LCD AND KEYBOARD INTERFACING

LCD (LIQUID CRYSTAL DISPLAY) INTERFACE

LCDs can display numbers, characters, and graphics. To produce a proper display, the information has to be periodically refreshed. This can be done by the CPU or internally by the LCD device itself. Incorporating a refreshing controller into the LCD, relieves the CPU of this task and hence many LCDs have built-in controllers. These controllers also facilitate flexible programming for characters and graphics. Table 5.1 shows the pin description of an LCD. from Optrex.

Pin no.	Symbol	External connection	Function
1	Vss	Power supply	Signal ground for LCM
2	V_{DD}		Power supply for logic for LCM
3	V_0		Contrast adjust
4	RS	MPU	Register select signal
5	R/W	MPU	Read/write select signal
6	E	MPU	Operation (data read/write) enable signal
7~10	DB0~DB3	MPU	Four low order bi-directional three-state data bus lines. Used for data transfer between the MPU and the LCM. These four are not used during 4-bit operation.
11~14	DB4~DB7	MPU	Four high order bi-directional three-state data bus lines. Used for data transfer between the MPU

Table 5.4.1 Pin description of LCD

[Source: "The 8051Microcontroller and Embedded Systems: Using Assembly and C" by Mohamed Ali Mazidi, Janice Gillispie Mazidi, Rolin McKinlay]

- Vss and VDD provide +5v and ground, V0 is used for controlling LCD contrast.
- If RS=0, the instruction command register is selected, allowing the user to send a command such as clear display, cursor at home, etc.
- If RS=1 the data register is selected, allowing the user to send data to be displayed on the LCD.
- R/W input allows the user to Read/ Write the information to the LCD.
- The enable pin is used by the LCD to latch information presented to its data pins. The 8-bit data pins are used to send information to LCD.
- Section 5.1 discusses about command codes for writing the instructions on the LCD register. Section 5.2 gives an example program for displaying a character on the LCD.

LCD COMMAND CODES

The LCD's internal controller can accept several commands and modify the display accordingly. These commands would be things like:

✓ Clear screen

✓ Return home

✓ Decrement/Increment cursor

After writing to the LCD, it takes some time for it to complete its internal operations. During this time, it will not accept any new commands or data. Figure 5.4.1 shows the command codes of LCD and Figure 5.4.2 shows the LCD interfacing. We need to insert a time delay between any two commands or data sent to LCD.

Code (Hex)	Command to LCD Instruction Register
1	Clear display screen
2	Return home
4	Decrement cursor (shift cursor to left)
6	Increment cursor (shift cursor to right)
5	Shift display right
7	Shift display left
8	Display off, cursor off
A	Display off, cursor on
C	Display on, cursor off
E	Display on, cursor blinking
F	Display on, cursor blinking
10	Shift cursor position to left
14	Shift cursor position to right
18	Shift the entire display to the left
1C	Shift the entire display to the right
80	Force cursor to beginning to 1st line
C0	Force cursor to beginning to 2nd line
38	2 lines and 5x7 matrix

Figure 5.4.1 LCD Command Codes

[Source: "The 8051Microcontroller and Embedded Systems: Using Assembly and C" by Mohamed Ali Mazidi, Janice Gillispie Mazidi, Rolin McKinlay, pg.no.353]

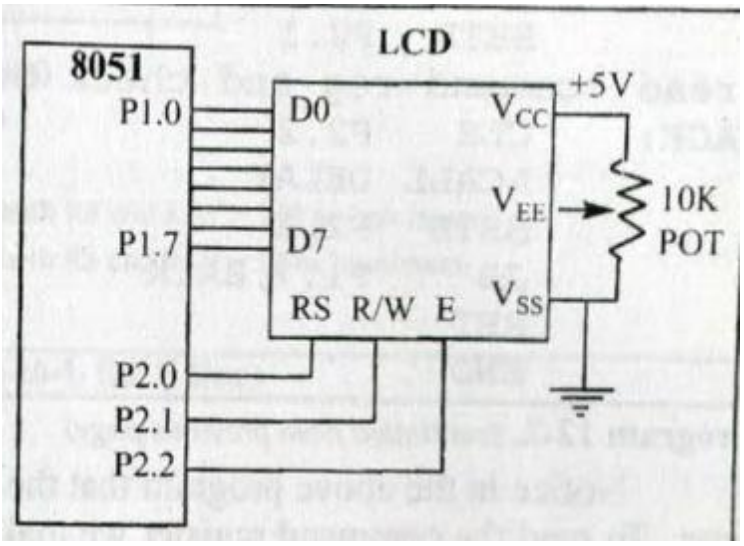

Figure 5.4.2 LCD Connections to 8051

[Source: "The 8051Microcontroller and Embedded Systems: Using Assembly and C" by Mohamed Ali Mazidi, Janice Gillispie Mazidi, Rolin McKinlay, pg.no.355]

PROGRAM TO DISPLAY CHARACTERS ON LCD

To send any of the commands to the LCD, make pin RS=0. For data, make RS=1. Then send a high-to-low pulse to the E pin to enable the internal latch of the LCD. This is shown in the code below.

```
;calls a time delay before sending next data/command
;P1.0-P1.7 are connected to LCD data pins D0-D7
;P2.0 is connected to RS pin of LCD
;P2.1 is connected to R/W pin of LCD
;P2.2 is connected to E pin of LCD
MOV A,#38H          ;INIT. LCD 2 LINES, 5X7 MATRIX
ACALL COMNWRT       ;call command subroutine
ACALL DELAY         ;give LCD some time
MOV A,#0EH          ;display on, cursor on
ACALL COMNWRT       ;call command subroutine
ACALL DELAY         ;give LCD some time
MOV A,#01           ;clear LCD
ACALL COMNWRT       ;call command subroutine
ACALL DELAY         ;give LCD some time
```

```
MOV A,#06H          ;shift cursor right
ACALL COMNWRT       ;call command subroutine
ACALL DELAY         ;give LCD some time
MOV A,#84H          ;cursor at line 1, pos. 4
ACALL COMNWRT       ;call command subroutine
ACALL DELAY         ;give LCD some time
MOV A,#'N'          ;display letter N
ACALL DATAWRT       ;call display subroutine
ACALL DELAY         ;give LCD some time
MOV A,#'O'          ;display letter O
ACALL DATAWRT       ;call display subroutine
AGAIN: SJMP AGAIN   ;stay here

COMNWRT:            ;send command to LCD
MOV P1,A            ;copy reg A to port 1
CLR P2.0            ;RS=0 for command
CLR P2.1            ;R/W=0 for write
SETB P2.2           ;E=1 for high pulse
CLR P2.2            ;E=0 for H-to-L pulse
RET

DATAWRT:            ;write data to LCD
MOV P1,A            ;copy reg A to port 1
SETB P2.0           ;RS=1 for DATA
CLR P2.1            ;R/W=0 for write
SETB P2.2           ;E=1 for high pulse
CLR P2.2            ;E=0 for H-to-L pulse
RET

DELAY: MOV R3,#50   ;50 or higher for fast CPUs
```

```
HERE 2: MOV R4,#255        ;R4 = 255
HERE: DJNZ R4,HERE         ;stay until R4 becomes 0
DJNZ R3, HERE 2
RET
```

KEYBOARD INTERFACING WITH 8051

Keys in a keyboard are arranged in a matrix of rows and columns. The controller access both rows and columns through ports. Using two ports, we can connect to an 8x8 or a 4x4 matrix keyboard. When a key is pressed, a row and column make a contact, otherwise there is no contact. We will look at the details using a 4x4 keyboard.

4X 4 KEYBOARD

Figure 5.4.31 shows a 4 x4 matrix connected to two ports.

- The rows are connected to an output port(Port 1) and the columns are connected to an input port. (Port 2)
- If no key has been pressed, reading the input port will yield 1s for all columns since they are all connected to high (Vcc).
- If all the rows are grounded and a key is pressed, one of the columns will have 0 since the key pressed provides the path to ground.
- It is the function of the microcontroller to scan the keyboard continuously to detect and identify the key pressed.

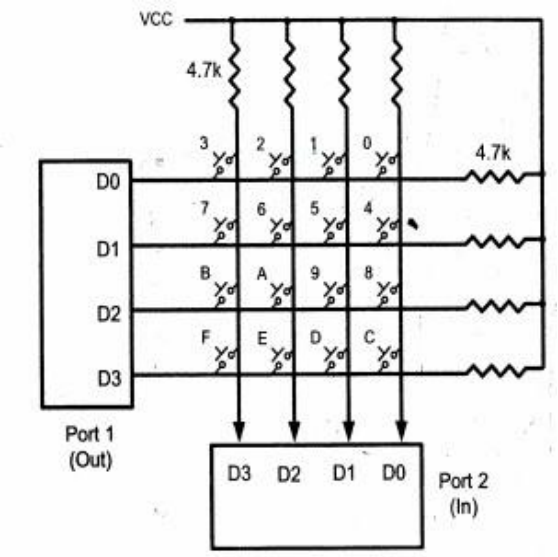

Figure 5.4.3 Matrix Keyboard Connections to Ports

[Source: "The 8051Microcontroller and Embedded Systems: Using Assembly and C" by Mohamed Ali Mazidi,

Janice Gillispie Mazidi, Rolin McKinlay]

KEY SCAN

To find out the key pressed , the controller grounds a row by sending a '0' on the corresponding line of the output port. It then reads the data at the columns using the input port. If data from columns is D3-D0=1111, then no key is pressed. If any bit of the column is '0', it indicates that a key is pressed in that column. In this example, the column is identified by the following values: 1110 – key pressed in column 0

1101 – key pressed in column 1

1011 – key pressed in column 2

0111 – key pressed in column 3

STEPS TO FIND OUT KEY PRESSED

Beginning with the row 0, the microcontroller grounds it by providing a low to row D0 only. It then reads the columns (port2). If the data read is all 1s, then no key in that row is activated and the process is moved to the next row. It then grounds the next row, reads the columns, and checks for any zero. This process continues until a row with a zero is identified. After identification of the row in which the key has been pressed, the column to which the pressed key belongs is identified as discussed above - by looking for a zero in the input values read.

Example:

(a) D3 – D0 = 1101 for the row, D3 – D0 = 1011 for the column, indicate row 1 and column 3 are selected. This indicates that key 6 is pressed.

(b) D3 – D0 = 1011 for the row, D3 – D0 = 0111 for the column, indicate row 2 and column 3 are selected. Then key 'B' is pressed.

PROGRAM:

The program used for detection and identification of the key activated goes through the following stages:

1. To make sure that the preceding key has been released, 0s are output to all rows at once, and the columns are read and checked repeatedly until all the columns are high.

- When all columns are found to be high, the program waits for a short amount of time before it goes to the next stage of waiting for a key to be pressed.

2. To see if any key is pressed, the columns are scanned over and over in an infinite loop until one of them has a 0 on it.

- Remember that the output latch is connected to rows, still have their initial zeros (in stage 1), making them grounded.
- After the key press detection, it waits for 20-ms for the bounce and then scans the columns again.

i) It ensures that the first key press detection was not an erroneous one due to spike noise.

ii) After the 20-ms delay, if the key is still pressed, then it goes to the loop (step 3) to detect the actual key pressed.

3. To detect which row the key pressed belongs to, it grounds one row at a time, reading the columns each time.

- If it finds that all columns are high, this means that the key press does not belong to that row. Therefore, it grounds the next row and continues until it finds the row, that the key pressed belongs to.
- Upon finding the row that the key pressed belongs to, it sets up the starting address for the lookup table holding the scan codes for that row.

4. To identify the key pressed, it rotates the column bits, one bit at a time, into the carry flag and checks to see if it is low.

- Upon finding the zero, it pulls out the ASCII code for that key from the look-up table.
- Otherwise, it increments the pointer to point to the next element of the look-up table.

Figure 5.4.4 provides the flowchart for keyboard interfacing Program for scanning and identifying the pressed key.

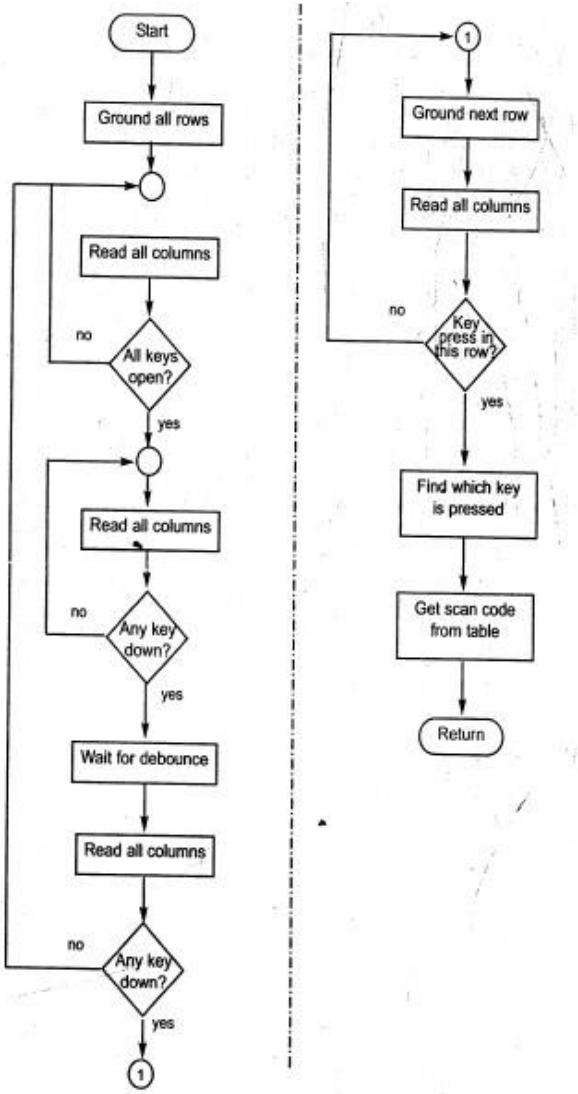

Figure 5.4.4 Flowchart for Keyboard Interfacing

[Source: "The 8051Microcontroller and Embedded Systems: Using Assembly and C" by Mohamed Ali Mazidi, Janice Gillispie Mazidi, Rolin McKinlay, pg.no.365]

PROGRAM:

```
;keyboard subroutine. This program sends the ASCII code for pressed key to P0.1
;P1.0-P1.3 connected to rows, P2.0-P2.3 to column
MOV P2,#0FFH                ;make P2 an input port
K1: MOV P1,#0               ;ground all rows at once
MOV A,P2                    ;read all col
                            ;(ensure keys open)
ANL A,00001111B             ;masked unused bits
CJNE A,#00001111B,K1        ;till all keys release
```

```
K2: ACALL DELAY          ;call 20 msec delay
MOV A,P2                 ;see if any key is pressed
ANL A,00001111B          ;mask unused bits
CJNE A,#00001111B,OVER   ;key pressed, find row
SJMP K2                  ;check till key pressed
OVER: ACALL DELAY        ;wait 20 msec debounce time
MOV A,P2                 ;check key closure
ANL A,00001111B          ;mask unused bits
CJNE A,#00001111B,OVER1  ;key pressed, find row
SJMP K2                  ;if none, keep polling
OVER1: MOV P1, #11111110B ;ground row 0
MOV A,P2                 ;read all columns
ANL A,#00001111B         ;mask unused bits
CJNE A,#00001111B,ROW_0  ;key row 0, find col.
MOV P1,#11111101B        ;ground row 1
MOV A,P2                 ;read all columns
ANL A,#00001111B         ;mask unused bits
CJNE A,#00001111B,ROW_1  ;key row 1, find col.
MOV P1,#11111011B        ;ground row 2
MOV A,P2                 ;read all columns
ANL A,#00001111B         ;mask unused bits
CJNE A,#00001111B,ROW_2  ;key row 2, find col.
MOV P1,#11110111B        ;ground row 3
MOV A,P2                 ;read all columns
ANL A,#00001111B         ;mask unused bits
CJNE A,#00001111B,ROW_3  ;key row 3, find col.
LJMP K2                  ;if none, false input,
ROW_0: MOV DPTR,#KCODE0  ;set DPTR=start of row 0
SJMP FIND                ;find col. Key belongs to
ROW_1: MOV DPTR,#KCODE1  ;set DPTR=start of row
```

```
SJMP FIND                ;find col. Key belongs to
ROW_2: MOV DPTR,#KCODE2  ;set DPTR=start of row 2
SJMP FIND                ;find col. Key belongs to
ROW_3: MOV DPTR,#KCODE3  ;set DPTR=start of row 3
FIND: RRC A              ;see if any CY bit low
JNC MATCH                ;if zero, get ASCII code
INC DPTR                 ;point to next col. addr
SJMP FIND                ;keep searching
MATCH: CLR A             ;set A=0 (match is found)
MOVC A,@A+DPTR           ;get ASCII from table
MOV P0,A                 ;display pressed key
LJMP K1
```

```
; ASCII LOOK-UP TABLE FOR EACH ROW
ORG 300H
KCODE0: DB '0','1','2','3'   ;ROW 0
KCODE1: DB '4','5','6','7'   ;ROW 1
KCODE2: DB '8','9','A','B'   ;ROW 2
KCODE3: DB 'C','D','E','F'   ;ROW 3
```