E-content
of
MICROCONTROLLERS

1. Microcontroller series (MCS) – 51 Overview (16 Periods)
   1.1. Architecture of 8051Microcontroller
   1.2. Pin details
   1.3. I/O Port structure
   1.4. Memory Organization
   1.5. Special Function Registers (SFRs)
   1.6. External Memory
2. Instruction Set (20 Periods)
   2.1. Instruction Set of 8051
   2.2. Addressing Modes,
   2.3. Types of Instructions
   2.4. Timer operation
   2.5. Serial Port operation
   2.6. Interrupts
3. Assembly/C programming for Micro controller (12 Periods)
   3.1. Assembler directives
   3.2. Assembler operation
   3.3. Compiler operations
   3.4. De bugger
4. Design and Interface (12 Periods)
   4.1. Keypad interface
   4.2. 7- segment interface
   4.3. LCD, A/D, D/A and RTC interface with programming.
5. Introduction of PIC Micro controllers (04 Periods)

MICROCONTROLLERS

LEARNING OBJECTIVES:

- Concept of 8051microcontroller.
- Study of pin diagram.
- About I/O port and memory organization.
- To know about SFRs and external memory.

CHAPTER-1(Microcontroller Series)

1.1. INTRODUCTION

What is a Microcontroller?

A Microcontroller is a VLSI IC that contains a CPU (Processor) along with some other peripherals like Memory (RAM and ROM), I/O Ports, Timers/Counters, Communication Interface, ADC, etc.

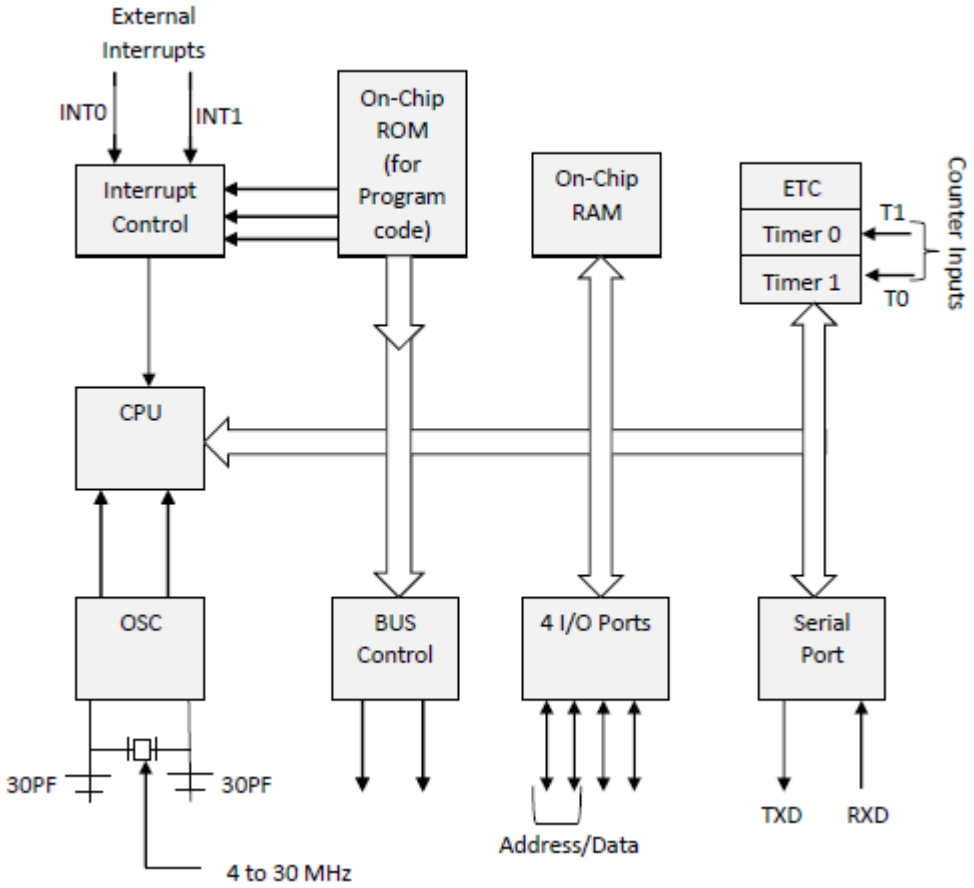
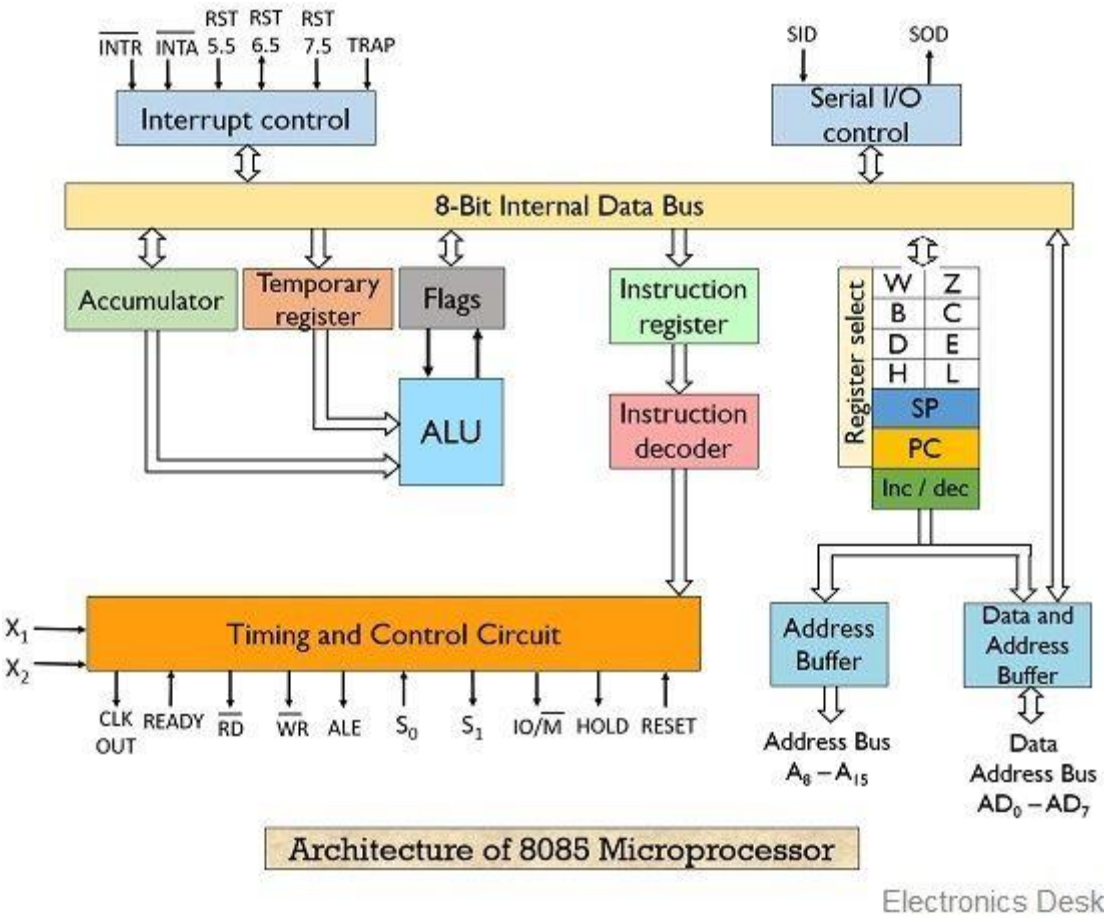


Diagram of Microcontroller

On the contrary, a Microprocessor (which was developed before Microcontroller) is just a Processor (CPU) and doesn't have the above-mentioned peripherals. In order to make it work or build a system around it, we need to interface the peripherals separately.



Until the development of Microcontrollers, almost all process and control tasks were implemented using Microprocessors. As Microprocessor need the additional peripherals to work as a system, the overall cost of the control system was high. But with the development of Microcontroller, the situation has changed completely including the world of Embedded Systems.

1.2. 8051 MICROCONTROLLER INTRODUCTION AND HISTORY

The 8051 Microcontroller Introduction gives a brief overview about the 8051 Microcontroller and its history. Intel's 8051 Microcontroller (Intel MSC-51 Architecture) was a successor to 8048 Microcontroller (Intel MSC-48 Architecture).

Originally, 8051 Microcontrollers were developed using N-MOS Technology but the use of battery powered devices and their low power consumption lead to usage of CMOS technology (which is famous for its low power consumption). Even though Intel developed 8051 Microcontrollers (which is discontinued in 2007), more than 20 semiconductor manufacturers are still producing 8051 compatible microcontrollers i.e., processors based on MSC-51 Architecture.
Some of the 8051 Microcontrollers produced by different manufacturers are: Atmel (AT89C51, AT89S51), Phillips (S87C654), STC Micro (STC89C52), Infineon (SAB-C515, XC800), Siemens (SAB-C501), Silicon Labs (C8051), NXP (NXP700, NXP900), etc.

Majority of the modern 8051 Microcontrollers are Silicon IP Cores (Intellectual Property Cores) but discrete 8051 Microcontroller ICs are also available. Because of their low power consumption, smaller size and simple architecture, 8051 IP Cores are used in FPGAs (Field

Programmable Gate Array) and SoCs (System on Chip) instead of Advanced ARM Architecture based MCUs.

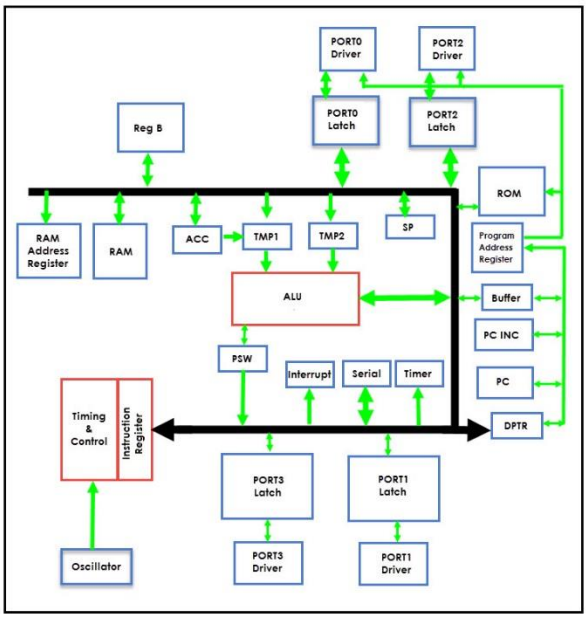
1.3. APPLICATION OF 8051 MICROCONTROLLER

Even with the development of many advanced and superior Microcontrollers, 8051 Microcontroller is still being used in many embedded system and applications.
Some of the applications of 8051 Microcontroller are mentioned below:

- Consumer Appliances (TV Tuners, Remote controls, Computers, Sewing Machines, etc.)
- Home Applications (TVs, VCR, Video Games, Camcorder, Music Instruments, Home
- Security Systems, Garage Door Openers, etc.)
- Communication Systems (Mobile Phones, Intercoms, Answering Machines, Paging Devices, etc.)
- Office (Fax Machines, Printers, Copiers, Laser Printers, etc.)
- Automobiles (Air Bags, ABS, Engine Control, Transmission Control, Temperature Control, Keyless Entry, etc)
- Aeronautical and Space
- Medical Equipment
- Defence Systems
- Robotics
- Industrial Process and Flow Control
- Radio and Networking Equipment
- Remote Sensing

1.4. 8051 MICROCONTROLLER BASICS

8051 is an 8 – bit Microcontroller i.e., the data bus of the 8051 Microcontroller (both internal and external) is 8 – bit wide. It is a CISC based Microcontroller with Harvard Architecture (Separate program and data memory). Since the basic layout of a microcontroller includes a CPU, ROM, RAM, etc. the 8051 microcontrollers also have a similar layout. The following image shows a brief layout of a typical 8051 Microcontroller.



1.5. 8051 MICROCONTROLLER FEATURES

- **8 – Bit ALU:** ALU or Arithmetic Logic Unit is the heart of a microcontroller. It performs arithmetic and bitwise operation on binary numbers. The ALU in 8051 is an 8 – Bit ALU i.e., it can perform operations on 8 – bit data.
- **8 – Bit Accumulator:**The Accumulator is an important register associated with the ALU. The accumulator in 8051 is an 8 – bit register.
- **RAM:** 8051 Microcontroller has 128 Bytes of RAM which includes SFRs and Input / Output Port Registers.
- **ROM:** 8051 has 4 KB of on-chip ROM (Program Memory).
- **I/O Ports:** 8051 has four 8 – bit Input / Output Ports which are bit addressable and bidirectional.
- **Timers / Counters:** 8051 has two 16 – bit Timers / Counters.
- **Serial Port:** 8051 supports full duplex UART Communication.
- **External Memory:** 8051Microcontroller can access two 16 – bit address line at once: one each for RAM and ROM. The total external memory that an 8051 Microcontroller can access for RAM and ROM is 64KB (216 for each type).
- **Additional Features:** Interrupts, on-chip oscillator, Boolean Processor, Power Down Mode, etc.

NOTE: Some of the features like size of RAM and ROM, number of Timers, etc. are not generic. They vary by manufacturer.

1.6. INTERRUPTS:

As the heading put forward, Interrupt is a sub-routine call that reads the Microcontroller's key function or job and helps it to perform some other program which is extra important at that point of time. The characteristic of 8051 Interrupt is extremely constructive as it aids in emergency cases. Interrupts provides us a method to postpone or delay the current process, carry out a sub-routine task and then all over again restart standard program implementation.

The Micro-controller 8051 can be assembled in such a manner that it momentarily stops or break the core program at the happening of interrupt. When sub-routine task is finished then the implementation of core program initiates automatically as usual. There are 5 interrupts supplies in 8051 Microcontroller, two out of five are peripheral interrupts, two are timer interrupts and one is serial port interrupt.

1.7. MEMORY:

Micro-controller needs a program which is a set of commands. This program enlightens Microcontroller to perform precise tasks. These programs need a storage space on which they can be accumulated and interpret by Microcontroller to act upon any specific process. The memory which is brought into play to accumulate the program of Microcontroller is recognized as Program memory or code memory. In common language it's also known as Read Only Memory or ROM.

Microcontroller also needs a memory to amass data or operands for the short term. The storage space which is employed to momentarily data storage for functioning is acknowledged as Data Memory and we employ Random Access Memory or RAM for this principle reason.

Microcontroller 8051 contains code memory or program memory 4K so that is has 4KB Rom and it also comprise of data memory (RAM) of 128 bytes.

1.7.1. BUS:

Fundamentally Bus is a group of wires which functions as a communication canal or mean for the transfer Data. These buses comprise of 8, 16 or more cables. As a result, a bus can bear 8 bits, 16 bits all together. There are two types of buses:

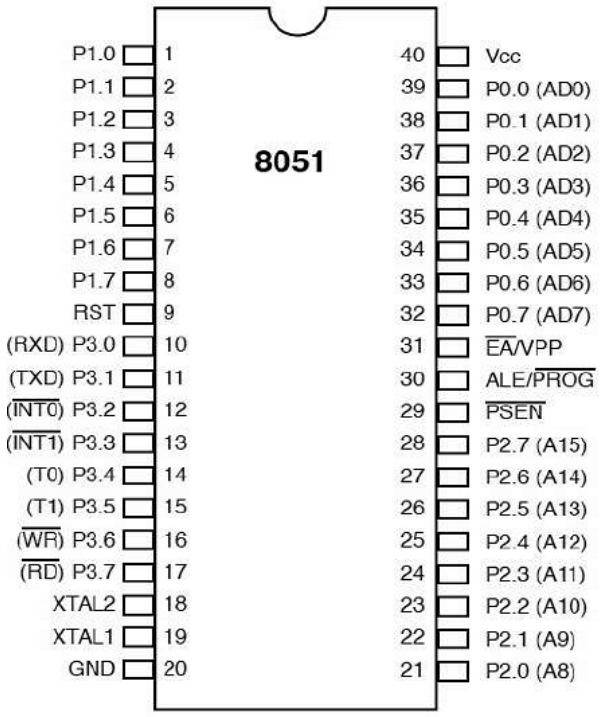
1. **Address Bus:** Microcontroller 8051 consists of 16 bit address bus. It is brought into play to address memory positions. It is also utilized to transmit the address from Central Processing Unit to Memory.
2. **Data Bus:** Microcontroller 8051 comprise of 8 bits data bus. It is employed to cart data.

1.7.2. OSCILLATOR:

As we all make out that Microcontroller is a digital circuit piece of equipment, thus it needs timer for its function. For this function, Microcontroller 8051 consists of an on-chip oscillator which toils as a time source for CPU (Central Processing Unit). As the productivity thumps of oscillator are steady as a result, it facilitates harmonized employment of all pieces of 8051 Microcontroller. Input/output Port: As we are acquainted with that Microcontroller is employed in embedded systems to manage the functions of devices.

Thus, to gather it to other machinery, gadgets or peripherals we need I/O (input/output) interfacing ports in Micro-controller. For this function Micro-controller 8051 consists of 4 input/output ports to unite it to other peripherals. Timers/Counters: Micro-controller 8051 is incorporated with two 16-bit counters & timers. The counters are separated into 8-bit registers. The timers are utilized for measuring the intervals, to find out pulse width etc.

1.8. 8051 MICROCONTROLLER PIN DIAGRAM



8051 Microcontroller Pin Diagram

For explaining the pin diagram and pin configuration of microcontroller 8051, we are taking into deliberation a 40 pin Dual inline package (DIP). Now let's study through pin configuration in brief: -

- **Pins 1 – 8: -** recognized as Port 1. Different from other ports, this port doesn't provide any other purpose. Port 1 is a domestically pulled up, quasi bi directional Input/output port.
- **Pin 9: -** As made clear previously RESET pin is utilized to set the micro-controller 8051 to its primary values, whereas the micro-controller is functioning or at the early beginning of application. The RESET pin has to be set elevated for two machine rotations.
- **Pins 10 – 17: -** recognized as Port 3. This port also supplies a number of other functions such as timer input, interrupts, serial communication indicators TxD & RxD, control indicators for outside memory interfacing WR & RD, etc. This is a domestic pull up port with quasi bi directional port within.
- **Pins 18 and 19: -** These are employed for interfacing an outer crystal to give system clock.
- **Pin 20: -** Titled as Vss – it symbolizes ground (0 V) association.
- **Pins- 21-28: -** recognized as Port 2 (P 2.0 – P 2.7) – other than serving as Input/output port, senior order address bus indicators are multiplexed with this quasi bi directional port.
- **Pin- 29: -** Program Store Enable or PSEN is employed to interpret sign from outer program memory.
- **Pin-30: -** External Access or EA input is employed to permit or prohibit outer memory interfacing. If there is no outer memory need, this pin is dragged high by linking it to Vcc.
- **Pin-31: -** Aka Address Latch Enable or ALE is brought into play to de-multiplex the address data indication of port 0 (for outer memory interfacing). Two ALE throbs are obtainable for every machine rotation.
- **Pins 32-39:** recognized as Port 0 (P0.0 to P0.7) – other than serving as Input/output port, low order data & address bus signals are multiplexed with this port (to provide the use of outer memory interfacing). This pin is a bi directional Input/output port (the single one in microcontroller 8051) and outer pull up resistors are necessary to utilize this port as Input/output.
- **Pin-40:** termed as Vcc is the chief power supply. By and large it is +5V DC.

1.9. APPLICATION OF 8051 MICROCONTROLLER:

The microcontroller 8051 applications include large number of machines, principally because it is simple to incorporate in a project or to assemble a machine around it. The following are the key spots of spotlight:

1. **Energy Management:** Competent measuring device systems aid in calculating energy consumption in domestic and industrialized applications. These meter systems are prepared competent by integrating microcontrollers.
2. **Touch screens:** A high degree of microcontroller suppliers integrate touch sensing abilities in their designs. Transportable devices such as media players, gaming devices & cell phones are some illustrations of micro-controller integrated with touch sensing screens.
3. **Automobiles:** The microcontroller 8051 discovers broad recognition in supplying automobile solutions. They are extensively utilized in hybrid motor vehicles to control

engine variations. In addition, works such as cruise power and anti-brake mechanism has created it more capable with the amalgamation of micro-controllers.

4. **Medical Devices:** Handy medicinal gadgets such as glucose & blood pressure monitors bring into play micro-controllers, to put on view the measurements, as a result, offering higher dependability in giving correct medical results.
5. **Medical Devices:** Handy medicinal gadgets such as glucose & blood pressure monitors bring into play micro-controllers, to put on view the measurements, as a result, offering higher dependability in giving correct medical results.

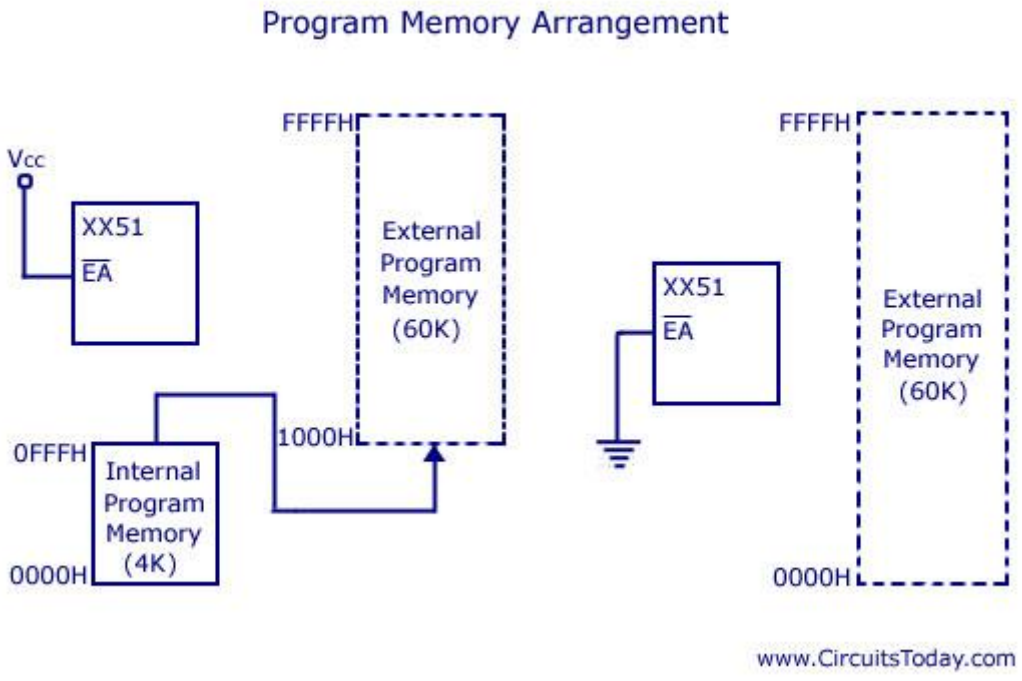
1.10. 8051 MEMORY ORGANIZATION

Before going deep into the memory architecture of 8051, let's talk a little bit about two variations available for the same. They are Princeton architecture and Harvard architecture. Princeton architecture treats address memory and data memory as a single unit (does not distinguish between two) whereas Harvard architecture treats program memory and data memory as Â separate Â entities. Thus, Harvard architecture demands address, data and control bus for accessing them Â separately Â whereas Princeton architecture does not demand any such separate bus.

Example: - 8051 micro controller is based on Harvard architecture and 8085 micro processor is based on Princeton architecture.

Thus 8051 has two memories: - Program memory and Data memory

1.10.1. PROGRAM MEMORY ORGANIZATION

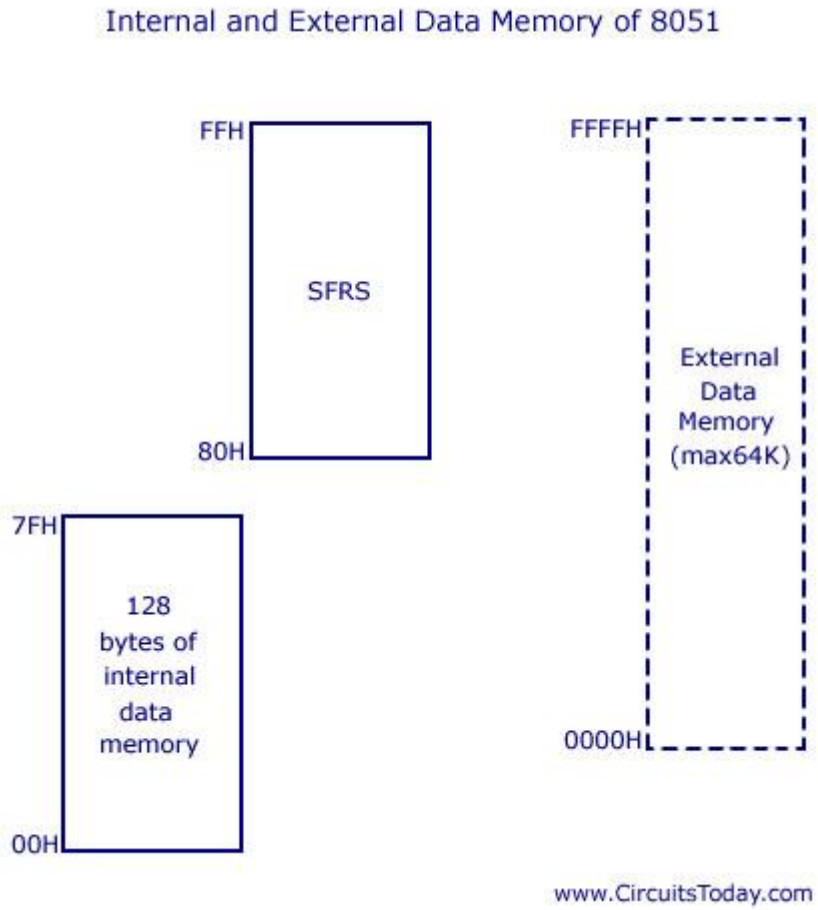


Now let's dive into the program memory organization of 8051. It has an internal program of 4K size and if needed an external memory can be added (by interfacing) of size 60K maximum. So, in total 64K size memory is available for 8051Â micro controller. Â by default, the External Access (EA) pin should be connected Vcc so that instructions are fetched from internal memory initially. When the limit of internal memory (4K) is crossed, control will automatically move

to external memory to fetch remaining instructions. If the programmer wants to fetch instruction from external memory only (bypassing the internal memory), then he must connect External Access (EA) pin to ground (GND). You may already know that 8051 has a special feature of locking the program memory (internal) and hence protecting against software piracy. This feature is enable by program lock bits. Once these bits are programmed, contents of internal memory cannot be accessed using an external circuitry. However, locking the software is not possible if external memory is also used to store the software code. Only internal memory can be locked and protected. Â Once locked, these bits can be unlocked only by a memory-erase operation, which in turn will erase the programs in internal memory too.

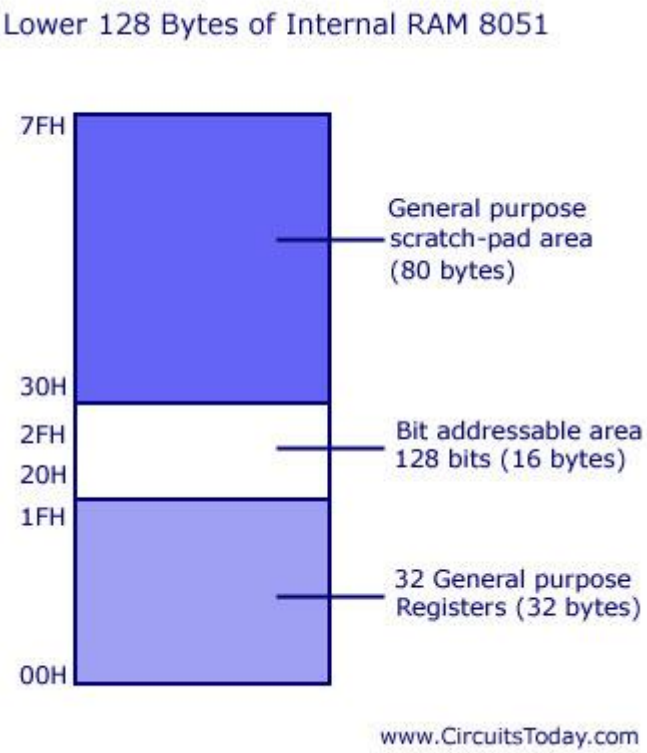
8051 is capable of Â pipelining. Pipelining makes a processor capable of fetching the next instruction while executing previous instruction. It's something like multi-tasking, doing more than one operation at a time. 8051 is capable of fetching first byte of the next instruction while executing the previous instruction.

1.10.2. DATA MEMORY ORGANIZATION



In the MCS-51 family, 8051 has 128 bytes of internal data memory and it allows interfacing external data memory of maximum size up to 64K. So, the total size of data memory in 8051 can be up to 64K (external) Â + Â 128 bytes (internal). Â Observe the diagram carefully to get more understanding. So, there are 3 separations/divisions of the data memory: -

1) Register banks
2) Bit Â addressable Â area
3) Scratch pad area.



Register banks form the lowest 32 bytes on internal memory and there are 4 register banks designated bank #0, #1, #2 and #3. Each bank has 8 registers which are designated as R0, R1…R7. At a time only one register bank is selected for operations and the registers inside the selected bank are accessed using mnemonics R0... R1… etc. Other registers can be accessed simultaneously only by direct addressing. Â Registers are used to store data or operands during executions. Â By default register bank #0 is selected (after a system reset).

The bit addressable areas of 8051 is usually used to store bit variables. The bit addressable area is formed by the 16 bytes next to register banks. They are designated from address 20H to 2FH (total 128 bits). Each bits can be accessed from 00H to 7FH within this 128 bits from 20H to 2FH. Bit addressable area is mainly used to store bit variables from application program, like status of an output device like LED or Motor (ON/OFF) etc. We need only a bit to store this status and using a complete byte addressable area for storing this is really bad programming practice, since it results in wastage of memory.

The scratch pad area is the upper 80 bytes which is used for general purpose storage. Scratch pad area is from 30H to 7FH Â and this includes stack too.

1.11. SPECIAL FUNCTION REGISTERS (SFR)

Special function registers are upper RAM memory in the 8051 microcontrollers. These registers contain all peripheral related registers like P0, P1, P2, P3, timers or counters, serial port and interrupts-related registers. The SFR memory address starts from 80h to FFh. The SFR register is implemented by bit-address registers and byte-address registers.

80	P0	SP	DPL	DPH				PCON	87
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
90	P1								97
98	SCON	SBUF							9F
A0	P2								A7
A8	IE								AF
B0	P3								B7
B8	IP								B9
C0									C7
C8									CF
D0	PSW								D7
D8									DF
E0	ACC								E7
E8									EF
F0	B								F7
F8									FF

Blue background are I/O port SFRs
Yellow background are control SFRs
Green background are other SFRs

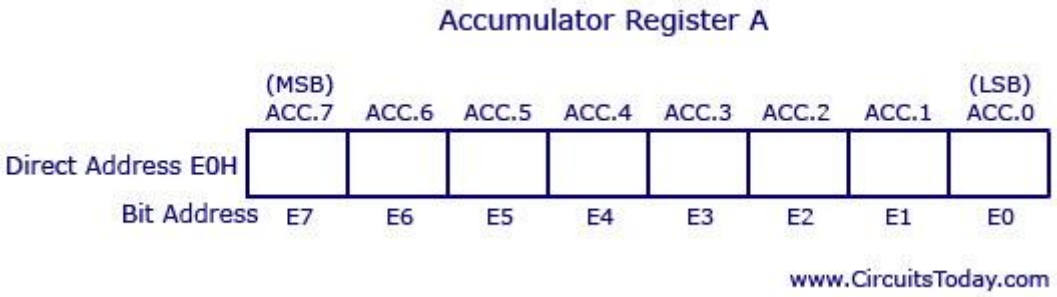
Special Function Registers (SFR)

The accumulator, B register, P0, P1, P2, P3, IE registers are bit-addressable register remaining all are byte-addressable registers.

Accumulator

The accumulator which is also known as ACC or A is a bit as well as a byte-addressable register by an address of the accumulator. If you want to use a bit-addressable register, you can use a single bit (E0) of the register and you can use an 8-bit of the accumulator as a byte-addressable register. The accumulator holds the results of most Arithmetic and logical operations.

Accumulator register



The Assembly program for subtraction used with an Accumulator

```
Org                                     0000h
MOV             R0,             #       09h
MOV     A,      #03h            (1byte  data)
SUBB    A,      01h             (1byte data)
END
```

B-Register

The B-register is a bit and byte-addressable register. You can access 1-bit or all 8-bits by a physical
address F0h. Suppose to access a bit 1, we have to use f1. The B register is only used for multiplication and division operations.

B-Register

The Assembly program for multiplication used with a B-Register

```
Org 0000h
MOV A, #09h
MOV B, #03h
MUL A, B (Final value stored in A)
END
```

The Assembly program for Division used with a B-Register

```
Org 0000h
MOV A, #09h
MOV B, #03h
DIC A, B (Final value stored in A)
END
```

Port Registers

The 8051 microcontroller consists of 4-input and output ports (P0, P1, P2, and P3) or 32-I/O pins.

Each pin is designed with a transistor and P registers. The pin configuration is very important for a
microcontroller that depends on the logic states of the registers. The pin configuration as input given by 1 or output 0 depends on the logic states. If logic 1 is applied to the bit of the P register, the output transistor switches off the appropriate pin that acts as an input pin.

Port Registers of 8051

Assembly program to toggle LEDs of Port0

```
ORG 0000h
RETURN: MOV P0, #00h
```



```
ACALL DEL1
MOV P0, #0FF
ACALL DEL1
SJMP RETURN
DEL1: MOV R2, #200
DEL: DJNZ R0, #230
DJNZ R2, DEL
RET
END
```

Counters and registers

Many microcontrollers consist of one or more timers and counters. The timers are used to generate
precious time delay and the source for the timers is crystal oscillator. The counters are used to count the number of external events – for instance, the objective counter , and the source for counters are external pulses applied across the counter pin.

The 8051 microcontroller consists of two 16-bit timers and counters such as timer 0 and timer 1.

Both the timers consist of a 16-bit register in which the lower byte is stored in the TL and the

higher byte is stored in the TH. The Timer can be used as a counter as well as for timing operation
that depends on the source of the clock pulses to the counters.
The Counters and Timers in 8051 microcontrollers contain two special function registers: TMOD
(Timer Mode Register) and TCON (Timer Control Register), which are used for activating and configuring timers and counters.
Types of Shift Register
Shift registers are a type of sequential logic circuits that are mainly used for storage of digital data.
The shift registers are bit-addressable registers that store only one bit of data. The shift registers are
constructed with flip-flops – a group of flip-flops connected as a chain so that the output from one
flip-flop becomes the input of the next flip-flop.
All the flip-flops are driven by the clock signals that are implemented by the D-flip-flap. The shift
registers are mainly used for serial communication.
These are classified into 4- types:
- Serial in Serial out (SISO)
- Serial in Parallel out (SIPO)
- Parallel in Serial out (PISO)
- Parallel in Parallel out (PIPO)

D- flipflop register

These are all different types of registers in an 8051 microcontroller. We hope that we have successfully given you relevant content with the appropriate program for each register. Furthermore, for any sort of help to know the coding of several other registers, you can contact us by commenting below.

QUESTIONS:
MULTIPLE CHOICE QUESTIONS:
1)which of following is pin of 8051:
(a) XTAL 1 (b) RST 5.5
(c) RST 6.5 (d)RST 7.5
2) which of following is a part microprocessor:
(a)ALU (b) register
(c) interrupt (d) all of these above

SHORT ANSWER TYPES QUESTIONS:
1) What is microprocessor?
2) ALU stands for ……..
3) What do you mean by bus?
4) What is function of ALE?
5) Write all types of memory.
LONG ANSWER TYPES QUESTIONS:
1) Explain in detail pin diagram of 8051microprocessor.
2) Explain in detail applications of 8051 microprocessor.

MICROCONTROLLERS

LEARNING OBJECTIVES:

 Concept of keypad interface.
 Study of 7 segment interface.
 To know about LCD,A/D,D/A.

CHAPTER-4(DESIGN AND INTERFACE)

4.1 KEYPAD INTERFACING 8051 MICROCONTROLLER : In this 8051 microcontroller tutorial you will learn how to interface 8051 microcontroller with keypad. In this article, I will guide you step by step programming part as well as structure of keypad. you can interface any size
keypad with 8051 microcontroller. Before starting this article, you should know how to write your
first program in keil and how to used input output ports of 8051 microcontroller. Because keypad is used as a input with 8051 microcontroller. Matrix Keypads are mostly used in calculators, mobile phones, telephones, ATM etc. It is used when a number of input switches are
required. In this article we will study how to interface keypad with 8051 microcontroller. An experiment will show the keypad interfacing. User will give input through keypad and then that
input will be displayed on LCD.

4.1.1 KEYPAD STRUCTURE:

In a keypad, push button switches are arranged in rows and columns. For a 4×4 keypad 16 switches
are used and to connect to microcontroller we need 16 inputs pins. But the arrangement is changed
by connecting switches in a special way. Now we need only 8 pins of microcontroller to connect
keypad to
it.

The status of each key/switch is determined by Scanning the rows or columns. The column pins
(Col 1–Col4) are connected to the microcontroller as the inputs pinsand the rows pins (Row 1–Row
4) are connected to the output pins of the microcontroller. Normally, all the column pins are pulled
high by internal or external pull up resistors. Now we can read the status of each switch through scanning.

4.1.2 READING DATA:

Scanning is done in a different way. Columns pins are used as input pins, and rows pins are used as
output. If a low logic is given to all the Rows and high logic is given to each Column.

For finding Column number:

 When a switch/key is pressed, the corresponding row and column will get short.
 Output of the corresponding column goes to go low.

 Since we have made all the rows zero so this gives the column number of the pressed key.

For Finding Row number:

 After the detection of column number, the controller set's all the rows to high.
 Each row is one by one set to zero by the microcontroller and the earlier detected column is checked and obviously it becomes zero.

 The row due to which the column gets zero is the row number of the pressed key.

4.2 7 Segment display Interfacing with 8051 Microcontroller

Interfacing a 8051 microcontroller with a 7 segment Common Anode Display.

1. The 7 segment display is found in many displays such as microwaves or fancy toaster ovens and occasionally in non cooking devices. It is just 7 LEDs that have been combined into one case to make a convenient device for displaying numbers and some letters. The display is shown on the left. The pin out of the display is on the right.

This version is a common anode version. That means that the positive leg of each LED is connected to a common point which is pin 3 in this case. Each LED has a negative leg that is connected to one of the pins of the device. To make it work you need to connect pin 3 to 5 volts.

Then to make each segment light up, connect the ground pin for that led to ground. A resistor is required to limit the current. Rather than using a resistor from each LED to ground, you can just use one resistor from Vcc to pin 3 to limit the current.

The following table shows how to form the numbers 0 to 9 and the letters A, b, C, d, E, and F. '0' means that pin is connected to ground. '1' means that pin is connected to Vcc.

Now, we want to run the display with the AT89C51 microcontroller. We will use Port 0 to run the display. Connect the AT89C51 to the 7 segment display as follows.

4.3 LCD interfacing with 8051 Microcontroller

4.3.1 LCD DISPLAY

We always use devices made up of Liquid Crystal Displays (LCDs) like computers, digital watches and also DVD and CD players. They have become very common and have taken a giant leap in the screen industry by clearly replacing the use of Cathode Ray Tubes (CRT). CRT draws more power than LCD and are also bigger and heavier. All of us have seen a LCD, but no one knows the exact working of it. Let us take a look at the working of a LCD.

Here we are using alphanumeric LCD 16×2. A 16×2 LCD display is very basic module and is very commonly used in various devices and circuits. These modules are preferred over seven segments and other multi segment LEDs. The reasons being: LCDs are economical; easily programmable; have no limitation of displaying special & even custom characters (unlike in seven segments), animations and so on.

A 16×2 LCD means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5×7 pixel matrix. This LCD has two registers, namely, Command and Data. The command register stores the command instructions given to the LCD. A command is

an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. The data register stores the data to be displayed on the
LCD. The data is the ASCII value of the character to be displayed on the LCD.
Pin Diagram

Pin Description
Pin
No Function Name
1 Ground (0V) Ground
2 Supply voltage; 5V (4.7V – 5.3V) Vcc
3 Contrast adjustment; through a variable resistor VEE
4 Selects command register when low; and data register when high Register Select
5 Low to write to the register; High to read from the register Read/write
6 Sends data to data pins when a high to low pulse is given Enable
7
8-bit data pins
DB0
8 DB1
9 DB2
10 DB3
11 DB4
12 DB5
13 DB6
14 DB7
15 Backlight VCC (5V) Led+
16 Backlight Ground (0V) Led-

The LCD display module requires 3 control lines as well as either 4 or 8 I/O lines for the data bus.
The user may select whether the LCD is to operate with a 4-bit data bus or an 8-bit data bus. If a 4-
bit data bus is used the LCD will require a total of 7 data lines (3 control lines plus the 4 lines for
the data bus). If an 8-bit data bus is used the LCD will require a total of 11 data lines (3 control lines plus the 8 lines for the data bus).
The three control lines are referred to as EN, RS, and RW.
The EN line is called “Enable.” This control line is used to tell the LCD that you are sending it data. To send data to the LCD, your program should make sure this line is low (0) and then set the
other two control lines and/or put data on the data bus. When the other lines are completely ready,
bring EN high (1) and wait for the minimum amount of time required by the LCD datasheet (this
varies from LCD to LCD), and end by bringing it low (0) again.
The RS line is the “Register Select” line. When RS is low (0), the data is to be treated as a command or special instruction (such as clear screen, position cursor, etc.). When RS is high (1),
the data being sent is text data which should be displayed on the screen. For example, to display the

letter “T” on the screen you would set RS high.
The RW line is the “Read/Write” control line. When RW is low (0), the information on the data
bus is being written to the LCD. When RW is high (1), the program is effectively querying (or reading) the LCD. Only one instruction (“Get LCD status”) is a read command. All others are write
commands–so RW will almost always be low.
Finally, the data bus consists of 4 or 8 lines (depending on the mode of operation selected by the
user). In the case of an 8-bit data bus, the lines are referred to as DB0, DB1, DB2, DB3, DB4, DB5, DB6, and DB7.
4.3.2 LCD COMMANDS
Now let’s move to programming.

4.3.3. CIRCUIT DIAGRAM
RS is connected to Port 0.0 (P0.0)
RW is connected to Port 0.1 (P0.1)
EN is connected to Port 0.2 (P0.2)
Data lines are connected into Port 2 (P2)
ADC (Analog to digital converter) forms a very essential part in many embedded projects and this article is about interfacing an ADC to 8051 embedded controller. ADC 0804 is the ADC used
here and before going through the interfacing procedure, we must neatly understand how the ADC
0804 works.
4.4 ADC INTERFACE
ADC0804 is an 8 bit successive approximation analogue to digital converter from National semiconductors. The features of ADC0804 are differential analogue voltage inputs, 0-5V input voltage range, no zero adjustment, built in clock generator, reference voltage can be externally adjusted to convert smaller analogue voltage span to 8 bit resolution etc. The pin out diagram of
ADC0804 is shown in the figure below.

ADC0804 pinout
The voltage at Vref/2 (pin9) of ADC0804 can be externally adjusted to convert smaller input voltage spans to full 8 bit resolution. Vref/2 (pin9) left open means input voltage span is 0-5V and
step size is 5/255=19.6V. Have a look at the table below for different Vref/2 voltages and corresponding analogue input voltage spans.
Vref/2 (pin9) (volts) Input voltage span (volts) Step size (mV)
Left open 0 – 5 5/255 = 19.6
2 0 – 4 4/255 = 15.69
1.5 0 – 3 3/255 = 11.76
1.28 0 – 2.56 2.56/255 = 10.04
1.0 0 – 2 2/255 = 7.84
0.5 0 – 1 1/255 = 3.92
4.4.1 Steps for converting the analogue input and reading the output from ADC0804.
 Make CS=0 and send a low to high pulse to WR pin to start the conversion.
 Now keep checking the INTR pin. INTR will be 1 if conversion is not finished and INTR will be 0

if conversion is finished.
 If conversion is not finished (INTR=1) , poll until it is finished.
 If conversion is finished (INTR=0), go to the next step.
 Make CS=0 and send a high to low pulse to RD pin to read the data from the ADC.

4.4.2 Circuit diagram.
Interfacing ADC to 8051
The figure above shows the schematic for interfacing ADC0804 to 8051. The circuit initiates the
ADC to convert a given analogue input , then accepts the corresponding digital data and displays it
on the LED array connected at P0. For example, if the analogue input voltage Vin is 5V then all
LEDs will glow indicating 11111111 in binary which is the equivalent of 255 in decimal. AT89s51
is the microcontroller used here. Data out pins (D0 to D7) of the ADC0804 are connected to the
port pins P1.0 to P1.7 respectively. LEDs D1 to D8 are connected to the port pins P0.0 to P0.7
respectively. Resistors R1 to R8 are current limiting resistors. In simple words P1 of the
microcontroller is the input port and P0 is the output port. Control signals for the ADC (INTR,
WR, RD and CS) are available at port pins P3.4 to P3.7 respectively. Resistor R9 and capacitor C1
are associated with the internal clock circuitry of the ADC. Preset resistor R10 forms a voltage
divider which can be used to apply a particular input analogue voltage to the ADC. Push button S1,
resistor R11 and capacitor C4 forms a debouncing reset mechanism. Crystal X1 and capacitors
C2,C3 are associated with the clock circuitry of the microcontroller.
4.5 RTC INTERFACE
4.5.1 RTC (DS1307)

Introduction
The DS1307 serial real-time clock (RTC) is a low-power, full binary-coded decimal (BCD)
clock/calendar plus 56 bytes of NV SRAM. Address and data are transferred serially through an
I²C, bidirectional bus. The clock/calendar provides seconds, minutes, hours, day, date, month, and
year information. The end of the month date is automatically adjusted for months with fewer than
31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour
format with AM/PM indicator. The DS1307 has a built-in power-sense circuit that detects power
failures and automatically switches to the backup supply. Timekeeping operation continues while
the part operates from the backup supply.
Features
 Completely Manages All Timekeeping Functions
 Real-Time Clock Counts Seconds, Minutes, Hours, Date of the Month, Month, Day of the
Week, and Year with Leap-Year Compensation Valid Up to 2100
 56-Byte, Battery-Backed, General-Purpose RAM with Unlimited Writes

 Programmable Square-Wave Output Signal
 Simple Serial Port Interfaces to Most Microcontrollers
 I
2C Serial Interface
 Low Power Operation Extends Battery Backup Run Time
 Consumes Less than 500nA in Battery-Backup Mode with Oscillator Running
 Automatic Power-Fail Detect and Switch Circuitry
 8-Pin DIP and 8-Pin SO Minimizes Required Space
 Optional Industrial Temperature Range: -40°C to +85°C Supports Operation in a Wide Range
of Applications
 Underwriters Laboratories®
(UL) Recognized
VCC, GND: These pins are used to provide the power to the chip. When 5V is applied within normal limits, the device is fully accessible and data can be written and read. When a 3V battery is
connected to the device and VCC is below 1.25 x VBAT, reads and writes are inhibited. However,
the timekeeping function continues unaffected by the lower input voltage. As VCC falls below

VBAT the RAM and timekeeper are switched over to the external power supply (nominal 3.0V DC) at VBAT.
X1-X2:Pins to connect the external 32.768kHz oscillator that provides the clock source to the chip.
Vbat: A 3.3v Lithium battery can be connected to this pin in order to provide the power source when the external supply voltage is not available. Battery voltage must be held between 2.0V and
3.5V for proper operation.
SCL: This pin must be connected to SCL pin of the I2C Bus/Master.
SDA: This pin must be connected to SDA pin of the I2C Bus/Master.
SQW/OUT: When enabled, the SQWE bit set to 1, the SQW/OUT pin outputs one of four square
wave frequencies (1Hz, 4kHz, 8kHz, 32kHz).
 Note: The SCL,SDA and SQW are open drain and must be pulled up with appropriate pull up
resistors
Connection circuit
DS1307 requires very less number of components to operate. One of the mandatory component is
the crystal oscillator of 32.768kHz. The battery backup is optional as you can connect power source to Vbatt pin. But It is good to have battery connected. When running on battery it only consumes 500 nA of current just for the operation of clock and to maintain content of NV RAM.
Your can use a coin cell type battery as backup source for DS1307.
Following image shows typical connection diagram for DS1307 RTC.
DS1307 Memory
All the registers of DS1307 stores value as BCD, i.e. if clock is at 49 seconds then register 0x00
will show 0x49 as binary coded decimal value of 49. This makes it easy for programmer to read

and display number on LCD or serial terminal. So if you want to use register values of RTC then
first convert the value from BCD to decimal and then use it.
The RTC keeps the date and time arranged in it's memory as shown below:

Register 0x07 is control register for square wave output pin. RS0 and RS1 bit selects the output frequency as per table below:
You can also use SQW/Out pin as GPO pin, when SQW function of DS1307 is not used. Bit 7 controls the output level of the pin. If OUT bit is 1 then OUT pin is high and when 0 OUT pin will
be low.
4.5.2 RTC Interfacing with 8051
DS1307 uses I2C Protocol and act as a slave device and I2C Master can read/write register of RTC.
To communicate with the slave device, master need the slave address of device connected on bus.
DS1307 has fixed slave address which makes it impossible to connect two RTC devices on same
bus, don't worry occurrence of such a scenario is close to zero.
Slave address for I2C Write: 0b11010000 = 0xD0
Slave address for I2C Read: 0b11010001 = 0xD1
Components Required
- 8051 Development board
- LCD Module
- Keypad
- RTC DS1307
- Buzzer

Circuit Diagram

EXPLANATION
- The RTC is connected to the Port 3.
- The LCD data lines are given to the Port2. RS is given to port pin P0.0, RW to P0.1and enable
to P0.2.
- Keypad is connected in to P1.
- Buzzer is connected into P0.7.

QUESTIONS:
MULTIPLE CHOICE QUESTIONS:
1)which of following is ADC:
(a) 0804 (b) timer 1
(c) ORG (d)all
2) which of following is a part of LCD:
(a)CRT (b) register
(c) intruppt (d) all of these above

SHORT ANSWER TYPES QUESTIONS:
1) What is keypad?
2) LCD stands for ……..
3) What do you mean by 7 segment display?

4) What is function of ADC?
5) Write any five LCD commands.

LONG ANSWER TYPES QUESTIONS:

1) Explain in detail keypad interface.
2) Explain in detail RTC interface.

Left
 RLC - Rotate Accumulator Left Through Carry
 RR - Rotate Accumulator Right
 RRC - Rotate Accumulator Right Through Carry
 SETB - Set Bit
 SJMP - Short Jump
 SUBB - Subtract From Accumulator With Borrow
 SWAP - Swap Accumulator Nibbles
 XCH - Exchange Bytes
 XCHD - Exchange Digits
 XRL - Bitwise Exclusive OR
 Undefined - Undefined Instruction

8051 Instruction Set: ACALL
Operation: ACALL
Function: Absolute Call Within 2K Block
Syntax: ACALL code address

Instructions	OpCode	Bytes	Flags
ACALL page0	0x11	2	None
ACALL page1	0x31	2	None
ACALL page2	0x51	2	None
ACALL page3	0x71	2	None
ACALL page4	0x91	2	None
ACALL page5	0xB1	2	None
ACALL page6	0xD1	2	None
ACALL page7	0xF1	2	None

Description: ACALL unconditionally calls a subroutine at the indicated code address. ACALL pushes the address of the instruction that follows ACALL onto the stack, least-significant-byte first, most-significant-byte second. The Program Counter is then updated so that program execution
continues at the indicated address.

The new value for the Program Counter is calculated by replacing the least-significant-byte of the
Program Counter with the second byte of the ACALL instruction, and replacing bits 0-2 of the most-significant-byte of the Program Counter with 3 bits that indicate the page. Bits 3-7 of the most-significant-byte of the Program Counter remain unchaged.
Since only 11 bits of the Program Counter are affected by ACALL, calls may only be made to routines located within the same 2k block as the first byte that follows ACALL.
See Also: LCALL, RET
8051 Instruction Set: ADD
Operation: ADD, ADDC
Function: Add Accumulator, Add Accumulator With Carry
Syntax: ADD A,operand
ADDC A,operand

Instructions	OpCode	Bytes	Flags
ADD A,#data	0x24	2	C, AC, OV
ADD A,iram addr	0x25	2	

C, AC,
OV
ADD A,@R0 0x26 1
C, AC,
OV
ADD A,@R1 0x27 1
C, AC,
OV
ADD A,R0 0x28 1
C, AC,
OV
ADD A,R1 0x29 1
C, AC,
OV
ADD A,R2 0x2A 1
C, AC,
OV
Instructions OpCode Bytes Flags
ADDC A,#data 0x34 2
C, AC,
OV
ADDC A,iram
addr 0x35 2
C, AC,
OV
ADDC A,@R0 0x36 1
C, AC,
OV
ADDC A,@R1 0x37 1
C, AC,
OV
ADDC A,R0 0x38 1
C, AC,
OV
ADDC A,R1 0x39 1
C, AC,
OV
ADDC A,R2 0x3A 1
C, AC,
OV

ADD A,R3 0x2B 1
C, AC,
OV
ADD A,R4 0x2C 1
C, AC,
OV
ADD A,R5 0x2D 1
C, AC,
OV

ADD A,R6 0x2E 1
C, AC,
OV
ADD A,R7 0x2F 1
C, AC,
OV
ADDC A,R3 0x3B 1
C, AC,
OV
ADDC A,R4 0x3C 1
C, AC,
OV
ADDC A,R5 0x3D 1
C, AC,
OV
ADDC A,R6 0x3E 1
C, AC,
OV
ADDC A,R7 0x3F 1
C, AC,
OV
Description: Description: ADD and ADDC both add the value operand to the value of the Accumulator, leaving the resulting value in the Accumulator. The value operand is not affected. ADD and ADDC function identically except that ADDC adds the value of operand as well as the
value of the Carry flag whereas ADD does not add the Carry flag to the result.
The Carry bit (C) is set if there is a carry-out of bit 7. In other words, if the unsigned summed value of the Accumulator, operand and (in the case of ADDC) the Carry flag exceeds 255 Carry is
set. Otherwise, the Carry bit is cleared.
The Auxillary Carry (AC) bit is set if there is a carry-out of bit 3. In other words, if the unsigned summed value of the low nibble of the Accumulator, operand and (in the case of ADDC) the Carry
flag exceeds 15 the Auxillary Carry flag is set. Otherwise, the Auxillary Carry flag is cleared.
The Overflow (OV) bit is set if there is a carry-out of bit 6 or out of bit 7, but not both. In other words, if the addition of the Accumulator, operand and (in the case of ADDC) the Carry flag treated as signed values results in a value that is out of the range of a signed byte (-128 through +127) the Overflow flag is set. Otherwise, the Overflow flag is cleared.
See Also: SUBB, DA, INC, DEC
8051 Instruction Set: AJMP
Operation: AJMP
Function: Absolute Jump Within 2K Block
Syntax: AJMP code address

Instructions OpCode Bytes Flags
AJMP page0 0x01 2 None
AJMP page1 0x21 2 None
AJMP page2 0x41 2 None
AJMP page3 0x61 2 None
AJMP page4 0x81 2 None

AJMP page5 0xA1 2 None
AJMP page6 0xC1 2 None
AJMP page7 0xE1 2 None
Description: AJMP unconditionally jumps to the indicated code address. The new value for the Program Counter is calculated by replacing the least-significant-byte of the Program Counter with
the second byte of the AJMP instruction, and replacing bits 0-2 of the most-significant-byte of the
Program Counter with 3 bits that indicate the page of the byte following the AJMP instruction. Bits
3-7 of the most-significant-byte of the Program Counter remain unchaged.
Since only 11 bits of the Program Counter are affected by AJMP, jumps may only be made to code
located within the same 2k block as the first byte that follows AJMP.
See Also: LJMP, SJMP
8051 Instruction Set: ANL
Operation: ANL
Function: Bitwise AND
Syntax: ANL operand1, operand2
Instructions OpCode Bytes Flags
ANL iram addr,A 0x52 2 None

ANL iram addr,#data 0x53 3 None
ANL A,#data 0x54 2 None
ANL A,iram addr 0x55 2 None
ANL A,@R0 0x56 1 None
ANL A,@R1 0x57 1 None
ANL A,R0 0x58 1 None
ANL A,R1 0x59 1 None
ANL A,R2 0x5A 1 None
ANL A,R3 0x5B 1 None
ANL A,R4 0x5C 1 None
ANL A,R5 0x5D 1 None
ANL A,R6 0x5E 1 None
ANL A,R7 0x5F 1 None
ANL C,bit addr 0x82 2 C
ANL C,/bit addr 0xB0 2 C
Description: ANL does a bitwise "AND" operation between operand1 and operand2, leaving the
resulting value in operand1. The value of operand2 is not affected. A logical "AND" compares the
bits of each operand and sets the corresponding bit in the resulting byte only if the bit was set in
both of the original operands, otherwise the resulting bit is cleared.
See Also: ORL, XRL
8051 Instruction Set: CJNE
Operation: CJNE

Function: Compare and Jump If Not Equal
Syntax: CJNE operand1,operand2,reladdr

Instructions OpCode Bytes Flags
CJNE A,#data,reladdr 0xB4 3 C
CJNE A,iram addr,reladdr 0xB5 3 C
CJNE @R0,#data,reladdr 0xB6 3 C
CJNE @R1,#data,reladdr 0xB7 3 C
CJNE R0,#data,reladdr 0xB8 3 C
CJNE R1,#data,reladdr 0xB9 3 C
CJNE R2,#data,reladdr 0xBA 3 C
CJNE R3,#data,reladdr 0xBB 3 C
CJNE R4,#data,reladdr 0xBC 3 C
CJNE R5,#data,reladdr 0xBD 3 C
CJNE R6,#data,reladdr 0xBE 3 C
CJNE R7,#data,reladdr 0xBF 3 C
Description: CJNE compares the value of operand1 and operand2 and branches to the indicated relative address if operand1 and operand2 are not equal. If the two operands are equal program flow continues with the instruction following the CJNE instruction.
The Carry bit (C) is set if operand1 is less than operand2, otherwise it is cleared.
See Also: DJNZ
8051 Instruction Set: CLR
Operation: CLR

Function: Clear Register
Syntax: CLR register
Instructions OpCode Bytes Flags
CLR bit addr 0xC2 2 None
CLR C 0xC3 1 C
CLR A 0xE4 1 None
Description: CLR clears (sets to 0) all the bit(s) of the indicated register. If the register is a bit (including the carry bit), only the specified bit is affected. Clearing the Accumulator sets the Accumulator's value to 0.
See Also: SETB
8051 Instruction Set: CPL
Operation: CPL
Function: Complement Register
Syntax: CPL operand
Instructions OpCode Bytes Flags
CPL A 0xF4 1 None
CPL C 0xB3 1 C
CPL bit addr 0xB2 2 None
Description: CPL complements operand, leaving the result in operand. If operand is a single bit
then the state of the bit will be reversed. If operand is the Accumulator then all the bits in the Accumulator will be reversed. This can be thought of as "Accumulator Logical Exclusive OR 255"
or as "255-Accumulator." If the operand refers to a bit of an output Port, the value that will be complemented is based on the last value written to that bit, not the last value read from it.
See Also: CLR, SETB

8051 Instruction Set: DA
Operation: DA

Function: Decimal Adjust Accumulator
Syntax: DA A
Instructions OpCode Bytes Flags
DA 0xD4 1 C
Description: DA adjusts the contents of the Accumulator to correspond to a BCD (Binary Coded
Decimal) number after two BCD numbers have been added by the ADD or ADDC instruction. If
the carry bit is set or if the value of bits 0-3 exceed 9, 0x06 is added to the accumulator. If the carry
bit was set when the instruction began, or if 0x06 was added to the accumulator in the first step, 0x60 is added to the accumulator.
The Carry bit (C) is set if the resulting value is greater than 0x99, otherwise it is cleared.
See Also: ADD, ADDC
8051 Instruction Set: DEC
Operation: DEC
Function: Decrement Register
Syntax: DEC register
Instructions OpCode Bytes Flags
DEC A 0x14 1 None
DEC iram addr 0x15 2 None
DEC @R0 0x16 1 None
DEC @R1 0x17 1 None
DEC R0 0x18 1 None
DEC R1 0x19 1 None

DEC R2 0x1A 1 None
DEC R3 0x1B 1 None
DEC R4 0x1C 1 None
DEC R5 0x1D 1 None
DEC R6 0x1E 1 None
DEC R7 0x1F 1 None
Description: DEC decrements the value of register by 1. If the initial value of register is 0, decrementing the value will cause it to reset to 255 (0xFF Hex). Note: The Carry Flag is NOT set
when the value "rolls over" from 0 to 255.
See Also: INC, SUBB
8051 Instruction Set: DIV
Operation: DIV
Function: Divide Accumulator by B
Syntax: DIV AB
Instructions OpCode Bytes Flags
DIV AB 0x84 1 C, OV
Description: Divides the unsigned value of the Accumulator by the unsigned value of the "B" register. The resulting quotient is placed in the Accumulator and the remainder is placed in the "B"
register.
The Carry flag (C) is always cleared.
The Overflow flag (OV) is set if division by 0 was attempted, otherwise it is cleared.
See Also: MUL AB

8051 Instruction Set: DJNZ

Operation: DJNZ
Function: Decrement and Jump if Not Zero
Syntax: DJNZ register,reladdr

Instructions	OpCode	Bytes	Flags
DJNZ iram addr,reladdr	0xD5	3	None
DJNZ R0,reladdr	0xD8	2	None
DJNZ R1,reladdr	0xD9	2	None
DJNZ R2,reladdr	0xDA	2	None
DJNZ R3,reladdr	0xDB	2	None
DJNZ R4,reladdr	0xDC	2	None
DJNZ R5,reladdr	0xDD	2	None
DJNZ R6,reladdr	0xDE	2	None
DJNZ R7,reladdr	0xDF	2	None

Description: DJNZ decrements the value of register by 1. If the initial value of register is 0, decrementing the value will cause it to reset to 255 (0xFF Hex). If the new value of register is not 0
the program will branch to the address indicated by relative addr. If the new value of register is 0
program flow continues with the instruction following the DJNZ instruction.
See Also: DEC, JZ, JNZ
8051 Instruction Set: INC
Operation: INC
Function: Increment Register
Syntax: INC register

Instructions	OpCode	Bytes	Flags
INC A	0x04	1	None
INC iram addr	0x05	2	None
INC @R0	0x06	1	None
INC @R1	0x07	1	None
INC R0	0x08	1	None
INC R1	0x09	1	None
INC R2	0x0A	1	None
INC R3	0x0B	1	None
INC R4	0x0C	1	None
INC R5	0x0D	1	None
INC R6	0x0E	1	None
INC R7	0x0F	1	None
INC DPTR	0xA3	1	None

Description: INC increments the value of register by 1. If the initial value of register is 255 (0xFF
Hex), incrementing the value will cause it to reset to 0. Note: The Carry Flag is NOT set when the
value "rolls over" from 255 to 0.
In the case of "INC DPTR", the value two-byte unsigned integer value of DPTR is incremented. If
the initial value of DPTR is 65535 (0xFFFF Hex), incrementing the value will cause it to reset to 0.

Again, the Carry Flag is NOT set when the value of DPTR "rolls over" from 65535 to 0.
See Also: ADD, ADDC, DEC
8051 Instruction Set: JB

Operation: JB
Function: Jump if Bit Set
Syntax: JB bit addr, reladdr
Instructions OpCode Bytes Flags
JB bit addr,reladdr 0x20 3 None
Description: JB branches to the address indicated by reladdr if the bit indicated by bit addr is set.
If the bit is not set program execution continues with the instruction following the JB instruction.
See Also: JBC, JNB
8051 Instruction Set: JBC
Operation: JBC
Function: Jump if Bit Set and Clear Bit
Syntax: JB bit addr, reladdr
Instructions OpCode Bytes Flags
JBC bit addr,reladdr 0x10 3 None
Description: JBC will branch to the address indicated by reladdr if the bit indicated by bit addr is
set. Before branching to reladdr the instruction will clear the indicated bit. If the bit is not set program execution continues with the instruction following the JBC instruction.
See Also: JB, JNB
8051 Instruction Set: JC
Operation: JC
Function: Jump if Carry Set
Syntax: JC reladdr
Instructions OpCode Bytes Flags

JC reladdr 0x40 2 None
Description: JC will branch to the address indicated by reladdr if the Carry Bit is set. If the Carry
Bit is not set program execution continues with the instruction following the JC instruction.
See Also: JNC
8051 Instruction Set: JMP
Operation: JMP
Function: Jump to Data Pointer + Accumulator
Syntax: JMP @A+DPTR
Instructions OpCode Bytes Flags
JMP @A+DPTR 0x73 1 None
Description: JMP jumps unconditionally to the address represented by the sum of the value of DPTR and the value of the Accumulator.
See Also: LJMP, AJMP, SJMP
8051 Instruction Set: JNB
Operation: JNB
Function: Jump if Bit Not Set
Syntax: JNB bit addr,reladdr
Instructions OpCode Bytes Flags

JNB bit addr,reladdr 0x30 3 None
Description: JNB will branch to the address indicated by reladdress if the indicated bit is not set.
If the bit is set program execution continues with the instruction following the JNB instruction.
See Also: JB, JBC

8051 Instruction Set: JNC
Operation: JNC
Function: Jump if Carry Not Set
Syntax: JNC reladdr
Instructions OpCode Bytes Flags
JNC reladdr 0x50 2 None
Description: JNC branches to the address indicated by reladdr if the carry bit is not set. If the carry bit is set program execution continues with the instruction following the JNB instruction.
See Also: JC
8051 Instruction Set: JNZ
Operation: JNZ
Function: Jump if Accumulator Not Zero
Syntax: JNZ reladdr
Instructions OpCode Bytes Flags
JNZ reladdr 0x70 2 None
Description: JNZ will branch to the address indicated by reladdr if the Accumulator contains any
value except 0. If the value of the Accumulator is zero program execution continues with the instruction following the JNZ instruction.
See Also: JZ
8051 Instruction Set: JZ
Operation: JZ
Function: Jump if Accumulator Zero
Syntax: JNZ reladdr

Instructions OpCode Bytes Flags
JZ reladdr 0x60 2 None
Description: JZ branches to the address indicated by reladdr if the Accumulator contains the value
0. If the value of the Accumulator is non-zero program execution continues with the instruction following the JNZ instruction.
See Also: JNZ
8051 Instruction Set: LCALL
Operation: LCALL
Function: Long Call
Syntax: LCALL code addr
Instructions OpCode Bytes Flags
LCALL code addr 0x12 3 None
Description: LCALL calls a program subroutine. LCALL increments the program counter by 3 (to
point to the instruction following LCALL) and pushes that value onto the stack (low byte first, high
byte second). The Program Counter is then set to the 16-bit value which follows the LCALL opcode, causing program execution to continue at that address.

See Also: ACALL, RET
8051 Instruction Set: LJMP
Operation: LJMP
Function: Long Jump
Syntax: LJMP code addr
Instructions OpCode Bytes Flags
LJMP code addr 0x02 3 None
Description: LJMP jumps unconditionally to the specified code addr.

See Also: AJMP, SJMP, JMP
8051 Instruction Set: MOV
Operation: MOV
Function: Move Memory
Syntax: MOV operand1,operand2
Instructions OpCode Bytes Flags
MOV @R0,#data 0x76 2 None
MOV @R1,#data 0x77 2 None
MOV @R0,A 0xF6 1 None
MOV @R1,A 0xF7 1 None
MOV @R0,iram addr 0xA6 2 None
MOV @R1,iram addr 0xA7 2 None
MOV A,#data 0x74 2 None
MOV A,@R0 0xE6 1 None
MOV A,@R1 0xE7 1 None
MOV A,R0 0xE8 1 None
MOV A,R1 0xE9 1 None
MOV A,R2 0xEA 1 None
MOV A,R3 0xEB 1 None
MOV A,R4 0xEC 1 None

MOV A,R5 0xED 1 None
MOV A,R6 0xEE 1 None
MOV A,R7 0xEF 1 None
MOV A,iram addr 0xE5 2 None
MOV C,bit addr 0xA2 2 C
MOV DPTR,#data16 0x90 3 None
MOV R0,#data 0x78 2 None
MOV R1,#data 0x79 2 None
MOV R2,#data 0x7A 2 None
MOV R3,#data 0x7B 2 None
MOV R4,#data 0x7C 2 None
MOV R5,#data 0x7D 2 None
MOV R6,#data 0x7E 2 None
MOV R7,#data 0x7F 2 None
MOV R0,A 0xF8 1 None
MOV R1,A 0xF9 1 None
MOV R2,A 0xFA 1 None
MOV R3,A 0xFB 1 None
MOV R4,A 0xFC 1 None
MOV R5,A 0xFD 1 None

MOV R6,A 0xFE 1 None
MOV R7,A 0xFF 1 None
MOV R0,iram addr 0xA8 2 None
MOV R1,iram addr 0xA9 2 None
MOV R2,iram addr 0xAA 2 None
MOV R3,iram addr 0xAB 2 None
MOV R4,iram addr 0xAC 2 None
MOV R5,iram addr 0xAD 2 None
MOV R6,iram addr 0xAE 2 None
MOV R7,iram addr 0xAF 2 None
MOV bit addr,C 0x92 2 None
MOV iram addr,#data 0x75 3 None
MOV iram addr,@R0 0x86 2 None
MOV iram addr,@R1 0x87 2 None
MOV iram addr,R0 0x88 2 None
MOV iram addr,R1 0x89 2 None
MOV iram addr,R2 0x8A 2 None
MOV iram addr,R3 0x8B 2 None
MOV iram addr,R4 0x8C 2 None
MOV iram addr,R5 0x8D 2 None

MOV iram addr,R6 0x8E 2 None
MOV iram addr,R7 0x8F 2 None
MOV iram addr,A 0xF5 2 None
MOV iram addr,iram addr 0x85 3 None
Description: MOV copies the value of operand2 into operand1. The value of operand2 is not affected. Both operand1 and operand2 must be in Internal RAM. No flags are affected unless the
instruction is moving the value of a bit into the carry bit in which case the carry bit is affected or
unless the instruction is moving a value into the PSW register (which contains all the program flags).
** Note: In the case of "MOV iram addr,iram addr", the operand bytes of the instruction are stored
in reverse order. That is, the instruction consisting of the bytes 0x85, 0x20, 0x50 means "Move the
contents of Internal RAM location 0x20 to Internal RAM location 0x50" whereas the opposite would be generally presumed.
See Also: MOVC, MOVX, XCH, XCHD, PUSH, POP
8051 Instruction Set: MOVC
Operation: MOVC
Function: Move Code Byte to Accumulator
Syntax: MOVC A,@A+register
Instructions OpCode Bytes Flags
MOVC A,@A+DPTR 0x93 1 None
MOVC A,@A+PC 0x83 1 None
Description: MOVC moves a byte from Code Memory into the Accumulator. The Code Memory

address from which the byte will be moved is calculated by summing the value of the Accumulator
with either DPTR or the Program Counter (PC). In the case of the Program Counter, PC is first incremented by 1 before being summed with the Accumulator.
See Also: MOV, MOVX

8051 Instruction Set: MOVX
Operation: MOVX
Function: Move Data To/From External Memory (XRAM)
Syntax: MOVX operand1,operand2
Instructions OpCode Bytes Flags
MOVX @DPTR,A 0xF0 1 None
MOVX @R0,A 0xF2 1 None
MOVX @R1,A 0xF3 1 None
MOVX A,@DPTR 0xE0 1 None
MOVX A,@R0 0xE2 1 None
MOVX A,@R1 0xE3 1 None
Description: MOVX moves a byte to or from External Memory into or from the Accumulator.
If operand1 is @DPTR, the Accumulator is moved to the 16-bit External Memory address indicated by DPTR. This instruction uses both P0 (port 0) and P2 (port 2) to output the 16-bit address and data. If operand2 is DPTR then the byte is moved from External Memory into the Accumulator.
If operand1 is @R0 or @R1, the Accumulator is moved to the 8-bit External Memory address indicated by the specified Register. This instruction uses only P0 (port 0) to output the 8-bit address
and data. P2 (port 2) is not affected. If operand2 is @R0 or @R1 then the byte is moved from External Memory into the Accumulator.
See Also: MOV, MOVC
8051 Instruction Set: MUL
Operation: MUL
Function: Multiply Accumulator by B
Syntax: MUL AB

Instructions OpCode Bytes Flags
MUL AB 0xA4 1 C, OV
Description: Multiples the unsigned value of the Accumulator by the unsigned value of the "B" register. The least significant byte of the result is placed in the Accumulator and the mostsignificant-byte is placed in the "B" register.
The Carry Flag (C) is always cleared.
The Overflow Flag (OV) is set if the result is greater than 255 (if the most-significant byte is not
zero), otherwise it is cleared.
See Also: DIV
8051 Instruction Set: NOP
Operation: NOP
Function: None, waste time
Syntax: No Operation
Instructions OpCode Bytes Flags
NOP 0x00 1 None

Description: NOP, as it's name suggests, causes No Operation to take place for one machine cycle.
NOP is generally used only for timing purposes. Absolutely no flags or registers are affected.
8051 Instruction Set: ORL
Operation: ORL
Function: Bitwise OR
Syntax: ORL operand1,operand2
Instructions OpCode Bytes Flags
ORL iram addr,A 0x42 2 None

ORL iram addr,#data 0x43 3 None
ORL A,#data 0x44 2 None
ORL A,iram addr 0x45 2 None
ORL A,@R0 0x46 1 None
ORL A,@R1 0x47 1 None
ORL A,R0 0x48 1 None
ORL A,R1 0x49 1 None
ORL A,R2 0x4A 1 None
ORL A,R3 0x4B 1 None
ORL A,R4 0x4C 1 None
ORL A,R5 0x4D 1 None
ORL A,R6 0x4E 1 None
ORL A,R7 0x4F 1 None
ORL C,bit addr 0x72 2 C
ORL C,/bit addr 0xA0 2 C
Description: ORL does a bitwise "OR" operation between operand1 and operand2, leaving the resulting value in operand1. The value of operand2 is not affected. A logical "OR" compares the
bits of each operand and sets the corresponding bit in the resulting byte if the bit was set in either
of the original operands, otherwise the resulting bit is cleared.
See Also: ANL, XRL
8051 Instruction Set: POP
Operation: POP

Function: Pop Value From Stack
Syntax: POP
Instructions OpCode Bytes Flags
POP iram addr 0xD0 2 None
Description: POP "pops" the last value placed on the stack into the iram addr specified. In other words, POP will load iram addr with the value of the Internal RAM address pointed to by the current Stack Pointer. The stack pointer is then decremented by 1.
See Also: PUSH
8051 Instruction Set: PUSH
Operation: PUSH
Function: Push Value Onto Stack
Syntax: PUSH
Instructions OpCode Bytes Flags
PUSH iram addr 0xC0 2 None
Description: PUSH "pushes" the value of the specified iram addr onto the stack. PUSH first

increments the value of the Stack Pointer by 1, then takes the value stored in iram addr and stores
it in Internal RAM at the location pointed to by the incremented Stack Pointer.
See Also: POP
8051 Instruction Set: RET
Operation: RET
Function: Return From Subroutine
Syntax: RET
Instructions OpCode Bytes Flags

RET 0x22 1 None
Description: RET is used to return from a subroutine previously called by LCALL or ACALL. Program execution continues at the address that is calculated by popping the topmost 2 bytes off
the stack. The most-significant-byte is popped off the stack first, followed by the least-significantbyte.
See Also: LCALL, ACALL, RETI
8051 Instruction Set: RETI
Operation: RETI
Function: Return From Interrupt
Syntax: RETI
Instructions OpCode Bytes Flags
RETI 0x32 1 None
Description: RETI is used to return from an interrupt service routine. RETI first enables interrupts
of equal and lower priorities to the interrupt that is terminating. Program execution continues at the
address that is calculated by popping the topmost 2 bytes off the stack. The most-significant-byte is
popped off the stack first, followed by the least-significant-byte.
RETI functions identically to RET if it is executed outside of an interrupt service routine.
See Also: RET
8051 Instruction Set: RL
Operation: RL
Function: Rotate Accumulator Left
Syntax: RL A
Instructions OpCode Bytes Flags
RL A 0x23 1 C

Description: Shifts the bits of the Accumulator to the left. The left-most bit (bit 7) of the Accumulator is loaded into bit 0.
See Also: RLC, RR, RRC
8051 Instruction Set: RLC
Operation: RLC
Function: Rotate Accumulator Left Through Carry
Syntax: RLC A
Instructions OpCode Bytes Flags
RLC A 0x33 1 C
Description: Shifts the bits of the Accumulator to the left. The left-most bit (bit 7) of the

Accumulator is loaded into the Carry Flag, and the original Carry Flag is loaded into bit 0 of the
Accumulator. This function can be used to quickly multiply a byte by 2.
See Also: RL, RR, RRC
8051 Instruction Set: RR
Operation: RR
Function: Rotate Accumulator Right
Syntax: RR A
Instructions OpCode Bytes Flags
RR A 0x03 1 None
Description: Shifts the bits of the Accumulator to the right. The right-most bit (bit 0) of the Accumulator is loaded into bit 7.
See Also: RL, RLC, RRC
8051 Instruction Set: RRC

Operation: RRC
Function: Rotate Accumulator Right Through Carry
Syntax: RRC A
Instructions OpCode Bytes Flags
RRC A 0x13 1 C
Description: Shifts the bits of the Accumulator to the right. The right-most bit (bit 0) of the Accumulator is loaded into the Carry Flag, and the original Carry Flag is loaded into bit 7. This function can be used to quickly divide a byte by 2.
See Also: RL, RLC, RR
8051 Instruction Set: SETB
Operation: SETB
Function: Set Bit
Syntax: SETB bit addr
Instructions OpCode Bytes Flags
SETB C 0xD3 1 C
SETB bit addr 0xD2 2 None
Description: Sets the specified bit.
See Also: CLR
8051 Instruction Set: SJMP
Operation: SJMP
Function: Short Jump
Syntax: SJMP reladdr

Instructions OpCode Bytes Flags
SJMP reladdr 0x80 2 None
Description: SJMP jumps unconditionally to the address specified reladdr. Reladdr must be within
-128 or +127 bytes of the instruction that follows the SJMP instruction.
See Also: LJMP, AJMP
8051 Instruction Set: SUBB
Operation: SUBB
Function: Subtract from Accumulator With Borrow
Syntax: SUBB A,operand
Instructions OpCode Bytes Flags
SUBB A,#data 0x94 2 C, AC, OV

SUBB A,iram addr 0x95 2 C, AC, OV
SUBB A,@R0 0x96 1 C, AC, OV
SUBB A,@R1 0x97 1 C, AC, OV
SUBB A,R0 0x98 1 C, AC, OV
SUBB A,R1 0x99 1 C, AC, OV
SUBB A,R2 0x9A 1 C, AC, OV
SUBB A,R3 0x9B 1 C, AC, OV
SUBB A,R4 0x9C 1 C, AC, OV
SUBB A,R5 0x9D 1 C, AC, OV
SUBB A,R6 0x9E 1 C, AC, OV

SUBB A,R7 0x9F 1 C, AC, OV
Description: SUBB subtract the value of operand and the Carry Flag from the value of the Accumulator, leaving the resulting value in the Accumulator. The value operand is not affected. The Carry Bit (C) is set if a borrow was required for bit 7, otherwise it is cleared. In other words,
if the unsigned value being subtracted is greater than the Accumulator the Carry Flag is set.
The Auxillary Carry (AC) bit is set if a borrow was required for bit 3, otherwise it is cleared. In
other words, the bit is set if the low nibble of the value being subtracted was greater than the low
nibble of the Accumulator.
The Overflow (OV) bit is set if a borrow was required for bit 6 or for bit 7, but not both. In other
words, the subtraction of two signed bytes resulted in a value outside the range of a signed byte (-
128 to 127). Otherwise it is cleared.
See Also: ADD, ADDC, DEC
8051 Instruction Set: SWAP
Operation: SWAP
Function: Swap Accumulator Nibbles
Syntax: SWAP A
Instructions OpCode Bytes Flags
SWAP A 0xC4 1 None
Description: SWAP swaps bits 0-3 of the Accumulator with bits 4-7 of the Accumulator. This instruction is identical to executing "RR A" or "RL A" four times.
See Also: RL, RLC, RR, RRC
8051 Instruction Set: Undefined Instruction
Operation: Undefined Instruction
Function: Undefined
Syntax: ???

Instructions OpCode Bytes Flags
??? 0xA5 1 C
Description: The "Undefined" instruction is, as the name suggests, not a documented instruction.
The 8051 supports 255 instructions and OpCode 0xA5 is the single OpCode that is not used by any
documented function. Since it is not documented nor defined it is not recommended that it be

executed. However, based on my research, executing this undefined instruction takes 1 machine
cycle and appears to have no effect on the system except that the Carry Bit always seems to be set.
Note: We received input from an 8052.com user that the undefined instruction really has a format
of Undefined bit1,bit2 and effectively copies the value of bit2 to bit1. In this case, it would be a
three-byte instruction. We haven't had an opportunity to verify or disprove this report, so we present it to the world as "additional information."
Note: It has been reported that Philips 8051 model P89C669 uses instruction prefix 0xA5 to let the
user access a different (extended) SFR area.
8051 Instruction Set: XCH
Operation: XCH
Function: Exchange Bytes
Syntax: XCH A,register
Instructions OpCode Bytes Flags
XCH A,@R0 0xC6 1 None
XCH A,@R1 0xC7 1 None
XCH A,R0 0xC8 1 None
XCH A,R1 0xC9 1 None
XCH A,R2 0xCA 1 None
XCH A,R3 0xCB 1 None
XCH A,R4 0xCC 1 None

XCH A,R5 0xCD 1 None
XCH A,R6 0xCE 1 None
XCH A,R7 0xCF 1 None
XCH A,iram addr 0xC5 2 None
Description: Exchanges the value of the Accumulator with the value contained in register.
See Also: MOV
8051 Instruction Set: XCHD
Operation: XCHD
Function: Exchange Digit
Syntax: XCHD A,[@R0/@R1]
Instructions OpCode Bytes Flags
XCHD A,@R0 0xD6 1 None
XCHD A,@R1 0xD7 1 None
Description: Exchanges bits 0-3 of the Accumulator with bits 0-3 of the Internal RAM address pointed to indirectly by R0 or R1. Bits 4-7 of each register are unaffected.
See Also: DA
8051 Instruction Set: XRL
Operation: XRL
Function: Bitwise Exclusive OR
Syntax: XRL operand1,operand2
Instructions OpCode Bytes Flags

XRL iram addr,A 0x62 2 None
XRL iram addr,#data 0x63 3 None

XRL A,#data 0x64 2 None
XRL A,iram addr 0x65 2 None
XRL A,@R0 0x66 1 None
XRL A,@R1 0x67 1 None
XRL A,R0 0x68 1 None
XRL A,R1 0x69 1 None
XRL A,R2 0x6A 1 None
XRL A,R3 0x6B 1 None
XRL A,R4 0x6C 1 None
XRL A,R5 0x6D 1 None
XRL A,R6 0x6E 1 None
XRL A,R7 0x6F 1 None

2.2 Different Types Of Addressing Modes In 8051.

Addressing:-The CPU can access data in a register or in memory or be provided as an immediate
value. These various ways of accessing data are called addressing modes.
The 8051 provides a total of five distinct addressing modes.
They are as follows:-

1) Immediate addressing modes.
2) Register addressing modes.
3) Direct addressing modes.
4) Register indirect - addressing mode.
5)Indexed addressing mode.

Let us study different modes with selective examples:-

1) Immediate Addressing mode:-

In This addressing mode, the source operated is a constant i.e. immediate
data. Note that the immediate data must be preceded by the pound sign"#".

Example:-

```
MOV A, # 82H ; load 82H into A
MOV R4, # 100 ; load the decimal value 100 in to R4
MOV B, # 40H ; load 40H in to B
MOV P1, # 55H ; load 55H directly to port 1
MOV DPTR , # 4521 H ; DPTR =4521 H
```

2) Register - Addressing Mode:-

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

Example:-

```
MOV A,RO ;copy the contents of R0 in to A
MOV R2,A ;copy the contents of A in to R2
ADD A,R5 ;Add contents of R5 to A
MOV R7,DPL ;load R7 with contents of DPL
```

3) Direct Addressing mode:-

In direct addressing mode the data is in a RAM Memory location whose
addressing is known and this address is given as a part of the instruction.
Although the entire 128 bytes of RAM com be addressed Using direct
addressing mode
It is most often used to access RAM locations 30-7FH

Example:-

MOV R0, 40H ; save content of RAM location 40H in Ro
MOV 7FH, A ; save content of A in to RAM location 7FH
MOV A,07H ;same as move A1 R7 is copies data in R7 to A
MOV 03H, 04H ; it copies data in R4 to R3

4) Indirect addressing mode:-

In the register indirect addressing mode a register is used as a pointer to the data .only register R0 and R1 are used for internal RAM indirect Data transfer. When they hold the address of Rom locations they must be preceded by the "@" symbol.

Example :-

MOV ,A, @R0 ;move contents of Ram location whose address is help By R0 in to A
MOV @ R1 ,B ;move contents of B in to RAM location whose Address is help by R1.

5) Indexed addressing mode :-

Indexed addressing mode is widely used in accessing data elements of look-up table entries located in the program Rom space of the 8051.

Example :-

MOV A, @ A + DPTR
MOV A, @ A+ PC

2.3 INTERRUPT 8051 MICROCONTROLLER

The most powerful and important features are interrupts in 8051 microcontroller. In most of the real-time processes, to handle certain conditions properly, the actual task must be halt for some time – it takes required action – and then must return to the main task. For executing such type of programs, interrupts are necessary. It entirely differs from the polling method wherein the processor must check sequentially each device and ask whether the service is required or not while consuming more processor time.

Interrupts in 8051 microcontroller

Interrupts in 8051 microcontroller are more desirable to reduce the regular status checking of the interfaced devices or inbuilt devices. Interrupt is an event that temporarily suspends the main program, passes the control to a special code section, executes the event-related function and resumes the main program flow where it had left off.

Interrupts are of different types like software and hardware, maskable and non-maskable, fixed and vector interrupts, and so on. Interrupt Service Routine (ISR) comes into the picture when interrupt occurs, and then tells the processor to take appropriate action for the interrupt, and after ISR execution, the controller jumps into the main program.

2.3.1 TYPES OF INTERRUPTS IN 8051 MICROCONTROLLER

The 8051 microcontroller can recognize five different events that cause the main program to interrupt from the normal execution. These five sources of interrupts in 8051are:

1. Timer 0 overflow interrupt- TF0
2. Timer 1 overflow interrupt- TF1
3. External hardware interrupt- INT0
4. External hardware interrupt- INT1

5. Serial communication interrupt- RI/TI
The Timer and Serial interrupts are internally generated by the microcontroller, whereas the external interrupts are generated by additional interfacing devices or switches that are externally
connected to the microcontroller. These external interrupts can be edge triggered or level triggered.
When an interrupt occurs, the microcontroller executes the interrupt service routine so that memory
location corresponds to the interrupt that enables it. The Interrupt corresponding to the memory location is given in the interrupt vector table below.
Interrupt vector Table
2.3.2 INTERRUPT STRUCTURE OF 8051 MICROCONTROLLER
Upon 'RESET' all the interrupts get disabled, and therefore, all these interrupts must be enabled by
a software. In all these five interrupts, if anyone or all are activated, this sets the corresponding interrupt flags as shown in the figure. All these interrupts can be set or cleared by bit in some special function register that is Interrupt Enabled (IE), and this in turn depends on the priority, which is executed by IP interrupt priority register.
Interrupt structure of 8051
microcontroller
Interrupt Enable (IE) Register: This register is responsible for enabling and disabling the interrupt. It is a bit addressable register in which EA must be set to one for enabling interrupts. The
corresponding bit in this register enables particular interrupt like timer, external and serial inputs.
In the below IE register, bit corresponding to 1 activates the interrupt and 0 disables the interrupt.

Interrupt Enable (IE) Register
Interrupt Priority Register (IP): It is also possible to change the priority levels of the interrupts by setting or clearing the corresponding bit in the Interrupt priority (IP) register as shown in the
figure. This allows the low priority interrupt to interrupt the high-priority interrupt, but prohibits
the interruption by another low-priority interrupt. Similarly, the high-priority interrupt cannot be
interrupted. If these interrupt priorities are not programmed, the microcontroller executes in predefined manner and its order is INT0, TF0, INT1, TF1, and SI.
IP register
TCON Register: In addition to the above two registers, the TCON register specifies the type of external interrupt to the 8051 microcontroller, as shown in the figure. The two external interrupts,
whether edge or level triggered, specify by this register by a set, or cleared by appropriate bits in it.
And, it is also a bit addressable register.
Timers in 8051 microcontroller
Timer is an important application in Embedded systems, it maintains the timing of an operation in
sync with a system clock or an external clock. Timer has so many applications such as measure

time generating delays, they can also be used for generating baud rates.
A normal delay function might be using for loop iterating for a few 1000 cycles. But these types of
delays need not be accurate and fundamentally it is not a good programming practice. So,
TIMER/COUNTER is a software designed to count the time interval between events. It counts the
cycle of the peripheral clock or an externally-supplied clock.
Timer in 8051
The AT89S8253 has three timers/counters marked T0, T1 and T2. Timers T0 and T1 completely
fall under the 8051 Standard. Their main purpose is to measure time and count external events.
Besides, they are used for generating clock pulses that can be used in serial communication, so called Baud Rate.
The microcontroller can also generate/measure the required time delays by running loops, but the
timer/counter relieves the CPU from that redundant and repetitive task, allowing it to allocate maximum processing time for other tasks.

Timer T2
Timer 2 is a 16-bit timer/counter installed only in new versions of the 8051 family. Unlike timers
T0 and T1, this timer consists of 4 registers. Two of them, TH2 and TL2, are connected serially in
order to form a larger 16-bit timer register. Like timers 0 and 1, it can operate either as a timer or as
an event counter. Another two registers, RCAP2H and RCAP2L, are also serially connected and
operate as capture registers. They are used to temporarily store the contents of the counter register.
The main advantage of this timer compared to timers 0 and 1 is that all read and swap operations are easily performed using one instruction. Similar to T0 and T1, it has four different modes of operation,
 Timer T2 in Capture mode
 Timer T2 in auto-reload mode
 Timer T2 as a baud rate generator
 Timer T2 as a clock generator
Register configuration for the Timer
TCON
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.
 TF1 bit is automatically set when the Timer 1 overflow.
 TR1 bit enables the Timer 1.
o 1 – Timer 1 is enabled.
o 0 – Timer 1 is disabled.
 TF0 bit is automatically set when the Timer 0 overflow.
 TR0 bit enables the timer 0.
o 1 – Timer 0 is enabled.
o 0 – Timer 0 is disabled.
TMOD

This register contains bits controlling the operation of timer 0 & 1. To select the operating mode
and the timer/counter operation of the timers we use TMOD register. Timer 0 and timer 1 are two
timer registers in 8051. Both of these registers use the same register called TMOD to set various timer operation modes.
 TMOD is an 8-bit register.
 The lower 4 bits are for Timer 0.

 The upper 4 bits are for Timer 1.
 In each case,
o The lower 2 bits are used to set the timer mode.
o The upper 2 bits to specify the operation.
Bits of this register have the following function:
 GATE enables and disables Timer by means of a signal brought to the INTx pin:
o 1 – Timer operates only if the INTx bit is set.
o 0 – Timer operates regardless of the logic state of the INTx bit.
 C/T selects pulses to be counted up by the timer/counter:
o 1 – Timer counts pulses brought to the Tx(Timer) pin.
o 0 – Timer counts pulses from the internal oscillator.
 M1, M0 These two bits select the operational mode Timer.
TIMER MODE 0 (13 bit mode)
MODE 0 is a 13 bit mode. In this mode the THx acts as an 8 bit timer & TLx acts as a 5 bit timer.
The TLx counts up to 31 & then resets to 00 & increment THx by 1. Suppose you load 0 in the timer then the timer will overflow in 2^13 i.e. 8192 machine cycles.
TIMER MODE 1 (16 bit mode)
MODE 1 is similar to MODE 0 except it is a 16 bit mode. In this mode the THx & TLx both acts as
an 8 bit timer. The TLx counts upto 255 & then resets to 00 & increment THx by 1. Since this is a
full 16 bit timer we can get maximum of 2^16 i.e. 65536 Machine cycle before the timer overflows.
TIMER MODE 2 (8 bit mode)
In this Mode TLx acts as the timer & THx contains the Reload Value i.e. THx is loaded in TLx everytime it overflows i.e. when TLx reaches 255 & is incremented then instead of resetting it to 0
it will be reset to the value stored in THx. This mode is very commonly used for generating baud
rate used in serial communication.
TIMER MODE 3 (Split Mode)
Timer mode “3” is known as split-timer mode. Timers 0 and 1 may be programmed to be in mode
0, 1, or 2 independently of a similar mode for the other timer. But in mode 3 the timers do not operate independently, if mode 3 is chosen for timer 0. When Timer 0 is placed in mode 3, it essentially becomes two separate 8-bit timers. Timer 0 is TL0 and Timer 1 is TH0. Both timers count from 0 to 255 and overflow back to 0. All the bits that are related to Timer 1 will now be tied
to TH0. Now placing timer 1 in mode 3 causes it to stop counting, the control bit TR1 and the

Timer 1 flag TF1 are now used by timer 0. So even if you use Timer 1 in Mode 0, 1 or 2 you won't
be able to START or STOP the timer & no INTERRUPT will be generated by Timer 1. The real
Timer 1 will be incremented every machine cycle no matter what.
Example :
 TMOD = 00000001, mode 1 of timer 0 is selected.
P a g e | 40
 TMOD = 00100000, mode 2 of timer 1 is selected.
 TMOD = 00010010, mode 2 of timer 0, and mode 1 of timer 1 are selected.
QUESTIONS:
MULTIPLE CHOICE QUESTIONS:
1)which of following is intruppt of 8051:
(a) timer o (b) timer 1
(c) internal hardware (d)all
2) which of following is a part instruction:
(a)opcode (b) register
(c) intruppt (d) all of these above

SHORT ANSWER TYPES QUESTIONS:
1) What is intruppt?
2) IP stands for ……..
3) What do you mean by addressing mode?
4) What is function of timer?
5) Write all types of addressing modes.
LONG ANSWER TYPES QUESTIONS:
1) Explain in detail addressing modes 8051microprocessor.
2) Explain in detail instructions of 8051 microprocessor.
P a g e | 41
P a g e | 1
MICROCONTROLLERS
LEARNING OBJECTIVES:
 Concept of assembler directives.
 Study of assembler operation.
 To know about debugger.
CHAPTER-3(ASSEMBLY PROGRAMMING FOR MICROCONTROLLER)
3.1. ASSEMBLER DIRECTIVES
Microcontroller 8051 has only one 8-bits data type and the size of each register is also 8 bits. The
job of the programmer is to break down data larger than 8 bits [00 to FFH, or 0 to 255 in decimal]
to be processed by the CPU. The data type used by the 8051 can be positive or negative.
DB [Define Byte]:
The DB directive is used to define the 8-bit data and the numbers can be in decimal,binary, hex or
ASCII formats. For decimal, the "D" after the decimal number is optional, but using "B" [binary]
and "H" [hexadecimal] is required. The assembler will convert the numbers in hex. To indicate
ASCII, simply put it in quotation marks 'like this'. The assembler will assign the ASCII code for

the numbers or characters automatically. The DB directive is the only directive that can be used to
define ASCII strings larger than two characters. It should be used for all ASCII data definitions.

```
ORG 300H
DATA1: DB 30 ;Decimal number [1E in HEX]
DATA2: DB 01010101B ;Binary [55 in HEX]
DATA3: DB 40H ; HEX number
ORG 500H
DATA4: DB "2482" ;ASCII Numbers
DATA5: DB "My name is Ayesha Imtiaz" ;ASCII Characters
```

Either single or double quotes can be used around ASCII strings. Can be useful for strings, which
contain a single quote such as "O'Really". DB is also used to allocate memory in byte-sized chunks.

3.2 Assembler Directives:

The following Assembler directives are widely used in 8051 Assembly language programming.

 ORG [Origin]:

The ORG directive is used to indicate the beginning of the address. The number
that comes after ORG can be either in HEX or in decimal. If the number is not followed by 'H', it is
decimal and the assembler will convert it into hex. Some assemblers use ".ORG" [notice the dot]
instead of "ORG" for the origin directive. For tha you need to check your assembler.

 EQU [Equate]:

EQU is used to define a constant without occupying a memory location. The
EQU directive does not set aside storage for a data item but associates a constant value with a data
label so that when the label appears in the program, its constant value will be substituted for the
label. Here uses EQU for the counter constant ans then the constant is used to load the R4 register.

```
COUNT EQU 22
.... ....
MOV R4,#COUNT
```

After executing the instruction "MOV R4,#COUNT", the register R4 will be loaded with the value
22 [# sign indicates it is a value, notice it].

3.2.1 What is the advantage of using EQU?

The answer is that, lets say in a program there is a constant value [a fixed value] used in many different places in the program, and the programmer wants to change its value through out the entire program. By the use of EQU, a programmer can change all valves at once and the assembler
will change all of it occurrences, rather than search the entire program and to change the value one
by one to fine every occurrence, just change the constant value followed by EQU results changing
the all occurrences at once.

 END Directive:

END directive pseudocode is very important. END indicates to the assemblere the

end of the source [asm] file. The END directive is the last line of an 8051 program. In assembly language programming anything after the END directive is ignored by the assembler. Some assembler uses ".END" [notice the dot] instead of "END"

3.3 Labels in Assembly Language Programming and its Rules:

Programmer can make a program easier to read and maintain by choosing label names that are meaningful. There are several rules that names must follow. First, each label name must be unique.

The names used for labels in assembly language programming consists of alphabetic letters in both
upper and lower case, the digits 0 through 9, and the special characters question mark [?], period
[.], at [@], underline [_], and dollar sign [$]. The first character of the label must be an alphabetic
character, it cannot be a number. Every assembler has some reserved words which must not be used as labels in the program. These reserve words are the mnemonics for the instructions e.g. like
"MOV", "DJNZ", "MUL" and "ADD" are reserved as these are the instruction mnemonics.

3.4 DEBUGGER

The Crossware 8051 debugger downloads and runs your program on an 8051 target board.
It shares a common user interface with the simulator and many of the simulator and debugger features are identical.

The debugger can be run from the debug menu as shown on the left, by clicking on items in the
debugger tool bar or by using the keyboard accelerator keys.

The debugger requires a connection to the target board and this is provided either by a debug monitor which runs on the target board and communicates with the PC via a serial link or, for chips
such as those from Silicon Laboratories that have integrated debugging hardware, the manufacturers' debugger interface.

Take a look at some of the tutorials videos to see the debugger in action.

The debugger shares the following features with the simulator:

- Multiple memory views
- Multiple watch windows
- Multiple register views with register tooltips
- Disassembly view
- Complex heirarchical source level drag-and-drop breakpoints
- Machine level breakpoints
- Call stack view
- Variable tooltips
- Multiple application debugging

The following features are specific to the debugger:

- Downloading into memory on the target chip
- Hardware breakpoints in flash (Silicon Laboratories chips)
- Unlimited software execution breakpoints in RAM and flash memory (debug monitor)

QUESTIONS:

MULTIPLE CHOICE QUESTIONS:

1)which of following is directives of 8051:

(a) timer o (b) timer 1

(c) ORG (d)all

2) which of following is a part of debugger:
(a)tooltips (b) register
(c) intruppt (d) all of these above

SHORT ANSWER TYPES QUESTIONS:
1) What is directives?
2) EQU stands for ……..
3) What do you mean by debugger?
4) What is function of timer?
5) Write all types of assembler directives.

LONG ANSWER TYPES QUESTIONS:
1) Explain in detail assembler directives of 8051microprocessor.
2) Explain in detail debugger of 8051 microprocessor.

MICROCONTROLLERS
LEARNING OBJECTIVES:

- Concept of PIC microcontroller.

.
CHAPTER-5(INTRODUCTION OF PIC MICROCONTROLLER)
5.1. INTRODUCTION
Microcontrollers give you a fantastic way of creating projects. A PIC microcontroller is a processor
with built in memory and RAM and you can use it to control your projects (or build projects around it). So, it saves you building a circuit that has separate external RAM, ROM and peripheral
chips.
What this really means for you is that you have a very powerful device that has many useful built
in modules e.g.

- EEPROM.
- Timers.
- Analogue comparators.
- UART.

Even with just these four modules (note these are just example modules - there are more) you can
make up many projects e.g.:

- ► Frequency counter - using the internal timers and reporting through UART (RS232) or output to LCD.
- ► Capacitance meter - analogue comparator oscillator.
- ► Event timer - using internal timers.
- ► Event data logger -capturing analogue data using an internal ADC and using the internal
- ► EEPROM for storing data (using an external I2C for high data storage capacity.
- ► Servo controller (Control through UART) - using the internal PWM module or using a software created PWM.

The PIC Micro is one of the most popular microcontrollers and in case you were wondering the
difference between a microprocessor and a microcontroller is that a microcontroller has an internal
bus with in built memory and peripherals.
In fact the 8 pin (DIL) version of the 12F675 has an amazing number of internal peripherals. These
are:

- Two timers.
- One 10bit ADC with 4 selectable inputs.
- An internal oscillator (or you can use an external crystal).
- An analogue comparator.
- 1024 words of program memory.
- 64 Bytes of RAM.
- 128 Bytes of EEPROM memory.
- External interrupt (as well as interrupts from internal peripherals).
- External crystal can go up to 20MHz.
- ICSP : PIC standard programming interface.

And all of these work from within an 8 pin DIL package!
In the mid-range devices the memory space ranges from 1k to 8k (18F parts have more) - this does
not sound like a lot but the processor has an efficient instruction set and you can make useful projects even with 1k e.g. LM35 temperature sensing project that reports data to the serial port easily fits within 1k.
Features
In fact a PIC microcontroller is an amazingly powerful fully featured processor with internal RAM,
EEROM FLASH memory and peripherals. One of the smallest ones occupies the space of a 555
timer but has a 10bit ADC, 1k of memory, 2 timers, high current I/O ports a comparator a watch dog timer... I could go on as there is more!
Programming
One of the most useful features of a PIC microcontroller is that you can re-program them as they
use flash memory (if you choose a part with an F in the part number e.g. 12F675 not 12C509). You
can also use the ICSP serial interface built into each PIC Microcontroller for programming and even do programming while it's still plugged into the circuit!
You can either program a PIC microcontroller using assembler or a high level language and I recommend using a high level language such as C as it is much easier to use (after an initial learning curve). Once you have learned the high level language you are not forced to use the same
processor e.g. you could go to an AVR or Dallas microcontroller and still use the same high level
language.
Input / Output - I/O
A PIC Microcontroller can control outputs and react to inputs e.g. you could drive a relay or read
input buttons.
With the larger devices it's possible to drive LCDs or seven segment displays with very few control
lines as all the work is done inside the PIC Micro.

Comparing a frequency counter to discrete web designs you'll find two or three chips for the microcontroller design and ten or more for a discrete design. So using them saves prototype design
effort as you can use built in peripherals to take care of lots of the circuit operation.
Many now have a built in ADC so you can read analogue signal levels so you don't need to add an
external devices e.g. you can read an LM35 temperature sensordirectly with no interface logic.
Peripherals
The PIC microcontroller has many built in peripherals and this can make using them quite daunting
at first which is why I have made this introductory page with a summary of each major peripheral
block.
At the end is a short summary of the main devices used in projects shown on this site.
The best way to start is to learn about the main features of a chip and then begin to use each

peripheral in a project. I think learning by doing is the best way.
PIC
microcontroller
Feature
PIC microcontroller
feature description
Flash memory Re-programmable program storage.
RAM Memory storage for variables.
EEPROM
Long term stable memory : Electrically
Erasable Programmable Read
Only Memory.
I/O ports High current Input/Output ports (with pin
direction change).
Timers/Counters Typically 3.
USART Built in RS232 protocol (only needs level
translator chip).
CCP Capture/Compare/PWM module.
SSP I2C and SPI Interfaces.
Comparator An analogue comparator and internal

voltage reference.
ADC Analogue to digital converter.
PSP Parallel Slave Port (for 8 bit
microprocessor systems).
LCD LCD interface.
Special features ICSP,WDT,BOR,POR,PWRT,OST,SLEEP
ICSP Simple programming using In Circuit
Serial Programming.
Note:these are some of the main features
(some chips have all of these and some don't).
Flash memory
This is the program storage area and gives you the most important benefit for using a PIC microcontroller - You program the device many times. Since when does anyone get a program right first time ?
Devices used in projects on this site can be re-programmed up to 100,000 times (probably more) as
they use Flash memory - these have the letter F in the part name. You can get cheaper (OTP) devices but these are One-Time-Programmable; once programmed you can't program it again!
ICSP
In Circuit Serial Programming (ICSP) is the next most important benefit. Instead of transferring your chip from the programmer to the development board you just leave it in the board. By arranging the programming connections to your circuit correctly you won't need to remove the chip!
You can re-program the device while it's still in the circuit so once your programmer is setup you
can leave it on the bench and test your programs without moving the chip around and it makes the
whole process much easier.
I/O Ports

Input / Output ports let you communicate with the outside world so you can control leds, LCDs or
just about anything with the right interface. You can also set them as inputs to gather information.

Pin direction
Most PIC microcontroller pins can be set as an input or and output and this can be done on the fly
e.g. for a dallas 1 wire system a pin can be written to generate data and read at a later stage. The
TRIS register controls the I/O direction and setting a bit in this register to zero sets the pin
as output while setting it as one sets the pin as input.
This allows you to use a pin for multiple operations e.g. the Real Time clock project uses RA0, the
first pin of PORTA, to output data to a seven segment display and at a later point in the program
read the analogue value as an input.
Current
The PIC I/O ports are high current ports capable of directly driving LEDs (up to 25ma output
current) - the total current allowed usually ~200mA this is often for the whole chip (or specified for
several ports combined together).
Timer / Counters
Each PIC microcontroller has up to three timers that you can either use as a timer or a counter
(Timer 1 & 2) or a baud clock (Timer 2).
Timer 0
The original timer: Timer 0 was the first timer developed and you can find it in all the earliest
devices e.g. 16F84 up to the most current e,g, 16F877A.
It is an 8 bit timer with an 8 bit prescaler that can be driven from an internal (Fosc/4) or external
clock. It generates an interrupt on overflow when the count goes from 255 to zero.
Timer 0 always synchronizes the input clock (when using external clock).
Note: You can read and write timer 0 but you can not read the prescaler.
Note: The prescaler changes its effect depending on whether it is a timer prescaler or a watch dog
prescaler - so the same prescaler setting may prescale by 2 or by 1 depending on its use!
Timer 1
This is a 16 bit timer that generates an overflow interrupt when it goes from 65535 to zero. It has
an 8 bit programmable prescaler and you can drive it from the internal clock (Fosc/4) or an
external pin.
To eliminate false triggering it also has an optional input synchronizer for external pin input.
This timer can be used in sleep mode and will generate a wakeup interrupt on overflow.
Timer 1 is also read by the CCP module to capture an event time.
Note: Using this timer in sleep mode will use more current.

In addition it can be used to drive a low power watch crystal. This is something that sounds good
but I don't recommend you do it as watch crystals are extremely difficult to drive correctly. You
should only use it if you are going to make a pcb and follow all the guidelines in making it noise

free. I used a DS1307 in the Real Time clock project which drives the crystal directly but even this
is difficult to get operating accurately.

Timer 2

This is an 8 bit timer with an 8 bit prescaler and an 8 bit postscaler. It takes its input only from the
internal oscillator (Fosc/4).

This timer is used for the timebase of a PWM when PWM is active and it can be software selected
by the SSP module as a baud clock.

It also has a period register that allows easy control of the period. When timer 2 reaches the PR2
register value then it resets. This saves having to check the timer value in software and then reset
the timer and since it is done in hardware the operation is much faster - so you can generate fast
clocks with periods that are multiples of the main clock.

USART

The USART is a useful module and saves having to code up a software version so it saves valuable
program memory. You can find more information on RS232here and how to make it
work. Look here for pin outs.

All you need to interface it to a PC serial port is a MAX232 chip (or equivalent).

Note: An equivalent MAX232 chip is the SP202ECP that has the same pinout as the MAX232 but
lets you use 100nF capacitors - so you don't need the large 1uF caps.

Baud Rates

You have to be careful using the baud rates as they depend on the main clock in use and normal oscillator values in general do not fit very well with 'real' baud rates.

There is a table of baud rates in microchip data sheet DS33023A which indicates the expected percentage error for a specific clock rate and in general the higher the main clock the lower the error.

You sometimes have to play around with the register settings to get a better fit with your clock rate
and the baud rate you want. An example is for an 8MHz clock - if you use BRGH=1 and an 8MHz
clock (see the 16F88 datasheet) you get accurate baud rates up to 38.4kbaud. You have to force this to work e.g. in mikroC the built in USART routines use BRGH=0 so at 8MHz the baud rate is
only accurate to 9.6kbaud.

If you want a super-accurate baud rate the best way is to use a clock crystal that ends up giving you
that baud rate i.e. work back through the baud rate equations to find the crystal you need.

CCP

The Capture/Compare/PWM module has three modes of operation:

- Capture - Capture the time of an event.
- Compare - Generate an output when Timer 1 reaches a value.
- PWM - Pulse Width Modulation.

Capture

Capture mode is used to capture the value of Timer 1 when a signal at the CCP pin goes high (or
low depending on how the CCP is set up). The CCP can accurately capture the arrival time of a
signal at the CCP pin so it can be used for pulse time measurement.

Compare

Compare mode is used to generate an output when Timer 1 reaches a value you put into CCPR1. One special event trigger mode lets you start the ADC when the compare mode triggers.

PWM

PWM gives you one Pulse Width Modulation output with 10 bit resolution and with no software
overhead - once started it operates all by itself unless you want to change the duty cycle.
It uses Timer 2 to define its operation using Timer 2 period register to define the frequency of the
PWM.
Note: The duty cycle is not a percentage it is the number of periods of the PWM clock that the output is high!

SSP

The Synchronous Serial Port lets you communicate with devices that use either the SPI (Serial Peripheral Interface) or I2C (Inter IC communication) protocols. Note that for full Master mode I2C operation you need to choose a PIC device that has the MSSP device (Master Synchronous Serial Port).
SPI and I2C are shared so you can only use one at a time (or you could use the I2C bit banged routines in the Real Time Clock project to have both at the same time).
You can find a project that uses I2C here and you can find more information on I2C here.

Comparator and comparator voltage reference

The comparator is module that has two analogue comparators which can be set up in one of 8 different ways. Either digital or analogue inputs can be compared to reference voltages.
In one mode an internally generated voltage reference is used as an input to both comparators and
in the same mode multiplexing lets you monitor up to four different input pins.
You can even send the output of the comparator to a pin so that it is used independently from the
microcontroller e.g. in a circuit where you need a comparator you don't need an extra chip!
The analogue level must be between Vdd and Vss as protection diodes won't allow anything else.
The module will generate an interrupt if the comparator output changes.
You can use it in sleep mode and the interrupt will wake it up.
The source impedance of the analogue signal must be smaller than 10k.

ADC

The single 10 bit Analogue to Digital Converter can have up to 8 inputs for a device multiplexed
from input pins.
The ADC can be used during sleep but you have to use the RC clock mode. One benefit of this is
that there will be no digital switching noise so you will get better conversion accuracy.
For the 16F877A you can not just choose to use an analogue input if you feel the need as there are

only a specific and limited number of ways that the analogue input pins can be enabled. It is best
to start with AN0 and add more as necessary - see the datasheet for which analogue inputs can be
enabled e.g. if you started a design using only AN5 you would find that you may have to enable a
few more analogue inputs as well!
The 16F675 can measure 4 analogue input pins!
PSP
The Parallel Slave Port lets you to connect the PIC microcontroller directly into a microprocessor
system. It provides an 8 bit read/write data bus and RD (read) WR (write) and CS (chip select) inputs - all active low.
This will let you add a PIC microcontroller to a system so that the PIC microcontroller can be treated as a memory mapped peripheral. It will let the microcontroller behave just as though it was
another microprocessor building block e.g. some memory or ram but in this case you have full control over exactly what the building block is i.e. you can re-program the PIC microcontroller to
do just about anything.
This provides an easy route to adding a PIC microcontroller to an 8 bit system that already exists.

LCD
The LCD interface lets you directly interface to an LCD saving you having to use an LCD module
such as the HD44780. I have not used this feature as it is another commercial requirement where
removing a chip (HD44780) saves money in a production run. I think it is capable of driving a graphic LCD.
Special Features
ICSP In Circuit Serial Programming
click here (jumps to ICSP section).
WDT Watch dog timer This is a software error protector.
BOR Brown Out reset
This detects if the power supply dips slightly and resets the device if so.
POR Power on reset
This starts microcontroller initialization.
PWRT PoWeR up Time A time delay to let Vdd rise.
OST Oscillator start up timer
Wait for 1024 cycles after PWRT.
SLEEP

PIC
microcontroller
sleepmode
Enter low power mode.
WDT
If your software goes haywire then this timer resets the processor. To stop the reset the well behaved software must periodically issue the CLRWDT instruction to stop a resert. The WDT runs using its own oscillator. It runs during sleep and shares Timer 0 prescaler.
POR
Power On Reset starts PIC microcontroller initialization when it detects a rising edge on MCLR.
PWRT
If you enable this then 72ms after a POR the PIC microcontroller is started.

OST
Oscillator Startup Timer delays for 1024 oscillator cycles after PWRT (if PWRT is enabled) ensuring that the oscillator has started and is stable. It is automatic and only used for crystal oscillator modes and is active after POR or wake from sleep.
SLEEP
Sleep mode (or low power consumption mode) is entered by executing the 'SLEEP' command. The
device can wake from sleep caused by an external reset, Watch Dog Timer timeout, INT pin RB
port change or peripheral interrupt.
Project device overview
This site mainly uses three PIC devices out of the hundreds of different chips that microchip produces. This does not sound like a lot but you can use the devices in almost any project and they
have so many built in peripherals that you can make hundreds of projects with them.
The other microchip devices are all useful in different situations - perhaps they have more memory
or different peripherals - this is useful if you want to tailor your designs to the system you build -
but probably more useful in a commercial environment where every cent counts in a production run.
All three devices are extremely powerful and the main difference is that they have different numbers of pins and memory size.
Note: There are differences in using the devices i.e. there are some registers that are different but in
the generally you can interchange them - this is made easier using a high level language.
The devices used in this site are:
PIC
microcontroller
Device
PIC
microcontroller
No. Pins
PIC
microcontroller
Flash memory

WORDS
12F675 8 1k
16F88 18 4k
16F877A 40 8k
Note : When looking at the microchip site the memory size is kwords - ignore kbytes - you need
the kword size as this is what each instruction occupies - the kbyte size is for comparison to other
P a g e | 11
types of micros (probably). But the microcontroller data bus is 8 bits wide so it is an 8 bit microcontroller (different program memory and data memory due to using Harvard architecture).
(Note: that all of them have the letter F in - this means it is a Flash re-programmable part - don't go
and buy a part with O in as its OTP - programmable only once! - only do that if you are really really sure it's the final design).
PIC Microcontroller Flash Memory size
You may think that 1k or even 8k is so tiny that it won't be useful but each PIC microcontroller uses RISC (Reduced Instruction Set Computing) which simply means that it has a cleverly arranged instruction set that only has a few instructions. The mid range parts have 35 instructions.
If you use the high level language as recommended in this site then you won't need to be too aware
of the instruction set it just means you can do a lot with a small amount of memory. Most of the
projects on this site although they are fully working projects fit within 2k words!
Note: If you need more memory you can always move to the 18F series of PIC microcontrollers. Another option is to add an I2C serial eprom.
PIC microcontroller RAM and EEPROM size
The PIC microcontroller RAM size is also important as it stores all your variables and intermediate
data.
Note: You can usually alter the program to use less RAM by choosing the right variable sizes or
changing how your program works
For example don't use floating point alter it to use a different variable type e.g. you can use long
integers with fixed point operation to avoid floating point.
PIC microcontroller EEROM : Electrically Erasable ROM is used to store data that must be saved between power up and power down.
This area is readable and writable and has a much longer life than the main program store i.e. it has
been designed for more frequent use.
QUESTIONS:
MULTIPLE CHOICE QUESTIONS:
1)which of following is PIC microcontroller:
P a g e | 12
(a) XTAL 1 (b) RST 5.5
(c) 16F88 (d)RST 7.5
2) which of following is a memory:

(a)ALU (b) register
(c) EPROM (d) all of these above

SHORT ANSWER TYPES QUESTIONS:

1) What is ADC?
2) WDT stands for ……..
3) What do you mean by bus?
4) What is function of PSP?
5) Write all types of memory.

LONG ANSWER TYPES QUESTIONS:

1) Explain in detail PIC microcontroller.