The mechanical relays or contactors, however, have several serious problems. When the contacts are opened and closed, arcing takes place between the contact, which causes the contacts to oxidize and pit. As the contacts are oxidized, they become higher resistance contant and may get hot enough to melt. Another disadvantage of mechanical relays is that when they switch ON or OFF at high-voltage point, they produce large amount of electrical noise, called **electromagnetic interference (EMI)**.

9.14.3.2 Solid State Relays

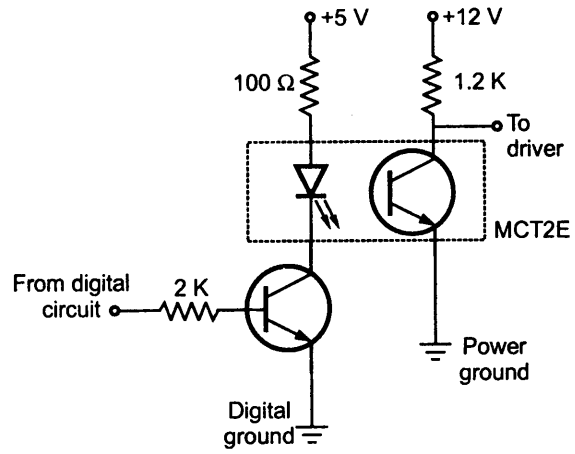


Fig. 9.33 Opto-isolator circuitry

To avoid the problems of mechanical relays, solid state relays are used. In this triac is used as a switching element and isolation is provided by opto isolators or pulse transformer. The Fig. 9.33 shows the typical opto isolator circuitry. It consists of LED and a photo transistor. LED glows when digital input is high, making phototransistor ON. Thus digital input controls the voltage at the collector of phototransistor without any physical connection between them, providing isolation. When digital input is low, LED and hence phototransistor is OFF.

The Fig. 9.34 shows the typical pulse transformer circuit. The pulse transformer magnetically couples the control and power circuitry avoiding electrical contact between them.

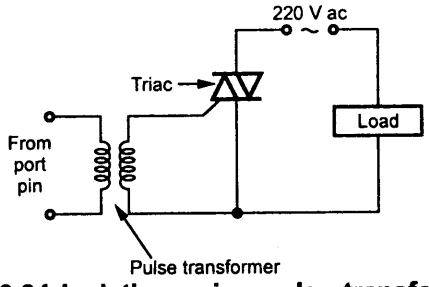


Fig. 9.34 Isolation using pulse transformer

The opto isolator circuits give better performance at relatively low switching speed. Because it has switching speed limitations. On the other hand, at high switching speeds pulse transformer provides better performance. At low switching speed pulse transformer may get saturated to detoriate its performance.

Review Questions

1. *Discuss the organization and architecture of 8255 programmable peripheral interface IC with a functional block diagram.*
2. *Illustrate the different modes of operation of 8255.*
3. *Explain the control word formation with suitable examples.*

4. *Draw the interfacing scheme of 8255 and 8086 in I/O mapped I/O mode.*
5. *Draw the interfacing scheme of 8255 and 8086 in memory mapped I/O mode.*
6. *Explain mode 1 input and output operation of 8255 with the help of timing diagram. Also give the format of BSR word in 8255.*
7. *Explain the bit set/reset mode of 8255.*
8. *Write a program in 8086 assembly language using 8255 to generate a square wave of 1 MHz frequency.*
9. *An 8255 is used with Port A as output, Port B as input and with Port C used for handshaking for Port A and Port B. Assume address of port A = 80H.*
   i) Write the instruction sequence to program the 8255 for this mode of operation.
   ii) Write the format of the status word obtained if PORT C is read.
   iii) Draw the scheme of connection required.
10. *Draw the interfacing diagram for 8086 system in minimum mode with the following specifications :*
    i) 16 kB RAM
    ii) 8 kB EPROM
    iii) 8255 PPI in I/O address space.
    Also show the required latches, buffers and decoder. Draw the memory map for the above interface.
11. *Draw the interfacing diagram for 8086 system in minimum mode with the following specifications :*
    i) 32 kB RAM
    ii) 8 kB EPROM
    iii) 8255 PPI in memory address space.
    Also show the required latches, buffers and decoder. Draw the memory map for the above.
12. *Draw the interfacing diagram between 8086 CPU and 8255 with 8 LEDs and 8 SPDT switches. Also write a program in 8086 assembly language to read the switch status and display it on the LEDs. Add proper comments in the program.*
13. *Write a short note on centronics interface.*
14. *Explain with neat diagram and flowchart interfacing of stepper motor to microprocessor.*
15. *Discuss various options available for interfacing high power devices.*
16. *Why isolation circuitry is required to control ac drives ?*
17. *Explain various circuits that provide isolation between control circuit and power circuit.*

❑❑❑

10 Keyboard and Display Interfacing

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.

10.1 Interfacing of Switches

For interfacing switches 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. 10.1. 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.**

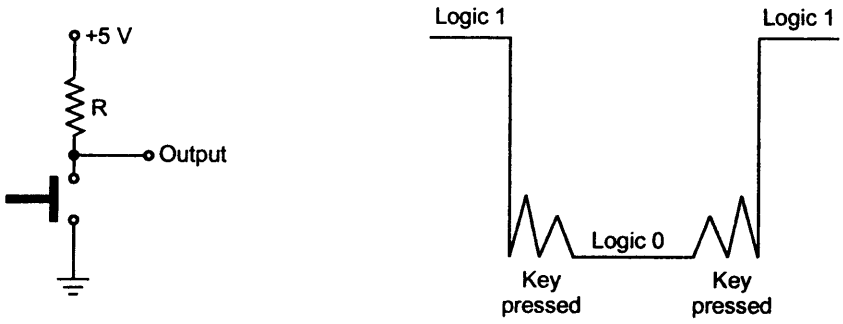


Fig. 10.1 Bouncing of key switch

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

10.1.1 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 10.1

Fig. 10.2 shows the circuit diagram of key debounce. It consists of flip-flop. The output of flip-flop shown in Fig. 10.2 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 10.1. 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.

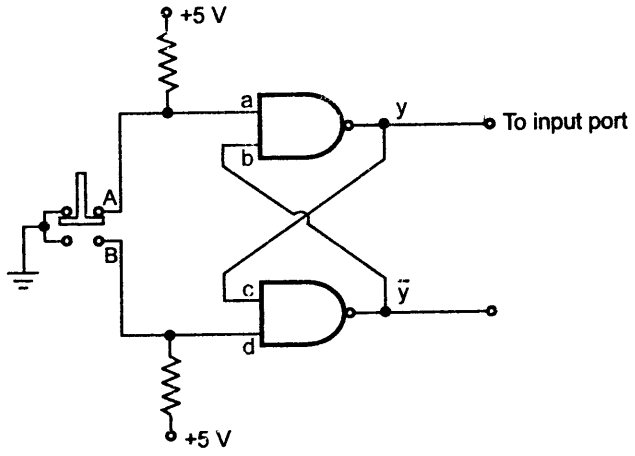


Fig. 10.2

10.1.2 Key Debouncing using Software

In the software technique, when a key press is found, the microprocessor waits for atleast 10 ms before it accepts the key as an input. This 10 ms period is sufficient to settle key at steady state. Fig. 10.3 shows the flowchart with key debounce technique.

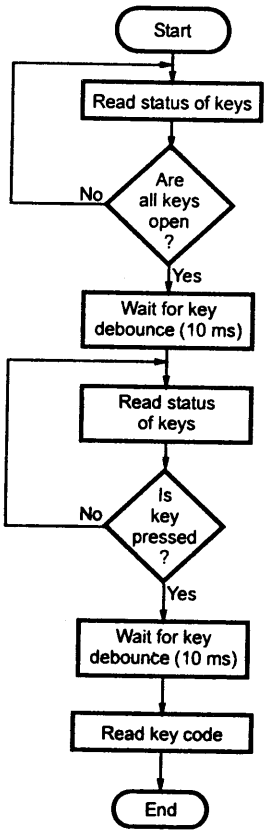


Fig. 10.3 Flowchart of key input with debounce

10.2 Simple Keyboard Interface

Fig. 10.4 shows the simple keyboard interface.

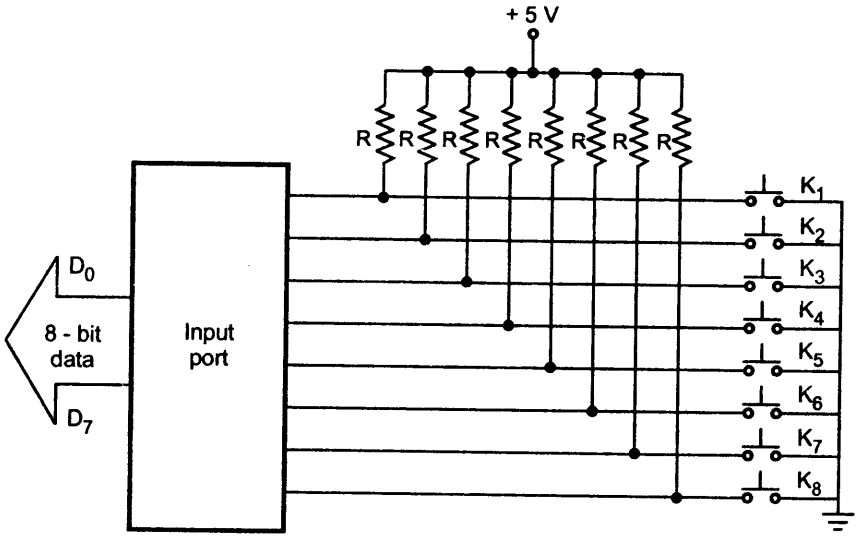


Fig. 10.4 Simple keyboard interface

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   AL,IN_PORT          ; Read key status
         CMP  AL,FFH              ; check if keys are open
         JNZ  START               ; if no, goto start otherwise
                                  ; continue
         CALL DEBOUNCE_DELAY      ; call debounce delay
AGAIN :  IN   AL, IN_PORT         ; Read key status
         CMP  AL, FFH             ; check if any key is pressed
         JZ AGAIN                 ; if no, goto AGAIN; otherwise
                                  ; continue
         CALL DEBOUNCE_DELAY      ; call debounce delay
         IN   AL, 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 CMP AL, FFH set the zero flag, and the program waits for key debounce. 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 10.2

10.3 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. 10.5.

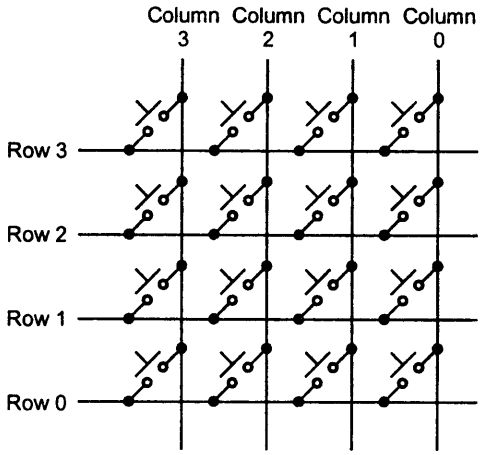


Fig. 10.5 Matrix keyboard

Fig. 10.5 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. 10.5.

Fig. 10.6 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

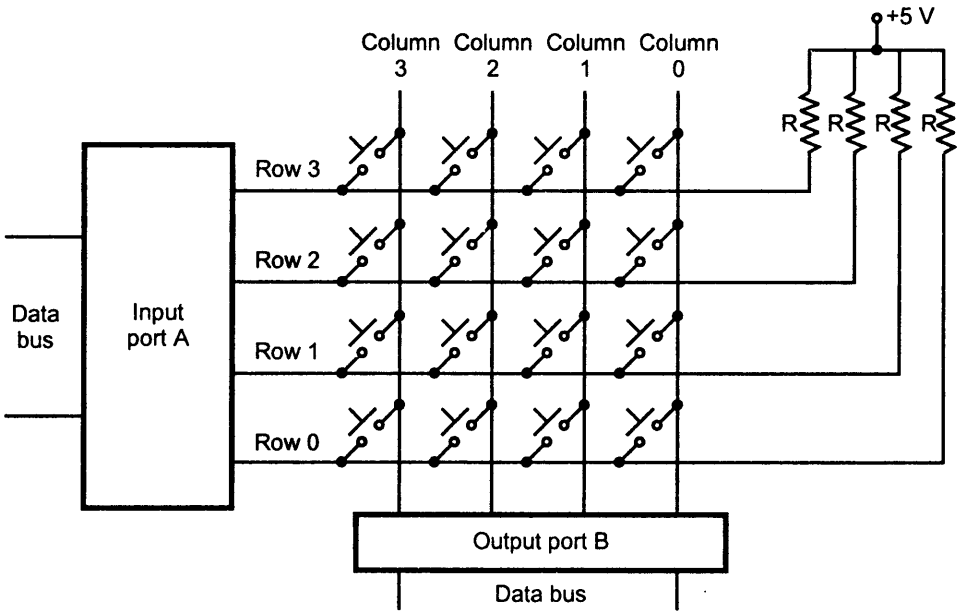


Fig. 10.6 Matrix keyboard connections

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.

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 is 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.

Example 10.1 : *Interface 4 × 4 keyboard with 8086 microprocessor.*

Solution : Hardware : Fig. 10.7 shows a matrix keyboard with 16 keys connected to the 8086 microprocessor using 8255. A matrix keyboard reduces the number of connections, thus the number of interfacing lines. In this example the keyboard with 16 keys, is arranged in 4 × 4 (4 rows and 4 columns) matrix. This requires eight lines from the microprocessor to make all the connections instead of 16 lines if the keys are connected individually. The interfacing of matrix keyboard requires two ports : one input port and one output port. Rows are connected to the Input Port (return lines) and columns are connected to the Output Port (scan lines). 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.

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.

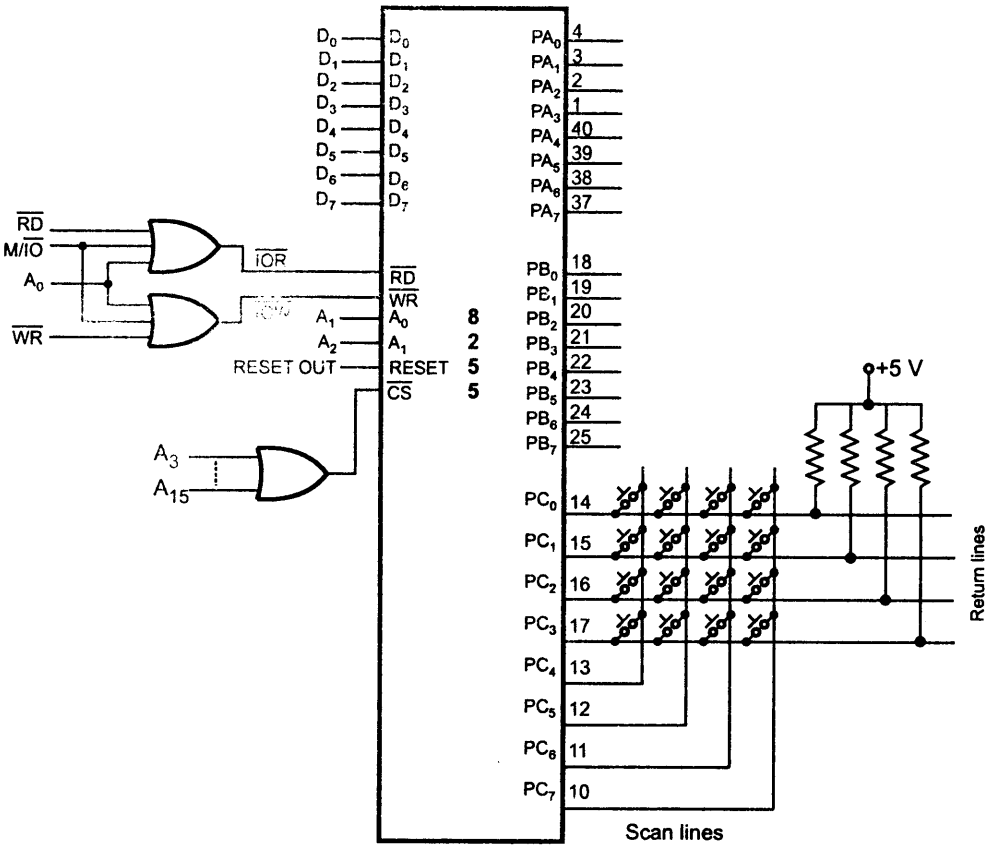


Fig. 10.7 Interfacing of 4 × 4 keyboard with 8086

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.

In Fig. 10.7 the scan lines are connected to the port C_L of 8255 and returns lines are connected to the port C_U of 8255.

Flowchart

(See flowchart on next page).

Source program

```
PORTA    EQU    0000
PORTC    EQU    0004
CR       EQU    0006
```

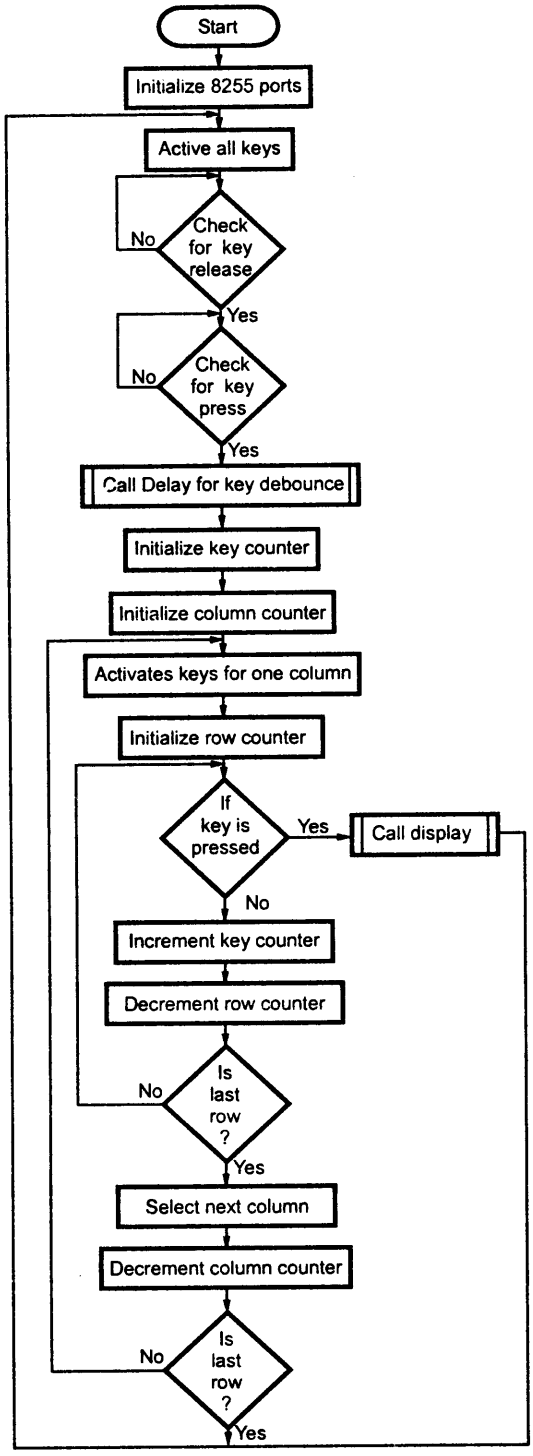


Fig. 10.8 Flowchart

```
     PROC KEY NEAR
START:   MOV AL,81H    ; Initialize Port C_L as input and Port C_U
                       ; as output
         MOV DX,CR     ; [ Initialize
         OUT DX,AL     ;     8255 ]
         MOV AL,00H
         MOV DX,PORTC
         OUT DX,AL     ; Make all scan lines zero
BACK:    IN  AL,DX
         AND AL,0FH
         CMP AL,0FH    ; Check for key release
         JNZ BACK      ; If not, wait for key release
BACK1:   IN  AL,DX
         AND AL,0FH
         CMP AL,0FH    ; Check for key press
         JZ BACK1      ; If not, wait for key press
         CALL DELAY    ; Wait for key debounce
         MOV BL,00H    ; Initialize key counter
         MOV CL,04H
         MOV BH,FEH    ; Make one column low
NEXTCOL: MOV AL,BH
         OUT DX,AL
         MOV CH,04H    ; Initialize row counter
         MOV DX,PORTA
         IN AL,DX      ; Read return line status
NEXTROW: RCR AL,1      ; Check for one Row
         JNC DISPLAY   ; If zero, goto display
                       ; otherwise continue
         INC BL        ; Increment key counter
         DEC CH        ; Decrement row counter
         JNZ NEXTROW   ; Check for next row
         MOV AL,BH
         RCL AL,1      ; Select the next column
         MOV BH,AL
         DEC C         ; Decrement column count
         JNZ NEXTCOL   ; Check for last column if not repeat
         JMP START     ; Goto start
         RET
         KEY ENDP
         END START
```

10.4 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. The following sections explains you how to interface LED displays to microprocessors.

10.4.1 LED Displays

LED displays are available in two very common formats. 7 segment displays and 5 by 7 dot-matrix displays.

Seven-Segment display

Seven segment displays are generally used as numerical indicators and consists of a number of LEDs arranged in seven segments as shown in the Fig. 10.9.

Any number between 0 and 9 can be indicated by lighting the appropriate segments.

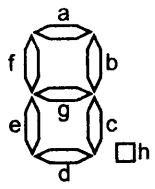


Fig. 10.9 Seven segment display

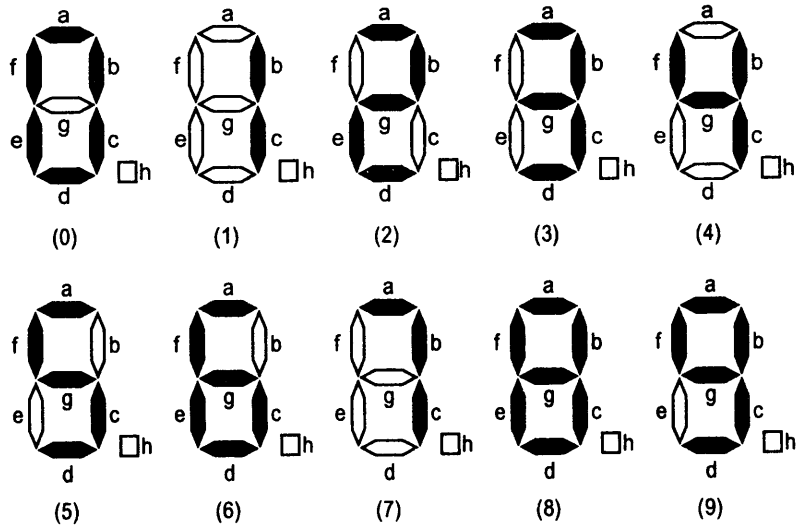


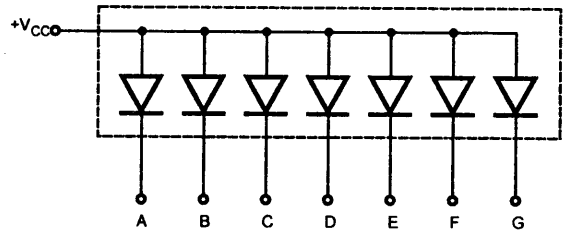
Fig. 10.10

The seven segments are labeled a to g and dot is labeled as h. By forward biasing different LED segments, we can display the digits 0 through 9. For instance, to display 0, we need to light up a, b, c, d, e, and f. To light up 5, we need segments a, f, g, c and d.

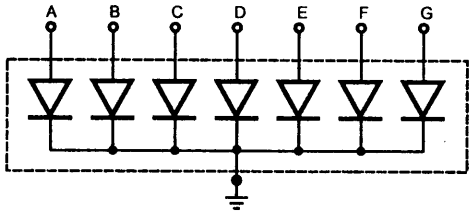
These 7 segment displays are of two types :

- Common Anode Type
- Common Cathode Type

In common anode, all anodes of LEDs are connected together as shown in Fig. 10.11 (a) and in common cathode, all cathodes are connected together, as shown in Fig. 10.11 (b).



(a) Common anode type



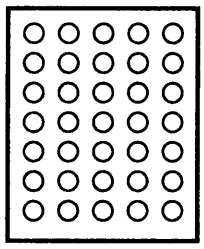
(b) Common cathode type

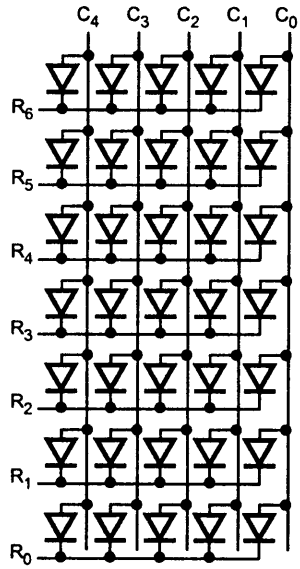
Fig. 10.11 Internal diagram of 7-segment LED

5 by 7 DOT matrix LED

Fig. 10.12 (a) and (b) show the 5 by 7 dot matrix LED display and its circuit connections.

This display can be used to display number as well as alphabets.





(a) 5 × 7 dot matrix LED display (b) 5 × 7 dot matrix circuit connections

Fig. 10.12

10.4.2 Interfacing LED Displays

Static display

Fig. 10.13 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 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.

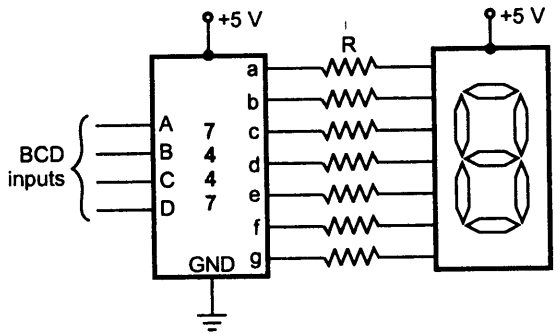


Fig. 10.13 Circuit for driving single seven segment LED display

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.

$$\therefore \quad IR = V_{CC} - 1.5\ V$$

$$= 5 - 1.5\ V$$

$$= 3.5\ V$$

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

$$\therefore \quad R = \frac{3.5\ V}{15\ mA}$$

$$= 233\ \Omega$$

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 220 Ω 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

$$I = 8 \text{ (Digits)} \times 7 \text{ (Segment)} \times 15 \text{ mA (Current per segment)}$$

$$= 840 \text{ mA}$$

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

Multiplexed display

To solve the problems of the static display approach, multiplexed display method is used. Fig. 10.14 shows the 4 seven segment displays connected using multiplexed method. Here, common anode seven segment LEDs are used.

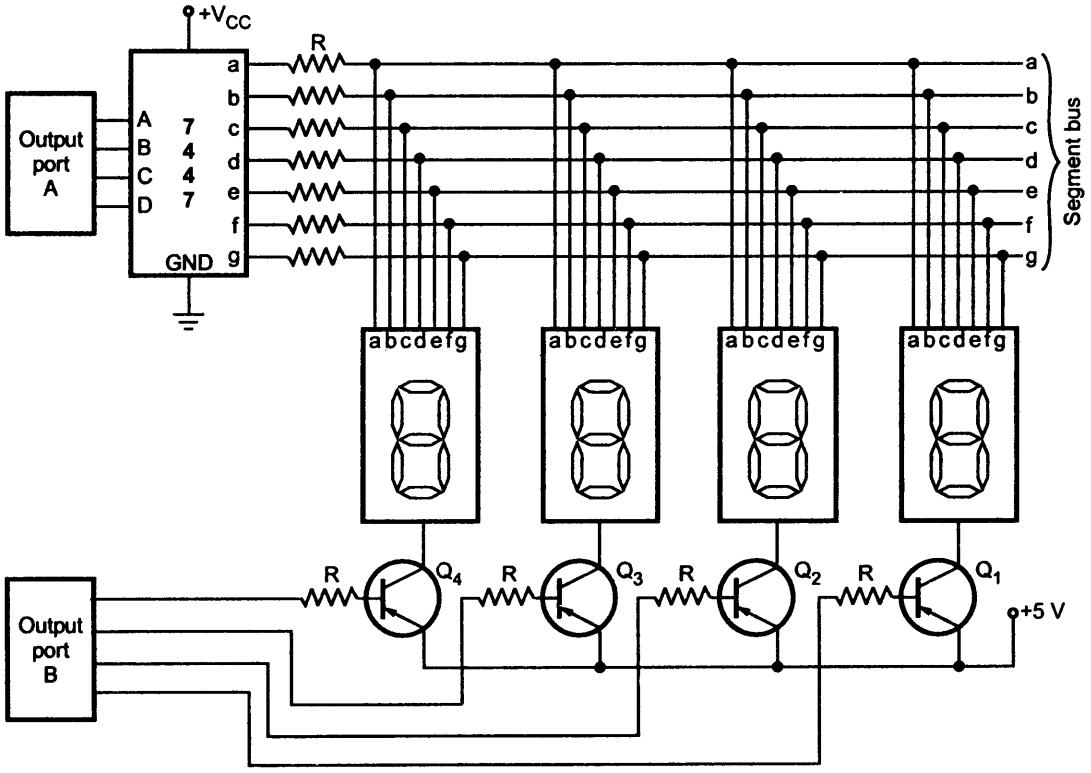


Fig. 10.14 Seven segment display in multiplexed connection

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 the

Fig. 10.14, 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.

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 Q_1 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.

Example 10.2 : *Interface an 8-digit 7 segment LED display using 8255 to the 8086 microprocessor system and write an 8086 assembly language routine to display message on the display.*

Solution : Hardware : Fig. 10.15 shows the multiplexed eight digit 7-segment display connected in the 8086 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. The 8255 is addressed using direct addressing mode, so only A_0-A_7 lines are used to decode the addresses for 8255.

For this circuit different addresses are :

PA = 00H PC = 04H

PB = 02H CR = 06H

The register values are chosen in Fig. 10.15 so 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-segments × 80 mA = 560 mA), but the average anode current is 70 mA.

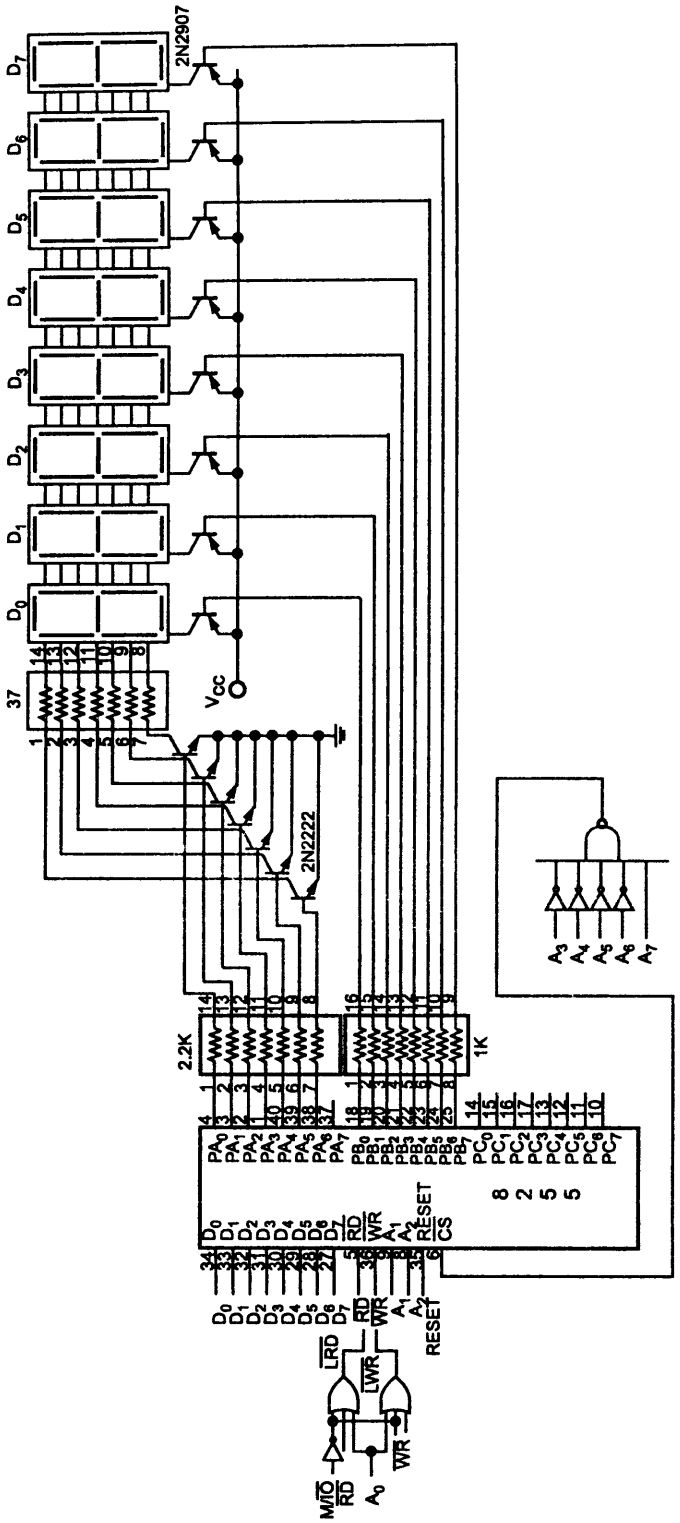


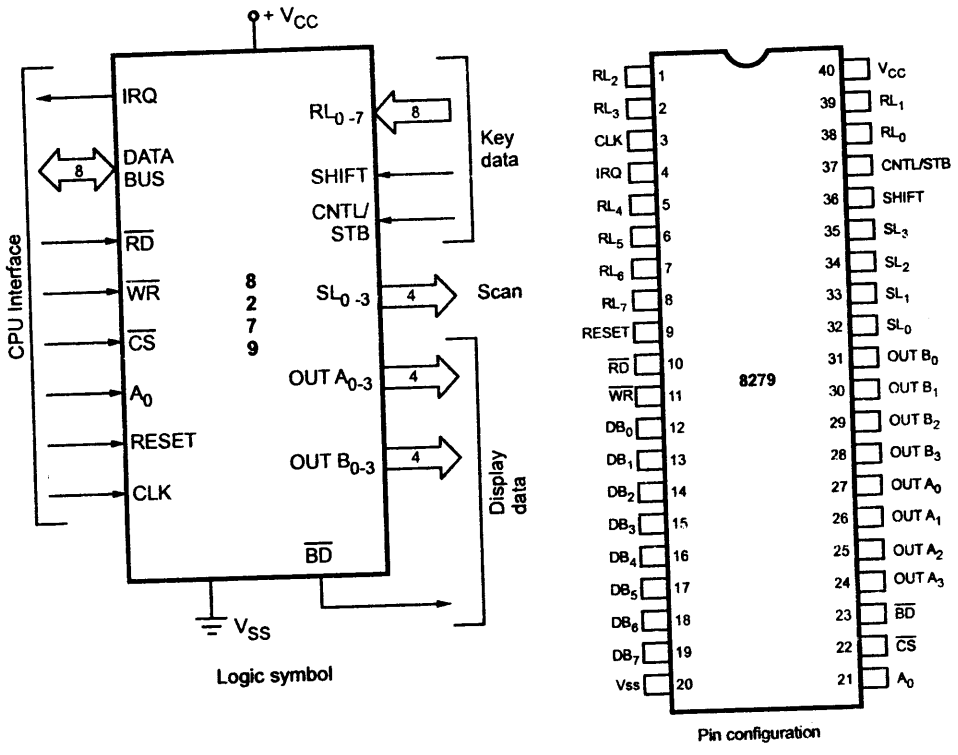
Fig. 10.15 8 Digit 7-segment display interface using 8255

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.

10.5.2 Pin Description

Fig. 10.16 shows functional and pin diagram of 8279. It is a 40 pin device and looking at Fig. 10.16 (a) we can see that these pins are divided in four functional groups :

- CPU interface
- Key data
- Display data
- Scan.



a) Functional diagram Fig. 10.16 b) Pin diagram

10.5.2.1 CPU Interface Pins

As shown in Fig. 10.16 (a), it consists of 8-bit data bus, $\overline{RD}$, $\overline{WR}$, A_0, $\overline{CS}$, RESET, CLK and IRQ lines.

DB_0-DB_7 : 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.

$\overline{RD}$: Read

It is an active low signal. When $\overline{RD}$ 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 A_0 signal.

$\overline{WR}$: Write

It is an active low signal. When $\overline{WR}$ signal is low, CPU loads the data into selected register (control register or display register) depending on the status of A_0 signal.

A_0 : Address line

When A_0 is high, signals are interpreted as a command or status. When A_0 is low signals are interpreted as a data.

$\overline{CS}$: 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

Auto-increment flag is set to one. The interrupt feature of 8279 eliminates the need of polling the keyboard.

10.5.2.2 Keyboard Data

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

RL_0-RL_7 : 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.

10.5.2.3 Display Data

This group consists of OUT A_3-A_0, OUT B_3-B_0 and $\overline{BD}$ lines.

OUT A_3-A_0 and OUT B_3-B_0 : 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 (SL_0-SL_3) 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.

10.5.3 Block Diagram

Fig. 10.17 shows the block diagram of 8279. It consists of four main sections :

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

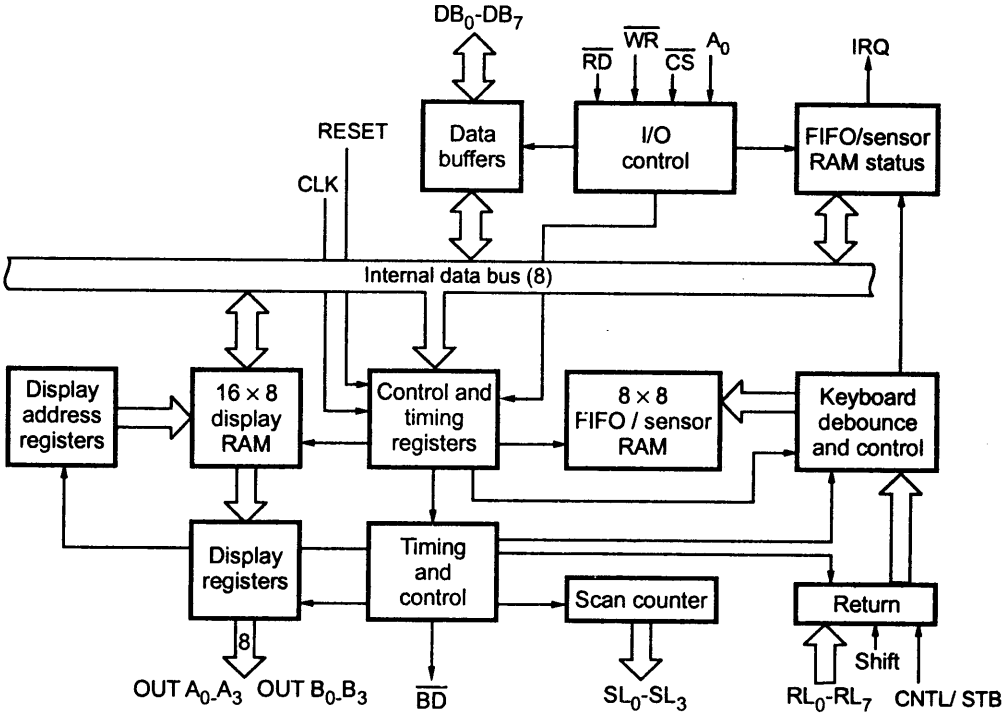


Fig. 10.17

10.5.3.1 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 A_0, $\overline{CS}$, $\overline{RD}$ and $\overline{WR}$ 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 $\overline{CS}$ = 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 10.3. 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 10.3

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 10.4.

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

Table 10.4 Internal timings of 8279

10.5.3.2 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 911 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.

10.5.3.3 Keyboard Section

This section consists of return buffers, keyboard debounce and control, FIFO/sensor RAM and FIFO/sensor RAM status. There 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.

10.5.3.4 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.

10.5.4 Operating Modes

10.5.4.1 Input Modes

The 8279 provides 3 basic input modes :

- Scanned keyboard
- Scanned sensor matrix
- Strobed input.

Scanned keyboard

In this mode, keyboard can be scaned in two ways : **Encoded scan and decoded scan.**