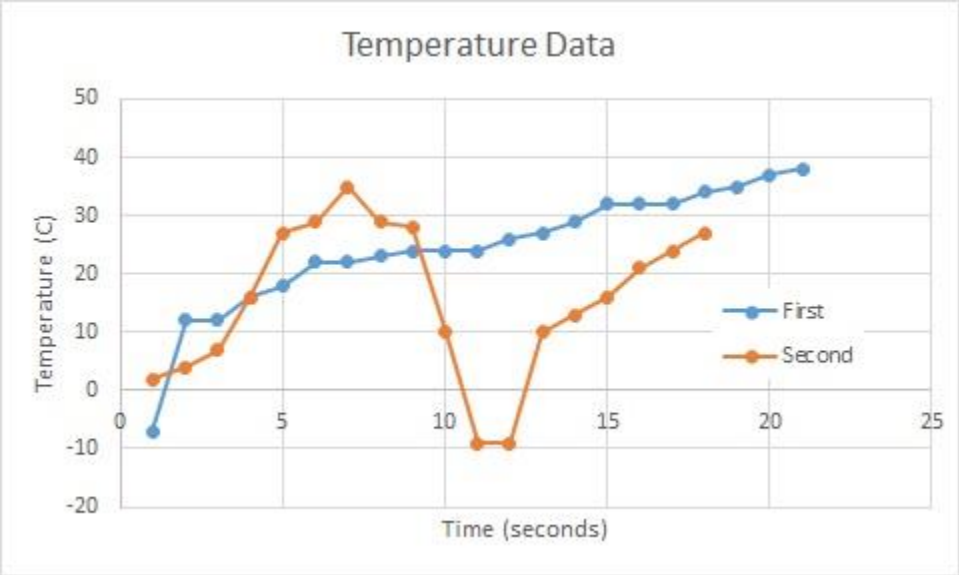
   Example1: If the readings are as follows:
   int16_t First[21]={
   -7,12,12,16,18,22,22,23,24,24,24,26,27,29,32,32,32,34,35,37,38};

then they are indeed a non-decreasing monotonic series, and so the routine would return a True(1).

Example2: If the readings are as follows:

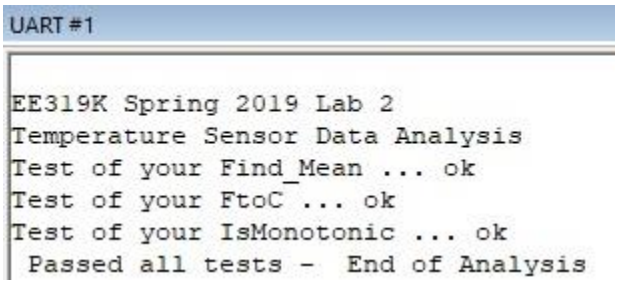
uint16_t Second[18]={2,4,7,16,27,29,35,29,28,10,-9,-9,10,13,16,21,24,27};

then they are *not* a a non-decreasing monotonic series as shown by the plot **Second** data, and so the routine would return a False(0):



Procedure

The starter project provided (Lab2_EE319K) has one assembly file Startup.s and multiple C files, main.c, PLL.c, UART.c, and Lab2.c. All your tasks are performed by writing code for the three subroutines (called functions in C) whose blank stubs are provided in Lab2.c. To test whether your implementations of these functions are correct, you can run the project in the simulator and you should get the following result on the UART window:



Demonstration

During the demonstration, you will be asked to run your program to verify proper operation. You should be able to single step your program and explain what your program is doing and why. You need to know how to set and clear breakpoints, watch variables like the Readings array and any local variables you declare in your subroutines.

Deliverables

Upload the following two files as part of your lab report.

1. Lab Report. A screenshot of your UART1 window.
2. Lab 2 C source code from your Lab2.c