UNIT-III

8086 Memory and Digital Interfacing

- 8086 Addressing and Address Decoding
- Interfacing RAM, ROM, EPROM to 8086
- 8255 Programmable Peripheral Interface
- Various Modes of Operation of 8255 and Interfacing to 8086
- Interfacing to
  1. Keyboard
  2. Alphanumeric Displays
  3. Seven segment LED displays
  4. Stepper motor
  5. D/A and A/D Converters
Introduction to Memories

Memory types

Two basic types:

- ROM: read only memory
- RAM: read write memory

Four commonly used memories

- ROM
- FLASH (EEPROM)
- STATIC RAM (SRAM)
- DYNAMIC RAM (DRAM)

Introduction to Memories Cont..

Generic pin configuration

[Diagram showing address connection, output/input-output connection, and control signals such as WE (Write Enable), OE (Output Enable), and CS (Chip Select).]
Introduction to Memories Cont..

- **Generic pin configuration**
  - The number of address pins are related to the number of memory locations. Common sizes today are **1K to 256M** locations.
  - Therefore between 10 and 28 address pins are present.
  - The data pins are bi-directional in read write memories. The number of data pins are related to size of the memory location.
    - An 8-bit wide memory device has 8 data pins.
  - Each memory device has at least one chip select pin (logic 0) to enable the memory device.
  - If more than one pin is present, then all must be 0 in order to perform a read or write

- Each memory chip has at least one control pin.
  - For ROMs an output enable or read pin is present (logic 0).
  - For RAMs a write enable and a read enable pins are present (logic 0).
  - But both RD and WR cannot be 0 at once
Introduction to Memories Cont..

- **ROMs:**
  - Non volatile memory: Maintains its state when power down.
  - There are several forms:
    - PROM: Programmable Read Only Memory.
      Field programmable but only once.
    - EPROM: Erasable Programmable Read Only Memory.
      Reprogramming requires 20 min of high intensity UV light exposure.
    - Flash EEPROM: Electrically Erasable Programmable ROM.

### EPROMs

Intel 2716 EPROM (2K x 8):

- **V_{pp}** is used to program the device by applying 25V and pulsing PGM while holding CS high.

### Address Inputs

- \( A_0 \) to \( A_7 \): Address
- PD/PGM: Power down/Program
- CS: Chip Select
- \( O_n, O_2 \): Outputs

### Data Outputs

- **Output Buffers**
- **Y-Gating**
- **16,384 Cell Matrix**
- **X Decoder**
Introduction to Memories Cont..

- **SRAMs:**
  - TI TMS-4016 SRAM (2K X 8):
  - Virtually identical to the EPROM with respect to the pinout.

**DRAMs:**
- SRAMS are limited in size (up to about 128K × 8).
- DRAMS are available in much larger sizes, e.g., 64M × 1
- DRAMS must refreshed (re-written) for every 2ms to 4ms since they store their values on an integrated capacitor that looses charging over time.
- This refresh performed by special circuit in DRAM which refreshes the entire memory.
The processor can usually address a memory space that is much larger than the memory space covered by an individual memory chip.

In order to splice a memory device into address space of a processor, a decoding is necessary.

For example, 8086 issues 20-bit addresses for a total of 1 MB memory of address space.

However, the BIOS on a 2716 EPROM has only 2K × 8 (2 KB) of memory and 11 address lines.

An decoder can be used to decode additional 9 address pins and allow the EPROM to be placed in any 2 KB section of the 1 MB address space.

Memory Address Decoding Cont..
Memory Address Decoding Cont..

To determine the address range that a device is mapped into:

This 2KB memory segment maps into the reset location of the 8086 (FFFFOH)
NAND gate decoders are not often used. Rather the 3-8 Line Decoder (74LS138) is more common.

Interfacing of Static RAM and ROM

- The semiconductor memories are organized as two dimensional arrays of memory locations.
- In general to address a memory location out of N memory locations, we will require at least n bits of address i.e. n address lines where \( n = \log_2 N \).
- Thus if microprocessor has n address lines, then it is able to address at the most N locations of memory, where \( 2^n = N \).
- If out of N locations only p memory locations are to be interfaced, then the least significant p address lines out of the available n lines can be directly connected from the microprocessor to the memory chip while the remaining (n-p) high order lines may be used for address decoding (as input to the chip selection).
Interfacing of Static RAM and ROM Cont..

- The general procedure of static memory interfacing with 8086
  1. Arrange the available memory chips so as to obtain 16-bit data bus width.
     • The upper 8-bit bank is called ‘odd address memory bank’.
     • The lower 8-bit bank is called ‘even address memory bank’.
  2. Connect available memory address lines of memory chips with those of the microprocessor and also connect the RD and WR inputs to the corresponding processor control signals.
  3. Connect the 16-bit data bus of memory bank with that of the microprocessor 8086.
  4. The remaining address lines of the microprocessor, BHE and A0, are used for decoding the required chip select signals for the odd and even memory banks. The CS of memory is derived from the output of the decoding circuit.

- Interface two 4K×8 EPROMs and two 4K×8 RAM chips with 8086. select suitable maps.
  - We know that, after reset the IP and CS are initialized to form address FFFF0H. Hence this address must lie in the EPROM.
  - The address of RAM may be selected anywhere in the 1MB address of 8086.
Interfacing of Static RAM and ROM Cont..

Memory chip selection

<table>
<thead>
<tr>
<th>Decoder I/O</th>
<th>A2</th>
<th>A1</th>
<th>A0</th>
<th>Selection/ Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address/BHE</td>
<td>A2</td>
<td>A1</td>
<td>A0</td>
<td>BHE</td>
</tr>
<tr>
<td>Word transfer on $D_0 - D_{15}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Even and odd addresses in RAM</td>
</tr>
<tr>
<td>Byte transfer on $D_2 - D_6$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Only even address in RAM</td>
</tr>
<tr>
<td>Byte transfer on $D_2 - D_{15}$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Only odd address in RAM</td>
</tr>
<tr>
<td>Word transfer on $D_2 - D_{15}$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Even and odd addresses in ROM</td>
</tr>
<tr>
<td>Byte transfer on $D_0 - D_7$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Only even address in ROM</td>
</tr>
<tr>
<td>Byte transfer on $D_2 - D_{15}$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Only odd address in ROM</td>
</tr>
</tbody>
</table>
Memory Address Decoding Cont..

Introduction to I/O Ports Interfacing

- I/O ports or Input/output ports are the devices through which the microprocessor communicates with other devices or external data source/destinations.
- Input activity, as one may expect, is the activity that enables the microprocessor to read data from external devices, for example keyboards. These devices are known as input devices as they feed data into microprocessor system.
- Output activity transfers data from the microprocessor to the external devices, for example CRT display. These devices which accept the data from a microprocessor system are called output devices.
- Thus for a microprocessor the input activity is similar to read operation, while the output activity is similar to write operation.
Steps in Interfacing an I/O Device

- Connect the data bus of the microprocessor system with the data bus of the I/O port.
- Derive a device address pulse by decoding the required address of the device and use it as the chip select of the device.
- Use a suitable control signal i.e. IORD or IOWR to carry out device operations.

I/O Structure of a Typical Microcomputer

- The IO devices connected to a microprocessor system provide an efficient means of communication between the microprocessor system and outside world.
- The characteristics of the IO devices are normally different from the characteristics of the microprocessor.
- Since the characteristics of the IO devices are not compatible with that of microprocessor, an interface hardware circuitry between microprocessor and the IO device is necessary.
There are three major types of data transfer between the microprocessor and an IO device. They are:

1. Programmed IO
2. Interrupt driven IO
3. Direct memory access (DMA)

In programmed IO, the data transfer is accomplished through IO port and controlled by a software.

In Interrupt driven IO, the IO device will interrupt the processor and initiate data transfer.

In DMA, the data transfer between the memory and IO can be performed by bypassing the microprocessor.

Interfacing IO and Peripheral devices

IO devices are generally slow devices and so they are connected to system bus through ports.

The ports are buffered IC’s which are used to temporarily hold the data transmitted from the microprocessor to the IO device.

For data transfer from the input device to the processor, the following operations are performed:

1. The input device will load the data to port.
2. When the port receives a data, it sends a message to the processor to read the data.
3. The processor will read the data from the port.
4. After a data has been read by the processor, the input device will load the next data into the port.
Interfacing IO and Peripheral devices Cont..

For data transfer from processor to the output device, the following operations are performed:

1. The processor will load the data to port.
2. The port will send a message to the output device to read the data.
3. The output device will read the data from the port.
4. After a data has been read by the output device, the processor can load the next data to the port.

- Latch acts as a good output port.
- Buffer acts as a good input port.

Fig. a) Latch (O/P Port) b) Buffer (I/P Port)
Methods of Interfacing IO devices

- There are two ways of interfacing IO devices in a microprocessor/microcontroller based system:
  - Memory Mapped IO Device
  - Standard IO Mapped IO Device or Isolated IO Mapping

Comparison of Memory mapping and IO mapping

<table>
<thead>
<tr>
<th>Memory Mapping</th>
<th>IO mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. 20-bit addresses are provided for IO devices.</td>
<td>1. 8-bit or 16-bit address are provided for IO devices</td>
</tr>
<tr>
<td>2. The IO ports or peripherals can be treated like memory locations and so all instructions related to memory can be used for data transfer.</td>
<td>2. Only IN and OUT instructions can be used for data transfer between IO device and the processor.</td>
</tr>
<tr>
<td>3. In memory mapped ports, the data can be moved from any register to port and vice versa</td>
<td>3. In IO mapped ports, the data transfer can take only between the accumulator and the ports</td>
</tr>
<tr>
<td>4. When memory mapping is used for IO devices, the full memory address space cannot be used for addressing memory.</td>
<td>4. When IO mapping is used for IO devices, the full address space can be used for addressing memory.</td>
</tr>
<tr>
<td>5. For accessing memory mapped devices, the processor executes the memory read or write cycle. During this cycle, ( M/IO ) is asserted high.</td>
<td>5. For accessing IO mapped devices, the processor executes the IO read or write cycle. During this cycle, ( M/IO ) is asserted low.</td>
</tr>
</tbody>
</table>
Problem

- Interface an input port 74LS245 to read the status of switches SW1 to SW8. The switches, when shorted, input a 1 else input a 0 to the microprocessor system. Store the status in register BL. The address of the port is 0740H.

  - The ALP is given as follows:
    - MOV BL,00
    - MOV DX,0740H
    - IN AL, DX
    - MOV BL, AL
    - HLT
The parallel input-output port chip 8255 is also known as programmable peripheral input-output port.

- It has 24 input/output lines which may be individually programmed in two groups of twelve lines each, or three groups of eight lines.
- The two groups of I/O pins are named as Group A and Group B.
- Each of these two group contains a subgroup of eight I/O lines called as 8-bit port and another subgroup of four I/O lines or a 4-bit port.
- Thus Group A contains an 8-bit port A along with a 4-bit port, C upper. The port A lines are identified by symbols PA₀ – PA₇ while the port C lines are identified as PC₄–PC₇.
Similarly Group B contains an 8-bit port and a 4-bit port C with lower bits. The port C upper and port C lower can be used in combination as an 8-bit port C. All of these ports function independently either as input or as output ports. This can be achieved by programming the bits of internal register of 8255 called as Control Word Register. The 8-bit data bus buffer is controlled by read/write control logic. The read/write control logic manages all of the internal and external transfer of both data and control words.
**Pin Diagram of Programmable Peripheral Interface (PPI) 8255**

- **PA<sub>7</sub>-PA<sub>0</sub>** These are eight port A lines that act as either latched output or buffered input lines depending upon the control word loaded into the control word register.
- **PC<sub>7</sub>-PC<sub>0</sub>** Upper nibble of port C lines. They may act as either output latches or input buffers lines. This port also can be used for generation of handshake lines in mode 1 or mode 2.
- **PB<sub>7</sub>-PB<sub>0</sub>** These are the eight port B lines which are used as latched output lines or buffered input lines in the same way as port A.
- **RD** This is the input line driven by the microprocessor and should be low to indicate read operation to 8255.
- **WR** This is an input line driven by the microprocessor. A low on this line indicates write operation.
- **CS** This is a chip select line. If this line goes low, it enables the 8255 to respond to RD and WR signals, otherwise RD and WR signals are neglected.
- **A<sub>2</sub>-A<sub>0</sub>** These are the address input lines and are driven by the microprocessor. These lines (A<sub>2</sub> - A<sub>0</sub>) with RD, WR and CS form the following operations for 8255. These address lines are used for addressing any one of the four registers, i.e., three ports and a control word register.

---

**Table (a)**

<table>
<thead>
<tr>
<th>RD</th>
<th>WR</th>
<th>CS</th>
<th>A&lt;sub&gt;1&lt;/sub&gt;</th>
<th>A&lt;sub&gt;0&lt;/sub&gt;</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Port A to data bus</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Port B to data bus</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Port C to data bus</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>CWR to data bus</td>
</tr>
</tbody>
</table>

**Table (b)**

<table>
<thead>
<tr>
<th>RD</th>
<th>WR</th>
<th>CS</th>
<th>A&lt;sub&gt;1&lt;/sub&gt;</th>
<th>A&lt;sub&gt;0&lt;/sub&gt;</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Data bus to Port A</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Data bus to Port B</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Data bus to Port C</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Data bus to CWR</td>
</tr>
</tbody>
</table>

**Table (c)**

<table>
<thead>
<tr>
<th>RD</th>
<th>WR</th>
<th>CS</th>
<th>A&lt;sub&gt;1&lt;/sub&gt;</th>
<th>A&lt;sub&gt;0&lt;/sub&gt;</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>Data bus tristated</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>Data bus tristated</td>
</tr>
</tbody>
</table>
There are two basic modes of operation of 8255 - I/O mode and BSR (Bit-Set-Reset) mode.

In the I/O mode, the 8255 ports work as programmable I/O ports, while in BSR mode only port C (PC0 – PC7) can be used to set or reset its individual port bits.

Under the I/O mode of operation, further there are three modes of operation of 8255, so as to support different types of applications, viz. mode 0, mode 1 and mode 2.
BSR Mode

- In this mode, any of the 8-bits of port C can be set or reset depending on B_0 of the control word.
- The bit to be set or reset is selected by bit select flags B_3, B_2 and B_1 of the CWR as given in the table below.

<table>
<thead>
<tr>
<th>B_3</th>
<th>B_2</th>
<th>B_1</th>
<th>Selected Bits of port C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>B_0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>B_1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>B_2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>B_3</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>B_4</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>B_5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>B_6</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>B_7</td>
</tr>
</tbody>
</table>

BSR Mode

- The CWR format is as shown below:

```
 B_7 | B_6 | B_5 | B_4 | B_3 | B_2 | B_1 | B_0
```

- B_3, B_2, B_1, are from 000 to 111 for bits PC_3 to PC_7.
I/O Modes – MODE 0 (Basic I/O Mode)

- This mode is also known as basic input/output mode.
- This mode provides input and output capability using each of the three ports.
- Data can be simply read from or written to the input and output ports respectively, after appropriate initialization.

The salient features of this mode are:
1. Two 8 bit ports (port A and port B) and two 4-bit ports (port C upper and lower) are available. The two 4-bit ports can be combinedly used as a third 8-bit port.
2. Any port can be used as an input and output port
3. Output ports are latched. Input ports are not latched.
4. A maximum of four ports are available so that overall 16 I/O configurations are possible.

All these modes can be selected by programming a register internal to 8255, known as Control Word Register (CWR) which has two formats. The first format is valid for I/O modes of operation i.e., mode 0, mode 1 and mode 2 while the second format is valid for bit set/reset (BSR) mode of operation.
I/O Modes

The CWR format is as shown below:

Features of MODE 1

1. Two groups—group A and group B are available for strobed data transfer.
2. Each group contains one 8-bit data I/O port and one 4-bit control/data port.
3. The 8-bit port can be either used as input or an output port.
4. Out of 8-bit port C, PC_0-PC_2 are used to generate control signals for port B and PC_3-PC_5 are used to generate control signals for port A. The lines PC_6,PC_7 may be used as independent data lines.
I/O Modes – Mode 1 (Strobed I/O Mode)

The control signals for both the groups in input and output modes are explained as follows:

**Input Control Signal Definitions (Mode 1):**

- **STB (Strobe input)**—If this line falls to logic low level, the data available at 8-bit input port is loaded into input latches.
- **IBF (Input buffer full)**—If this signal rises to logic 1, it indicates that data has been loaded into the latches, i.e., it works as an acknowledgement. IBF is set by a low on STB and is reset by the rising edge of RD input.
- **INTR (Interrupt request)**—This active high output signal can be used to interrupt the CPU whenever an input device requests the service. INTR is set by a high at STB pin and a high at IBF pin. INTR is reset by a falling edge on RD input. Thus an external input device can request the service of the processor by putting the data on the bus and sending the strobe signal.

**Output Control Signal Definitions (Mode 1):**

- **OBF (Output buffer full)**—This status signal, whenever falls to logic low, indicates that the CPU has written data to the specified output port. The OBF flip-flop will be set by a rising edge of WR signal and reset by a low going edge at the ACK input.
- **ACK (Acknowledge input)**—ACK signal acts as an acknowledgement to be given by an output device. ACK signal, whenever low, informs the CPU that the data transferred by the CPU to the output device through the port is received by the output device.
- **INTR (Interrupt request)**—Thus an output signal that can be used to interrupt the CPU when an output device acknowledges the data received from the CPU. INTR is set when ACK, OBF and INTE are ‘1’. It is reset by a falling edge on WR input.
I/O Modes – Mode 1 (Strobed I/O Mode)

Input control signal definitions in Mode 1

Output control signal definitions Mode 1

(a) Mode 1 Control Word Group A (b) Mode 1 Control Word Group B
I/O Modes – Mode 2 (Strobed Bidirectional I/O Mode)

This mode is also known as strobed Bidirectional input/output mode. This mode of operation provides 8255 with an additional feature for communicating with a peripheral device on an 8-bit data bus. Handshaking signals are provided to maintain proper data flow and synchronization between data transmitter and receiver.

Features of MODE 2
1. The single 8-bit port in group A is available.
2. The 8-bit port is bidirectional and additionally a 5-bit control port is available.
3. Three I/O lines are available at port C
4. Input and output ports are both latched
5. The 5-bit control port C is used for generating/accepting handshake signals for 8-bit data transfer on port A.

Control signal definitions in mode 2

INTR (Interrupt request) As in mode 1, this control signal is active high and is used to interrupt the microprocessor to ask for transfer of the next data byte to/from it. This signal is used for input (read) as well as output (write) operations.

Control signals for output operations
ÖBF (Output buffer full) This signal, when falls to logic low level, indicates that the CPU has written data to port A.
ACK (Acknowledgment) This control input, when falls to logic low level, acknowledges that the previous data byte is received by the destination and the next byte may be sent by the processor. This signal enables the internal tristate buffers to send out the next data byte on port A.
INTEI (A flag associated with ÖBF) This can be controlled by bit set/reset mode with PCn.

Control signals for input operations
STB (Strobe input) A low on this line is used to strobe in the data into the input latches of 8255.
IBF (Input buffer full) When the data is loaded into the input buffer, this signal rises to logic ‘1’. This can be used as an acknowledgement that the data has been received by the receiver.
I/O Modes – Mode 2 (Strobed Bidirectional I/O Mode)

Interfacing 8255 to 8086
Interfacing 8255 to 8086

Problem: Interface an 8255 with 8086 to work as an I/O port.
Initialize port A as output port, port B as input port and port C as output port. Port A address should be 0740H. Write a program to sense switch positions SW0-SW7 connected to port B, the sensed pattern is to display on port A, to which 8 LEDs are connected, while the port C lower displays number of on switches out of the total eight switches.

Solution:

Thus 82H is the control word for the requirements in the problem. The port address can be done as given below.

<table>
<thead>
<tr>
<th>8255 Ports</th>
<th>8255 I/O Address lines</th>
<th>Hex Port Addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port A</td>
<td>A12 A14 A16 A17 A22 A24 A26 A28 A29 A30 A31 A32 A33 A34 A36 A38 A40</td>
<td>0740H</td>
</tr>
<tr>
<td>Port B</td>
<td>A12 A14 A16 A17 A22 A24 A26 A28 A29 A30 A31 A32 A33 A34 A36 A38 A40</td>
<td>0740H</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>8255 Ports</th>
<th>8255 I/O Address lines</th>
<th>Hex Port Addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port C</td>
<td>A12 A14 A16 A17 A22 A24 A26 A28 A29 A30 A31 A32 A33 A34 A36 A38 A40</td>
<td>0744H</td>
</tr>
</tbody>
</table>

| CWR        | A12 A14 A16 A17 A22 A24 A26 A28 A29 A30 A31 A32 A33 A34 A36 A38 A40 | 0746H |
Interfacing 8255 to 8086 Cont..

Solution: The 8255 is to be interfaced with lower order data bus, i.e. D₀-D₇.
- The A₀ and A₁ pins of 8255 are connected to A₀₁ and A₀₂ pins of microprocessor respectively.
- The A₀₀ pin of the microprocessor is used for selecting lower byte of data bus.
- Hence any change in the status of A₀₀ does not affect the port to be selected.
- Let us use absolute decoding scheme that uses all the 16 address lines for deriving the device address pulse.
- Out of A₀-A₁₅ lines, two address lines A₀₂ and A₀₃ are directly required by 8255 for three port and CWR address decoding.
- Hence only A₁ to A₁₅ are used for decoding address.
Interfacing 8255 to 8086 Cont..

Flow Chart

Program

MOV DX,0746H ; Initialize CWR with
MOV AL,82H ; control word 82H
OUT DX,AL ;
SUB DX,04 ; Get address of port B in DX
IN AL,DX ; Read port B for switches
SUB DX,02 ; positions in AL and get port A
; address in DX
OUT DX,AL ; Display switch positions on port A
MOV BL,00H ; Initialize BL for switch count
MOV CH,08H

YY: ROL AL ; Rotate AL through carry to check,
JNC XX ; whether the switches are on or
INC BL ; off, i.e. either 1 or 0

XX: DEC CH ; Check for next switch. If all
JNZ YY ; switches are checked, the
MOV AL,BL ; number of on switches are
ADD DX,04 ; in BL. Display it on port C
OUT DX,AL ; lower
HLT
Interfacing to Alphanumeric Displays

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

LED Displays

- LED Displays are available in two very common formats.
  a. 7 segment displays
  b. 5 by 7 dot-matrix displays
Interfacing to Alphanumeric Displays Cont..

Seven-Segment displays

- 7 segment displays are generally used as numerical indicators and consists of a number of LEDs arranged in seven segments.
- Any number between 0 to 9 can be indicated by lighting the appropriate segments.
- The seven segments are labeled a to g and dot is labeled as h.
- By forward biasing different LED segments, we can display the digits 0 through 9.

These 7 segment displays are of two types:

a. Common Anode Type
b. Common Cathode Type
Interfacing to Alphanumeric Displays Cont..

5 by 7 DOT-matrix LED
➢ These displays can be used to display numbers as well as alphabets.

Interfacing LED Displays

Static Display
Fig shows a circuit to drive a single, seven segment, common anode LED display.
Interfacing LED Displays

Static Display

- For common anode, when anode is connected to positive supply, a low voltage is applied to a cathode to turn it on.
- Here BCD to seven segment decoder, IC7447 is used to apply low voltages at cathodes according to BCD input applied to 7447.
- To limit the current through LED segments resistors are connected in series with the segments.
- This circuit connection is referred to as a static display because current is being passed through the display at all times.
Interfacing LED Displays

Multiplexed Display

- Here, common anode seven segment LEDs are used.
- Anodes are connected to +5V through transistors. Cathodes of all seven segments are connected in parallel and then to the output of 7447 IC through resistors.
- In multiplexed display the segment information sent for all digits on the common lines, but only one display digit is turned on at a time.
- The PNP transistors connected in series with the common anode of each digit act as an ON and OFF switch for that digit.

Interfacing to Alphanumeric Displays

Problem: Interface an 8255 with 8086 at 80H as an I/O address of port A. Interface five 7 segment displays with 8255. Write a sequence of instructions to display 1, 2, 3, 4 and 5 over five displays continuously as per their positions starting with 1 at the least significant position.
Solution: In this scheme, I/O port A is multiplexed to carry data to all the 7-segment displays. The port B selects one of the display at a time.

The display used are common anode type.

<table>
<thead>
<tr>
<th>Number to be displayed</th>
<th>PA_2</th>
<th>PA_3</th>
<th>PA_4</th>
<th>PA_5</th>
<th>PA_6</th>
<th>PA_7</th>
<th>PA_8</th>
<th>PA_9</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>CF</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>92</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

All these codes, decided above, are stored in a look up table starting at 2000:0001
Interfacing to Alphanumeric Displays

Program:

AGAIN:    MOV CL,05H ;Count for displays
          MOV BX,2000H ;Initialize data segment
          MOV DS,BX ;for look up table
          MOV CH,01H ;1st number to be displayed
          MOV AL,80H ;Load control word in the
          OUT 86H,AL ;CWR
          MOV DL,01H ;Enable code for Least
          ;significant 7-seg display

NEXTDGT: MOV BX,0000H;Set pointer to look up table
          MOV AL,CH ;First no to display
          ;Store number to be displayed in AL
          XLAT ;Find code from look up table
          OUT 80H,AL ;Display the code
          MOV AL,DL ;Enable the display
          OUT 82H,AL ;
          ROL DL ;Go for selecting next display
          INC CH ;Next number to display
          DEC CL
          JNZ NXTDGT ;Go for next digit display
          JMP AGAIN
Interfacing to Keyboard

Problem: Interface a 4*4 keyboard with 8086 using 8255 and write an ALP for detecting a key closure and return the key code in AL. The debouncing period for a key is 10ms. Use software key debouncing technique. DEBOUNCE is an available 10ms delay routine.

Interfacing to Keyboard Cont..

Keyboard Types:
- When you press a key on your computer, you are activating a switch. There are many different ways of making these switches.
  - Mechanical Key Switches
  - Membrane Key Switches
  - Capacitive Key Switches
  - Hall Effect Key Switches
Keyboard Types:

In most keyboards, the key switches are connected in a matrix of rows and columns. We use simple mechanical switches but the principle is same for other types of switches.

Getting a meaningful data from a keyboard requires three major tasks.

1. Detect a key press.
2. Debounce the key press
3. Encode the key press

The three tasks can be done with hardware, software, or a combination of two depending on application.
Interfacing to Keyboard Cont..

Solution:
- Port A is used as output port for selecting a row of key while port B is used as an input port for sensing a closed key.
- Thus keyboard lines are selected one by one through port A and the port B lines are polled continuously till a key closure is sensed.
- Then routine DEBOUNCE is called for debouncing.
- The key code is decided depending upon the selected row and a low sensed column.
- The higher order lines of port A and port B are left unused.
- The address of port A and port B will be respectively 8000H and 8002H while the address of CWR will be 8006H.
Interfacing to Keyboard Cont..

Flow chart:

1. Make all row lines zero by sending low on all output lines. This activates all keys in the keyboard matrix.
2. Read the status of return lines. If status of all lines is logic high, key is not pressed; otherwise pressed.

Check 2:
1. Activate keys from any one column by making any one column line zero.
2. Read the status of return lines. The zero on any return line indicates key is pressed from the corresponding row and selected column.
3. Activate the keys from next column and repeat 2 and 3 for all columns.
Interfacing to Keyboard Cont..

Program:

```
CODE SEGMENT
ASSUME CS : CODE
START: MOV AL, 82H ; Load CHR with
        MOV DX, 8006H ; control word
        OUT DX, AL ; required
        MOV BL, 00H ; Initialize BL for key code
        XOR AX, AX ; Clear all flags
        MOV DX, 8000H ; Port Address in AK.
        OUT DX, AL ; Ground all rows.
        ADD DX, 02H ; Port B address in DX.
        WAIT: IN AL, DX ; Read all columns.
        AND AL, OFH ; Mask data lines D0-D4.
        CMP AL, OFH ; Any key closed?
        JZ WAIT ; If not, wait till key
        CALL DEBOUNCE ; closure else wait for 10 ms
        MOV AL, 7FH ; Load data byte to ground
        MOV BH, 04H ; a row and set row counter.
```

Interfacing to Keyboard Cont..

Program:

```
NXTROW : ROL AL, 01 ; Rotate AL to ground next row.
        MOV CH, AL ; Save data byte to ground next row.
        SUB DX, 02H ; Output port address is in DX.
        OUT DX, AL ; Ground one of the rows.
        ADD DX, 02H ; Input port address is in DX.
        IN AL, DX ; Read input port for key closure.
        AND AL, OFH ; Mask D5-D7.
        MOV CL, 04H ; Set column counter.
```
Interfacing to Keyboard Cont..

Program:

NXTCOL : ROR AL, 01    ; Move D0 in CF.
JNC CODEXY    ; Key closure is found, if CF=0.
INC BL        ; Increment BL for next binary
              ; key code.
DEC CL        ; Decrement column counter.
              ; if no key closure found.
JNZ NXTCOL    ; Check for key closure in next column
MOV AL, CH    ; Load data byte to ground next row.
DEC BH        ; if no key closer found in column
              ; get ready to ground next row.
JNZ NXTROW    ; Go back to ground next row.
JMP WAIT      ; Jump back to check for key.
              ; closure again.

Interfacing to Keyboard Cont..

Program:

CODEXY : MOV AL, BL    ; Key code is transferred to AL.
          MOV AH, 4CH    ; Return to DOS prompt.
          INT 21H

DEBOUNCE PROC NEAR
          MOV CL, OE2H
BACK:    NOP
          DEC CL
          JNZ BACK
          RET
DEBOUNCE ENDP
CODE ENDS
END START
Interfacing to Stepper Motor

- A stepper motor is a device used to obtain an accurate position control of rotating shafts.
- It employs rotation of its shafts in terms of steps, rather than continuous rotation as in case of AC or DC motors.
- To rotate the shafts of the stepper motor, a sequence of pulses is needed to be applied to the windings of the stepper motor in a proper sequence.
- The number of pulses required for one complete rotation of the shaft of the stepper motor are equal to its number of internal teeth or its rotor.
- The stator teeth and the rotor teeth lock with each other to fix a position of the shaft.

With a pulse applied to the winding input the rotor rotates by one teeth position or an angle $x$.

$$x = \frac{360^\circ}{\text{no. of rotor teeth}}$$

After the rotation of the shaft through angle $x$, the rotor locks itself with the next tooth in sequence on the internal surface of stator.
A typical stepper motor may have parameters like torque 3Kg-cm, operating voltage 12V, current rating 0.2A and a step angle 1.8°, i.e. 200 steps/revolution (number of rotor teeth).

The stepper motors are designed to work with digital circuits.

Binary level pulses of 0-5V are required at its winding inputs to obtain the rotation of shafts.

The sequence of pulses can be decided upon the required motion of the shaft.

The circuit for interfacing a winding $W_n$ with an I/O port is

> Each of the windings of a stepper motor need this circuit for its interfacing with output port
Interfacing to Stepper Motor

- A simple scheme for rotating the shaft of stepper motor is called a wave scheme.
- In this scheme, the windings \( W_a, W_b, W_c \) and \( W_d \) are applied with required voltage pulses, in a cyclic fashion.

### Excitation Sequences of a Stepper Motor Using Wave Switching Scheme

<table>
<thead>
<tr>
<th>Motion</th>
<th>Step</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clockwise</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Anticlockwise</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Interfacing to Stepper Motor

- Design a stepper motor controller and write an ALP to rotate shaft of 4-phase stepper motor:
  a. In clockwise 5 rotations
  b. In anticlockwise 5 rotations

The 8255 port A address is 0740H. The stepper motor has 200 rotor teeth. The port A bit \( PA_0 \) drives winding \( W_a \), \( PA_1 \) drives \( W_b \) and so on. The stepper motor has an inertial delay of 10ms. Assume that the routine for this delay is already available.
The stepper motor connections for all the four windings are:

![Stepper Motor Windings Connections](image)

```asm
ASSUME CS:CODE
CODE SEGMENT
START: MOV AL,80H
   OUT CWR,AL
   MOV AL,88H ;Bit pattern 10001000 to start
   MOV CX,1000 ;the sequence of excitation
AGAIN1: OUT PORTA,AL ;from Wa. Excite Wa, Wb,
   CALL DELAY ;Wc, Wd in sequence with
   ROL AL,01 ; delay. For clockwise
   DEC CX ; rotations the count is
   JNZ AGAIN1 ; 200*5=1000. Excite till
   ;count=0
CODE ENDS
```
Interfacing to Stepper Motor

MOV AL, 88H ; Bit pattern to excite Wa
MOVCX, 1000 ; count for 5 rotations
AGAIN2: OUT PORTA, AL ; excite Wa, Wb, Wc, Wd
CALL DELAY ; Wait
ROR AL, 01 ; anticlockwise
DEC CX
JNZ AGAIN2
MOV AH, 4CH
INT 21H
CODE ENDS
END START

The count for rotating the shafts of stepper motor through a specified angle may be calculated from the number of rotor teeth. The number of rotor teeth is equal to the count of one rotation i.e. 360°. Hence for any specified angle θ° the count is calculated as

\[ C = \frac{\text{Number of rotor teeth}}{360°} \times \theta° \]
Interfacing to A/D Converter

- The analog to digital converter is treated as an input device by the microprocessor, that sends an initializing signal to ADC to start the analog to digital data conversion process.
- The start of conversion signal is a pulse of a specific duration.
- The process of analog to digital conversion is a slow process, and the microprocessor has to wait for the digital data till the conversion is over.
- After conversion is over, the ADC sends end of conversion (EOC) signal to inform the microprocessor about it and the result is ready at the output buffer of the ADC.

Interfacing to A/D Converter Cont..

- These tasks of issuing an SOC pulse to ADC, reading EOC signal from the ADC and reading the digital output of the ADC are carried out by the CPU using 8255 I/O ports.
- The time taken by the ADC from the active edge of SOC pulse till the active edge of EOC signal is called as the conversion delay of the ADC.
- The available ADCs in the market use different conversion techniques for conversion of analog signal to digital signals.
- Successive approximation and dual slope integration techniques are most popular techniques used in integrated ADC chips.
Interfacing to A/D Converter Cont..

A general algorithm for ADC interfacing contains:
1. Ensure the stability of analog input applied to the ADC
2. Issue start of conversion (SOC) pulse to ADC
3. Read end of conversion (EOC) signal to mark the end of conversion process
4. Read digital data output of the ADC as equivalent digital output

➢ It may be noted that the analog input voltage must be a constant at the input of the ADC right from the beginning to end of the conversion to get correct results.

Interfacing to A/D Converter Cont..

➢ This may be ensured by a sample and hold circuit which samples the analog signal and holds it constant for a specified time duration.
➢ The microprocessor may issue a hold signal to the sample and hold circuit.
➢ If the applied input changes before the complete conversion process is over, the digital equivalent of the analog input calculated by the ADC may not be correct.
ADC 0808/0809

The analog to digital converter chips 0808 and 0809 are 8-bit CMOS, successive approximation converters.

> These converters internally have 3:8 analog multiplexer so that at a time eight different analog inputs can be connected to chip.
> Out of these eight inputs only one can be selected for conversion using address lines ADD A, ADD B and ADD C

<table>
<thead>
<tr>
<th>Analog I/P selected</th>
<th>C</th>
<th>B</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/P 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>I/P 1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>I/P 2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>I/P 3</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>I/P 4</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>I/P 5</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>I/P 6</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>I/P 7</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
These are unipolar analog to digital converters, i.e. they are able to convert only positive analog input voltages to their digital equivalents.

- Problem: Interface ADC 0808 with 8086 using 8255 ports. Use port A of 8255 for transferring digital data output of ADC to the CPU and port C for control signals. Assume that an analog input is present at I/P 2 of the ADC and a clock input of suitable frequency is available for ADC. Draw the schematic and write required ALP.
The analog input I/P is used and therefore address pins A, B, C should be 010 respectively to select I/P₂.
The OE and ALE pins are already kept at +5V to select the ADC and enable the outputs.
Port C upper acts as the input port to receive the EOC signal while port C lower acts as output port to send SOC to the ADC.
Port A acts as a 8-bit input data port to receive the digital data output from the ADC.
The required ALP

```
MOV AL, 98 H ; Initialise 8255 as
OUT CWR,AL ; discussed above
MOV AL,02H ; Select I/P 2 as analog
OUT Port B,AL ; input
MOV AL,00H ; Give start of conversion
OUT Port C,AL ; pulse to the ADC.
MOV AL,01 H
OUT Port C,AL
MOV AL,00H
OUT Port C,AL
WAIT : IN AL,Port:C ; Check for EOC by
RCL ; reading port C upper and
JNC WAIT ; rotating through carry.
IN AL,Port A ; If EOC, read digital equivalent in AL
HLT ; Stop
```

Interfacing to D/A Converters

- The digital to analog converters convert binary numbers into their corresponding analog equivalent voltages.
- The DAC find applications in areas like digitally controlled gains, motor speed controls, programmable gain amplifiers etc.
DAC 0800

- The DAC 0800 is a monolithic 8-bit DAC manufactured by National Semiconductor.

Problem: Write an ALP to generate a triangular wave of frequency 500 Hz using the DAC 0800 interfacing. The 8086 system operates at 8 MHz the amplitude of the triangular wave should be +5V.
DAC 0800 Cont..

- The $V_{ref+}$ should be tied to +5V to generate a wave of +5V amplitude.
- The required frequency of the output is 500 Hz i.e. the period is 2ms.
- Assuming the wave to be generated is symmetric, the waveform will rise for 1 ms and fall for 1ms. This will be repeated continuously.

```
ASSUME   CS : CODE
CODE     SEGMENT
START    : MOV AL,80H   ; Initialise 8255 ports
          OUT CWR,AL    ; suitably.
          MOV AL,00H   ; Start rising ramp from
BACK     : OUT Port A,AL ; 0V by sending 00H to DAC.
          INC AL      ; Increment ramp till 5V
          CMP AL,FFH  ; i.e. FFH.
          JB BACK    ; If it is FFH then,
BACK1    : OUT Port A,AL ; Output it and start the falling
          DEC AL      ; ramp by decrementing the
          CMP AL,00   ; counter till it reaches
          JA BACK1    ; zero. Then start again
          JMP BACK   ; for the next cycle.
.CODE    ENDS
END START
```