ARDUINO BASED AUTOMATIC ENGINE LOCKING SYSTEM FOR DRUNKEN DRIVERS

INSTITUTE OF AERONAUTICAL ENGINEERING

ARDUINO BASED AUTOMATIC ENGINE LOCKING SYSTEM FOR DRUNKEN DRIVERS

Submitted by

SIPURUSETTY SRAVANKUMAR

20955a0241

INDEX

CHAPTER 1
INTRODUCTION

1.1 Introduction of Embedded System

An Embedded System is a combination of computer hardware and software, and perhaps additional mechanical or other parts, designed to perform a specific function. A good example is the microwave oven. Almost every household has one, and tens of millions of them are used everyday, but very few people realize that a processor and software are involved in the preparation of their lunch or dinner.

This is in direct contrast to the personal computer in the family room. It too is comprised of computer hardware and software and mechanical components (disk drives, for example). However, a personal computer is not designed to perform a specific function rather; it is able to do many different things. Many people use the term general-purpose computer to make this distinction clear. As shipped, a general-purpose computer is a blank slate; the manufacturer does not know what the customer will do wish it. One customer may use it for a network file server another may use it exclusively for playing games, and a third may use it to write the next great American novel.

Frequently, an embedded system is a component within some larger system. For example, modern cars and trucks contain many embedded systems. One embedded system controls the anti-lock brakes, other monitors and controls the vehicle's emissions, and a third displays information on the dashboard. In some cases, these embedded systems are connected by some sort of a communication network, but that is certainly not a requirement.

At the possible risk of confusing you, it is important to point out that a general-purpose computer is itself made up of numerous embedded systems. For example, my computer consists of a keyboard, mouse, video card, modem, hard drive, floppy drive, and sound card-each of which is an embedded system. Each of these devices contains a processor and software and is designed to perform a specific function. For example, the modem is designed to send and receive digital data over analog telephone line. That's it and all of the other devices can be summarized in a single sentence as well.

If an embedded system is designed well, the existence of the processor and software could be completely unnoticed by the user of the device. Such is the case for a microwave oven, VCR, or alarm clock. In some cases, it would even be possible to build an equivalent device that does not contain the processor and software. This could be done by replacing the combination with a custom integrated circuit that performs the same functions in hardware. However, a lot of flexibility is lost when a design is hard-cooled in this way. It is mush easier, and cheaper, to change a few lines of software than to redesign a piece of custom hardware.

1.1.2 Real Time Systems:

One subclass of embedded is worthy of an introduction at this point. As commonly defined, a real-time system is a computer system that has timing constraints. In other words, a real-time system is partly specified in terms of its ability to make certain calculations or decisions in a timely manner. These important calculations are said to have deadlines for completion. And, for all practical purposes, a missed deadline is just as bad as a wrong answer.

The issue of what if a deadline is missed is a crucial one. For example, if the real-time system is part of an airplane's flight control system, it is possible for the lives of the passengers and crew to be endangered by a single missed deadline. However, if instead the system is involved in satellite communication, the damage could be limited to a single corrupt data packet. The more severe the consequences, the more likely it will be said that the deadline is "hard" and thus, the system is a hard real-time system. Real-time systems at the other end of this discussion are said to have "soft" deadlines.

All of the topics and examples presented in this book are applicable to the designers of real-time system who is more delight in his work. He must guarantee reliable operation of the software and hardware under all the possible conditions and to the degree that human lives depend upon three system's proper execution, engineering calculations and descriptive paperwork.

1.1.3 Application Areas

Nearly 99 per cent of the processors manufactured end up in embedded systems. The embedded system market is one of the highest growth areas as these systems are used in very market segment- consumer electronics, office automation, industrial automation, biomedical engineering, wireless communication, data communication, telecommunications, transportation, military and so on.

1.1.4 Consumer appliances:

At home we use a number of embedded systems which include digital camera, digital diary, DVD player, electronic toys, microwave oven, remote controls for TV and air-conditioner, VCO player, video game consoles, video recorders etc. Today' s high-tech car has about 20 embedded systems for transmission control, engine spark control, air-conditioning, navigation etc. Even wristwatches are now becoming embedded systems. The palmtops are powerful embedded systems using which we can carry out many general-purpose tasks such as playing games and word processing.

1.1.5 Office automation:

The office automation products using em embedded systems are copying machine, fax machine, key telephone, modem, printer, scanner etc.

1.1.6 Industrial automation:

Today a lot of industries use embedded systems for process control. These include pharmaceutical, cement, sugar, oil exploration, nuclear energy, electricity generation and transmission. The embedded systems for industrial use are designed to carry out specific tasks such as monitoring the temperature, pressure, humidity, voltage, current etc., and then take appropriate action based on the monitored levels to control other devices or to send information to a centralized monitoring station. In hazardous industrial environment, where human presence has to be avoided, robots are used, which are programmed to do specific jobs. The robots are now becoming very powerful and carry out many interesting and complicated tasks such as hardware assembly.

1.1.7 Medical electronics:

Almost every medical equipment in the hospital is an embedded system. These equipments include diagnostic aids such as ECG, EEG, blood pressure measuring devices, X-ray scanners; equipment used in blood analysis, radiation, colonscopy, endoscopy etc. Developments in medical electronics have paved way for more accurate diagnosis of diseases.

1.1.8 Computer networking:

Computer networking products such as bridges, routers, Integrated Services Digital Networks (ISDN), Asynchronous Transfer Mode (ATM), X.25 and frame relay switches are embedded systems which implement the necessary data communication protocols. For example, a router interconnects two networks. The two networks may be running different protocol stacks. The router' s function is to obtain the data packets from

incoming pores, analyze the packets and send them towards the destination after doing necessary protocol conversion. Most networking equipments, other than the end systems (desktop computers) we use to access the networks, are embedded systems

1.1.9 Telecommunications:

In the field of telecommunications, the embedded systems can be categorized as subscriber terminals and network equipment. The subscriber terminals such as key telephones, ISDN phones, terminal adapters, web cameras are embedded systems. The network equipment includes multiplexers, multiple access systems, Packet Assemblers Dissemblers (PADs), sate11ite modems etc. IP phone, IP gateway, IP gatekeeper etc. are the latest embedded systems that provide very low-cost voice communication over the Internet.

1.1.10 Wireless technologies:

Advances in mobile communications are paving way for many interesting applications using embedded systems. The mobile phone is one of the marvels of the last decade of the 20' h century. It is a very powerful embedded system that provides voice communication while we are on the move. The Personal Digital Assistants and the palmtops can now be used to access multimedia services over the Internet. Mobile communication infrastructure such as base station controllers, mobile switching centers are also powerful embedded systems.

1.1.11 Security:

Security of persons and information has always been a major issue. We need to protect our homes and offices; and also the information we transmit and store. Developing embedded systems for security applications is one of the most lucrative businesses nowadays. Security devices at homes, offices, airports etc. for authentication and verification are embedded systems. Encryption devices are nearly 99 per cent of the processors that are manufactured end up in~ embedded systems. Embedded systems find applications in . every industrial segment- consumer electronics, transportation, avionics, biomedical engineering, manufacturing, process control and industrial automation, data communication, telecommunication, defense, security etc. Used to encrypt the data/voice being transmitted on communication links such as telephone lines. Biometric systems using fingerprint and face recognition are now being extensively used for user authentication in banking applications as well as for access control in high security buildings.

1.1.13 Finance:

Financial dealing through cash and cheques are now slowly paving way for transactions using smart cards and ATM (Automatic Teller Machine, also expanded as Any Time Money) machines. Smart card, of the size of a credit card, has a small micro-controller and memory; and it interacts with the smart card reader! ATM machine and acts as an electronic wallet. Smart card technology has the capability of ushering in a cashless society. Well, the list goes on. It is no exaggeration to say that eyes wherever you go, you can see, or at least feel, the work of an embedded system!

1.1.14 What are microcontrollers and what are they used for?

Like all good things, this powerful component is basically very simple. It is made by mixing tested and high- quality "ingredients" (components) as per following receipt:

1. The simplest computer processor is used as the "brain" of the future system.
2. Depending on the taste of the manufacturer, a bit of memory, a few A/D converters, timers, input/output lines etc. are added
3. All that is placed in some of the standard packages.
4. A simple software able to control it all and which everyone can easily learn about has been developed.

On the basis of these rules, numerous types of microcontrollers were designed and they quickly became man's invisible companion. Their incredible simplicity and flexibility conquered us a long time ago and if you try to invent something about them, you should know that you are probably late, someone before you has either done it or at least has tried to do it. The following things have had a crucial influence on development and success of the microcontrollers:

- Powerful and carefully chosen electronics embedded in the microcontrollers can independetly or via input/output devices (switches, push buttons, sensors, LCD displays, relays etc.), control various processes and devices such as industrial automation, electric current, temperature, engine performance etc.
- Very low prices enable them to be embedded in such devices in which, until recent time it was not worthwhile to embed anything. Thanks to that, the world is overwhelmed today with cheap automatic devices and various “ smart” appliences.
- Prior knowledge is hardly needed for programming. It is sufficient to have a PC (software in use is not demanding at all and is easy to learn) and a simple device

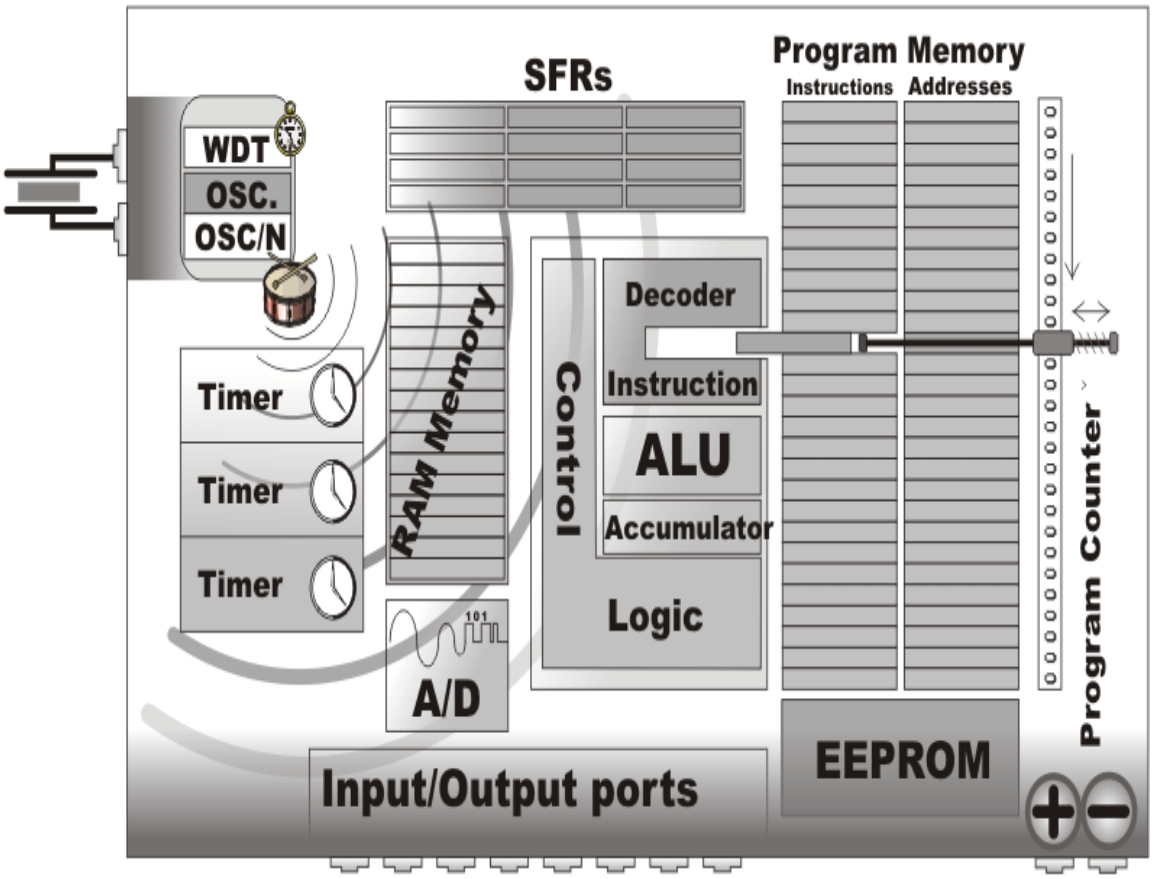
(called the programmer) used for " loading" raedy-to-use programs into the microcontroller.

So, if you are infected with a virus called electronics, there is nothing left for you to do but to learn how to use and control its power.

1.1.15 How does the microcontroller operate?

Even though there is a large number of different types of microcontrollers and even more programs created for their use only, all of them have many things in common. Thus, if you learn to handle one of them you will be able to handle them all. A typical scenario on the basis of which it all functions is as follows:

1. Power supply is turned off and everything is still…the program is loaded into the microcontroller, nothing indicates what is about to come…
2. Power supply is turned on and everything starts to happen at high speed! The control logic unit keeps everything under control. It disables all other circuits except quartz crystal to operate. While the preparations are in progress, the first milliseconds go by.
3. Power supply voltage reaches its maximum and oscillator frequency becomes stable. SFRs are being filled with bits reflecting the state of all circuits within the microcontroller. All pins are configured as inputs. The overall electronis starts operation in rhythm with pulse sequence. From now on the time is measured in micro and nanoseconds.
4. Program Counter is set to zero. Instruction from that address is sent to instruction decoder which recognizes it, after which it is executed with immediate effect.
5. The value of the Program Counter is incremented by 1 and the whole process is repeated...several million times per second.



1.1.16 What is what in the microcontroller?

As you can see, all the operations within the microcontroller are performed at high speed and quite simply, but the microcontroller itself would not be so useful if there are not special circuits which make it complete. In continuation, we are going to call your attention to them.

1.1.17 Read Only Memory (ROM)

Read Only Memory (ROM) is a type of memory used to permanently save the program being executed. The size of the program that can be written depends on the size of this memory. ROM can be built in the microcontroller or added as an external chip, which depends on the type of the microcontroller. Both options have some disadvantages. If ROM is added as an external chip, the microcontroller is cheaper and the program can be considerably longer. At the same time, a number of available pins is reduced as the microcontroller uses its own input/output ports for connection to the chip. The internal ROM is usually smaller and more expensive, but leaves more pins available for connecting to peripheral environment. The size of ROM ranges from 512B to 64KB.

Random Access Memory (RAM)

Random Access Memory (RAM) is a type of memory used for temporary storing data and intermediate results created and used during the operation of the microcontrollers. The content of this memory is cleared once the power supply is off. For example, if the program performes an addition, it is necessary to have a register standing for what in everyday life is called the " sum" . For that purpose, one of the registers in RAM is called the "sum" and used for storing results of addition. The size of RAM goes up to a few KBs.

1.1.18 Electrically Erasable Programmable ROM (EEPROM)

The EEPROM is a special type of memory not contained in all microcontrollers. Its contents may be changed during program execution (similar to RAM), but remains permanently saved even after the loss of power (similar to ROM). It is often used to store values, created and used during operation (such as calibration values, codes, values to count up to etc.), which must be saved after turning the power supply off. A disadvantage of this memory is that the process of programming is relatively slow. It is measured in miliseconds.

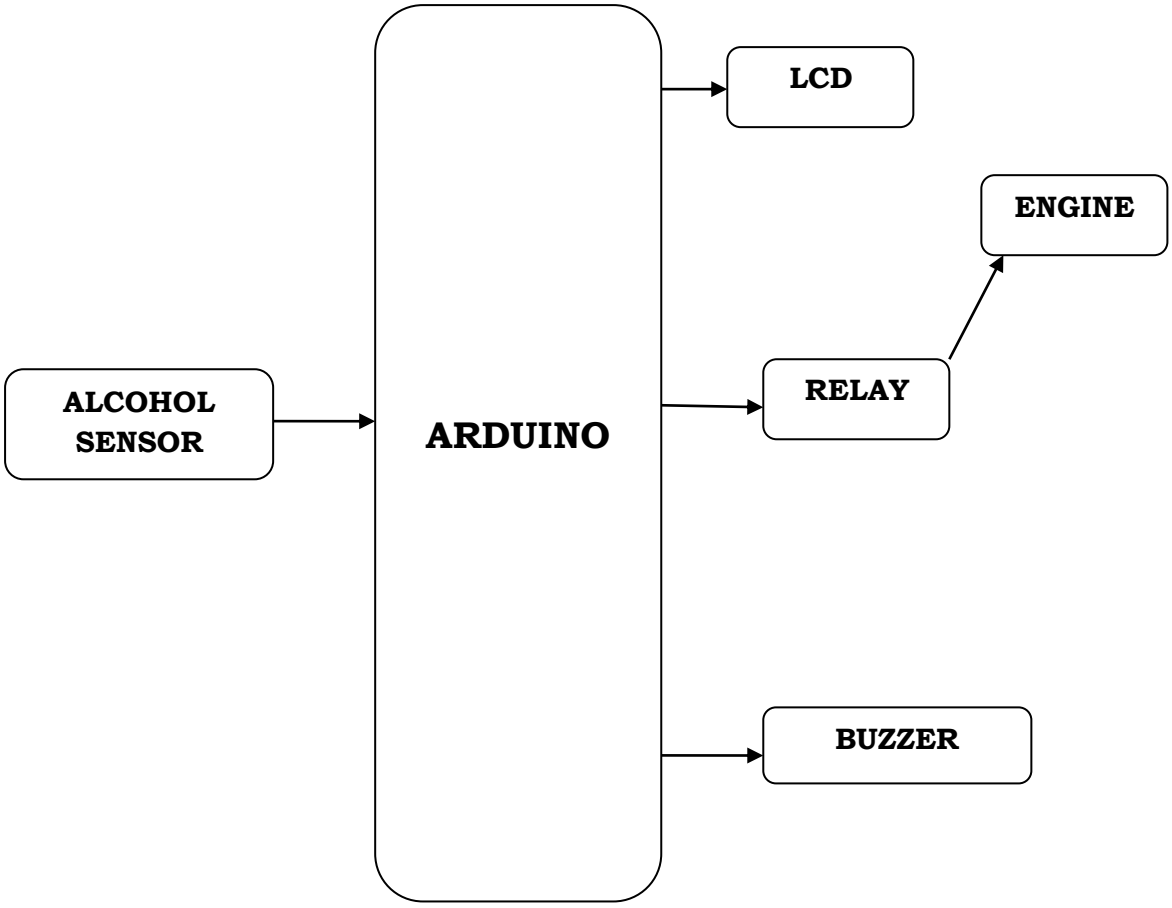
1.1.19 Special Function Registers (SFR)

Special function registers are part of RAM memory. Their purpose is predefined by the manufacturer and cannot be changed therefore. Since their bits are physically connected to particular circuits within the microcontroller, such as A/D converter, serial communication module etc., any change of their state directly affects the operation of the microcontroller or some of the circuits. For example, writing zero or one to the SFR controlling an input/output port causes the appropriate port pin to be configured as input or output. In other words, each bit of this register controls the function of one single pin.

1.1.20 Program Counter

Program Counter is an engine running the program and points to the memory address containing the next instruction to execute. After each instruction execution, the value of the counter is incremented by 1. For this reason, the program executes only one instruction at a time just as it is written. However…the value of the program counter can be changed at any moment, which causes a " jump" to a new memory location. This is how subroutines and branch instructions are executed. After jumping, the counter resumes even and monotonous automatic counting +1, +1, +1…

1.2 BLOCK DIAGRAM



1.3 Objective of the Project

The main purpose of this project is " Drunk driving detection" . Nowadays, many accidents are happening because of the alcohol consumption of the driver or the person who is driving the vehicle. Thus Drunk driving is a major reason for accidents in almost all countries all over the world. Alcohol Detector in Car project is designed for the safety of the people seating inside the car. Alcohol Detection with Vehicle Controlling project helps to control the vehicle in case the driver has consumed the alcohol. An alcohol breath analyzer project should be fitted/ installed inside the vehicle.

We here propose an alcohol sensing system that measures alcohol intake, displays percentage of alcohol and also sounds an alarm if it is above a particular threshold. Here we use an alcohol sensor circuit along with lcd display and a buzzer alarm. Our system first uses the alcohol sensor in order to detect alcohol. The sensor provides analog output. This analog output is now provided to the microcontroller for further processing. Based on the input the microcontroller calculates the percentage of alcohol and displays the same on an LCD display. It also sounds an alarm if the amount of alcohol exceeds a particular amount. Our system thus allows measuring amount of alcohol and then displaying percentage of alcohol measured. Also a alarm is sounded that indicates that measured alcohol is above a particular percentage.

Here' s what you' ll need to know about the alcohol gas sensor:

- Manufacturer' s model MQ-3
- Very high alcohol sensitivity
- Technology offering long life and stability
- Simple drive circuit
- Fast response time and high reaction sensitivity

Technical specifications of the alcohol gas sensor

Below are more details about the alcohol gas sensor:

- Voltage required: 5 V DC
- Current required: ~160 mA
- Interface type: Resistive
- Dimensions: 19.1 mm diameter × 16.55 mm high (excluding electrodes)
- Operating temperature range: -10 to +50 °C

CHAPTER 2

DESCRIPTION OF HARDWARE COMPONENTS

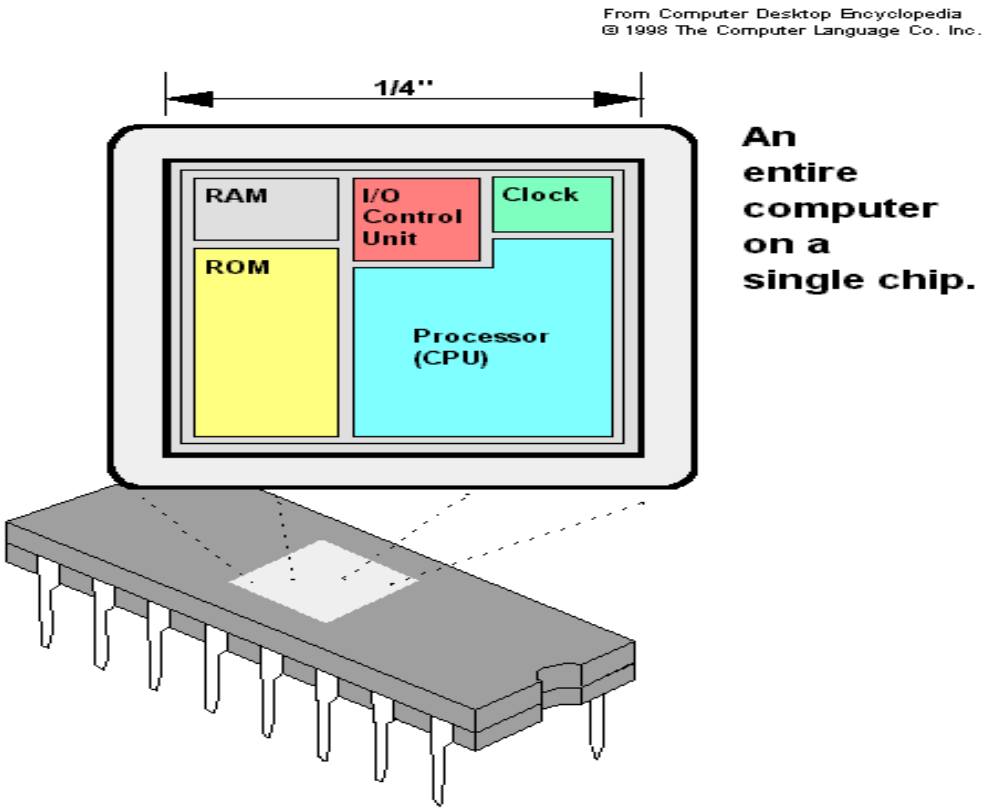
2.1 AT89S52

2.2.1 A BRIEF HISTORY OF 8051

In 1981, Intel Corporation introduced an 8 bit microcontroller called 8051. This microcontroller had 128 bytes of RAM, 4K bytes of chip ROM, two timers, one serial port, and four ports all on a single chip. At the time it was also referred as " A SYSTEM ON A CHIP"

AT89S52:

The AT89S52 is a low-power, high-performance CMOS 8-bit microcontroller with 8K bytes of in-system programmable Flash memory. The device is manufactured using Atmel' s high-density nonvolatile memory technology and is compatible with the industry-standard 80C51 instruction set and pinout. The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional nonvolatile memory pro-grammer. By combining a versatile 8-bit CPU with in-system programmable Flash on a monolithic chip, the Atmel AT89S52 is a powerful microcontroller, which provides a highly flexible and cost-effective solution to many, embedded control applications. The AT89S52 provides the following standard features: 8K bytes of Flash, 256 bytes of RAM, 32 I/O lines, Watchdog timer, two data pointers, three 16-bit timer/counters, a six-vector two-level interrupt architecture, a full duplex serial port, on-chip oscillator, and clock circuitry. In addition, the AT89S52 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port, and interrupt system to continue functioning. The Power-down mode saves the RAM con-tents but freezes the oscillator, disabling all other chip functions until the next interrupt



- 8031 has 128 bytes of RAM, two timers and 6 interrupts.
- 8051 has 4K ROM, 128 bytes of RAM, two timers and 6 interrupts.
- 8052 has 8K ROM, 256 bytes of RAM, three timers and 8 interrupts.
- Of the three microcontrollers, 8051 is the most preferable. Microcontroller supports both serial and parallel communication.

In the concerned project 8052 microcontroller is used. Here microcontroller used is AT89S52, which is manufactured by ATMEL laboratories. The 8051 is the name of a big family of microcontrollers. The device which we are going to use along this tutorial is the *'AT89S52'* which is a typical 8051 microcontroller manufactured by Atmel™. Note that this part doesn't aim to explain the functioning of the different components of AT89S52 microcontroller, but rather to give you a general idea of the organization of the chip and the available features, which shall be explained in detail along this tutorial.

The block diagram provided by Atmel™ in their datasheet showing the architecture the 89S52 device can seem very complicated, and since we are going to use the C high level language to program it, a simpler architecture can be represented as the figure 1.2.A.

This figures shows the main features and components that the designer can interact with. You can notice that the 89S52 has 4 different ports, each one having 8 Input/output lines providing a total of 32 I/O lines. Those ports can be used to output DATA and orders do other devices, or to read the state of a sensor, or a switch. Most of the ports of the 89S52 have 'dual function' meaning that they can be used for two different functions: the fist one is to perform input/output operations and the second one is used to implement special features of the microcontroller like counting external pulses, interrupting the execution of the program according to external events, performing serial data transfer or connecting the chip to a computer to update the software.

NECESSITY OF MICROCONTROLLERS:

Microprocessors brought the concept of programmable devices and made many applications of intelligent equipment. Most applications, which do not need large amount of data and program memory, tended to be costly.

The microprocessor system had to satisfy the data and program requirements so, sufficient RAM and ROM are used to satisfy most applications .The peripheral control equipment also had to be satisfied. Therefore, almost all-peripheral chips were used in the design. Because of these additional peripherals cost will be comparatively high.

An example:

8085 chip needs:

An Address latch for separating address from multiplex address and data.32-KB RAM and 32-KB ROM to be able to satisfy most applications. As also Timer / Counter, Parallel programmable port, Serial port, and Interrupt controller are needed for its efficient applications.

In comparison a typical Micro controller 8051 chip has all that the 8051 board has except a reduced memory as follows.

4K bytes of ROM as compared to 32-KB, 128 Bytes of RAM as compared to 32-KB.

Bulky:

On comparing a board full of chips (Microprocessors) with one chip with all components in it (Microcontroller).

Debugging:

Lots of Microprocessor circuitry and program to debug. In Micro controller there is no Microprocessor circuitry to debug.

Slower Development time: As we have observed Microprocessors need a lot of debugging at board level and at program level, where as, Micro controller do not have the excessive circuitry and the built-in peripheral chips are easier to program for operation.

So peripheral devices like Timer/Counter, Parallel programmable port, Serial Communication Port, Interrupt controller and so on, which were most often used were integrated with the Microprocessor to present the Micro controller .RAM and ROM also were integrated in the same chip. The ROM size was anything from 256 bytes to 32Kb or more. RAM was optimized to minimum of 64 bytes to 256 bytes or more.

Microprocessor has following instructions to perform:

1. Reading instructions or data from program memory ROM.
2. Interpreting the instruction and executing it.
3. Microprocessor Program is a collection of instructions stored in a Nonvolatile memory.
4. Read Data from I/O device
5. Process the input read, as per the instructions read in program memory.
6. Read or write data to Data memory.
7. Write data to I/O device and output the result of processing to O/P device.

Introduction to AT89S52

The system requirements and control specifications clearly rule out the use of 16, 32 or 64 bit micro controllers or microprocessors. Systems using these may be earlier to implement due to large number of internal features. They are also faster and more reliable but, the above application is satisfactorily served by 8-bit micro controller. Using an inexpensive 8-bit Microcontroller will doom the 32-bit product failure in any competitive market place. Coming to the question of why to use 89S52 of all the 8-bit Microcontroller available in the market the main answer would be because it has 8kB Flash and 256 bytes of data RAM32 I/O lines, three 16-bit timer/counters, a Eight-vector two-level interrupt architecture, a full duplex serial port, on-chip oscillator, and clock circuitry.

In addition, the AT89S52 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode

stops the CPU while allowing the RAM, timer/counters, serial port, and interrupt system to continue functioning. The Power Down Mode saves the RAM contents but freezes the oscillator, disabling all other chip functions until the next hardware reset. The Flash program memory supports both parallel programming and in Serial In-System Programming (ISP). The 89S52 is also In-Application Programmable (IAP), allowing the Flash program memory to be reconfigured even while the application is running.

By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89S52 is a powerful microcomputer which provides a highly flexible and cost effective solution to many embedded control applications.

FEATURES

- Compatible with MCS-51 Products
- 8K Bytes of In-System Reprogrammable Flash Memory
- Fully Static Operation: 0 Hz to 33 MHz
- Three-level Program Memory Lock
- 256 x 8-bit Internal RAM
- 32 Programmable I/O Lines
- Three 16-bit Timer/Counters
- Eight Interrupt Sources
- Programmable Serial Channel
- Low-power Idle and Power-down Modes
- 4.0V to 5.5V Operating Range
- Full Duplex UART Serial Channel
- Interrupt Recovery from Power-down Mode
- Watchdog Timer
- Dual Data Pointer
- Power-off Flag
- Fast Programming Time
- Flexible ISP Programming (Byte and Page Mode)

PIN DIAGRAM

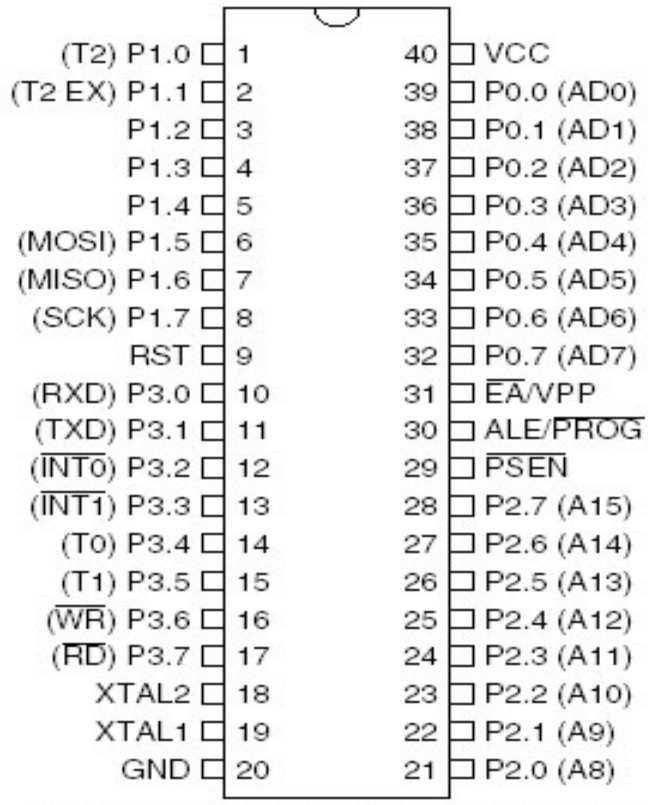


FIG-2 PIN DIAGRAM OF 89S52 IC

2.1.4 PIN DESCRIPTION

Pin Description

VCC

Supply voltage.

GND

Ground.

Port 0

Port 0 is an 8-bit open drain bidirectional I/O port. As an output port, each pin can sink eight TTL inputs. When 1s are written to port 0 pins, the pins can be used as highimpedance inputs.Port 0 can also be configured to be the multiplexed loworder address/data bus during accesses to external program and data memory. In this mode, P0

has internal pullups. Port 0 also receives the code bytes during Flash programming and outputs the code bytes during program verification.

External pullups are required during program verification.

Port 1

Port 1 is an 8-bit bidirectional I/O port with internal pullups. The Port 1 output buffers can sink/source four TTL inputs. When 1s are written to Port 1 pins, they are pulled high by the internal pullups and can be used as inputs. As inputs,Port 1 pins that are externally being pulled low will source current (IIL) because of the internal pullups. In addition, P1.0 and P1.1 can be configured to be the timer/counter 2 external count input (P1.0/T2) and the timer/counter 2 trigger input (P1.1/T2EX), respectively, as shown in the following table. Port 1 also receives the low-order address bytes during Flash programming and verification.

Port Pin	Alternate Functions
P1.0	T2 (external count input to Timer/Counter 2), clock-out
P1.1	T2EX (Timer/Counter 2 capture/reload trigger and direction control)
P1.5	MOSI (used for In-System Programming)
P1.6	MISO (used for In-System Programming)
P1.7	SCK (used for In-System Programming)

Port 2

Port 2 is an 8-bit bidirectional I/O port with internal pullups. The Port 2 output buffers can sink/source four TTL inputs. When 1s are written to Port 2 pins, they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (IIL) because of the internal pullups. Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @ DPTR). In this application, Port 2 uses strong internal pull-ups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOVX @ RI), Port 2 emits the contents of the P2 Special Function Register. Port 2 also receives the high-order address bits and some control signals during Flash programming and verification.

Port 3

Port 3 is an 8-bit bidirectional I/O port with internal pullups. The Port 3 output buffers can sink/source four TTL inputs. When 1s are written to Port 3 pins, they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (IIL) because of the pullups. Port 3 also serves the functions of various special features of the AT89S52, as shown in the following table. Port 3 also receives some control signals for Flash programming and verification.

Port Pin	Alternate Functions
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{\text{INT0}}$ (external interrupt 0)
P3.3	$\overline{\text{INT1}}$ (external interrupt 1)
P3.4	T0 (timer 0 external input)
P3.5	T1 (timer 1 external input)
P3.6	$\overline{\text{WR}}$ (external data memory write strobe)
P3.7	$\overline{\text{RD}}$ (external data memory read strobe)

RST

Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device. This pin drives High for 96 oscillator periods after the Watchdog times out. The DISRTO bit in SFR AUXR (address 8EH) can be used to disable this feature. In the default state of bit DISRTO, the RESET HIGH out feature is enabled. **ALE/PROG** Address Latch Enable (ALE) is an output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input (PROG) during Flash programming. In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external data memory. If desired, ALE operation can be disabled by setting bit 0 of SFR location 8EH. With the bit set, ALE is active only during a MOVX or MOVC instruction. Otherwise, the pin is weakly pulled high. Setting the ALE-disable bit has no effect if the microcontroller is in external execution mode.

PSEN

Program Store Enable (PSEN) is the read strobe to external program memory. When the AT89S52 is executing code from external program memory, PSEN is activated twice each machine cycle, except that two PSEN activations are skipped during each access to external data memory.

EA/VPP

External Access Enable. EA must be strapped to GND in order to enable the device to fetch code from external program memory locations starting at 0000H up to FFFFH. Note, however, that if lock bit 1 is programmed, EA will be internally latched on reset. EA should be strapped to VCC for internal program executions. This pin also receives the 12-volt programming enable voltage
(VPP) during Flash programming.

XTAL1

Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

XTAL2

Output from the inverting oscillator amplifier.

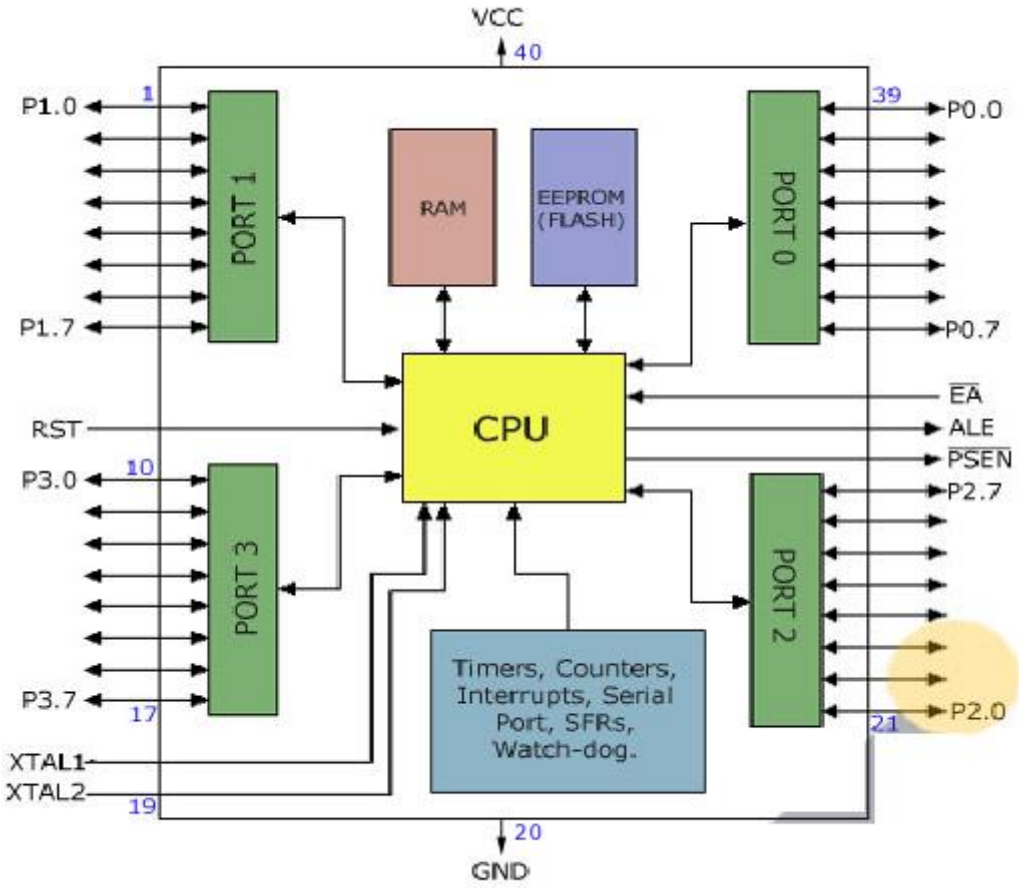


FIG-3 Functional block diagram of micro controller

The 8052 Oscillator and Clock:

The heart of the 8051 circuitry that generates the clock pulses by which all the internal all internal operations are synchronized. Pins XTAL1 And XTAL2 is provided for connecting a resonant network to form an oscillator. Typically a quartz crystal and capacitors are employed. The crystal frequency is the basic internal clock frequency of the microcontroller. The manufacturers make 8051 designs that run at specific minimum and maximum frequencies typically 1 to 16 MHz.

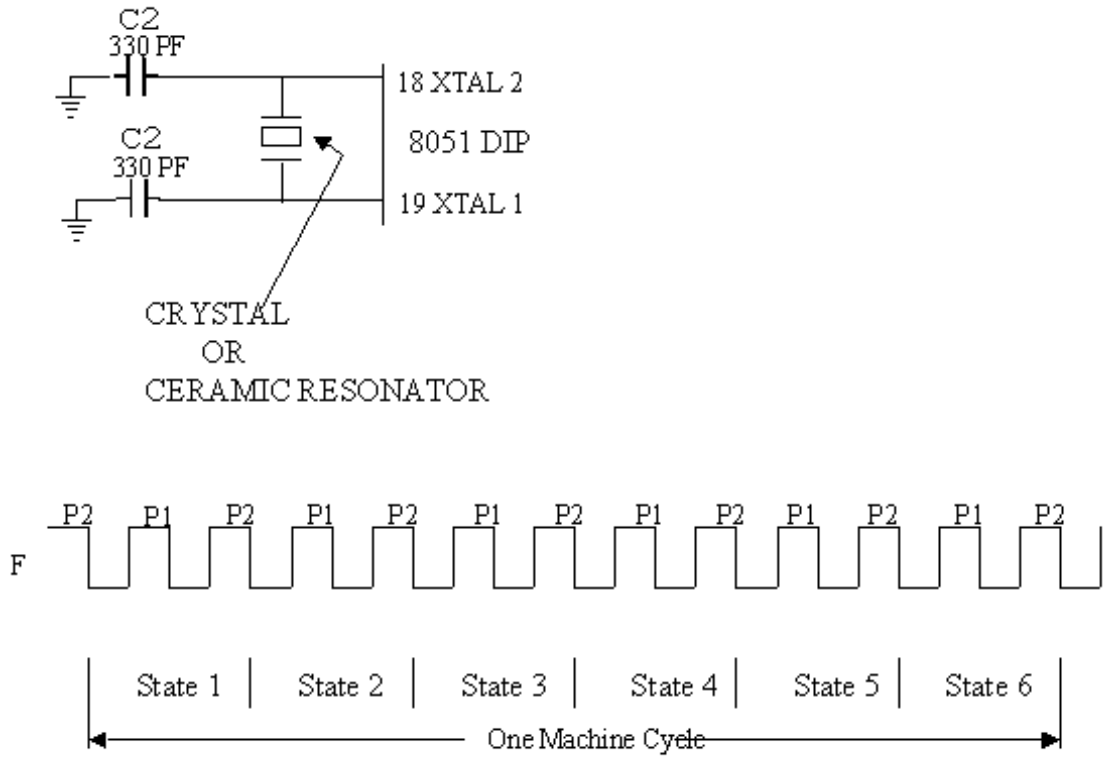


Fig-4 Oscillator and timing circuit

MEMORIES

Types of memory:

The 8052 have three general types of memory. They are on-chip memory, external Code memory and external Ram. On-Chip memory refers to physically existing memory on the micro controller itself. External code memory is the code memory that resides off chip. This

is often in the form of an external EPROM. External RAM is the Ram that resides off chip. This often is in the form of standard static RAM or flash RAM.

a) Code memory

Code memory is the memory that holds the actual 8052 programs that is to be run. This memory is limited to 64K. Code memory may be found on-chip or off-chip. It is possible to have 8K of code memory on-chip and 60K off chip memory simultaneously. If only off-chip memory is available then there can be 64K of off chip ROM. This is controlled by pin provided as EA

b) Internal RAM

The 8052 have a bank of 256 bytes of internal RAM. The internal RAM is found on-chip. So it is the fastest Ram available. And also it is most flexible in terms of reading and writing. Internal Ram is volatile, so when 8051 is reset, this memory is cleared. 256 bytes of internal memory are subdivided. The first 32 bytes are divided into 4 register banks. Each bank contains 8 registers. Internal RAM also contains 256 bits, which are addressed from 20h to 2Fh. These bits are bit addressed i.e. each individual bit of a byte can be addressed by the user. They are numbered 00h to FFh. The user may make use of these variables with commands such as SETB and CLR.

Special Function registered memory:

Special function registers are the areas of memory that control specific functionality of the 8052 micro controller.

a) Accumulator (0E0h)

As its name suggests, it is used to accumulate the results of large no of instructions. It can hold 8 bit values.

b) B registers (0F0h)

The B register is very similar to accumulator. It may hold 8-bit value. The b register is only used by MUL AB and DIV AB instructions. In MUL AB the higher byte of the product gets stored in B register. In div AB the quotient gets stored in B with the remainder in A.

c) Stack pointer (81h)

The stack pointer holds 8-bit value. This is used to indicate where the next value to be removed from the stack should be taken from. When a value is to be pushed onto the stack, the 8052 first store the value of SP and then store the value at the resulting memory location. When a value is to be popped from the stack, the 8052 returns the value from the memory location indicated by SP and then decrements the value of SP.

d) Data pointer

The SFRs DPL and DPH work together work together to represent a 16-bit value called the data pointer. The data pointer is used in operations regarding external RAM and some instructions code memory. It is a 16-bit SFR and also an addressable SFR.

e) Program counter

The program counter is a 16 bit register, which contains the 2 byte address, which tells the 8052 where the next instruction to execute to be found in memory. When the 8052 is initialized PC starts at 0000h. And is incremented each time an instruction is executes. It is not addressable SFR.

f) PCON (power control, 87h)

The power control SFR is used to control the 8051' s power control modes. Certain operation modes of the 8051 allow the 8051 to go into a type of " sleep mode" which consumes much lee power.

SMOD	--	--	--	GF1	GF0	PD	IDL

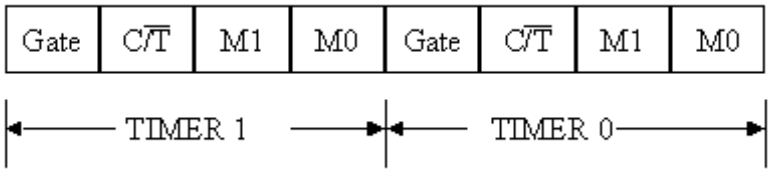
g) TCON (timer control, 88h)

The timer control SFR is used to configure and modify the way in which the 8051' s two timers operate. This SFR controls whether each of the two timers is running or stopped and contains a flag to indicate that each timer has overflowed. Additionally, some non-timer related bits are located in TCON SFR. These bits are used to configure the way in which the external interrupt flags are activated, which are set when an external interrupt occurs.

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

h) TMOD (Timer Mode, 89h)

The timer mode SFR is used to configure the mode of operation of each of the two timers. Using this SFR your program may configure each timer to be a 16-bit timer, or 13 bit timer, 8-bit auto reload timer, or two separate timers. Additionally you may configure the timers to only count when an external pin is activated or to count " events" that are indicated on an external pin.



i) TO (Timer 0 low/high, address 8A/8C h)

These two SFRs taken together represent timer 0. Their exact behavior depends on how the timer is configured in the TMOD SFR; however, these timers always count up. What is configurable is how and when they increment in value.

j) T1 (Timer 1 Low/High, address 8B/ 8D h)

These two SFRs, taken together, represent timer 1. Their exact behavior depends on how the timer is configured in the TMOD SFR; however, these timers always count up.

k) P0 (Port 0, address 90h, bit addressable)

This is port 0 latch. Each bit of this SFR corresponds to one of the pins on a micro controller. Any data to be outputted to port 0 is first written on P0 register. For e.g., bit 0 of port 0 is pin P0.0, bit 7 is pin p0.7. Writing a value of 1 to a bit of this SFR will send a high level on the corresponding I/O pin whereas a value of 0 will bring it to low level.

l) P1 (port 1, address 90h, bit addressable)

This is port latch1. Each bit of this SFR corresponds to one of the pins on a micro controller. Any data to be outputted to port 0 is first written on P0 register. For e.g., bit 0 of port 0 is pin P1.0, bit 7 is pin P1.7. Writing a value of 1 to a bit of this SFR will send a high level on the corresponding I/O pin whereas a value of 0 will bring it to low level

m) P2 (port 2, address 0A0h, bit addressable):

This is a port latch2. Each bit of this SFR corresponds to one of the pins on a micro controller. Any data to be outputted to port 0 is first written on P0 register. For e.g., bit 0 of port 0 is pin P2.0, bit 7 is pin P2.7. Writing a value of 1 to a bit of this SFR will send a high level on the corresponding I/O pin whereas a value of 0 will bring it to low level.

n) P3 (port 3, address B0h, bit addressable) :

This is a port latch3. Each bit of this SFR corresponds to one of the pins on a micro controller. Any data to be outputted to port 0 is first written on P0 register. For e.g., bit 0 of port 0 is pin P3.0, bit 7 is pin P3.7. Writing a value of 1 to a bit of this SFR will send a high level on the corresponding I/O pin whereas a value of 0 will bring it to low level.

o) IE (interrupt enable, 0A8h):

The Interrupt Enable SFR is used to enable and disable specific interrupts. The low 7 bits of the SFR are used to enable/disable the specific interrupts, where the MSB bit is used to enable or disable all the interrupts. Thus, if the high bit of IE is 0 all interrupts are disabled regardless of whether an individual interrupt is enabled by setting a lower bit.

EA	--	ET2	ES	ET1	EX1	ET0	EX0

p) IP (Interrupt Priority, 0B8h)

The interrupt priority SFR is used to specify the relative priority of each interrupt. On 8051, an interrupt maybe either low or high priority. An interrupt may interrupt interrupts. For e.g., if we configure all interrupts as low priority other than serial interrupt. The serial interrupt always interrupts the system, even if another interrupt is currently executing. However, if a serial interrupt is executing no other interrupt will be able to interrupt the serial interrupt routine since the serial interrupt routine has the highest priority.

--	--	PT2	PS	PT1	PX1	PT0	PX0

q) PSW (Program Status Word, 0D0h)

The program Status Word is used to store a number of important bits that are set and cleared by 8052 instructions. The PSW SFR contains the carry flag, the auxiliary carry flag, the parity flag and the overflow flag. Additionally, it also contains the register bank select flags, which are used to select, which of the " R" register banks currently in use.

CY	AC	F0	RS1	RS0	OV	--	P

r) SBUF (Serial Buffer, 99h)

SBUF is used to hold data in serial communication. It is physically two registers. One is writing only and is used to hold data to be transmitted out of 8052 via TXD. The other is read only and holds received data from external sources via RXD. Both mutually exclusive registers use address 99h.

2.2 POWER SUPPLY

All digital circuits require regulated power supply. In this article we are going to learn how to get a regulated positive supply from the mains supply.

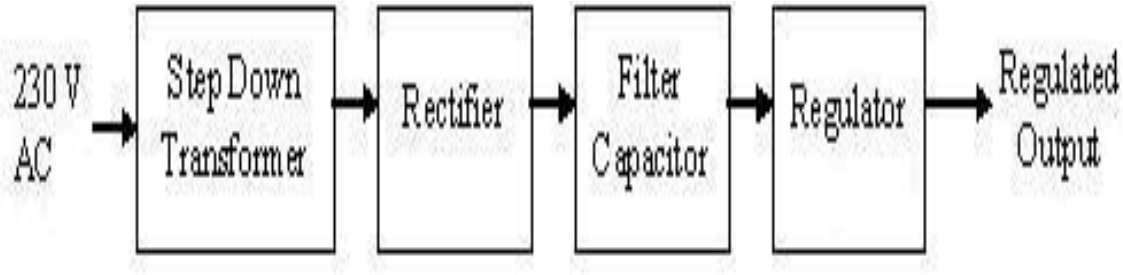
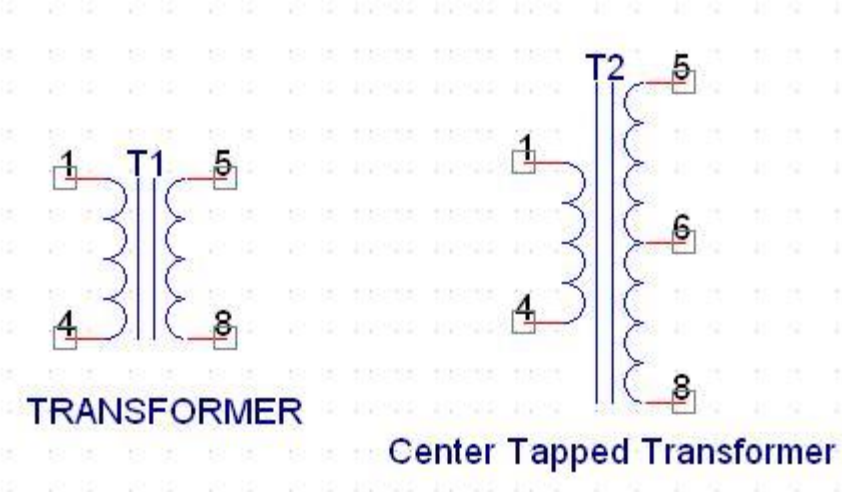


Figure 1 shows the basic block diagram of a fixed regulated power supply. Let us go through each block.

TRANSFORMER



A transformer consists of two coils also called as " WINDINGS" namely PRIMARY & SECONDARY. They are linked together through inductively coupled electrical conductors also called as CORE. A changing current in the primary causes a change in the Magnetic Field in the core & this in turn induces an alternating voltage in the secondary coil. If load is applied to the secondary then an alternating current will flow through the load. If we consider an ideal condition then all the energy from the primary circuit will be transferred to the secondary circuit through the magnetic field.

$$P_{primary} = P_{secondary}$$

So

$$I_p V_p = I_s V_s$$

The secondary voltage of the transformer depends on the number of turns in the Primary as well as in the secondary.

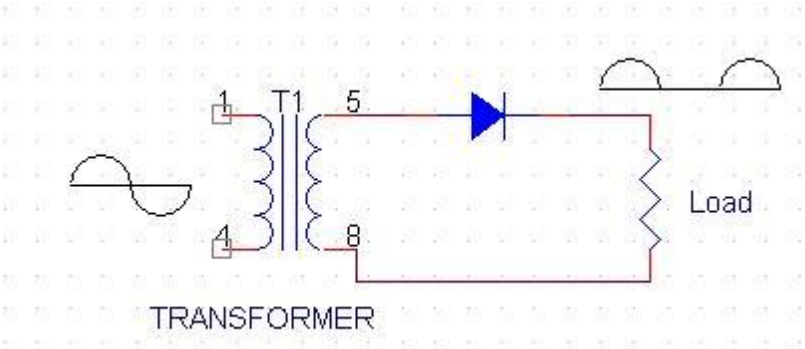
$$\frac{V_s}{V_p} = \frac{N_s}{N_p}$$

Rectifier

A rectifier is a device that converts an AC signal into DC signal. For rectification purpose we use a diode, a diode is a device that allows current to pass only in one direction i.e. when the anode of the diode is positive with respect to the cathode also called as forward biased condition & blocks current in the reversed biased condition.

Rectifier can be classified as follows:

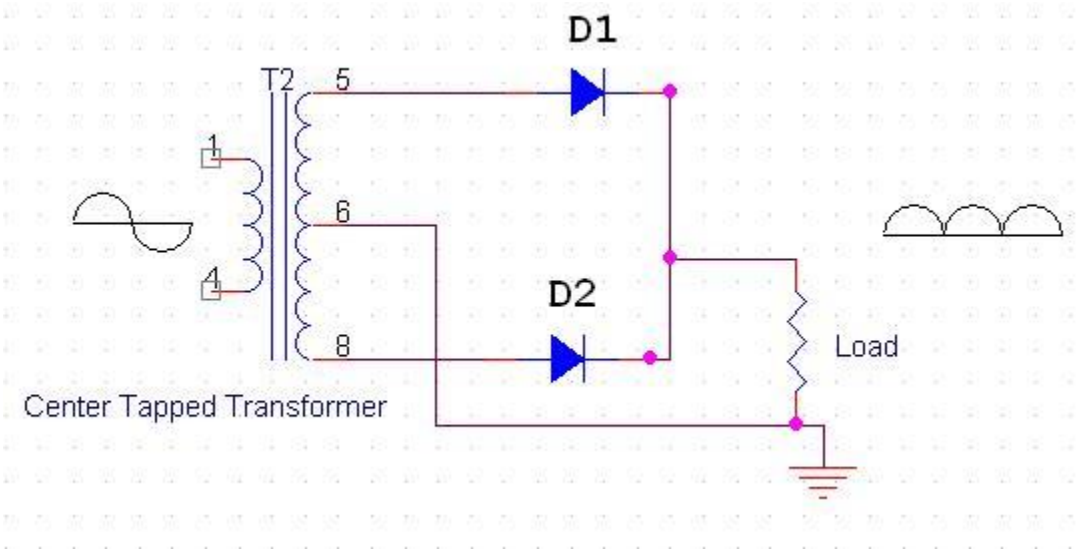
1) Half Wave rectifier.



This is the simplest type of rectifier as you can see in the diagram a half wave rectifier consists of only one diode. When an AC signal is applied to it during the positive half cycle

the diode is forward biased & current flows through it. But during the negative half cycle diode is reverse biased & no current flows through it. Since only one half of the input reaches the output, it is very inefficient to be used in power supplies.

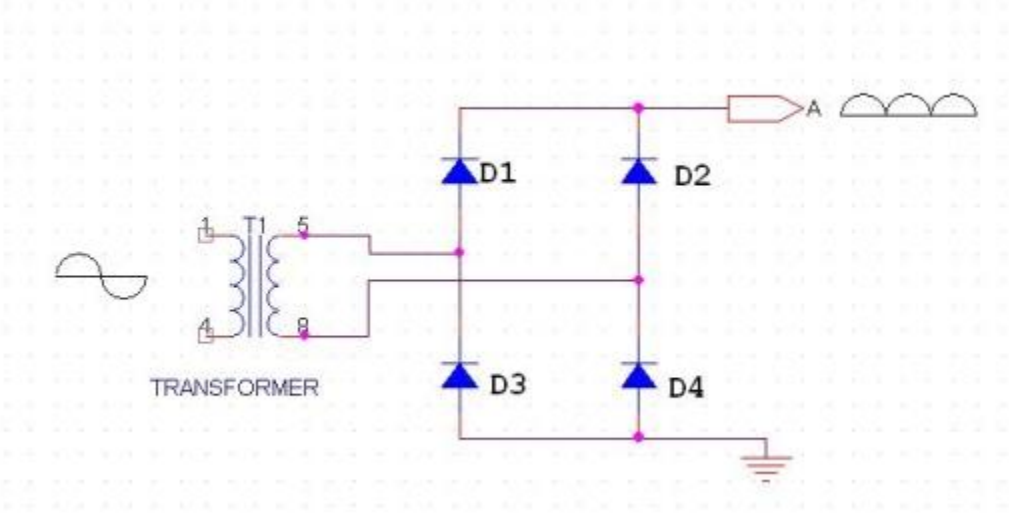
2) Full wave rectifier.



Half wave rectifier is quite simple but it is very inefficient, for greater efficiency we would like to use both the half cycles of the AC signal. This can be achieved by using a center tapped transformer i.e. we would have to double the size of secondary winding & provide connection to the center. So during the positive half cycle diode D1 conducts & D2 is in reverse biased condition. During the negative half cycle diode D2 conducts & D1 is reverse biased. Thus we get both the half cycles across the load.

One of the disadvantages of Full Wave Rectifier design is the necessity of using a center tapped transformer, thus increasing the size & cost of the circuit. This can be avoided by using the Full Wave Bridge Rectifier.

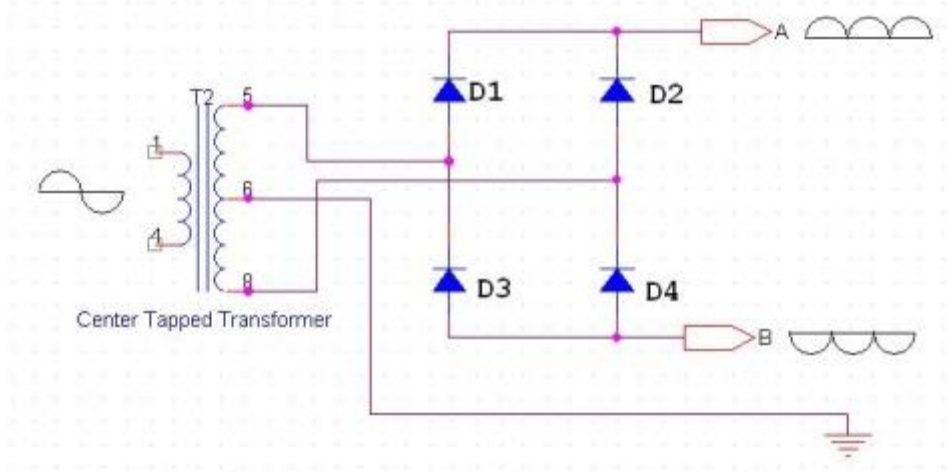
3) Bridge



Rectifier.

As the name suggests it converts the full wave i.e. both the positive & the negative half cycle into DC thus it is much more efficient than Half Wave Rectifier & that too without using a center tapped transformer thus much more cost effective than Full Wave Rectifier.

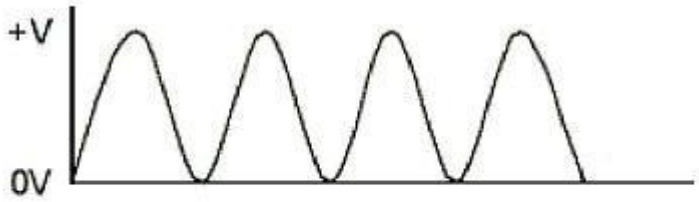
Full Bridge Wave Rectifier consists of four diodes namely D1, D2, D3 and D4. During the positive half cycle diodes D1 & D4 conduct whereas in the negative half cycle diodes D2 & D3 conduct thus the diodes keep switching the transformer connections so we get positive half cycles in the output.



If we use a center tapped transformer for a bridge rectifier we can get both positive & negative half cycles which can thus be used for generating fixed positive & fixed negative voltages.

Filter Capacitor

Even though half wave & full wave rectifier give DC output, none of them provides a constant output voltage. For this we require to smoothen the waveform received from the rectifier. This can be done by using a capacitor at the output of the rectifier this capacitor is also called as " FILTER CAPACITOR" or " SMOOTHING CAPACITOR" or " RESERVOIR CAPACITOR" . Even after using this capacitor a small amount of ripple will remain.We place the Filter Capacitor at the output of the rectifier the capacitor will charge to the peak voltage during each half cycle then will discharge its stored energy slowly through the load while the rectified voltage drops to zero, thus trying to keep the voltage as constant as possible.



If we go on increasing the value of the filter capacitor then the Ripple will decrease. But then the costing will increase. The value of the Filter capacitor depends on the current consumed by the circuit, the frequency of the waveform & the accepted ripple.

$$C = \frac{V_r F}{I}$$

Where,

Vr= accepted ripple voltage.(should not be more than 10% of the voltage)

I= current consumed by the circuit in Amperes.

F= frequency of the waveform. A half wave rectifier has only one peak in one cycle so F=25hz Whereas A Full Wave Rectifier Has Two Peaks In One Cycle So F=100hz.

VOLTAGE REGULATOR

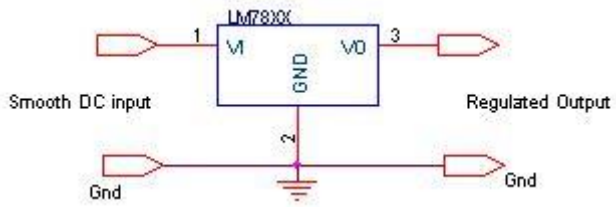
A Voltage regulator is a device which converts varying input voltage into a constant regulated output voltage. Voltage regulator can be of two types

1) Linear Voltage Regulator

Also called as Resistive Voltage regulator because they dissipate the excessive voltage resistively as heat.

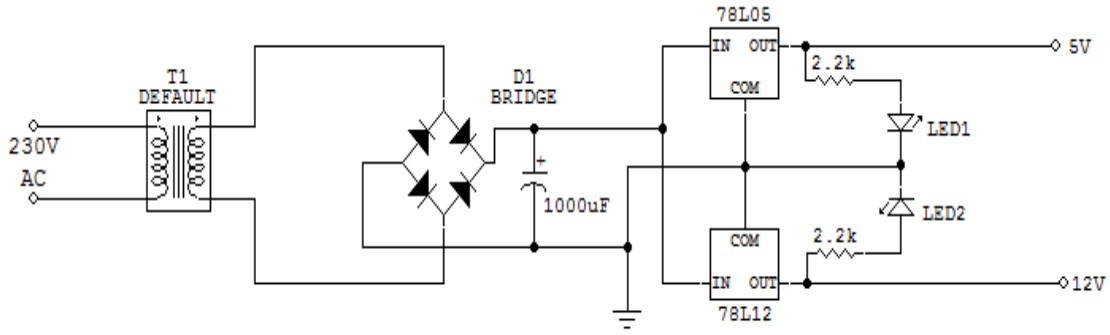
2) Switching Regulators.

They regulate the output voltage by switching the Current ON/OFF very rapidly. Since their output is either ON or OFF it dissipates very low power thus achieving higher efficiency as compared to linear voltage regulators. But they are more complex & generate high noise due to their switching action. For low level of output power switching regulators tend to be costly but for higher output wattage they are much cheaper than linear regulators. The most commonly available Linear Positive Voltage Regulators are the 78XX series where the XX indicates the output voltage. And 79XX series is for Negative Voltage Regulators.



After filtering the rectifier output the signal is given to a voltage regulator. The maximum input voltage that can be applied at the input is 35V.Normally there is a 2-3 Volts drop across the regulator so the input voltage should be at least 2-3 Volts higher than the output voltage. If the input voltage gets below the Vmin of the regulator due to the ripple voltage or due to any other reason the voltage regulator will not be able to produce the correct regulated voltage.

(i) Circuit diagram:



Circuit Diagram of power supply

(ii) IC 7805:

7805 is an integrated three-terminal positive fixed linear voltage regulator. It supports an input voltage of 10 volts to 35 volts and output voltage of 5 volts. It has a current rating of 1 amp although lower current models are available. Its output voltage is fixed at 5.0V. The 7805 also has a built-in current limiter as a safety feature. 7805 is manufactured by many companies, including National Semiconductors and Fairchild Semiconductors.

The 7805 will automatically reduce output current if it gets too hot.The last two digits represent the voltage; for instance, the 7812 is a 12-volt regulator. The 78xx series of regulators is designed to work in complement with the 79xx series of negative voltage regulators in systems that provide both positive and negative regulated voltages, since the 78xx series can't regulate negative voltages in such a system.

The 7805 & 78 is one of the most common and well-known of the 78xx series regulators, as it's small component count and medium-power regulated 5V make it useful for powering TTL devices.

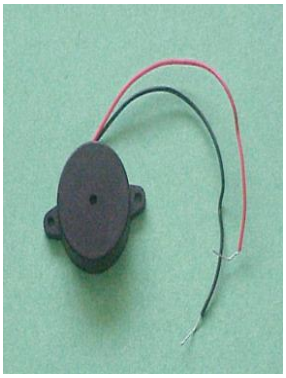
Table 2.1. Specifications of IC7805

SPECIFICATIONS	IC 7805
V_{out}	5V

V_{ein} - V_{out} Difference	5V - 20V
Operation Ambient Temp	0 - 125°C
Output I_{max}	1A

2.3 BUZZER

A **buzzer** or **beeper** is an audio signaling device, which may be mechanical, electromechanical, or electronic. Typical uses of buzzers and beepers include alarms, timers and confirmation of user input such as a mouse click or keystroke.



FEATURES

- The PB series are high-performance buzzers with a unimorph piezoelectric ceramic element and an integral self-excitation oscillator circuit.
- They exhibit extremely low power consumption in comparison to electromagnetic units.
- They are constructed without switching contacts to ensure long life and no electrical noise.
- Compact, yet produces high acoustic output with minimal voltage.

Mechanical

A joy buzzer is an example of a purely mechanical buzzer.

Electromechanical

Early devices were based on an electromechanical system identical to an electric bell without the metal gong. Similarly, a relay may be connected to interrupt its own actuating current, causing the contacts to buzz. Often these units were anchored to a wall or ceiling to use it as a sounding board. The word "buzzer" comes from the rasping noise that electromechanical buzzers made.

VOLTAGE BUZZER SOUND CONTROLS

When resistance is connected in series (as shown in illustrations (a) and (b)), abnormal oscillation may occur when adjusting the sound volume. In this case, insert a capacitor in parallel to the voltage oscillation board (as shown in illustration (c)). By doing so, abnormal oscillation can be prevented by grounding one side. However, the voltage VB added to the voltage oscillation board must be within the maximum input voltage range, and as capacitance of 3.3μF or greater should be connected.

Electronic

A piezoelectric element may be driven by an oscillating electronic circuit or other audio signal source. Sounds commonly used to indicate that a button has been pressed are a click, a ring or a beep. Electronic buzzers find many applications in modern days.

Uses

- Annunciator panels
- Electronic metronomes
- Game shows
- Microwave ovens and other household appliances
- Sporting events such as basketball games

2.4 DC MOTOR

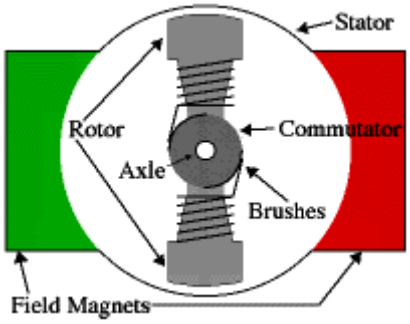
DC motors are configured in many types and sizes, including brush less, servo, and gear motor types. A motor consists of a rotor and a permanent magnetic field stator. The magnetic field is maintained using either permanent magnets or electromagnetic windings. DC motors are most commonly used in variable speed and torque.

Motion and controls cover a wide range of components that in some way are used to generate and/or control motion. Areas within this category include bearings and bushings, clutches and brakes, controls and drives, drive components, encoders and resolves, Integrated motion control, limit switches, linear actuators, linear and rotary motion components, linear position sensing, motors (both AC and DC motors), orientation position sensing, pneumatics and pneumatic components, positioning stages, slides and guides, power transmission (mechanical), seals, slip rings, solenoids, springs.

Motors are the devices that provide the actual speed and torque in a drive system. This family includes AC motor types (single and multiphase motors, universal, servo motors, induction, synchronous, and gear motor) and DC motors (brush less, servo motor, and gear motor) as well as linear, stepper and air motors, and motor contactors and starters.

In any electric motor, operation is based on simple electromagnetism. A current-carrying conductor generates a magnetic field; when this is then placed in an external magnetic field, it will experience a force proportional to the current in the conductor, and to the strength of the external magnetic field. As you are well aware of from playing with magnets as a kid, opposite (North and South) polarities attract, while like polarities (North and North, South and South) repel. The internal configuration of a DC motor is designed to harness the magnetic interaction between a current-carrying conductor and an external magnetic field to generate rotational motion.

Let's start by looking at a simple 2-pole DC electric motor (here red represents a magnet or winding with a "North" polarization, while green represents a magnet or winding with a "South" polarization).

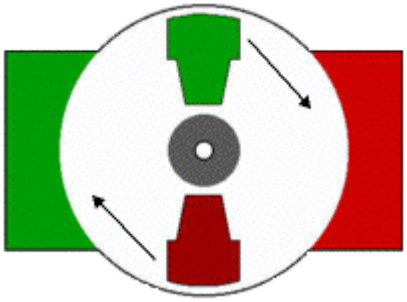


Every DC motor has six basic parts -- axle, rotor (a.k.a., armature), stator, commutator, field magnet(s), and brushes. In most common DC motors (and all that Beamers will see), the external magnetic field is produced by high-strength permanent magnets1. The stator is the stationary part of the motor -- this includes the motor casing, as well as two or more permanent magnet pole pieces. The rotor (together with the axle and attached commutator) rotates with respect to the stator. The rotor consists of windings (generally on a core), the windings being electrically connected to the commutator. The above diagram shows a common motor layout -- with the rotor inside the stator (field) magnets.

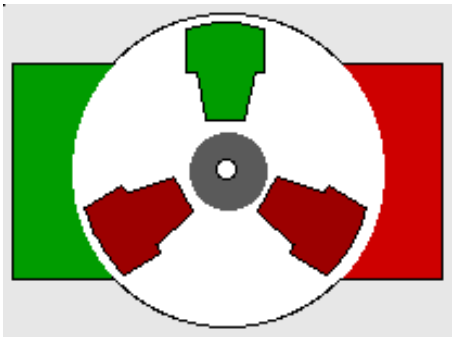
The geometry of the brushes, commutator contacts, and rotor windings are such that when power is applied, the polarities of the energized winding and the stator magnet(s) are misaligned, and the rotor will rotate until it is almost aligned with the stator's field magnets. As the rotor reaches alignment, the brushes move to the next commutator contacts,

and energize the next winding. Given our example two-pole motor, the rotation reverses the direction of current through the rotor winding, leading to a "flip" of the rotor's magnetic field, and driving it to continue rotating.

In real life, though, DC motors will always have more than two poles (three is a very common number). In particular, this avoids "dead spots" in the commutator. You can imagine how with our example two-pole motor, if the rotor is exactly at the middle of its rotation (perfectly aligned with the field magnets), it will get "stuck" there. Meanwhile, with a two-pole motor, there is a moment where the commutator shorts out the power supply (i.e., both brushes touch both commutator contacts simultaneously). This would be bad for the power supply, waste energy, and damage motor components as well. Yet another disadvantage of such a simple motor is that it would exhibit a high amount of torque" ripple" (the amount of torque it could produce is cyclic with the position of the rotor).

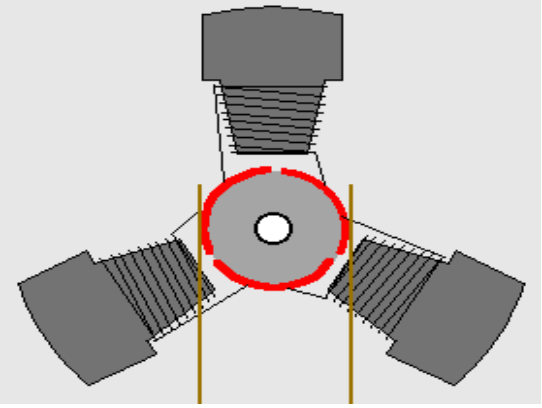


So since most small DC motors are of a three-pole design, let's tinker with the workings of one via an interactive animation (JavaScript required):



You'll notice a few things from this -- namely, one pole is fully energized at a time (but two others are "partially" energized). As each brush transitions from one commutator contact to the next, one coil's field will rapidly collapse, as the next coil's field will rapidly

charge up (this occurs within a few microsecond). We'll see more about the effects of this later, but in the meantime you can see that this is a direct result of the coil windings' series wiring:



There's probably no better way to see how an average dc motor is put together, than by just opening one up. Unfortunately this is tedious work, as well as requiring the destruction of a perfectly good motor.

2.5 L293D

L293D is a typical Motor driver or Motor Driver IC which allows DC motor to drive on either direction. L293D is a 16-pin IC which can control a set of two DC motors simultaneously in any direction. It means that you can control two DC motor with a single L293D IC. Dual H–bridge *Motor Driver integrated circuit* (*IC*).

The l293d can drive small and quiet big motors as well, check the Voltage Specification at the end of this page for more info.

You can Buy L293D IC in any electronic shop very easily and it costs around 70 Rupees (INR) or around 1 $ Dollar (approx Cost) or even lesser cost. You can find the necessary pin diagram, working, a circuit diagram, Logic description and Project as you read through.

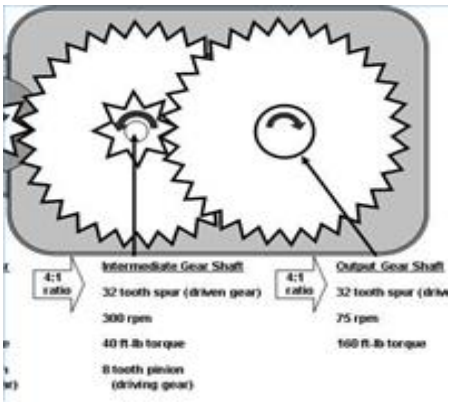
It works on the concept of H-bridge. H-bridge is a circuit which allows the voltage to be flown in either direction. As you know voltage need to change its direction for being able to rotate the motor in clockwise or anticlockwise direction, Hence H-bridge IC are ideal for driving a DC motor.

In a single L293D chip there are two h-Bridge circuit inside the IC which can rotate two dc motor independently. Due its size it is very much used in robotic application for controlling DC motors. Given below is the pin diagram of a L293D motor controller.

2.6 Gear motor

What Is a Gear Motor?

Gear motors are complete motive force systems consisting of an electric motor and a reduction gear train integrated into one easy-to-mount and -configure package. This greatly reduces the complexity and cost of designing and constructing power tools, machines and appliances calling for high torque at relatively low shaft speed or RPM. Gear motors allow the use of economical low-horsepower motors to provide great motive force at low speed such as in lifts, winches, medical tables, jacks and robotics. They can be large enough to lift a building or small enough to drive a tiny clock.



12V High Torque DC GEAR MOTOR

Operation Principle

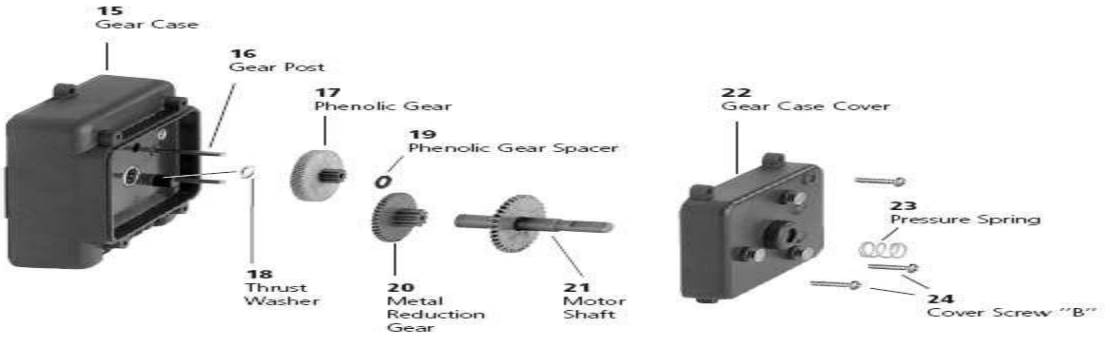
Most synchronous AC electric motors have output ranges of from 1,200 to 3,600 revolutions per minute. They also have both normal speed and stall-speed torque specifications. The reduction gear trains used in gear

motors are designed to reduce the output speed while increasing the torque. The increase in torque is inversely proportional to the reduction in speed. Reduction gearing allows small electric motors to move large driven loads, although more slowly than larger electric motors. Reduction gears consist of a small gear driving a larger gear. There may be several sets of these reduction gear sets in a reduction gear box.

Article II. Gear

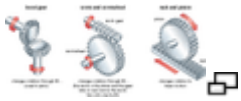
Toothed wheel that transmits the turning movement of one shaft to another shaft. Gear wheels may be used in pairs or in threes if both shafts are to turn in the same direction. The gear ratio – the ratio of the number of teeth on the two wheels – determines the torque ratio, the turning force on the output shaft compared with the turning force on the input shaft. The ratio of the angular velocities of the shafts is the inverse of the gear ratio.

The common type of gear for parallel shafts is the **spur gear**, with straight teeth parallel to the shaft axis. The **helical gear** has teeth cut along sections of a helix or corkscrew shape; the double form of the helix gear is the most efficient for energy transfer. **Bevel gears**, with tapering teeth set on the base of a cone, are used to connect intersecting shafts.



The toothed and interlocking wheels which make up a typical gear movement.

Gear ratio is calculated by dividing the number of teeth on the driver gear by the number of teeth on the driven gear (gear ratio = driver/driven); the idler gears are ignored. Idler gears change the direction of rotation but do not affect speed. A high driven to driver ratio (middle) is a speed-reducing ratio.



Different gears are used to perform different engineering functions depending on the change in direction of motion that is needed. Rack and pinion gears are the commonest gears and are used in car steering mechanics.

Speed Reduction

- Sometimes the goal of using a gear motor is to reduce the rotating shaft speed of a motor in the device being driven, such as in a small electric clock where the tiny synchronous motor may be spinning at 1,200 rpm but is reduced to one rpm to drive the second hand, and further reduced in the clock mechanism to drive the minute and hour hands. Here the amount of driving force is irrelevant as long as it is sufficient to overcome the frictional effects of the clock mechanism.

Torque Multiplication

- Another goal achievable with a gear motor is to use a small motor to generate a very large force albeit at a low speed. These applications include the lifting mechanisms on hospital beds, power recliners, and heavy machine lifts where the great force at low speed is the goal.

Motor Varieties

- Most industrial gear motors are AC-powered, fixed-speed devices, although there are fixed-gear-ratio, variable-speed motors that provide a greater degree of control. DC gear motors are used primarily in automotive applications such as power winches on trucks, windshield wiper motors and power seat or power window motors.

Many Applications

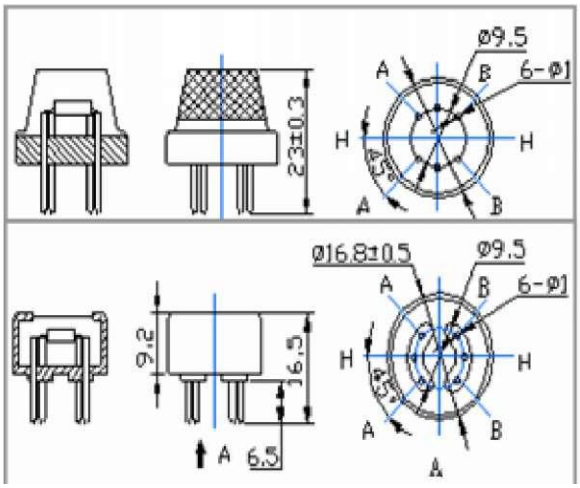
- What power can openers, garage door openers, stair lifts, rotisserie motors, timer cycle knobs on washing machines, power drills, cake mixers and electromechanical clocks have in common is that they all use various integrations of gear motors to derive a large force from a relatively small electric motor at a manageable speed. In industry, gear motor applications in jacks, cranes, lifts, clamping, robotics, conveyance and mixing are too numerous to count.

2.6 ALCOHOL SENSOR MQ-3 Semiconductor Sensor for Alcohol

Sensitive material of MQ-3 gas sensor is SnO_2, which with lower conductivity in clean air. When the target alcohol gas exist, The sensor's conductivity is more higher along with the gas concentration rising. Please use simple electrocircuit, Convert change of conductivity to correspond output signal of gas concentration.

MQ-3 gas sensor has high sensitity to Alcohol, and has good resistance to disturb of gasoline, smoke and vapor. The sensor could be used to detect alcohol with different concentration, it is with low cost and suitable for different application.

Configuration



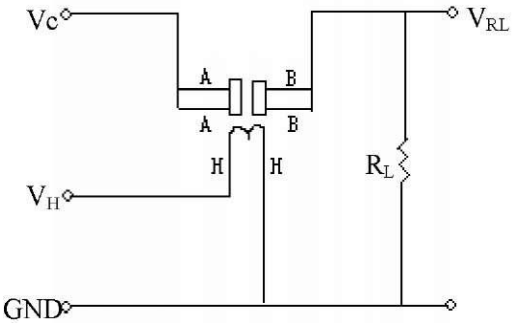
Character

* Good sensitivity to alcohol gas
* Long life and l ow cost
* Simple drive circuit

Application

* Vehicel alcohol detector
* Portable alcohol detector

Technical Data



The above is basic test circuit of the sensor. The sensor need to be put 2 voltage, heater voltage(VH) and test voltage(VC). VH used to supply certified working temperature to the sensor, while VC used to detect voltage (VRL) on load resistance (RL) whom is in series with sensor. The sensor has light polarity, Vc need DC power. VC and VH could use same power circuit with precondition to assure performance of sensor. In order to make the sensor with better performance, suitable RL value is needed: Power of Sensitivity body(Ps):

Model No.			MQ-3
Sensor Type			Semiconductor
Standard Encapsulation			Bakelite (Black Bakelite)
Detection Gas			Alcohol gas
Concentration			0.04-4mg/l alcohol
Circuit	Loop Voltage	c	<24V DC
	Heater Voltage	Vh	5.0V±0.2V AC or DC
	Load Resistance	Rl	Adjustable
Character	Heater Resistance	Rh	31Q±3Q (Room Tem.)
	Heater consumption	Ph	<900mW
	Sensing Resistance	s	2KQ-20KQ(in 0.4mg/l alcohol)
	Sensitivity	S	Rs(in air)/Rs(0.4mg/L Alcohol)>5

	Slope	a	<0.6(R300ppm/R100ppm Alcohol)
Condition	Tem. Humidity		20°C±2°C; 65%±5%RH
	Standard test circuit		Vc:5.0V±0.1V; VH: 5.0V±0.1V
	Preheat time		Over 48 hours

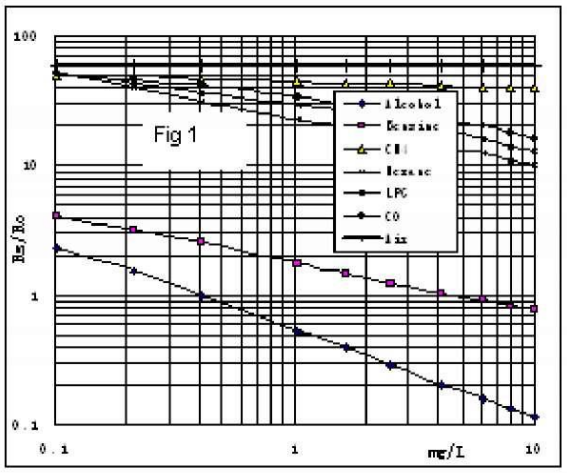
$Ps=Vc^2*Rs/(Rs+RL)^2$

Basic test loop

Resistance of sensor(Rs): Rs=(Vc/VRL-1)*RL

Sensitivity Characteristics

Influence of Temperature/Humidity



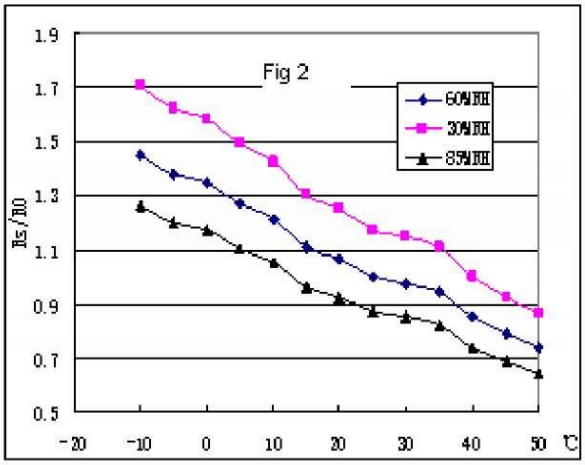
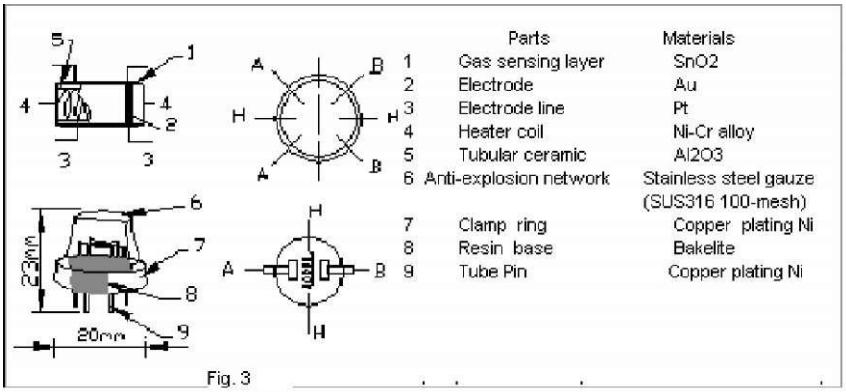


Fig.2 shows the typical temperature and humidity characteristics. Ordinate means resistance ratio of the sensor (Rs/Ro), Rs means resistance of sensor in 0.4mg/l alcohol under different tem. and humidity. Ro means resistance of the sensor in environment of 0.4mg/l alcohol, 20°C/65%RH Fig.1 shows the typical sensitivity characteristics of the MQ-3, ordinate means resistance ratio of the sensor (Rs/Ro), abscissa i s concentration of gases. Rs means resistance in different gases, Ro means resistance of sensor in 0. 4mg/l alcohol. All test are under standard test conditions.

P.S.: Sensitivity to smoke is ignite 10pcs cigarettes in $8m^3$ room, and the output equals to 0.1mg/l alcohol

Structure and configuration



	Parts	Materials
1	Gas sensing layer	SnO2
2	Electrode	Au
3	Electrode line	Pt
4	Heater coil	Ni-Cr alloy
5	Tubular ceramic	Al2O3
6	Anti-explosion network	Stainless steel gauze (SUS316 100-mesh)
7	Clamp ring	Copper plating Ni
8	Resin base	Bakelite
9	Tube Pin	Copper plating Ni

Fig. 3

Structure and configuration of MQ-3 gas sensor is shown as Fig. 3, sensor composed by micro AL2O3 ceramic tube, Tin Dioxide (SnO2) sensitive layer, measuring electrode and heater are fixed into a crust made by plastic and stainless steel net. The heater provides necessary work conditions for work of sensitive components. The enveloped MQ-4 have 6 pin, 4 of them are used to fetch signals, and other 2 are used for providing heating current.

Notification

1 Following conditions must be prohibited

1. Exposed to organic silicon steam

Organic silicon steam cause sensors invalid, sensors must be avoid exposing to silicon bond, fixature, silicon latex, putty or plastic contain silicon environment

2. High Corrosive gas

If the sensors exposed to high concentration corrosive gas (such as H_2Sz, SO_X, Cl_2, HCl etc), it will not only result in corrosion of sensors structure, also it cause sincere sensitivity attenuation.

3. Alkali, Alkali metals salt, halogen pollution

The sensors performance will be changed badly if sensors be sprayed polluted by alkali metals salt especially brine, or be exposed to halogen such as fluorin.

4. Touch water

Sensitivity of the sensors will be reduced when spattered or dipped in water.

5. Freezing

Do avoid icing on sensor'surface, otherwise sensor would lose sensitivity.

6. Applied voltage higher

Applied voltage on sensor should not be higher than stipulated value, otherwise it cause down-line or heater damaged, and bring on sensors' sensitivity characteristic changed badly.

7. Voltage on wrong pins . _ For 6 pins sensor, if apply voltage on 3 pins or 6 pins

2 Following conditions must be avoided

2.1 Water Condensation '

Indoor conditions, slight water condensation will effect sensors performance lightly. However, if water condensation on sensors surface and keep a certain period, sensor' sensitivity will be decreased.

2.2 Used in high gas concentration

No matter the sensor is electrified or not, if long time placed in high gas concentration, if will affect sensors characteristic.

2.3 Long time storage

The sensors resistance produce reversible drift if it's stored for long time without electrify, this drift is related with storage conditions. Sensors should be stored in airproof without silicon gel bag with clean air. For the sensors with long time storage but no electrify, they need long aging time for stbility before using.

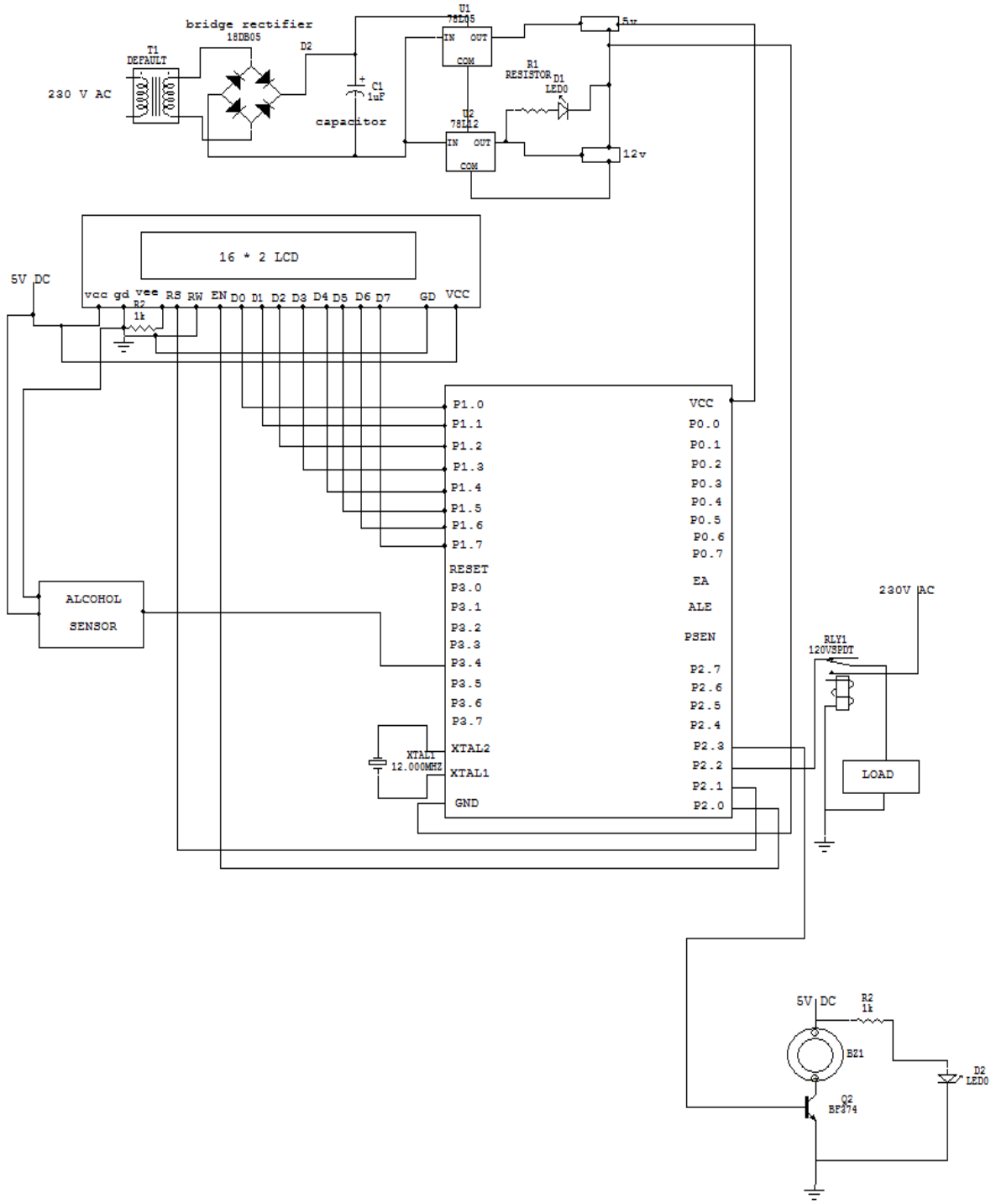
2.4 Long time exposed to adverse environment

No matter the sensors electrified or not, if exposed to adverse environment for long time, such as high humidity, high temperature, or high pollution etc, it will effect the sensors performance badly.

CHAPTER 3

CIRCUITS AND THEIR OPERATION

3.1 Circuit diagram



2.1 source code

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(6, 7, 2, 3, 4, 5);


int alcohol=8;
int buzzer=9;
int motor=10;

void setup()
{
   lcd.begin(16, 2);    ////16*2 LCD
   lcd.print("VEHICLE SECURITY");
  lcd.setCursor(0, 1);   ///    first zero row no, second zero
     line
  lcd.print("TO AVOD ACCIDENTS");
   delay(200);
   delay(200);
}
void loop()
{
  if(digitalRead(alcohol)==0)
  {
  digitalWrite(buzzer, HIGH);
  digitalWrite(motor, LOW);
  lcd.clear();
  lcd.print(" ENGINE STOPPED  ");
  delay(200);
  }
  else
  {
    lcd.clear();
   lcd.setCursor(0, 0);    ///     first zero row no, second
     zero line
```

```
   lcd.print("VEHICLE IS SAFE ");
  lcd.setCursor(0, 1);   ///    first zero row no, second zero
     line
  lcd.print("  ENGINE RUNNING ");
  digitalWrite(buzzer, LOW);
  digitalWrite(motor, HIGH);
  delay(200);
  }
}
```

CHAPTER 4

SOFTWARE DEVELOPMENT

4.1 Introduction:

In this chapter the software used and the language in which the program code is defined is mentioned and the program code dumping tools are explained. The chapter also documents the development of the program for the application. This program has been termed as " Source code" . Before we look at the source code we define the two header files that we have used in the code.

4.2 Tools Used:

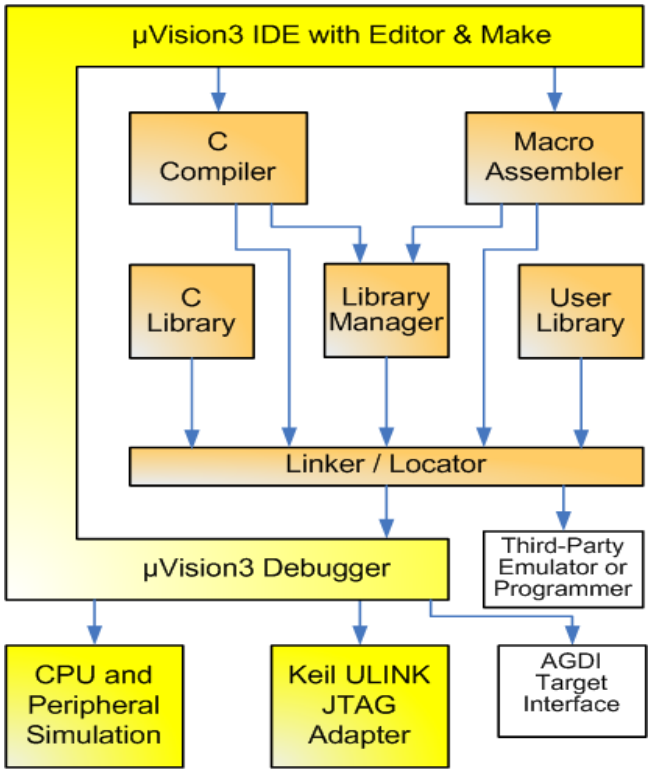


Figure 5.1 Keil Software- internal stages

Keil development tools for the 8051 Microcontroller Architecture support every level of software developer from the professional applications

4.3 C51 Compiler & A51 Macro Assembler:

Source files are created by the µVision IDE and are passed to the C51 Compiler or A51 Macro Assembler. The compiler and assembler process source files and create replaceable object files.

The Keil C51 Compiler is a full ANSI implementation of the C programming language that supports all standard features of the C language. In addition, numerous features for direct support of the 8051 architecture have been added.

4.4 START µVISION

What's New in µVision3?

µVision3 adds many new features to the Editor like Text Templates, Quick Function Navigation, and Syntax Coloring with brace high lighting Configuration Wizard for dialog based startup and debugger setup. µVision3 is fully compatible to µVision2 and can be used in parallel with µVision2.

What is µVision3?

µVision3 is an IDE (Integrated Development Environment) that helps you write, compile, and debug embedded programs. It encapsulates the following components:

- A project manager.
- A make facility.
- Tool configuration.
- Editor.
- A powerful debugger.

To help you get started, several example programs (located in the **\C51\Examples**, **\C251\Examples**, **\C166\Examples**, and **\ARM\...\Examples**) are provided.

- **HELLO** is a simple program that prints the string "Hello World" using the Serial Interface.
- **MEASURE** is a data acquisition system for analog and digital systems.
- **TRAFFIC** is a traffic light controller with the RTX Tiny operating system.
- **SIEVE** is the SIEVE Benchmark.
- **DHRY** is the Dhrystone Benchmark.
- **WHETS** is the Single-Precision Whetstone Benchmark.

Additional example programs not listed here are provided for each device architecture.

7.3 BUILDING AN APPLICATION IN µVISION

To build (compile, assemble, and link) an application in µVision2, you must:

1. Select Project -(forexample,**166\EXAMPLES\HELLO\HELLO.UV2**).

2. Select Project - Rebuild all target files or Build target.

µVision2 compiles, assembles, and links the files in your project.

Creating Your Own Application in µVision2

To create a new project in µVision2, you must:

1. Select Project - New Project.
2. Select a directory and enter the name of the project file.
3. Select Project - Select Device and select an 8051, 251, or C16x/ST10 device from the Device Database™.
4. Create source files to add to the project.
5. Select Project - Targets, Groups, Files. Add/Files, select Source Group1, and add the source files to the project.
6. Select Project - Options and set the tool options. Note when you select the target device from the Device Database™ all special options are set automatically. You typically only need to configure the memory map of your target hardware. Default memory model settings are optimal for most applications.
7. Select Project - Rebuild all target files or Build target.

Debugging an Application in µVision2

To debug an application created using µVision2, you must:

1. Select Debug - Start/Stop Debug Session.
2. Use the Step toolbar buttons to single-step through your program. You may enter **G, main** in the Output Window to execute to the main C function.
3. Open the Serial Window using the **Serial #1** button on the toolbar.

Debug your program using standard options like Step, Go, Break, and so on.

Starting µVision2 and Creating a Project

µVision2 is a standard Windows application and started by clicking on the program icon. To create a new project file select from the µVision2 menu

Project – New Project…. This opens a standard Windows dialog that asks you for the new project file name.

We suggest that you use a separate folder for each project. You can simply use the icon Create New Folder in this dialog to get a new empty folder. Then select this folder and enter the file name for the new project, i.e. Project1.

µVision2 creates a new project file with the name PROJECT1.UV2 which contains a default target and file group name. You can see these names in the Project **Window – Files.**

Now use from the menu Project – Select Device for Target and select a CPU for your project. The Select Device dialog box shows the µVision2 device database. Just select the microcontroller you use. We are using for our examples the Philips 80C51RD+ CPU. This selection sets necessary tool options for the 80C51RD+ device and simplifies in this way the tool Configuration

Building Projects and Creating a HEX Files

Typical, the tool settings under Options – Target are all you need to start a new application. You may translate all source files and line the application with a click on the Build Target toolbar icon. When you build an application with syntax errors, µVision2 will display errors and warning messages in the Output Window – Build page. A double click on a message line opens the source file on the correct location in a µVision2 editor window. Once you have successfully generated your application you can start debugging.

After you have tested your application, it is required to create an Intel HEX file to download the software into an EPROM programmer or simulator. µVision2 creates HEX files with each build process when Create HEX files under Options for Target – Output is enabled. You may start your PROM programming utility after the make process when you specify the program under the option Run User Program #1.

CPU Simulation

µVision2 simulates up to 16 Mbytes of memory from which areas can be mapped for read, write, or code execution access. The µVision2 simulator traps and reports illegal memory accesses.

In addition to memory mapping, the simulator also provides support for the integrated peripherals of the various 8051 derivatives. The on-chip peripherals of the CPU you have selected are configured from the Device

Database selection

You have made when you create your project target. Refer to page 58 for more

Information about selecting a device. You may select and display the on-chip peripheral components using the Debug menu. You can also change the aspects of each peripheral using the controls in the dialog boxes.

Start Debugging

You start the debug mode of µVision2 with the Debug – Start/Stop Debug Session command. Depending on the Options for Target – Debug Configuration, µVision2 will load the application program and run the startup

code µVision2 saves the editor screen layout and restores the screen layout of the last debug session. If the program execution stops, µVision2 opens an

editor window with the source text or shows CPU instructions in the disassembly window. The next executable statement is marked with a yellow arrow. During debugging, most editor features are still available.

For example, you can use the find command or correct program errors. Program source text of your application is shown in the same windows. The µVision2 debug mode differs from the edit mode in the following aspects:

⇒ The " Debug Menu and Debug Commands" described below are available. The additional debug windows are discussed in the following.

⇒ The project structure or tool parameters cannot be modified. All build Commands are disabled.

Disassembly Window

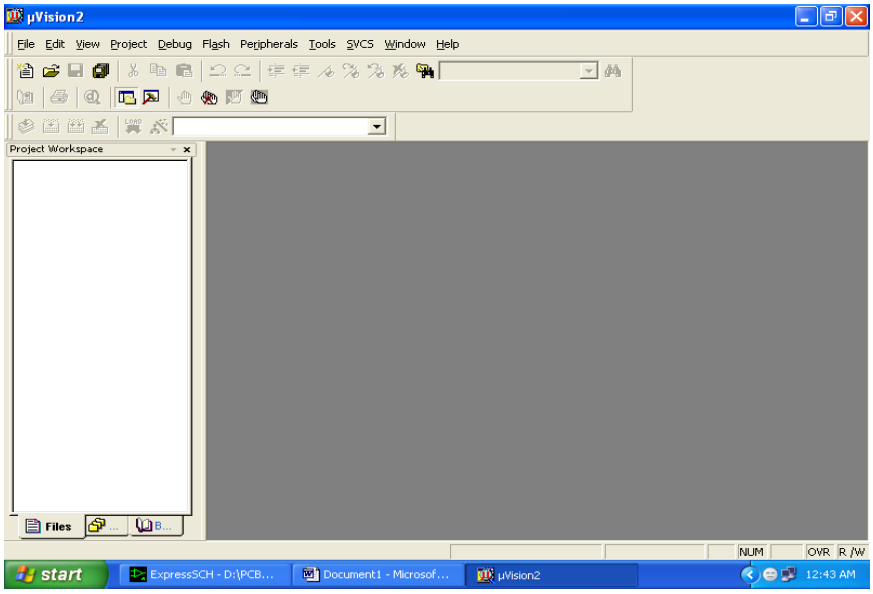
The Disassembly window shows your target program as mixed source and assembly program or just assembly code. A trace history of previously executed instructions may be displayed with Debug – View Trace Records. To enable the trace history, set Debug – Enable/Disable Trace Recording.

If you select the Disassembly Window as the active window all program step commands work on CPU instruction level rather than program source lines. You can select a text line and set or modify code breakpoints using toolbar buttons or the context menu commands.

You may use the dialog Debug – Inline Assembly…to modify the CPU instructions. That allows you to correct mistakes or to make temporary changes to the target program you are debugging.

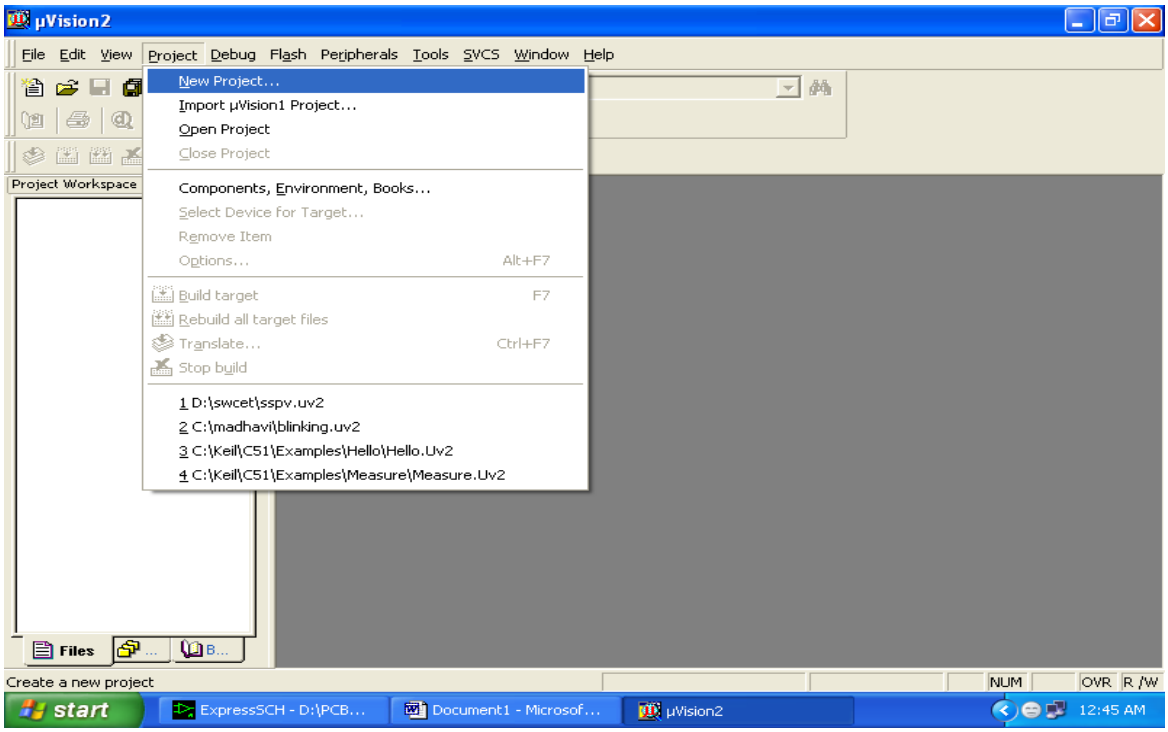
5.5 OVERVIEW OF KEIL UVISION SOFTWARE

1. Click on the Keil uVision Icon on Desktop
2. The following fig will appear

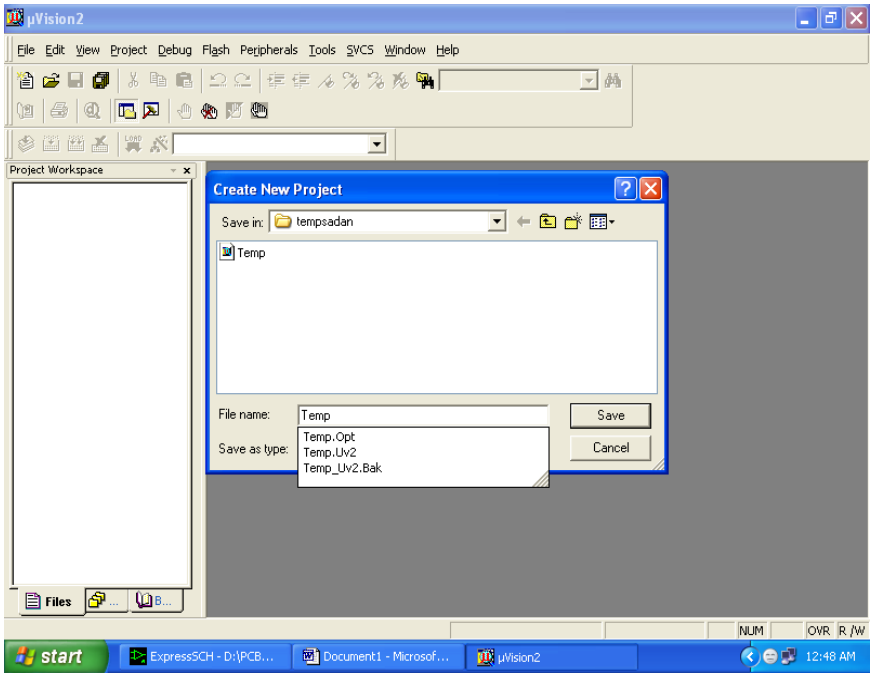


3. Click on the Project menu from the title bar

4. Then Click on New Project



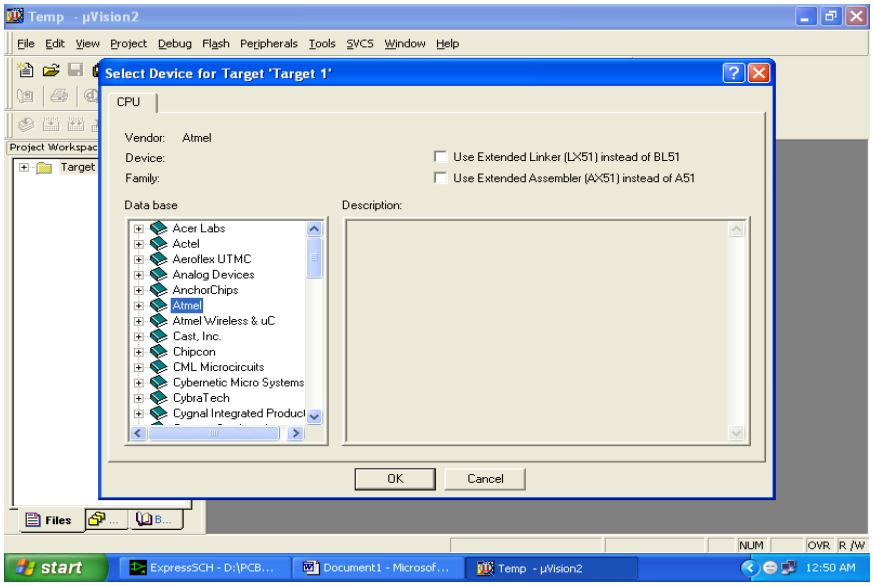
5. Save the Project by typing suitable project name with no extension in u r own folder sited in either C:\ or D:\



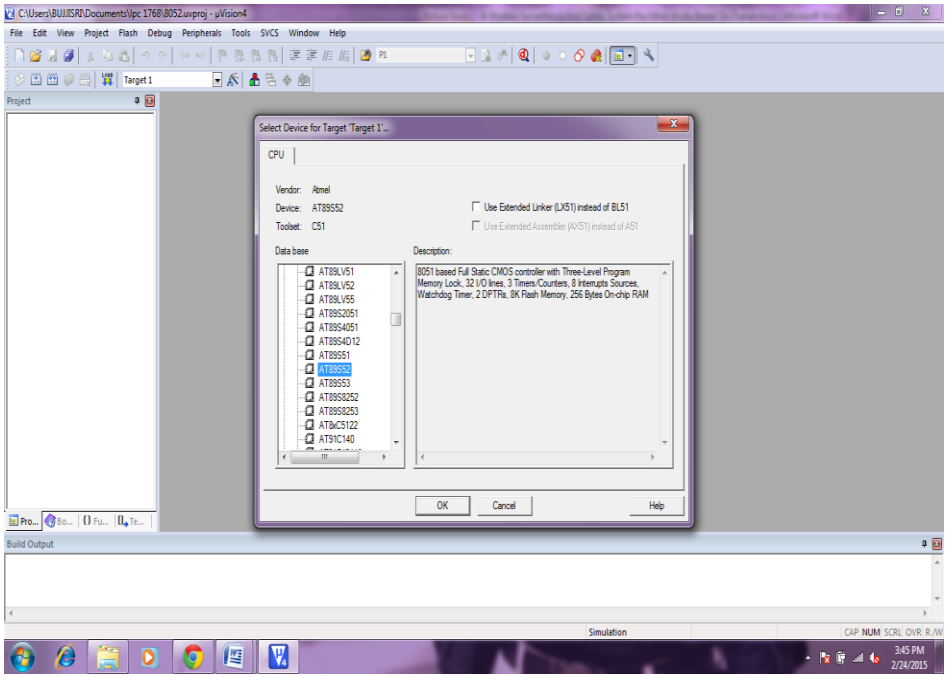
6. Then Click on Save button above.

7. Select the component for u r project. i.e. Atmel……

8. Click on the + Symbol beside of Atmel

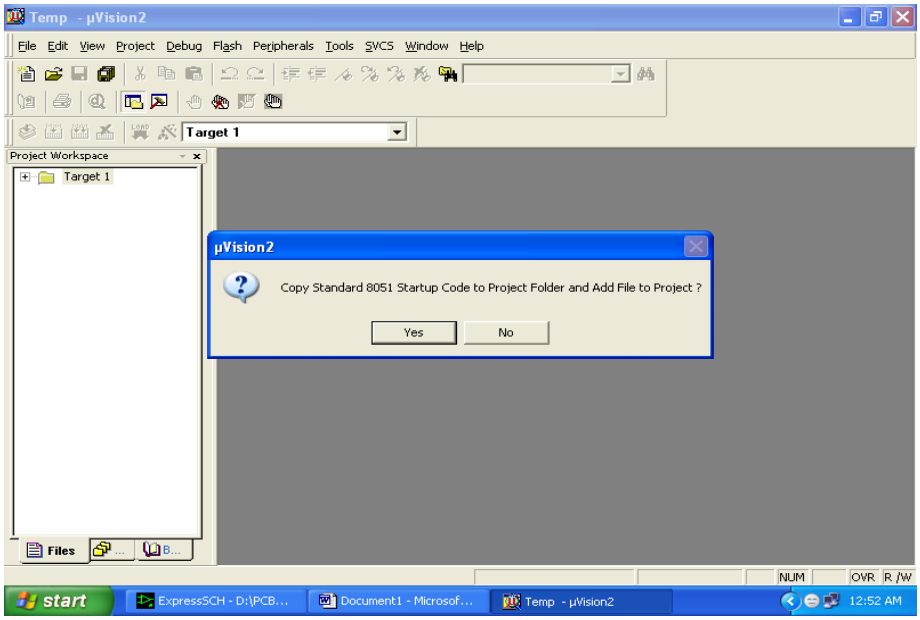


9. Select AT89S52 as shown below



10. Then Click on " OK"

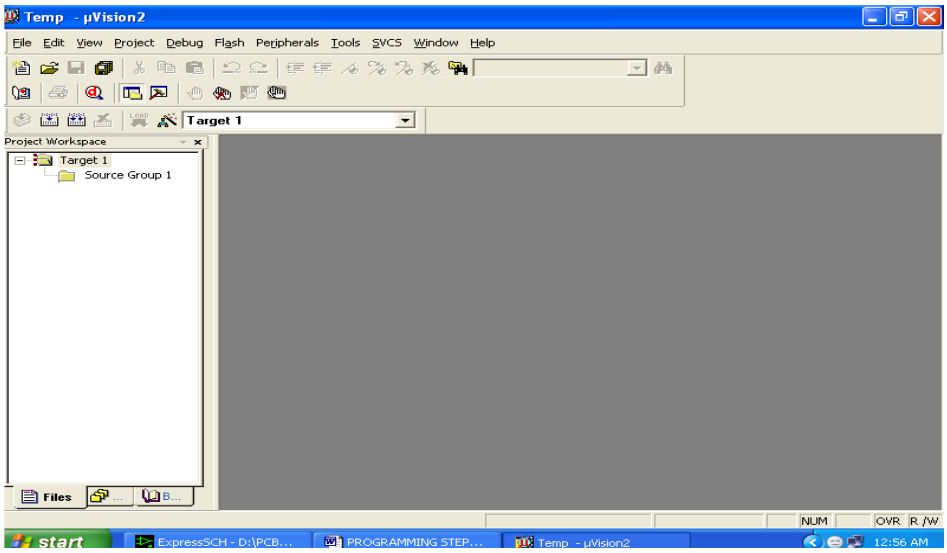
11. The Following fig will appear



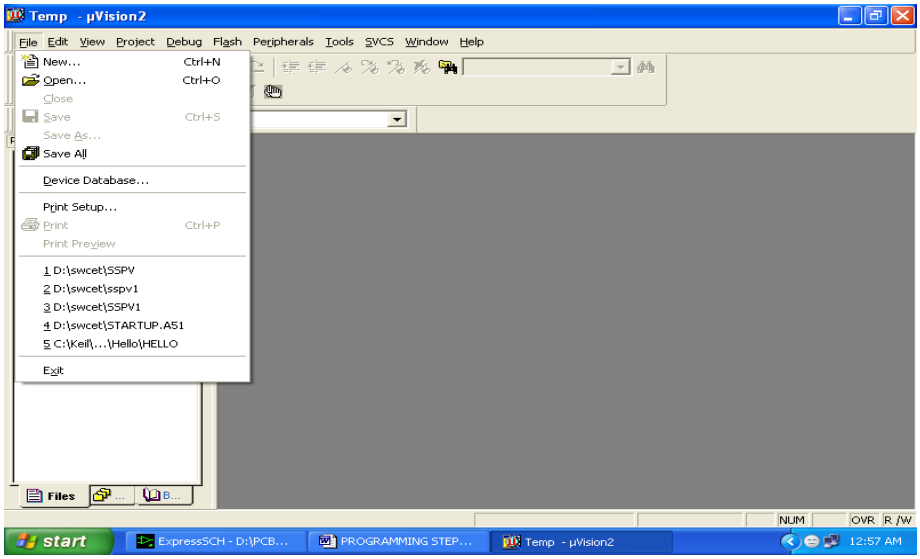
12. Then Click either YES or NO……mostly " NO"

13. Now your project is ready to USE

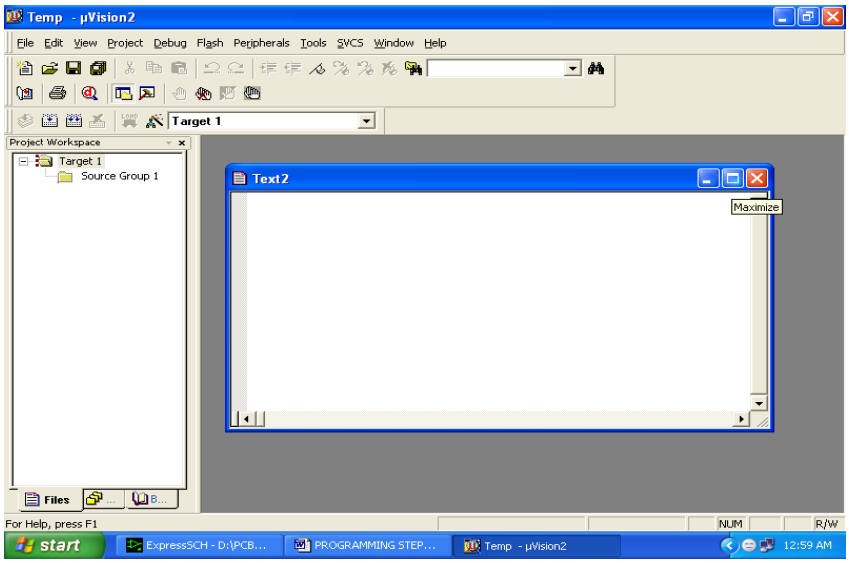
14. Now double click on the Target1, you would get another option " Source group 1" as shown in next page.



15. Click on the file option from menu bar and select " new"

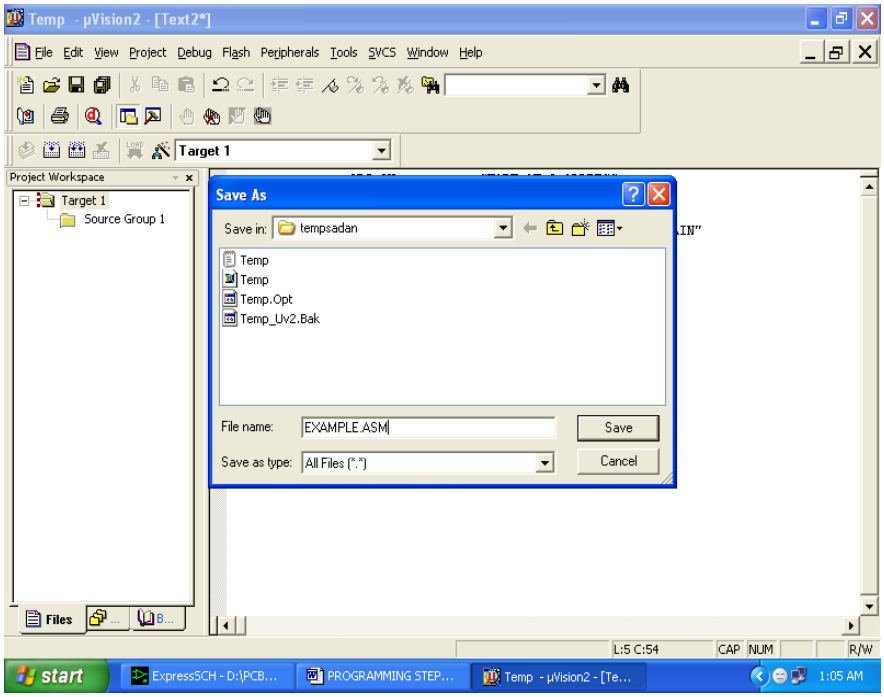


16. The next screen will be as shown in next page, and just maximize it by double clicking on its blue boarder.

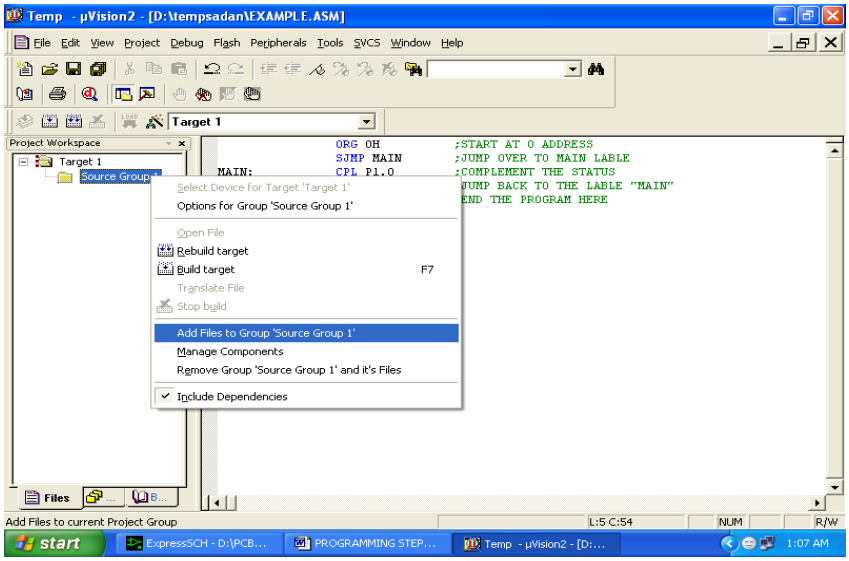


17. Now start writing program in either in “ C” or “ ASM”

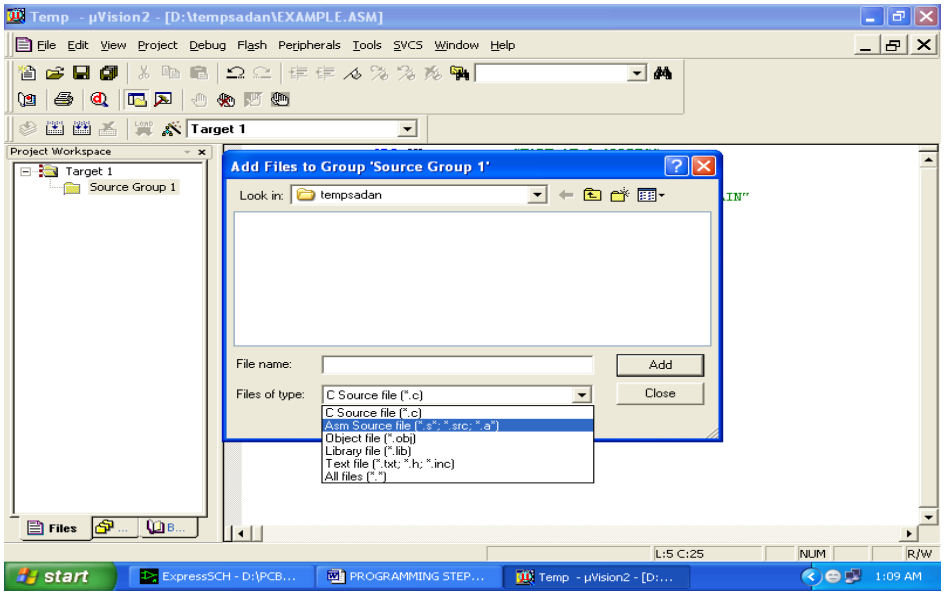
18. For a program written in Assembly, then save it with extension “ . asm” and for “ C” based program save it with extension “ .C”



19. Now right click on Source group 1 and click on “ **Add files to Group Source**”



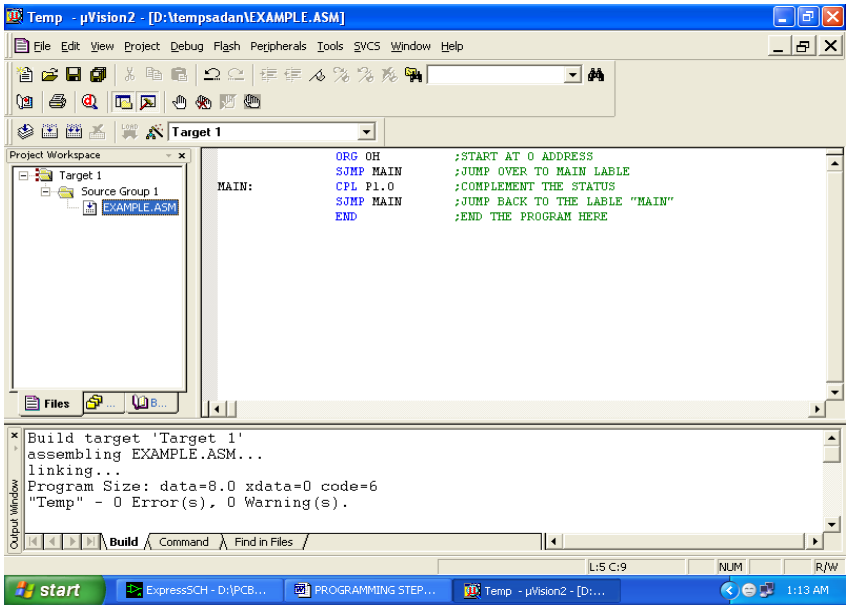
20. Now you will get another window, on which by default " C" files will appear.



21. Now select as per your file extension given while saving the file

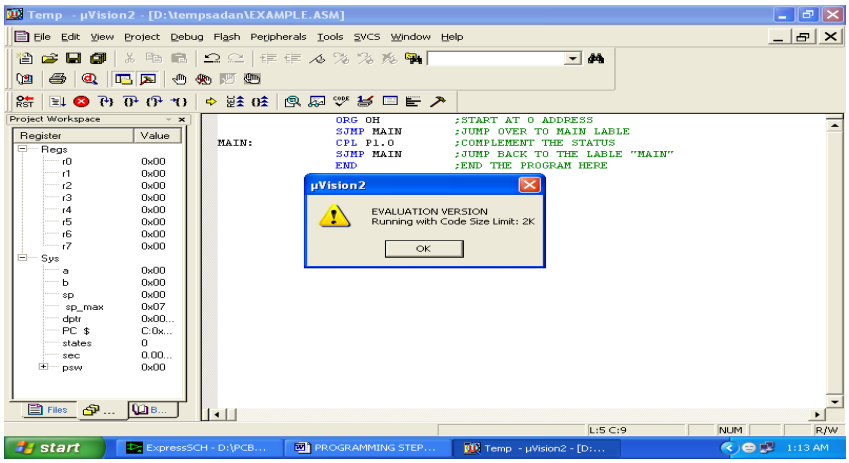
22. Click only one time on option " **ADD**"

23. Now Press function key F7 to compile. Any error will appear if so happen.



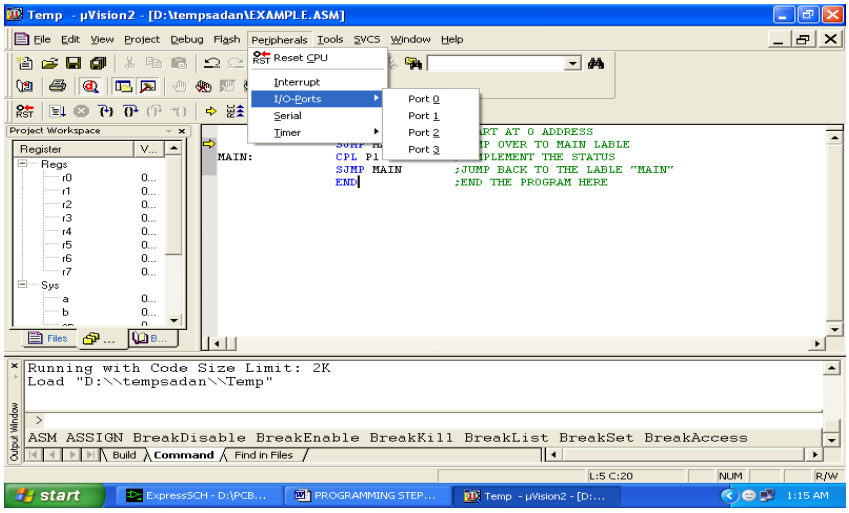
24. If the file contains no error, then press Control+F5 simultaneously.

25. The new window is as follows

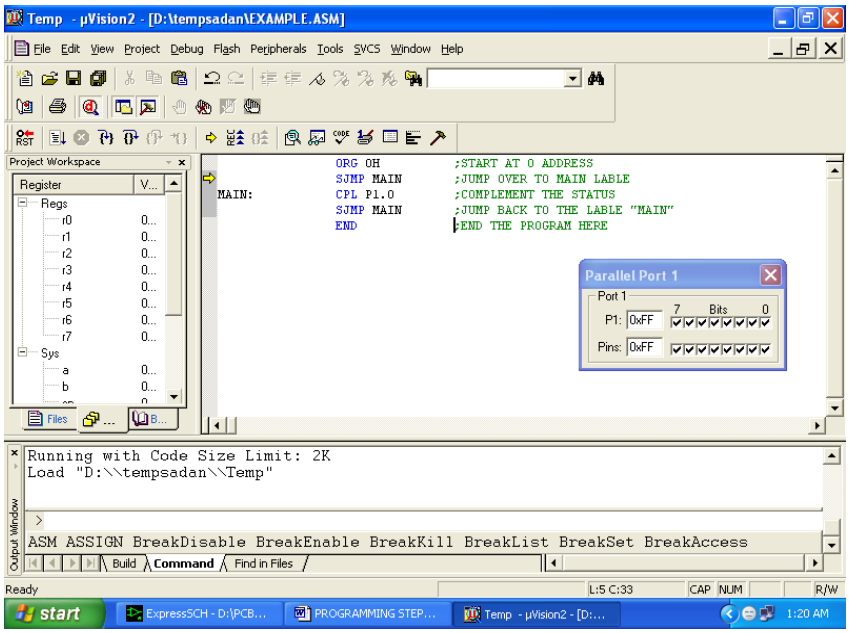


26. Then Click " OK"

27. Now Click on the Peripherals from menu bar, and check your required port as shown in fig below



28. Drag the port a side and click in the program file.



29. Now keep Pressing function key " F11" slowly and observe.

You are running your program successfully

SAMPLE PROGRAMS

Example 1:

```
        org 00h             // Starting  Of The Program From 00h memory
back:   mov P1,#55h         //Move 55h to Port1
        acall delay         // Call Delay Function
        mov P1,#0AAh        //Move 55h to Port1
        lcall delay         // Call Delay Function
        sjmp back
delay:  mov r5,#30h
again:  djnz r5,again       // Generating delay
        ret                 // Return Of Loop
        end                 // End Of Program
```

Example 2:

```
#include<reg51.h>
void delay(unsigned int);  //Global Declaration  Of Delay
void main()
 {
 P0=0x00;                  // Clearing Of Port O
while(1)                   //Infinite Loop
{
P0=0xAA;
delay(30);
P0=0x55;
delay(30);
}
 }
void delay(unsigned int x)         //Delay Main Function
```

```
{
unsigned int i,j;
for(i=0;i<=x;i++)
for(j=0;j<=1275;j++);
}
```

Example3:

```
                    #include<reg51.h>
                    sbit  SWITCH=P1^0;          // Input  to P1.0
                    sbit  LED =P2^5;            // Out  to P2.5
                    void main()
{
  while(1)                  //Infinite Loop
    {
         if (SWITCH==0)
           {
           LED=1;
           }
        else
           {
           LED=0;
           }
    }
 }
```

Example4:

```
#include<reg51.h>
unsigned char str[10]="MAGNI5";   // String Of Data
void main()
{
        unsigned int i=0;
        TMOD=0X20;                  // Timer1, Mode2
        SCON=0X50;                 //1 Start Bit And 1 Stop Bit
        TH1=-3;                     // Baud Rate 9600
        TR1=1;                     //Start Timer 1
While(1)
  {
                for(i=0;i<10;i++)
                {
                SBUF=str[i];
                while(TI==0);     // Wait  Data Till Bit Of Data
                TI=0;
                   }
          }
  }
```

CHAPTER 5

APPLICATIONS, ADVANTAGES AND CONCLUSION

ADVANTAGES:

1. By this method we can reduce time of the travelers .
2. By allowing ambulance in mean time by giving info to it we can save the patient in mean time .
3. Controlling traffic is easy.
4. GSM technology gives Seamless interoperability between networks and handsets

DISADVANTAGES:

1.

2.

APPLICATIONS:

1.

Conclusion

The project " **ALCOHOL DETECTION ROBOT"** has been successfully designed and tested. Integrating features of all the hardware components used have developed it. Presence of every module has been reasoned out and placed carefully thus contributing to the best working of the unit. Secondly, using highly advanced IC' s and with the help of growing technology the project has been successfully implemented.

CHAPTER 6

REFERENCE AND BIBILOGRAPHY

REFERENCE BOOKS

1. "The 8051 Microcontroller Architecture, Programming & Applications"

   By Kenneth J Ayala.

2. "The 8051 Microcontroller & Embedded Systems" by Mohammed Ali Mazidi and Janice Gillispie Mazidi

3. "Power Electronics" by M D Singh and K B Khanchandan

4. "Linear Integrated Circuits" by D Roy Choudary & Shail Jain

5. "Electrical Machines" by S K Bhattacharya

6. "Electrical Machines II" by B L Thereja

7. www.8051freeprojectsinfo.com

BABILIOGRAPHY

1. WWW.MITEL.DATABOOK.COM

2. WWW.ATMEL.DATABOOK.COM

3. WWW.FRANKLIN.COM

4. WWW.KEIL.COM