Lab 2 (Team Lab): A 4-digit Password System

Start: After demo Lab 1

Demo Due: October 17th, 2014

Points: 50

C Code Submission on D2L: All C files for each lab assignment and final project should be submitted on D2L in the Dropbox. After your demo, ***submit your C file*** on D2L Dropbox by ***11.59 PM on October 17th.*** Otherwise, 20% will be deducted from your lab 2 points.

Lab Overview: In this lab,

- Interface the microcontroller with a 4x3 Keypad for a user input and
- Implement (write C code) a keypad scanning function along with configuring the keypad to utilize change notification (CN) interrupts.
  Note: the LCD software driver developed in your previous lab assignment will be utilized to display results using the 8x2 LCD Display.
- You will then build a 4-digit password system with its requirement specification stated in Part 2 of this lab handout. The 32-bit timer should be used to create a 2-second delay stated in the requirement.

NOTE: As you will be incorporating additional components with your PIC microcontroller, those components can either be directly integrated in the prototyping area of the 16-bit 28-pin Starter Board or using a separate prototype board. Each team is provided with a prototype board within their Parts Kits.
If your team decides to integrate the components in the prototype area of the Starter Board, one of the team member must be willing to use his/her board for this purpose.

Wire Wrapping and Color Coding Requirements:
The provided jumper wires are free to be used for developing and testing your system, however, when it comes to the ***final demo, everything should be*** **wire wrapped** ***cleanly with "reasonable" color coding*** (using black wires all the time is obviously a BAD color coding), otherwise 5% of the points will be deducted.

Datasheets and References (on D2L)
4x3 Keypad Datasheet
LCD Display Datasheet
Microchip 16-bit 28-pin Starter Board User's Guide
PIC24FJ64GA002 Datasheet
PIC24F Family Reference Manuals
PICkit 3 User's Guide

Parts Needed for Lab (all parts are provided in your team's Parts Kit):
1 - 8x2 LCD Screen (1.50” x 1”)
1 – 4x3 Keypad
Male Headers
Optional: 4x4 Vector Prototype Board

Stand-Alone Program Demo and Code Explanation (all team members must be presented at the demo time):
Before October 17th, 2014, the TAs and/or instructor will be available on the lab and *open* lab sections for you to demonstrate your Lab 2 assignment (Part 2). During this demo, your team will need to provide both a demonstration of your working stand-alone programmed board and an explanation of the required modifications made to the provided software code.

Provided Source Code for Lab 2 Assignment: Available on class D2L
lab2.c, keypad.h, keypad.c

When writing your code, follow C Coding and Commenting Style Guidelines (the last page of this handout).

Lab Procedure and Demo:

Part 1: Keypad Interface and Keypad Scanning Driver

Hardware: Connect the 4x3 Keypad to the PIC24F microcontroller. The keypad is organized as 4 rows and 3 columns. Use the following configurations:

- Configure the ***4 rows as outputs*** and the ***3 columns as inputs***.
- For the 3 inputs, since CN interrupt will be used:
  - Use input pins (of PIC24F) that have a change notification *(labeled CNx)* assigned to it.
  - Use (internal (preferred) or external) Pull-up resistor for each input pin

  Do not use RB9 for a digital input pin.
  RB0, RB1, RB4 and RA4 can be used only in stand-alone mode.
- For the 4 outputs,
  - The same pins that are used for LCD data pins can be used. If you decide to use this, make sure that you clear the data pins in the LCD function used for sending info to LCD.
  - Use Open Drain Configuration for the outputs

Software: Display the pressed key using the LCD Display

Using the provided C code (lab2.c, keypad.h, keypad.c) complete the keypad initialization, keypad scanning, and change notification interrupt code required to interface with the 4x3 keypad and display the pressed key using the LCD Display.

Before start writing code, carefully read all comments given in all code templates. Comments are used to give you general ideas on what should be done.

To complete this functionality, you will need to complete the following:

- Develop a KeypadInitialize( ) function that will configure IOs and CN interrupt for keypad scanning. This configuration should ensure that if any key is pressed, a change notification interrupt will be generated.

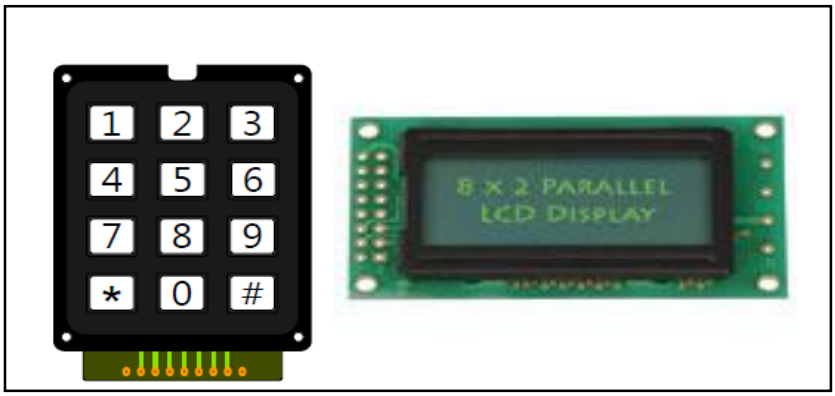
- Develop a KeypadScan() function implementing the keypad scanning procedure to detect if ***exactly*** one button of the keypad is pressed. The function should immediately return the current state of the keypad as:
  - '0' - '9' : Return the ASCII character '0' to '9' if one of the numeric (0 - 9) keys are pressed.

- '*' : Return the ASCII character '*' if the * key is pressed.
- '#' : Return the ASCII character '#' if the # key is pressed.
- -1 : Return -1 if
  - no key is pressed or
  - more than one key is pressed simultaneously.

- Code idea in general: a change notification interrupt is used to detect if *any* key of the keypad is *pressed*, and update a ***scanKeypad*** variable to indicate keypad scanning process must be executed. The keypad scanning procedure [KeypadScan()] should be called from within the main execution loop of the main( ) function.

NOTE: In implementing the KeypadScan(), a for/while/do..while loop with bit masking and shifting operations should be utilized in the scanning process for either the rows of the columns of the scanning process. Full credit will not be awarded if your code is written in such a way that both individual rows and columns are set and checked.

Part 2: A 4-digit Password System



***Software*:** Write the C program that meets the specification requirement below.

Specification Requirement:
1) The 4-digit password system lets a user enter a four-digit number (‘0’ – ‘9’) and then shows whether the entered four digits is
- a good password (match with one of the stored passwords in the database or
- a bad password (no match with any password in the database)

2) It has 2 modes: ***User mode*** (pages 3-4 of this handout) and ***Program mode*** (page 5).
3) It has password database. The first (default) password is 1234 or ‘1’, ‘2’, ‘3’, ‘4’
Note: Create an array in the main code that will be used as “database” to keep all good (valid) passwords. The first (default) password is 1234 or ‘1’, ‘2’, ‘3’, ‘4’ (depending on how your team decides on the way the valid password will be kept).

A) User mode:

A.1) It first displays a word ‘Enter’ on LCD screen (1st line of the LCD) and wait for a user to enter digits (keys pressed on the keypad). The entered keys should be displayed on the 2nd line of the LCD as it is entered.

A.2) **Each digit** of the **password can only** be **'0' – '9'.** (Note: * and # cannot be part of the good password.)

A2.1) Each entered digit must be displayed on the LCD (2nd line) as it is entered.

A2.2) If the "*****" key, followed by "*****" key are entered (i.e. the first 2 pressed keys are '*'), it should go into the **program mode** (see section B).

A2.3) If the "*" OR "#" key (except the case mentioned in A2.2) is entered, the screen should be cleared and the word 'Bad' should be displayed on the 1st line of LCD for ***2 seconds*** and it then should go back to A.1 (so that a user can enter another set of 4 digits).

A.3) If the entered **4 digits**

A3.1) **Match** with **one of** the 4-digit **passwords stored in the database**, the screen should be cleared and the word 'Good' should be displayed on the 1st line of LCD for ***2 seconds*** and it then should go back to A.1 (so that a user can enter another set of 4 digits).

A3.2) **Do not match** with **any** 4-digit **passwords stored in the database**, the screen should be cleared and the word 'Bad' should be displayed on the 1st line of LCD for ***2 seconds*** and it then should go back to A.1 (so that a user can enter another set of 4 digits).

For example: At the beginning, the **default password** already in the database is **'1', '2', '3', '4'**

- When the system starts, the LCD displays

Enter

Note: As the key get pushed, each key should be displayed on the LCD). It then makes a decision whether the entered digits are good or bad password or the user wants to change to the Program mode.

Ex 1 (Step A2.2 above): If the user pushes '*', '*', it should go into the ***Program mode*** (see section B on the next page)

Enter **	⇨	Set Mode

Ex 2 (Step A2.3 above): If the user pushes '1', '#'

Enter 1#	⇨	Bad	After 2 sec ⇨	Enter

Ex 3 (Step A3.1 above): If the user pushes '1', '2', '3', '4'

Enter 1234	⇨	Good	After 2 sec ⇨	Enter

Since these 4 digits match the default password in the database

Ex 4 (Step A3.2 above): If the user pushes '5', '3', '3', '7'

Enter 5337	⇨	Bad	After 2 sec	Enter

Since these do not match any password in the database

B) Program mode: To add a new password (to the existing database)

Note: After the new one is added into the database, all (existing and recently added) passwords in the database MUST be working.

As stated in A2.2), if the first two entered keys are '*', the system will be in the program mode. In the program mode,

B1) It first displays a word 'Set Mode' on LCD screen (1st line) and wait for a user to enter digits (keys pressed on the keypad). The entered keys should be displayed on the 2nd line of the LCD as it is entered.

B2) It will allow users to enter 4 digits followed by the '#' key. Again, each digit can only be '0' – '9'.

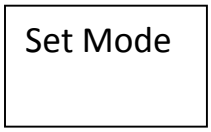
B3) If the 4 entered digits are **valid** (only contains '0' – '9'),

- this password should be added onto the existing database
- the screen should be cleared and the word 'Valid' should be displayed on the 1st line of LCD for ***2 seconds*** and it then should go back to A.1 (*back to the user mode).*

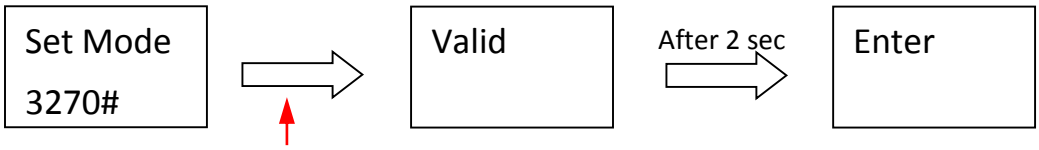
If the entered digits are **Invalid**,

- Discard these entered keys.
- the screen should be cleared and the word 'Invalid' should be displayed on the 1st line of LCD for ***2 seconds*** and it then should go back to A.1 (*back to the user mode).*

For example:
In the Program mode, the LCD screen first displays below. This results from the user pushed '*', '*' in the user mode.

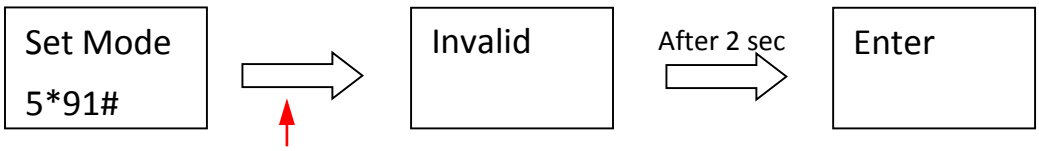


Ex 5: If the user pushes '3', '2', '7', '0', '#'



Since these 4 digits are a valid password, **"3270" will be added on to the existing database.** If this is the first new password that the user enters, it should be added right after the default password (1234)).

Ex 6: If the user pushes '5', '*', '9', '8', '#'



Since these 4 digits are NOT valid, **they will be discarded.**

Lab Score distribution (total of 50 points): (See D2L for an in-depth rubric)

- Half points if Only Part 1 (simple keypad and LCD) of this lab works
- Eighty percent of points if (up to) Part 2A (user mode of the 4-digit password system) of this lab works
- Full score if Part 2 works (both user mode and program mode)
- Extra 5 points if your system can accept at least 3 new passwords and all passwords [default (1234) and 3 new ones that your system lets the user enter] must be working.

Note: The requirement specifications in Part 2 are *minimum* requirements. To get full scores, your system must meet them. However, your team is very welcome to improve your 4-digit password system (make sure you discuss with your instructor an improvement (or modification or (better) idea) that your team wants to make before you start implementing it.

C Coding and Commenting Style Guidelines

a. All files should have the names and date. If working in a group, then all group members should be listed.
b. All functions should have a descriptive note of what it does.
c. All variables should have a descriptive name and a comment on what it's used for.
d. Any PIC configuration setting should have a comment indicating what the configuration corresponds to.
e. In the later labs, if you can use an interrupt instead of a busy wait, then you should use the interrupt.

Note: Class examples don't always adhere to this standard, be more diligent then we were.