Dalhousie University
Department of Electrical and Computer Engineering

ECED 4402 – Real Time Systems

Assignment 2: Stopwatch

Objectives

In this assignment you are expected to implement a module that allows a user to interact with a simple monitor. The monitor is to allow the user to enter commands that set the time-of-day, query the time-of-day, and run a stopwatch. A cursor is to indicate the current screen position; the screen is to scroll when the cursor reaches the "bottom" of the screen (i.e., line 25).

To achieve this, you are expected to write:

- a clock driver for the Cortex (SYSTICK),
- a UART driver for the Stellaris (UART0),
- queue support software, and
- control software.

Overview

This assignment consists of four distinct parts: a clock interrupt handler, a UART interrupt handler, a queue, and control software.

Both the clock and the UART are devices that can be programmed to generate interrupts when a device state change occurs. In the case of the clock, it is each time SYSTICK reaches zero, whereas with UART0, it is when a character has been received or transmitted.

The queue software must allow the addition of messages to the queue using **enqueue()**, and the removal of messages from the queue using **dequeue()**. The queue is a *critical region* in that two objects should not be in the region (i.e., the queue) at the same time; if they are allowed to be, there is a potential for disastrous (or exciting—your choice) results occurring. The software must be written in such a way as to ensure that at most one object can be in the region at any one time. Characters received by the UART as well as one-tenth-of-a-second timing signals from SYSTICK are to be queued by their respective ISRs (using **enqueue()**); these are to be removed by the control software (using **dequeue()**). Characters are to be passed to the UART for transmission by the control software.

The control software is to take messages from the queue and interpret them. Input received by the UART is to be echoed back to the user (i.e., output to the UART from the control software via the queue) as well as being formed into strings for subsequent processing. Clock messages should cause the elapsed time to be updated. The elapsed time should handle hours, minutes, seconds, and tenths-of-a-second; the initial value is 00:00:00.0. Hours range from 00 to 23, minutes and seconds from 00 to 59, and tenths-of-a-second from 0 to 9.

Three input commands are to be supported (all others should produce the ever-useful diagnostic '?'):

- TIME - when the user enters the string "time", the current time-of-day should be displayed.

- SET hh:mm:ss - sets the current time of day; the string "set" is to be followed by a space then "hh" (digits 00 through 23), "mm" (digits 00 through 59), and "ss" (digits 00 through 59), and the punctuation symbols ":" in the appropriate positions.
- GO - the string "go" starts the stop watch (from 00:00:00.0).

The output from the stopwatch should appear on the line following the "GO" command; when any key is pressed, the stopwatch should stop and its value remain on the screen. The stopwatch value must not affect the time-of-day.

The stop watch software should run until the machine is reset or the user closes the laptop's terminal-emulator window.

Marking

This assignment will be marked as follows:

Design

The design description must describe the algorithms associated with each of the four major divisions of the software. It must also explain which part of the queue is the critical region and how it is protected.

Total points: 5.

Software

A fully commented, indented, magic-numberless, tidy piece of software that meets the requirements described above and follows the design description.

Total points: 7.

Testing

A set of at least four tests should be conducted. The tests should be in three parts: a description of the test, a description of what the test is to achieve, and a description of the test results (i.e., whether the software met the objectives of the test). Demonstrating the software is *not* a test.

Total points: 3.

Important Dates

Available: 28 September 2015
Due: 19 October 2015
Demonstration: 21 October 2015 (in the lab)

Late assignments will be penalized 0.25 points per day or fraction thereof.

Assignments must be successfully demonstrated before they will be marked.

Miscellaneous

It will be necessary to have a terminal emulator on your laptop or other mobile device for this assignment. One possible, recommended, emulator is **putty**, which can be downloaded from the URL specified on the course website.

The screen cursor can be controlled using the **VT-100 Terminal Control Escape Sequences**. These are strings of ASCII characters defined in the 1970s by DEC to allow their character-based terminals to support rudimentary graphics and forms. Given the popularity of the VT-100 family and the usefulness of the escape sequences, putty can be put into VT-100 mode to accept VT-100 escape sequences (see Configuration > Terminal > Keyboard and then select VT 100+). There are numerous websites devoted to the VT-100 escape sequences; for example, a brief list of some of the escape sequences that may be of use to the assignment can be found at the URL specified on the course website.

Although there are literally hundreds of escape sequences, limit your implementation to those that are of use to this assignment, such as cursor control. If you feel daring, you can try adding colour to the screen.

The USB cable connecting your laptop with the Stellaris carries both the UART signals and powers the Stellaris.

Do *not* discard this work once it is completed, since this makes up part of the real-time operating system to be implemented in subsequent assignments.

You must work alone on this assignment.

If you are having *any* difficulty with this assignment, *please* contact me as soon as possible.