SEIA1504	MICROPROCESSORS, MICROCONTROLLERS AND EMBEDDED SYSTEMS	L	T	P	Credits	Total Marks
		3	0	0	3	100

COURSE OBJECTIVES

- To impart basic knowledge about 8085 microprocessor, 8051 and PIC microcontrollers.
- To introduce about microprocessor interfacing with external systems.
- To provide basics of 8085, 8051 and PIC programming with the knowledge of instructions and interfacing chips.
- To discuss the major components that constitutes an embedded system.

UNIT 1 ARCHITECTURE AND INSTRUCTION SET OF 8085, 8086 11 Hrs.

8085 Architecture and its operation- Instruction set classification-Addressing modes – 8086 Architecture and Instruction set- Basic programs.

UNIT 2 ARCHITECTURE OF 8051 AND INSTRUCTION SET 9 Hrs.

Introduction - Architecture of 8051 - Memory organization - Addressing modes - Instruction set – Assembly Language Programming - Jump, Loop and Call Instructions - Arithmetic and Logic Instructions - Bit Operations.

UNIT 3 PIC MICROCONTROLLER 9 Hrs.

PIC Microcontrollers and Instruction Set: PIC Micro-controllers - overview; features, PIC-18Fxxx architecture, file selection register, Memory organization, Addressing modes, Instruction set, Interrupt handling. PIC-18Fxxx - Reset, low power operations, oscillator connections, I/O ports - serial; parallel, Timers, Interrupts, ADC.

UNIT 4 INTERFACING 9 Hrs.

Basic Interface concepts, Fundamentals of memory interface- memory mapped I/O and I/O mapped I/O, Interrupt and vectored interrupt, Programmable peripheral interface 8255 - Programmable Interval timer 8253 - Programmable interrupt controller 8259 - Programmable DMA controller 8257.

UNIT 5 INTRODUCTION TO EMBEDDED SYSTEM 7 Hrs.

Embedded system- characteristics of embedded system- categories of embedded system- requirements of embedded systems- challenges and design issues of embedded system- trends in embedded system- system integration- hardware and software partition- applications of embedded system - control system and industrial automation-biomedical-data communication system-network information appliances- IVR systems- GPS systems.

Max.45 Hrs.

COURSE OUTCOMES

On completion of the course, student will be able to

CO1 - Recall architecture and operation of 8085, 8086, 8051 and PIC microcontroller.
CO2 - Explain the instruction set of the 8085,8086, 8051 & PIC Microcontroller.
CO3 - Discuss interfacing concepts with Microprocessors.
CO4 - Demonstrate small programs using microprocessors and microcontrollers.
CO5 - Design of embedded systems.
CO6 - Design a control system using 8051, PIC.

TEXT / REFERENCE BOOKS

1. Ramesh Goankar, "Microprocessor architecture programming and applications with 8085 / 8088", 5th Edition, Penram International Publishing, 2002.
2. Mazidi & McKinlay, “The 8051 Microcontroller and Embedded Systems using Assembly and C”, PHI, 2007.
3. MykePredko, “Programming and Customizing the 8051 Micro-controller”, Tata McGraw-Hill edition, 2007.
4. R A Gaonkar, “Fundamentals of Microcontrollers and Applications in Embedded Systems (with the PIC18 Microcontroller Family)”, Penram Publishing India, 2007.
5. Kenneth Ayala ,”The 8051 Microcontroller”, 3rd Edition, Thomson Delmar Learning, 2004.
6. Kenneth J. Ayala, Dhananjay V. Gadre, “The 8051 Microcontroller & Embedded Systems Using Assembly and C”, Cengage Learning India Publication, 2007.
7. Ajay V Deshmukh, “Microcontrollers: Theory and Applications”, Tata McGraw-Hill, 2005.
8. Raj Kamal, “Embedded Systems Architecture, Programming, and Design”. (2/e), Tata McGraw Hill, 2008.

END SEMESTER EXAM QUESTION PAPER PATTERN

Max. Marks : 100 **Exam Duration: 3 Hrs.**

PART A: 10 Questions of 2 marks each-No choice **20 Marks**

PART B: 2 Questions from each unit of internal choice, each carrying 16 marks **80 Marks**

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF ELECTRICAL AND ELECTRONICS ENGINEERING

DEPARTMENT OF ELECTRONICS AND INSRUMENTATION ENGINEERING

UNIT – I – MICROPROCESSORS , MICROCONTROLLERS AND EMBEDDED SYSTEMS– SEIA 1504

UNIT 1 ARCHITECTURE AND INSTRUCTION SET OF 8085, 8086

8085 Architecture and its operation- Instruction set classification-Addressing modes – 8086 Architecture and Instruction set- Basic programs.

1. History of microprocessor:

The invention of the transistor in 1947 was a significant development in the world of technology. It could perform the function of a large component used in a computer in the early years. Shockley, Brattain and Bardeen are credited with this invention and were awarded the Nobel prize for the same. Soon it was found that the function this large component was easily performed by a group of transistors arranged on a single platform. This platform, known as the integrated chip (IC), turned out to be a very crucial achievement and brought along a revolution in the use of computers. A person named Jack Kilby of Texas Instruments was honored with the Nobel Prize for the invention of IC, which laid the foundation on which microprocessors were developed. At the same time, Robert Noyce of Fairchild made a parallel development in IC technology for which he was awarded the patent.

ICs proved beyond doubt that complex functions could be integrated on a single chip with a highly developed speed and storage capacity. Both Fairchild and Texas Instruments began the manufacture of commercial ICs in 1961. Later, complex developments in the IC led to the addition of more complex functions on a single chip. The stage was set for a single controlling circuit for all the computer functions. Finally, Intel corporation's Ted Hoff and Frederico Fagin were credited with the design of the first microprocessor.

The work on this project began with an order from a Japanese calculator company Busicom to Intel, for building some chips for it. Hoff felt that the design could integrate a number of functions on a single chip making it feasible for providing the required functionality. This led to the design of Intel 4004, the world's first microprocessor. The next in line was the 8 bit 8008 microprocessor. It was developed by Intel in 1972 to perform complex functions in harmony with the 4004.

This was the beginning of a new era in computer applications. The use of mainframes and huge computers was scaled down to a much smaller device that was affordable to many. Earlier, their use was limited to large organizations and universities. With the advent of microprocessors, the use of computers trickled down to the common man. The next processor in line was Intel's 8080 with an 8 bit data bus and a 16 bit address bus. This was amongst the most popular microprocessors of all time.

Very soon, the Motorola corporation developed its own 6800 in competition with the Intel's 8080. Fagin left Intel and formed his own firm Zilog. It launched a new microprocessor Z80 in 1980 that was far superior to the previous two versions. Similarly, a break off from Motorola prompted the design of 6502, a derivative of the 6800. Such attempts continued with some modifications in the base structure.

The use of microprocessors was limited to task-based operations specifically required for company projects such as the automobile sector. The concept of a 'personal computer' was still a distant dream for the world and microprocessors were yet to come into personal use. The 16 bit microprocessors started becoming a commercial sell-out in the 1980s with the first popular one being the TMS9900 of Texas Instruments.

Intel developed the 8086 which still serves as the base model for all latest advancements in the microprocessor family. It was largely a complete processor integrating all the required features in it. 68000 by Motorola was one of the first microprocessors to develop the concept of microcoding in its instruction set. They were further developed to 32 bit architectures. Similarly, many players like Zilog, IBM and Apple were successful in getting their own products in the market. However, Intel had a commanding position in the market right through the microprocessorers.

The 1990s saw a large scale application of microprocessors in the personal computer applications developed by the newly formed Apple, IBM and Microsoft corporation. It witnessed a revolution in the use of computers, which by then was a household entity.

This growth was complemented by a highly sophisticated development in the commercial use of microprocessors. In 1993, Intel brought out its 'Pentium Processor' which is one of the most popular processors in use till date. It was followed by a series of excellent processors of the Pentium family, leading into the 21st century. The latest one in commercial use is the Pentium Dual Core technology and the Xeon processor. They have opened up a whole new world of diverse applications. Supercomputers have become common, owing to this amazing development in microprocessors.

1.1 Introduction

A microprocessor is a programmable electronics chip that has computing and decision making capabilities similar to central processing unit of a computer. Any microprocessor-based systems having limited number of resources are called microcomputers. Nowadays, microprocessor can be seen in almost all types of electronics devices like mobile phones, printers, washing machines etc. Microprocessors are also used in advanced applications like radars, satellites and flights. Due to the rapid advancements in electronic industry and large scale integration of devices results in a significant cost reduction and increase application of microprocessors and their derivatives.

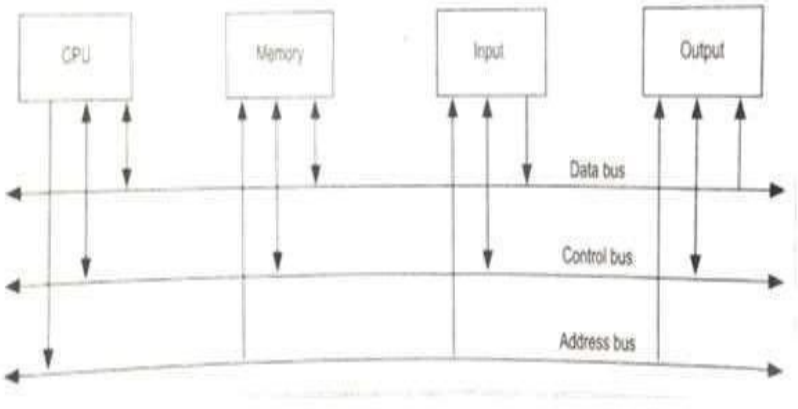


Fig.1.1 Microprocessor-based system

Bit: A bit is a single binary digit.

Word: A word refers to the basic data size or bit size that can be processed by the arithmetic and logic unit of the processor. A 16-bit binary number is called a word in a 16-bit processor.

Bus: A bus is a group of wires/lines that carry similar information.

System Bus: The system bus is a group of wires/lines used for communication between the microprocessor and peripherals.

Memory Word: The number of bits that can be stored in a register or memory element is called a memory word.

Address Bus: It carries the address, which is a unique binary pattern used to identify a memory location or an I/O port. For example, an eight bit address bus has eight lines and thus it can address 28 = 256 different locations. The locations in hexadecimal format can be written as 00H – FFH.Data Bus: The data bus is used to transfer data between memory and processor or between I/O device and processor. For example, an 8-bit processor will generally have an 8-bit data bus and a 16-bit processor will have 16-bit data bus.

Control Bus: The control bus carry control signals, which consists of signals for selection of memory or I/O device from the given address, direction of data transfer and synchronization of data transfer in case of slow devices.A typical microprocessor consists of arithmetic and logic unit (ALU) in association with control unit to process the instruction execution. Almost all the microprocessors are based on the principle of store- program concept. In store-program concept, programs or instructions are sequentially stored in the memory locations that are to be executed. To do any task

using a microprocessor, it is to be programmed by the user. So the programmer must have idea about its internal resources, features and supported instructions. Each microprocessor has a set of instructions, a list which is provided by the microprocessor manufacturer. The instruction set of a microprocessor is provided in two forms: binary machine code and mnemonics.

Microprocessor communicates and operates in binary numbers 0 and 1. The set of instructions in the form of binary patterns is called a machine language and it is difficult for us to understand. Therefore, the binary patterns are given abbreviated names, called mnemonics, which forms the assembly language. The conversion of assembly-level language into binary machine-level language is done by using an application called assembler.

Technology Used:

The semiconductor manufacturing technologies used for chips are:

- Transistor-Transistor Logic (TTL)
- Emitter Coupled Logic (ECL)
- Complementary Metal-Oxide Semiconductor (CMOS)

Classification of Microprocessors:

Based on their specification, application and architecture

microprocessors are classified. Based on size of data bus:

- 4-bit microprocessor
- 8-bit microprocessor
- 16-bit microprocessor
- 32-bit microprocessor

Based on application:

- General-purpose microprocessor- used in general computer system and can be used by programmer for any application. Examples, 8085 to Intel Pentium.
- Microcontroller- microprocessor with built-in memory and ports and can be programmed for any generic

control
application. Example, 8051.

• Special-purpose processors- designed to handle special functions required for an application. Examples, digital signal processors and application-specific integrated circuit (ASIC) chips.

Based on architecture:

- Reduced Instruction Set Computer (RISC) processors
- Complex Instruction Set Computer (CISC) processors

2. 8085 Microprocessor Architecture

The 8085 microprocessor is an 8-bit processor available as a 40-pin IC package and uses +5 V for power. It can run at a maximum frequency of 3 MHz. Its data bus width is 8-bit and address bus width is 16-bit, thus it can address 216 = 64 KB of memory. The internal architecture of 8085 is shown is Fig. 1.2.

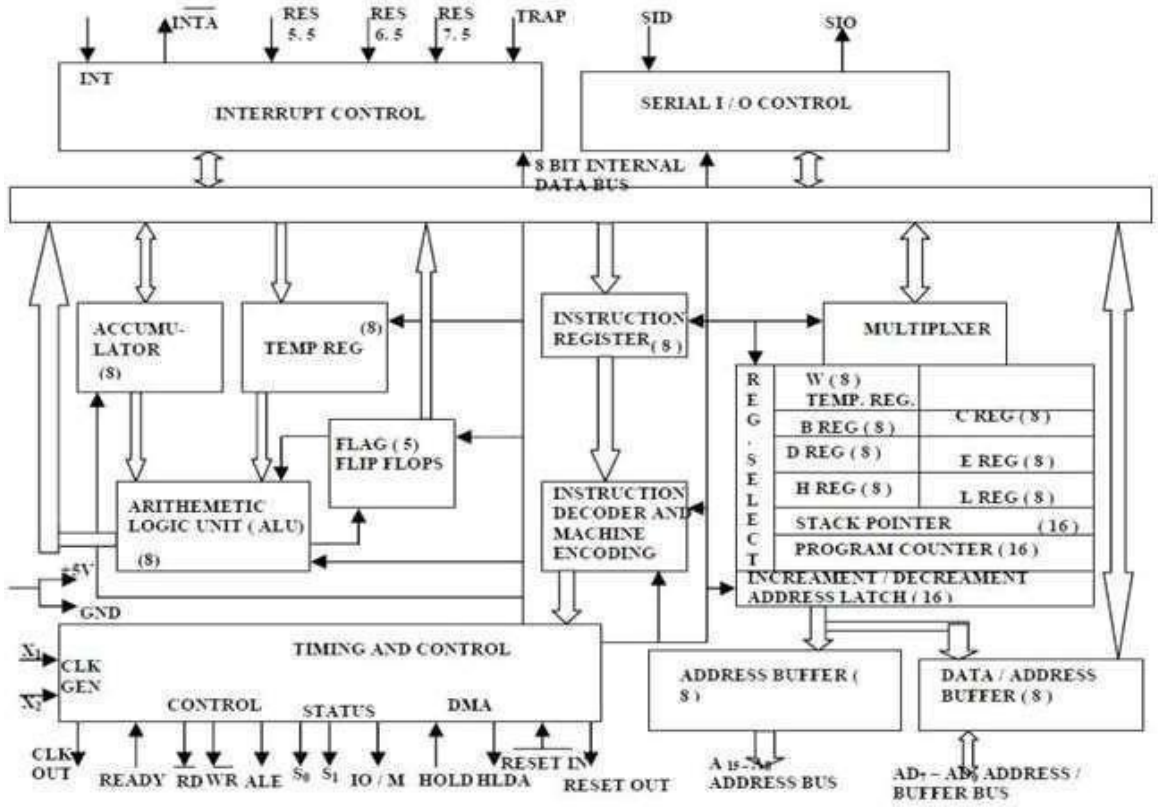


Fig 1.2: 8085 Architecture

Arithmetic and Logic Unit

The ALU performs the actual numerical and logical operations such as Addition (ADD), Subtraction (SUB), AND, OR etc. It uses data from memory and from Accumulator to perform operations. The results of the arithmetic and logical operations are stored in the accumulator.

Registers

The 8085 includes six registers, one accumulator and one flag register, as shown in Fig. 1.3. In addition, it has two 16-bit registers: stack pointer and program counter. They are briefly described as follows.

The 8085 has six general-purpose registers to store 8-bit data; these are identified as B, C, D, E, H and L. they can be combined as register pairs - BC, DE and HL to perform some 16-bit operations. The programmer can use these registers to store or copy data into the register by using data copy instructions.

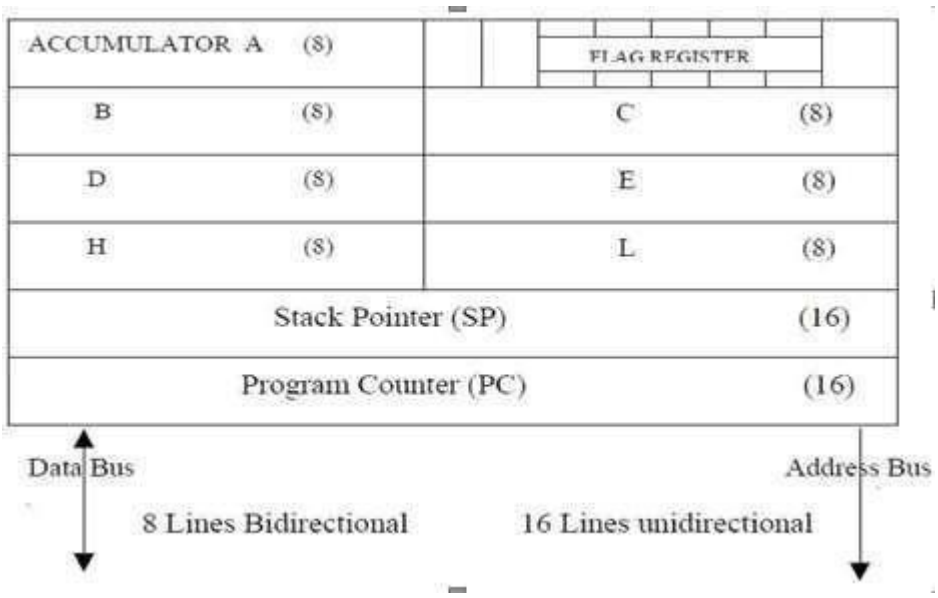


Fig 1.3: Register Organization

Accumulator

The accumulator is an 8-bit register that is a part of ALU. This register is used to store 8-bit data and to perform arithmetic and logical operations. The result of an operation is stored in the accumulator. The accumulator is also identified as register A.

Flag register

The ALU includes five flip-flops, which are set or reset after an operation according to data condition of the result in the accumulator and other registers. They are called Zero (Z), Carry (CY), Sign (S), Parity (P) and Auxiliary Carry (AC) flags. Their bit positions in the flag register are shown in Fig. 4. The microprocessor uses these flags to test data conditions.

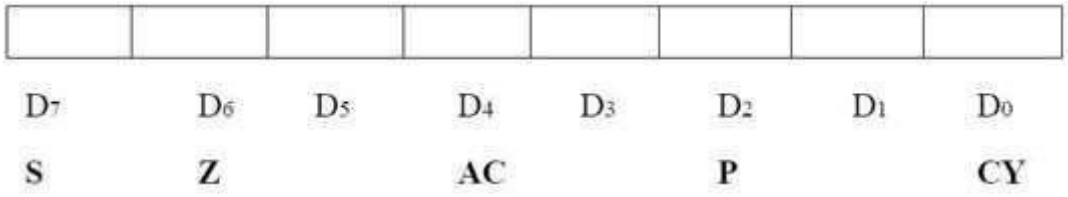


Fig 1.4: PSW

For example, after an addition of two numbers, if the result in the accumulator is larger than 8-bit, the flip-flop uses to indicate a carry by setting CY flag to 1. When an arithmetic operation results in zero, Z flag is set to 1. The S flag is just a copy of the bit D7 of the accumulator. A negative number has a 1 in bit D7 and a positive number has a 0 in 2‘s complement representation. The AC flag is set to 1, when a carry result from bit D3 and passes to bit D4. The P flag is set to 1, when the result in accumulator contains even number of 1s.

Program Counter (PC)

This 16-bit register deals with sequencing the execution of instructions. This register is a memory pointer. The microprocessor uses this register to sequence the execution of the instructions. The function of the program counter is to point to the memory address from which the next byte is to be fetched. When a byte is being fetched, the program counter is automatically incremented by one to point to the next memory location.

Stack Pointer (SP)

The stack pointer is also a 16-bit register, used as a memory pointer. It points to a memory location in R/W memory, called stack. The beginning of the stack is defined by loading 16-bit address in the stack pointer.

Instruction Register/Decoder

It is an 8-bit register that temporarily stores the current instruction of a program. Latest instruction sent here from memory prior to execution. Decoder then takes instruction and decodes or interprets the instruction. Decoded instruction then passed to next stage.

Control Unit

Generates signals on data bus, address bus and control bus within microprocessor to carry out the instruction, which has been decoded. Typical buses and their timing are described as follows:

- Data Bus: Data bus carries data in binary form between microprocessor and other external units such as memory.

It is used to transmit data i.e. information, results of

arithmetic etc between memory and the microprocessor. Data bus is bidirectional in nature. The data bus width of 8085 microprocessor is 8-bit i.e. 28 combination of binary digits and are typically identified as D0 – D7. Thus

size of the data bus determines what arithmetic can be done. If only 8-bit wide then largest number is 11111111 (255 in decimal). Therefore, larger numbers have to be broken down into chunks of 255. This slows
microprocessor.

- Address Bus: The address bus carries addresses and is one way bus from microprocessor to the memory or other devices. 8085 microprocessor contain 16-bit address bus and are generally identified as A0 - A15. Thehigher

order address lines (A8 – A15) are unidirectional and the lower order lines (A0 – A7) are multiplexed (time- shared) with the eight data bits (D0 – D7) and hence, they are bidirectional.

- Control Bus: Control bus are various lines which have specific functions for coordinating and controlling

microprocessor operations. The control bus carries control signals partly unidirectional and partly bidirectional. The following control and status signals are used by 8085 processor:

I. ALE (output): Address Latch Enable is a pulse that is provided when an address appears on the AD0 – AD7 lines, after which it becomes 0.
II. RD (active low output): The Read signal indicates that data are being read from the selected I/O or memory device and that they are available on the data bus.
III. WR (active low output): The Write signal indicates that data on the data bus are to be written into a selected memory or I/O location.
IV. IO/M (output): It is a signal that distinguished between a memory operation and an I/O operation. When IO/M = 0 it is a memory operation and IO/M = 1 it is an I/O operation.
V. S1 and S0 (output): These are status signals used to specify the type of operation being performed; they are listed in Table 1.1

Table 1.1: Status signals and associated operations

S1	S0	States
0	0	Halt
0	1	Write
1	0	Read
1	1	Fetch

The schematic representation of the 8085 bus structure is as shown in Fig. 1.5. The microprocessor performs primarily four operations:

1.Memory Read: Reads data (or instruction)

from memory. 2.Memory Write: Writes data

(or instruction) into memory. 3.I/O Read:

Accepts data from input device.

4.I/O Write: Sends data to output device.

The 8085 processor performs these functions using address bus, data bus and control bus as shown in Fig. 1.5.

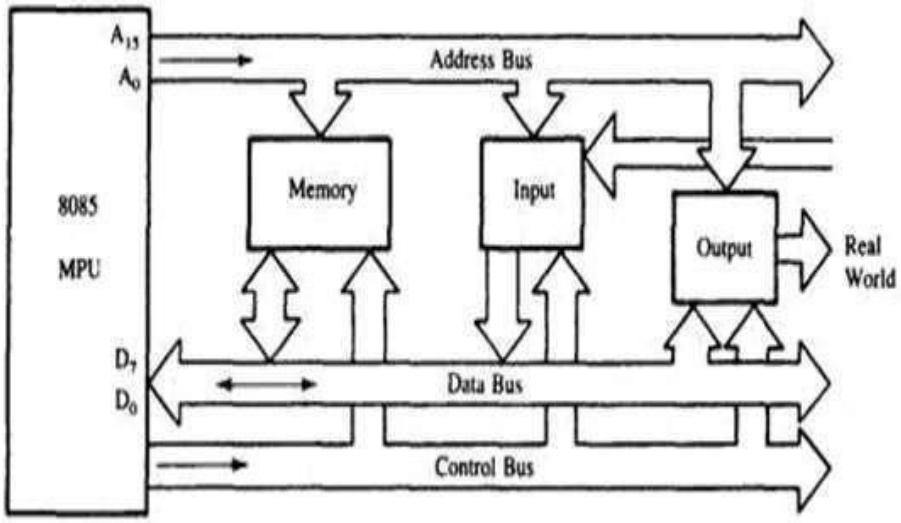


Fig 1.5: 8085 Bus structure

3. 8085 Pin Description

- It is a 8-bit microprocessor
- Manufactured with N-MOS technology
- 40 pin IC package

- It has 16-bit address bus and thus has 216 = 64 KB addressing capability.
- Operate with 3 MHz single-phase clock
- +5 V single power supply

The logic pin layout and signal groups of the 8085nmicroprocessor are shown in Fig. 1.6. All the signals are classified into six groups:

- Address bus
- Data bus
- Control & status signals
- Power supply and frequency signals
- Externally initiated signals
- Serial I/O signals

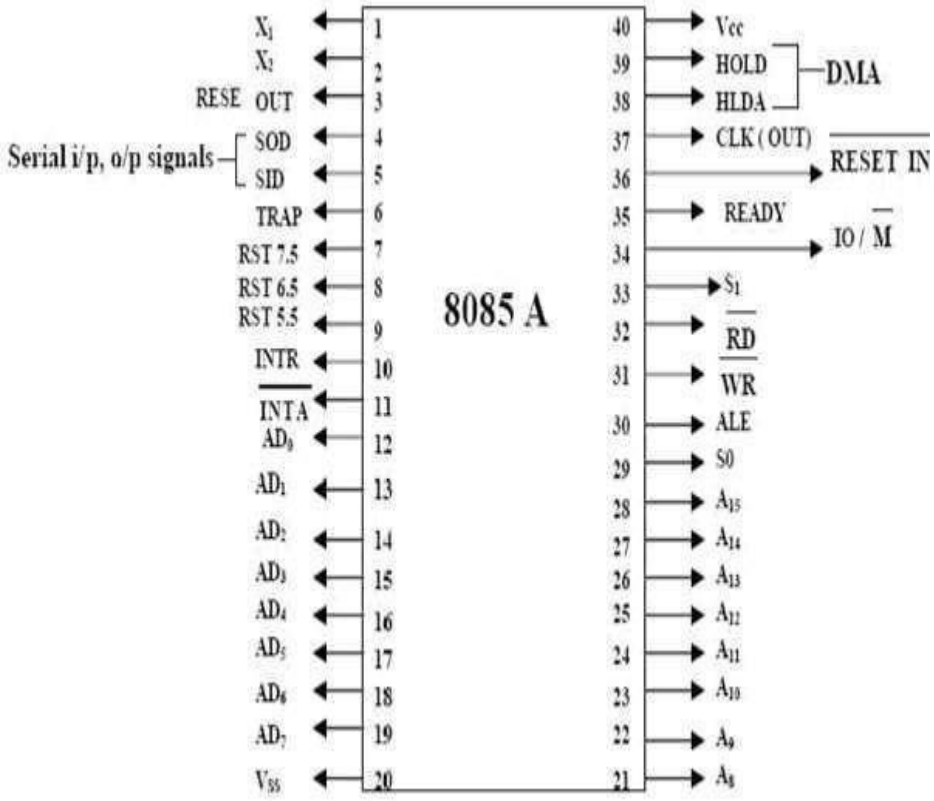


Fig 1.6: 8085-Pin Diagram

Address and Data Buses:

- A8 – A15 (output, 3-state): Most significant eight bits of memory addresses and the eight bits of the I/O addresses. These lines enter into tri-state high impedance state during HOLD and HALT modes.

- AD0 – AD7 (input/output, 3-state): Lower significant bits of memory addresses and the eight bits of the I/O addresses during first clock cycle. Behaves as data bus during third and fourth clock cycle. These lines enter into tri-state high impedance state during HOLD and HALT modes.

Control & Status Signals:

- ALE: Address latch enable
- RD : Read control signal.
- WR :Write control signal
  :
- IO/M , S1 and S0 : Status

signals. Power Supply & Clock Frequency:

- Vcc: +5 V power supply
- Vss: Ground reference
- X1, X2: A crystal having frequency of 6 MHz is connected at these two pins
- CLK: Clock output

Externally Initiated and Interrupt Signals:

- RESET IN : When the signal on this pin is low, the PC is set to 0, the buses are tri-stated and the processor is reset.
- RESET OUT: This signal indicates that the processor is being reset. The signal can be used to reset other
  devices.
- READY: When this signal is low, the processor waits for an integral number of clock cycles until it goes high.
- HOLD: This signal indicates that a peripheral like DMA (direct memory access) controller is requesting the use of address and data bus.
- HLDA: This signal acknowledges the HOLD request.
- INTR: Interrupt request is a general-purpose interrupt.
- INTA : This is used to acknowledge an interrupt.
- RST 7.5, RST 6.5, RST 5,5 – restart interrupt: These are vectored interrupts

and have highest priority than INTR interrupt.

- TRAP: This is a non-maskable interrupt and has the highest priority.

Serial I/O Signals:

- SID: Serial input signal. Bit on this line is loaded to D7 bit of register A using RIM instruction.
- SOD: Serial output signal. Output SOD is set or reset by using SIM instruction.

4. Instruction Set And Execution In 8085

Based on the design of the ALU provides and decoding unit, the microprocessor manufacturer microprocessor. The instruction set for every machine code and instruction set consists of both

mnemonics.

An instruction is a binary pattern designed inside a microprocessor to perform a specific function. The entire group of instructions that a microprocessor supports is called instruction set. Microprocessor instructions can be classified based on the parameters such functionality, length and operand addressing.

Classification based on functionality:

I. Data transfer operations: This group of instructions copies data from source to destination. The content of the source is not altered.

II. Arithmetic operations: Instructions of this group perform operations like addition, subtraction, increment & decrement. One of the data used in arithmetic operation is stored in accumulator and the result is also stored in accumulator.

III. Logical operations: Logical operations include AND, OR, EXOR, NOT. The operations like AND, OR and EXOR uses two operands, one is stored in accumulator and other can be any register or memory location. The result is stored in accumulator. NOT operation requires single operand, which is stored in accumulator.

IV. Branching operations: Instructions in this group can be used to transfer program sequence from one memory location to another either conditionally or unconditionally.

V. Machine control operations: Instruction in this group control execution of other instructions and control operations like interrupt, halt etc.

Classification based on length:

I. One-byte instructions: Instruction having one byte in machine code. Examples are depicted in Table 1.2.

I. Two-byte instructions: Instruction having two byte in machine code. Examples are depicted in Table 1.3

II. Three-byte instructions: Instruction having three byte in machine code. Examples are depicted in Table 1.4.

Table 1.2: Example of one byte instruction

Opcode	Operand	Machine code/Hex code
MOV	A, B	78
ADD	M	86

Table 1.3 Examples of two byte instructions

Opcode	Operand	Machine code/Hex code	Byte description
MVI	A, 7FH	3E	First byte
		7F	Second byte
ADI	0FH	C6	First byte
		0F	Second byte

Table 1.4 Examples of three byte instructions

Opcode	Operand	Machine code/Hex code	Byte description
JMP	9050H	C3	First byte
		50	Second byte
		90	Third byte
LDA	8850H	3A	First byte
		50	Second byte
		88	Third byte

5. Addressing Modes in Instructions:

The process of specifying the data to be operated on by the instruction is called addressing. The various formats for specifying operands are called addressing modes. The 8085 has the following five types of addressing:

1. Immediate addressing

2. Memory direct addressing

3. Register direct addressing

4. Indirect addressing

5. Implicit

addressing Immediate

Addressing:

In this mode, the operand given in the instruction - a byte or word – transfers to the destination register or memory location.

Ex: MVI A, 9AH

- The operand is a part of the instruction.
- The operand is stored in the register mentioned in the instruction.

Memory Direct Addressing:
Memory direct addressing moves a byte or word between a memory location and register. The memory location address is given in the instruction.
Ex: LDA 850FH
This instruction is used to load the content of memory address 850FH in the accumulator. Register Direct Addressing:

Register direct addressing transfer a copy of a byte or word from source register to destination register.

Ex: MOV B, C

It copies the content of register C to register B.

Indirect Addressing:

Indirect addressing transfers a byte or word between a register and a memory

location. Ex: MOV A, M

Here the data is in the memory location pointed to by the contents of HL pair. The data is moved to the accumulator.

Implicit Addressing

In this addressing mode the data itself specifies the data to be

operated upon. Ex: CMA

The instruction complements the content of the accumulator. No specific data or operand is mentioned in the instruction

6.8085 Interrupts

Interrupt Structure:

Interrupt is the mechanism by which the processor is made to transfer control from its current program execution to another program having higher priority. The interrupt signal may be given to the processor by any external peripheral device.

The program or the routine that is executed upon interrupt is called interrupt service routine (ISR). After execution of ISR, the processor must return to the interrupted program. Key features in the interrupt structure of any microprocessor are as follows:

i. Number and types of interrupt signals available.
ii. The address of the memory where the ISR is located for a particular interrupt signal. This address is called interrupt vector address (IVA).
iii. Masking and unmasking feature of the interrupt signals.
iv. Priority among the interrupts.
v. Timing of the interrupt signals.
vi. Handling and storing of information about the interrupt program (status information).

Types of Interrupts:

Interrupts are classified based on their maskability, IVA and source. They are classified as:

i. Vectored and Non-Vectored Interrupts

Vectored interrupts require the IVA to be supplied by the external device that gives the interrupt signal. This technique is vectoring, is implemented in number of ways.

Non-vectored interrupts have fixed IVA for ISRs of

different interrupt signals. ii.Maskable and Non-Maskable

Interrupts

Maskable interrupts are interrupts that can be blocked. Masking can be done by software or hardware means.

Non-maskable interrupts are interrupts that are always recognized; the corresponding ISRs are executed.

iii. Software and Hardware Interrupts

Software interrupts are special instructions, after execution transfer the control to predefined ISR.

Hardware interrupts are signals given to the processor, for recognition as an interrupt and execution of the corresponding ISR.

Interrupt Handling Procedure:

The following sequence of operations takes place when an interrupt signal is recognized:

i. Save the PC content and information about current state (flags,

registers etc) in the stack. ii.Load PC with the beginning address of

an ISR and start to execute it.

iii. Finish ISR when the return instruction is executed.

iv. Return to the point in the interrupted program where execution was interrupted.

Interrupt Sources and Vector Addresses in 8085:

Software Interrupts:

8085 instruction set includes eight software interrupt instructions called Restart (RST) instructions. These are one byte instructions that make the processor execute a subroutine at predefined locations. Instructions and their vector addresses are given in Table 1.6

Table 1.6 Vector address

Instruction	Machine hex code	Interrupt Vector Address
RST 0	C7	0000H
RST 1	CF	0008H
RST 2	D7	0010H
RST 3	DF	0018H
RST 4	E7	0020H
RST 5	EF	0028H
RST 6	F7	0030H
RST 7	FF	0032H

The software interrupts can be treated as CALL instructions with default call locations. The concept of priority does not apply to software interrupts as they are inserted into the

program as instructions by the programmer and executed by the processor when the respective program lines are read.

Hardware Interrupts and Priorities:

8085 have five hardware interrupts – INTR, RST 5.5, RST 6.5, RST 7.5 and TRAP. Their IVA and priorities are given in Table 1.7.

Table 1.7 Hardware interrupts of 8085

Interrupt	Interrupt vector address	Maskable or non-maskable	Edge or level Triggered	priority
TRAP	0024H	Non-makable	Level	1
RST 7.5	003CH	Maskable	Rising edge	2
RST 6.5	0034H	Maskable	Level	3
RST 5.5	002CH	Maskable	Level	4
INTR	Decided by hardware	Maskable	Level	5

Addressing Modes

Implied - the data value/data address is implicitly associated with the instruction.

Register - references the data in a register or in a register pair.

Immediate - the data is provided in the instruction.

Direct - the instruction operand specifies the memory address where data is located.

Register indirect - instruction specifies a register containing an address, where data is located. This addressing mode works with SI, DI, BX and BP registers.

Based :- 8-bit or 16-bit instruction operand is added to the contents of a base register (BXor BP), the resulting value is a pointer to location where data resides.

Indexed :- 8-bit or 16-bit instruction operand is added to the contents of an index register(SI or DI), the resulting value is a pointer to location where data resides.

Based Indexed :- the contents of a base register (BX or BP) is added to the contents of anindex register (SI or DI), the resulting value is a pointer to location where data resides.

Based Indexed with displacement :- 8-bit or 16-bit instruction operand is added to the contents of a base register (BX or BP) and index register (SI or DI), the resulting value is a pointer to location where data resides.

Interrupts

The processor has the following interrupts:

INTR is a maskable hardware interrupt. The interrupt can be enabled/disabled using STI/CLI instructions or using more complicated method of updating the FLAGS register with the help of the POPF instruction.

When an interrupt occurs, the processor stores FLAGS register into stack, disables further interrupts, fetches from the bus one byte representing interrupt type, and jumps to interrupt processing routine address of which is stored in location 4 * <interrupt type>. Interrupt processing routine should return with the IRET instruction.

NMI is a non-maskable interrupt. Interrupt is processed in the same way as the INTR interrupt. Interrupt type of the NMI is 2, i.e. the address of the NMI processing routine is stored in location 0008h. This interrupt has higher priority then the maskable interrupt.

Software interrupts can be caused by:

INT instruction - breakpoint interrupt. This is a type 3 interrupt.
INT <interrupt number> instruction - any one interrupt from available 256 interrupts.INTO instruction - interrupt on overflow
Single-step interrupt - generated if the TF flag is set. This is a type 1 interrupt. When the CPU processes this interrupt it clears TF flag before calling the interrupt processing routine.

Processor exceptions: Divide Error (Type 0),

Unused Opcode (type 6) and Escape opcode

(type 7).

Software interrupt processing is the same as for the hardware interrupts.

The figure below shows the 256 interrupt vectors arranged in the interrupt vector table inthe memory.

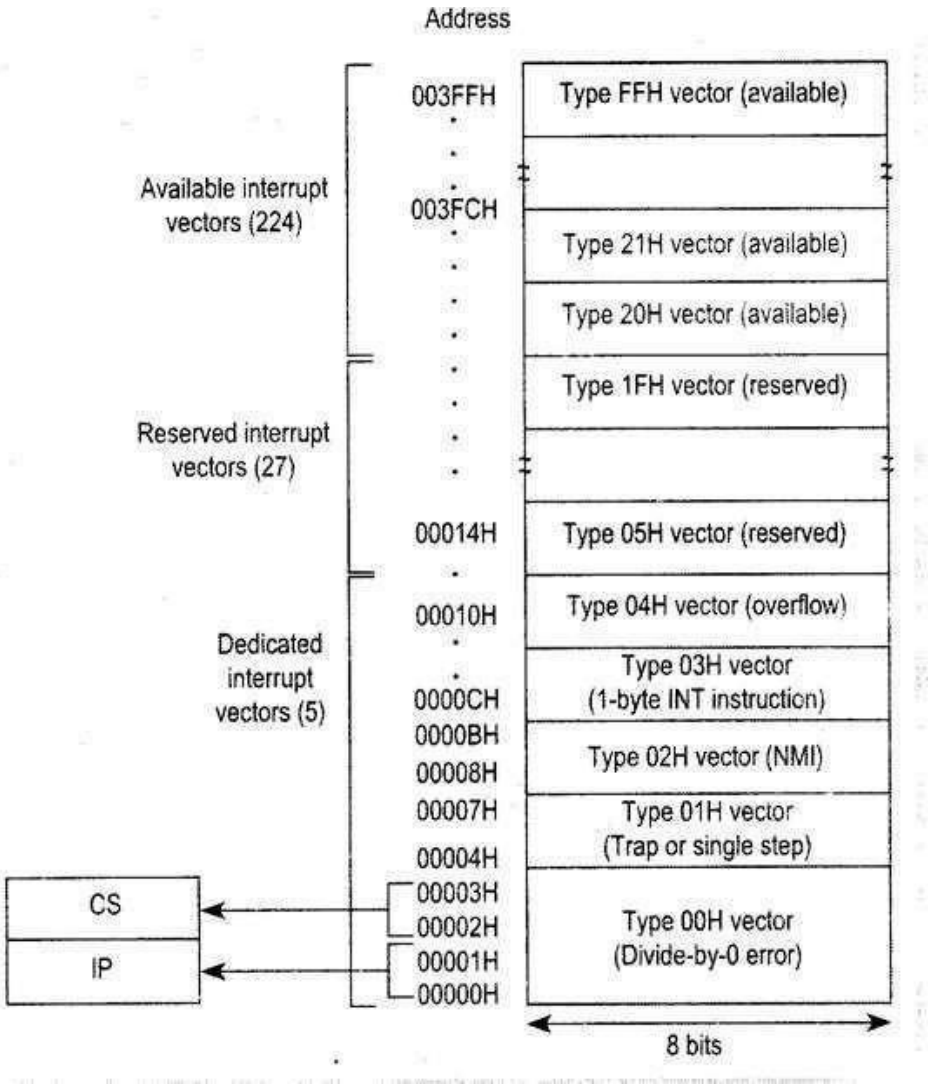


Fig 1.7 Interrupt Vector Table in the 8086

Minimum Mode Interface

When the Minimum mode operation is selected, the 8086 provides all control signals needed to implement the memory and I/O interface. The minimum mode signal can be divided into the following basic groups : address/data bus, status, control, interrupt and DMA.

Address/Data Bus : these lines servetwo functions. As an address bus is 20 bits long and consists of signal lines A0 through A19. A19 represents the MSB and A0 LSB. A 20bit address gives the 8086 a 1Mbyte memory address space. More over it has an independent I/O address space which is 64K bytes in length.

The 16 data bus lines D0 through D15 are actually multiplexed with address lines A0

through A15 respectively. By multiplexed we mean that the bus work as an address bus during first machine cycle and as a data bus during next machine cycles. D15 is the MSB and D0 LSB. When acting as a data bus, they carry read/write data for memory, input/output data for I/O devices, and interrupt type codes froman interrupt controller.

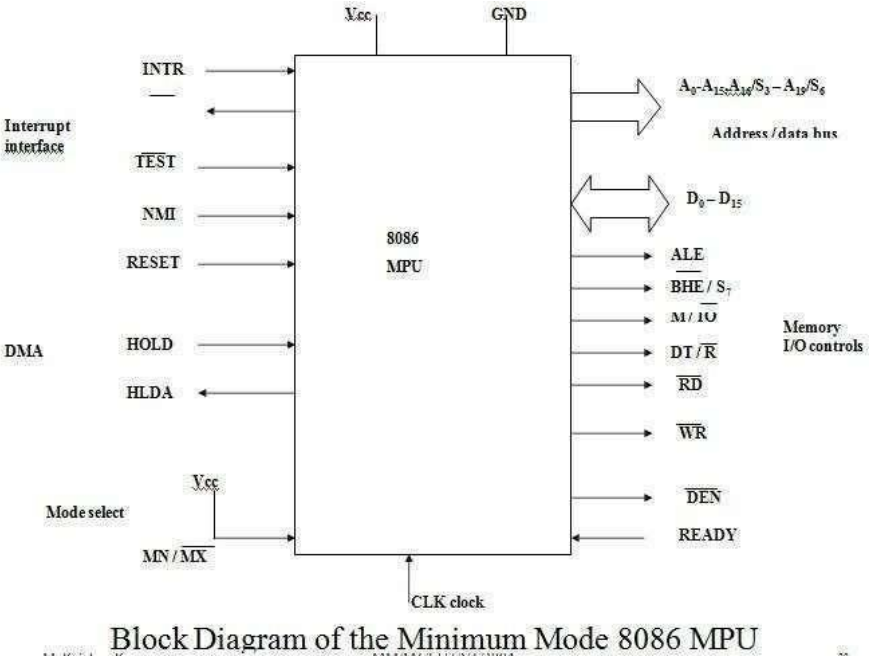


Fig 1.8: Block diagram of Minimum mode

Status signal : The four most significant address lines A19 through A16 are also multiplexed but in this case with status signals S6 through S3. These status bits are output on the bus at the same time that data are transferred over the other bus lines. Bit S4 and S3 together from a 2 bit binary code that identifies which of the 8086 internal segment registersare used to generate the physical address that was output on the address bus during the current bus cycle. Code S4S3 = 00 identifies a register known as *extra segment register* as the source of the segment address.

S_4	S_3	Segment Register
0	0	Extra
0	1	Stack
1	0	Code / none
1	1	Data

Fig 1.9:Memory segment status code

Status line S5 reflects the status of another internal characteristic of the 8086. It is the logic level of the internal enable flag. The last status bit S6 is always at the logic 0 level.

Control Signals : The control signals are provided to support the 8086 memory I/O interfaces. They control functions such as when the bus is to carry a valid address in which direction data are to be transferred over the bus, when valid write data are on the bus and when to put read data on the system bus.

ALE is a pulse to logic 1 that signals external circuitry when a valid address word is on the bus. This address must be latched in external circuitry on the 1-to-0 edge of the pulse at ALE.

Another control signal that is produced during the bus cycle is BHE bank high enable. Logic 0 on this used as a memory enable signal for the most significant byte half of the data bus D8 through D1. These lines also serves a second function, which is as the S7 status line.

Using the M/IO and DT/R lines, the 8086 signals which type of bus cycle is in progress and in which direction data are to be transferred over the bus.

The logic level of M/IO tells external circuitry whether a memory or I/O transfer istaking place over the bus. Logic 1 at this output signals a memory operation and logic 0 an

I/O operation.

The direction of data transfer over the bus is 74ignallin by the logic level output at DT/R. When this line is logic 1 during the data transfer part of a bus cycle, the bus is in the transmit mode. Therefore, data are either written into memory or output to an I/O device.

On the other hand, logic 0 at DT/R signals that the bus is in the receive mode. This corresponds to reading data from memory or input of data from an input port.

The signal read RD and write WR indicates that a read bus cycle or a write bus cycle is in progress. The 8086 switches WR to logic 0 to signal external device that valid write or output data are on the bus.

On the other hand, RD indicates that the 8086 is performing a read of data of the bus. During read operations, one other control signal is also supplied. This is DEN

(data

enable) and it signals external devices when they should put data on the bus.

There is one other control signal that is involved with the memory and I/O interface. This isthe READY signal.

READY signal is used to insert wait states into the bus cycle such that it is extended by a number of clock periods. This signal is provided by an external clock generator device and can be supplied by the memory or I/O sub- system to signal the 8086 when they are ready topermit the data transfer to be completed.

Maximum Mode Interface (cont..)

Status Inputs			CPU Cycles	8288 Command
$\overline{S}_2$	$\overline{S}_1$	$\overline{S}_0$		
0	0	0	Interrupt Acknowledge	INTA
0	0	1	Read I/O Port	IORC
0	1	0	Write I/O Port	IOWC, AIOWC
0	1	1	Halt	None
1	0	0	Instruction Fetch	MRDC
1	0	1	Read Memory	MRDC
1	1	0	Write Memory	MWTC, AMWC
1	1	1	Passive	None

Bus Status Codes

Fig 1.10: Maximum mode

Maximum Mode Interface

When the 8086 is set for the maximum-mode configuration, it provides signals for implementing a multiprocessor / coprocessor systemenvironment. By multiprocessor environment we mean that one microprocessor exists in the system and that each processor is executing its own program. Usually in this type of system environment, there are some system resources that are common to all processors. They are called as *global resources*.

There are also other resources that are assigned to specific processors. These are known as *local* or *private resources*. Coprocessor also means that there is a second processor in the system. In this two processor does not access the bus at the same time. One passes the control of the system bus to the other and then may suspend its operation. In the maximum-mode 8086 system, facilities are provided for implementing allocation of global resources and passing bus control to other microprocessor or coprocessor.

8288 Bus Controller – Bus Command and Control Signals: 8086 does not directly provide all the signals that are required to control the memory, I/O and interrupt interfaces. Specially the WR, M/IO, DT/R, DEN, ALE and INTA, signals are no longer produced by the 8086. Instead it outputs three status signals S0, S1, S2 prior to the initiation of each bus cycle. This 3- bit bus status code identifies which type of bus cycle is to follow. S2S1S0 are input to the external bus controller device, the bus controller generates the appropriately timed command and control signals. The 8288 produces one or two of these eight command signals for each b us cycles. For instance, when the 8086

outputs the code S2S1S0 equals 001, it indicates that an *I/O read cycle* is to be performed. In the code 111 is output by the 8086, it is 7 signalling that no bus

activity is to take place.

The control outputs produced by the 8288 are DEN, DT/R and ALE. These 3 signals provide the same functions as those described for the minimum system mode. This set of bus commands and control signals is compatible with the Multibus and industry standard for interfacing microprocessor systems.

This device permits processors to reside on the system bus. It does this by implementing the Multibus arbitration protocol in an 8086-based system. Addition of the 8288 bus controller and 8289 bus arbiter frees a number of the 8086 pins for use to produce control signals that are needed to support multiple processors. Bus priority lock (LOCK) is one of these signals.It is input to the bus arbiter together with status signals S0 through S2.

Queue Status Signals: Two new signals that are produced by the 8086 in the maximum-mode system are queue status outputs QS0 and QS1. Together they form a 2-bit queue status code, QS1QS0. Following table shows the four different queue status.

Table 1.8: Queue status code

QS_1	QS_0	Queue Status
0 (low)	0	No Operation. During the last clock cycle, nothing was taken from the queue.
0	1	First Byte. The byte taken from the queue was the first byte of the instruction.
1 (high)	0	Queue Empty. The queue has been reinitialized as a result of the execution of a transfer instruction.
1	1	Subsequent Byte. The byte taken from the queue was a subsequent byte of the instruction.

Queue status codes

AX - the Accumulator BX - the Base Register CX - the Count Register DX - the Data Register Normally used for storing temporaryresults. Each of the registers is 16 bits wide (AX, BX, CX, DX). Can be accessed as either 16 or 8 bits AX, AH, AL

AX-Accumulator Register. Preferred register to use in arithmetic, logic and data transfer instructions because it generates the shortest Machine Language Code. Must be used in multiplication and division operations.Must also be used in I/O operations.

BX-Base Register.Also serves as an address register

CX- Count register. Used as a loop counter. Used in shift and rotate operations

DX- Data register. Used in multiplication and division. Also used in I/O

operations

Pointer and Index Registers

SP	Stack Pointer
BP	Base Pointer
SI	Source Index
DI	Destination Index
IP	Instruction Pointer

Fig 1.11 Pointers and index registers

- All 16 bits wide, L/H bytes are notaccessible. Used as memory pointers
- Example: MOV AH,[SI]
- *Move the byte stored in memory location whose address is contained in register SIto register AH.*

IP is not under direct control of theprogrammer

The Stack

The stack is used for temporary storage of information such as data or addresses. When a CALL is executed, the 8086 automatically PUSH the current value of CS and IP onto the stack. Other registers can also be pushed. Before return from the subroutine, POP instructions can be used to pop values back from the stack into the corresponding registers.

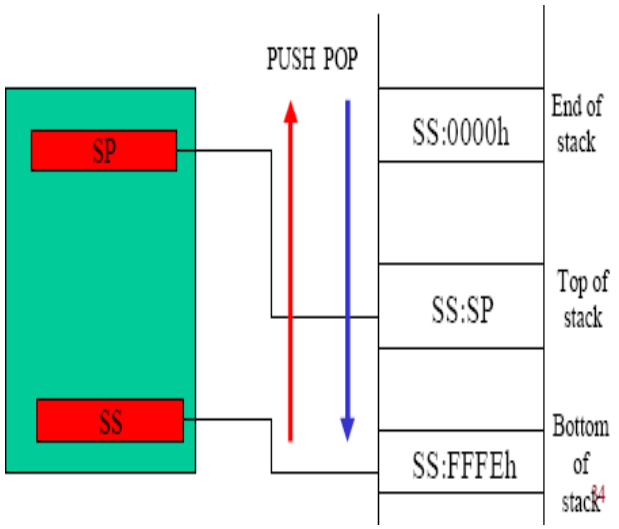


Fig 1.12 stack operation

Test signals in 8086

TEST is an input pin and is only used by the wait instruction .the 8086 enter a wait state after execution of the wait instruction until a low is Seen on the test pin. Used in conjunction with the WAIT instruction in multiprocessing environments. This is input from the 8087 coprocessor. During execution of a wait instruction, the CPU checks this signal. If it is low, execution of the signal will continue; if not, it will stop executing.

Coprocessor Execution

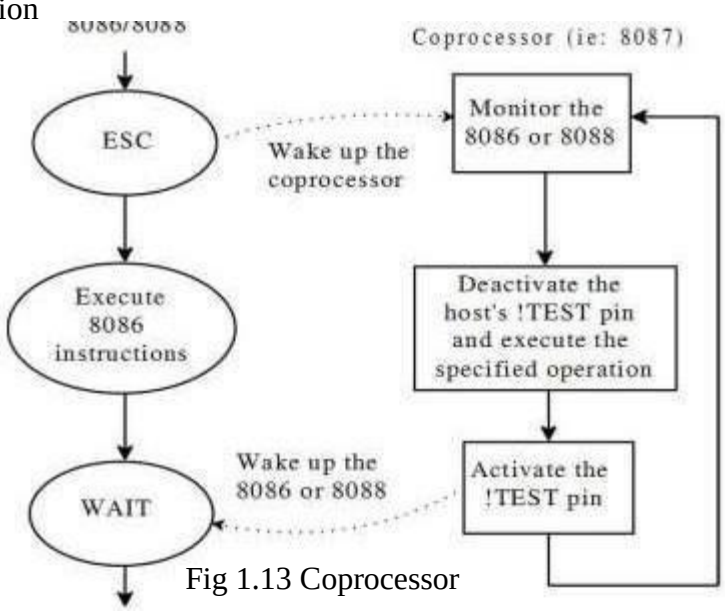


Fig 1.13 Coprocessor

Multiprocessor configuration

High system throughput can be achieved by having more than one CPU. The system can be expanded in modular form. Each bus master module is an independent unit and normally resides on a separate PC board. One can be added or removed without affecting the others in the system. A failure in one module normally does not affect the breakdown of the entire system and the faulty module can be easily detected and replaced. Each bus master has its own local bus to access dedicated memory or IO devices. So a greater degree of parallel processing can be achieved.

8.8085ROPROCESSOR INSTRUCYIONS

8.1 Instruction Set of 8085

- An instruction is a binary pattern designed inside a microprocessor to perform a specific function.
- The entire group of instructions that a microprocessor supports is called Instruction Set.
- 8085 has 246 instructions.
- Each instruction is represented by an 8-bit binary value.
- These 8-bits of binary value is called Op-Code or Instruction Byte.

Classification of Instruction Set

- Data Transfer Instruction
- Arithmetic Instructions
- Logical Instructions
- Branching Instructions
- Control Instructions

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.

Arithmetic Instructions • These instructions perform the operations like: • Addition • Subtract • Increment • Decrement

Logical Instructions • These instructions perform logical operations on data stored in registers, memory and status flags. • The logical operations are: • AND • OR • XOR • Rotate • Compare • Complement

Branching Instructions • The branching instruction alter the normal sequential flow. • These instructions alter either unconditionally or conditionally

Control Instructions • The control instructions control the operation of microprocessor.

DATA TRANSFER INSTRUCTIONS

Copy of data

- MOV Moves data from register to register / memory
- MVI Moves immediate data to register /

memory Load Instructions

- LDA Load accumulator direct
- LDAX Load accumulatorindirect
- LHLD Load H&L registersdirect
- LXI Load register pair

immediate Store Instructions

- STA Store accumulator direct
- SPHL Copy H&L registers to stack pointer.
- STAX Store accumulator indirect

Opcode	Operand	Meaning	Explanation
MOV	Rd, Sc M, Sc Dt, M	Copy from the source (Sc) to the destination(Dt)	This instruction copies the contents of the source register into the destination register without any alteration. Example − MOV A, L
MVI	Rd, data M, data	Move immediate 8-bit	The 8-bit data is stored in the destination register or memory. Example − MVI H, 55H
LDA	16-bit address	Load the accumulator	The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator. Example − LDA 2034H
LDAX	B/D Reg. pair	Load the accumulator indirect	The contents of the designated register pair point to a memory location. This instruction copies the contents of that memory location into the accumulator. Example − LDAX B
LXI	Reg. pair, 16-bit data	Load the register	The instruction loads 16-bit data in the register pair designated in the register or the memory. Example − LXI H, 3225H
LHLD	16-bit address	Load H and L registers direct	The instruction copies the contents of the memory location pointed out by the address into register L and copies the contents of the next memory location into register H.

			Example − LHLD 3225H
STA	16-bit address	16-bit address	The contents of the accumulator are copied into the memory location specified by the operand. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address. Example − STA 3257H
STAX	16-bit address	Store the accumulator indirect	The contents of the accumulator are copied into the memory location specified by the contents of the operand. Example − STAX D
SHLD	16-bit address	Store H and L registers direct	The contents of register L are stored in the memory location specified by the 16-bit address in the operand and the contents of H register are stored into the next memory location by incrementing the operand. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address. Example − SHLD 3225H
XCHG	None	Exchange H and L with D and E	The contents of register H are exchanged with the contents of register D, and the contents of register L are exchanged with the contents of register E. Example − XCHG

SPHL	None	Copy H and L registers to the stack pointer	The instruction loads the contents of the H and L registers into the stack pointer register. The contents of the H register provide the high-order address and the contents of the L register provide the low-order address. Example − SPHL
XTH L	None	Exchange H and L with top of stack	The contents of the L register are exchanged with the stack location pointed out by the contents of the stack pointer register. The contents of the H register are exchanged with the next stack location (SP+1). Example − XTHL
PUSH	Reg. pair	Push the register pair onto the stack	The contents of the register pair designated in the operand are copied onto the stack in the following sequence. The stack pointer register is decremented and the contents of the high order register (B, D, H, A) are copied into that location. The stack pointer register is decremented again and the contents of the low-order register (C, E, L, flags) are copied to that location. Example − PUSH PSW
POP	Reg. pair	Pop off stack to the register pair	The contents of the memory location pointed out by the stack pointer register are copied to the low-order register (C, E, L, status flags) of the operand.

			The stack pointer is incremented by 1 and the contents of that memory location are copied to the high- order register (B, D,H, A) of the operand. The stack pointer register is again incremented by 1. Example – POP D
OUT	8-bit port address	Output the data from the accumulator to a port with 8bit address	The contents of the accumulator are copied into the I/O port specified by the operand. Example − OUT 12H
IN	8-bit port address	Input data to accumulator from a port with 8-bit address	The contents of the input port designated in the operand are read and loaded into the accumulator. Example – IN 55H

Store accumulator direct

STA 16-bit address

The contents of the accumulator are copied into the memory location specified by the operand. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address.
Example: STA 4350 or STA XYZ

Store accumulator indirect

STAX Reg. pair

The contents of the accumulator are copied into the memory location specified by the contents of the operand (register pair). The contents of the accumulator are not altered.
Example: STAX B

Store H and L registers direct

SHLD 16-bit address

The contents of register L are stored into the memory location specified by the 16-bit address in the operand and the contents of H register are stored into the next memory location by incrementing the operand. The contents of registers HL are not altered. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address.
Example: SHLD 2470

Exchange H and L with D and E

XCHG none

The contents of register H are exchanged with the contents of register D, and the contents of register L are exchanged with the contents of register E.
Example: XCHG

Copy H and L registers to the stack pointer

SPHL none

The instruction loads the contents of the H and L registers into the stack pointer register, the contents of the H register provide the high-order address and the contents of the L register provide the low-order address. The contents of the H and L registers are not altered.
Example: SPHL

Exchange H and L with top of stack

XTHL none

The contents of the L register are exchanged with the stack location pointed out by the contents of the stack pointer register. The contents of the H register are exchanged with the next stack location (SP+1); however, the contents of the stack pointer register are not altered.
Example: XTHL

Push register pair onto stack

PUSH	Reg. pair	The contents of the register pair designated in the operand are copied onto the stack in the following sequence. The stack pointer register is decremented and the contents of the high-order register (B, D, H, A) are copied into that location. The stack pointer register is decremented again and the contents of the low-order register (C, E, L, flags) are copied to that location. Example: PUSH B or PUSH A

Pop off stack to register pair

POP	Reg. pair	The contents of the memory location pointed out by the stack pointer register are copied to the low-order register (C, E, L, status flags) of the operand. The stack pointer is incremented by 1 and the contents of that memory location are copied to the high-order register (B, D, H, A) of the operand. The stack pointer register is again incremented by 1. Example: POP H or POP A

Output data from accumulator to a port with 8-bit address

OUT	8-bit port address	The contents of the accumulator are copied into the I/O port specified by the operand. Example: OUT 87

Input data to accumulator from a port with 8-bit address

IN	8-bit port address	The contents of the input port designated in the operand are read and loaded into the accumulator. Example: IN 82

Arithmetic Instructions:

Opcode	Operand	Description
Add register or memory to accumulator		
ADD	R M	The contents of the operand (register or memory) are added to the contents of the accumulator and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the addition. Example: ADD B or ADD M
Add register to accumulator with carry		
ADC	R M	The contents of the operand (register or memory) and the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the addition. Example: ADC B or ADC M
Add immediate to accumulator		
ADI	8-bit data	The 8-bit data (operand) is added to the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the addition. Example: ADI 45
Add immediate to accumulator with carry		
ACI	8-bit data	The 8-bit data (operand) and the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the addition. Example: ACI 45
Add register pair to H and L registers		
DAD	Reg. pair	The 16-bit contents of the specified register pair are added to the contents of the HL register and the sum is stored in the HL register. The contents of the source register pair are not altered. If the result is larger than 16 bits, the CY flag is set. No other flags are affected. Example: DAD H

Subtract register or memory from accumulator

SUB	R M	The contents of the operand (register or memory) are subtracted from the contents of the accumulator, and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the subtraction. Example: SUB B or SUB M

Subtract source and borrow from accumulator

SBB	R M	The contents of the operand (register or memory) and the Borrow flag are subtracted from the contents of the accumulator and the result is placed in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the subtraction. Example: SBB B or SBB M

Subtract immediate from accumulator

SUI	8-bit data	The 8-bit data (operand) is subtracted from the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the subtraction. Example: SUI 45

Subtract immediate from accumulator with borrow

SBI	8-bit data	The 8-bit data (operand) and the Borrow flag are subtracted from the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the subtracion. Example: SBI 45

Increment register or memory by 1

INR	R M	The contents of the designated register or memory) are incremented by 1 and the result is stored in the same place. If the operand is a memory location, its location is specified by the contents of the HL registers. Example: INR B or INR M

Increment register pair by 1

INX	R	The contents of the designated register pair are incremented by 1 and the result is stored in the same place. Example: INX H

Decrement register or memory by 1

DCR R
M

The contents of the designated register or memory are decremented by 1 and the result is stored in the same place. If the operand is a memory location, its location is specified by the contents of the HL registers.
Example: DCR B or DCR M

Decrement register pair by 1

DCX R

The contents of the designated register pair are decremented by 1 and the result is stored in the same place.
Example: DCX H

Decimal adjust accumulator

DAA none

The contents of the accumulator are changed from a binary value to two 4-bit binary coded decimal (BCD) digits. This is the only instruction that uses the auxiliary flag to perform the binary to BCD conversion, and the conversion procedure is described below. S, Z, AC, P, CY flags are altered to reflect the results of the operation.

If the value of the low-order 4-bits in the accumulator is greater than 9 or if AC flag is set, the instruction adds 6 to the low-order four bits.

If the value of the high-order 4-bits in the accumulator is greater than 9 or if the Carry flag is set, the instruction adds 6 to the high-order four bits.

Example: DAA

BRANCHING INSTRUCTIONS

Opcode Operand Description

Jump unconditionally

JMP 16-bit address

The program sequence is transferred to the memory location specified by the 16-bit address given in the operand.
Example: JMP 2034 or JMP XYZ

Jump conditionally

Operand: 16-bit address

The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW as described below.
Example: JZ 2034 or JZ XYZ

Opcode	Description	Flag Status
JC	Jump on Carry	CY = 1
JNC	Jump on no Carry	CY = 0
JP	Jump on positive	S = 0
JM	Jump on minus	S = 1
JZ	Jump on zero	Z = 1
JNZ	Jump on no zero	Z = 0
JPE	Jump on parity even	P = 1
JPO	Jump on parity odd	P = 0

Unconditional subroutine call

CALL 16-bit address

The program sequence is transferred to the memory location specified by the 16-bit address given in the operand. Before the transfer, the address of the next instruction after CALL (the contents of the program counter) is pushed onto the stack.
Example: CALL 2034 or CALL XYZ

Call conditionally

Operand: 16-bit address

The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW as described below. Before the transfer, the address of the next instruction after the call (the contents of the program counter) is pushed onto the stack.
Example: CZ 2034 or CZ XYZ

Opcode	Description	Flag Status
CC	Call on Carry	CY = 1
CNC	Call on no Carry	CY = 0
CP	Call on positive	S = 0
CM	Call on minus	S = 1
CZ	Call on zero	Z = 1
CNZ	Call on no zero	Z = 0
CPE	Call on parity even	P = 1
CPO	Call on parity odd	P = 0

Return from subroutine unconditionally

RET none

The program sequence is transferred from the subroutine to the calling program. The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address.
Example: RET

Return from subroutine conditionally

Operand: none

The program sequence is transferred from the subroutine to the calling program based on the specified flag of the PSW as described below. The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address.
Example: RZ

Opcode	Description	Flag Status
RC	Return on Carry	CY = 1
RNC	Return on no Carry	CY = 0
RP	Return on positive	S = 0
RM	Return on minus	S = 1
RZ	Return on zero	Z = 1
RNZ	Return on no zero	Z = 0
RPE	Return on parity even	P = 1
RPO	Return on parity odd	P = 0

Load program counter with HL contents

PCHL none

The contents of registers H and L are copied into the program counter. The contents of H are placed as the high-order byte and the contents of L as the low-order byte.
Example: PCHL

Restart

RST 0-7

The RST instruction is equivalent to a 1-byte call instruction to one of eight memory locations depending upon the number. The instructions are generally used in conjunction with interrupts and inserted using external hardware. However these can be used as software instructions in a program to transfer program execution to one of the eight locations. The addresses are:

Instruction	Restart Address
RST 0	0000H
RST 1	0008H
RST 2	0010H
RST 3	0018H
RST 4	0020H
RST 5	0028H
RST 6	0030H
RST 7	0038H

The 8085 has four additional interrupts and these interrupts generate RST instructions internally and thus do not require any external hardware. These instructions and their Restart addresses are:

Interrupt	Restart Address
TRAP	0024H
RST 5.5	002CH
RST 6.5	0034H
RST 7.5	003CH

LOGICAL INSTRUCTIONS

Opcode	Operand	Description

Compare register or memory with accumulator

CMP R
M

The contents of the operand (register or memory) are compared with the contents of the accumulator. Both contents are preserved . The result of the comparison is shown by setting the flags of the PSW as follows:
if (A) < (reg/mem): carry flag is set, s=1
if (A) = (reg/mem): zero flag is set, s=0
if (A) > (reg/mem): carry and zero flags are reset, s=0
Example: CMP B or CMP M

Compare immediate with accumulator

CPI 8-bit data

The second byte (8-bit data) is compared with the contents of the accumulator. The values being compared remain unchanged. The result of the comparison is shown by setting the flags of the PSW as follows:
if (A) < data: carry flag is set, s=1
if (A) = data: zero flag is set, s=0
if (A) > data: carry and zero flags are reset, s=0
Example: CPI 89

Logical AND register or memory with accumulator

ANA R
M

The contents of the accumulator are logically ANDed with the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY is reset. AC is set.
Example: ANA B or ANA M

Logical AND immediate with accumulator

ANI 8-bit data

The contents of the accumulator are logically ANDed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY is reset. AC is set.
Example: ANI 86

Exclusive OR register or memory with accumulator

XRA	R M	The contents of the accumulator are Exclusive ORed with the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY and AC are reset. Example: XRA B or XRA M

Exclusive OR immediate with accumulator

XRI	8-bit data	The contents of the accumulator are Exclusive ORed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY and AC are reset. Example: XRI 86

Logical OR register or memory with accumulaotr

ORA	R M	The contents of the accumulator are logically ORed with the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY and AC are reset. Example: ORA B or ORA M

Logical OR immediate with accumulator

ORI	8-bit data	The contents of the accumulator are logically ORed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY and AC are reset. Example: ORI 86

Rotate accumulator left

RLC	none	Each binary bit of the accumulator is rotated left by one position. Bit D_7 is placed in the position of D_0 as well as in the Carry flag. CY is modified according to bit D_7. S, Z, P, AC are not affected. Example: RLC

Rotate accumulator right

RRC	none	Each binary bit of the accumulator is rotated right by one position. Bit D_0 is placed in the position of D_7 as well as in the Carry flag. CY is modified according to bit D_0. S, Z, P, AC are not affected. Example: RRC

Rotate accumulator left through carry

RAL	none	Each binary bit of the accumulator is rotated left by one position through the Carry flag. Bit D7 is placed in the Carry flag, and the Carry flag is placed in the least significant position D0. CY is modified according to bit D7. S, Z, P, AC are not affected. Example: RAL

Rotate accumulator right through carry

RAR	none	Each binary bit of the accumulator is rotated right by one position through the Carry flag. Bit D0 is placed in the Carry flag, and the Carry flag is placed in the most significant position D7. CY is modified according to bit D0. S, Z, P, AC are not affected. Example: RAR

Complement accumulator

CMA	none	The contents of the accumulator are complemented. No flags are affected. Example: CMA

Complement carry

CMC	none	The Carry flag is complemented. No other flags are affected. Example: CMC

Set Carry

STC	none	The Carry flag is set to 1. No other flags are affected. Example: STC

CONTROL INSTRUCTIONS

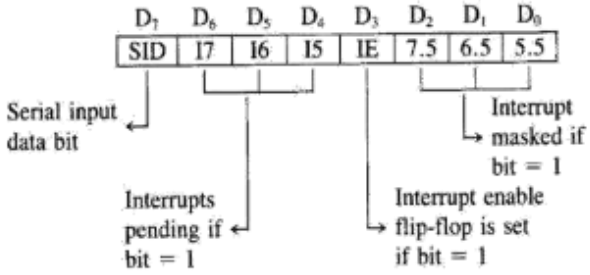
Opcode	Operand	Description
No operation		
NOP	none	No operation is performed. The instruction is fetched and decoded. However no operation is executed. Example: NOP
Halt and enter wait state		
HLT	none	The CPU finishes executing the current instruction and halts any further execution. An interrupt or reset is necessary to exit from the halt state. Example: HLT
Disable interrupts		
DI	none	The interrupt enable flip-flop is reset and all the interrupts except the TRAP are disabled. No flags are affected. Example: DI
Enable interrupts		
EI	none	The interrupt enable flip-flop is set and all interrupts are enabled. No flags are affected. After a system reset or the acknowledgement of an interrupt, the interrupt enable flip-flop is reset, thus disabling the interrupts. This instruction is necessary to reenable the interrupts (except TRAP). Example: EI

Read interrupt mask

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. The instruction loads eight bits in the accumulator with the following interpretations.

Example: RIM

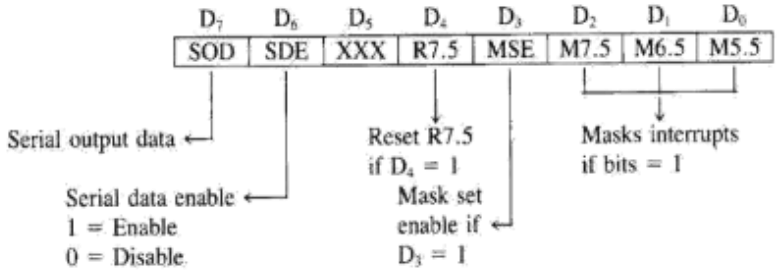


Set interrupt mask

SIM none

This is a multipurpose instruction and used to implement the 8085 interrupts 7.5, 6.5, 5.5, and serial data output. The instruction interprets the accumulator contents as follows.

Example: SIM



- SOD—Serial Output Data: Bit D_7 of the accumulator is latched into the SOD output line and made available to a serial peripheral if bit $D_6 = 1$.
- SDE—Serial Data Enable: If this bit = 1, it enables the serial output. To implement serial output, this bit needs to be enabled.
- XXX—Don't Care
- R7.5—Reset RST 7.5: If this bit = 1, RST 7.5 flip-flop is reset. This is an additional control to reset RST 7.5.
- MSE—Mask Set Enable: If this bit is high, it enables the functions of bits D_2, D_1, D_0. This is a master control over all the interrupt masking bits. If this bit is low, bits D_2, D_1, and D_0 do not have any effect on the masks.
- M7.5—D_2 = 0, RST 7.5 is enabled.
  = 1, RST 7.5 is masked or disabled.
- M6.5—D_1 = 0, RST 6.5 is enabled.
  = 1, RST 6.5 is masked or disabled.
- M5.5—D_0 = 0, RST 5.5 is enabled.
  = 1, RST 5.5 is masked or disabled.

QUESTION BANK

PART A

Part A

1. Define microprocessor
2. In how many groups can the signals of 8085 be classified?
3. What is the technology used in the manufacture of 8085?
4. Draw the block diagram of the built-in clock generator of 8085
5. What is the purpose of CLK signal of 8085?
6. What are the widths of data bus (DB) and address bus (AB) of 8085?
7. The address capability of 8085 is 64 KB. Explain.
8. Does 8085 have serial I/O control
9. What jobs ALU of 8085 can perform?
10. How many hardware interrupts 8085 supports?
11. How many I/O ports can 8085 access?
12. Why the lower byte address bus (A0 – A7) and data bus (D0 – D7) are multiplexed?
13. Why the lower byte address bus (A0 – A7) and data bus (D0 – D7) are multiplexed?
14. List the interrupts of 8085
15. List the flag bits of 8086

PART B

1. Explain the architecture of 8085
2. Discuss the addressing modes of 8085
3. Explain the 8086 architecture with neat diagram
4. Explain a. XTHL b.SPHL c.PCHL d.RAR e.SIM
5. Explain with example a.LDAX B b.PUSH PSW c.RLCd.JNC 16bit e.XRA A
6. Write an ALP to sort a given array in ascending order
7. Explain the interrupts of 8085

TEXT / REFERENCE BOOKS

1. Ramesh Goankar, "Microprocessor architecture programming and applications with 8085 / 8088", 5th Edition, Penram International Publishing, 2002.

2. Mazidi & McKinlay, “The 8051 Microcontroller and Embedded Systems using Assembly and C”, PHI, 2007.

3. MykePredko, “Programming and Customizing the 8051 Micro-controller”, Tata McGraw-Hill edition, 2007.

4. R A Gaonkar, “Fundamentals of Microcontrollers and Applications in Embedded Systems (with the PIC18 Microcontroller Family)”, Penram Publishing India, 2007

5. Kenneth Ayala ,”The 8051 Microcontroller”, 3rd Edition, Thomson Delmar Learning, 2004.

6. Kenneth J. Ayala, Dhananjay V. Gadre, “The 8051 Microcontroller & Embedded Systems Using Assembly and C”, Cengage Learning India Publication, 2007.

7. Ajay V Deshmukh, “Microcontrollers: Theory and Applications”, Tata McGraw-Hill, 2005

8. Raj Kamal, “Embedded Systems Architecture, Programming, and Design”. (2/e), Tata McGraw Hill, 2008.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF ELECTRICAL AND ELECTRONICS ENGINEERING

DEPARTMENT OF ELECTRONICS AND INSTRUMENTATION ENGINEERING

UNIT – II– MICROPROCESSORS , MICROCONTROLLERS AND EMBEDDED SYSTEMS – SEIA 1504

UNIT 2 ARCHITECTURE OF 8051 AND INSTRUCTION SET

Introduction - Architecture of 8051 - Memory organization - Addressing modes - Instruction set – Assembly Language Programming - Jump, Loop and Call Instructions - Arithmetic and Logic Instructions - Bit Operations.

1. Architecture of 8051 Microcontroller

An 8051 microcontroller has the following 11 major components:

1. ALU (Arithmetic and Logic Unit)
2. PC (Program Counter)
3. Registers
4. Timers and counters
5. Internal RAM and ROM
6. Four general purpose parallel input/output ports
7. Interrupt control logic with five sources of interrupt
8. Serial date communication
9. PSW (Program Status Word)
10. Data Pointer (DPTR)
11. Stack Pointer (SP)

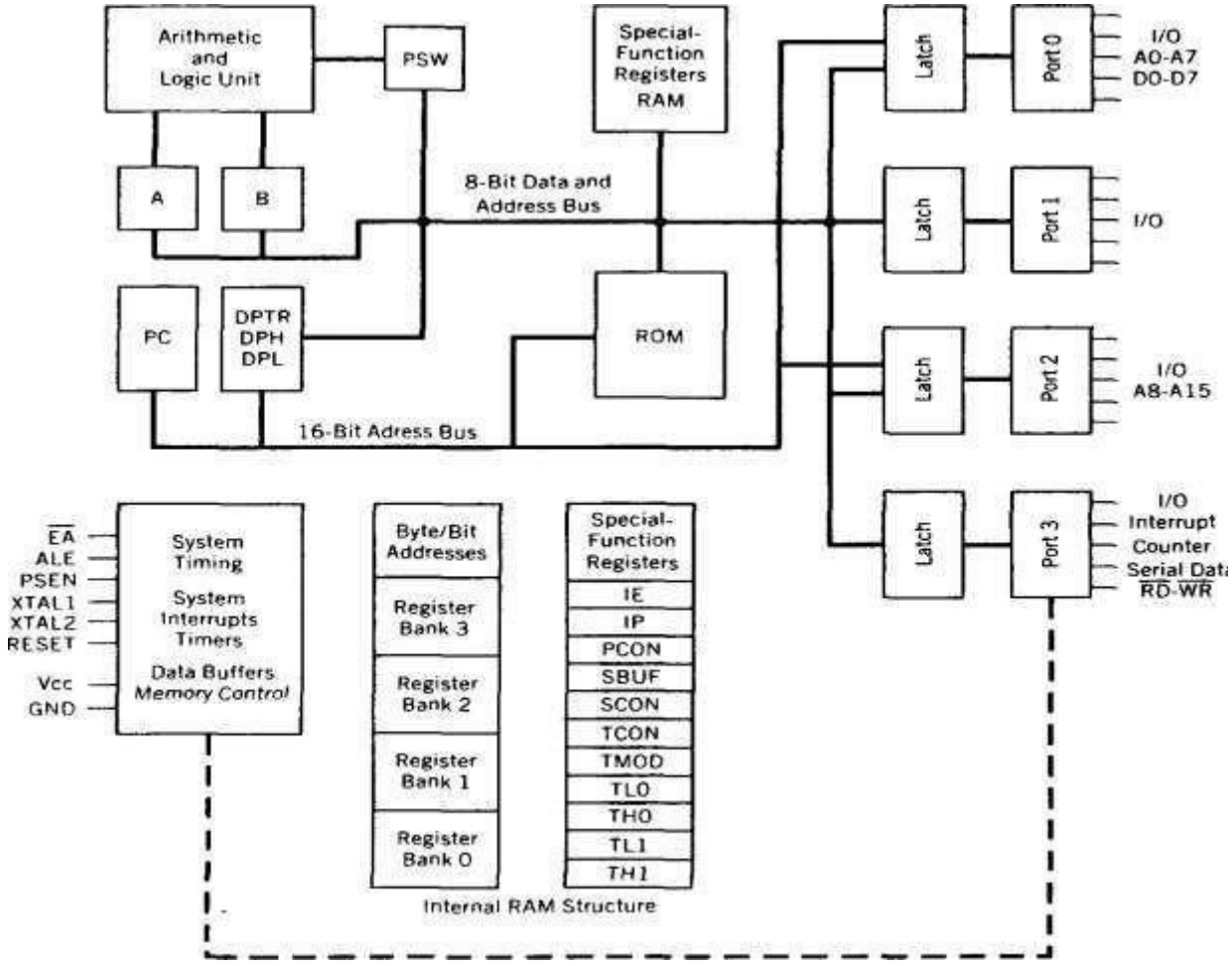


Fig 2.1: 8051- Architecture

The unique features are

Internal ROM and RAM, I/O ports with programmable pins, Timers and counters, Serial Data communication

2.Programming Model of 8051

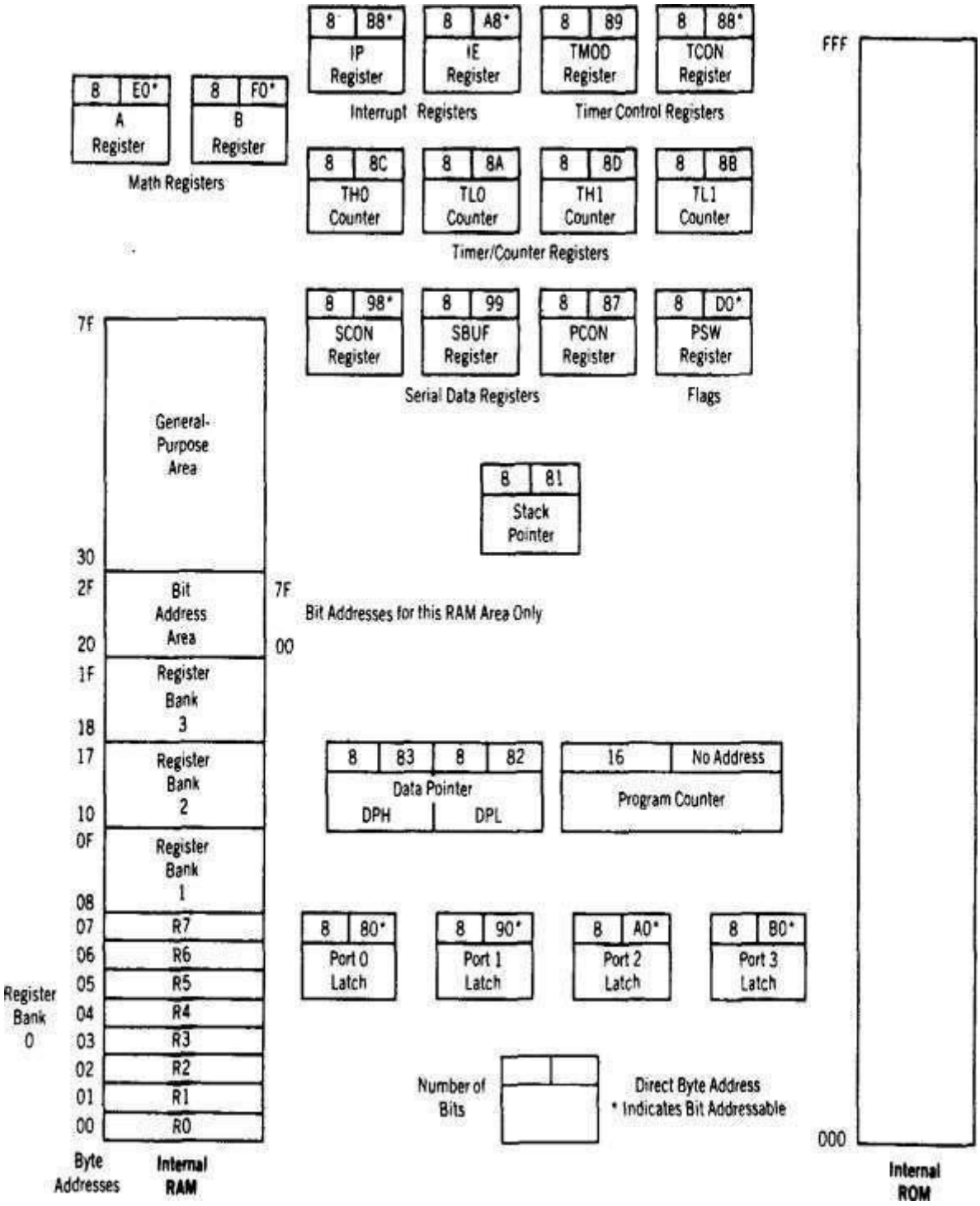


Fig 2.2: Programming Model

The above diagram shows the programming model of

8051. The 8051 architecture consists of these specific

features:

- 8 bit CPU with registers A and B
- 16 bit PC and DPTR
- 8 bit Program status word (PSW)
- 8 bit Stack pointer(SP)
- Internal ROM (4K)
- Internal RAM of 128 bytes
    - 4 register banks, each containing 8 registers
    - 16 bytes, which may be addressed at the bitlevel
    - 80 bytes of general purpose data memory
- 32 input/output pins arranged as four 8 bit ports: P0-P3
- Two 16 bit Timers/Counters: T0 and T1
- Full duplex serial data receiver/transmitter: SBUF
- Control Register: TCON,TMOD,SCON,PCON,IP and IE
- Two external and three internal interrupt sources
- Oscillator and

clock circuits Special

Function Registers (SFRs)

Special Function Registers (SFRs) are a sort of control table used for running and monitoring the operation of the microcontroller. Each of these registers as well as each bit they include, has its name, address in the scope of RAM and precisely defined purpose such as timer control, interrupt control, serial communication control etc. Even though there are 128 memory locations intended to be occupied by them, the basic core, shared by all types of 8051 microcontrollers, has only 21 such registers. Rest of locations are intensionally left

unoccupied in order to enable the manufacturers to further develop microcontrollers keeping them compatible with the previous versions. It also enables programs written a long time ago for microcontrollers which are out of production now to be used today.

F8									FF
F0	B								F7
E8									EF
E0	ACC								E7
D8									DF
D0	PSW								D7
C8									CF
C0									C7
B8	IP								BF
B0	P3								B7
A8	IE								AF
A0	P2								A7
98	SCON	SBUF							9F
90	P1								97
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
80	P0	SP	DPL	DPH				PCON	87

Bit-addressable Registers

Fig 2.3 : Special Function Register

A Register (Accumulator)

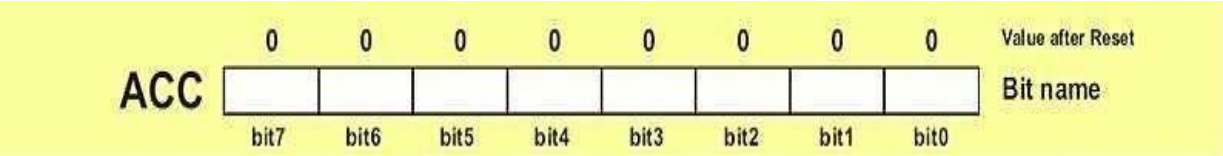


Fig 2.4: Accumulator

A register is a general-purpose register used for storing intermediate results obtained during operation. Prior to executing an instruction upon any number or operand it is necessary to store itin the accumulator first. All results obtained from arithmetical operations performed by the ALU are stored in the accumulator. Data to be moved from one register to another must go through theaccumulator. In other words, the A register is the most commonly used register and it is impossible to imagine a microcontroller without it. More than half instructions used by the 8051 microcontroller use somehow the accumulator.

B Register

Multiplication and division can be performed only upon numbers stored in the A and B registers.All other instructions in the program can use this register as a spare accumulator (A).

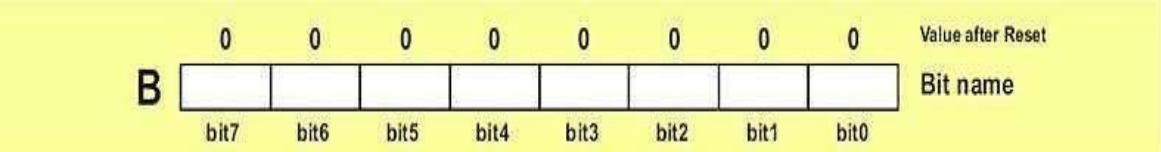


Fig 2.5: B Register

R Registers
(R0-R7)

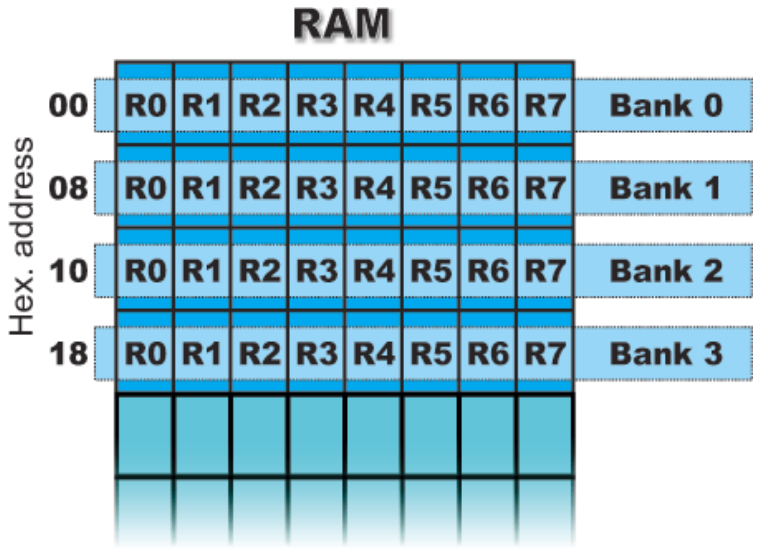


Fig 2.6: Register Banks

This is a common name for 8 general-purpose registers (R0, R1, R2 ...R7). Even though they are not true SFRs, they deserve to be discussed here because of their purpose. They occupy 4 banks within RAM. Similar to the accumulator, they are used for temporary storing variables and intermediate results during operation. Which one of these banks is to be active depends on two bits of the PSW Register. Active bank is a bank the registers of which are currently used.

The following example best illustrates the purpose of these registers. Suppose it is necessary to perform some arithmetical operations upon numbers previously stored in the R registers: (R1+R2) - (R3+R4). Obviously, a register for temporary storing results of addition is needed. This is how it looks in the program:

MOV A,R3; Means: move number from R3 into accumulator
ADD A,R4; Means: add number from R4 to accumulator (result remains in accumulator)
MOV R5,A; Means: temporarily move the result from accumulator into R5
MOV A,R1; Means: move number from R1 to accumulator
ADD A,R2; Means: add number from R2 to accumulator
SUBB A,R5; Means: subtract number from R5 (there are R3+R4)

Program Status Word (PSW) Register

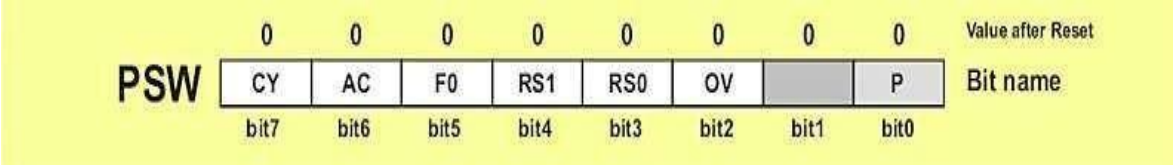


Fig 2.7: PSW

PSW register is one of the most important SFRs. It contains several status bits that reflect the current state of the CPU. Besides, this register contains Carry bit, Auxiliary Carry, two register bank select bits, Overflow flag, parity bit and user-definable status flag.

P - Parity bit. If a number stored in the accumulator is even then this bit will be automatically set (1), otherwise it will be cleared (0). It is mainly used during data transmit and receive via serial communication.

- Bit 1. This bit is intended to be used in the future versions of microcontrollers.

OV Overflow occurs when the result of an arithmetical operation is larger than 255 and cannot be stored in one register. Overflow condition causes the OV bit to be set (1). Otherwise, it will becleared (0).

RS0, RS1 - Register bank select bits. These two bits are used to select one of four register banks of RAM. By setting and clearing these bits, registers R0-R7 are stored in one of four banksof RAM.

RS1 RS2 Space in RAM

0 0 Bank0 00h-07h
0 1 Bank1 08h-0Fh
1 0 Bank2 10h-17h
1 1 Bank3 18h-1Fh

F0 - Flag 0. This is a general-purpose bit available for use.

AC - Auxiliary Carry Flag is used for BCD operations only.

CY - Carry Flag is the (ninth) auxiliary bit used for all arithmetical operations and shift instructions.

Data Pointer Register (DPTR)

DPTR register is not a true one because it doesn't physically exist. It consists of two separate registers: DPH (Data Pointer High) and (Data Pointer Low). For this reason it may be treated as a16-bit register or as two independent 8-bit registers. Their 16 bits are primarly used for external memory addressing. Besides, the DPTR Register is usually used for storing data and intermediateresults.

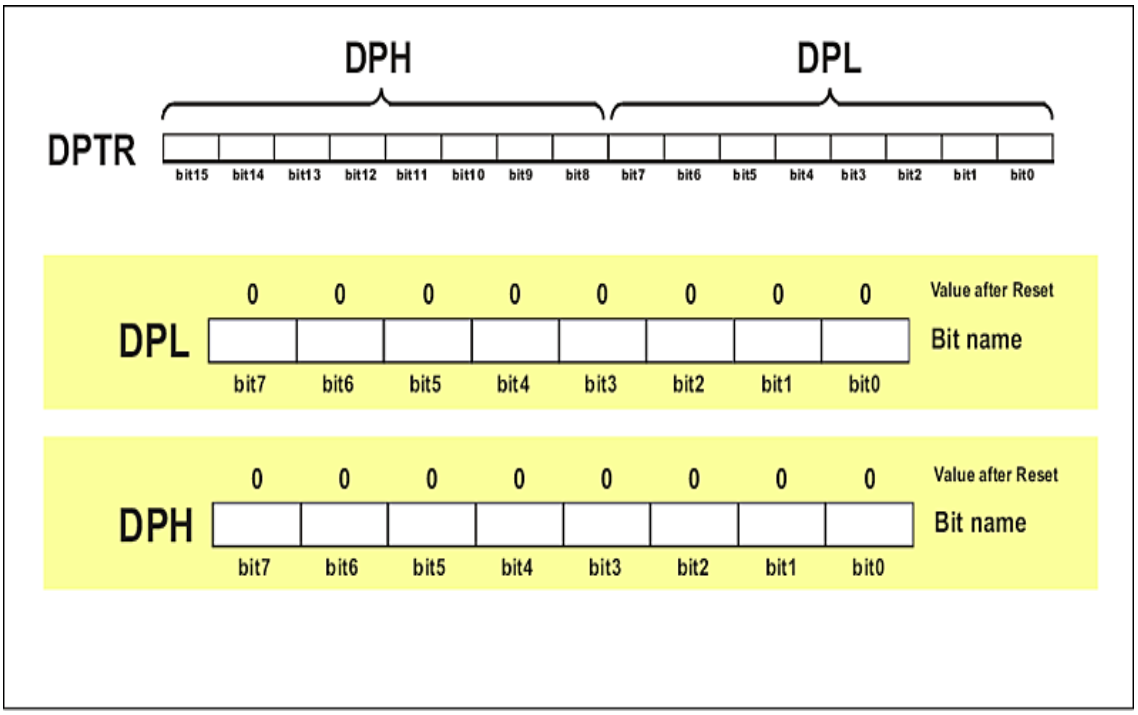


Fig 2.8: DPTR

Stack Pointer (SP)
Register

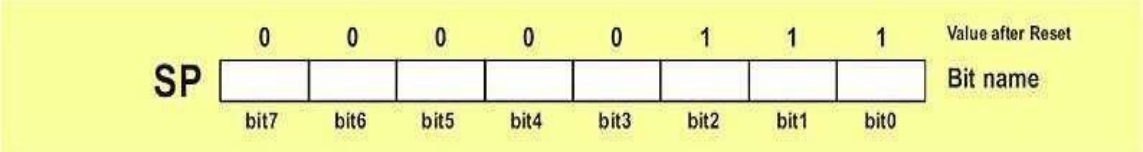


Fig 2.9: Stack Pointer

A value stored in the Stack Pointer points to the first free stack address and permits stack availability. Stack pushes increment the value in the Stack Pointer by 1. Likewise, stack pops decrement its value by 1. Upon any reset and power-on, the value 7 is stored in the Stack Pointer,which means that the space of RAM reserved for the stack starts at this location. If another value is written to this register, the entire Stack is moved to the new memory location.

P0, P1, P2, P3 - Input/Output Registers

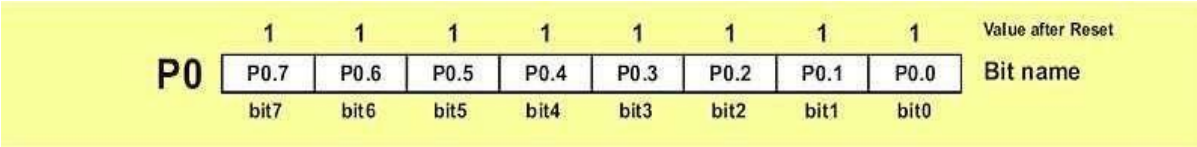


Fig 2.10: P0

If neither external memory nor serial communication system are used then 4 ports with in total of32 input/output pins are available for connection to peripheral environment. Each bit within theseports affects the state and performance of appropriate pin of the microcontroller.

Thus, bit logic state is reflected on appropriate pin as a voltage (0 or 5 V) and vice versa, voltage on a pin reflects the state of appropriate port bit.

As mentioned, port bit state affects performance of port pins, i.e. whether they will be configuredas inputs or outputs. If a bit is cleared (0), the appropriate pin will be configured as an output, while if it is set (1), the appropriate pin will be configured as an input. Upon reset and power-on, all port bits are set (1), which means that all appropriate pins will be configured as inputs.

Pinout Description

Pins 1-8: Port 1 Each of these pins can be configured as an input or an output.

Pin 9: RS A logic one on this pin disables the microcontroller and clears the contents of most registers. In other words, the positive voltage on this pin resets the microcontroller. By applying logic zero to this pin, the program starts execution from the beginning.

Pins10-17: Port 3 Similar to port 1, each of these pins can serve as general input or output. Besides, all of them have alternative functions:

Pin 10: RXD Serial asynchronous communication input or Serial synchronous communication output.

Pin 11: TXD Serial asynchronous communication output or Serial synchronous communication clock output.

Pin 12: INT0 Interrupt 0 input.

Pin 13: INT1 Interrupt 1 input.

Pin 14: T0 Counter 0 clock

Pin 15: T1 input.Counter 1

Pin 16: WRclock input.

Write to external (additional)

Pin 17: RD RAM.Read from external

Pin 18, 19: X2, X1 RAM.

Internal oscillator input and output. A quartz crystal which specifies Operating frequency is usually connected to these pins. Instead of it, miniature ceramics resonators can also be used for frequency stability. Later versions of microcontrollers operate ata frequency of 0 Hz up to over 50 Hz.

Pin 20: GND Ground.

Pin 21-28: Port 2 If there is no intention to use external memory then these port pins areconfigured as general inputs/outputs. In case external memory is used, the higher address byte,
i.e. addresses A8-A15 will appear on this port. Even though memory with capacity of 64Kb is not used, which means that not all eight port bits are used for its addressing, the rest of them arenot available as inputs/outputs.

Pin 29: PSEN If external ROM is used for storing program then a logic zero (0) appears on itevery time the microcontroller reads a byte from memory.

Pin 30: ALE Prior to reading from external memory, the microcontroller puts the lower address byte (A0-A7) on P0 and activates the ALE output. After receiving signal from the

ALE pin, the external register (usually 74HCT373 or 74HCT375 add-on chip) memorizes the state of P0 and uses it as a memory chip address. Immediately after that, the ALU pin is
returned its previous logic state and P0 is now used as a Data Bus. As seen, port data multiplexing is performed by means of only one additional (and cheap) integrated circuit. In other words, this port is used for both data and address transmission.

Pin 31: EA By applying logic zero to this pin, P2 and P3 are used for data and address transmission with no

PIN DIAGRAM OF 8051

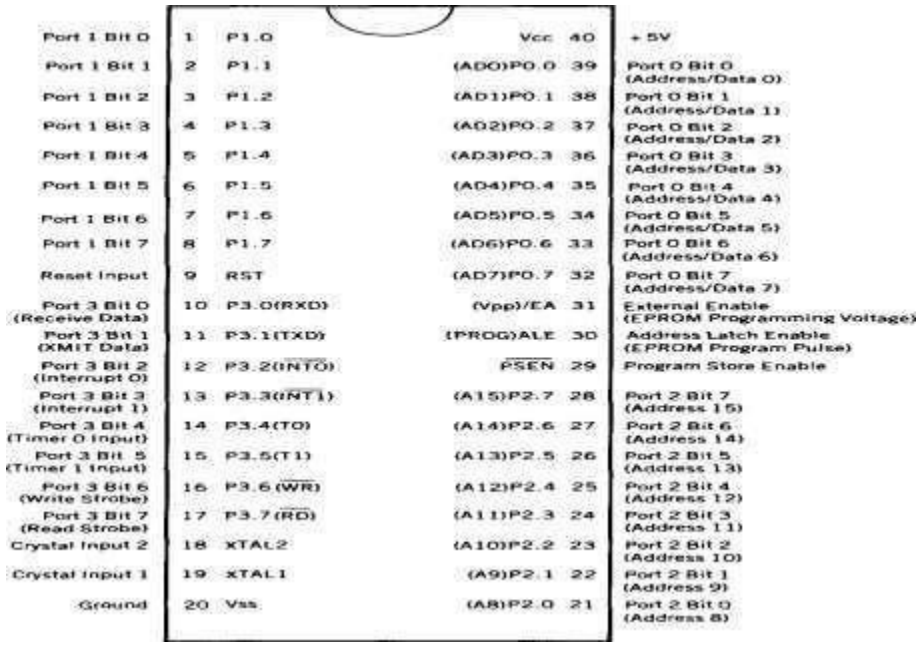


Fig 2.11: Pin Diagram-8051

3.Memory Organization

The 8051 has two types of memory and these are Program Memory and Data Memory. Program Memory (ROM) is used to permanently save the program being executed, while Data Memory (RAM) is used for temporarily storing data and intermediate results created and used during the operation of the microcontroller. Depending on the model in use (we are still talking about the 8051 microcontroller family in general) at most a few Kb of ROM and 128 or 256 bytes of RAM is used. However All 8051 microcontrollers have a 16-bit addressing bus and are capable of addressing 64 kb memory. It is neither a mistake nor a big ambition of engineers who were working on basic core development. It is a matter of smart memory organization which makes these microcontrollers a real ―programmers' goody―.Program Memory. The first models of the 8051 microcontroller family did not have internal program memory. It was added as an external separate chip. These models are recognizable by their label beginning with 803 (for example 8031 or 8032). All later models have a few Kbyte ROM embedded. Even though such an amountof memory is sufficient for writing most of the programs, there are situations when it is necessaryto use additional memory as well. A typical example is so called lookup tables. They are used in cases when

equations describing some processes are too complicated or when there is no time forsolving them. In such cases all necessary estimates and approximates are executed in advance and the final results are put in the tables (similar to logarithmic tables).

How does the microcontroller handle external memory depends on the EA pin logic state:

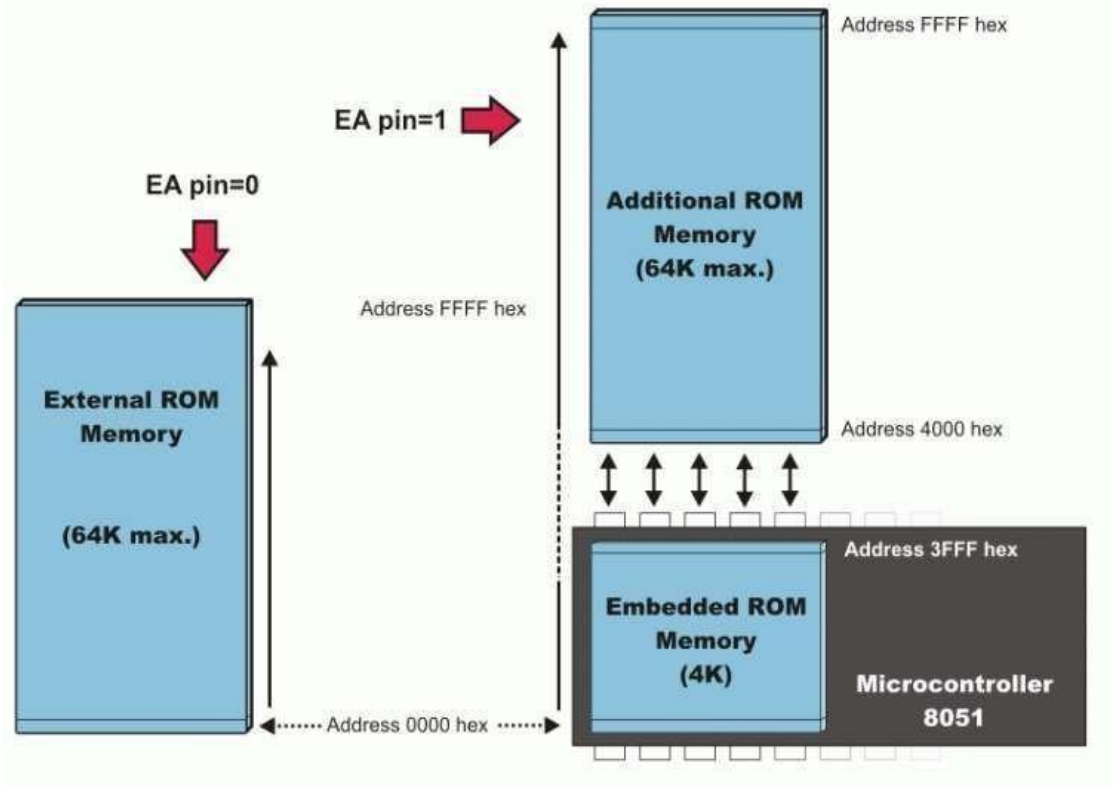


Fig 2.12: External memory EA pin

EA=0 In this case, the microcontroller completely ignores internal program memory andexecutes only the program stored in external memory.

EA=1 In this case, the microcontroller executes first the program from built-in ROM, then theprogram stored in external memory.

In both cases, P0 and P2 are not available for use since being used for data and addresstransmission. Besides, the ALE and PSEN pins are also used.

Data Memory

As already mentioned, Data Memory is used for temporarily storing data and intermediate resultscreated and used during the operation of the microcontroller. Besides, RAM memory built in the 8051 family includes many registers such as hardware counters and timers, input/output ports, serial data buffers etc. The previous models had 256 RAM locations, while for the later models this number was incremented by additional 128 registers. However, the first 256 memory locations (addresses 0-FFh) are the heart of memory common to all the models belonging to the 8051 family. Locations available to the user occupy memory space with addresses 0-7Fh, i.e. first 128 registers. This part of RAM is divided in several blocks.

The first block consists of 4 banks each including 8 registers denoted by R0-R7. Prior to accessing any of these registers, it is necessary to select the bank containing it. The next

memory block (address 20h-2Fh) is bit- addressable, which means that each bit has its own address (0- 7Fh). Since there are 16 such registers, this block contains in total of 128 bits with separate addresses (address of bit 0 of the 20h byte is 0, while address of bit 7 of the 2Fh byte is 7Fh). The third group of registers occupy addresses 2Fh-7Fh, i.e. 80 locations, and does not have any special functions or features.

Additional RAM

In order to satisfy the programmers‘ constant hunger for Data Memory, the manufacturers decided to embed an additional memory block of 128 locations into the latest versions of the 8051 microcontrollers. However, it‘s not as simple as it seems to be… The problem is that electronics performing addressing has 1 byte (8 bits) on disposal and is capable of reaching only the first 256 locations, therefore. In order to keep already existing 8-bit architecture and compatibility with other existing models a small trick was done.

What does it mean? It means that additional memory block shares the same addresses with locations intended for the SFRs (80h- FFh). In order to differentiate between these two physically separated memory spaces, different ways of addressing are used. The SFRs memory locations are accessed by direct addressing, while additional RAM memory locations are accessed by indirect addressing.

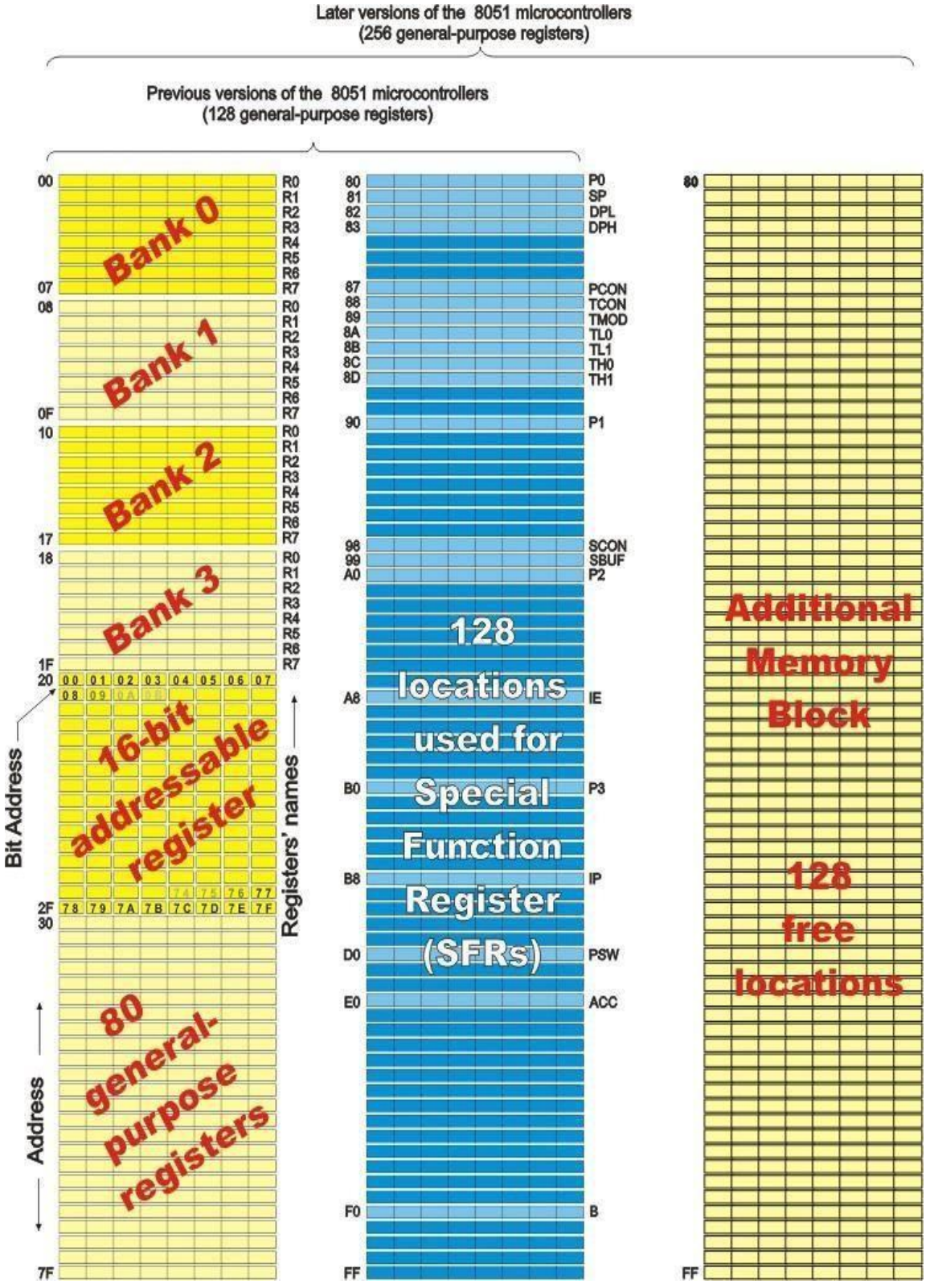


Fig 2.13 : Internal RAM

Memory expansion

In case memory (RAM or ROM) built in the microcontroller is not sufficient, it is possible to addtwo external memory chips with capacity of 64Kb each. P2 and P3 I/O ports are used for their addressing and data transmission.

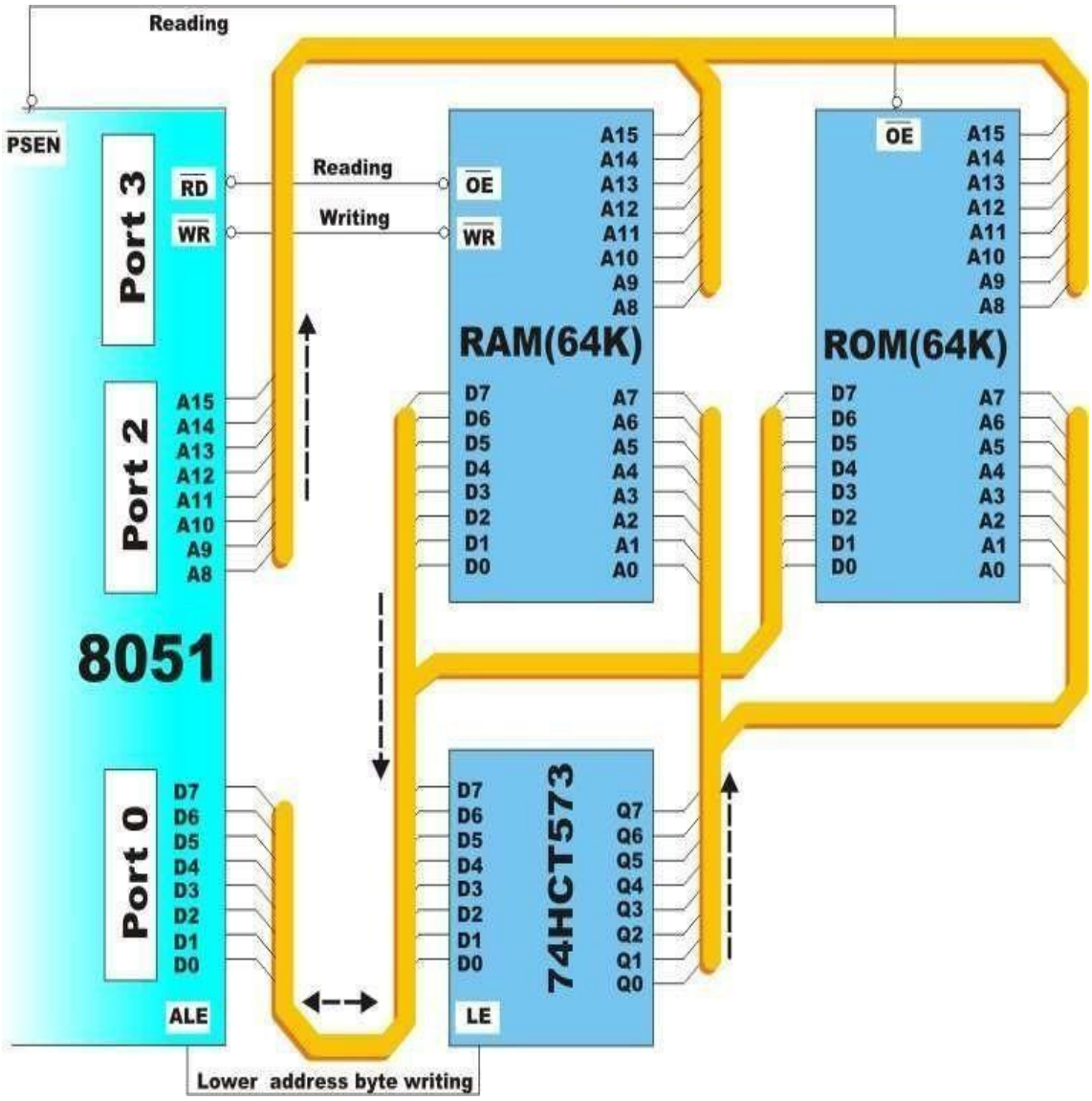


Fig 2.14: External Memory Interfacing

From the user‘s point of view, everything works quite simply when properly connected because most operations are performed by the microcontroller itself. The 8051 microcontroller has two pins for data read RD#(P3.7) and PSEN#. The first one is used for reading data from external data memory (RAM), while the other is used for reading data from external program memory (ROM). Both pins are active low. A typical example of memory expansion by adding RAM and ROM chips (Hardward architecture), is shown in figure above.

Even though additional memory is rarely used with the latest versions of the microcontrollers, we will describe in short what happens when memory chips are connected according to the previous schematic. The whole process described below is performed automatically.

- When the program during execution encounters an instruction which resides in external memory (ROM), the microcontroller will activate its control output ALE and set the first 8 bits of address (A0-A7) on P0. IC circuit 74HCT573 passes the first 8 bits to memory address pins.
- A signal on the ALE pin latches the IC circuit 74HCT573 and immediately afterwards 8 higher bits of address (A8-A15) appear on the port. In this way, a desired location of additional program memory is addressed. It is left over to read its content.
- Port P0 pins are configured as inputs, the PSEN pin is activated and the microcontroller reads from memory chip.

Similar occurs when it is necessary to read location from external RAM. Addressing is performed in the same way, while read and write are performed via signals appearing on the control outputs RD (is short for read) or WR (is short for write).

3. Addressing Modes

An "addressing mode" refers to how you are addressing a given memory location. In summary, the addressing modes are as follows, with an example of each:

Immediate Addressing	MOV A,#20h
Direct Addressing	MOV A,30h
Indirect Addressing	MOV A,@R0
External Direct	MOVX A,@DPTR
Code Indirect	MOVC A,@A+DPTR

Each of these addressing modes provides important flexibility.

Immediate Addressing

Immediate addressing is so-named because the value to be stored in memory immediately follows the operation code in memory. That is to say, the instruction itself dictates what value will be stored in memory.

For example, the instruction:

```
MOV A,#6Ah
```

This instruction uses Immediate Addressing because the Accumulator will be loaded with the value that immediately follows; in this case 6A (hexidecimal).

Immediate addressing is very fast since the value to be loaded is included in the instruction. However, since the value to be loaded is fixed at compile-time it is not very flexible.

Direct Addressing

Direct addressing is so-named because the value to be stored in memory is obtained by directly retrieving it from another memory location. For example:

```
MOV A,30h
```

This instruction will read the data out of Internal RAM address 30 (hexidecimal) and store it in the Accumulator.
Direct addressing is generally fast since, although the value to be loaded isnt included in the instruction, it is quickly accessable since it is stored in the 8051s Internal RAM. It is also much more flexible than Immediate Addressing since the value to be loaded is whatever is found at thegiven address--which may be variable.

Also, it is important to note that when using direct addressing any instruction which refers to an address between 00h and 7Fh is referring to Internal Memory. Any instruction which refers to an address between 80h and FFh is referring to the SFR control registers that control the 8051 microcontroller itself.

Indirect Addressing

Indirect addressing is a very powerful addressing mode which in many cases provides an exceptional level of flexibility. Indirect addressing is also the only way to access the extra 128 bytes of Internal RAM found on an 8052.

Indirect addressing appears as follows:

```
MOV A,@R0
```

This instruction causes the 8051 to analyze the value of the R0 register. The 8051 will then load the accumulator with the value from Internal RAM which is found at the address indicated by R0.

For example, lets say R0 holds the value 40h and Internal RAM address 40h holds the value 67h.When the above instruction is executed the 8051 will check the value of R0. Since R0 holds 40h the 8051 will get the value out of Internal RAM address 40h (which holds 67h) and store it in theAccumulator. Thus, the Accumulator ends up holding 67h.

Indirect addressing always refers to Internal RAM; it never refers to an SFR. Thus, in a prior example we mentioned that SFR 99h can be used to write a value to the serial port. Thus one may think that the following would be a valid solution to write the value 1 to the serial port:

```
MOV R0,#99h ;Load the address of the serial port
MOV @R0,#01h ;Send 01 to the serial port -- WRONG!!
```

This is not valid. Since indirect addressing always refers to Internal RAM these two instructions would write the value 01h to Internal RAM address 99h on an 8052. On an 8051 these two instructions would produce an undefined result since the 8051 only has 128 bytes

of Internal RAM.

External Direct

External Memory is accessed using a suite of instructions which use what I call "External Direct"addressing. I call it this because it appears to be direct addressing, but it is used to access externalmemory rather than internal memory.
There are only two commands that use External Direct addressing mode:

```
MOVXA,@DPT R
        MOVX
  @DPTR,A
```

Both commands utilize DPTR. In these instructions, DPTR must first be loaded with the address of external memory that you wish to read or write. Once DPTR holds the correct external memory address, the first command will move the contents of that external memory address into the Accumulator. The second command will do the opposite: it will allow you to write the value of the Accumulator to the external memory address pointed to by DPTR.

External Indirect

External memory can also be accessed using a form of indirect addressing which I call External Indirect addressing. This form of addressing is usually only used in relatively small projects that have a very small amount of external RAM. An example of this addressing mode is:

```
MOVX @R0,A
```

Once again, the value of R0 is first read and the value of the Accumulator is written to that address in External RAM. Since the value of @R0 can only be 00h through FFh the project would effectively be limited to 256 bytes of External RAM. There are relatively simple hardware/software tricks that can be implemented to access more than 256 bytes of memory using External Indirect addressing.

4. Instruction Set

The process of writing program for the microcontroller mainly consists of giving instructions (commands) in the specific order in which they should be executed in order to carry out a specific task. As electronics cannot ―understand‖ what for example an instruction ―if the push button is pressed- turn the light on‖ means, then a certain number of simpler and precisely defined orders that decoder can recognise must be used. All commands are known as INSTRUCTION SET. All microcontrollers compatibile with the 8051 have in total of 255 instructions, i.e. 255 different words available for program writing.

At first sight, it is imposing number of odd signs that must be known by heart. However, It is not so complicated as it looks like. Many instructions are considered to be ―different‖, even though they perform the same operation, so there are only 111 truly different commands. For example: ADD A,R0, ADD A,R1, ... ADD A,R7 are instructions that perform the same

operation (additon of the accumulator and register). Since there are 8 such registers, each instruction is counted separately. Taking into account that all instructions perform only 53 operations (addition, subtraction, copy etc.) and most of them are rarely used in practice, there are actually 20-30 abbreviations to be learned, which is acceptable.

Types of instructions

Depending on operation they perform, all instructions are divided in several groups:

- Arithmetic Instructions
- Branch Instructions
- Data Transfer Instructions
- Logic Instructions
- Bit-oriented Instructions

The first part of each instruction, called MNEMONIC refers to the operation aninstruction performs (copy, addition, logic operation etc.). Mnemonics are abbreviations of the name of operation being executed. For example:

- INC R1 - Means: Increment register R1 (increment register R1);
- LJMP LAB5 - Means: Long Jump LAB5 (long jump to the address marked as LAB5);
- JNZ LOOP - Means: Jump if Not Zero LOOP (if the number in the accumulator is not 0, jump to the address marked as LOOP);

The other part of instruction, called OPERAND is separated from mnemonic by at least one whitespace and defines data being processed by instructions. Some of the instructions have no operand, while some of them have one, two or three. Ifthere is more than one operand in an instruction, they are separated by a comma. For example:

- RET - return from a subroutine;
- JZ TEMP - if the number in the accumulator is not 0, jump to the address marked as TEMP;
- ADD A,R3 - add R3 and accumulator;
- CJNE A,#20,LOOP - compare accumulator with 20. If they are not equal, jump to the address marked as LOOP;

Arithmetic instructions

Arithmetic instructions perform several basic operations such as addition, subtraction, division, multiplication etc. After execution, the result is stored in the first operand. For example:

ADD A,R1 - The result of addition (A+R1) will be stored in the accumulator.

Arithmetic Instructions

Mnemonic	Description	Byte	Cycle
ADD A,Rn	Adds the register to the accumulator	1	1
ADD A,direct	Adds the direct byte to the accumulator	2	2
ADD A,@Ri	Adds the indirect RAM to the accumulator	1	2
ADD A,#data	Adds the immediate data to the accumulator	2	2

ADDC A,Rn	Adds the register to the accumulator with a carry flag	1	1
ADDC A,direct	Adds the direct byte to the accumulator with a carry flag	2	2
ADDC A,@Ri	Adds the indirect RAM to the accumulator with a carry flag	1	2
ADDC A,#data	Adds the immediate data to the accumulator with a carry flag	2	2
SUBB A,Rn	Subtracts the register from the accumulator with a borrow	1	1
SUBB A,direct	Subtracts the direct byte from the accumulator with a borrow	2	2
SUBB A,@Ri	Subtracts the indirect RAM from the accumulator with a borrow	1	2
SUBB A,#data	Subtracts the immediate data from the accumulator with a borrow	2	2
INC A	Increments the accumulator by 1	1	1
INC Rn	Increments the register by 1	1	2
INC Rx	Increments the direct byte by 1	2	3
INC @Ri	Increments the indirect RAM by 1	1	3
DEC A	Decrements the accumulator by 1	1	1
DEC Rn	Decrements the register by 1	1	1
DEC Rx	Decrements the direct byte by 1	1	2
DEC @Ri	Decrements the indirect RAM by 1	2	3
INC DPTR	Increments the Data Pointer by 1	1	3
MUL AB	Multiplies A and B	1	5
DIV AB	Divides A by B	1	5
DA A	Decimal adjustment of the accumulator according to BCD code	1	1

Branch Instructions

There are two kinds of branch instructions:
Unconditional jump instructions: upon their execution a jump to a new location from where theprogram continues execution is executed.
Conditional jump instructions: a jump to a new program location is executed only if a specifiedcondition is met. Otherwise, the program normally proceeds with the next instruction.

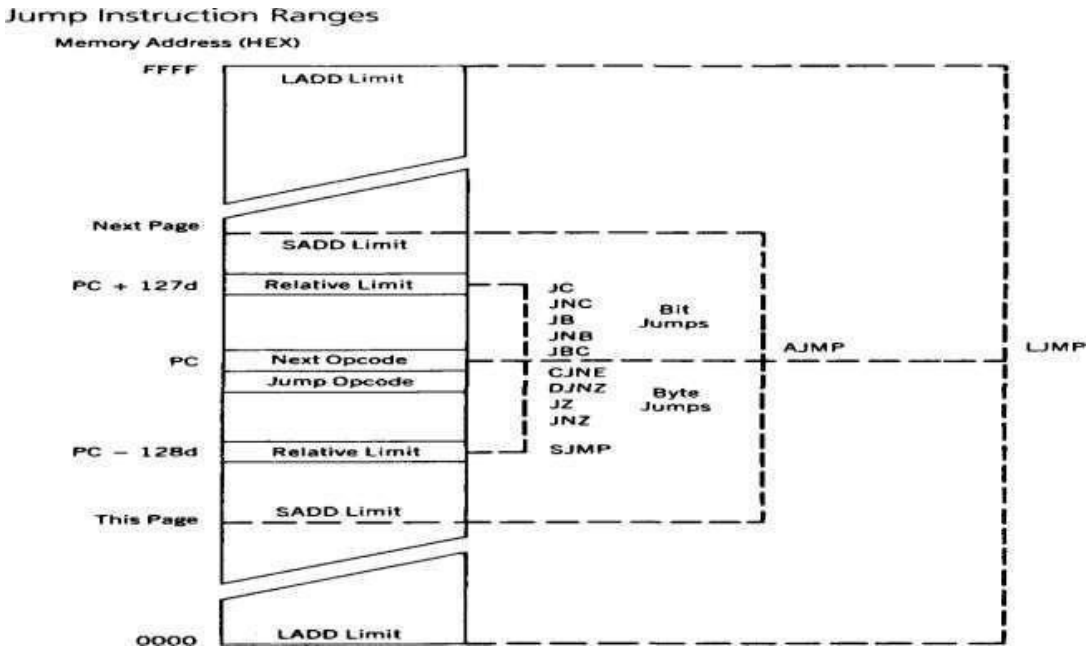


Fig 2.15: Jump Address Range

Branch Instructions

Mnemonic	Description	Byte	Cycle
ACALL addr11	Absolute subroutine call	2	6
LCALL addr16	Long subroutine call	3	6
RET	Returns from subroutine	1	4
RETI	Returns from interrupt subroutine	1	4
AJMP addr11	Absolute jump	2	3
LJMP addr16	Long jump	3	4
SJMP rel	Short jump (from –128 to +127 locations relative to the following instruction)	2	3
JC rel	Jump if carry flag is set. Short jump.	2	3
JNC rel	Jump if carry flag is not set. Short jump.	2	3
JB bit,rel	Jump if direct bit is set. Short jump.	3	4
JBC bit,rel	Jump if direct bit is set and clears bit. Short jump.	3	4
JMP @A+DPTR	Jump indirect relative to the DPTR	1	2
JZ rel	Jump if the accumulator is zero. Short jump.	2	3
JNZ rel	Jump if the accumulator is not zero. Short jump.	2	3
CJNE A,direct,rel	Compares direct byte to the accumulator and jumps if not equal. Short jump.	34	
CJNE A,#data,rel	Compares immediate data to the accumulator and jumps if not equal. Short jump.	34	
CJNE	Rn,#data,rel		

CJNE			
DJNZ Rn,rel	Decrements register and jumps if not 0. Short jump.	2	3
DJNZ Rx,rel	Decrements direct byte and jump if not 0. Short jump.	3	4
NOP	No operation	1	1

Data Transfer Instructions

Data transfer instructions move the content of one register to another. The register the content of which is moved remains unchanged. If they have the suffix ―X‖ (MOVX), the data is exchanged with external memory.

Data Transfer Instructions

Mnemonic	Description	Byte	Cycle
MOV A,Rn	Moves the register to the accumulator	1	1
MOV A,direct	Moves the direct byte to the accumulator	2	2
MOV A,@Ri	Moves the indirect RAM to the accumulator	1	2
MOV A,#data	Moves the immediate data to the accumulator	2	2
MOV Rn,A	Moves the accumulator to the register	1	2
MOV Rn,direct	Moves the direct byte to the register	2	4
MOV Rn,#data	Moves the immediate data to the register	2	2
MOV direct,A	Moves the accumulator to the direct byte	2	3
MOV direct,Rn	Moves the register to the direct byte	2	3
MOV direct,direct	Moves the direct byte to the direct byte	3	4
MOV direct,@Ri	Moves the indirect RAM to the direct byte	2	4
MOV direct,#data	Moves the immediate data to the direct byte	3	3
MOV @Ri,A	Moves the accumulator to the indirect RAM	1	3
MOV @Ri,direct	Moves the direct byte to the indirect RAM	2	5
MOV @Ri,#data	Moves the immediate data to the indirect RAM	2	3
MOV DPTR,#data	Moves a 16-bit data to the data pointer	3	3
MOVC A,@A+DPTR	Moves the code byte relative to the DPTR to the accumulator (address=A+DPTR)	1	3
MOVC A,@A+PC	Moves the code byte relative to the PC to the accumulator (address=A+PC)	1	3
MOVX A,@Ri	Moves the external RAM (8-bit address) to the accumulator	1	3-10
OVX A,@DPTR	Moves the external RAM (16-bit address) to the accumulator	1	3-10
MOVX @Ri,A	Moves the accumulator to the external RAM (8-bit address)	1	4-11
MOVX @DPTR,A	Moves the accumulator to the external RAM (16-bit address)	1	4-11
PUSH direct	Pushes the direct byte onto the stack	2	4
POP direct	Pops the direct byte from the stack/td>	2	3
XCH A,Rn	Exchanges the register with the accumulator	1	2
XCH A,direct	Exchanges the direct byte with the accumulator	2	3
XCH A,@Ri	Exchanges the indirect RAM with the accumulator	1	3

XCHD A,@Ri	Exchanges the low-order nibble indirect RAM with the	1	3 accumulator

Logic Instructions

Logic instructions perform logic operations upon corresponding two registers. After bits of execution, the result is stored in the first operand.

Logic Instructions

Mnemonic	Description	Byte	Cycle
ANL A,Rn	AND register to accumulator	1	1
ANL A,direct	AND direct byte to accumulator	2	2
ANL A,@Ri	AND indirect RAM to accumulator	1	2
ANL A,#data	AND immediate data to accumulator	2	2
ANL direct,A	AND accumulator to direct byte	2	3
ANL direct,#data	AND immediae data to direct register	3	4
ORL A,Rn	OR register to accumulator	1	1
ORL A,direct	OR direct byte to accumulator	2	2
ORL A,@Ri	OR indirect RAM to accumulator	1	2
ORL direct,A	OR accumulator to direct byte	2	3
ORL direct,#data	OR immediate data to direct byte	3	4
XRL A,Rn	Exclusive OR register to accumulator	1	1
XRL A,direct	Exclusive OR direct byte to accumulator	2	2
XRL A,@Ri	Exclusive OR indirect RAM to accumulator	1	2
XRL A,#data	Exclusive OR immediate data to accumulator	2	2
XRL direct,A	Exclusive OR accumulator to direct byte	2	3
XORL direct,#data	Exclusive OR immediate data to direct byte	3	4
CLR A	Clears the accumulator	1	1
CPL A	Complements the accumulator (1=0, 0=1)	1	1
SWAP A	Swaps nibbles within the accumulator	1	1
RL A	Rotates bits in the accumulator left	1	1
RLC A	Rotates bits in the accumulator left through carry	1	1
RR A	Rotates bits in the accumulator right	1	1
RRC A	Rotates bits in the accumulator right through carry	1	1

Bit-oriented Instructions

Similar to logic instructions, bit-oriented instructions perform logic operations. The difference isthat these are performed upon single bits.

Bit-oriented Instructions

Mnemonic	Description	Byte	Cycle
CLR C	Clears the carry flag	1	1
CLR bit	Clears the direct bit	2	3
SETB C	Sets the carry flag	1	1
SETB bit	Sets the direct bit	2	3
CPL C	Complements the carry flag	1	1
CPL bit	Complements the direct bit	2	3
ANL C,bit	AND direct bit to the carry flag	2	2
ANL C,/bit	AND complements of direct bit to the carry flag		
2	2 ORL C,bit OR direct bit to the carry flag		
2 2 ORL C,/bit	OR complements of direct bit to the carry		
flag 2	2 MOV C,bit Moves the direct bit to the carry flag		
2	2 MOV bit,C Moves the carry flag to the direct bit		
2	3		

6.8051 Microcontroller Interrupts

There are five interrupt sources for the 8051, which means that they can recognize 5 different events that can interrupt regular program execution. Each interrupt can be enabled or disabled bysetting bits of the IE register. Likewise, the whole interrupt system can be disabled by clearing the EA bit of the same register. Refer to figure below.

Now, it is necessary to explain a few details referring to external interrupts- INT0 and INT1. Ifthe IT0 and IT1 bits of the TCON register are set, an interrupt will be generated on high to lowtransition, i.e. on the falling pulse edge (only in that moment). If these bits are cleared, an interrupt will be continuously executed as far as the pins are held low.

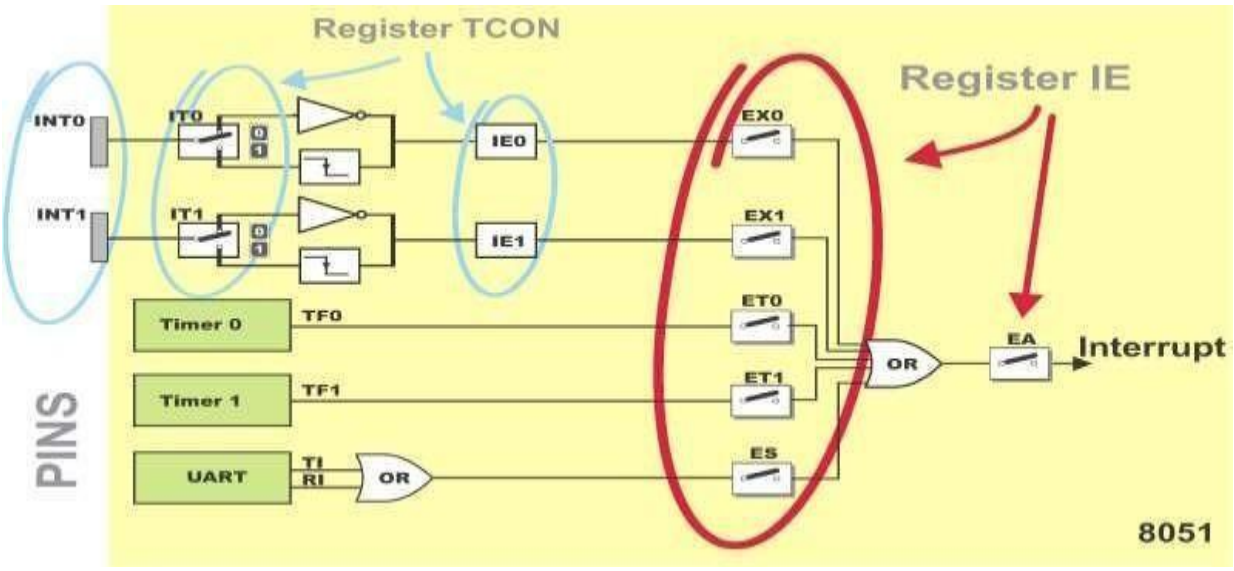


Fig 2.16:TCON

IE Register (Interrupt Enable)

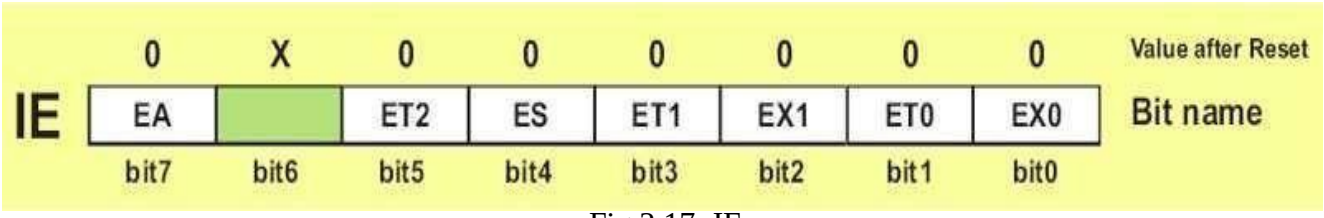


Fig 2.17: IE

- EA - global interrupt enable/disable:
  - 0 - disables all interrupt requests.
  - 1 - enables all individual interrupt requests.
- ES - enables or disables serial interrupt:
  - 0 - UART system cannot generate an interrupt.
  - 1 - UART system enables an interrupt.
- ET1 - bit enables or disables Timer 1 interrupt:
  - 0 - Timer 1 cannot generate an interrupt.
  - 1 - Timer 1 enables an interrupt.
- EX1 - bit enables or disables external 1 interrupt:
  - 0 - change of the pin INT0 logic state cannot generate an interrupt.
  - 1 - enables an external interrupt on the pin INT0 state change.
- ET0 - bit enables or disables timer 0 interrupt:
  - 0 - Timer 0 cannot generate an interrupt.
  - 1 - enables timer 0 interrupt.
- EX0 - bit enables or disables external 0 interrupt:
  - 0 - change of the INT1 pin logic state cannot generate an interrupt.
  - 1 - enables an external interrupt on the pin INT1 state change.

Interrupt Priorities

It is not possible to forseen when an interrupt request will arrive. If several interrupts areenabled, it may happen that while one of them is in progress, another one is requested. In order

that the microcontroller knows whether to continue operation or meet a new interrupt request,there is a priority list instructing it what to do.

The priority list offers 3 levels of interrupt priority:

1. Reset! The apsolute master. When a reset request arrives, everything is stopped and the microcontroller restarts.
2. Interrupt priority 1 can be disabled by Resetonly.
3. Interrupt priority 0 can be disabled by both Reset and interrupt priority1.

The IP Register (Interrupt Priority Register) specifies which one of existing interrupt sourceshave higher and which one has lower priority. Interrupt priority is usually specified at the beginning of the program. According to that, there are several possibilities:

- If an interrupt of higher priority arrives while an interrupt is in progress, it willbe immediately stopped and the higher priority interrupt will be executed first.
- If two interrupt requests, at different priority levels, arrive at the same time then the higher priority interrupt is serviced first.
- If the both interrupt requests, at the same priority level, occur one after another, the one which came later has to wait until routine being in progress ends.
- If two interrupt requests of equal priority arrive at the same time then the interrupt to be serviced is selected according to the following priority list:

1. External interrupt INT0
2. Timer 0 interrupt
3. External Interrupt INT1
4. Timer 1 interrupt
5. Serial Communication Interrupt

IP Register (Interrupt Priority)

The IP register bits specify the priority level of each interrupt (high or low priority).

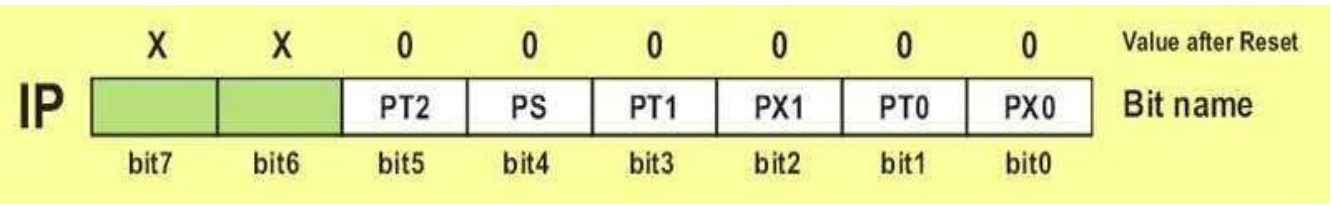


Fig 2.18: IP

- PS - Serial Port Interrupt priority bit
  - Priority 0
  - Priority 1
- PT1 - Timer 1 interrupt priority
  - Priority 0
  - Priority 1
- PX1 - External Interrupt INT1 priority
  - Priority 0
  - Priority 1
- PT0 - Timer 0 Interrupt Priority
  - Priority 0
  - Priority 1
- PX0 - External Interrupt INT0 Priority
  - Priority 0
  - Priority 1

Handling Interrupt

When an interrupt request arrives the following occurs:

1. Instruction in progress is ended.
2. The address of the next instruction to execute is pushed on the stack.
3. Depending on which interrupt is requested, one of 5 vectors (addresses) is written tothe program counter in accordance to the table below:
4.

Interrupt Source	Vector (address)
IE0	3 h
TF0	B h
TF1	1B h
RI, TI	23 h

   All addresses are in hexadecimal format
5. These addresses store appropriate subroutines processing interrupts. Instead of them, there are usually jump instructions specifying locations on which these subroutines reside.
6. When an interrupt routine is executed, the address of the next instruction to execute is poped from the stack to the program counter and interrupted program resumes operation from where it left off.

Reset

Reset occurs when the RS pin is supplied with a positive pulse in duration of at least 2 machine cycles (24 clock cycles of crystal oscillator). After that, the microcontroller generates an internal reset signal which clears all SFRs, except SBUF registers, Stack Pointer and ports (the state of the first two ports is not defined, while FF value is written to the ports configuring all their pins as inputs). Depending on surrounding and purpose of device, the RS pin is usually connected to apower-on reset push button or circuit or to both of them. Figure below illustrates one of the simplest circuit providing safe power-on reset.

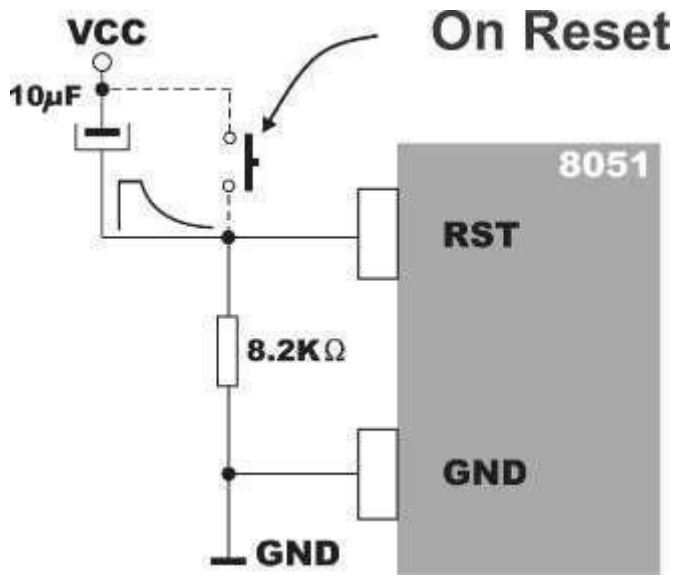


Fig 2.19:Reset

Basically, everything is very simple: after turning the power on, electrical capacitor is being charged for several milliseconds throgh a resistor connected to the ground. The pin is driven highduring this process. When the capacitor is charged, power supply voltage is already stable and the pin remains connected to the ground, thus providing normal operation of the microcontroller.Pressing the reset button causes the capacitor to be temporarily discharged and the microcontroller is reset. When released, the whole process is repeated…

Through the program- step by step...

microcontrollers normally operate at very high speed. The use of 12 Mhz quartz crystal enables instructions to be executed per second. Basically, there is no need for higher operating rate. In case it is needed, it is easy to built in a crystal for high frequency. The problem arises when it is necessary to slow down the operation of the microcontroller. For example during testing in real environment when it is necessary to execute several instructions step by step in order to check I/O pins' logic state.

Interrupt system of the 8051 microcontroller practically stops operation of the microcontrollerand enables instructions to be executed one after another by pressing the button. Two interruptfeatures enable that:

- Interrupt request is ignored if an interrupt of the same priority level is inprogress.
- Upon interrupt routine execution, a new interrupt is not executed until at least one instruction from the main program is executed.

In order to use this in practice, the following steps should be done:

1. External interrupt sensitive to the signal level should be enabled (for example INT0).
2. Three following instructions should be inserted into the program (at the 03hex. address):

```
JNB     P3.2$   ← Means: wait here until the pin P3.2 (INT0) is set to "1".
JB      P3.2$   ← Means: wait here until the pin P3.2 (INT0) is set to "0".
RETI            ← Means: go back to the main program
```

What is going on? As soon as the P3.2 pin is cleared (for example, by pressing the button), themicrocontroller will stop program execution and jump to the 03hex address will be executed. This address stores a short interrupt routine consisting of 3 instructions.

The first instruction is executed until the push button is realised (logic one (1) on the P3.2 pin). The second instruction is executed until the push button is pressed again. Immediately after that, the RETI instruction is executed and the processor resumes operation of the main program. Uponexecution of any program instruction, the interrupt INT0 is generated and the whole procedure is repeated (push button is still pressed). In other words, one button press - one instruction

6. Input/Output Ports

All 8051 microcontrollers have 4 I/O ports each comprising 8 bits which can be configured as inputs or outputs. Accordingly, in total of 32 input/output pins enabling the microcontroller to beconnected to peripheral devices are available for use.

Pin configuration, i.e. whether it is to be configured as an input (1) or an output (0), depends on its logic state. In order to configure a microcontroller pin as an output, it is necessary to apply a logic zero (0) to appropriate I/O port bit. In this case, voltage level on appropriate pin will be 0.

Similarly, in order to configure a microcontroller pin as an input, it is necessary to apply a logic one (1) to appropriate port. In this case, voltage level on appropriate pin will be 5V (as is the case with any TTL input). This may seem confusing but don't loose your patience. It all becomesclear after studying simple electronic circuits connected to an I/O pin.

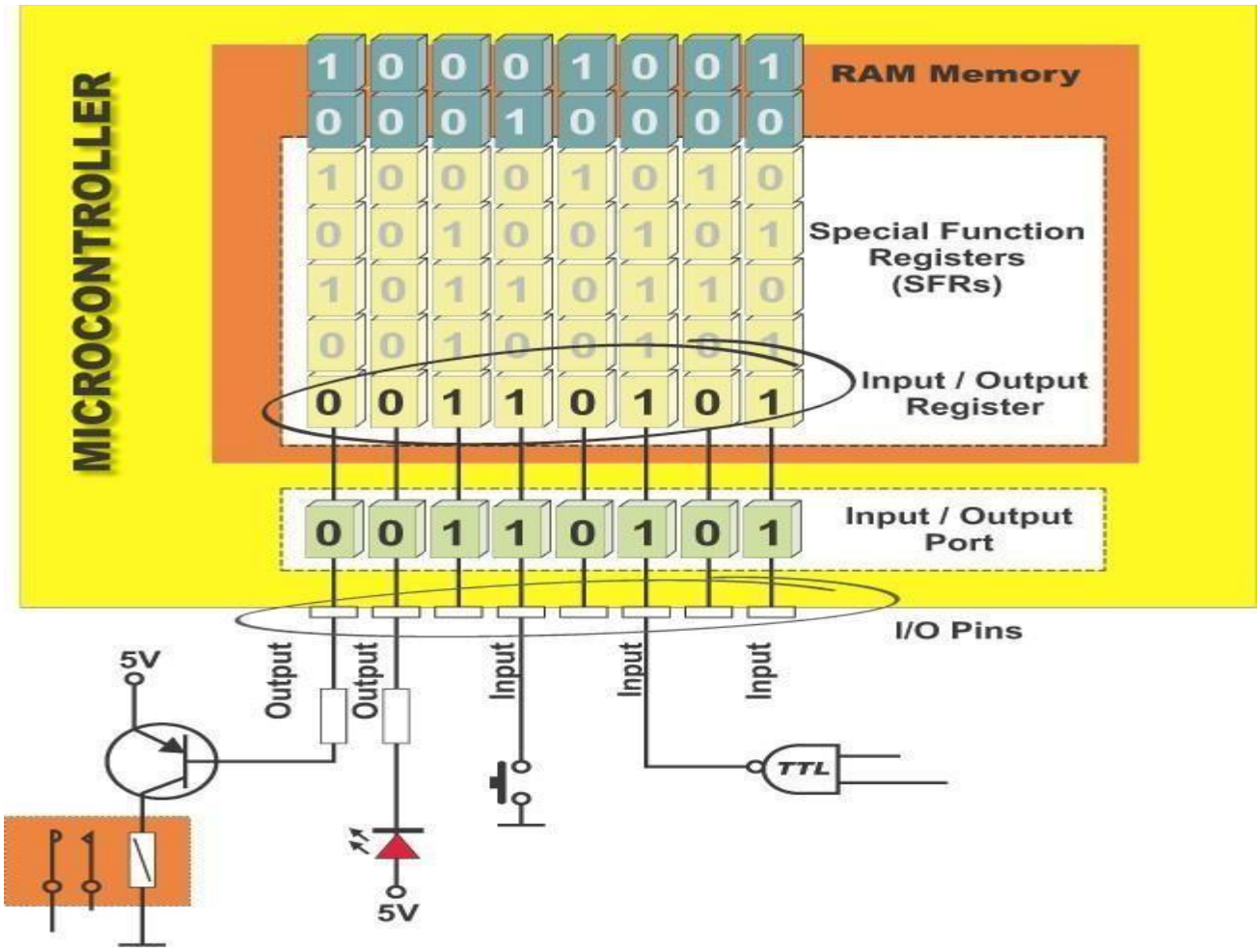


Fig 2.20: Input / Output

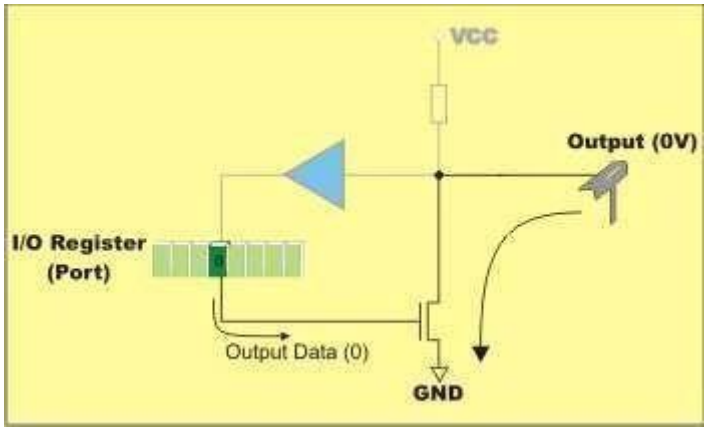


Fig 2.21: Output

Input/Output (I/O) pin Figure abov

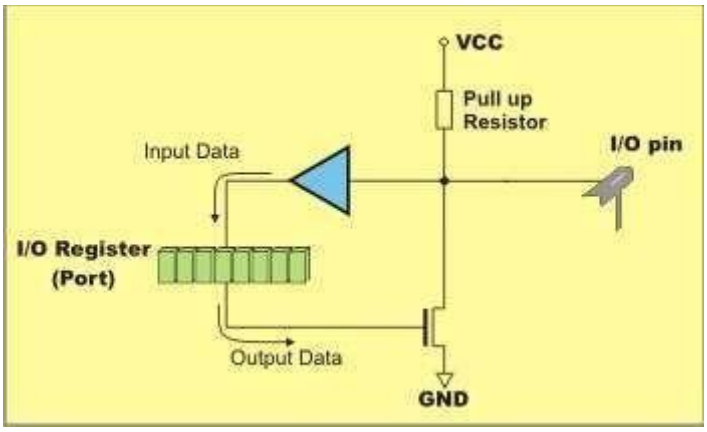


Fig 2.22: Input / output

Output pin

A logic zero (0) is applied to a bit of the P register. The output FE transistor is turned on, thus connecting the appropriate pin to ground.

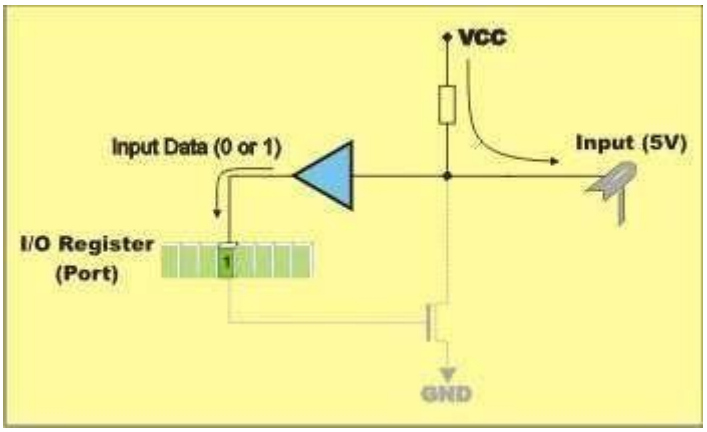


Fig 2.23 output

Hardware interrupts of 8085

Input pin

A logic one (1) is applied to a bit of the P register. The output FE transistor is turned off and theappropriate pin remains connected to the power supply voltage over a pull-up resistor of high resistance.

Logic state (voltage) of any pin can be changed or read at any moment. A logic zero (0) and logic one (1) are not equal. A logic one (0) represents a short circuit to ground. Such a pin acts asan output.

A logic one (1) is ―loosely‖ connected to the power supply voltage over a resistor of high resistance. Since this voltage can be easily ―reduced‖ by an external signal, such a pin acts as.

The P0 port is characterized by two functions. If external memory is used then the lower address byte (addresses A0-A7) is applied on it. Otherwise, all bits of this port are configured as inputs/outputs.The other function is expressed when it is configured as an output. Unlike other ports consisting of pins with built-in pull-up resistor connected by its end to 5 V power supply, pins of this port have this resistor left out. This apparently small difference has its consequences:

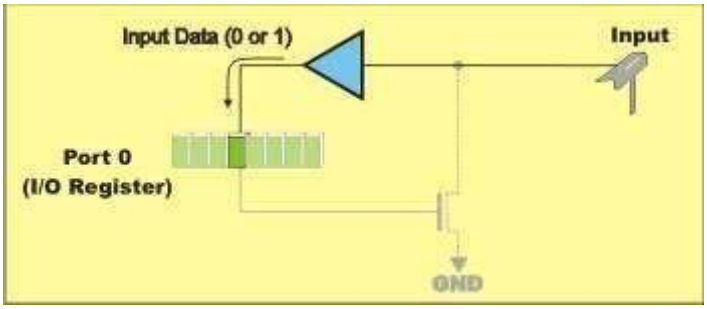


Fig 2.24: Port 0 configuration-input

If any pin of this port is configured as an input then it acts as if it ―floats‖. Such an input has unlimited input resistance and indetermined potential.

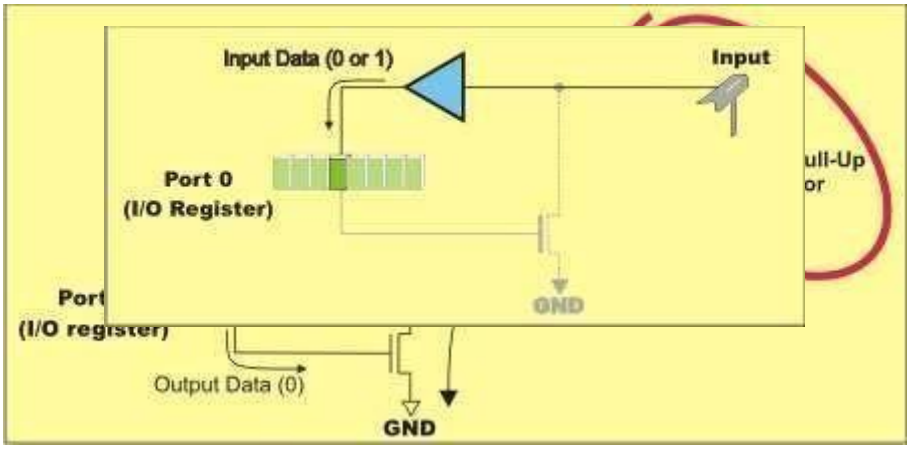


Fig 2.25: Port 0 configuration-output

When the pin is configured as an output, it acts as an ―open drain‖. By applying logic 0 to aport bit, the appropriate pin will be connected to ground (0V). By applying logic 1, the external output will keep on ―floating‖. In order to apply logic 1 (5V) on this output pin, it is necessary to built in an external pull-up resistor.Only in case P0 is used for addressing external memory, the microcontroller will provide internalpower supply source in order to supply its pins with logic one. There is no need to add

PORT A

P2 acts similarly to P0 when external memory is used. Pins of this port occupy addresses intended for external memory chip. This time it is about the higher address byte with addresses A8-A15. When no memory is added, this port can be used as a general input/output port showingfeatures similar to P1.

Port 3

All port pins can be used as general I/O, but they also have an alternative function. In order to use these alternative functions, a logic one (1) must be applied to appropriate bit of the P3 register. In tems of hardware, this port is similar to P0, with the difference that its pins have a pull-up resistor built-in.

Pin's Current limitations

When configured as outputs (logic zero (0)), single port pins can receive a current of 10mA. If all8 bits of a port are active, a total current must be limited to 15mA (port P0: 26mA). If all ports (32 bits) are active, total maximum current must be limited to 71mA. When these pins are configured as inputs (logic 1), built-in pull-up resistors provide very weak current, but strong enough to activate up to 4 TTL inputs of LS series.

As seen from description of some ports, even though all of them have more or less similar architecture, it is necessary to pay attention to which of them is to be used for what and how.

For example, if they shall be used as outputs with high voltage level (5V), then P0 should be avoided because its pins do not have pull-up resistors, thus giving low logic level only. When using other ports, one should have in mind that pull-up resistors have a relatively high resistance,so that their pins can give a current of several hundreds microamperes only.

Counters and Timers

As you already know, the microcontroller oscillator uses quartz crystal for its operation. As the frequency of this oscillator is precisely defined and very stable, pulses it generates are always of the same width, which makes them ideal for time measurement. Such crystals are also used in quartz watches. In order to measure time between two events it is sufficient to count up pulses coming from this oscillator. That is exactly what the timer does. If thetimer is properly programmed, the value stored in its register will be incremented (or decremented) with each coming pulse, i.e. once per each machine cycle. A single machine-cycle instruction lasts for 12 quartz oscillator periods, which means that by embedding quartz with oscillator frequency of 12MHz, a number stored in the timer register will be changed million times per second, i.e. each microsecond.

The 8051 microcontroller has 2 timers/counters called T0 and T1. As their names suggest, their main purpose is to measure time and count external events. Besides, they can be used for generating clock pulses to be used in serial communication, so called Baud Rate.

Timer T0

As seen in figure below, the timer T0 consists of two registers – TH0 and TL0 representing a lowand a high byte of one 16-digit binary number.

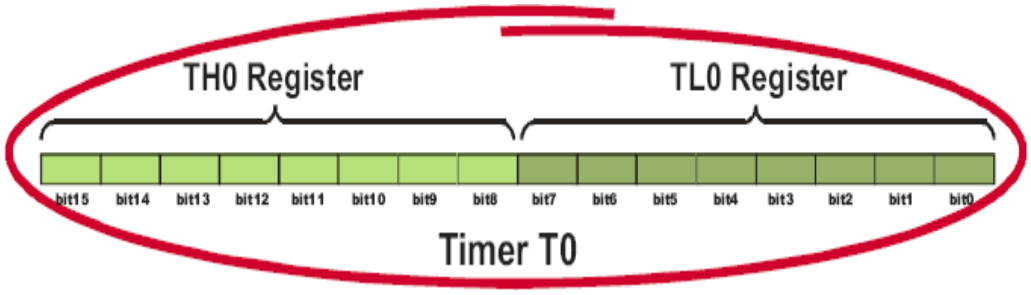


Fig 2.26: Timer 0

Accordingly, if the content of the timer T0 is equal to 0 (T0=0) then both registers it consists ofwill contain 0. If the timer contains for example number 1000 (decimal), then the TH0 register (high byte) will contain the number 3, while the TL0 register (low byte) will contain decimal number 232.

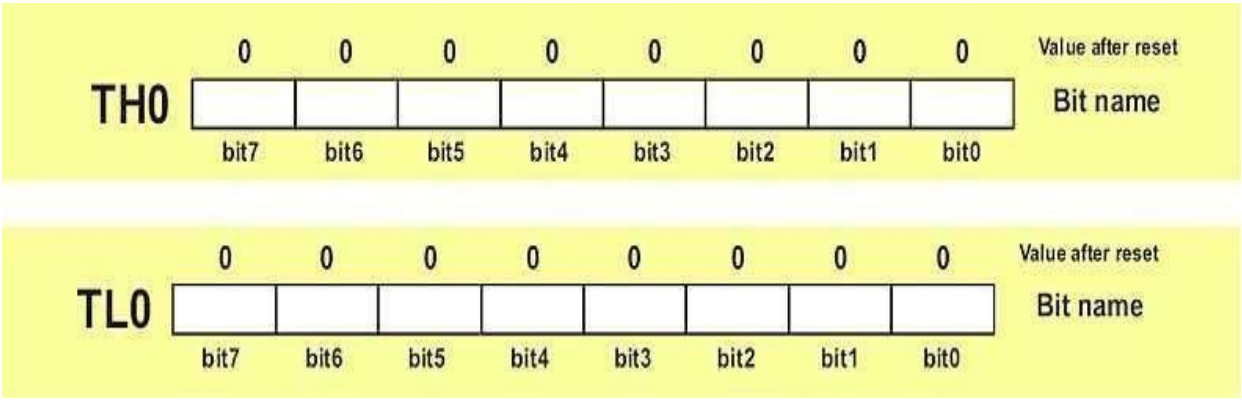


Fig 2.27: Timer 0-TLO& TL1

Formula used to calculate values in these two registers is very simple:TH0 × 256 + TL0 = T
Matching the previous example it would be as follows:3 × 256 + 232 = 1000

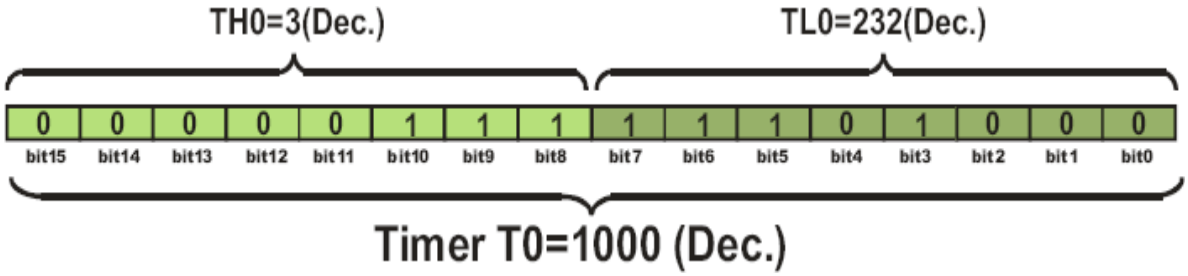


Fig 2.28: Timer 0

Since the timer T0 is virtually 16-bit register, the largest value it can store is 65 535. In case ofexceeding this value, the timer will be automatically cleared and counting starts from 0. This condition is called an overflow. Two registers TMOD and TCON are closely connected to thistimer and control its operation.

TMOD Register (Timer Mode)

The TMOD register selects the operational mode of the timers T0 and T1. As seen in figure below, the low 4 bits (bit0 - bit3) refer to the timer 0, while the high 4 bits (bit4 - bit7) refer tothe timer 1. There are 4 operational modes and each of them is described herein.

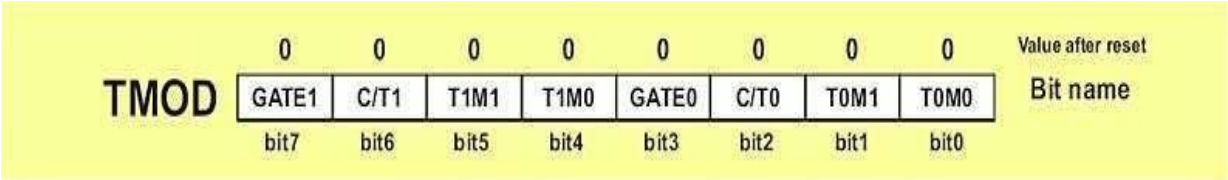


Fig 2.29: TMOD

Bits of this register have the following function:

- GATE1 enables and disables Timer 1 by means of a signal brought to the INT1 pin (P3.3):
  - 1 - Timer 1 operates only if the INT1 bit is set.
  - 0 - Timer 1 operates regardless of the logic state of the INT1 bit.
- C/T1 selects pulses to be counted up by the timer/counter 1:
  - 1 - Timer counts pulses brought to the T1 pin (P3.5).
  - 0 - Timer counts pulses from internal oscillator.
- T1M1,T1M0 These two bits select the operational mode of the Timer 1.

T1M1 T1M0 Mode Description

0	0	0	13-bit timer
0	1	1	16-bit timer
1	0	2	8-bit auto-reload

1	1	3	Split mode

- GATE0 enables and disables Timer 1 using a signal brought to the INT0 pin (P3.2):
    - 1 - Timer 0 operates only if the INT0 bit is set.
    - 0 - Timer 0 operates regardless of the logic state of the INT0 bit.
- C/T0 selects pulses to be counted up by the timer/counter 0:
    - 1 - Timer counts pulses brought to the T0 pin (P3.4).
    - 0 - Timer counts pulses from internal oscillator.
- T0M1,T0M0 These two bits select the oprtaional mode of the Timer 0.

T0M1	T0M0	Mode	Description
0	0	0	13-bit timer
0	1	1	16-bit timer
1	0	2	8-bit auto-reload
1	1	3	Split mode

Timer 0 in mode 0 (13-bit timer)

This is one of the rarities being kept only for the purpose of compatibility with the previuos versions of microcontrollers. This mode configures timer 0 as a 13-bit timer which consists of all8 bits of TH0 and the lower 5 bits of TL0. As a result, the Timer 0 uses only 13 of 16 bits. How does it operate? Each coming pulse causes the lower register bits to change their states. After receiving 32 pulses, this register is loaded and automatically cleared, while the higher byte (TH0)is incremented by 1. This process is repeated until registers count up 8192 pulses. After that, both registers are cleared and counting starts from 0.

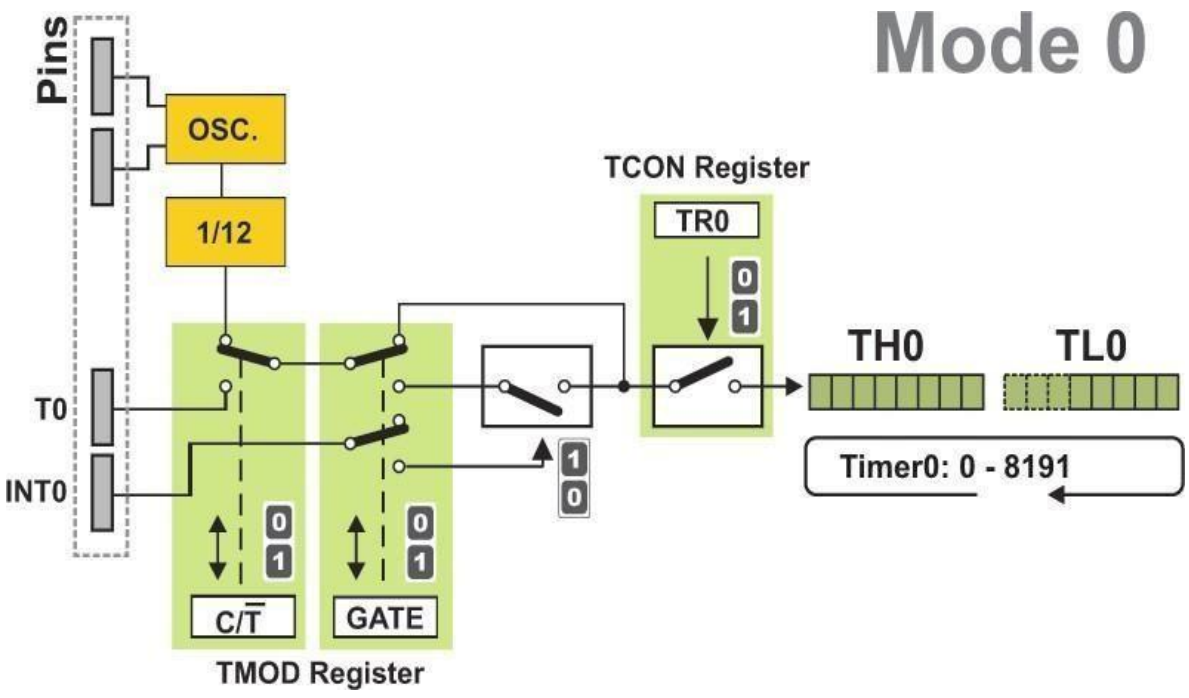


Fig 2.30: Timer Mode 0

Timer 0 in mode 1 (16-bit timer)

Mode 1 configures timer 0 as a 16-bit timer comprising all the bits of both registers TH0 and TL0. That's why this is one of the most commonly used modes. Timer operates in the same wayas in mode 0, with difference that the registers count up to 65 536 as allowable by the 16 bits.

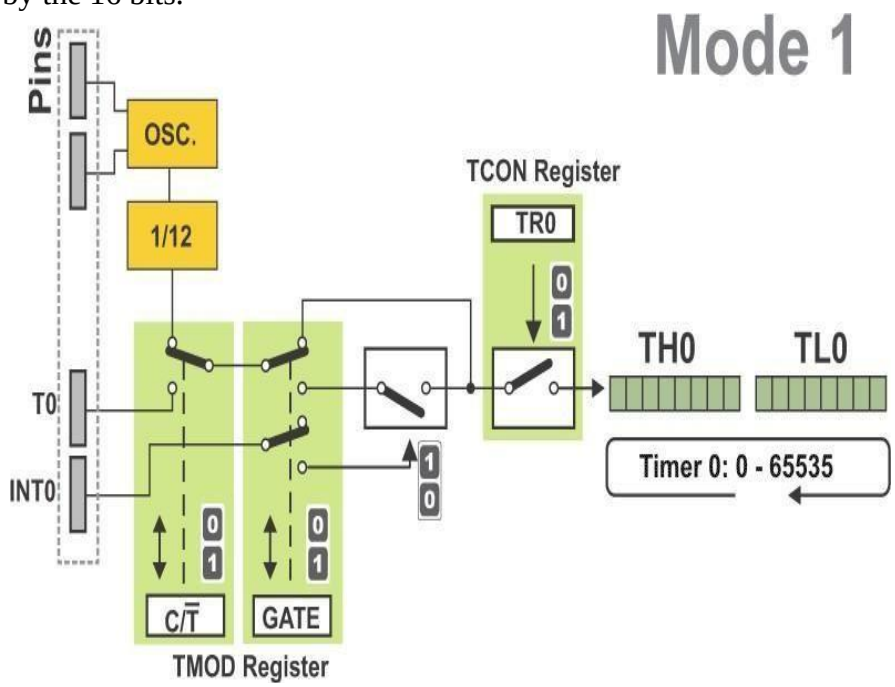


Fig 2.31: Timer Mode 1

Timer 0 in mode 2 (Auto-Reload Timer)

Mode 2 configures timer 0 as an 8-bit timer. Actually, timer 0 uses only one 8-bit register for counting and never counts from 0, but from an arbitrary value (0-255) stored in another (TH0)register.

The following example shows the advantages of this mode. Suppose it is necessary to constantlycount up 55 pulses generated by the clock.

If mode 1 or mode 0 is used, It is necessary to write the number 200 to the timer registers and constantly check whether an overflow has occured, i.e. whether they reached the value 255. When it happens, it is necessary to rewrite the number 200 and repeat the whole procedure. The same procedure is automatically performed by the microcontroller if set in mode 2. In fact, only the TL0 register operates as a timer, while another (TH0) register stores the value from which thecounting starts. When the TL0 register is loaded, instead of being cleared, the contents of TH0 will be reloaded to it. Referring to the previous example, in

order to register each 55th pulse, the best solution is to write the number 200 to the TH0 register and configure the timer to operate in mode 2.

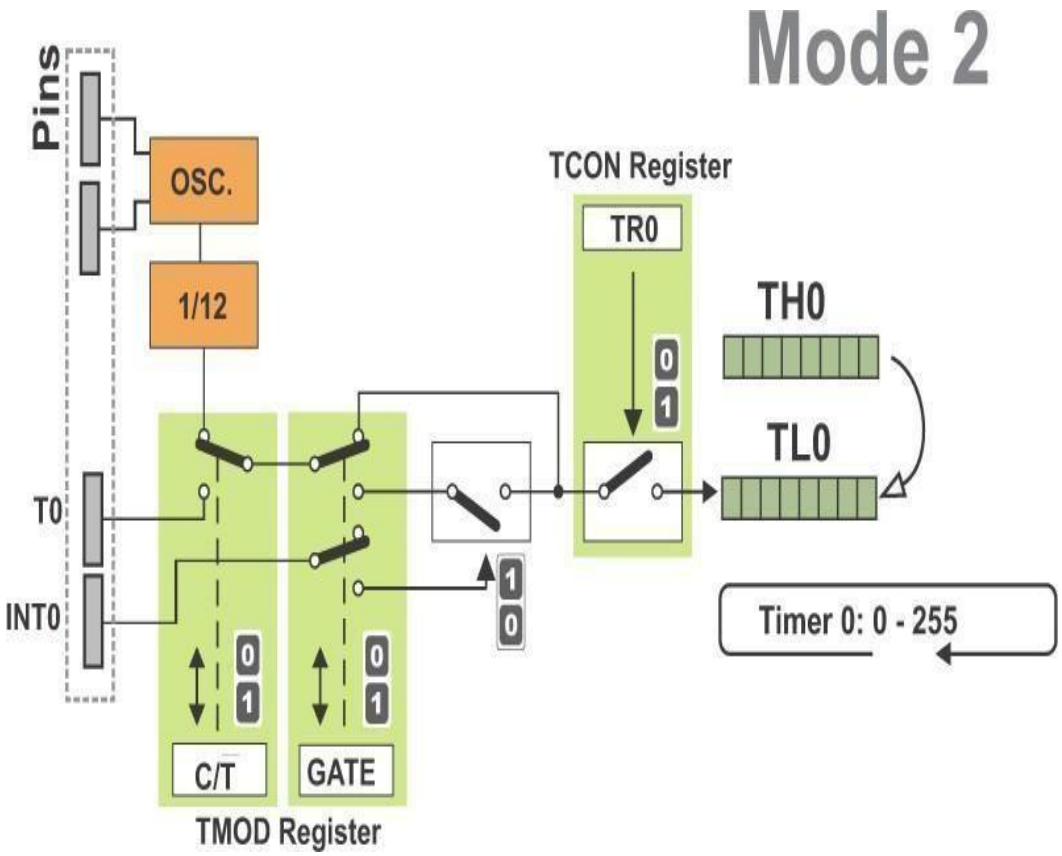


Fig 2.32: Timer Mode 2

Timer 0 in Mode 3 (Split Timer)

Mode 3 configures timer 0 so that registers TL0 and TH0 operate as separate 8-bit timers. In other words, the 16-bit timer consisting of two registers TH0 and TL0 is split into two independent 8-bit timers. This mode is provided for applications requiring an additional 8-bit timer or counter. The TL0 timer turns into timer 0, while the TH0 timer turns into timer **1.** In addition, all the control bits of 16-bit Timer 1 (consisting of the TH1 and TL1 register), now control the 8-bit Timer 1. Even though the 16-bit Timer 1 can still be configured to operate in any of modes (mode 1, 2 or 3), it is no longer possible to disable it as there is no control bit to doit. Thus, its operation is restricted when timer 0 is in mode 3.

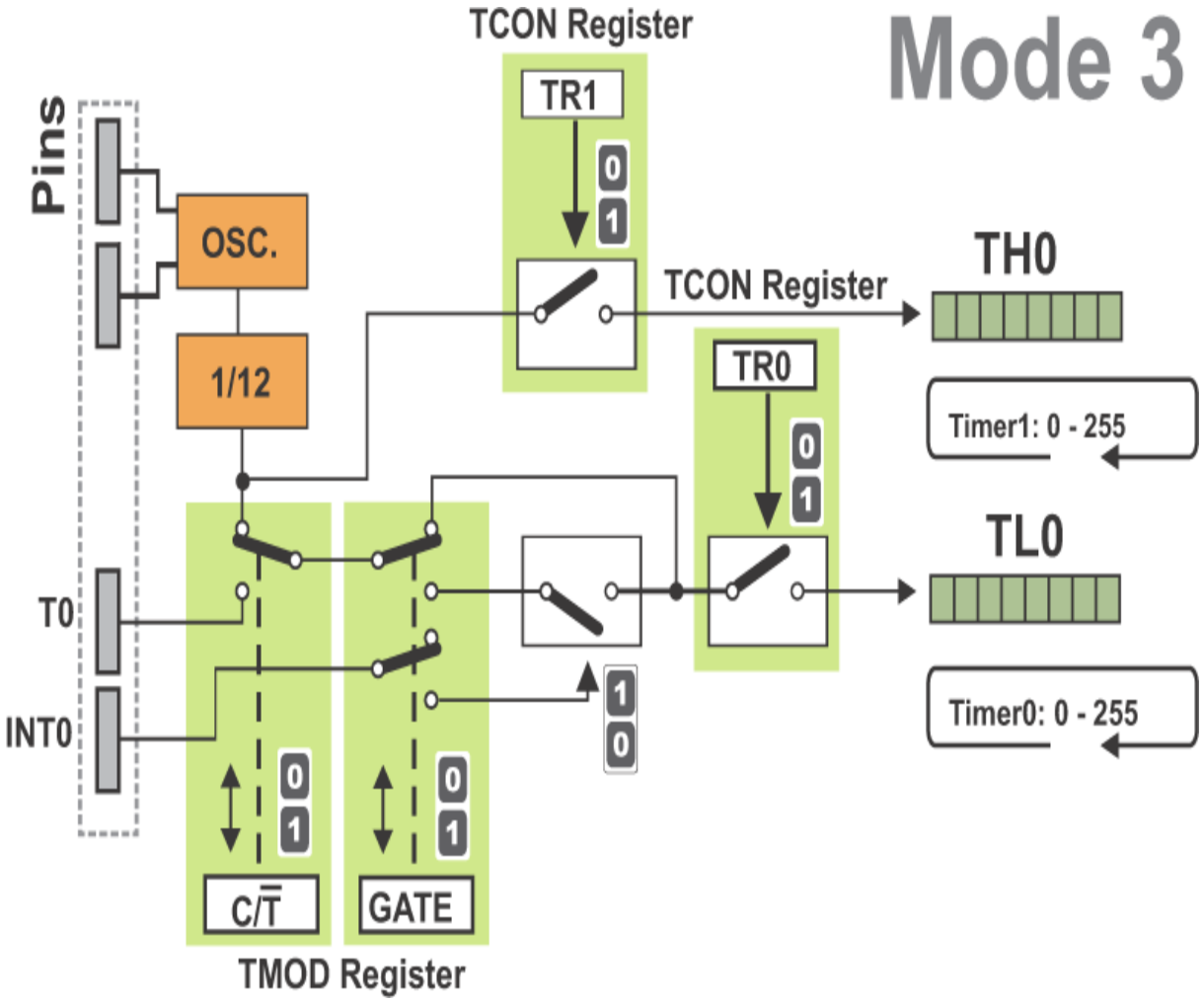


Fig 2.33: Timer Mode 3

The only application of this mode is when two timers are used and the 16-bit Timer 1 the operation of which is out of control is used as a baud rate generator.

Timer Control (TCON) Register

TCON register is also one of the registers whose bits are directly in control of timer operation. Only 4 bits of this register are used for this purpose, while rest of them is used for

interrupt control to be discussed later.

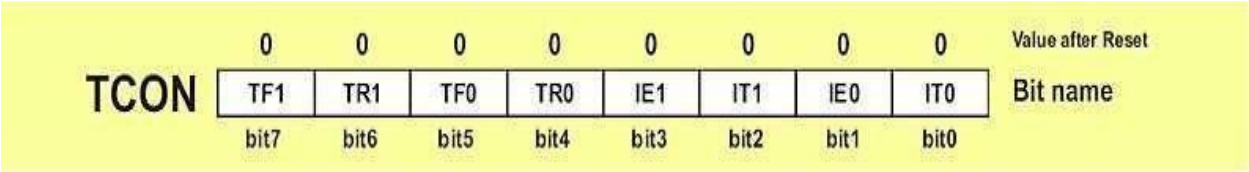


Fig 2.34: TCON

- TF1 bit is automatically set on the Timer 1 overflow.
- TR1 bit enables the Timer 1.
  - 1 - Timer 1 is enabled.
  - 0 - Timer 1 is disabled.
- TF0 bit is automatically set on the Timer 0 overflow.
- TR0 bit enables the timer 0.
  - 1 - Timer 0 is enabled.
  - 0 - Timer 0 is disabled.

How to use the Timer 0 ?

In order to use timer 0, it is first necessary to select it and configure the mode of its operation.Bits of the TMOD register are in control of it:

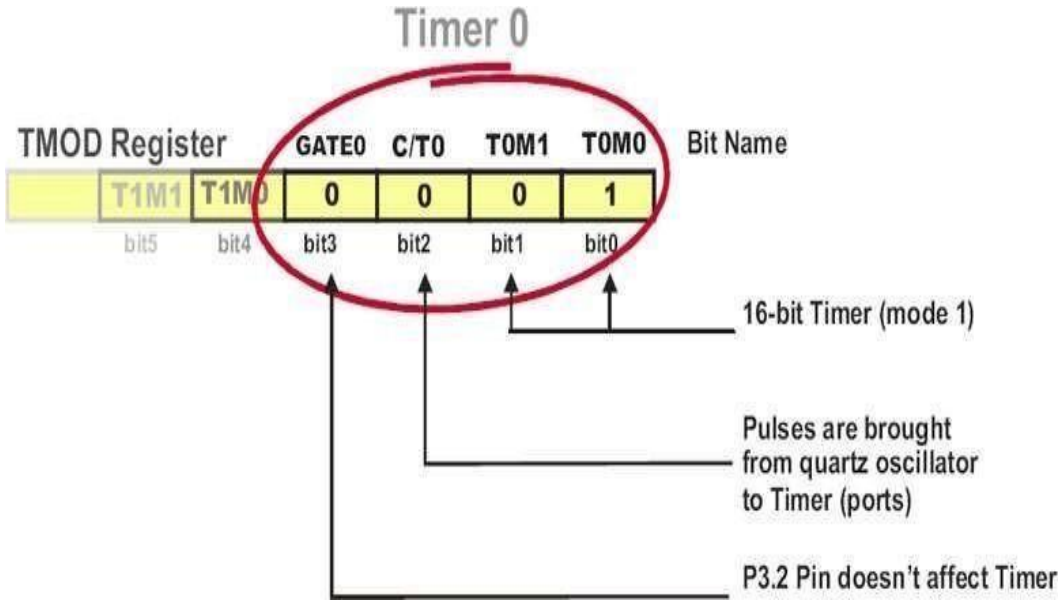


Fig 2.35: Timer 0 configuration

Referring to figure above, the timer 0 operates in mode 1 and counts pulses generated by internalclock the frequency of which is equal to 1/12 the quartz frequency.
Turn on the timer:

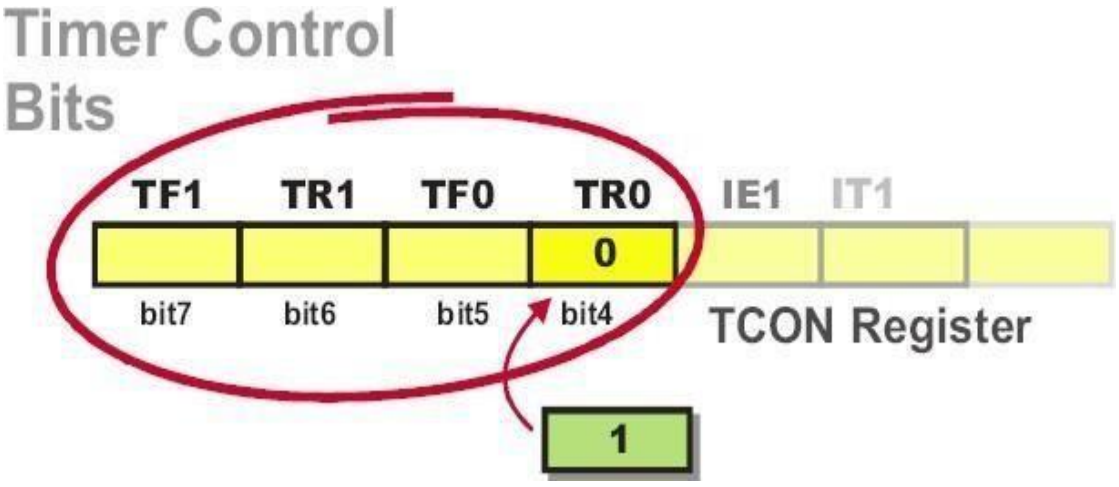


Fig 2.36: TCON control bits

The TR0 bit is set and the timer starts operation. If the quartz crystal with frequency of 12MHz isembedded then its contents will be incremented every microsecond. After 65.536 microseconds, the both registers the timer consists of will be loaded. The microcontroller automatically clears them and the timer keeps on repeating procedure from the beginning until the TR0 bit value is logic zero (0).

How to 'read' a timer?

Depending on application, it is necessary either to read a number stored in the timer registers or to register the moment they have been cleared.

- It is extremely simple to read a timer by using only one register configured in mode 2 or 3. It is sufficient to read its state at any moment. That's all!

- It is somehow complicated to read a timer configured to operate in mode 2. Suppose the lower byte is read first (TL0), then the higher byte (TH0). The result is:

TH0 = 15 TL0 = 255

Everything seems to be ok, but the current state of the register at the moment of reading

was:TH0 = 14 TL0 = 255

In case of negligence, such an error in counting (255 pulses) may occur for not so obvious but quite logical reason. The lower byte is correctly read (255), but at the moment the program counter was about to read the higher byte TH0, an overflow occurred and the contents of both registers have been changed (TH0: 14→15, TL0: 255→0). This problem has a simple solution.The higher byte should be read first, then the lower byte and once again the higher byte. If the number stored in the higher byte is different then thissequence

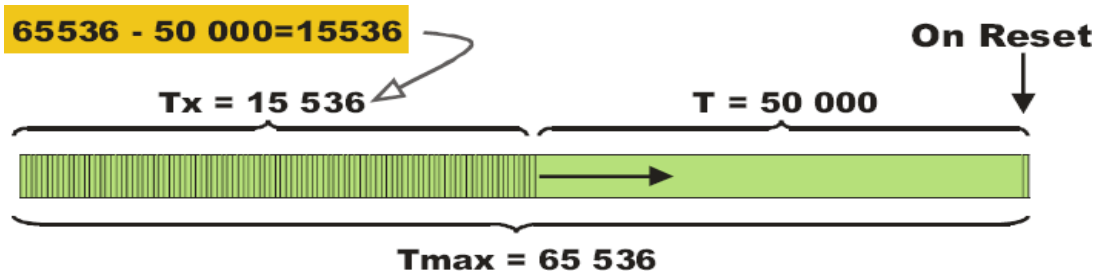
should be repeated. It's about ashort loop consisting of only 3 instructions in the program.

There is another solution as well. It is sufficient to simply turn the timer off while reading is going on (the TR0 bit of the TCON register should be cleared), and turn it on again after readingis finished.

Timer 0 Overflow Detection

Usually, there is no need to constantly read timer registers. It is sufficient to register the moment they are cleared, i.e. when counting starts from 0. This condition is called an overflow. When it occurrs, the TF0 bit of the TCON register will be automatically set. The state of this bit can be constantly checked from within the program or by enabling an interrupt which will stop the mainprogram execution when this bit is set. Suppose it is necessary to provide a program delay of
0.05 seconds (50 000 machine cycles), i.e. time when the program seems to be

stopped:First a number to be written to the timer registers should be calculated:



Then it should be written to the timer registers TH0 and TL0:

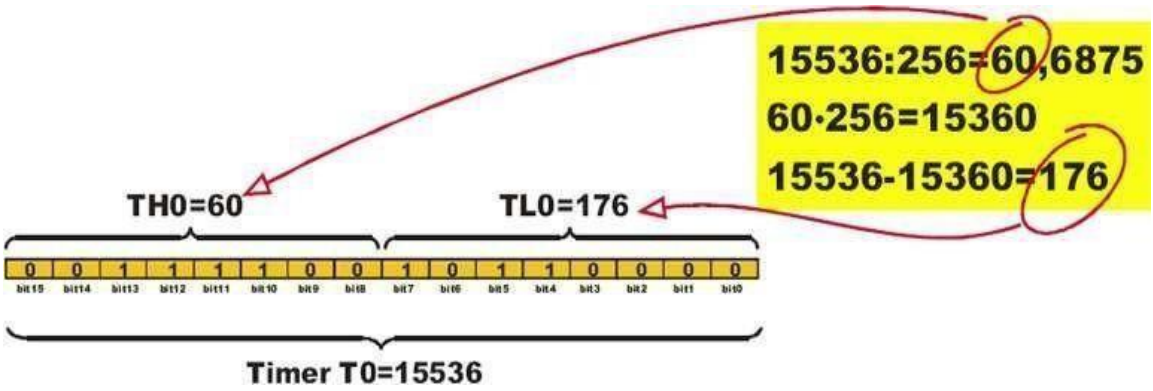


Fig 2.37: Timer 0 -TLO & THO count write

When enabled, the timer will resume counting from this number. The state of the TF0 bit, i.e. whether it is set, is checked from within the program. It happens at the moment of overflow, i.e. after exactly 50.000 machine cycles or 0.05 seconds.

How to measure pulse duration?

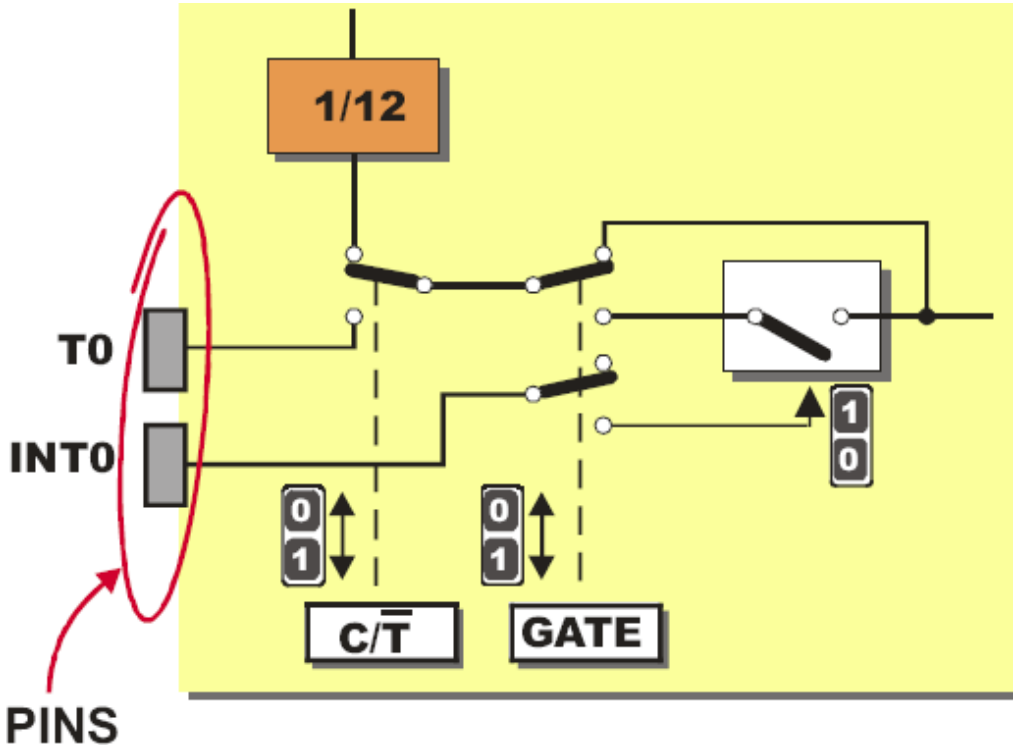


Fig 2.38: Measure Pulse duration

Suppose it is necessary to measure the duration of an operation, for example how long a device has been turned on? Look again at the figure illustrating the timer and pay attention to the function of the GATE0 bit of the TMOD register. If it is cleared then the state of the P3.2 pin doesn't affect timer operation. If GATE0 = 1 the timer will operate until the pin P3.2 is cleared. Accordingly, if this pin is supplied with 5V through some external switch at the moment the device is being turned on, the timer will measure duration of its operation, which actually was the objective.

How to count up pulses?

Similarly to the previous example, the answer to this question again lies in the TCON register. This time it's about the C/T0 bit. If the bit is cleared the timer counts pulses generated by the internal oscillator, i.e. measures the time passed. If the bit is set, the timer input is provided withpulses from the P3.4 pin (T0). Since these pulses are not always of the same width, the timer cannot be used for time measurement and is turned into a counter, therefore. The highest frequency that could be measured by such a counter is 1/24 frequency of used quartz-crystal.

Timer 1

Timer 1 is identical to timer 0, except for mode 3 which is a hold-count mode. It means that theyhave the same function, their operation is controlled by the same registers TMOD and TCON and both of them can operate in one out of 4 different modes.

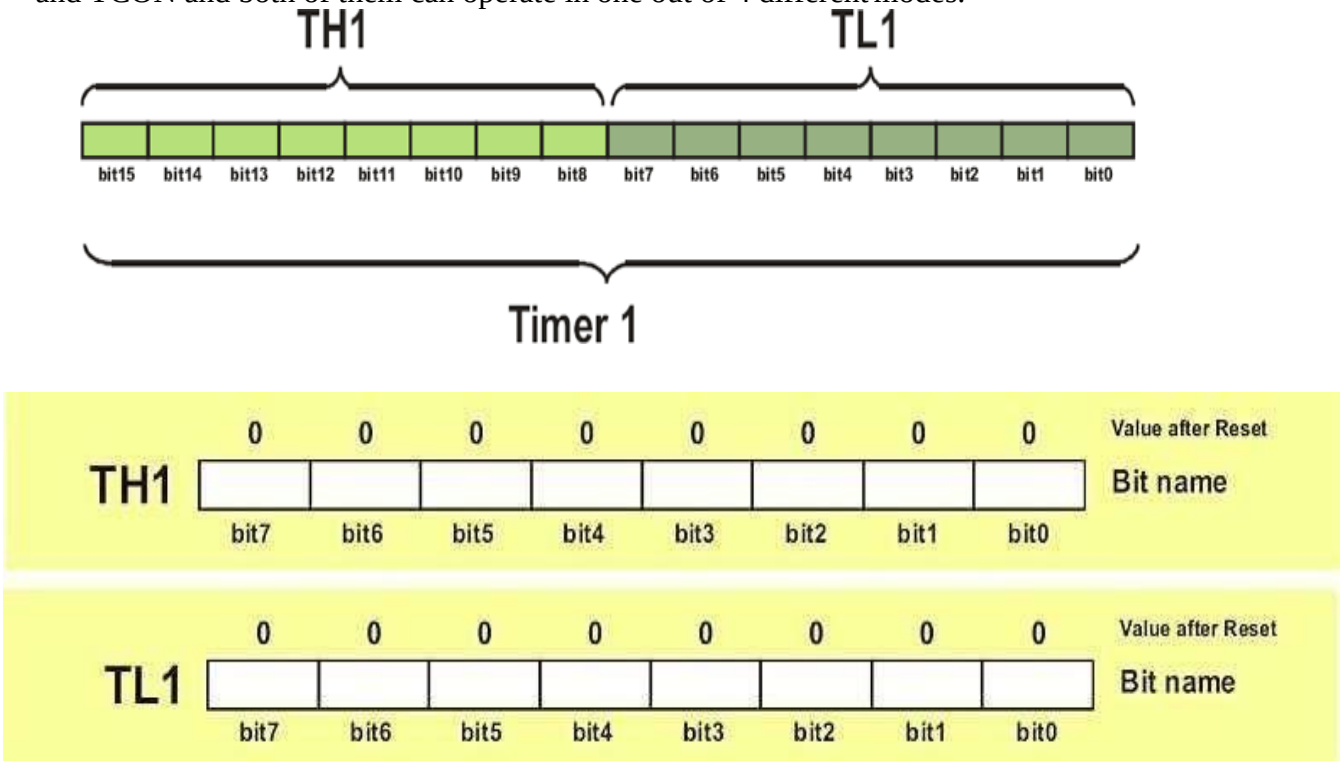


Fig 2.39: timer 1

7. Serial Communication

One of the microcontroller features making it so powerful is an integrated UART, better known as a serial port. It is a full-duplex port, thus being able to transmit and receive data simultaneously and at different baud rates. Without it, serial data send and receive would be an enormously complicated part of the program in which the pin state is constantly changed and checked at regular intervals. When using UART, all the programmer has to do is to simply selectserial port mode and baud rate. When it's done, serial data transmit is nothing but writing to the SBUF register, while data receive represents reading the same register. The microcontroller takescare of not making any error during data transmission.

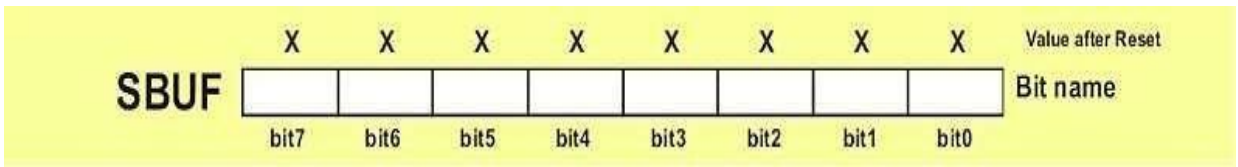


Fig 2.40: SBUF

Serial port must be configured prior to being used. In other words, it is necessary to determinehow many bits is contained in one serial ―word‖, baud rate and synchronization clock source. The whole process is in control of the bits of the SCON register (Serial Control).

Serial Port Control (SCON) Register

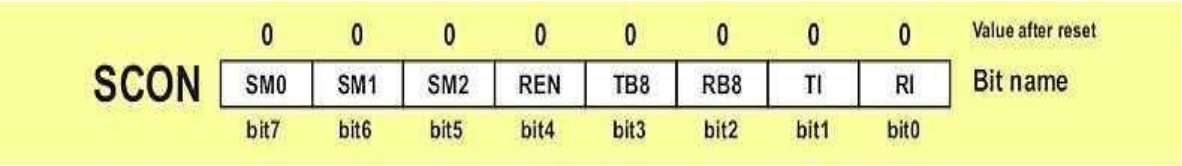


Fig 2.41: SCON

- SM0 - Serial port mode bit 0 is used for serial port modeselection.
- SM1 - Serial port mode bit 1.
- SM2 - Serial port mode 2 bit, also known as multiprocessor communication enable bit. When set, it enables multiprocessor communication in mode 2 and 3, and eventually mode 1. It should be cleared in mode 0.
- REN - Reception Enable bit enables serial reception when set. When cleared, serial reception is disabled.
- TB8 - Transmitter bit 8. Since all registers are 8-bit wide, this bit solves the problemof transmiting the 9th bit in modes 2 and 3. It is set to transmit a logic 1 in the 9th bit.
- RB8 - Receiver bit 8 or the 9th bit received in modes 2 and 3. Cleared by hardware if9th bit received is a logic 0. Set by hardware if 9th bit received is a logic 1.
- TI - Transmit Interrupt flag is automatically set at the moment the last bit of one byte is sent. It's a signal to the processor that the line is available for a new byte transmite. It must be cleared from within the software.
- RI - Receive Interrupt flag is automatically set upon one byte receive. It signals that byte is received and should be read quickly prior to being replaced by a new data. This bit is also cleared from within the software.

As seen, serial port mode is selected by combining the SM0 and SM2 bits:

SM0	SM1	Mode	Description	Baud Rate
0	0	0	8-bit Shift Register	1/12 the quartz frequency
0	1	1	8-bit UART	Determined by the timer 1
1	0	2	9-bit UART	1/32 the quartz frequency (1/64 the quartz frequency)
1	1	3	9-bit UART	Determined by the timer 1

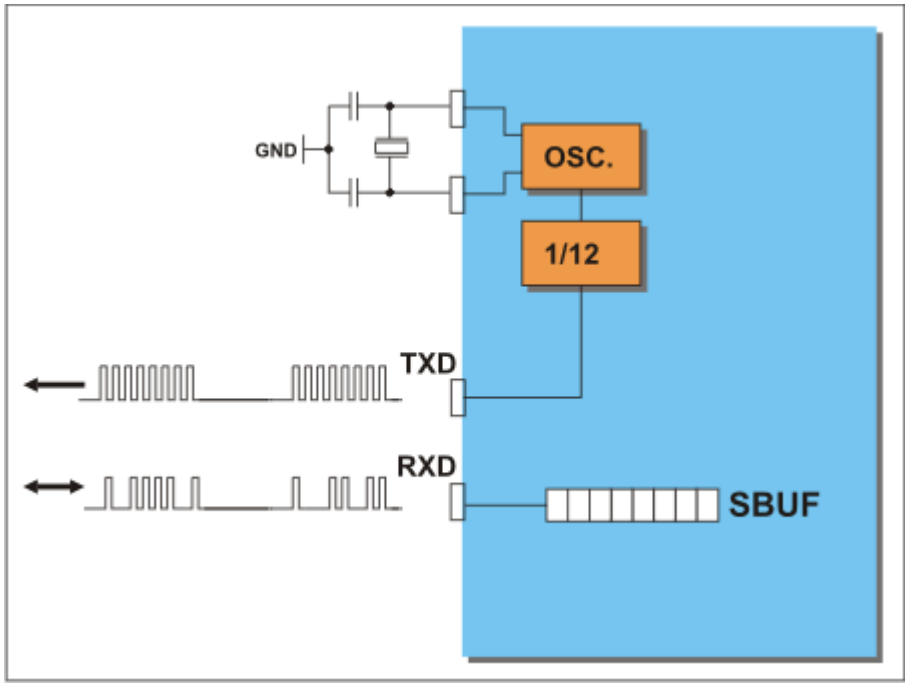


Fig 2.42: TXD , RXD

In mode 0, serial data are transmitted and received through the RXD pin, while the TXD pin output clocks. The bout rate is fixed at 1/12 the oscillator frequency. On transmit, the least significant bit (LSB bit) is sent/received first.

TRANSMIT - Data transmit is initiated by writing data to the SBUF register. In fact, this process starts after any instruction being performed upon this register. When all 8 bits have beensent, the TI bit of the SCON register is automatically set.

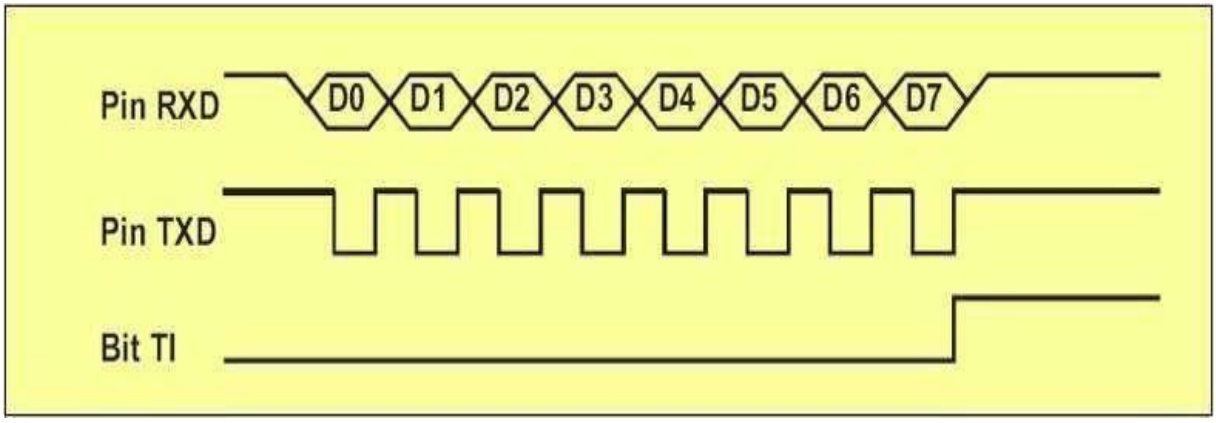


Fig 2.43: TXD , RXD status- TI-mode 0

RECEIVE - Data receive through the RXD pin starts upon the two following conditions are met: bit REN=1 and RI=0 (both of them are stored in the SCON register). When all 8 bits havebeen received, the RI bit of the SCON register is automatically set indicating that one byte receive is complete.

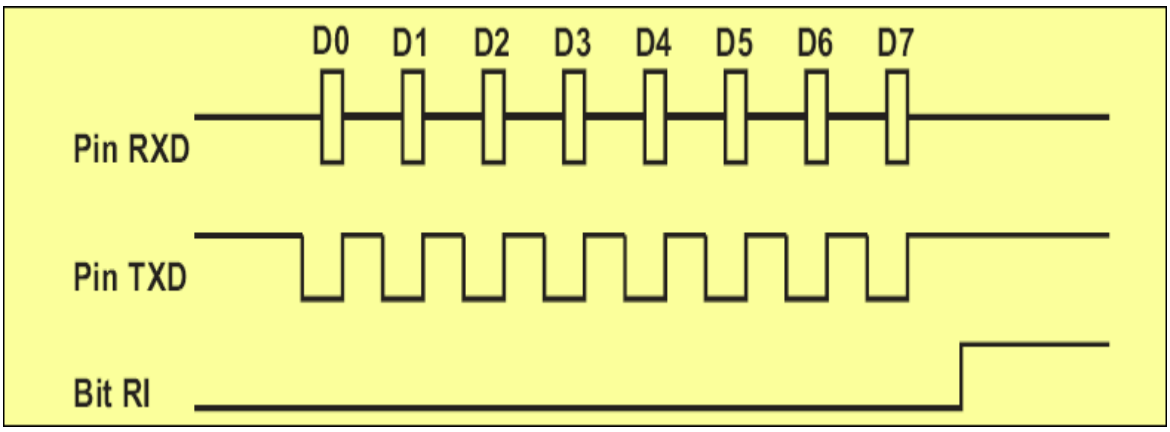


Fig 2.44: TXD , RXD-RI-mode 0

Since there are no START and STOP bits or any other bit except data sent from the SBUF register in the pulse sequence, this mode is mainly used when the distance between devices is short, noise is minimized and operating speed is of importance. A typical example is I/O port expansion by adding a cheap IC (shift registers 74HC595, 74HC597 and similar).

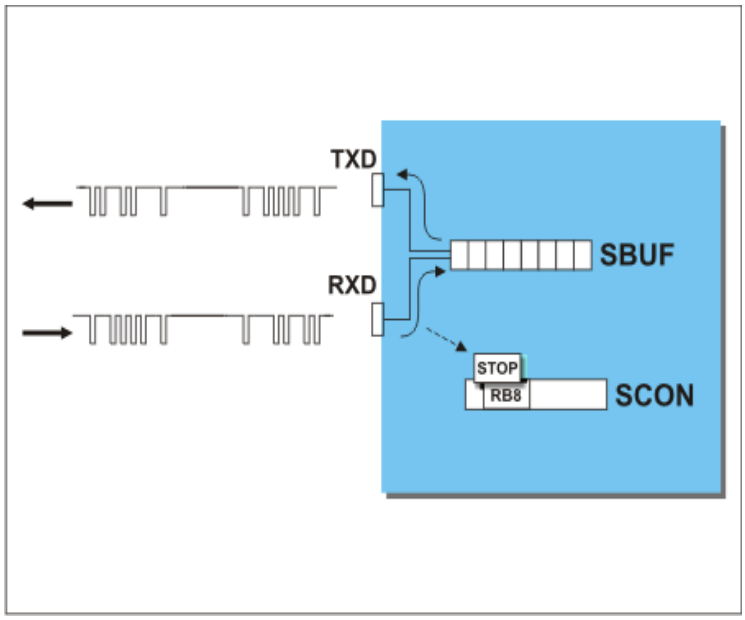


Fig 2.45: TXD , RXD, SBUF,SCON-mode 1

In mode 1, 10 bits are transmitted through the TXD pin or received through the RXD pin in thefollowing manner: a START bit (always 0), 8 data bits (LSB first) and a STOP bit (always 1).
The START bit is only used to initiate data receive, while the STOP bit is automatically writtento the RB8 bit of the SCON register.

TRANSMIT - Data transmit is initiated by writing data to the SBUF register. End of datatransmission is indicated by setting the TI bit of the SCON register.

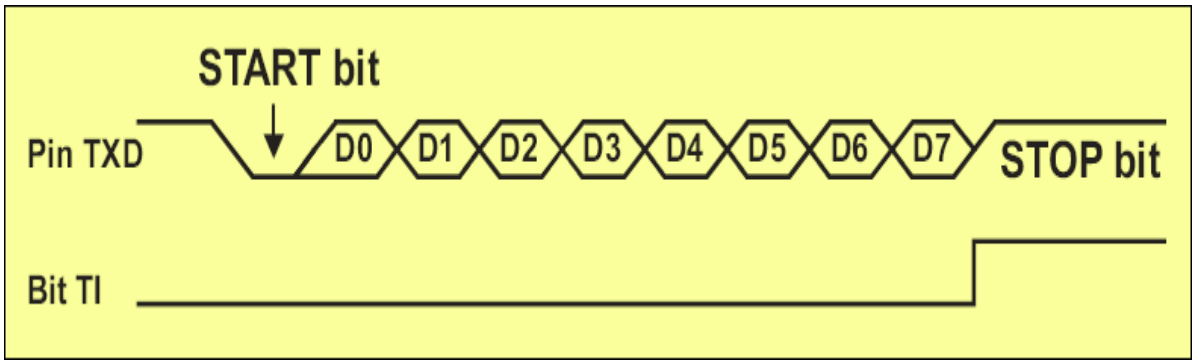


Fig 2.46: TXD , TI-mode 1

RECEIVE - The START bit (logic zero (0)) on the RXD pin initiates data receive. The following two conditions must be met: bit REN=1 and bit RI=0. Both of them are stored in theSCON register. The RI bit is automatically set upon data reception is complete.

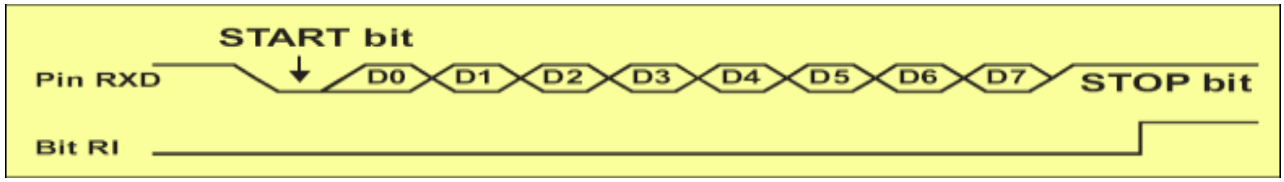


Fig 2.47: RXD-RI-mode 1

The Baud rate in this mode is determined by the timer 1 overflow.

Mode 2

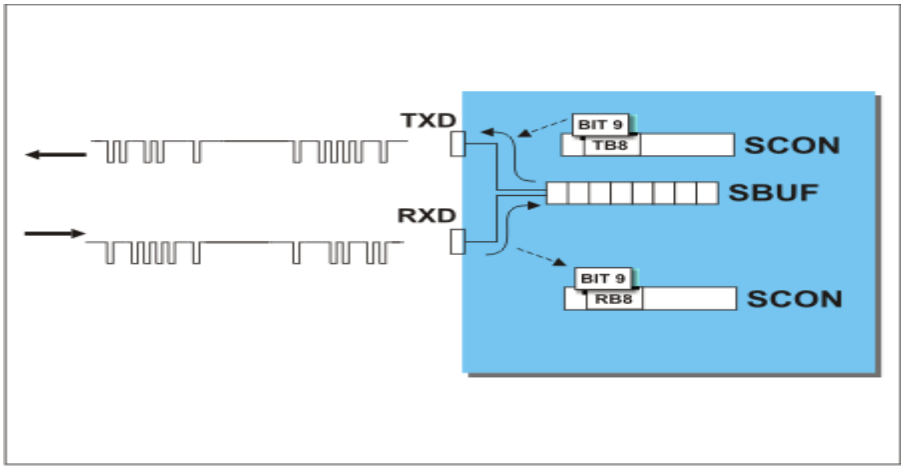


Fig 2.48: TXD , RXD-mode 2

In mode 2, 11 bits are transmitted through the TXD pin or received through the RXD pin: a START bit (always 0), 8 data bits (LSB first), a programmable 9th data bit and a STOP bit (always 1). On transmit, the 9th data bit is actually the TB8 bit of the SCON register. This bit usually has a function of parity bit. On receive, the 9th data bit goes into the RB8 bit of the sameregister (SCON).The baud rate is either 1/32 or 1/64 the oscillator frequency.

TRANSMIT - Data transmit is initiated by writing data to the SBUF register. End of datatransmission is indicated by setting the TI bit of the SCON register.

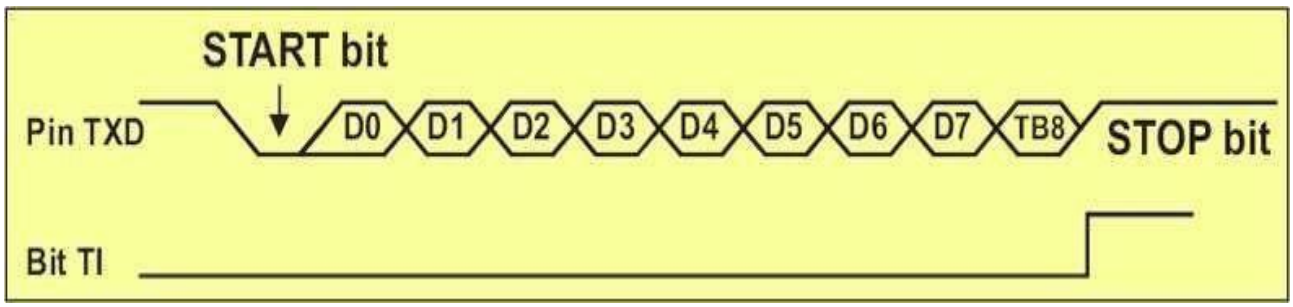


Fig 2.49: mode 2

RECEIVE - The START bit (logic zero (0)) on the RXD pin initiates data receive. The following two conditions must be met: bit REN=1 and bit RI=0. Both of them are stored in theSCON register. The RI bit is automatically set upon data reception is complete.

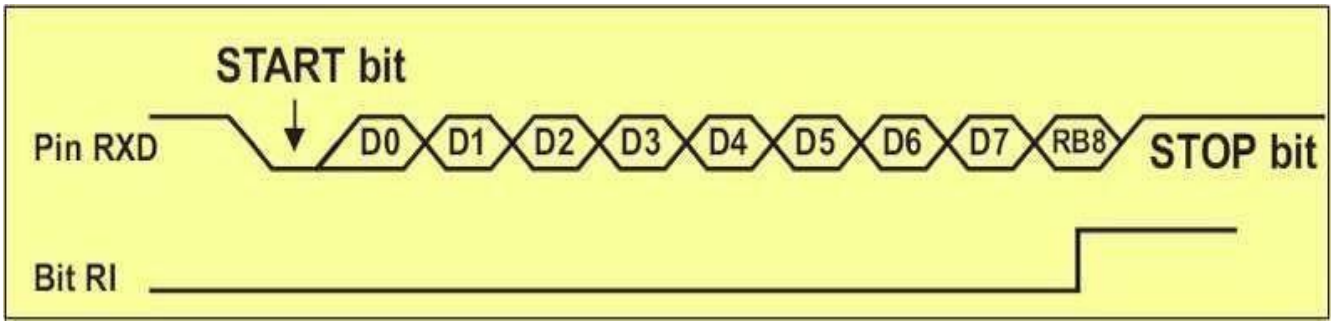


Fig 2.50: mode 2

Mode 3 is the same as Mode 2 in all respects except the baud rate. The baud rate in Mode 3 is variable.

Baud Rate

Baud Rate is a number of sent/received bits per second. In case the UART is used, baud rate depends on: selected mode, oscillator frequency and in some cases on the state of the SMOD bitof the SCON register. All the necessary formulas are specified in the table:

	BAUD RATE	BITSMOD
Mode 0	Fosc. / 12	
Mode 1	1 Fosc. 16 12 (256-TH1)	BitSMOD
Mode 2	Fosc. / 32 Fosc. / 64	1 0
Mode 3	1 Fosc. 16 12 (256-TH1)	

Timer 1 as a clock generator

Baud Rate	Fosc. (MHz)				Bit SMOD
	11.0592	12	14.7456	16 20	
150	40 h	30 h	00 h	0	
300	A0 h	98 h	80 h	75 h 52 h 0	
600	D0 h	CC h C0 h		BB h A9 h 0	
1200	E8 h	E6 h E0 h		DE h D5 h 0	
2400	F4 h	F3 h F0 h		EF h EA h 0	

4800		F3 h EF h	EF h	1
4800	FA h	F8 h	F5 h	0
9600	FD h	FC h		0
9600			F5 h	1
19200	FD h	FC h		1
38400		FE h		1
76800		FF h		1

Multiprocessor Communication

As you may know, additional 9th data bit is a part of message in mode 2 and 3. It can be used forchecking data via parity bit. Another useful application of this bit is in communication between two or more microcontrollers, i.e. multiprocessor communication. This feature is enabled by setting the SM2 bit of the SCON register. As a result, after receiving the STOP bit, indicating end of the message, the serial port interrupt will be generated only if the bit RB8 = 1 (the 9th bit).

This is how it looks like in practice:

Suppose there are several microcontrollers sharing the same interface. Each of them has its own address. An address byte differs from a data byte because it has the 9th bit set (1), while this bitis cleared (0) in a data byte. When the microcontroller A (master) wants to transmit a block of data to one of several slaves, it first sends out an address byte which identifies the target slave. An address byte will generate an interrupt in all slaves so that they can examine the received byteand check whether it matches their address.

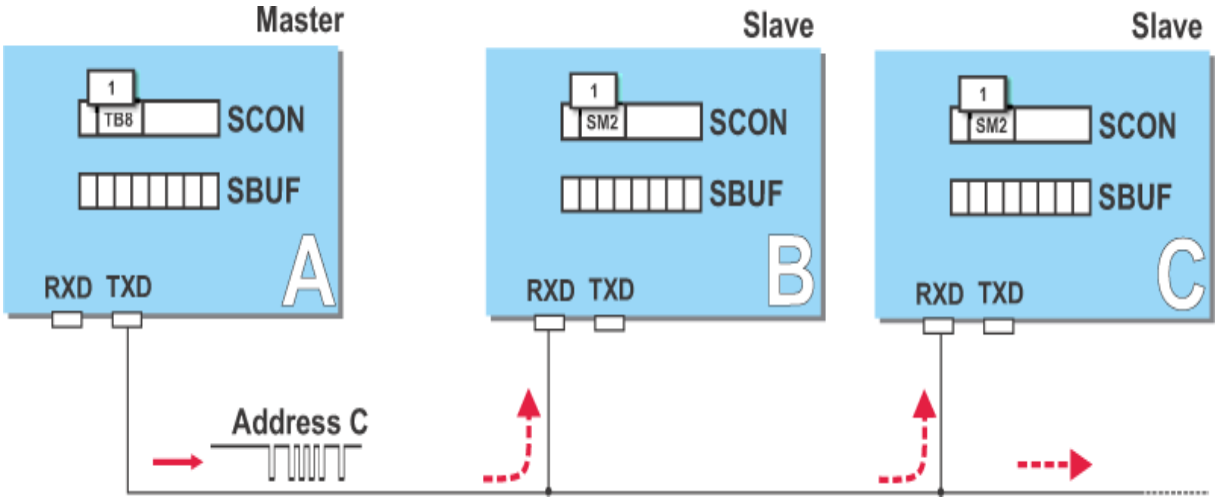


Fig 2.51: multiprocessor communication

Of course, only one of them will match the address and immediately clear the SM2 bit of the SCON register and prepare to receive the data byte to come. Other slaves not being addressed leave their SM2 bit set ignoring the coming data bytes.

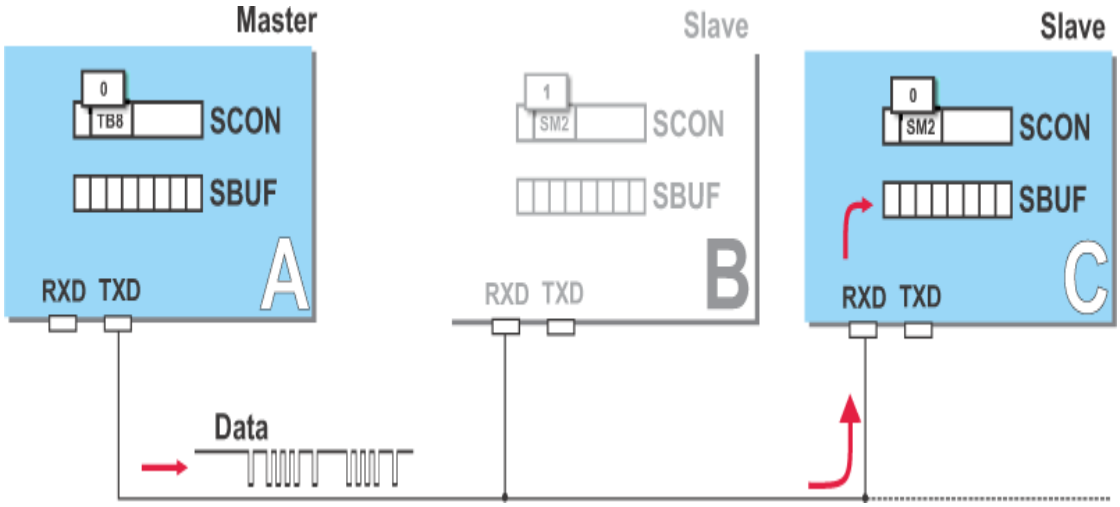


Fig 2.52: multiprocessor communication

QUESTION BANK

PART A

1. What are the addressing modes of 8051.
2. Differentiate microcontroller and microprocessor.
3. Write short notes on interrupts.
4. Write briefly about the timer of 8051.
5. What is an SFR.
6. List the SFR in 8051.
7. Write an assembly language program to transfer
8. a.10 data from internal to external
   b.10 data from external to internal
9. Explain how to interface I/O devices to 8051.
10. Write a program to find a square of a number using look up table.10.Write a program to find the given number is odd or even.
11. Write a program to generate a square wave of 1ms using timer.
12. List the bits of PSW.
13. What are the different ranges of jump.
14. Classify jump instruction
15. Write about stack
16. On reset the value of SP is_____, I/O ports are configured as_____.
17. Write about EA pin of 8051.
18. Draw one machine cycle of 8051.
19. What is ALE?
20. The internal RAM size is_____and the internal ROM size is_____.

PART B

1. With neat diagram explain the architecture of 8051.
2. Classify the instruction set of8051 and explain the instruction with suitable examples.
3. Write in detail how serial communication is carried out in 8051.
4. Explain in detail about timers in 8051 microcontroller
5. Explain the interrupts of 8051 microcontroller
6. Write the following programs
   a. programs using arithmetic and logical instruction
   b. Programs to convert hexa to ascii and ascii to hexa
   c. Programs using program transfer instructions. d.Programs using I/O ports
7. Explain the following instructions with example
   a. movc a,@a+dptr b. movx @r0,a c. JBC b,radd
   d. XCHD A,@Rp e. Swap A

TEXT / REFERENCE BOOKS

1. Ramesh Goankar, "Microprocessor architecture programming and applications with 8085 / 8088", 5th Edition, Penram International Publishing, 2002.

2. Mazidi & McKinlay, “The 8051 Microcontroller and Embedded Systems using Assembly and C”, PHI, 2007.

3. MykePredko, “Programming and Customizing the 8051 Micro-controller”, Tata McGraw-Hill edition, 2007.

4. R A Gaonkar, “Fundamentals of Microcontrollers and Applications in Embedded Systems (with the PIC18 Microcontroller Family)”, Penram Publishing India, 2007

. 5. Kenneth Ayala ,”The 8051 Microcontroller”, 3rd Edition, Thomson Delmar Learning, 2004.

6. Kenneth J. Ayala, Dhananjay V. Gadre, “The 8051 Microcontroller & Embedded Systems Using Assembly and C”, Cengage Learning India Publication, 2007.

7. Ajay V Deshmukh, “Microcontrollers: Theory and Applications”, Tata McGraw-Hill, 2005

. 8. Raj Kamal, “Embedded Systems Architecture, Programming, and Design”. (2/e), Tata McGraw Hill, 2008.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF ELECTRICAL AND ELECTRONICS ENGINEERING

DEPARTMENT OF ELECTRONICS AND INSTRUMENTATION ENGINEERING

UNIT – III MICROPROCESSORS, MICROCONTROLLERS AND EMBEDDED SYSTEMS: SEIA1504

UNIT 3: PIC MICROCONTROLLER AND INSTRUCTION SET

PIC Microcontrollers and Instruction Set: PIC Micro-controllers - overview; features, PIC-18Fxxx architecture, file selection register, Memory organization, Addressing modes, Instruction set, Interrupt handling. PIC-18Fxxx - Reset, low power operations, oscillator connections, I/O ports - serial; parallel, Timers, Interrupts, ADC.

3.1 PIC ARCHITECTURE:

- High performance RISC CPU.
- ONLY 35 simple word instructions.
- All single cycle instructions except for program branches which are two cycles.
- Operating speed: clock input (200MHz), instruction cycle (200nS).
- Up to 368×8bit of RAM (data memory), 256×8 of EEPROM (data memory), 8k×14 of flash memory.
- Pin out compatible to PIC 1 6C74B, PIC 1 6C76, PIC 1 6C77.
- Eight level deep hardware stack.
- Interrupt capability (up to 14 sources).
- Different types of addressing modes (direct, Indirect, relative addressing modes).
- Power on Reset (POR).
- Power-Up Timer (PWRT) and oscillator start-up timer.
- Low power- high speed CMOS flash/EEPROM.
- Fully static design.

- Wide operating voltage range (2.0 – 5.56)volts.
- High sink/source current (25mA).
- Commercial, industrial and extended temperature ranges.
- Low power consumption (<0.6mA typical @3v-4MHz, 20µA typical @3v-32MHz and <1 A typical standby).

Peripheral Features

- Timer 0: 8 bit timer/counter with pre-scalar.
- Timer 1:16 bit timer/counter with pre-scalar.
- Timer 2: 8 bit timer/counter with 8 bit period registers with pre-scalar and post-scalar.
- Two Capture (16bit/12.5nS), Compare (16 bit/200nS), Pulse Width Modules (10bit).
- 1 0bit multi-channel A/D converter
- Synchronous Serial Port (SSP) with SPI (master code) and I2C (master/slave).
- Universal Synchronous Asynchronous Receiver Transmitter (USART) with 9 bit address detection.
- Parallel Slave Port (PSP) 8 bit wide with external RD, WR and CS controls (40/46pin).
- Brown Out circuitry for Brown-Out Reset (BOR).

Key Features

- Maximum operating frequency is 20MHz.
- Flash program memory (14 bit words), 8KB.
- Data memory (bytes) is 368.

- EEPROM data memory (bytes) is 256.
- 5 input/output ports.
- 3 timers and 2 CCP modules.
- 2 serial communication ports (MSSP, USART).
- PSP parallel communication port and 10bit A/D module (8 channels)

Analog Features

- 1 0bit, up to 8 channel A/D converter.
- Brown Out Reset function.
- Analog comparator module.

Special Features

- 100000 times erase/write cycle enhanced memory.
- 1000000 times erase/write cycle data EEPROM memory.
- Self programmable under software control.
- In-circuit serial programming and in-circuit debugging capability.
- Single 5V,DC supply for circuit serial programming
- WDT with its own RC oscillator for reliable operation.
- Programmable code protection.
- Power saving sleep modes.
- Selectable oscillator options.

2. Architecture of PIC

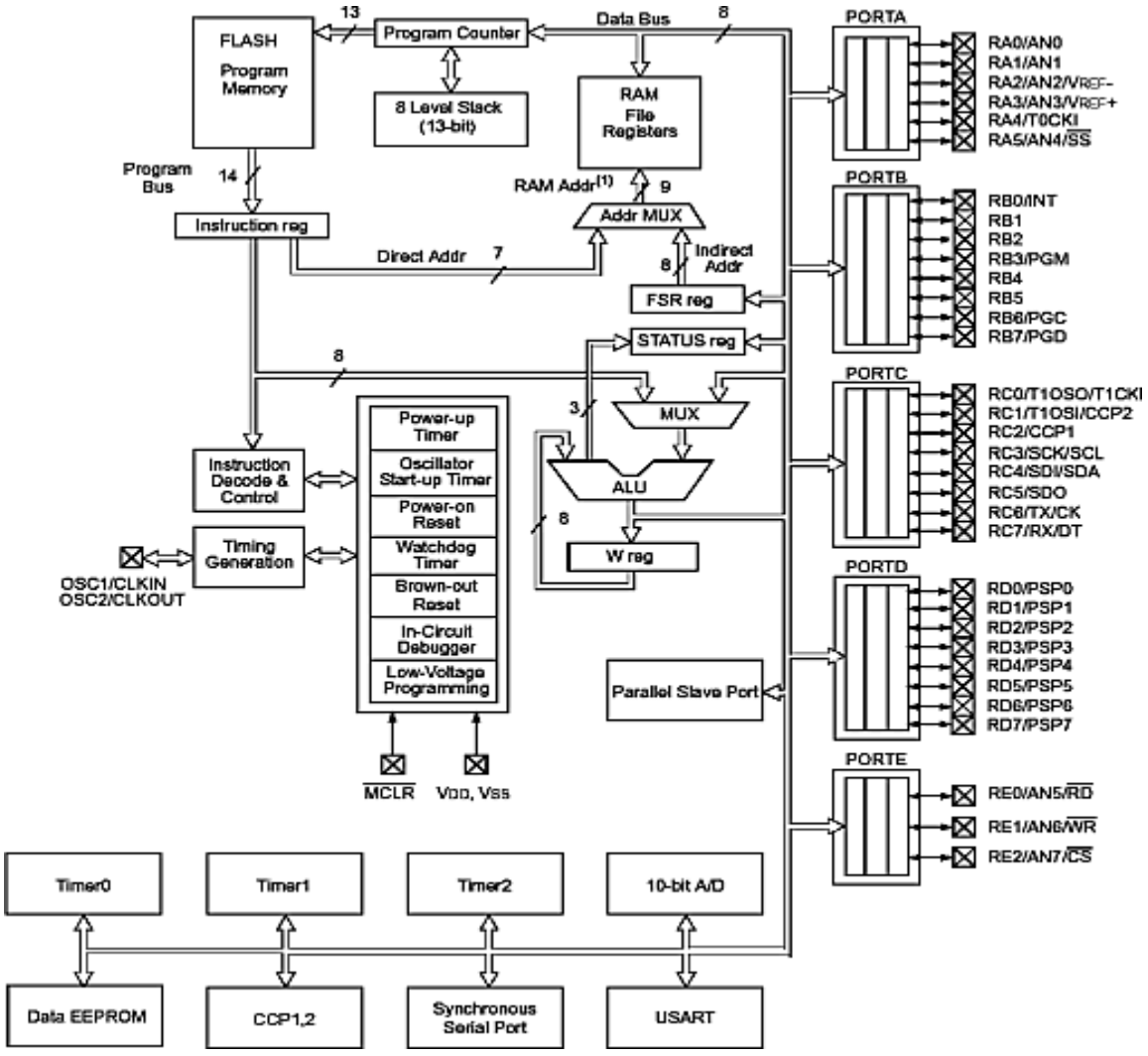


Fig.3.1 Block Diagram of PIC

The function of CPU in PIC is same as a normal microcontroller CPU. A PIC, CPU consists of several sub units such as instruction decoder, ALU, accumulator, control unit, etc. The CPU in PIC normally supports Reduced Instruction Set Computer (RISC) architecture. RISC design is based on the premise that most of the instructions. The computer decodes and executes are simple. As a result, RISC architecture limits the number of instructions. Execution Time is less.

MEMORY

The memory in a PIC chip used to store the data and programs (temporary or permanently). PIC also has certain amount of memory space for RAM, ROM, and EEPROM and other flash memory, etc. ROM memory is used for permanent storage memory. The contents in the EEPROM changes during run time and at that time it acts like a RAM memory. But the difference is after the power goes off, the data remains in this ROM chip. This is the one of the special advantages of EEPROM. In the PIC chip the function of EPROM is to store the values created during the runtime.RAM memory is the one of the complex memory module in a PIC chip. This memory associated with various type of registers (special function registers and general purpose registers) and memory BANK modules (BANK 0, BANK 1, etc.). Once the power goes off, the contents in the RAM will be cleared. As like normal microcontrollers, the RAM memory is used to store temporary data and provide immediate results. The flash memory is a special type of memory where READ, WRITE, and ERASE operations can be done many times.

3.REGISTERS

Information is stored in a CPU memory location called a register. Registers can be thought of as the CPUs tiny scratchpad, temporarily storing instructions or data. Registers basically classified into the following.

General Purpose Register (GPR)

A general purpose register (or processor register) is a small storage area available on a CPU whose contents can be accessed more quickly than other storage that available on PIC. A general purpose register can store both data addresses simultaneously.

Special Function registers (SFR)

These are also a part of RAM memory locations. As compared to GPR, their purpose is predetermined during the manufacturing time and cannot be changed by the user. It is only for special dedicated functions.

4.INTERRUPTS

Interrupt is the temporary delay in a running program. These delays stop the current execution for a particular interval. This interval/delay is usually called as interrupt. When an interrupt request arrives into a current execution program, then it stops its regular execution. Interrupt can be performed by externally (hardware interrupt) or internally (by using software).

BUS

BUS is the communication or data transmission/reception path in a microcontroller unit. In a normal microcontroller chip, two types of buses are normally available.

Data bus

Data bus is used for memory addressing. The function of data bus is interfacing all the circuitry components inside the PIC chip.

Address bus

Address bus mostly used for memory addressing. The function of address bus is to transmit the address from the CPU to memory locations.

USART or UART

These ports are used for the transmission (TX) and reception (RX) of data. These transmissions possible with help of various digital data transceiver modules like RF, IR, Bluetooth, etc. This is the one of the simplest way to communicate the PIC chip with other devices.

5.OSCILLATORS

Oscillator unit basically an oscillation/clock generating circuit which is used for providing proper clock pulses to the PIC chip. This clock pulses also helps the timing and counting applications. A PIC chip normally use various types of clock generators. According to the application and the type of PIC used, the oscillators and its frequencies may vary. RC (Resistor-Capacitor), LC (Inductor-Capacitor), RLC (Resistor-Inductor-capacitor), crystal oscillators, etc are the normal oscillators used with A PIC chip.

Stack

The entire PIC chip has an area for storing the return addresses. This area or unit called Stack is used in some Peripheral interface controllers. The hardware stack is not accessible by software. But for most of the controllers, it can be easily accessible.

6.INPUT/ OUTPUT PORTS

These ports are used for the interfacing various input/output devices and memories. According to the type of PIC, the number of ports may change.

Advanced functioning blocks

These sections include various advanced features of a PIC chip. According to the type of PIC, these features may change. Various advanced features in a peripheral interface controller are power up timer, oscillator start up timer, power on reset, watch dog timer, brown out reset, in circuit debugger, low voltage programming, voltage comparator, CCP modules etc.

7. MEMORY ORGANIZATION OF PIC16F877

The memory of a PIC 1 6F877 chip is divided into 3 sections. They are

1. Program memory
2. Data memory and
3. Data EEPROM

Program memory

Program memory contains the programs that are written by the user. The program counter (PC) executes these stored commands one by one. Usually PIC1 6F877 devices have a 13 bit wide program counter that is capable of addressing 8K×14 bit program memory space. This memory is primarily used for storing the programs that are written (burned) to be used by the PIC. These devices also have 8K*14 bits of flash memory that can be electrically erasable /reprogrammed. Each time we write a new program to the controller, we must delete the old one at that time. The figure below shows the program memory map and stack.

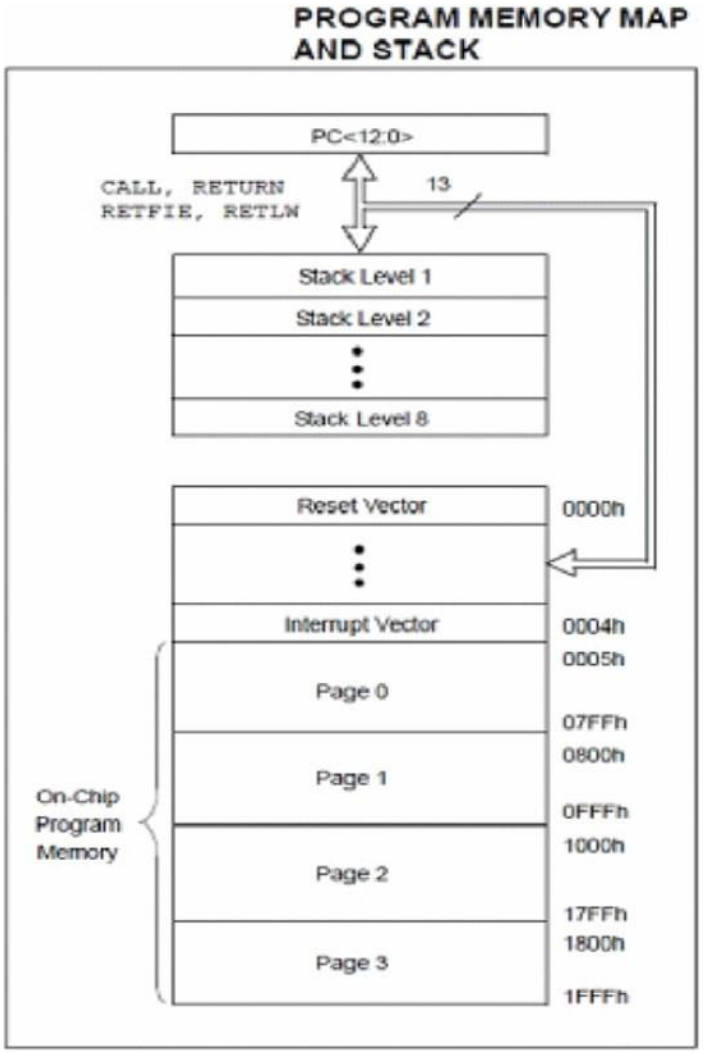


Fig.3.2 :Memory Map

PIC16f877 Program Memory

Program counters (PC) is used to keep the track of the program execution by holding the address of the current instruction. The counter is automatically incremented to the next instruction during the current instruction execution.

The PIC16F87XA family has an 8-level deep 13-bit wide hardware stack. The stack space is not a part of either program or data space and the stack pointers are not readable or writable. In the PIC microcontrollers, this is a special block of RAM memory used only for this purpose. Each time the main program execution starts at address 0000 – Reset Vector. The address 0004 is "reserved" for the "interrupt service routine" (ISR).

PIC16F87XA Data Memory Organization

The data memory of PIC1 6F877 is separated into multiple banks which contain the general purpose registers (GPR) and special function registers (SPR). According to the type of the microcontroller, these banks may vary. The PIC1 6F877 chip only has four banks (BANK 0, BANK 1, BANK 2, and BANK4). Each bank holds 128 bytes of addressable memory.

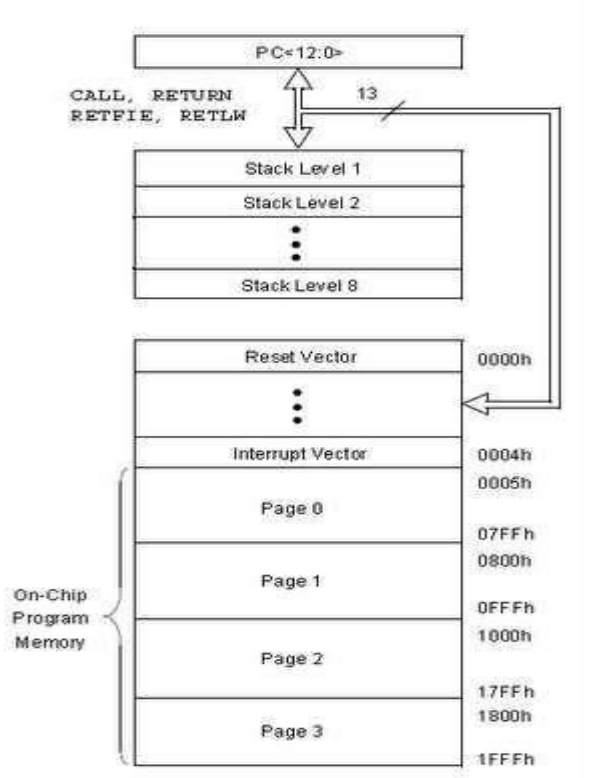


Fig.3.3 :Memory Map

The banked arrangement is necessary because there are only 7 bits are available in the instruction word for the addressing of a register, which gives only 128 addresses. The selection of the banks are determined by control bits RP1, RP0 in the STATUS registers Together the RP1, RP0 and the specified 7 bits effectively form a 9 bit address. The first 32 locations of Banks 1 and 2, and

the first 16 locations of Banks2 and 3 are reserved for the mapping of the Special Function Registers (SFR).

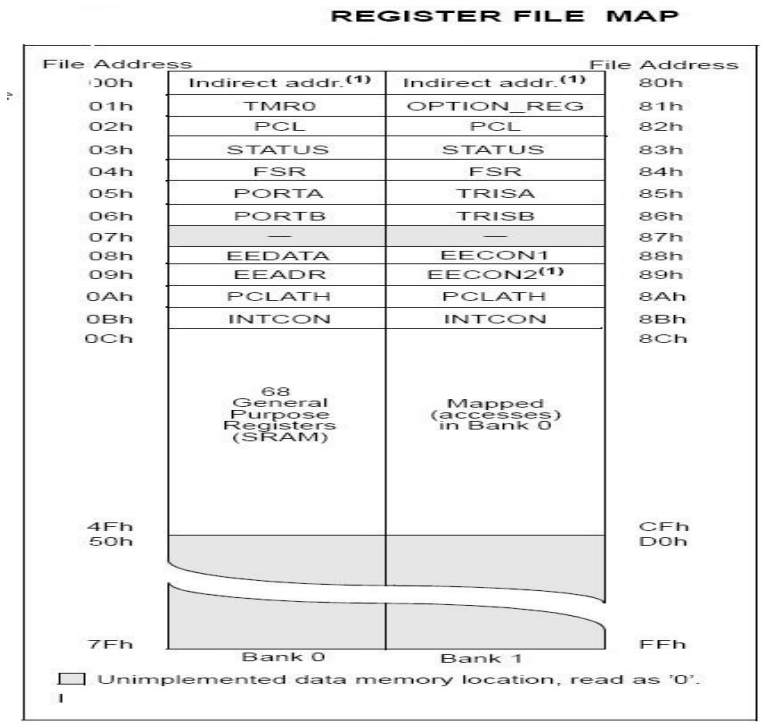


Fig.3.4 :Memory Map

Data EEPROM and FLASH

The data EEPROM and Flash program memory is readable and writable during normal operation (over the full VDD range). This memory is not directly mapped in the register file space. Instead, it is indirectly addressed through the Special Function Registers. There are six SFRs used to read and write this memory:

- EECON1
- EECON2
- EEDATA
- EEDATH
- EEADR
- EEADRH

The EEPROM data memory allows single-byte read and writes. The Flash program memory allows single-word reads and four-word block writes. Program memory write operations automatically perform an erase-before write on blocks of four words. A byte write in data EEPROM memory automatically erases the location and writes the new data (erase-beforewrite). The write time is controlled by an on-chip timer. The write/erase voltages are generated by an on-chip charge pump, rated to operate over the voltage range of the device for byte or word operations.

8. PIN DIAGRAM

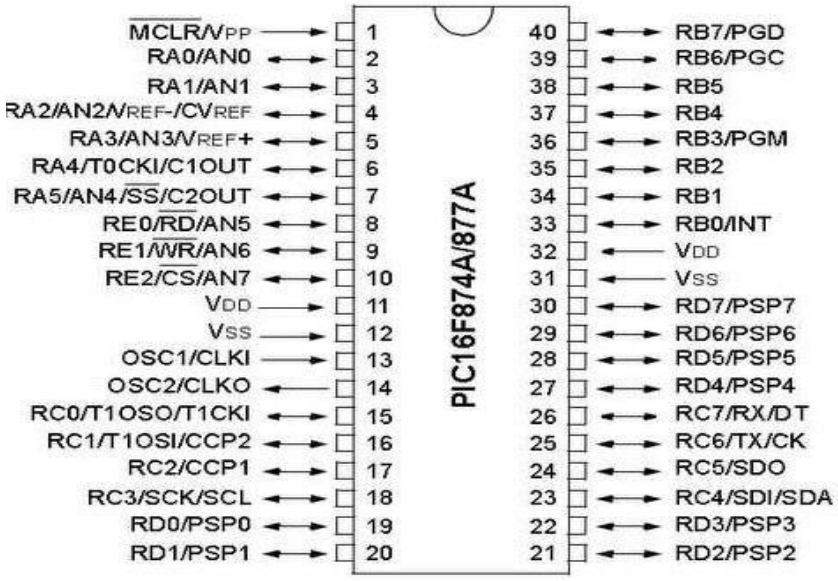


Fig.3.5:Pin details

PIC16F877 chip is available in different types of packages. According to the type of applications and usage, these packages are differentiated. The pin diagrams of a PIC16F877 chip in different packages. PIC1 6F877 has 5 basic input/output ports. They are usually denoted by PORT A (R A), PORT B (RB), PORT C (RC), PORT D (RD), and PORT E (RE). These ports are used for input/ output interfacing. In this controller, PORT A is only 6 bits wide (RA-0 to RA-7), PORT B , PORT C,PORT D are only 8 bits wide (RB-0 to RB-7,RC-0 to RC-7,RD-0 to RD-7), PORT E has only 3 bit wide (RE-0 to RE-7).

All these ports are bi-directional. The direction of the port is controlled by using TRIS(X) registers (TRIS A used to set the direction of PORT-A, TRIS B used to set the direction for PORT-B, etc.). Setting a TRIS(X) bit1 will set the corresponding PORT(X) bit as input. Clearing a TRIS(X) bit 0 will set the corresponding PORT(X) bit as output.(If we want to set PORT A as an input, just set TRIS(A) bit to logical ,,1 and want to set PORT B as an output, just set the PORT B bits to logical 0.)

- Analog input port (AN0 TO AN7): these ports are used for interfacing analog inputs.
- TX and RX: These are the USART transmission and reception ports.
- SCK: These pins are used for giving synchronous serial clock input.
- SCL: These pins act as an output for both SPI and I2C modes.
- DT: These are synchronous data terminals.
- CK: Synchronous clock input.
- SD0: SPI data output (SPI Mode).
- SD1: SPI Data input (SPI mode).
- SDA: Data input/output in I2C Mode.
- CCP1 and CCP2: These are capture/compare/PWM modules.
- OSC1: Oscillator input/external clock.
- OSC2: Oscillator output/clock out.
- MCLR: Master clear pin (Active low reset).
- Vpp: programming voltage input.
- THV: High voltage test mode controlling.
- Vref (+/-): reference voltage.
- SS: Slave select for the synchronous serial port.
- T0CK1: clock input to TIMER 0.
- T1OSO: Timer 1 oscillator output.
- T1OS1: Timer 1 oscillator input.
- T1CK1: clock input to Timer 1.
- PGD: Serial programming data.

- PGC: serial programming clock.
- PGM: Low Voltage Programming input.
- INT: external interrupt.
- RD: Read control for parallel slave port.
- CS: Select control for parallel slave.
- PSP0 to PSP7: Parallel slave port.
- VDD: positive supply for logic and input pins.
- VSS: Ground reference for logic and input/output pins

Input/ Output Ports

In order to synchronize the operation of I/O ports with the internal 8-bit organization of the microcontroller, they are, similar to registers, grouped into five ports denoted by A, B, C, D and E. All of them have several features in common: If a pin performs any of these functions, it may not be used as a general-purpose input/output pin. TRIS register: TRISA, TRISB, TRISC etc which determines the performance of port bits, but not their contents. By clearing any bit of the TRIS register (bit=0), the corresponding port pin is configured as an output. Similarly, by setting any bit of the TRIS register (bit=1), the corresponding port pin is configured as an input. This rule is easy to remember 0 = Output, 1 = Input.

PORTC and TRISC register

Port C is an 8-bit wide, bidirectional port. Bits of the TRISC register determine the function of its pins. Similar to other ports, a logic one (1) in the TRISC register configures the appropriate portC pin as an input. Port D is an 8 bit wide, bidirectional port Bits .

PORTE and TRISE register

Port E is a 4-bit wide, bidirectional port. The TRISE registers bits determine the function of its pins. Similar to other ports, a logic one in the TRISE register configures the appropriate portE pin as an input. The exception is the RE3 pin which is always configured as an input. Similar to ports A and B, three pins can be configured as analog inputs in this case. The

ANSELH register bits determine whether a pin will act as an analog input (AN) or digital input/output:

RE0 = AN5

RE1 = AN6

RE2 = AN7

INSTRUCTION SET OF PIC 16F877

The instruction set for the 16F8XX includes 35 instructions. The reason for such a small number of instructions lies in the RISC architecture. It means that instructions are well optimized from the aspects of operating speed, simplicity in architecture and code compactness.

Instruction Execution Time

All instructions are single-cycle instructions. The only exception may be conditional branch instructions (if condition is met) or instructions performed upon the program counter. In both cases, two cycles are required for instruction execution, while the second cycle is executed as an NOP *(No Operation).* Single-cycle instructions consist of four clock cycles. If 4MHz oscillator is used, the nominal time for instruction execution is 1μs. As for jump instructions, the instruction execution time is 2μs.

- Data transfer Instruction
- Arithmetic and Logic Instruction
- Bit oriented Instruction
- Program Control Instruction

Data transfer Instruction:

The data is copied from source to Destination without any change.

EX: MOVLW k-□Move constant to W.

MOV WF f□Move W to F

CLR W Clear W

Arithmetic and Logic Instruction:

To perform arithmetic operation such as addition, subtraction, Increment and decrement. The group of instruction perform logical operation such as AND, OR, Exclusive-OR, Rotate, Compare, and Complement the content.

EX: ADDLW k Add W and Constant

SUB LW k Subtract W from constant

IORLW k Logical OR with W with constant

Bit oriented Instruction:

BC F f, b Clear bit b in f

Program Control Instruction:

CALL k Call subroutine

RETURN Return from subroutine

9. PIC ADDRESSING MODES.

1. Immediate addressing mode
2. Direct addressing mode
3. Register addressing mode
4. Indexed ROM addressing mode

1. Immediate addressing mode

- In immediate addressing mode, the immediate data is specified in the instruction.
- The immediate addressing mode is used to load the data into PIC registers and WREG register.
- However, it cannot use to load data into any of the file register.

Example:

1. MOVLW 50H

2. ANDLW 40H 3. IORLW 60H

2. Direct addressing mode:

- In direct addressing mode, the 8- bit data in RAM memory location whose address is specified in the instruction.
- This mode is used for accessing the RAM file register.

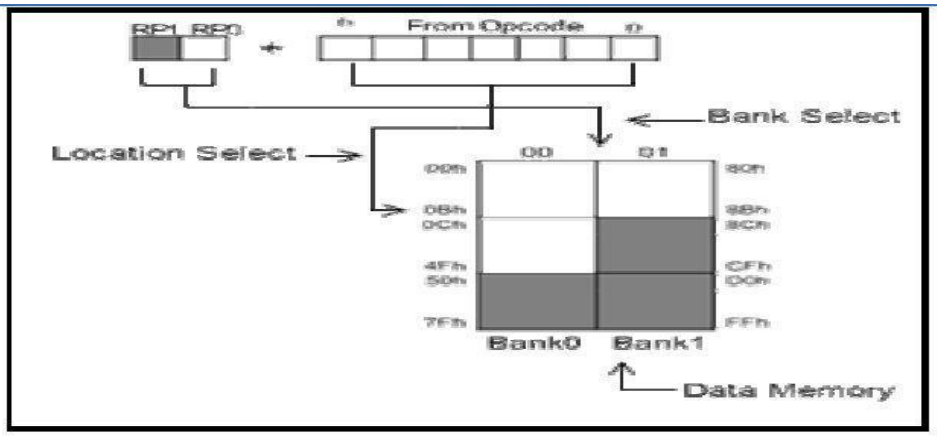


Fig.3.6 Pictorial representation of Direct Addressing

Example:

1. MOVWF 0X10

2. MOVFF PORTB, POTRC

3. MOVFF 0X30, PORTC

3. Register indirect addressing mode:

- Register indirect addressing mode is used for accessing data stored in the RAM part of file register.
- In this addressing mode a register is used as pointer to the memory location of the file register.
- Three file select registers are used. They are FSR0, FSR1 and FSR2.

Example:

1. LFSR1,0X55

2. MOVWF INDF2

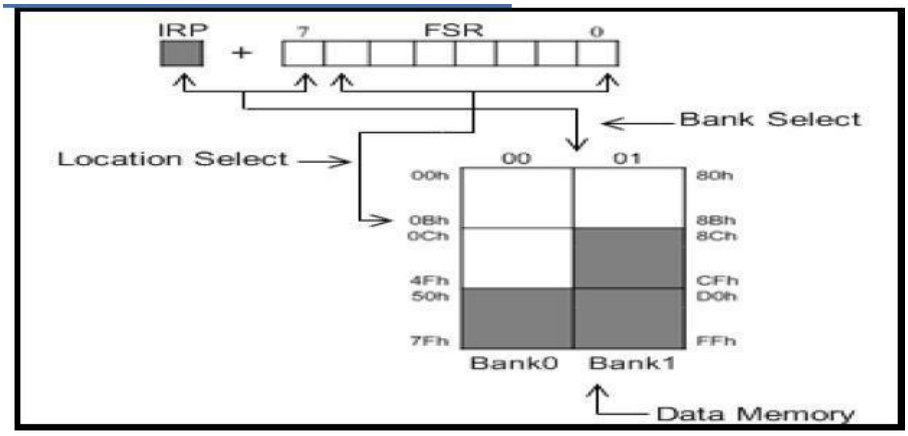


Fig3.7 Pictorial representation of Indirect Addressing

4. Indexed ROM addressing mode:

- This addressing mode is used for accessing the data from look up tables that reside in the PIC18 program ROM.

11. Watch Dog Timer

- Watchdog Timer (WDT) can be helpful to automatically reset the system whenever a timeout occurs
- A system reset is required for preventing the failure of the system in a situation of a hardware fault or program error.
- There are countless applications where the system cannot afford to get stuck at a point (not even for a small duration of time).
- For example, in a radar system, if the system hangs for 5 minutes, it can result in serious repercussions (an enemy plane or missile may go undetected resulting in huge losses).
- The system should be robust enough to automatically detect the failures quickly and reset itself in order to recover from the failures and function normally without errors.

- One can manually reset the system to recover from errors. But it is not always feasible to manually reset the system, especially once it has been deployed.
- To overcome such problems, a watchdog timer is necessary to automatically reset the system without human intervention.
- The watchdog timer is loaded with a timeout period which is dependent on the application.
- The watchdog timer starts its counting independent of a system clock i.e. it has a separate internal oscillator to work independently of a system clock.
- The watchdog timer cleared through software each time before the timeout period occurs.
- Whenever software failed to clear the watchdog timer before its timeout period, the watchdog timer resets the system.
- For this purpose, a watchdog timer is used to overcome software failures in real-time applications.
- The watchdog timer is also used to wake up the microcontroller from sleep mode.
- In PIC18F4550, the watchdog timer uses a different 31 kHz INTRC clock and it is independent of a system clock.
- Watchdog Timer can be enabled in two ways through Configuration Register (CONFIG2H) and through WDTCON Register.
- CONFIG2H has a WDTEN bit to enable/disable the watchdog timer.
- WDTCON (WDT control register) has the SWDTEN bit which is used to enable/disable the WDT through software.

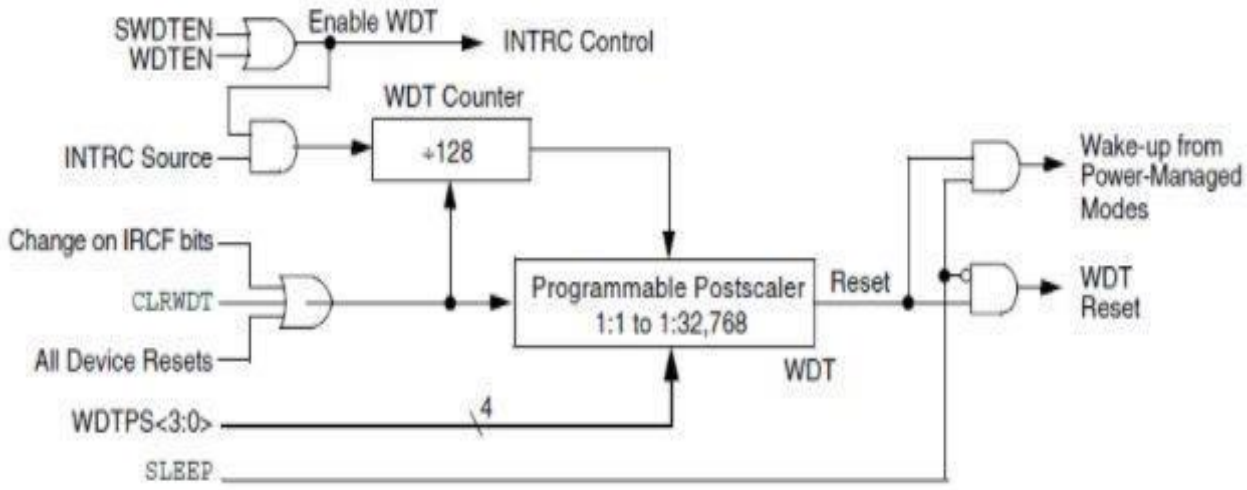


Fig3.8 Operation of Watch dog Timer

- When WDT is enabled, 31 kHz INTRC source gets initialized and provides a clock for the watchdog timer.
- This clock is then divided by 128 (pre-scaler). This pre-scaler gives a nominal time-out period of 4 ms.
- PIC18F4550 also has a programmable Post-scaler which helps to divide down the WDT pre-scaler output and increase the time-out periods. So now we can vary the time-out period in the range of 4ms to 131.072 sec (2.18 min) using Post-scaler.

Enabling and Disabling WDT

There are two ways to enable or disable the WDT which are given as follows

1. Through Configuration Register:

7	6	5	4	3	2	1	0
			WDTPS3	WDTPS2	WDTPS1	WDTPS0	WDTEN

Fig.3.9 CONFIG2H Register: Configuration Register 2 High

Bit 0 – WDTEN: Watchdog Timer Enable bit

0 = Disables WDT (possible to enable WDT through SWDTEN)

1 = Enables WDT

Bit 4:1 – WDTPS3:WDTPS0: Watchdog Timer Post-scale select bit

1111 = 1:32768

1110 = 1:16384

1101 = 1:8192

1100 = 1:4096

1011 = 1:2048

1010 = 1:1024

1001 = 1:512

1000 = 1:256

0111 = 1:128

0110 = 1:64

0101 = 1:32

0100 = 1:16

0011 = 1:8

0010 = 1:4

0001 = 1:2

0000 = 1:1

Through WDTCON Register

Fig3.10 WDTCON Register: Watchdog Timer Control

Register Bit 0 – SWDTEN: Software Controlled Watchdog Timer Enable bit

0 = Disable Watchdog Timer

1 = Enable Watchdog Timer

This software controlled watchdog timer can enable watchdog timer only if configuration bit has disabled the WDT.

If the WDTEN (configuration bit) is enabled, then SWDTEN has no effect.

Calculate the WDT Timeout Period

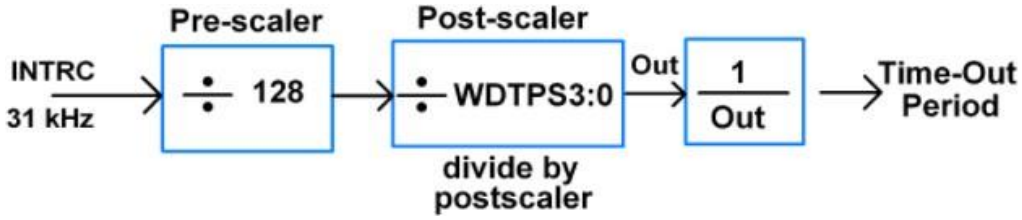


Fig.3.11 WDT

Question Bank

UNIT III

PART A

1Compare RISC vs CISC controllers.
2.What is the role of program counter in accessing program memory in PIC microcontroller?
3.Write about serial connector
4.What is meant by PIC
5.What is meant by interrupt controller
6What are the various types of PIC
7.Explain the different types of addressing 8.Explain the memory organization
9 Define RISC processor 10.Define CSIC processor

PART- B

1. Explain with a neat diagram the architecture of micro controller.
2. Explain the addressing modes of micro controller
3. List the features of PIC Micro controller
4. Explain about the RISC architecture.
5. Discuss the addressing modes of PIC microcontroller with ports.
6. List out the different instruction group in PIC microcontroller and explain the and Compare CICS and RISC.
7. Explain the instruction set of PIC microcontroller. 8..Write in detail about ports,interrupt and timer
8. Explain the architecture of microprocessor with watch dog timer.
9. Describe the operations carried out whenexecutes the following instructions:
10. Movlw 50H (ii) Swapf INTCON, W
11. Clrwdt (iv) bsf Port B,O

TEXT / REFERENCE BOOKS

1. Ramesh Goankar, "Microprocessor architecture programming and applications with 8085 / 8088", 5th Edition, Penram International Publishing, 2002.
2. Mazidi & McKinlay, “The 8051 Microcontroller and Embedded Systems using Assembly and C”, PHI, 2007.
3. MykePredko, “Programming and Customizing the 8051 Micro-controller”, Tata McGraw-Hill edition, 2007.
4. R A Gaonkar, “Fundamentals of Microcontrollers and Applications in Embedded Systems (with the PIC18 Microcontroller Family)”, Penram Publishing India, 2007

5. Kenneth Ayala ,”The 8051 Microcontroller”, 3rd Edition, Thomson Delmar Learning, 2004.
6. Kenneth J. Ayala, Dhananjay V. Gadre, “The 8051 Microcontroller & Embedded Systems Using Assembly and C”, Cengage Learning India Publication, 2007.
7. Ajay V Deshmukh, “Microcontrollers: Theory and Applications”, Tata McGraw-Hill, 2005
8. Raj Kamal, “Embedded Systems Architecture, Programming, and Design”. (2/e), Tata McGraw Hill, 2008.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF ELECTRICAL & ELECTRONICS ENGINEERING

DEPARTMENT OF ELECTRONICS AND INSTRUMENTATION ENGINEERING

UNIT IV-MICROPROCESSORS,MICROCONTROLLERS AND EMBEDDED SYSTEMS: SEIA 1504

UNIT 4 INTERFACING

INTERFACING 9 Hrs. Basic Interface concepts, Fundamentals of memory interface- memory mapped I/O and I/O mapped I/O, Interrupt and vectored interrupt, Programmable peripheral interface 8255 - Programmable Interval timer 8253 - Programmable interrupt controller 8259 - Programmable DMA controller 8257

1.PROGRAMMABLE PERIPHERAL INTERFACE -825PPI

The Intel 8255 (or i8255) Programmable Peripheral Interface (PPI) chip is a peripheral chip, is used to give the CPU access to programmable parallel I/O. It can be programmable to transfer data under various conditions from simple I/O to interrupt I/O. it is flexible versatile

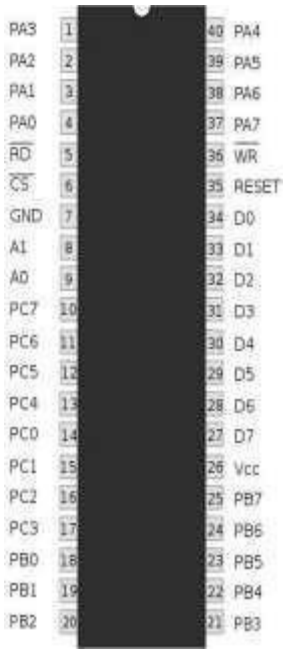


Fig 4.1: Pin diagram

and economical (when multiple I/O ports are required) but some what complex. It is an important general purpose I/O device that can be used with almost any microprocessor. Functional block of 8255 – Programmable Peripheral Interface (PPI)

The 8255A has 24 I/O pins that can be grouped primarily in two 8-bit parallel ports: A and B with the remaining eight bits as port C. The eight bits of port C can be used as individual bits or be grouped in to 4-bit ports: CUpper (Cu) and CLower (CL) as in Figure **2.** The function of these ports is defined by writing a control word in the control register as

shown in Figure 3.3

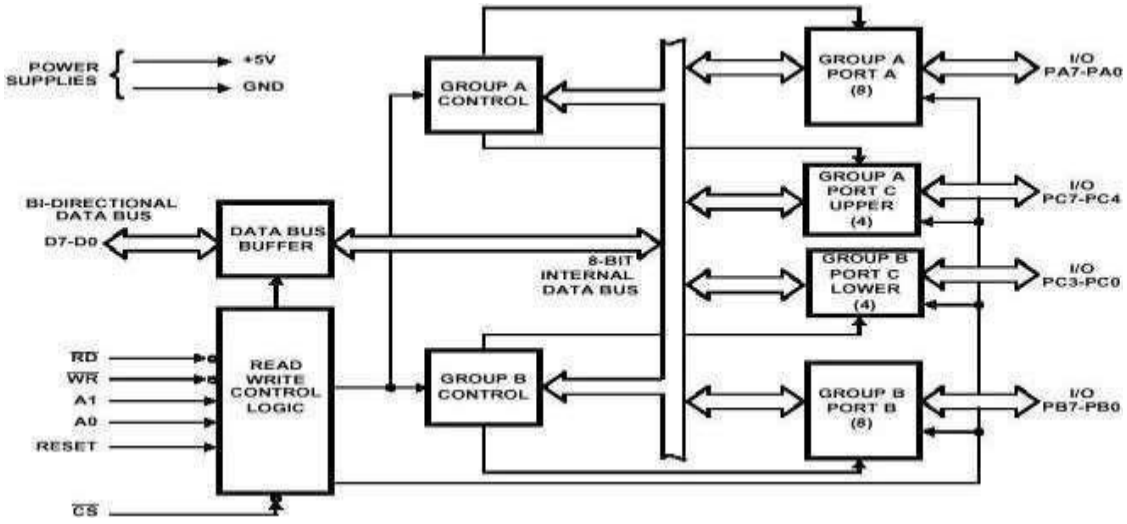


Fig 4.2 : Block diagram of 8255

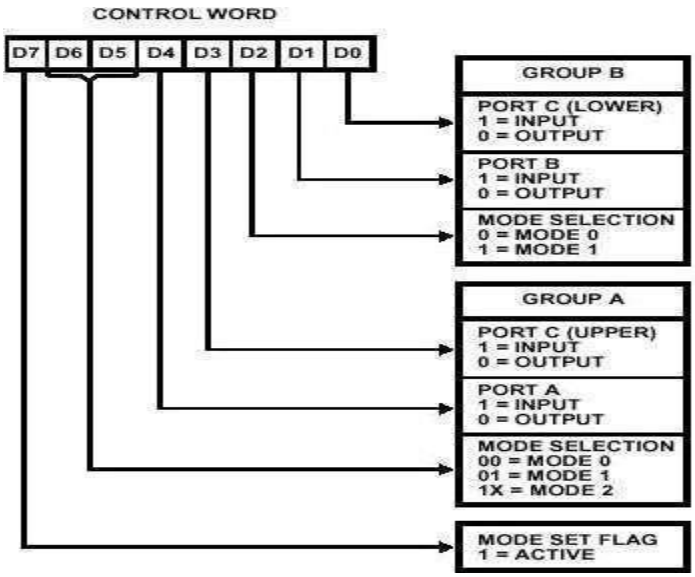


Fig 4.3. Control word Register format

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 theCPU. 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 andWR 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).

A1	A0	SELECTION
0	0	PORT A
0	1	PORT B
1	0	PORT C
1	1	CONTROL

Fig 4.4 selection of Ports and Control reg

(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.

Group A and Group B Controls

The functional configuration of each port is programmed by the systems software. In essence, theCPU "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. Eachof 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

e 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.

I. Operational modes of 8255

There are two basic operational modes of 8255:

1. Bit set/reset Mode (BSR Mode).
2. 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.

1. 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 as shown in Figure 4. BSR mode and I/O mode are independent and selection of BSR mode does not affect the operation of other portsin I/O mode.

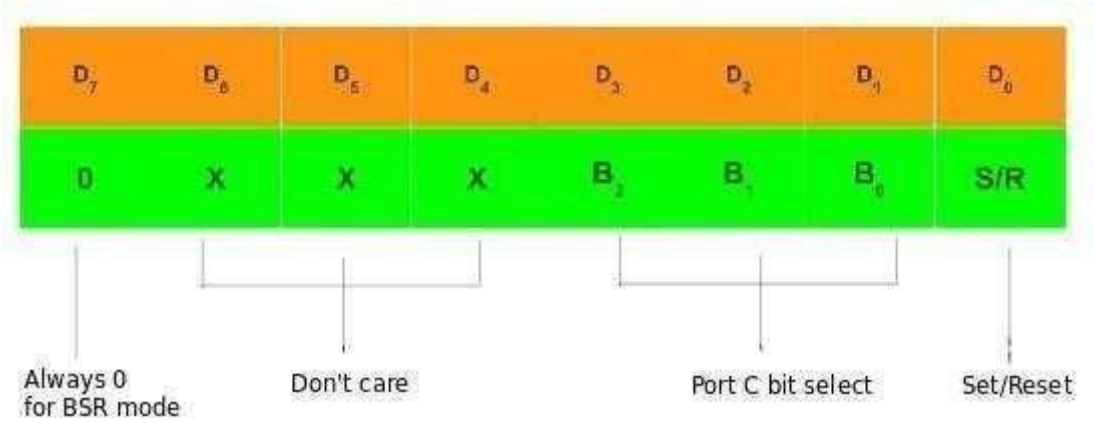


Fig 4.5: 8255 Control register format for BSR mode

- D7 bit is always 0 for BSR mode. Bits D6, D5 and D4 are don't care bits.
- Bits D3, D2 and D1 are used to select the pin of Port C.Bit D0 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	PC_0
0	0	1	PC_1
0	1	0	PC_2
0	1	1	PC_3
1	0	0	PC_4
1	0	1	PC_5
1	1	0	PC_6
1	1	1	PC_7

As an example, if it is needed that PC_5 be set, then in the control word,

1. Since it is BSR mode, D_7 = '0'.
2. Since D_4, D_5, D_6 are not used, assume them to be'0'.
3. PC_5 has to be selected, hence, D_3 = '1', D_2 = '0', D_1 = '1'.
4. PC_5 has to be set, hence, D0 = '1'.

Thus, as per the above values, 0B (Hex) will be loaded into the Control Word Register (CWR).

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	1	0	1	1

2. Input/Output mode

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

1. Mode 0 - Simple I/O
2. Mode 1 - Strobed I/O
3. Mode 2 - Strobed Bi-directional I/O

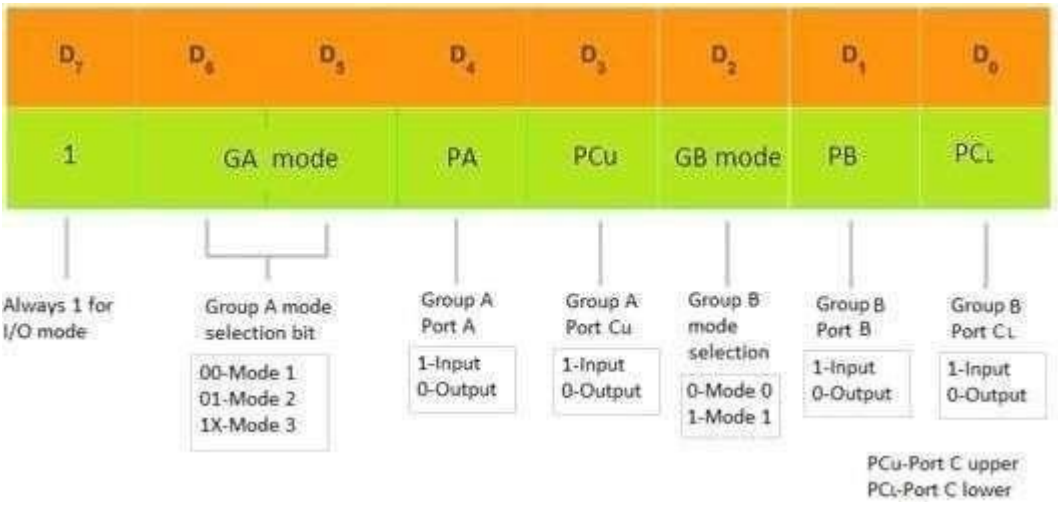


Figure 4.6: 8255 Control word for I/O mode

- D0, D1, D3, D4 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 D0 = D4 = 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 D1 = D3 = 0, then port B and upper port C act as output ports as shown in Figure 5.
- D2 is used for mode selection of Group B (port B and lower port C). When D2 = 0, mode 0 is selected and when D2 = 1, mode 1 is selected.
- D5& D6 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, D7 = 1.

For example, if port B and upper port C have to be initialized as input ports and lower port Cand port A as output ports (all in mode 0):

1. Since it is an I/O mode, D7 = 1.
2. Mode selection bits, D2, D5, D6 are all 0 for mode 0 operation.
3. Port B and upper port C should operate as Input ports, hence, D1 = D3 = 1.
4. Port A and lower port C should operate as Output ports, hence, D4 = D0 = 0.

Hence, for the desired operation, the control word register will have to be loaded with "10001010" = 8A (hex).

- 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:

1. Output ports are latched.

2. Input ports are buffered, not latched.
3. Ports do not have handshake or interrupt capability.
4. 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 $\overline{CS}$ low. Then it selects the desired port using A_0 and A_1 lines.
- The CPU then issues an $\overline{RD}$ 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 $\overline{CS}$ low. It then selects the desired port using A_0 and A_1 lines.
- CPU then issues a $\overline{WR}$ signal to write data to the selected port via the system databus. 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 initilalised 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:

1. Two ports i.e. port A and B can be used as 8-bit i/o ports.
2. Each port uses three lines of port c as handshake signal and remaining two signals can be used as i/o ports.
3. Interrupt logic is supported.
4. 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 external device receives the data via the signal. this pin is qualified by the internal INTE(interrupt enable) bit.

4. 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.

- Mode 2

Only group A can be initialized in this mode. Port A can be used for bidirectional handshake data transfer. This means that data can be input or output on the same eight lines (PA0 - PA7). Pins PC3 - PC7 are used as handshake lines for port A. The remaining pins of port C (PC0 - PC2) can be used as input/output lines if group B is initialized in mode 0 or as handshaking for port B if group B is initialized in mode 1. In this mode, the 8255 may be used to extend the system bus to a slave microprocessor or to transfer data bytes to and froma floppy disk controller. Acknowledgement and handshaking signals are provided to maintain proper data flow and synchronisation between the data transmitter and receiver.

II. Interfacing 8255 with 8085 processor

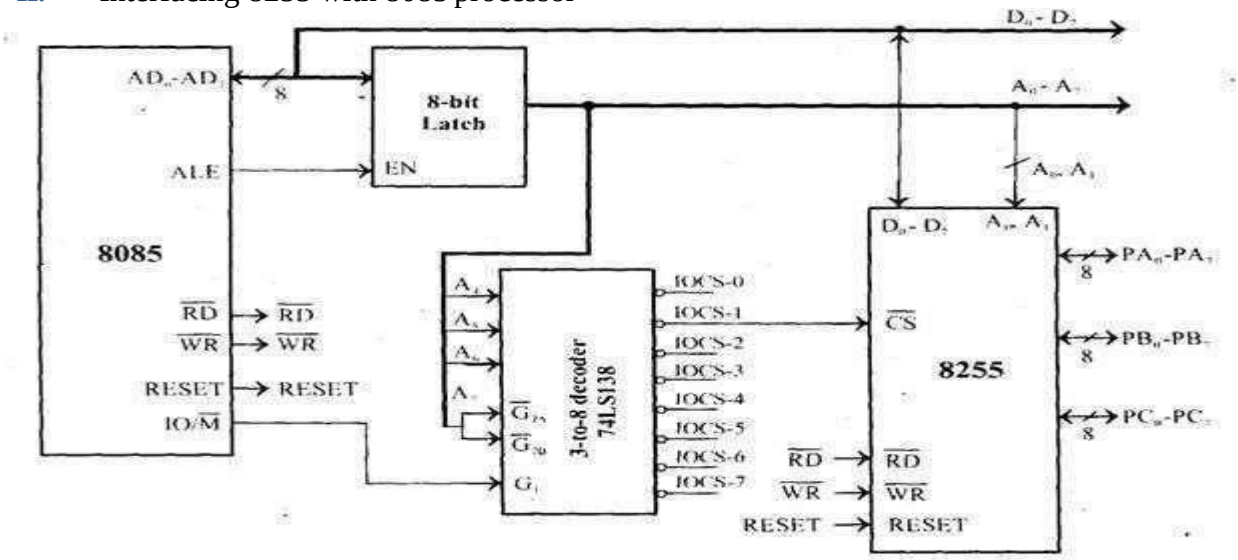


Fig 4.7. Interfacing 8255 with 8085 processor

- The 8255 can be either memory mapped or I/O mapped in the system. In the schematic shown in above is I/O mapped in the system.

Using a 3-to-8decoder generates the chip select signals for I/O mapped devices.

- The address lines A4, A5 and A6 are decoded to generate eight chip select signals (IOCS-0 to IOCS-7) and in this, the chip select IOCS- 1 is used to select 8255 as shown in Figure 3.7.
- The address line A7 and the control signal IO/M (low) are used as enable for thedecoder.
- The address line A0 of 8085 is connected to A0 of 8255 and A1 of 8085 is connected toA1 of 8255 to provide the internal addresses.
- The data lines D0-D7 are connected to D0-D7 of the processor to achieve parallel datatransfer.
- The I/O addresses allotted to the internal devices of 8255 are listed in table.

Internal Device	Binary Address								Hexa Address
	Decoder input and enable				Input to address pins of 8255				
	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
Port-A	0	0	0	1	x	x	0	0	10
Port-B	0	0	0	1	x	x	0	1	11
Port-C	0	0	0	1	x	x	1	0	12
Control Register	0	0	0	1	x	x	1	1	13

Note : *Don't care "x" is considered as zero.*

2. USART 8251 (Universal Synchronous/ Asynchronous Receiver Transmitter)

The 8251 is a USART (Universal Synchronous Asynchronous Receiver Transmitter) for serial data communication. As a peripheral device of a microcomputer system, the 8251 receives parallel data from the CPU and transmits serial data after conversion. This device also receives serial data from the outside and transmits parallel data to the CPU after conversion as shown in Figure 3.8.

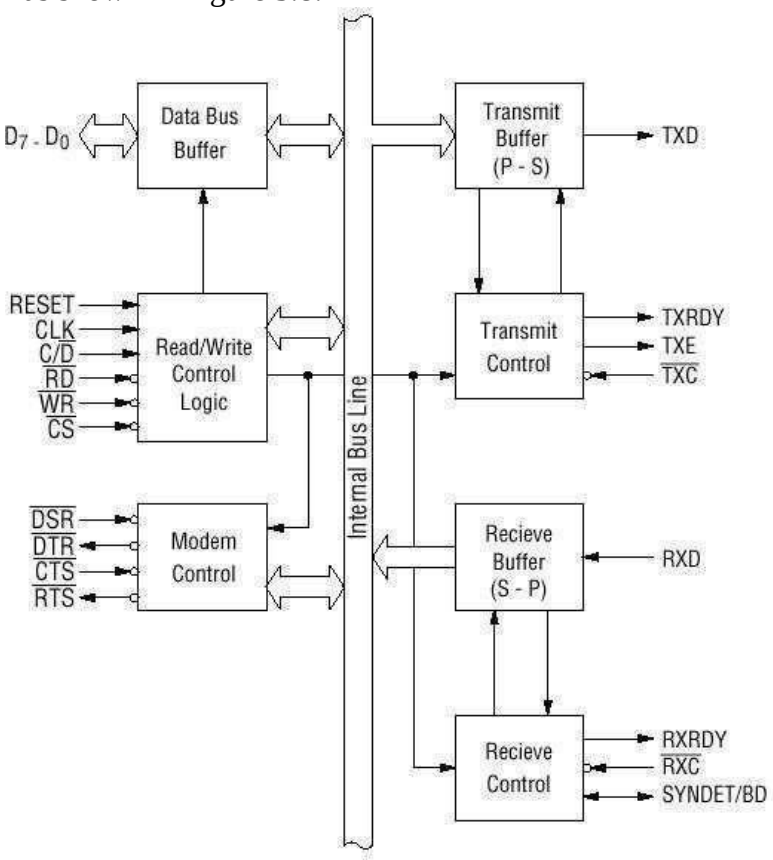


Figure 4.8 : Architecture of 8251

Transmitter Section

The transmitter section consists of three blocks—transmitter buffer register, output register and the transmitter control logic block. The CPU deposits (when TXRDY = 1, meaning that the transmitter buffer register is empty) data into the transmitter buffer register, which is subsequently put into the output register (when TXE = 1, meaning that the output buffer is empty). In the output register, the eight bit data is converted into serial form and comes out

via TXD pin. The serial data bits are preceded by START bit and succeeded by STOP bit, which are known as framing bits. But this happens only if transmitter is enabled and the CTS is low. TXC signal is the transmitter clock signal which controls the bit rate on the TXD line (output line). This clock frequency can be 1, 16 or 64 times the baud.

Receiver Section

The receiver section consists of three blocks — receiver buffer register, input register and the receiver control logic block. Serial data from outside world is delivered to the input register via RXD line, which is subsequently put into parallel form and placed in the receiver buffer register. When this register is full, the RXRDY (receiver ready) line becomes high. This line is then used either to interrupt the MPU or to indicate its own status. MPU then accepts the data from the register. RXC line stands for receiver clock. This clock signal controls the rate at which bits are received by the input register. The clock can be set to 1, 16 or 64 times the baud in the asynchronous mode.

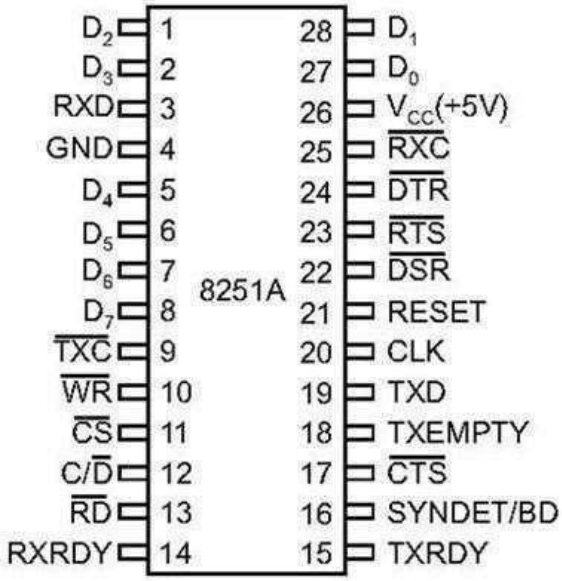


Fig 4.9 : Pin Configuration of 8251

Pin Configuration of 8251 is shown in figure

11. D 0 to D 7 (l/O terminal)

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

RESET (Input terminal)

A "High" on this input forces the 8251 into "reset status." The device waits for the writing of"mode instruction." The min. reset width is six clock inputs during the operating status of CLK.

CLK (Input terminal)

CLK signal is used to generate internal device timing. CLK signal is independent of RXC or TXC. However, the frequency of CLK must be greater than 30 times the RXC and TXC at Synchronous mode and Asynchronous "x1" mode, and must be greater than 5 times at Asynchronous "x16" and "x64" mode.

WR (Input terminal)

This is the "active low" input terminal which receives a signal for writing transmit data andcontrol words from the CPU into the 8251.

RD (Input terminal)

This is the "active low" input terminal which receives a signal for reading receive data and statuswords from the 8251.

C/D (Input terminal)

This is an input terminal which receives a signal for selecting data or command words and statuswords when the 8251 is accessed by the CPU. If C/D = low, data will be accessed. If C/D = high, command word or status word will be accessed.CS (Input terminal)

This is the "active low" input terminal which selects the 8251 at low level when the CPU accesses. Note: The device won‘t be in "standby status"; only setting CS = High. TXD (output terminal)

This is an output terminal for transmitting data from which serial-converted data is sent out. The device is in "mark status" (high level) after resetting or during a status when transmit is disabled. It is also possible to set the device in "break status" (low level) by a command.

TXRDY (output terminal)

This is an output terminal which indicates that the 8251is ready to accept a transmitted data character. But the terminal is always at low level if CTS = high or the device was set in "TX disable status" by a command. Note: TXRDY status word indicates that transmit data character isreceivable, regardless of CTS or command. If the CPU writes a data character, TXRDY will be reset by the leading edge or WR signal.

TXEMPTY (Output terminal)

This is an output terminal which indicates that the 8251 has transmitted all the characters and hadno data character. In "synchronous mode," the terminal is at high level, if transmit data charactersare no longer remaining and sync characters are automatically transmitted.

If the CPU writes a data character, TXEMPTY will be reset by the leading edge of WR signal. Note : As the transmitter is disabled by setting CTS "High" or command, data written before disable will be sent out. Then TXD and TXEMPTY will be "High". Even if a data is written after disable, that data is not sent out and TXE will be "High". After the transmitter is enabled, it sent out. (Refer to Timing Chart of Transmitter Control and Flag Timing)
TXC (Input terminal)

This is a clock input signal which determines the transfer speed of transmitted data. In "synchronous mode," the baud rate will be the same as the frequency of TXC. In "asynchronous mode", it is possible to select the baud rate factor by mode instruction. It can be 1, 1/16 or 1/64 the TXC. The falling edge of TXC sifts the serial data out of the 8251.
RXD (input terminal)

This is a terminal which receives
serial data.RXRDY (Output
terminal)

This is a terminal which indicates that the 8251 contains a character that is ready to READ. If theCPU reads a data character, RXRDY will be reset by the leading edge of RD signal.

\Unless the CPU reads a data character before the next one is received completely, the preceding data will belost. In such a case, an overrun error flag status word will be set.

RXC (Input terminal)

This is a clock input signal which determines the transfer speed of received data. In "synchronous mode," the baud rate is the same as the frequency of RXC. In "asynchronous mode," it is possible to select the baud rate factor by mode instruction. It can be 1, 1/16, 1/64 the RXC.
SYNDET/BD (Input or output terminal)

This is a terminal whose function changes according to mode. In "internal synchronous mode." this terminal is at high level, if sync characters are received and synchronized. If a status word is read, the terminal will be reset. In "external synchronous mode, "this is an input terminal. A "High" on this input forces the 8251 to start receiving data characters.
In "asynchronous mode," this is an output terminal which generates "high level"output

upon the detection of a "break" character if receiver data contains a "low-level" space between the stopbits of two continuous characters. The terminal will be reset, if RXD is at high level. After Reset is active, the terminal will be output at low level.

DSR (Input terminal)

This is an input port for MODEM interface. The input status of the terminal can be recognizedby the CPU reading status words.

DTR (Output terminal)

This is an output port for MODEM interface. It is possible to set the status of DTR by a command.

CTS (Input terminal)

This is an input terminal for MODEM interface which is used for controlling a transmit circuit. The terminal controls data transmission if the device is set in "TX Enable" status by a command.

Data is transmitable if the terminal is

atlow level.

RTS (Output terminal)

This is an output port for MODEM interface. It is possible to set the status RTS by a command. The 8251 functional configuration is programmed by software. Operation between the 8251 anda CPU is executed by program control. Table 1 shows the operation between a CPU and thedevice.

Summary of Control Signals for 8251

$\overline{CS}$	$\overline{C/D}$	$\overline{RD}$	$\overline{WR}$	*Function*
0	1	1	0	MPU writes instructions in the control register
0	1	0	1	MPU reads status from the status register
0	0	1	0	MPU outputs data to the Data Buffer
0	0	0	1	MPU accepts data from the Data Buffer
1	X	X	X	USART is not selected

Control Words

There are two types of control word.

1. Mode instruction (setting of function)

2. Command (setting of operation)

1) Mode Instruction

Mode instruction is used for setting the function of the 8251. Mode instruction will be in "wait for write" at either internal reset or external reset. That is, the writing of a control word after resetting will be recognized as a "mode instruction."

Items set by mode instruction are as follows:

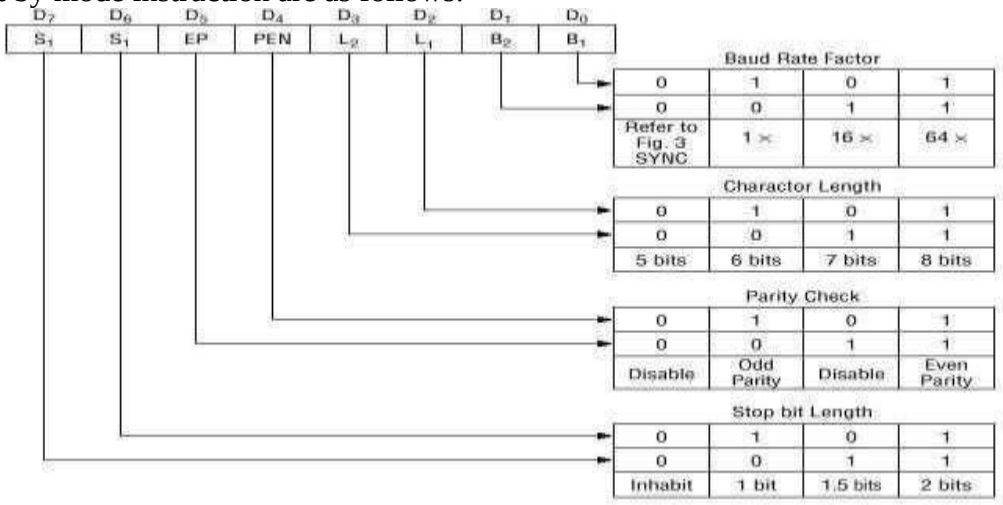


Fig. 2 Bit Configuration of Mode Instruction (Asynchronous)

Fig 4.10: Bit configuration of Mode instruction (Asynchronous)

- Synchronous/asynchronous mode
- Stop bit length (asynchronous mode)
- Character length
- Parity bit

• Baud rate factor (asynchronous mode)

• Internal/external synchronization (synchronous mode)

• Number of synchronous characters (Synchronous mode)

The bit configuration of mode instruction is shown in Figures 12 and 13. In the case of synchronous mode, it is necessary to write one-or two byte sync characters. If sync characters were written, a function will be set because the writing of sync characters constitutes part of mode instruction.

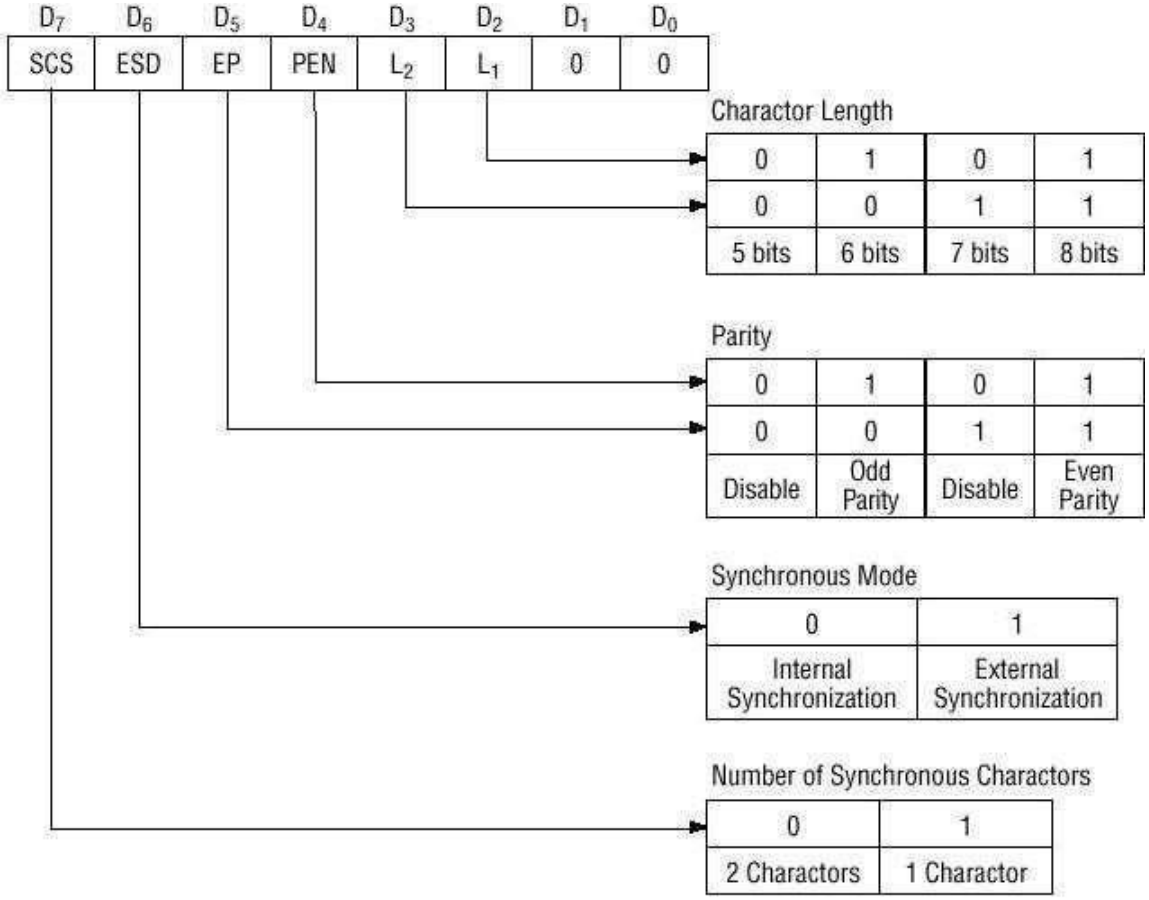


Fig. 3 Bit Configuration of Mode Instruction (Synchronous)

Fig 4.11: Bit configuration of mode instruction(synchronous)

2) Command

Command is used for setting the operation of the 8251. It is possible to write a commandwhenever necessary after writing a mode instruction and sync characters as shown in figure 14.

Items to be set by command are as follows:

- Transmit Enable/Disable
- Receive Enable/Disable
- DTR, RTS Output of data.
- Resetting of error flag.
- Sending to break characters
- Internal resetting
- Hunt mode (synchronous mode)

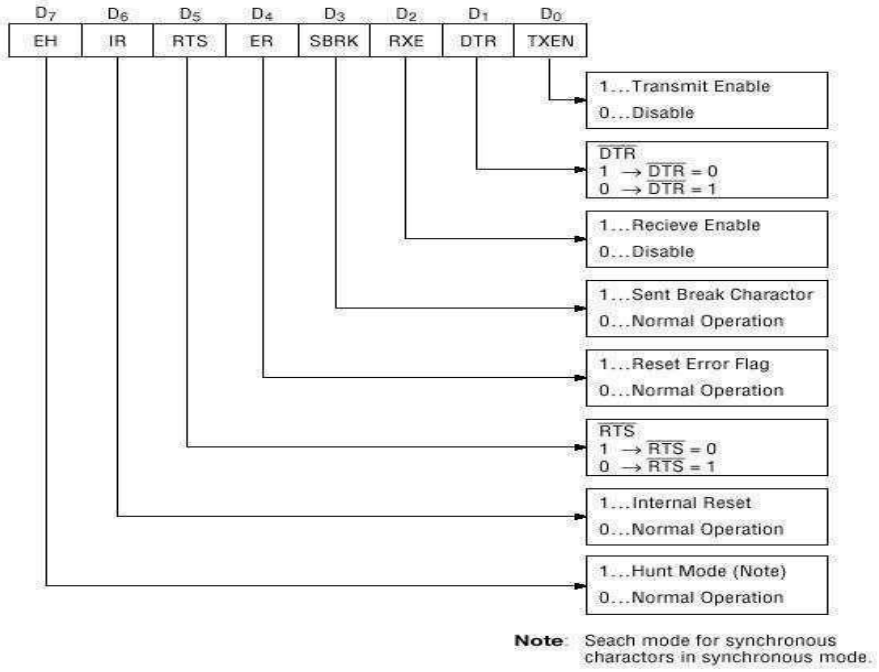


Fig. 4 Bit Configuration of Command

Fig 4.12: Bit configuration of command

Status Word

It is possible to see the internal status of the 8251 by reading a status word. The bit configuration of status word is shown in Fig.15.

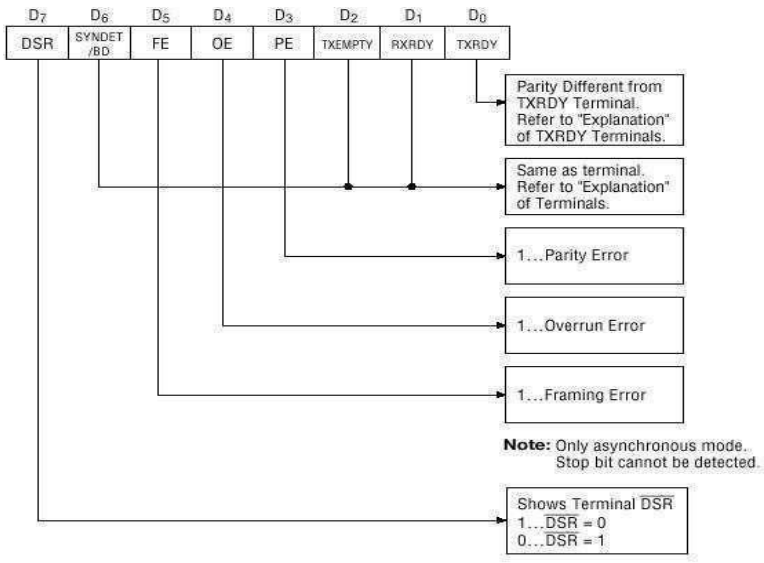


Fig. 5 Bit Configuration of Status Word

Fig 4.13: Bit configuration of Status Word

3.8253/8254 PROGRAMMABLE INTERVAL TIMER:PIT

The 8254 programmable Interval timer consists of three independent 16-bit programmable counters (timers). Each counter is capable of counting in binary or binary coded decimal. The maximum allowable frequency to any counter is 10MHz. This device is useful whenever the microprocessor must control real-time events. The timer in a personal computer is an 8253. 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, which interrupts the processor. The count can count either in binary or BCD Each counter in the block diagram has 3 logical lines connected to it. Two of these lines, clock and gate, are inputs. The third, labeled OUT is an output.

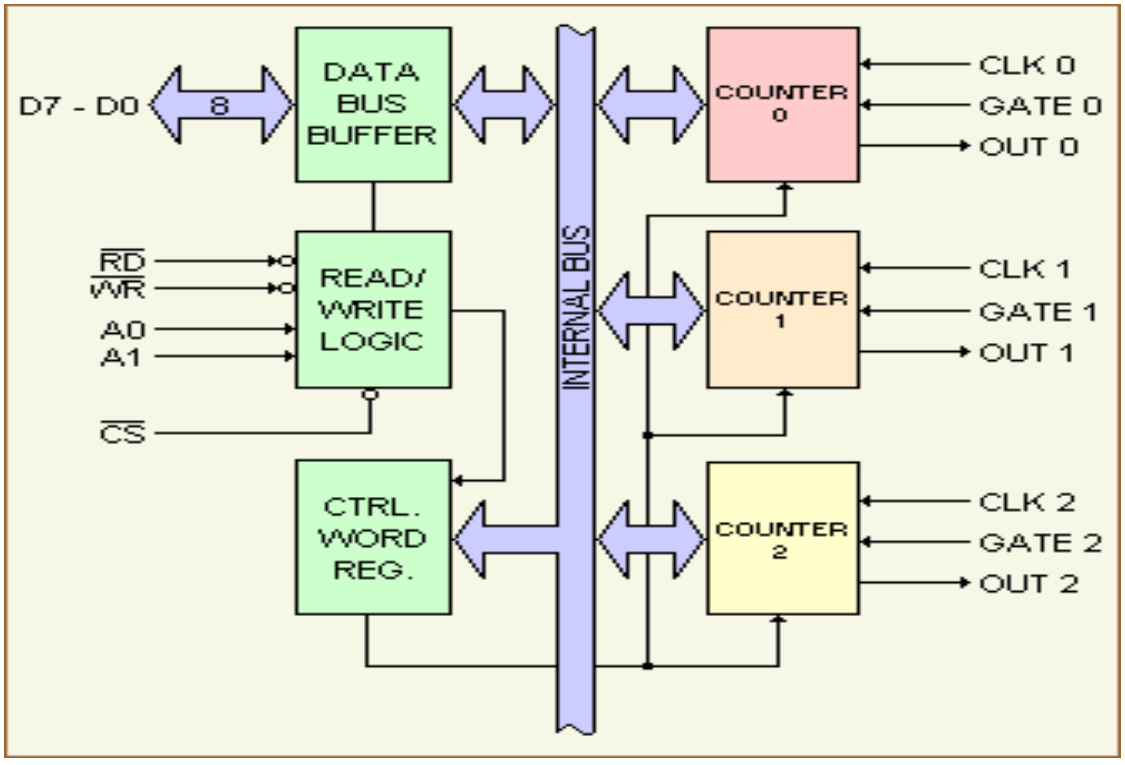


Fig : 4.14 Block Diagram of 8253 programmable interval timer

Data bus buffer- It is a communication path between the timer and the microprocessor. The buffer is 8-bit and bidirectional. It is connected to the data bus of the microprocessor. Read /write logic controls the reading and the writing of the counter registers. Control word register, specifies the counter to be used and either a Read or a write operation. Data is transmitted or received by the buffer upon execution of INPUT instruction from CPU as shown in figure 16. The data bus buffer has three basic functions,

(i). Programming the modes of 8253. (ii). Loading the count value in times (iii).Reading the count value from timers.

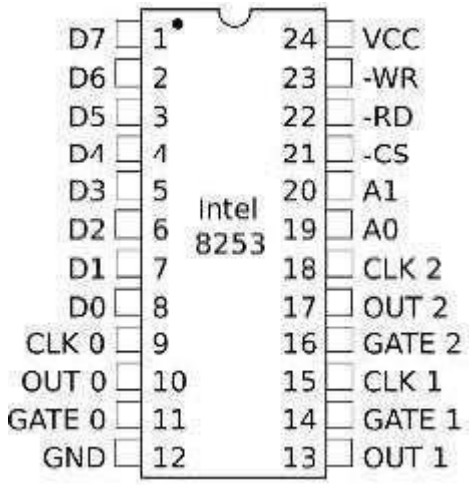


Fig 4.15:Pin Diagram of 8253

The data bus buffer is connected to microprocessor using D7 – D0 pins which are also bidirectional. The data transfer is through these pins. These pins will be in high- impedance (or this state) condition until the 8253 is selected by a LOW or *CS* and either the read operation requested by a LOW *RD* on the input or a write operation *WR* requested by the input going LOW.

Read/ Write Logic:

It accepts inputs for the system control bus and in turn generation the control signals for overall device operation. It is enabled or disabled by *CS* so that no operation can occur to change the function unless the device has been selected as the system logic.

$\overline{CS}$: The chip select input is used to enable the communicate between 8253 and themicroprocessor by means of data bus. A low an *CS* enables the data bus buffers, while a high disable the buffer. The *CS* input does not have any affect on the operation of threetimes once they have been initialized. The normal configuration of a system employs an decode logic which actives *CS* line, whenever a specific set of addresses thatcorrespond to 8253 appear on the address bus.

$\overline{RD}$ & $\overline{WR}$:

The read ($\overline{RD}$) and write $\overline{WR}$ pins central the direction of data transfer on the 8-bit bus.

Whenthe input $\overline{RD}$ pin is low. Then CPU is inputting data from 8253 in the form of counter value. When *WR* pins is $\overline{low}$, then CPU is sending data to 8253 in the form of mode information or loading counters. The *RD* &*WR* should not both be low simultaneously. When *RD* & *WR* pins are HIGH, the data bus buffer isdisabled.

A0 & A1:

These two input lines allow the microprocessor to specify which one of the internal register in the 8253 is going to be used for the data transfer. Fig shows how these two lines are used to select either the control word register or one of the 16-bit counters.

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A_1	A_0	operation
0	1	0	0	0	Load counter '0'
0	1	0	0	1	Load counter '1'
0	1	0	1	0	Load counter '2'
0	1	0	1	1	Write mode word
0	0	1	0	0	Read TM_0
0	0	1	0	1	Read TM_1
0	0	1	1	0	Read TM_2
0	0	1	1	1	No- operation 3- state
1	X	X	X	X	Disable -- state
0	1	1	X	X	No- operation 3- state

Control word register:

It is selected when A0 and A1 . It the accepts information from the data bus buffer and stores itin a register. The information stored in then register controls the operation mode of each counter,selection of binary or BCD counting and the loading of each counting and the loading of each count register. This register can be written into, no read operation of this content is available.

Counters:

Each of the times has three pins associated with it. These are CLK (CLK) the gate (GATE) and the output (OUT).

CLK:

This clock input pin provides 16-bit times with the signal to causes the times to decrement max^{m} clock input is 2.6MHz. Note that the counters operate at the negative edge (H1 to L0) of this

clock input. If the signal on this pin is generated by a fixed oscillator then the user has implemented a standard timer. If the input signal is a string of randomly occurring pulses, then it is called implementation of a counter.

GATE:

The gate input pin is used to initiate or enable counting. The exact effect of the gate signal dependson which of the six modes of operation is chosen.

OUTPUT:

The output pin provides an output from the timer. It actual use depends on the mode of operationof the timer. The counter can be read —in the fly| without inhibiting gate pulse or clock input.

CONTROL WORD OF 8253

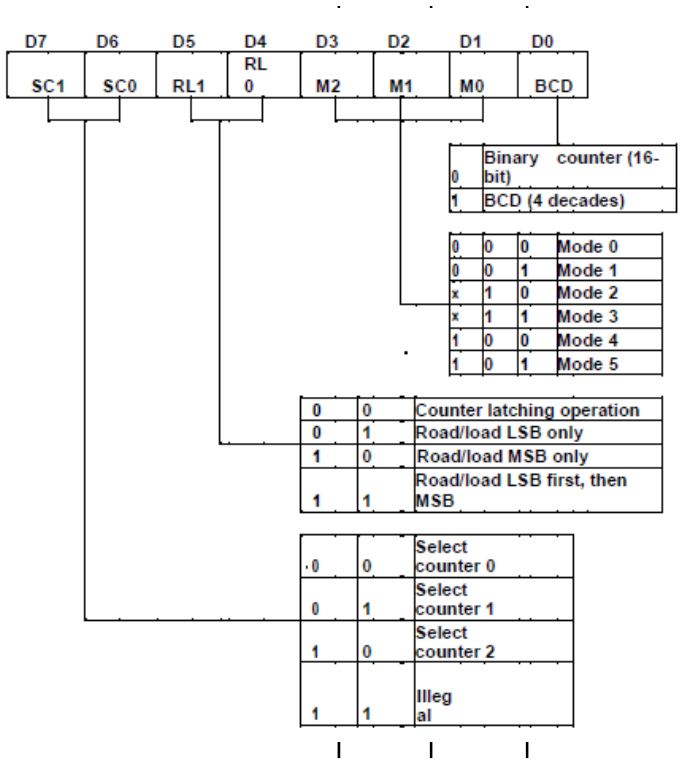


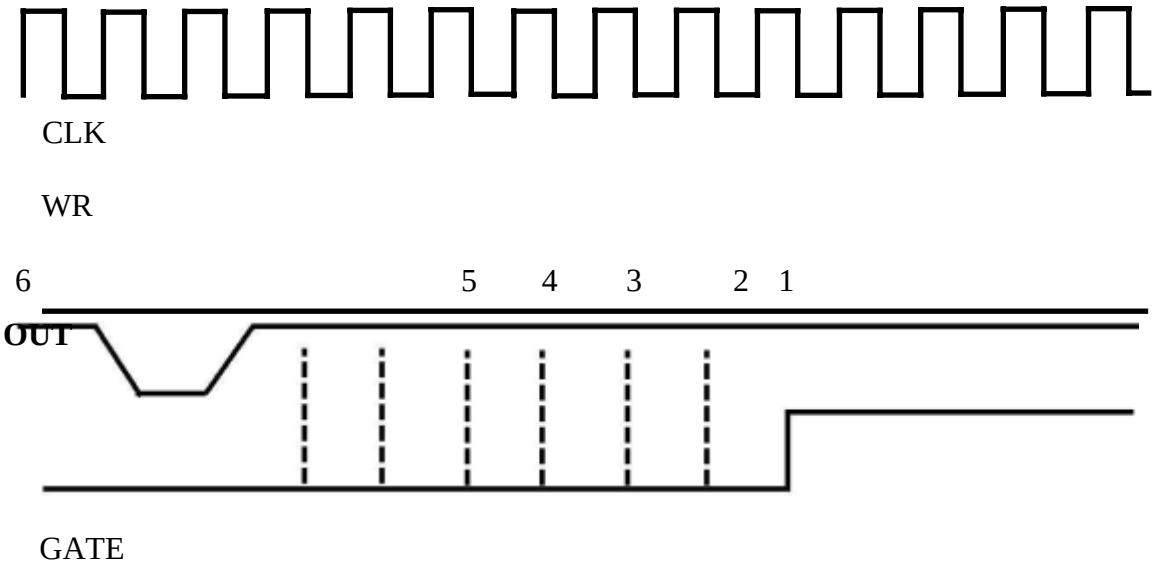
Fig 4.16: Control word format-8253

Control Register

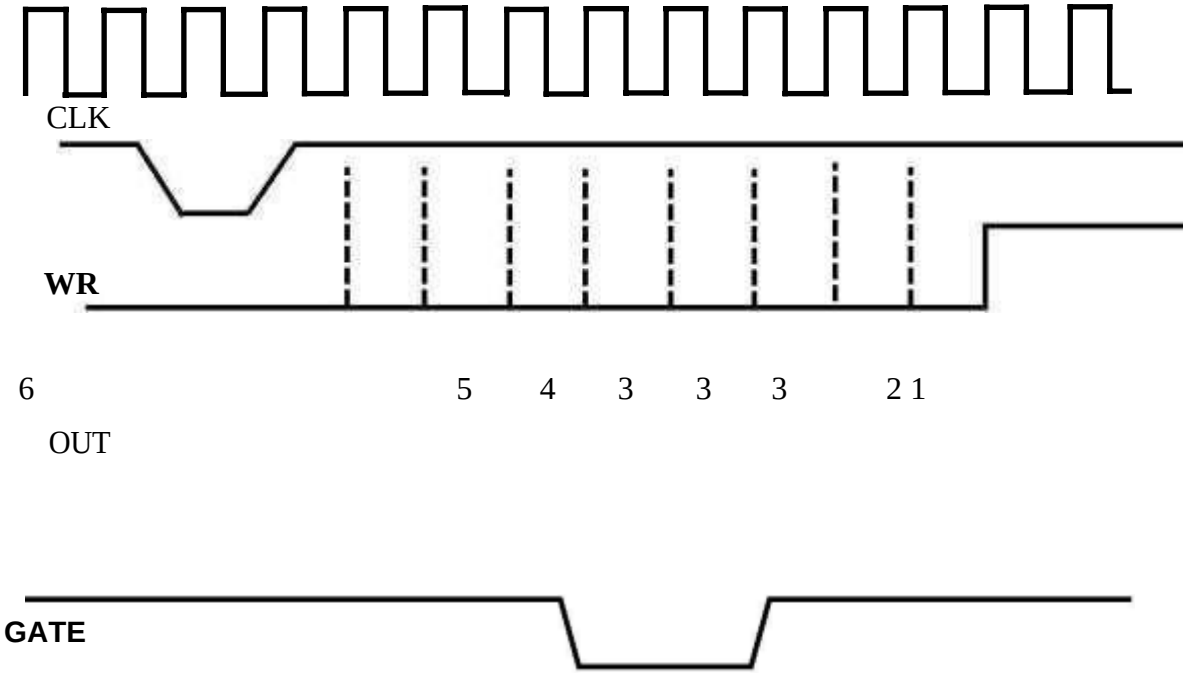
MODES OF OPERATION

Mode 0 Interrupt on terminal count Mode 1 Programmable one shot Mode 2 Rate Generator Mode 3 Square wave rate Generator Mode 4 Software triggered strobe Mode 5 Hardware triggeredstrobe

Mode 0: The output goes high after the terminal count is reached. The counter stops if the Gate islow.. The timer count register is loaded with a count (say 6) when the WR line is made low by the processor. The counter unit starts counting down with each clock pulse. The output goes highwhen the register value reaches zero. In the mean time if the GATE is made low the count is suspended at the value(3) till the GATE is enabled again .



Mode 0 count when Gate is high (enabled)

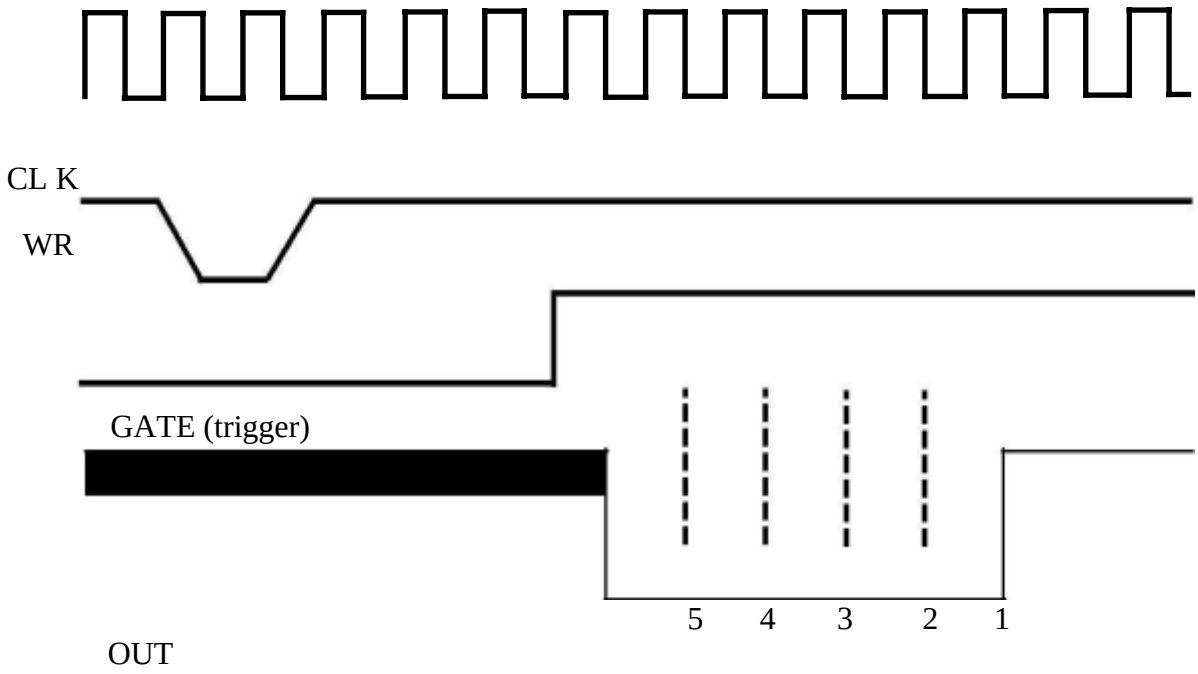


Mode 0 count when Gate is low temporarily (disabled) Mode 1 Programmable mono-shot

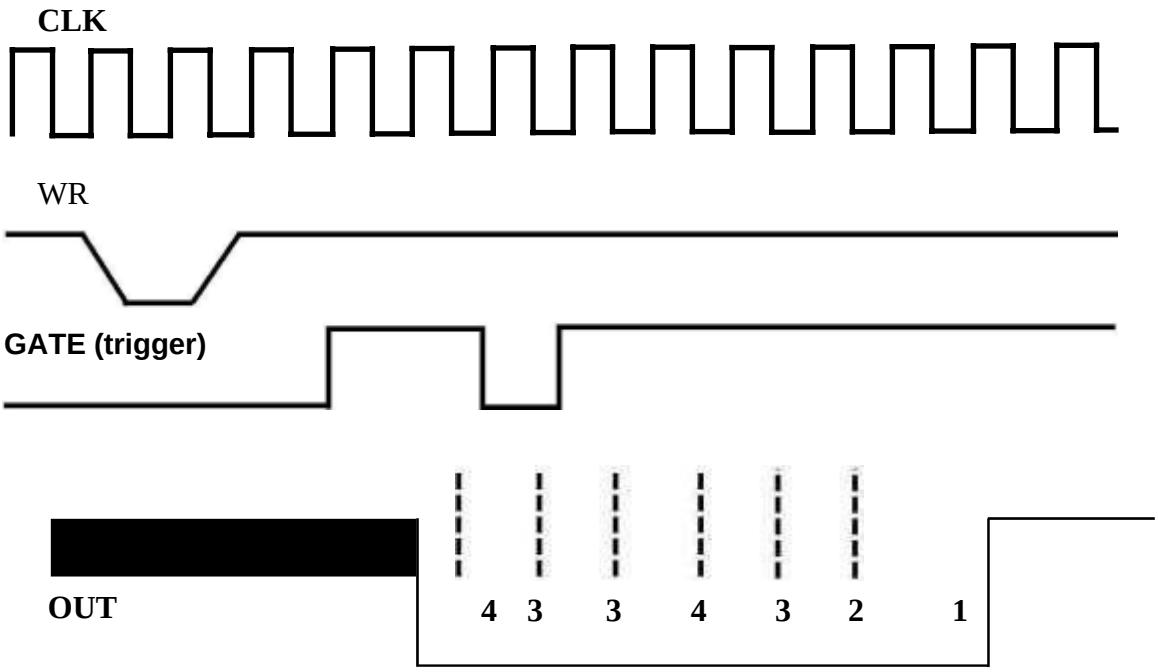
The output goes low with the Gate pulse for a predetermined period depending on the

counter. The counter is disabled if the GATE pulse goes momentarily low.The counter register isloaded with a count value as in the previous case (say 5). The output responds to the GATE inputand goes low for period that equals the count down period of the register (5 clock pulses in this period). By changing the value of this count the duration of the output pulse can be changed. If the GATE becomes low before the count down is completed then the counter will be suspended at that state as long as GATE is low. Thus it works as a mono- shot.

\



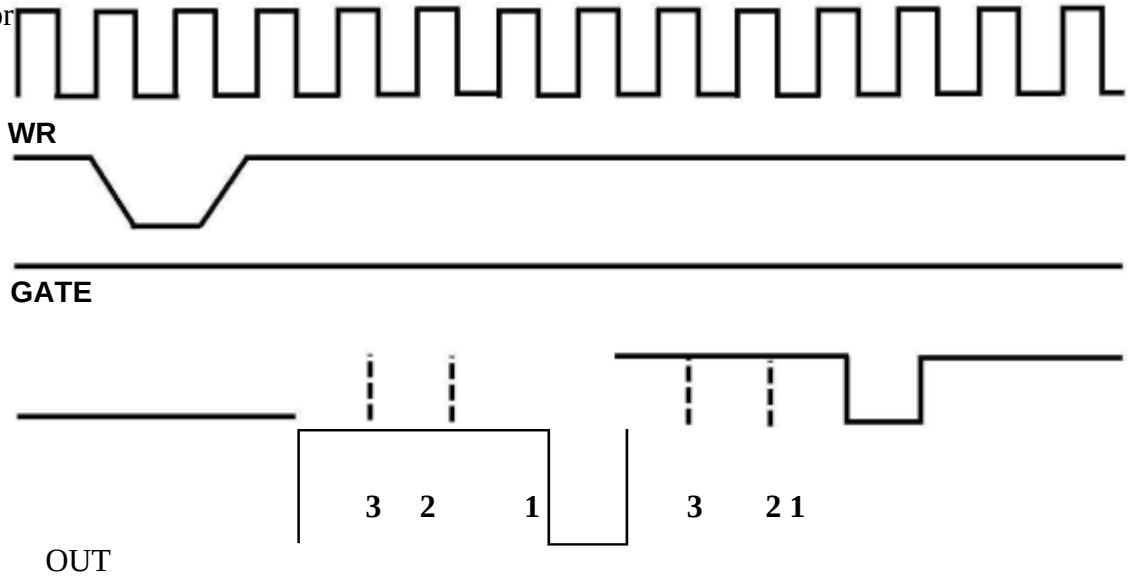
Mode 1 The Gate goes high. The output goes low for the period depending on the count



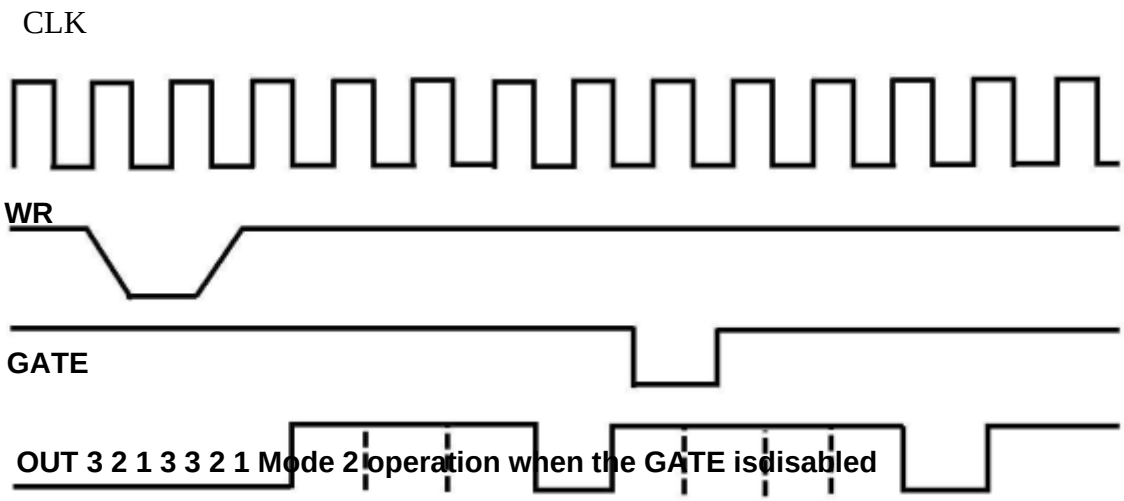
Mode 1 The Gate pulse is disabled momentarily causing the counter to stop.

Mode 2 Programmable Rate Generator

In this mode it operates as a rate generator. The output goes high for a period that equals the timeof count down of the count register (3 in this case). The output goes low exactly for



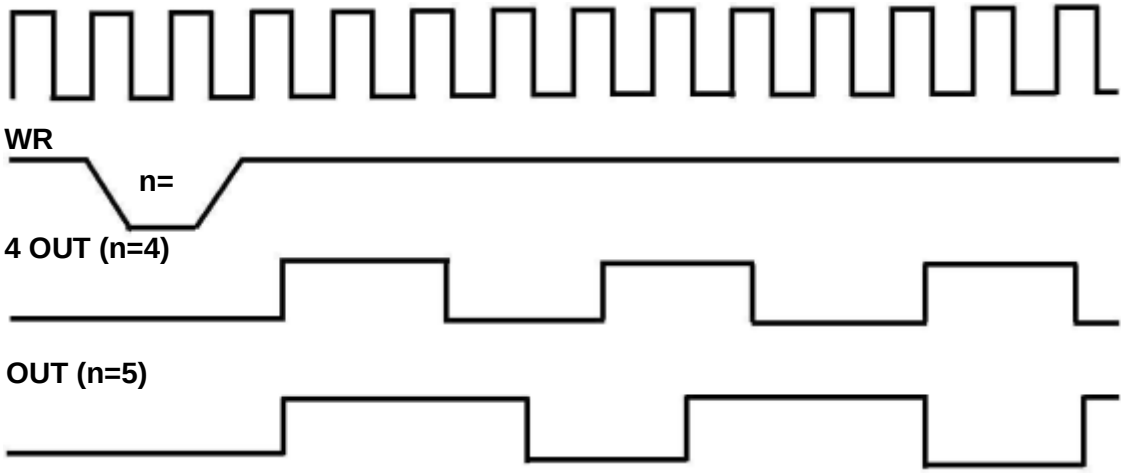
Mode 2 Operation when the GATE is kept high



momentarily.

Mode 3 Programmable Square Wave Rate Generator

It is similar to Mode 2 but the output high and low period is symmetrical. The output goes high after the count is loaded and it remains high for period which equals the count down period of the counter register. The output subsequently goes low for an equal period and hence generates a symmetrical square wave unlike Mode 2. The GATE has no role here.

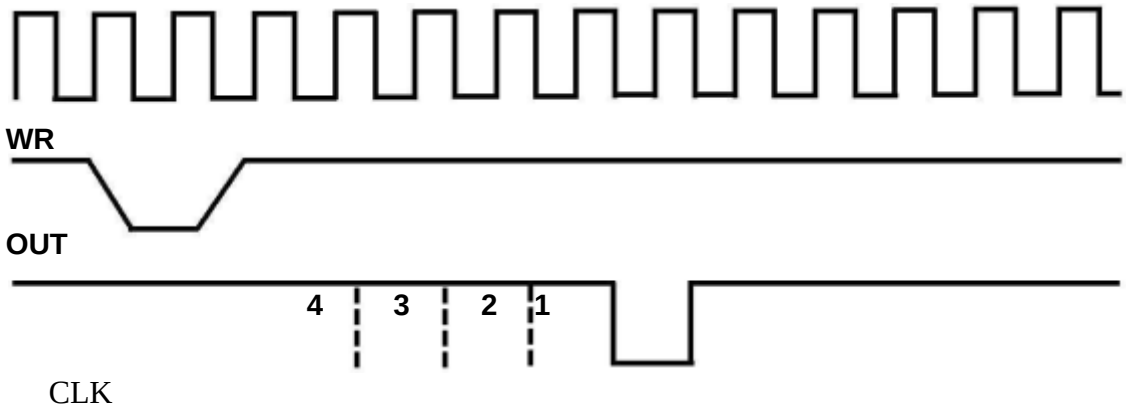


CLK

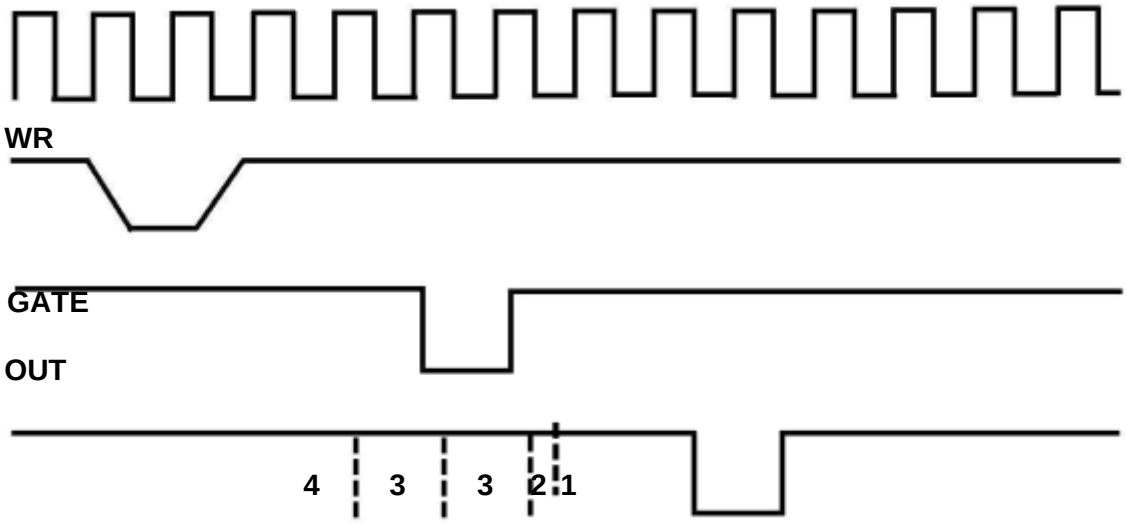
Mode3 Operation: Square Wave generator

Mode 4 Software Triggered Strobe

In this mode after the count is loaded by the processor the count down starts. The output goes low for one clock period after the count down is complete. The count down can be suspended bymaking the GATE low . This is also called a software triggered strobe as the count down is initiated by a program.



Mode 4 Software Triggered Strobe when GATE is high

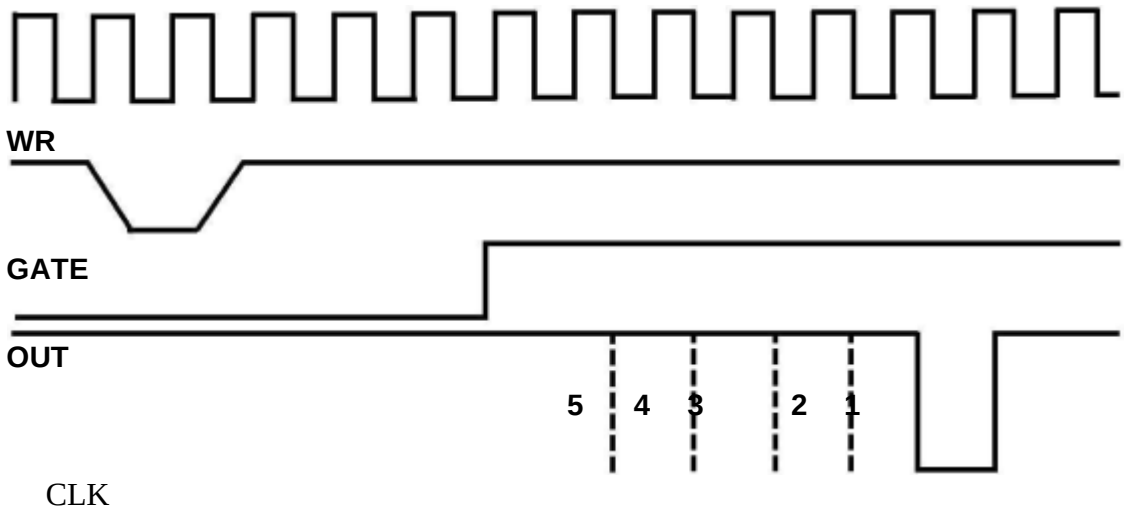


LK

Mode 4 Software Triggered Strobe when GATE is momentarily low

Mode 5 Hardware Triggered Strobe

The count is loaded by the processor but the count down is initiated by the GATE pulse. The transition from low to high of the GATE pulse enables count down. The output goes low for one clock period after the count down is complete.



Mode 5 Hardware Triggered Strobe

4. PROGRAMMABLE INTERRUPT CONTROLLER-8259

FEAUTURES OF 8259

- Eight-Level PriorityController Expandable to 64Levels Programmable Interrupt Modes

- 8086, 8088 Compatible
- MCS-80, MCS-85 Compatible
  - Individual Request Mask Capability Single +5V Supply (No Clocks)
  - Available in 28-Pin DIP and 28-Lead PLCC Package Available in EXPRESS
    1. Standard Temperature Range
    2. Extended Temperature Range

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 a5V 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 8259A is fully upward compatible with the Intel 8259. Software originally written for the 8259 will operate the 8259Ain all 8259 equivalent modes (MCS-80/85, Non-Buffered, Edge Triggered). Pin Diagram of 8259is shown in figure 3.17.

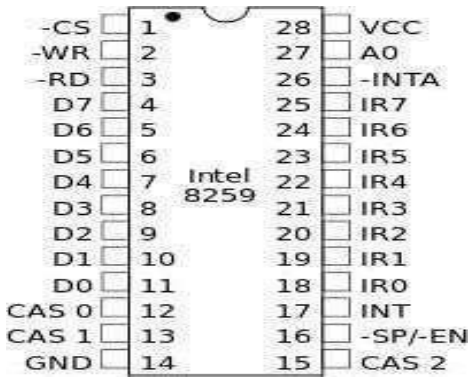


Fig.4.17 **Pin** Diagram of 8259

Pin Description of 8259

Symbol	Pin No.	Type	Name and Function
V_{CC}	28	I	**SUPPLY:** +5V Supply.
GND	14	I	**GROUND**
$\overline{CS}$	1	I	**CHIP SELECT:** A low on this pin enables $\overline{RD}$ and $\overline{WR}$ communication between the CPU and the 8259A. INTA functions are independent of CS.
$\overline{WR}$	2	I	**WRITE:** A low on this pin when CS is low enables the 8259A to accept command words from the CPU.
$\overline{RD}$	3	I	**READ:** A low on this pin when CS is low enables the 8259A to release status onto the data bus for the CPU.
D_7–D_0	4–11	I/O	**BIDIRECTIONAL DATA BUS:** Control, status and interrupt-vector information is transferred via this bus.
CAS_0–CAS_2	12, 13, 15	I/O	**CASCADE LINES:** The CAS lines form a private 8259A bus to control a multiple 8259A structure. These pins are outputs for a master 8259A and inputs for a slave 8259A.
$\overline{SP}/\overline{EN}$	16	I/O	**SLAVE PROGRAM/ENABLE BUFFER:** This is a dual function pin. When in the Buffered Mode it can be used as an output to control buffer transceivers (EN). When not in the buffered mode it is used as an input to designate a master (SP = 1) or slave (SP = 0).
INT	17	O	**INTERRUPT:** This pin goes high whenever a valid interrupt request is asserted. It is used to interrupt the CPU, thus it is connected to the CPU's interrupt pin.
IR_0–IR_7	18–25	I	**INTERRUPT REQUESTS:** Asynchronous inputs. An interrupt request is executed by raising an IR input (low to high), and holding it high until it is acknowledged (Edge Triggered Mode), or just by a high level on an IR input (Level Triggered Mode).
$\overline{INTA}$	26	I	**INTERRUPT ACKNOWLEDGE:** This pin is used to enable 8259A interrupt-vector data onto the data bus by a sequence of interrupt acknowledge pulses issued by the CPU.
A_0	27	I	**AO ADDRESS LINE:** This pin acts in conjunction with the $\overline{CS}$, $\overline{WR}$, and $\overline{RD}$ pins. It is used by the 8259A to decipher various Command Words the CPU writes and status the CPU wishes to read. It is typically connected to the CPU A0 address line (A1 for 8086, 8088).

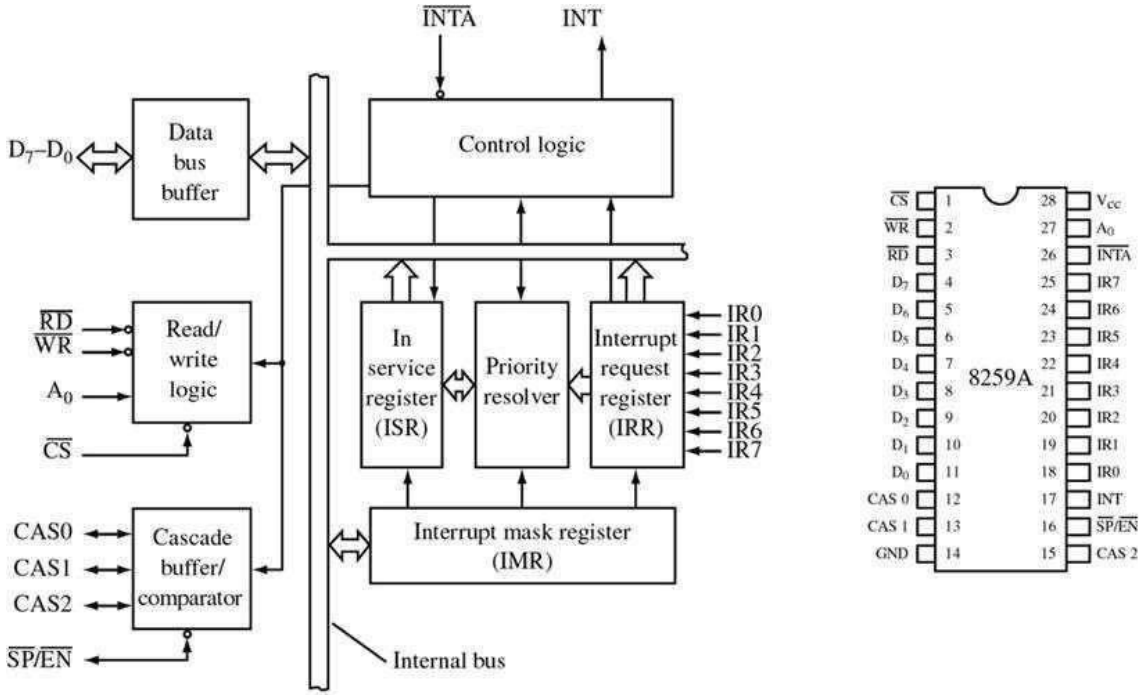


Fig. 4.18 Block Diagram of 8259

A more desirable method would be one that would allow the microprocessor to be executing its main program and only stop to service peripheral devices when it is told to do so bythe device itself. In effect, the method would provide an external asynchronous input that would inform the processor that it should complete whatever instruction that is currently being executedand fetch a new routine that will service the requesting device. Once this servicing is complete, however, the processor would resume exactly where it left off. This method is called Interrupt. It is easy to see that system throughput would drastically increase, and thus more tasks could be assumed by the micro-computer to further enhance its cost effectiveness. Block Diagram of 8259is shown in figure 18.

The Programmable Interrupt Controller (PIC) functions as an overall manager in an Interrupt-Driven system environment. It accepts requests from the peripheral equipment, determines which of the in-coming requests is of the highest importance (priori-ty),

ascertains whether the incoming request has a higher priority value than the level currently being serviced, and issues an interrupt to the CPU based on this determination.

The 8259A is a device specifically designed for use in real time, interrupt driven microcomputer systems. It manages eight levels or requests and has built-in features for expandability to other 8259A's (up to 64 levels). It is programmed by the system's software as an I/O peripheral. A selection of priority modes is available to the programmer so that the manner in which the requests are processed by the 8259A can be configured to match his system requirements. The priority modes can be changed or reconfigured dynamically at any time during the main program. This means that the complete interrupt structure can be defined as required, based on the total system environment.

5. Interrupt Request Register (Irr) And In-Service Register (Isr)

The interrupts at the IR input lines are handled by two registers in cascade, the Interrupt Request Register (IRR) and the In-Service (ISR). The IRR is used to store all the interrupt levels which are requesting service; and the ISR is used to store all the interrupt levels which are being serviced.

PRIORITY RESOLVER

This logic block determines the priorites of the bits set in the IRR. The highest priority is selected and strobed into the corresponding bit of the ISR during INTA pulse.

INTERRUPT MASK REGISTER (IMR)

The IMR stores the bits which mask the interrupt lines to be masked. The IMR operates on the IRR. Masking of a higher priority input will not affect the interrupt request lines of lower quality.

INT (INTERRUPT)

This output goes directly to the CPU interrupt input. The VOH level on this line is designed to befully compatible with the 8080A, 8085A and 8086 input levels.

INTA (INTERRUPT ACKNOWLEDGE)

INTA pulses will cause the 8259A to release vectoring information onto the data bus. The formatof this data depends on the system mode (mPM) of the 8259A.

DATA BUS BUFFER

This 3-state, bidirectional 8-bit buffer is used to inter-face the 8259A to the system Data Bus. Control words and status information are transferred through the Data Bus Buffer.

READ/WRITE CONTROL LOGIC

The function of this block is to accept Output commands from the CPU. It contains the Initialization Command Word (ICW) registers and Operation Command Word (OCW) registers which store the various control formats for device operation. This function block also allows the status of the 8259A to be transferred onto the Data Bus.

CS (CHIP SELECT)

A LOW on this input enables the 8259A. No reading or writing of the chip willoccur unless the device is selected.

WR (WRITE)

A LOW on this input enables the CPU to write con-trol words (ICWs and OCWs)
to the8259A. RD (READ)

A LOW on this input enables the 8259A to send the status of the Interrupt Request Register (IRR), In Service Register (ISR), the Interrupt Mask Register (IMR), or the Interrupt level onto the Data Bus.

A0

This input signal is used in conjunction with WR and RD signals to write commands into the various command registers, as well as reading the various status registers of the chip. This line can be tied directly to one of the address lines.

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 specificinterrupt 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 MCS-80/85 sys-tem:

1. One or more of the INTERRUPT REQUEST lines (IR7±0) are raised high, setting the correspond-ing 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 correspond-ing 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 re-lease its preprogrammed subroutine address onto the Data Bus. The lower 8-bit address is released at the first INTA pulse andthe higher 8-bit address is released at the second INTA pulse.
7. This completes the 3-byte CALL instruction re-leased by the 8259A. In the AEOI mode the ISR bit is reset at the end of the third INTA pulse. Otherwise, the ISR

bit remains set until an appropriate EOI command is issued at the end of the interruptsequence.

8. The events occurring in an 8086 system are the same until step 4.
9. Upon receiving an INTA from the CPU group, the highest priority ISR bit is set and the corresponding IRR bit is reset. The 8259A does not drive the Data Bus during thiscycle.
10. The 8086 will initiate a second INTA pulse. During this pulse, the 8259A releases an 8- bit pointer onto the Data Bus where it is read by the CPU.
11. This completes the interrupt cycle. In the AEOI mode the ISR bit is reset at the end of the second INTA pulse. Otherwise, the ISR bit remains set until an appropriate EOI command is issued at the end of the interrupt subroutine.

If no interrupt request is present at step 4 of either sequence (i.e., the request was too short in duration) the 8259A will issue an interrupt level 7. Both the vectoring bytes and the CASlines will look like an interrupt level 7 was requested.

When the 8259A PIC receives an interrupt, INT be-comes 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 un-specified 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. The designer should be aware of this consideration when designing a sys-tem which usesthe 8259A. It is recommended that proper asynchronous design techniques be followed.

INITIALIZATION COMMAND WORDS

Whenever a command is issued with A0 e 0 and D4 e 1, this is interpreted as Initialization Command Word 1 (ICW1). ICW1 starts the initialization sequence during which the following automatically occur.

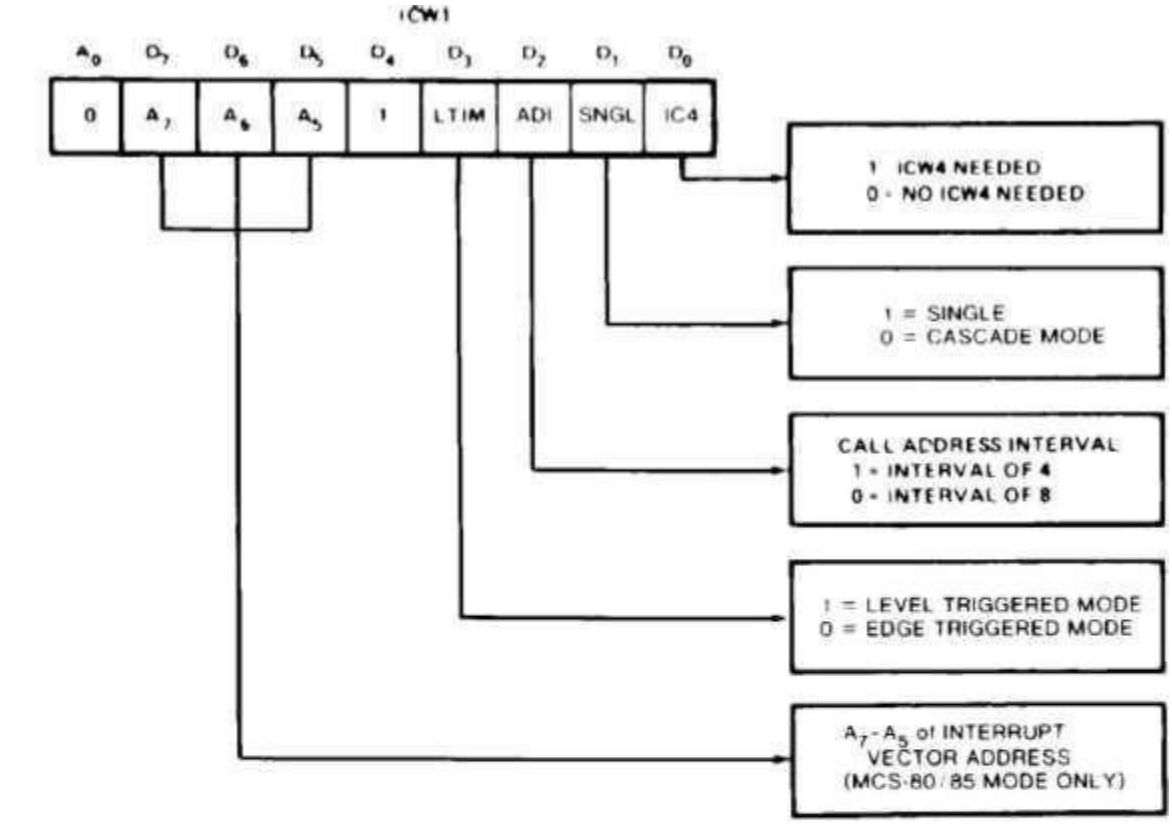
a. The edge sense circuit is reset, which means that following initialization, an interrupt request (IR) input must make a low-to-high transition to generate an interrupt.
b. The Interrupt Mask Register is cleared.
c. IR7 input is assigned priority 7.

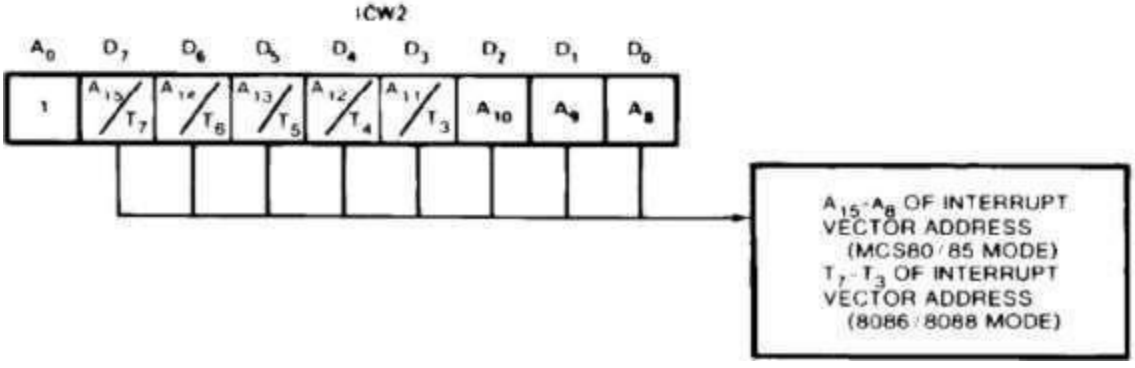
d. The slave mode address is set to 7.

e. Special Mask Mode is cleared and Status Read isset to IRR.

f. If IC4 e 0, then all functions selected in ICW4are set to zero. (Non-Buffered mode*, no Auto-EOI, MCS-80, 85 system).

Initialization Command Word Format is as shown infigure





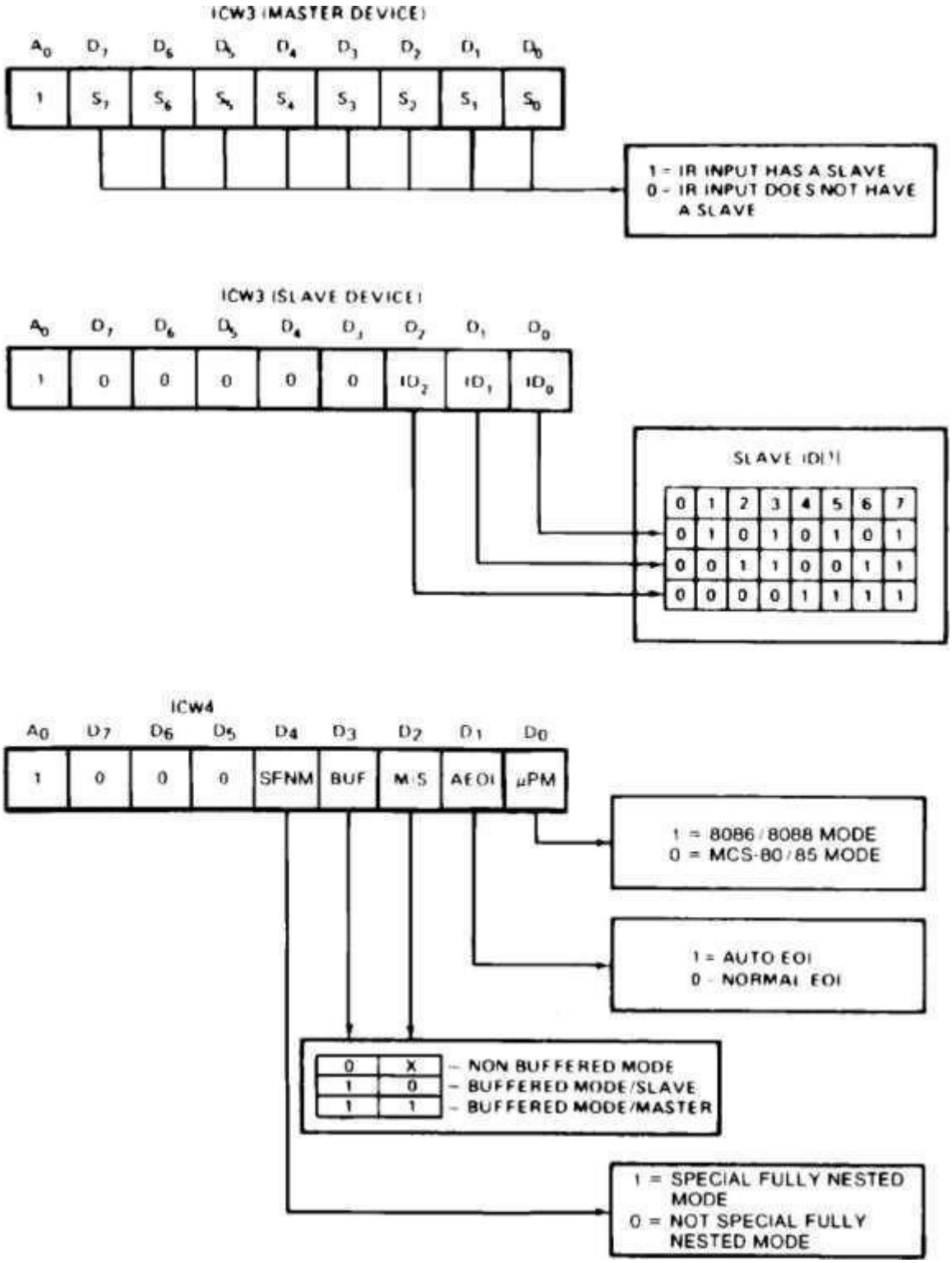
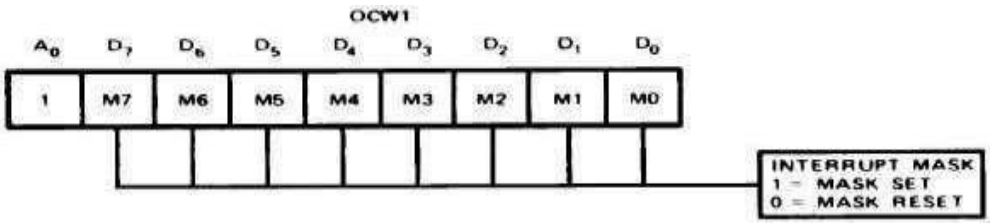


Fig 4.19 . Initialization Command Word Format

OPERATION COMMAND WORDS

After the Initialization Command Words (ICWs) are programmed into the 8259A, the chip is ready to accept interrupt requests at its input lines. However, during the 8259A operation, a selection of algorithms can command the 8259A to operate in various modes through the Operation Command Words (OCWs). Operation Command Word format is as shown in figure

Fig 4.20 a. Operational Control Words



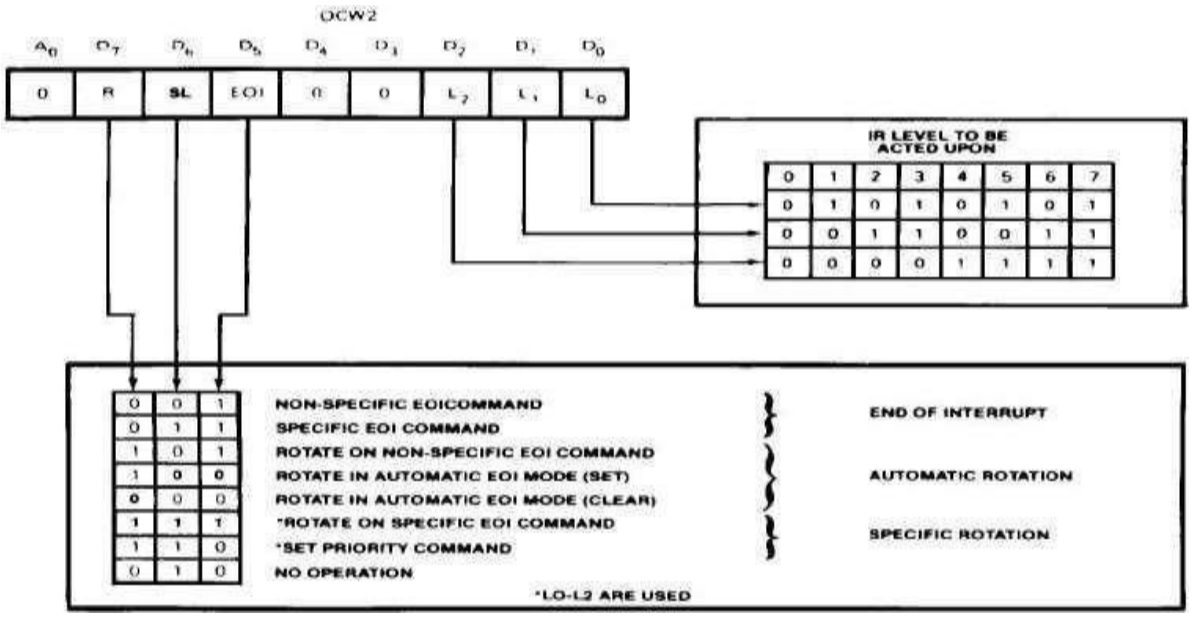


Fig 4.20 b. Operation Command Word Format

INTERFACING MEMORY CHIPS WITH 8085

8085 has 16 address lines (A0 - A15), hence a maximum of 64 KB (= 2^{16} bytes) of memory locations can be interfaced with it. The memory address space of the 8085 takes values from0000H to FFFFH.

The 8085 initiates set of signals such as IO/M , RD and WR when it wants to read from and write into memory. Similarly, each memory chip has signals such as CE or CS (chip enable or chip select), OE or RD (output enable or read) and WE or WR (write enable or write) associated with it.

Generation of Control Signals for Memory:

When the 8085 wants to read from and write into memory, it activates IO/M , RD and WR signals as shown in Table .

Table 8 Status of IO/M , RD and WR signals during memory read and write operations

$IO/\overline{M}$	$\overline{RD}$	$\overline{WR}$	Operation
0	0	1	8085 reads data from memory
0	1	0	8085 writes data into memory

Using IO/M , RD and WR signals, two control signals MEMR (memory read) and MEMW (memory write) are generated. Fig. 16 shows the circuit used to generate thesesignals.

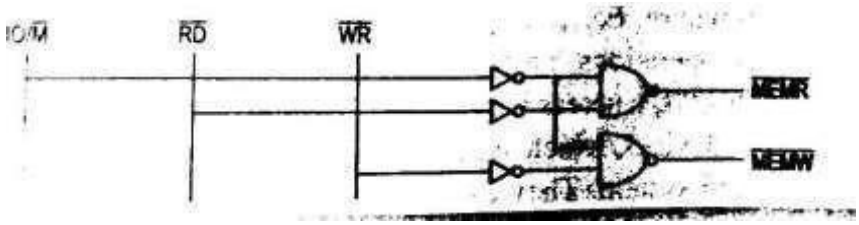


Fig. 4.21 Circuit used to generate MEMR and MEMW signals

When is IO/M high, both memory control signals are deactivated irrespective of the statusof $\overline{RD}$ and WR signals.

Ex: Interface an IC 2764 with 8085 using NAND gate address decoder such that theaddress range allocated to the chip is 0000H – 1FFFH.

Specification of IC 2764:

- 8 KB (8 x 2^{10} byte) EPROM chip
- 13 address lines (2^{13} bytes = 8 KB) Interfacing:

- 13 address lines of IC are connected to the corresponding address lines of
- 8085. Remaining address lines of 8085 are connected to address decoder formed using logic gates, the output of which is connected to the CE pin of
- IC. Address range allocated to the chip is shown in Table 9.
- Chip is enabled whenever the 8085 places an address allocated to EPROM chip in the address bus. This is shown in Fig. 17.

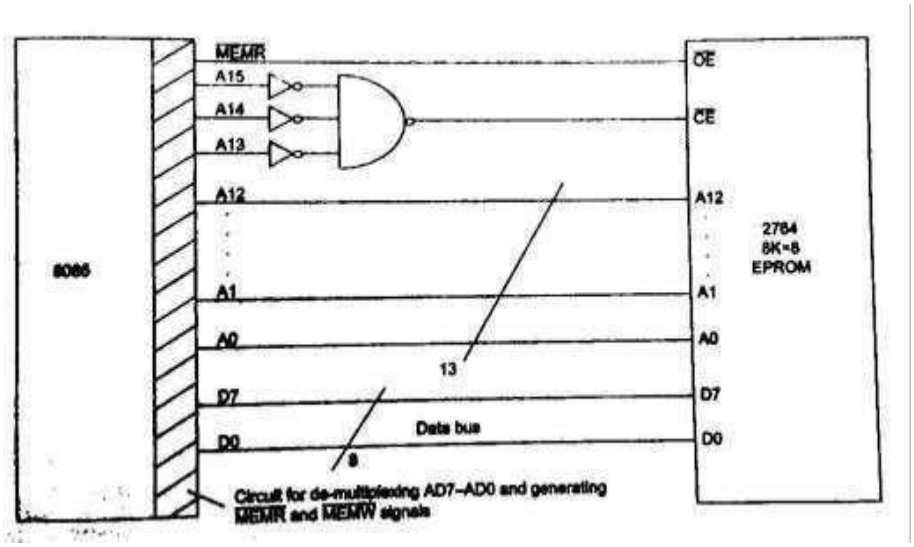


Fig. 4.22 Interfacing IC 2764 with the 8085 Table 9 Address allocated to IC 2764

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Address
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000H
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0001H
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1FFEH
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFFH

Ex: Interface a 6264 IC (8K x 8 RAM) with the 8085 using NAND gate decoder suchthat the starting address assigned to the chip is 4000H.

Specification of IC 6264:

- 8K x 8 RAM

- 8 KB = 2^{13} bytes
  13 address lines

The ending address of the chip is 5FFFH (since 4000H + 1FFFH = 5FFFH). When the address 4000H to 5FFFH are written in binary form, the values in the lines A15, A14, A13 are 0, 1 and 0 respectively. The NAND gate is designed such that when the lines A15 and A13 carry 0 and A14 carries 1, the output of the NAND gate is 0. The NAND gate output is in turn connected to the CE1 pin of the RAM chip. A NAND output of 0 selects the RAM chip for read or write operation, since CE2 is already 1 because of its connection to +5V. Fig. 18 shows the interfacing of IC 6264 with the 8085.

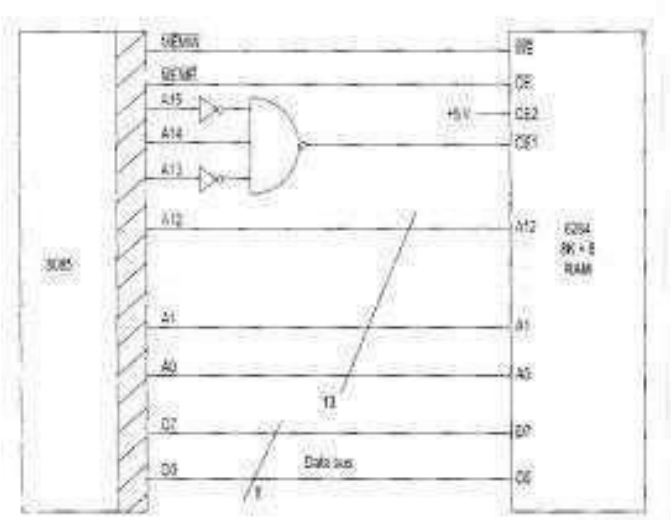


Fig. 4.23 Interfacing 6264 IC with the 8085

Ex: Interface two 6116 ICs with the 8085 using 74LS138 decoder such that the starting addresses assigned to them are 8000H and 9000H, respectively.

Specification of IC 6116:

- 2 K x 8 RAM
- 2 KB = 2^{11} bytes
  11 address lines

6116 has 11 address lines and since 2 KB, therefore ending addresses of 6116 chip 1 is and chip 2 are 87FFH and 97FFH, respectively. Table 10 shows the address range of the two chips.

Table 4.1 Address range for IC 6116

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Address
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8000H
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	87FFH (RAM chip 1)
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	9000H
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
1	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	97FFH (RAM chip 2)

Interfacing:

- Fig. 19 shows the interfacing.
- A0 – A10 lines of 8085 are connected to 11 address lines of the RAMchips.
- Three address lines of 8085 having specific value for a particular RAM are connected to the three select inputs (C, B and A) of 74LS138 decoder.
- Table 10 shows that A13=A12=A11=0 for the address assigned to RAM 1 and A13=0, A12=1 and A11=0 for the address assigned to RAM 2.
- Remaining lines of 8085 which are constant for the address range assigned to thetwo RAM are connected to the enable inputs of decoder.
- When 8085 places any address between 8000H and 87FFH in the address bus, the select inputs C, B and A of the decoder are all 0. The Y0 output of the

- When 8085 places any address between 9000H and 97FFH in the address bus, the select inputs C, B and A of the decoder are 0, 1 and 0. The Y2 output of the decoder is also 0, selecting RAM 2.

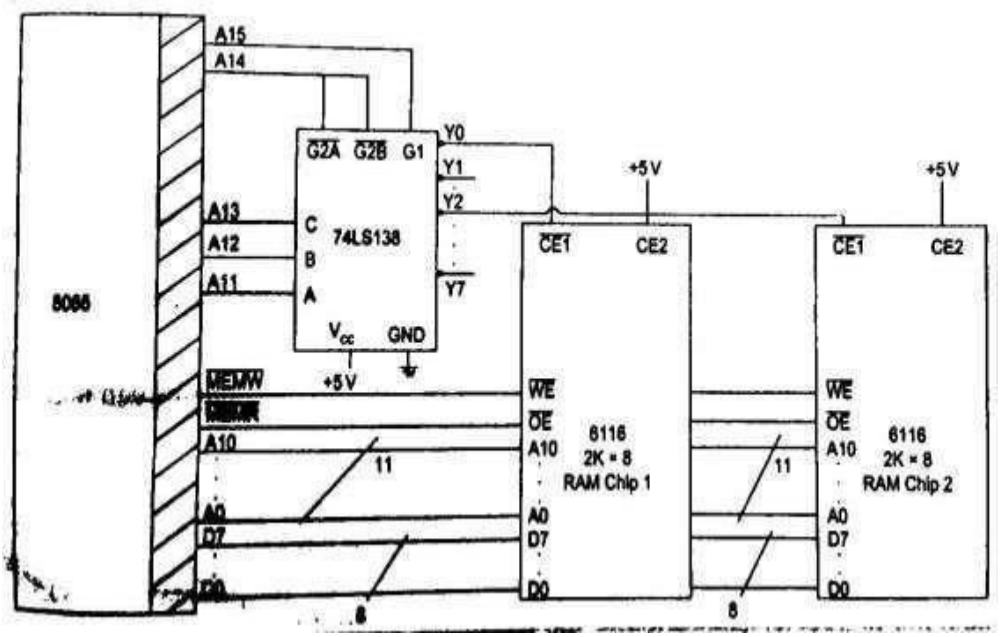


Fig. 4.24 Interfacing two 6116 RAM chips using 74LS138 decoder

3. PERIPHERAL MAPPED I/O INTERFACING

In this method, the I/O devices are treated differently from memory chips. The control signals I/O read (IOR) and I/O write (IOW), which are derived from the IO/M , RD and

WR signals of the 8085, are used to activate input and output devices, respectively.

Generation of these control signals is shown in Fig. 20. Table 11 shows the status of IO/M ,RD and WR signals during I/O read and I/O write operation.

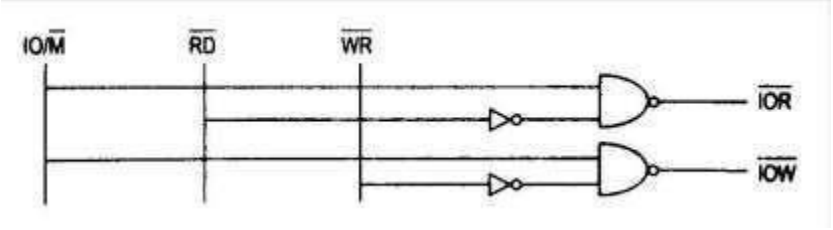


Fig. 4.25 Generation of IOR and IOW signals

IN instruction is used to access input device and OUT instruction is used to access output device. Each I/O device is identified by a unique 8-bit address assigned to it. Since the control signals used to access input and output devices are different, and all I/O device use 8-bit address, a maximum of 256 (2^8) input devices and 256 output devices can be interfaced with 8085.

Table 4.2 Status of IOR and IOW signals in 8085.

IO/M	RD	WR	IOR	IOW	Operation
1	0	1	0	1	I/O read operation
1	1	0	1	0	I/O write operation
0	X	X	1	1	Memory read or write operation

Ex: Interface an 8-bit DIP switch with the 8085 such that the address assigned to the DIPswitch if F0H.

IN instruction is used to get data from DIP switch and store it in accumulator.
Stepsinvolved in the execution of this instruction are:

- Address F0H is placed in the lines A0 – A7 and a copy of it in lines A8 –A15.

ii. The $\overline{IOR}$ signal is activated (IOR = 0), which makes the selected input device to place its data in the data bus.

iii. The data in the data bus is read and store in the accumulator.

Fig. 3.26 shows the interfacing of DIP switch.

A7	A6	A5	A4	A3	A2	A1	A0	
1	1	1	1	0	0	0	0	= F0H

A0 – A7 lines are connected to a NAND gate decoder such that the output of NAND gate is 0. The output of NAND gate is ORed with the IOR signal and the output of OR gateis

connected to $\overline{1}$G and $\overline{2}$G of the 74LS244. When 74LS244 is enabled, data from the DIP switch is placed on the data bus of the 8085. The 8085 read data and store in the accumulator. Thus data from DIP switch is transferred to the accumulator.

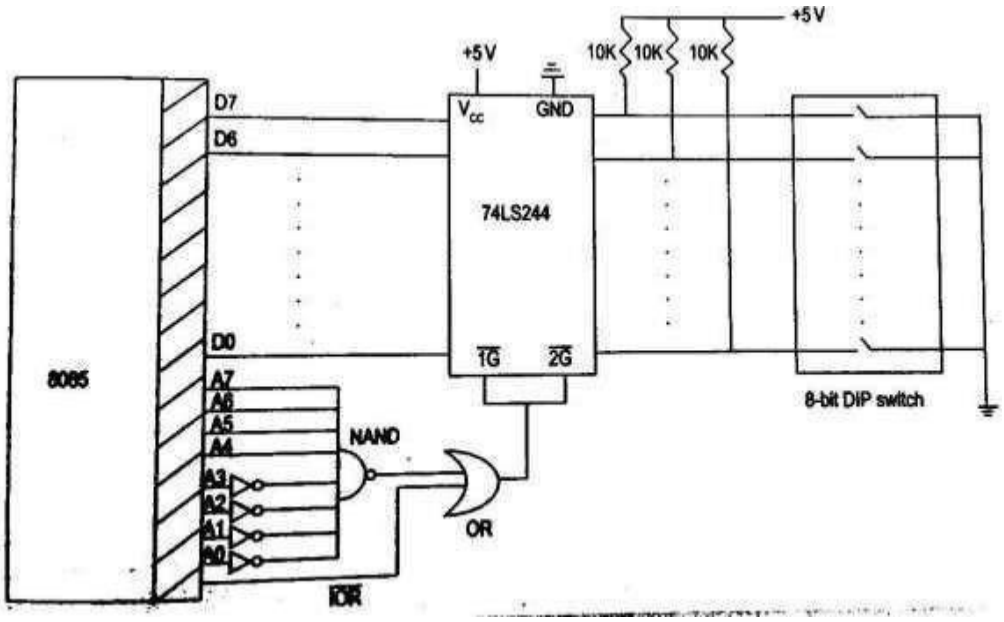


Fig. 4.26 interfacing of 8-bit DIP switch with 8085

6. Memory Mapped I/O Interfacing

In memory-mapped I/O, each input or output device is treated as if it is a memory location. The MEMR and MEMW control signals are used to activate the devices. Each input or output device is identified by unique 16-bit address, similar to 16-bit address assigned to memory location. All memory related instruction like LDA 2000H, LDAX B, MOV A, M can be used.Since the I/O devices use some of the memory address space of 8085, the maximum memory capacity is lesser than 64 KB in this method. Ex: Interface an 8- bit DIP switch with the 8085 using logic gates such that the address assigned to it is F0F0H. Since a 16-bit address has to be assigned to a DIP switch, the memory- mapped I/O technique must be used. Using LDA F0F0H instruction, the data from the 8-bit DIP switch can be transferred to the accumulator. The steps involved are:

- The address F0F0H is placed in the address bus A0 – A15.The MEMR signal is made low for some time.
- The data in the data bus is read and stored in the accumulator.

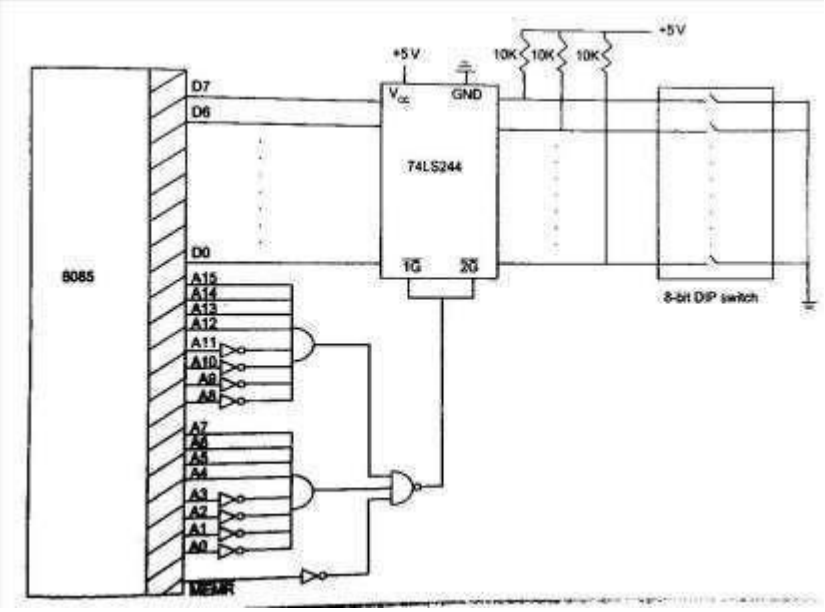


Fig. 4.27 shows the interfacing diagram.

When 8085 executes the instruction LDA F0F0H, it places the address F0F0Hin the address lines A0 – A15 as:

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	= F0F0H

The address lines are connected to AND gates. The output of these gates along with MEMR signal are connected to a NAND gate, so that when the address F0F0H is placed in the address bus and MEMR = 0 its output becomes 0, thereby enabling the buffer 74LS244. The data from the DIP switch is placed in the 8085 data bus. The 8085 reads the data from the data bus and stores it in the accumulator.

nterfacing 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.

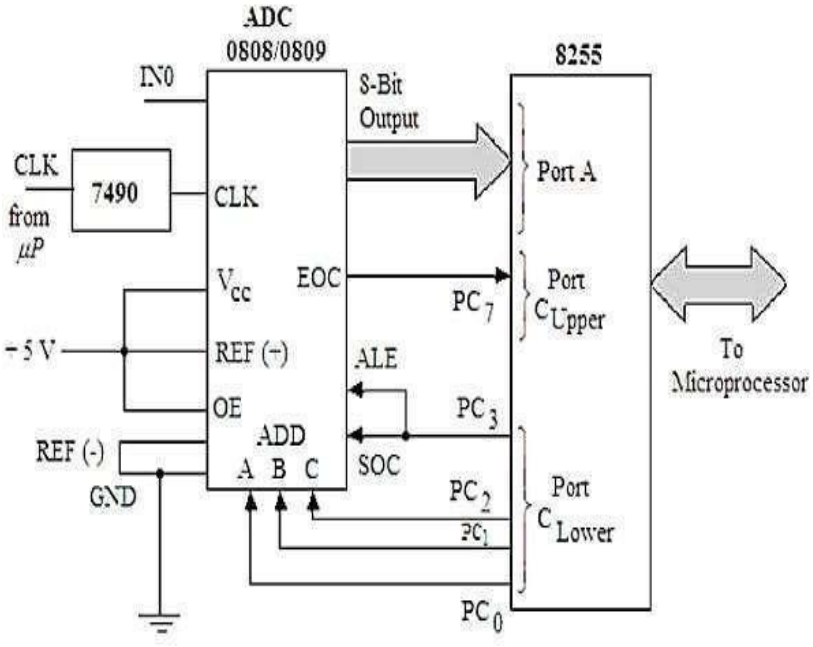


Fig 4.28: ADC interfacing

The PortA 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.

Program

```
        MVI A, 98H ; Set Port A and Cupper as input, CLower as
        output OUT 03H ; Write control word 8255-I to control Word
        register XRA A ; Clear the accumulator
        OUT 02H ; 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
```

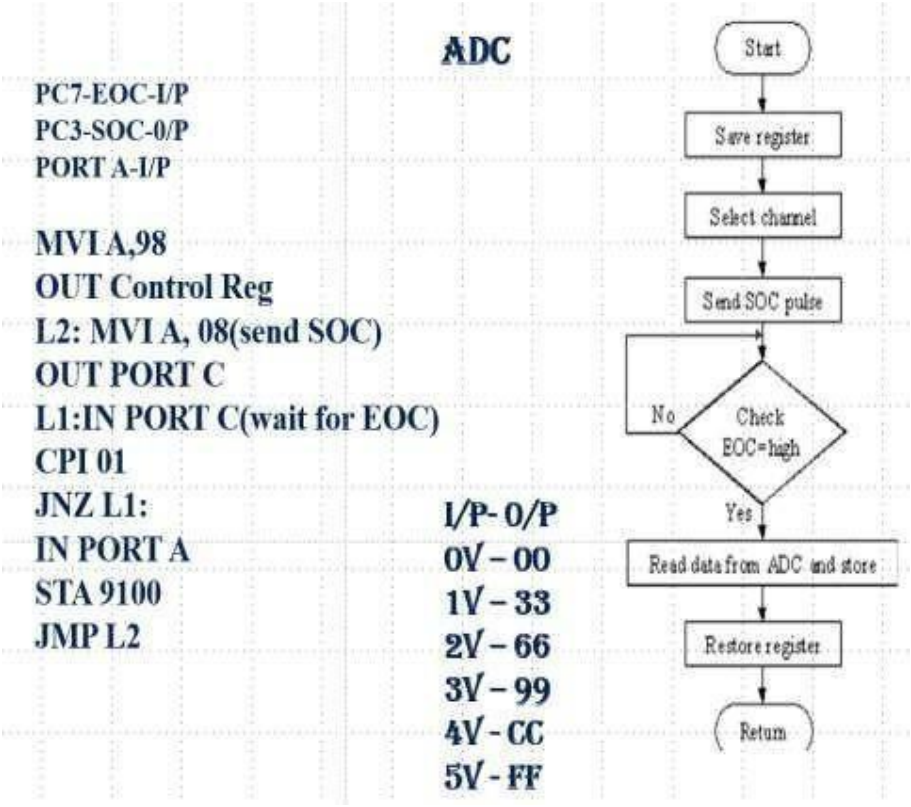


Fig 4.29: Flow chart-

ADC Either of the method can write the program.

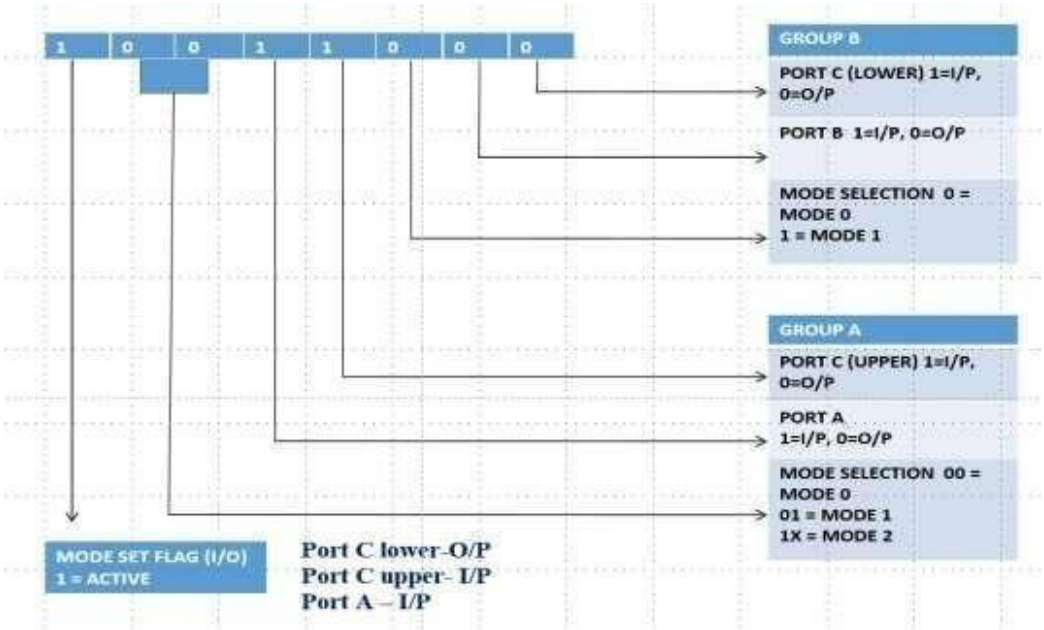


Fig 4.30: control word format

DAC

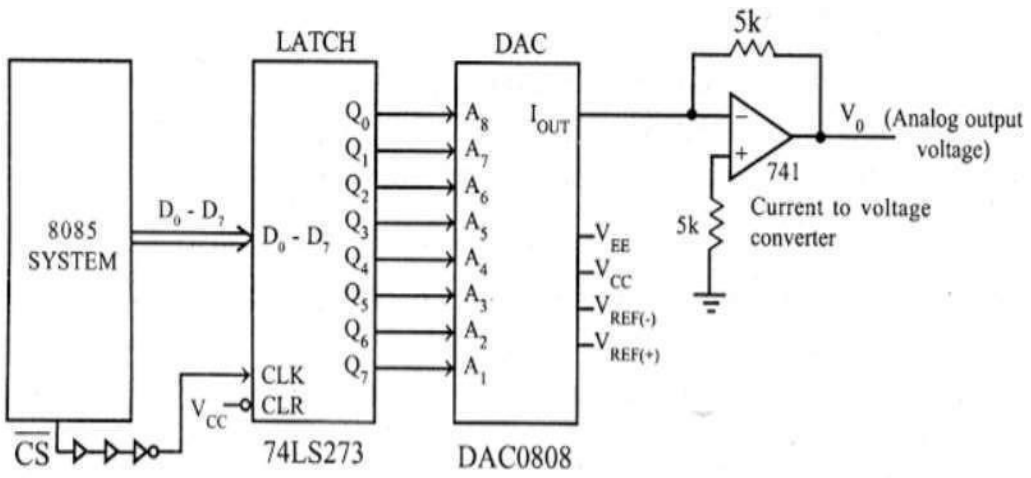


Fig 4.31: DAC Interfacing

- The processor sends an address, which is decoded by decoder in the microprocessor system to produce chip select signal.
- Then the processor sends a digital data to latch. The buffer and inverter will produce sufficient delay for CS signal so that, the latch is clocked only after the data is arrived at the input lines of the latch.
- When the latch is clocked the digital data is send to DAC. The DAC will produce a corresponding current signal, which is converted to voltage signal by the op-amp 741.
- The typical settling time of DAC0800 is 150nsec. Therefore the processor need not wait for loading next data

PROGRAMS FOR VARIOUS WAVEFORM GENERATION USING DAC

SAW TOOTH

```
L1:MVI A,00
OUT DAC
INR A
JMP L1:
```

TRIANGULAR

```
L1:MVI A,00
OUT DAC
INR A
CPI FF
JNZ L1:
L2:OUT DAC
DCR A
JNZ L2
JMP L1:
```

SQUARE WAVE

```
L1:MVI A,00
OUT DAC
CALL DELAY
MVI A, FF
OUT DAC
CALL DELAY
JMP L1:

DELAY
MVI B,55
L2:DCR B
JNZ L2:
RET
```

STAIR CASE

```
L1:LXI H,9100
MVI C, 06
L2:MOV A, M
OUT DAC
CALL DELAY
INX H
DCR C
JNZ L2:
JMP L1:

9100: 00
9101: 55
9102: AA
9103: FF
9104: AA
9105: 55
```

QUESTION BANK

PART A

1. What is interfacing
2. Distinguish memory mapped I/O and I/O mapped I/O
3. Draw the control word for 8255
4. Configure 8255 as PORT A-I/P, PORT B-O/P & PORT C LOWER-I/P , PORTCUPPER-O/P
5. Set PCO using bit set reset mode
6. Write the control word to generate square wave using 8253
7. What is the need of Priority resolver in 8259
8. How many interrupts maximum a 8259 can support
9. What is USART
10. Define resolution in DAC and ADC
11. What is EOC and SOC in ADC
12. Write an ALP to generate sawtooth using DAC
13. What are the two command words used in 8259
14. Explain mode 5 of 8253
15. Explain the transmitter section of 8251 USART

PART B

1. Explain with neat diagram 8255 PPI
2. With neat diagram explain how serial communication is done using 8251
3. With neat diagram explain the 8253 timer
4. Explain the various modes of 8253 timer
5. Discuss about 8259 PIC
6. Interface ADC to 8085 and explain
7. Interface DAC with 8085 and generate various waveforms

TEXT / REFERENCE BOOKS

1. Ramesh Goankar, "Microprocessor architecture programming and applications with 8085 / 8088", 5th Edition, Penram International Publishing, 2002.
2. Mazidi & McKinlay, “The 8051 Microcontroller and Embedded Systems using Assembly and C”, PHI, 2007.
3. MykePredko, “Programming and Customizing the 8051 Micro-controller”, Tata McGraw-Hill edition, 2007.
4. R A Gaonkar, “Fundamentals of Microcontrollers and Applications in Embedded Systems (with the PIC18 Microcontroller Family)”, Penram Publishing India, 2007
. 5. Kenneth Ayala ,”The 8051 Microcontroller”, 3rd Edition, Thomson Delmar Learning, 2004.
6. Kenneth J. Ayala, Dhananjay V. Gadre, “The 8051 Microcontroller & Embedded Systems Using Assembly and C”, Cengage Learning India Publication, 2007.
7. Ajay V Deshmukh, “Microcontrollers: Theory and Applications”, Tata McGraw-Hill, 2005
.8. Raj Kamal, “Embedded Systems Architecture, Programming, and Design”. (2/e), Tata McGraw Hill, 2008.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF ELECTRICAL AND ELECTRONICS ENGINEERING

DEPARTMENT OF ELECTRONICS AND INSTRUMENTATION ENGINEERING

UNIT V- MICROPROCESSORS,MICROCONTROLLERS AND EMBEDDED SYSTEMS: SEIA 1504

I INTRODUCTION TO EMBEDDED SYSTEM

Embedded system- characteristics of embedded system- categories of embedded system- requirements of embedded systems- challenges and design issues of embedded system- trends in embedded system- system integration- hardware and software partition- applications of embedded system- control system and industrial automation-biomedical- data communication system- network information appliances- IVR systems- GPS systems

1.1INTRODUCTION TO EMBEDDED SYSTEM

An embedded system is one kind of a computer system mainly designed to perform several tasks like to access, process, store and also control the data in various electronics- based systems. Embedded systems are a combination of hardware and software where software is usually known as firmware that is embedded into the hardware. One of its most important characteristics of these systems is, it gives the o/p within the time limits. Embedded systems support to make the work more perfect and convenient. So, we frequently use embedded systems in simple and complex devices too. The applications of embedded systems mainly involve in our real life for several devices like microwave, calculators, TV remote control, home security and neighborhood traffic control systems

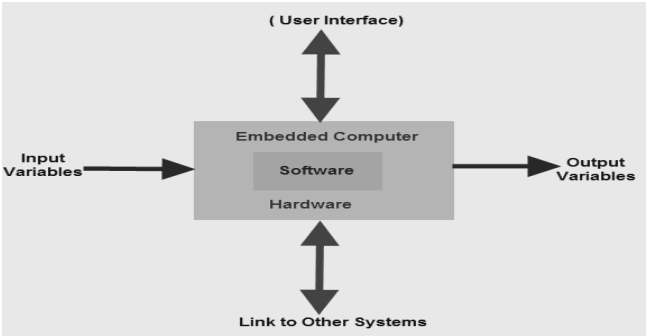


Fig: 5.1: Block Diagram of Embedded system

The embedded system basics are the combination of embedded system hardware and embedded system software.

Embedded System Hardware

An embedded system uses a hardware platform to perform the operation. Hardware of the embedded system is assembled with a microprocessor/microcontroller. It has the elements such as input/output interfaces, memory, user interface and the display unit. Generally, an embedded system comprises of the following

- Power Supply
- Memory
- Processor
- Timers
- Output/Output circuits
- Serial communication ports
- SASC (System application specific circuits)

Embedded System Software

The software of an embedded system is written to execute a particular function. It is normally written in a high-level setup and then compiled down to offer code that can be stuck within a non-volatile memory in the hardware. An embedded system software is intended to keep in view of the following three limits

- Convenience of system memory
- Convenience of processor's speed
- When the embedded system runs constantly, there is a necessity to limit power dissipation for actions like run, stop and wake up.

2EMBEDDED SYSTEM CHARACTERISTICS

- Generally, an embedded system executes a particular operation and does the similar continually. For instance: A pager is constantly functioning as a pager.
- All the computing systems have limitations on design metrics, but those can be especially tight. Design metric is a measure of an execution features like size, power, cost and also performance.
- It must perform fast enough and consume less power to increase battery life.

- Several embedded systems should constantly react to changes in the system and also calculate particular results in real time without any delay. For instance, a car cruise controller; it continuously displays and responds to speed & brake sensors. It must calculate acceleration/de-accelerations frequently in a limited time; a delayed computation can consequence in letdown to control the car.
- It must be based on a microcontroller or microprocessor based.
- It must require a memory, as its software generally inserts in ROM. It does not require any secondary memories in the PC.
- It must need connected peripherals to attach input & output devices.
- An Embedded system is inbuilt with hardware and software where the hardware is used for security and performance and Software is used for more flexibility and features.

Embedded System Applications

The applications of an embedded system basics include smart cards, computer networking, satellites, telecommunications, digital consumer electronics, missiles, etc

3 CATEGORIES OF EMBEDDED SYSTEM

Embedded systems are classified into four categories based on their performance and functional requirements:

- Stand alone embedded systems
- Real time embedded systems
- Networked embedded systems
- Mobile embedded systems

Embedded Systems are classified into three types based on the performance of the microcontroller such as

- Small scale embedded systems
- Medium scale embedded systems

- Sophisticated embedded systems

Stand Alone Embedded Systems

Stand alone embedded systems do not require a host system like a computer, it works by itself. It takes the input from the input ports either analog or digital and processes, calculates and converts the data and gives the resulting data through the connected device- Which either controls, drives and displays the connected devices. Examples for the stand alone embedded systems are mp3 players, digital cameras, video game consoles, microwave ovens and temperature measurement systems.

Real Time Embedded Systems

A real time embedded system is defined as, a system which gives a required o/p in a particular time. These types of embedded systems follow the time deadlines for completion of a task. Real time embedded systems are classified into two types such as soft and hard real time systems. Further this Real-Time Embedded System is divided into two types i.e.

Soft Real Time Embedded Systems

In these types of embedded systems time/deadline is not so strictly followed. If deadline of the task is passed (means the system didn't give result in the defined time) still result or output is accepted.

Hard Real-Time Embedded Systems

In these types of embedded systems time/deadline of task is strictly followed. Task must be completed in between time frame (defined time interval) otherwise result/output may not be accepted.

Examples

- Traffic control system
- Military usage in defense sector
- Medical usage in health sector

Networked Embedded Systems

These types of embedded systems are related to a network to access the resources. The connected network can be LAN, WAN or the internet. The connection can be any wired or wireless. This type of embedded system is the fastest growing area in embedded system applications. The embedded web server is a type of system wherein all embedded devices are connected to a web server and accessed and controlled by a web browser.Example for the LAN networked embedded system is a home security system wherein all sensors are connected and run on the protocol TCP/IP

Mobile Embedded Systems

Mobile embedded systems are used in portable embedded devices like cell phones, mobiles, digital cameras, mp3 players and personal digital assistants, etc.The basic limitation of these devices is the other resources and limitation of memory.

Small Scale Embedded Systems

These types of embedded systems are designed with a single 8 or 16-bit microcontroller, that may even be activated by a battery. For developing embedded software for small scale embedded systems, the main programming tools are an editor, assembler, cross assembler and integrated development environment (IDE).

Medium Scale Embedded Systems

These types of embedded systems design with a single or 16 or 32 bit microcontroller, RISCs or DSPs. These types of embedded systems have both hardware and software complexities. For developing embedded software for medium scale embedded systems, the main programming tools are C, C++, JAVA, Visual C++, RTOS, debugger, source code engineering tool, simulator and IDE.

Sophisticated Embedded Systems

These types of embedded systems have enormous hardware and software complexities, that may need ASIPs, IPs, PLAs, scalable or configurable processors. They are used for cutting- edge applications that need hardware and software Co-design and components which have to assemble in the final system.

Requirements of Embedded system

Reliability

Cost-effectiveness

Low power consumption Efficient

use of processing power Efficient

use of memory

Appropriate execution time

Reliability

Embedded system have to work without the need for resetting or rebooting. This call for a very reliable hardware and software. For example : if an embedded system comes to a halt because of hardware error, the system must reset itself without the need for human intervention. However the embedded software developers must make the reliability of the hardware as well as that of the software

Cost-effectiveness

If an embedded system is designed for a very special purpose such as for deep space or for nuclear power plant station cost may not be an issue. However if the embedded system is designed for a mass market purpose like CD players, toys and mobile devices cost is a major concern. Application

Specific Integrated Circuit (ASIC) is used by the designers to reduce the hardware components and hence the cost

Low power consumption

Most of the embedded systems are powered by battery, rather than a main supply. In such case the power consumption should be minimized to avoid draining the Batteries. For example : by reducing the number of hardware component the power consumption can be reduced. As well as by designing the processor to revert to low power or sleep mode when there is no operation to perform

Efficient use of processing power

A wide variety of processors with varying processing powers are available to embedded systems. Developers must keep processing power, memory and cost in mind while choosing the right processor. The processing power requirement is specified in
,Million Instruction Per Second (MIPS). With the availability of so many processor, choosing a processor has become a tough task nowadays

Efficient use of memory

Most of the embedded systems do not have secondary storage such as hard disk. The memory chip available on the embedded systems are only Read Only memory and Random Access memory. As most of the embedded systems do not have secondary storage, “flash memory” is used to store the program. Nowadays micro-controller and Digital signal processors also comes with onboard memory. Such processors are used for small embedded system as the cost generally is low and the execution generally is fast

Appropriate execution time

In real time embedded systems, certain task must be performed within a specified period of time. Normally desktop pc cannot achieve real time performance. Therefore,

special operating system known as real time operating systems run on these embedded systems. In hard real time embedded system deadlines has to be strictly met but whereas in soft real time embedded system the task may not be performed in a timely manner. The software developer needs to ascertain whether the embedded system is a hard real time or soft one and has to perform the performance analysis accordingly

CHALLENGES AND ISSUES

Co-design

Embedding an operating system

Code optimization

Efficient input or output

Testing and debugging

Co-design

An embedded system consists of hardware and software, deciding which function of the system should be implemented in hardware and software is of a major consideration. For example in hardware implementation the task execution is faster compared with the other one. On the downside a chip cost money, consumes valuable power and occupies space. A software implementation is better if these are the major concern. This issue of choosing between hardware and software implementation is known as a co-design issue

Embedding an operating system

It is possible to write embedded software without any operating system embedded into the system. Developers can implement services such as memory management, input/output management and so on. Writing your own routines necessary for a particular application results in compact and efficient coding. Embedded operating system provide the necessary Application Programming Interfaces(API).

code optimization

Developers need not worry much about the code optimization, because the processor is highly powerful, plenty of memory is generally available. Memory and Execution time are the important constraints in embedded system. Sometimes to achieve the required response time the programmer has to write certain portion of coding in assembly language. Of course, with the availability of sophisticated development tools, this is less of an issue in recent years

Efficient input or output

In most of the embedded system, the input interfaces have limited functionality. Writing embedded software is a different ball game compared with writing a user interface with a full-fledged keyboard, a mouse and a large display. Many systems available in process control take electrical signal as input and produce electrical signal as output, since they don't use I/O devices. Developing, testing and debugging such systems is much more challenging than doing the same with the desktop systems.

4.TESTING AND DEBUGGING

Software for an embedded system cannot be tested on the target hardware during the development phase because debugging will be extremely difficult. Testing and debugging the software on the host system by actual simulation of field conditions is very challenging. Nowadays, the job is made a bits simpler with the availability of "profilers" that tell you clearly which line of code are executed and which lines are not executed. Using the output of such profilers we can locate the untested lines of code and ensure that they are also executed by providing the necessary test input data. It is these challenges that made embedded software development a "black art" in earlier days. This is no longer the case, however the developments in embedded software are changing the scenario completely.

Recent trends in embedded system

Processors

Memory

Operating Systems

Programming Languages

Development Tools

Processors

In an effort to cater to different applications, several semiconductor electronics vendors have released many processors. We can find 8-bit, 16-bit, and 32 bit processors with different processing powers and memory addressing capabilities. Many sophisticated DSP are available to cater to numerous application needs including audio and video coding and image processing. The processor boards around which the embedded systems can be built come with the necessary RAM and ROM as well as peripherals such as a serial port, USB port and Ethernet connectivity.

Memory

Both RAM and ROM memory devices are becoming increasingly cheaper paving the way for devices that can store large numbers of programs and their data. Secondary devices such as Hard disk are also being integrated into embedded systems such as mobile communication and computing devices . Devices that do not have secondary storage use flash memory and the capacity of flash memory chips is also rising very rapidly making it possible to incorporate heavy OS

Operating Systems

As most everyone knows Microsoft currently holds the lion share of the market in operating systems that run desktop computers. Many operating systems which are available nower days are categorized as embedded operating systems, real time operating systems and mobile operating system. These operating system occupies much less memory. This reduces the development time and the effort considerably

Programming Languages

The era of writing the embedded software in assembly languages is now almost history. High level languages are extensively used for embedded software development. Object oriented programming languages are also extensively used. Another important development is the use of JAVA. Because of JAVA platform independence it has become very popular for embedded

software development

Development Tools

Many advances in development tools are accelerating embedded software development. These development tools include Cross compiler, Debbugers and Emulators. Using these tools developers can write programs on host machines, test the software thoroughly and port to the target hardware. The cycle time for the development has been reduced considerably in recent years because of these development tools. Many of them are available free of cost from major software vendors

6. CONTROL SYSTEM AND INDUSTRIAL AUTOMATION

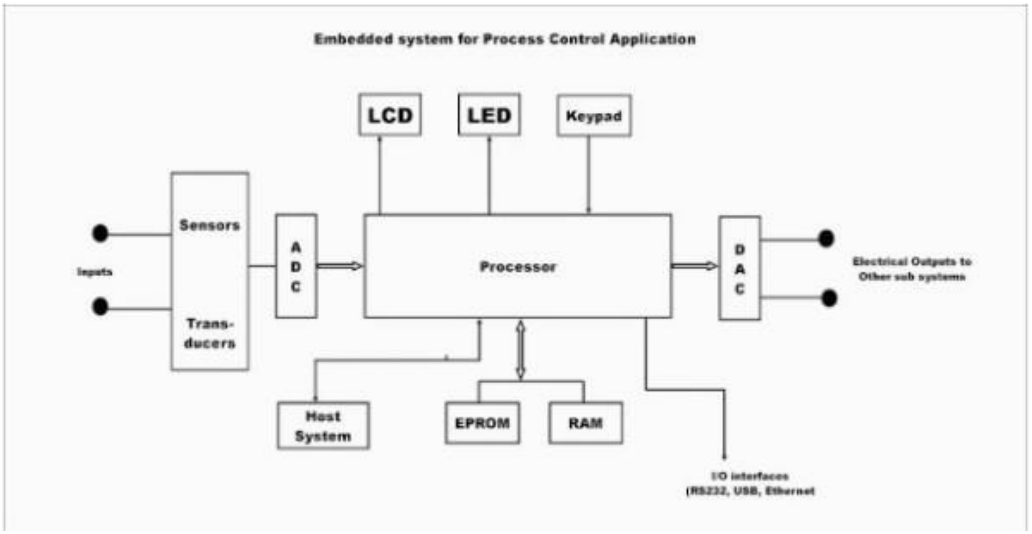


Fig.2: Block Diagram of control system and industrial automation

The embedded system takes electrical signals as input. Generally sensors or transducers are used to convert the physical entity into an electrical signal. The processor can process only digital signals, the ADC(Analog to Digital Convertor) converts the analog signals to its equivalent digital signals, which is an electrical representation of a bit stream of 0s and 1s. RAM is used to store the volatile data. A DAC(Digital to Analog Convertor) is used to convert the output digital signal to analog format. The processor board also includes input/output interface, such as serial interface , USB port and an Ethernet port for connectivity to the external systems. For the user interaction

LCD and LED and a keypad are provided. These modules may or may not be required depending on the application. Depending on the application the designer chooses the necessary modules and carries out the design. While designing the reliability, performance and the cost need to be kept in the mind. Some of the typical process control applications in nuclear plants and telemetry and tele command units in satellite communication systems. Some of the embedded systems have to operate in very hostile environments

7. BIOMEDICAL SYSTEMS

Much of the progress made in the health care industry is due to the development in the electronic industry Hospitals are full of embedded systems, including X-ray control units, EEG and ECG units and equipments used for diagnostic testing such as endoscopy and so on. These systems use PC add on cards which take the ECG signals and process them and the PC monitor is used for the display. Even the PC secondary storage is used to store the ECG records. Biometric systems for finger print and face recognition are gaining wide
use in the agencies concerned with the securities

The input fingerprint must be processed and compared with the available database using pattern recognition algorithm, which requires intensive processing. The biometric systems use a Digital Signal Processor(DSP) for signal processing such as filtering and edge enhancement of the image. And a general purpose processor for implementing the pattern matching algorithms

13.DATA COMMUNICATION SYSTEM

Internet has acted as the catalyst for the embedded system. Modem that connects two computers is an embedded system. Dialup modems normally used to access the internet are embedded systems with a DSP inside. Using the DSP and the associated software the modem establishes the connection using the standard protocols. As the digital signal is modulated a lot of signal processing is involved therefore, DSP is used.

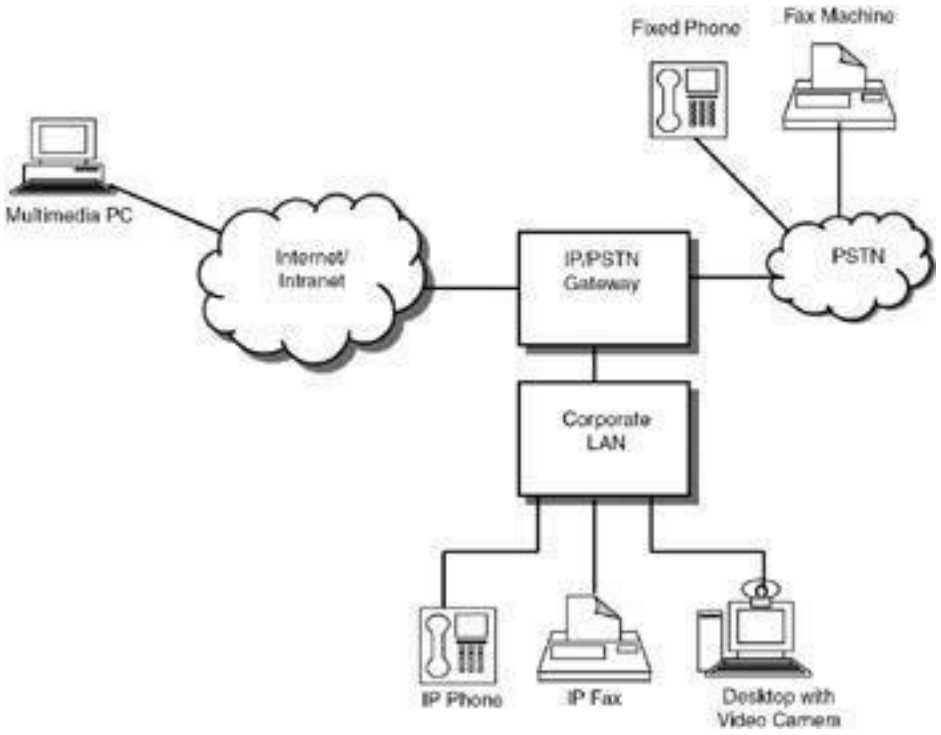


Fig. 5.3:Multimedia Communication over IP Network

"Convergence" is the mantra nowadays. For years we used different networks for different services. Telephone network(PSTN) for making voice call and sending fax messages. Internet for data rate services such as email, file transfer and web services. WAN act as the backbone network supporting the data, voice and video communication services

Telecommunication

Telecommunication infrastructure element includes networking components such as Telephone switches, Loop carriers, terminal adapters, ATM switches, frame relays and so on. Mobile communication components includes base station, Mobile switching centers and so on. Satellite communication equipment includes earth station controller, onboard processing elements, telemetry and so on

Audio codec

Normally when voice is transmitted over the telephone network the voice is coded at the rate of 64kbps using a technique known as PCM. In radio system speech is compressed to save the bandwidth. At the transmitting side the audio signal are compressed to achieve data rates and at the receiving end the audio signal is expanded to retrieve the original Signal. These codecs use DSP extensively and gets embedded into cell phones and equipment of mobile and fixed communication systems. MP3 player is a good example, where are the signals are encoded and transmitted from a music kiosk to be played on the MP3 device

Automatic speech generation system

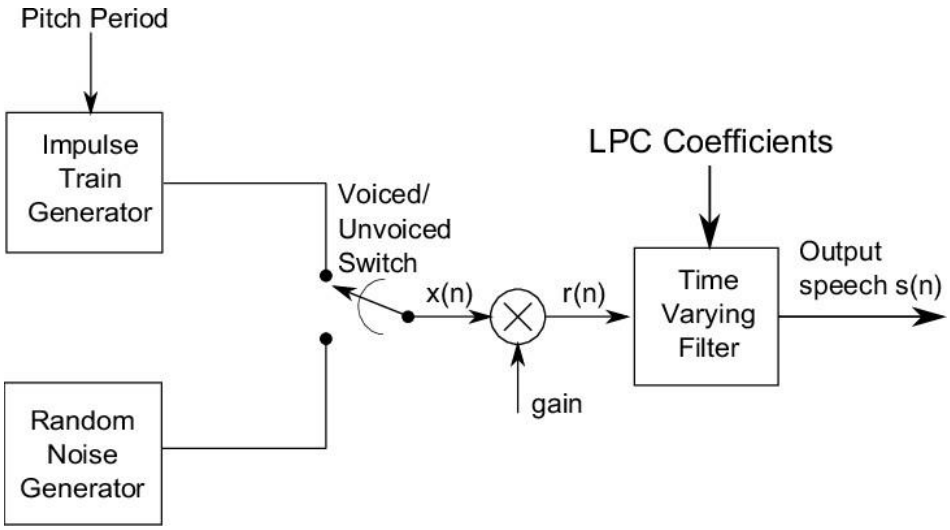


Fig.5.4: Human speech model

Video codec

Video conferencing has become very popular in recent years. Video occupies very large bandwidth however and to transmit video over the internet, video signals must be compressed to reduce the data Standards such as MPEG and JPEG are used to achieve

video compression. To compress the video signal a video coder is used and to bring to the original signal a decoder is used. These embedded systems use DSP to implement video compression Algorithm

8. IVR SYSTEM

It is a stand alone embedded system connected to the computer through a parallel port or USB port or it can be implemented on a PC with an add on card. IVR system is an embedded system connected to the computer holding the bank database. IVR system also has a telephone interface and it is connected in parallel to a telephone line. Once the bank assigns a specific number to the IVR system any subscriber can call this number to get the information about his/her bank account details. IVR system comprises of PSTN interface, ADC and DAC, S to P and P to S Convertors and an interface circuitry with microphone and speaker PSTN interface receives the telephone calls and answers them. Filters limit the audio signal to the desired frequency band up to 4khz. ADC converts input to digital format and digitized voice data is converted to parallel format using S to P convertor and vice versa. FIFO are buffers that temporarily holds the speech data. An IC MT8880 is used. Using this technology coupled with speech recognition and speech synthesis we can develop applications to browse the web through voice commands

15. GPS SYSTEM

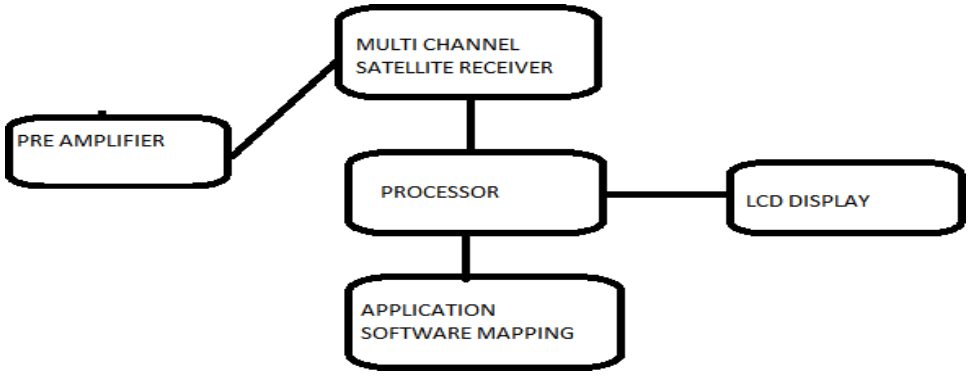


Fig.5.5: Block Diagram of GPS Receiver

It is a gift from the U.S from DOD to the humankind. Using a set of 24 NAVSTAR satellites, the DOD provides the GPS service for any moving or fixed object. A GPS receiver receives the satellite signals and process them to find the position parameters of the GPS receiver location. GPS receiver is a powerful embedded system that uses a DSP to process the satellite signals. GPS receiver computes its latitude, longitude, altitude, velocity and so on. It has an RS 232 serial communication interface or a USB interface from which the position data is available.

Question Bank

Part-A

1. What is the technique used for power&energy management in a system.
2. What do you mean by real time Operating system.
3. What is the role of RAM in an Embedded System. 6.When do we need multitasking Operating system.
4. When do we need an RTOS?
5. What are the challenges faced in designing an Embedded System.
6. What are the requirements of Embedded System.
7. List the issues in Embedded software development.

PART B

1. Explain the different categories of Embedded System.
2. Explain about the Networked Embedded System with a neat block diagram.
3. Discuss about the issues in Embedded software development.
4. Explain the requirements of Embedded System in detail.
5. Give an example of real time embedded system.
6. Give the basic steps in developing an embedded system.
7. Explain about in trend in embedded system
8. Explain about stand alone system with three example.

TEXT / REFERENCE BOOKS

1. Ramesh Goankar, "Microprocessor architecture programming and applications with 8085 / 8088", 5th Edition, Penram International Publishing, 2002.

2. Mazidi & McKinlay, “The 8051 Microcontroller and Embedded Systems using Assembly and C”, PHI, 2007.

3. MykePredko, “Programming and Customizing the 8051 Micro-controller”, Tata McGraw-Hill edition, 2007.

4. R A Gaonkar, “Fundamentals of Microcontrollers and Applications in Embedded Systems (with the PIC18 Microcontroller Family)”, Penram Publishing India, 2007

. 5. Kenneth Ayala ,”The 8051 Microcontroller”, 3rd Edition, Thomson Delmar Learning, 2004.

6. Kenneth J. Ayala, Dhananjay V. Gadre, “The 8051 Microcontroller & Embedded Systems Using Assembly and C”, Cengage Learning India Publication, 2007.

7. Ajay V Deshmukh, “Microcontrollers: Theory and Applications”, Tata McGraw-Hill, 2005

. 8. Raj Kamal, “Embedded Systems Architecture, Programming, and Design”. (2/e), Tata McGraw Hill, 2008.