Dalhousie University
Department of Electrical and Computer Engineering

ECED 4402 – Real Time Systems

Assignment 1: An interrupt-driven monitor

Objectives

In this assignment you are to implement an interrupt-driven monitor. The monitor is to allow a user to enter commands that set and query the time-of-day, set and query the date, and set the time for an alarm to trigger. A cursor is to indicate the next input character position; the screen is to scroll when the cursor reaches the "bottom" of the screen (i.e., line 25).

Overview

This assignment consists of four distinct components: an interrupt service routine for the Cortex timer (SYSTICK), a UART interrupt service routine for the Tiva (UART0), the monitor, a queue between the ISRs and the monitor, and command-support routines.[1]

Both the timer and the UART are devices that can be programmed to generate interrupts when they undergo a status change. In the case of the timer, it occurs each time SYSTICK reaches zero (every tenth-of-a-second), while with the UART, it is when a character has been received or transmitted by UART0.

The queue routines must the exchange of messages between the ISRs and the monitor. Two operations must be supported: the queuing of messages sent from an ISR to the monitor or vice-versa and the dequeuing of messages by either an ISR or the monitor. Characters received by the UART as well as the timing signals from SYSTICK are to be queued by their respective ISRs and removed by the monitor. Characters are to be passed to the UART ISR for transmission, again, using a queue structure.

In addition to supporting the flow of input messages from the ISRs to the monitor, the queuing routines must also allow character and string messages to be passed from the control and command-support routines to the UART for output.

The control routine is to take messages from the input queue and direct them to the appropriate command-support routine. Input received from the UART is to be echoed back to the user by the control software via a queue not the ISR itself, as well as being formed into strings for subsequent processing by one of the command-support routines. 'Tick' messages from SYSTICK should cause the elapsed time (in tenths of a second) to be updated.

All input commands, in either upper-case or lower-case, or both end with an end-of-line indication, such as '\r'. Three input commands are to be supported (all others should result in the diagnostic '?'):

TIME {*time-value*}: The TIME command allows the user to set the current time (24 hour clock) or query the current time. If *time-value* is omitted, the current time is to be displayed on the

[1] The supplied SYSTICK and UART interrupt handle code is "bare-bones" and will need to be adapted for this assignment.

next line in the format *HH:MM:SS.T*. If *time-value* is supplied, it must be in the format *HH:MM:SS*, where *HH* is hours (00 through 23), *MM* is minutes (00 through 59), *SS* is seconds (00 through 59), and *T* is tenths of a second (0 through 9). The time-value characters are ASCII characters separated by colons. For example:

```
> time 12:30:00.5
12:30:00.5
> TIME
12:30:59.3
> timex
?
> time 99:99:99
?
> TimE
12:32:09.4
>
```

Any of *HH*, *MM*, or *SS* can be omitted, giving a default value of zero. For example, 9:23 sets hours to 9, minutes to 23, and seconds to 0 (default). Similarly, ::30 sets hours and minutes to 0 and seconds to 30. The default (initial) time is 00:00:00.

DATE {*date-value*}: When issued, the DATE command either returns the current date or lets the user set the date. The *date-value* is in the format *DD-MMM-YYYY*, where *DD* is the day-of-the-month (1 through 28, 29, 30, or 31, depending on the month and year), *MMM* is the month (Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, or Dec), and *YY* is the year, any four-digit value (0000 through 9999). If the *date-value* is omitted, the current date is displayed. The default *date-value* is 1-Sep-2019. For example:

```
> date 1-jan-2019
1-Jan-2019
> DaTe
1-Jan-2019
> time 23:59:59.0
23:25:59.0
> time
00:01:28.7
> date
2-Jan-2019
> datum
?
> date 99-Zog-1900
?
>
```

It is necessary to handle leap-years and leap-centuries.

ALARM {*delta-value*}: The ALARM command allows the user to request an “alarm” to be signaled at a specific time in the future or clear an existing alarm. The *delta-value* is in the same format as used with the TIME command (see above). The alarm-time (i.e., the time the

alarm is to signal) is the sum of the *delta-value* and the current time. If *delta-value* is omitted, the alarm is cleared. For example, to signal an alarm five minutes into the future:

```
> time
14:06:56.4
> alarm 0:5:0.0
Alarm at 14:12:10.9
> time
14:10:47.0
>
* ALARM * 14:12:10.9 *
>
```

In the above example, the alarm value of 0:5:0.0 (five minutes into the future) is added to the current time, giving 14:12:10.9. The alarm routine indicates this after the time is set. After five minutes, the alarm is indicated.

An alarm can be cleared once it is set; for example, clearing an alarm set 10 minutes into the future:

```
> alarm 10:0.0
Alarm at 06:33:12.3
> alarm
Alarm cleared
>
```

In this assignment, as in space, no one can hear the alarm or your scream. However, outputting the ASCII BEL character (0x07) should produce an audible beep.

The commands can be entered in upper or lower case, or a combination of both.

The monitor is responsible for supplying the specified command-support routine with the arguments from the command line. The monitor is to build the command-line, echoing each character received (that is, the UART's ISR is not to echo the character). When end-of-line is detected, the monitor can determine which, if any, command-support routine is to be called.

Characters can be deleted using CTRL-H (ASCII BS or backspace, 0x08). The command-line buffer is 80 characters long. The screen must scroll when the cursor reaches the bottom of the screen.

A string is output one character at a time. If the UART output routine is supplied with a string, it can output the next character after the current character has been transmitted (i.e., a transmission completion interrupt is signaled). At the end-of-string, the output should stop. As usual, it is the boundary conditions that cause the problems: how is the first character to be sent and what is the UART output routine to do if, on completion of a transmission there is a pending string? Single characters can be output using the UART output routine as well.

The software must be downloaded and run on the Tiva.

Marking

This assignment will be marked as follows:

Design

The design description must describe the algorithms and data structure associated with each of the four a major components.

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.

System Testing

A set of at least six system tests should be conducted.[2] Each test is to consist of three parts: a description of the test, a description of what the test is to achieve, a description of the test results based on specific input scenarios, and an indication as to whether the software met the objectives of the test. Demonstrating the software is *not* a system test.

Total points: 3.

Important Dates

Available: 10 September 2019

Due: 3 October 2019 (in lab)

Demonstration (Acceptance testing): 3 October 2019 (during lab)

Late assignments will be penalized 1.5 points per day or fraction thereof to a maximum of 4.5 points. Assignments will not be accepted or marked if submitted two weeks after the due date.

Assignments must be successfully demonstrated before they will be accepted and marked.

This assignment is worth 8% of your overall assignment grade.

Miscellaneous

It will be necessary to have a terminal emulator on your laptop or other mobile device for this assignment. One possible 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; a brief list of

[2] See System Testing Fundamentals (http://softwaretestingfundamentals.com/system-testing/) for a description and definition of system testing and other stages in the test cycle.

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 to the Tiva both powers it and carries the UART signals.

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.

Assignment 2 will be released no later than 1 October 2019.

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