Manonmaniam Sundaranar University,

Directorate of Distance & Continuing Education,

Tirunelveli - 627 012 Tamilnadu, India

OPEN AND DISTANCE LEARNING (ODL) PROGRAMMES

(FOR THOSE WHO JOINED THE PROGRAMMES FROM THE ACADEMIC YEAR 2023–2024)

II YEAR

M.Sc. Physics

Course Material

Microprocessor 8085 and Microcontroller 8051

Prepared

By

Dr. S. Shailajha

Assistant Professor

Department of Physics

Manonmaniam Sundaranar University

Tirunelveli – 12

UNIT I	8085 PROGRAMMING, PERIPHERAL DEVICES AND THEIR INTERFACING
Instruction set – Addressing Modes – Programming techniques – Memory mapped I/O scheme – I/O mapped I/O scheme – memory and I/O Interfacing – Data Transfer schemes – Interrupts of 8085 – Programmable peripheral Interface (PPI)- Control group and control word-Programmable DMA controller – Programmable Interrupt controller-Programmable communication Interface- Programmable counter/Interval timer.	
UNIT – II	**8085 INTERFACING APPLICATIONS**
Seven segment display Interface – Interfacing of Digital to Analog converter and Analog to Digital Converter – Stepper motor Interface – Measurement of electrical quantities – voltage and current), Measurement of Physical quantities (Temperature and Strain).	
UNIT-III	**8051 MICROCONTROLLER HARDWARE**
Introduction-Features of 8051-8051 Microcontroller Hardware pin -Out 8051, Central Processing Unit (CPU), Internal RAM, Internal ROM, Register set of 8051- Memory organization of 8051-Input/Output pins, Ports and Circuits-External data memory and Program memory: External Program Memory, External data memory.	
UNIT-IV	**8051 INSTRUCTION SET AND ASSEMBLY LANGUAGE PROGRAMMING**
Addressing Modes-Data moving (data Transfer) Instructions: Instructions to Access external data memory, external ROM/Program memory, PUSH and POP instructions, Data exchange Instructions- Logical Instructions byte and bit level logical operations, Rotate and swap operations – Arithmetic instructions, Flags, Incrementing and decrementing, Addition, Subtraction, Multiplication and Division, Decimal Arithmetic – JUMP and CALL Instructions Jump and Call Program range, Jump, Call and subroutines – Programming.	
UNIT -V	**INTERUPPT PROGRAMMING AND INTERFACING TO EXTERNAL WORLD**
8051 Interrupts-Interrupt vector table-Enabling an disabling an interrupt – Timer Interrupts and Programming -Programming external hardware interrupts – serial communication interrupts and programming – Interrupt priority in the 8051: Nested interrupts, software trigerring of interrupt.LED interface seven segment display Interface – Interfacing of Digital to Analog converter and Analog to Digital Convertor – Stepper motor Interface- Measurement of electrical quantities – voltage and current), Measurement of Physical quantities (Temperature and Strain).	
UNIT VI	**PROFESSIONAL COMPONENTS**
Expert Lectures, Online Seminars-Webinars on Industrial Interactions/Visits, Competitive Examinations, Employable and Communication Skill Enhancement, Social Accountability and Patriotism.	

TEXT BOOKS

A. NAgoorkani, Microprocessors & Microcontrollers, RBA Publications (2009)

A.P.Godse and D.A.Godse, Microprocessors, Technical Publications, Pune (2009)

Ramesh Gaonkar, Microprocessor Architecture, Programming and 98 Applications with 8085, Penram International Publishing (2013)

B.Ram, Fundamentals of Microprocessors & Microcontrollers, DhanpatRai Publications New Delhi(2016)

V.Vijayendran, 2005, Fundamentals of Microprocessor – 8085, 3rd Edition S. Visvanathan Pvt.Ltd.

Unit 1 - 8085 Programming, Peripheral Devices and their Interfacing

INTRODUCTION

Microcomputer:

The term microcomputer is generally synonymous with personal computer, or a computer that depends on a microprocessor.

- Microcomputers are designed to be used by individuals, whether in the form of PCs, workstations or notebook computers.
- A microcomputer contains a CPU on a microchip (the microprocessor), a memory system (typically ROM and RAM), a bus system and I/O ports, typically housed in a motherboard.

Microprocessor:

A silicon chip that contains a CPU. In the world of personal computers, the terms microprocessor and CPU are used interchangeably.

- A microprocessor (sometimes abbreviated μP) is a digital electronic component with miniaturized transistors on a single semiconductor integrated circuit (IC).
- One or more microprocessors typically serve as a central processing unit (CPU) in a computer system or handheld device.
- Microprocessors made possible the advent of the microcomputer.
- At the heart of all personal computers and most working stations sits a microprocessor.
- Microprocessors also control the logic of almost all digital devices, from clock radios to fuel-injection systems for automobiles.
- Three basic characteristics differentiate microprocessors:
- Instruction set: The set of instructions that the microprocessor can execute.
- Bandwidth: The number of bits processed in a single instruction.
- Clock speed: Given in megahertz (MHz), the clock speed determines how many instructions per second the processor can execute.
- In both cases, the higher the value, the more powerful the CPU. For example, a 32-bit microprocessor that runs at 50MHz is more powerful than a 16-bit microprocessor that runs at 25MHz.

In addition to bandwidth and clock speed, microprocessors are classified as being either RISC (reduced instruction set computer) or CISC (complex instruction set computer).

8085 Microprocessor:

The Intel 8085 is an 8-bit microprocessor introduced by Intel in 1977. It was binary compatible with the more-famous Intel 8080 but required less supporting hardware, thus allowing simpler and less expensive microcomputer systems to be built. The "5" in the model number came from the fact that the 8085 requires only a +5-Volt (V) power supply rather than the +5 V, −5 V and +12 V supplies the 8080 needed. The main features of 8085 μP are:

- It is an 8-bit microprocessor.
- It is manufactured with N-MOS technology.
- It has 16-bit address bus and hence can address up to 216= 65536 bytes (64KB) memory locations through A_0–A_{15}.
- The first 8 lines of address bus and 8 lines of data bus are multiplexed AD_0–AD_7
- Data bus is a group of 8 lines D_0–D_7.
- It supports external interrupt request.
- A 16-bit program counter (PC)
- A 16-bit stack pointer (SP)
- Six 8-bit general purpose register arranged in pairs: BC, DE, HL.
- It requires a signal +5V power supply and operates at 3.2 MHZ single phase clock.
- It is enclosed with 40 pins DIP (Dual in line package).

Instruction sets are instruction codes to perform some tasks. It is classified into five categories.

Data Transfer Instructions: Includes the instructions that moves (copies) data between registers or between memory locations and registers. In all data transfer operations, the content of source register is not altered. Hence the data transfer is copying operation.

Arithmetic Instructions: Includes the instructions, which performs the addition, subtraction, increment or decrement operations. The flag conditions are altered after execution of an instruction in this group.

Logical Instructions: The instructions which performs the logical operations like AND, OR, EXCLUSIVE-OR, complement, compare and rotate instructions are grouped under this heading. The flag conditions are altered after execution of an instruction in this group.

Branching Instructions: The instructions that are used to transfer the program control from one memory location to another memory location are grouped under this heading.

Machine Control Instructions: Includes the instructions related to interrupts and the instruction used to halt program execution.

Data Transfer Instructions

- These instructions move data between registers, or between memory and registers.
- These instructions copy data from source to destination.
- While copying, the contents of source are not modified.

Opcode	Operand	Description
MOV	Rd, Rs M, Rs Rd, M	Copy from source to destination.
MVI	Rd, Data M, Data	Move immediate 8-bit
LDA	16-bit address	Load Accumulator
LDAX	B/D Register Pair	Load accumulator indirect
LXI	Reg. pair, 16-bit data	Load register pair immediate
STA	16-bit address	Store accumulator direct
STAX	Reg. pair	Store accumulator indirect
XCHG	None	Exchange H-L with D-E

Arithmetic Instructions

These instructions perform arithmetic operations such as addition, subtraction, increment, and decrement.

Opcode	Operand	Description
ADD	R M	Add register or memory to accumulator
ADC	R M	Add register or memory to accumulator with carry
ADI	8-bit data	Add immediate to accumulator
ACI	8-bit data	Add immediate to accumulator with carry
SUB	R M	Subtract register or memory from accumulator
SUI	8-bit data	Subtract immediate from accumulator
INR	R M	Increment register or memory by 1
INX	R	Increment register pair by 1
DCR	R M	Decrement register or memory by 1
DCX	R	Decrement register pair by 1

Logical Instructions

These instructions perform various logical operations with the contents of the accumulator

Opcode	Operand	Description
CMP	R M	Compare register or memory with accumulator
CMP	R M	Compare register or memory with accumulator
CPI	8-bit data	Compare immediate with accumulator
ANA	R M	Logical AND register or memory with accumulator
ANI	8-bit data	Logical AND immediate with accumulator
XRA	R M	Exclusive OR register or memory with accumulator
ORA	R M	Logical OR register or memory with accumulator
ORI	8-bit data	Logical OR immediate with accumulator
XRA	R M	Logical XOR register or memory with accumulator
XRI	8-bit data	XOR immediate with accumulator

Branching Instructions

This group of instructions alters the sequence of program execution either conditionally or unconditionally.

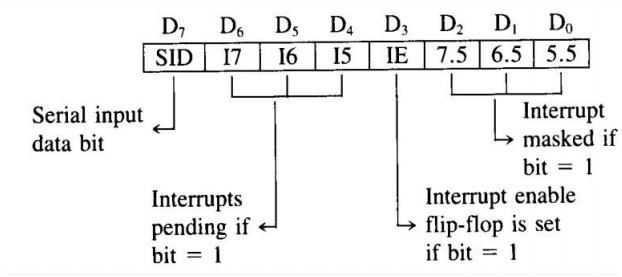
Opcode	Operand	Description
JMP	16-bit address	Jump unconditionally
Jx	16-bit address	Jump conditionally

Machine Control Instructions

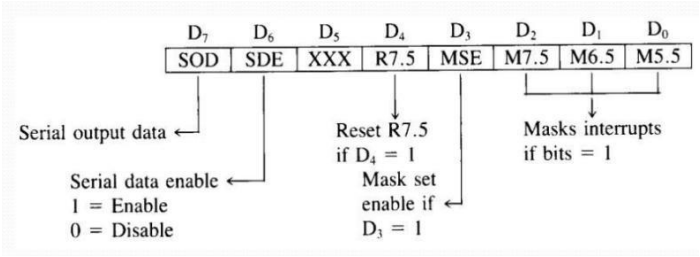
These instructions control machine functions such as Halt, Interrupt, or do nothing.

Opcode	Operand	Description
HLT	None	Halt
NOP	None	No operation

EI	None	The interrupt enable flip-flop is set and all interrupts are enabled. No flags are affected.
DI	None	The interrupt enable flip-flop is reset and all the interrupts except the TRAP are disabled. No flags are affected.
SIM	None	This is a multipurpose instruction and used to implement the 8085 interrupts 7.5, 6.5, 5.5, and serial data output.
RIM	None	This is a multipurpose instruction used to read the status of interrupts 7.5, 6.5, 5.5 and read serial data input bit.



SIM Instruction



RIM Instruction

Addressing Modes

Every instruction of a program has to operate on a data. The method of specifying the data to be operated by the instruction is called Addressing. The 8085 has the following 5 different types of addressing.

- Immediate Addressing
- Direct Addressing
- Register Addressing
- Register Indirect Addressing
- Implied Addressing

Immediate Addressing

In immediate addressing mode, the data is specified in the instruction itself. The data will be a part of the program instruction.

EX. MVI B, 3EH - Move the data 3EH given in the instruction to B register; LXI SP, 2700H.

Direct Addressing

In direct addressing mode, the address of the data is specified in the instruction. The data will be in memory. In this addressing mode, the program instructions and data can be stored in different memory.

EX. LDA 1050H - Load the data available in memory location 1050H in to accumulator; SHLD 3000H

Register Addressing

In register addressing mode, the instruction specifies the name of the register in which the data is available.

EX. MOV A, B - Move the content of B register to A register; SPHL; ADD C.

Register Indirect Addressing

In register indirect addressing mode, the instruction specifies the name of the register in which the address of the data is available. Here the data will be in memory and the address will be in the register pair.

EX. MOV A, M - The memory data addressed by H L pair is moved to A register. LDAX B.

Implied Addressing

In implied addressing mode, the instruction itself specifies the data to be operated. EX. CMA - Complement the content of accumulator; RAL

Programming techniques Memory Mapped I/O

There are two ways to interface 8085 with I/O devices in parallel data transfer mode:

- Memory Mapped IO
- IO Mapped IO

I/O devices are interfaced using address from memory space. That means IO device address are part of addresses given to memory locations.8085 uses 16-bit address to memory interfacing. So, any address between 0000H-FFFFH can be given to each peripheral. But the addresses given to peripheral can't be used for memory. Memory control signals are used as read and write control signals for peripherals. And all the operations that can be performed on memory can also be performed on peripherals. No need of using IO instructions such as IN, OUT.

IO mapped I/O

In this method separate address space is given to IO devices. Each IO device is given an 8-bit address. Hence maximum 256 devices can be interfaced to the processor. The address range for the IO devices is 00H-FFH. IO control signals are used to perform read, write operations. For reading data from IO device or writing data to IO device IN, OUT instructions need to be used.

IO mapped IO vs. Memory Mapped IO

Memory Mapped IO	IO mapped IO
IO is treated as memory.	IO is treated IO.
16-bit addressing.	8- bit addressing.
More Decoder Hardware.	Less Decoder Hardware.
Can address 2^{16}=64k locations.	Can address 2^{8}=256 locations.
Less memory is available.	Whole memory address space is available.
Memory Instructions are used.	Special Instructions are used like IN, OUT.
Memory control signals are used.	Special control signals are used.
Arithmetic and logic operations can be performed on data.	Arithmetic and logic operations cannot be performed on data.
Data transfer b/w register and IO.	Data transfer b/w accumulator and IO.

Arithmetic and logical operations can't be performed directly on IO devices as in memory mapped IO. IO devices can be interfaced, by using buffers for simple IO i.e. by using address decoding circuit to enable buffer. For handshake IO or to interface more peripherals ICs like 8255 peripheral programmable interfaces (PPI) can be used.

Memory Interfacing

The following are the steps involved in interfacing memory with 8085 processor.

1. First decide the size of memory requires to be interfaced. Depending on this we can say how many address lines are required for it. For example, if you want to interface 4KB (2^{12}) memory it requires 12 address lines. Remaining address lines can be used in address decoding.
2. Depending on the size of memory required and given address range, construct address decoding circuitry. This address decoding circuitry can be implemented with NAND gates and/or decoders or using PAL (when board size is a constraint).
3. Connect data bus of memory to processor data bus.
4. Generate the control signals required for memory using IO/M', WR', RD' signals of 8085 processor.

Interfacing I/O Devices

1. Using I/O devices data can be transferred between the microprocessor and the outside world.
2. This can be done in groups of 8 bits using the entire data bus. This is called parallel I/O.
3. The other method is serial I/O where one bit is transferred at a time using the SID and SOD pins on the Microprocessor.

The interfacing of output devices

Output devices are usually slow.

- Also, the output is usually expected to continue appearing on the output device for a long period of time.
- Given that the data will only be present on the data lines for a very short period (microseconds), it has to be latched externally.
- To do this the external latch should be enabled when the port's address is present on the address bus, the IO/M signal is set high and WR is set low.
- The resulting signal would be active when the output device is being accessed by the microprocessor.
- Decoding the address bus (for memory-mapped devices) follows the same techniques discussed in interfacing memory.

Interfacing of Input Devices

- The basic concepts are similar to interfacing of output devices.
- The address lines are decoded to generate a signal that is active when the particular port is being accessed.
- An IORD signal is generated by combining the IO/M and the RD signals from the microprocessor.
- A tri-state buffer is used to connect the input device to the data bus.
- The control (Enable) for these buffers is connected to the result of combining the address signal and the signal IORD.

Data transfer schemes

At the time of executing one 8085 program, interruption can be done in the mid-way by the virtue of the program by an Input Output device. Interruption can be done by the method according to which the processor works, since it wants urgent communication with the processor. The data transfer schemes always want for sending information to the processor, rather receiving information from the 8085 processor. It is so because sending and receiving information in the entire 8085 data transfer scheme process plays a vital role for executing the entire program rather process.

The communication is not done directly with the Input Output device. The communication processes are carried out systematically by the help of Input Output device by the virtue of an Input.

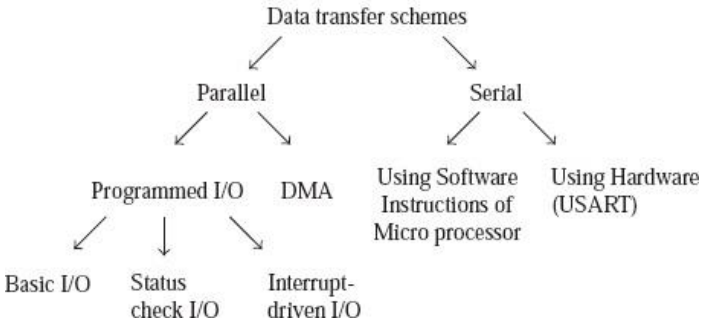
Output port. The data transfer can be either in two forms namely parallel or serial respectively. By the virtue of the Programmed Input Access or rather transferring data in parallel the data transfer

can be possible by using the Input Output programmed part or by Direct Memory Access (DMA) scheme. For transferring data in parallel we have three different ways by means of which microprocessor communicates with an Input or Output. They are:

Basic or simple data transfer scheme: The simplest data transfer scheme is the basic or simple data transfer. This method is beneficial to us when we have accurate know-ledge of the Input Output device timing characteristics.

Status check data transfer: Status check data transfer process is a much more complex process than simple data transfer. We use this method is used when there is lack of accurate knowledge of the Input Output device consisting of the timing characteristics. Status information is received by the processor regarding the readiness of the Input Output device for performing the data transfer.

Interrupt driven data transfer: We use this method when there lacks accurate knowledge of the timing characteristics of the Input Output device which takes maximum time for the device to be ready for use. Suppose we resort for the checking of data transfer; the processor here wastes a huge time in the loop for the device to get ready up to the mark.



Interrupt of 8085

Consider that a microprocessor is executing the main program. Now whenever the interrupt signal is enabled or requested the microprocessor shifts the control from main program to process the incoming request and after the completion of request, the control goes back to the main program. For example, an Input/output device may send an interrupt signal to notify that the data is ready for input. The microprocessor temporarily stops the execution of main program and transfers control to I/O device. After collecting the input data, the control is transferred back to main program. Interrupt signals present in 8085 are:

- INTR
- RST 7.5
- RST 6.5

- RST 5.5
- TRAP

INTR is maskable 8080A compatible interrupt. When the interrupt occurs the processor fetches from the bus one instruction, usually one of these instructions: One of the 8 RST instructions (RST0 - RST7). The processor saves current program counter into stack and branches to memory location N * 8 (where N is a 3 - bit number from 0 to 7 supplied with the RST instruction).
CALL instruction (3-byte instruction). The processor calls the subroutine, address of which is specified in the second and third bytes of the

RST5.5 is a maskable interrupt. When this interrupt is received the processor saves the contents of the PC register into stack and branches to 2CH (hexadecimal) address.
RST6.5 is a maskable interrupt. When this interrupt is received the processor saves the contents of the PC register into stack and branches to 34H (hexadecimal) address.
RST7.5 is a maskable interrupt. When this interrupt is received the processor saves the contents of the PC register into stack and branches to 3CH (hexadecimal) address.
TRAP is a non-maskable interrupt. When this interrupt is received the processor saves the contents of the PC register into stack and branches to 24H (hexadecimal) address.
All maskable interrupts can be enabled or disabled using EI and DI instructions. RST5.5, RST6.5 and RST7.5 interrupts can be enabled or disabled individually using SIM instruction.

NON-VECTORED INTERRUPT: The address of the service routine is not known in prior to the microprocessor. It is sent by the interrupting device. When the interrupt flipflop is enabled and INTR is high, microprocessor executes the current instruction and makes INTA low. Based on the flexibility to enable or disable interrupt, the interrupts are classified as maskable interrupt and non maskable interrupt. Maskable Interrupt: Even if the interrupt signals are high, microprocessor will respond to these signals only when interrupt flip flop is enabled. Example RST 7.5, RST 6.5, RST 5.5, INTR Non-Maskable Interrupt: Once the signal is enabled, the microprocessor immediately responds to this interrupt. Example: TRAP

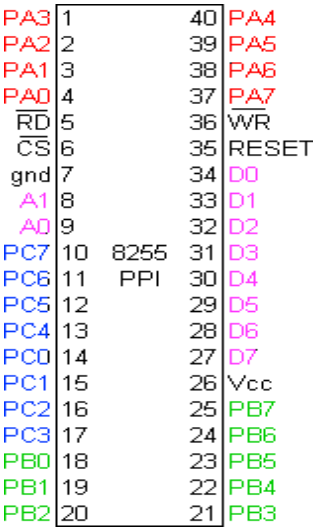
Programmable Peripheral Interface 8255

The 8255 is a widely used, programmable, parallel I/O device. It can be programmed to transfer data under various conditions, from simple I/O to interrupt I/O. It is flexible, versatile and economical and complex.

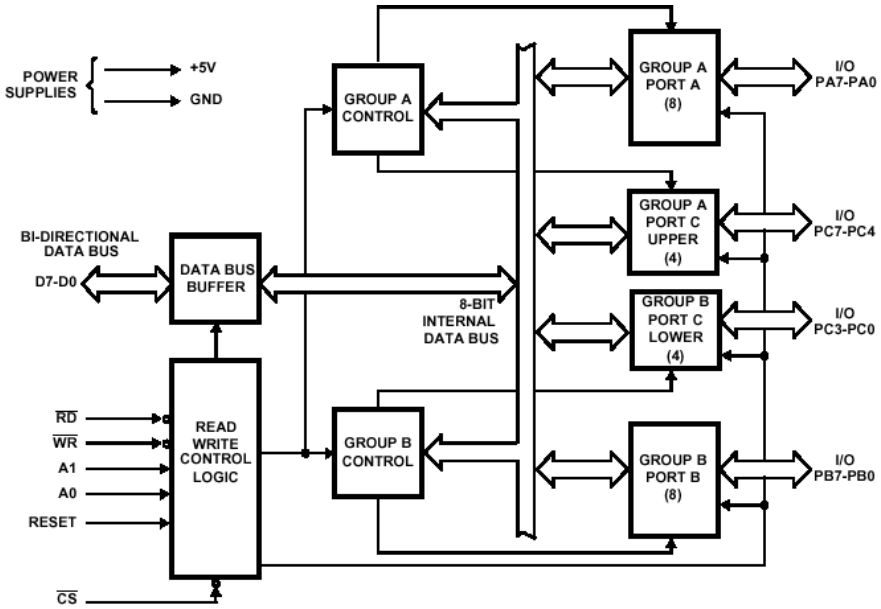
Features

Three 8-bit IO ports PA, PB, PC

- PA can be set for Modes 0, 1, 2. PB for 0,1 and PC for mode 0 and for BSR. Modes 1 and 2 are interrupt driven.
- PC has 2 4-bit parts: PC upper (PCU) and PC lower (PCL), each can be set independently for I or O. Each PC bit can be set/reset individually in BSR mode.
- PA and PCU are Group A (GA) and PB and PCL are Group B (GB)
- Address/data bus must be externally demultiplexed.
- TTL compatible.
- Improved dc driving capability



8255 Pin Diagram



8255 Block Diagram

Data Bus Buffer: This three-state bi-directional 8-bit buffer is used to interface the 8255 to the system data bus. Data is transmitted or received by the buffer upon execution of input or output instructions by the CPU. Control words and status information are also transferred through the data bus buffer.

Read/Write and Control Logic: The function of this block is to manage all of the internal and external transfers of both Data and Control or Status words. It accepts inputs from the CPU Address and Control busses and in turn, issues commands to both of the Control Groups.

(CS) Chip Select. A "low" on this input pin enables the communication between the 8255 and the CPU.

(RD) Read: A "low" on this input pin enables 8255 to send the data or status information to the CPU on the data bus. In essence, it allows the CPU to "read from" the 8255.

(WR) Write: A "low" on this input pin enables the CPU to write data or control words into the 8255.

(A0 and A1) Port Select 0 and Port Select 1: These input signals, in conjunction with the RD and WR inputs, control the selection of one of the three ports or the control word register. They are normally connected to the least significant bits of the address bus (A0 and A1).

(RESET) Reset. A "high" on this input initializes the control register to 9Bh and all ports (A, B, C) are set to the input mode.

A1	A0	Selection
0	0	Port A
0	1	Port B
1	0	Port C
1	1	Control

Group A and Group B Controls

The functional configuration of each port is programmed by the systems software. In essence, the CPU "outputs" a control word to the 8255. The control word contains information such as "mode", "bit set", "bit reset", etc., that initializes the functional configuration of the 8255. Each of the Control blocks (Group A and Group B) accepts "commands" from the Read/Write Control logic, receives "control words" from the internal data bus and issues the proper commands to its associated ports.

Ports A, B, and C

The 8255 contains three 8-bit ports (A, B, and C). All can be configured to a wide variety of functional characteristics by the system software but each has its own special features or "personality" to further enhance the power and flexibility of the 8255.

Port A One 8-bit data output latch/buffer and one 8-bit data input latch. Both "pull-up" and "pull-down" bus-hold devices are present on Port A.

Port B One 8-bit data input/output latch/buffer and one 8-bit data input buffer.

Port C One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input).

This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal output and status signal inputs in conjunction with ports A and B.

Operational modes of 8255

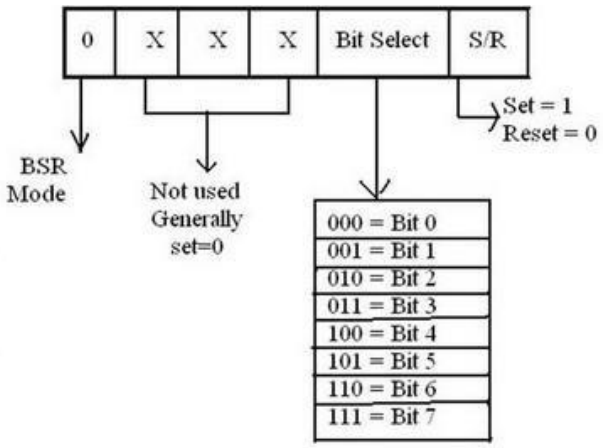
There are two basic operational modes of 8255:

- Bit set/reset Mode (BSR Mode).
- Input/output Mode (I/O Mode).

The two modes are selected on the basis of the value present at the D7 bit of the Control Word Register. When D7 = 1, 8255 operates in I/O mode and when D7 = 0, it operates in the BSR mode.

Bit set/reset (BSR) mode

The Bit Set/Reset (BSR) mode is applicable to port C only. Each line of port C (PC0 - PC7) can be set/reset by suitably loading the control word register. BSR mode and I/O mode are independent and selection of BSR mode does not affect the operation of other ports in I/O mode.



control word BSR mode

D_7 bit is always 0 for BSR mode.

Bits D_6, D_5 and D_4 are don't care bits.

Bits D_3, D_2 and D_1 are used to select the pin of Port C.

Bit D_0 is used to set/reset the selected pin of Port C.

Selection of port C pin is determined as follows

B3	B2	B1	Bit/pin of port C selected
0	0	0	PC0
0	0	1	PC1
0	1	0	PC2
0	1	1	PC3
1	0	0	PC4
1	0	1	PC5
1	1	0	PC6
1	1	1	PC7

Input/Output mode

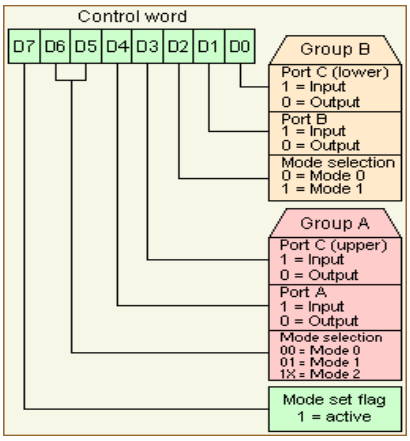
This mode is selected when D7 bit of the Control Word Register is 1. There are three I/O modes:

Mode 0 - Simple I/O

Mode 1 - Strobed I/O

Mode 2 - Strobed Bi-directional I/O

D_0, D_1, D_3, D_4 are assigned for lower port C, port B, upper port C and port A respectively. When these bits are 1, the corresponding port acts as an input port. For e.g., if $D_0 = D_4 = 1$, then lower port C and port A act as input ports. If these bits are 0, then the corresponding port acts as an output port. For e.g., if $D_1 = D_3 = 0$, then port B and upper port C act as output ports. D_2 is used for mode selection of Group B (port B and lower port C). When $D_2 = 0$, mode 0 is selected and when $D_2 = 1$, mode 1 is selected.



Control word I/O mode

D_5 & D_6 are used for mode selection of Group A (port A and upper port C).
The selection is done as follows:

D6	D5	Mode
0	0	0
0	1	1
1	X	2

As it is I/O mode, $D_7 = 1$.

Mode 0 - Simple I/O

In this mode, the ports can be used for simple I/O operations without handshaking signals. Port A, port B provide simple I/O operation. The two halves of port C can be either used together as an additional 8-bit port, or they can be used as individual 4-bit ports. Since the two halves of port C are

independent, they may be used such that one-half is initialized as an input port while the other half is initialized as an output port.

- The input/output features in mode 0 are as follows:
- Output ports are latched.
- Input ports are buffered, not latched.
- Ports do not have handshake or interrupt capability.
- With 4 ports, 16 different combinations of I/O are possible.

Mode 0 – input mode

- In the input mode, the 8255 gets data from the external peripheral ports and the CPU reads the received data via its data bus.
- The CPU first selects the 8255 chip by making CS' low. It then selects the desired port using A0 and A1 lines.
- The CPU then issues an RD' signal to read the data from the external peripheral device via the system data bus.

Mode 0 - Output mode

- In the output mode, the CPU sends data to 8255 via system data bus and then the external peripheral ports receive this data via 8255 port.
- CPU first selects the 8255 chip by making CS' low. It then selects the desired port using A0 and A1 lines.
- CPU then issues a WR' signal to write data to the selected port via the system data bus. This data is then received by the external peripheral device connected to the selected port.

Mode 1

When we wish to use port A or port B for handshake (strobed) input or output operation, we initialise that port in mode 1 (port A and port B can be initialised to operate in different modes, i.e., for e.g., port A can operate in mode 0 and port B in mode 1). Some of the pins of port C function as handshake lines.

For port B in this mode (irrespective of whether is acting as an input port or output port), PC0, PC1 and PC2 pins function as handshake lines. If port A is initialised as mode 1 input port, then, PC3, PC4 and PC5 function as handshake signals. Pins PC6 and PC7 are available for use as input/output lines.

The mode 1 which supports handshaking has following features:

- Two ports i.e. port A and B can be used as 8-bit i/o ports.
- Each port uses three lines of port c as handshake signal and remaining two signals can be used as i/o ports.
- Interrupt logic is supported.
- Input and Output data are latched.

Input Handshaking signals

1. IBF (Input Buffer Full) - It is an output indicating that the input latch contains information.
2. STB (Strobed Input)-The strobe input loads data into the port latch, which holds the information until it is input to the microprocessor via the IN instruction.
3. INTR (Interrupt request)-It is an output that requests an interrupt. The INTR pin becomes a logic 1 when the STB input returns to a logic 1, and is cleared when the data are input from the port by the microprocessor.
4. INTE (Interrupt enable)-It is neither an input nor an output; it is an internal bit programmed via the port PC4(port A) or PC2(port B) bit position.

Output Handshaking signals

1. OBF (Output Buffer Full)-It is an output that goes low whenever data are output (OUT) to the port A or port B latch. This signal is set to a logic 1 whenever the ACK pulse returns from the external device.
2. ACK(Acknowledge)-It causes the OBF pin to return to a logic 1 level. The ACK signal is a response from an external device, indicating that it has received the data from the 82C55 port.
3. INTR (Interrupt request)-It is a signal that often interrupts the microprocessor when the
4. External device receives the data via the signal. this pin is qualified by the internal INTE (interrupt enable) bit.

5. INTE (Interrupt enable)-It is neither an input nor an output; it is an internal bit programmed to enable or disable the INTR pin. The INTE A bit is programmed using the PC6 bit and INTE B is programmed using the PC2 bit.

DMA Controller

DMA Signals: HOLD, HLDA, READY

- **HOLD:** Indicates that another master is requesting the use of the address and data buses. The CPU, upon receiving the hold request, will relinquish the use of the bus as soon as the completion of the current bus transfer. Internal processing can continue. The processor can

regain the bus only after the HOLD is removed. When the HOLD is acknowledged, the Address, Data RD, WR and IO/M' lines are tri-stated.

- **HLDA:** Hold Acknowledge: Indicates that the CPU has received the HOLD request and that it will relinquish the bus in the next clock cycle HLDA goes low after the Hold request is removed. The CPU takes the bus one half-clock cycle after HLDA goes low.

- **READY:** This signal synchronizes the fast CPU and the slow memory, peripherals. If READY is high during a read or write cycle, it indicates that the memory or peripheral is ready to send or receive data. If READY is low, the CPU will wait an integral number of clock cycle for READY to go high before completing the read or write cycle. READY must conform to specified setup and hold times.

Reset Signals: Reset in, Reset Out

RESET IN: A low on this pin;

- Sets the program counter to zero (0000H)
- Resets the interrupt enables and HLDA flip-flops.
- Tri-states the data bus, address bus and control bus.
- Affects the content of processors internal registers randomly.

On Reset, The Program counter sets to 0000h which causes the 8085 to execute; the first instruction from address 0000H.

RESET OUT: This active high signal indicates that the processor; is being reset. This signal is synchronized to the processor clock and it can be used to reset other devices connected in the system.

Serial Input/output control

The input and output of serial data can be carried out using 2 instructions in 8085.

- SID-Serial Input Data
- SOD-Serial Output Data

Two more instructions are used to perform serial-parallel conversion needed for serial I/O devices.

- SIM
- RIM

Address buffer and Address-Data buffer

The contents of the stack pointer and program counter are loaded into the address buffer and address-data buffer. These buffers are then used to drive the external address bus and address-data bus. As the

memory and I/O chips are connected to these buses, the CPU can exchange desired data to the memory and I/O chips.

The address-data buffer is not only connected to the external data bus but also to the internal data bus which consists of 8-bits. The address data buffer can both send and receive data from internal data bus.

Address bus and Data bus

We know that 8085 is an 8-bit microprocessor. So, the data bus present in the microprocessor is also 8-bits wide. So, 8-bits of data can be transmitted from or to the microprocessor. But 8085 processor requires 16-bit address bus as the memory addresses are 16-bit wide. The 8 most significant bits of the address are transmitted with the help of address bus and the 8 least significant bits are transmitted with the help of multiplexed address/data bus. The eight- bit data bus is multiplexed with the eight

least significant bits of address bus. The address/data bus is time multiplexed. This means for few microseconds, the 8 least significant bits of address are generated, while for next few seconds the same pin generates the data. This is called Time multiplexing. But there are situations where there is a need to transmit both data and address simultaneously. For this purpose, a signal called ALE (address latch enables) is used. ALE signal holds the obtained address in its latch for a long time until the data is obtained and so when the microprocessor sends the data next time the address is also available at the output latch. This technique is called Address/Data demultiplexing.

Programmable Interrupt Controller

The Intel 8259A Programmable Interrupt Controller handles up to eight vectored priority interrupts for the CPU. It is cascadable for up to 64 vectored priority interrupts without additional circuitry. It is packaged in a 28-pin DIP, uses NMOS technology and requires a single a 5V supply. Circuitry is static, requiring no clock input. The 8259A is designed to minimize the software and real time overhead in handling multi-level priority interrupts. It has several modes, permitting optimization for a variety of system requirements. The PIC receives an interrupt request from an I/O device and tells the microprocessor. The CPU completes whatever instruction it is currently executing and then fetches a new routine that will service the requesting device. Once this peripheral service is completed, the CPU resumes doing exactly what it was doing when the interrupt request occurred. The PIC functions as an overall manager of hardware interrupt requests in an interrupt driven system environment.

Features

- 8 levels of interrupts.
- Can be cascaded in master-slave configuration to handle 64 levels of interrupts.
- Internal priority resolver, Fixed priority mode and rotating priority mode.
- Individually maskable interrupts.
- Modes and masks can be changed dynamically.

Programmable Interval Timer

The 8253 solves one of most common problem in any microcomputer system, the generation of accurate time delays under software control. Instead of setting up timing loops in system software, the programmer configures the 8253 to match his requirements, initializes one of the counters of the 8253 with the desired quantity, then upon command the 8253 will count out the delay and interrupt the CPU when it has completed its tasks.

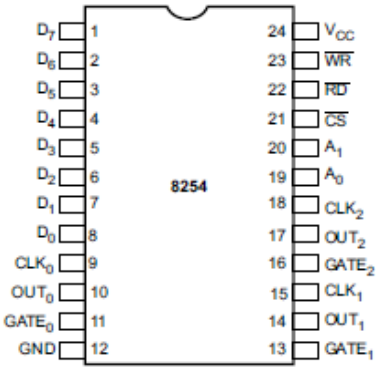
It is easy to see that the software overhead is minimum and that multiple delays can be easily be maintained by assignment of priority levels. The 8253 includes three identical 16-bit counters that can operate independently. To operate a counter, a 16-bit count is loaded in its register and, on command, it begins to decrement the count until it reaches 0. At the end of the count, it generates a pulse that can be used to interrupt the CPU. The counter can count either in binary or BCD. In addition, a count can be read by the CPU while the counter is decrementing.

Features

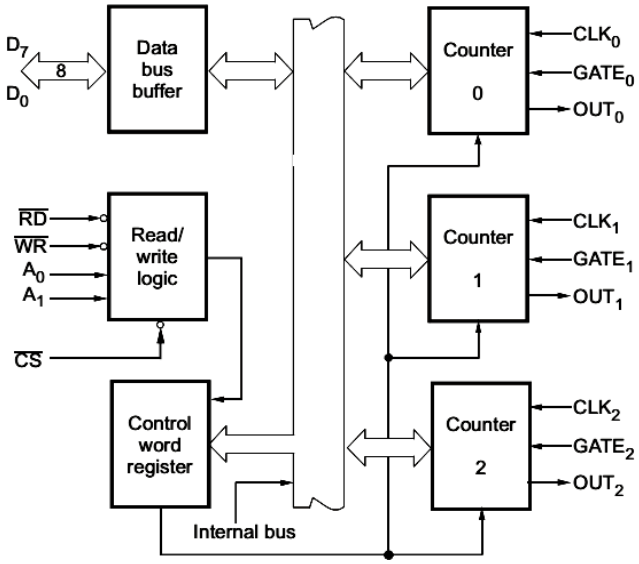
1. Three independent 16-bit down counters.
2. 8253 can operate from DC up to 2.6 MHz
3. Three counters are identical presettable, and can be programmed for either binary or BCD count.
4. Counter can be programmed in six different modes.
5. Compatible with all Intel and most other microprocessors.

It includes three counters, a data bus buffer, Read/Write control logic, and a control register. Each counter has two input signals CLOCK and GATE and one output signal OUT.

Pin Diagram



8254 Pin Diagram



8254 Block Diagram

Data Bus Buffer: This tri-state, bi-directional, 8-bit buffer is used to interface the 8253 to the system data bus. The Data bus buffer has three basic functions.

1. Programming the modes of 8253.
2. Loading the count registers.
3. Reading the count values.

Read/Write Logic: The Read/Write logic has five signals: RD, WR, CS and the address lines A0 and A1. In the peripheral I/O mode, the RD, and WR signals are connected to IOR and IOW, respectively. In memory-mapped I/O, these are connected to MEMR and MEMW. Address lines A0

and A1 of the CPU are usually connected to lines A0 and A1 of the 8253, and CS is tied to a decoded address. The control word register and counters are selected according to the signals on lines A0 and A1. It includes three counters, a data bus buffer, Read/Write control logic, and a control register. Each counter has two input signals CLOCK and GATE and one output signal OUT.

A_1	A_0	Selection
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Control word Register

Control Word Register: This register is accessed when lines A_0 and A_1 are at logic 1. It is used to write a command word which specifies the counter to be used (binary or BCD), its mode, and either a read or write operation.

Counters: These three functional blocks are identical in operation. Each counter consists of a single, 16-bit, pre-settable, down counter. The counter can operate in either binary or BCD and its input, gate and output are configured by the selection of modes stored in the control word register. The counters are fully independent. The programmer can read the contents of any of the three counters without disturbing the actual count in process.

Operation Description

The complete functional definition of the 8253 is programmed by the system software. Once programmed, the 8253 is ready to perform whatever timing tasks it is assigned to accomplish.

Mode 0: Interrupt on terminal count

1. The output will be initially low after the mode set operation.
2. After the count is loaded into the selected count Register the output will remain low and the counter will count.
3. When the terminal count is reached the output will go high and remain high until the selected count is reloaded.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC_1	SC_0	RW_1	RW_0	M_2	M_1	M_0	BCD

SC - Select counter

SC_1	SC_0	
0	0	Select counter 0
0	1	Select counter 1
1	0	Select counter 2
1	1	Illegal for 8253 Read -Back command for 8254 (See Read operations)

M - Mode

M_2	M_1	M_0	
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

RW - Read /Write

RW_1	RW_0	
0	0	Counter latch command (See Read operations)
0	1	Read / Write least significant byte only
1	0	Read / Write most significant byte only
1	1	Read / write least significant byte first, then most significant byte

BCD :

0	Binary counter 16 - bits
1	Binary coded decimal (BCD) Counter (4 Decades)

Mode 1: Hardware Retriggerable One-shot

1. The output will be initially high
2. The output will go low on the CLK pulse following the rising edge at the gate input.
3. The output will go high on the terminal count and remain high until the next rising edge at the gate input.

Mode 2: Rate generator

This mode functions like a divide by-N counter.

1. The output will be initially high.
2. The output will go low for one clock pulse before the terminal count.
3. The output then goes high, the counter reloads the initial count and the process is repeated.
4. The period from one output pulse to the next equals the number of input counts in the count register.

Mode 3: Square wave mode

1. Initially output is high.
2. For even count, counter is decremented by 2 on the falling edge of each clock pulse. When the counter reaches terminal count, the state of the output is changed and the counter is reloaded with the full count and the whole process is repeated.
3. If the count is odd and the output is high the first clock pulse (after the count is loaded) decrements the count by 1. Subsequent clock pulses decrement the clock by 2. After timeout, the output goes low and the full count is reloaded. The first clock pulse decrements the count by 3 and subsequent clock pulse decrement the count by two. Then the whole process is repeated. In this way, if the count is odd, the output will be high for (n+1)/2 counts and low for (n-1)/2 counts.

Mode 4: Software Triggered Strobe

1. The output will be initially high
2. The output will go low for one CLK pulse after the terminal count (TC).

Mode 5: Hardware triggered strobe (Retriggerable)

1. The output will be initially high.
2. The counting is triggered by the rising edge of the Gate.
3. The output will go low for one CLK pulse after the terminal count (TC).

Programming the 8253

Each counter of the 8253 is individually programmed by writing a control word into the control word register (A_0 - A_1 = 11). The above figure shows the control word format. Bits SC_1 and SC_0 select the counter, bits RW_1 and RW_0 select the read, write or latch command, bits M_2, M_1 and M_0 select the mode of operation and bit BCD decides whether it is a BCD counter or binary counter.

WRITE Operation

1. Write a control word into control register.
2. Load the low-order byte of a count in the counter register.
3. Load the high-order byte of count in the counter register.

READ Operation

In some applications, especially in event counters, it is necessary to read the value of the count in process. This can be done by following possible methods:

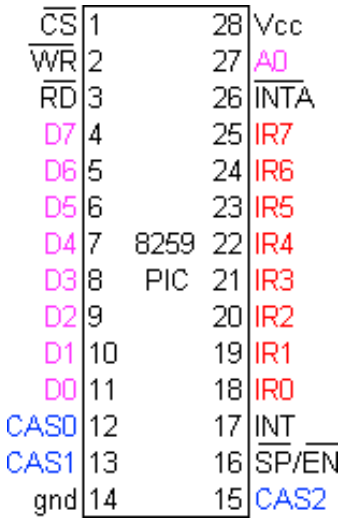
Simple Read: It involves reading a count after inhibiting the counter by controlling the gate input or the clock input of the selected counter, and two I/O read operations are performed by the CPU. The first I/O operation reads the low-order byte, and the second I/O operation reads the high order byte.

Counter Latch Command: In the second method, an appropriate control word is written into the control register to latch a count in the output latch, and two I/O read operations are performed by the CPU. The first I/O operation reads the low-order byte, and the second I/O operation reads the high order byte.

- Accepts IRQ, determines priority, checks whether incoming priority > current level being serviced, issues interrupt signal.
- In 8085 mode, provides 3-byte CALL instruction. In 8086 mode, provides 8-bit vector number.
- Polled and vectored mode.

- Starting address of ISR or vector number is programmable.
- No clock required.

Pin Diagram



8259 Pin Diagram

D_0-D_7	Bi-directional, tristate, buffered data lines. Connected to data bus directly or through buffers
RD-bar	Active low read control
WR-bar	Active low write control
A0	Address input line, used to select control register
CS-bar	Active low chip select
CAS0-2	Bi-directional, 3-bit cascade lines. In master mode, PIC places slave ID no. on these lines. In slave mode, the PIC reads slave ID no. from master on these lines. It may be regarded as slave- select.
SP-bar / EN-bar	Slave program / enable. In non-buffered mode, it is SP-bar input, used to distinguish master/slave PIC. In buffered mode, it is output line used to enable buffers
INT	Interrupt line, connected to INTR of microprocessor
INTA-bar	Interrupt ack, received active low from microprocessor
IR0-7	Asynchronous IRQ input lines, generated by peripherals.

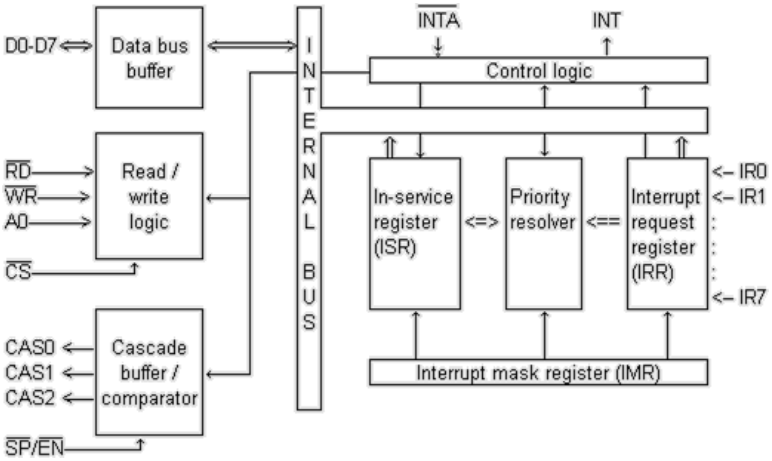
Interrupt Request Register (IRR): The interrupts at IRQ input lines are handled by Interrupt Request Register internally. IRR stores all the interrupt requests in it in order to serve them one by one on the priority basis.

In-Service Register (ISR): This register stores all the interrupt requests those are being served, i.e. ISR keeps a track of the requests being served.

Priority Resolver: This unit determines the priorities of the interrupt requests appearing simultaneously. The highest priority is selected and stored into the corresponding bit of ISR during

INTA pulse. The IR0 has the highest priority while the IR7 has the lowest one, normally in fixed priority mode. The priorities however may be altered by programming the 8259A in rotating priority mode.

Interrupt Mask Register (IMR): This register stores the bits required to mask the interrupt puts. IMR operates on IRR at the direction of the Priority Resolver.



Interrupt Control Logic: This block manages the interrupt and interrupt acknowledge signals to be sent to the CPU for serving one of the eight interrupt requests. This also accepts interrupt acknowledge (INTA) signal from CPU that causes the 8259A to release vector address on to the data bus.

Data Bus Buffer: This tristate bidirectional buffer interfaces internal 8259A bus to the microprocessor system data bus. Control words, status and vector information pass through buffer during read or write operations.

Read write Control Logic: This circuit accepts and decodes commands from the CPU. This also allows the status of the 8259A to be transferred on to the data bus.

Cascade Buffer/Comparator: This block stores and compares the ID's of all the 8259As used in the system. The three I/O pins CAS0-2 are outputs, when the 8259A is used as a master. The same pins act as inputs when the 8259A is in slave mode. The 8259A in master mode sends the ID of the interrupting slave device on these lines. The slave thus selected, will send its pre-programmed vector address on the data bus during the next INTA pulse.

Interrupt Sequence

The powerful features of the 8259A in a microcomputer system are its programmability and the interrupt routine addressing capability. The latter allows direct or indirect jumping to the specific interrupt routine requested without any polling of the interrupting devices. The normal sequence of events during an interrupt depends on the type of CPU being used. The events occur as follows in an 8085 system:

1. One or more of the INTERRUPT REQUEST lines (IR7–0) are raised high, setting the corresponding IRR bit(s).
2. The 8259A evaluates these requests, and sends an INT to the CPU, if appropriate.
3. The CPU acknowledges the INT and responds with an INTA pulse.
4. Upon receiving an INTA from the CPU group, the highest priority ISR bit is set, and the corresponding IRR bit is reset. The 8259A will also release a CALL instruction code (11001101) onto the 8-bit Data Bus through its D7–0 pins.
5. This CALL instruction will initiate two more INTA pulses to be sent to the 8259A from the CPU group.
6. These two INTA pulses allow the 8259A to release its preprogrammed subroutine address onto the Data Bus. The lower 8-bit address is released at the first INTA pulse and the higher 8-bit address is released at the second INTA pulse.
7. This completes the 3-byte CALL instruction released by the 8259A. In the AEOI mode the ISR bit is reset at the end of the third INTA pulse.
8. Otherwise, the ISR bit remains set until an appropriate EOI command is issued at the end of the interrupt sequence.

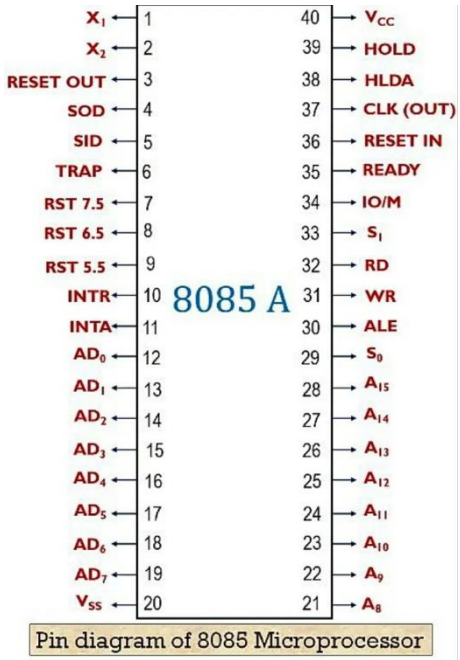
When the 8259A PIC receives an interrupt, INT becomes active and an interrupt acknowledge cycle is started. If a higher priority interrupt occurs between the two INTA pulses, the INT line goes inactive immediately after the second INTA pulse. After an unspecified amount of time the INT line is activated again to signify the higher priority interrupt waiting for service. This inactive time is not specified and can vary between parts.

II 8085 INTERFACING APPLICATIONS

Seven Segment display Interface

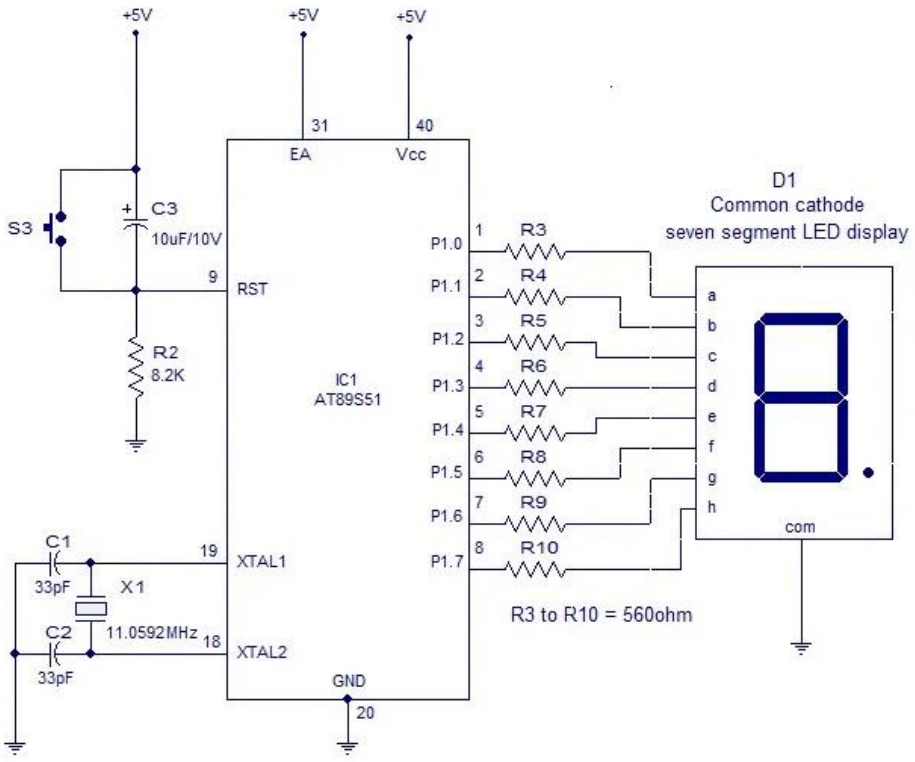
A seven-segment display to 8085 to display alphabet, number, signs or symbols e.t.c., Even the programming aspect of it which is done with assembly language will also be shown. But before we continue what is a microcontroller? microcontroller is a small computer on a single metal-oxide-semiconductor (MOS) integrated circuit (IC) chip or A microcontroller is an integrated circuit (IC) device used for controlling other portions of an electronic system, usually via a microprocessor unit (MPU), memory, and some peripherals. It has 40 pinout the name or functions of the pins are labelled on the picture below.

A seven-segment display is the most basic electronic display device that can display digits from 0–9. The most common configuration has an array of eight LEDs arranged in a special pattern to display these digits. They are laid out as a squared-off figure. Seven segment displays are of two types, common cathode and common anode. In common cathode type , the cathode of all LEDs are tied together to a single terminal and the anode of all LEDs are left alone as individual pins labeled as a, b, c, d, e, f, g & h (for dot). In common anode type, the anode of all LEDs are tied together as a single terminal and cathodes are left alone as individual pins. It will take a high (1) to turn a LED ON in common cathode and will take a low (0) to turn a LED ON in a common anode seven segment display.



Pin diagram of 8085 Microprocessor

The circuit diagram shown below is of an AT89S51 microcontroller based 0 to 9 counter which has a 7 segment LED display interfaced to it in order to display the count. This simple circuit illustrates two things. How to setup simple 0 to 9 up counter using 8051 and more importantly how to interface a seven segment LED display to 8051 in order to display a particular result. The common cathode seven segment display D1 is connected to the Port 1 of the microcontroller (AT89S51) as shown in the circuit diagram. R3 to R10 are current limiting resistors. S3 is the reset switch and R2,C3 forms a debouncing circuitry. C1, C2 and X1 are related to the clock circuit. The software part of the project has to do the following tasks.



Form a 0 to 9 counter with a predetermined delay (around 1/2 second here).

Convert the current count into digit drive pattern.

Put the current digit drive pattern into a port for displaying.

All the above said tasks are accomplished by the program given below.

Program.

```
ORG 000H //initial starting address
START: MOV A, #00001001B // initial value of accumulator
MOV B, A
MOV R0, #0AH //Register R0 initialized as counter which counts from 10 to 0
LABEL: MOV A, B
```

```
INC A
MOV B, A
MOVC A, @A+PC // adds the byte in A to the program counters address
MOV P1, A
ACALL DELAY // calls the delay of the timer
DEC R0//Counter R0 decremented by 1
MOV A, R0 // R0 moved to accumulator to check if it is zero in next instruction.
JZ START //Checks accu
```

ADC and DAC Interfacing:

The Analog to Digital Conversion is a quantizing process. Here the analog signal is represented by equivalent binary states. The A/D converters can be classified into two groups based on their conversion techniques.

In the first technique it compares given analog signal with the initially generated equivalent signal. In this technique, it includes successive approximation, counter and flash type converters. In another technique it determines the changing of analog signals into time or frequency. This process includes integrator-converters and voltage-to- frequency converters. The first process is faster but less accurate, the second one is more accurate. As the first process uses flash type, so it is expensive and difficult to design for high accuracy.

ADC 0808/0809 Chip

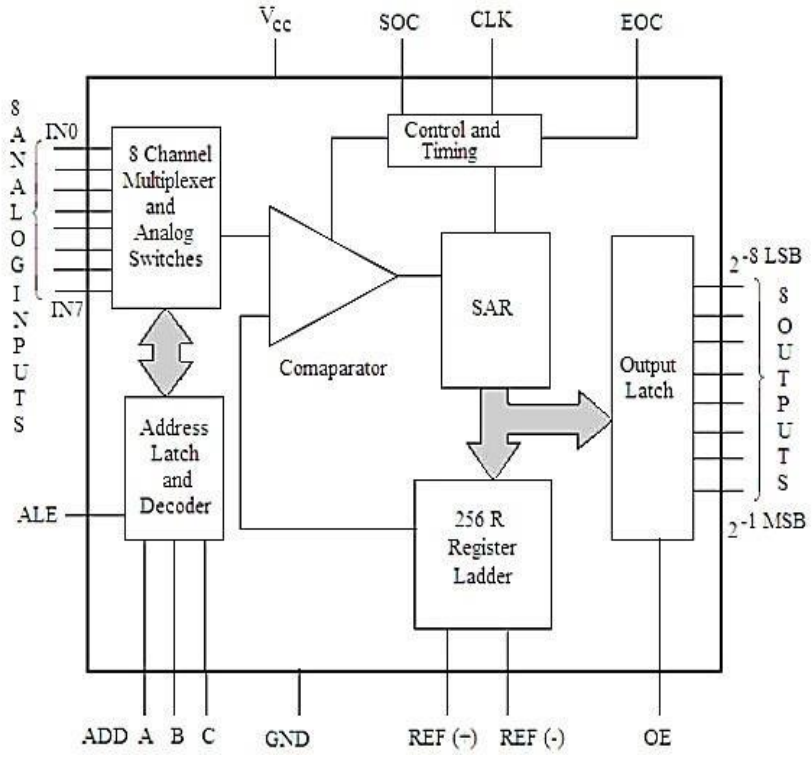
The ADC 0808/0809 is an 8-bit analog to digital converter. It has 8 channel multiplexers to interface with the microprocessor.

This chip is popular and widely used ADC. ADC 0808/0809 is a monolithic CMOS device. This device uses successive approximation technique to convert analog signal to digital form. One of the main advantages of this chip is that it does not require any external zero and full scale adjustment, only +5V DC supply is sufficient.

Let us see some good features of ADC 0808/0809:

- The conversion speed is much higher
- The accuracy is also high
- It has minimal temperature dependence

- Excellent long-term accuracy and repeatability
- Less power consumption

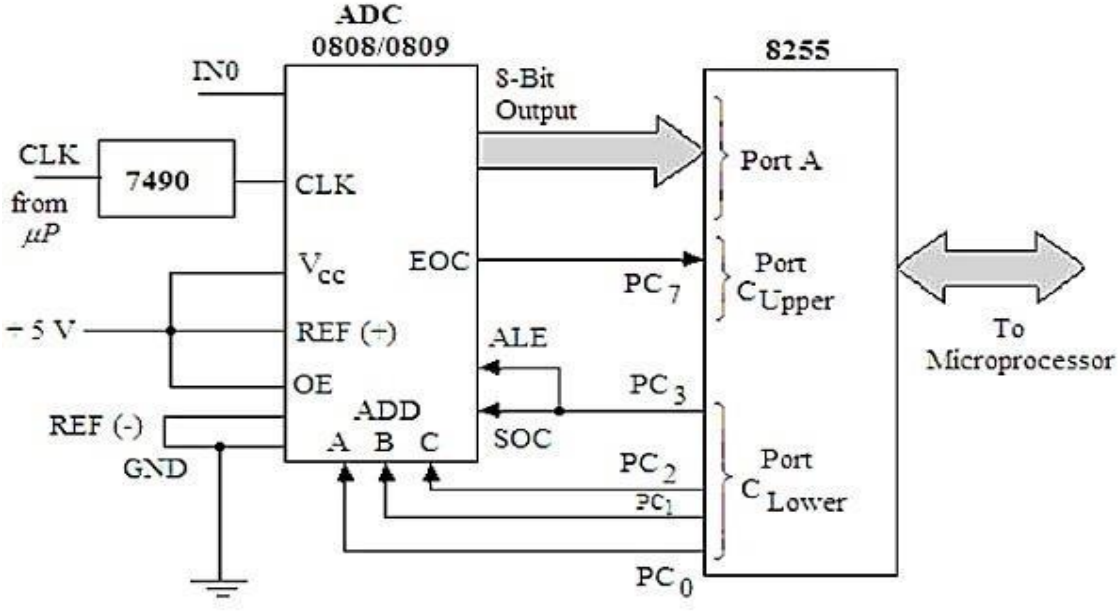


Architecture of ADC

[Source: "Microprocessor Architecture Programming and Application" by R.S. Gaonkar, page-]

Interfacing ADC with 8085 Microprocessor

To interface the ADC with 8085, we need 8255 Programmable Peripheral Interface chip with it. Let us see the circuit diagram of connecting 8085, 8255 and the ADC converter.



Interfacing ADC with 8085 Microprocessor

[Source: "Microprocessor Architecture Programming and Application" by R.S. Gaonkar]

The Port A of 8255 chip is used as the input port. The PC7 pin of Port Cupper is connected to the End of Conversion (EOC) Pin of the analog to digital converter. This port is also used as input port. The Clower port is used as output port. The PC2-0 lines are connected to three address pins of this chip to select input channels. The PC3 pin is connected to the Start of Conversion (SOC) pin and ALE pin of ADC 0808/0809.

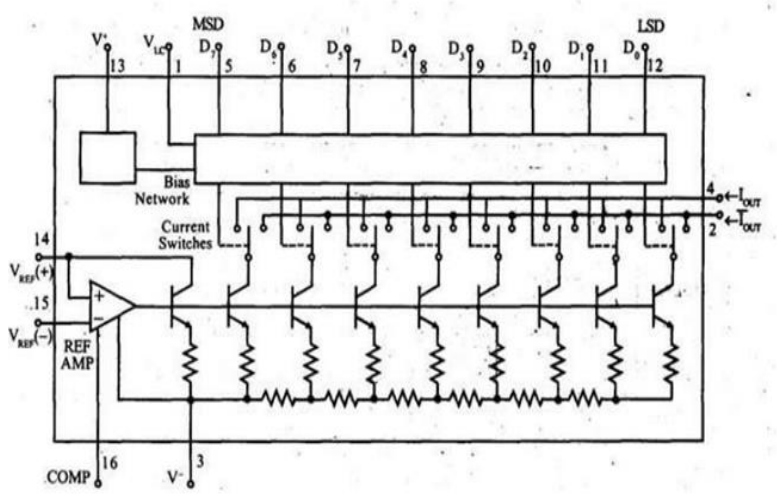
Now let us see a program to generate digital signal from analog data. We are using IN0 as input pin, so the pin selection value will be 00H.

MVI A, 98H; Set Port A and Cupper as input, C Lower as output OUT 03H; Write control word 8255-I to control Word register XRA A; Clear the accumulator
OUT 02 H; Send the content of Acc to Port Clower to select IN0
MVI A, 08H; Load the accumulator with 08H OUT 02H ; ALE and SOC will be 0
XRA A; Clear the accumulator
OUT 02H; ALE and SOC will be low. READ: IN 02H ; Read from EOC (PC7)
RAL; Rotate left to check C7 is 1.
JNC READ; If C7 is not 1, go to READ IN 00H ; Read digital output of ADC STA 8000H ; Save result at 8000H
HLT; Stop the program

DAC Interfacing with 8085 Microprocessor

- DAC 0800 Features
- To convert the digital signal to analog signal a Digital-to-Analog Converter (DAC) has to be employed.
- The DAC will accept a digital (binary) input and convert to analog voltage or current.
- Every DAC will have "n" input lines and an analog output.
- The DAC require a reference analog voltage (Vref) or current (Iref) source.
- The smallest possible analog value that can be represented by the n-bit binary code is called resolution.
- The resolution of DAC with n-bit binary input is 1/2nof reference analog value.
- The DAC0800 is an 8-bit, high speed, current output DAC with a typical settling time (conversion time) of 100 ns.

- It produces complementary current output, which can be converted to voltage by using simple resistor load.
- The DAC0800 require a positive and a negative supply voltage in the range of ± 5V to ±18V.



Circuit Diagram of 0800
[Source: "Microprocessor Architecture Programming and Application" by R.S. Gaonkar]

- It can be directly interfaced with TTL, CMOS, PMOS and other logic families.
- For TTL input, the threshold pin should be tied to ground (VLC = 0V).
- The reference voltage and the digital input will decide the analog output current, which can be converted to a voltage by simply connecting a resistor to output terminal or by using an op-amp I to V converter.
- The DAC0800 is available as a 16-pin IC in DIP.

Stepper motor Interfacing/Control using 8085 and 8051

Stepper Motor

A stepper motor is a device that translates electrical pulses into mechanical movement in steps of fixed step angle.

- The stepper motor rotates in steps in response to the applied signals.
- It is mainly used for position control.
- It is used in disk drives, dot matrix printers, plotters and robotics and process control circuits.

Square Wave Generation Using DAC 0800:

ADDRESS	LABEL	MNEMONICS	OPCODE
	START	MVIA,00H	
		OUT C8	
		CALL DISPLAY	
		MVI A,FF	
		OUT C8	
		CALL DELAY	
		JMP START	
		MVI B,05H	
		MVI C,FF	
	DELAY	DCR C	
	L2	JNZ L1	
		DCR C	
		JNZ L1	
		DCR B	
	L1	JNL L2	
		RET	

Structure

Stepper motors have a permanent magnet called rotor (also called the shaft) surrounded by a stator. The most common stepper motors have four stator windings that are paired with a center-tap. This type of stepper motor is commonly referred to as a four-phase or unipolar stepper motor. The center tap allows a change of current direction in each of two coils when a winding is grounded, thereby resulting in a polarity change of the stator.

Interfacing

Even a small stepper motor requires a current of 400 mA for its operation. But the ports of the microcontroller cannot source this much amount of current. If such a motor is directly connected to the microprocessor/microcontroller ports, the motor may draw large current from the ports and damage it. So, a suitable driver circuit is used with the microprocessor/microcontroller to operate the motor.

Motor Driver Circuit (ULN2003)

Stepper motor driver circuits are available readily in the form of ICs. ULN2003 is one such driver IC which is a High-Voltage High-Current Darlington transistor array and can give a current of 500mA.This current is sufficient to drive a small stepper motor. Internally, it has protection diodes used

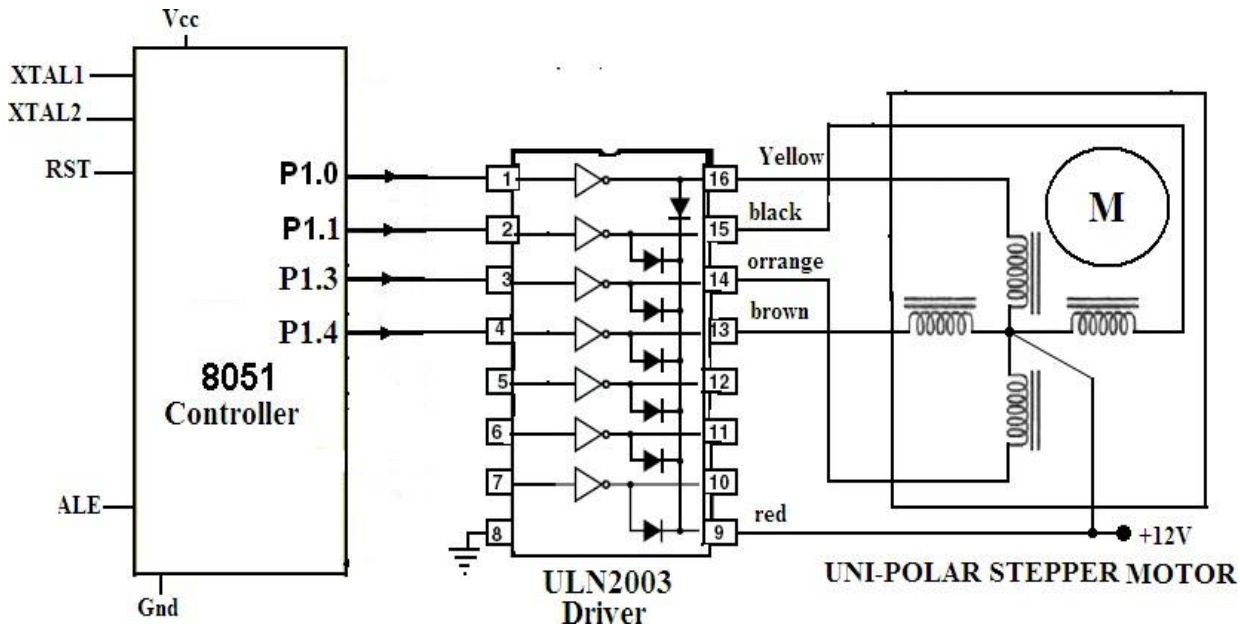
to protect the motor from damage due to back emf and large eddy currents. So, this ULN2003 is used as a driver to interface the stepper motor to the microcontroller.

Operation

The important parameter of a stepper motor is the step angle. It is the minimum angle through which the motor rotates in response to each excitation pulse. In a four-phase motor if there are 200 steps in one complete rotation then then the step angle is 360/200 = 1.8^{O}. So, to rotate the stepper motor we have to apply the excitation pulse. For this the controller should send a hexa decimal code through one of its ports. The hex code mainly depends on the construction of the stepper motor. So, all the stepper motors do not have the same Hex code for their rotation.

For example, let us consider the hex code for a stepper motor to rotate in clockwise direction is 77H , BBH , DDH and EEH. This hex code will be applied to the input terminals of the driver through the assembly language program. To rotate the stepper motor in anti-clockwise direction the same code is applied in the reverse order.

Stepper Motor interface- Schematic Diagram (for 8051)

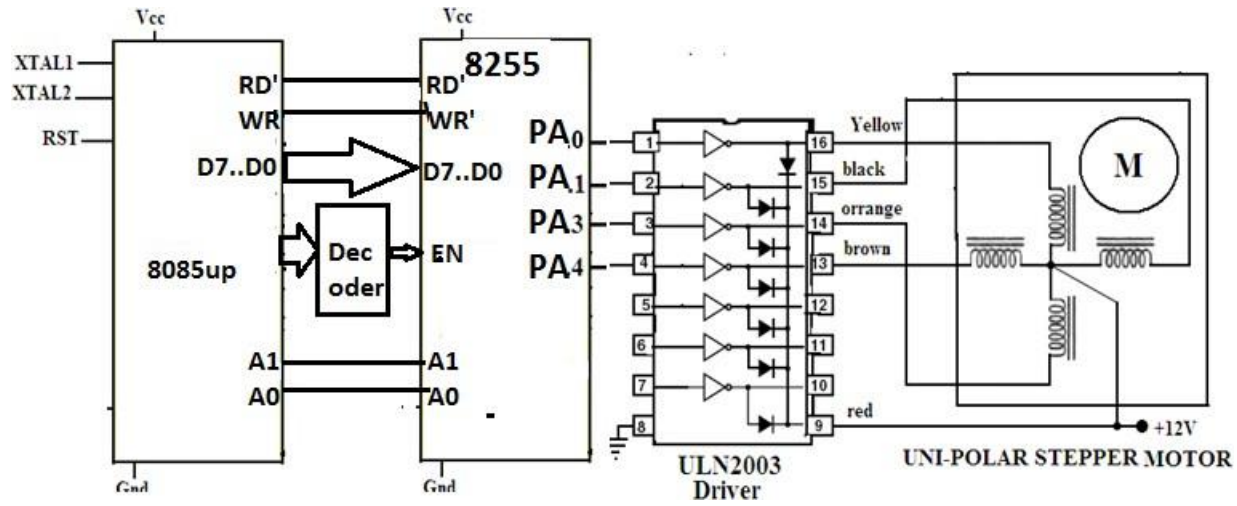


The assembly language program for 8051 is given below.

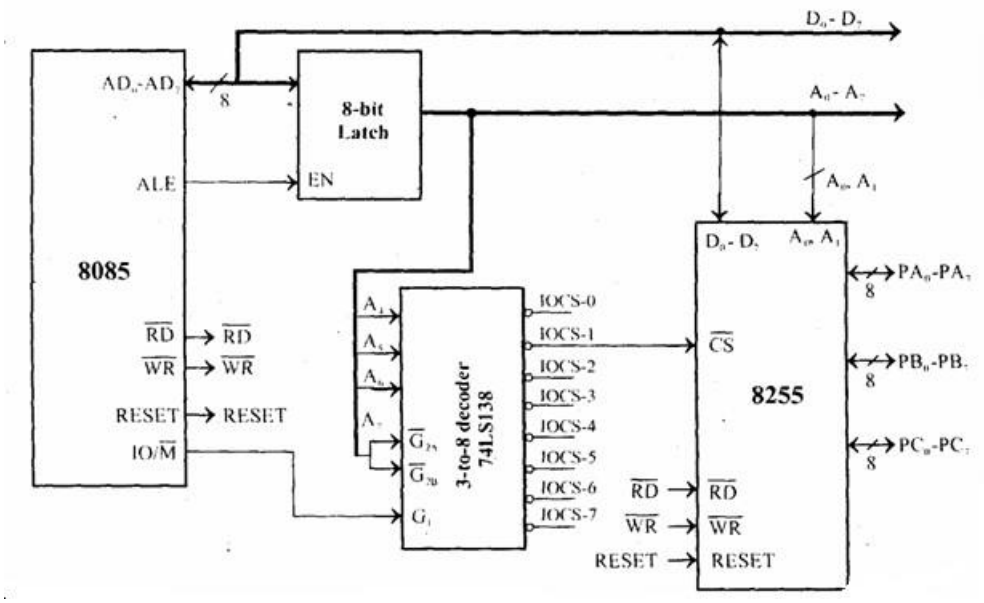
ASSEMBLY LANGUAGE PROGRAM (8051)

```
Main  : MOV A, # 0FF H        ;  Initialization of Port 1
        MOV P1, A             ;
        MOV A, #77 H          ; Code for the Phase 1
        MOV P1, A             ;
        ACALL DELAY           ;  Delay subroutine
        MOV A, # BB H         ; Code for the Phase II
        MOV P1, A             ;
        ACALL DELAY           ;  Delay subroutine.
        MOV A, # DD H         ; Code for the Phase III
        MOV P1, A             ;
        ACALL DELAY           ; Delay subroutine
        MOV A, # EE H         ;  Code for the Phase 1
        MOV P1, A             ;
        ACALL DELAY           ;  Delay subroutine SJMP
        MAIN; Keep the motor rotating continuously.
              DELAY Subroutine
        MOV R4, #0FF H        ;  Load R4 with FF
        MOV R5, # 0FF         ; Load R5 with FF
LOOP1:  DJNZ R4, LOOP1        ; Decrement R4 until zero,wait LOOP2:
DJNZ R5, LOOP2                ; Decrement R5 until zero,wait
        RET                   ; Return to main program
```

Stepper Motor interface - Schematic Diagram for (8085)



Detailed Connection diagram between 8085 and 8255



ASSEMBLY LANGUAGE PROGRAM (8085)

```
Main  : MVI A, 80          ;  80H → Control word to configure PA,PB,PC in O/P
        OUT CWR_Address    ; Write control word in CWR of 8255
        MVI A, 77          ; Code for the Phase 1
        OUT PortA_Address  ; sent to motor via port A of 8255  ;
        CALL DELAY         ;  Delay subroutine
        MVI A,  BB         ; Code for the Phase II
        OUT PortA_Address  ; sent to motor via port A of 8255
        CALL DELAY         ;  Delay subroutine.
        MVI A,  DD         ; Code for the Phase III
        OUT PortA_Address  ; sent to motor via port A of 8255;
        CALL DELAY         ; Delay subroutine
        MVI A,  EE H       ;  Code for the Phase 1
        OUT PortA_Address  ; sent to motor via port A of 8255  ;
        CALL DELAY         ;  Delay subroutine
        JMP MAIN           ; Keep the motor rotating continuously.
```

DELAY Subroutine

```
MVI C, FF          ;  Load C with FF  -- Change it for the speed variation LOOP1:
MVI D,FF           ;  Load D with FF
LOOP2:  DCR D
JNZ LOOP2 DCR C
```

```
JNZ LOOP1
RET                    ; Return to main program .
```

Basic electrical quantities

The basic electrical quantities are electrical current and voltage, electrical charge, resistance, capacitance, inductance and electric power. Electricity is a flow of free electrons carrying negative electric charge from the place with their excess (place with negative charge) to the place with their deficiency (place with positive charge). 2.1. Electrical current and charge According the convention, the positive direction of electric current is opposite, i.e. the positive direction of electric current is from the place with positive charge to the place with negative charge. The symbol for electric current is I (or i if the current is time varying) and the basic unit of measure is ampere (symbol A) after André-Marie Ampère. The ampere is one of seven basic units according to international convention (SI), which definition is: "the ampere is that constant current which, if maintained in two straight parallel conductors of infinite length, of negligible circular cross-section, and placed 1 meter apart in vacuum, would produce between these conductors a force equal to 2×10^{-7} newton per meter of length". Another simpler definition comes from the fact that electric current is movement of elementary electric charges carried by electrons. When there is a constant flow of approximately $1.602176487 \times 10^{19}$ electrons per second trough a surface, the current of 1 amper is flowing.

The unit of measure charge is coulomb (symbol C) after Charles Augustin de Coulomb. One coulomb is approximately 1.602176487×1019 of elementary charges. The SI definition of coulomb is "the coulomb is the quantity of electricity carried in 1 second by a current of 1 ampere". Although electrical charge is one of basic electrical quantities it is measured very rarely in praxis and if needed usually only calculated from measurement of other electrical quantities.

Electrical voltage Electrical voltage is a difference of potential between two places with different charges. Voltage provides the ability to move charges and hence: do a work and therefore voltage is also sometimes called electromotive force (EMF). The symbol for voltage is V or sometimes U (v or u if the voltage is time varying quantities) and the unit of measure is volt (V) after Alessandro Volta. The SI definition is: "The volt is the potential difference between two points of a conducting wire carrying a constant current of 1 ampere, when the power dissipated between these points is equal to 1 watt". Electrical voltage and current are manifestation of electrical charge movement and the can be supposed to be "active" quantities. They can carry information in electronic circuits and systems or they can be supposed to be only expression of supplied and consumed electrical energy. Measurement methods and instrumentation as well as measured parameters differ from a point of view where and why the voltage and current are measured.

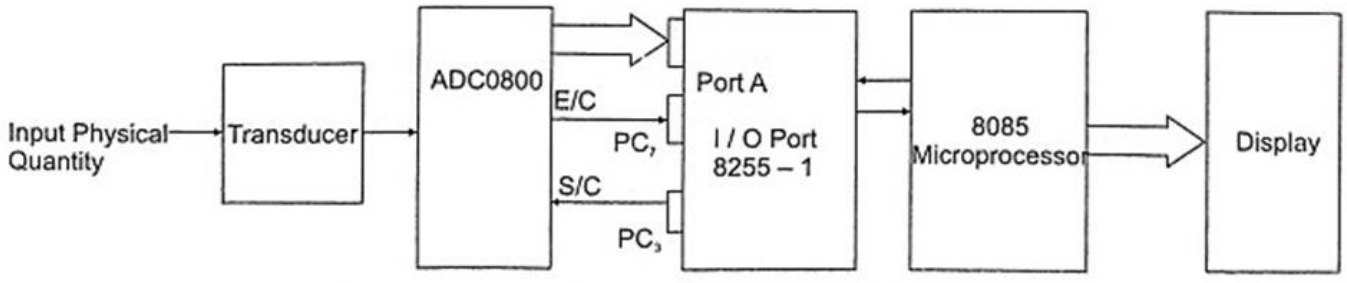
Measurement of Physical Quantities:

Measurement of Physical Quantities is given by

- Displacement Measurement
- Strain Measurement
- Force Measurement
- Torque Measurement
- Pressure Measurement
- Temperature Measurement
- Water-Level Indicator
- Measurement and Display of Speed of a Motor

Nowadays, physical quantities such as force, displacement, acceleration, velocity, speed, temperature, pressure, flow and level, etc., are measured and displayed using microprocessors and interfacing devices in industry. For the measurement of any physical quantity, transducers are used to convert energy from motion, displacement, acceleration, velocity, flow, pressure, level, heat, light, sound and any other Measurement of Physical Quantities into electrical energy. A transducer consists of sensor and signal conditioning circuit. Most commonly used transducers are potentiometers, capacitive and inductive transducers, level transducers, strain gauge, accelerometer, Linear Variable Differential Transformer (LVDT), piezoelectric crystals and diaphragm, etc. Electrical output of a transducer is very small and it is not in measurable condition; therefore it should be amplified by using amplifiers.

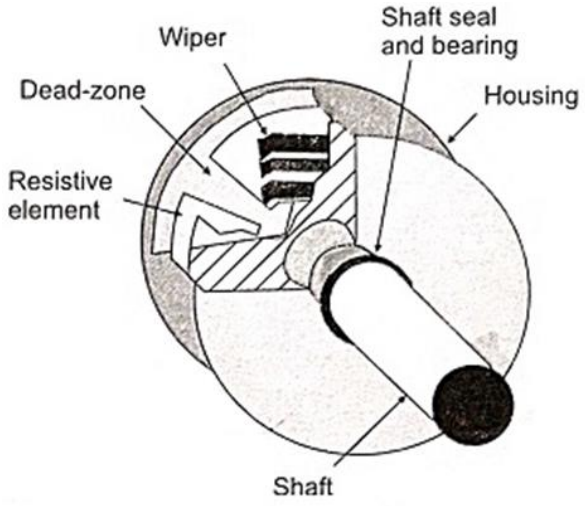
Block diagram of a Measurement of Physical Quantities



The output electrical signal from transducer is fed to an A/D converter, which converts analog signal to digital form and then applies to the 8085 microprocessor through 8255 PPI. The 8085 microprocessor reads this digital data and displays it in seven-segment display. When it is required to measure and display more than one Measurement of Physical Quantities, a multiplexer should be incorporated in between transducers and the A/D converter. In section. the working principle of measuring displacement, strain, pressure, force, torque, speed and temperature are discussed in detail.

Displacement Measurement

In a displacement-measurement potentiometer, capacitive transducers and Linear Variable Differential Transformers (LVDT) are generally used. In a potentiometer, the object moves the tap on a variable resistance and output voltage is directly proportional to displacement. Pots are used as potentiometers, shown in Fig.



In a pot, an electrically conductive wiper slides against a fixed resistor element. To measure displacement, the potentiometer is typically wired in a voltage divider configuration as depicted in Fig. The output voltage is a function of the wiper's position and it is an analog voltage.

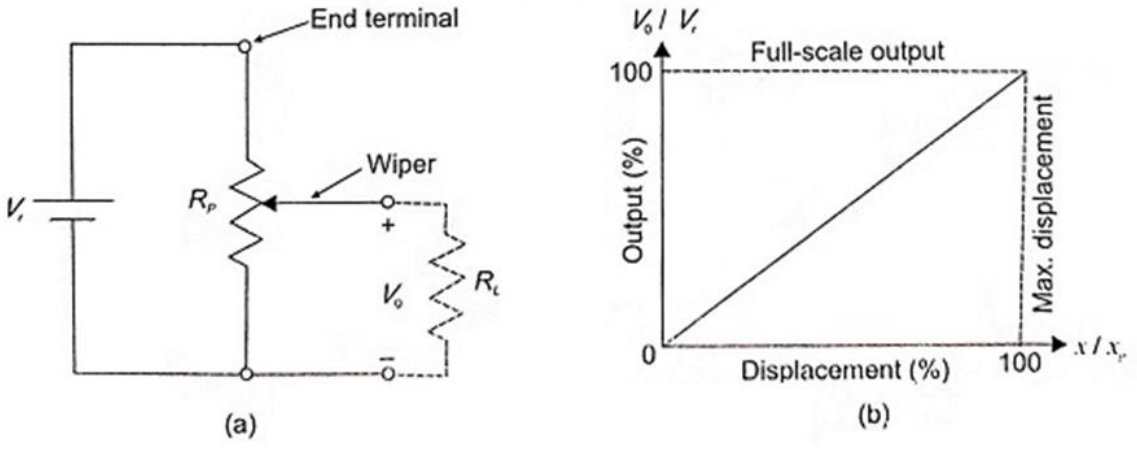


Fig.(a) Potentiometer (b) Linear Output Function

The output voltage V_0 can be expressed as

$$V_0 = V_r \frac{x}{x_r}$$

where

- V_r = the reference voltage,
- V_0 = output voltage,
- x_p = the maximum wiper position, and
- x = displacement.

This type of resistive displacement sensor has some advantages such as ease of use, low cost, high-amplitude voltage signal and passivity. But its disadvantages are limited bandwidth, frictional loading, inertial loading and wear. The potentiometer is commonly used in positioning of robotics like artificial limbs and servo systems.

In capacitive displacement transducer, one plate of the capacitor is mounted to a fixed surface and the other plate mounted to the object. With the position of object, capacitance value changes. The capacitive displacement sensor generates an output signal due to change in capacitance. The capacitance is a function of distance (d) between the electrodes, the surface area (A) of the electrodes, and the permittivity ε as given below:

$$C = \varepsilon \frac{A}{d} = \varepsilon_0 \varepsilon_r \frac{A}{d}$$

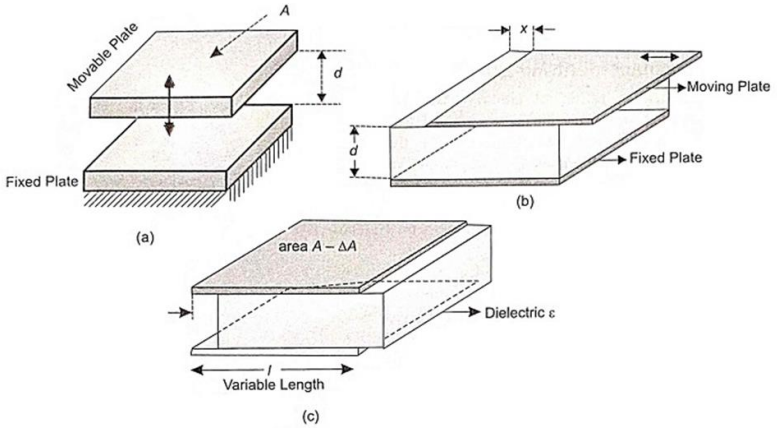
where

- ε_0 is permittivity of air, and
- ε_r is the relative permittivity.

The change in capacitance due to change in distance is

$$\Delta C = \varepsilon_0 \varepsilon_r \left(\frac{A}{d} - \frac{A}{d + \Delta d} \right)$$

Capacitor sensors are variable-distance displacement sensors, variable-area displacement sensors, and variable-dielectric displacement sensors as depicted in Fig. (a), (b) and (c) respectively.



Capacitors Sensors

Capacitor value in variable-area displacement sensors is

$$C = \varepsilon_0 \varepsilon_r \frac{A - wx}{d}$$

where

- w width,
- w_x then reduction in the area due to movement of the plate.

In variable dielectric displacement capacitive sensors,

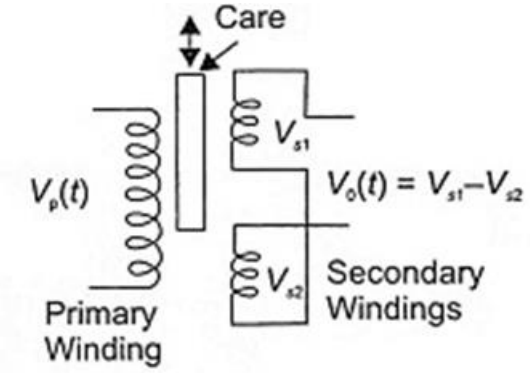
$$C = \varepsilon_0 \, w[\varepsilon_0 l - (\varepsilon_2 - \varepsilon_1)x]$$

where

- ε_2 is the permittivity of the displacing material, and
- ε_1 is the relative permittivity of the dielectric material.

Generally, a capacitive transducer can be placed in a bridge circuit and ac voltage is connected across the bridge. Then the bridge output voltage is amplified, rectified and measured.

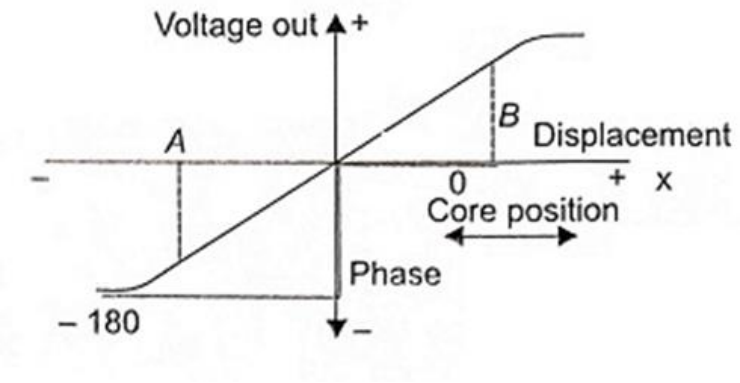
A Linear Variable Differential Transformer (LVDT) is a three-coil inductive transducer and object moves core of three winding. A three winding transformer (one primary and two secondary) with a movable core is shown in Fig. It is a passive inductive transducer. The two secondary's are having equal sizes, shapes and number of turns. Primary winding of transformer is supplied by 1-10 V, 50 Hz – 25 kHz ac signal. Each secondary winding covers one half of transformer, secondary's connected to oppose each other and the object is connected to the core.



LVDT

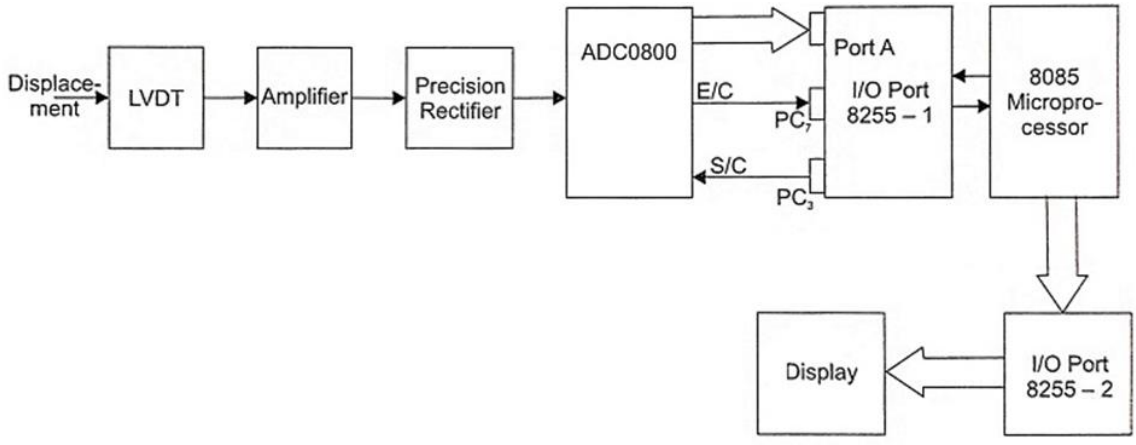
The mutual inductance between the primary and secondary windings is changed with the change in position of a high permeability rod. The induced voltages at secondary's are $V_{s1}(t) = K_1 V_p(t)$, $V_{s2}(t) = K_2 V_p(t)$. As secondary windings are connected in series opposition, the output voltage $V_0(t) = V_{s1}(t) - V_{s2}(t)$. As the rod moves from the Centre, K_1 increases while K_2 decreases.

When the core is centered, voltages at two secondary's are equal and the output voltage is zero. While core is off center, voltage at one secondary is higher than other one. Thus output voltage is linearly related to core position as depicted in Fig.



Plot of output voltage against core position

The model function of LVDT is $V_0(x, t) = KV_p(t)$ where K is a constant, t is time, $V_p(t)$ is the primary voltage, x is displacement and it will be either positive or negative. If x is negative, phase of output voltage is reversed.



The schematic block diagram of the displacement or deflection measurement is shown in Fig. LVDT is used to sense the deflection of a beam as the movable core of the linear variable differential transformer is connected to the beam. When the core is in centered position, the voltages induced in two secondary's of the LVDT are equal. Hence output voltage will be zero. While the core is moved in upward or downward directions, the voltage induced in two secondary's will not be equal and output voltage is equal to the difference between secondary induced voltages as expressed by $V_0 = V_{s1} - V_{s2}$. This output voltage V_0 is directly proportional to the displacement of core. The output voltage of LVDT is low and in the range of 100-500 mV. Therefore, an amplifier should be used to amplify LVDT output and fed to a precision rectifier for rectification. Then precision rectifier output voltage is applied

to A/D converter for analog to digital conversion. The digital output of the A/D converter is fed to Port A of 8255-1 as depicted.

Strain Measurement

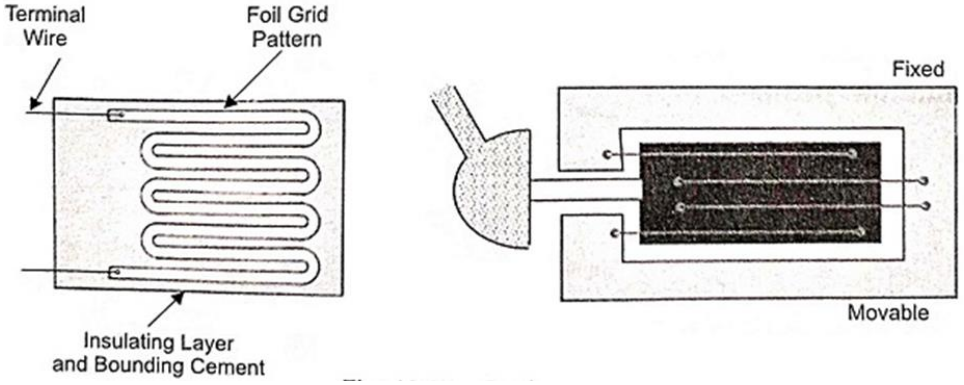
Strain is the change in shape of an object due to some force. Assume an object in two conditions: with and without a force applied. When an external force is applied along a dimension, there will be some deformation in the object.

Let L_1 be the length of the object along the dimension when no force is applied and L_2 be the length when the force is applied. Then the object's strain is

$$\frac{L_2 - L_1}{L_1} = \frac{\Delta L}{L_1}$$

Where $\Delta L = L_2 - L_1$ change in length.

A stain transducer is used for strain measurement. Strain gauge is a stain transducer and is used to measure strains and stresses in any structures. A strain gauge is a flexible card with strip of some copper–nickel alloy conductor wires arranged in special pattern as shown in Fig. 10.36. The grids of fine wires forming a strain gauge are cemented to a thin paper membrane. The strain gauge is mounted on the object being measured. A strain-gauge conductor is usually made of metal or semiconductor. The pattern is chosen in such a way that the conductor maintains an almost constant volume with strain. That is, the conductor is not compressible.



Strain Gauge

The resistance of a conductor is

$$R = \rho \frac{L}{A}$$

where

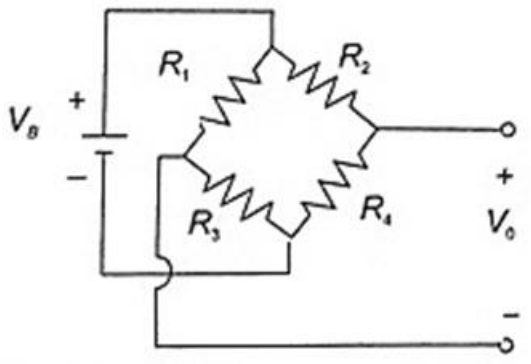
L is its length, A is its area, and ρ is its resistivity.

Assume force causes length of the conductor to decrease. Since volume does not change much, area must increase. Thus, resistance decreases.

The model function of resistance is equal to $R_1 = R_0 (1 + G_f\varepsilon)$ where G_f is a constant and known as the gauge factor which is the ratio $\Delta R/R / \Delta L/L$ and may he considered as the sensitivity of the sensor. R_0 is resistance without strain, R_1 is resistance due to strain, and ε is strain.

For metal-wire strain gauges (constantan), $G_f = 2$ while semiconductor strain gauges have much higher G_f of about 200. Bonded strain gauges have folded wires bonded to a semiflexible backing material, with unbonded gauges having flexible wires connected between fixed and movable frames as shown in Fig.

The sensitivity of a strain gauge is very low which is approximately 1% over full operating range. But it is very important that the above change must be accurately detected.



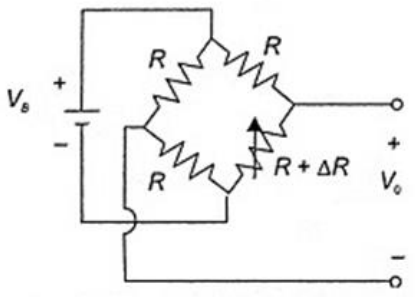
Wheatstone Bridge

To increase the sensitivity, two and more active sensor elements can be used in a bridge circuit. In strain measurement, a Wheatstone bridge is used a device. which can read a difference voltage directly. The output voltage of the Wheatstone bridge as shown in Fig. is expressed as

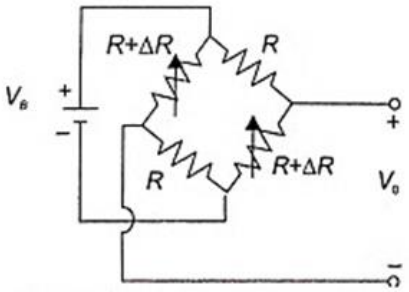
$$V_0 = \left(\frac{R_4}{R_2 + R_4} - \frac{R_3}{R_1 + R_3}\right) V_B$$

The sensor bridge usually consists of four identical sensor elements. Assume that only one of these is sensitive to the strain, which can be measured, and other sensors are 'dummy' sensors. For example, for a maximum of 1% change in resistance, the output voltage is about $0.0025V_B$. This is approximately 2.5 mV for a 10 V supply. Therefore, the range of V_0 in this case is 0-2.5 mV.

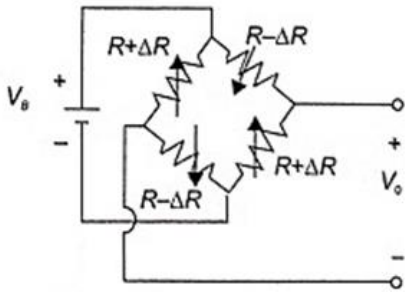
$$V_0 = \left(\frac{R+\Delta R}{2R+\Delta R} - \frac{R}{2R}\right) V_B \quad = \frac{\Delta R}{2(2R+\Delta R)} V_B \cong \frac{\Delta R}{4R} V_B$$



One strain Gauge in Wheatstone Bridge



Two strain Gauge in Wheatstone Bridge

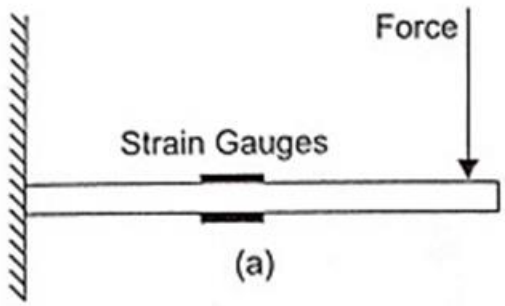


Four strain Gauge in Wheatstone Bridge

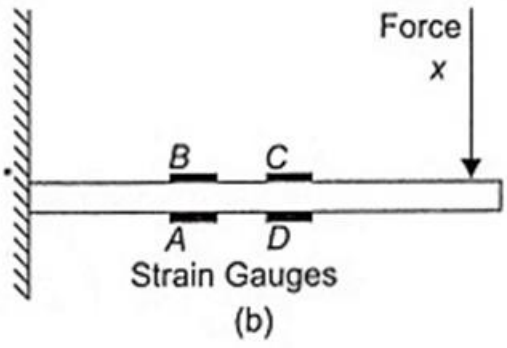
with only one of them subjected to the strain, the temperature effect is cancelled. The output voltage can be expressed

- for two Strain gauges in Wheatstone bridge, and
- for four Strain gauges in Wheatstone

$$V_0 = \left(\frac{R+\Delta R}{2R} - \frac{R-\Delta R}{2R}\right) V_B = \frac{\Delta R}{R} V_B$$



Complementary Pairs of strain Gauge



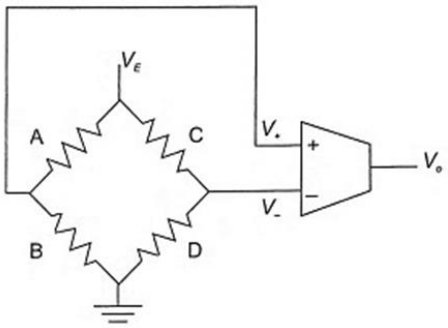
Four strain Gauges

In some cases, the strain in two places on the object will be of equal magnitude but of opposite sign. For example, a cantilever beam is as shown in Fig. The upper part of the beam is stretched (positive strain) and the lower part of the beam is compressed (negative strain). The two strain gauges, therefore, form complementary pairs.

Physically, a strain gauge is not much different from an RTD and so, a strain gauge is affected by temperature. Hence, temperature compensation is required. The model function including temperature.

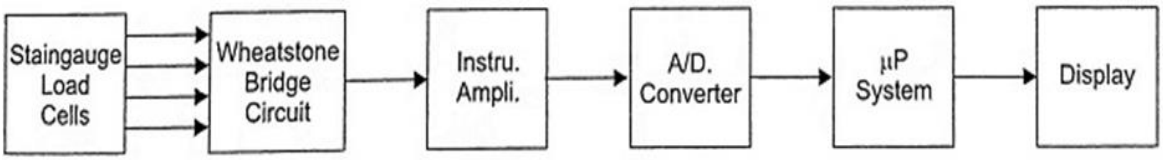
$R_1 = R_0(T)(1 + G_f\varepsilon)$, where $R_0(T)$ is a function of temperature. Conditioning circuit can remove $R_0(T)$. A bridge circuit does this very well.

For a complimentary pair, $R_1 = R_0(T)(1 \pm G_f\varepsilon)$



Bridge Circuit with amplifier

Four strain gauges are placed in the bridge form as shown in Fig. 10.42. The temperature terms will be canceled and the output voltage $V_0 = AV_E G_f\varepsilon$, where A = gain of amplifier, V_E = input voltage, G_f = gauge factor, and ε = strain.



Block diagram for Strain measurement

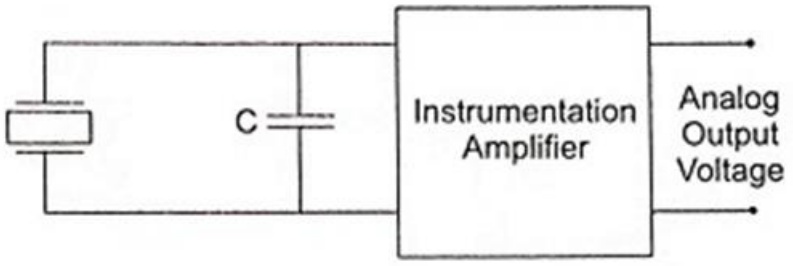
Figure shows the block diagram for strain measurement when two gauges are mounted to a cantilever beam. Due to change in resistance, bridge output voltage changes and its magnitude is directly proportional to strain. Then output of the bridge is fed to a instrumentation amplifier and amplified to a certain voltage in the range of 0-5 V so that it can be processed by the microprocessor. Output is 0 V for no strain and 5 V for the maximum strain. When four strain gauges are mounted in a cantilever beam, the output voltage increases two times. A look-up table between the hex code of digital voltage and strain is stored in memory. After converting the analog voltage into digital form through an A/D converter, find the strain from the look-up table and display it in seven-segment display unit.

Force Measurement

There are many types of sensors which can be used to measure force. Resistance-type force sensors, such as gauges and load cells, are very commonly used in force measurement. The force can be measured by the following ways:

Elastic sensing	$F = E.x$
Strain sensing	$F = \sigma.A$
Pressure sensing	$F = P.A$
Acceleration sensing	$F = m.a$

Force can move a part of the transducer. This movement can be measured using displacement sensors. Piezoelectric crystals are also used in force measurement A piezoelectric crystal consists of a crystal of a material with piezoelectric properties, i.e., a piezoelectric material emits charge when compressed. The material may be quartz, or special ceramics. Contacts are always placed along two faces of the crystal.



Piezo electric crystals in force measurement

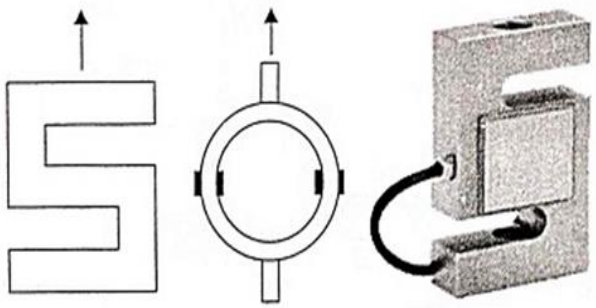
When force or pressure is applied to the crystal, a charge appears on the surface of the crystal and the amount of charge directly proportional to the force. Therefore, the output of the transducer is charge. The output of the crystal is converted to voltage using a capacitor. The voltage generated from piezoelectric crystals is V, which can be expressed as

$$V = \frac{Q}{C}$$

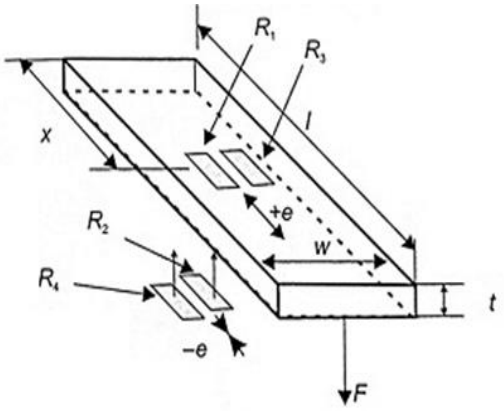
The capacitor should be chosen in such a way, so as to get the desired voltage range. The capacitor voltage is fed to very high input impedance amplifier for amplification, and amplifier output is applied to A/D converter for analog-to-digital conversion.

When an unchanging force is applied, the voltage will decrease over time. Therefore, piezoelectric crystals are best used for measuring changes in force and vibration.

A load cell is most commonly used to measure mechanical force. Strain gauges are called load cells and normally used for force measurement. The force bends, compresses, or stretches a part of the transducer and change in shape is usually measured using strain gauges. Two common load-cell configurations are illustrated in Fig. Usually. load cells are sold including a bridge circuit.

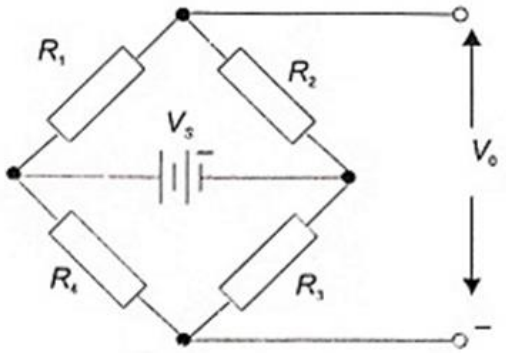


Load-cell Configurations



Cantilever beam with four strain Gauge

Resistance of a strain gauge increases if it is stretched. Strain gauges are cemented over the mechanical structure whose deformation under the influence of stress is to be measured. Figure 10.46 shows a cantilever beam with four strain gauges. The force is applied at a predetermined point. Strain gauges are placed at locations chosen so that their output is linearly related to force. The choice of location for the strain gauges and the derivation of the resulting load-cell model function are beyond the scope of this book. Load cells are usually packaged with strain gauges connected in bridge configuration. Strain gauges 1 and 2 are mounted so that after applying load, they come under tension. Similarly, strain gauges 3 and 4 will be under compression under loaded condition. Strain gauges are normally used in a full bridge to give the bridge output proportional to the applied force. To maximize the bridge sensitivity, the strain gauge is connected in a bridge. Under loaded conditions, resistance of strain gauges 1 and 2 increases and 3 and 4 decreases. Therefore, the potential at Point A of the bridge will be elevated much as compared to. As all four gauges are at the same temperature, this system also provides temperature compensation. Strain gauges are bonded in such a way that can provide the maximum output deformation ratio. The strain gauges are wired in full-bridge configuration for temperature compensation and for better accuracy of measurement. The complete assembly must be housed within a protective case and properly sealed so that the external environment cannot affect strain gauges though strain gauges are capable to deform after application of the force.



Bridge Configurations

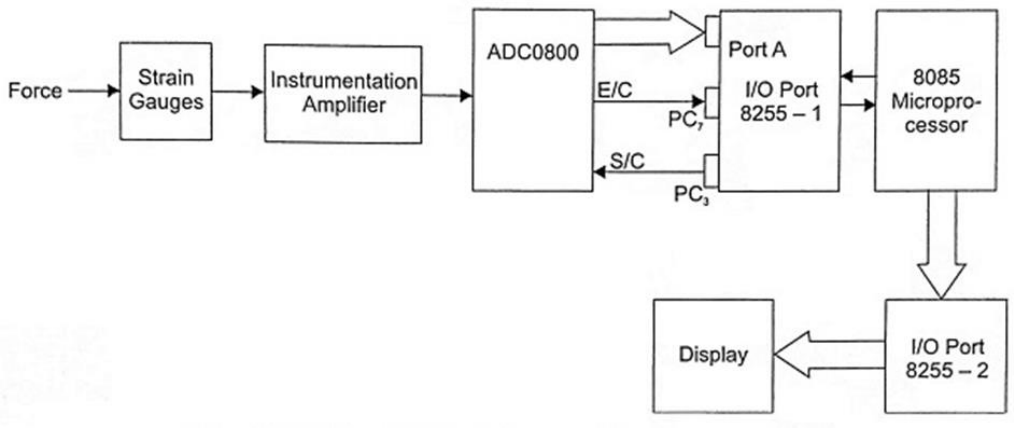
When strain gauges are used in cantilever type load cells, the strain can he expressed as

$$e = \frac{6(l-x)}{wt^2E} F$$

where F = force, l = length, w = width, t = thickness, and E Young's modulus.
The output voltage of the bridge will be

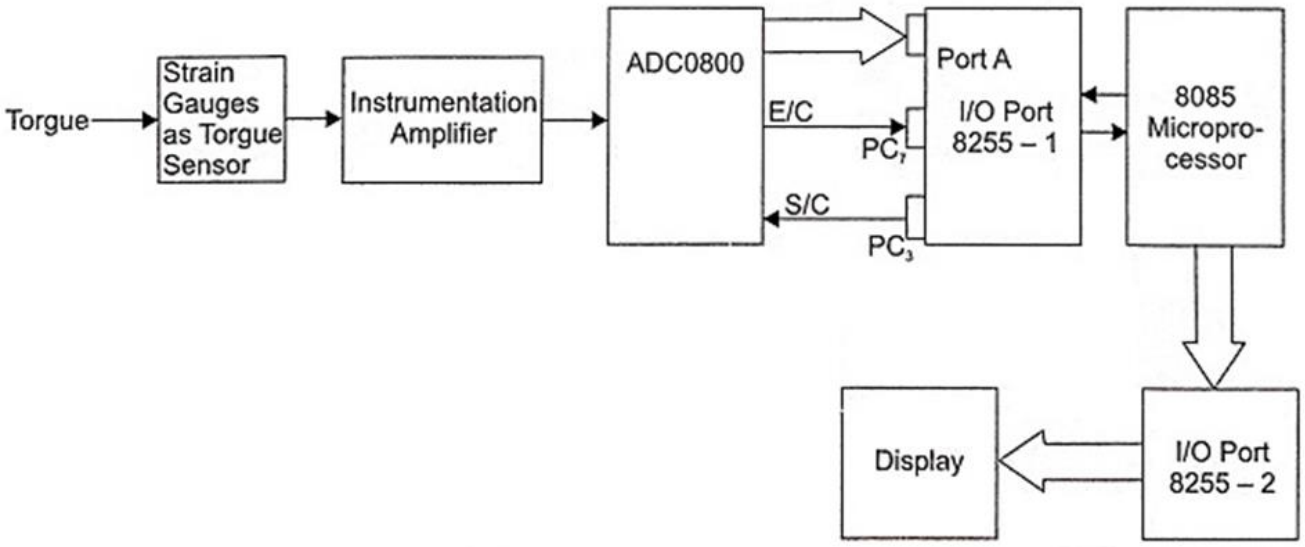
$$V_0 = \frac{V_s R_2 R_3}{(R_2 + R_3)^2}\left[\frac{\Delta R_3}{R_3} + \frac{\Delta R_1}{R_1} - \frac{\Delta R_2}{R_2} - \frac{\Delta R_4}{R_4}\right]$$

The block diagram for force measurement is shown in Fig. The program for displacement measurement can he used in force measurement, but there will be sonic modification in the look-up table. A look-up table between the hex code of digital voltage corresponding to force is stored in memory. The 8085 microprocessor reads the digital output of A/D converter for a force input and determines the force from the look-up table and display it in seven-segment display unit.



Block diagram of Force Measurement

Torque Measurement



Torque Measurement using strain Gauges

Generally, torque is transmitted through a rotating shaft between a power source and a power sink. Strain gauges are commonly used in torque cells. Figures 10.49 (a) and (b) show the torque measurement using strain gauges. Here, four strain gauges are mounted on the shaft. Strain gauges 1 and 3 are compressed, but strain gauges 2 and 4 are under tension due a torque in the shaft. The strain of the strain gauge 1 is approximately

$$e_1 = \frac{T}{\pi . G . r^3}$$

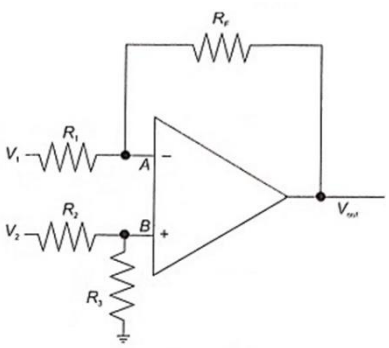
where

- T = torque,
- G = E/2(1+υ) = shear modulus,
- r = radius of shaft.

The relationship between strains of all four gauges is $e_2 = e_4 = -e_1 = -e_3$. When all four gauges are connected in bridge form as shown in Fig. 10.50, the output voltage can be expressed as

$$\frac{V_0}{V} = G_f . e_1 = \frac{G_f . T}{\pi . G . r^3}$$

The output of bridge circuit fed to a differential amplifier using an operational amplifier as shown in Fig. and output voltage becomes measurable. The microprocessor can be used to measure this voltage and display it in seven-segment display after proper calibration based on a look-up table.



Bridge Configuration

$R_1 = R_2$

$R_3 = R_F$

Gain: $R_F = R_1$

Temperature Measurement

Temperature is widely measured and controlled in industrial process control system. For temperature measurement, one of the following devices are used:

Device Name	*Temperature range*
Resistance thermometers	–100°C to + 300°C
Platinum Resistance thermometers	–0°C to 700°C
Thermocouples	–250 °C to + 2000°C
Thermistors	–100°C to + 100°C)
Pyrometers	+ 100°C to + 5000°C

Platinum wires are frequently used in resistance thermometers for industrial application because of their greater resolution, and mechanical and electrical stability as compared to copper or nickel wires. A change in temperature causes a change in resistance. The resistance thermometer is placed in an arm of a Wheatstone bridge to get a voltage proportional to temperature. A thermistor is a

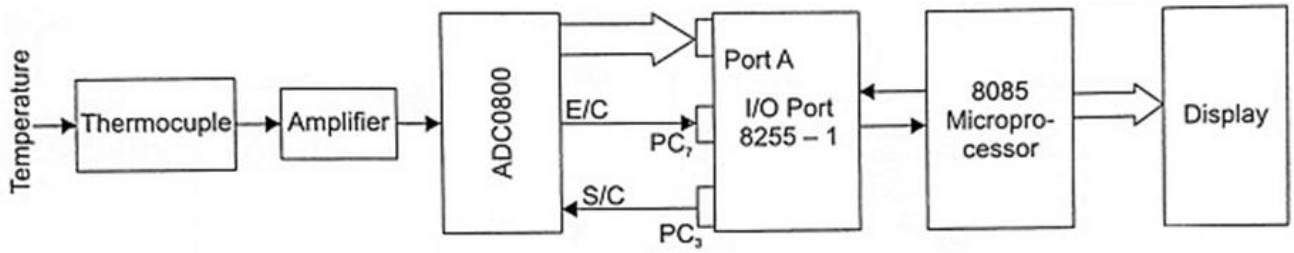
semiconductor device fabricated from a sintered mixture of metal alloys, having a large negative temperature coefficient. A thermistor is used in a Wheatstone bridge to get a voltage proportional to temperature. The thermistor is a thermally sensitive variable resistor made of semiconductor material. The substance used may be oxides of nickel, copper, manganese, iron, cobalt, etc., usually a high negative temperature coefficient. It can be used in the range of -100 to +100° C for greater accuracy as compared to a platinum resistance thermometer. Positive thermistors are also used but in the low range of 50°C to + 100°C.

In industry, the most widely used temperature transducer is the thermocouple. This temperature transducer works on the principle that contact potential between two dissimilar metals changes with temperature. When two dissimilar metals are joined and the junctions are placed at two different temperatures, an emf is induced which will he used for temperature measurement. Thermocouple materials for different ranges of temperature are given below:

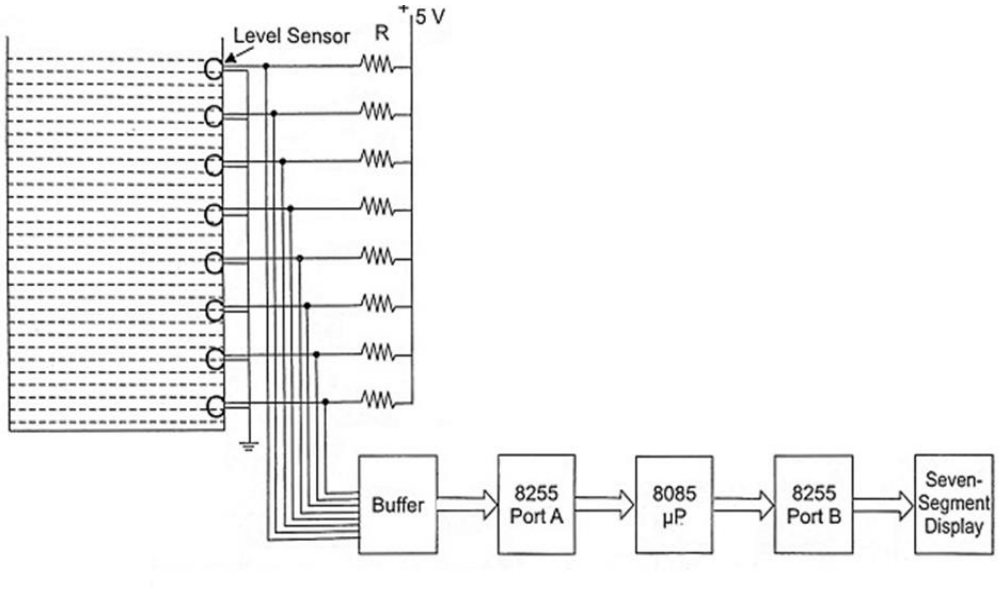
Material	*Temperature Range*
Copper–Constantan	–200°C to + 350°C
Iron–Constantan	–200°C to + 100°C
Iron–Nickel	+300°C to + 600°C
Nickel Chrome	+600°C to +1000°C
Platinum–Platinum Rhodium	+1000°C to +1750°C
Tungesten–Rhenium	0°C to + 2000°C

The microprocessor-based temperature measurement of an electrical furnace is shown in Fig. 10.54. Here, a thermocouple⁻ is used as a sensor for temperature measurement. The output of a thermocouple is directly proportional to the furnace temperature, which is in millivolt range. As output voltage is not in a measurable condition, it must be amplified using an instrumentation amplifier. 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 an A/D converter completes conversion, it sends an end-of-conversional signal to the microprocessor. Having received an end-of-conversion signal from the 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. Then the microprocessor displays the measured temperature.

Water-Level Indicator



A water-level indicator works by converting water levels into electrical signals and measures them by electrical or electronics circuits. The most simplest type water level indicator is the resistive method. This is also known as contact point type. A number of resistances of suitable values have their one end inserted in the column. Resistance may be a function of level.

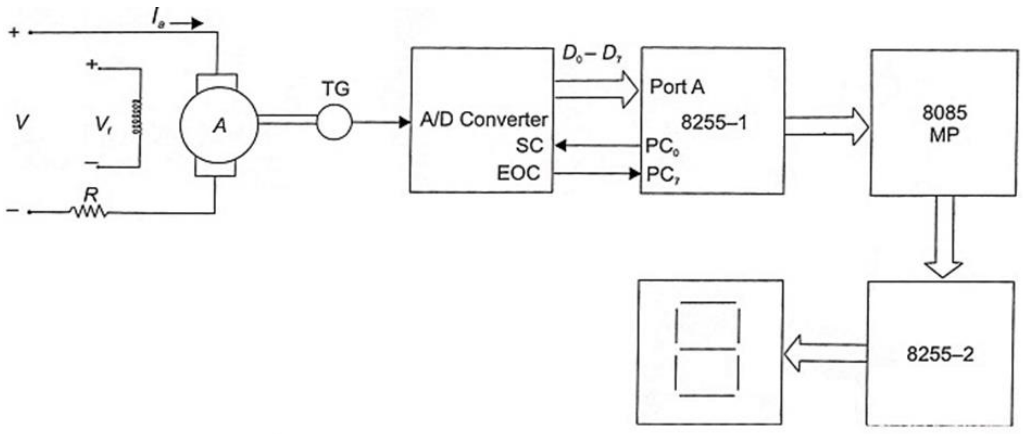


Block diagram for Water level Measurement

The microprocessor-based water level indicator is shown in Fig. The contact-type level sensors are connected to + 5 V through series resistors and other terminals arc grounded and assume water tank is also at ground potential. When the level sensor is immersed in water, it will be at ground potential and its output will be logic 0. If the level sensor is not immersed in water, its output is + 5 V or logic 1. As shown in Fig. there are eight level sensors, which are used to indicate eight different levels of the tank. The output of level sensors are connected to a buffer and buffer outputs are applied to Port A of 8255. The microprocessor reads the buffer output through Port A of 8255 and determines the water level based on look-up table, which is stored in the memory. After finding out the level, it will be displayed in a seven-segment display.

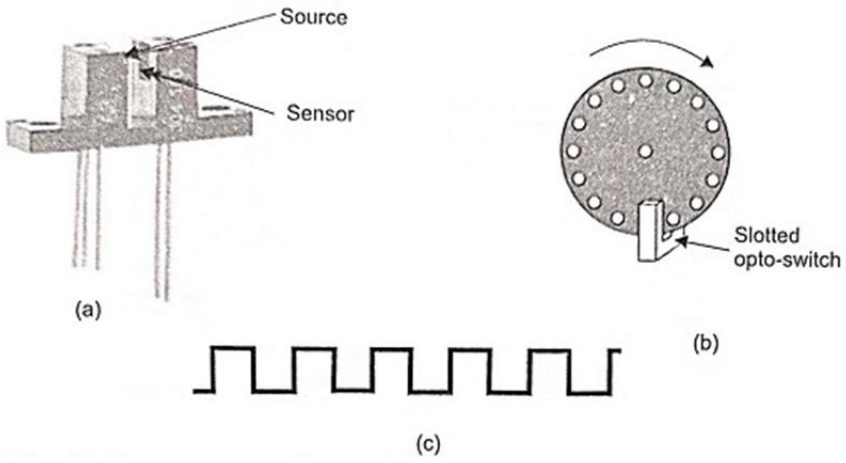
Measurement and Display of Speed of a Motor

Figure 10.58 shows the microprocessor-based speed measurement. A tacho-generator is coupled at the shaft of the motor and generates a voltage proportional to the speed. The output of the tachogenerator (TG) used in this measurement scheme is 0 to 5 volts dc for speed variation of 0 to 1500 RPM. The output voltage is connected to ADC 0808 for analog-to-digital conversion. The output of the A/D converter is applied to Port A of 8255-1. Seven-segment display units are connected to Port A and Port B of 8255-2. The control word for both 8255-1 is 98H and 8255-2 is 80H. The look-up table consists of the hex code of techo-generator voltage and its corresponding speed in rpm. The microprocessor reads the digital output of the A/D converter for different speeds of the motor. After that, the microprocessor measures the speed using a look-up table and displays the speed of the motor in a seven-segment display.



Block diagram for Speed Measurement

For better accuracy of measurement, the search-table properly calibrated. For this, each digital input voltage, and corresponding speed is measured accurately and stored in the memory location. For one digital input voltage, two memory locations are located in the search table where decimal data corresponding to the speed are stored. After getting a digital input corresponding to a speed, the speed will be searched from memory and displayed in the displayed screen. If we want to measure speed accurately, a train of pulses can be generated using a photoelectric switch sensor. Opto-switches consist of a light source and a light sensor within a single unit as shown in Fig. 10.59. This scheme of speed measurement has a light source, a semiconductor device sensitive to light and an attachment disc on the shaft containing a hole to pass light. The microprocessor will count the number of pulses per second, which is directly proportional to the speed.



(a) Slotted opto-switch (b) Opto switch sensor (c) Train of output Pulse

III – 8051 MICROCONTROLLERHARDWARE

Introduction to 8051 Microcontroller

Microcontroller is a single chip microcomputer which consists of CPU, Memory, I/O ports,timers and other peripherals. The difference between microprocessor and microcontroller is microprocessor is a single integrated CPU whereas microcontroller is single chip microcomputer.The world leaders of manufacturing of microprocessor and microcontroller are Intel, Motorola, IBM, Cyrix etc. Here we have to focus on microcontroller 8051.

In 1981 Intel Corporation introduced an 8-bit microcontroller called 8051.this microcontroller had 128 bytes of RAM, 4K bytes of on-chip ROM, two timers, one serial port and four ports (each 8bit wide) all on a single chip. It is an 8-bit processor means it can process8 bit of data at a time. It has total of four I/O ports, each 8-bit wide.

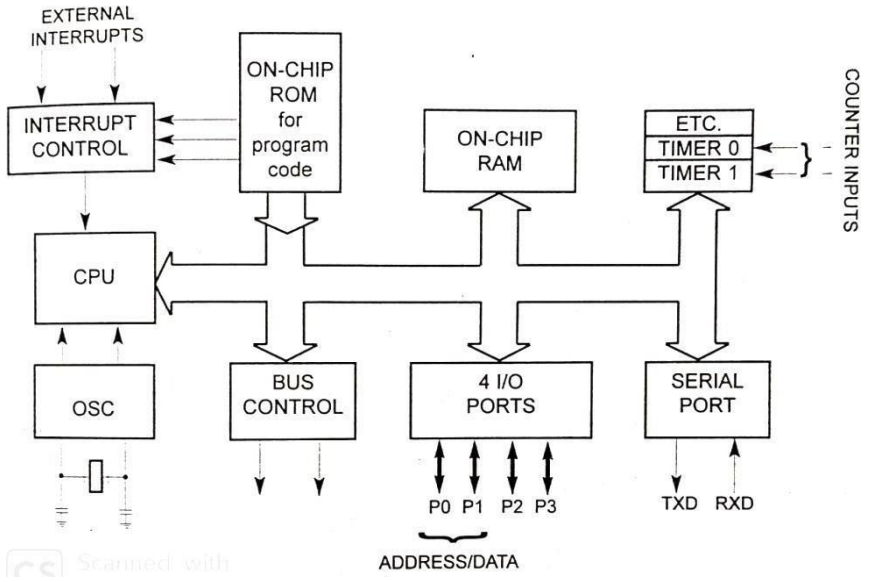
Features of 8051

Feature	Quantity
ROM	4K bytes
RAM	128bytes
Timer	2
I/O pins	32
Serial Port	1
Interrupt sources	6

Architecture of 8051

Fig shows a simplified architecture for the internal Hardware. It shows an overview ofthe internal hardware architecture of the 8051/8031 microcontrollers.

The CPU has the controlled and sequencing logic circuits with signals as in a microprocessor. The MCU has, besides the CPU, ROM, Interrupt control circuit, internal timing devices (timersT0, T1), serial interface (SI), RAM and special function registers (SFRs). It has four ports P0, P1, P2 and P3 as shown in Fig.



Simplified architecture of 8051

Description of Sub units in the hardware architecture and meaning of the symbol

PC- Program Counter -A 16 bit register to hold the program memory address of the instruction being currently fetched. Increments continuously to point to the next instruction, unless there is change in the program flow path.

DPTR- Data Pointer register

A 16-bit register to hold the external data memory address of the data being currently fetched orto be fetched in indirect addressing mode.

A-Accumulator

An 8-bit register to save an operand for an ALU or data transfer operation and is also used toaccumulate result after an ALU operation.

B- B register

An 8-bit register to save a second operand for the ALU and also accumulate the result after ALU operation for multiplication or division.

ALU- Arithmetic logic unit

A unit to perform an arithmetic and logical operation at an instance as per the instruction to beexecuted and give result.

PSW- Processor Status Word

A register to save the bits of different flags.

P0- Port P0

An 8-bit port for the I/Os in a single chip mode and for the data bus-cum- lower order address inthe expanded mode.

P2- Port2

An 8-bit port for the I/Os in a single chip mode and for the higher order address in the expanded mode

P1- Port1

An 8-bit port for the I/Os in a single chip mode and a few device operations related bits in certain8051 family variants in the expanded mode.

P3- Port3

An 8-bit port for the I/Os in a single chip mode and the serial interface (SI) bits , timer T0 and T1 inputs, Interrupts INT0 and INT1 inputs , RD and WR for the memory read-write in the expanded mode.

SI- Serial Interface Device

Serial device for full duplex UART serial I/O operations through the set of two pins of P3, RxD and TxD and for the half duplex synchronous communication of the bits through the same set of pins, DATA and CLOCK.

T0 and T1- Timers T0 and T1

Timing devices in 8051 family using four registers TH1, TH0, TL1, and TL0.

SFRs- Special Function Registers

All registers the SP, PSW, A, B, IE, IP, SCON, TCON, SMOD, SBUF, PCON, , TL0, TH0,

TL1, TH1 are called SFRs

ROM- Read only Program memory

Masked ROM EPROM or flash EEPROM of 4kB in 8051 classic family.

Internal RAM- Internal Random Access Memory

For read and write the 128 B memory is indirectly and directly addressable in address space.

Register banks- Four set of registers

Four register banks each of 8 registers and these are also part of the internal RAM.

XTAL1 and XTAL2 – Pins to the Crystal

Pins to the crystal in the oscillator circuit, usually 12 MHz

$\overline{EA}$ - External Enable

To enable use of external memory addresses to external ROM.

RST- Reset Pin

Reset circuit input and also reset few output cycles to the external peripheral devices to let processor reset and synchronize with devices.

INT **0 and** INT **1- Interrupt pins**

Active low two external interrupts.

VCC and GND- Voltage supply pi and ground pin

For 5 V supply and ground connections respectively.

PSEN - Program Store Enable

Active low when reading the external program memory bytes

$\overline{RD}$ **-Read**

Active low when reading the byte from external data memory. $\overline{WR}$ **- Write**

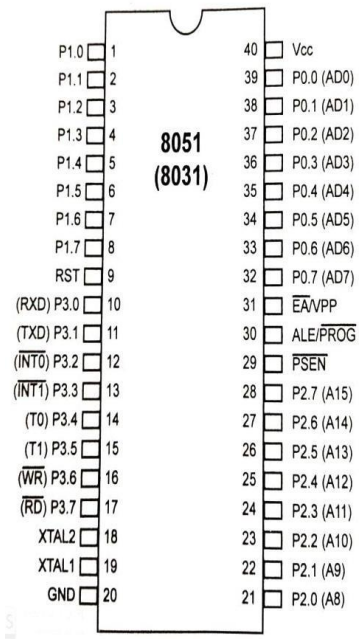
Active low when writing the byte to external data memory

Pin Configuration

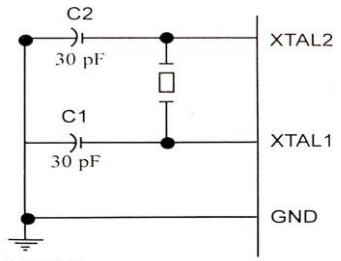
Fig. shows 40 pin signals in an 8051 series microcontroller. It shows the I/O pins, P0.0 to P0.7, P1.0 to P1.7, P2.0 to P2.7 and P3.0 to P3.7. It shows other remaining 8 pins, V_{DD}, V_{SS}, XTAL1 and XTAL2, RST, ALE, $\overline{EA}$ and PSEN . **Vcc - Pin** 40 provides supply voltage to the chip. The voltage source is +5V

GND- Pin 20 is the ground.

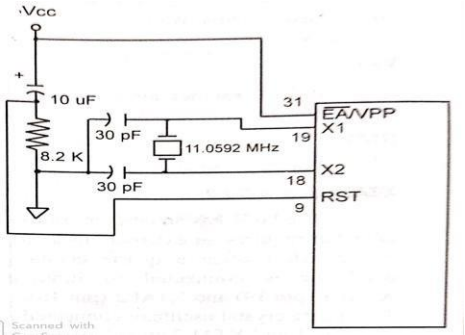
XTAL1 and XTAL2- 8051 has an on-chip oscillator but requires an external clock to run it. Most upon a quartz crystal oscillator is connected to inputs XTAL1 (pin 19 and XTAL@ (pin- 18) The quartz crystal oscillator connected also needs two capacitors of 30 pF. If frequency source other than crystal oscillator such as TTL oscillator will be connected to XTAL1 and XTAL2 is left unconnected.



8051 Pin diagram



TAL connection to8051



Power-On RESET

RST (I/P)- Pin 9 is the RESET pin and is active high (normally low). Upon applying high pulse to this pin the microcontroller will reset and terminate all activities. This often referred to as power on reset. Once it is activated the contents of all registers become zero except the contentof SP which is 07H.

EA **(External Access)** - This pin is connected to V_{CC} for those have on-chip ROM otherwise it is grounded incase 8031 and 8032. Because in case of 8031 and 8032 there is no on-chipROM.

PSEN **(o/p) (Program Store Enable)-** In case of 8031 based system in which an external ROM holds the program code . To read the code this pin is connected to OE pin of ROM chip.

AlE (o/p) (address Latch enable)- When 8051is connected to external memory, both address and data are transferred through port 0 pins. ALE signal is active high used to demultiplex address/data bus.

P0, P1, P2 and P3 are explained in port section.

Register banks in the 8051

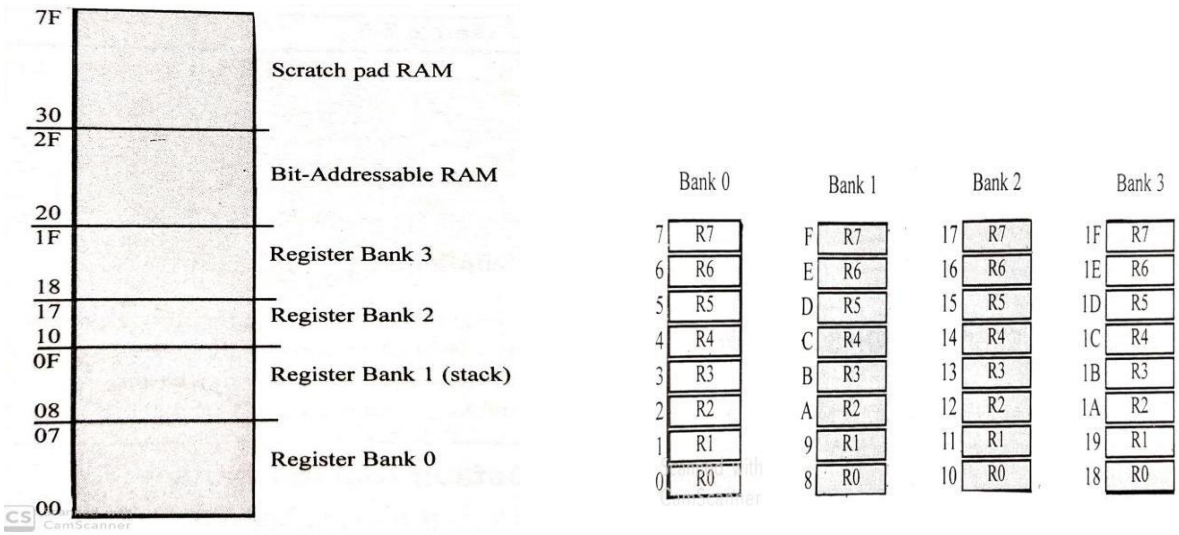
As mentioned, a total of 32 bytes of RAM are set aside for the register banks an stack. These 32 bytes are divided into 4 banks of registers in which each bank has 8 registers, R0-R7. RAM locations from 0 to 7 are set aside for bank 0 of R0-R7 where R0 is RAM location 0 , R2 is location 2 and so on. The second bank of registers R0-R7 start RAM location 08 and goes to location 1FH.
The third bank of R0-R7 starts at memory location 10 H and goes to location 17H. finally RAM location 18H to 1FH are set aside for the fourth bank of R0-R7. The following shows how 32 bytes are allocated into 4 banks.

Memory Organization

The 8051 micro controller has a total of 128 bytes of RAM. The 128 bytes of RAM inside the8051 are assigned addresses 00H to 7FH and divided into three different groups as follows.

1. A total of 32 bytes from location s 00H to 1FH are set aside for register banks and the stacks.
2. A total of 16 bytes from locations 20H to 2FH are set aside for bit addressable read/write memory.

3. A total of 80 bytes from locations 30H to 7FH are used for read and write storage, or what is normally called a scratch pad. These 80 locations of RAM are widely used for the purpose of storing data and parameters by 8051 programmers.



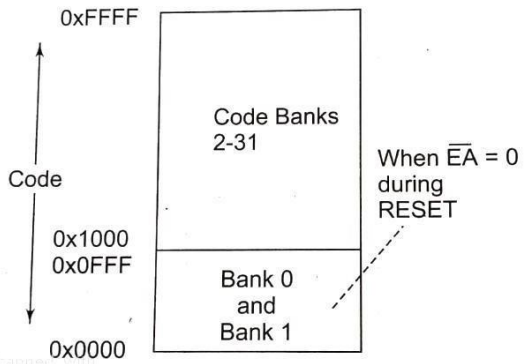
RAM allocation in the 8051

External Program Memory

Fig. shows a layout of the external code memory addresses in the classic 8051 architecture.

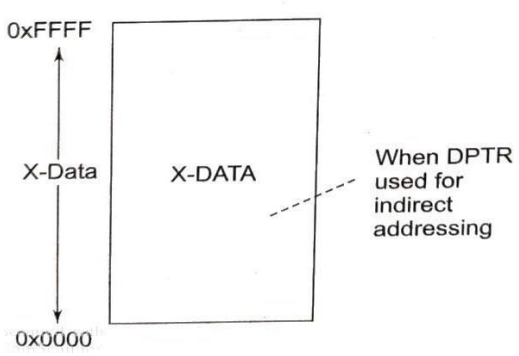
1. When the the EA =0 at RESET, the PC (MCU program counter) starts from 0x0000 and accesses the external addresses from the memory. Memory addresses are between 0x0000 and 0xFFFF.

2. When the EA =1 at RESET, the PC starts from 0x0000 for banks0 and 1 and accesses the internal addresses and the 0x1000 onwards from the external addresses from the memory.



External Data Memory

Fig. shows a layout of the external data (X-DATA) memory addresses in the classic 8051architecture. It can be accessed through the indirect addressing mode used.



Memory for X-Data in classic 8051

Special Function Registers (SFR)

For a programmer, the SFRs are at the directly addressable space special registers. These can be accessed by their names or by their addresses. The SFRs have addresses between 80H and FFH. These addresses are above 80H, since the addresses 00 to 7FH are addresses of RAM memory inside the 8051.Not all the address space of 80 to FF is used by the SFR. The unused locations 80H to FFH are reserved and must not be used by the 8051 programmer. The meaning of each symbol is enlisted in Table.

Special Function Register (SFR) Address.

Symbol	Name	Address
ACC*	Accumulator	0E0H
B*	B-register	0F0H
PSW*	Program Status Word	0D0H
SP	Stack Pointer	81H
DPTR	Data Pointer 2 bytes	
	DPL lower byte	82H
	DPH higher byte	83H
P0*	Port0	80H
P1*	Port1	90H
P2*	Port2	0A0H

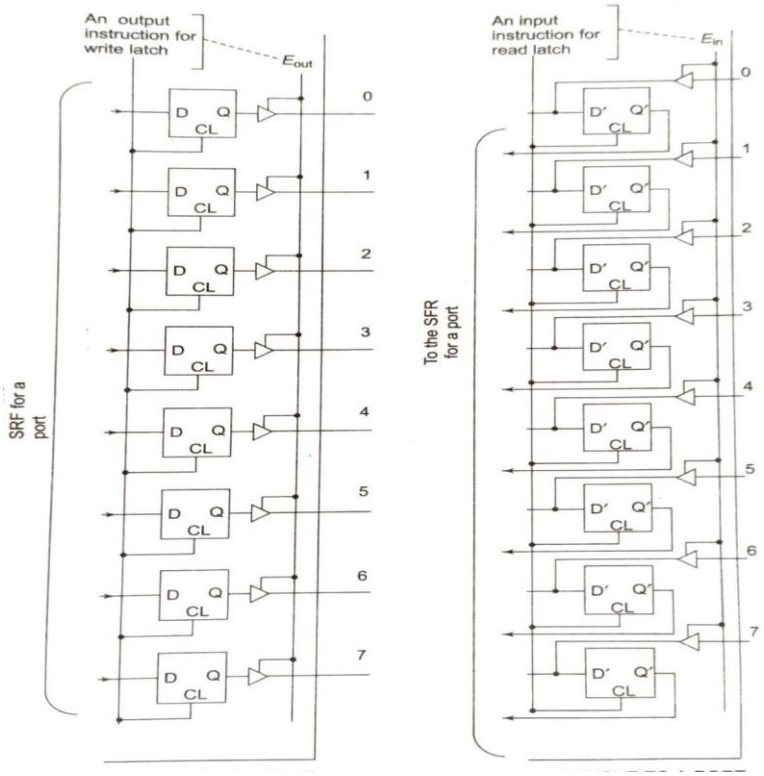
P3*	Port3	0B0H
IP*	Interrupt Priority Control	0B8H
IE*	Interrupt Enable Control	0A8H
TMOD	Timer /counter mode control	89H
TCON*	Timer/counter control	88H
T2CON*	Timer/counter 2 control	0C8H
T2MOD	Timer /counter mode control	0C9H
TH0	Timer/counter0 high byte	8CH
TL0	Timer/counter0 low byte	8AH
TH1	Timer/counter 1 high byte	8DH
TL1	Timer/counter 1 low byte	8BH
TH2	Timer/counter 2 high byte	0CDH
TL2	Timer/counter 2 low byte	0CCH
RCAP2H	T/C2 capture register high byte	0CBH
RCAP2L	T/C2 capture register high byte	0CAH
SCON*	Serial control	98H
SBUF	Serial data buffer	99H
PCON8	Power control	87H
* indicate Bit addressable		

Port Operation

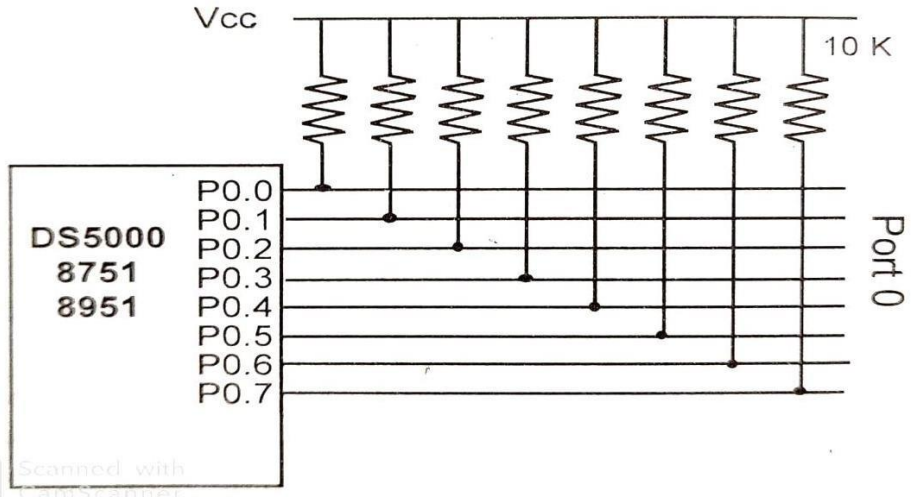
The four ports P0, P1, P2 and P3 each use 8 pins, making them 8-bit ports. All the ports upon RESET are configured as output, ready to be used as output ports. To use any of these ports as an input port , it must be programmed. The port structure is depicted in Fig.

Port 0

It can be used for input or output. It occupies total of 8 pins (pins 32-39). To use the pins of port 0 as both input and out ports, each pin must be connected externally to a 10 K ohm pull-up resistor. P0 is an open drain unlike P1, P2 and P3. With external pull-up resistors connected uponreset, port0 is configured as an output port.



Port Structure



Port 0 with pull up Resistors

With resistors connected to port 0 , in order to make it as input the port must be programmed by writing 1 to all the bits. In the following code.

```
MOV          A, #0FFHMOV    P0, A
BACK:        MOVA, P0
MOV          P1, A
SJMP         BACK.
```

Port 1

Port 1 occupies a total of 8 pins (pins 1 through 8) . It can be used as input or output. In contrastto Port 0 , this port does not need any pull-up resistors since it already has pull-up resistors internally. Upon reset port 1 is configured as an output port. To make Port 1 an input port it mustbe programmed as such by writing 1 to all its bits.

Port 2

Port 2 occupies a total of 8 pins (pins 21 through 28). It can be used as input or output. Just like P1, port 2 does not need any pull-up resistors since it already has pull-up resistors internally. Upon reset, port 2 is configured as an output port. To make port 2 as input, it must programmed as such by writing 1 to all its bits. The dual role of port 2 is also accomplished by providing higher byte address through A8-A15 to access the external memory.

Port 3

Port 3 occupies a total of 8 pins, pin 10 through 17. It can be used as input or output. P3 does notneed any pull-up resistors , the same as P1 and P2. Although Port 3 is configured as an output port upon reset, Port 3 has additional function of providing some extremely important signals such as interrupts. Table depicts the alternate functions of port 2.

Port 3 alternate functions

P3 bit	Functions	Pin
P3.0	RxD	10
P3.1	TxD	11
P3.2	$\overline{\text{INT0}}$	12
P3.3	$\overline{\text{INT1}}$	13
P3.4	T0	14
P3.5	T1	15
P3.6	$\overline{\text{WR}}$	16
P3.7	$\overline{\text{RD}}$	17

P3.0 and P3.1 are used for the RxD and TxD serial communication signals. P3.2 and P3.3 are used for external interrupts. Bits P3.4 and P3.5 are used for timers 0 and 1. Bits P3.6 and P3.7 are used to provide WR and RD signals for external memories in 8051 based system.

Memory interfacing

Semiconductor memory

In the design of all microprocessor based system, Semiconductor memory are used as primary storage for code and data. It can be in units of K bits, M bits and so on. Semiconductor memories are conducted directly to the CPU and is also called as primary memory. The widely used semiconductor memories are ROM and RAM.

Characteristics of Semiconductor Memory

Memory capacity- The number of bits that a semiconductor memory chip can store is called chip capacity.

Memory organization- Memory chips are organized into number of locations within the IC. Each location hold 1 bit, 4bits, 8bits or even 16 bits, depending on how it is designed internally.

1. A memory chip contains 2^x locations where x is the number of address pins.
2. Each location contains y bits, where y is the number of data pins on the chip.
3. The entire chip will contain 2^x x y bits, where x is the number of address pins and y isthe number of data pins

Speed- One of the most important characteristics of a memory chip is the speed at which its data can be accessed.

ROM (Read-Only-Memory)- It is a type of memory that does not loss its contents when the power is turned off. For this reason ROM is called volatile memory. There are different types of read-only-memory such as PROM, EPROM, EEPROM, Flash EPROM and mask ROM.

PROM- It refers to the kind of ROM that the user can burn information into it. That's why it is called as user-programmable memory. For every bit of the PROM, there exists a fuse. So it is programmed by blowing of fuses. It is also referred to as OTP (one –time programmable)

EPROM (Erasable Programmable ROM)- In EPROM, one can program the memory chip and erase it thousands of times. A widely used EPROM is called UV-EPROM. The content of UV-EPROM is erased when it is exposed to ultra violet light. Its erase time is near about 20 minutes.

EEPROM (Electrically Erasable Programmable ROM)- Its desired contents are erased by electrically.

Flash memory EPROM- This memory has become popular user-programmable memory chip, due to the process of erasure of the entire contents takes less than a second. As the erasure method is

electrical sometimes it is called as Flash EEPROM.

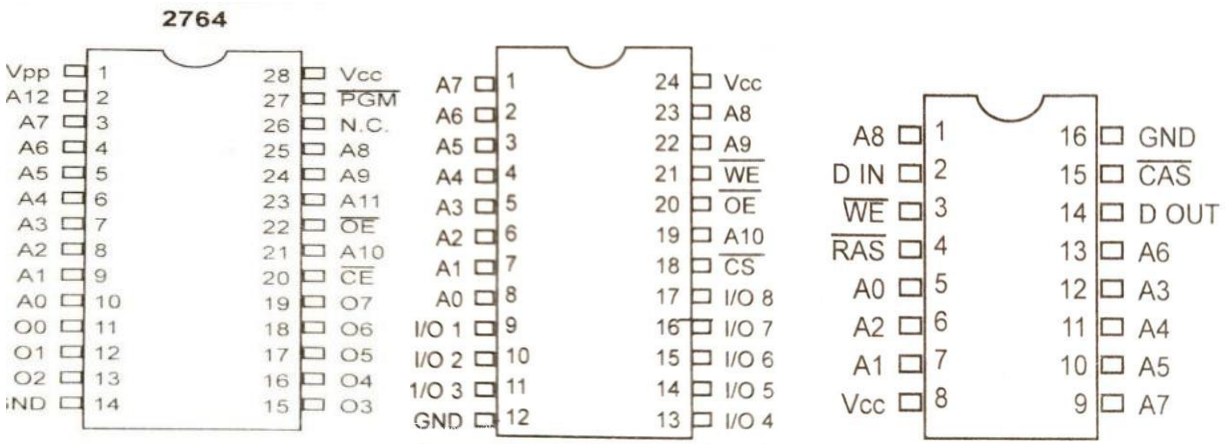
Mask ROM- Mask ROM refers to kind of ROM in which the contents are programmed by theIC manufacturer. It is not a user-programmable ROM.

RAM (Random Access Memory)- It is called volatile memory since cutting of the power tothe IC will result in the loss of data. Sometimes it is called as read and write memory (RAWM). There are three types of RAM: Static RAM (SRAM), NV-RAM (Nonvolatile RAM) anddynamic RAM (DRAM).

NV-RAM (Nonvolatile RAM)-This RAM is nonvolatile. Like other RAMs it allows the CPU toread write to it, but when the power is turned off, the contents are not lost. To retain its content every NV-RAM chip internally is made of the following components.

1. It uses extremely power-efficient. SRAM cells built out of CMOS
2. It uses an internal lithium battery as back energy source.
3. It uses an intelligent control circuitry. The main job of internal circuitry to monitor the V_{cc} pin constantly to detect the loss of external power supply. If the power to the Vcc pin falls the below out-of-tolerance condition, the control circuitry switches automaticallyto its internal power source, lithium battery.

DRAM (Dynamic RAM)-The use of a capacitor as a means to store data cuts down the number of transistors needed to build up the cell; however it requires constant refreshing due to leakage.This is in contrast to SRAM whose cells are made of flip-lops. The use of capacitor as storage cells in DRAM results in much smaller net memory size.



2764 ROM 8Kx8 2Kx8 SRAM pins 256Kx1 DRAM

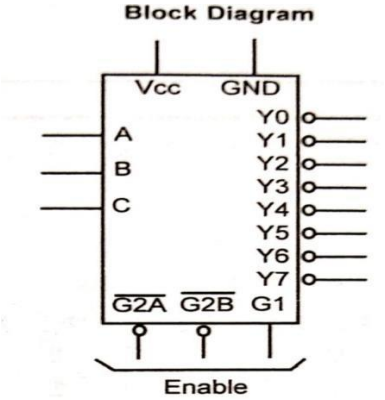
Memory Address Decoding

The job of the decoding circuitry to locate the selected memory block that CPU has access to desired data in memory chip. Memory chips have one more pin called CS (chip select) which must be

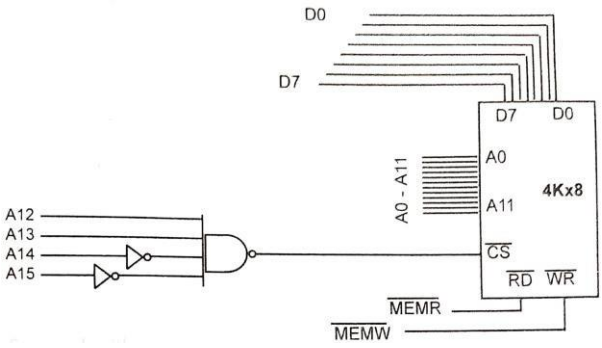
activated for the memory contents to be accessed. Sometimes the chip select is also referred to as Chip Enable (CE).

Following points are required for interfacing the memory to the CPU.

- The data bus of the CPU is connected directly to data pins of the memory chip.
- Control signals RD (read) and WR write from the CPU are connected to the OE (output enable) and WE (write enable) pins of memory chips respectively.
- In case of the address buses, while lower bits of the addresses from the CPU areconnected directly to the address pins of the memory chips and upper address pins are used to activate the CS or CE pin of the memory chip. The CS or CE pin along with RD/WR allows the flow of data in or out of the memory chip.



74LS138 Decoder



logic Gate as Decoder

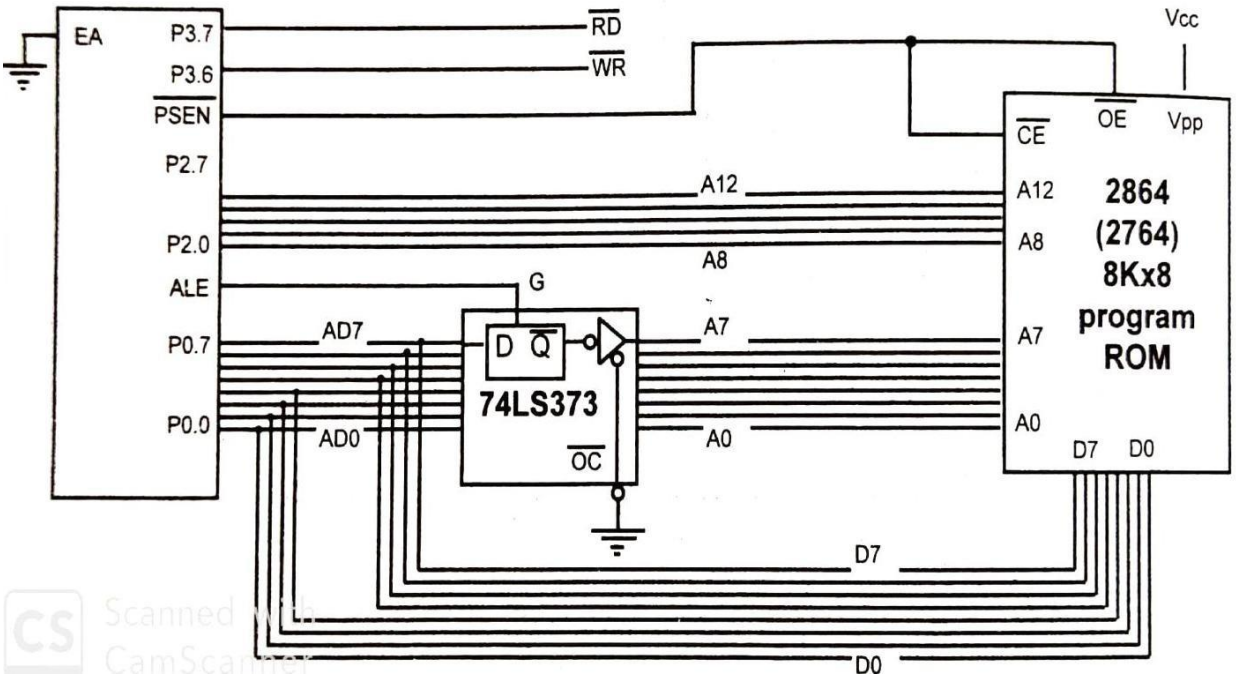
Interfacing with External ROM/RAM as Program and Data Memory

For interfacing to external ROM some pins have important role that to be discussed here.

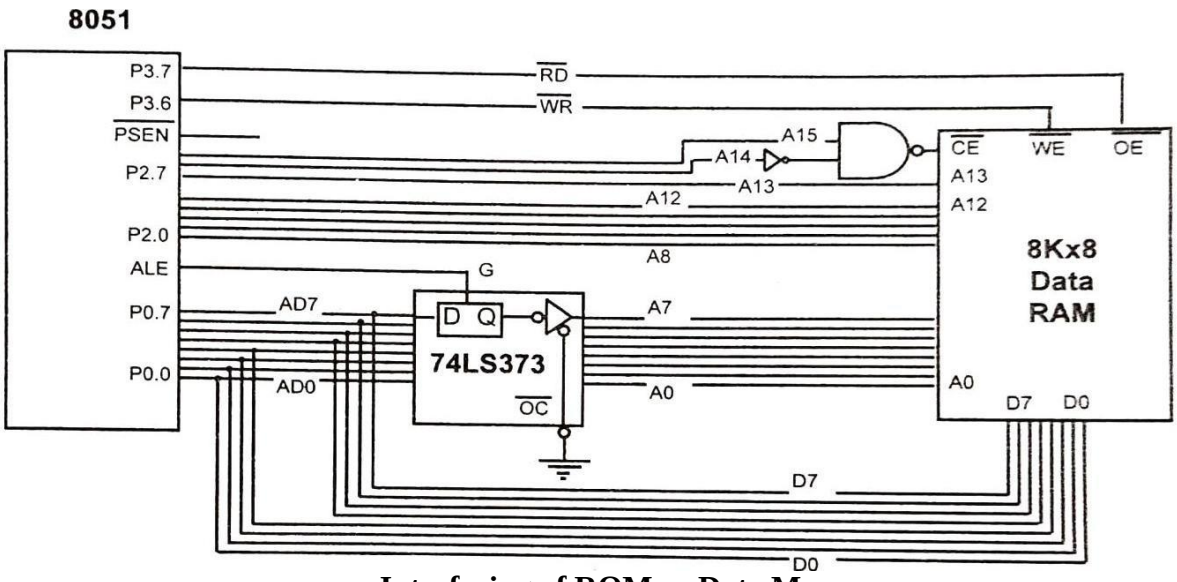
EA-When this pin is connected to Vcc, that indicates the program code is stored in the microcontroller on-chip ROM. For external ROM access tis pin is grounded.

P0 and P2 role in providing addresses- In 8051 P0 and P2 provides the 16-bit address to access external memory. Of these ports P0 provides the lower 8 bit addresses A0-A7, and P2 provides the upper 8 bit addresses A8-A15. More importantly, P0 is also used to provide 8 bit- data bus D0-D7. In other words P0.0- P0.7 are used for both address and Data is called as address/data multiplexing. The sharing of this bus is accomplished by ALE (address latch enable.) Pin. When ALE=0, the 8051 uses P0 for the data path and when ALE=1, it is used for address path.

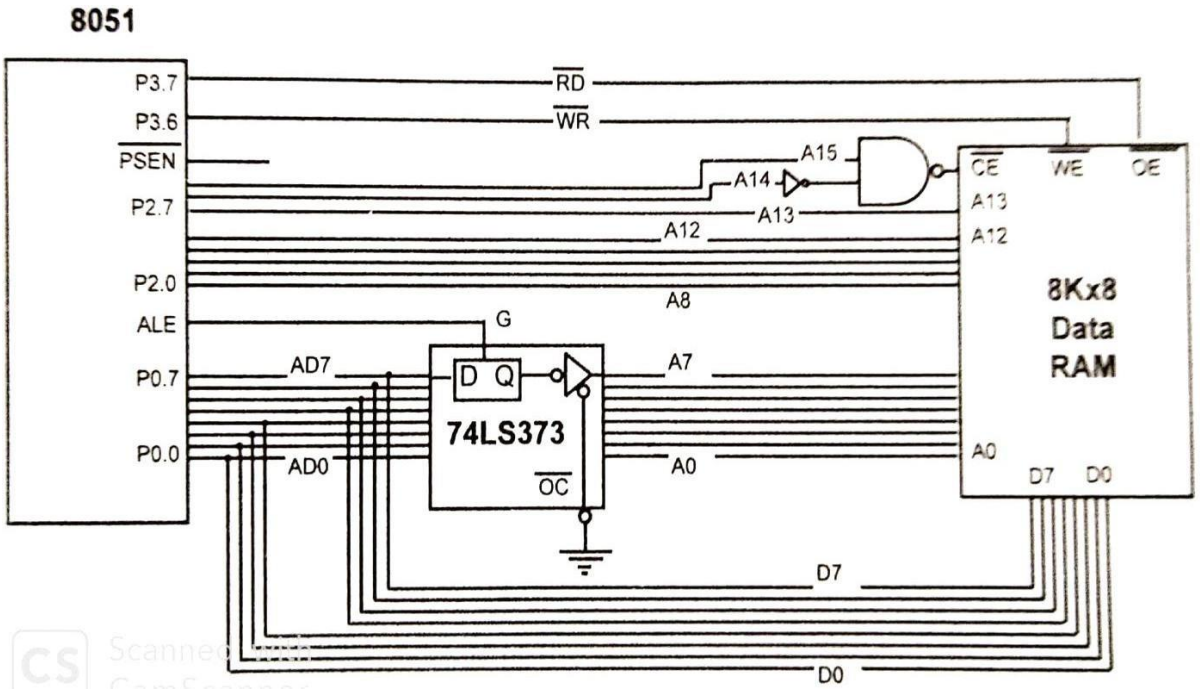
PSEN **(program store enable)-** It is an output signal must be connected to OE pin of a ROM containing the program code. When EA pin is connected to ground the 8051 fetches opcodefrom external ROM by using PSEN .



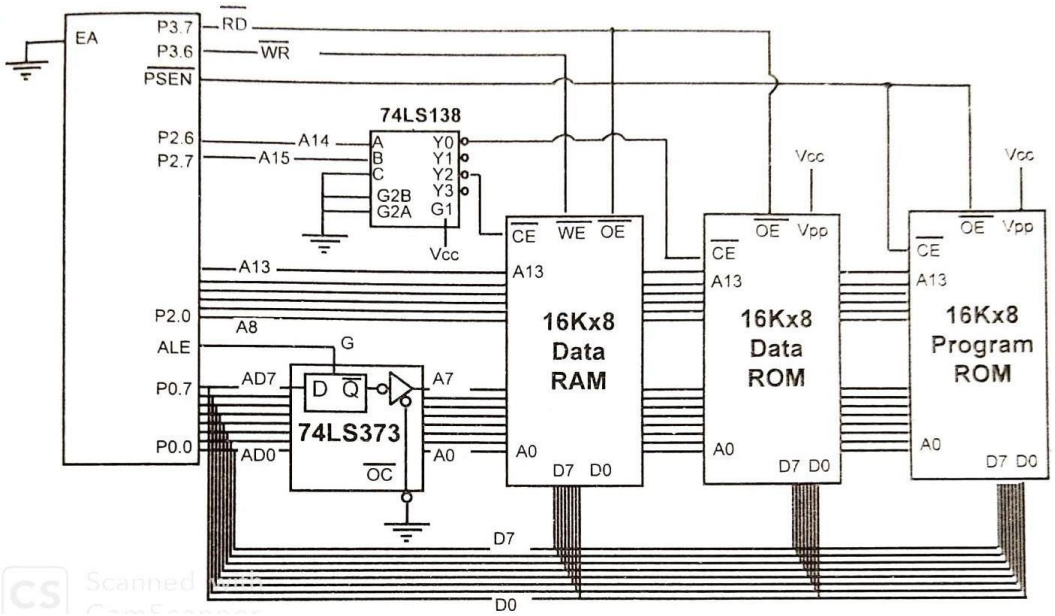
Interfacing of ROM to 8051 as program memory



Interfacing of ROM as Data Memory



Interfacing with Data RAM



Interfacing with Data RAM Data ROM and Program ROM

UNIT-IV 8051 Instruction Set and Assembly Language Programming

An addressing mode is a method of specifying the data source or destination in aninstruction. There are 5 types of addressing modes is supported by 8051.

1. Register
2. Immediate
3. Direct (memory related)
4. Register Indirect (memory related)
5. Index register addressing

Register addressing mode

This addressing mode involves the use of registers to hold the data to be manipulated.

Examples:

MOV A, R0 ; Copy the contents of R0 int A

ADD A, R7 ; Add the contents of R7 to contents of A and the result is stored in A

Immediate addressing mode

In this addressing mode immediate data is specified in instruction as a source operand.**Examples:**

MOV B, #40H; load 40H into B registerMOV DPTR, #2000H; load 2000H into DPTR

Direct addressing mode

As we know the on-chip RAM of 8051 is 128 byte, it can be accessed through memory address from 00H to FF H. The allocations of 128 bytes are as follows.

RAM location 00H-1FH are assigned to register banks and stack

RAM location 20H-2FH is set aside as bit-addressable space to save single bit data.

RAM location 30H-7F is available as place to save bite-sized data.

Although the entire 128 bytes of RAM can be accessed through direct addressing mode, it is most often used to access RAM location 30H-7FH. This is due to fact that register banks are accessed through their names.

Examples:

```
MOV R4, 70H        ; move the contents of RAM location 70H to R4.MOV 56H, A
                   ; save the content of A in RAM location 56H
PUSH  05           ; push R5 onto the stack
```

Register indirect addressing mode

In this mode the address (of 8bits) is indirectly specified in the instruction by the contents of pointer. This addressing mode so called because the source operand is from the address specified indirectly by another register in the instruction. The limitation is that only R0 and R1 register can be used in 8051 for indirect addressing. SFRs are directly accessible.

Examples

MOV R1, #55H ; load pointer R1=55H

MOV A, @R1 ; the content of pointer is transferred to A

Index registers addressing

Suppose we need to access external data RAM and external code space of on-chip ROM16 bit address must be required. In this case we have to use DPTR. This mode is widely used in accessing data elements of look-up table entries in the program ROM space of 8051.

Examples;

MOV DPTR, #0200H ; load DPTR with 0200

CLR A ; clear accumulator

MOVC A,@A+DPTR; Move the content 0200 location into A

Instruction set

The instruction set of 8051 can be classified into following group.

- Data Transfer Instructions
- Arithmetic Instructions
- Logic Instructions
- Boolean Variable manipulation Instructions
- Program flow control (Processor and Machine control) Instructions
- Interrupt flow Control instruction
- Data Transfer Instruction

Three types of the data transfer can be done by move instruction. First type is transfer within the internal RAM and SFRs, second type is transfer using code memory area (CODE) andthe third is using the external data memory X-DATA).

MOV instruction

A MOV instruction means move (copy) the bits from one source to a destination.

MOV instructions within the registers, internal RAM and Special Function Registers (SFRs) in 8051

Instruction (Mnemonic)	Action	Addressing	Length in bytes	cycles
MOV A, Rn	Move Rn into A	Register	1	1
MOV Rn, A	Move into Rn from A	Register	1	1
MOV A, #data	Move immediate 8-bit data into A	Immediate	2	1
MOV Rn, #data	Move into Rn the data.	immediate	2	1
MOV A, direct	Move byte at the direct address into A	Direct	2	1
MOV Rn, direct	Move from direct address into Rn	Direct	2	2
MOV direct, A	Move byte to the direct address form A	Direct	2	1
MOV direct, Rn	Move a byte to the direct address from Rn	Direct	2	2
M OV direct, direct	Move byte to the direct address from the direct address	Direct	3	2
MOV direct, #data	Move immediate data byte to the direct address	Immediate	3	2
MOV a,@Ri	Move into A the byte from the address pointed by Ri	Indirect	2	2
MOV @Ri, A	Move A into address pointed by Ri	Indirect	1	1
MOV direct, @Ri	Move into direct address from address pointed by Ri	indirect	1	1
MOV @Ri, direct	Move from the direct address to the address poined by ri	Indirect	2	2
MOV @Ri, #data	Move data ino address pointed by Ri	immediate	2	2
MOV DPTR, data16	Mov e16 bit data	immediate	3	2

MOVC-type Instruction

It moves the 8-bit code from one source at the program memory (internal and external) to the register A destination.

MOVC Instructions for transfer from the program memory area address code orconstant to accumulator in 8051

Instruction	Action	Addressing	Length in bytes	Cycles
MOVC A, @A+DPTR	Moves the code or constant into A the byte from the program memory address pointed by hypothetical addition of DPTR with the A itself.	Indirect	1	2
MOVC A, @A+PC	Move the code or constant into A the byte from the program memory address pointedby hypothetical addition of PC with the A itself	Indirect	1	2

MOX-type Instructions

A MOVX instruction means move (copy) the 8-bit data into A and from A using the external data memory address using DPTR or Ri as the pointer.

MOVX instruction

Instruction	Action	Addressing	Length inbytes	Cycles
MOVX A, @DPTR	Move the external data byte (X-DATA) into A from the data memory address pointed by DPTR	Indirect	1	2
MOVX @DPTR,A	Move into the external data memory from A to the address pointed by DPTR	Indirect	1	2
MOVX A,@Ri	Move the external data byteinto a from the memory address pointed by Ri	Indirect	1	2
MOVX @Ri, A	Move into the external datamemory from A to the memory address pointed by Ri	Indirect	1	2

Instructions to Access External Data Memory

The Table explains the instruction to access external data memory.

MOVX A, @Ri **Example :**	Copy the contents of the external address in Ri to A. **MOVX A, @ R0** : This instruction copies data from the 8-bit address in R0 to A.
MOVX A, @DPTR	This instruction copies data from the 16-bit address in DPTR to A
MOVX @Ri, A **Example :**	Copy data from A to the external address in Ri. **MOVX @R1, A** : This instruction copies data from A to the 8-bit address in R1.
MOVX @DPTR, A	This instruction copies data from A to the 16-bit address in DPTR.

Important Points to Remember in Accessing External Data Memory

• All external data moves with external RAM involve the A register.

• While accessing external RAM, Rp can address 256 bytes and DPTR can address 64 kbytes.

• MOVX instruction is used to access external RAM or I/O addresses.

Instructions to Access External ROM/Program Memory

The Table explains the instructions to access external ROM/program memory.

MOVC A, @A + DPTR	Copy the contents of the external ROM address formed by adding A and the DPTR, to A.
MOVC A, @A + PC	Copy the contents of the external ROM address formed by adding A and the PC, to A.

Important Points to Remember in Accessing External Read Only Memory

• When PC is used to access external ROM, it is incremented by 1 (to point to the next instruction) before it is added to A to form the physical address of external ROM.

• All external data moves with external ROM involve the A register.

• MOVC is used with internal or external ROM and can address 4 K of internal code or 64 K of external code.

• The DPTR and the PC are not changed.

PUSH and POP instructions for using the Stack Area employing SP

Instruction	Action	Addressing	Length inbytes	Cycles
PUSH direct	Move byte from a direct internal RAM or SFR into thestack after first incrementing the stack pointer by 1	Direct	2	2
POP direct	Move byte to a direct internalRAM or SFR into the stack and then decrement the stack pointer by 1.	Direct	2	2

Data Exchange Instructions

When 8051 executes MOV, PUSH or POP instruction, the 'copy operation' takes place. The data from the source address is copied to the destination address. The data at the source address remains unchanged

The Exchange instructions move data from source address to destination address and vice versa.

Important Points to Remember in Exchange Instructions

• All exchanges involve the A register.

• All exchanges take place internally within 8051.

XCH A, Rn **Example :**	Exchange data bytes between register Rn and A. **XCH A, R0** : This instruction exchanges contents of accumulator with the contents of register R0 of selected register bank.
XCH A, direct **Example :**	Exchange data bytes between address directly given within instruction and A. **XCH A, 20H** : This instruction exchanges contents of accumulator with the contents of memory whose address is given within the instruction (20H).
XCH A, @Ri **Example :**	Exchange data bytes between A and address in Ri. **MOV A, @R2** : This instruction exchanges the contents of accumulator with the contents of memory location whose address is given by the contents of register R2 of selected register bank.
XCHD A, @Ri **Example :**	XCHD exchanges the low-order nibble of the Accumulator (bits 3-0), with that of the internal RAM location indirectly addressed by the specified register. The high-order nibbles (bits 7-4) of each register are not affected. R0 contains the address 20H. The Accumulator holds the value 36H (00110110B). Internal RAM location 20H holds the value 75H (01110101B). The instruction, **XCHD A, @R0** will leave RAM location 20H holding the value 76H (01110110B) and 35H (00110101B) in the Accumulator.

• When XCHD A, @ Ri instruction is executed, the upper nibble of A and the upper nibble of the address in Ri do not change.

• Immediate addressing mode cannot be used in the exchange instructions.

Logical Instruction

Table gives features of 8-bit AND, OR and XOR instruction. These instructions have 4addressing modes such as register, immediate, direct and indirect.

ANL, ORL XRL instruction

Instruction	Action	Addressing	Length in bytes	Cycles
ANL A, Rn	AND Rn into A	Register	1	1
ANL A, direct	AND byte at the direct address into A	Direct	2	1
ANL A, @Ri	AND into the byte from the address pointed by the Ri	Indirect	1	1
ANL A, #data	AND immediate data byte into A	immediate	2	1
ANL direct, A	AND A into byte at the direct address	Direct	2	1

ANL direct, #data	AND immediate byte into byte at the direct address	Direct	3	2
ORL A, Rn	OR Rn into A	Register	1	1
ORL A, direct	OR byte at the direct address into A	Direct	2	1
ORL A, @Ri	OR into the byte from the address pointed by Ri	Indirect	1	1
ORL A, #data	OR immediate data byte to the A	immediate	2	1
ORL direct, A	OR A into byte at the direct address	Direct	2	1
ORL direct,#data	OR immediate byte into byte at the direct address	Direct	3	2
XRL A, Rn	XOR Rn into A	Register	1	1
XRL A, direct	XOR byte at the direct address into A	Direct	2	1
XRL A, @Ri	XOR the byte at the address pointed by Ri into A	Indirect	1	1
XRL A, #data	XOR immediate data byte to the A	immediate	2	1
XRL direct, A	XOR A into byte at the direct address	Direct	2	1
XRL direct, #data	XOR immediate byte into byte at the direct address	Direct	3	2

Boolean Variable manipulation Instructions

MOV, CLR, CPL, SETB, ANL, and ORL Boolean Processing Instruction

Instruction	Action	Addressing	Length (bytes)	Cycles
MOV C, bit	Move bit into CF	Direct bit addressing	2	1
MOV bit, C	Move CF into the bit	Direct bit addressing	2	2
CLR C	Clear CF	PSW Register CF bit addressing	1	1
CLR bit	Clear bit	Direct bit addressing	2	1
CPL C	Complement CF	PSW Register CF bit addressing	1	1
CPL bit	Complement bit	Direct bit addressing	2	1
SETB C	Set CF=1	PSW Register CF bit addressing	1	1
SETB bit	Set bit =1	Direct bit addressing	2	1
ANL C,bit	AND between CF and bit, place the result in CF	Direct bit addressing	2	2
ANL C, bit	AND between CF and , place the result in C	Direct bit addressing	2	2
ORL C, bit	OR between CF and bit, place theresult in C	Direct bit addressing	2	2
ORL C, bit	OR between CF and bit , place theresult in C	Direct bit addressing	2	2

Bit Level Logical Instructions

Bit level manipulations are very convenient when it is necessary to set or reset a particular bit in the internal RAM or SFRs. The internal RAM of 8051 from address 20H through 2FH is both byte addressable and bit addressable. However, byte and bit addresses are different. The Table 15.6.1 shows the correspondence between byte and bit addresses.

Byte Address in Hex	Bit Address in Hex	Byte Address in Hex	Bit Address in Hex
20	00-07	28	40-47
21	08-0F	29	48-4F
22	10-17	2A	50-57
23	18-1F	2B	58-5F
24	20-27	2C	60-67
25	28-2F	2D	68-6F
26	30-37	2E	70-77
27	38-3F	2F	78-7F

Bit and Byte Address of Internal RAM

As shown in the Table addresses of bit 0 and bit 7 of internal RAM byte address 20H are 00H and 07H respectively. From Table 15.6.1 we can easily interpolate addresses of bit 1 and bit 6 of internal RAM byte address 26H as 31H and 36H, respectively.

Like internal RAM, some SFRs are bit addressable. The Table shows the bit addressable SFR and the corresponding bit addresses.

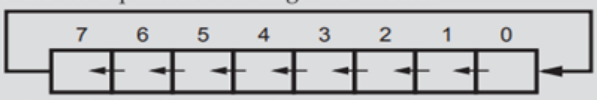
SFR	Direct Address in Hex	Bit Address in Hex
A	E0	E0-E7
B	F0	F0-F7
IE	A8	A8-AF
IP	B8	B8-BF
P0	80	80-87
P1	90	90-97
P2	A0	A0-A7
P3	B0	B0-B7
PSW	D0	D0-D7
TCON	88	88-8F
SCON	98	98-9F

Rotate and Swap Instructions

The Table gives the list of rotate and swap operations supported by 8051.

RL A : Rotate Accumulator Left — **Bytes : 1 Cycles : 1**

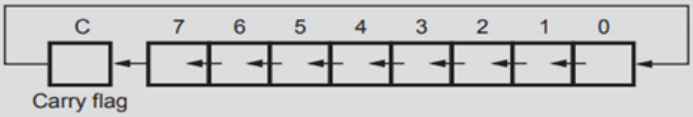
Description : The eight bits in the Accumulator are rotated one bit to the left. Bit 7 is rotated into the bit 0 position. No flags are affected.



Example : The Accumulator holds the value C5H (11000101B). The instruction **RL A** leaves the Accumulator holding the value 8BH (10001011B) with the **carry unaffected**.

RLC A : Rotate A Left through the Carry flag — **Bytes : 1 Cycles : 1**

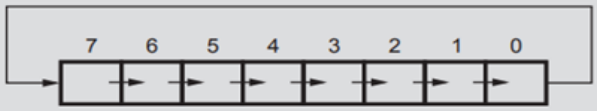
Description : The eight bits in the Accumulator and the carry flag are together rotated one bit to the left. Bit 7 moves into the carry flag; the original state of the carry flag moves into the bit 0 position. No other flags are affected.



Example : The Accumulator holds the value C5H (11000101B), and the carry is zero. The instruction, **RLC A** leaves the Accumulator holding the value 8BH (10001010B) with the carry set.

RR A : Rotate Accumulator Right — **Bytes : 1 Cycles : 1**

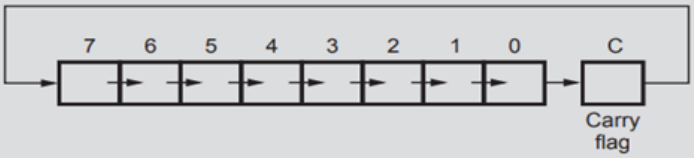
Description : The eight bits in the Accumulator are rotated one bit to the right. Bit 0 is rotated into the bit 7 position. No flags are affected.



Example : The Accumulator holds the value C5H (11000101B). The instruction, **RR A** leaves the Accumulator holding the value E2H (11100010B) with the **carry unaffected**.

RRC A : Rotate A Right through Carry flag — **Bytes : 1 Cycles : 1**

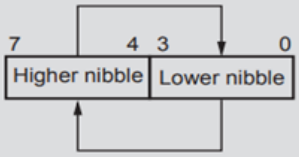
Description : The eight bits in the Accumulator and the carry flag are together rotated one bit to the right. Bit 0 moves into the carry flag; the original value of the carry flag moves into the bit 7 position. No other flags are affected.



Example : The Accumulator holds the value C5H (11000101B), the carry is zero. The instruction **RRC A** leaves the Accumulator holding the value 62 (01100010B) with the carry set.

SWAP A : Swap nibbles within the Accumulator — **Bytes : 1 Cycles : 1**

Description : Swap A interchanges the low and high-order nibbles (four-bit fields) of the Accumulator (bits 3-0 and bits 7-4). The operation can also be thought of as a four-bit rotate instruction. No flags are affected.



Example : The Accumulator holds the value C5H (11000101B). The instruction **SWAP A** leaves the Accumulator holding the value 5CH (01011100B)

Arithmetic Instruction

These instructions include 8 bit addition, subtraction, increment, decrement, multiply anddivision instruction.

Arithmetic ADD, SUB, MUL, DIV, INC and DEC instruction s in 8051

Instruction	Action	Addressing	Flags affected	Length (bytes)	Cycles
ADD A,Rn	Add Rn into A	Register	C,AC,OV	1	1
ADD A, direct	Add the byte at the direct address into A	Direct	C,AC,OV	2	1
ADD A, @Ri	Add the byte from the addresspointed by the Ri into A	Indirect	C,AC,OV	1	1
ADD A, #data	Add immediate data byte to the A	Immediate	C,AC,OV	2	1
ADDC A, Rn	Add CF(carry) bit and Rn into A	Register	C,AC,OV	1	1
ADDC A, direct	Add CF bit and byte at the direct address ito A	Direct	C,AC,OV	2	1
ADDC A @Ri	Add CF bit and the byte from theaddress pointed by the Ri	Indirect	C,AC,OV	1	1
ADDC A, #data	Add CF bit and immediate data byte to the A	Immediate	C,AC,OV	2	1
SBBB A,Rn	Subtract borrow at CF bit and Rn into A	Rgister	C,AC,OV	1	1
SBBB A, direct	Subtract borrow at CF bit and byteat the direct address into A	Direct	C,AC,OV	2	1
SBBB A, @Ri	Subtract borrow at C bit and byte at the byte from the address pointed	Indirect	C,AC,OV	1	1
SBBB A, #data	Subtract borrow at CF bit and immediate data byte into A	Immediate	C,AC,OV	2	1
INC A	Increment	Register	None	1	1

INC Rn	Increment Rn	Register	None	1	1
INC direct	Increment byte at the direct address	Direct	None	2	1
INC @Ri	Increment the byte at the address pointed by Ri	Indirect	None	1	1
DEC A	Decrement A	Register	None	1	1
DEC Rn	Decrement Rn	Register	None	1	1
DEC direct	Decrement byte at the direct address	Direct	None	2	1
DEC @Ri	Decrement the byte at the address pointed by the Ri	Indirect	None	1	1
MUL AB	Multiply A and B Result MSB in Band LSB in A	Register	OV	1	4
DIV AB	Divide A (Numerator) and B(denominator) Remainder in B Quotient in A	Register	OV	1	4
DAA	Decimal adjust accumulator	Register	C	1	1

Long, Absolute and Short Jump

8051 has three jump instructions: Long- it jumps to 16-bit address, Absolute- it jumps within 2 Kbytes and Short- it jumps to address within 128 bytes above or below the present address.

Long, absolute and short jump instructions

Instruction	**Action**	**Addressing**	**Length in bytes**	**Cycles**
LJMP addr16	Jump to the next address given by two bytes in the instruction	Direct 16 bit address	3	2
AJMP addr11	Jump to the next address	Direct 11-bit address	2	2
SJMP *rel*	Jump in the range between -128and +127 from the address of next instruction	Direct 8-bit	2	2
JMP @A+DPTR	Jump in the next address given by addition of 8-bits of A with 16-bits of DPTR	Indirect 16-bit relative address		

Conditional Short Relative Jumps

Instruction	Action	Addressing	Length in bytes	Cycles
JNZ rel	Jump to a relative address if a is not zero	Relative(offset)	2	2
JZ rel	Jump to a relative address if A is zero	Relative(offset)	2	2
JNC rel	Jump to a relative address if CF is not 1	Relative(offset)	2	2
JC rel	Jump to a relative address if CF=1	Relative(offset)	2	2
JB bit, rel	Jump to a relative address if addressed bit 1 (bit not set)	Relative(offset)	2	2
JNB bit,rel	Jump to a relative address if addressed bit 0 (bit not set)	Relative(offset)	2	2
JBC bit, rel	Jump to a relative address if addressed bit 1(bit set) and reset carry(Make CF=0)	Relative(offset)	2	2

Decrement and Conditional jump on Zero

Instruction for decrement and then jump in program-loops in 8051

Instruction	Action	Addressing	Length in bytes	Cycles
DJNZ Rn, Rel	Decrement Rn and jump if Rn is still not zero.	Relative (offset)	2	2
DJNZ direct, Rel	Decrement byte at the direct and jump if byte is still not zero	Relative (offset)	2	2

Jump after comparison

Compare then conditional jump after comparison

Instruction	**Action**	**Addressing**	**Flag affected**	**Length in bytes**	**Cycles**
CJNE A, #data, rel	Compare A and immediate data and jump if both are notequal.	Relative (offset)	C	3	2
CJNE Rn, #data, rel	Compare Rn and immediate data andjump if both are not equal.	Relative (offset	C	3	2
CJNE A, direct, rel	Compare the bytes at A and direct and jump if both are not equal.	Relative (offset	C	3	2

CJNE @Ri, #data, rel	Compare byte from the address pointed by Ri and immediatedata and jump if both are not equal	Relative (offset)	C	3	2

Call to a Routine

Long, absolute call and return instruction

Instruction	**Action**	**Addressing**	**Length in bytes**	**Cycles**
LCALL addr16	Call to the next address given by two bytes in the instruction	Direct 16-bit address	3	2
ACALL addr11	Call the next address given by 11 bits in	Direct 11	2	**2**
RET	Return to PC the saved PCL and PCH from the stack.	Stack address	1	**2**

Interrupt Control Flow (RETI instruction)

RETI instruction

Instruction	**Action**	**Addressing**	**Length In bytes**	**cycles**
RETI	Return into PC the saved PCL and	Stack adddress	1	2

Programming

While the CPU can work only in binary, it can do so at a very high speed, however, it is quite tedious and slow for humans to deal with 0s and 1s in order to program the computer. A programthat consists of 0s and 1s is called machine language. In the early days of the computer programmers coded programs in machine language. Although the hexadecimal system was used as a more efficient way to represent binary numbers, the process of working in machine code was still cumbersome for humans. Eventually, assembly language were developed which provided **mnemonics** for the machine code instructions. Plus other features which made programming faster and less prone to error. Assembly language is referred to as low level language because it deals directly with internal structure of CPU. Programmer needs assemblerto convert the assembly language to machine language for execution purpose. Assembly language consists mnemonics optionally followed by one or two operands.

Programs

P1. Write an ALP (Assembly Language Program) to find the sum of values and store the result in A (lower byte and in R7 (higher byte). Assume that RAM locations 40-44 have the followingvalues. 40=(7B), 41=(EC), 42=(C4), 43=(5B), 44=(30)

Solution:			
	MOV	R0, #40H	; load pointer
	MOV	R2, #05H	; load counter
	CLR	A	; A=0
	MOV	R7, A	; clear R7
AGAIN:	ADD	A, @R0	; add the byte pointer
	JNC	NEXT	; if CY=0 it can jump to NEXT label
	INC	R7	; increment counter
NEXT:	INC	R0	; increment pointer
	DJNZ	R2,AGAIN	; repeat until R2is zero
HERE:	SJMP	HERE	

P2. Assume that 5 BCD data items are stored in RAM locations starting a 40H as shown below. Write an ALP to find the sum of all numbers. The result must be in BCD.

Solution:			
	MOV	R0, #40H	; load pointer
	MOV	R2, #05H	; load counter
	CLR	A	; A=0
	MOV	R7, A	; clear R7
AGAIN:	ADD	A, @R0	; add the byte pointer
	DA	A	
	JNC	NEXT	; if CY=0 it can jump to NEXT label
	INC	R7	; increment counter
NEXT:	INC	R0	; increment pointer
	DJNZ	R2,AGAIN	; repeat until R2is zero
HERE:	SJMP	HERE	

P3. Write an ALP to get hex data in the range of 00-FFH from port 1 and convert it to decimal.Save the digits in R7, R6 and R5, where the least significant digit in R7.

MOV	A, #0FFH	
MOV	P1, A	; make an P1 an input port
MOV	A1, P1	; read data from P1
MOV	B, #0AH	; move 0AH to register b
DIV	AB	; divide by the contents of A by B
MOV	R7, B	; Save lower digit in R7 register
MOV	B , #0AH	;
DIV	AB	;
MOV	R6, B	; save the next digit
MOV	R5, A	; save the last digit
HERE: SJMP	HERE	

P4. Read and test P1 to see whether it has the value 45H. if it does send 99H to P2; otherwise, it stays cleared.

Solution :			
	MOV	P2, 00H	; clear P2
	MOV	P1, #0FFH	; make P1 an input port
	MOV	R3, #45H	; R3=45H
	MOV	A, P1	; read P1
	XRL	A, R3	;
	JNZ	EXIT	
	MOV	P2, #99H	
EXIT:	……		

P5. Find the 2's complement of the value 78 H

Solution:

MOV A, #78H ; A=85H

```
CPL     A          ; make 1's complement a

ADD     A, #01H; make 2's complemt

HERE:   SJMP       HERE
```

P6. Write an ALP to determine if register A contains the value 99H, if so, make R1=FFH otherwise make R1=0.

Solution:

```
        MOV     R1, #00H            ; clear R1

        CJNE    A, #99H, NEXT       ; if A is not equal 99H then jump

        MOV     R1, #0FFH           ;  make R1=FFH

NEXT    .....
```

P7. Assume that P1 is an input port connected to a temperature sensor. Write an ALP to read the temperature and test it for the value 75. According to the rest result, place the temperature value into the registers indicated by the following.

If T=75 then A=75 If T<75 then R1=T If T>75 then R2=T

Solution:			
	MOV	P1, # 0FFH	; make P1 an input port
	MOV	A, P1	; read P1 port, temperature
	CJNE	A, #75, OVER	; jump if A is not equal 75
	SJMP	EXIT	
OVER:	JNC	NEXT	; if CY=0, then A>75
	MOV	R1, A	; if CY=1, A<75
	SJMP	EXIT	; Exit
NEXT:	MOV	R2, A	
EXIT	……		

P8. Write an ALP that finds the number of 1s in a given byte 97H.

Solution:			
	MOV	R1, #00H	; clear R1
	MOV	R7, 08H	; Counter=08
	MOV	A, 97H	
AGAIN:	RLC	A	; rotate through CY once
	JNC	NEXT	; check for CY
	INC	R1	; if CY=1 then increment R1
NEXT:	DJNZ	R7, AGAIN	; go through 8times
HERE:	SJMP	HERE	

P9. Assume that register a has packed BCD 29H, write an ALP to convert packed BCD toASCII numbers and place them in R2 and R6.

Solution:			
	MOV	A, #29H	; A=29H, packed BCD
	MOV	R2, A	; keep a copy of BCD data in R2
	ANL	A, #0FH	; mask the upper nibble (A=09)
	ORL	A, #30H	; make it an ASCII, A=39H
	MOV	A, R6	; save in R6
	MOV	A, R2	; A=29H
	ANL	A, #0F0H	; mask the lower nibble
	RR	A	; rotae right
	RR	A	; rotae right
	RR	A	; rotate right
	RR	A	; rotate right

	ORL	A, #30 H	; A=32H
	MOV	R2, A	; save the ASCII character in R2
HERE:	SJMP	HERE	

P10. Write an ALP to create a square wave of 50% duty cycle on bit 0 of port 1.

Solution:			
HERE:	SETB	P1.0	; set to high bit 0of port 1
	LCALL	DELAY	; call the delay subroutine
	CLR	P1.0	; p1.0=0
	LCALL	DELAY	
	SJMP	HERE	

P11. Assume that the bit P2.2 is used to control the outdoor light and bit P2.5 to control the lightinside the building. Write an ALP to turn on outside light and to turn the inside one.

Solution:

```
SETB    C          ; CY=1
ORL     C, P2.2    ; CY=P2.2
MOV     P2.2, C    ;  turn  it "on" if not already "on"
CLR     C          ; CY=0
ANL     C, P2.5    ; CY=P2.5 ANDed with CY
MOV     P2.5, C    ; turn it off if not already off.
```

V INTERUPPT PROGRAMMING AND INTERFACING TO EXTERNAL WORLD

8051 Microcontroller is a widely used embedded system, that incorporates a robust interrupt system which are important for external communications and real-time applications. Interrupts are the important feature of a microcontroller which enables the microcontroller to respond to the external events and requests, which enhances the multitasking abilities of the microcontroller. An interrupt is an external or internal event/command that interrupts the normal processing of an event and informs the microcontroller that a device needs its service. Whenever a device needs its service, the device sends an interrupt signal to the microcontroller to send a notification. Upon receiving the interrupt signal, the microcontroller stops its existing program and serves the external device request. The program which is associated with the interrupt is known as interrupt Service Routine (IRS) or interrupt handler.

The 8051 features two main types of interrupts, i.e. Hardware interrupts and software interrupts. The hardware interrupts are triggered by external signal such as peripheral events or external devices. The microcontroller can be configured to respond to specific events, allowing for efficient event-driven programming. Whereas, the Software interrupts, are initiated by specific instructions in the program code. They provide a mechanism for the programmer to force the microcontroller to interrupt its normal execution and execute a predefined routine.

The address of the corresponding interrupt service routine (ISR) is included in the suitable interrupt vector associated with every interrupt source in the 8051. The microcontroller automatically maintains its state on interrupt, fetches the interrupt vector's ISR address, and executes the ISR's operation. Once the ISR is finished, the microcontroller restarts the task which has been interrupted.

What is an 8051 Microcontroller?

8051 microcontroller is an 8-bit data bus and 16-bit address bus Microcontroller. A 64K (216) byte code memory space and an additional 64K byte data memory space can be addressed using the 16-bit address bus. It has 40 pins and 4K on-chip read only code memory and 128 bytes of internal RAM. It also has various *Special Function Registers (SFR)* such as the accumulator, the B register, and many other control registers. At a time, the ALU executes an 8-bit operation. It also has two 16-bit counter timers and 3 internal interrupts and 2 external interrupts and four 8 bit I/O ports.

8051 Microcontroller Interrupt

The timer and serial interrupts are internally generated by the microcontroller whereas, the external interrupts are generated by additional peripheral devices or switches that are connected to the microcontroller externally. There are two types of external interrupts: edge-triggered and level-triggered. The interrupt service routine is carried out by the microcontroller as a reaction to an interrupt, enabling memory locations to coincide with interrupts.

Interrupt Vector Table

The addresses of different interrupt service routines (ISRs) are stored in a table called the Interrupt Vector Table (IVT) in an 8051 microcontroller. It is a vital aspect of the interrupt handling mechanism in the microcontroller. When an interrupt occurs the interrupt specific ISR is executed by jumping the program counter to the corresponding address in the IVT. There are memory areas set aside specifically for the IVT in the 8051 microprocessors. Every interrupt has a specific place in the IVT, and the addresses kept there point to the program memory's associated ISR's start. By guiding the program flow to the proper place, the IVT enables the microcontroller to respond to external events like hardware interrupts or external signals quickly and effectively.

Interrupt	Flag	Interrupt Vector Address
Reset	-	0000H
INT0 (External Interrupt 0)	IE0	0003H
Timer 0	TF0	000BH
INT1 (External Interrupt 1)	IE1	0013H
Timer 1	TF1	001BH
Serial Interrupt	TI/RI	0023H

Interrupt structure of 8051 Microcontroller

All of the interrupts are disabled by **"RESET"** thus software is required to enable all of these interrupts. If any one of these five interrupts or all five are activated, the relevant interrupt flags are set. The priority, which is managed by the IP interrupt priority register, determines which of these interrupts can be set or cleared bit by bit in a specific function register that is Interrupt Enabled (IE).

Two SFRs controls the function of interrupts in 8051 microcontrollers. IE is Responsible for disable/enable the function and IP is Responsible for priority assignment: The priority list offers 3 levels of interrupt priority: Reset: When a reset request arrives, everything is stopped and the microcontroller restarts. Reset can be used to disable the interrupt priority 1. Interrupt priority 0 can be disabled by both Reset and interrupt.

Some of the registers used in this microcontroller are :

- IE (Interrupt Enable) Register
- IP (Interrupt Priority) Register
- TCON (Timer Control) Register

IE (Interrupt Enable) Register

Interrupts can be enabled and disabled using IE Register. It is a register in the 8051 microcontroller that controls interrupt prioritization and triggering. It includes many bits, such as:

EA-Global Interrupt Enable/Disable - When it is set it enables all interrupt, if cleared disables all interrupts

- 0- Disables all interrupt requests
- 1- Enables all interrupt requests

ES (Serial Communication Interrupt Enable)- This bit enables or disables the interrupt for serial communication.

- 1- Interrupt is enabled by UART system
- 0-Interrupt cannot be generated by UART system

ET0- Bit enables or disables timer 0 interrupt.

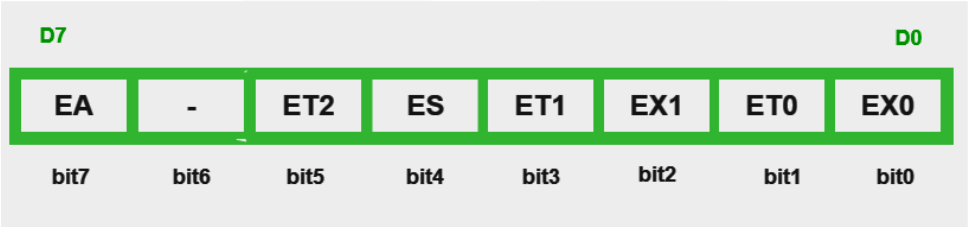
- 1-Timer 0 is enables an interrupt
- 0-Interrupt cannot be generated by the timer 0

ET1- Bit enables or disables timer 1 interrupt.

- 1- Interrupt is enabled by timer 1
- 0-Interrupt is disabled by timer 1

EX0 and EX1 (External Interrupt 0 and External Interrupt 1 Enable)

- These bits control the interrupts from external devices.
- EX0 - bit enables or disables external 0 interrupt: 0 - change of the INT1 pin logic state cannot generate an interrupt and 1 - enables an external interrupt on the pin INT1 state change.
- EX1 - bit enables or disables external 1 interrupt: 0 - change of the pin INT0 logic state cannot generate an interrupt and 1 - enables an external interrupt on the pin INT0 state change.
- **IT0 and IT1 (External Interrupt 0 and External Interrupt 1 Type)**- These bits determine the type of trigger for external interrupts (level or edge-triggered).



P (Interrupt Priority) Register

One cannot predict when one may receive an interrupt request. If multiple interrupts are enabled, it can happen that a request for another interrupt is made while the first one is ongoing. There is a priority list that tells the microcontroller what to do in order to determine whether to respond to a new interrupt request or to carry on with existing operations. The microcontroller restarts once everything stops in response to a reset request. Only Reset has the ability to disable Interrupt priority 1. Both Reset and interrupt priority 1 have the ability to disable interrupt priority 0. The interrupt priority register, or IP Register, indicates which of the current interrupt sources is more significant than other. The program's start typically defines the interrupt priority. An interrupt will be immediately paused and given preference over any other interrupt if the one with greater priority comes while the other is still in progress. Whenever two interrupt requests that have different priorities occurs simultaneously, the higher priority interrupt is handled first. If two interrupt requests with the same priority level arise one after the other, the subsequent request needs to wait until the entire process is accomplished.

Bit0 (PX0)- External0 interrupt priority bit

- 0- Sets low priority to external 0 interrupt
- 1- Sets high priority to external 0 interrupt

Bit1 (PT0)- Timer 0 interrupt priority bit

- 0- Assigns low priority to Timer0 interrupt

- 1- Assigns high priority to Timer0 interrupt

Bit2 (PX1)- External1 interrupt priority bit

- 0- Sets low priority to external1 interrupt
- 1- Sets high priority to external1 interrupt

Bit3 (PT1)- Timer1 interrupt priority bit

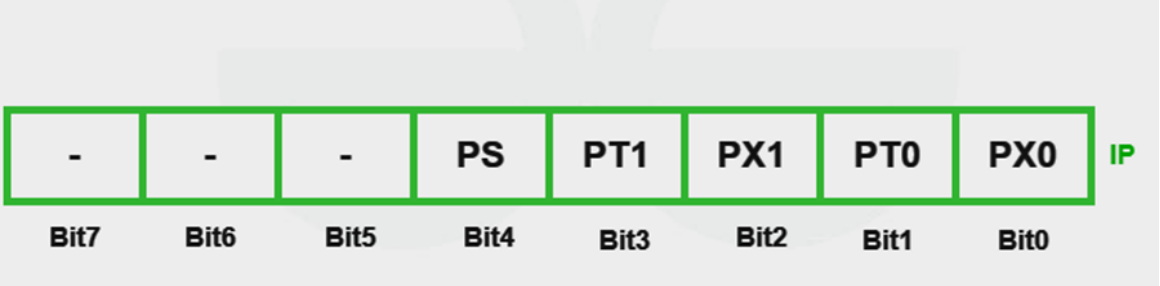
- 0- Sets low priority to timer1 interrupt
- 1- Sets high priority to timer1 interrupt

Bit4 (PS)- Serial Input priority bit

- 0- Assigns low priority to serial interrupt
- 1- Assigns high priority to serial interrupt

Bit 5,6 & 7- These bits are called as the Reserved bits.

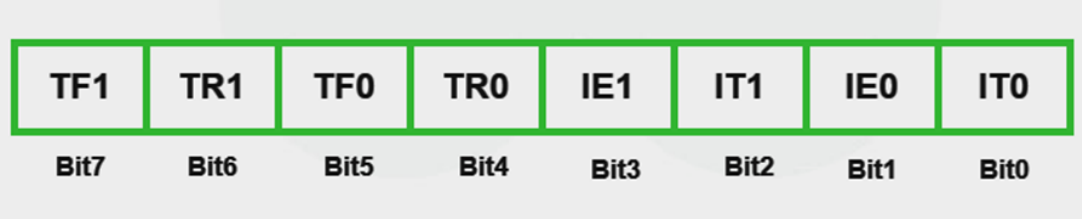
IP (Interrupt Priority) Register



TCON (Timer Control) Register

The interruptions that the microcontroller interfaces with (external) devices are known as external interrupts. They are received by the controller's INTx pins. These may be triggered by edges or levels. Interrupt is enabled for a low at the INTx pin when it is level triggered, and for a high to low transition at the INTx pin when it is edge triggered. The TCON register determines whether the triggering is edge or level trigger. The INTx pin for a level trigger interrupt needs to remain low until the interrupt begins and needs to go back to high prior to the interrupt terminating. An interrupt won't be produced if the low at the INTx pin rises to a high value before the ISR begins. Additionally, the interrupt will be created once more if the INTx pin is low even after the ISR has ended. The level trigger interrupt (low) at the INTx pin must therefore be four machine cycles long, neither longer nor shorter than this.

- **IE0- External interrupt 0 edge flag-** When an external interrupt edge is detected, hardware sets it. Cleared by the device upon processing the interrupt.
- **IE1- External interrupt 1 edge flag-**Set by hardware when external interrupt edge is detected. Cleared by hardware when the interrupt is processed.
- **TF0- Timer 0 overflow flag-**This bit is set whenever timer 0 overflows and is processed by the hardware.
- **TF1- Timer 1 overflow flag -** This bit is set whenever timer 1 overflows and is processed by the hardware.
- **TR0- Timer 0 Run Control-** Set this bit to start Timer 0 and clear it to stop the timer. This is important because the timer needs to be running for it to generate interrupts.
- **TR1 Timer 1 Run Control - S**imilar to TR0, this bit controls the running state of Timer 1.
- **IT0- Interrupt 0 type control bit-** Set/cleared by the device or software to indicate falling edge/low-level triggered external interrupts.
- **IT1- Interrupt 1 type control bit-** Set/cleared by the device or software to indicate falling edge/low-level triggered external interrupts. Whenever the IT0 and IT1 bits are set, the external interrupts 0 and 1 edge-triggered respectively. These bits are cleared by default, which causes the external interrupt to be level triggered.



Types of 8051 Microcontroller Interrupts

8051 Microcontroller suffers five different types of interrupts that hampers the main program execution. These five types of interrupts are:

- Timer 0 overflow interrupt- TF0
- Timer 1 overflow interrupt-TF1
- External hardware interrupt- INT0
- External hardware interrupt- INT1

- Serial communication interrupt- RI/TI

External Hardware Interrupt- (INT0 & INT1)

The 8051 microcontrollers are able to respond to external events through its external interrupts, INT0 and INT1.

External Interrupt 0 (INT0)

- It is connected to the 8051's pin PORT3.2.
- An interrupt request is issued when this pin transitions from low to high in response to an external signal.
- It is possible to program the microcontroller to carry out a particular Interrupt Service Routine (ISR) in response to this interrupt.
- Set the IE (Interrupt Enable) bit for INT0 in the TCON register and configure the IT0 (Interrupt Type 0) bit in the TCON register corresponding to the desired triggering condition (edge or level-triggered) in order to enable and configure INT0.

External Interrupt 1 (INT1)

- It is connected to the 8051's pin PORT3.3
- When that particular pin encounters a low-to-high transitions, INT1, like INT0, creates an interrupt request.
- By configuring the IT1 (Interrupt Type 1) bit in the TCON register and setting the IE bit for INT1 in the TCON register, one can enable and configure INT1.
- A specific ISR can be executed by the microcontroller in response to INT1.

Timer Interrupts (Timer0 and Timer1)

Timer 0 and Timer 1 are hardware timers with internal timer interrupts featured in the 8051 microcontrollers. In microcontroller applications, these timers are used to measure time intervals and generate precise delays. The interrupt system of the microcontroller enables it to react quickly to outside events. Interrupts for Timer 0 and Timer 1 are produced when their respective timers exceed their limit. The microcontroller will run the interrupt service routine (ISR) for that timer if the related interrupt is enabled, and the associated interrupt flag is set upon overflow.

Timer 0 Interrupt

- Since Timer 0 is an 8-bit timer, its count range is 0 to 255.
- There are two modes of operation for it, 13-bit and 16-bit. It employs the TH0 (Timer 0 High) and TL0 (Timer 0 Low) registers in 13-bit mode and only the TH0 register in 16-bit mode.
- It is possible to set timer 0 to interrupt when it approaches zero instead of staying at its maximum value. The microcontroller can perform a particular interrupt service routine (ISR) in response to the interrupt request that this overflow generates.

Timer 1 Interrupt

- Timer 1 is a 16-bit timer with a counting range of 0 to 65,535.
- It can operate in 16- or 8-bit mode. It employs the TL1 (Timer 1 Low) and TH1 (Timer 1 High) registers in 8-bit mode and only the TH1 register in 16-bit mode.
- Timer 1 can be set up to produce an interrupt when it overflows, just like Timer 0. This interruption may cause a certain ISR to be executed.

Serial Communication Interrupts (UART)

UART (Universal Asynchronous Receiver/Transmitter) is a serial communication protocol used with 8051 microcontrollers. Data is sent over a single cable, bit by bit, in serial transmission. In this sense, "interrupts" refers to the processes that enable the microcontroller to react quickly to external events.

Addressing UART communication with the 8051's interrupts:

- **Initialization of UART-** Set the data format, baud rate, and enable the UART module by configuring the UART registers.
- **Interrupt Enable**- Depending on the operation you wish to interrupt for, enable the UART's transmit interrupt (TI) or receive interrupt (RI).
- **ISR (interrupt service routine)-** To handle the interrupt, write an ISR. The ISR in UART communication normally verifies whether the transmit buffer is ready (TI) or whether data has been received (RI).
- **Clearing the Flag- To** recognize the interrupt and get ready for the next one, in the ISR, clear the associated interrupt flag (RI or TI).

Applications of 8051 Interrupts

- In real-time systems, interrupts are essential for speedy responses to external occurrences. Due to its ability to swap tasks fast through interrupts, the 8051 is a good choice for applications that need exact timing.
- Interrupts serve in the control of incoming data in communication applications, assuring timely transmission or rapid handling of received information.
- In embedded systems, 8051 interrupts are frequently utilized for activities like sensor interfacing, where the microcontroller must react quickly to environmental changes.
- Interruptions assist in ensuring timely data sampling and processing in situations when data must be obtained from sensors or external devices.
- By enabling instantaneous reactions to sensor triggers or alarm situations, interrupts in security systems can ensure prompt alerting and suitable responses.

Nested Interrupts

Nested interrupts allow an interrupt service routine (ISR) to be interrupted by another ISR

- Enables higher-priority interrupts to be serviced even when a lower-priority ISR is executing
- Requires support from the microcontroller architecture and proper configuration

Nesting depth determines how many levels of interrupts can be nested

- Deeper nesting allows more flexibility but increases complexity and memory usage
- Shallower nesting limits the number of concurrent interrupts but simplifies the system design

Nested interrupts require careful management of shared resources and data structures

- ISRs must save and restore any registers they modify to avoid corrupting the state of other ISRs
- Reentrancy issues can arise when multiple ISRs access the same resources (semaphores, mutexes)

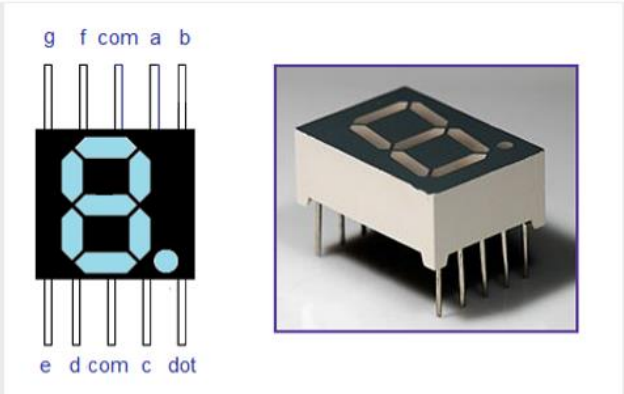
Nested interrupts can significantly improve system responsiveness and real-time performance

- Allows high-priority events to be handled promptly, even during the execution of lower-priority tasks

- Commonly used in applications with strict timing requirements (motor control, communication protocols)

Interfacing Seven segment display to 8051-LED

7 segment LED display is very popular and it can display digits from 0 to 9 and quite a few characters like A, b, C, ., H, E, e, F, n, o,t,u,y, etc. Knowledge about how to interface a seven-segment display to a micro controller is very essential in designing embedded systems. A seven-segment display consists of seven LEDs arranged in the form of a squarish **'8'** slightly inclined to the right and a single LED as the dot character. Different characters can be displayed by selectively glowing the required LED segments. Seven segment displays are of two types, ***common cathode and common anode.*** In common cathode type, the cathode of all LEDs are tied together to a single terminal which is usually labeled as '**com**' and the anode of all LEDs are left alone as individual pins labelled as a, b, c, d, e, f, g & h (or dot) . In common anode type, the anode of all LEDs are tied together as a single terminal and cathodes are left alone as individual pins. The pin out scheme and picture of a typical 7 segment LED display is shown in the image below.



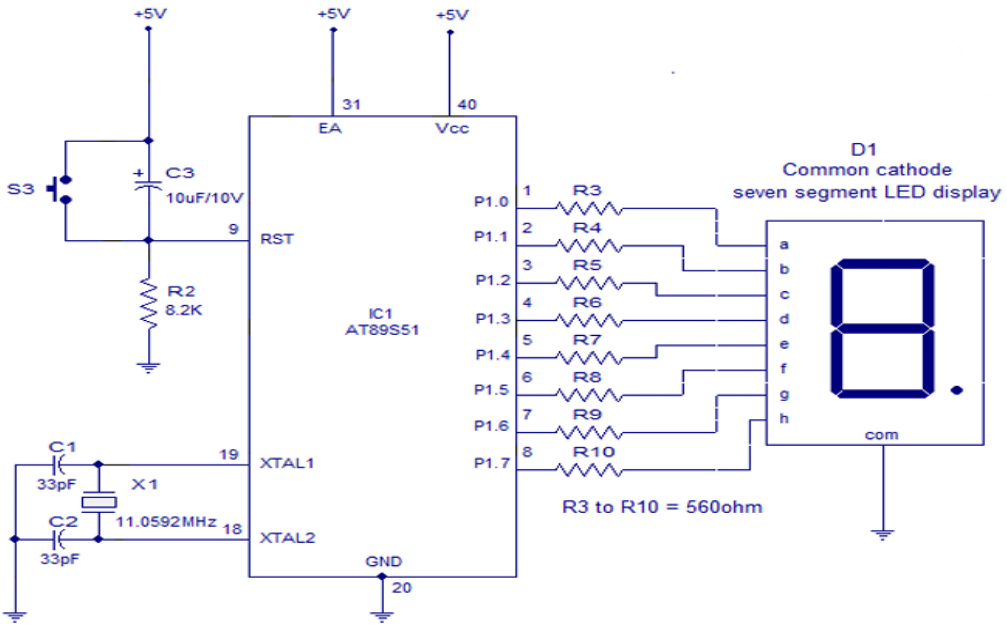
7 segment LED display

Digit drive pattern.

Digit drive pattern of a seven segment LED display is simply the different logic combinations of its terminals **'a' to 'h'** in order to display different digits and characters. The common digit drive patterns (0 to 9) of a seven segment display are shown in the table below.

Digit	a	b	c	d	e	f	g
0	1	1	1	1	1	1	0
1	0	1	1	0	0	0	0
2	1	1	0	1	1	0	1
3	1	1	1	1	0	0	1
4	0	1	1	0	0	1	1
5	1	0	1	1	0	1	1

6	1	0	1	1	1	1	1
7	1	1	1	0	0	0	0
8	1	1	1	1	1	1	1
9	1	1	1	1	0	1	1



Interfacing seven segment display to 8051.

The circuit diagram shown above is of an AT89S51 microcontroller based 0 to 9 counter which has a 7 segment LED display interfaced to it in order to display the count. This simple circuit illustrates two things. How to setup simple 0 to 9 up counter using 8051 and more importantly how to interface a seven segment LED display to 8051 in order to display a particular result. The common cathode seven segment display D1 is connected to the Port 1 of the microcontroller (AT89S51) as shown in the circuit diagram. R3 to R10 are current limiting resistors. S3 is the reset switch and R2,C3 forms a debouncing circuitry. C1, C2 and X1 are related to the clock circuit. The software part of the project has to do the following tasks.

- Form a 0 to 9 counter with a predetermined delay (around 1/2 second here).
- Convert the current count into digit drive pattern.
- Put the current digit drive pattern into a port for displaying.

All the above said tasks are accomplished by the program given below.

Program.

```
ORG000H//initial starting address
START:MOVA,#00001001B // initial value of accumulator
MOV B,A
MOV R0,#0AH //Register R0 initialized as counter which counts from 10 to 0
LABEL: MOV A,B
INC A
MOV B,A
MOVC A,@A+PC // adds the byte in A to the program counters address
MOV P1,A
ACALL DELAY // calls the delay of the timer
DEC R0//Counter R0 decremented by 1
MOV A,R0 // R0 moved to accumulator to check if it is zero in next instruction.
JZ START //Checks accumulator for zero and jumps to START. Done to check if counting has been finished.
SJMP LABEL
DB 3FH // digit drive pattern for 0
DB 06H // digit drive pattern for 1
DB 5BH // digit drive pattern for 2
DB 4FH // digit drive pattern for 3
DB 66H // digit drive pattern for 4
DB 6DH // digit drive pattern for 5
DB 7DH // digit drive pattern for 6
DB 07H // digit drive pattern for 7
DB 7FH // digit drive pattern for 8
DB 6FH // digit drive pattern for 9
DELAY: MOV R4,#05H // subroutine for delay
WAIT1: MOV R3,#00H
WAIT2: MOV R2,#00H
WAIT3: DJNZ R2,WAIT3
DJNZ R3,WAIT2
DJNZ R4,WAIT1
RET
END
```

About the program.

Instruction MOVC A,@A+PC is the instruction that produces the required digit drive pattern for the display. Execution of this instruction will add the value in the accumulator A with the content of the program counter(address of the next instruction) and will move the data present in the resultant address to A. After this the program resumes from the line after MOVC A,@A+PC.

In the program, initial value in A is 00001001B. Execution of MOVC A,@A+PC will add oooo1001B to the content in PC (address of next instruction). The result will be the address of label DB 3FH (line15) and the data present in this address ie 3FH (digit drive pattern for 0) gets moved into the accumulator. Moving this pattern in the accumulator to Port 1 will display 0 which is the first count.

At the next count, value in A will advance to 00001010 and after the execution of MOVC A,@+PC ,the value in A will be 06H which is the digit drive pattern for 1 and this will display 1 which is the next count and this cycle gets repeated for subsequent counts.

The reason why accumulator is loaded with 00001001B (9 in decimal) initially is that the instructions from line 9 to line 15 consumes 9 bytes in total.

The lines 15 to 24 in the program which starts with label DB can be called as a **Look Up Table (LUT)**. label DB is known as Define Byte – which defines a byte. This table defines the digit drive patterns for 7 segment display as bytes (in hex format). MOVC operator fetches the byte from this table based on the result of adding PC and contents in the accumulator.

Register B is used as a temporary storage of the initial value of the accumulator and the subsequent increments made to accumulator to fetch each digit drive pattern one by one from the look up table(LUT).

INTERFACING OF DAC & ADC

Digital-to-analog conversion is a process in which signals having a few (usually two) defined levels or states are converted into signals having a theoretically infinite number of states. A common example is the processing, of computer data into audio-frequency (AF) tones that can be transmitted over a twisted pair by a modem of computer data telephone line. The circuit that performs this function is a digital-to-analog converter(DAC).

Basically, digital-to-analog conversion is the opposite of analog-to-digital conversion. In most cases, if an analog-to-digital converter (ADC) is placed in a communications circuit after a DAC, the digital signal output is identical to the digital signal input. Also, in most instances when a DAC is placed after an ADC, the analog signal output is identical to the analog signal input.

Binary digital impulses, all by themselves, appear as long strings of ones and zeroes, and have no apparent

meaning to a human observer. But when a DAC is used to decode the binary digital signals, meaningful output appears. This might be a voice, a picture, a musical tune, or mechanical motion. Both the DAC and the ADC are of significance in some applications of digital signal processing. The intelligibility or fidelity of an analog signal can often be improved by converting the analog input to digital form using an ADC, then clarifying the digital signal, and finally converting the "cleaned-up" digital impulses back to analog form using a DAC. A common use of digital-to-analog converters is generation of audio signals from digital information in music players. Digital video signals are converted to analog in televisions and cell phones to display colors and shades.

Digital-to-analog conversion can degrade a signal, so conversion details are normally chosen so that the errors are negligible. Due to cost and the need for matched components DACs are almost exclusively manufactured on integrated circuits (ICs). There are many DAC architectures which have different advantages and disadvantages. The suitability of a particular DAC for an application is determined by a variety of measurements including speed and resolution. The digital to analog converter is a device widely used to convert digital pulses to analog signals. The two methods of creating DAC are binary weighted and R-2R ladder. DAC 0808 uses the R-2R method since it can achieve a high degree of precision. The first criterion for judging a DAC is its resolution, which is the function of the number of binary inputs. The common ones are 8, 10 and 12 bits. The number of data bit inputs decides the resolution of the DAC since the number of analog output levels is equal to 2n, where n is the number of data inputs. DAC 0808 provides 256 discrete voltage or current levels of output. In DAC 0808, the digital inputs are converted into current Iout and by connecting a resistor to Iout pin, we convert the result to voltage. The total current provided by IOUT pin is a function of binary numbers at the D0-D7 pins inputs to DAC 0808 and reference current (Iref) is as follows:

$$\mathbf{I_{out}=I_{ref}\,(D7/2+D6/4+D5/8+D4/16+D3/32\ldots+D0/256)}$$

Where D0 is the LSB, D7 is the MSB for the inputs and I_{ref} is the input current that must be applied.

Algorithm for interface 8051 with DAC:

Step1: Connect the P1 of 8051 with D0-D7 pins of DAC

Step2: Give +5v to VCC & V_{ref} of DAC

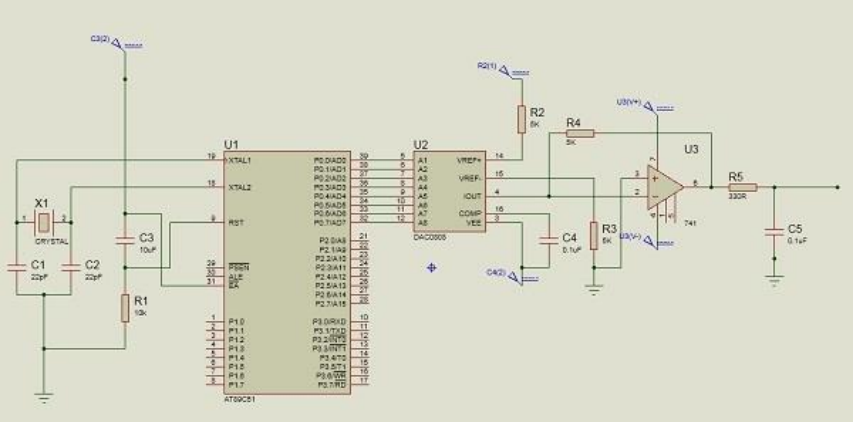
Step3: Connect -12v to VEE of DAC

Step4: Connect OPAMP to OUT pin of the DAC With 5K resistor

Step5: Connect the oscilloscope to the OPAMP to View the output

Digital to Analog converters are required when a digital code must be converted to analog signal. It has eight digital input lines and an output line for analog signal. The number of data bits reduces resolution of DAC. Outputting digital data 00 to FF at regular intervals to DAC, results in generation of different waveforms namely square wave, triangular wave,stair case wave etc.

INTERFACING OF DAC WITH CONTROLLER



Analog to Digital Converter using 8051 microcontroller

In electronics, an analog-to-digital converter (ADC, A/D, or A-to-D) is a system that converts an analog signal, such as a sound picked up by a microphone or light entering a digital camera, into a digital signal. An ADC may also provide an isolated measurement such as an electronic device that converts an input analog voltage or current to a digital number representing the magnitude of the voltage or current. Typically the digital output is a two's complement binary number that is proportional to the input, but there are other possibilities.

There are several ADC architectures. Due to the complexity and the need for precisely matched components, all but the most specialized ADCs are implemented as integrated circuits (ICs).A digital-to-analog converter (DAC) performs the reverse function; it converts a digital signal into an analog signal.An ADC converts a continuous-time and continuous-amplitude analog signal to a discrete-time and discrete-amplitude digital signal. The conversion involves quantization of the input, so it necessarily introduces a small amount of error or noise. Furthermore, instead of continuously performing the conversion, an ADC does the conversion periodically, sampling the input, limiting the allowable bandwidth of the input signal.

The performance of ADC is characterized by its bandwidth and its signal-to-noise ratio. The bandwidth of an ADC is characterized primarily by its sampling rate. The dynamic range of an ADC is influenced by many factors, including the resolution, linearity and accuracy (how well the quantization levels match the true analog signal), aliasing and jitter. The dynamic range of an ADC is often summarized in terms of its effective number of bits (ENOB), the number of bits of each measure it returns that are on average not noise. An ideal ADC has an ENOB equal to its resolution. ADCs are chosen to match the bandwidth and required signal-to-noise ratio of the signal to be quantized. If an ADC operates at a sampling rate greater than twice the bandwidth of the signal, then perfect reconstruction is possible given an ideal ADC and neglecting quantization error. The presence of quantization error limits the dynamic range of even an ideal ADC. However, if the dynamic range of the ADC exceeds that of the input signal, its effects may be neglected resulting in an essentially perfect digital representation of the input signal.

Types of ADC's

ADCs come in various speeds, use different interfaces, and provide differing degrees of accuracy. The most common types of ADCs are flash, successive approximation, and sigma-delta.

Flash ADC

The flash ADC is the fastest type available. A flash ADC uses comparators, one per voltage step, and a string of resistors. A 4-bit ADC will have 16 comparators, an 8-bit ADC will have 256 comparators. All of the comparator outputs connect to a block of logic that determines the output based on which comparators are low and which are high.The conversion speed of the flash ADC is the sum of the comparator delays and the logic delay (the logic delay is usually negligible). Flash ADCs are very fast, but consume enormous amounts of IC real estate. Also, because of the number of comparators required, they tend to be power hogs, drawing significant current. A 10-bit flash ADC may consume half an amp.A variation on the flash converter is the half-flash, which uses an internal digital-to-analog converter (DAC) and subtraction to reduce the number of internal comparators. Half-flash converters are slower than true flash converters but faster than other types of ADCs. We'll lump them into the flash converter category.

Successive approximation converter

A successive approximation converter uses a comparator and counting logic to perform a conversion. The first step in the conversion is to see if the input is greater than half the reference voltage. If it is, the most significant bit (MSB) of the output is set. This value is then subtracted from the input, and the result is checked for one quarter of the reference voltage. This process continues until all the output bits have been set or reset. A successive approximation ADC takes as many clock cycles as there are output bits to perform a conversion.

Sigma-delta

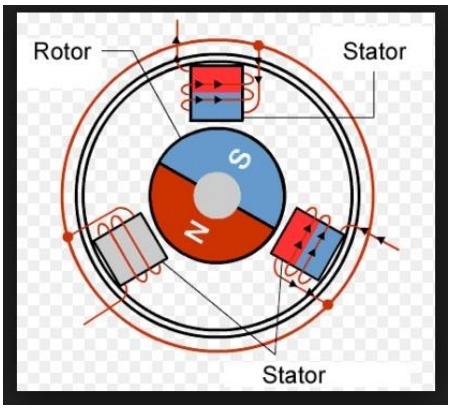
A sigma-delta ADC uses a 1-bit DAC, filtering, and oversampling to achieve very accurate conversions. The conversion accuracy is controlled by the input reference and the input clock rate.The primary advantage of a sigma-delta converter is high resolution. The flash and successive approximation ADCs use a resistor ladder or resistor string. The problem with these is that the accuracy of the resistors directly affects the accuracy of the conversion result. Although modern ADCs use very precise, laser-trimmed resistor networks, some inaccuracies still persist in the resistor ladders. The sigma-delta converter does not have a resistor ladder but instead takes a number of samples to converge on a result.The primary disadvantage of the sigma-delta converter is speed. Because the converter works by oversampling the input, the conversion takes many clock cycles. For a given clock rate, the sigma-delta converter is slower than other converter types. Or, to put it another way, for a given conversion rate, the sigma-delta converter requires a faster clock. Another disadvantage of the sigma-delta converter is the complexity of the digital filter that converts the duty cycle information to a digital output word. The sigma-delta converter has become more commonly available with the ability to add a digital filter or DSP to the IC die.

Stepper Motor

Stepper motors are used to translate electrical pulses into mechanical movements. In some disk drives, dot matrix printers, and some other different places the stepper motors are used. The main advantage of using the stepper motor is the position control. Stepper motors generally have a permanent magnet shaft (rotor), and it is surrounded by a stator.

Normal motor shafts can move freely but the stepper motor shafts move in fixed repeatable increments. Some parameters of stepper motors –

- **Step Angle** − The step angle is the angle in which the rotor moves when one pulse is applied as an input of the stator. This parameter is used to determine the positioning of a stepper motor.
- **Steps per Revolution** − This is the number of step angles required for a complete revolution. So the formula is 360° /Step Angle.
- **Steps per Second** − This parameter is used to measure a number of steps covered in each second.
- **RPM** − The RPM is the Revolution Per Minute. It measures the frequency of rotation. By this parameter, we can measure the number of rotations in one minute.



The relation between RPM, steps per revolution, and steps per second is like below:

Steps per Second = rpm x steps per revolution / 60

Interfacing Stepper Motor with 8051 Microcontroller

Weare using Port P0 of 8051 for connecting the stepper motor. HereULN2003 is used. This is basically a high voltage, high current Darlington transistor array. Each ULN2003 has seven NPN Darlington pairs. It can provide high voltage output with common cathode clamp diodes for switching inductive loads.

The Unipolar stepper motor works in three modes.

- **Wave Drive Mode** − In this mode, one coil is energized at a time. So all four coils are energized one after another. This mode produces less torque than full step drive mode.

The following table is showing the sequence of input states in different windings.

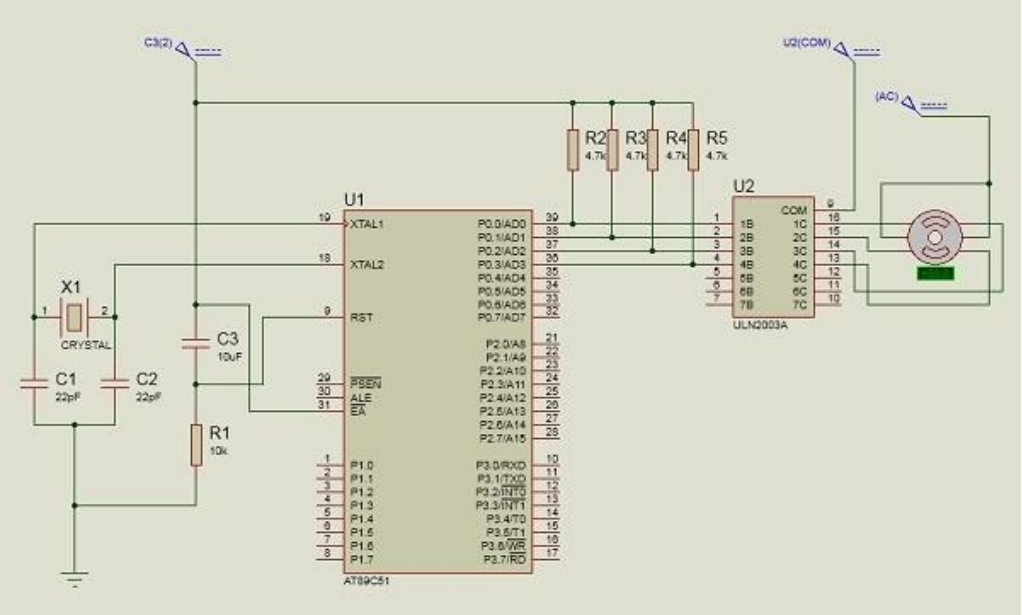
Steps	Winding A	Winding B	Winding C	Winding D
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

- **Full Drive Mode** − In this mode, two coils are energized at the same time. This mode produces more torque. Here the power consumption is also high
- **Half Drive Mode** − In this mode, one and two coils are energized alternately. At first, one coil is energized then two coils are energized. This is basically a combination of wave and full drive mode. It increases the angular rotation of the motor

The following table is showing the sequence of input states in different windings.

Steps	Winding A	Winding B	Winding C	Winding D
1	1	0	0	0
2	1	1	0	0
3	0	1	0	0
4	0	1	1	0
5	0	0	1	0
6	0	0	1	1
7	0	0	0	1
8	1	0	0	1

The circuit diagram is like below: We are using the full drive mode.



MEASUREMENT OF CURRENT- 8051 CONTROLLER:

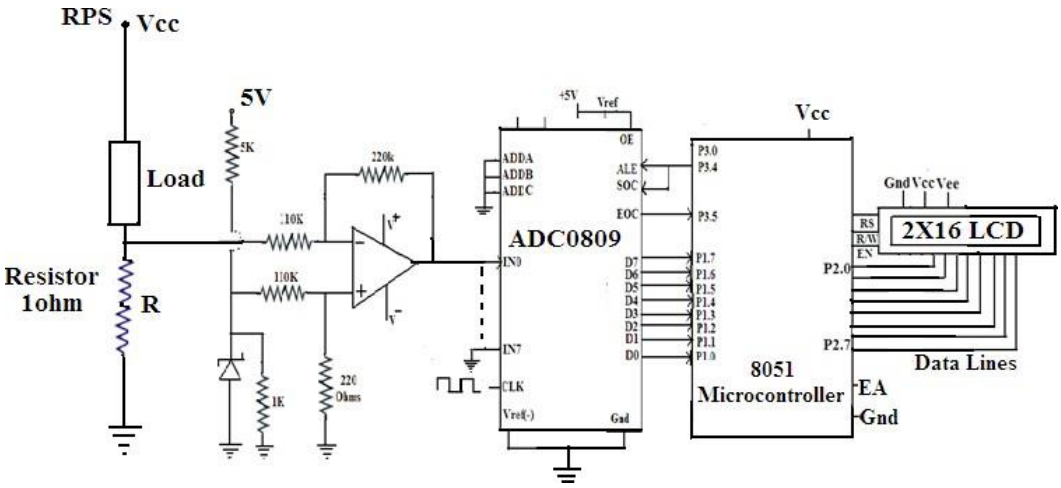
Small currents can be measured using the microcontroller. For this a resistor of small value(0.1 ohm to 1 ohm) is connected to the load and V_{cc} in series as shown in the diagram. The smallvoltage drop (mV) across the resistor is connected to the inverting terminal of op-amp for signal conditioning. The signal conditioning circuit will make the voltage suitable to apply to one of theinputs of the ADC0809.This ADC will convert this analog voltage into equivalent digital value. This digital signal is applied to the Port 1 of the microcontroller. The data is processed and displayed on the LCD module.

To calculate the current Ohm's law, V= IR, is used. If the sense resistor's value is 1 ohm and when 1 ampere of current is flowing in the resistor , then 1 Volt will be developed across the senseresistor. So, I = V / R

By measuring the voltage using the microcontroller, the current in the resistor can be found.

Note : This method is useful to measure small currents only. Because with large currents the small resistor will dissipate large heat. So, to measure large currents high wattage wire wound resistors or Toroid must be used.

The circuit diagram is shown below.



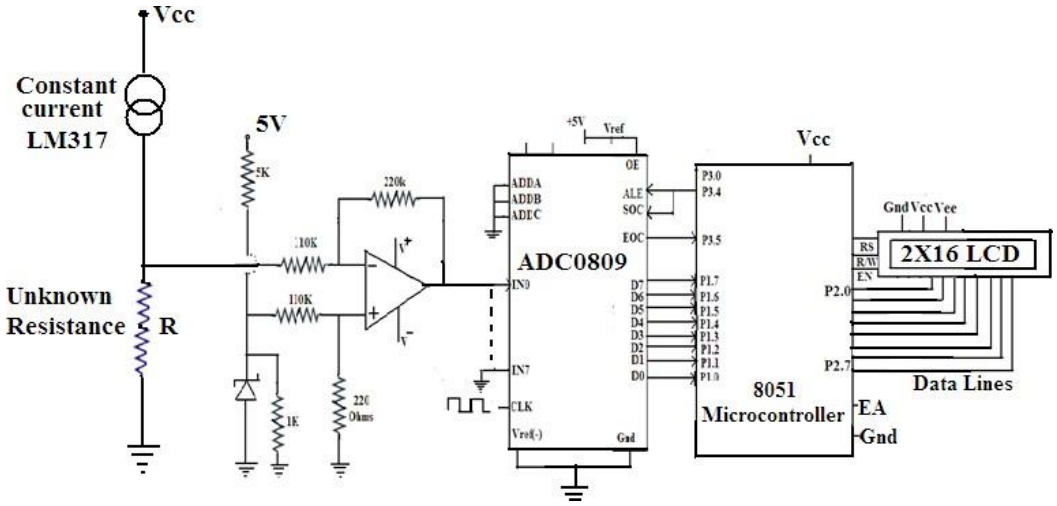
Assembly Language Program :he assembly language program is basically an ADC programAs given below.

Address(Hex)	Label	Mnemonics	Operands	Comments
9000	START	SETB	P3.4	Send ALE & SOC high
		NOP		No operation
		NOP		No operation
		CLR	P3.4	ALE &SOC low
	LOOP1	JNB	P3.5 , LOOP1	Is conversion over ?
		MOV	A,P1	Read digital data(Voltage)
		CALL	CONV	Convert the hex value into Decimal value
		MOV	B,#Resistor Value	Load the resistor value into B register
		DIV	AB	Divide voltage by resistance
		CALL	DISPLAY	Display subroutine to display current on LCD
		SJMP	START	Measure the current continuously

MEASUREMENT OF RESISTANCE - 8051 CONTROLLER

Resistance is measured using the current-voltage relation. i.e using the Ohm's law. In the relationV= IR if the voltage and current are known, the resistance can be determined. In this method a **constant current source i**s used. A resistor is connected in series to the current source .The voltagedrop across the resistor is signal conditioned using the op-amp circuit and this analog signal is then converted into equivalent digital signal by using ADC. The digital signal is fed to the Port1 of the microcontroller. The microcontroller will process the data .Using the relation R=V/I , the resistor value is determined and by developing a suitable program the resistance value is displayedon the LCD. In the circuit LM317 regulator can be used as constant current source.

The interface diagram is shown below.



An assembly language program is developed to find the resistance value. The program is given below.

Address(Hex)	Label	Mnemonics	Operands	Comments
9000	START	SETB	P3.4	Send ALE & SOC high
		NOP		No operation
		NOP		No operation
		CLR	P3.4	ALE &SOC low
	LOOP1	JNB	P3.5 , LOOP1	Is conversion over ?
		MOV	A,P1	Read digital data(Voltage)
		CALL	CONV	Convert the hex value into Decimal value

		MOV	B, # Current value	Load the known value of the current into B register
		DIV	AB	Divide voltage by current
		CALL	DISPLAY	Display subroutine to display resistance value on LCD
		SJMP	START	Measure the resistance continuously