

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 LEDs.
- Implement keypad security system.

2 Reading

- Read 3.1-3.5 and 8.1 in the textbook.
- Read the [keypad datasheet](#).

3 Background

Two versions of the keypad interface will be designed. In this lab, you will use gadfly synchronization. In the next lab, you will redesign it using interrupts. This lab demonstrates how a microcomputer can be used to control a keypad matrix via a parallel interface. 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".

Once again, make sure the USER option jumpers have been disabled for this lab. It is recommended that you use Port T as the interface to your keypad, as it contains an internal pull-up, pull-down network.

4 Parts

This lab will require a 12-key matrix keypad ([Grayhill 96AB2-102-R](#)) which you can purchase in the ECE lab.

5 Software

Below is a prototype for your device driver:

1. Data structures: global, private (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 `OpenFlag` to true, initializes the hardware (timers, interrupts, etc.), 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 `OpenFlag` to false, returns error code if not previously open.
3. Regular I/O calls (called by user to perform I/O)
`KeyIn` - input a 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.).

6 Tasks

1. Prepare a schematic for your design including all components used (note, if you do not use Port T, then 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. You should modify and use the FSM for the security code access system that you designed in lab 3. You will need to make the following modifications:
 - (a) Each access code now will consist of 4 digits between 0-9 which will be checked against an access code specified in global memory. The * key will be used as reset.
 - (b) The keypad will now be used to enter these digits.
4. Connect your circuit and debug your software. Check off your working system with your TA.
5. Use a scope to measure the bounce time and verify the sharpness of the digital inputs/outputs.

7 Prelab

Complete tasks 1-3 before you come to lab.

8 Writeup

1. Your hardware schematic.
2. A printout of your code.
3. Present your bounce time measurements (and waveforms if you desire). Describe the technique you used to debounce the switch and why this method is effective based on your measurements.