Assignment 6
Debouncing

Embedded Logic Design

September 5, 2015

1 Debouncing

In this lab assignment you are asked to minimize the usage of functions such as `_delay_ms()`, `_delay_us()` and their derivatives with the exception that if you need delays to control the LCD, you can use those functions. Take the LCD code from your source code that you submitted in the earlier assignment.

1. The shield that comes with your Arduino kit, is equipped with buttons b_n with $1 \leq n \leq 5$. What pins on the shield are the buttons connected to?
2. Are those buttons active low (i.e. a button connects the pin to ground, if they are pressed) or active high (on activation the button is connecting the pin to VCC)?

The problem of a user interaction with a microcontroller is that the microcontroller is able to perceive events at a much higher rate than the user. If for instance a button is pressed, it is not one event, in which the button is activated, but multiple, maybe hundred events. Imagine that a user increases the pressure on the button slowly. Just before the button is activated, it might already establish a contact to VCC or ground. This contact is instable and fluctuating, essentially giving the impression that the button is pressed multiple times from the point of view of the microcontroller.

To prevent undesired behavior of the program, these accidental button presses need to be filtered out. This process is called *debouncing*. Among several variations of debouncing the simplest one is to prevent any further button presses to be processed for a specific amount of time after the first button activation.

1.1 Assignment

1. In the upper row of the LCD display an integer i_1 which is incremented, when button b_1 is activated.
2. In the lower row of the LCD display an integer i_2 which is decremented, when button b_2 is activated. i_1 and i_2 start from 0.
3. Connect 2 LEDs (l_1 and l_2) to a pin of your choice. l_1 and l_2 toggle, when button b_3 and b_4 is activated respectively.

Use Interrupt Service Routines (ISR) to handle all button activations. You may use `INT0` and `INT1` due to their ability to detect the pulse edges which eases your programming. However there are only two such pins available and the remaining buttons need to be handled somehow differently.

After a button has been activated, you are required to disallow any further button presses for 200ms. For an exact timing, use the Interrupt Service Routines given to you by the timers. Refer to your code in the previous assignments and modify it according to your needs.

2 Deliverables

All files must be submitted to `nanu.iiitd.edu.in` via `git` or `subversion`. Late submissions are not evaluated nor will be submissions through `https://www.usebackpack.com` or mail. Your repository has to contain:

- Source code
- Makefile

2.1 Remarks

If you encounter a problem, ask Google, DuckDuckGo, Bing, etc. first. The TAs will not type the question that you have, into the mask in the search engine for you. Required resources, textbooks, etc. are available on the ELD course website of `https://www.usebackpack.com` or in the Internet (datasheets, AVR library documentation, etc.)