Laboratory 8

(Due date: Nov. 21st)

Objectives

- Learn basic mechanisms for Real-Time Systems: handle signals (setup and detect).
- Configure and test Real-Time Clock.

Reference Material

- Refer to the board website or the Tutorial: Embedded Intel for User Manuals and Guides.
- Refer to the Tutorial: High-Performance Embedded Programming with the Intel® Atom™ platform → *Tutorial 8* for associated examples.

Activities

First Activity: Handling Signals (60/100)

- In this experiment, you are asked to implement an application (`.c`) that implements the following:
  - ✓ Prints the message "`i: Inside main function`" every second. `i = 1,2, …`
  - ✓ Every 2 seconds, an alarm (`SIGALRM` signal) goes off that interrupts the execution of the main function and prints the message "`Inside handler function for SIGALRM`".
  - ✓ The user should have the ability to snooze the alarm for 5 seconds. This is done via the `SIGINT` signal (***Ctrl-c***); here, the message "`Inside handler function for SIGINT`" is printed.
  - ✓ To exit the program, the user can use the `SIGQUIT` signal (***Ctrl-***\).

Suggestions

- You need to setup a ***handler function*** for both the `SIGALRM` and `SIGINT` signals.
- You can set up the 2-second alarm before an infinite loop. In order to continuously setup the 2-secodn alarm, you might want to setup (restart) the 2-second alarm every time the ***handler function*** for `SIGALRM` is executed.
- When the user issues the `SIGINT` signal, you can set up a 5-second alarm in the ***handler function*** for `SIGINT`.

- Take a screenshot of the software running on the Terminal. It should show: i) the messages being printed every second, ii) the alarm going off every 2 seconds, and iii) the user generating the `SIGINT` signal. Fig. 1 shows an execution example.

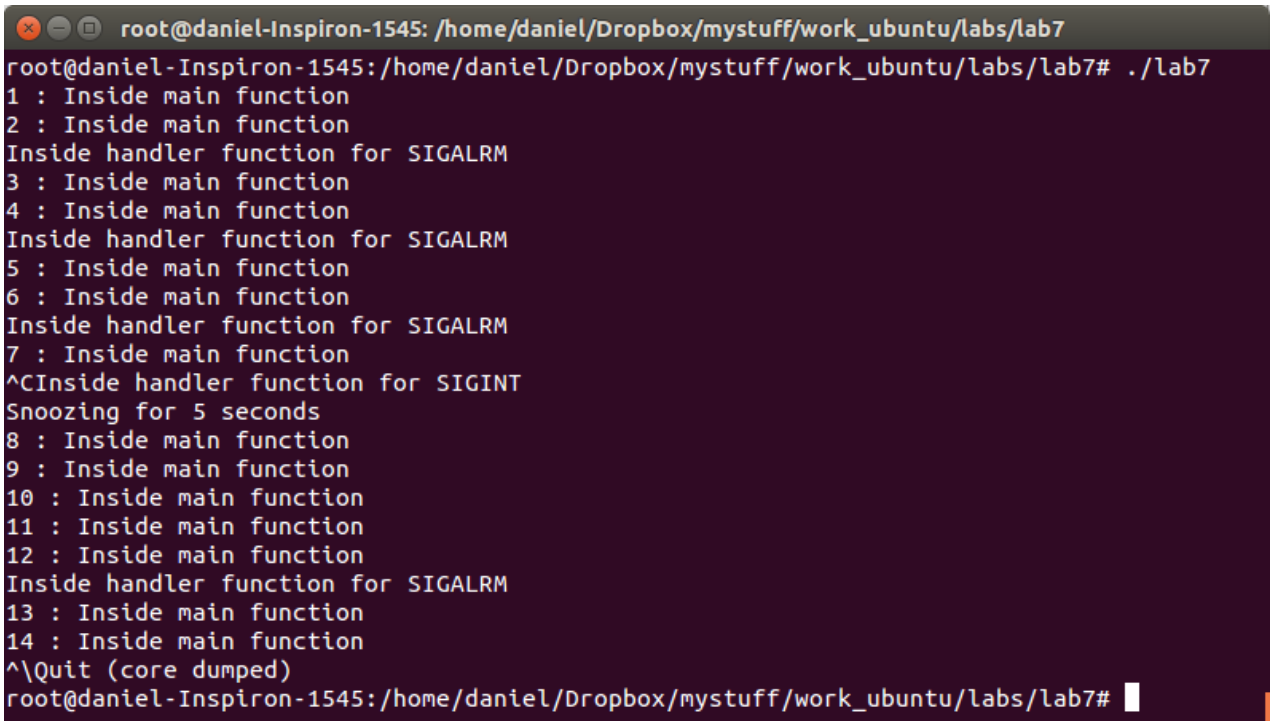

Figure 1. Sample execution for the application in the First Activity.

SECOND ACTIVITY: REAL-TIME CLOCK CONFIGURATION (40/100)

- In this experiment, you are asked to implement an application that:
  - ✓ Read data (current date/time) from RTC.
  - ✓ Configure and test the ***Alarm Interrupt***.
  - ✓ Configure and test ***Periodic Interrupts***.

- You need to use the RTC driver template (`rtctst.c`) available from the Tutorial: High-Performance Embedded Programming with the Intel® Atom™ platform → *Tutorial 8*. Refer to this tutorial for a detailed explanation of the code.

- You are asked to perform the following (these are minor modifications to `rtctst.c`):
  - ✓ Read the RTC time/date. Print it in format `mm-dd-yy, hours:minutes:seconds`.
  - ✓ Configure and test ***Alarm Interrupt***:
    - Set the Alarm Interrupt to **10** seconds in the future.
    - Read current alarm settings. Print the time the alarm is set to go off: `hours:minutes:seconds`.
    - Enable Alarm Interrupts.
    - Wait until Alarm Interrupt comes by executing a blocking `read()` on RTC.
    - Disable Alarm Interrupts.

  - ✓ Configure and test ***Periodic Interrupts***:
    - Read periodic IRQ rate (it will print the last one it has been used)
    - For a set of periodic interrupts (from 2 to 256 Hz, only powers of 2), do: set the frequency, enable period interrupts, detect 20 interrupt of a given frequency, and disable periodic interrupts.
      - Set frequency (`2, 4, 8, 16, 32, 64, 128, 256`)
      - Enable Periodic Interrupts
      - For a given frequency, wait for 20 periodic interrupts; use a blocking `read()` to detect each.
      - Disable Periodic Interrupts

- Note that you need be `root` to execute this code (use `sudo -i`).

- Take a screenshot of the software running on the Terminal. It should show (Fig. 2 shows an execution example)
  - ✓ The current RTC data/time,
  - ✓ The time the alarm is set to go off, and when the alarm is detected.
  - ✓ The current Periodic Interrupt rate, and the detection of 20 periodic interrupts for each frequency (2, 4, 6, 16, 32, 64, 128, 256).

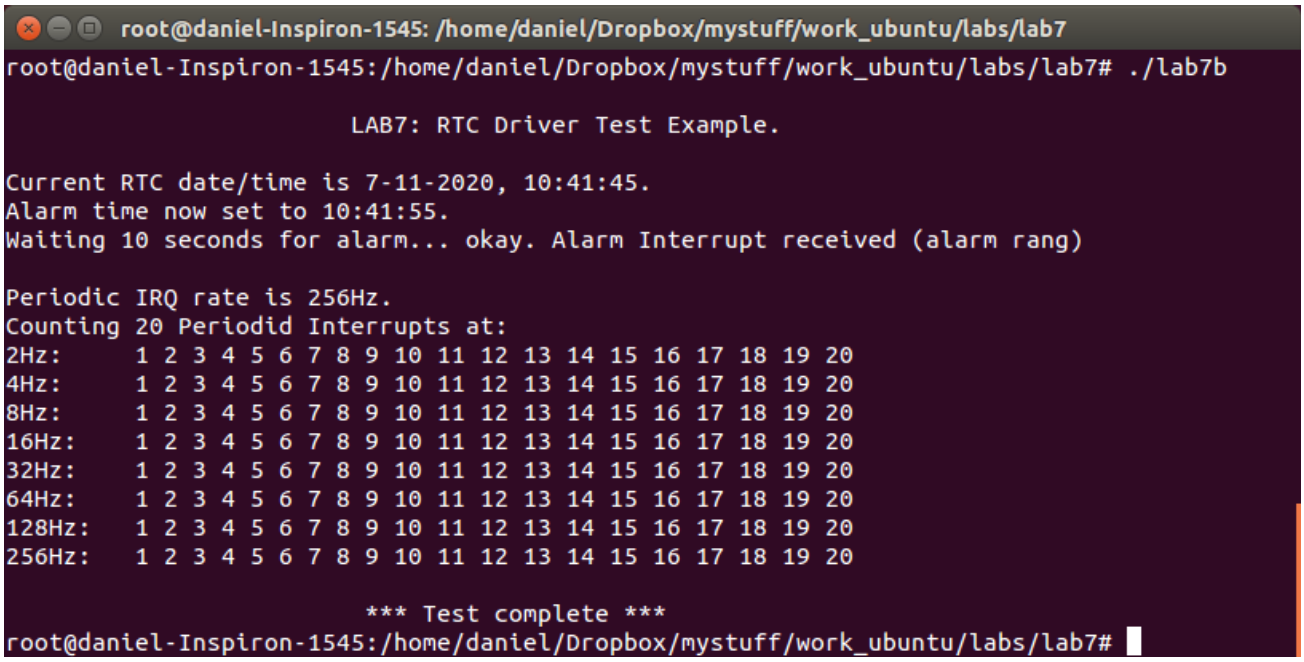

Figure 2. Sample execution for the application in the Second Activity.

SUBMISSION

- Demonstration: In this Lab 8, the requested screenshot of the software routine running in the Terminal suffices.
  - ✓ If you prefer, you can request a virtual session (Zoom) with the instructor and demo it.

- Submit to Moodle (an assignment will be created):
  - ✓ Two .zip files (one for the 1st Activity and one for the 2nd Activity).
    - 1st Activity: The .zip file must contain the source files (`.c, .h, Makefile`) and the requested screenshot.
    - 2nd Activity: The .zip file must contain the source files (`.c, .h, Makefile`) and the requested screenshot.

TA signature: _____ **Date:** _____