ECE3411 – Fall 2016

Lab 2c.

Debugging General Purpose Digital Input LCD Interfacing

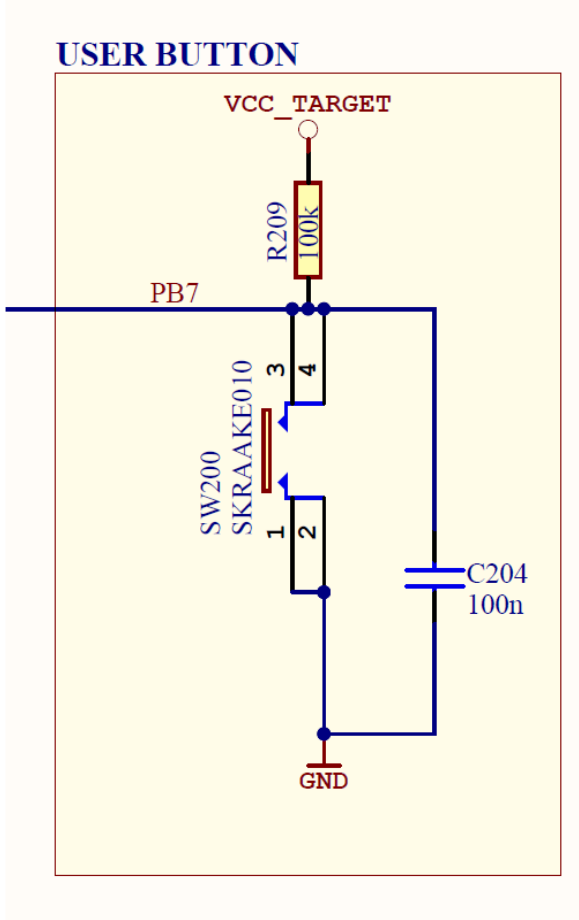
Marten van Dijk, Chenglu Jin
Department of Electrical & Computer Engineering
University of Connecticut
Email: {marten.van_dijk, chenglu.jin}@uconn.edu

UCONN

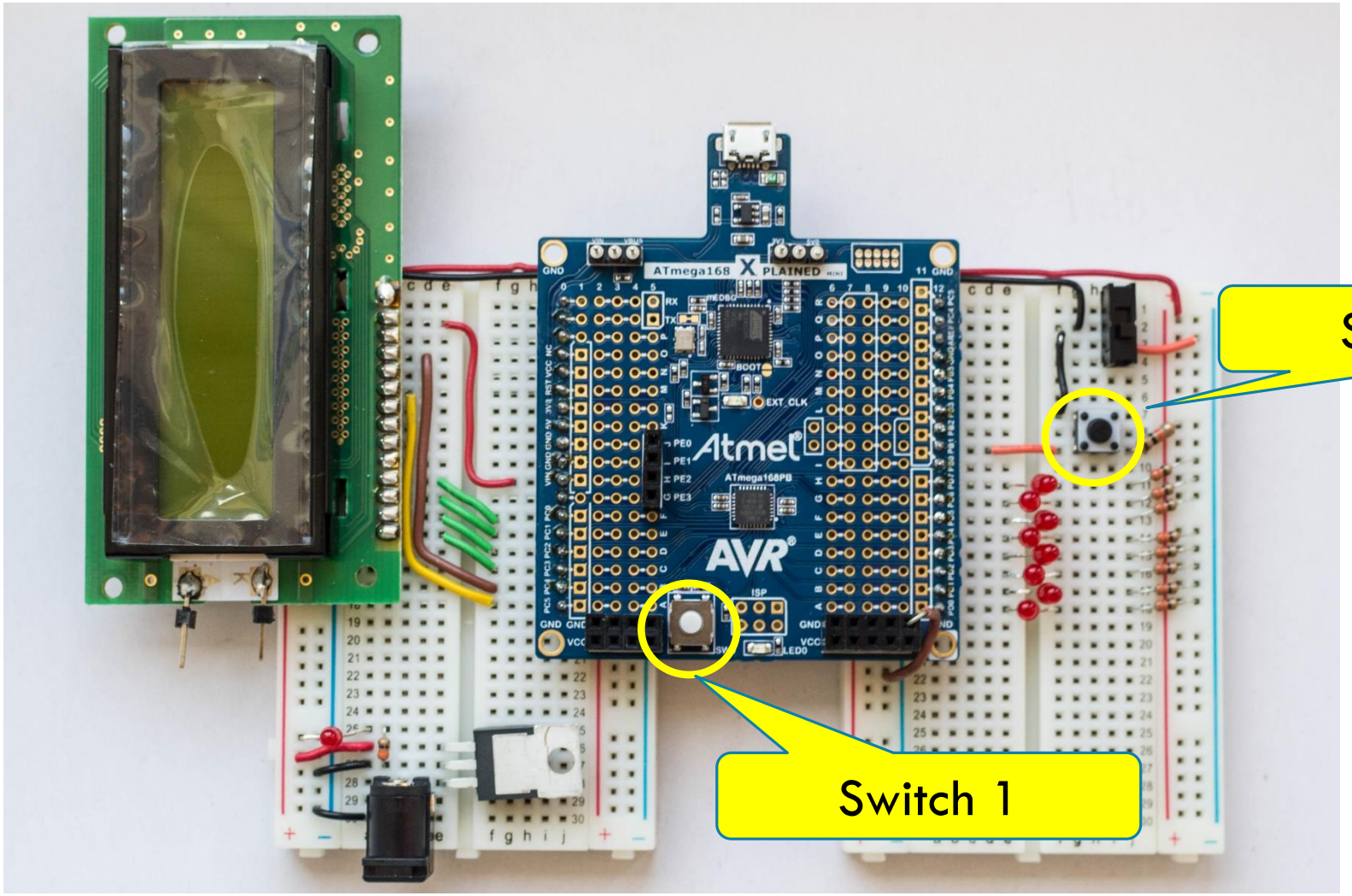
Adopted from Lab 2c slides "General Purpose Digital Input LCD Interfacing" by Marten van Dijk and Syed Kamran Haider, Fall 2015.

Push Switch Interface

- A push switch provides a logic HIGH or LOW value to the microcontroller pin to which it is connected
  - HIGH: When the switch is not pressed
  - LOW: When the switch is pressed
- Figure shows the schematic of the push button onboard ATmega328p Xplained Mini kit
  - The switch is connected to PB7
- We have another push switch on the bread board which is connected to PB1
- You should use the switch on the bread board (Switch 2) for debouncing tasks



Available Push Switches



Task 1: Debugging

1. Download the buggy code (Lab2c_1.c) from Piazza under resources. (Buggy code)
   - Correct the syntax errors in it.

2. Read the slide deck about debugging techniques.
   - The spec of the buggy code is that we want to use eight LEDs to show the number of button presses, but if you program your board with this buggy code, you will find that the count keeps incrementing.
   - Use simulator or DebugWire to help you fix this code.

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

Using LCD Library

- In order to facilitate you, we provide a library file “lcd_lib.c” which defines some useful basic LCD functions.
  - “lcd_lib.h” and “lcd_lib.c” can be downloaded from Piazza under Resources.

- The corresponding prototypes of the functions are declared in “lcd_lib.h” file which comes along with “lcd_lib.c” file.

- In order to use the function provided by “lcd_lib.c”, you need to:
  1. Add “lcd_lib.c” and “lcd_lib.h” files in your Atmel Studio project source files
  2. Include “lcd_lib.h” as a header file in your code, i.e. #include "lcd_lib.h"

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 2: Reading a Non-Debounced & Debounced Switch

- Read the input of a push switch (PINB1) and print a character ‘ * ’ on the LCD for each button push
  - Whenever the button connected to PINB1 is pushed, one ‘ * ’ is printed on LCD. (So, no matter the duration, a single button push should result in printing only one ‘ * ’.)
- Once a row of LCD is filled with characters ‘ * ’, the subsequent button pushes should start clearing the LCD
  - Most recently printed character is cleared first, and so on until all ‘ * ’ are cleared.
- Implement this task with both non-debounced and debounced switch.