Since 1951

JAMAL MOHAMED COLLEGE (AUTONOMOUS)

Accredited (3rd Cycle) with 'A' Grade by NAAC

Affiliated to Bharathidasan University

TIRUCHIRAPPALLI – 620 020

DEPARTMENT OF COMPUTER SCIENCE

MICROPROCESSOR INTERFACING AND APPLICATIONS

Prepared By: Dr. T. Abdul Razak

I/O INTERFACES

Memories and I/O devices are interfaced to the microprocessor to form a microcomputer. In the case of large and minicomputers the memories and input/output devices are interfaced to CPU by the manufacturer. In a microprocessor-based system the designer has to select suitable memories and input/output devices for his task and interface them to the microprocessor. The selected memories and I/O devices should be compatible with microprocessor.

ADDRESS SPACE PARTITIONING

The Intel 8086 uses 20-bit wide address bus for addressing memories and I/O devices. Using 20-bit wide address bus it can access 2^{20} = 1 MB of memory and I/O devices. The addresses are to be assigned to memories and I/O devices for their addressing. There are two schemes for allocation of addresses to memories and I/O devices:

1. Memory mapped I/O scheme
2. I/O mapped I/O scheme

Memory mapped I/O scheme

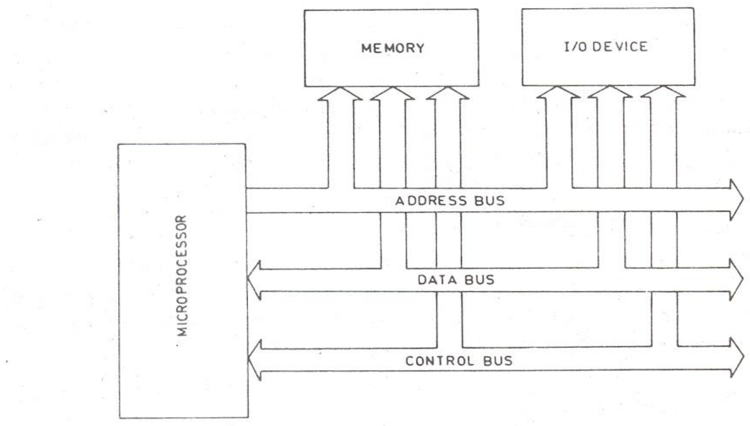
In this scheme, there is only one address space. Address space is defined as set of all possible addresses that a microprocessor can generate. Some addresses are assigned to memories and some addresses to I/O devices. An I/O device is also treated as a memory location and one address is assigned to it. One address is assigned to each memory location. The addresses for I/O devices are different from the addresses which have been assigned to memories and vice-versa. In this scheme, all the data transfer instructions of microprocessor can be used for both memory as well as I/O devices. This scheme is suitable for small systems.

I/O mapped I/O scheme

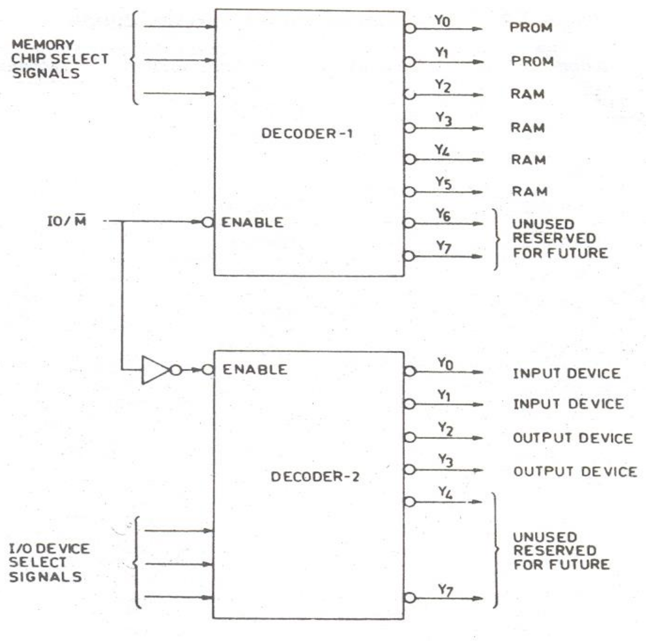
In this scheme, the addresses assigned to memory locations can also be assigned to I/O devices. Since the same address may be assigned to memory location or I/O devices, the microprocessor must issue a special signal to distinguish whether the address on the address bus is for a memory location or an I/O device. The Intel 8086 issues an $M/\overline{IO}$ signal for this purpose. When it is high, address on address bus is for an I/O device. If it is low, the address on the address bus is for a memory location. Two extra instructions IN and OUT are used to address I/O devices. The IN instruction is used to read data from an input device. The OUT instruction is used to send data to an output device. This scheme is suitable for large systems.

MEMORY AND I/O INTERFACING

Several memory chips and I/O devices are connected to a microprocessor. The following diagram shows a schematic diagram to interface memory chips or I/O devices to the microprocessor.



An address decoding circuit is employed to select the required I/O device or memory chip. The following diagram shows a schematic diagram of the decoding circuit.



: Interfacing of Memory and I/O Devices.

If IO/M' is high the DECODER-2 is activated and required I/O device is selected. If it is low, DECODER-1 is activated and required memory chip is selected. A few MSBs of address lines are applied to decoder to select a memory chip or an I/O device.

DATA TRANSFER SCHEMES

In a microprocessor-based system, data transfer takes place between two devices such as microprocessor and memory, microprocessor and I/O devices, and memory and I/O devices. Such a system has several I/O devices of different speed. A slow I/O device cannot transfer data when the microprocessor issues instruction for the same because it takes some time to get ready. To solve the problem of speed mismatch between a microprocessor and I/O devices a number of data transfer schemes have been developed. The data transfer schemes are classified into two broad categories:

- Programmed data transfer schemes
- DMA (Direct Memory Access) data transfer scheme.

Programmed data transfer schemes

These schemes are controlled by the central processing unit. Data are transferred from an I/O device to CPU which resides in memory. These programs are executed by CPU when an I/O device is ready to transfer data. The programmed data transfer schemes are employed when small amount of data are to be transferred. They are classified into following categories.

- Synchronous data transfer scheme
- Asynchronous data transfer scheme
- Interrupt driven data transfer scheme

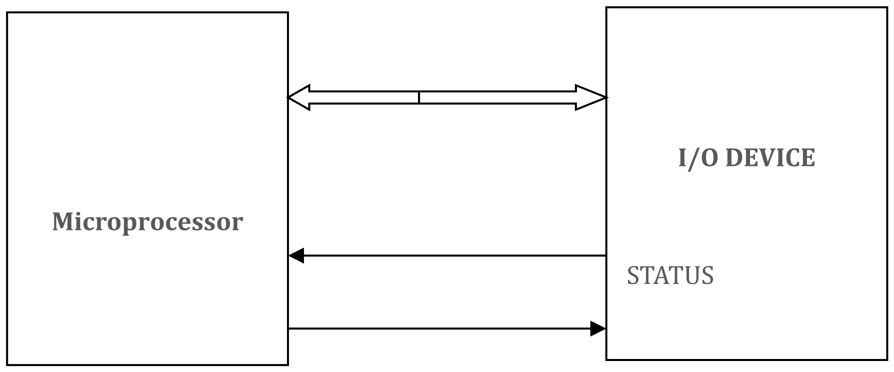
Synchronous Data Transfer

Synchronous means at the same time. The device which sends data and the device which receives data are synchronized with the same clock. When the CPU and I/O devices match in speed, this technique of the data transfer is employed. The data transfer with I/O device is performed executing IN or OUT instructions for I/O mapped I/O devices or using memory read/write instructions for memory mapped I/O devices. The IN instruction is used to read data from an input device or input port. The OUT instruction is used to send data from the CPU to an output device or output port. As the CPU and the I/O device match in speed, the I/O device is ready to transfer data when IN or OUT instruction is executed by the CPU. But I/O devices compatible with microprocessors in speed are usually not available. Hence this technique of data transfer is rarely used for I/O devices.

Asynchronous Data Transfer

This technique of data transfer is used when the speed of an I/O device does not match the speed of the microprocessor, and the timing characteristic of I/O device is not predictable. In this technique the status of the I/O device is checked by the microprocessor before the data are transferred. The microprocessor initiates the I/O device to get ready and then continuously checks the status of the I/O device till the I/O device becomes ready to transfer data. This mode of data transfer is called handshaking mode of transfer. In this handshaking mode of data transfer some signals are exchanged between the I/O device and the microprocessor before the actual data transfer takes place. The microprocessor issues an initiating signal to the I/O device to get ready. When an I/O device becomes ready it sends signals to the processor to indicate that it is ready. Such signals are called handshake signals.

The following diagram shows the schematic diagram for asynchronous data transfer. Asynchronous data transfer is used for slow I/O devices. This technique is inefficient technique because time is wasted.



Interrupt Driven Data Transfer

In this scheme, the microprocessor initiates an I/O device to get ready, and then it executes its main program instead of remaining in a program loop to check the status of the I/O device. When the I/O device becomes ready to transfer data, it sends a high signal to the microprocessor through a special input line called an interrupt line. In other words, it interrupts the normal processing sequence of the microprocessor. On receiving an interrupt, the microprocessor completes the current instruction at hand, and then attends the I/O device. It saves the contents of the program counter on the stack first, and then takes up a subroutine called ISS (Interrupt Service Subroutine). It executes ISS to transfer data from or to the I/O device. After completing the data transfer, the microprocessor returns back to the main program which it was executing before the interrupt occurred. Interrupt driven data transfer is used for slow I/O devices. It is an efficient technique as compare asynchronous data transfer scheme because precious time of the microprocessor is not wasted in waiting while an I/O device is getting ready.

DMA data transfer scheme

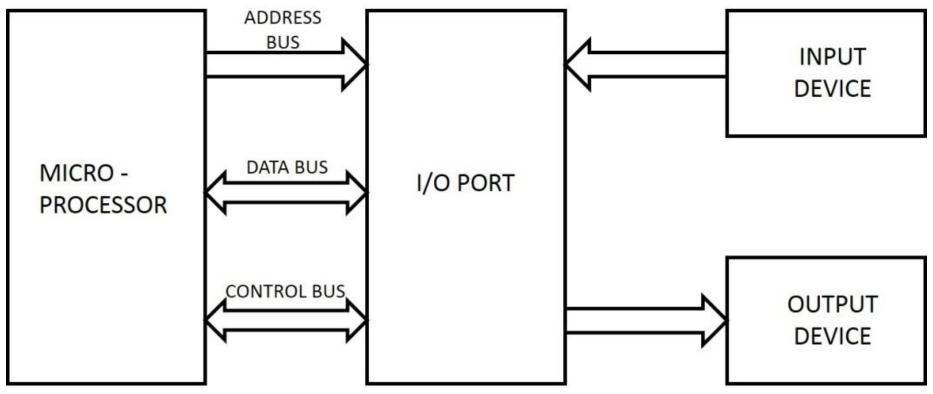
In DMA (Direct Memory Access) method of data transfer, the CPU does not participate. Data are directly transferred from an I/O device to memory or vice-versa. The data transfer is controlled by the I/O device or a DMA controller. It is employed when large amount of data are to be transferred. If bulk data are transferred through the CPU, it takes more time and the process becomes slow. An I/O device, which wants to send data using DMA technique, sends the HOLD signal to the CPU. On receiving the HOLD signal from an I/O device, the CPU gives the control of buses as soon as the current bus cycle is completed. The CPU sends a HLDA (Hold Acknowledge) signal to the I/O device to indicate that it has received the HOLD request and it has released the buses. The I/O device takes over the control of buses and directly transfers data to the memory or reads data from memory. When data transfer is over, CPU regains the control over the buses. There are two methods of DMA data transfer schemes:

Burst mode of DMA data transfer: A scheme of DMA data transfer, in which I/O device withdraws the DMA request only after all data bytes have been transferred, is called burst mode of data transfer. A block of data is transferred. It is employed by magnetic disk drives. In case of magnetic disks, data transfer cannot be stopped or slowed without loss of data.

Cycle Stealing Technique: In this method, after transferring one byte or several bytes, the I/O device withdraws DMA request. It reduces interference in CPU's activities. The interference can be eliminated completely by designing an interfacing circuitry which can steal bus cycle for DMA data transfer only when CPU is not using the system bus.

I/O PORTS

An input device is connected to microprocessor through an input port. An input port is a place for unloading data. An input device unloads data into port. The microprocessor reads data from input port. Similarly, an output device is connected to microprocessor through an output port. As the output port is connected to output device, data are transferred to output device. An I/O port may be programmable or non-programmable. A non-programmable port behaves as an input port if it has been designed and connected in input mode. A port connected in output mode acts as an output port. But, a programmable I/O port can be programmed to act either as an input port or output port.



Interfacing of I/O Devices through I/O Port

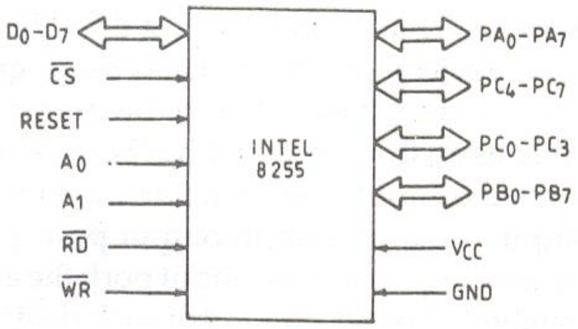
PROGRAMMABLE PERIPHERAL INTERFACE (PPI)

A programmable peripheral interface is a multiport device. The ports may be programmed in a variety of ways as required by the programmer. The device is very useful for interfacing peripheral devices. The term PIA called as peripheral Interface Adapter is also used by some manufacturer.

Intel 8255:

The Intel 8255 is a PPI. Its main function is to interface peripheral devices to microcomputer. It has three 8-bit ports, Port A, B and C. The Port C has been further divided into two of 4-bit ports, Port C upper and Port C lower. Thus a total of 4 ports are available, two 8-bit ports and two 4-bit ports. Each port can be programmed either as an input port or an output port. These four ports are further categorized into two Groups: Group-A and Group-B. Group-A consists of Port A and Port C_{upper}, and Group-B consists of Port-B and Port $C_{lower.}$

Pin Configuration of Intel 8255



Schematic Diagram of Intel 8255 A.

The pins for various ports are as follows:

$PA_0 – PA_7$	8 pins of Port A
$PB_0 – PB_7$	8 pins of Port B
$PC_0 – PC_3$	4 pins of Port C_{lower}
$PC_4 – PC_7$	4 pins of Port C_{upper}

The important control signals are as follows:

$\overline{CS}$: It is a chip select signal. The LOW status of this signal enables communication between the CPU and 8255.

$\overline{RD}$ (Read) : When it goes low the 8255 sends out data or status information to CPU on data bus. In other words, it allows CPU to read data from input port of 8255.

$\overline{WR}$ (Write) : When it goes low the CPU writes data or control word into 8255. The CPU writes data into output port of 8255 and control word into control word register.

A_0 and A_1 : The selection of input port and control word register is done using this in conjunction with $\overline{RD}$ and $\overline{WR}$. They are normally connected to LSBs of address bus.

A_1	A_0	Selected Port
0	0	Port A
0	1	Port B
1	0	Port C
1	1	Control Word Register

Operating modes of 8255

The 8255 has the following three modes of operation:

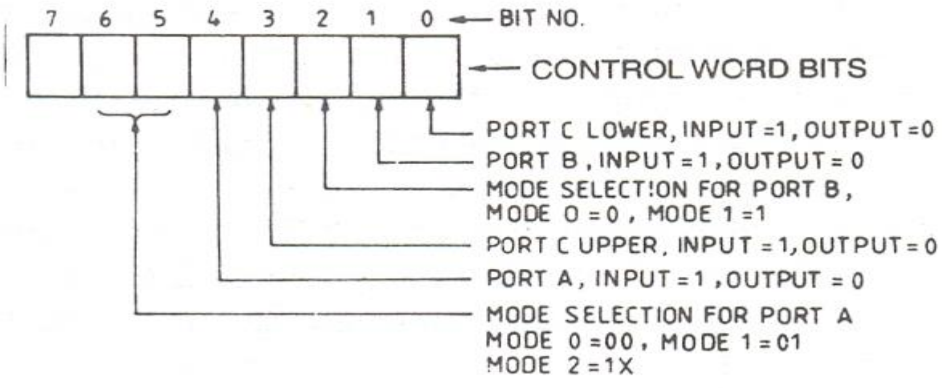
Mode-0 (Simple I/O): In this mode, a port can be operated as simple I/O port. Each of four ports of 8255 can be programmed to be either an input or output port. In this mode, the data transfer takes without any control signals.

Mode-1 (Strobed I/O): Mode-1 is a strobed input/output mode of operation. Port A and Port B both are designed to operate in this mode of operation. When Port A and Port B are programmed in Mode1, Port C is used for generating control signals.

Mode-2 (Bidirectional Port): Mode-2 is strobed bidirectional mode of operation. In this mode, only Port A can be programmed to operate as a bidirectional port. Port C is used for generating control signals. When Port A is programmed in Mode 2, the Port B can be used in either Mode-1 or Mode-0.

Control Word Format of 8255

According to the requirement a port can be programmed to act either as an input port or an output port. For programming the ports of 8255, a control word is formed. The bits of the control word are shown in the following diagram:


7 6 5 4 3 2 1 0 BIT NO.
CONTROL WORD BITS
PORT C LOWER, INPUT=1, OUTPUT=0
PORT B, INPUT=1, OUTPUT=0
MODE SELECTION FOR PORT B, MODE 0=0, MODE 1=1
PORT C UPPER, INPUT=1, OUTPUT=0
PORT A, INPUT=1, OUTPUT=0
MODE SELECTION FOR PORT A MODE 0=00, MODE 1=01 MODE 2=1X

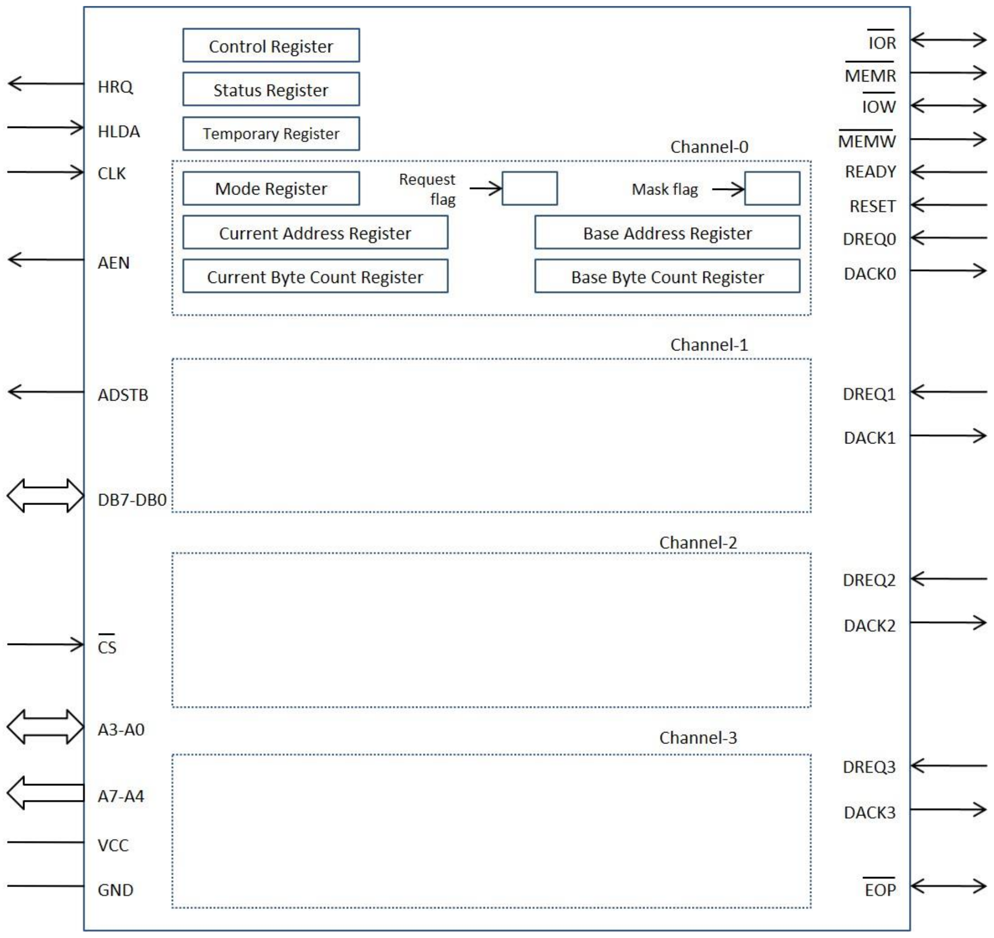
Control Word Bits for Intel 8255

The control word is written onto the control word register which is within the 8255. The control word bit corresponding to a particular port is set to either 1 or 0 depending upon the definition of the port. If a particular port to be made an input port, the bit corresponding to that port is set to 1. For making a port an output port, the corresponding bit for the port is set to 0. Bit 7 is set to 1 if Port A, B and C are defined as I/O port. It is set to 0 if the individual pins of the Port C are to be set or reset.

DMA CONTROLLER

The Intel 8237 is a programmable DMA controller. It is a 4-channel programmable Direct Memory Access (DMA) controller. It is in a 40 pin IC. Four I/O devices can be interfaced to the microprocessor through this device. It is capable of performing three operations, namely read, write and verify. During the *read* operation data are directly transferred from the memory to the I/O device. During the *write* operation data are transferred from the I/O device to the memory. On receiving a request from an I/O device, the 8237 generates a sequential memory address which allows the I/O device to read or write directly to or from the memory. Each channel incorporates two 16-bit registers, namely a DMA address register and a byte count register. These registers are initialised before a channel is enabled. Initially, the DMA address register is loaded with the address of the first memory location to be accessed. During DMA operation. it stores the next memory location to be accessed in the next DMA cycles. The byte count register stores the number of bytes to be transferred. Besides these registers the 8237 also includes a mode set register and a status register.

The following figure shows the pin configuration of Intel 8237 DMA controller.



The functions of its pins are as follows:

DREQ0-DREQ3: These are DMA request lines. An I/O device sends its DMA request on one of these lines. A HIGH status of the line generates a DMA request

DACK0-DACK3: These are DMA acknowledge lines. The Intel 8237 sends an acknowledge signal through one of these lines informing an I/O device that it has been selected for DMA data transfer. A LOW on the line acknowledges the I/O device.

A0-A7: These are address lines. A0-A3, are bidirectional lines. In the master mode these lines carry 4 LSBs of 16-bit memory address generated by the 8237. In the slave mode these lines are input lines. The inputs select one of the registers to be read or programmed. A4-A7 lines give tri-stated outputs which carry 4 through 7 of the 16-bit memory address generated by the 8237.

DB0-DB7: These are data lines. These are bidirectional three-state lines. While programming the controller the CPU sends data for the DMA address register, the byte count register and the mode set register through these data lines. During DMA cycle, the 8237 sends the 8 MSBs of the memory address through these lines at the beginning of the DMA cycle. Thereafter the data bus is made available to handle memory data transfer during rest of the DMA cycle.

ADSTB: A HIGH on this line latches the 8 MSBs of the address, which are sent on D-bus, into an address latch connected for this purpose.

$\overline{\textbf{CS}}$: It is an active-low chip select signal.

$\overline{\textbf{IOR}}$: It is a bidirectional line. In output mode it is used to access data from the I/O device during the DMA write cycle. In input mode, this line is used by CPU to read control registers of 8237.

$\overline{\textbf{IOW}}$: It is a bidirectional line. In output mode it allows the transfer of data to the I/O device during the DMA read cycle. Data is transferred from the memory. In input mode, this line is used by CPU to write information to 8237.

$\overline{\textbf{MEMR}}$: It is the active-low memory read signal, which is used to read the data from the addressed memory locations during DMA read operation.

$\overline{\textbf{MEMW}}$: It is the active-low memory write signal which is used to write the data to the addressed memory location during DMA write operation.

$\overline{\textbf{EOP}}$: End of Process. EOP goes low when data transfer is complete. The 8237 also allows an external signal to terminate an active DMA service through this line.

CLK: It is a clock signal which is required for the internal operation of 8237.

HRQ: This signal is used to receive the hold request signal from the output device.

HLDA: It is the hold acknowledgement signal which indicates the DMA controller that the bus has been granted to the requesting peripheral by the CPU.

RESET: It is an active-high input signal which clears the internal registers of 8237.

AEN: It is an Address latch Enable signal that enables the 8-bit latch containing the upper 8 address bits onto the system address bus.

READY: Ready is an input signal that is used to indicate that I/O peripheral device is ready to communicate with the 8237.

V_{CC}: +5V power supply signal (Two pins)

V_{SS}: Ground signal.

An I/O device sends its request for DMA transfer through one of the four DRQ lines. On receiving the DMA request for DMA data transfer from an I/O device, the Intel 8237 sends the hold request to the CPU through the HRQ line. The 8237 receives the hold acknowledge signal from the CPU through HLDA line. After receiving the hold acknowledge from the CPU, it sends DMA acknowledge to the I/O device through DACK line. The memory address is sent out on address and data lines. For DMA read cycle, in which data are transferred from memory to I/O devices, two control signals MEMR and IOW are issued by 8237. The MEMR enables the addressed memory for reading data from it. The IOW enables the I/O device to accept data. Similarly, for DMA write cycle, in which data are transferred from the I/O device to the memory, two control signals MEMW and IOR are issued by the controller. The MEMW enables the addressed memory for writing data to it. The IOR enables the I/O device to output data. The byte count is decremented by one after the transfer of one byte of data. When byte count becomes zero, a low signal is sent out through EOP indicating that the data transfer using DMA is complete. The four DMA channels are programmed either in a fixed priority mode or rotating mode of operation. READY line is used by slow memory or I/O devices.

Register of 8237:

Current Address Register: Each channel contains a 16-bit current address register. After each transfer the word count is decremented. It holds the memory address during DMA transfer. The address can be automatically incremented or decremented following each data transfer.

Current Word Count Register: Each channel contains a 16-bit current word count register. The number of transfers is determined by this register.

Base Address and Base Word Count Registers: Each channel is provided with these registers. These 16-bit registers hold the original value of the address and word count respectively.

In addition to above registers there are some more registers such as command register, mode register, request flag, mask flag, status register and temporary register.

DELAY SUBROUTINES

Delay Subroutine using one register:

A delay program is used to provide the desired delay in industrial control before issuing the control signal by the microcomputer. To generate delay a few registers of the microprocessor are loaded with desired numbers and then decremented to zero. The delay time depends on the numbers loaded in the registers. Illustrative examples of delay programs are given below.

```
                MOV   BL, 10H
NEXT:           DEC   BL
                JNZ   NEXT
                RET
```

To generate very small delay only one register can be used. In the above program register BL has been loaded by 10H (16 decimal). Then the register BL is decremented and the program moves in a loop till the content of register BL becomes zero. After this the program returns to the main program.

Delay time calculation:

The instruction JNZ takes 16 states when the content of register BL is not zero and the program jumps to the label NEXT. It takes only 4 states when the content of register BL has become zero and the program proceeds further to execute RET instruction. The instructions MOV BL, 10H and RET are executed only once. The instruction DEC BL is executed 16 times. The instruction JNZ is executed 16 times, out of which 15 times the program jumps to the label NEXT as the content of register BL has not become zero, and takes 16 states each time. At last, when the content of register BL becomes zero, JNZ is executed and the program proceeds to RET instruction. The last execution of instruction JNZ takes only 4 states.

To calculate the delay time, the number of times each instruction of the above program is executed, the number of states required for the execution of each instruction and the time required for one state are used. This is illustrated below:

Instruction	No. of times the instruction is executed in the program	No. of T-states
MOV BL, 10H	1	4 x 1 = 4
DEC BL	16	3 x 16 = 48
JNZ NEXT	16	16 x 15 + 4 x 1 = 244
RET	1	8 x 1 = 8
	Total No. of T-states	304

Clock frequency of Intel 8086 is 5 MHz

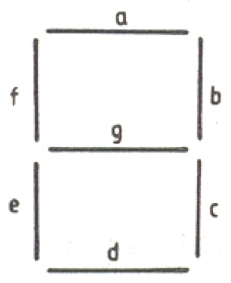
Time for one T-state for Intel 8086 = $1/(5 \times 10^6) = 0.2 \times 10^{-6}$ Second
Delate time for the above subroutine = $304 \times 0.2 \times 10^{-6}$ Second = 60.8×10^{-6} Second
= 60.8 μs

Delay Subroutine using three registers:

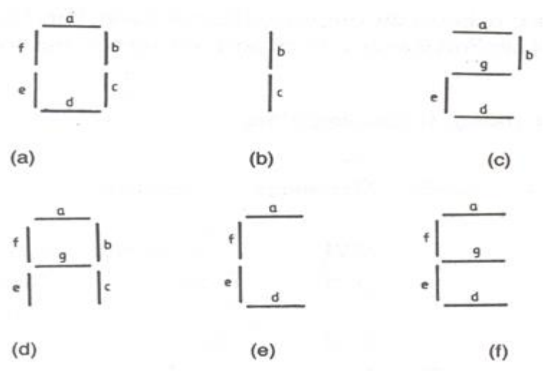
```
                MOV   BH, 50H
        LOOP1:  MOV   BL, FFH
        LOOP2:  MOV   DH, FFH
        LOOP3:  DEC   DH
                JNZ   LOOP3
                DEC   BL
                JNZ   LOOP2
                DEC   BH
                JNZ   LOOP1
                RET
```

7-SEGMENT LED DISPLAYS

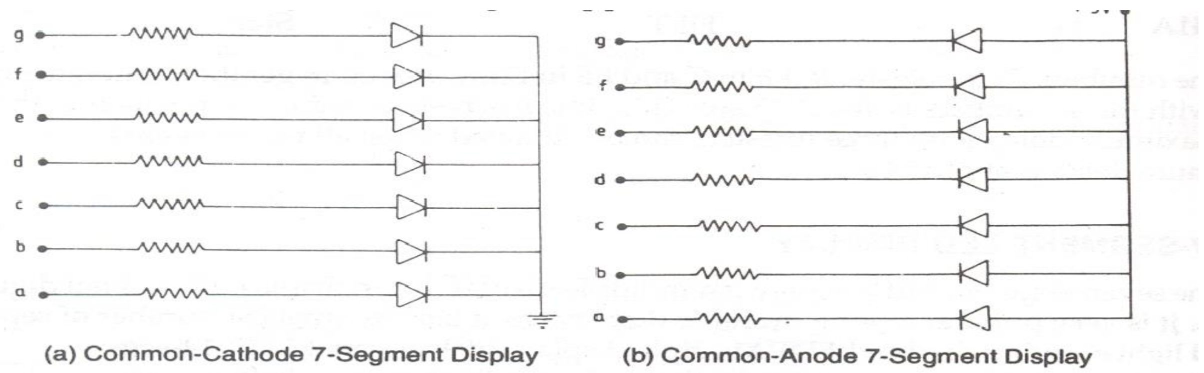
The seven-segment LED display is a multiple display. It can display all decimal digits and some letters. In seven-segment displays there are seven light emitting diodes (LED) as shown the following figure.



Each LED can be controlled separately. To display a digit or letter the desired segments are made ON as shown in the following figures.

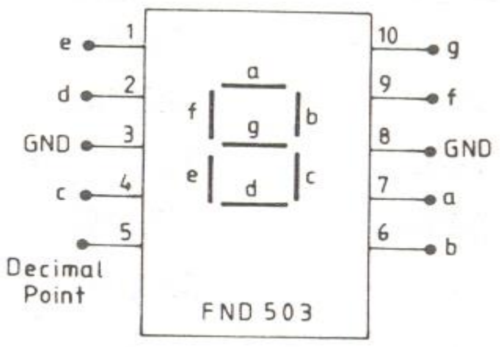


There are two types of 7-segment displays - namely, common-cathode type and common anode type. In a common-cathode type display, all the 7 cathodes of LEDs are tied together to the ground. When a +5V is applied to any segment the corresponding diode emits light. Thus, applying logic '1' i.e., positive logic to the desired segments, the desired letter or decimal number can be displayed. In a common-anode type display all the 7 anodes are tied together and connected to + 5 V supply. A particular segment will emit light when 0 logic is applied to it. The two types of 7-segment displays are illustrated in the figures below:



(a) Common-Cathode 7-Segment Display (b) Common-Anode 7-Segment Display

FND 500 and FND 503 are common-cathode 7-segment displays. The following diagram shows the pin configuration of FND 503. FND 507 and FND 510 are common-anode 7-segment displays. The pin configuration of FND 503 is shown in the figure below:



The seven-segment displays are not connected to I/O ports directly. They are connected through buffers or drivers / decoders.

Display of Alphanumeric Characters:

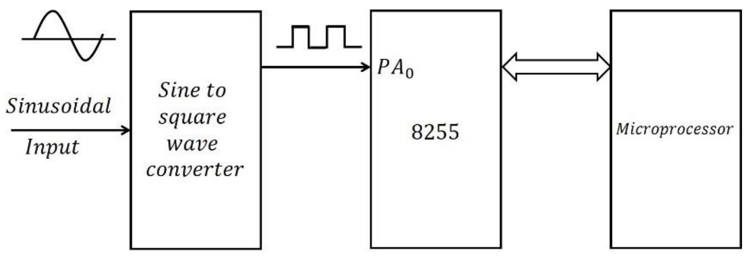
The program shown below illustrates the display of alphanumeric characters (0 to 9 and A to F). The codes required to display these 16 characters are assumed to be available in the memory locations from 2000H to 200FH. A delay subroutine is used to cause a delay after the display of each character. After displaying all the characters, the program will repeat the process of the displaying the characters again and again.

```
                MOV   AL, 98H
                OUT   CW#, AL
START:          MOV   CL, 10H
                MOV   SI, 2000H
NEXT:           MOV   AL, [SI]
                OUT   PB#, AL
                CALL  DELAY
                INC   SI
                DEC   CL
                JNZ   NEXT
                JMP   START

DELAY:          MOV   BH, 0FH
LOOP1:          MOV   BL, FFH
LOOP2:          MOV   CH, FFH
LOOP3:          DEC   CH
                JNZ   LOOP3
                DEC   BL
                JNZ   LOOP2
                DEC   BH
                JNZ   LOOP1
                RET
```

FREQUENCY MEASUREMENT

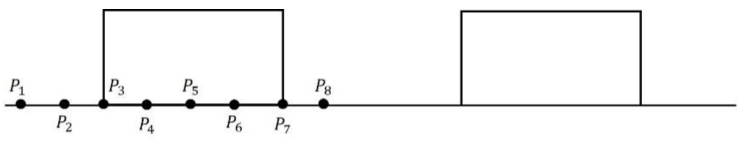
The interfacing circuit for frequency measurement is shown in the figure below.



Frequency Measurement

A sinusoidal signal is converted to square wave using a sine-to-square wave converter. The output of the converter is connected to PA_0 of 8255. The 8255 is connected to the microprocessor.

A program has been developed to sense the zero instant point of the rectified square wave. The microprocessor measures the magnitude of the square wave at two consecutive points as shown in Figure below. The two magnitudes are compared and decision is taken on the basis of carry and zero status flags, whether the point is at zero instant.



Rectified Square Wave

Suppose, the microprocessor takes reading at P_1 and P_2 where both magnitudes are zero. The difference of the two magnitudes is zero and hence, this is not the zero instant of the wave. At points P_5 and P_6, the difference of the two magnitudes is zero, so it is also not a zero instant point. The zero instant point is detected when P_2 and P_3 is compared. At P_7 and P_8, the difference is non-zero but there is carry. So, it is the end point of the half-square wave.

As soon as the zero instant point is detected the microprocessor initiates a register to count the number how many times the loop is executed. The microprocessor reads the magnitude of the square wave again and again, and moves in the loop. It crosses the loop when the magnitude of the square wave becomes zero. Thus, the time for half cycle is measured. The count can be compared with the stored numbers in a look-up table and the frequency can be displayed. The count which is inversely proportional to the frequency of the input signal can be used for further processing and control as desired.

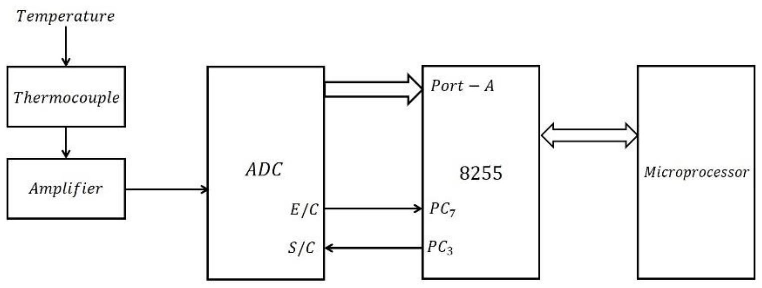
The program that detects the zero instant point and calculates the count is given below.

```
                MOV   AL, 98H
                OUT   CW#, AL
READ:           IN    AL, PA#
                MOV   BL, AL
                IN    AL, PA#
                CMP   AL, BL
                JZ    READ
                JC    READ
                MOV   SI, 0000H
AGAIN:          INC   SI
                IN    AL, PA#
                RCR   AL, 01
                JC    AGAIN
                HLT
```

The instruction, IN AL, PA#, receives the 1st reading of the magnitude of the square wave. MOV BL, AL transfers the 1st reading from the accumulator to the register BL. Again, IN AL, PA# takes the 2nd reading of the magnitude. CMP AL, BL compares these two readings. JZ READ indicates the condition that both readings are of equal magnitudes, i.e., readings are either 0, 0 or 1, 1. Therefore, it is not a condition of zero instant and the program jumps to the label READ. JC READ indicates that the 1st reading is 1 and the 2nd reading is 0. This is the condition for the end of rectified square wave. So, the program again jumps to the label READ. The program will move further only when the result is non-zero and there is no carry. This will be the case when the 1st reading is 0 and the 2nd reading is 1. This is the case of zero instant point. Once the zero instant is detected the program moves in a loop till the square wave exists. Thus, the time period for half cycle is measured. The count is inversely proportional to the frequency. If frequency is to be displayed, 7-segment displays can be interfaced and using look-up table technique it can be displayed. The program given above is only for the frequency measurement.

TEMPERATURE MEASUREMENT

The following figure shows a microprocessor-based scheme for temperature measurement and control.



Temperature Measurement

The output of a thermocouple proportional to the temperature in millivolt. It is amplified using an amplifier before it is processed by microprocessor. The amplified voltage is applied to an A/D converter. The microprocessor sends a start-of-conversion signal to the A/D converter through the port of 8255 PPI. When A/D converter completes conversion, it sends an end-of-conversion signal to the microprocessor. Having received an end-of-conversion signal from A/D converter the microprocessor reads the output of the A/D converter which is a digital quantity proportional to the temperature to be measured. The microprocessor displays the measured temperature. If the temperature is to be controlled, the microprocessor first measures its temperature, and then compares the measured temperature with a reference temperature at which the temperature is to be maintained. If the measured temperature is higher than the reference temperature, the microprocessor sends a control signal to reduce the temperature. If the measured temperature is less than the reference temperature, the microprocessor sends a control signal to increase the temperature.

Program:

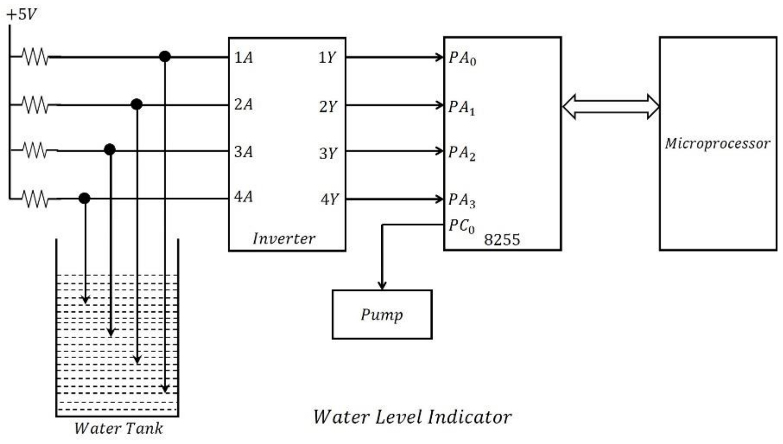
If the temperature is to be displayed in ^{o}C, a look-up table can be used.

```
                MOV   AL, 98H
                OUT   CW#, AL
                MOV   AL, 00H
                OUT   PC#, AL
                MOV   AL, 08H
                OUT   PC#, AL
READ:           IN    AL, PC#
                RCL   AL, 01
                JNC   READ
                IN    AL, PA#
                MOV   BX, 2000H
                MOV   CL, [BX]
SEARCH:         INC   BX
                CMP   AL, [BX]
                JC    GO
                JZ    GO
                DEC   CL
                JNZ   SEARCH
                JMP   START
GO:             INC   BH
                MOV   AL, [BX]
                CALL  DISPLAY
                JMP   START
```

(The DISPLAY subroutine displays the temperature on the 7-segment displays)

WATER LEVEL INDICATOR

A microprocessor-based water level indicator can be developed as shown in the figure below.



Water Level Indicator

The water tank is at ground potential. Probes are connected to 5V, through resistors. When a probe is immersed in water it is at ground potential and is at logic 0. When it is not immersed in water, it is at 5V potential and is at logic 1. These probes are connected to an inverter which inverts the logic of the probes. After inversion, the probe which is not immersed in water gives logic 0 and the probe immersed in water outputs logic 1. The outputs of the inverter are connected to Port-A of 8255. In the interface only four probes have been used. The 4 output points of the inverter are connected to 4 pins of Port-A. The microprocessor reads the binary logic corresponding to water level from the port, and displays it on 7-segment displays which are connected to Port-B.

The values of water levels at which probes are placed in the tank are stored in the memory in the form of a look-up table as shown below.

Address	Water level in cms
5000	00
5001	20
5002	40
5003	60
5004	80

The microprocessor counts how many probes are immersed in water and determines the LSB of the memory address of the look-up table. Suppose that the 1st, 2nd and 3rd probes are immersed in water. It picks up the water level corresponding to memory location FD03 which is 30 cm. The values of water levels may be stored in metres or cms depending on the actual situation. The values given in the program are for laboratory demonstration.

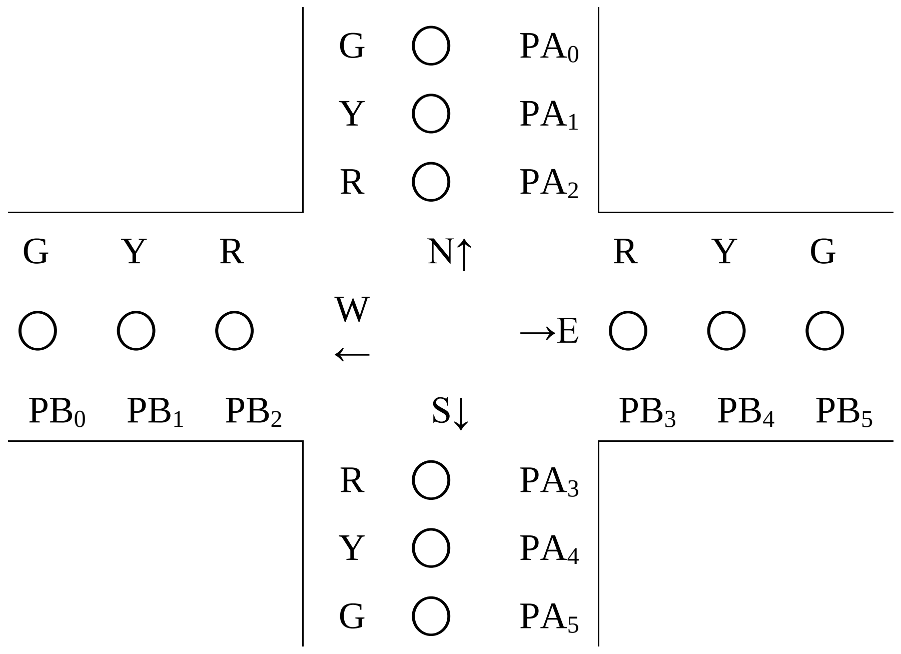
Automatic pumping can be done on the basis of water level in the tank. The program given below has been developed to switch on a pump when the water level is less than or equal to 10 cm. The pump is switched off when water level reaches 80 cm.

```
                MOV   AL, 98H
                OUT   CW#, AL
START:          MOV   BX, 2000H
                IN    AL, PA#
READ:           RCR   AL, 01
                JNC   NEXT
                INC   BL
                JMP   READ
NEXT:           MOV   AL, [BX]
                OUT   PB#, AL
                CALL  DELAY
                CMP   AL, 10H
                JNC   GO
                MOV   AL, 01H
                OUT   PC#, AL
                JMP   START
GO:             CMP   AL, 80H
                JC    START
                MOV   AL, 00H
                OUT   PC#, AL
                JMP   START

DELAY:          MOV   BH, 0FH
LOOP1:          MOV   BL, FFH
LOOP2:          MOV   CH, FFH
LOOP3:          DEC   CH
                JNZ   LOOP3
                DEC   BL
                JNZ   LOOP2
                DEC   BH
                JNZ   LOOP1
                RET
```

MICROPROCESSOR-BASED TRAFFIC CONTROL

Figure below shows a simple arrangement and port connections for microprocessor-based traffic control. All ports of 8255 have been programmed as output ports. The control word to make all the ports output ports in Mode 0 operation is 80H. The connection of pins of the ports to LED have been made through buffers (7407). Positive logic has been used to switch on LEDs. Three types of LEDs have been used Red (R), Yellow (Y) and Green (Y). Green light glows to allow crossing, Yellow to make alert, and Red does not allow crossing.



Port-A is used for the LEDs in the North-South direction and Port-B is used for the LEDs in the East-West direction. Green LEDs are connected to PA_0 and PA_5 in the North-South direction and to PB_0 and PB_5 in the East-West direction. Red LEDs are connected to PA_2 and PA_3 in the North-South direction and to PB_2 and PB_3 in the East-West direction. Yellow LEDs are connected to PA_1 and PA_4 in the North-South direction and to PB_1 and PB_4 in the East-West direction. The codes required to switch on the Green, Yellow and Red LEDs in both the directions are given in the table below:

LEDs	Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0	Code
Green	0	0	1	0	0	0	0	1	21
Yellow	0	0	0	1	0	0	1	0	12
Red	0	0	0	0	1	1	0	0	0C

Green lights for right turns have not been shown. One can add some more LEDs for this purpose, connect them to ports and make additions to the program. The program will become a longer one. Once the user understands the circuit and program illustrated in this section, the further extension is very easy. The procedure for the implementation of a simple traffic lights system is given below:

Step-1:	Initialize the I/O ports of 8255
Step-2:	Switch on RED lights in the North-South direction
Step-3:	Switch on GREEN lights in the East-West direction
Step-4:	Call Delay Subroutine (DELAY1) to display these lights for 20 seconds
Step-5:	Switch on YELLOW lights in both directions
Step-6:	Call Delay Subroutine (DELAY2) to display these lights for 5 seconds
Step-7:	Switch on RED lights in the East-West direction

Step-8:	Switch on GREEN lights in the North-South direction
Step-9:	Call Delay Subroutine (DELAY1) to display these lights for 20 seconds
Step-10:	Switch on YELLOW lights in both directions
Step-11:	Call Delay Subroutine (DELAY2) to display these lights for 5 seconds
Step-12:	Go to Step-2

The program for a traffic lights system is given below:

```
                MOV   AL, 80H
                OUT   CW#, AL
START:          MOV   AL, 0CH
                OUT   PA#, AL
                MOV   AL, 21H
                OUT   PB#, AL
                CALL  DELAY1
                MOV   AL, 12H
                OUT   PA#, AL
                OUT   PB#, AL
                CALL  DELAY2
                MOV   AL, 0CH
                OUT   PB#, AL
                MOV   AL, 21H
                OUT   PA#, AL
                CALL  DELAY1
                MOV   AL, 12H
                OUT   PA#, AL
                OUT   PB#, AL
                CALL  DELAY2
                JMP   START

DELAY1:         MOV   BH, 20H
LOOP1:          MOV   BL, FFH
LOOP2:          MOV   CH, FFH
LOOP3:          DEC   CH
                JNZ   LOOP3
                DEC   BL
                JNZ   LOOP2
                DEC   BH
                JNZ   LOOP1
                RET

DELAY2:         MOV   BH, 05H
                JMP   LOOP1
```