CHAPTER 12

LCD AND KEYBOARD INTERFACING

LCD Operation

- LCD is finding widespread use replacing LEDs
  - The declining prices of LCD
  - The ability to display numbers, characters, and graphics
  - Incorporation of a refreshing controller into the LCD
    - Relieving the CPU of the task of refreshing the LCD
  - Ease of programming for characters and graphics

Table 12-1: Pin Descriptions for LCD

Pin	Symbol	I/O	Description
1	V_{SS}	--	Ground
2	V_{CC}	--	+5V power supply
3	V_{EE}	--	Power supply to control contrast
4	RS	I	RS = 0 to select command register, RS = 1 to select data register
5	R/W	I	R/W = 0 for write, R/W = 1 for read
6	E	I/O	Enable
7	DB0	I/O	The 8-bit data bus
8	DB1	I/O	The 8-bit data bus
9	DB2	I/O	The 8-bit data bus
10	DB3	I/O	The 8-bit data bus
11	DB4	I/O	The 8-bit data bus
12	DB5	I/O	The 8-bit data bus
13	DB6	I/O	The 8-bit data bus
14	DB7	I/O	The 8-bit data bus

- Send displayed information or instruction command codes to the LCD
- Read the contents of the LCD's internal registers

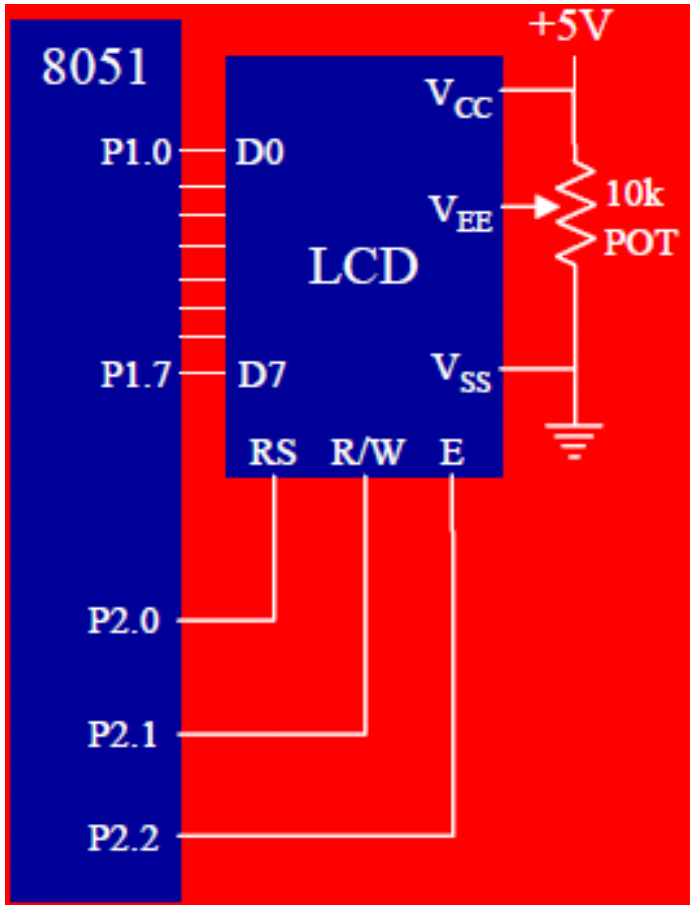
used by the LCD to latch information presented to its data bus

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.

Table 12-2: LCD Command Codes

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 of 1st line
C0	Force cursor to beginning of 2nd line
38	2 lines and 5x7 matrix

Note: This table is extracted from Table 12-4.



```
;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
          ORG    0H
          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 port1
         CLR    P2.0           ;RS=0 for command
         CLR    P2.1           ;R/W=0 for write
         SETB   P2.2           ;E=1 for high pulse
         ACALL  DELAY          ;give LCD some time
         CLR    P2.2           ;E=0 for H-to-L pulse
         RET
DATAWRT:                       ;write data to LCD
         MOV    P1,A           ;copy reg A to port1
         SETB   P2.0           ;RS=1 for data
         CLR    P2.1           ;R/W=0 for write
         SETB   P2.2           ;E=1 for high pulse
         ACALL  DELAY          ;give LCD some time
         CLR    P2.2           ;E=0 for H-to-L pulse
         RET
```

Program 12-1: Communicating with LCD using a delay *(continued on next page)*

```
DELAY:   MOV    R3,#50         ;50 or higher for fast CPUs
HERE2:   MOV    R4,#255        ;R4=255
HERE:    DJNZ   R4,HERE        ;stay until R4 becomes 0
         DJNZ   R3,HERE2
         RET
         END
```

Program 12-1. *(continued from previous page)*

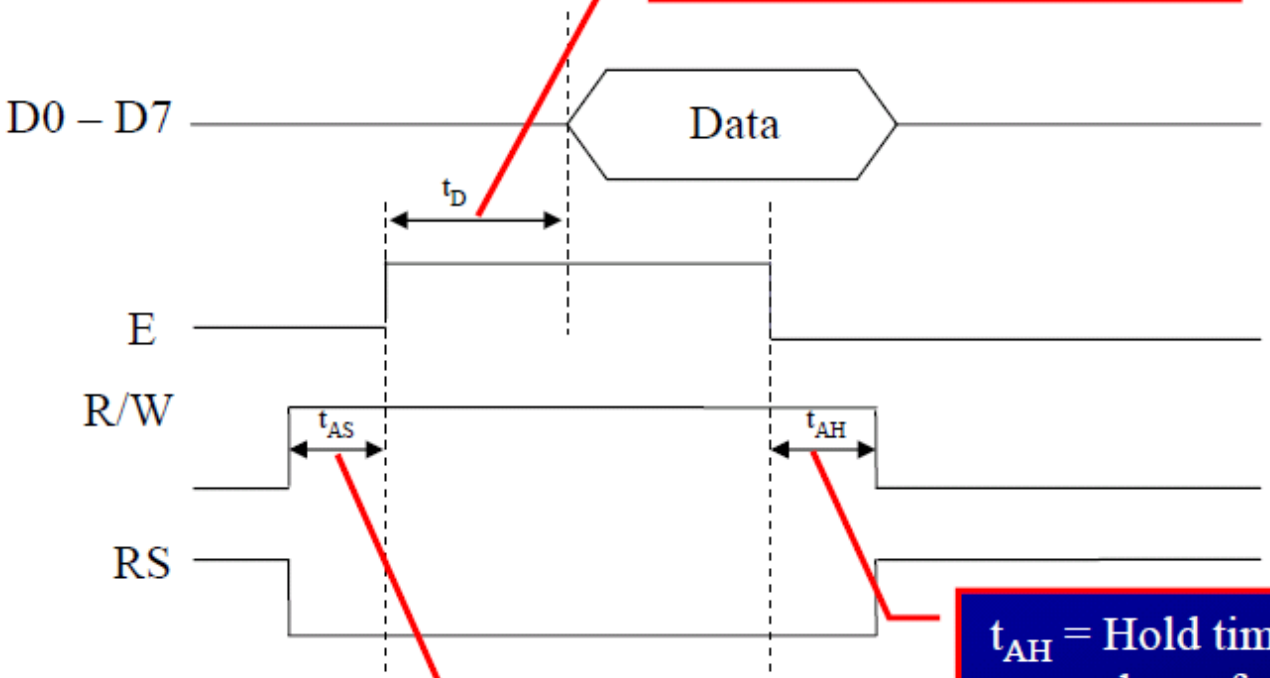
```
;Check busy flag before sending data, command to LCD
;p1=data pin
;P2.0 connected to RS pin
;P2.1 connected to R/W pin
;P2.2 connected to E pin
      ORG   0H
      MOV   A,#38H        ;init. LCD 2 lines ,5x7 matrix
      ACALL COMMAND       ;issue command
      MOV   A,#0EH        ;LCD on, cursor on
      ACALL COMMAND       ;issue command
      MOV   A,#01H        ;clear LCD command
      ACALL COMMAND       ;issue command
      MOV   A,#06H        ;shift cursor right
      ACALL COMMAND       ;issue command
      MOV   A,#86H        ;cursor: line 1, pos. 6
      ACALL COMMAND       ;command subroutine
      MOV   A,#'N'        ;display letter N
      ACALL DATA_DISPLAY
      MOV   A,#'O'        ;display letter O
      ACALL DATA_DISPLAY
HERE:SJMP  HERE           ;STAY HERE
.....
```

```
.....
COMMAND:
      ACALL READY          ;is LCD ready?
      MOV   P1,A           ;issue command code
      CLR   P2.0           ;RS=0 for command
      CLR   P2.1           ;R/W=0 to write to LCD
      SETB  P2.2           ;E=1 for H-to-L pulse
      CLR   P2.2           ;E=0,latch in
      RET
DATA_DISPLAY:
      ACALL READY          ;is LCD ready?
      MOV   P1,A           ;issue data
      SETB  P2.0           ;RS=1 for data
      CLR   P2.1           ;R/W =0 to write to LCD
      SETB  P2.2           ;E=1 for H-to-L pulse
      CLR   P2.2           ;E=0 latch in
      RET
READY:
      SETB  P1.7           ;make P1.7 input port
      CLR   P2.0           ;RS=0 access command reg
      SETB  P2.1           ;R/W=1 read command reg
;read command reg and check busy flag
BACK:SETB  P2.2            ;E=1 for H-to-L pulse
      CLR   P2.2           ;E=0 H-to-L pulse
      JB    P1.7,BACK      ;stay until busy flag=0
      RET
      END
```

To read the command register, we make R/W=1, RS=0, and a H-to-L pulse for the E pin.

If bit 7 (busy flag) is high, the LCD is busy and no information should be issued to it.

LCD Timing for Read



t_D = Data output delay time

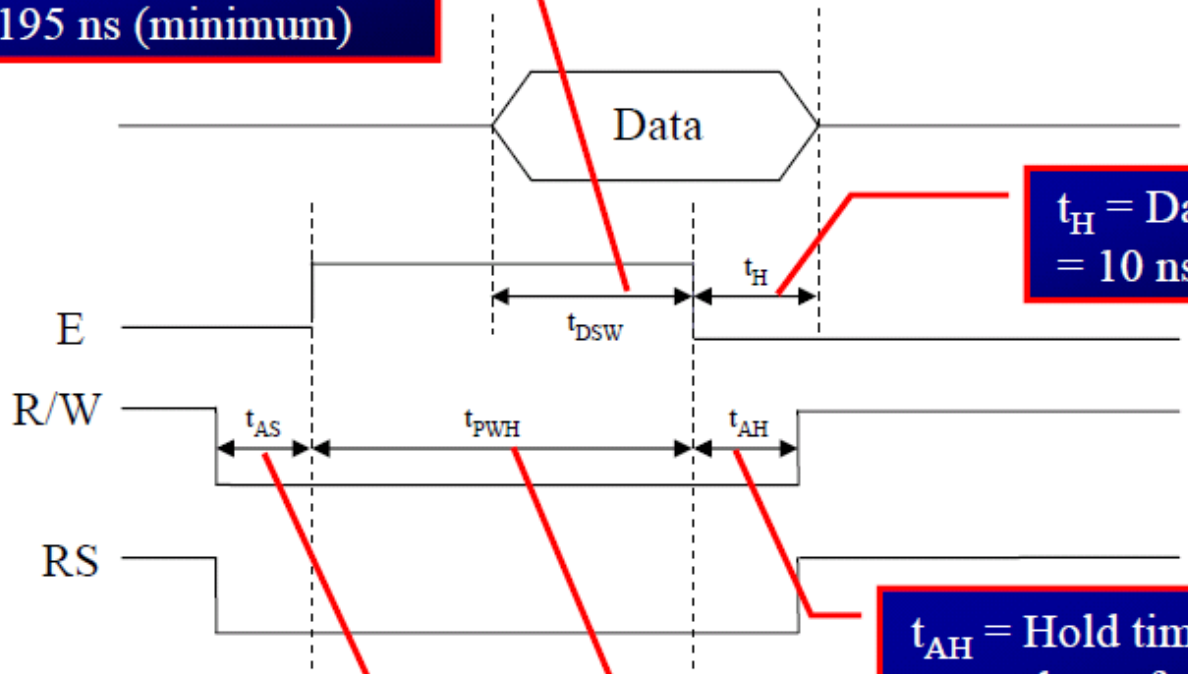
t_{AH} = Hold time after E has come down for both RS and R/W = 10 ns (minimum)

t_{AS} = Setup time prior to E (going high) for both RS and R/W = 140 ns (minimum)

Note : Read requires an L-to-H pulse for the E pin

LCD Timing for Write

t_{DSW} = Data set up time = 195 ns (minimum)



t_H = Data hold time = 10 ns (minimum)

t_{AH} = Hold time after E has come down for both RS and R/W = 10 ns (minimum)

t_{PWH} = Enable pulse width = 450 ns (minimum)

t_{AS} = Setup time prior to E (going high) for both RS and R/W = 140 ns (minimum)

LCD Data Sheet

- One can put data at any location in the LCD
  - The following shows address locations and how they are accessed
    - AAAAAAA=000_0000 to 010_0111 for line1
    - AAAAAAA=100_0000 to 110_0111 for line2
      - The upper address range can go as high as 0100111 for the 40-character-wide LCD
      - Corresponds to locations 0 to 39

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	1	A	A	A	A	A	A	A

```
;Call a time delay before sending next data/command
; P1.0-P1.7=D0-D7, P2.0=RS, P2.1=R/W, P2.2=E

        ORG   0
        MOV   DPTR,#MYCOM
C1:     CLR   A
        MOVC  A,@A+DPTR
        ACALL COMNWRT  ;call command subroutine
        ACALL DELAY    ;give LCD some time
        INC   DPTR
        JZ    SEND_DAT
        SJMP  C1
SEND_DAT:
        MOV   DPTR,#MYDATA
D1:     CLR   A
        MOVC  A,@A+DPTR
        ACALL DATAWRT  ;call command subroutine
        ACALL DELAY    ;give LCD some time
        INC   DPTR
        JZ    AGAIN
        SJMP  D1
AGAIN:  SJMP  AGAIN    ;stay here
.....
```

```
.....
COMNWRT:                ;send command to LCD
        MOV   P1,A      ;copy reg A to P1
        CLR   P2.0      ;RS=0 for command
        CLR   P2.1      ;R/W=0 for write
        SETB  P2.2      ;E=1 for high pulse
        ACALL DELAY     ;give LCD some time
        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
        ACALL DELAY     ;give LCD some time
        CLR   P2.2      ;E=0 for H-to-L pulse
        RET
DELAY:  MOV   R3,#250   ;50 or higher for fast CPUs
HERE2:  MOV   R4,#255   ;R4 = 255
HERE:   DJNZ  R4,HERE   ;stay until R4 becomes 0
        DJNZ  R3,HERE2
        RET
        ORG   300H
MYCOM:  DB    38H,0EH,01,06,84H,0 ; commands and null
MYDATA: DB    "HELLO",0
        END
```

Keyboard Interfacing

- Keyboards are organized in a matrix of rows and columns
  - The CPU accesses both rows and columns through ports
    - With two 8-bit ports, an 8 x 8 matrix of keys can be connected to a microprocessor
    - When a key is pressed, a row and a column make a contact
      - Otherwise, there is no connection between rows and columns
  - In IBM PC keyboards, a microcontroller takes care of hardware and software interfacing

Keyboard Interfacing (cont.)

- A 4x4 matrix connected to two ports
  - The rows are connected to an output port
  - The columns are connected to an input port
    - If no key has been pressed, reading the input port will yield 1s for all columns
      - Since they are all connected to high (V_{cc})
    - 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

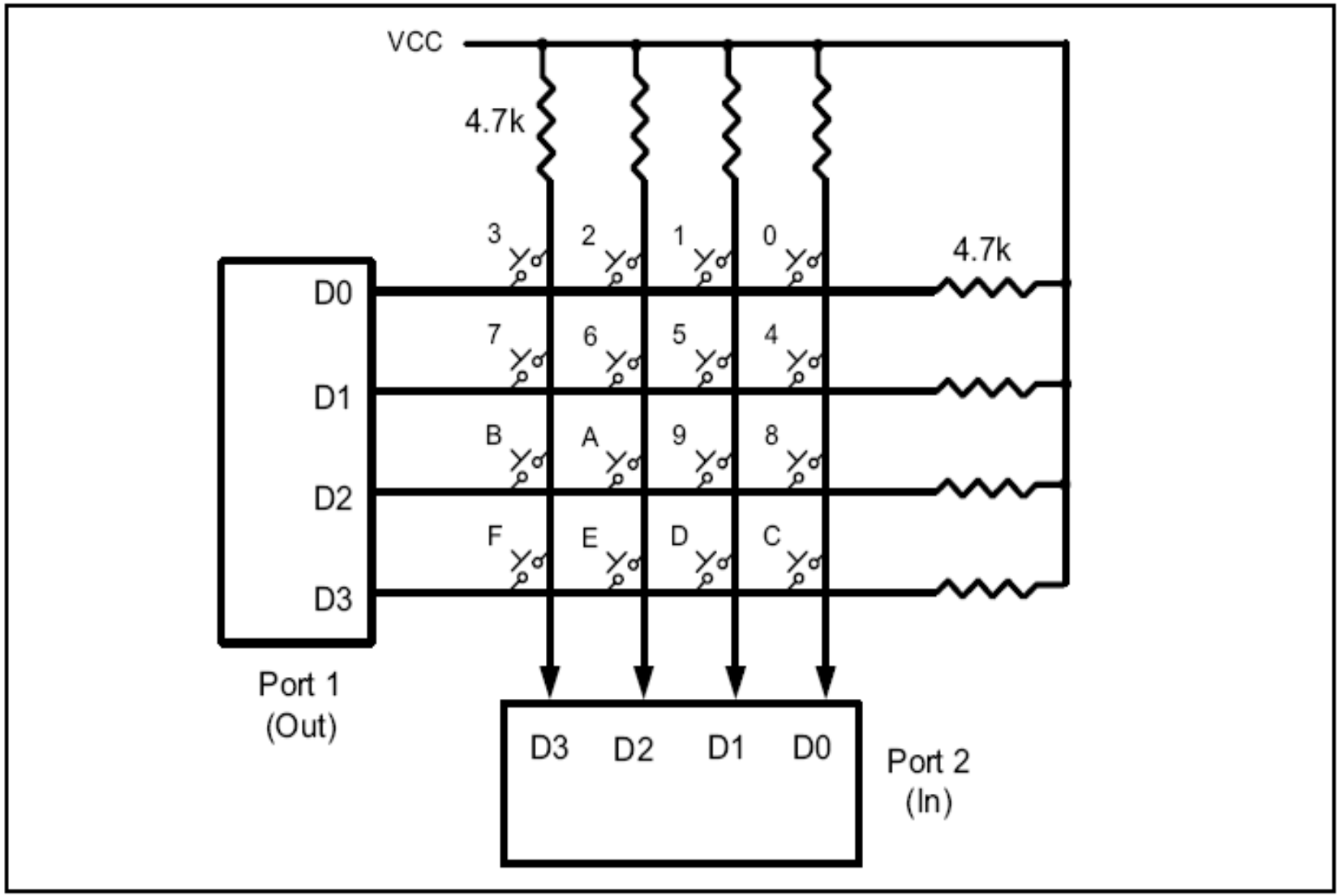


Figure 12-6. Matrix Keyboard Connection to Ports

Grounding Rows and Reading Columns

- To detect a pressed key
  - The microcontroller grounds all rows by providing 0 to the output latch
  - Then it reads the columns
    - If the data read from columns is D3 – D0 = 1111, no key has been pressed
      - The process continues till key press is detected
    - If one of the column bits has a zero, this means that a key press has occurred
    - For example, if D3 – D0 = 1101, this means that a key in the D1 column has been pressed

Grounding Rows and Reading Columns (cont.)

- After detecting a key press, the microcontroller will go through the process of identifying the key
  - Starting with the top row, the microcontroller grounds it by providing a low to row D0 only
    - It reads the columns, if the data read is all 1s, no key in that row is activated
      - The process is moved to the next row
  - It grounds the next row, reads the columns, and checks for any zero

Grounding Rows and Reading Columns (cont.)

- This process continues until the row is identified
- After identification of the row in which the key has been pressed
  - Find out which column the pressed key belongs to

Example 12-3

From Figure 12-6, identify the row and column of the pressed key for each of the following.

(a) D3 - D0 = 1110 for the row, D3 - D0 = 1011 for the column

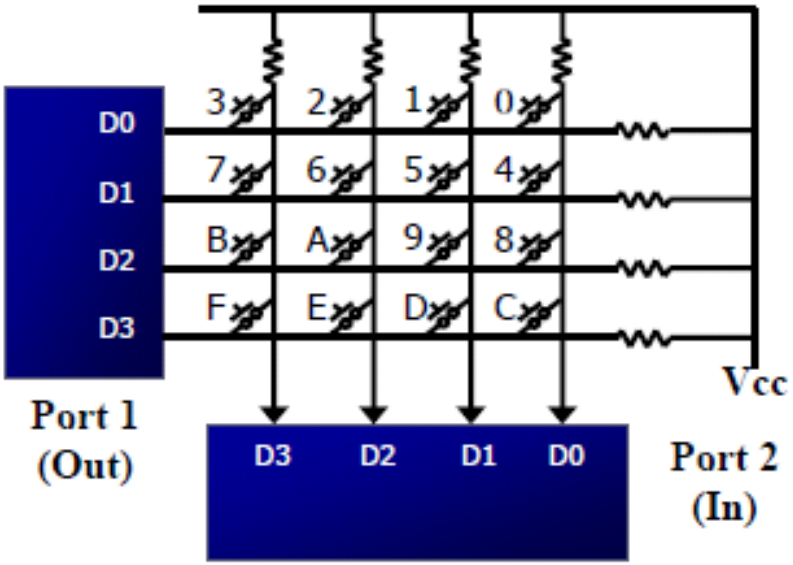
(b) D3 - D0 = 1101 for the row, D3 - D0 = 0111 for the column

Solution:

From Figure 12-6 the row and column can be used to identify the key.

(a) The row belongs to D0 and the column belongs to D2; therefore, key number 2 was pressed.

(b) The row belongs to D1 and the column belongs to D3; therefore, key number 7 was pressed.



Grounding Rows and Reading Columns (cont.)

- Detection and identification of key activation goes through the following:
  - 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

Grounding Rows and Reading Columns (cont.)

- 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 latches connected to rows still have their initial zeros (provided in stage 1), making them grounded
  - After the key press detection, it waits 20 ms for the bounce and then scans the columns again
    - It ensures that the first key press detection was not an erroneous one due a spike noise
    - If after the 20-ms delay the key is still pressed, it goes back into the loop to detect a real key press

Grounding Rows and Reading Columns (cont.)

- To detect which row key press 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 cannot belong to that row
    - It grounds the next row and continues until it finds the row the key press belongs to
  - Upon finding the row that the key press belongs to, it sets up the starting address for the look-up table holding the scan codes (or ASCII) for that row

Grounding Rows and Reading Columns (cont.)

- To identify the key press, 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

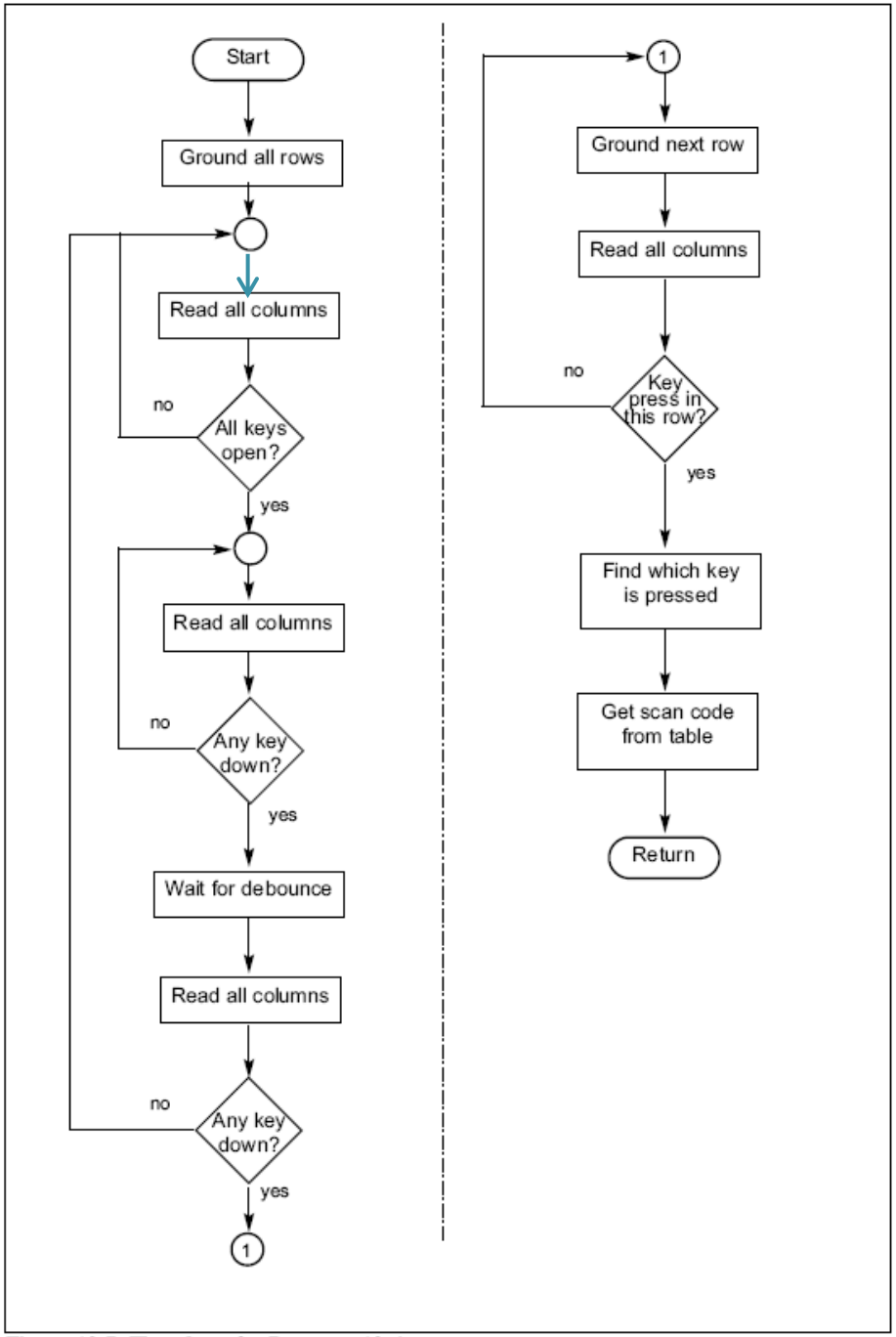


Figure 12-7. Flowchart for Program 12-4

```
;Keyboard subroutine. This program sends the ASCII code
;for pressed key to P0
;P1.0-P1.3 connected to rows P2.0-P2.3 connected to columns
        MOV   P2,#0FFH              ;make P2 an input port
K1:     MOV   P1,#0                 ;ground all rows at once
        MOV   A,P2          ;read all col. ensure all keys open
        ANL   A,#00001111B          ;masked unused bits
        CJNE  A,#00001111B,K1       ;check til all keys released
K2:     ACALL DELAY                 ;call 20 ms delay
        MOV   A,P2                  ;see if any key is pressed
        ANL   A,#00001111B          ;mask unused bits
        CJNE  A,#00001111B,OVER     ;key pressed, await closure
        SJMP  K2                    ;check if key pressed
OVER:   ACALL DELAY                 ;wait 20 ms 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 the 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 the 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 the 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 the col.
        LJMP  K2                  ;if none, false input, repeat

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 1
        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 is low
        JNC   MATCH               ;if zero, get the ASCII code
        INC   DPTR                ;point to next col. address
```

Program 12-4: Keyboard Program *(continued on next page)*

```
        SJMP  FIND                ;keep searching
MATCH:  CLR   A                   ;set A=0 (match is found)
        MOVC  A,@A+DPTR           ;get ASCII code 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
       END
```

Program 12-4. *(continued from previous page)*