

Keyboard and Display Controller

We have seen that keyboard and display devices are the two main components of microprocessor based system. Using them user can give and receive information from the microprocessor based system. In this chapter we will discuss keyboard and display interfacing in detail and study the programmable keyboard/display interface, 8279.

8.1 Keyboard Interfacing

For interfacing keyboard to the microprocessor based systems, usually push button keys are used. These push button keys when pressed, bounces a few times, closing and opening the contacts before providing a steady reading, as shown in the Fig. 8.1. Reading taken during bouncing period may be faulty. Therefore, microprocessor must wait until the key reach to a steady state; this is known as key debounce.

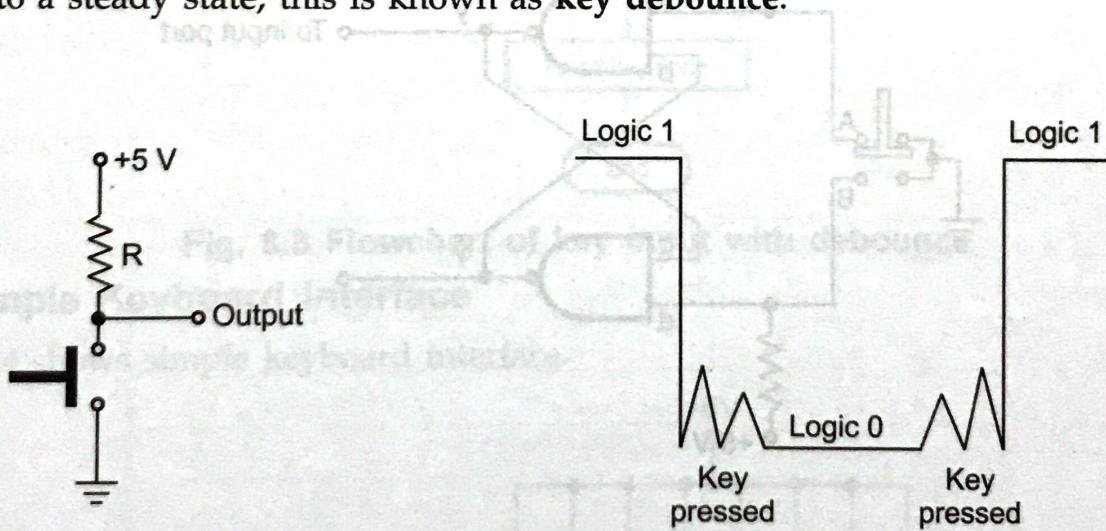


Fig. 8.1 Bouncing of key switch

The problem of key bounce can be eliminated using key debounce technique, either hardware or software.

8.1.1 Key Debounce using Hardware

Key position	a	b	y	c	d	\bar{y}
A	0	0	1	1	1	0
B	1	1	0	0	0	1
Between A and B	1	\bar{y}	No change	y	1	No change

Table 8.1

Fig. 8.2 shows the circuit diagram of key debounce. It consists of flip-flop. The output of flip-flop shown in Fig. 8.2 is logic 1 when key is at position A (unpressed) and it is logic 0 when key is at position B, as shown in table 8.1. It is important to note that, when key is in between A and B, output does not change, preventing bouncing of key output. In other words we can say that output does not change during transition period, eliminating key debouncing.

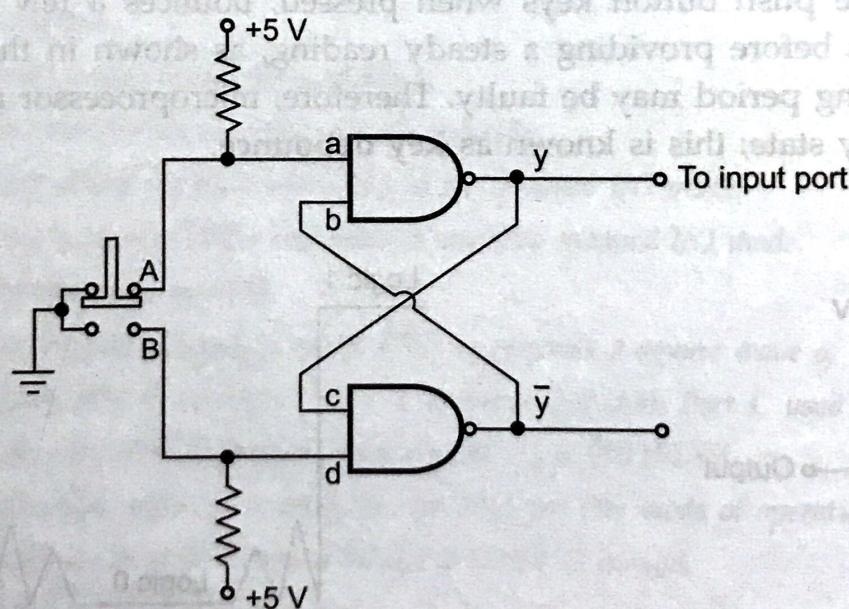


Fig. 8.2

8.1.2 Key Debouncing using Software

In the software technique, when a key press is found, the microprocessor waits for at least 10 ms before it accepts the key as an input. This 10 ms period is sufficient to settle key at steady state. Fig. 8.3 shows the flowchart with key debounce technique.

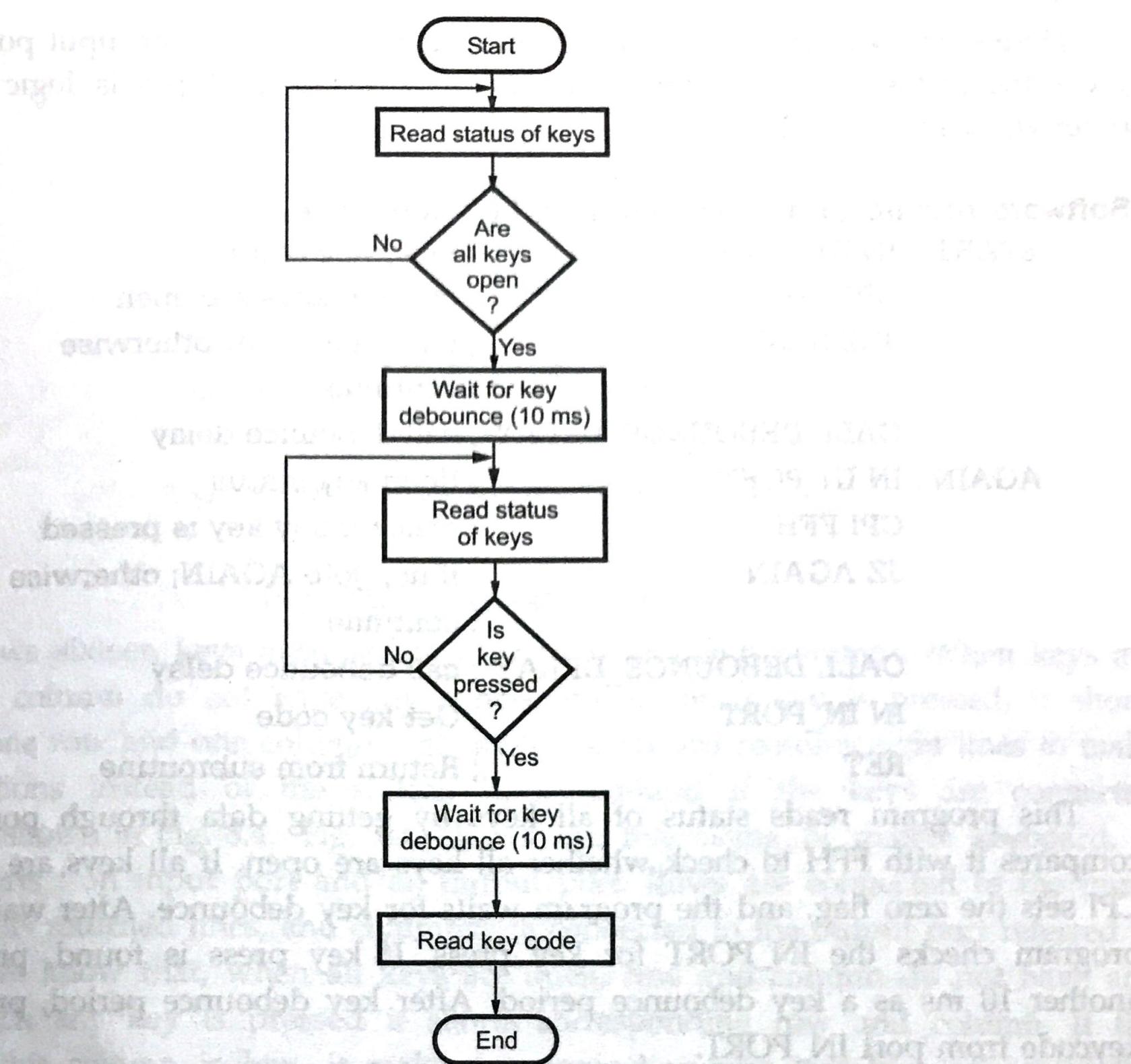


Fig. 8.3 Flowchart of key input with debounce

8.1.3 Simple Keyboard Interface

Fig. 8.4 shows simple keyboard interface.

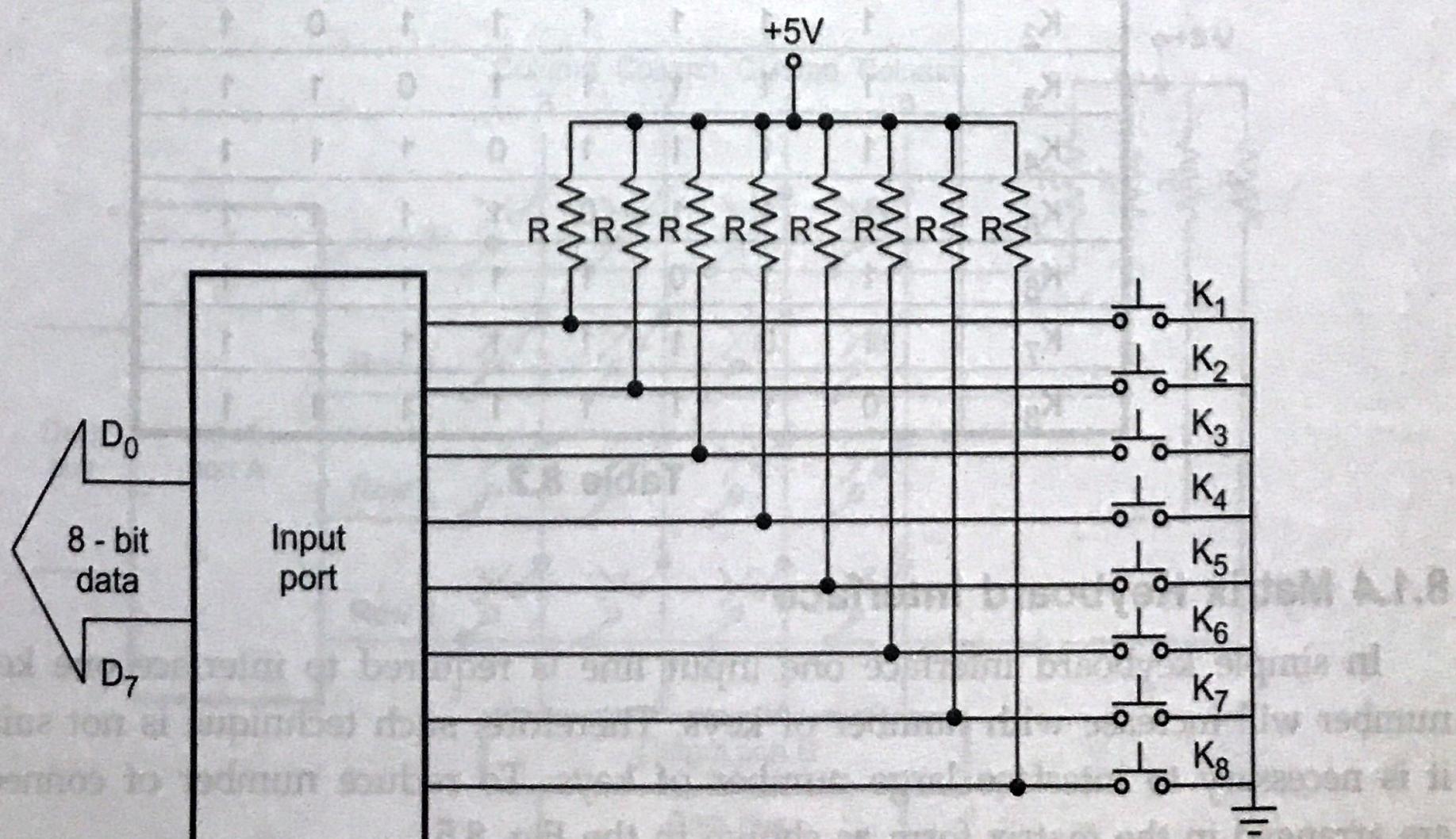


Fig. 8.4 Simple keyboard interface

Here eight keys are individually connected to specific pins of input port. Each port pin gives the status of key connected to that pin. When port pin is logic 1, key is open, otherwise key is closed.

Software routine to get keycode with key debounce.

```

START : IN IN_PORT ; Read key status
        CPI FFH ; check if keys are open
        JNZ START ; if no, goto start otherwise
                  ; continue
        CALL DEBOUNCE_DELAY ; call debounce delay
AGAIN : IN IN_PORT ; Read key status
        CPI FFH ; check if any key is pressed
        JZ AGAIN ; if no, goto AGAIN; otherwise
                  ; continue
        CALL DEBOUNCE_DELAY ; call debounce delay
        IN IN_PORT ; Get key code
        RET ; Return from subroutine
    
```

This program reads status of all keys by getting data through port IN_PORT and compares it with FFH to check whether all keys are open. If all keys are open, instruction CPI sets the zero flag, and the program waits for key debounce. After waiting about 10ms, program checks the IN_PORT for key press. If key press is found, program waits for another 10 ms as a key debounce period. After key debounce period, program reads the keycode from port IN_PORT.

Key	Keycode							
	D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
K ₁	1	1	1	1	1	1	1	0
K ₂	1	1	1	1	1	1	0	1
K ₃	1	1	1	1	1	0	1	1
K ₄	1	1	1	1	0	1	1	1
K ₅	1	1	1	0	1	1	1	1
K ₆	1	1	0	1	1	1	1	1
K ₇	1	0	1	1	1	1	1	1
K ₈	0	1	1	1	1	1	1	1

Table 8.2

8.1.4 Matrix Keyboard Interface

In simple keyboard interface one input line is required to interface one key and this number will increase with number of keys. Therefore, such technique is not suitable when it is necessary to interface large number of keys. To reduce number of connections keys are arranged in the matrix form as shown in the Fig. 8.5.

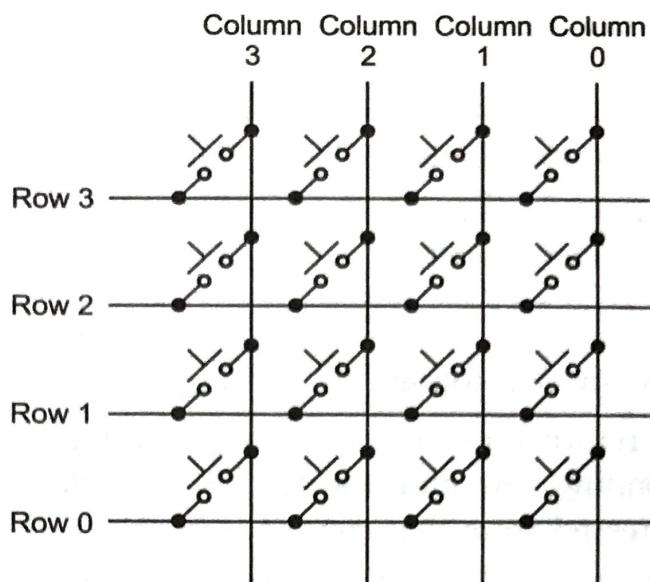
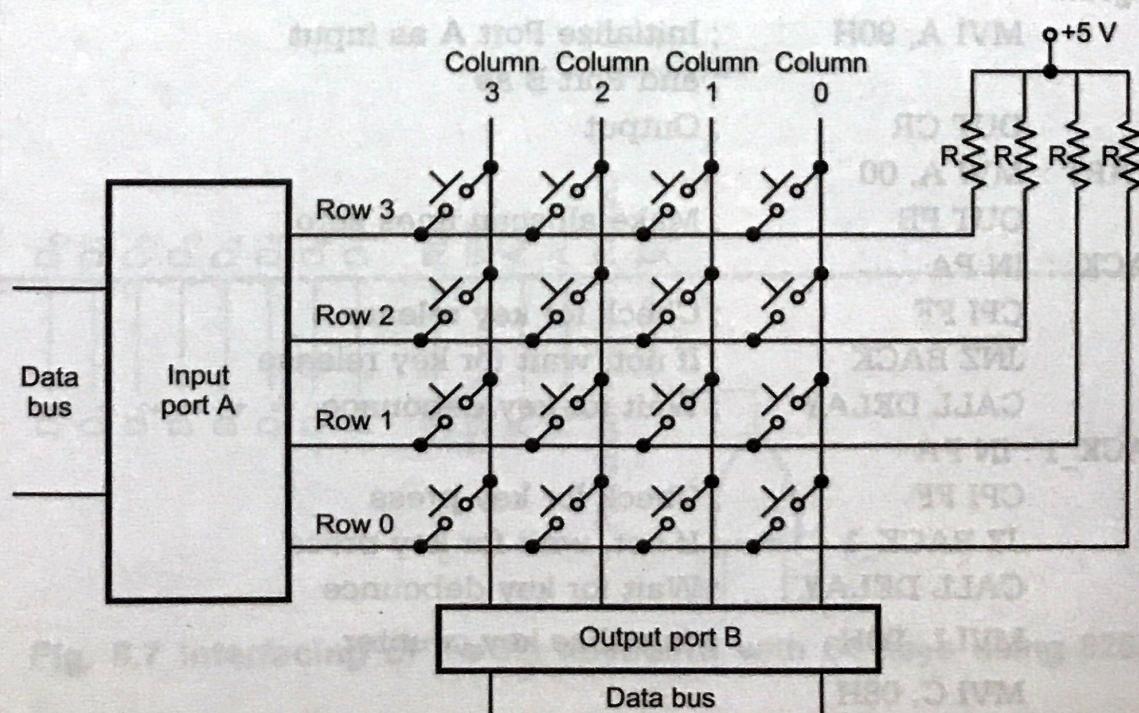
**Fig. 8.5 Matrix keyboard**

Fig. 8.5 shows sixteen keys arranged in four rows and four columns. When keys are open, row and column do not have any connection. When a key is pressed, it shorts corresponding one row and one column. This matrix keyboard requires eight lines to make all the connections instead of the sixteen lines required if the keys are connected individually, as shown in Fig. 8.4. Fig. 8.6 shows the interfacing of matrix keyboard. It requires two ports : an input port and an output port. Rows are connected to the input port referred to as returned lines, and columns are connected to the output port referred to as scan lines. We know that, when all keys are open, row and column do not have any connection. When any key is pressed it shorts corresponding row and column. If the output line of this column is low, it makes corresponding row line low; otherwise the status of row line is high. The key is identified by data sent on the output port and input code received from the input port. The following section explains the steps required to identify pressed key.

**Fig. 8.6 Matrix keyboard connections**

Check 1 : Whether any key is pressed or not

1. Make all column lines zero by sending low on all output lines. This activates all keys in the keyboard matrix. (Note : When scan lines are logic high, the status on the return lines do not change, it will remain logic high.)
2. Read the status of return lines. If the status of all lines is logic high, key is not pressed; otherwise key is pressed.

Check 2 :

1. Activate keys from any one column by making any one column line zero.
2. Read the status of return lines. The zero on any return line indicates key is pressed from the corresponding row and selected column. If the status of all lines is logic high, key is not pressed from that column.
3. Activate the keys from the next column and repeat 2 and 3 for all columns.

Lab Experiment 63 : Hardware and software for 64-key matrix keyboard interface

Statement : Interface 64-key matrix keyboard to the 8085 microprocessor using 8255. Write an 8085 assembly language program to initialize 8255 and to read the key code.

Hardware : Fig. 8.7 shows a matrix keyboard with 64 keys connected to the 8085 microprocessor using 8255. A matrix keyboard reduces the number of connections, thus the number of interfacing lines. In this example, the keyboard with 64 keys, is arranged in 8×8 (8 rows and 8 columns) matrix. This requires sixteen lines from the microprocessor to make all the connections instead of 64 lines if the keys are connected individually. The interfacing of matrix keyboard requires two ports : one input port and other output port. Rows are connected to the input port, port A and columns are connected to the output port, port B.

Source program

```

MVI A, 90H      ; Initialize Port A as input
                  ; and Port B as
OUT CR          ; Output

START : MVI A, 00
        OUT PB      ; Make all scan lines zero

BACK :  IN PA
        CPI FF      ; Check for key release
        JNZ BACK    ; If not, wait for key release
        CALL DELAY   ; Wait for key debounce

BACK_1 : IN PA
        CPI FF      ; Check for key press
        JZ BACK_1   ; If not, wait for key press
        CALL DELAY   ; Wait for key debounce
        MVI L, 00H    ; Initialize key counter
        MVI C, 08H

```

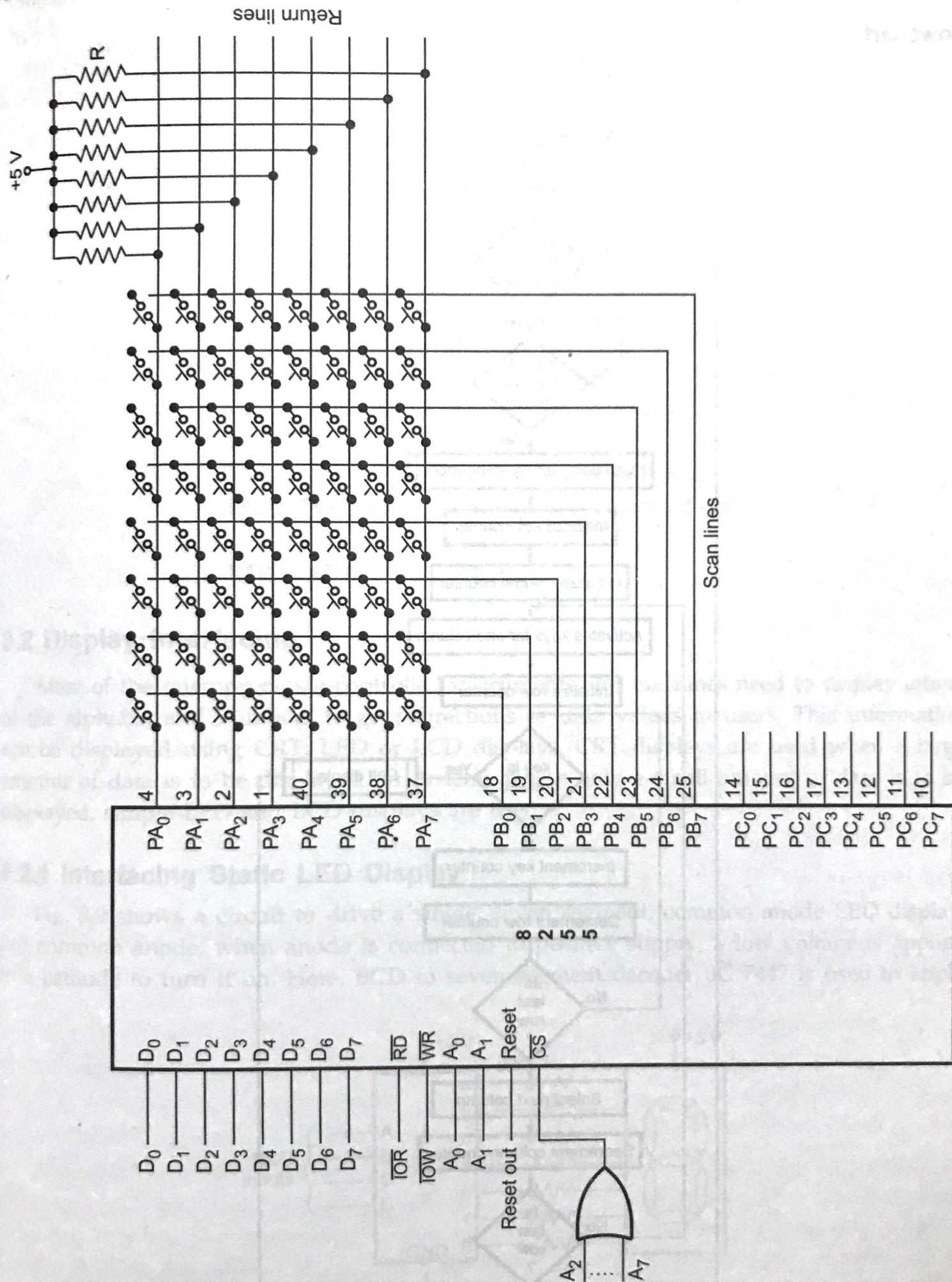
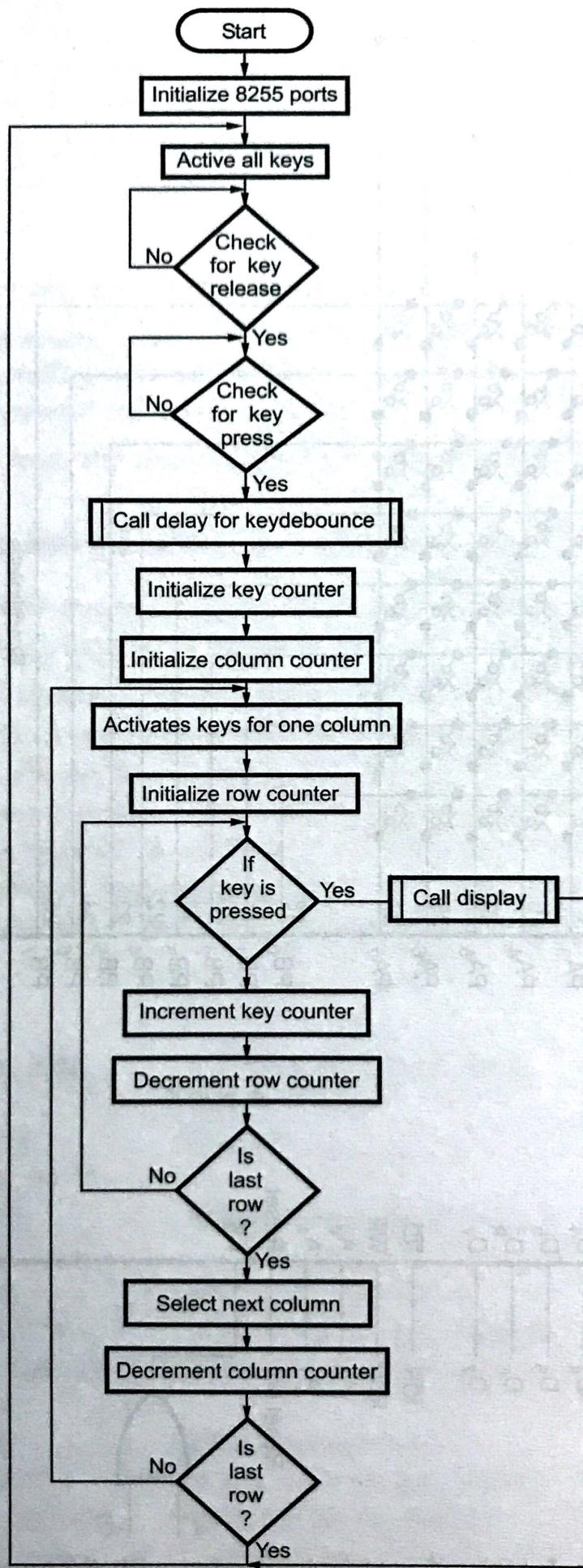


Fig. 8.7 Interfacing of matrix keyboard with 64 keys using 8255

Flowchart**Fig. 8.8 Flowchart**

MVI B, FEH	; Make one column low
NEXTCOL : MOV A, B	;
OUT PB	;
MVI D,08H	; Initialize row counter
IN PA	; Read return line status
NEXTROW : RRC	; Check for one row
JNC DISPLAY	; If zero, goto display ; otherwise continue
INR L	; Increment key counter
DCR D	; Decrement row counter
JNZ NEXTROW	; Check for next row
MOV A,B	;
RLC	; Select the next column
MOV B,A	;
DCR C	; Decrement column count
JNZ NEXTCOL	; Check for last column if not ; repeat
JMP START	; Goto start

8.2 Display Interfacing

Most of the microprocessor-controlled instruments and machines need to display letters of the alphabet and numbers to give directions or data values to users. This information can be displayed using CRT, LED or LCD displays. CRT displays are used when a large amount of data is to be displayed. In systems where only a small amount of data is to be displayed, simple LED and LCD displays are used.

8.2.1 Interfacing Static LED Display

Fig. 8.9 shows a circuit to drive a single, seven segment, common anode LED display. For common anode, when anode is connected to positive supply, a low voltage is applied to a cathode to turn it on. Here, BCD to seven segment decoder, IC 7447 is used to apply

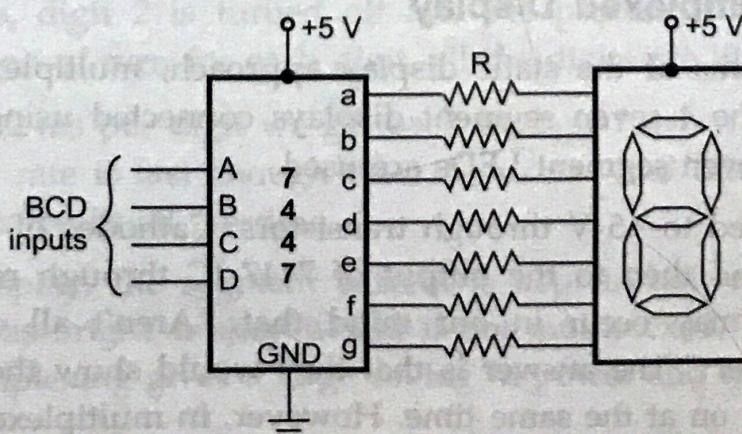


Fig. 8.9 Circuit for driving single seven segment LED display

low voltages at cathodes according to BCD input applied to 7447. To limit the current through LED segments resistors are connected in series with the segments. This circuit connection is referred to as a static display because current is being passed through the display at all times.

The value of the resistor in series with the segment can be calculated as follows :

We know, V_{CC} – Drop across LED segment – $IR = 0$

Drop across LED segment is nearly 1.5 V.

$$\begin{aligned}\therefore IR &= V_{CC} - 1.5 \text{ V} \\ &= 5 - 1.5 \text{ V} \\ &= 3.5 \text{ V}\end{aligned}$$

Each LED segment requires a current of between 5 and 30 mA to light. Let's assume that current through LED segment is 15 mA

$$\begin{aligned}\therefore R &= \frac{3.5 \text{ V}}{15 \text{ mA}} \\ &= 233 \Omega\end{aligned}$$

In practice, the voltage drop across the LED and the output of 7447 are not exactly predictable and the exact current through the LED is not critical as long as we don't exceed its maximum current rating. Therefore, a standard value 220Ω can be used.

The static display circuits work well for driving just one or two LED digits. However, these circuits are not suitable for driving more LED digits, say 8 digits. When there are more number of digits, the first problem is power consumption. For worst-case calculations, assume that all eight digits with all segments are lit. Therefore, worst case current required is

$$\begin{aligned}I &= 8 \text{ (digits)} \times 7 \text{ (segment)} \\ &\quad \times 15 \text{ mA (current per segment)} \\ &= 840 \text{ mA}\end{aligned}$$

A second problem of the static approach is that each display digit requires a separate BCD to 7 segment decoder.

8.2.2 Interfacing Multiplexed Display

To solve the problems of the static display approach, multiplexed display method is used. Fig. 8.10 shows the 4 seven segment displays connected using multiplexed method. Here, common anode seven segment LEDs are used.

Anodes are connected to +5 V through transistors. Cathodes of all seven segments are connected in parallel and then to the output of 7447 IC through resistors. Looking at Fig. 8.10, the question may occur in our mind that, "Aren't all of the digits going to display the same number?" The answer is that they would show the same number only if all the digits are turned on at the same time. However, in multiplexed display the segment information is sent for all digits on the common lines (output lines of 7447), but only one

display digit is turned on at a time. The PNP transistors connected in series with the common anode of each digit act as an ON and OFF switch for that digit. Here's how the multiplexing process works.

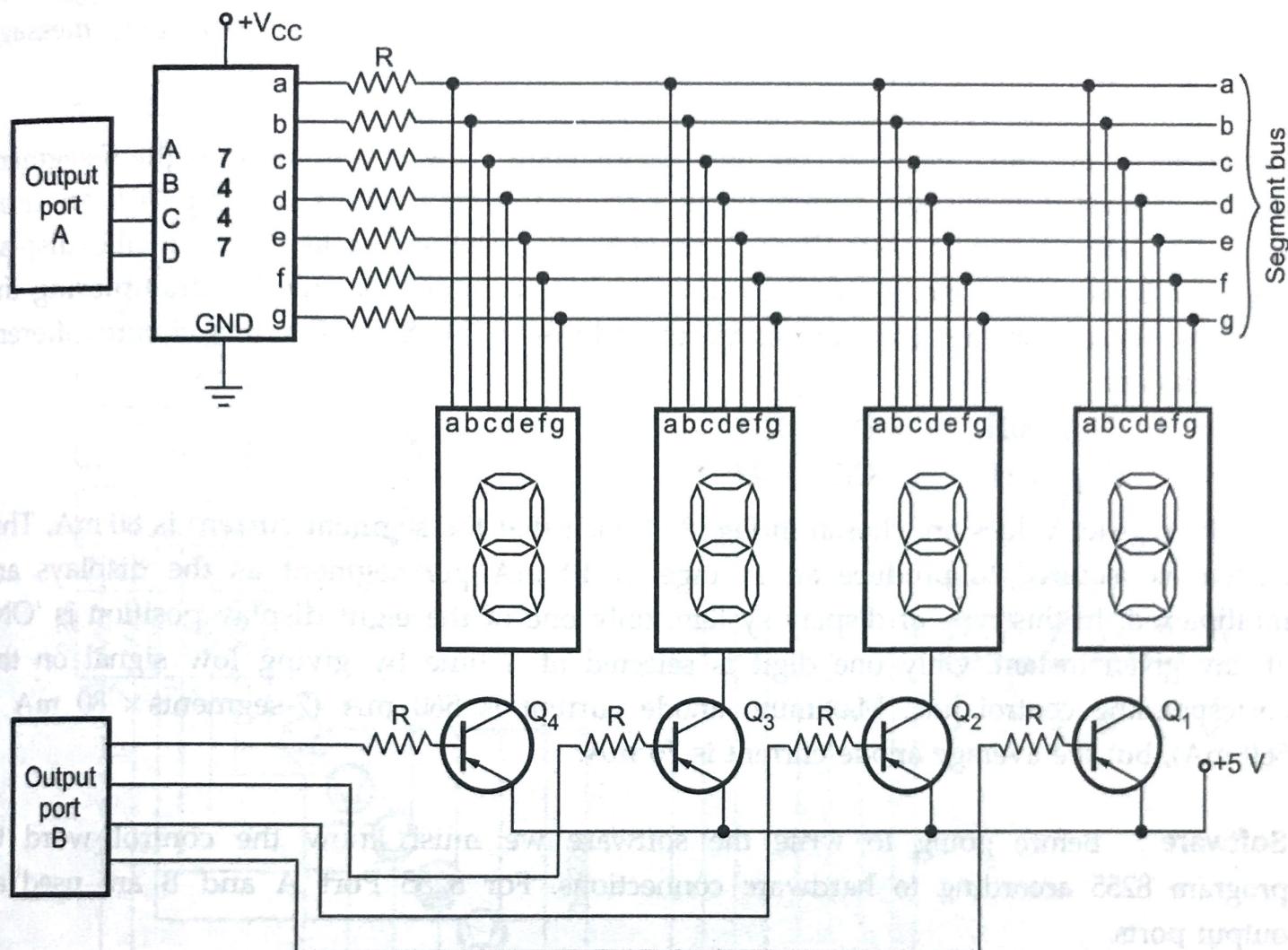


Fig. 8.10 Seven segment display in multiplexed connection

The BCD code for digit 1 is first output from port A, to the 7447. The 7447, BCD to seven segment decoder outputs the corresponding seven segment code on the segment bus lines. The transistor Q₁ connected to digit 1 is then turned on by outputting a low to that bit of port B. All of the rest of the bits of port B are made high to ensure no other digits are turned on. After 2 ms, digit 1 is turned OFF outputting all highs to port B. The BCD code for digit 2 is then output to the port A, and bit pattern to turn on digit 2 is output on port B. After 2 ms, digit 2 is turned off and the process is repeated for digit 3 and digit 4. After completion of turn for each digit, all the digits are lit again in turn.

With 4 digits and 2 ms per digit we get back to digit 1 every 8 ms or about 125 times a second. This refresh rate is fast enough that, to our eye and due to persistence of vision all digits will appear to be lit all the time.

In multiplexed display, the segment current is kept in between 40 mA to 60 mA so that they will appear as bright as they would if not multiplexed. Even with this increased segment current, multiplexing gives a large saving in power and hardware components.

Lab Experiment 64 : Hardware and software for interfacing 8-digit 7 segment display.

Statement : Interface an 8-digit 7 segment LED display using 8255 to the 8085 microprocessor system and write an 8085 assembly language routine to display message on the display.

Hardware : Fig. 8.11 (see Fig. on next page) shows the multiplexed eight 7-segment display connected in the 8085 system using 8255. In this circuit port A and port B are used as simple latched output ports. Port A provides the segment data inputs to the display and port B provides a means of selecting a display position at a time for multiplexing the displays. A_0-A_7 lines are used to decode the addresses for 8255. For this circuit different addresses are :

$$\begin{array}{ll} PA = 00H & PC = 02H \\ PB = 01H & CR = 03H. \end{array}$$

The register values are chosen in Fig. 8.11 such that the segment current is 80 mA. This current is required to produce an average of 10 mA per segment as the displays are multiplexed. In this type of display system, only one of the eight display position is 'ON' at any given instant. Only one digit is selected at a time by giving low signal on the corresponding control line. Maximum anode current is 560 mA ($7\text{-segments} \times 80\text{ mA} = 560\text{ mA}$), but the average anode current is 70 mA.

Software : Before going to write the software we must know the control word to program 8255 according to hardware connections. For 8255 Port A and B are used as output ports.

BSR	Mode A		P_A	P_{CU}	Mode B	P_B	P_{CL}
1	0	0	0	X	0	0	X

Fig. 8.12 Control word format for 8255

; Software to initialize 8255

MVI A, 80H ; Load control word in AL

OUT CR ; Load control word in CR

; Subroutine to display message on multiplexed LED display

; set up registers for display

MVI B, 08H ; load count

MVI C, 7FH ; load select pattern

LXI H, 6000H ; starting address of message

; display message

DISP_1 : MOV A, C ; select digit

OUT PB

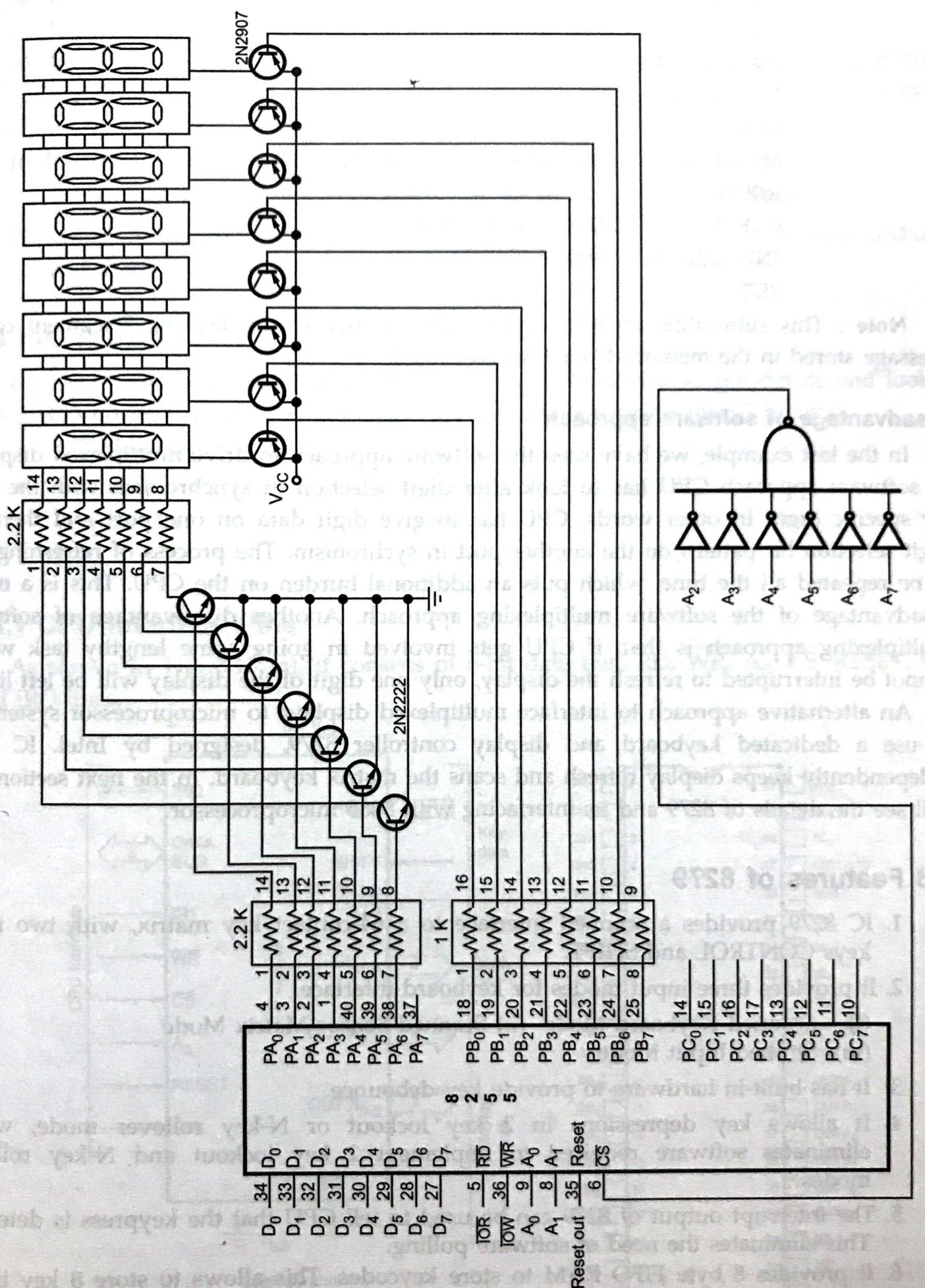


Fig. 8.11 Interfacing of multiplexed eight 7-segment display using 8255

```

MOV A, M      ; get data
OUT PA        ; display data
CALL DELAY   ; wait for some time
MOV A, C
RRC
MOV C, A      ; adjust selection pattern
INX H
DCR B         ; Decrement count
JNZ DISP_1    ; repeat 8 times
RET

```

Note : This subroutine must be called continuously to display the 7-segment coded message stored in the memory from address 6000H.

Disadvantage of software approach

In the last example, we have seen the software approach to drive multiplexed displays. In software approach CPU has to look after digit selection in synchronism with the data for specific digit. In other words, CPU has to give digit data on one port and then the digit selection bit pattern on the another port in synchronism. The process of refreshing has to be repeated all the time, which puts an additional burden on the CPU. This is a major disadvantage of the software multiplexing approach. Another disadvantage of software multiplexing approach is that, if CPU gets involved in going some lengthy task which cannot be interrupted to refresh the display, only one digit of the display will be left lit.

An alternative approach to interface multiplexed displays to microprocessor systems is to use a dedicated keyboard and display controller 8279, designed by Intel. IC 8279 independently keeps display refresh and scans the matrix keyboard. In the next section, we will see the details of 8279 and its interfacing with 8085 microprocessor.

8.3 Features of 8279

1. IC 8279 provides a scanned interface to a 64-contact key matrix, with two more keys CONTROL and SHIFT.
2. It provides three input modes for keyboard interface.
 - (i) Scanned Keyboard Mode (ii) Scanned Sensor Matrix Mode
 - (iii) Strobed Input Mode
3. It has built-in hardware to provide key debounce.
4. It allows key depressions in 2 key lockout or N-key rollover mode, which eliminates software required to implement 2 key lockout and N-key rollover modes.
5. The interrupt output of 8279 can be used to tell CPU that the keypress is detected. This eliminates the need of software polling.
6. It provides 8 byte FIFO RAM to store keycodes. This allows to store 8 key board inputs when CPU is busy in performing his own computation.
7. It provides multiplexed display interface with blanking and inhibit options.

8. It provides sixteen byte display RAM to store display codes for 16 digits, allowing to interface 16 digits.
9. In autoincrement mode, address of display RAM and FIFO RAM is incremented automatically which eliminates extra command after each read/write operation to access successive locations of display RAM and FIFO RAM.
10. It provides two output modes for display interface.
 - (i) Left Entry (typewriter type)
 - (ii) Right Entry (calculator type)
11. Simultaneous keyboard and display operation facility allows to interleave keyboard and display software.

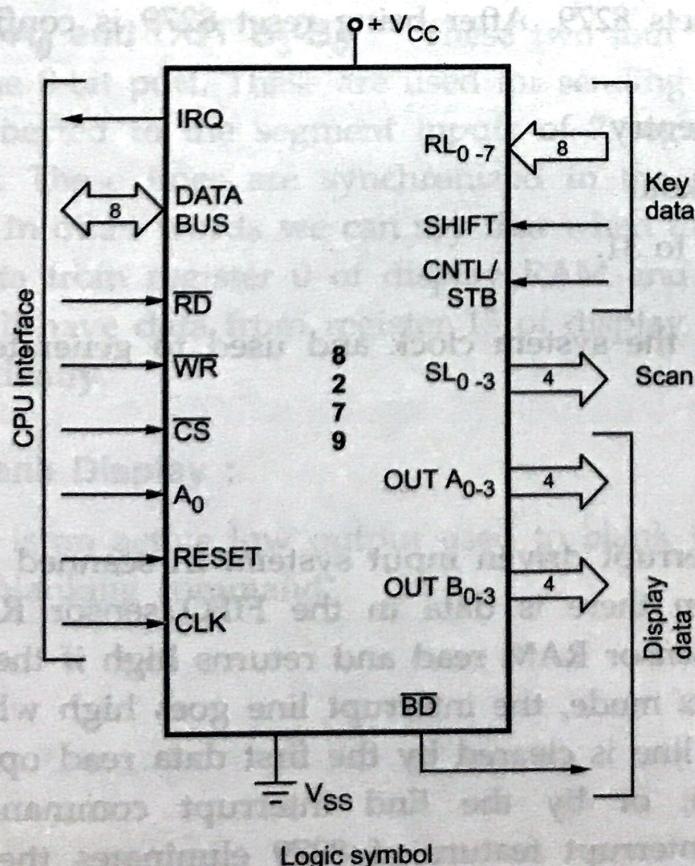
8.4 Pin Description

Fig. 8.13 shows functional and pin diagram of 8279. It is a 40 pin device and looking at Fig. 8.13 (a) we can see that these pins are divided in four functional groups :

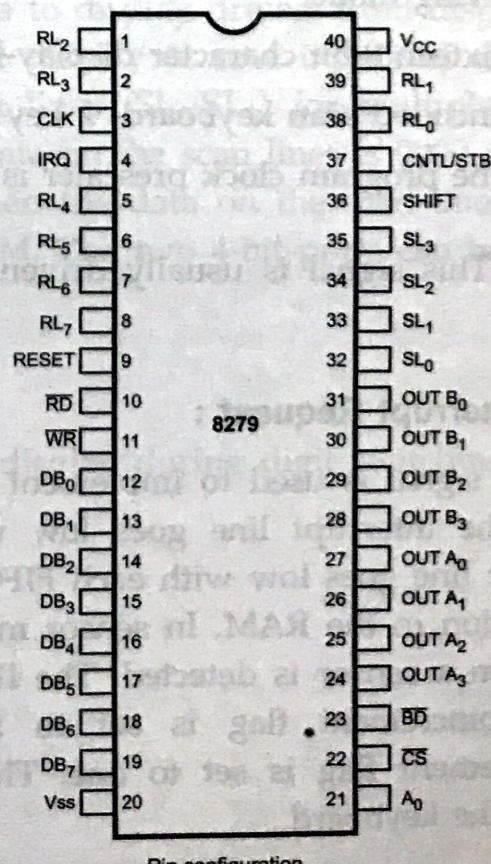
- CPU interface
- Key data
- Display data
- Scan

8.4.1 CPU Interface Pins

As shown in Fig. 8.13 (a), it consists of 8-bit data bus, \overline{RD} , \overline{WR} , A_0 , \overline{CS} , RESET, CLK and IRQ lines.



a) Functional diagram



b) Pin diagram

DB₀-DB₇ : Bi-directional data bus :

All data, commands and status information between the CPU and the 8279 are transmitted on these bi-directional 8-bit data bus.

RD : Read

It is an active low signal. When RD signal is low CPU reads the contents of selected register (display RAM, status register or FIFO RAM) from 8279; depending on the type of command and the status of the A₀ signal.

WR : Write

It is an active low signal. When WR signal is low, CPU loads the data into selected register (control register or display register) depending on the status of A₀ signal.

A₀ : Address line

When A₀ is high, signals are interpreted as a command or status. When A₀ is low signals are interpreted as a data.

CS : Chip select :

It is an active low signal. When low, enables the communication between CPU and 8279.

RESET : A high signal on this pin resets 8279. After being reset 8279 is configured in the following mode.

1. Sixteen 8-bit character display-left entry
2. Encoded scan keyboard- 2 key lockout
3. The program clock prescaler is set to 31.

CLK : This signal is usually driven by the system clock and used to generate internal timings.

IRQ : Interrupt Request :

This signal is used to implement interrupt driven input system. In scanned keyboard mode, the interrupt line goes low when there is data in the FIFO/sensor RAM. The interrupt line goes low with each FIFO/sensor RAM read and returns high if there is still information in the RAM. In sensor matrix mode, the interrupt line goes high whenever a change in a sensor is detected. The IRQ line is cleared by the first data read operation if the autoincrement flag is set to zero, or by the End interrupt command if the autoincrement flag is set to one. The interrupt feature of 8279 eliminates the need of polling the keyboard.

8.4.2 Keyboard Data

This group consists of return, SHIFT and CNTL/STB lines.

RL₀-RL₇ : Return lines : These input lines are used to interface matrix keyboard. These lines have active internal pullups which keep their status high. When the key from the matrix keyboard is pressed corresponding return line goes low. In the strobed input mode these lines are used as 8 input lines.

SHIFT : It is a special key input line. Its status is stored along with the key position on the key closure in the scanned keyboard modes. It has an active internal pullup to keep it high until a switch closure pulls it low.

CNTL/STB : Control/strobe

For scanned keyboard mode this line is used as a control input. Like SHIFT key, its status is stored along with the key position on the key closure. It also has an active internal pullup to keep it high until a switch closure pulls it low.

In the strobed input mode this line is used as a strobe input. When activated, loads the status of keyboard into the FIFO RAM.

8.4.3 Display Data

This group consists of OUT A₃-A₀, OUT B₃-B₀ and \overline{BD} lines.

OUT A₃-A₀ and OUT B₃-B₀ : These two four bit output ports, which can be considered as an one 8-bit port. These are used for sending data to display drivers from display RAM and connected to the segment inputs of 7 segment display or row inputs of dot matrix displays. These lines are synchronized to the scan lines (SL₀-SL₃) for multiplexed digit display. In other words we can say that when the data on the scan lines is 0000, ports will have data from register 0 of display RAM and when the data on the scan lines is 1111, ports will have data from register 15 of display RAM. The two 4-bit ports can be blanked independently.

\overline{BD} : Blank Display :

This is an active low output used to blank the display during digit switching or by a display blanking command.

8.5 Block Diagram

Fig. 8.14 shows the block diagram of 8279. It consists of four main sections :

- CPU interface and control section
- Scan section
- Keyboard section
- Display section

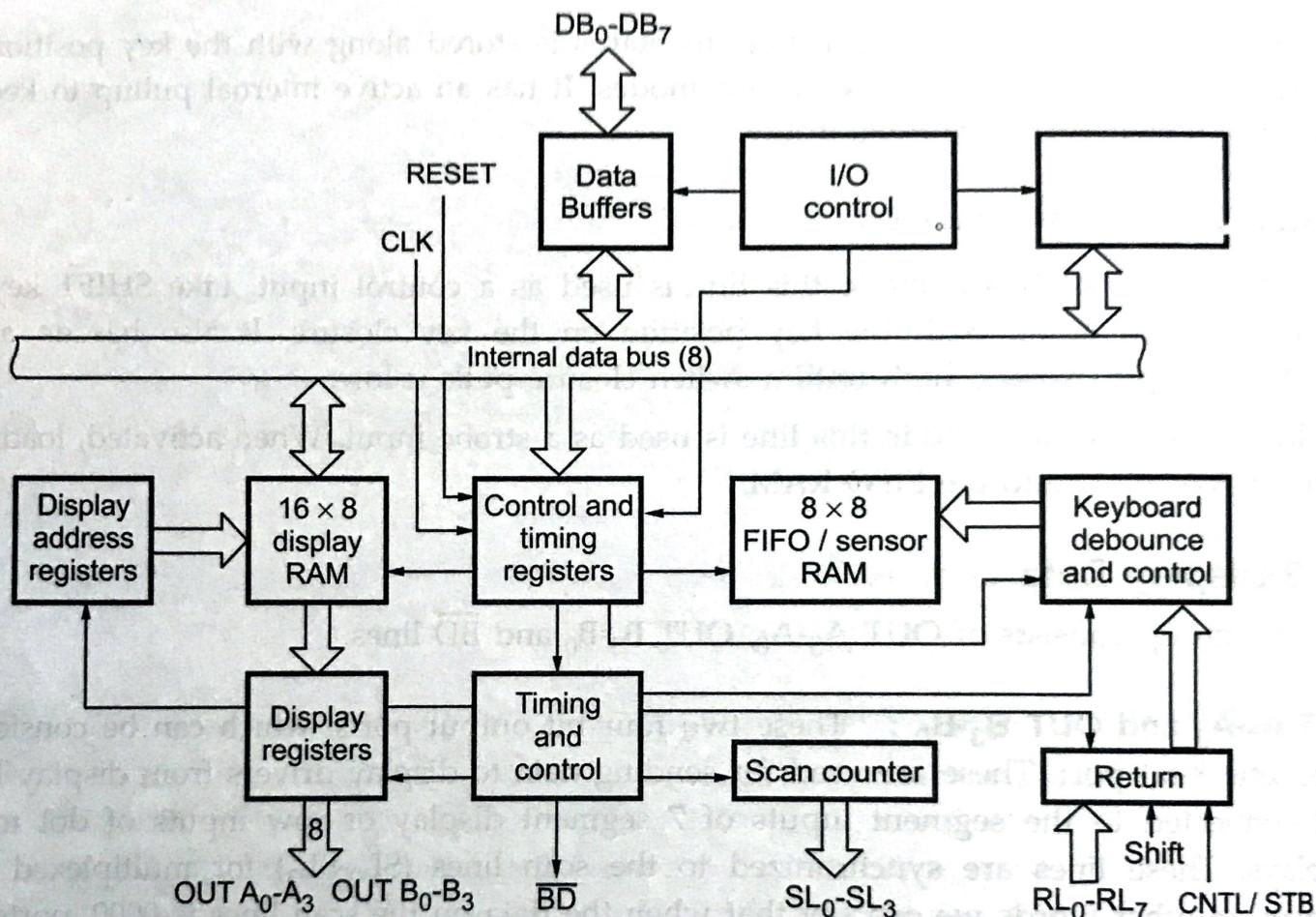


Fig. 8.14

8.5.1 CPU Interface and Control Section

This section consists of data buffers, I/O control, control and timing registers, and timing and control logic.

Data Buffers

The data buffers are 8-bit bi-directional buffers that connect the internal data bus to the external data bus.

I/O Control

The I/O control section uses the A₀, CS, RD and WR signals to control data flow to and from the various internal registers and buffers. The data flow to and from the 8279 is enabled only when CS = 0; otherwise the 8279 signals are in a high impedance state. The

8279 interprets the data given or desired by the CPU with the help of A_0 , \overline{RD} and \overline{WR} signals, as shown in table 8.3. When A_0 is logic 0 data is transferred and when A_0 is logic 1 command word or status word is transferred. \overline{RD} and \overline{WR} determine the direction of data flow through the data buffers.

A_0	\overline{RD}	\overline{WR}	Interpretation
0	1	0	Data from CPU to 8279
0	0	1	Data to CPU from 8279
1	1	0	Command word from CPU to 8279
1	0	1	Status word to CPU from 8279

Table 8.3

Control and Timing Registers

The control and timing registers store the keyboard and display modes and other operating conditions programmed by the CPU. The modes are programmed by sending the proper command on the data lines with $A_0 = 1$. The command is latched on the rising edge of \overline{WR} . The command is then decoded and the appropriate mode/function is set.

Timing Control

The timing control consists of the basic timing counter chain. The first counter is divided by N prescaler that can be programmed to give an internal frequency of 100 kHz. The other counters divide down the basic internal frequency to provide the proper keyscan, row scan, keyboard matrix scan, and display scan times. The internal frequency of 100 kHz gives the internal timings as shown in the table 8.4.

Parameter	Timings
Keyboard scan time	5.1 msec
Keyboard debounce time	10.3 msec
Key scan time	80 μ sec
Display scan time	10.3 msec
Digit ON time	480 μ sec
Blanking time	160 μ sec
Internal clock cycle	10 μ sec

Table 8.4 Internal timings of 8279

8.5.2 Scan Section (Scan Counter)

The scan section has a scan counter which has two modes : Encoded mode and decoded mode.

Encoded Mode

In the encoded mode, the scan counter provides a binary count from 0000 to 1111 on the four scan lines ($SC_3 - SC_0$) with active high outputs. This binary count must be externally decoded to provide 16 scan lines.

Display can use all 16 scan lines to interface 16 digit 7-segment display, but keyboard can use only 8 scan lines out of 16 scan lines.

Decoded Mode

In the decoded mode, the internal decoder decodes the least significant 2 bits of binary count and provides four possible combinations on the scan lines ($SC_3 - SC_0$) : 1110, 1101, 1011 and 0111. Thus the output of decoded scan is active low. These four active low output lines can be used directly to interface 4 digit 7 segment display, 8×4 matrix keyboard, eliminating the external decoder.

8.5.3 Keyboard Section

This section consists of return buffers, keyboard debounce and control, FIFO/sensor RAM and FIFO/sensor RAM status. These functions depend on selected keyboard mode out of three keyboard input modes : scanned keyboard, sensor matrix and strobed input.

Return buffers

The 8 return lines ($RL_7 - RL_0$) are buffered and latched by the return buffers during each row scan in scanned keyboard or sensor matrix mode. In strobed input mode, the contents of the return lines are transferred to the FIFO RAM on the rising edge of the CNTL/STB line pulse.

Keyboard debounce and control

Keyboard and debounce control is enabled only when scanned keyboard mode is selected. In the scanned keyboard mode, return lines are scanned, looking for key closures in that row. If the debounce circuit detects a close switch, it waits about 10 msec to check if the switch remains closed. If it does, the address of the switch in the matrix plus the status of SHIFT and CONTROL keys are transferred to the FIFO RAM.

FIFO/Sensor RAM

This is a dual function 8×8 RAM. In scanned keyboard and strobed input modes, it is a FIFO. Each new entry is written into successive RAM positions and then read in order of entry. In sensor matrix mode, the memory is referred to as sensor RAM. Each row of the

sensor RAM is loaded with the status of the corresponding row of sensor in the sensor matrix.

FIFO/sensor RAM status

FIFO RAM status keeps track of the number of characters in the FIFO and whether it is full or empty. The status logic also makes IRQ signal high when the FIFO is not empty, which can be used to interrupt CPU telling that key press is detected and keycode is available in FIFO RAM.

8.5.4 Display Section

The display section consists of display RAM, display address registers and display registers.

Display RAM

It is 16×8 RAM, which stores the display codes for 16 digits. It can be accessed directly by CPU. In decoded mode, 8279 uses only first four locations of display RAM. In encoded mode, 8279 uses first eight locations for 8 digit display and all 16 locations for 16 digits display.

Display address registers

The display address registers hold the address of the byte currently being written or read by the CPU and scan count value. The read/write addresses are programmed by CPU command. If set in autoincrement mode, address in the address register is incremented for each read or write.

Display registers

Display registers are two 4-bit registers A and B. They hold the bit pattern of character to be displayed. The contents of display registers A and B can be blanked and inhibited individually.

8.6 Operating Modes

8.6.1 Input Modes

The 8279 provides 3 basic input modes :

- Scanned keyboard
- Scanned sensor matrix
- Strobed input