Institute of Aeronautical Engineering (IARE)

(Autonomous Institute NAAC Accreditation with "A" Grade Accredited by NBA Affiliation Status from JNTUH)

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

A

PROJECT REPORT

ON

"TRANSMISSION LINE FAULT DETECTION USING ANDROID APPLICATION VIA BLUETOOTH"

BATCH MEMBERS

20955A0239 – M. SHIVA GANESH

20955A0251 – S. VINITH KUMAR

20955A0253 – M. YASWANTH MITRA

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 airconditioner, 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:

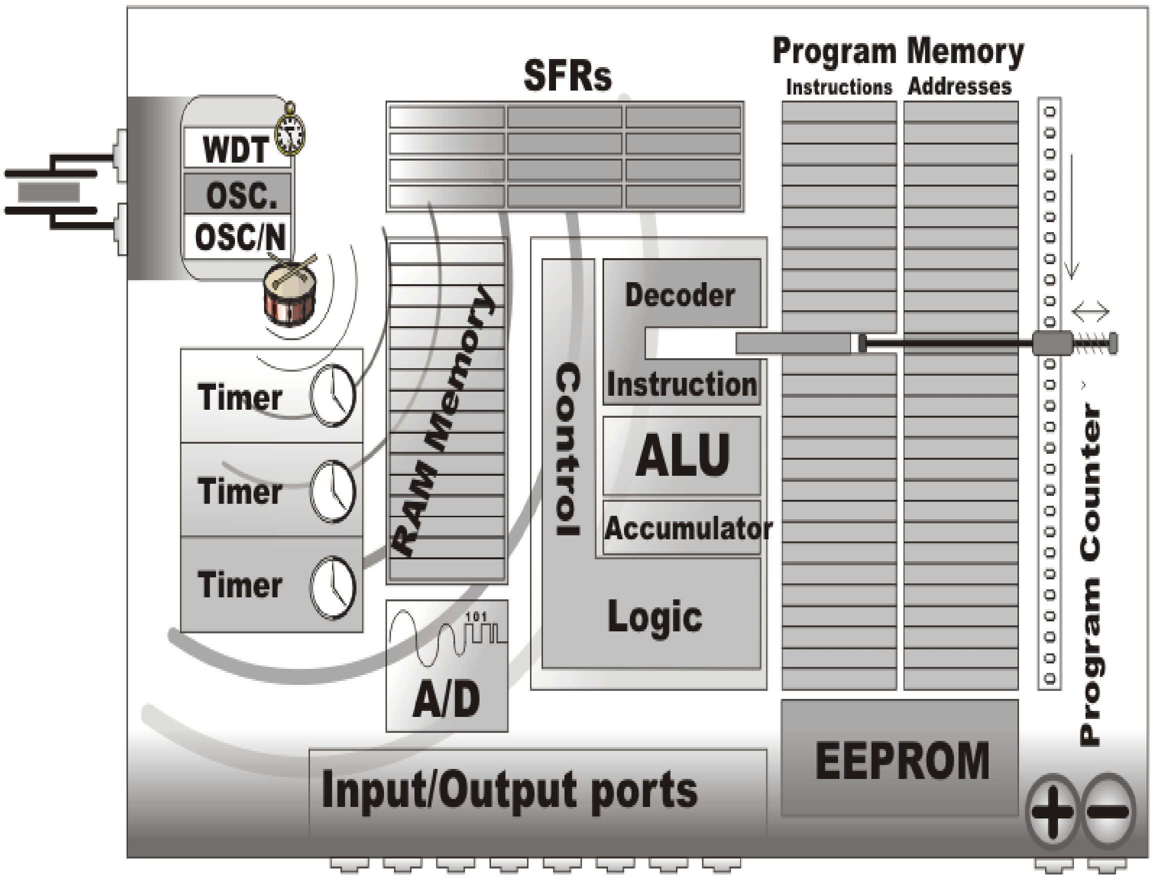
- ➢ Powerful and carefully chosen electronics embedded in the microcontrollers can independetly or via input/output devices (switches, push buttons, sensors, LCD displays, relays etc.), control various processes and devices such as industrial automation, electric current, temperature, engine performance 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.

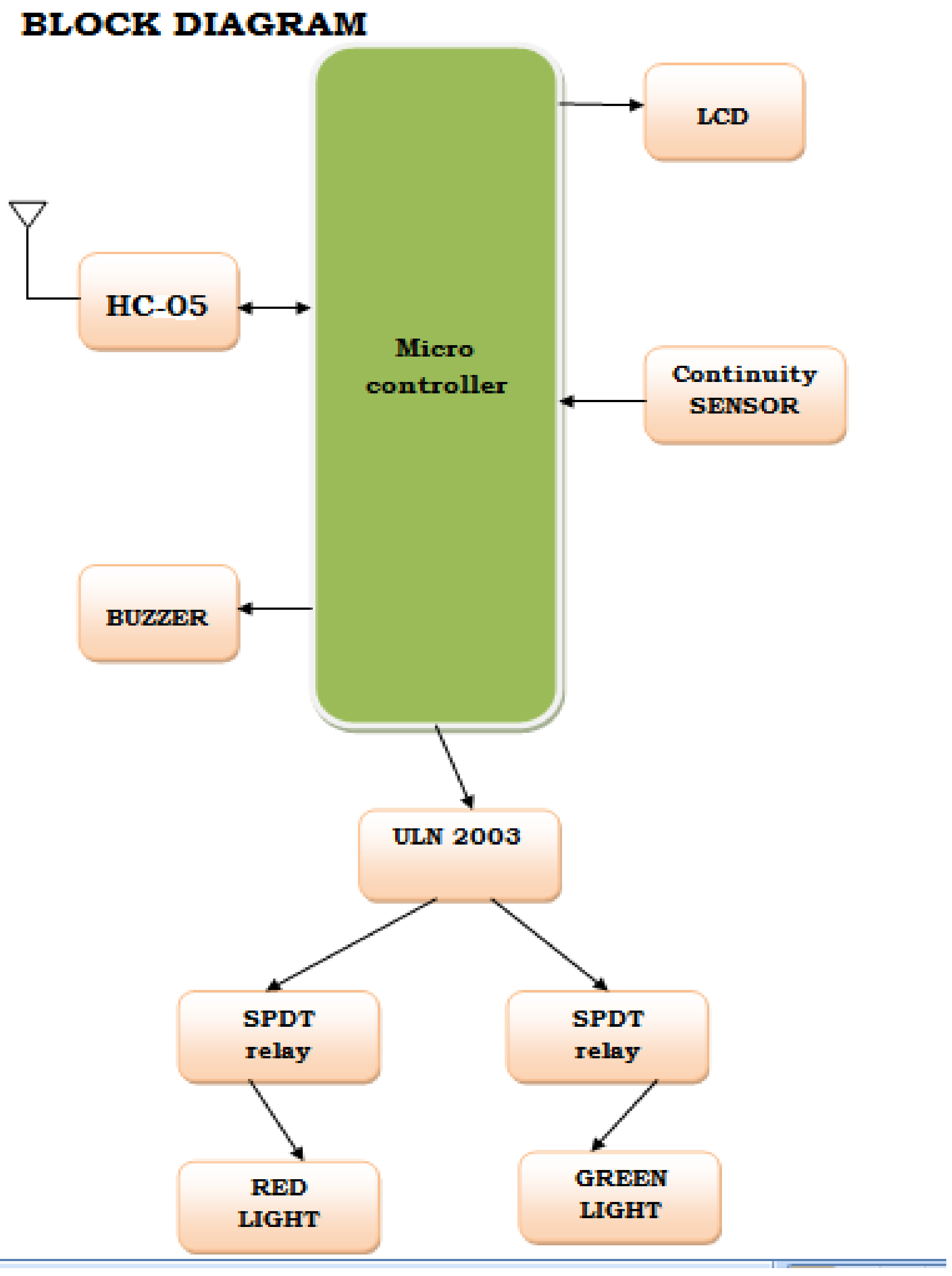
1.1.19 Special Function Registers (SFR)

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

Power grids are vast complex networks that make up a large part of an infrastructure. Many precautions are taken, and operators hired to maintain reliability, however three fourths of power outages are caused by operator errors. These errors can be avoided by automatic adjustments based on models of the grid system. The model explored is ensuring generator synchronization within the system. Finally, not only will the grid not have destructive interference, constructive interference will occur which increases the total power the grid can produce which optimizes the grid.

This paper presents the design of a system to detect the synchronization failure of any external supply source to the power grid on sensing the abnormalities in frequency and voltage and thereby protecting the load. There are several power generation units connected to the grid such as hydra, thermal, solar etc., to supply power to the load. These generating units need to supply power according to the rules of the grid. These rules involve maintaining a voltage variation within limits and also the frequency. If any deviation from the acceptable limit of the grid, it is mandatory that the same feeder should automatically get disconnected from the grid which by effect is termed as islanding. This prevents in large scale brown out or black out of the grid power. So, it is preferable to have a system which can warn the grid in advance so that alternate arrangements are kept on standby to avoid complete grid failure. This system is based on AT89S52 microcontroller. The microcontroller monitors the under/over voltage being derived from a set of comparators and a standard AT89S52 is used to vary the input voltage to test the functioning of the paper. A lamp load (indicating a predictable blackout, brownout) being driven from the microcontroller in case of voltage/frequency going out of acceptable range. GSM technologies are used to indicate the fault location.

CHAPTER 2

DESCRIPTION OF HARDWARE COMPONENTS

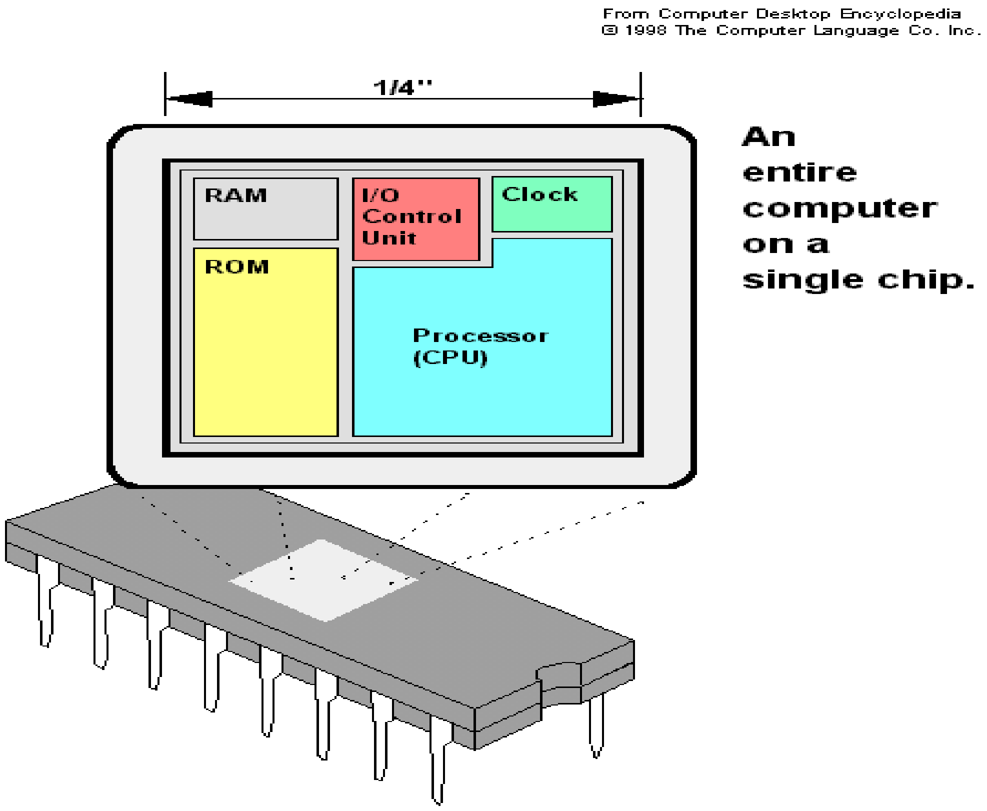
2.1 AT89S52

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


From Computer Desktop Encyclopedia
© 1998 The Computer Language Co. Inc.
1/4"
RAM
I/O Control Unit
Clock
ROM
Processor (CPU)
An entire computer on a single chip.

- ❖ 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

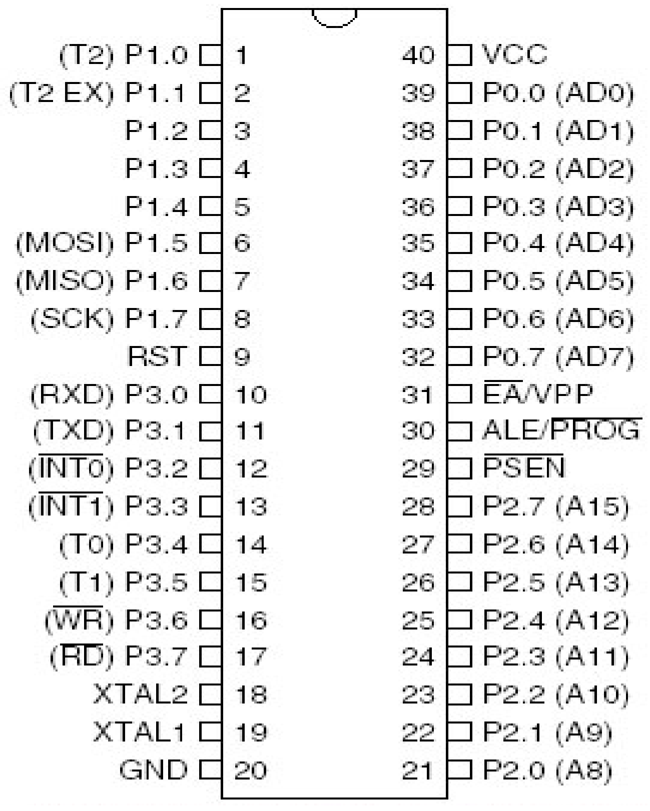


FIG-2 PIN DIAGRAM OF 89S52 IC

2.1.4 PIN DESCRIPTION

Pin Description

VCC

Supply voltage.

GND

Ground.

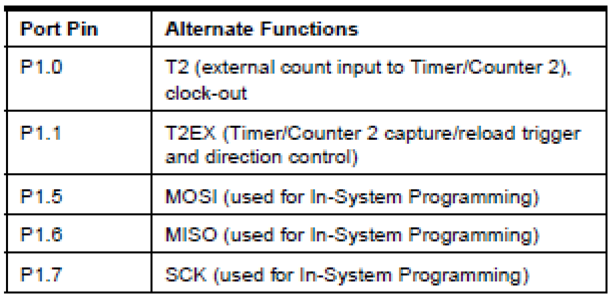
Port 0

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.



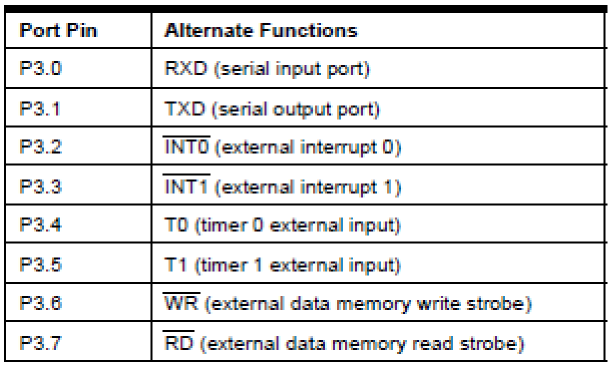
Port Pin	Alternate Functions
P1.0	T2 (external count input to Timer/Counter 2), clock-out
P1.1	T2EX (Timer/Counter 2 capture/reload trigger and direction control)
P1.5	MOSI (used for In-System Programming)
P1.6	MISO (used for In-System Programming)
P1.7	SCK (used for In-System Programming)

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.

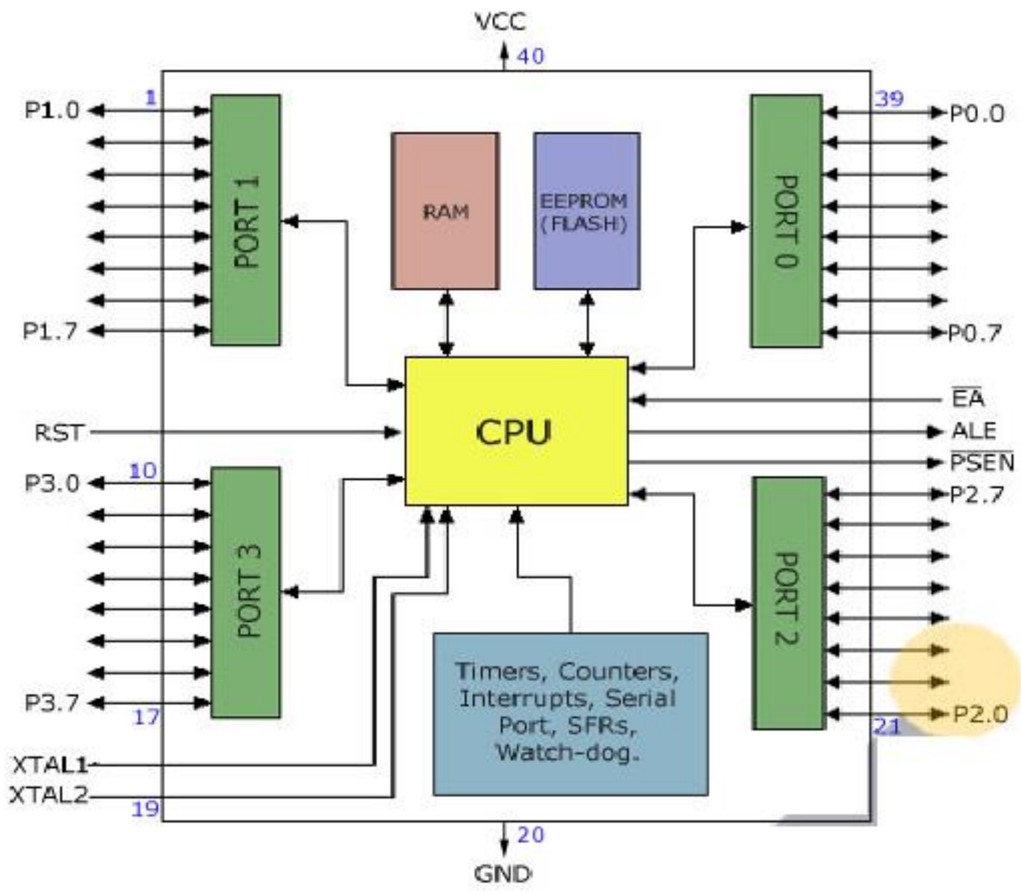
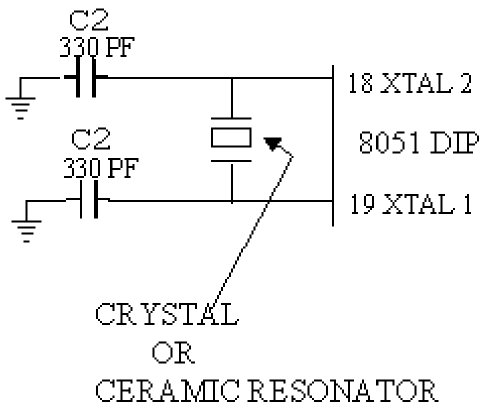


FIG-3 Functional block diagram of micro controller

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.



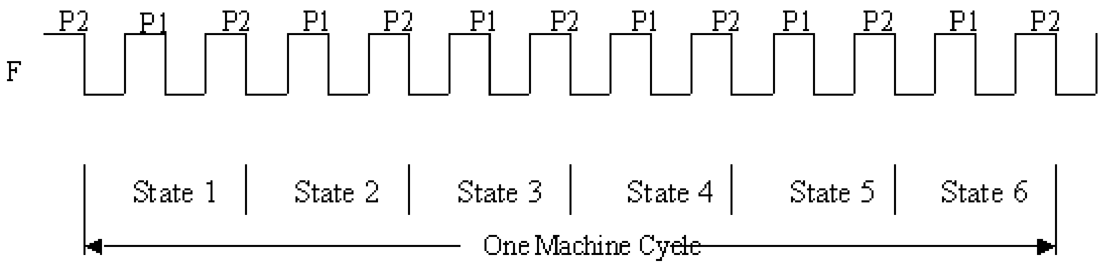


Fig-4 Oscillator and timing circuit

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

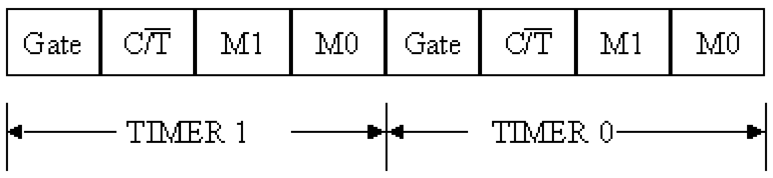
g) TCON (timer control, 88h)

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.

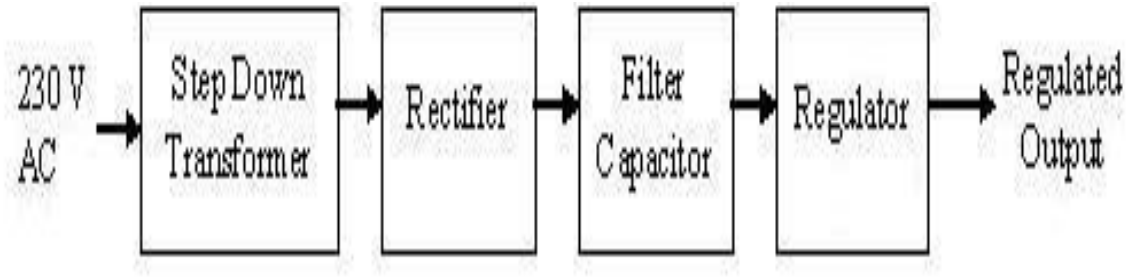
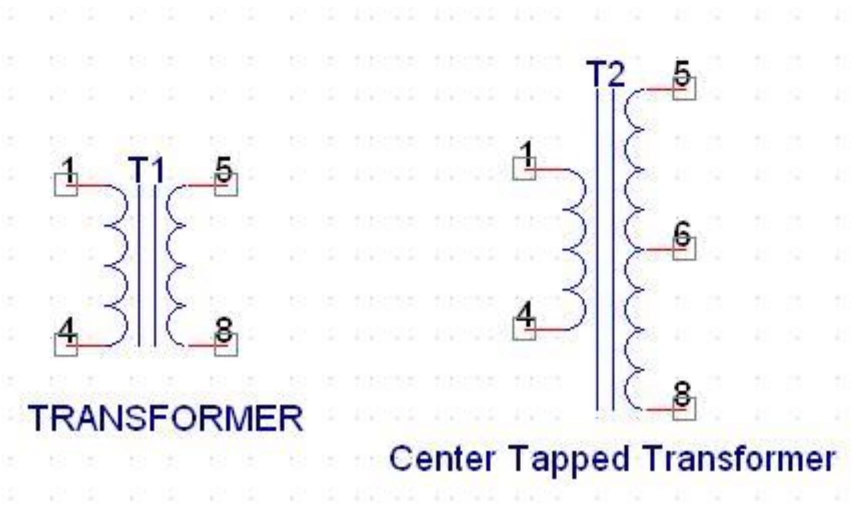


Figure 1 shows the basic block diagram of a fixed regulated power supply. Let us go through each block.

<u>TRANSFORMER</u>



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.

So
$$P_{primary} = P_{secondary}$$
$$I_p V_p = I_s V_s$$

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

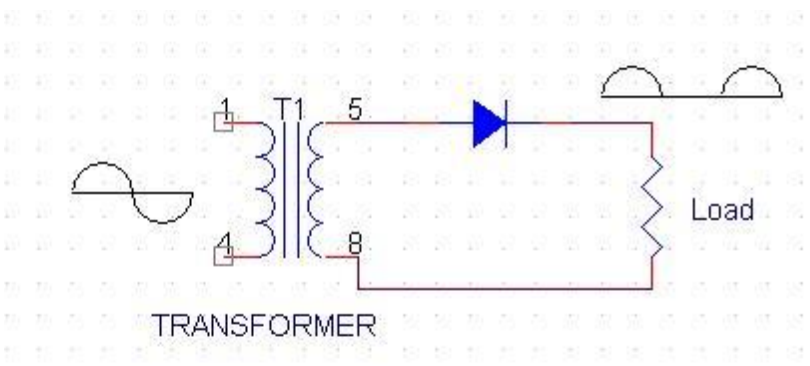
$$\frac{V_s}{V_p} = \frac{N_s}{N_p}$$

<u>Rectifier</u>

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.

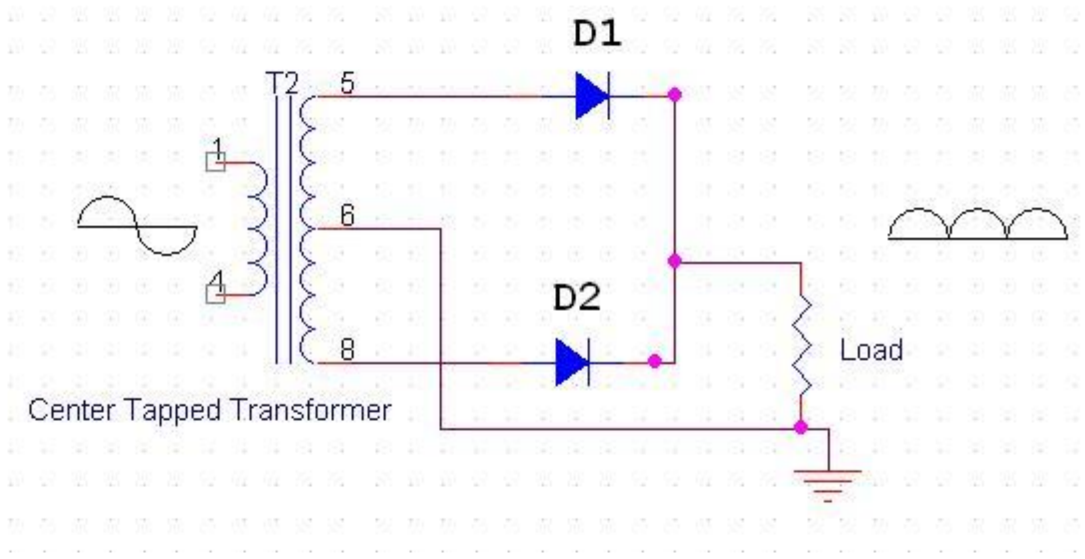
Rectifier can be classified as follows:

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.

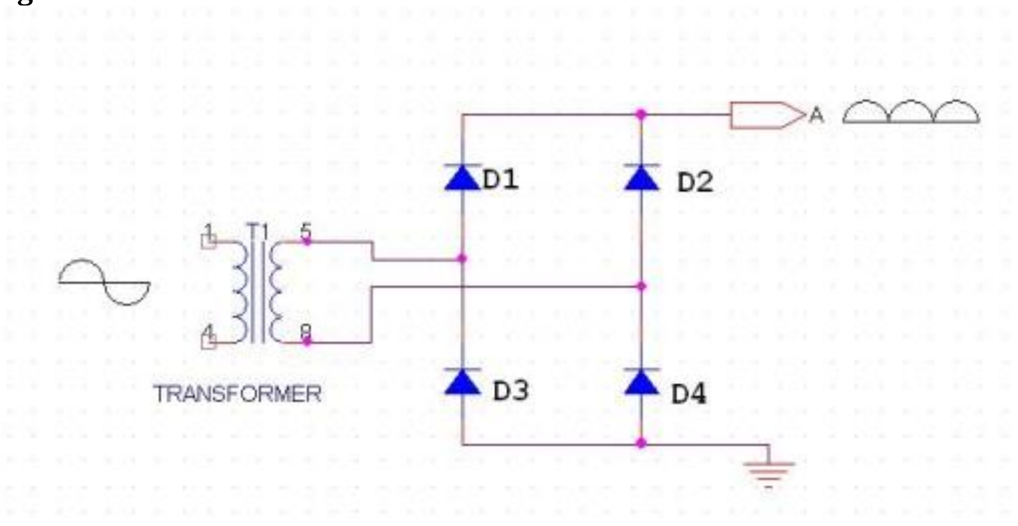
2) Full wave rectifier.



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.

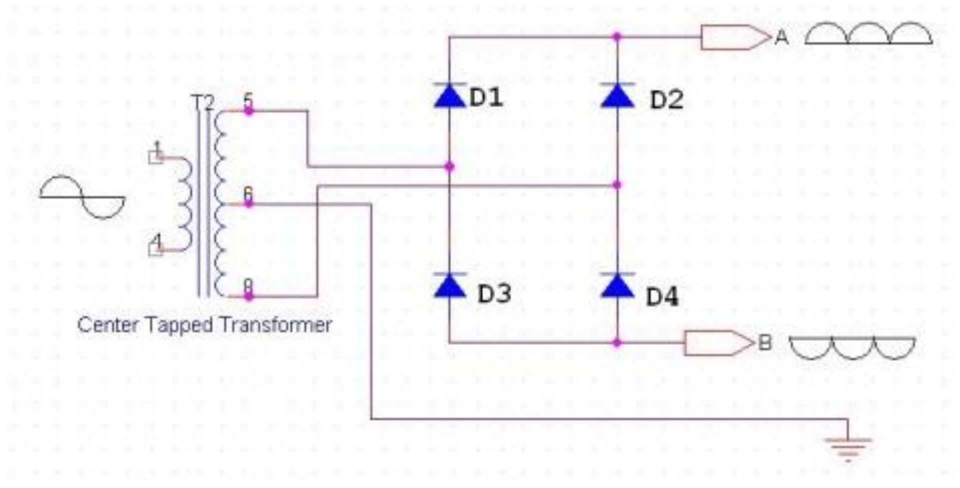
3) Bridge



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.

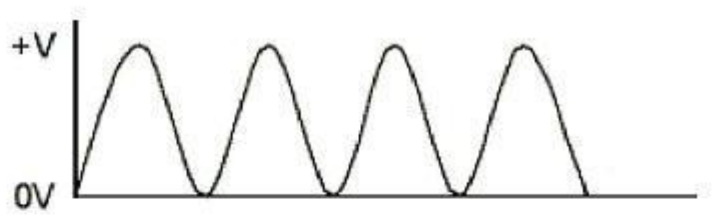
Full Bridge Wave Rectifier consists of four diodes namely D1, D2, D3 and D4. During the positive half cycle diodes D1 & D4 conduct whereas in the negative half cycle diodes D2 & D3 conduct thus the diodes keep switching the transformer connections so we get positive half cycles in the output.

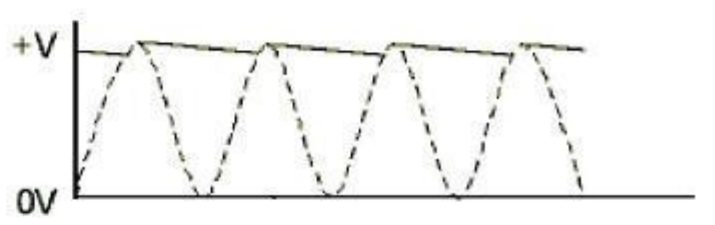


If we use a center tapped transformer for a bridge rectifier we can get both positive & negative half cycles which can thus be used for generating fixed positive & fixed negative voltages.

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

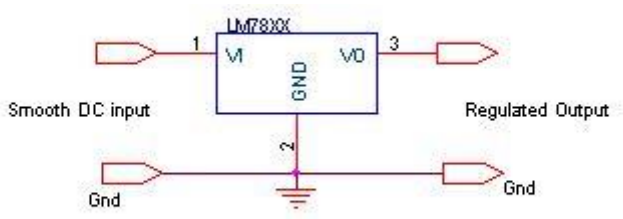
A Voltage regulator is a device which converts varying input voltage into a constant regulated output voltage. Voltage regulator can be of two types

1) Linear Voltage Regulator

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

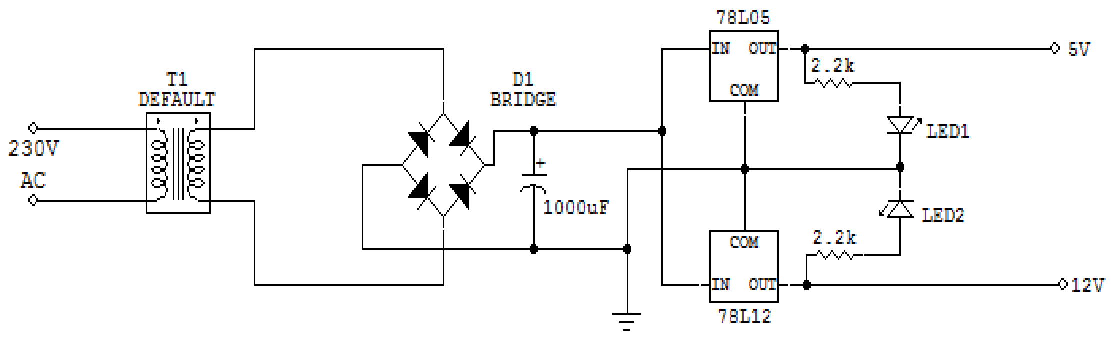
2) Switching Regulators.

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.

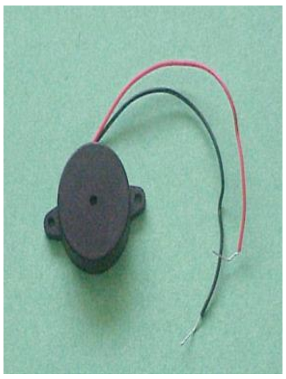
Table 2.1. Specifications of IC7805

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 BLUETOOTH

Bluetooth is an open wireless technology standard for exchanging data over short distances (using short wavelength radio transmissions) from fixed and mobile devices, creating personal area networks (PANs) with high levels of security. Created by telecoms vendor Ericsson in 1994 it was originally conceived as a wireless alternative to RS-232 data cables. It can connect several devices, overcoming problems of synchronization.

Bluetooth Overview

Why Bluetooth?

Bluetooth is a short-range radio link intended to replace the calble(s) connecting portable and/or fixed electronic devices. Key features are robustness, low complexity, low power and low cost [1]. There are already similar standards in this market, such as IrDA, HomeRF and IEEE 802.11 family. Bluetooth is designed to offer some unique advantages that none of the others can provide.

For example, IrDA uses infrared as medium, so its range is limited to around 1 meter, and it requires a line-of-sight communication. In comparison, Bluetooth can operate at a range up to 10 meters, or even 100 meters with enhanced transmitters. RF signals goes through walls, so a Bluetooth network can span several rooms.

Compared with HomeRF and IEEE 802.11 family, Bluetooth has much lower data rate and transmission range (10 meter). While HomeRF supports 1.6 ~ 10 Mbps data rate and IEEE 802.11a/b supports 54/11 Mbps, Bluetooth supports only 780 Kbps, which can be used for 721 kbps downstream and 57.6 kbps upstream asymmetric data transfer, or 432.6 kbps symmetric data transfer. Both HomeRF and IEEE 802.11 operates at 100 meter range, while Bluetooth operates at up to 10 meter.

However, as a result of the lower data rate and transmission range, Bluetooth offers much lower cost per node (approximately 5 ~ 10% of HomeRF and IEEE 802.11). So it is more suitable for applications involving low data rate (data and voice), small number of devices (8 at maximum), low power consumption and short range (up to 10 meter), such as PC-to-peripheral networking, home networking, hidden computing, data synchronization (such as between PC and PDA), mobile phone devices, and future smart devices or entertainment equipment.

BLUETOOTH ARCHITECTURE

The Bluetooth architecture and its mapping to OSI model

Bluetooth communication occurs between a master radio and a slave radio. Bluetooth radios are symmetric in that the same device may operate as a master and also the slave. Each radio has a 48-bit unique device address (BD_ADDR) that is fixed.

Two or more radio devices together form ad-hoc networks called piconets. All units within a piconet share the same channel. Each piconet has one master device and one or more slaves. There may be up to seven active slaves at a time within a piconet. Thus, each active device within a piconet is identifiable by a 3-bit active device address. Inactive slaves in unconnected modes may continue to reside within the piconet.

A master is the only one that may initiate a Bluetooth communication link. However, once a link is established, the slave may request a master/slave switch to become the master. Slaves are not allowed to talk to each other directly. All communication occurs within the slave and the master. Slaves within a piconet must also synchronize their internal clocks and frequency hops with that of the master. Each piconet uses a different frequency hopping sequence.

Radio devices used Time Division Multiplexing (TDM). A master device in a piconet transmits on even numbered slots and the slaves may transmit on odd numbered slots.

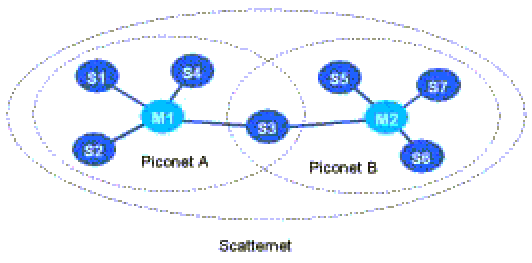
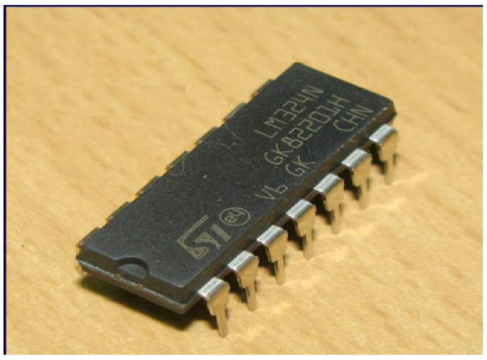


Fig 1: Bluetooth Scatternets and Piconets

Multiple piconets with overlapping coverage areas form a scatternet. Each piconet may have only one master, but slaves may participate in different piconets on a time-division multiplex basis. A device may be a master in one piconet and a slave in another or a slave in more than one piconet.

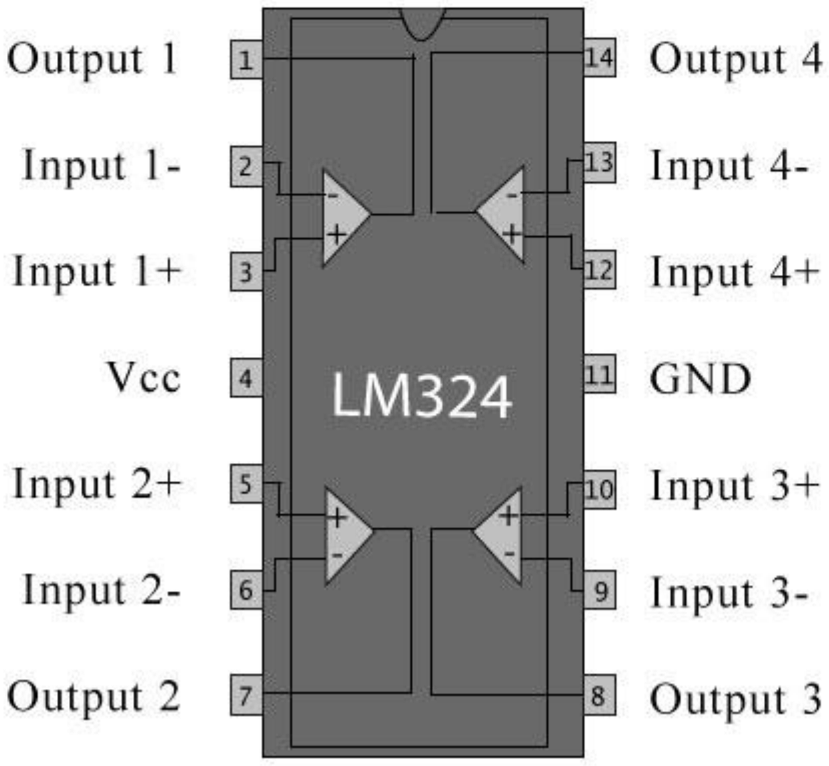
2.5 CONTINUITY SENSOR

LM324 is a 14pin IC consisting of four independent operational amplifiers (op-amps) compensated in a single package. Op-amps are high gain electronic voltage amplifier with differential input and, usually, a single-ended output. The output voltage is many times higher than the voltage difference between input terminals of an op-amp.



These op-amps are operated by a single power supply LM324 and need for a dual supply is eliminated. They can be used as amplifiers, comparators, oscillators, rectifiers etc.

The conventional op-amp applications can be more easily implemented with LM324.



Application areas include transducer amplifiers, DC gain blocks and all the conventional op amp circuits which now can be more easily implemented in single power supply systems. For example, the LM124 series can be directly operated off of the standard +5V power supply voltage which is used in digital systems and will easily provide the required interface electronics without requiring the additional ±15V power supplies.

Unique Characteristics

In the linear mode the input common-mode voltage range includes ground and the output voltage can also swing to ground, even though operated from only a single power supply voltage the unity gain cross frequency is temperature compensated. The input bias current is also temperature compensated

Advantages

Eliminates need for dual supplies four internally compensated op amps in a single package Allows directly sensing near GND and VOUT also goes to GND Compatible with all forms of logic Power drain suitable for battery operation.

Features

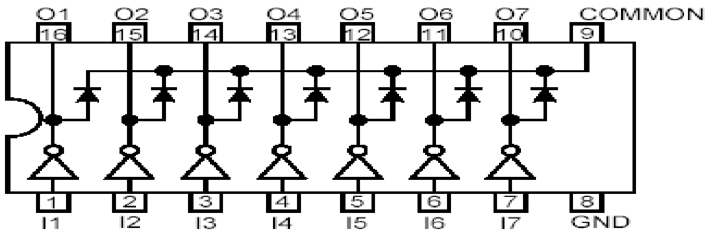
Internally frequency compensated for unity gain Large DC voltage gain 100 Db Wide and width (unity gain) 1 MHz (temperature compensated) n Wide power supply range:
Single supply 3V to 32V or dual supplies ±1.5V to ±16V Very low supply current drain (700 µA)—essentially. Independent of supply voltage Low input biasing current 45 nA (temperature compensated) Low input offset voltage 2 mV and offset current: 5 nA Input common-mode voltage range includes ground Differential input voltage range equal to the power supply voltage
Large output voltage swing 0V to V+ − 1.5V

2.6 ULN 2003

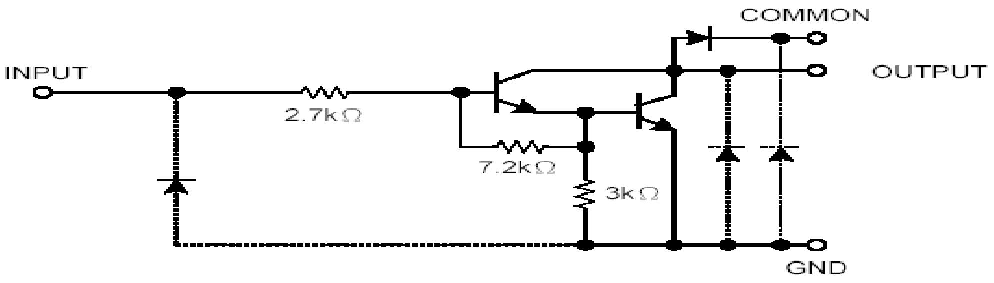
ULN2003

ULN is mainly suited for interfacing between low-level circuits and multiple peripheral power loads,.The series ULN20XX high voltage, high current darlington arrays feature continuous load current ratings. The driving circuitry in- turn decodes the coding and conveys the necessary data to the stepper motor, this module aids in the movement of the arm through steppers

■ **PIN CONNECTION**



■ **BLOCK DIAGRAM**

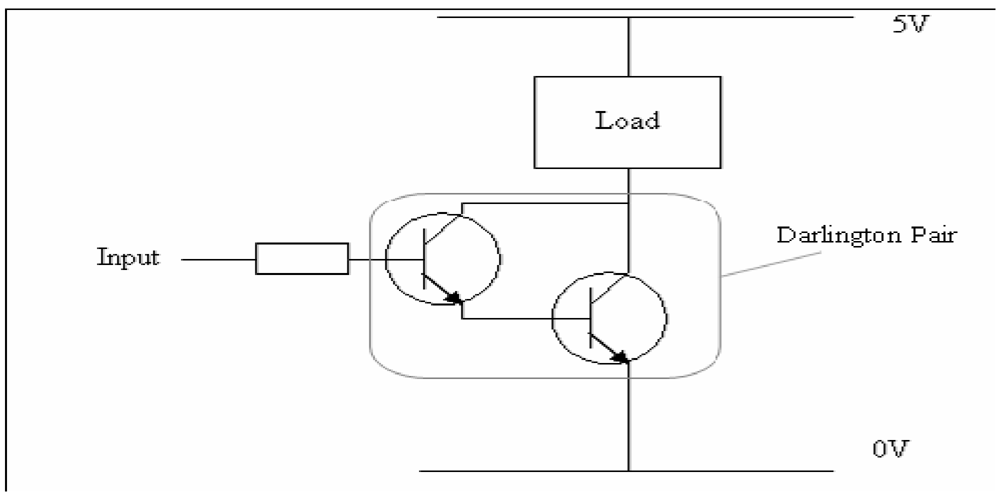


The driver makes use of the ULN2003 driver IC, which contains an array of 7 power Darlington arrays, each capable of driving 500mA of current. At an approximate duty cycle, depending on ambient temperature and number of drivers turned on, simultaneously typical power loads totaling over 230w can be controlled.

The device has base resistors, allowing direct connection to any common logic family. All the emitters are tied together and brought out to a separate terminal. Output protection diodes are included; hence the device can drive inductive loads with minimum extra components. Typical loads include relays, solenoids, stepper motors, magnetic print hammers, multiplexed LED, incandescent displays and heaters.

What is current gain?

Transistors have a characteristic called current gain. This is referred to as its hFE. The amount of current that can pass through the load when connected to a transistor that is turned on equals the input current x the gain of the transistor (hFE) The current gain varies for different transistor and can be looked up in the data sheet for the device. Typically it may be 100. This would mean that the current available to drive the load would be 100 times larger than the input to the transistor.



Why use a Darlington Pair?

In some application the amount of input current available to switch on a transistor is very low. This may mean that a single transistor may not be able to pass sufficient current required by the load.

As stated earlier this equals the input current x the gain of the transistor (hFE). If it is not be possible to increase the input current then we need to increase the gain of the transistor. This can be achieved by using a Darlington Pair.

A Darlington Pair acts as one transistor but with a current gain that equals:

Total current gain (hFE total) = current gain of transistor 1 (hFE t1) x current gain of transistor 2 (hFE t2)

So for example if you had two transistors with a current gain (hFE) = 100:

(hFE total) = 100 x 100

(hFE total) = 10,000

You can see that this gives a vastly increased current gain when compared to a single transistor. Therefore this will allow a very low input current to switch a much bigger load current.

Base Activation Voltage

Normally to turn on a transistor the base input voltage of the transistor will need to be greater that 0.7V. As two transistors are used in a Darlington Pair this value is doubled. Therefore the base voltage will need to be greater than 0.7V x 2 = 1.4V.

It is also worth noting that the voltage drop across collector and emitter pins of the Darlington Pair when the turn on will be around 0.9V Therefore if the supply voltage is 5V (as above) the voltage across the load will be will be around 4.1V (5V – 0.9V)

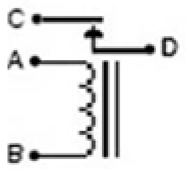
2.6 SPDT RELAY

Overview OF Relays

A relay is an electrically operated switch used to isolate one electrical circuit from another. In its simplest form, a relay consists of a coil used as an electromagnet to open and close switch contacts. Since the two circuits are isolated from one another, a lower voltage circuit can be used to trip a relay, which will control a separate circuit that requires a higher voltage or amperage. Relays can be found in early telephone exchange equipment, in industrial control circuits, in car audio systems, in automobiles, on water pumps, in high-power audio amplifiers and as protection devices.

Relay Switch Contacts

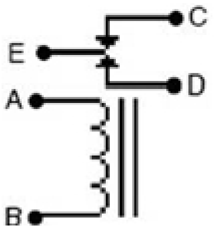
The switch contacts on a relay can be "normally open" (NO) or "normally closed" (NC)--that is, when the coil is at rest and not energized (no current flowing through it), the switch contacts are given the designation of being NO or NC. In an open circuit, no current flows, such as a wall light switch in your home in a position that the light is off. In a closed circuit, metal switch contacts touch each other to complete a circuit, and current flows, similar to turning a light switch to the "on" position. In the accompanying schematic diagram, points A and B connect to the coil. Points C and D connect to the



switch. When you apply a voltage across the coil at points A and B, you create an electromagnetic field, which attracts a lever in the switch, causing it to make or break contact in the circuit at points C and D (depending if the design is NO or NC). The switch contacts remain in this state until you remove the voltage to the coil. Relays come in different switch configurations. The switches may have more than one "pole," or switch contact. The diagram shows a "single pole single throw" configuration, referred to as SPST. This is similar to a wall light switch in your home. With a single "throw" of the switch, you close the circuit.

The Single Pole Double Throw Relay

A single pole double throw (SPDT) relay configuration switches one common pole to two other poles, flipping between them. As shown in the schematic diagram, the common point E completes a circuit with C when the relay coil is at rest, that is, no voltage is applied to it.



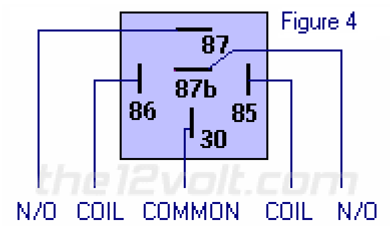
This circuit is "closed." A gap between the contacts of point E and D creates an "open" circuit. When you apply power to the coil, a metal level is pulled down, closing the circuit between points E and D and opening the circuit between E and C. A single pole double throw relay can be used to alternate which circuit a voltage or signal will be sent to.

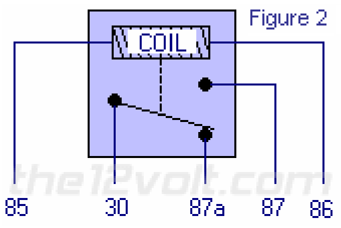
SPDT Relay:

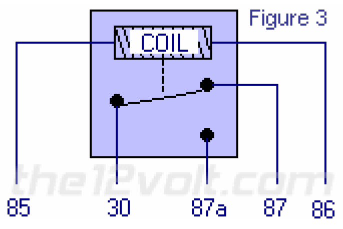
(**S**ingle **P**ole **D**ouble **T**hrow **Relay**) an electromagnetic switch, consist of a coil (terminals 85 & 86), 1 common terminal (30), 1 normally closed terminal (87a), and one normally open terminal (87) (Figure 1). When the coil of an SPDT relay (Figure 1) is at rest (not energized), the common terminal (30) and the normally closed terminal (87a) have continuity. When the coil is energized, the common terminal (30) and the normally open terminal (87) have continuity.

The diagram below center (Figure 2) shows an SPDT relay at rest, with the coil not energized. The diagram below right (Figure 3) shows the relay with the coil energized. As you can see, the coil is an electromagnet that causes the arm that is always connected to the common (30) to pivot when energized whereby contact is broken from the normally closed terminal (87a) and made with the normally open terminal (87).

When energizing the coil of a relay, polarity of the coil does not matter unless there is a diode across the coil. If a diode is not present, you may attach positive voltage to either terminal of the coil and negative voltage to the other, otherwise you must connect positive to the side of the coil that the cathode side (side with stripe) of the diode is connected and negative to side of the coil that the anode side of the diode is connected.







Why do I want to use a relay and do I really need to?

Anytime you want to switch a device which draws more current than is provided by an output of a switch or component you'll need to use a relay. The coil of an **SPDT** or an **SPST** relay that we most commonly use draws very little current (less than 200 milliamps) and the amount of current that you can pass through a relay's common, normally closed, and normally open contacts will handle up to 30 or 40 amps. This allows you to switch devices such as headlights, parking lights, horns, etc., with low amperage outputs such as those found on keyless entry and alarm systems, and other components. In some cases you may need to switch multiple things at the same time using one output. A single output connected to multiple relays will allow you to open continuity and/or close continuity simultaneously on multiple wires.

There are far too many applications to list that require the use of a relay, but we do show many of the most popular applications in the pages that follow and many more in our Relay Diagrams - Quick Reference application. If you are still unclear about what a relay does or if you should use one after you browse through the rest of this section, please post a question in the12volt's install bay. (We recommend Tyco (formerly Bosch) or Potter & Brumfield relays for all of the SPDT and SPST relay applications shown on this site.)

2.8 LCD

To display interactive messages we are using LCD Module. We examine an intelligent LCD display of two lines,16 characters per line that is interfaced to the controllers. The protocol (handshaking) for the display is as shown. Whereas D0 to D7th bit is the Data lines, RS, RW and EN pins are the control pins and remaining pins are +5V, -5V and GND to provide supply. Where RS is the Register Select, RW is the Read Write and EN is the Enable pin.

The display contains two internal byte-wide registers, one for commands (RS=0) and the second for characters to be displayed (RS=1). It also contains a user-programmed RAM area (the character RAM) that can be programmed to generate any desired character that can be formed using a dot matrix. To distinguish between these two data areas, the hex command byte 80 will be used to signify that the display RAM address 00h will be chosen.Port1 is used to furnish the command or data type, and ports 3.2 to3.4 furnish register select and read/write levels.

The display takes varying amounts of time to accomplish the functions as listed. LCD bit 7 is monitored for logic high (busy) to ensure the display is overwritten.

Liquid Crystal Display also called as LCD is very helpful in providing user interface as well as for debugging purpose. The most common type of LCD controller is HITACHI 44780 which provides a simple interface between the controller & an LCD. These LCD's are very simple to interface with the controller as well as are cost effective.

2x16 Line Alphanumeric LCD Display

The most commonly used *ALPHANUMERIC* displays are *1x16* (Single Line & 16 characters), *2x16* (Double Line & 16 character per line) & *4x20* (four lines & Twenty characters per line).

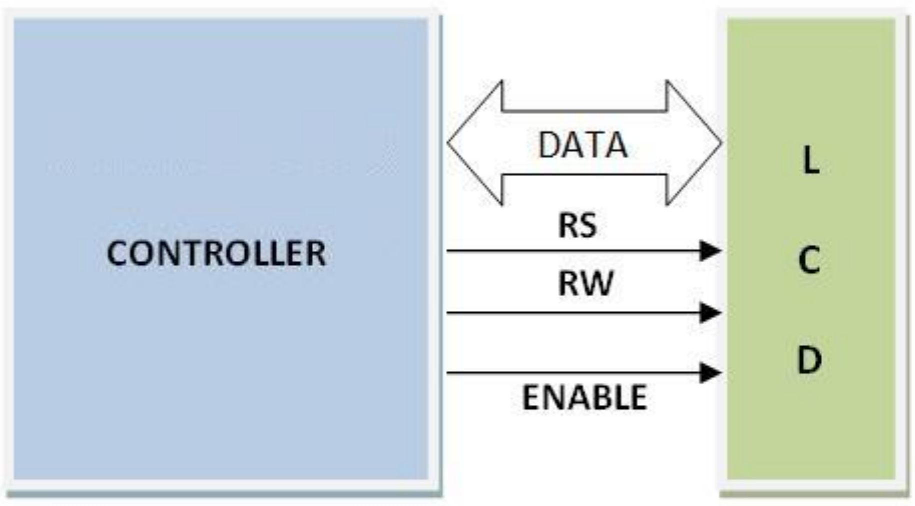
The LCD requires 3 control lines (RS, R/W & EN) & 8 (or 4) data lines. The number on data lines depends on the mode of operation. If operated in 8-bit mode then 8 data lines + 3 control lines i.e. total 11 lines are required. And if operated in 4-bit mode then 4 data lines + 3 control lines i.e. 7 lines are required. How do we decide which mode to use? It's simple if you have sufficient data lines you can go for 8 bit mode & if there is a time constrain i.e. display should be faster then we have to use 8-bit mode because basically 4-bit mode takes twice as more time as compared to 8-bit mode.

Pin	Symbol	Function
1	Vss	Ground
2	Vdd	Supply Voltage
3	Vo	Contrast Setting
4	RS	Register Select
5	R/W	Read/Write Select
6	En	Chip Enable Signal
7-14	DB0-DB7	Data Lines
15	A/Vee	Gnd for the backlight
16	K	Vcc for backlight

When *RS* is low (0), the data is to be treated as a command. When RS is high (1), the data being sent is considered as text data which should be displayed on the screen.

When *R/W* is low (0), the information on the data bus is being written to the LCD. When RW is high (1), the program is effectively reading from the LCD. Most of the times there is no need to read from the LCD so this line can directly be connected to Gnd thus saving one controller line.

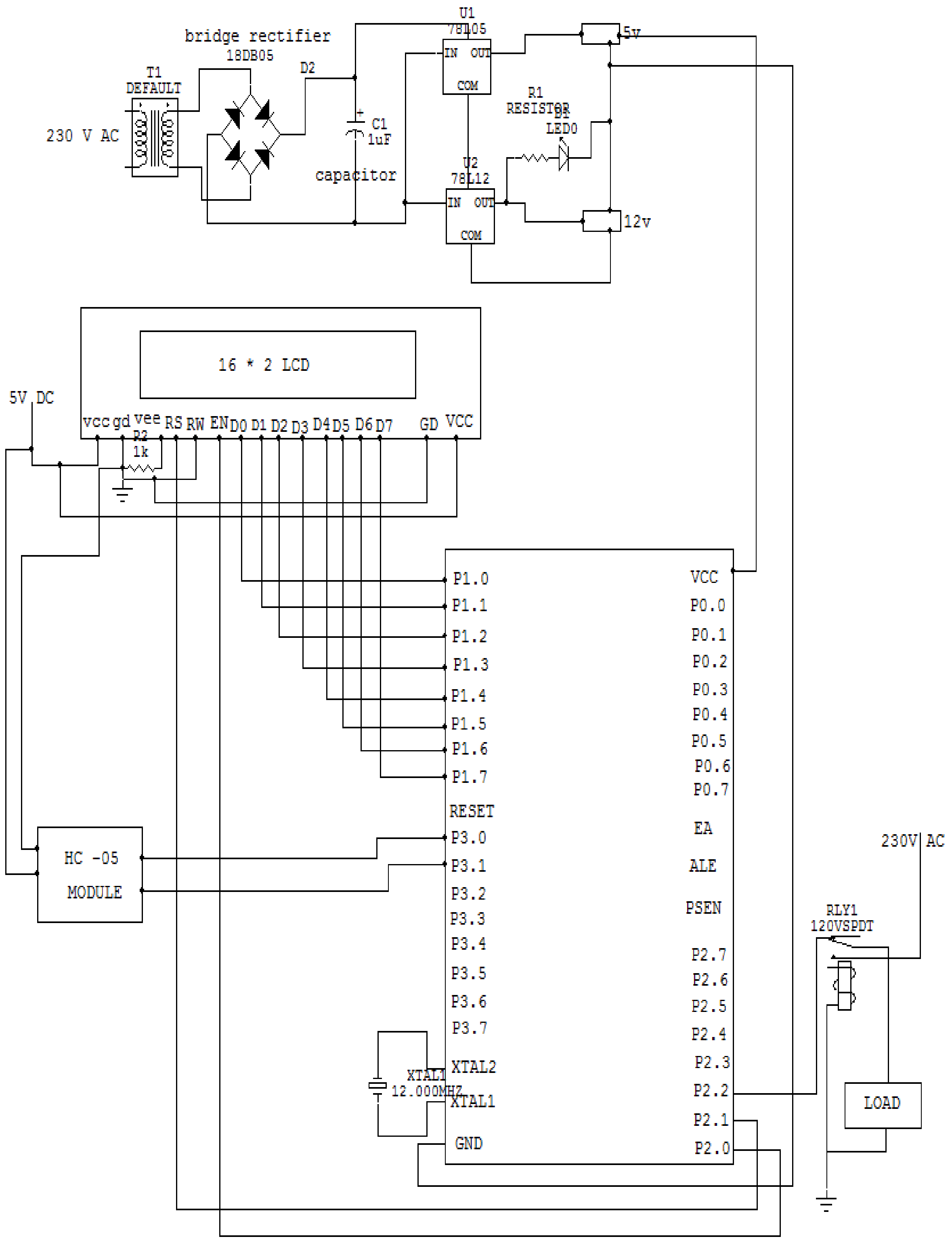
The *ENABLE* pin is used to latch the data present on the data pins. A HIGH - LOW signal is required to latch the data. The LCD interprets and executes our command at the instant the EN line is brought low. If you never bring EN low, your instruction will never be executed.


CONTROLLER
DATA
RS
RW
ENABLE
L
C
D

CHAPTER 3

CIRCUITS AND THEIR OPERATION

3.1 Circuit diagram



2.1 source code

```
#include <reg51.H>
 sbit line1=P2^0;
 sbit line2=P2^1;
 sbit buzzer=P2^2;

 sbit red1=P3^4;
 sbit green1=P3^5;
 sbit red2=P3^6;
 sbit green2=P3^7;

 sfr ldata=0X90;

sbit rs=P2^6;
sbit en=P2^7;



void transmit(char temp[]);
receive();



void delay(unsigned int t);



unsigned int t=50;
  unsigned int j;
   unsigned char rx;

 void lcd_str(unsigned char temp[]);
 void lcdcmd(unsigned char);
void lcddata(unsigned char);

void main()
{
     TMOD=0X20;
     SCON=0X50;
     TH1=-3;
     TR1=1;


lcdcmd(0x38);
lcdcmd(0x0E);
lcdcmd(0x01);
lcdcmd(0x80);
lcd_str("TRANSMSSION LINE");
lcdcmd(0xC0);
lcd_str("SECURITY SYSTERM");

transmit("Transmission    line    monitoring    and    security
system\r\n");

green1=green2=1;
red1=red2=buzzer=0;


while(1)
{

if(line1==1)
{
 lcdcmd(0x01);
 lcdcmd(0x80);
 lcd_str("FAULT OCCURED IN");
  lcdcmd(0xc0);
 lcd_str("LINE1 TRANSMISSN");
 transmit("Fault Occurred in Line1 Transmission... Please take
action immediately\r\n");
```

```
  ///transmit("Doors  Opened  and  Sprinklers  switched  on...
please Move out of train\r\n");
buzzer=1;
red1=1;
green1=0;
 }
 else
 {
  buzzer=0;
red1=0;
green1=1;
   }

   if(line2==1)
{
 lcdcmd(0x01);
 lcdcmd(0x80);
 lcd_str("FAULT OCCURED IN");
  lcdcmd(0xc0);
 lcd_str("LINE2 TRANSMISSN");
 transmit("Fault Occurred in Line2 Transmission... Please take
action immediately\r\n");
  ///transmit("Doors  Opened  and  Sprinklers  switched  on...
please Move out of train\r\n");
buzzer=1;
red2=1;
green2=0;
 }
 else
 {
  buzzer=0;
red2=0;
green2=1;
   }
```

```
}
}
void delay(unsigned int t)
{
unsigned int a,b;
for(a=0;a<t;a++)
for(b=0;b<1275;b++);
}
```

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:

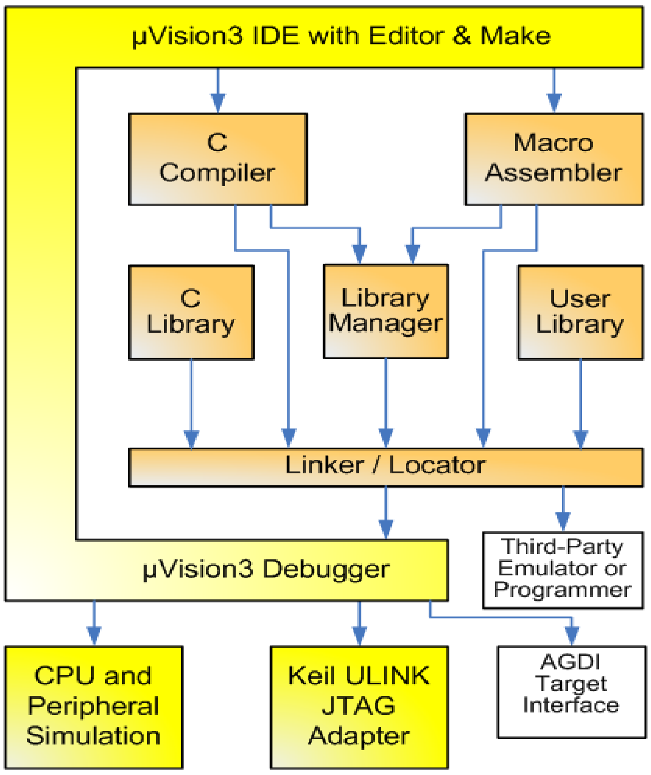


Figure 5.1 Keil Software- internal stages

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

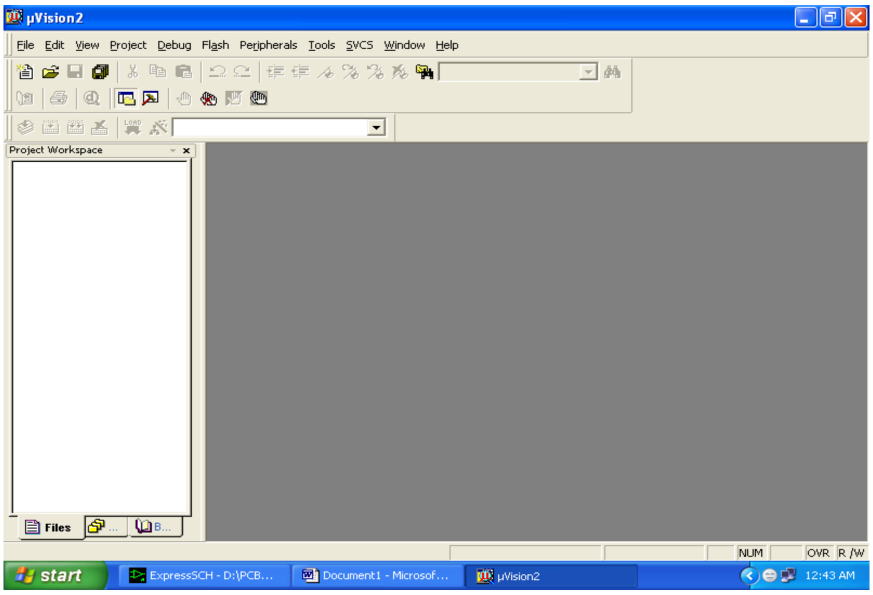
The Disassembly window shows your target program as mixed source and assembly program or just assembly code. A trace history of previously executed instructions may be displayed with Debug – View Trace Records. To enable the trace history, set Debug – Enable/Disable Trace Recording.

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.

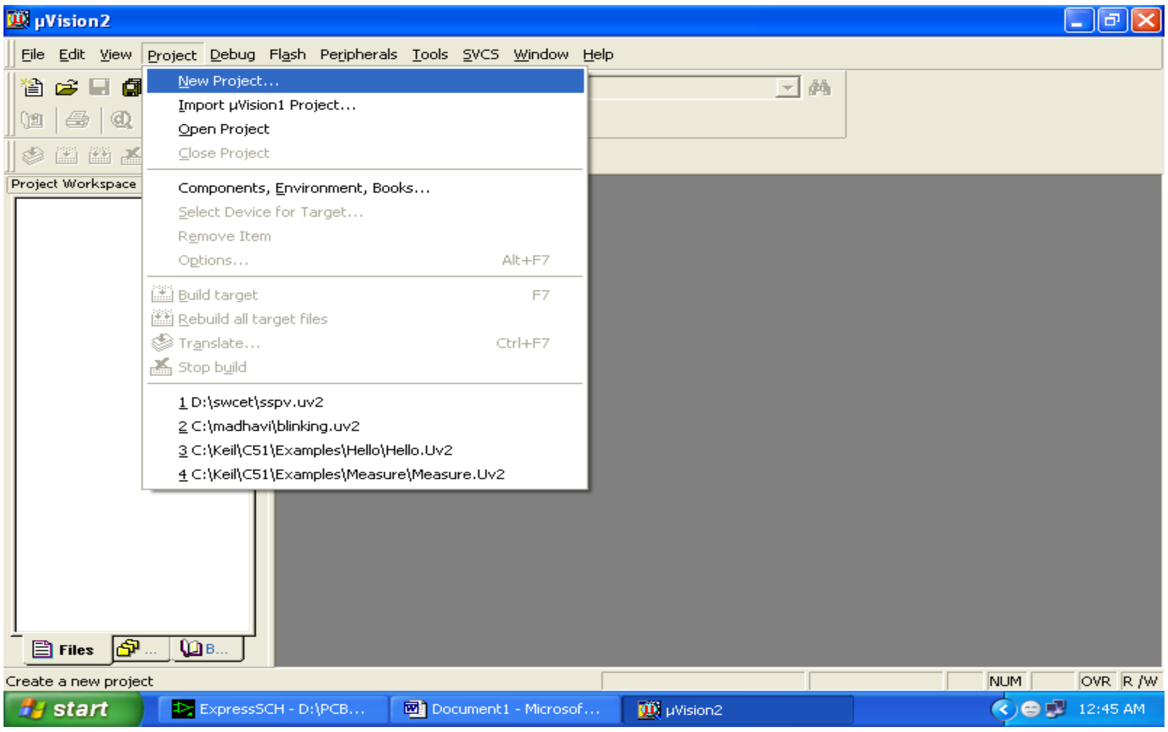
You may use the dialog Debug – Inline Assembly… to modify the CPU instructions. That allows you to correct mistakes or to make temporary changes to the target program you are debugging.

5.5 OVERVIEW OF KEIL UVISION SOFTWARE

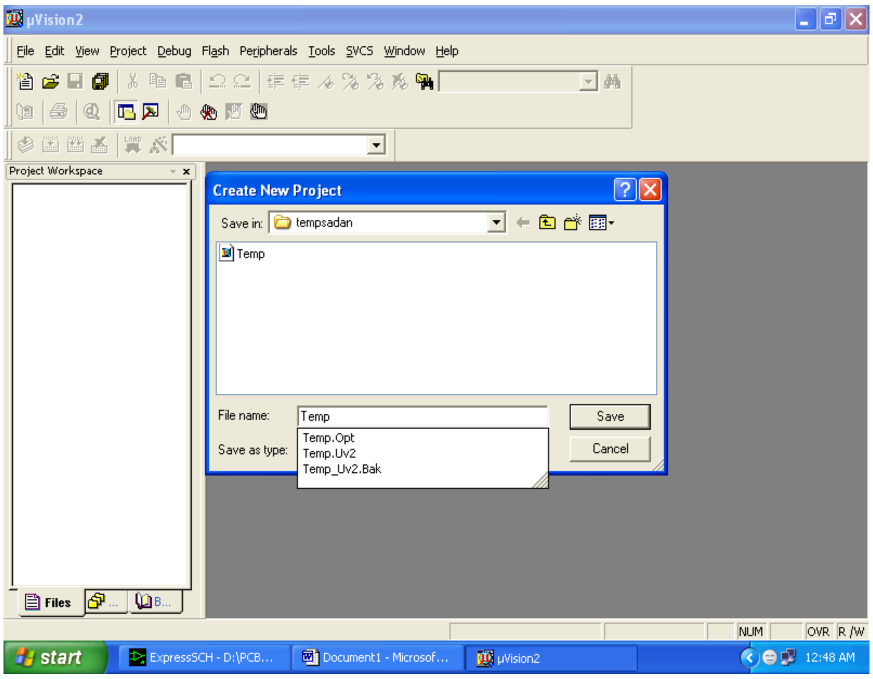
1. Click on the Keil uVision Icon on Desktop
2. The following fig will appear



3. Click on the Project menu from the title bar
4. Then Click on New Project



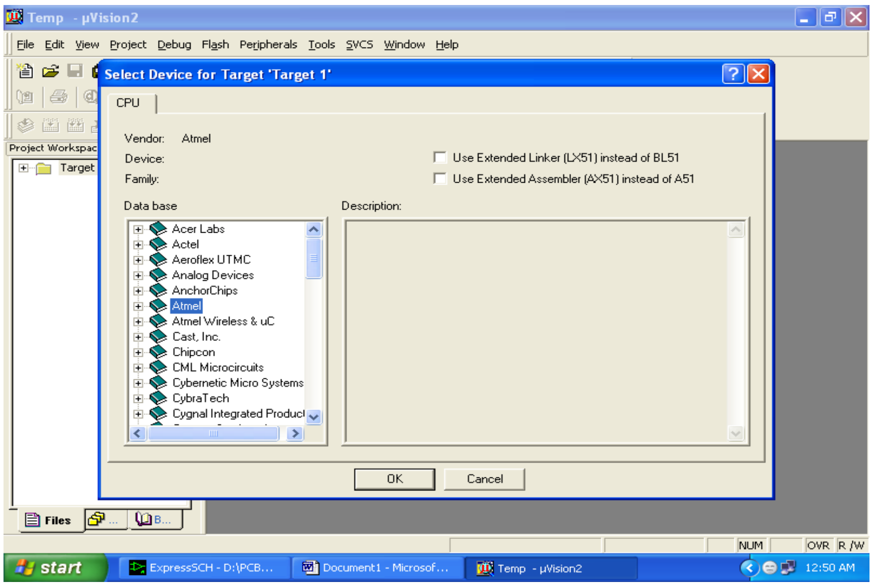
5. Save the Project by typing suitable project name with no extension in u r own folder sited in either C:\ or D:\



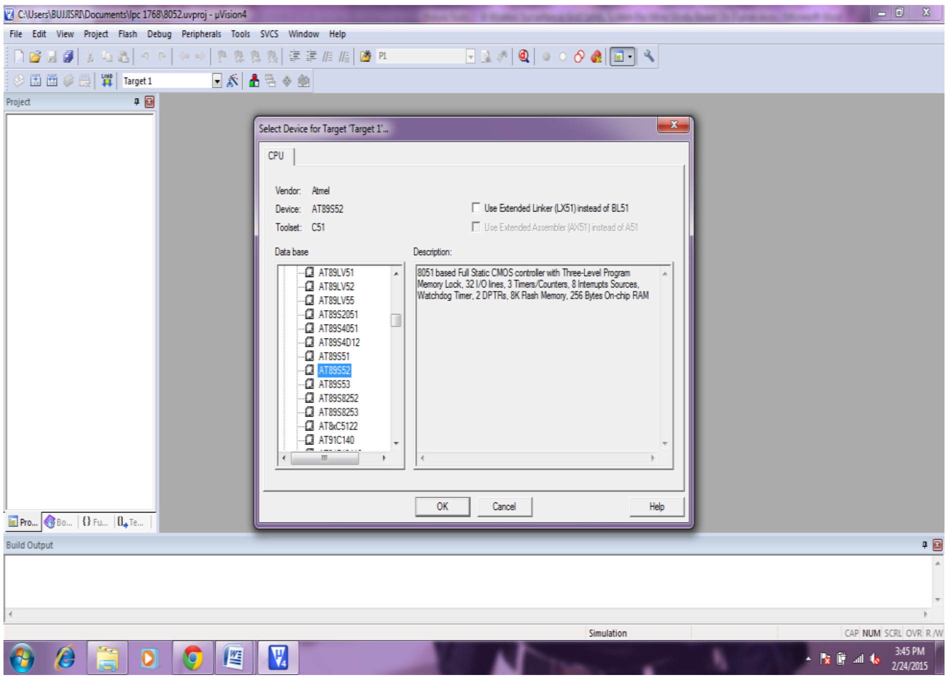
6. Then Click on Save button above.

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


Select Device for Target 'Target 1'...

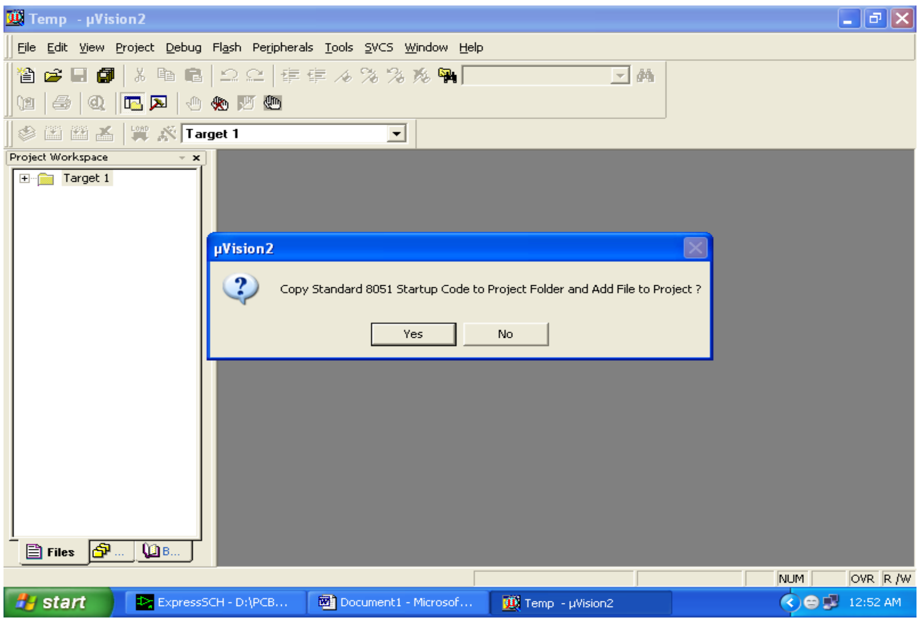
Vendor: Atmel
Device: AT89S52
Toolset: C51

Use Extended Linker (LX51) instead of BL51
Use Extended Assembler (AX51) instead of A51

Data base: AT89LV51, AT89LV52, AT89LV55, AT89S2051, AT89S4051, AT89S4D12, AT89S51, AT89S52, AT89S53, AT89S8252, AT89S8253, AT8xC5122, AT91C140

Description:
8051 based Full Static CMOS controller with Three-Level Program Memory Lock, 32 I/O lines, 3 Timers/Counters, 8 Interrupts Sources, Watchdog Timer, 2 DPTRs, 8K Flash Memory, 256 Bytes On-chip RAM

10. Then Click on “OK”
11. The Following fig will appear

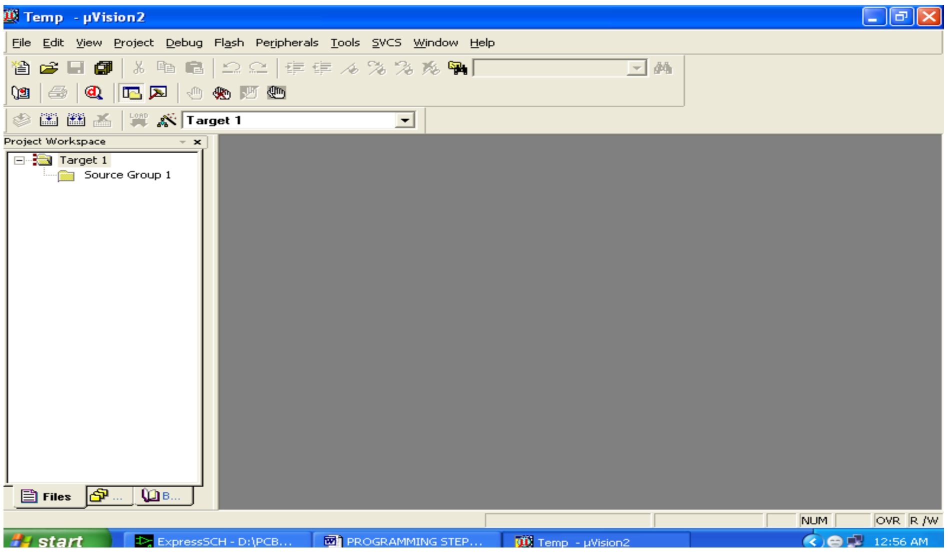

Temp - µVision2

µVision2

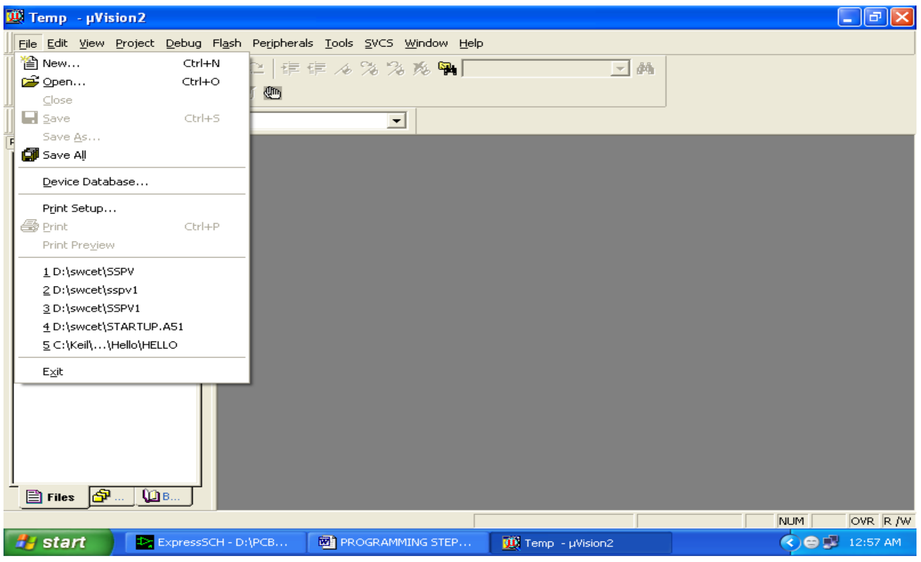
Copy Standard 8051 Startup Code to Project Folder and Add File to Project ?

12. Then Click either YES or NO………mostly “NO”

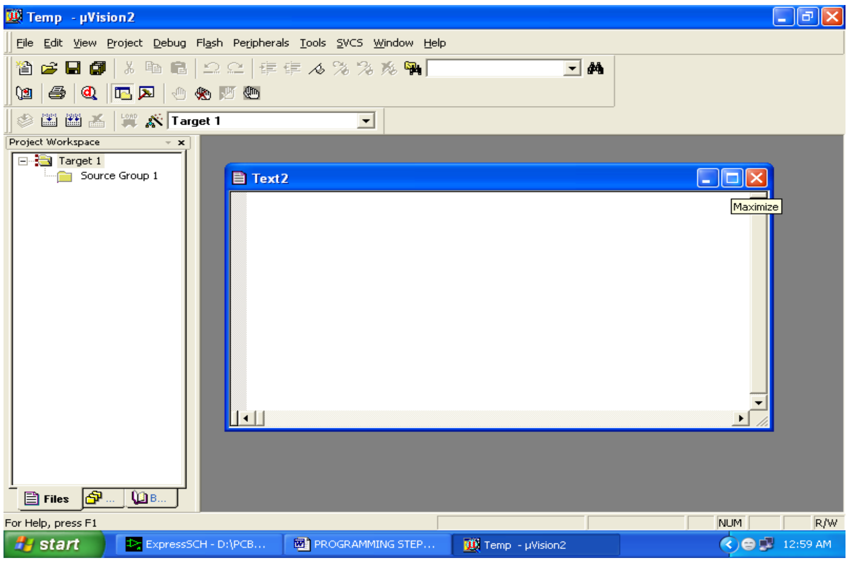
13. Now your project is ready to USE
14. Now double click on the Target1, you would get another option “Source group 1” as shown in next page.



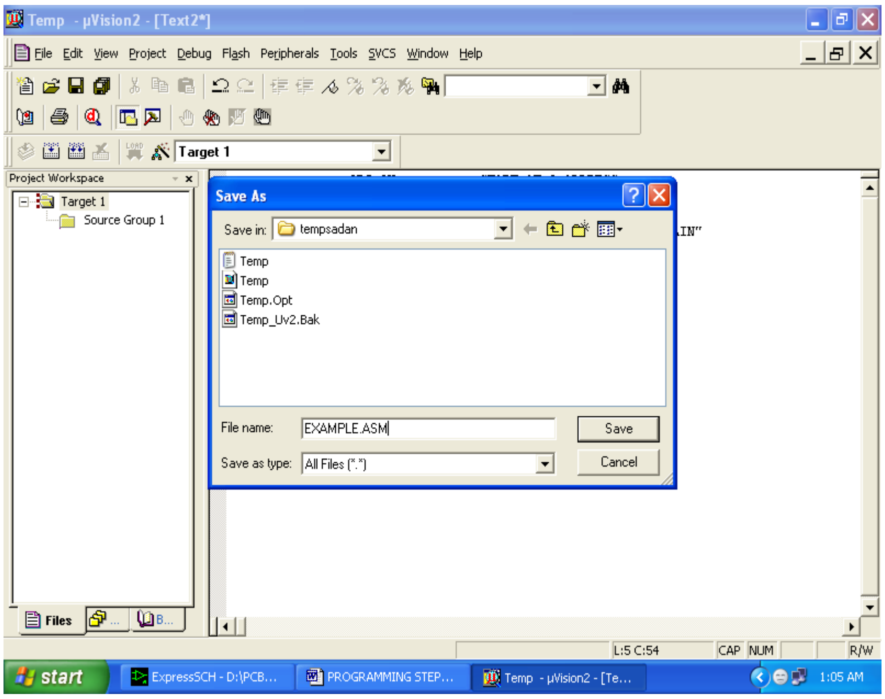
15. Click on the file option from menu bar and select “new”



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”

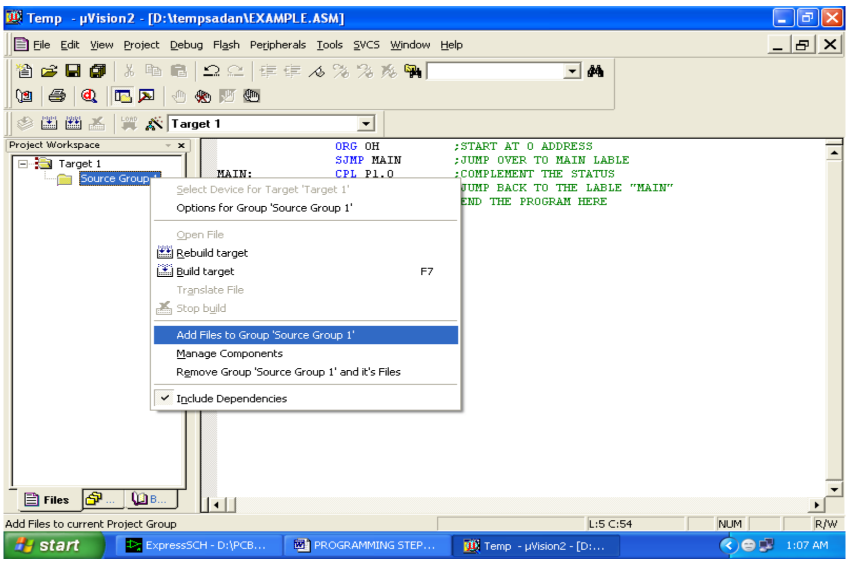


Save in: tempsadan

File name: EXAMPLE.ASM

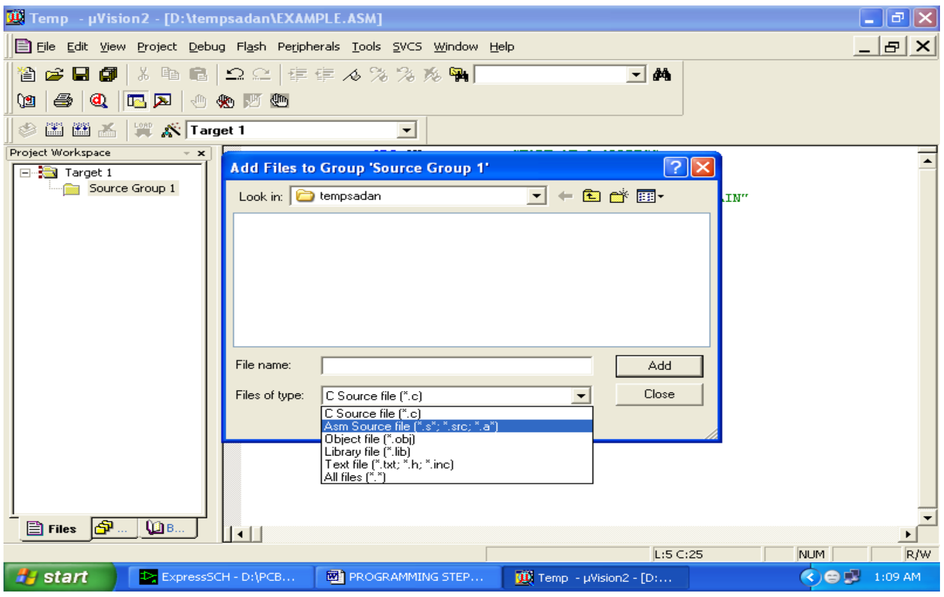
Save as type: All Files (*.*)

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


Temp - µVision2 - [D:\tempsadan\EXAMPLE.ASM]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Target 1
Project Workspace
Target 1
Source Group 1
Select Device for Target 'Target 1'
Options for Group 'Source Group 1'
Open File
Rebuild target
Build target F7
Translate File
Stop build
Add Files to Group 'Source Group 1'
Manage Components
Remove Group 'Source Group 1' and it's Files
Include Dependencies
Files
Add Files to current Project Group
L:5 C:54
NUM
R/W
start
ExpressSCH - D:\PCB...
PROGRAMMING STEP...
Temp - µVision2 - [D:...
1:07 AM

```
            ORG OH          ;START AT 0 ADDRESS
            SJMP MAIN       ;JUMP OVER TO MAIN LABLE
MAIN:       CPL P1.0        ;COMPLEMENT THE STATUS
                            JUMP BACK TO THE LABLE "MAIN"
                            END THE PROGRAM HERE
```

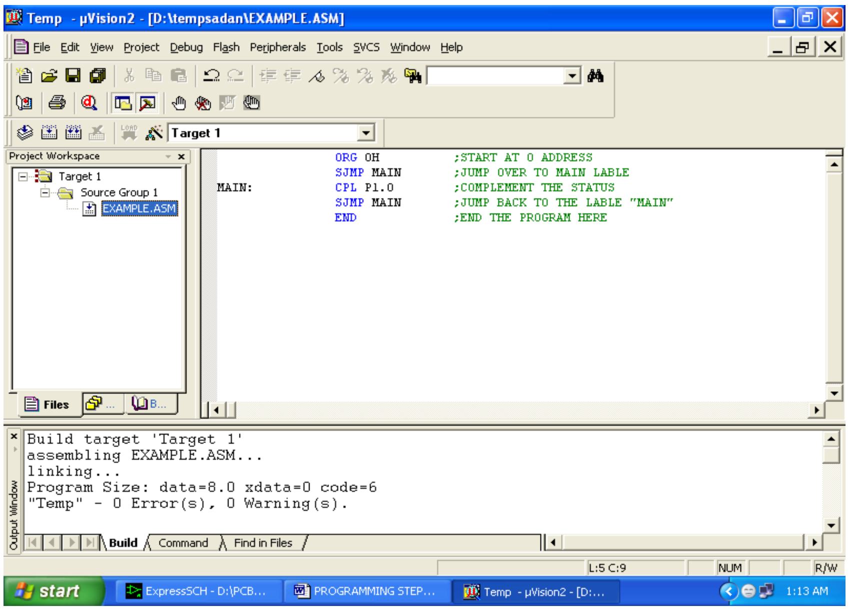
20. Now you will get another window, on which by default “C” files will appear.



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.

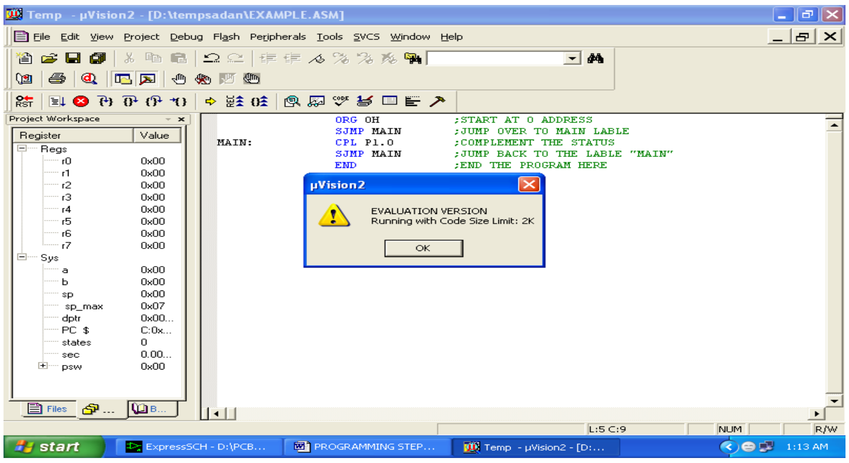

Temp - µVision2 - [D:\tempsadan\EXAMPLE.ASM]

```
        ORG OH          ;START AT 0 ADDRESS
        SJMP MAIN       ;JUMP OVER TO MAIN LABLE
MAIN:   CPL P1.0        ;COMPLEMENT THE STATUS
        SJMP MAIN       ;JUMP BACK TO THE LABLE "MAIN"
        END             ;END THE PROGRAM HERE
```

```
Build target 'Target 1'
assembling EXAMPLE.ASM...
linking...
Program Size: data=8.0 xdata=0 code=6
"Temp" - 0 Error(s), 0 Warning(s).
```

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

25. The new window is as follows


Temp - µVision2 - [D:\tempsadan\EXAMPLE.ASM]

```
        ORG OH          ;START AT 0 ADDRESS
        SJMP MAIN       ;JUMP OVER TO MAIN LABLE
MAIN:   CPL P1.0        ;COMPLEMENT THE STATUS
        SJMP MAIN       ;JUMP BACK TO THE LABLE "MAIN"
        END             ;END THE PROGRAM HERE
```

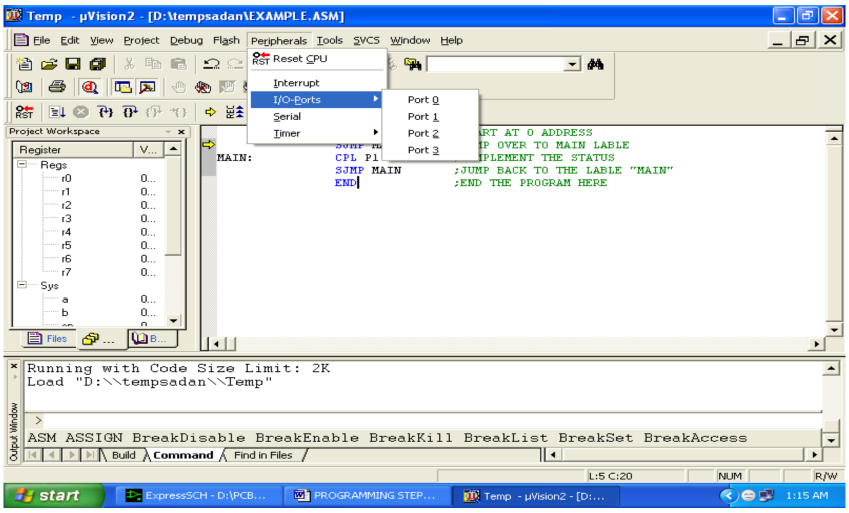
Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x00
sp_max	0x07
dptr	0x00...
PC $	C:0x...
states	0
sec	0.00...
psw	0x00

µVision2

EVALUATION VERSION
Running with Code Size Limit: 2K

26. Then Click “OK”

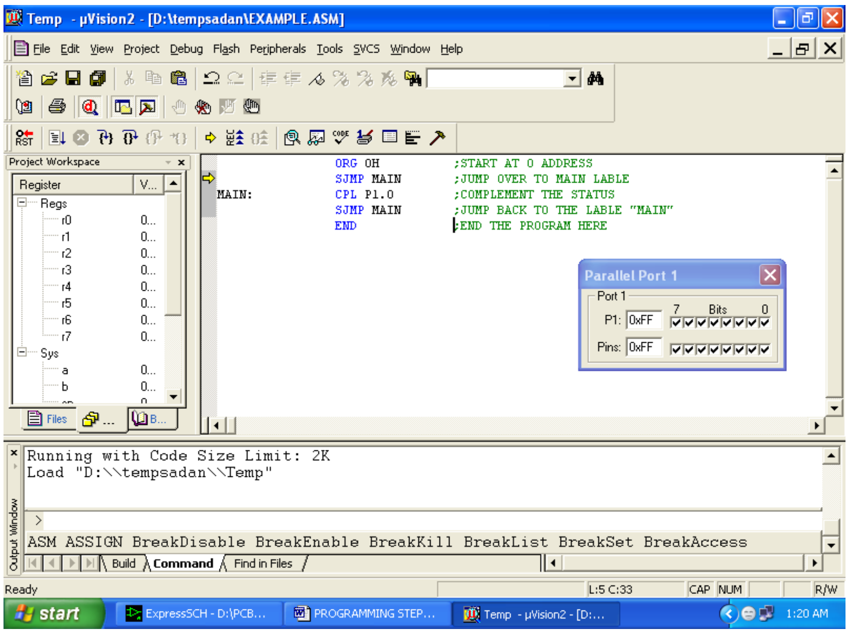
27. Now Click on the Peripherals from menu bar, and check your required port as shown in fig below


Temp - µVision2 - [D:\tempsadan\EXAMPLE.ASM]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Reset CPU
Interrupt
I/O-Ports
Serial
Timer
Port 0
Port 1
Port 2
Port 3
Project Workspace
Register
Regs
Sys

```
                              ...ART AT 0 ADDRESS
                              ...P OVER TO MAIN LABLE
MAIN:          CPL  P1        ...PLEMENT THE STATUS
               SJMP MAIN      ;JUMP BACK TO THE LABLE "MAIN"
               END            ;END THE PROGRAM HERE
```

```
Running with Code Size Limit: 2K
Load "D:\\tempsadan\\Temp"
>
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess
```

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


Temp - µVision2 - [D:\tempsadan\EXAMPLE.ASM]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Project Workspace
Register
Regs
Sys

```
               ORG  0H        ;START AT 0 ADDRESS
               SJMP MAIN      ;JUMP OVER TO MAIN LABLE
MAIN:          CPL  P1.0      ;COMPLEMENT THE STATUS
               SJMP MAIN      ;JUMP BACK TO THE LABLE "MAIN"
               END            ;END THE PROGRAM HERE
```

Parallel Port 1
Port 1
P1: 0xFF Bits 7 ... 0
Pins: 0xFF

```
Running with Code Size Limit: 2K
Load "D:\\tempsadan\\Temp"
>
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess
```

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. Devices are enabling by wireless communication.
2. Coverage area is large compared to existing system.
3. Less number of components and manual observations. so it is economically reliable and low cost.

DISADVANTAGES:

1. It transmits small area due to the range of Bluetooth.

2. Does not work properly in congested network.

3. Cannot find out the exact number of fault nodes.

4. Increases time latency in fault detection.

5. Difficult in deciding the parameter.

APPLICATIONS:

1. Used In Traffic Light Controlling System.

2. Used in transmission line.

3. Used in textile mills.

4. Used in food industry.

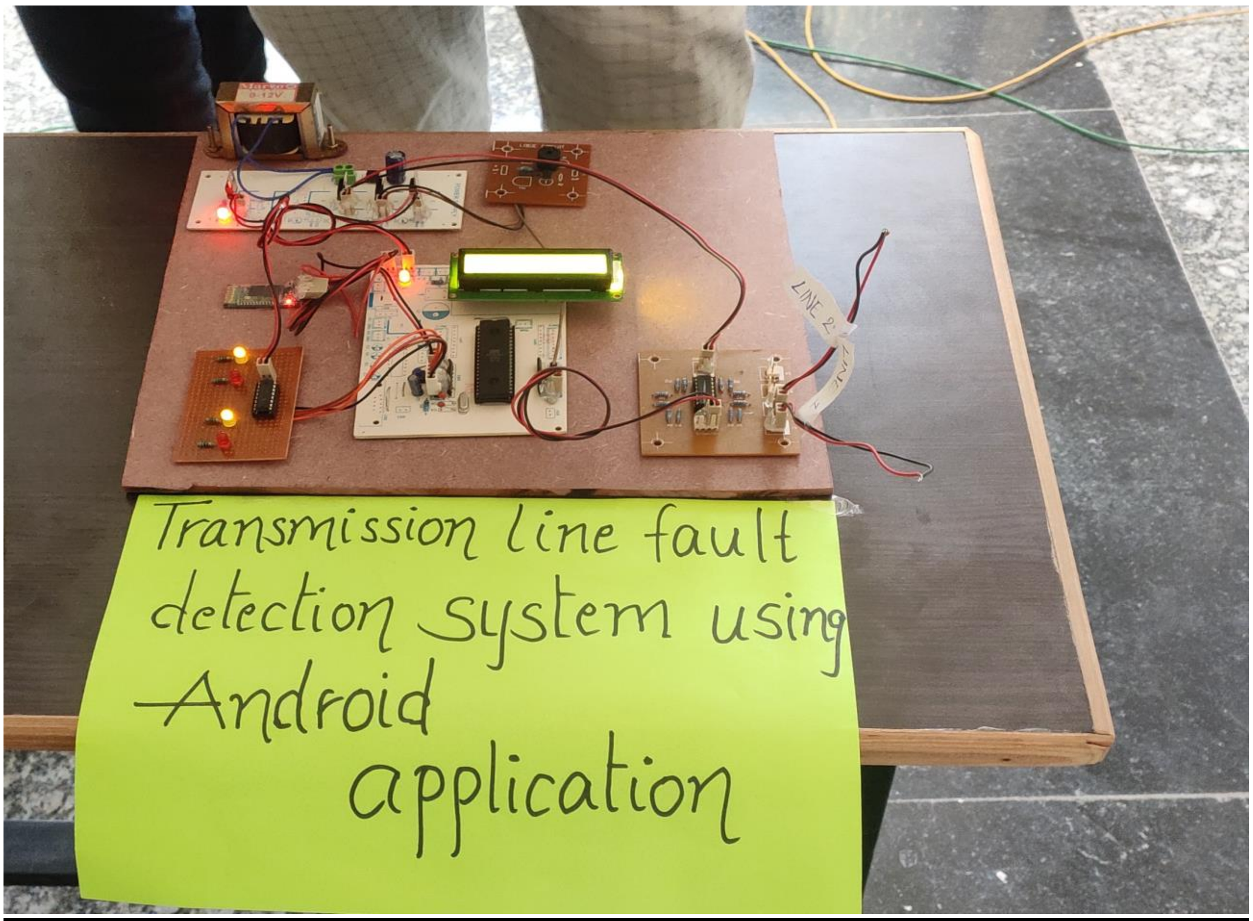
5. This Can Be Implemented In Medical Field For Protection Of Patient With Authorization Control System.

Conclusion

"TRANSMISSION LINE FAULT DETECTION USING ANDROID APPLICATION VIA BLUETOOTH"

Microcontroller And Bluetooth Based Fault Detection System Is A Reliable Technique For Monitoring And Controlling The Electric Distribution System, The Microcontroller Works Up To 100ºC Temperature. For Short Distance Data Transmission Bluetooth Technology Is A Reliable And Robust One. Any Kind Of Fault Occurring In The Distribution System Results The Bluetooth Modules To Send Instant Messages Automatically To The Nearest User. Bluetooth Based Microcontroller Fault Detection System Will Serve As A Reliable, Easy And Cost Effective Solution For Monitoring And Controlling The Electric Distribution System.

Project Kit Output


Transmission line fault detection system using Android application
LINE 2

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