UNIT III
I/O INTERFACING

Memory Interfacing and I/O interfacing - Parallel communication interface – Serial communication interface – D/A and A/D Interface - Timer – Keyboard /display controller – Interrupt controller – DMA controller – Programming and applications Case studies: Traffic Light control, LED display , LCD display, Keyboard display interface and Alarm Controller.

3.1 MEMORY INTERFACING

The interfacing of the 8086 microprocessor to memory is the main step in the design of a microprocessor-based system. Since the 8086 has 20 address lines, a total of 1MB memory space can be interfaced. Various types of memory chips such as RAM,ROM, EPROM, etc may be connected within this space.

The number of memory chips and the address space used depends upon the application needs. It is not necessary that the address space of any microprocessor based system should start only from 00000. It may start from any boundary of 1 KB, i.e. the address bits A_0 to A_9 must be zero at the starting address. Even this restriction is for convenience sake only and can overcome by deep decoding.

The first task is to allocate the address range to different memory chips and devise the address decoding circuit so that based on address information on address lines, the chip select signal to select the particular memory chip may be generated. After the selection of memory chip , the read/write operation may be performed.

The block diagram of address decoder 74LS138 which is widely used is shown in fig 3.1. The decoder is selected when G1.($\overline{G2B}$. $\overline{G2A}$) =1. On selection, the decoder generates the signals Y0 to Y7 based on the information available at A,B and C pins, and based on the truth table shown in fig 3.2.

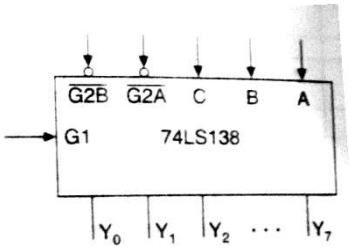


Fig 3.1: Block diagram of 74LS138

Suppose a 2KB chip 2716/6116 is to be interfaced to the 8086 starting from 00000 to 007FFH. If you examine the address information, you will find that the bits A_{19} to A_{11} in address information will always be zero. Thus, we may connect address lines to 74LS138 pins in the following manner.

$$A=A_{11},\ B=A_{12},\ C=A_{13}$$
$$\overline{G2A}=A_{14},\ \overline{G2B}=A_{15}$$
$$G1 = (\overline{A16}.\ \overline{A17}.\ \overline{A18}.\ \overline{A19})\ .\ (M/\overline{IO})$$

AllAbtEngg Android Application for Anna University, Polytechnic & School

Inputs			Outputs							
C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
0	0	0	0	1	1	1	1	1	1	1
0	0	1	1	0	1	1	1	1	1	1
0	1	0	1	1	0	1	1	1	1	1
0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	1	1	1	0	1	1	1
1	0	1	1	1	1	1	1	0	1	1
1	1	0	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	0

Fig 3.2: Truth table of 74LS138 decoder

Since A,B,C are zero, Y0 will be low which can be directly used as a active low chip select ($\overline{CS}$) signal for the memory chip. Figure 3.3 explains the decoding strategy outlined above. For the next address range which starts from 00800H, A11 will become 1. Thus, the pin A of 74LS138 decoder will become 1 for the address range starting from 00800H and this will make Y1 low which can be used to select the next memory chip as shown in fig 3.3.

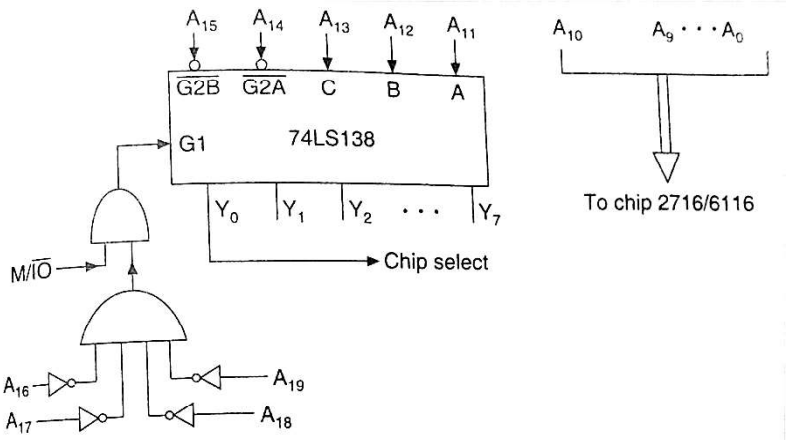


Fig 3.3: Generation of chip select signal for memory

Let us now work out the above decoding strategy again in the following manner.

G1 = M/$\overline{IO}$

G2A = (A14 OR A15 OR A16)

G2B = (A17 OR A18 OR A19)

A=A11, B=A12, C=A13

You will find not only the above but many combinations of the above may work well as effective decoding strategies.

Let us now take another example in which the 8KB memory chip 2764/6164 is to be interfaced starting from address FA000H. Bit patterns of starting address FA000H and end address FBFFFH are

FA000H=1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

FBFFFH=1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

AllAbtEngg Android Application for Anna University, Polytechnic & School

Bits which do not undergo change are A14 to A19. Thus these bits maybe utilized for G1, $\overline{G2A}$ and $\overline{G2B}$.

Bits A12,A13,A14 may be connected to A,B,C input pins respectively. Both Y2 and Y3 outputs of 74LS138 will get activated in the above address range. Thus Y2 AND Y3 = $\overline{CS}$ of the memory chip 2764/6164. There may be different combinations for G1, $\overline{G2A}$ and $\overline{G2B}$ as mentioned before. One possible combination is

$$G1=M/\overline{IO}$$
$$\overline{G2A}=\overline{A19} \text{ OR } \overline{A18}$$
$$\overline{G2B}=\overline{A16} \text{ OR } \overline{A17}$$

However as discussed earlier, the 8086 memory is organized into two memory banka. Memory bank-1, containing all even addresses, is called the lower order memory bank, whereas memory bank-2 containing all odd addresses, is known as the high-order memory bank. The low-order memory bank is selected by $\overline{A0}$ =1 whereas for higher order memory bank addresses(i.e. odd addresses), the 8086 outputs an active low signal $\overline{BHE}$. Thus, the address decoding scheme mentioned earlier needs to be modified in the following manner:

(a) A0 is not used for address decoding, but for memory bank selection along with $\overline{BHE}$.
(b) Memory is now organized into two distint parts-chip for high-order address bank and chip for low-order address bank.

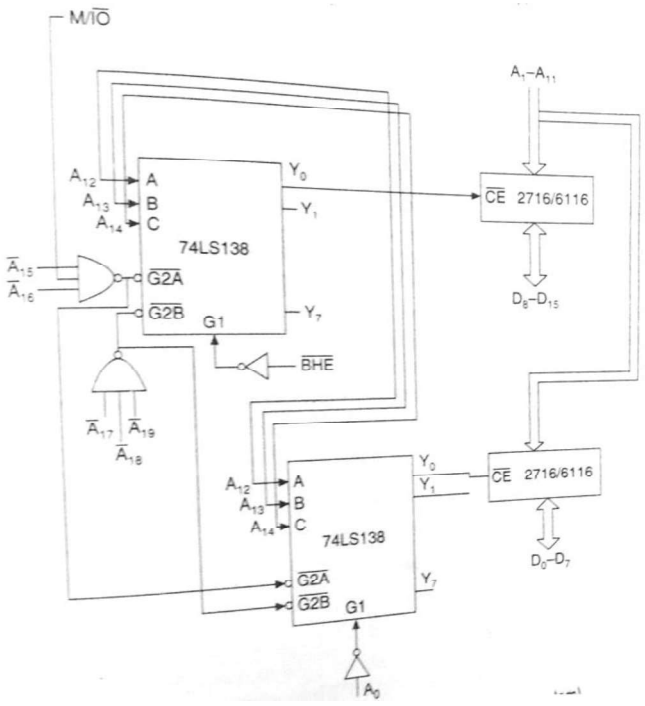


Fig 3.4: Memory address decoding (separate decoder system)

AllAbtEngg Android Application for Anna University, Polytechnic & School

This may be done in two ways:

- Separate decoder for high- and low- order banks.
- Single decoder for both the banks.

Figure 3.4 shows memory address decoding using two separate decoders. Address lines A_1 to A_{11} are connected to memory chips. The address range is 00000 to 00FFFH. i.e. total 4KB. The decoding is performed as

A=A12, B=A13, C=A14

$\overline{G2A}$= A15 OR A16 OR (M/$\overline{IO}$)) = ($\overline{\overline{A15}.\overline{A16}}$. (M/$\overline{IO}$))

$\overline{G2B}$= A17 OR A18 OR A19 = ($\overline{A17}$.$\overline{A18}$. $\overline{A19}$)

For lower decoder, G1=$\overline{A0}$

For upper decoder, G1=BHE

Thus the upper memory chip 2716/6116 (2KB) will be part of high-order memory bank whereas the lower memory chip 2716/6116 (2KB) will be part of low-order memory bank. Other output lines of decoders, i.e. Y1 to Y7 can be used to connect similar chips. Thus the total memory capacity of 32 KB may be interfaced in this way.

Since apart from $\overline{BHE}$ and A0 decoders have the same input signals, we may, therfore explore whether it is possible to use only one address decoder.

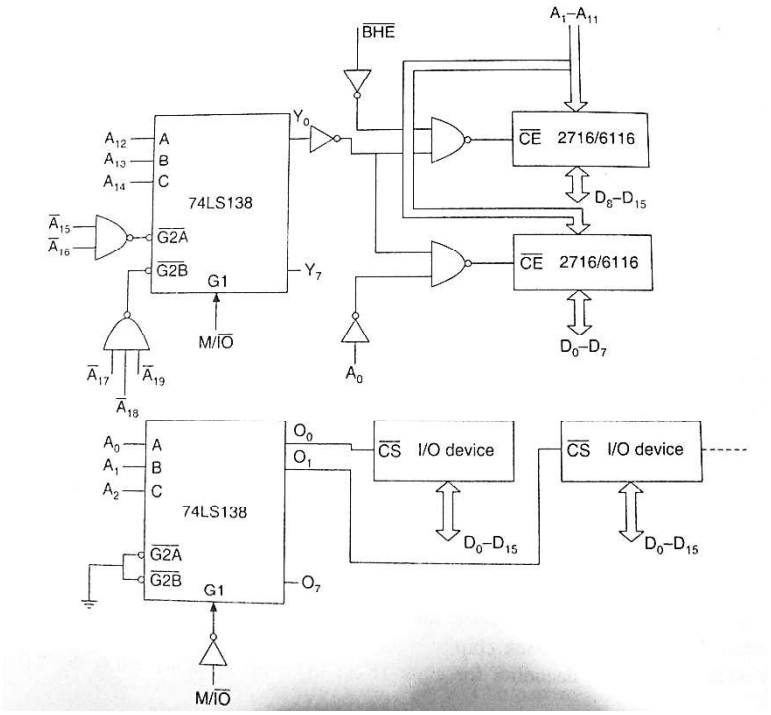


Fig.3.5: Memory and I/O address decoding

Fig 3.5 describes the memory address decoding using one 74LS138 decoder. In this scheme

AllAbtEngg Android Application for Anna University, Polytechnic & School

$$G1=M/\overline{IO}$$

$$\overline{G2B}=[\overline{A17}\,.\,\overline{A18}.\,\overline{A19}]'$$

$$G2A=[\overline{A15}\,.\overline{A16}]'$$

Y0 is combined with $\overline{BHE}$ to select the high-memory bank chip, and Y0 is combined with A0 to select the low-memory bank chip.

In addition, address decoding for eight 16-bit I/O ports is shown using the 74LS138 decoder.

$$A=A0,\ B=A1,\ C=A2$$

$$\overline{G2A}=\overline{G2B}=GND$$

$$G1=(M/\overline{IO})$$

It is evident that the decoder will be selected when $(M/\overline{IO})$ is low, i.e. only for I/O operations, and depending on the information at A0,A1, A2 bits, one of the eight I/O ports will be selected.

3.2 I/O INTERFACES

The I/O and memory interfaces are the counterparts to the bus control logic. What goes between the bus control logic and the interfaces is simply the conductors in the bus; therefore, the interfaces must be designed to accept and send signals that are compatible with the bus control logic and its timing. Although there are similarities in memory and I/O interfaces, there are also significant differences. An I/O interface must be able to:

l. Interpret the address and memory- I/O select signals to determine whether or not it is being referenced and, if so, determine which of its registers is being accessed.

2. Determine whether an input or output is being conducted and accept output data or control information from the bus or place input data or status information on the bus.

3. Input data from or output data to the associated I/O device and convert the data from parallel to the format acceptable to the I/O device, or vice versa.

4. Send a ready signal when data have been accepted from or placed on the data bus, thus informing the processor that a transfer has been completed.

5. Send interrupt requests and, if there is no interrupt priority management in the bus control logic, receive interrupt acknowledgments and send an interrupt type.

6. Receive a reset signal and reinitialize itself and, perhaps, its associated device.

Figure 3.6 contains a block diagram of a typical I/O interface. The function of an I/O interface is essentially to translate the signals between the system bus and the I/O device and provide the buffers needed to satisfy the two sets of timing constraints. Most of the work done by the interface is accomplished by the large block on the right side of the figure. Most often this block is implemented by a single IC device, but its functions could be scattered across several devices. Clearly, its functions vary radically, depending on the I/O device with which it is designed to communicate.

An interface can be divided into two parts, a part that interfaces to the I/O device and a part that interfaces to the system bus. Although little can be said about the I/O device side of the interface without knowing a lot about the device, the bus sides of all interfaces in a given system are very similar because they connect to the same bus. To support the main interface logic there must be data bus drivers and receivers, logic for translating the interface control

AllAbtEngg Android Application for Anna University, Polytechnic & School

signals into the proper handshaking signals, and logic for decoding the addresses that appear on the bus. In an 8086/8088 system, 8286 transceivers could drive the data bus, just as they are used to drive the bus at its bus control logic end. However, the main interface devices may have built-in drivers and receivers that are sufficient for small, single board systems.

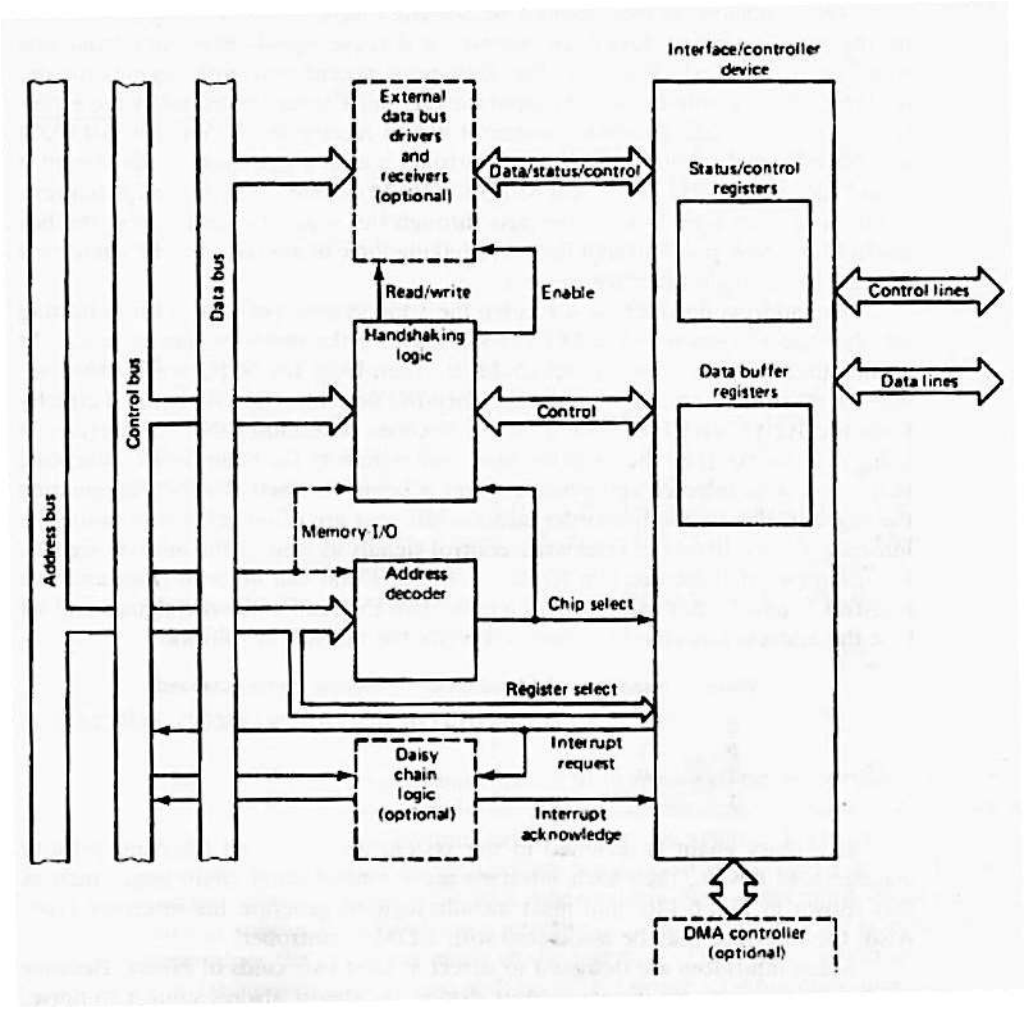


Figure 3.6: Typical block diagram of an I/O interface

The handshaking logic cannot be designed until the control signals needed by the main interface device are known, and these signals may vary from one interface to the next. Typically, this logic must accept read/write signals for determining the data direction and output the $\overline{OE}$ and$\overline{T}$ signals required by the 8286s. In a maximum mode 8086/8088 system it would receive the $\overline{IOWC}$ (or $\overline{AIOWC}$) and $\overline{IORC}$ signals from the 8288 bus controller and in a minimum mode system it would receive the $\overline{RD}$, $\overline{WR}$, and M/$\overline{IO}$ (or IO/$\overline{M}$) signals. The interrupt request, ready, and reset lines would also pass through this logic. In some cases, the bus control

AllAbtEngg Android Application for Anna University, Polytechnic & School

lines may pass through the handshaking logic unaltered (i.e., be connected directly to the main interface device).

The address decoder must receive the address and, perhaps, a bit indicating whether the address is in the I/O address space or the memory address space. In a minimum mode system this bit could be taken from the M/$\overline{IO}$ (or IO/$\overline{M}$) line, but in a maximum mode system the memory I/O determination is obtained directly from the $\overline{IOWC}$and$\overline{IORC}$ lines. If the decoder determines that its interface is being referenced, then the decoder must send signals to the main device indicating that it has been selected and which register is being accessed. The bits designating the register may be the low-order address bits, but are often generated inside the interface device from the read/write control signals as well as the address signals. For example, if there are two registers A and B that can be read from and two registers C and D that can be written into, then the read and write signals and bit 0of the address bus could be used to specify the register as follows:

Write	Read	Address Bit 0	Register being accessed
0	1	0	A
0	1	1	B
1	0	0	C
1	0	1	D

If a daisy chain is included in the system instead of an interrupt priority management device, then each interface must contain daisy chain logic, and must include logic to generate the interrupt type. Also, the interface may be associated with a DMA controller.

Many interfaces are designed to detect at least two kinds of errors. Because the lines connecting an interface to its device are almost always subject to noise, parity bits are normally appended to the information bytes as they are transmitted. If even parity is used the parity bit is set so that the total number of 1s, including the parity bit is even. For odd parity the total number of 1s is odd. As these bytes are received the parity is checked and, if it is in error, a certain status bit is set in a status register. Some interfaces are also designed to check error detection redundancy bytes that are placed after blocks of data. The other type of error most interfaces can detect is known as an overrun error. When a computer inputs data it brings the data in from a data-in buffer register. If, for some reason, the contents of this register are replaced by new data before they are input by the computer an overrun error occurs. Such an error also happens when data are put in a data-out buffer before the current contents of the register have been output. As with parity errors, overrun errors cause a certain status bit to be set. Interfaces can be categorized according to the way in which they communicate with their I/O devices.

3.3 PARALLEL COMMUNICATION INTERFACE

Parallel communication is accomplished by simultaneously transferring bits over separate lines. Its advantage over serial communication is that for lines with a given maximum bit rate, higher information rates can be attained due to the use of several lines. The disadvantage is, of course, the cost of the extra lines and, because these costs increase with distance, parallel communication is used over long distances only if high data rates are required.

AllAbtEngg Android Application for Anna University, Polytechnic & School

Unlike serial communication, parallel communication has not been well standardized. Parallel transfers may be performed by simply putting data in the interface's data-in buffer register or taking data out of the interface's data-out buffer register, or they may be conducted under the control of handshaking and/or timing signals. Normally, one character (or other piece of information) is transferred at a time and there is no problem in determining the end of a character or in finding the beginning of a transmission. There is no well-defined format for synchronous or asynchronous transmissions, but if a timing signal is to coordinate the activity between the device and the interface, the transmission would be considered synchronous. If only handshaking signals are employed, the transmissions would be asynchronous.

An interface may be designed for only outputting, only inputting, inputting and outputting over separate sets of lines, or performing I/O over a single set of bidirectional lines. If an interface is connected to a line printer, it would only output data, and if it services a card reader, it would only input data. An interface that services both a paper tape reader and a paper tape punch would require one set of input lines and one set of output lines, but an interface for a device that does not input and output data simultaneously could use only one set of bidirectional lines.

3.3.1 Programmable Peripheral Interface (PPI) 8255

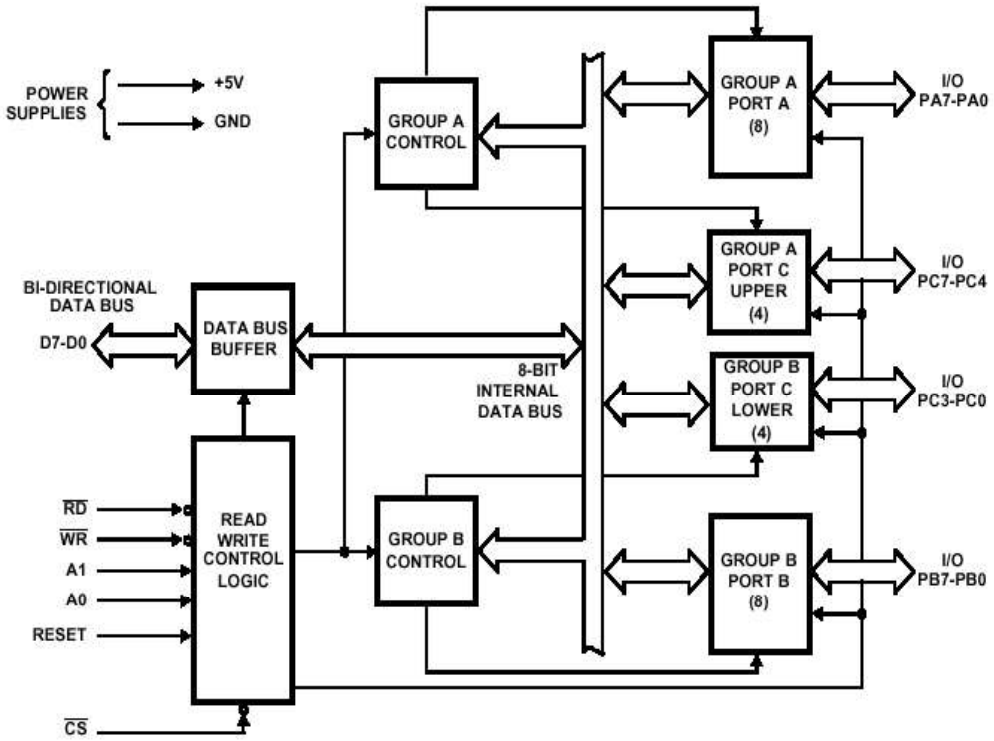


Fig 3.7 : 8255 Block Diagram

It has 24 I/O pins distributed to four ports- port A, port B, port C (upper) and port C (lower). These ports are divided into two groups group A and group B. Group A and B get the control signal from CPU and send the command to the individual control blocks. Group A send

AllAbtEngg Android Application for Anna University, Polytechnic & School

the control signal to port A and Port C (Upper) PC7-PC4.Group B send the control signal to port B and Port C (Lower) PC3-PC0.

3.3.2 Operation modes:

PPI operates under two modes:

1. Bit set/Reset mode.
2. I/O mode

1. Bit Set/Reset mode:

The PORT C can be Set or Reset by sending OUT instruction to the CONTROL registers.

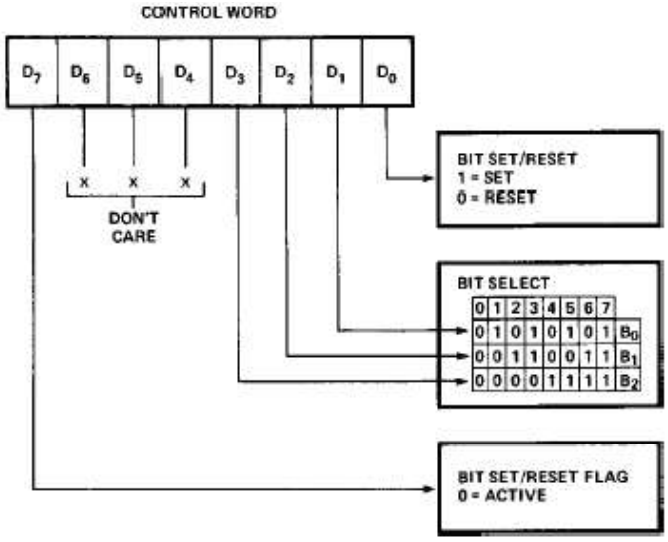


Fig 3.8: Control word format for Bit Set/Reset mode

2.I/O mode:

Control word format for I/O mode is given in fig 3.9. There are three basic modes of operation selected by the software.

a) Mode 0- Basic Input/output:

If a group is in mode 0, it is divided into two sets. For group A these sets are port A and the upper 4 bits of port C, and for group B they are port B and the lower 4 bits of port C. Each set may be used for inputting or outputting, but not both. Bits D4, D3, Dl, and D0 in the control register specify which sets are for input and which are for output. These bits are associated with the sets as follows:

- D4 - Port A.
- D3 - Upper half of port C.
- D1 - Port B.
- D0 - Lower half of port C.
- If a bit is 0, then the corresponding set is used for output; if it is 1, the set is for input.

AllAbtEngg Android Application for Anna University, Polytechnic & School

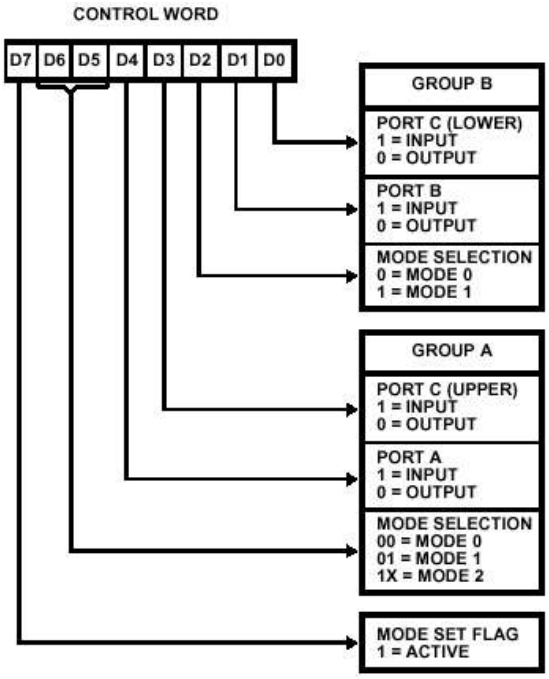


Fig 3.9: Control word format for I/O mode

b)Mode 1 – Strobed Input /output:

Port A and port B use the lines on port C for handshaking signals. When group A is in this mode port A is used for input or output according to bit D4 (D4 = 1 indicates input), and the upper half of port C is used for handshaking and control signals. For inputting, the four MSBs of port C are assigned the following symbols and definitions:

When Port A is used as input:		
PC4	$\overline{STB}$A	A 0 applied to this pin causes PA7-PA0 to be latched, or "strobed," into port A.
PC5	IBFA	Indicates that the input buffer is full. It is 1 when port A contains data that have not yet been input to the CPU. When a 0 is on this pin the device can input a new byte to the interface.
PC6,PC7	-	If D3 of the control register is 0, they are for outputting control signals; otherwise, they are for inputting status.
When Port A is used as output:		
PC4,PC5	-	If D3 of the control register is 0, they are for outputting control signals; otherwise, they are for inputting status.
PC7	$\overline{OBF}$A	Indicates that the output buffer is full. It outputs a 0 to the device when port A is outputting new data to be taken by the device.
PC6	$\overline{ACK}$A	Device puts a 0 on this pin when it accepts data from port A.

AllAbtEngg Android Application for Anna University, Polytechnic & School

PC3 is denoted INTRA and is associated with group A. It is used as an interrupt request line and is tied to one of the IR lines in the system bus.

- When inputting to port A, this pin becomes 1 when new data are put in port A (i.e., it is controlled by PC4) and is cleared when the CPU takes the data.
- For output, this pin is set to 1 when the contents of port A are taken by the device and is cleared when new data are sent from the CPU.

When Port B is used as input:		
PC2	$\overline{\text{STB}}$B	A 0 applied to this pin causes PB7-PB0 to be latched, or "strobed," into port B.
PC1	IBFB	Indicates that the input buffer is full. It is 1 when port B contains data that have not yet been input to the CPU. When a 0 is on this pin the device can input a new byte to the interface.
When Port B is used as output:		
PC2	$\overline{\text{ACK}}$B	Indicates that the device is ready to accept data from PB7-PB0.
PC1	$\overline{\text{OBF}}$B	Indicates that the output buffer is full. It outputs a 0 to the device when port B is outputting new data to be taken by the device.

PC0 becomes INTRB and its use is analogous to that of INTRA. The interrupt enable for group A is controlled by setting or clearing internal flags. Setting or clearing these flags is simulated by setting or clearing PC4, for input, or PC6, for output, using a Set/ Reset instruction. Similarly, the interrupt enable for group B is controlled by set/ clear of PC2 for both input and output.

Mode 2—Strobed bidirectional Bus:

This mode applies only to group A, although it also uses PC3 for making interrupt requests. In mode 2, port A is a bidirectional port and the four MSBs of port C are defined as follows:

PC4	$\overline{\text{STB}}$A	A 0 on this line causes the data on PA7-PA0 to be strobed into port A.
PC5	IBFA	Becomes 1 when port A is filled with new data from lines PA7-PA0 and is cleared when these data are taken by the CPU.
PC6	$\overline{\text{ACK}}$A	Indicates that the device is ready to accept data from PA7-PA0.
PC7	$\overline{\text{OBF}}$A	A Becomes 0 when port A is filled with new data from the CPU and is set to 1 when data are taken by the device.

3.4 SERIAL COMMUNICATION INTERFACE

3.4.1 Serial data communication:

Serial data communication uses two methods,

- Synchronous and
- Asynchronous.

Synchronous:

A synchronous transmitted character consists of 5 to 8 information bits optionally followed by a parity bit that contains no start or stop bit. All characters have the same number

AllAbtEngg Android Application for Anna University, Polytechnic & School

of bits. The same clock is used to generate the bits for the synchronization of queue process. The transmitter must transmit a character during each 'n' bit interval. If the character is not available at the beginning of the interval under run error will occur.

Asynchronous:

Asynchronous data communication is widely used for character-oriented transmissions i.e, transfers a single byte at a time

The data coming in at the receiving end of the data line in a serial data transfer is all 0s and 1s; it is difficult to make sense of the data unless the sender and receiver agree on a set of rules, a protocol, on how the data is packed, how many bits constitute a character, and when the data begins and ends.

Start and stop bits

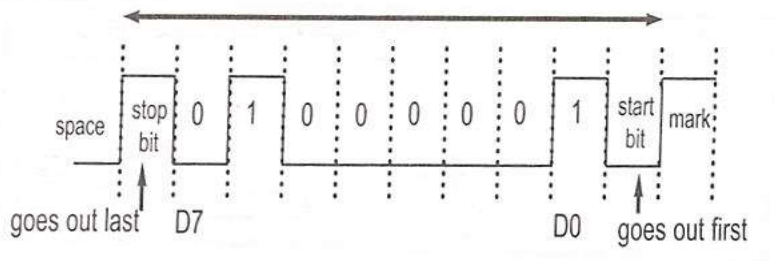


Fig 3.10: Framing ASCII "A" (41H)

In the asynchronous method, each character is placed between start and stop bits. This is called framing. In data framing for asynchronous communications, the data, such as ASCII characters, are packed between a start bit and a stop bit. The start bit is always one bit, but the stop bit can be one or two bits. The start bit is always a 0 (low) and the stop bit(s) is 1 (high). For example, look at Figure 5.12 in which the ASCII character "A" (8-bit binary 0100 0001) is framed between the start bit and a single stop bit. Notice that the LSB is sent out first.

When there is no transfer, the signal is 1 (high), which is referred to as mark. The 0 (low) is referred to as space. The transmission begins with a start bit followed by D0, which is the LSB, then the rest of the bits until the MSB (D7), and finally, the one stop bit indicating the end of the character "A".

3.4.2 Data transmission:

- Simplex
- Duplex

Simplex:

Simplex mode of transmission means transfer of data from source to destination only(one way transmission)

Duplex:

In data transmission if the data can be transmitted and received, it is a duplex transmission. This is in contrast to simplex transmissions. Duplex transmissions can be

- Half or
- Full duplex,

Half duplex:

If the data is transmitted one way at a time, it is referred to as half duplex.

AllAbtEngg Android Application for Anna University, Polytechnic & School

Full duplex :

If the data can go both ways at the same time, it is full duplex. Of course, full duplex requires two wire conductors for the data lines (in addition to the signal ground), one for transmission and one for reception, in order to transfer and receive data simultaneously.

3.4.3 Physical Communication Standard:

This standard affect:

1. Transmission rate
2. Electrical Characteristics
3. Line definition and notations

3.4.4 Programmable Communication Interface(8251):

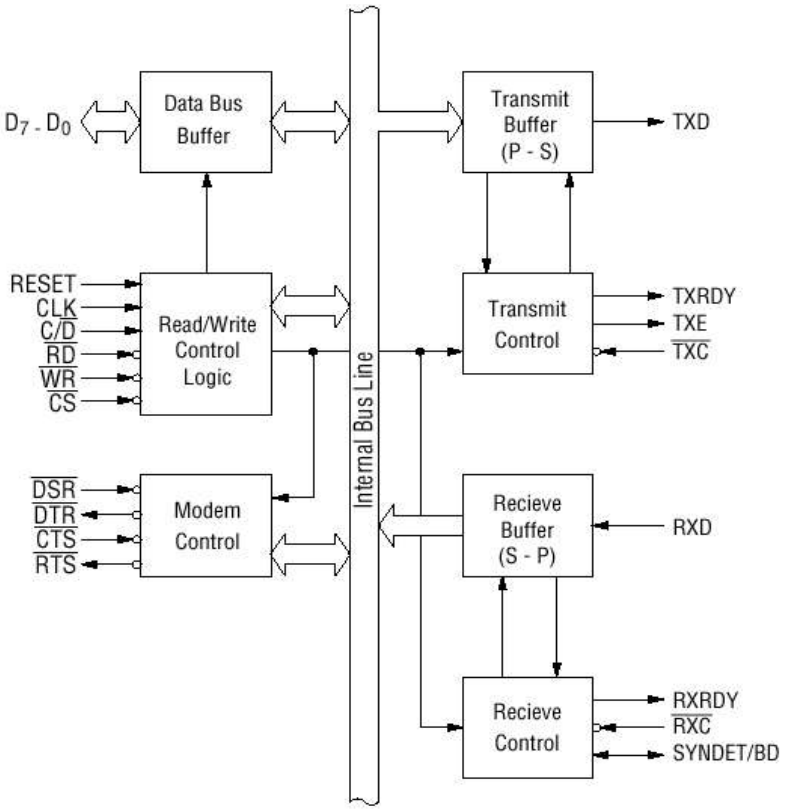


Fig 3.11: Block Diagram of 8251

Pin Description

Data bus buffer:

This is bidirectional data bus which receive control words and transmits data from the CPU and sends status words and received data to CPU.

Read/write control logic:

It has six input signal

AllAbtEngg Android Application for Anna University, Polytechnic & School

RESET (Input terminal): A "High" on this input forces the 8251 into "reset status."
CLK (Input terminal) :CLK signal is used to generate internal device timing.
WR (Input terminal) : This is the "active low" input terminal which receives a signal for writing transmit data and control words from the CPU into the 8251.
RD (Input terminal) : This is the "active low" input terminal which receives a signal for reading receive data and status words from the 8251.
C/D (Input terminal): This is an input terminal which receives a signal for selecting data or command words and status words when the 8251 is accessed by the CPU. If C/D = low, data will be accessed. If C/D = high, command word or status word will be accessed.
CS (Input terminal) : This is the "active low" input terminal which selects the 8251 at low level when the CPU accesses.
Modem control:
DSR (Input terminal) : This is an input port for MODEM interface. The input status of the terminal can be recognized by the CPU reading status words.
DTR (Output terminal) : This is an output port for MODEM interface. It is possible to set the status of DTR by a command.
CTS (Input terminal) : This is an input terminal for MODEM interface which is used for controlling a transmit circuit. The terminal controls data transmission if the device is set in "TX Enable" status by a command. Data is transmittable if the terminal is at low level.
RTS (Output terminal) : This is an output port for MODEM interface. It is possible to set the status RTS by a command.
Transmit buffer:
TXD (output terminal) :This is an output terminal for transmitting data from which serial-converted data is sent out.
Transmit control:
TXRDY (output terminal) : This is an output terminal which indicates that the 8251is ready to accept a transmitted data character.
TXEMPTY (Output terminal) :This is an output terminal which indicates that the 8251 has transmitted all the characters and had no data character
TXC (Input terminal) : This is a clock input signal which determines the transfer speed of transmitted data.
Receive Buffer:
RXD (input terminal) :This is a terminal which receives serial data.
Receive control:
RXRDY (Output terminal) : This is a terminal which indicates that the 8251 contains a character that is ready to READ.
RXC (Input terminal) : This is a clock input signal which determines the transfer speed of received data.
SYNDET/BD (Input or output terminal) : This is a terminal whose function changes according to mode.

AllAbtEngg Android Application for Anna University, Polytechnic & School

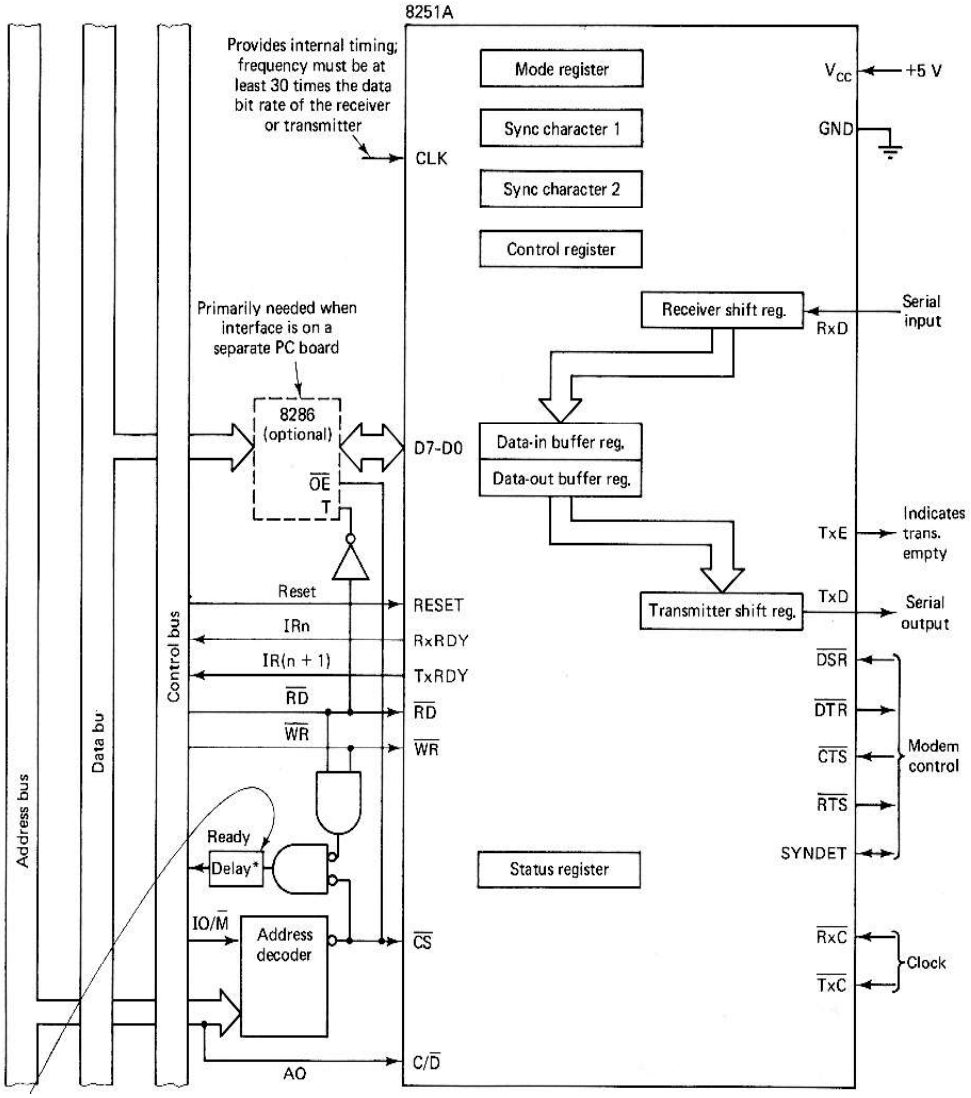


Fig 3.12: 8251 A serial communication interface

- The ***status register*** would error and other information concerning the state of current transmission.
- ***Control Register*** is for holding the information that determines the operating mode of the interface.
- For input, the serial bit stream arriving on the ***RxD*** pin is shifted into the ***receiver shift register*** and then the data bits are transferred to the data-in buffer register, where they can input by the CPU.

AllAbtEngg Android Application for Anna University, Polytechnic & School

- On output the data bits put in the data out buffer register by the CPU are transferred to the ***transmitter shift register*** along with synchronization bits are shifted out through the ***TXD*** pin.
- The contents of ***mode register*** determine whether 8251 is in asynchronous mode or synchronous mode.
- The ***sync character*** registers are for storing the sync characters needed for synchronous communication.
- The 8251A internally interprets the C/D,RD and WR signals as follow:

C/$\overline{D}$ (=A0)	RD	WR	
0	0	1	Data input from the data-in buffer
0	1	0	Data output to the data-out buffer
1	0	1	Status register is put on data bus
1	1	0	Data bus is put in mode, control or sync character register

3.5 D/A AND A/D INTERFACE

3.5.1 Digital-to-Analog Converter

Digital-to-analog converters (DACs) translate digital information into equivalent analog voltage or current. The DAC works on a very simple principle of decoder resistance network. The main constituents of DAC are the decoder network, an analog switch for each digital input, a buffer amplifier and the necessary control logic as shown in Figure 3.13.

The digital interface provides the storage of input data word and the control of the analog switch position. The analog switches under the control of digital interface either connect the reference source to the decoder network input terminal or ground the terminal.

The buffer amplifier provides impedance buffering and current driving capability. It is generally required because the output impedance of the decoder network is relatively high. Figure 3.14 shows the pin diagram of a 12-bit DAC. It is quite easy to interface a digital-to-analog converter to a microprocessor. If an n-bit microprocessor is to be interfaced to an m-bit DAC, the following three possibilities may arise.

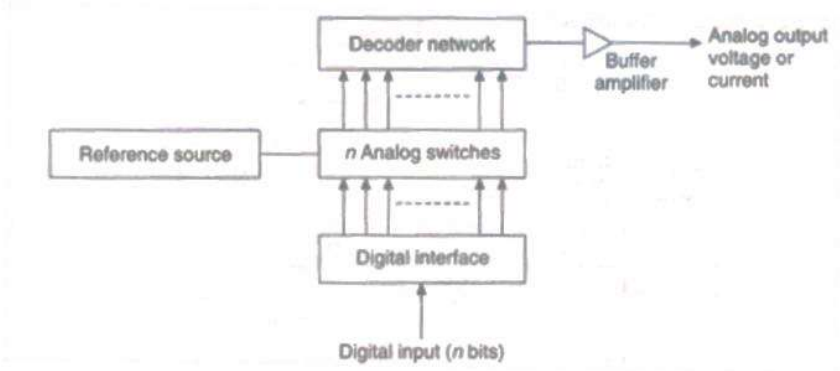


Fig 3.13:Digital-to-Analog Converter

AllAbtEngg Android Application for Anna University, Polytechnic & School

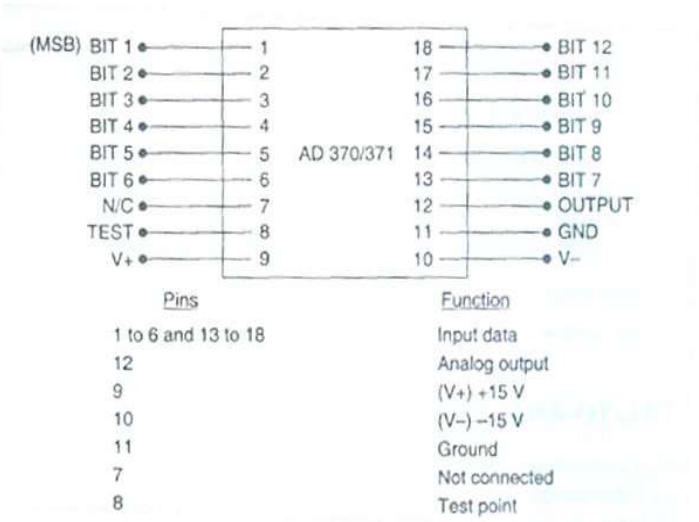


Fig 3.14: DAC AD 370/371 pin diagram

Case (i) n = m:

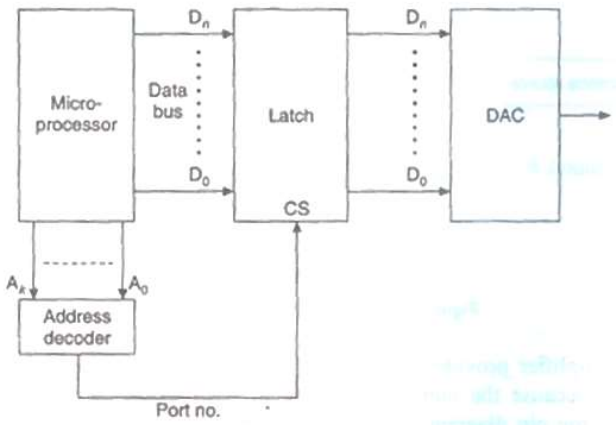


Figure 3.15: Microprocessor interface to DAC *(n* = m).

The output of the microprocessor can be directly connected to DAC through a latch as shown in Figure 3.15. The microprocessor outputs the 8-bit digital value to the specified latch which is connected to the DAC. The software interface consists of instructions to load the digital data to the latch.

Case (ii) n >m:

This case is abnormal. However, if one is forced to connect a higher-bit (n) microprocessor to a lower-bit (m) DAC, only then, the lower m bits of the microprocessor data bus are connected to the DAC latch.

For DAC, the microprocessor behaves like an *m-bit* microprocessor (Figure 3.16). The programmer must ensure that the data is represented only in *m* bits, otherwise the higher-order

AllAbtEngg Android Application for Anna University, Polytechnic & School

bits may be lost, resulting in loss in accuracy. The software consists of loading data into the latch.

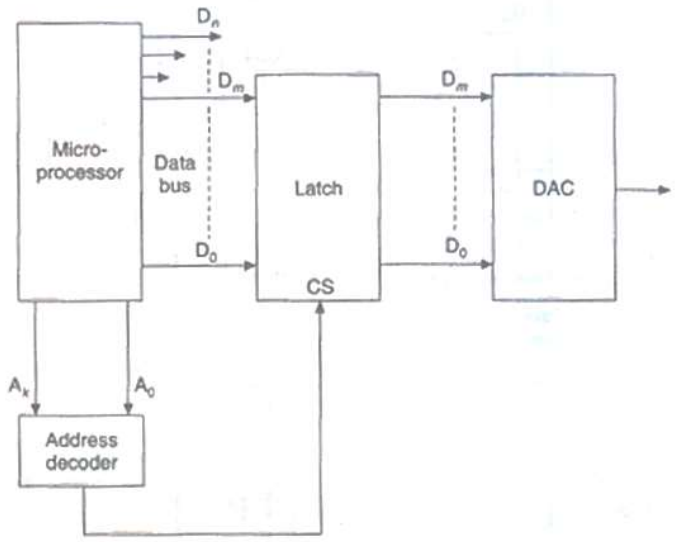


Fig3.16 :Microprocessor interface to DAC (n > m).

Case (iii) n <m:

Often it happens that 16-bit calculations are required to be performed on an 8-bit microprocessor. This is achieved by using double precision. If this result is to be output on a 16-bit DAC, then the 16-bit DAC will have to be interfaced to an 8-bit microprocessor. Since DAC works continuously without any start/stop pulse, the two bytes should be loaded to DAC at the same instant.

If the two bytes of the 16-bit data are loaded one by one, then the analog value at the output will not represent the 16-bit digital value. Thus, interfacing a lower bit (n) processor to a higher bit (m) DAC is quite tricky.

The interface block diagram is shown in Figure 3.17. First the lower-order n-bit data are stored in latch 1. The higher-order (m - n) bits are then stored in latch 2. The m-bit data are then stored. simultaneously in latch 3 and latch 4 using the same pulse. Latch 3 and latch 4 are connected to DAC directly.

The software involves storing the data in various latches. The latch 3 and latch 4 are selected simultaneously and thus the DAC gets all the m bits of data at the same instant. Some DACs output current while others output voltage as analog values. Often, it is required to convert voltage to current and vice versa. This can be achieved through an operational amplifier circuit.

AllAbtEngg Android Application for Anna University, Polytechnic & School

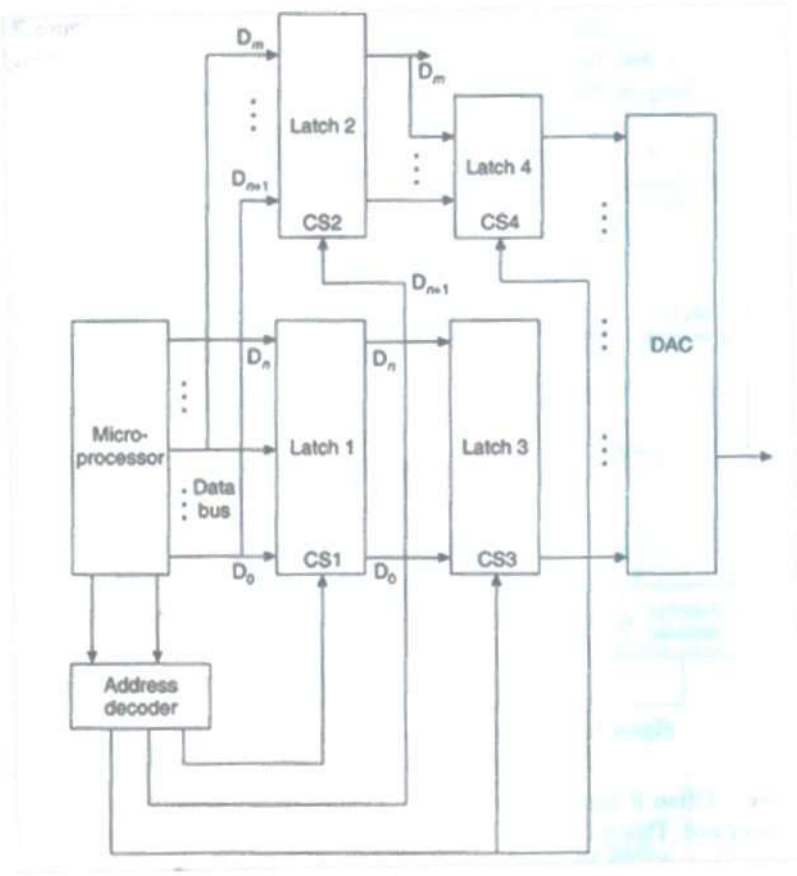


Fig3.17: Microprocessor interface to DAC (n < m).

3.5.2 Analog-to-Digital Converter

The analog-to-digital converter (ADC) takes an analog signal as input and presents an equivalent digital signal as output. The basic techniques include-ramp, integrating ramp, successive approximation, etc. Out of these, the successive approximation technique is quite prevalent. The configuration of an ADC chip is shown in Figure 3.18.

The whole operation is carried out by the following signals.

- Start Convert
- End of Conversion
- Output Enable

The control logic sends the Start Convert signal when the analog signal is stable on the analog-input line. The ADC converts the analog signal to digital and sends the End of Conversion signal on completion of conversion. To read the output from digital output lines, the control logic should send the Output Enable signal to ADC which will then enable the line of the digital output. In many ADCs the digital output is sent to digital output lines by ADC along with the End of Conversion signal.

AllAbtEngg Android Application for Anna University, Polytechnic & School

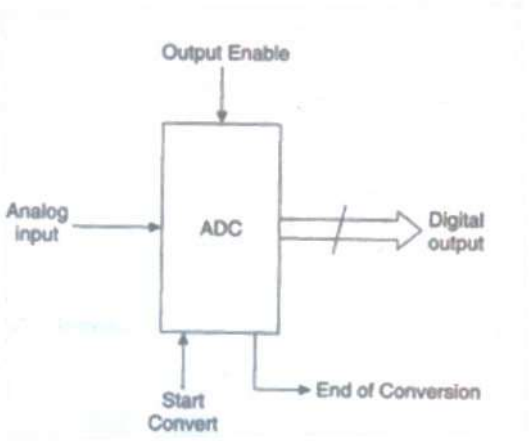


Fig 3.18: ADC chip configuration

Thus the Output Enable is not used in such cases. The analog-to-digital converter (ADC) deals with the following signals.

Input voltage	Input analog voltage. Input to ADC
Start Convert	Command to start conversion. Input to ADC.
End of Conversion	Information that conversion is complete and the output can be read. Output signal from ADC.
Digital output	Digital value of analog input derived from conversion. Output from ADC.

A microprocessor can be interfaced to ADC in any one of the following three modes

Asynchronous Mode

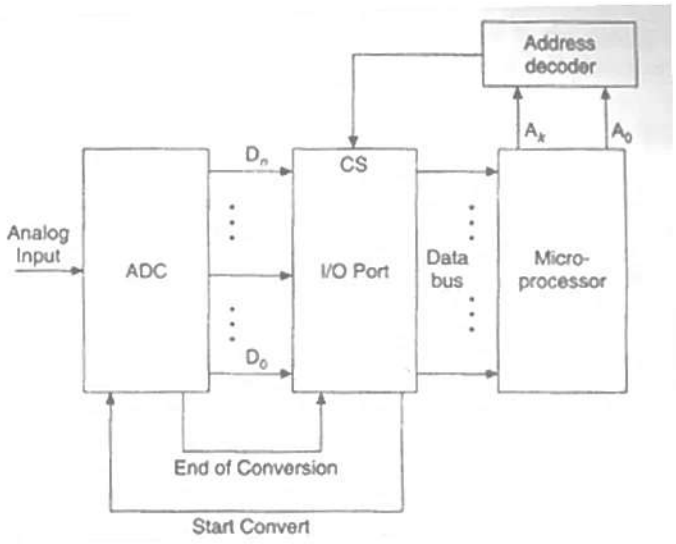


Fig 3.19: Microprocessor interface to ADC (asynchronous mode)

AllAbtEngg Android Application for Anna University, Polytechnic & School

Figure 3.19 shows the microprocessor interface with ADC in asynchronous mode. The microprocessor starts the conversion process by sending the Start Convert signal to the ADC. The ADC takes the analog input and converts it to digital. When the conversion is complete, the ADC outputs the End of Conversion signal to the microprocessor. On receiving this signal, the microprocessor reads the data present at the output of ADC.

The software flowchart for interface is presented in Figure 3.20. The microprocessor continuously checks for the End of Conversion signal after the Start Convert signal is output.

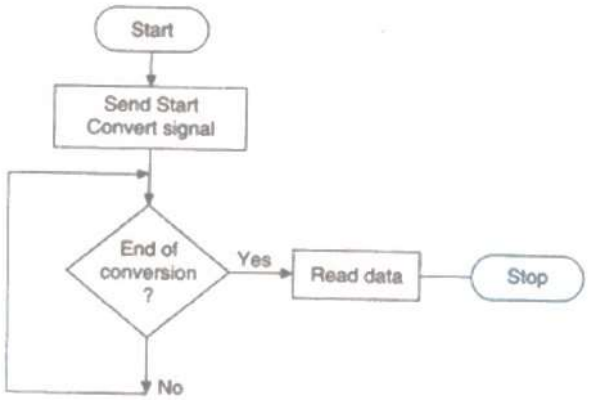


Fig 3.20: Asynchronous mode ADC- Microprocessor interface software flowchart

Synchronous Mode

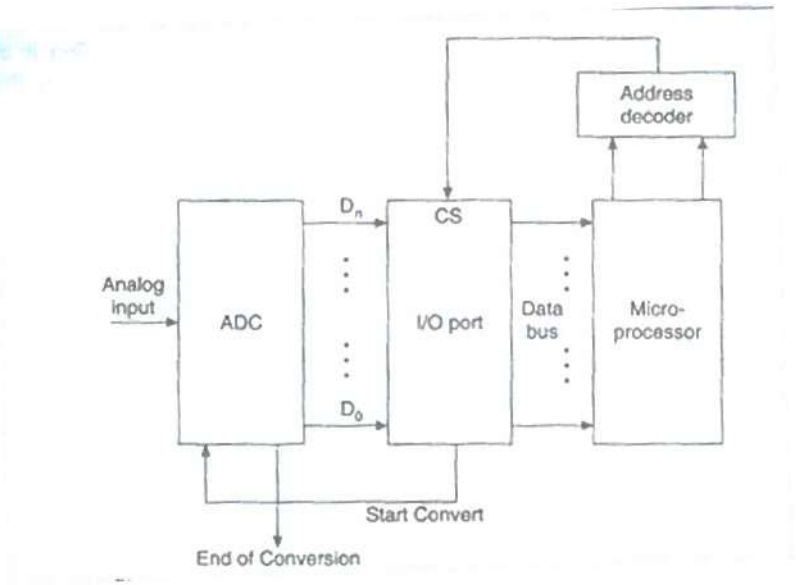


Fig 3.21 Microprocessor interface to ADC (Synchronous mode)

AllAbtEngg Android Application for Anna University, Polytechnic & School

It is evident that in the asynchronous mode, the microprocessor wastes time by waiting for the End of Conversion signal. The conversion time for an ADC is up to a few milliseconds. Thus, for every conversion, this much time of the microprocessor will be wasted.

Figure 3.21 shows the microprocessor interface with ADC in the synchronous mode. The microprocessor issues the Start Convert signal to initiate the conversion process. Since the conversion time is known, the microprocessor, then, executes certain instructions so That the execution time of the instructions is greater or equal to the conversion time. The microprocessor will then read the datafrom ADC directly.

The software flowchart for the interface is shown if fig 3.22. The microprocessor utilizes the conversion time in executing a part of the program. This program segment will clearly not use the ADC output. The End of Conversion signal is not used in this interface.

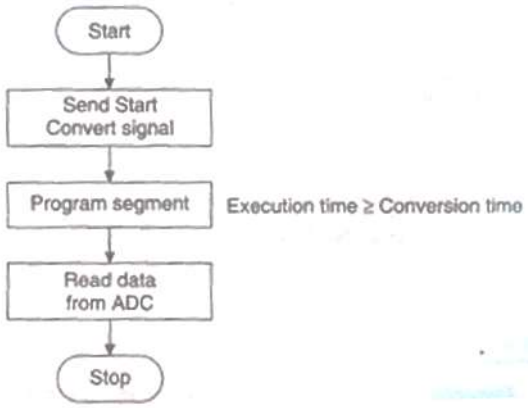


Fig 3.22 Synchronous mode ADC - Microprocessor interface software flowchart

Interrupt Mode

Many users, while accepting the benefits of the synchronous interface, find it inconvenient to write the program segment whose execution time is nearly equal to the conversion time. In a system containing 50 channels, 50 such different program segments will have to be written. The interrupt mode is used to avoid this. The interrupt mode interface is shown in Figure 3.23 and the software flowchart for the interface is presented in Figure 3.24

In this mode, the End of Conversion signal is physically connected to one of the interrupt inputs of the microproces.sor. The microprocessor initiates the conversion sending the Start Convert signal to the ADC. The ADC completes the conversion and sends the End of Conversion signal. Since the End of Conversion signal is connected to one of the interrupt inputs of the microprocessor, the microprocessor is interrupted. The microprocessor suspends its program execution, saves the status and then executes the Interrupt Service Routine to service the interrupt caused. The interrupt service routine, in the present case, contains instructions to read and store the digital data from ADC. It is clear that the interrupt mode is most advantageous for the interfacing of the microprocessor to the ADC. However, it uses one interrupt input of the microprocessor.

AllAbtEngg Android Application for Anna University, Polytechnic & School

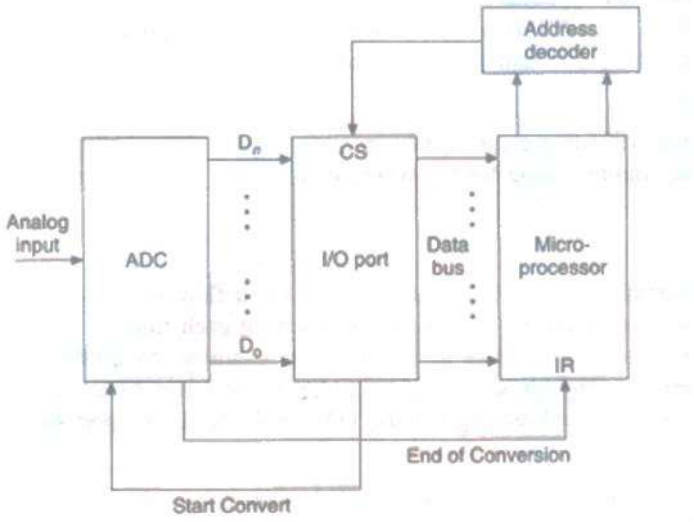


Fig 3.23 Microprocessor interface to ADC (interrupt mode)

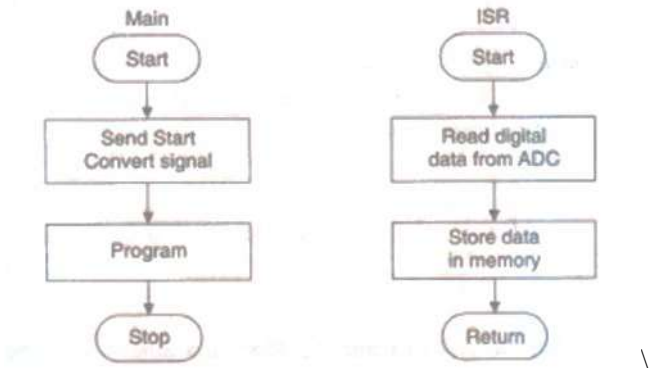


Fig 3.24 Interrupt mode ADC - Microprocessor interface software flowchart

3.6 PROGRAMMABLE TIMERS AND EVENT COUNTERS

One of the most common problem in any microcomputer system is the generation of accurate time delay under software control. In many application the processor has to wait for some time. In keyboard interface, the CPU waits about 5 ms in case of each key depression, in order to check for false depression.

The common software approach is to set up the timing loop by moving a fixed number to a register pair and decrementing that number till it becomes zero. The fixed number depends on the time delay required.

8253 solves this problem by facilitating three 16-bit programmable counters on the same chip.

Other functions that can be achieved by 8253 are:

AllAbtEngg Android Application for Anna University, Polytechnic & School

1. Event counter
2. Programmable rate generator
3. Binary Rate Multiplier
4. Real Time clock
5. Digital one-shot
6. Complex motor controller

3.6.1 Programmable Interval Timer 8253/8254

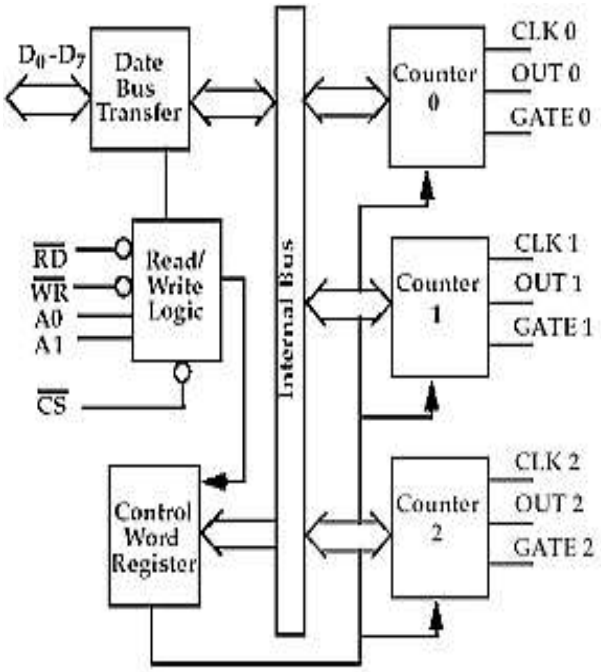
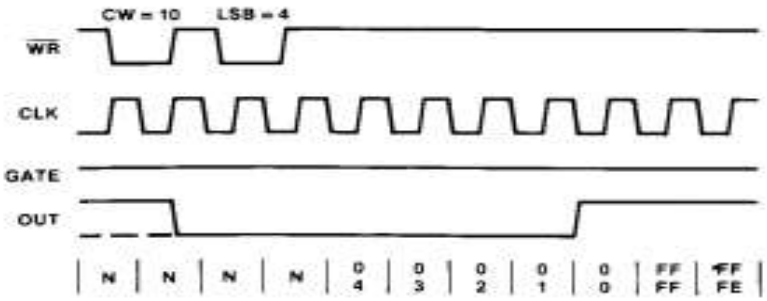


Fig 3.25: Functional block diagram of 8253

3.6.2 Operating Modes

The counters are fully independent and each have separate mode configurations and counting operations, binary or BCD.

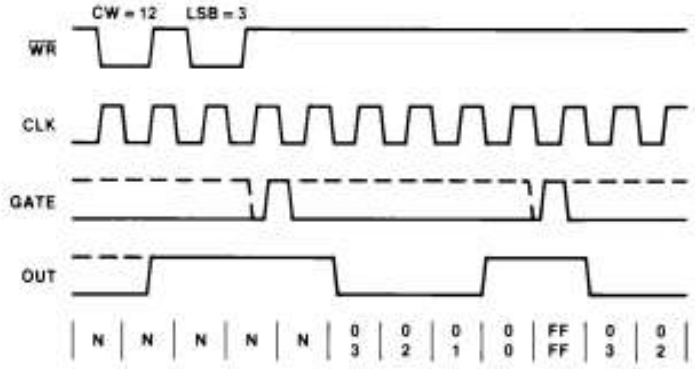
Mode 0: Interrupt on terminal count



AllAbtEngg Android Application for Anna University, Polytechnic & School

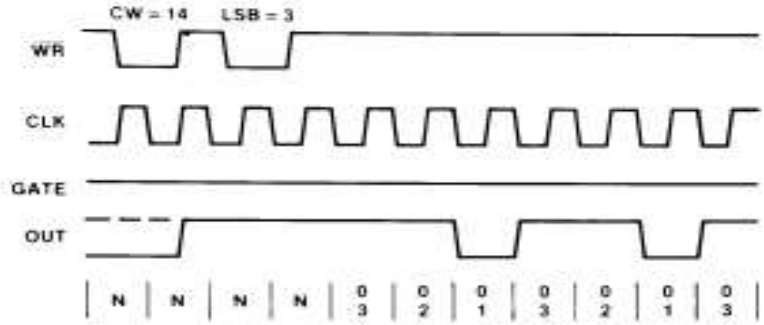
- This mode is used for event counting.
- The GATE input is used to enable or disable counting.
- When the GATE input is high counting is enabled and when it is low the counting is disabled.
- After the count is loaded to the selected count registers, OUT remains low and the counter starts counting one cycle after the counter is loaded.
- On reaching the terminal count OUT remains high until the count register is reloaded .

Mode 1: Programmable one shot



- The GATE input is used to trigger the OUT.
- The OUT will be initially high and go low on the count following the rising edge of the GATE.
- When the terminal count is reached, the OUT goes high and remains high until the clock pulse after the next trigger.

Mode 2: Rate Generator

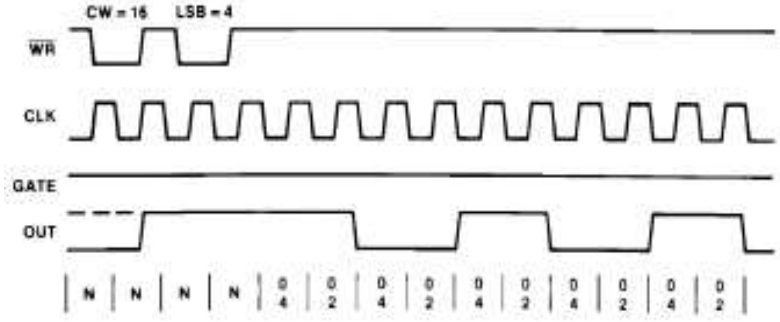


- The counter in this mode acts as divide-by-n counter.
- The OUT will be initially high.
- When the COUNT has decremented to 1, OUT goes low for one clock pulse.
- OUT then goes high again, the counter reloads the initial count and the process is repeated.

AllAbtEngg Android Application for Anna University, Polytechnic & School

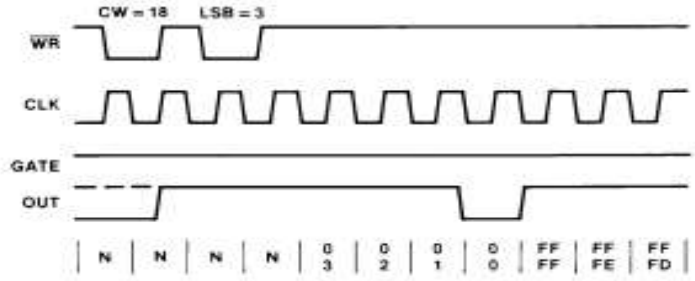
- When GATE input is low it disables counting.
- When the GATE input is high, the counter starts from the initial count.

Mode 3: Square Wave Generator



- OUT remains high for the first half of the count and goes low for the other half, thus generating the programmable square wave at OUT.
- Suppose n is the number loaded into the counter, then the OUT will be
- High for n/2 counts and low for n/2 counts if n=even
- High for (n+1)/2 counts and low for (n-1)/2 counts if n=odd.

Mode 4 : Software Triggered Strobe



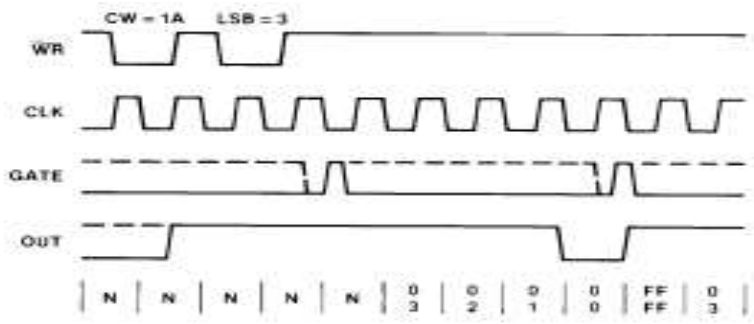
- The OUT is initially high and goes low for one clock period on the terminal count.
- When the GATE input is high counting is enabled and when it is low the counting is disabled.
- The OUT pulse is generated only once after N clock pulses

.

Mode 5 : Hardware Triggered Strobe

- The OUT is initially high
- The counter will start counting after the rising edge of the GATE input and the OUT will go low for one clock period when the terminal count is reached.
- The hardware circuit should trigger the GATE input to initiate the counting operation.

AllAbtEngg Android Application for Anna University, Polytechnic & School



3.6.3 Programming the 8253

The control words are used to program the counters with the desired mode and quantirty information. The format of control word register is as follows:

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD

SC1 , SC0:SC1 and SC0 bits select any one of the three counters.

SC1	SC0	
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Counter 3

RW1, RW0: Identify the Read/Write operation for the count as folllows

RW1	RW0	
0	0	Counter latching operation. Used when the programmer wants to read the contents of any counter “on the fly” i.e.without disturbing the counting
0	1	Read/Write the least significant byte only
1	0	Read/Write the most significant byte only
1	1	Read/Write the least significant byte first and then the most significant byte

M2, M1, M0: Select the mode

M2	M1	M0	
0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

BCD: Selects Binary or BCD counting

BCD = 0 => Binary counting

BCD = 1 =>BCD counting

AllAbtEngg Android Application for Anna University, Polytechnic & School

3.7 KEYBOARD/DISPLAY CONTROLLER (8279)

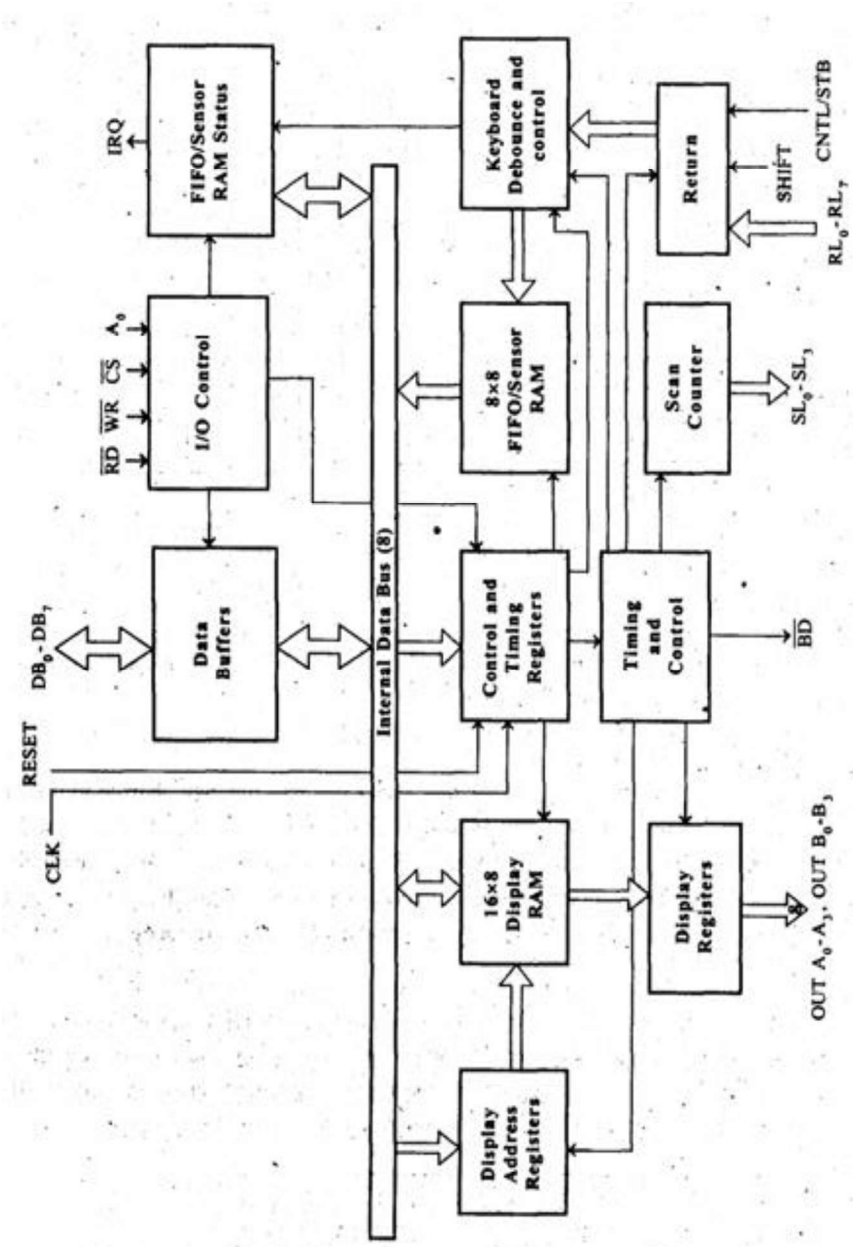


Fig 3.26: Internal block diagram of 8279

In the keyboard interface, the microprocessor scans the various keys, performs software debouncing and finally finds out the code of the depressed key. In the display interface, the microprocessor is busy in sending bit-by-bit information to the shift registers regarding the display of various segments. This puts the load on the CPU, which it may not be

AllAbtEngg Android Application for Anna University, Polytechnic & School

able to take in certain environments where the CPU has to handle a number of tasks in a very short duration.

The system designer, therefore needs an interface that can control these functions without placing the load on the CPU. The 8279 provides these function for the microprocessor. It has four main section:

1.CPU interface and control section
2.Scan section
3.Keyboard section
4.Display section

3.7.1.CPU interface and control section:

It consists of

a) Data buffers
b) I/O control
c) Control and timing registers
d) Timing and control logic

Data buffers

It is a 8-bit bidirectional buffer. It is used to connect the internal data bus and external data bus.

I/O control

- I/O control section uses the A0, $\overline{RD}, \overline{RD}, \overline{WR}$ and $\overline{CS}$ signal to control the data flow.
- The data flow is enabled by $\overline{CS}$ = 0, otherwise it is in high impedance state.
- A0=0 means data is transferred.
- A0=1 means status or command word is transferred.

***Control and timing register*:**

Store the keyboard and display modes and other operating condition programmed by the CPU.

Timing and control:

It consists of a control chain. First counter is divided by N prescalar that can be programmed to give an internal frequency of 100KHz.

3.7.2 Scan Section

It has twomodes.

1. Encoded mode
2. Decoded mode.

Encoded mode:

In **encoded scan** three scan lines SL2-SL0 is given to a 3:8 decoder and the 8 lines obtained from the decoder are connected to the rows and the return lines RL0-RL7 are connected to the columns. This enables the interfacing of 8 x 8 keyboard.

Decoded scan

In **decoded scan** four scan lines SL3-SL0 are interfaced with the rows and the return lines RL0-RL7 are connected to the columns. This enables the interfacing of 4 x 8 keyboard.

AllAbtEngg Android Application for Anna University, Polytechnic & School

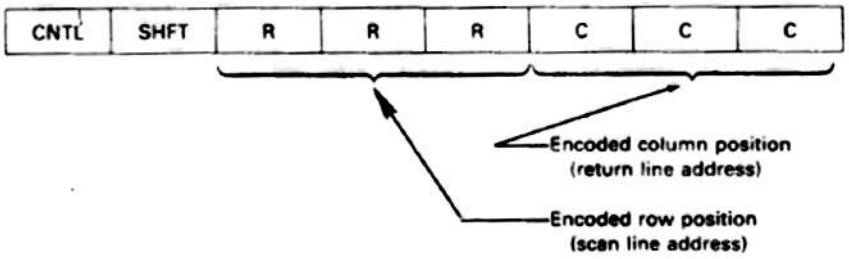
3.7.3 Keyboard section

It consists of

1. Return buffers
2. Keyboard debounce control
3. FIFO/sensor RAM
4. FIFO/sensor RAM status

Return buffers and keyboard debounce and control:

If a key closure is detected the key debounce unit debounces the key entry(ie. Wait for 10 ms). After the debounce period if the key continues to be detected the code of the key is directly transferred to the sensor RAM along with shift and control key status.



FIFO/ Sensor RAM and status logic:

In keyboard or strobed input mode, this block acts as 8-byte FIFO RAM. Each key code of the pressed key s entered in the order of the entry and in the meantime read by the CPU, till the RAM becomes empty.

The status logic generates an interrupt after each FIFO read operation till the FIFO is empty. In scanned sensor matrix mode, this unit acts as sensor RAM. Each row of sensor RAM is loaded with the status of the corresponding rows of sensors in the matrix. If a sensor changes its state, IRQ line goes high to interrupt the CPU.

3.7.4 Display section

It consists of display address register and display RAM

Display address register and display RAM:

The display address register holds the address of the word currently being written or read by the CPU to or from the display RAM. The contents of the registers are automatically updated by 8279 to accept the next data entry by CPU.

3.7.5 Modes of operation:

1. Input (keyboard) mode
2. Output(display) mode

Input (Keyboard) mode:

8279 provides three input modes, they are :

1. Scanned Keyboard Mode :
2. Scanned Sensor Matrix:
3. Strobed Input

AllAbtEngg Android Application for Anna University, Polytechnic & School

1. Scanned Keyboard Mode :

This mode allows a key matrix to be interfaced using either encoded or decoded scans. In the encoded scan, an 8 x 8 keyboard or in decoded scan , a 4 x 8 Keyboard can be interfaced. The code of key pressed with SHIFT and CONTROL status is stored into the FIFO RAM.

2. Scanned Sensor Matrix:

In this mode, a sensor array can be interfaced with 8279 using either encoder or decoder scans. With encoder scan 8 x 8 sensor matrix or with decoder scan 4 x 8 sensor matrix can be interfaced . The sensor codes are stored in the CPU addressable sensor RAM.

3. StrobedInput :

In this mode, if the control line goes low, the data on return lines, is stored in the FIFO byte by byte.

Output (Display) Modes :

8279 provides two output modes for selecting the display options.

1. Display Scan
2. Display Entry

1. ***Display Scan:***

In this mode, 8279 provides 8 or 16 character multiplexed displays those can be organized as dual 4-bit or single 8-bit display units.

2. ***Display Entry:***

The Display data is entered for display either from the right side or from the left side.

3.7.6 Software Commands:

The following commands are used to program the 8279 operating modes to perform read/write, to perform display blanking, etc. 8279 differentiates between the data and command by sensing the A0 input line. If A0 is high, the input byte is taken as command otherwise as data.

Keyboard Display Mode Set - Specifies the input and display modes and is used to initialize the 8279. Its format is:

MSB LSB

0	0	0	D	D	K	K	K

DD

00 – 8 – 8 bit character display – Left entry
01 – 16 – 8 bit character display – Left entry
10 – 8 – 8 bit character display – Right entry
11 –16 – 8 bit character display – Right entry

KKK

000 – Encoded scan keyboard – 2 key lockout
001 – Decoded scan keyboard – 2 key lockout
010 – Encoded scan keyboard – N key rollover
011 – Decoded scan keyboard – N key rollover
100 – Encoded scan sensor matrix
101 – Decoded scan sensor matrix

AllAbtEngg Android Application for Anna University, Polytechnic & School

110 – Strobed input Encoded display scan

111 – Strobed input decoded display scan

Read FIFO Sensor Memory - Specifies that a read from the data buffer register will input a byte from the FIFO memory and, if the 8279 is in the sensor mode, it indicates which row is .to be read. This command is required before inputting data from the FIFO memory. Its format is:

MSB							LSB
0	1	0	AI	X	A	A	A

- AI is the autoincrement flag for sensor RAM. If AI=1 the row select counter will be incremented after each read.
- AAA row number in FIFO/sensor RAM which is being read.

Write to Display Memory - Indicates that a write to the data buffer register will put data in the display memory. This command must be given before the CPU can send the characters to be displayed to the 8279. Its format is:

MSB							LSB
1	0	0	AI	A	A	A	A

- AI is the autoincrement bit. If AI = 1 the address will be incremented by 1 for each write
- AAAA is the address of the location in the display memory where the data for the next write will be stored

3.8 INTERRUPT CONTROLLER (8259)

Functional Description:

The 8259 A has eight interrupt request inputs, TR2 IR0. The 8259 A uses its INT output to interrupt the 8085A via INTR pin. The 8259A receives interrupt acknowledge pulses from the microprocessor at its $\overline{\text{INTA}}$ input. Vector address used by the 8085 A to transfer control to the service subroutine of the interrupting device, is provided by the 8259 A on the data bus. The 8259A is a programmable device that must be initialized by command words sent by the. After initialization the 8259 A mode of operation can be changed by operation command words.

The descriptions of various blocks are,

Data bus buffer:

This 3- state, bidirectional 8-bit buffer is used to interface the 8259A to the system data bus. Control words and status information are transferred through the data bus buffer.

Read/Write & control logic:

The function of this block is to accept OUTPUT commands from the CPU. It contains the initialization command word (ICW) register and operation command word (OCW) register which store the various control formats for device operation. This function block also allows the status of 8159A to be transferred to the data bus.

AllAbtEngg Android Application for Anna University, Polytechnic & School

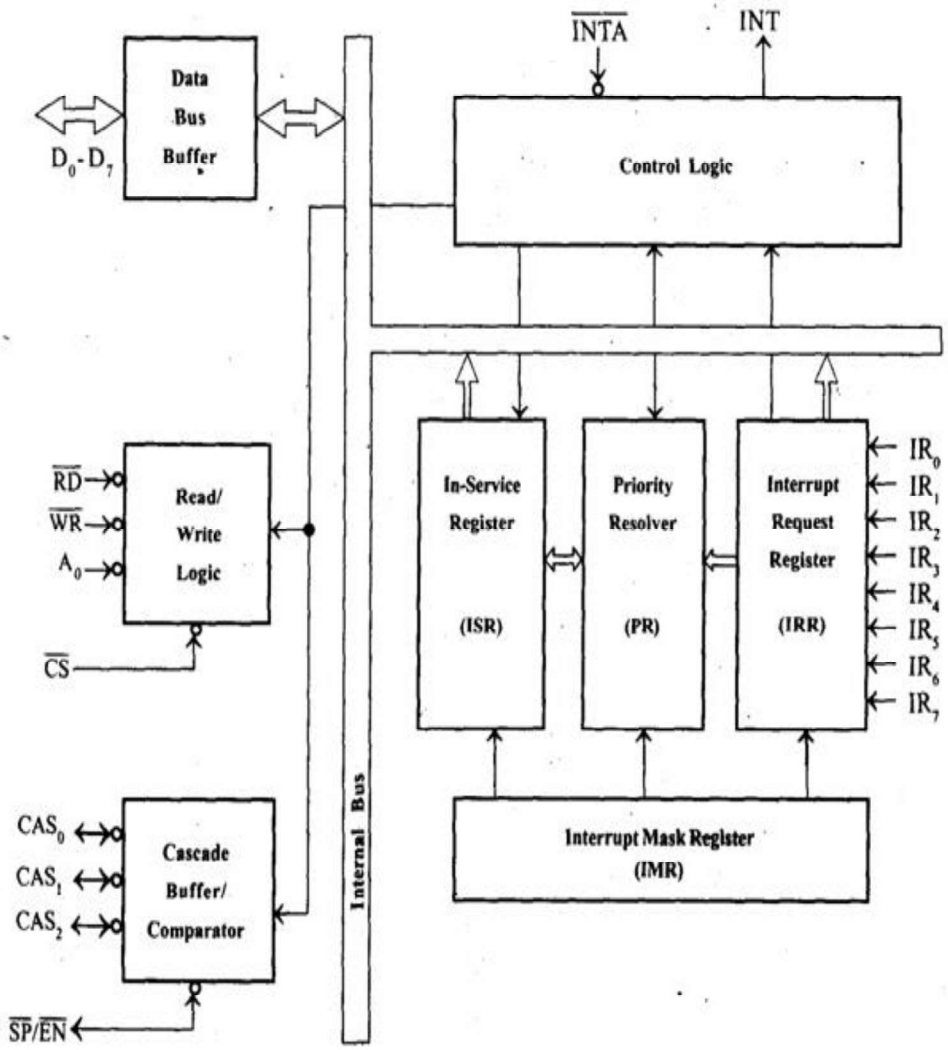


Fig 3.27: Internal block diagram of 8259

Interrupt request register (IRR):

IRR stores all the interrupt inputs that are requesting service. Basically, it keeps track of which interrupt inputs are asking for service. If an interrupt input is unmasked, and has an interrupt signal on it, then the corresponding bit in the IRR will be set.

Interrupt mask register (IMR):

The IMR is used to disable (Mask) or enable (Unmask) individual interrupt inputs. Each bit in this register corresponds to the interrupt input with the same number. The IMR operation on the IRR. Masking of higher priority input will not affect the interrupt request lines of lower priority. To unmask any interrupt the corresponding bit is set '0'.

AllAbtEngg Android Application for Anna University, Polytechnic & School

In service register (ISR):

The in service registers keeps tracks of which interrupt inputs are currently being serviced. For each input that is currently being serviced the corresponding bit will be set in the in service register. Each of these 3-reg can be read as status reg.

Priority Resolver:

This logic block determines the priorities of the set in the IRR. The highest priority is selected and strobed into the corresponding bit of the ISR during $\overline{\text{INTA}}$ pulse.

Cascade buffer/comparator:

This function blocks stores and compare the IDS of all 8259A's in the reg. The associated 3-I/O pins (CAS0-CAS2) are outputs when 8259A is used a master. Master and are inputs when 8259A is used as a slave. As a master, the 8259A sends the ID of the interrupting slave device onto the cas2-cas0. The slave thus selected will send its pre-programmed subroutine address on to the data bus during the next one or two successive $\overline{\text{INTA}}$ pulses.

Command words:

The command words of 8259are classified into two groups:

1. Initialization command words(ICW)
2. Operation command words(OCW)

Initialization command words(ICW)

Before it starts functioning, 8259 must be initialized by writing two or four command words into the respective command word register. These are called initialized command words.

Operation command words(OCW)

Once 8259 is initialized using the command word for initialization it is ready for the normal function, i.e., accepting the interrupts, but 8259 has its own way of handling the received interrupts called as modes of operation. These modes of operation can be selected by programming, i.e., by writing three internal registers called as operation command word.

Interrupt sequence in 8086 system:

1. One or more IR lines are raised high that sets the corresponding IRR bits.
2. 8259 resolves the priority and sends an INT signal to the CPU.
3. The CPU acknowledges with the INTA pulse.
4. Upon receiving an INTA signal from the CPU, the highest priority ISR bit is set and the corresponding IRR bit is reset.
5. 8086 will initiate a second INTA pulse. During this period 8259 releases an 8-bit pointer on the data bus from where it is read by the CPU.
6. This completes the interrupt cycle. The ISR is reset at the end of the second INTA pulse if automatic end of interrupt (AEOI) mode is programmed. Otherwise ISR bit remain set until an appropriate EOI command is issued at the end of interrupt subroutine.

Operating modes:

1. Fully nested mode
2. Specific mask mode
3. Buffered mode

AllAbtEngg Android Application for Anna University, Polytechnic & School

3.9 DMA CONTROLLERS

The DMA mode of data transfer is the fastest among all the modes of data transfer. In this mode, the device may transfer data directly from/to memory without any interface from the CPU.

The device requests the CPU through a DMA controller to hold its address, data and control bus so that the device may transfer data directly from/to memory. The DMA data transfer is initiated only after receiving HLDA signal from the CPU. For facilitation DMA type of data transfer between devices, DMA controller may be used.

3.9.1 Block diagram of 8257:

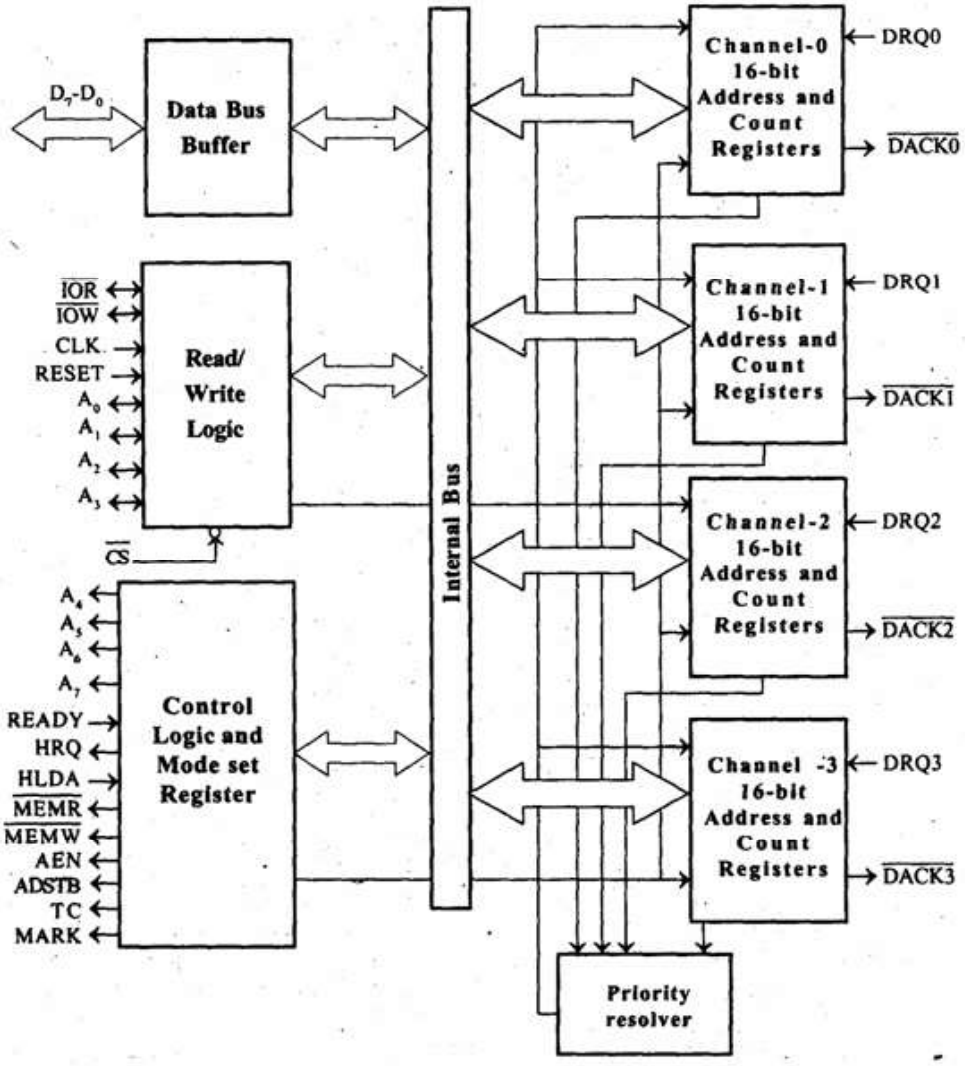


Fig 3.28: Internal block diagram of 8257

AllAbtEngg Android Application for Anna University, Polytechnic & School

It is a four channel DMA controller designed to be interfaced with their family of microprocessor. 8257 requests the CPU for bus access using local bus request input, i.e, HOLD in minimum mode and $\overline{RQ}/\overline{GT}$ pin is used as bus request input in maximum mode.

After receiving the HLDA signal in minimum mode or RQ/GT in maximum mode from the CPU, the requesting device gets the access of the bus and it completes the required number of DMA cycles for data transfer and hands over the control of bus back to the CPU.

3.9.2 Register organization of 8257

1. DMA address register
2. Terminal count register
3. Mode set register
4. Status register

DMA Address register:

Each DMA channel has one DMA address register and it is used to store the address of starting memory location.

The starting address of the memory block will be accessed by the device, it is loaded in the DMA address register of the channel.

If the device wants to transfer data over a DMA channel then it will access the block of memory with the starting address stored in DMA address register.

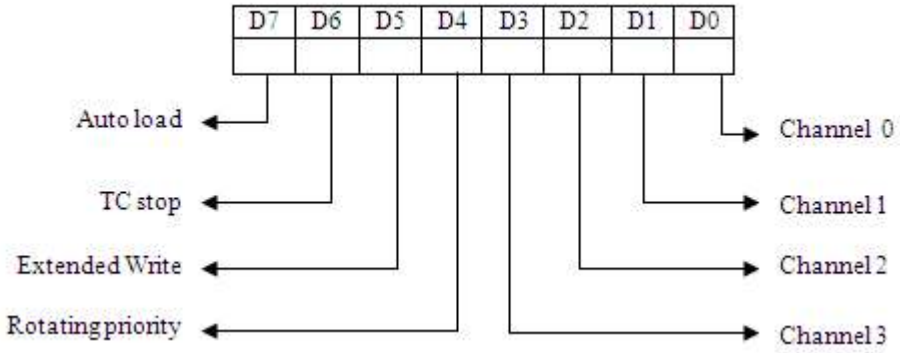
Terminal count register:

Each of the four DMA channels of 8257 has one terminal count (TC) register. This 16 bit register is used to hold the required number of DMA cycles.

After each DMA cycle, the TC content will be decremented by one and finally it becomes zero after the required number of cycles are over.

The bits 14 and 15 of the TC register indicate the type of DMA operation. At the time of write operation bit 14 is set to one and bit 15 will be set to zero.

Mode set register:



The mode set register is used for programming 8257 as per the requirements of the system. The mode set register is used to enable the DMA channel individually and also set the various modes of operation.

The DMA channel should not be enabled till the DMA address register and the terminal count register contain valid information. Otherwise an unwanted DMA request may initiate a DMA cycle, it destroys the valid memory data.

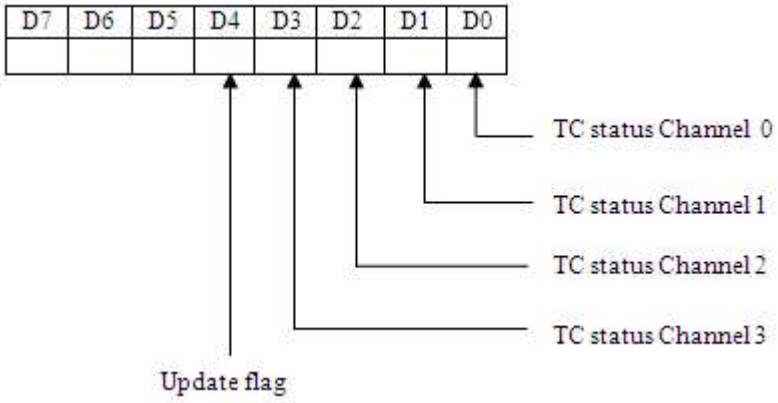
AllAbtEngg Android Application for Anna University, Polytechnic & School

- D0-D3 is used to enable one of the four DMA channels of 8257.
- If the TC bit is 1, the selected channel is disabled after the terminal count condition is reached and it further prevents any DMA cycle on the terminal.
- If the TC bit is 0 then the channel is not disabled even after the count reaches zero and further requests are allowed on the same channel.
- If the autoload bit is 1, it enables channel 2 foe repeat block chaining operations.
- If the extended write bit is 1, it extends the duration of $\overline{MEMW}$ and $\overline{IOW}$ signals by activating them earlier.

Status register:

The lower order 4 bits of this register contain the terminal count status for the individual channels. If any of these bits is set, it indicates that the specified channel has reached the terminal count condition.

The update flag is not accepted by read operation and it is cleared by resetting 8257. If the update flag is set. The contents of the channel register are reloaded to the corresponding registers of channel 2.



The update flag is set every time, the channel 2 registers are loaded with the contents of channel 3 register.

Each channel may be put in any of the four modes. The four possible modes are:

1. **Single Transfer Mode (01)**

After each transfer the controller will release the bus to the processor for at least one bus cycle, but will immediately begin testing for DREQ inputs and proceed to steal another cycle as soon as a DREQ line becomes active.

2. **Block Transfer Mode (10)**

DREQ need only be active until DACK becomes active, after which the bus is not released until the entire block of data has been transferred.

3. **Demand Transfer Mode (00)**

This mode is similar to the block mode except that DREQ is tested after each transfer. If DREQ is inactive, transfers are suspended until DREQ once again becomes active, at which

AllAbtEngg Android Application for Anna University, Polytechnic & School

time the block transfer continues from the point at which it was suspended. This allows the interface to stop the transfer in the event that its device cannot keep up.

4. Cascade Mode (11)

In this mode 8237s may be cascaded so that more than four channels can be included in the DMA subsystem. In cascading the controllers, those in the second level are connected to those in the first level by joining HRQ to DREQ and HLDA to DACK. To conserve space, this mode will not be considered further.

3.10 TRAFFIC LIGHT CONTROL

The traffic control sequence is as follows:

(i) Allow traffic from the North to the other directions.
(ii) Allow traffic from the East to the other directions.
(iii) Allow traffic from the South to the other directions.
(iv) Allow traffic from the West to the other directions

The traffic signal scheme for a four-road junction is shown in Fig. 3.29

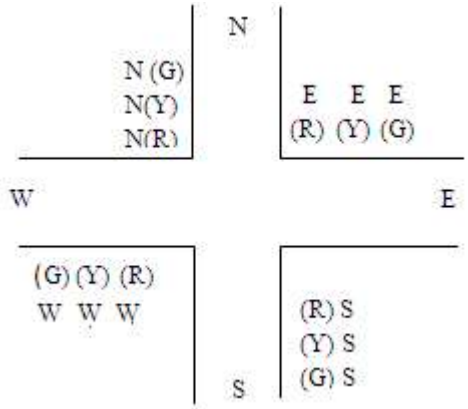


Fig 3.29: Traffic control signal scheme for a four-road junction

The red, yellow, and green signals have to be turned on in the proper sequence. The sub-sequences to be followed are listed in fig 3.30, along with the 8255 port pins used to control these lights.

AllAbtEngg Android Application for Anna University, Polytechnic & School

Sequence	PB7	PB6	PB5 GE	PB4 YE	PB3 RE	PB2 GW	PB1 YW	PB0 RW	PA7	PA6	PA5 GS	PA4 YS	PA3 RS	PA2 GN	PA1 YN	PA0 RW	Port B	Port A
North to other sides (green)	X	X	0	0	1	0	0	1	X	X	0	0	1	1	0	0	09H	0C H
North to other sides (yellow)	X	X	0	1	0	0	0	1	X	X	0	0	1	0	1	0	11H	0A H
East to other sides (green)	X	X	1	0	0	0	0	1	X	X	0	0	1	0	0	1	21H	09 H
East to other sides (yellow)	x	x	0	1	0	0	0	1	x	x	1	0	0	0	0	1	11H	11 H
South to other sides (green)	X	X	0	0	1	0	0	1	X	X	1	0	0	0	0	1	09H	21 H
South to other sides (yellow)	X	X	0	0	1	0	1	0	X	X	0	1	0	0	0	1	0AH	11 H
West to other sides (green)	X	X	0	0	1	1	0	0	X	X	0	0	1	0	0	1	0CH	09 H
West to other sides (yellow)	X	X	0	0	1	0	1	0	X	X	0	0	1	0	1	0	0AH	0A H

Fig 3.30: Traffic signal sub-sequences for a four-road junction

Instead of using LEDs, actual lights at 230 V can be used in this microprocessor-based system. The circuit diagram for interfacing 230V bulbs through relays is shown in Fig 3.31.

AllAbtEngg Android Application for Anna University, Polytechnic & School

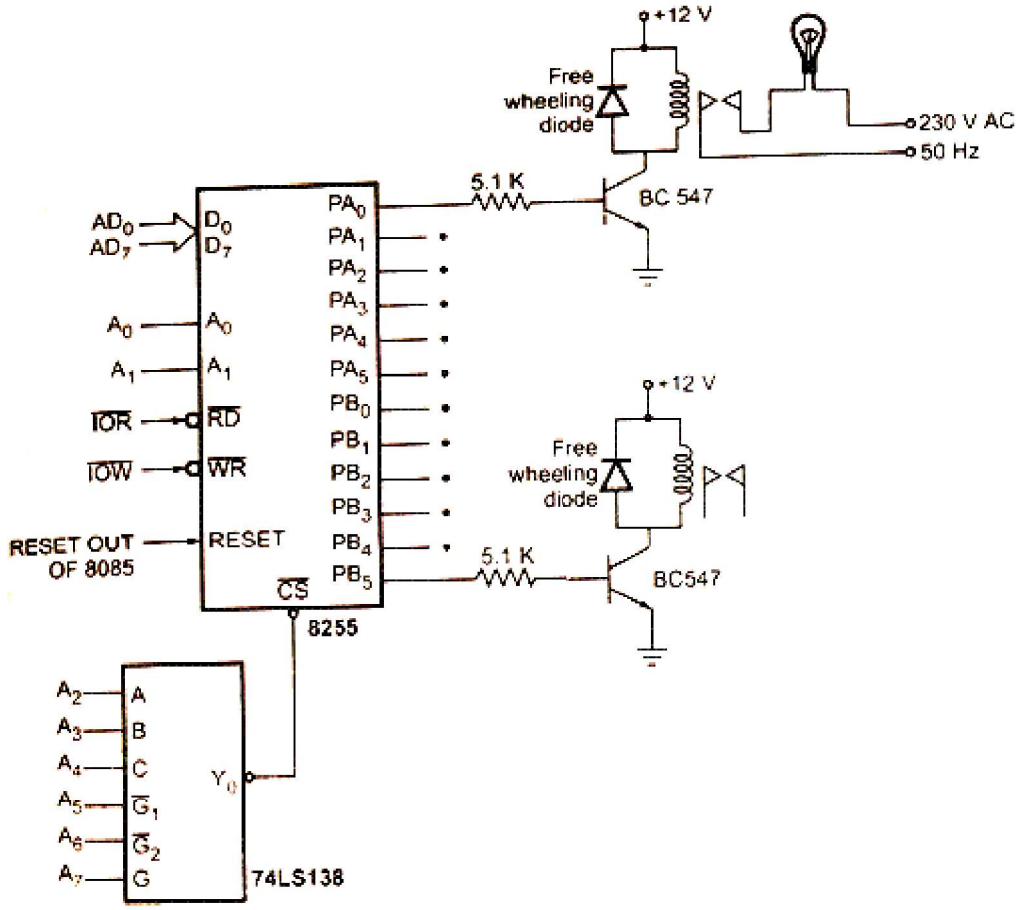


Fig 3.31: Traffic light control interface diagram

In this example, the port A and port B pins of the 8255 are assumed to control the traffic lights. The individual pins of these ports are connected to the different red, yellow and green lights. For example the red light facing the North direction is connected to the Port A pin PA0.

Source program:

This program assumes that the 8255 IC is interfaced to the microprocessor at the addresses 80H, 81H, 82H and 83H. The data for the port is available at the address 9000H. The port data for ports A and B are stored consecutively in the memory locations starting at 9000H.

Labels	Mnemonics	Comments
START:	MVI A,80H	;Initialize 8255 port A and port B
	OUT 83H(CR)	;in output mode
LOOP:	MVI C, 08H	;Initialize a count for eight sequences
	LXI H,9000H	;Initialize a pointer for output data
NEXT:	MOV A,M	;Get the data for one group
	OUT 80H	;Send it to port A
	INX H	;Point to the next data

AllAbtEngg Android Application for Anna University, Polytechnic & School

```
                MOV A,M         ;Get the data
                OUT 81H         ;Send it to port B
                CALL DELAY      ;Wait for a predefined delayed period
                INX H           ;Point to the next data
                DCR C           ;Reduce the sequence count
                JNZ NEXT        ;If it is not zero, go to the next sequence
                JMP LOOP        ;If it is zero, start the sequence again
Delay subroutine
DELAY:  MVI B, COUNT1   ;Initialize register B with COUNT1
LOOP2:  LXI D,COUNT2    ; Initialize the DE register pair with COUNTZ
LOOP1:  DCX D           ;Decrement the DE pair
        MOV A,D         ;Move the content of register D to the
                        accumulator
        ORA E           ;Check whether the DE pair has become zero
        JNZ LOOP1       ;If it has not, jump to LOOP1
        DCR B           ; If it has, decrement register B
        JNZ LOOP2       ;If it is not zero, loop again
        RET             ;If it is zero, return from subroutine
```

3.11 LED DISPLAYS

Seven-segment light emitting diode displays are commonly used low-cost displays and are easiest to interface with microprocessors. Seven-segment displays consist of seven LED segments. The arrangement of the seven segments and the appearance of the ten digits is shown in Fig. 3.32.

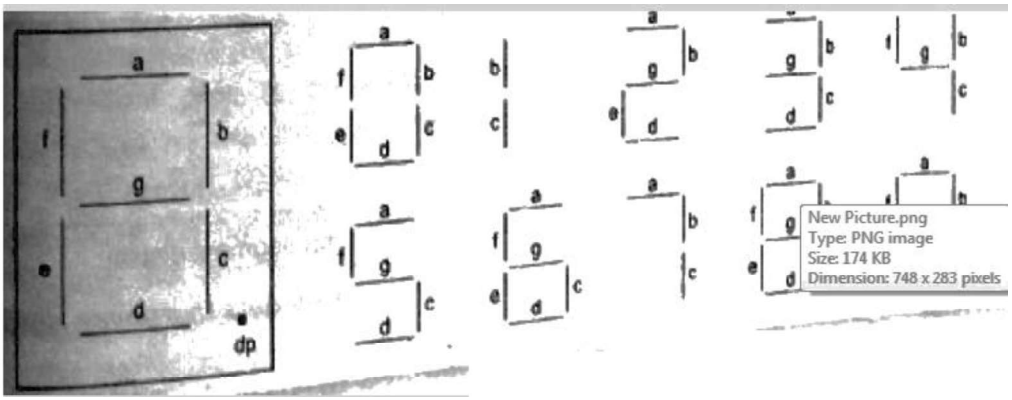


Fig 3.32: Arrangement of LEDs and appearance of digits in seven-segment displays

Seven-segment displays are available in a single dual in-line package (DIP). There is one pin for each segment and these pins are named a to f; another LED is available for decimal point(dp). In addition to these eight pins, seven-segment displays have one more pin for power supply. Seven-segment displays come in two types-common anode and common cathode.

In common anode display the anodes of all segments are connect6eed together. So to illuminate a segment, the common anode is connected to the supply and the corresponding segment input is connected to a low-level voltage or logic 0.

AllAbtEngg Android Application for Anna University, Polytechnic & School

In common cathode display the cathodes of all the LEDs are connected together. So to illuminate a segment input is connected to the high level voltage or logic 1and the common cathode is connected to the ground. This forward biases the LEDs and illuminates them.

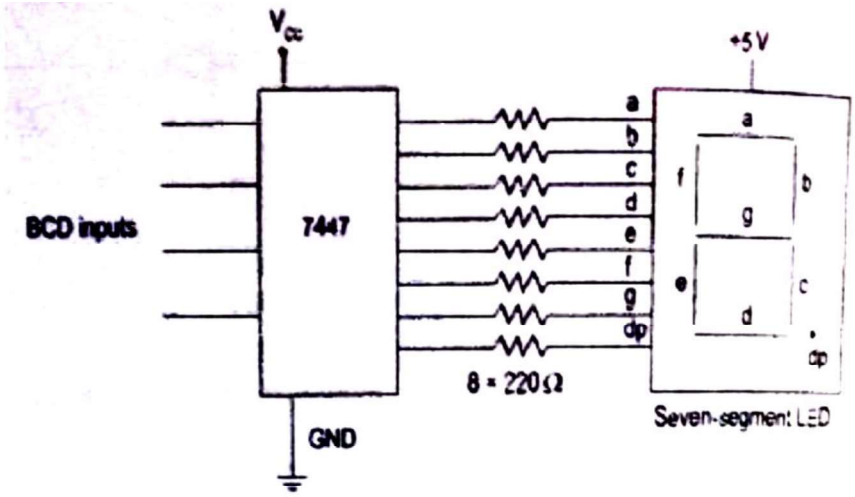


Fig 3.33: Driver circuit for single seven-segment display.

Figure 3.33 shows the circuit required to drive a single seven-segmcnt LED display from 4-bit BCD output. The BCD to seven-segment display decoder IC7447 converts the 4-bit BCD code applied at its input into the patterns required to display the BCD number. The patterns generated arc active low outputs, meaning that the output corresponding to segments that are to be illuminated is logic 0. So the common anode display is suitable for use with the 7447. The complete circuit used for interfacing the seven-segment display. along with the address bus decoder and latch. is given in Fig. 3.34

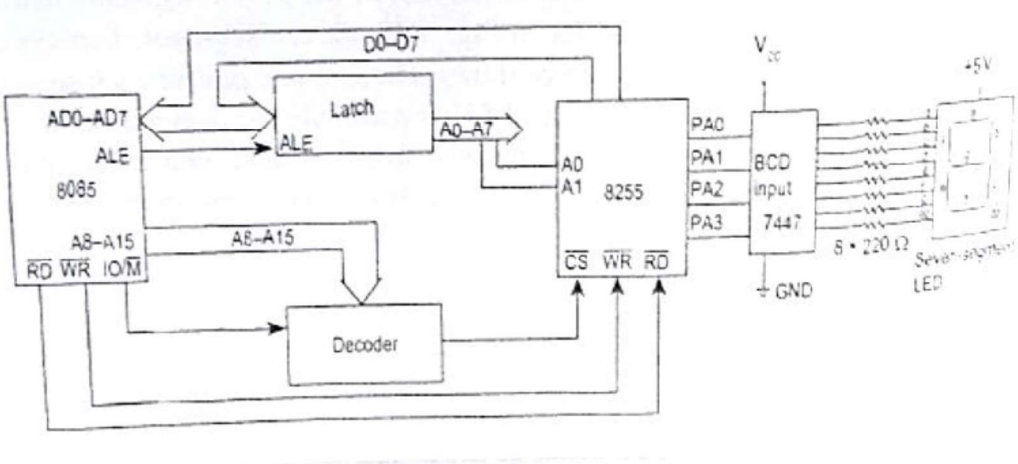


Fig 3.34: Complete circiut for interfacing single seven segment display.

AllAbtEngg Android Application for Anna University, Polytechnic & School

Fig 3.35 shows the instructions that can be used to display data in a seven segment display.

Mnemonics	Comments
MVI A, control word	;Load the accumulator with the 8255 word
OUT control_register	;Output it to the control register of the 8255
MVI A, data	;Load the accumulator with the data to be displayed
OUT PORT_A	;Output it to port A, where the display is connected

Fig 3.35 Instructions used to display in 7-segment display

Two seven segment displays can be connected to a single 8-bit port. One 7447 IC can be connected to the four lower-order bits and another 7447 can be connected to the four higher-order bits of port A. So six seven-segment displays can be connected to a single 8255 that has three parallel I/O ports. This results in a more complicated circuit. The complexity of the circuit can be reduced by using a technique called multiplexed display. By using multiplexed display, as many as eight displays can be connected to the two ports.

3.12 LCD DISPLAY

Many alphanumeric liquid crystal displays are available in the market. These displays have a built-in controller IC and a display section. These displays can be interfaced to any microprocessor or microcontroller. The data displayed in LCDs can be easily controlled and changed.

Liquid crystal displays are created by placing a thin layer of liquid crystal fluid between two glass plates. Transparent electrically conductive films are pasted on the front and back glass plates in the shape of the character to be displayed. When a voltage is applied between these films, the electric field changes the behavior of the liquid crystal. So the light is either transmitted through it or reflected by it. Hence ,the required display becomes visible.

Modern LCDs come with a controller IC and related control inputs.T hey get the ASII code of the data to be displayed and display the character in the exact location. The LCD displays come with many options such as 8-80 characters display ,one-line, two-line, or four-line display ,etc. Almost all these devices have 14 pins for interfacing with the microprocessor or microcontroller.

Three pins E,R/$\overline{W}$,ans RS are used for control and handshake signals.Eight pins are used for transferring data to the display and can be connected to the data bus of the system.Two pins are allotted for supply and ground and one pin is used for adjusting the contrast of display. The voltage applied to this pin can be varied to adjust the contrast.

LCDs contain internal RAM for storing the data to be displayed. The control signal RS given as input to the LCD indicates whether the data lines are carrying a command or a data for display. RS is made zero to indicate that a command to LCD is being sent on the data lines. It is made high to indicate that the data lines contain display data to be read or written. R/$\overline{W}$is also an input control signal given to the LCD to indicate the direction of data transfer. Enable is the control input to the LCD and must be pulsed to perform the read/write operation with the LCS.A transition from 1 to 0 on this line enables the corresponding operation, which is decided by the other control inputs.

AllAbtEngg Android Application for Anna University, Polytechnic & School

The LCD works with its own internal clock pulses. So any command or data written to the LCD must be enabled with the EN signal. This EN signal must be applied for a predefined duration. Each command and data requires between 40µs and 1.6µs,depending on the type of LCD and its clock frequency. So a separate subroutine is written to give proper control signals for the predefined delay time. Here, two subroutines-COMMAND and DISP-are used to write a command word and a data for display, respectively These two subroutines use a common delay routine.

The first step in the program is to clear the display, set the cursor to home position, and start display from there. The simple program given in table 7.24 is written to display an array of characters stored in memory locations starting from 9000H.The number of characters displayed is NUM-CHAR ;it is initialized as a count in register CALL the characters are displayed continuously.

3.13 KEYBOARD AND DISPLAY INTERFACE

3.13.1 Keyboard

A keyboard consists of number of a key switches used for entering data, event, etc. The important part of the interface between the keyboard and the microprocessor is to find out whether any key has been pressed; if yes, then the location of the key should be determined. A key, when pressed, makes the contact till it is kept pressed and the contact is broken when it is released. A key closure can be determined easily by supplying a voltage at one of the two points and finding out the voltage at the other point. For interfacing to microprocessor, one point is connected to +5 V and the other point is connected to sense the voltage

Fig 3.36: Basic keyboard operation – single key

The OUT point can be sensed as one bit of any port. In case of two key switches, the input points of the two keys can be connected together and a high voltage applied. The OUTI and OUT2 points can be connected to two separate bits of a port.

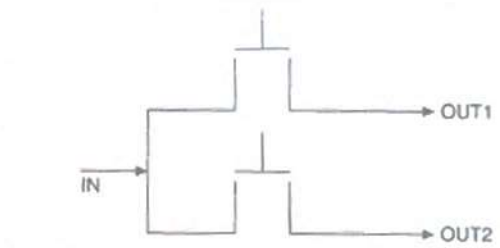


Fig 3.37: Basic keyboard operation – two keys

AllAbtEngg Android Application for Anna University, Polytechnic & School

Let us now consider the keyboard with four key switches. Extending the approach of Figure 3.37, we can interface this by connecting all the input points together and applying a high signal. The output points are connected to a port .

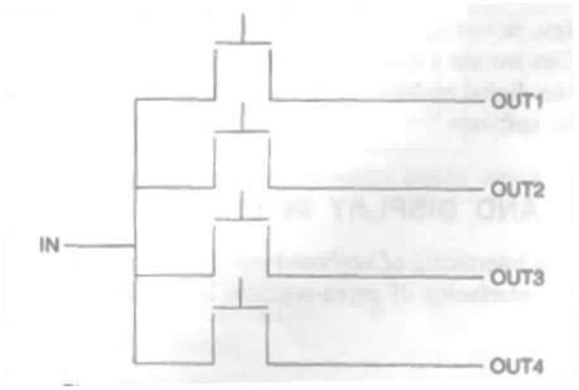


Fig 3.38: Basic keyboard operation – four keys

It is evident that this approach cannot be adopted for more number of keys. As an example, a 64-key keyboard will require eight 8-bit ports for the output points. To solve such problems, the keyboard is arranged as a matrix consisting of rows and columns. For a 4-key keyboard, Figure 3.39 shows two rows IN1 and IN2 and two columns OUT1 and OUT2. The rows are energized sequentially and the voltages in columns are sensed. In the example of Figure 3.39, if IN1 ishigh and OUT2 is high, then the key 2 of the first row has been pressed. If IN2 and OUT2 are high, then the key 2 of the second row has been pressed.

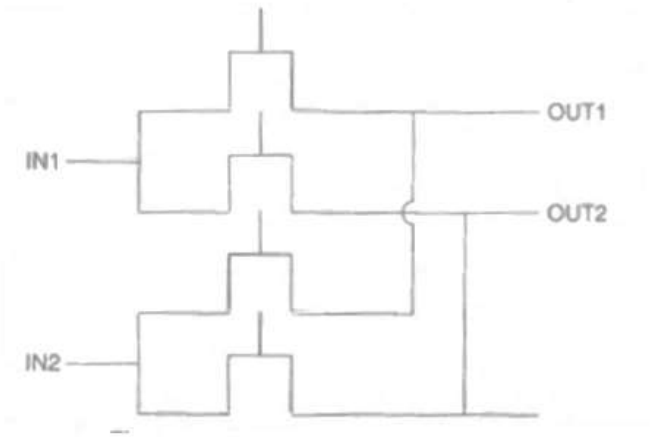


Fig 3.39: 2 X 2 keyboard operation

AllAbtEngg Android Application for Anna University, Polytechnic & School

Thus, a 64-key keyboard can be arranged as a matrixwith 8 rows and 8 columns. The interfacing will require two 8-bit ports for input and output lines. Let us assume that port 1 is connected to rows and port 2 to columns. Then the sequence of the operation is as follows:

(i) Output row enable signal on port 1 for the first row.
(ii) Input port 2 to check for key depression.
(iii) Determine the column number of the depressed key in the keyboard.
(iv) Determine the code of the key and store it in memory.

The sequence from (i) to (iv) is repeated, for all the rows in a sequence, by the microprocessor. Figure 3.40 shows an 8 x 8 keyboard interface to the microprocessor. To determine the column number of the depressed key in the keyboard, we would have to detect the bit in port 2 which is 1 and its relative position with respect to the bit corresponding to the first column. If bit 7 of port 2 corresponds to the first column, the bit 6 will then correspond to the second column and the bit 0 will correspond to the eighth column. This can be detected either by setting a mask pattern or by rotating.

A table consisting of the codes of various keys in the keyboard is stored in the microprocessor memory. Whenever a key is pressed, its code is taken from the code table and used in the program. For easy access, the code table is organized in the manner shown in Figure 3.41. The codes for the keys for rows 1 to 8 are stored sequentially. Within a row, the codes forthe columns are stored sequentially.

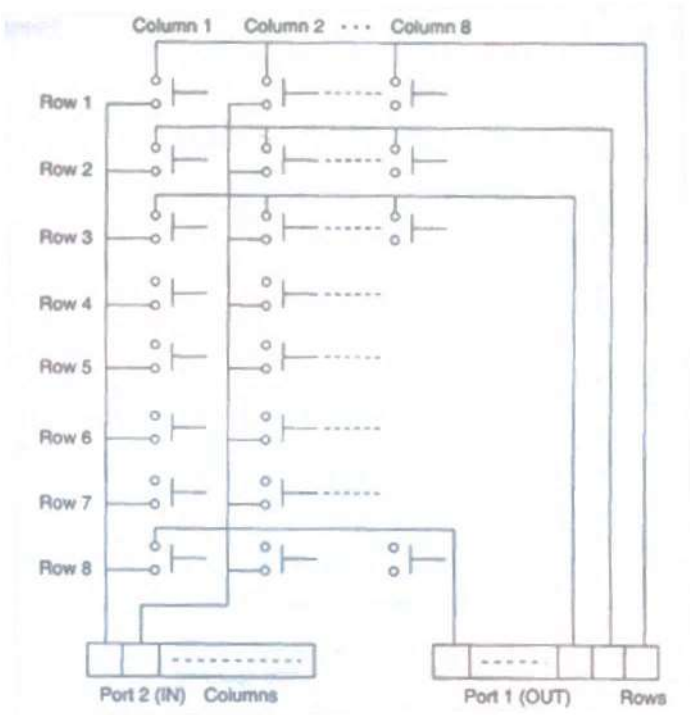


Fig 3.40: 8 X 8 keyboard interfacing with microprocessor

AllAbtEngg Android Application for Anna University, Polytechnic & School

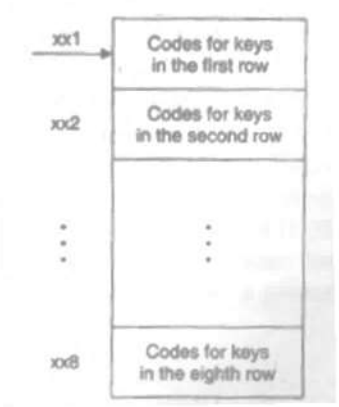


Fig 3.41: Code table for 8 X 8 keyboard

The addresses xxl, xx2, ... , xx8 are the starting addresses for the codes for the keys in the first, second, ... , eighth rows. Thus, if a key whose row number is 7 and the column number is 1 is depressed, the code of that key can be found at the address xx7. The program flowchart for the interface is shown in Figure 3.42.

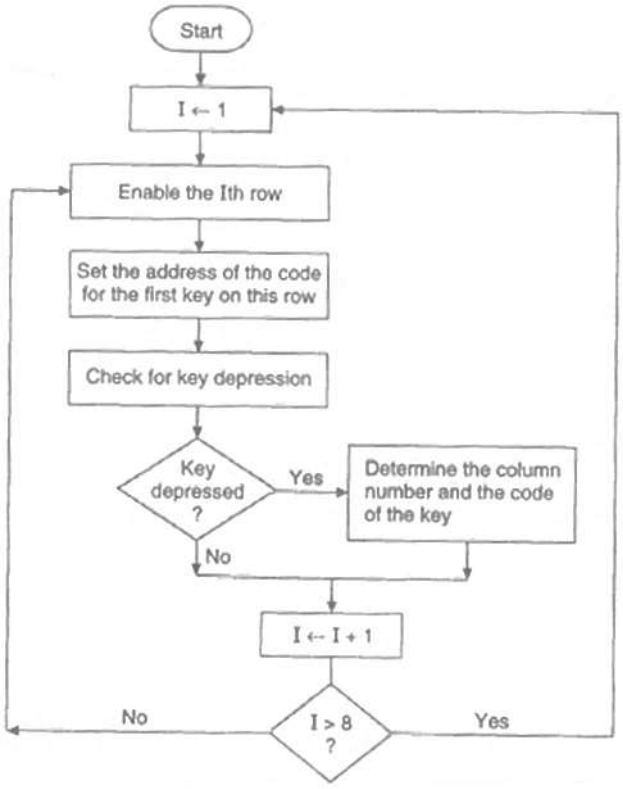


Fig 3.42: Keyboard – microprocessor interface software flowchart

AllAbtEngg Android Application for Anna University, Polytechnic & School

Key debouncing

When a key is depressed and released, the contact is not broken permanently. In fact, the key makes and breaks the contacts several times for a few milliseconds before the contact is broken permanently. If a key depression is detected by a microprocessor, it is possible that the depression may be false, i.e. it may be due to the bouncing of the key. It is, thus necessary to de bounce the key after depression. Debouncing by both hardware and software is possible. Hardware debouncing involves a circuit which ensures that the false depressions do not affect the out signal. In software debouncing, the microprocessor executes a delay routine for a few milliseconds after the key depression is detected. The same key is again checked for depression. Further action like finding the code etc. is taken, only if the key is found depressed.

3.13.2 Light Emitting Diode Display

The light emitting diodes (LEDs) are used to display limited data, results, events, or messages. The total number of characters that are displayed is fixed, unlike the CRT display where any amount of data may be displayed on the screen. The LEDs are commonly used on the front panel of instruments, digital clocks, etc. In most simple terms, the light emitting diode is a diode which when conducts, emits light energy. The conduction starts when the anode is held at a higher voltage than the cathode .

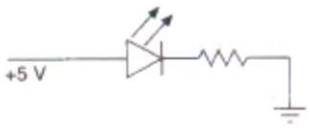


Fig 3.43: LED operation

LEDs can be directly interfaced with the microprocessor through an output port, as shown in Figures 3.44 and 3.45

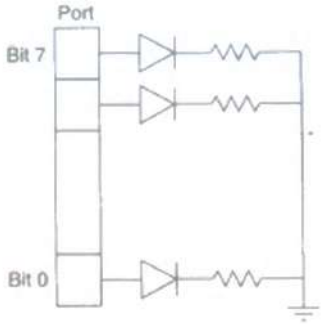


Fig 3.44: Microprocessor interface to LED (common cathode)

AllAbtEngg Android Application for Anna University, Polytechnic & School

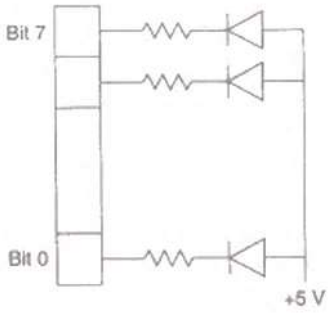


Fig 3.45: Microprocessor interface to LED (common anode)

For the interface shown in Figure 3.44, whenever a bit is 1, the corresponding LED will glow. In case of the interface shown in Figure 3.45, a 0 in any bit position will make the corresponding LED glow. So, by loading a particular bit pattern in the port, the desired LEDs may be lighted.

Seven-Segment LED

The LEDs can be arranged in the fashion shown in Figure 3.46

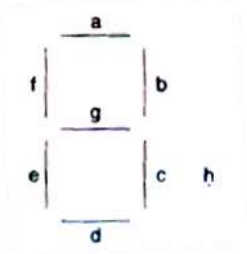


Fig 3.46: Seven-segment LED

This structure has eight segments marked a, b, c, d, e, f, g, and h and is. *very*useful in the display of the numeric and alphanumeric data. For example, to display character A, segments a, b, c, e, f and g should glow and to display character 6, segments a, c, d, e, f and g should glow (Figure 3.47). Originally, the structure had only seven segments a to g, and the eighth segment h (to display the decimal point) was added subsequently. Therefore, even though it has eight segments, the name seven-segment LED has been retained. Each segment may have more than one LED.

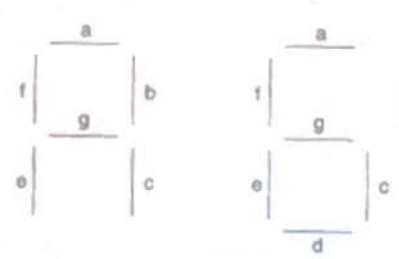


Fig 3.47: Character formation in seven-segment LEDs

AllAbtEngg Android Application for Anna University, Polytechnic & School

The On/Off information for eight segments can be arranged in one byte. The bit information can be connected to respective segments in the manner shown in Figure 3.48. Normally the LED anodes are connected to 5 V permanently and the cathodes are connected to bits as shown in Figure 3.60. Figure 3.47 shows the information for different segments

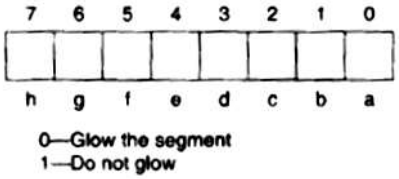


Fig 3.48: Control byte for seven-segment LEDs.

Parallel interface The parallel interface between the seven-segment LED display and the microprocessor is shown in Figure 3.49. The anodes of all LEDs are held permanently at +5 V, whereas the cathodes are connected to the port bits. The microprocessor will load any bit pattern in the Out port. Those segments, for which O is stored in the bit pattern of the code, will be lighted.

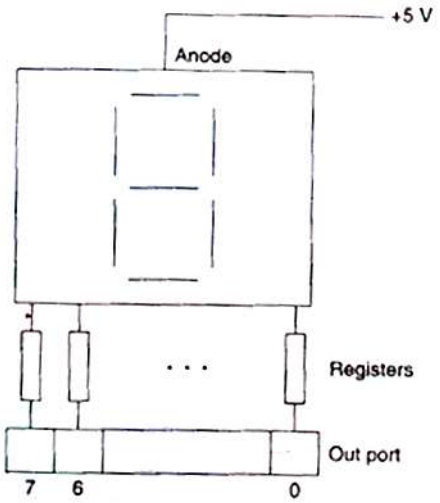


Fig 3.49: Microprocessor interface to seven-segment Led (parallel interface)

Serial interface The parallel interface, as described above, is fast, requires minimum extra hardware for interfacing and the software is also very simple. However, this is not always the case. A single seven-segment display can be used for only a single character display. However real-life situations call for a large number of seven-segment displays. This requires extra ports (number of ports = number of seven-segment displays) to be defined, i.e. extra hardware. As an example, an application requiring 64 digits to be displayed will require 64 ports to be defined, thus increasing the hardware complexity. The fast speed of parallel interface is not really required since the human eye, .due to its limitations, cannot perceive such fast changes. In fact,

AllAbtEngg Android Application for Anna University, Polytechnic & School

if the display continuously changes very fast, the eye will not be able to read anything. The serial interface which overcomes the disadvantages of the complex hardware in case of a large number of seven-segment displays is shown in Figure 3.50

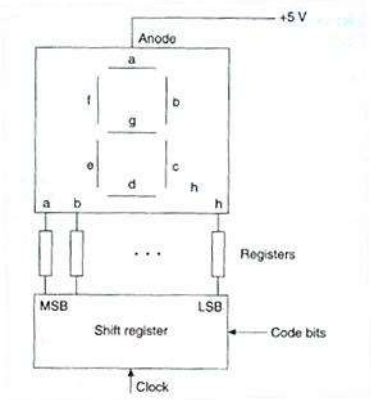


Fig 3.50: Microprocessor interface to seven segment LED (serial interface)

The flowchart in Figure 3.51 shows the operation sequence which displays a digit in a seven-segment LED.

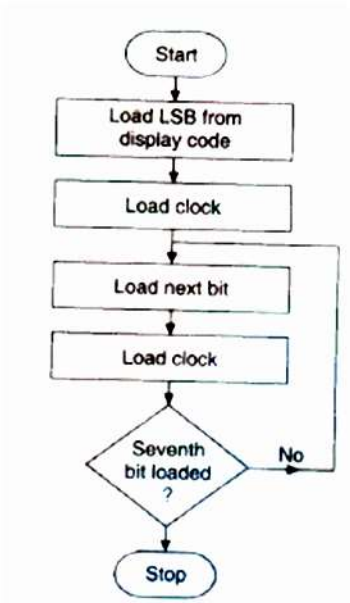


Fig 3.51: Serial interface of seven segment LED to microprocessor – software flowchart

As an example, let us assume that we want to display a character A. For character A, the segments a, b, c, e, f and g should glow. Thus, the display code for A will be

AllAbtEngg Android Application for Anna University, Polytechnic & School

7	6	5	4	3	2	1	0
1	0	0	0	1	0	0	0
h	g	f	e	d	c	b	a

The zero bits in the display code will light the corresponding segments. At the first step, LSB, i.e. the code for segment 'a' will be loaded in shift register. When the clock is loaded, the bit is passed to LSB of the shift register. When the second bit of the code (corresponding to segment 'b') is loaded in the shift register with the clock, the earlier bit shifts one step left and the later bit is included as the LSB of the shift register. The shifting occurs each time a bit is loaded. When the seventh bit has been loaded, the LSB of the display code reaches the MSB of the shift register and the character A appears. The character starts appearing as soon as the first bit is loaded in the shift register, but due to the fast execution and the limitation of the eye, the final character seems to appear instantly. Serial interface of the seven-segment LED display requires only two output lines--one for the clock and the other for the code bits. This can be achieved through a single output port. Interestingly, the increase in the number of the seven-segment displays has no effect on the number of lines. The shift registers are connected. Considering the limitations of the eye, it will still look instantaneous, except that the number of bits that will be entered are eight times the number of seven-segment LED displays interfaced. To display a character in display D0 in Figure 3.52, 32 clock pulses will be required. However, the users will soon realize that a large number of displays can be easily interfaced, without any burden on the speed, by the serial interface technique.

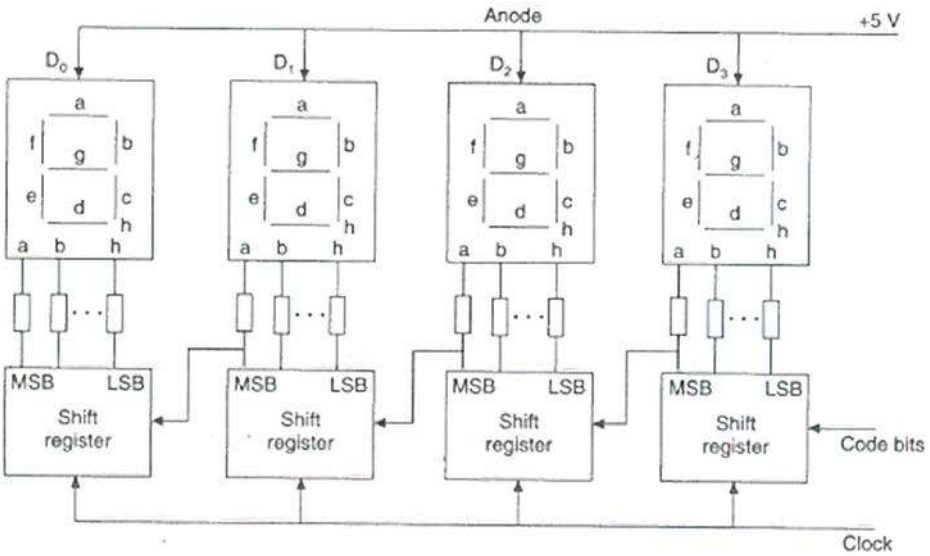


Fig 3.52: Cascading of seven-segment LEDs in a serial microprocessor interface

The keyboard and the display are very important parts of any system. A system may have a few push-button keys or a large keyboard. Similarly, simple LEDs on the front panel as indicators to large displays containing seven-segment LEDs are also possible.

AllAbtEngg Android Application for Anna University, Polytechnic & School

The 8086 Interfacing to Keyboard and Display

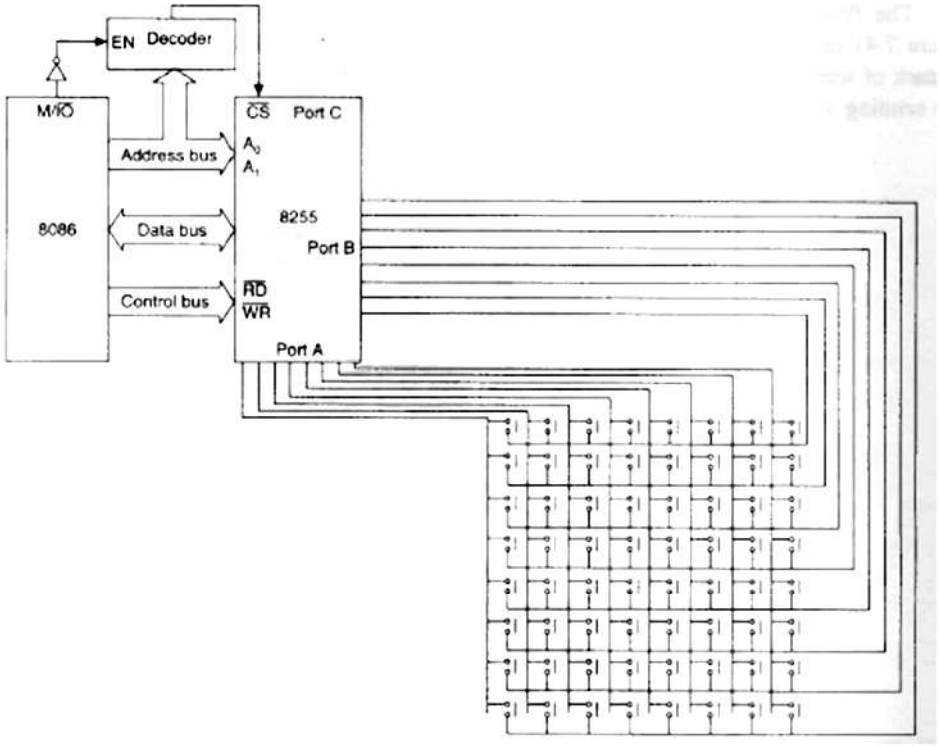


Fig 3.53: Keyboard interfacing to the 8086 using the 8255 PPI

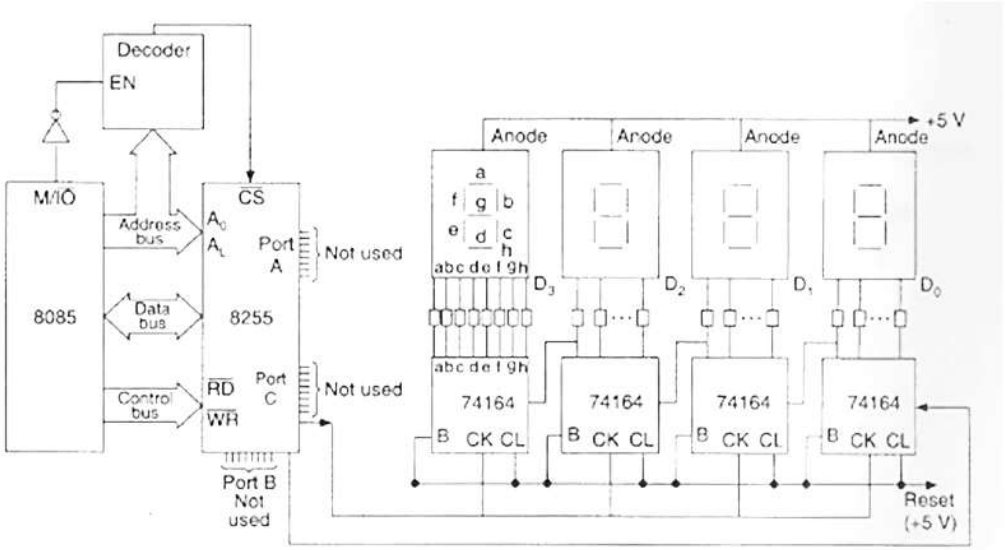


Fig 3.54: Seven-segment LED display interface to the 8086 using the 8255 PPI.

AllAbtEngg Android Application for Anna University, Polytechnic & School

The interfacing of the 8086 with an 8 x 8 key keyboard is shown in Figure 3.53. The 8086 is being used in minimum mode. Port A of the 8255 PPI is being used for the columns, whereas Port B is being used for the rows. The 8086 will determine the code of the depressed key using the look-up table stored in its memory and only then it will initiate the action. The interfacing of the seven-segment LED display (4 in number) in serial mode cascaded manner is shown in Figure 3.54. Only two port lines of the 8255 have been used for clock and data. Other lines are free for interfacing with other circuits of the system.

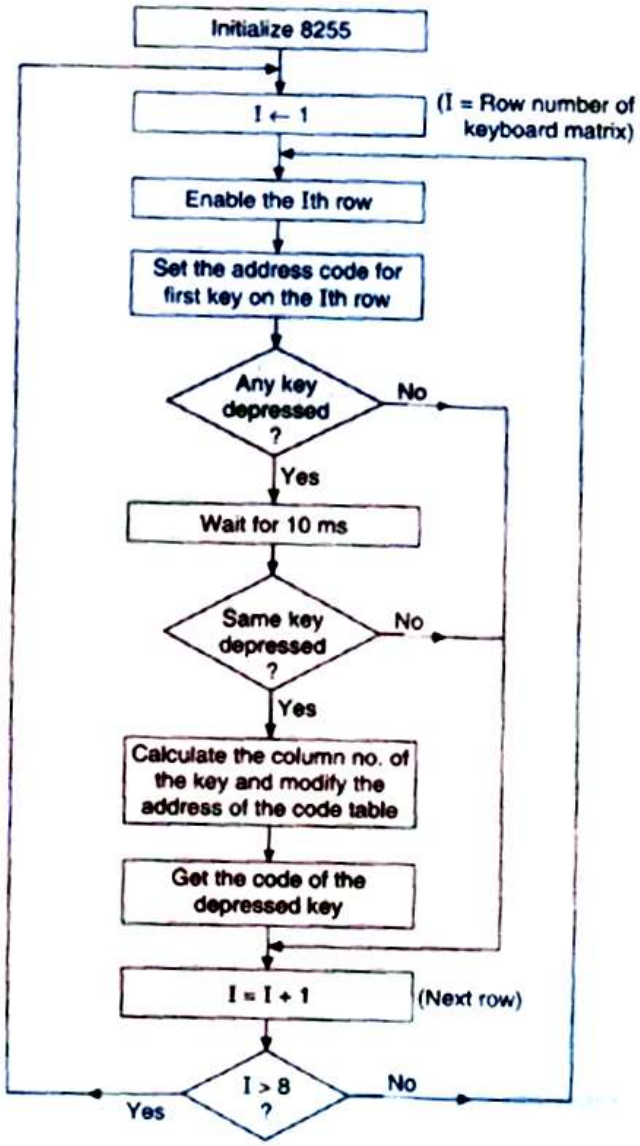


Fig 3.55: Flowchart for the 8 X 8 keyboard interface

AllAbtEngg Android Application for Anna University, Polytechnic & School

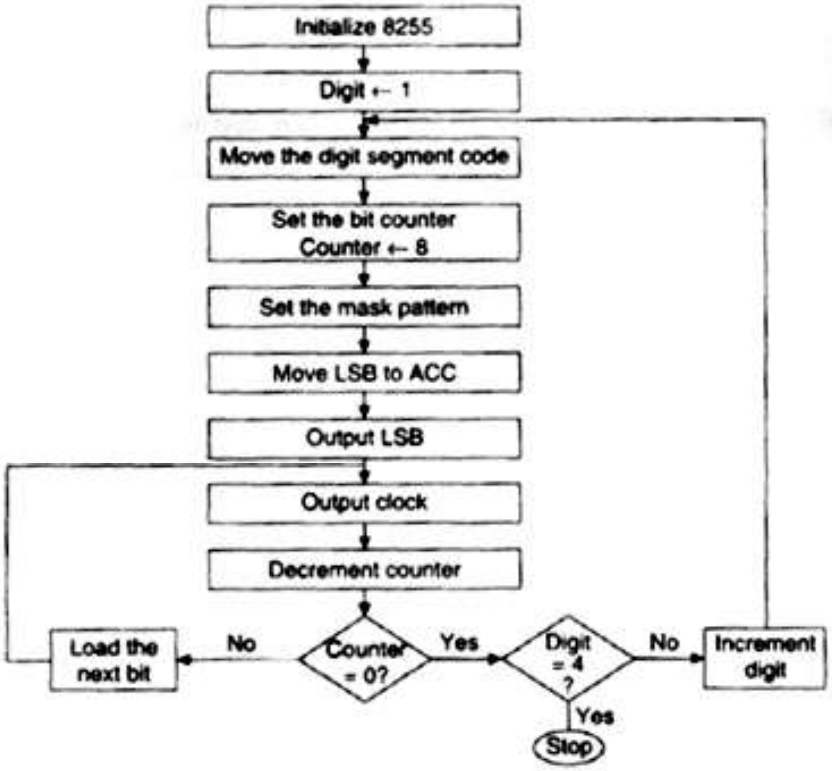


Fig 3.56: Flowchart for the seven-segment LED interface

The flowchart for the keyboard and the display interfaces is shown in Figure 3.55 and Figure 3.56 respectively. The software is resident in the microprocessor memory and performs the task of scanning, debouncing and finding the code for the key in case of keyboard interface, and sending various data bits and clock in case of display interface

3.14 ALARM CONTROLLER

To design a pre-settable alarm system use thumb wheel switches to accept 4 digit values in seconds. Thumb wheel switches are interfaced through 8255 ports. Timing parameter are derived from the 8253/8254. 74LS138 decoder is used to generate chip select signals for 8253/8254 and 8255. One more 74LS138 decoder is used to generate $\overline{IOR}$, $\overline{IOW}$, $\overline{MEMR}$ and $\overline{MEMW}$ signals.

Counter 0 of 8253/8254 is programmed in mode 0 to give the presettable time period and counter 2 is programmed in mode 0 to give a delay of 5 seconds. Clock input for 8253/8254 is 10 KHz count required to get 1 second time interval is,

$$Count = \frac{Requiedperiod}{Clockperiod} = \frac{1\,sec}{100\,s} = 1000 = 2710H$$

This count value is loaded in the count register of the counter 1 and a counter 1 is programmed in mode2 to generate square wave with frequency 1 Hz. The output of counter 1 is fed to the clock input of counter 0 and counter 2.

AllAbtEngg Android Application for Anna University, Polytechnic & School

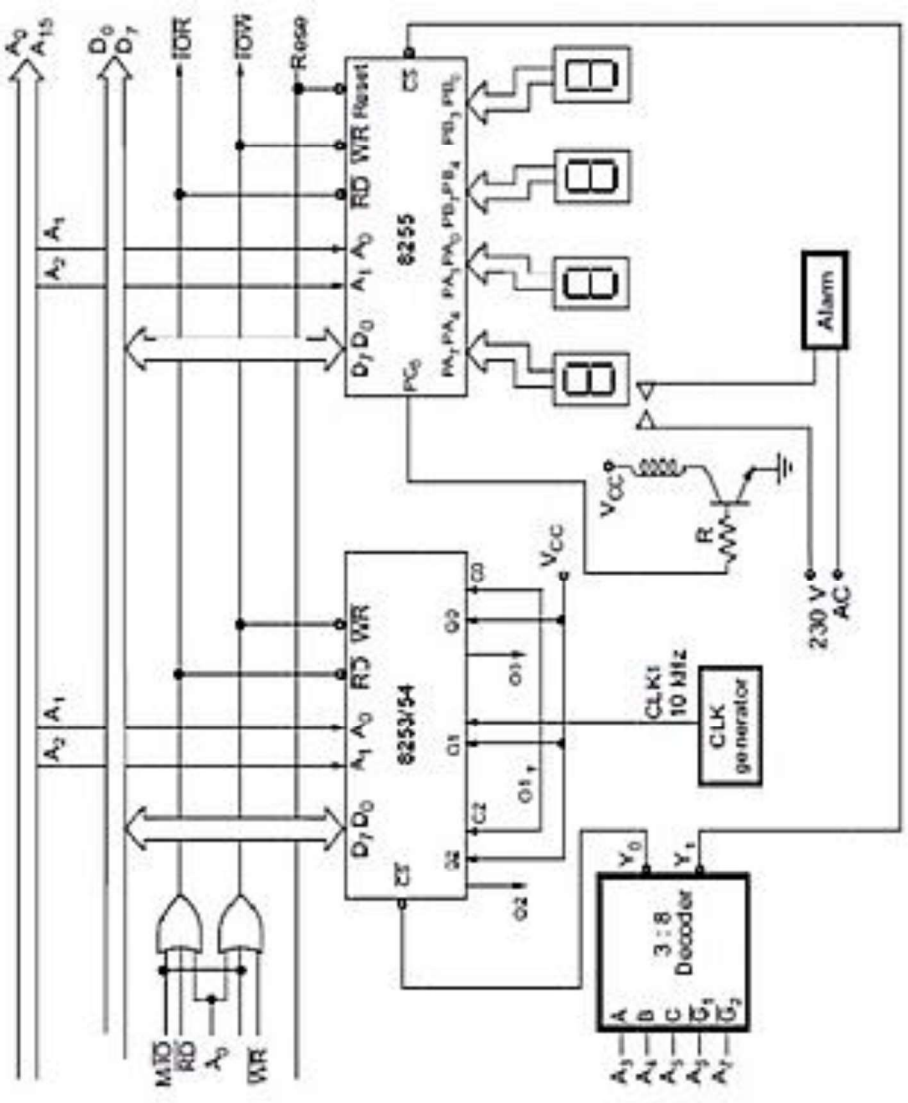


Fig 3.57: Pre-settable alarm system

To read four digit of count, we need four thumb wheels. One thumb wheel switch can be interfaced using four input lines. So to interface four thumbwheels, we need 16 lines. The iC8255 is used to interface these thumbwheel switches. Two thumb wheel switches are connected to port A and other two are connected to port B.

AllAbtEngg Android Application for Anna University, Polytechnic & School

Address Map

Ports/ Control Register	Address lines								Address
	A7	A6	A5	A4	A3	A2	A1	A0	
Counter 0	0	0	0	0	0	0	0	0	00H
Counter 1	0	0	0	0	0	0	1	0	02H
Counter 2	0	0	0	0	0	1	0	0	04H
Control Register	0	0	0	0	0	1	1	0	06H
Port A	0	0	0	0	1	0	0	0	08H
Port B	0	0	0	0	1	0	1	0	0AH
Port C	0	0	0	0	1	1	0	0	0CH
Control Register	0	0	0	0	1	1	1	0	0EH

Control word to read/write value in count register

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC2	RW1	RW0	M2	M1	M0	BCD
0	0	1	1	0	0	0	1

D0	= 1	BCD count
D1,D2,D3	= 000	Mode: Square wave rate generator
D4,D5	= 00	Read/write lower byte first and then higher byte
D6, D7	= 00	Counter 0

Alarm and IRQ output pinThe alarm interrupt can be programmed to occur at rates of (a) once per day, (b) once per hour, (c) once per minute, (d) once per second. Next, we look at each of these.

Once-per-day alarmTo program the alarm for once per day, we write the desired time for the alarm into the hour, minute, and second RAM locations l, 3, and 5. As the clock keeps the time, when all three bytes of hour, minute, and ond for the real time clock match the values in the alarm hour, minute, and second, the AF (alarm flag) bit in register of the 0512887 will go high. We can poll the AF bit in register C, which is a waste of microcontroller resources, or ow the IRQ pin to be activated upon matching the alarm time with the real time. It must be noted that in order to use IRQ pin of the DS12887 for an alarm, the interrupt-enable bit for alarm in register B (AIE) must be set high. How to able the AIE bit in register Bis shown shortly.

Once-per-hour alarm To program the alarm for once per hour, we write value llxxxxx into the alarm hour location of 5 only. Value Jxxxxx means any hex value of FCH to FFH. Very often we use value FFH.

Once-per-minute alarmTo program the alarm for once per minute, we write value FFH into both the alarm hour and alarm minute locations of 5 and 3.

Once-per-second alarm

To program the alarm for once per second, we write value FFH into all three locations of alarm hour, alarm minute, and alarm second.

AllAbtEngg Android Application for Anna University, Polytechnic & School

Using IRQ of 0512877 to activate the 8051 interrupt We can connect the IRQ of the DS12887 to the external interrupt pin of the 8051 (INTO). This allows us to perform a task once per day, once per minute, and so on. The program given in the next two pages will (a) sound the buzzer connected to SQW pin, and (b) will send the message "YES" to the serial port once per minute at exactly 8 seconds past the minute. The buzzer will stay on for 7 seconds before it is turned off.

LIST OF QUESTIONS
PART A

1. What is the need for interrupt controller?

When two or more device sends interrupt at the same time the interrupt controller will select one of the interrupt based on the priority.

2. What is keyboard debouncing?

When a key is depressed and released, the contact is not broken permanently. In fact, the key makes and breaks the contacts several times for a few milliseconds before the contact is broken permanently. If a key depression is detected by a microprocessor, it is possible that the depression may be false, i.e. it may be due to the bouncing of the key. It is, thus necessary to de bounce the key after depression.

3. What is settling or conversion time in DAC?

The time required to convert digital signal to analog signal is called as conversion time or settling time in DAC.

4. What is meant by two key lockout?

In two key lock out if another key is depressed while the first key is being debounced, the key which is released last will be entered into FIFO.

5. What is meant by N key roll over?

N-key roll over treats each key depression independently. If more than one key is depressed, after they were depressed they are all entered in the order they were sensed.

6. What is the necessity of programmable interval timer?

One of the most common problem in any microcomputer system is the generation of accurate time delay under software control. In many application the processor has to wait for some time. In keyboard interface, the CPU waits about 5 ms in case of each key depression, in order to check for false depression. 8253 solves this problem by facilitating three 16-bit programmable counters on the same chip.

7. What is the need for keyboard and display controller?

In the keyboard interface, the microprocessor scans the various keys, performs software debouncing and finally finds out the code of the depressed key. In the display interface, the microprocessor is busy in sending bit-by-bit information to the shift registers regarding the display of various segments. This puts the load on the CPU, which it may not be

AllAbtEngg Android Application for Anna University, Polytechnic & School