ECE3411 – Fall 2016

Lab 3a.

General Purpose Digital Input (Debouncing) Non-Blocking UART (using ISRs)

Marten van Dijk, Chenglu Jin
Department of Electrical & Computer Engineering
University of Connecticut
Email: {marten.van_dijk, chenglu.jin}@uconn.edu

Copied from Lab 3a, ECE3411 – Fall 2015, by
Marten van Dijk and Syed Kamran Haider

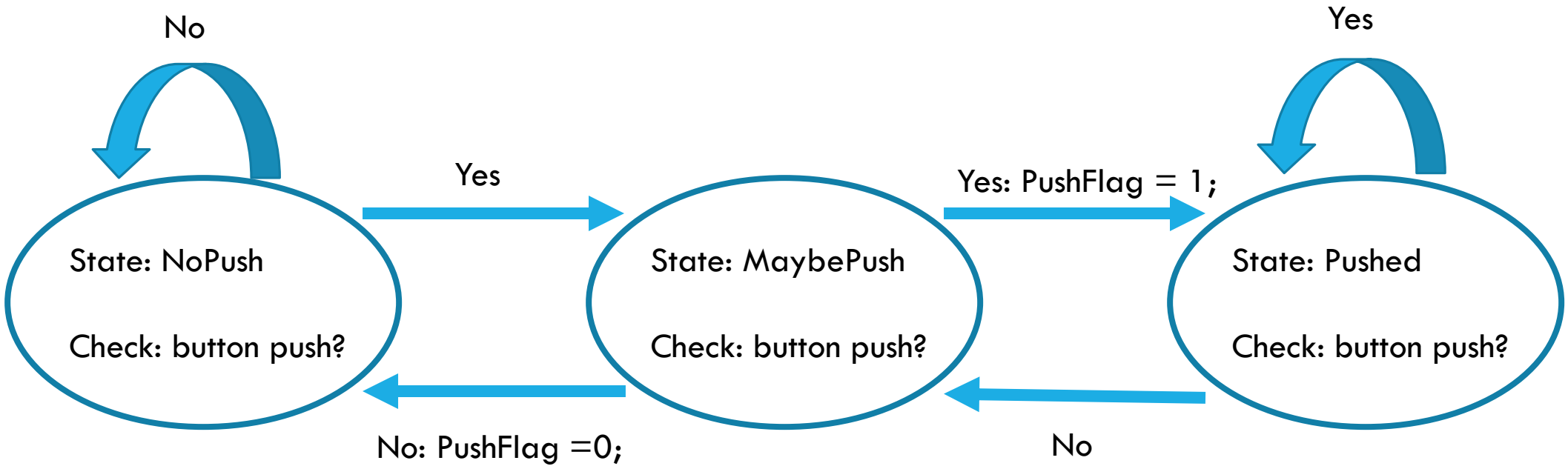
UCONN

Recap

In the last lab, we implemented the following:

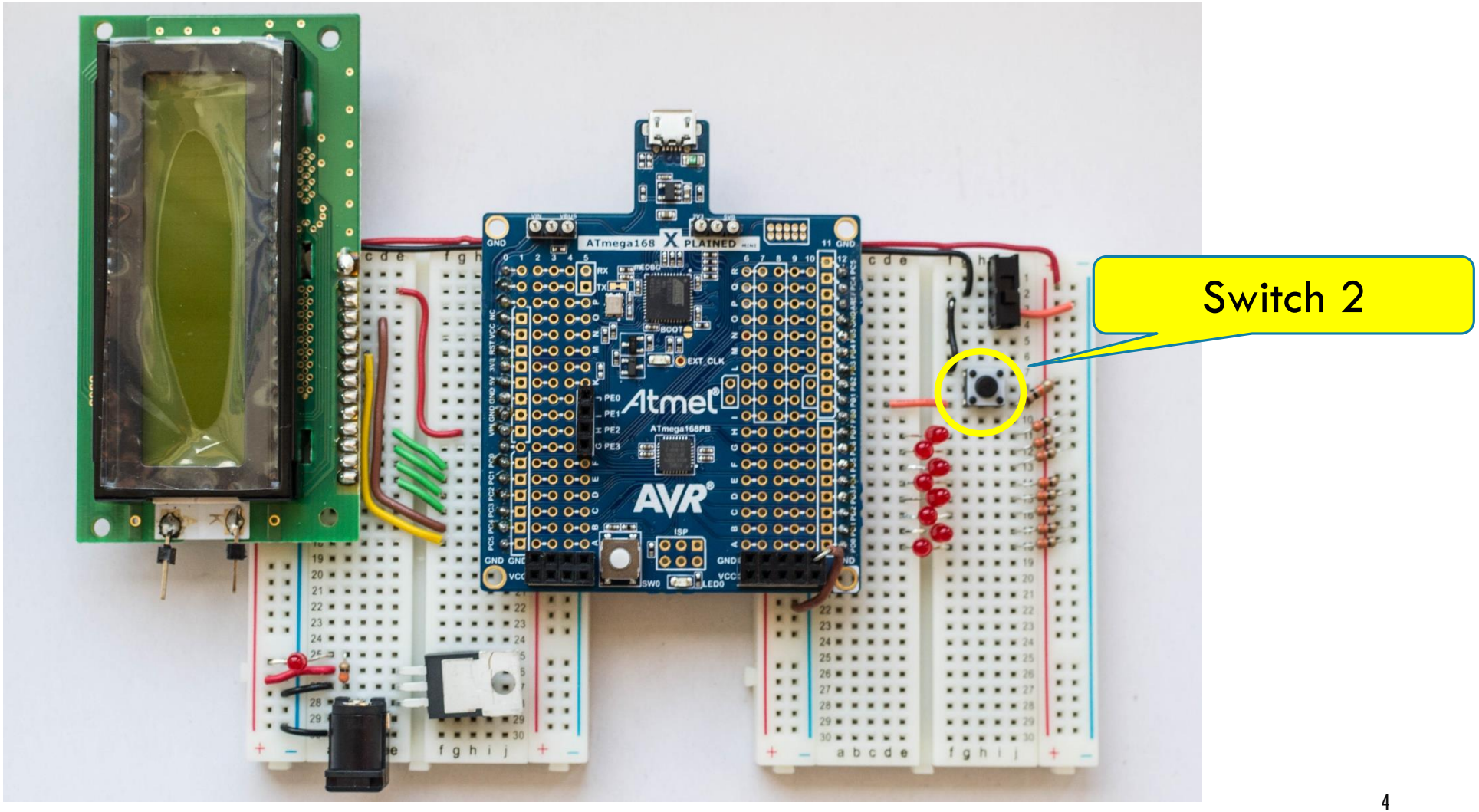
- Reading a Non-Debounced Switch
  - MCU may see a lot of glitches in the input from a Non-Debounced switch
- Reading a Debounced Switch
  - 3-state Debounce State Machine filters out glitches, but not all of them!
- Display some results on LCD

Do you see any problems with this Debounce State Machine?



- This state machine filters out glitches which result in **NoPush → MaybePush → NoPush** transitions
- What happens if a glitch causes **Pushed → MaybePush → Pushed** transitions sequence?
  - The software mistakenly thinks that a new button-push has occurred
  - Fix this problem in Task 1 of this lab

Push Switch to use



LCD Interfacing

- We are going to use the LCD in 4-bit mode
  - Only 4 data wires are required instead of 8
- LCD pin assignment is as follows:

No.	Symbol	Connections with ATmega328P
1, 3	V_{SS}, V_{EE}	GND
2	V_{CC}	5V
4	RS	PC4
5	R/W	GND (Always Write to LCD)
6	E	PC5
7-10	DB0-DB3	Not Connected
11-14	DB4-DB7	PC0-PC3

Pin1: V_{SS} → GND
Pin2: V_{CC} → 5V
Pin3: V_{EE} → GND
Pin4: RS → PC4
Pin5: R/W → GND
Pin6: E → PC5
Pin7: DB0 → N/C
Pin8: DB1 → N/C
Pin9: DB2 → N/C
Pin10: DB3 → N/C
Pin11: DB4 → PC0
Pin12: DB5 → PC1
Pin13: DB6 → PC2
Pin14: DB7 → PC3

Pin16: ANODE → 5V

Pin15: CATHODE → GND

LCD Test Program

```
// ------- Preamble -------- //
#define F_CPU 16000000UL      /* Tells the Clock Freq to the Compiler. */
#include <avr/io.h>           /* Defines pins, ports etc. */
#include <util/delay.h>       /* Functions to waste time */
#include "lcd_lib.h"          /* LCD Library */

int main(void) {
    // -------- Inits --------- //
     initialize_LCD();         /* Initialize LCD */

     LcdDataWrite('A');        /* Print a few characters for test */
     LcdDataWrite('B');
     LcdDataWrite('C');

    // ------ Event loop ------ //
    while (1) {
              /* Nothing to do */
    } /* End event loop */
    return (0);
}
```

Task 1: Extending the Debounce State Machine & LED Frequency Toggling

- Extend the 3-State Debounce State Machine such that the transition from the state **Pushed → Maybe → Pushed** is not considered a new button push
  - This eliminates the possible errors of the 3-State Debounce State Machine
- Use this extended debounce state machine to toggle the LED blinking frequency (Lab2b: Task1) using the switch
  - Each button push should toggle the LED blinking frequency between 2Hz and 8Hz. (So, no matter the duration of the button push, a single button push should toggle the frequency only once.)
  - Also print the frequency of the current mode on LCD
  - Don't forget you can use the debugging techniques we learned last week to fix your buggy code.

Task 2: Non-Blocking UART Reads

- Modify the LED frequency switching task (Lab2b: Task3) such that the UART reads are non-blocking. In other words, the LED should keep blinking when the user is asked if he wants to change the LED frequency.
  - Use UART interrupt service routine to receive the characters in a buffer (as shown in the lecture)

- Implement Task_InterpretReadBuffer() function to:
  - Properly handle the frequency switching
  - Display the current frequency on LCD