Department of Electronic Engineering
La Trobe University

ELE31EMP – Embedded Processors
Laboratory 2 -2004
Keypad and LCD Control using a PC Parallel Port

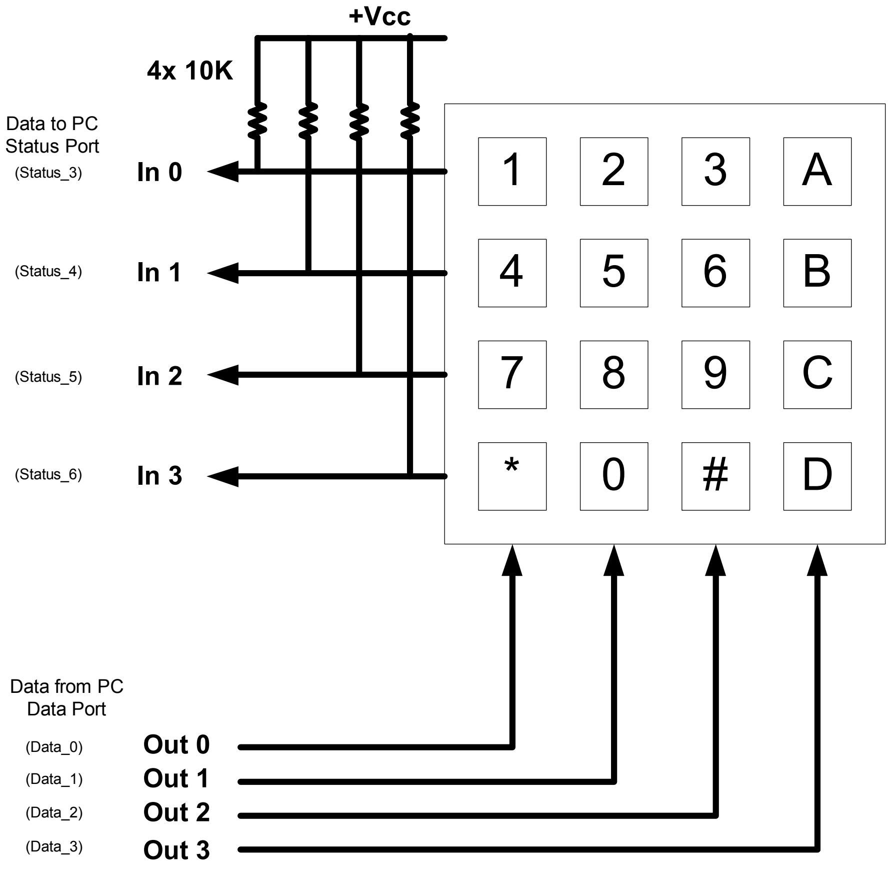
1. Introduction

In this lab you are required to be able to read a 4x4 matrix keypad and display the value of the key pressed in the command (DOS) window opened when Borland C runs. As a second part to the laboratory session, you are also required to initialise an LCD display and write the value of the key pressed on the keypad to this display.

1.1 Keypad

A matrix keypad must be scanned by
a. "activating" a row at a time and
b. then checking each key individually in that row to see if it has been pressed.

This is normally done by having 4 active low outputs and 4 inputs with pullup resistors (for a 4x4 keypad). This is illustrated in the diagram below.



Connection for reading a 4x4 Keypad

Figure 1

To scan a row, one column output is taken low, and then a row of 4 keys read. The row "data" byte (actually nibble) that has been read is then compared to the 4 values it would have been had any of the 4 keys in that row had been pressed. If it matches then that key must have been pressed. The other 3 rows are then processed similarly taking each of the other 3 outputs low in turn. (NOTE: this is a simplistic scanning system and does not allow two keys to be pressed at any one time).

The hardware connections for the scanning of the keypad will use 4 of the 8 data port lines of the parallel port as outputs and 4 of the 5 status port inputs of the parallel port for the scanned inputs. Note that the 4 unused inputs of the status port are in an indeterminate state – ie the PC interface card determines 3 of them while the last one is controlled by the test board that you have been given!

Debouncing

Any mechanical switch generally has mechanical contact bounce whenever the switch changes states. The waveform at any of the "In" ports of the above diagram (figure 1) will look something like:

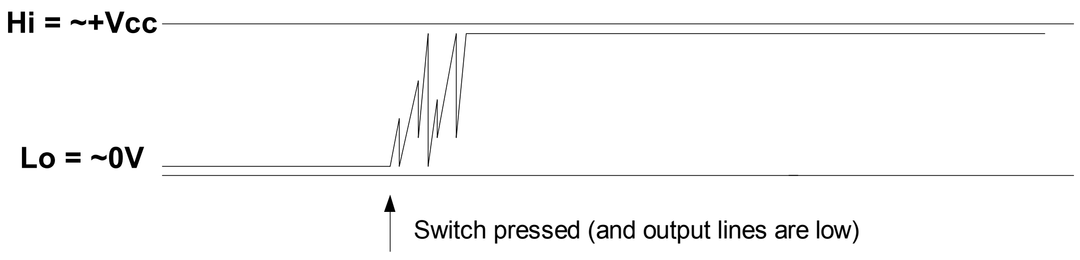


Figure 2. Mechanical switching " bounce"

The "bouncing" generally lasts 10msec or less. To stop this mechanical bounce causing misreads, a debounce routine must be introduced into the software. The simplest form of bedouncing is to read the input sequentially at an interval of about 20msec (ie something greater than 10msec, but fast enough so no latency is noticed by the user).

A sample program (only partially written) to read from the keypad is given at the website in the file:
http://www.ee.latrobe.edu.au/~jcd/emp/schedule.htm =>lab2_keypad.c

1.2 LCD

The LCD is designed as a peripheral to connect directly to the bus of a Motorola 6800 type processor. This connection can be simulated using the parallel port of a computer and synthesising the appropriate signals – in the correct sequence to meet the timing requirements as shown in the LCD data sheet (L2012.pdf). the connection of the parallel port to the LCD is shown in figure 3 below.

The data sheet for the LCD is available in directory:
http://www.ee.latrobe.edu.au/~jcd/emp/schedule.htm =>l2012.pdf

Read this data sheet carefully and understand the operation of the LCD unit. The write timing diagram should be followed in sequencing your writes to the control bits (E, R/W, RS) and the data bus.

For the purposes of this lab we will be only be writing to this display – we will not be reading from it. This requires that we connect the 8 data lines of the parallel port to the LCD's 8 data lines. The control port of the parallel port will be used to control the 3 control lines of the LCD interface.

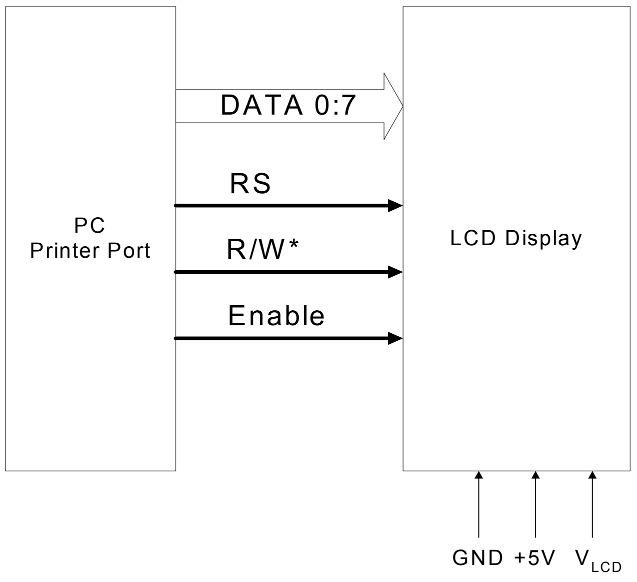
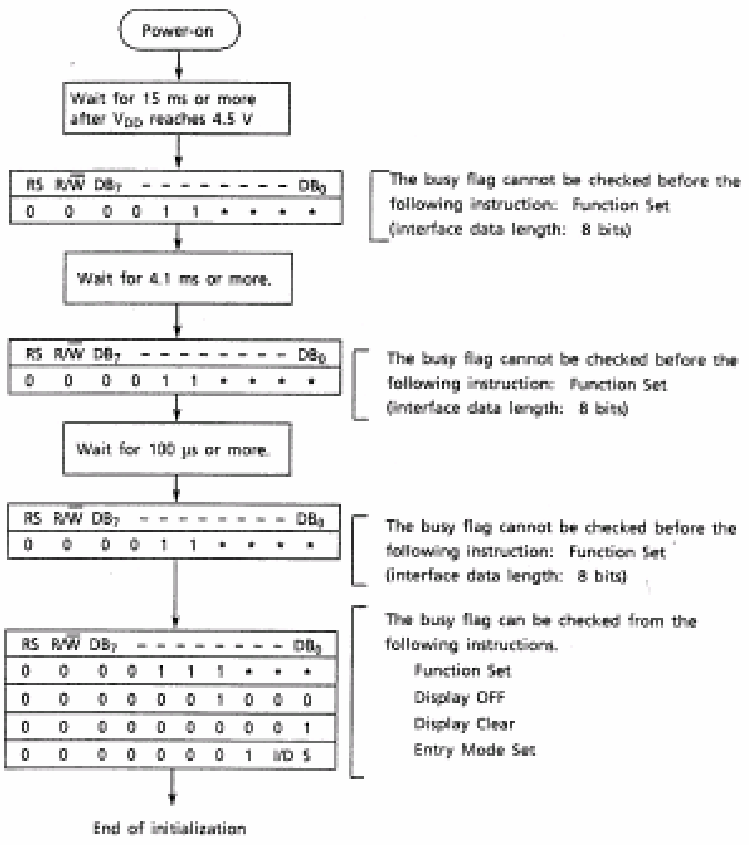


Figure 3. LCD connection to the PC Parallel Port.

The LCD is initialised with the sequence shown below in figure 4. (taken from the data sheet). The functionality to read and write to the LCD are also described in the data sheet.

A sample program to achieve LCD initialisation and write capability is given in in the file:
http://www.ee.latrobe.edu.au/~jcd/emp/schedule.htm =>lab2_LCD.c



Power-on
Wait for 15 ms or more after V_{DD} reaches 4.5 V
RS R/W DB7 - - - - - - - - DB0
0 0 0 0 1 1 * * * *
The busy flag cannot be checked before the following instruction: Function Set (interface data length: 8 bits)
Wait for 4.1 ms or more.
RS R/W DB7 - - - - - - - - DB0
0 0 0 0 1 1 * * * *
The busy flag cannot be checked before the following instruction: Function Set (interface data length: 8 bits)
Wait for 100 μs or more.
RS R/W DB7 - - - - - - - - DB0
0 0 0 0 1 1 * * * *
The busy flag cannot be checked before the following instruction: Function Set (interface data length: 8 bits)
RS R/W DB7 - - - - - - - - DB0
0 0 0 0 1 1 1 * * *
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 1 I/D S
The busy flag can be checked from the following instructions.
Function Set
Display OFF
Display Clear
Entry Mode Set
End of initialization

Figure 4 Initialisation sequence for the LCD

2. Program Specifications

2.1 Preliminaries

Two sample programs are given to help get you started. They are "**lab2_keypad.c**" and "**lab2_lcd.c**" as mentioned in section 1. They are located in the directory I:\latrobe\ele31emp\keypad_lcd. These programs are incomplete, but their code fragments give some idea of how to go about the task. Lab2_keypad.c has had some code removed and also has a simple mistake in it that some programmers, new to embedded systems, make. The removed code is associated with key encoding/decoding. The "mistake" is associated with unassigned bits.

Lab2_lcd.c is essentially complete. The initialisation code is fully functional. **Check that it meets the specifications of figure 3.** The **lcd_write()** function has had 3 lines of code removed. The required lines and the place from which they have been removed is documented in the code.

Create an appropriately named directory in your work area on H drive. A suggested directory is h:\emp\lab2. Copy these programs into it for reference.

NOTE: Borland C++ ver 5.02 still has some DOS hangover "bugs"! Ensure all directory names and file names are 8 characters or less and do not contain "whitespace"!

Start Borland C++. Create a new project with the same options as in lab 1 (easyWin application, C code, appropriate directory and path, appropriate program name). Use "options" to set the source and output paths. If you wish, copy the code fragments from lab2_keypad.c into your new c program. (You can copy the lab2_lcd.c fragments into your program later, when you are working on task 2 in the requirements section).

2.2 Requirements

2.2.1 Keypad Program

Write a simple "main program" and "scan function" to scan the keypad and detect a new key when one is pressed. (The sample code provides the basis of this.)

Write a debounce routine that calls the scan function twice, with a delay of about 20msec between calls. If the scan is different from the last time debounce was called AND the two sequential (20msec) scans are the same then a new (or no) key is pressed.

The scan_keypad function can be modelled on the lab2_keypad.c program.

2.2.2 LCD

Write

- An initialise function (actually suppled in the sample code)
- A write character function (partially supplied in the sample code)
- A write string function that calls the write character function for all characters in the string.

The LCD functions can be modelled on the lab2_lcd.c program of section 1.2

2.2.3 TASKS to implement requirements of sections 2.2.1 & 2.2.1

TASK 1:
Write a main program that calls the keypad function and displays the result (ie actual key pressed or no key pressed) to the command line continuously. The keypad must be "debounced".

TASK 2:
Using a series of "if-else" statements in the scanning code to decode the keys is not the preferred way to decode the keys. In a resource limited embedded processor system efficiency is all important. Rewrite these multiple if statements using the more efficient "switch-case" construct.

TASK 3:
Modify the main program of task 1 to initialise the LCD display at startup with a call to a separate LCD initialisation routine.

TASK 4:
Modify the main program to call a function to write a string to the LCD. Write this function. You may wish to write a separate function that writes a single character to the lcd at a time and have the write_string_to_lcd function call this function with separate characters. Then have the main program write the string "Hello World" to the LCD.

TASK 5:
With the main program scanning the keypad continually, write any new key pressed to the LCD, clearing the old key displayed on the LCD first.

3. Method

Use Borland C++ to write all the code. Use C rather than C++ code. Many of the concepts from Lab 1, as well as the two example programs (lab2_keypad.c and lab2_lcd.c) can be used as a template for Lab 2.

4. Report

A simple report for this laboratory session is required. The report should show that you:

- have done (and completed) the lab
- you understand the concepts of the lab

You may submit your software as an appendix to the report, or include it in the body of the report as like.

John Devlin
April 2002, March 2003, March 2004