ECE381 – Lab 1 – GPIO Basics

Objectives:

- Learn how to poll a GPIO pin
- Learn the structure of GPIO pins
- Learn how to debounce a switch
- Learn Hitachi LCDs and C Strings to display text for a user

Introduction:

In part1, you learned a bit about how to set up projects in PSoC Creator, as well as the very basics of Output on the PSoC4. In this lab, you will learn how to read a digital input through the process of polling. Polling is continually looping to check a pin's (or pins') value(s) to detect a change. We will use polling to scan and output keys pressed on a 4x4 keypad matrix.

Background:

Character LCDs

Character LCDs are a great, cheap way for basic text display. These devices typically use an interfacing scheme called Hitachi HD44780, after the original controller/LCD combo developed by Hitachi in the 80s. The interface consists of 3 control signals (E for enable, RS for register select, and RW for Read/Write) and 4 or 8 data signals. The protocol itself is pretty simple, *but* PSoC Creator provides a very nice LCD module (HAL) for us to use as long as you connect the control signals and data signals correctly! Then you can send C strings (char arrays that end in \0 or the byte 0) to the device with a single API call! And it even has API calls to do numbers to strings too! Excellent! *Make sure you pay real close attention to the power and ground connections on the LCD, it's easy to confuse them, and if you do, your LCD will die (and likely won't explode or burn or anything to let you know), so be very careful!* ***DOUBLE CHECK YOUR WIRING BEFORE APPLYING POWER!!!*** Our LCD is 16 columns by 2 rows (16x2).

Keypad Matrix

Numeric keypads are a common form of user input. They provide a matrix of buttons, where the left side of all the buttons in a row are common and the right side of all buttons in a column are common, as shown in Figure 1. To read the keypad, one of the common lines is *strobed* with a known voltage, then the orthogonal common lines are read one at a time. For example, if the keypad is read left-to-right, then top-to-bottom, R0 would be set to VDD (R1, R2, and R3 would be set to GND). Next, C0 would be read. If C0 reads as 1 (VDD), then the first button of row 0 would be pressed (button 1 in Figure 1) since it would short R0 to C0. Otherwise, C0 should read 0 (assuming C0 has a common pull-down resistor). Next, C1 would be read, and if 1, means the next button (button 2 in Figure 1) was pressed, then finally C3 would be read, and if 1, means the last button of row 0 is pressed (button A in Figure 1). R0 is then set to GND, R1 is set to VDD, and C0 (button 4), C1 (button 5), C2 (button 6), and C3 (button B) are read again in succession to see if any buttons in Row 1 are pressed. Repeat for R2 and R3 and all the buttons on the keypad are scanned. Repeat back with R0, and it's one big polling loop!

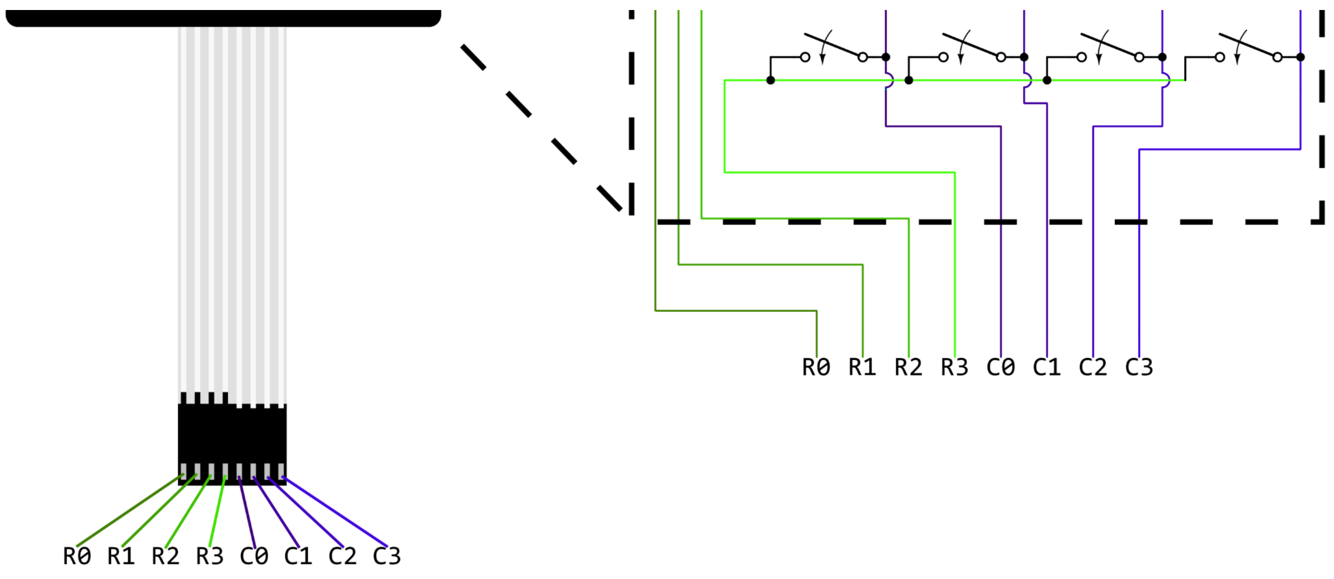

Figure 1: Keypad Diagram

If you wanted to read top-to-bottom, then left-to-right, you would strobe C0 with VDD, then read R0 (1), R1 (4), R2 (7), and R3 (*) in succession before setting C0 to GND then C1 to VDD and repeating.

The neat thing about matrices like this is that they are efficient in terms of pin usage. Normally, 16 discrete buttons would require 16 GPIO pins, but here we only need 8! This really scales for large inputs, as discrete buttons scale as Wx L, but a matrix is only W+L! (Think of a 100+ key keyboard).

Hardware Diagram:

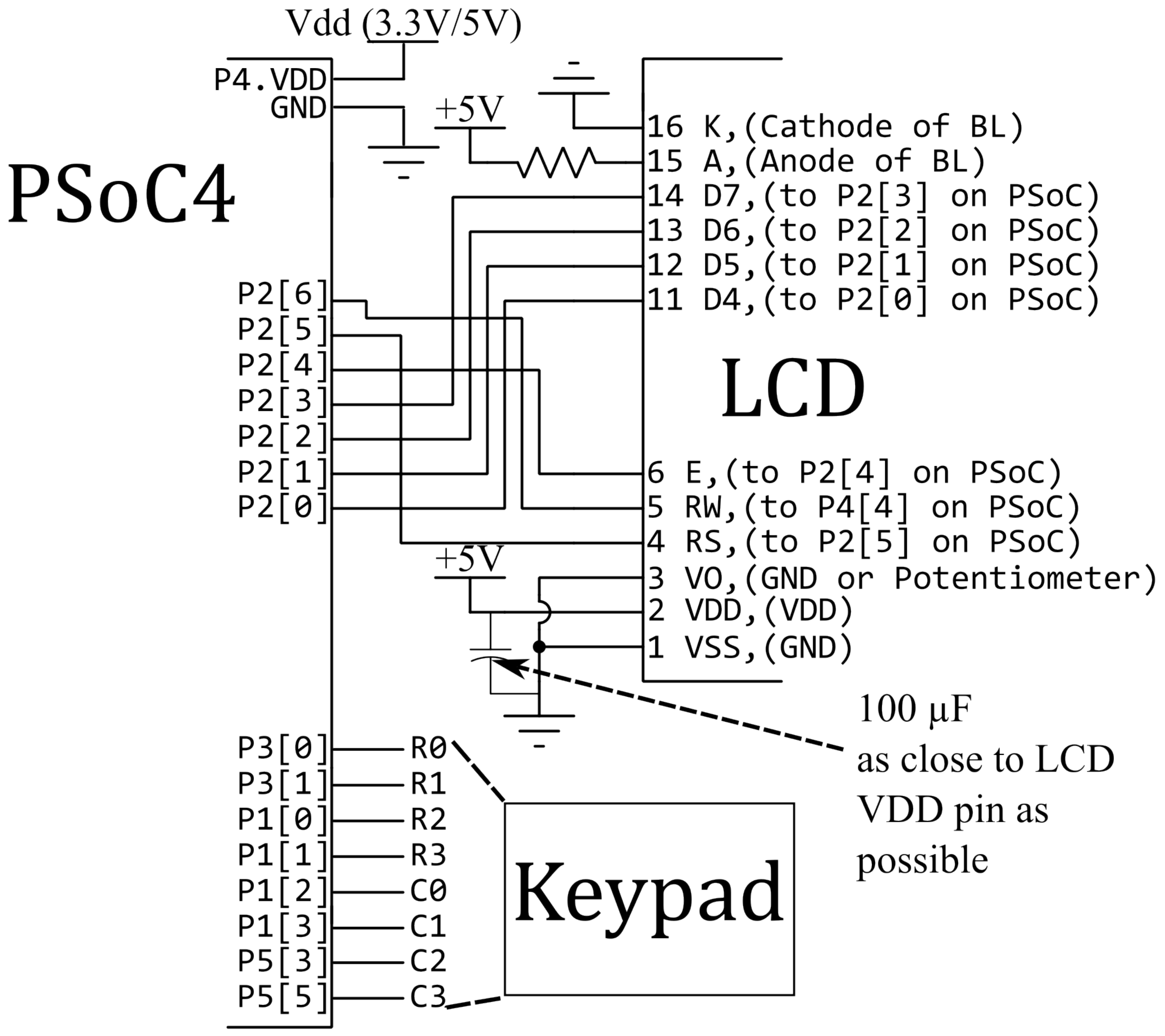

Figure 2: Hardware Schematic

Note that J9 on the CY8CKIT-044 can choose VDD to be either 5V or 3.3V. The default, I believe, is 3.3V. That's fine for all the Vdds on here EXCEPT the LCD. The LCD requires 5V for its VDD.

Configuration

- Grab a keypad and connect it as shown. J3 on the CY8CKIT-044 has those pins in a row (why I chose it)
- Grab an LCD and wire it as shown above. We will use Port2 on the PSoC4 to interface with the LCD. You will also probably want to keep it connected for future labs, so do a good job here with the wiring, and ***WATCH POWER AND GROUND! FLIPPING THEM WILL DESTROY THE LCD!!!*** Also, if your LCD has connections on pins 15 & 16, you can use

them with a resistor (ie. the 470Ω or larger in your parts kit) to turn on the LCD backlight. See the datasheet. The LCD may or may not have a built in resistor, so using one from the kit works best to limit brightness.

- Create a new PSoC Creator project. In the schematic window, from “Display” grab the “Character LCD” and drag it into the schematic window. Rename the module the “LCD” and make sure “LCD Custom Character Set” is set to None with “Include ASCII to Number Conversion Routines” checked.
- Place four, “Ports and Pins/Digital Output Pins”, and name them R0, R1, R2, and R3 respectively. For each one, right-click to configure, and uncheck the “HW Connection” box.
- Place four, “Ports and Pins/Digital Input Pins”, and name them C0, C1, C2, and C3 respectively. For each one, right-click to configure, and uncheck the “HW Connection” box. If you are not using external pull-down resistors, you should change the mode to “Resistive Pull-Down” instead of “High Impedance Digital”
- Double-click “Pins” under “Design Wide Resources” to bring up the pin assignment window. Assign “\LCD:LCDPort[6:0]\” to Port P2[6:0] as shown in Figure 2. Map R0-R3 to the pins in Figure 2, and C0-C3 to those pins in Figure 2.

    - NOTE: The Green LED is connected to P2[6], so you will likely see it on or blinking as you write to the LCD.

Requirements

- Write a program that scans the keypad for input, using row strobing as described above. Any key pressed should be output to the LCD
- Once a full row of keys is entered, the LCD should clear itself.
- Don’t forget to debounce (in software) your inputs! These are very cheap keypads, and are very bouncy.
- Also, don’t forget polling for buttons not only requires polling for press, but also waiting for release.
- Don’t forget, before your main loop, you have to have LCD_Start() for the LCD to be usable. ***ALWAYS START YOUR MODULES!!!***