Assignment 2:
I/O, Timers, Interrupts

1DT106: Programming Embedded Systems
Uppsala University

November 4th, 2016

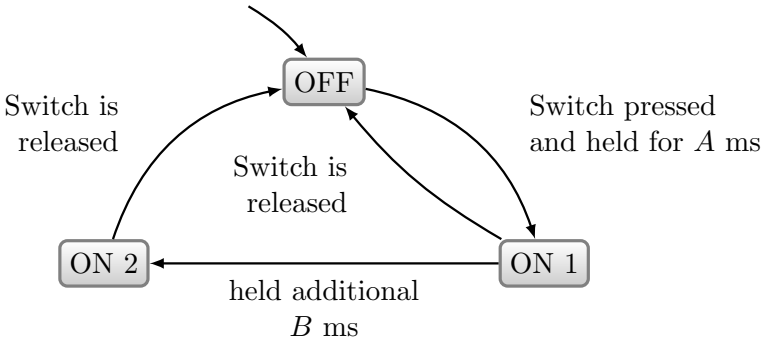
Assignments are to be solved by students *individually*. Several students handing in (literally) the same assignment solution will be considered as cheating, and reported accordingly.

Exercises

Exercise 1 Switch

In this exercise, you are supposed to implement software for a *multi-state* push-button switch. Imagine the "forward" button of a video player: pressing the button for a short time (say, less than 2s) will make the player forward slowly, while keeping the button pressed for a longer time makes the player change to a fast-forward mode.

The following state chart describes the behaviour of such a multi-state switch:



1. We assume that a push-button switch is connected to Pin 0 of GPIO C of an STM32F103 micro-controller. Implement a task function that

realises the multi-state switch behaviour as described above, for $A = 0.2$s and $B = 1.8$s. Start from the empty μVision project that you find on the student portal.
The effect of "ON 1" shall be to put a 1 on Pin 1 of GPIO C, the effect of "ON 2" to put a 1 on Pin 2.

2. Write a μVision debug function that exercises your implementation. The debug function is supposed to implement a test case that takes your system through all four transitions of the switch state chart (ignore the initial transition to state "OFF"). The debug function also has to observe the values on Pins 1 and 2 of GPIO C and to assess whether the behaviour is correct.
More information on debug functions are available at `http://www.keil.com/support/man/docs/uv4/uv4_debug_functions.htm`.

Exercise 2 Timers, Interrupts

Recall the program that you wrote in assignment 1, exercise 1, which was supposed to send seconds and tenths of seconds since controller start-up to the USART 1 device. In this exercise, you are supposed to implement a similar program generating the following sequence of outputs (again, at the correct points in time):

```
0.0000s
0.1001s
0.2002s
0.3003s
0.4004s
```

This means, the program is supposed to send a time stamp to the serial device every $0.1001s$.

1. Implement this program, starting from the empty μVision project that you find on the student portal. You are not allowed to use the FreeRTOS functions vTaskDelay or vTaskDelayUntil (which would not provide the required resolution anyway). Instead, set up one of the timers of the micro-controller to measure $0.1001s$, and generate the time stamps each time the timer overflows.
2. Modify your solution from the previous question to use interrupts, instead of polling until the timer overflows. The timer should now be set up to raise an interrupt every $0.1001s$. Time stamps are then generated in the interrupt service routine (ISR).
   Recall that it is not possible (due to FreeRTOS restrictions) to use the function printf in the immediate ISR; you have to use the "deferred interrupt handling" pattern.
3. Discuss, by listing advantages and disadvantages, the three solutions that you have implemented: the program developed in assignment 1, exercise 1; and the two programs written in part 1. and 2. of this exercise.

Submission

Solutions to this assignment are to be submitted
via Student Portal by
Friday, November 11th, 2016, 08:00.

Make sure that your solution is clearly marked with your name and personal number.

No solutions will be accepted after the deadline.

Please be prepared to discuss your solutions in the exercise slot.