LAB #4: Gadfly Keypad Interface

Lab writeup is due to your TA at the beginning of your next scheduled lab. Don’t put this off to the last minute! There is pre-lab work to complete before the start of the next lab. NO LATE LAB REPORTS WILL BE ACCEPTED.

1 Objectives

- Design the hardware interface between a keypad and microcomputer.
- Create the low-level gadfly device driver that can be used in other applications.
- Design the hardware interface between a microcomputer and a single LED.
- Implement keypad security system.

2 Reading

- Read section 2.7 about writing software device drivers.
- Read section 8.1 about keyboard scanning and debouncing.
- Description of the keypad (see link on our website).

3 Background

The keypad interface will be designed in two parts. In this lab, you will use gadfly synchronization. In the next lab, you will redesign it using interrupts. This lab demonstrates how a parallel port of the microcomputer can be used to control a keypad matrix. The hardware for the keypad is shown below. Your microcomputer should drive the rows (output 0 or HiZ) and read the columns. Your low-level software should input, scan, debounce, and save keystrokes. It should also handle two key rollover. For example, if I type “1,2,3”, I may push “1”, push “2”, release “1”, push “3”, release “2”, then release “3”.

4 Parts

This lab will require a 12-key keypad which you can purchase in the ECE lab later this week. You can use this or any 12-key keypad to do this lab. If you do find your own keypad, be sure you have specifications for it.

5 Software

Below is a prototype for your device driver:

1. Data structures: global, protected (accessed only by device driver, not the user)
   - **openFlag**: Boolean that is true if the keyboard port is open, initially false, set to true by **keyOpen**, set to false by **keyClose**, should be in static storage.
2. Initialization routines (called by user)
   KeyOpen - Initialization of the keyboard port, sets $\text{OpenFlag}$ to true, initializes the hardware, returns error code if unsuccessful (hardware non-existent, already open, etc.), no input parameters, output parameter is error code.
   KeyClose - release of keyboard port, sets $\text{OpenFlag}$ to false, returns error code if not previously open.

3. Regular I/O calls (called by user to perform I/O)
   KeyIn - input an key value from the keyboard port, waits for key to be pressed, then waits for it to be released, should support debouncing and two key rollover, returns data if successful, returns error code if unsuccessful (device not open, etc.).
   KeyStatus - returns the status of the keyboard port, returns true if a call to KeyIn would return with a key, returns false if a call to KeyIn would not return right away, but rather it would wait.

4. Support software (none in this lab, later you will add interrupt handlers here).

6 Tasks

Note: In order to use lab time efficiently, you should complete the first 3 tasks before your lab section.

1. Prepare a schematic for your design including all chips, pin numbers, and resistor values (note, you will need pull-up resistors on all the keypad inputs).
2. Write your low-level keypad device driver.
3. Write a main program which is an access code based security system.
   (a) Each access code will consist of 4 digits between 0-9.
   (b) The security system can recognize up to five access codes.
   (c) These codes will be specified in global memory.
   (d) The keypad will be used to enter these access codes.
   (e) If the access code is valid, an LED is turned on and remains on until a new key is pressed.
4. Connect your circuit and debug your software.
5. Use a scope to measure the bounce time and verify the sharpness of the digital inputs/outputs.

7 Writeup

Include the following items. In this lab, only one writeup per team is required.

1. Your hardware schematic.
2. A printout of your assembly code.
3. Collect some bounce time measurements and discuss.