VOICE OPERATED HOME APPLIANCES

Project report submitted in partial fulfillment for the Degree of

Bachelor of Technology

in

Electronics and Communication Engineering

Under the supervision of

Mr. VIKAS HASTIR

By

SANDEEP SINGH	101046
HARSHIT AGGARWAL	101071
RAMAN GUPTA	101082

May – 2014

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY,

WAKNAGHAT

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY,
(Established by H.P. State Legislature vide Act No. 14 of 2002)
P.O. Waknaghat, Teh. Kandaghat, Distt. Solan – 173234 (H.P.) INDIA
Website: www.juit.ac.in

CERTIFICATE

This is to certify that the project report entitled "**Voice Operated Home Appliances**" submitted by Sandeep Singh, Harshit Aggarwal and Raman Gupta in the partial fulfillment for award of degree of Bachelor of Technology in Electronics and Communication Engineering to Jaypee University of Information Technology, Waknaghat has been carried out under my supervision and guidance.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Date:

Mr. Vikas Hastir
(Assistant Professor)

…………………..

ACKNOWLEDGEMENT

At this level of understanding it is often difficult to understand the wide spectrum of knowledge without proper guidance and advice. Hence, we take this opportunity to express our heartfelt gratitude to our project guide **Mr. Vikas Hastir** who had faith in us and allowed us to work on this project. We owe him a great debt of gratitude for without his support, this work wouldn't have been accomplished, indeed. We just have no words to express our obligation for this learned and noble scholar.

We are equally indebted to all the staff members of Electronics and Communication department of our University. This project work has been greatly assisted by the cooperation of project lab staff that provided full support and facilities.

We have tried to produce the very best out of our endeavor.

Sandeep Singh (101046) ……………

Harshit Aggarwal (101071) ……………

Raman Gupta (101082) ……………

TALBLE OF CONTENTS

Title	Page No.
ACKNOWLEDGEMENT	**ii**
LIST OF FIGURES	**iv**
LIST OF TABLES	**v**
ABSTRACT	**vi**
CHAPTER 1 INTRODUCTION	**7**
1.1 Background	7
1.2 Objective of project	7
1.3 Realization	8
1.4 Introduction to embedded system	8
1.5 Technology consideration	9
CHAPTER 2 HARDWARE DESCRIPTION	**10**
2.1 Architecture of Voice Operated Home Appliances	10
2.2 Voice recognition module	10
2.3 Microcontroller and IC description	14
2.4 Relay circuit	29
CHAPTER 3 SOFTWARE DESCRIPTION	**32**
3.1 VRbot GUI	32
3.2 Keil uvision	33
3.3 Proteus 8.1	35
CHAPTER 4 INTERFACING OF MICROCONTROLLERS	**36**
4.1 Interfacing with LEDs	36
4.2 Interfacing with seven segments	37
4.3 Interfacing with LCD	39
Future scope and applications	**41**
References	**42**
Appendix A: Source code	43
Appendix B: Components	48
Appendix C: Data sheets	52

List of figures

FIGURE	NAME OF THE FIGURE	PAGE NO.
FIGURE 1	Circuit diagram of Voice Operated Home Appliances	10
FIGURE 2	Pin diagram of EasyVR shield	12
FIGURE 3	Communication protocol	13
FIGURE 4	Block diagram of microcontroller	15
FIGURE 5	Pin diagram of AT89S52	16
FIGURE 6	Crystal circuit	17
FIGURE 7	Pull-up resistors	18
FIGURE 8	Internal Pull-ups	18
FIGURE 9	Reset Pin diagram	19
FIGURE 10	MAX232 connections	27
FIGURE 11	Pin diagram of relay	29
FIGURE 12	Internal circuit diagram of relay	30
FIGURE 13	Main application windows	33
FIGURE 14	Type of programming language	33
FIGURE 15	Windows for selecting device	34
FIGURE 16	Creating HEX files	34
FIGURE 17	HEX code using Keil	35
FIGURE 18	LED interfacing	36
FIGURE 19	Eight LED interfacing	37
FIGURE 20	Types of seven segment displays	37
FIGURE 21	Interfacing with seven segment display	38
FIGURE 22	LCD interfacing with 89S52	40

List of Tables

TABLE	NAME OF THE TABLE	PAGE NO.
TABLE 1	Technical specification of EasyVR module	12
TABLE 2	Difference between microprocessor and microcontroller	14
TABLE 3	Features of other microcontrollers of 8051 family	15
TABLE 4	Functions of port pins	18
TABLE 5	Timer 2 operating mode	21
TABLE 6	TCON registers	21
TABLE 7	Timer modes	22
TABLE 8	Interrupts in AT89S52	24
TABLE 9	Seven segment conversion table	39
TABLE 10	Pin description of LCD	39

Abstract

We design a Voice operated home appliances system which allows people control their home devices by voice command at home. This is a wireless, voice control system. People could control almost all the electronics equipment at home including lights, fans or even back ground music. In this project we have implemented light and fan control. Except for basic turning on and off of equipment at home, we also realize the function of fixed-time control, and error detection when some device is broken. The system is quick enough for respond all the commands.

Mainly the system consists of two modules i.e. voice recognition module and other is voice processing module. For voice recognition and training we used EasyVR shield. It will store the commands and perform comparison between stored command and given command. After comparison, voice recognition module sends data to voice processing module which includes AT89S52 microcontroller. Now this microcontroller will send instruction to various appliances (such as light, fan etc.) according to command given to voice recognition module.

Such system can help the elderly and disabled with assisted living, patient monitoring and emergency response. It can be easily installed in existing home without adding any additional wires/switches.

CHAPTER 1

INTRODUCTION

1.1 Background

Microcontrollers are used in industrial world to control many types of equipment, ranging from consumer to specialized devices. They have replaced the older types of microcontrollers, including microprocessor. The demand will grow as more equipment uses more intelligence. Applications range from controlling engines in modern automobiles to controlling laser printer and other computer peripherals. Technology has evolved to the point where this same washing machine could be connected to internet. We can envision a future with wearable computing where wrist watch-type devices could communicate with and control the washing machine using wireless networking.

Generally there are lots of microcontrollers available in order to fulfill the needs of control. One of the most popular is ATMEL AT89S52 which is relatively easy to work with, yet it has most of the features essential for a complete control system.

In this particular project, AT89S52 will be used as controller. The microcontroller offers various functions that are suitable to design a Voice Control. It is very flexible. It makes designer's work become easier.[1]

1.2 Objective of the project

Voice operated home appliances (VOHA) is a wireless home automation system that is supposed to be implemented in existing home environments, without any changes in the infrastructure. VOHA let the user to control the home from his or her voice and assign actions that should happen depending on trigger and command user issues. Every box is a stand-alone device. It is connected to the mains and controls the power outlet of the electrical device that is plugged into it.

There will be a receiver and transmitter in each of the box, so they can exchange information with the master (a processor). People can control power supply of electrical devices in order to create an interactive home environment to facilitate the control without changing any home appliance. People can enjoy the high technology and simplicity modern life style.

Each device will be with standard setup and while adding it into network; it can be given an address and tasks to do. All the setting will be easily resettable to default value, so people can move the devices between different electrical devices and networks.

1.3 Realization

The project started off with a brainstorming session. All ideas about how the VOHA should work and what functions it should have were written on a Project notebook. We discussed all possible solutions and ideas that we had come up with and removed things that were not possible to implement within this project scope.

1.4 Introduction to Embedded System

Embedded systems are computers which is part of special-purpose devices. Due to the limited duties this systems can be highly optimized to the particular needs. Traditionally most of these systems are used for control and process measurement, as a side-effect of higher integration of integrated circuits more complex applications can be solved by embedded systems. To be able to solve these problems embedded systems are commonly equipped with various kinds of peripherals. Early applications of embedded devices include the guidance computer of the Minuteman I missiles and the Apollo guidance computer. The Minuteman I & II missiles are intercontinental ballistic nuclear warheads, produced by Boeing in the 1960's. Due to the large quantities of ICs used in the guidance system of Minuteman II missiles, prices for ICs fell from 1000$ each to 3$ each. This lead to wide adoption of embedded systems in consumer electronics in the 1980's.
Nowadays embedded systems can be found in devices from digital watches to traffic-control systems. The broad ranges of applications with totally different requirements lead to various implementation approaches. The range of hardware used in embedded systems reaches from FPGAs to full blown desktop CPUs which are accompanied by special purpose ICs such as DSPs. On the software side, depending on the needs, everything, from logic fully implemented in hardware, to systems with own operating system and different applications running on it, can be found. [2]

1.5 Technology Consideration

When designing a voice controlled home appliances three main requirements must be taken into account:

- Ease-of-use
- Reliability
- Low cost

Ease-of-use: When designing a voice operated home appliances for the mass market it is very important to realize that it is the average homeowner or a semi-skilled installer who typically installs the system. The technology must provide simple intuitive installation and require no network management by the user during the life time of installation. Finally the technology must be designed to support horizontal application: enabling different product types from various venders to seamless communicate with each other and use each other features.

Reliability: Robust and reliable voice recognition module is crucial in order to allow the home control system to handle sensitive operations. For example, if the home owner instructs the central door locking application to lock and arm the alarm system he or she must be guaranteed that the instruction is registered and executed. Furthermore voice recognition module is sensitive to change in voice, commands so a particular learning algorithm must be applied to make this module a reliable one.

Low cost: In order to have a true mass –market technology the physical voice operated platform must have a very low cost. The right tradeoffs between technology choice and cost must be taken without compromising the reliability of the network. As many home control products are both developed manufactured in low salary countries it is important to supply a hardware and software platform. We decided the idea of voice control smart home system at the beginning of this project. Because we have AT89S52, so we were looking for some voice recognition module to put on it. At final, we chose the EasyVR Shield - Voice Recognition Shield which is perfect to suit for AT89S52 and has the high level voice recognition function. It is the cheaper one which is under the budget. In order to make a tidy and beautiful circuit, we used much time to design the circuit to improve space utilization.

CHAPTER 2

HARDWARE DESCRIPTION

2.1 Architecture of Voice Operated Home Appliances

Circuit diagram of voice operated home appliance system is shown in fig.1. There is one microcontroller AT89S52 in our project which is connected to voice recognition module. There is a microphone connected to this module so that it could receive voice signal from people. When people say the voice instruction, microphone gets it first and then AT89S52 receive it. By the program controlling, AT89S52 will send the signal to transmitter LCD display and to relay circuit. LCD will show which function is performed and relay circuit will perform that function.

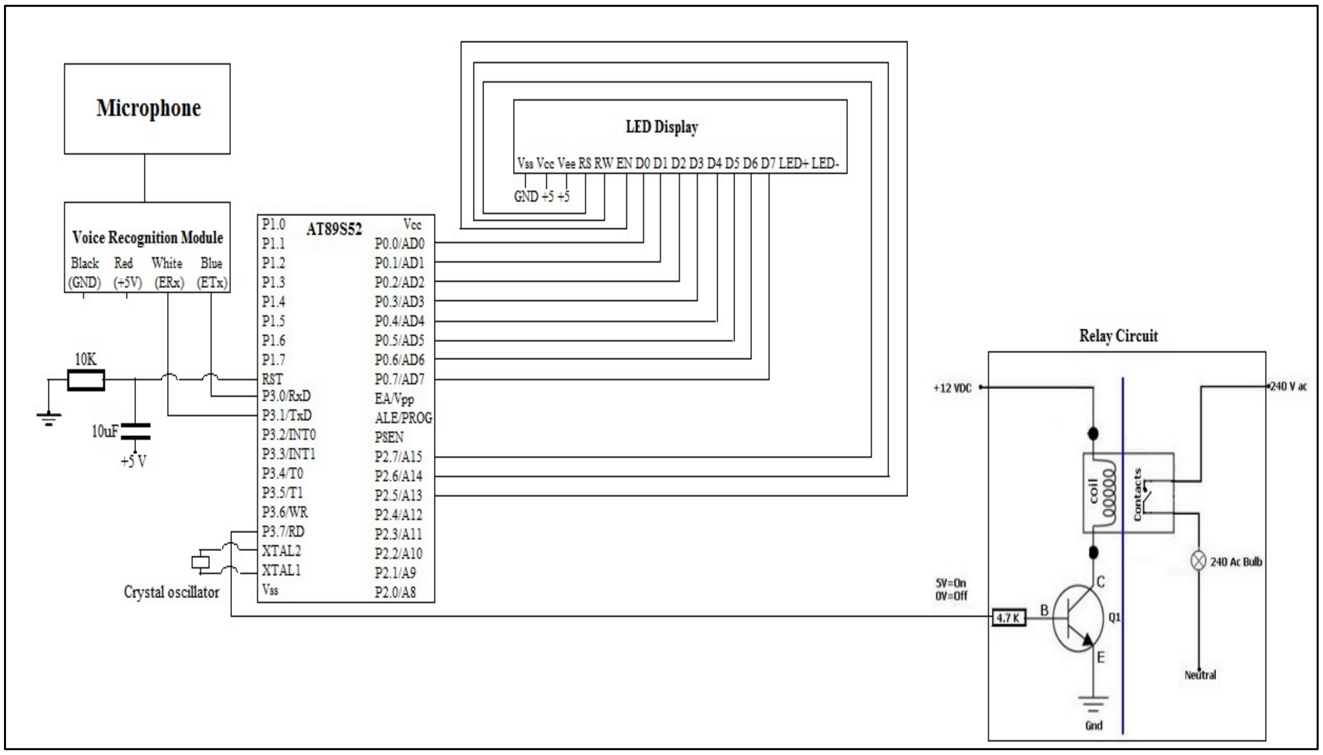


Figure – 1 Circuit diagram of voice operated home appliances

Now we will discuss in detail each part of this circuit diagram.

2.2 Voice Recognition Module

We used EasyVR Shield, an Arduino product in our project. EasyVR Commander could record the voice someone speak in the chip and then compare the voice said by people when they send out instructions. The hard part of voice recording and recognition is that we need to choose different pronounce to record in the chip, the more difference each instruction has, the better. After we choose and record and training with the most recognizable signals, we may use compiler to write programs in AT89S52.[3]

EasyVR features:

1. A host of built-in Speaker Independent (SI) commands for ready to run basic controls, in the followings languages:

- English (US)
- Italian
- German
- French
- Spanish
- Japanese

2. Supports up to 32 user-defined Speaker Dependent (SD) triggers or commands as well as voice Passwords. SD custom commands can be spoken in ANY language.
3. Easy-to-use and simple Graphical User Interface to program Voice Commands and audio.
4. Module can be used with any host with an UART interface (powered at 3.3V - 5V)
5. Simple and robust documented serial protocol to access and program through the host board
6. 3 GPIO lines (IO1, IO2, IO3) that can be controlled by new protocol commands.
7. PWM audio output that supports 8 ohm speakers.
8. Sound playback feature.

Technical Specifications:

The EasyVR module can be used with any host with an UART interface powered at 3.3V – 5V, such as PIC and Arduino boards. Some application examples include home automation, such as voice controlled light switches, locks or beds, or adding “hearing” to the most popular robots on the market. Physical dimension and pin diagram of EasyVR shield is shown in fig.2.

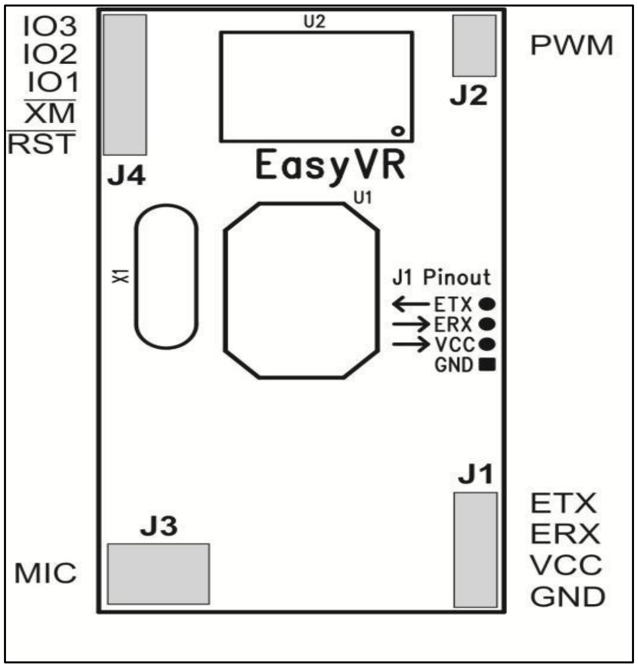


Figure-2 Pin diagram of EasyVR shield

Table – 1 Technical specifications of EasyVR module

Connector	Number	Name	Type	Description
J1	1	GND	-	Ground
	2	VCC	I	Voltage DC input
	3	ERX	I	Serial Data Receive
	4	ETX	O	Serial Data Transmit
J2	1-2	PWM	O	Differential audio output (can directly drive 8 ohm speaker)
J3	1	MIC_RET	-	Microphone reference ground
	2	MIC_IN	I	Microphone Input signal
	1	/RST	I	Active low asynchronous reset(Internal 100K pull-up)
	2	/XM	I	Boot select (Internal 1K pull down)

J4	**3**	**IO1**	**I/O**	**General purpose I/O (3.0 VDC TTL level)**
	4	**IO2**	**I/O**	**General purpose I/O (3.0 VDC TTL level)**
	5	**IO3**	**I/O**	**General purpose I/O (3.0 VDC TTL level)**

Communication Protocol:

Communication with the EasyVR module uses a standard UART interface compatible with 3.3-5V TTL/CMOS logical levels, according to the powering voltage VCC.[4]

A typical connection to an MCU-based host:

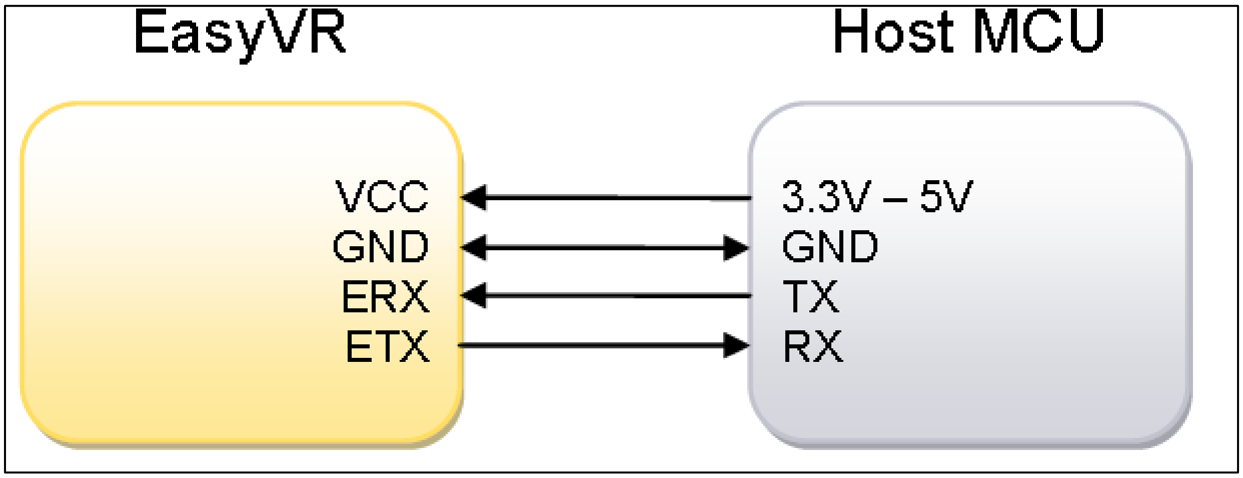


Figure- 3 Communication protocol

The initial configuration at power on is 9600 baud, 8 bit data, No parity, 1 bit stop. The baud rate can be changed later to operate in the range 9600 - 115200 baud.

2.3 Microcontroller and IC Description

In our project we have studied about the 8051 micro-controller and its family. To start with the AT89S52 micro-controller we have studied 8051 micro-controller and their family and the difference between 8051, AT89S51 and AT89S52.

Introduction to 8051

In our project we studied about the microcontroller which is the combination of the hardware and the software which is used to do about the specific task. A micro-controller can be compared to a small standalone computer; it is a very powerful device, which is capable of executing a series of pre-programmed tasks and interacting with other hardware devices.

Table – 2 Difference between Microprocessor and Microcontroller

Microprocessor	Microcontroller
In the microprocessor CPU stand alone, RAM, ROM, input/output ports, timers and the serial ports are separate.	In microcontroller CPU, RAM, ROM, input/output ports, timers and the serial ports all are on the single chip.
In microprocessor designer can decide the amount of RAM,ROM and the input/output ports	In microcontroller the RAM, ROM, input/output ports are fixed.
The microprocessors are expensive and versatile.	These are used where the cost, power and space are critical.
The microprocessors are used where we perform the multipurpose task.	The microcontrollers are used in the specific task.
It is concerned with the rapid movement of the code and the data from the external address to the chip.	It is concerned with the rapid movement of the bits within the chip.

Need of Microcontrollers: It reduces the size and cost compared to a design that uses a separate microprocessor, memory, and input/output devices, microcontrollers make it economical to digitally control even more devices and processes.

Block Diagram of Microcontroller:

The 8051 has 128 bytes of RAM,4K bytes of on chip ROM, two timers, one serial port, and four ports(each of 8 bits wide) all on a single chip. Also 8051 can address 64k bytes of external memory because it has 16 address lines.[5]

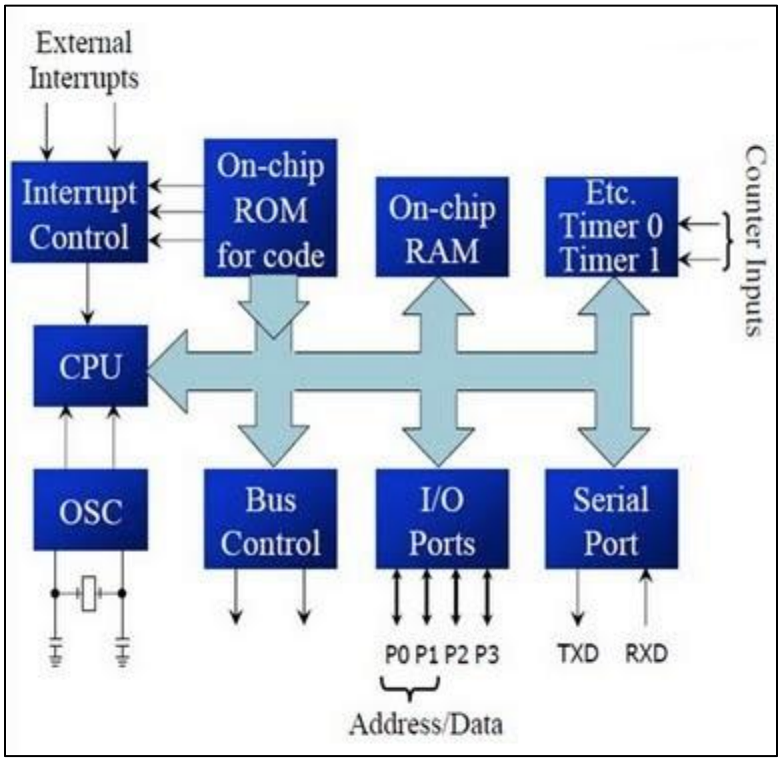


Figure – 4 Block diagram of microcontroller

Table – 3 Features of other members of 8051 family

Feature	**89S51**	**89S52**
ROM(on-chip program space in bytes)	4K	8K
RAM(bytes)	128	256
Timers	2	3
I/O pins	32	32
Serial port	1	1
Interrupt sources	6	8

We used AT89S52 in our project.

Pin configuration of AT89S52

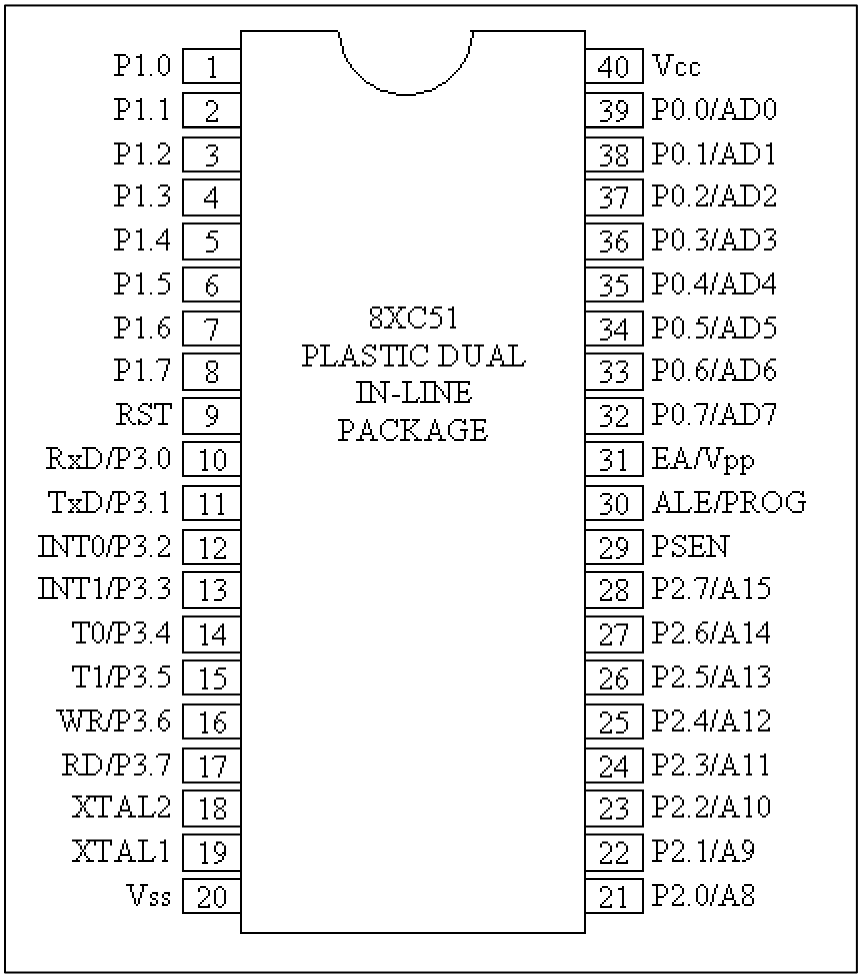


Figure – 5 Pin diagram of AT89S52

- **VCC:** Supply voltage.
- **GND**: Ground.
- **ALE/PROG**: Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory. ALE is emitted at a constant rate of 1/6 of the oscillator frequency, for external timing or clocking purposes, even when there are no accesses to external memory. (However, one ALE pulse is skipped during each access to external Data Memory.) This pin is also the program pulse input (PROG) during EPROM programming.
- **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 98 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.

- **PSEN:** Program Store Enable is the read strobe to external Program Memory. When the device is executing out of external Program Memory, PSEN is activated twice each machine cycle (except that two PSEN activations are skipped during accesses to external Data Memory). PSEN is not activated when the device is executing out of internal Program Memory.
- **EA/VPP:** When EA is held high the CPU executes out of internal Program Memory (unless the Program Counter exceeds 0FFFH in the 80C51). Holding EA low forces the CPU to execute out of external memory regardless of the Program Counter value. In the 80C31, EA must be externally wired low. In the EPROM devices, this pin also receives the programming supply voltage (VPP) during EPROM programming.
- **XTAL1:** Input to the inverting oscillator amplifier.
- **XTAL2:** Output from the inverting oscillator amplifier.

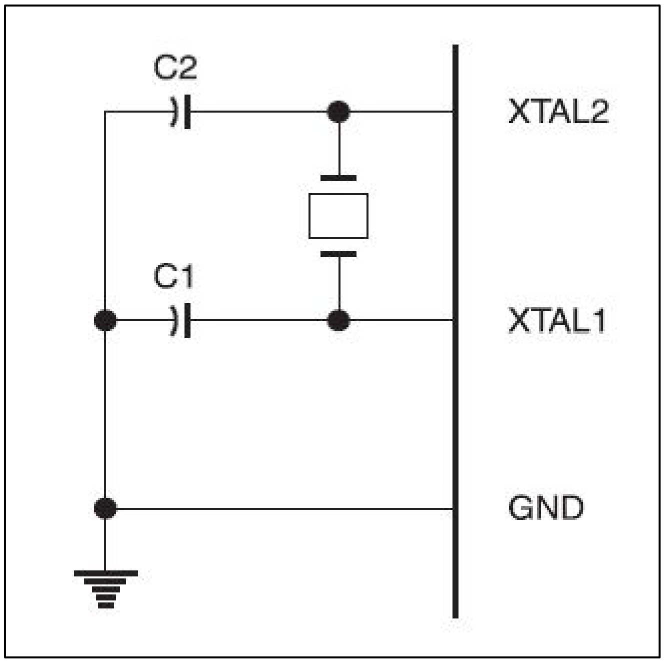


Figure – 6 Crystal circuit

- **Port 0:** Port 0 is an 8-bit open drain bidirectional port. As an open drain output port, it can sink eight LS TTL loads. Port 0 pins that have 1s written to them float, and in that state will function as high impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external memory. In this application it uses strong internal pull-ups when emitting 1s. Port 0 emits code bytes during program verification. In this application, external pull-ups are required.

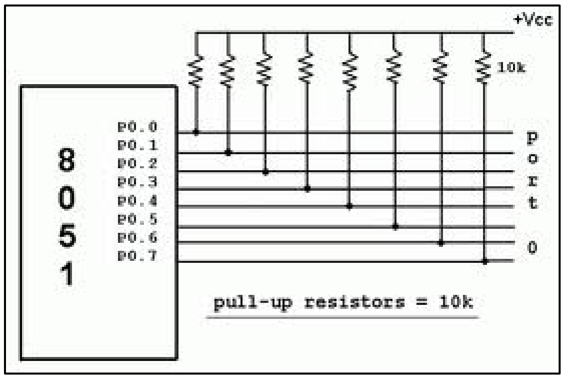


Figure – 7 Pull-up resistors

- **Port 1**: Port 1 is an 8-bit bidirectional I/O port with internal pull-ups. Port 1 pins that have 1s written to them are pulled high by the internal pull-ups, and in that state can be used as inputs. As inputs, port 1 pins that are externally being pulled low will source current because of the internal pull-ups.

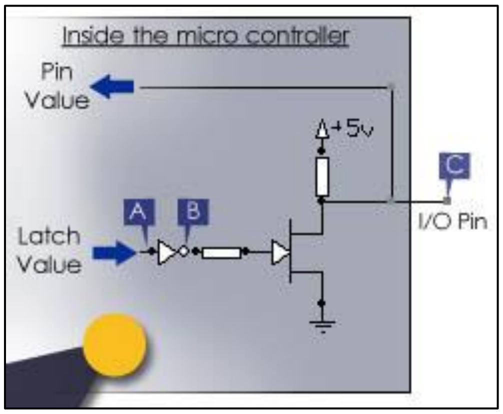


Figure – 8 internal pull-ups

- **Port 2:** Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. Port 2 emits the high-order address byte during accesses to external memory that use 16-bit addresses. In this application, it uses the strong internal pull-ups when emitting 1s.
- **Port 3:** Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. It also serves the functions of various special features of the 89S52 as follows:

Table – 4 Functions of Port pins

Port Pin	Alternate Function
P3.0	RXD (serial input port)
P3.1	TxD (serial output port)
P3.2	INT0 (external interrupt 0)

P3.3	INT1 (external interrupt 1)
P3.4	T0 (timer 0 external input)
P3.5	T1 (timer 1 external input)
P3.6	WR (external data memory write strobe)
P3.7	RD (external data memory read strobe)

- **VCC:** Supply voltage of +5v.
- **VSS:** Circuit ground potential
- **RST:** It is an input and is active high. Upon applying a high pulse to this pin, the microcontroller will reset and terminate all activities and all values in registers are lost. The reset should of minimum two machine cycles.

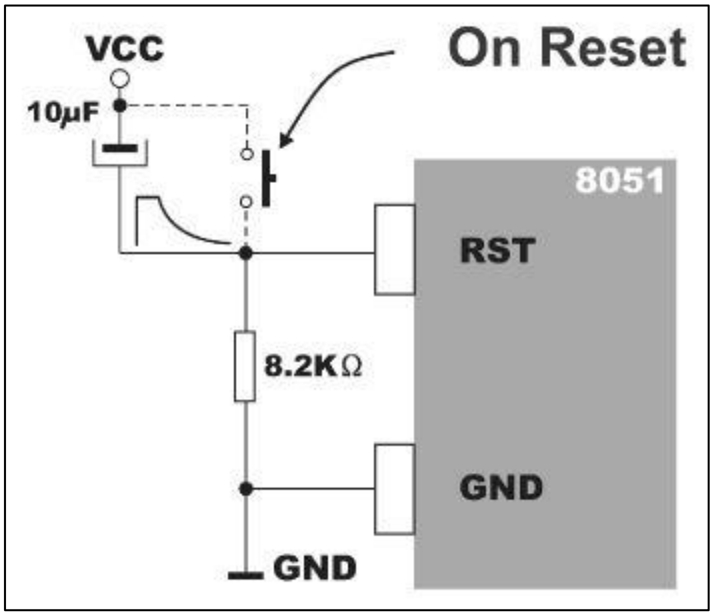


Figure – 9 Reset pin diagram

Time Delays:

There are two ways to create a time delay in 89S52:

1. Using a simple for loop.
2. Using the 89S52 timers.

Introduction to Timers

The AT89S52 has three **16-bit** Timer/Counters: **Timer0**, **Timer1 and Timer2**. The timers can be used to generate accurate delays and the counters can be used to count events. An event can be anything. It can be a pulse, a push, a pull, or any stimulus.

Basic Registers of Timer/Counter

Timer 0 and Timer 1 in the AT89S52 operate the same way as Timer 0 and Timer 1 in the 8051.

1. Timer0 Registers: Timer0 is a 16-bit timer/counter and it is accessed as **TH0** (Timer0 high byte) and **TL0** (Timer0 low byte).

TH0								TL0							
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

2. Timer1 Registers: Timer1 is a 16-bit timer/counter and it is accessed as **TH1** (Timer1 high byte) and **TL1** (Timer1 low byte).

TH1								TL1							
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

3. Timer2 Registers: Timer 2 is a 16-bit Timer/Counter that can operate as either a timer or an event counter. The type of operation is selected by bit C/T2 in the SFR T2CON. Timer 2 has three operating modes: capture, auto-reload (up or down counting), and baud rate generator. The modes are selected by bits in T2CON. Timer 2 consists of two 8-bit registers, TH2 and TL2. In the Timer function, the TL2 register is incremented every machine cycle.

4. TMOD Register: The TMOD (timer mode) register sets the operational modes of Timer0 and Timer1. It is 8 bits wide in which the upper nibble is for Timer1 and the lower nibble is for Timer0.

TMOD							
TIMER1				TIMER0			
GATE	C/T	M1	M0	GATE	C/T	M1	M0

GATE: Gating control when set. Timer/Counter x is enabled only while INTx pin is high and TRx control pin is set. When cleared, Timer x is enabled whenever TRx control bit is set.

C/T: If **C/T** = 1, the counter operation is selected. If **C/T**=0, the timer operation is selected

M1: Mode bit 1

M0: Mode bit 0

Table – 5 Timer 2 operating mode

RCLK+TCLK	CP/RL2	TR2	Mode
0	0	1	16-bit Auto reload
0	1	1	16-bit Capture
1	X	1	Baud Rate Generator
X	X	0	(Off)

When the timer operating mode is selected, the clock source of the timer is the same as the clock source of AT89S52 (ex: external quartz crystal). The frequency of the timer's clock source is equal to the frequency of At89S52′s clock source divided by 12. If the clock source is a 12MHz quartz crystal, the timer's clock frequency is equal to 12MHz/12 = 1MHz

4. T2CON Register: The TCON (Timer/counter 2 control) register is responsible to the control and status bits of AT89S52′s timers/counters (and external interrupts).

T2CON Address = 0C8H Reset Value = 0000 0000B

TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2
7	6	5	4	3	2	1	0

Table – 6 TCON registers

TF2	Timer 2 overflow flag set by a Timer 2 overflows and must be cleared by software. TF2 will not be set when either RCLK = 1or TCLK = 1.
EXF2	Timer 2 external flag set when either a capture or reload is caused by a negative transition on T2EX and EXEN2 = 1.When Timer 2 interrupt is enabled, EXF2 = 1 will

	cause the CPU to vector to the Timer 2 interrupt routine. EXF2 must be cleared by software. EXF2 does not cause an interrupt in up/down counter mode (DCEN = 1).
RCLK	Receive clock enable. When set, causes the serial port to use Timer 2 overflow pulses for its receive clock in serial port Modes 1 and 3. RCLK = 0 causes Timer 1 overflow to be used for the receive clock.
TCLK	Transmit clock enable. When set, causes the serial port to use Timer 2 overflow pulses for its transmit clock in serial port Modes 1 and 3. TCLK = 0 causes Timer 1 overflows to be used for the transmit clock.
EXEN2	Timer 2 external enables. When set, allows a capture or reload to occur as a result of a negative transition on T2EX if Timer 2 is not being used to clock the serial port. EXEN2 = 0 causes Timer 2 to ignore events at T2EX.
TR2	Start/Stop control for Timer 2. TR2 = 1 starts the timer.
C/T2	Timer or counter select for Timer 2. C/T2 = 0 for timer function. C/T2 = 1 for external event counter (falling edge triggered).
CP/RL2	Capture/Reload select. CP/RL2 = 1 causes captures to occur on negative transitions at T2EX if EXEN2 = 1. CP/RL2 = 0 causes automatic reloads to occur when Timer 2 overflows or negative transitions occur at T2EX when EXEN2 = 1. When either RCLK or TCLK = 1, this bit is ignored and the timer is forced to auto-reload on Timer 2 overflow.

Timer Modes

The timer/counter of AT89S52 has four modes. The mode of a timer is selected using the mode bits of the TMOD register.

Table – 7 Timer modes

Mode	M1	M0	Operating Mode
0	0	0	13-bit timer mode. 8-bit timer/counter THx with TLx as 5-bit prescaler
1	0	1	16-bit timer mode. THx and TLx are cascaded
2	1	0	8-bit auto reload. THx holds the value that is to be reloaded into TLx each time it overflows
3	1	1	Split Timer Mode. (Timer0) TL0 is an 8-bit Timer/Counter controlled by the

			standard Timer0 control bits. TH0 is an 8-bit timer only controlled by Timer 1 control bits
3	1	1	Timer 1) Timer/Counter 1 stopped.

1. Mode 0

Both Timers in Mode 0 are 8-bit Counters with a divide-by-32 pre-scaler. In this mode, the Timer register is configured as a 13-bit register. As the count rolls over from all 1s to all 0s, it sets the Timer interrupt flag **TF1**. The counted input is enabled to the Timer when **TR1** = 1 and either **GATE** = 0 or **INT1** = 1. Setting **GATE**=1 allows the Timer to be controlled by external input **INT**1, to facilitate pulse width measurements. **TR1** is a control bit in the Special Function Register **TCON**. **GATE** is in **TMOD**. The 13-bit register consists of all 8 bits of **TH1** and the lower 5 bits of **TL1**. The upper 3 bits of **TL**1 are indeterminate and should be ignored. Setting the run flag (**TR1**) does not clear the registers. Mode 0 operation is the same for Timer 0 as for Timer 1, except that **TR0**, **TF0** and **INT0** replace the corresponding Timer 1 signals. There are two different **GATE** bits, one for Timer 1 (TMOD.7) and one for Timer 0 (TMOD.3)

2. Mode 1

Mode 1 is the same as Mode 0, except that the Timer register is run with all 16 bits. The clock is applied to the combined high and low timer registers (**TL1**/**TH1**). As clock pulses are received, the timer counts up: 0000H, 0001H, 0002H, etc. An overflow occurs on the FFFFH-to-0000H overflow flag. The timer continues to count. The overflow flag is the **TF1** bit in **TCON** that is read or written by software.

3. Mode 2

Mode 2 configures the Timer register as an 8-bit Counter (**TL1**) with automatic reload. Overflow from **TL1** not only sets **TF1**, but also reloads **TL1** with the contents of **TH1**, which is preset by software. The reload leaves **TH1** unchanged. Mode 2 operation is the same for Timer/Counter 0.

4. Mode 3

Timer 1 in Mode 3 simply holds its count. The effect is the same as setting **TR1** = 0. Timer 0 in Mode 3 establishes **TL0** and **TH0** as two separate counters. **TL0** uses the Timer 0 control bits:

C/T, **GATE**, **TR0**, **INT0**, and **TF0**. **TH0** is locked into a timer function (counting machine cycles) and over the use of **TR1** and **TF1** from Timer 1. Thus, **TH0** now controls the Timer 1 interrupt. Mode 3 is for applications requiring an extra 8-bit timer or counter. With Timer 0 in Mode 3, the AT89C2051 can appear to have three Timer/Counters. When Timer 0 is in Mode 3, Timer 1 can be turned on and off by switching it out of and into its own Mode 3. In this case, Timer 1 can still be used by the serial port as a baud rate generator or in any application not requiring an interrupt.

Introduction to Interrupts

Interrupt: An interrupt is an asynchronous signal that needs attention. An interrupt stops the CPU of a microcontroller, leaving the tasks that it is currently doing, to give attention to the interrupt signal. Once the attention has been given to the interrupt signal, the CPU goes back to its unaccomplished task before the interrupt has occurred and continues the task.

Six Interrupts in 89S52

It has 6 interrupts sources (5 interrupts + RESET). The interrupt sources of AT89S52 are the following:

Table – 8 Interrupts in AT89S52

Srno	**Interrupt**	**ROM location(Hex)**	**Pin**	**Source**	**Priority**	**Flag clearing**
1	Reset	0000	9	RST	N/A	Auto
2	External Interrupt 0	0003	P3.2(12)	IE0	0	Auto
3	Timer 0 Interrupt	000B		TF0	1	Auto
4	External Interrupt 1	0013	P3.3(13)	IE1	2	Auto
5	Timer 1 Interrupt	001B		TF1	3	Auto
6	Serial COM interrupt	0023		RI or TI	4	By programmer

Interrupt Enable Register

The **Interrupt Enable** (**IE**) register is responsible in enabling and disabling the different interrupt sources of 8051.

D7 D0

EA	_____	ET2	ES	ET1	EX1	ET0	EX0

EA	Disables all interrupts. If EA = 0, no interrupt is acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
___	Not implemented
ET2	Enables or disables the Timer 2 overflow or capture interrupt
ES	Enables or disables the serial port interrupt.
ET1	Enables or disables the Timer 1 overflow interrupt.
EX1	Enables or disables External Interrupt 1.
ET0	Enables or disables the Timer 0 overflow interrupt.
EX0	Enables or disables External Interrupt 0.

How to Enable an Interrupt

1. Initialize the sources of interrupts such as Timers, External Interrupts, or UART.

2. Set the bits of the **IE** register that corresponds to the interrupt sources that you want to be enabled.

Example: If you want to enable the interrupt of the serial port or UART set **ES** to 1 or **ES**=1.

3. Enable the global interrupt by setting the **EA** bit of the **IE** register (EA=1).

How to Write an Interrupt Service Routine or ISR

The **interrupt service routine** or **ISR** is the routine that an MCU is servicing every time an interrupt occurs. It can be treated as an ordinary subroutine in a C program.

The format of the **ISR** is:

Void ***your_ISR_name***(void) interrupt ***interrupt_priority_number***

{

//your routine here

}

The *your_ISR_name* is user defined. It can be any name.

The *interrupt_priority_number* is fixed depending on the source of the interrupt.

Introduction to Embedded C

As we know that the C is a high level language and it is easy to understand, therefore we use C language to program the 8051 and this C language is called embedded C language. The advantages of C language over the assembly language are:

1. It is easier and less time consuming to write program in C.
2. C is easier to modify and upgrade.
3. You can use codes available in function libraries.
4. C code is portable to other microcontrollers with little or no modifications.

Data Types in C

1. **Unsigned char:** It is an 8 bit data type that takes values from 0-255. If we don't put unsigned keyword in front of char then it becomes signed char.
2. **Signed char:** It is an 8 bit data type that uses the most significant bit to represent +or- signs. Therefore we have only 7 bits for magnitude and giving us values from -128 to +127.
3. **Unsigned int:** It is a 16 bit data type that takes value from 0-65535. It is used to define the 16 bit variables such as memory addresses but it consume lot of memory space.
4. **Signed char:** It is a 16 bit data type that uses most significant bit to represent +or- signs. Thus it takes the value from -32,768 to +32,767.
5. **Sbit(single bit):**It is used to access single-bit addressable registers. It allows us to access the single bits of sfr registers and bit addressable ports.

Introduction to Serial Port Programming

UART: UART stands for **Universal Asynchronous Receiver/Transmitter**. As its name implies, it is universal. It can be used to establish a communication between a microcontroller and another device – microcontroller, USB controller, Bluetooth modules, GSM modules, GPS modules, personal computers, etc.

RS-232: **RS-232** is a standard for serial transmission of data between a **DTE** (Data Terminal Equipment) and a **DCE** (Data Circuit-terminating Equipment). It is commonly found in desktop computers where it is commonly referred as **COM port**.

AT89S51 UART: The AT89C2051 has one UART port. Its **TXD** (*Transmit*) pin is the same as its **P3.1** pin. Its **RXD** (*Receive*) pin is the same as its **P3.0** pin.

AT89S51 RS-232 Interface: The UART port of a microcontroller can be used to interface to a RS-232 port of a personal computer. However, the voltage levels of UART must be converted to voltage levels compatible to RS-232.

To convert UART voltage levels to RS-232 voltage levels, you may use the following circuit:

1. Using a MAX232

This is the most preferred circuit to convert UART using the TTL voltage levels to RS232 voltage levels. The 5V levels are converted by MAX232 to -9V to -12V and vice versa. The 0V levels are converted by MAX232 to +9V to +12V and vice versa.

The female DB-9 connector is used to connect with the RS-232 port of a personal computer.

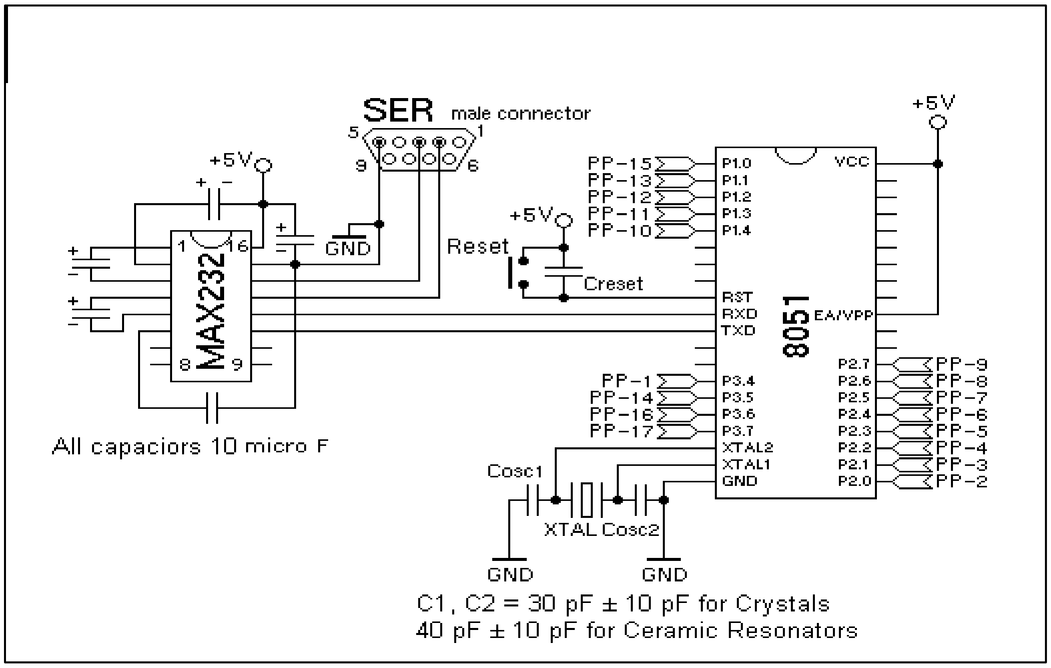


Figure – 10 MAX232 connections

Registers Used in UART Programming

1. SBUF Register

The **Serial Buffer (SBUF) Register** is the register used for serial communication using UART. The SBUF is actually two separate registers, a transmit buffer and a receive buffer register. When data is moved to SBUF, it goes to the transmit buffer where it is held for serial transmission. (Moving a byte to SBUF is what initiates the transmission.) When data is moved from SBUF, it comes from the receive buffer.

2. SCON Register

The **Serial Control (SCON) Register** is responsible to the data frame format, enabling of serial reception, etc. It is composed of 8 bits with each bit having a unique purpose. The SCON Register is both byte-addressable and bit-addressable.

SCON							
FE/SM0	SM1	SM2	REN	TB8	RB8	TI	RI

7	FE	**Framing Error bit (SMOD0=1**).Clear to reset the error state, not cleared by a valid stop bit.Set by hardware when an invalid stop bit is detected.SMOD0 must be set to enable access to the FE bit
	SM0	**Serial port Mode bit 0.** Refer to SM1 for serial port mode selection.SMOD0 must be cleared to enable access to the SM0 bi
6	SM1	**Serial port Mode bit 1**

SM0	**SM1**	**MODE**	**DESCRIPTION**	**BAUD RATE**
0	0	0	Shift Register	$F_{CPU\ PERIPH}/6$
0	1	1	8-bit UART	Variable
1	0	2	9-bit UART	$F_{CPU\ PERIPH}$ /32 or /16
1	1	3	9-bit UART	Variable

5	SM2	**Serial port Mode 2 bit / Multiprocessor Communication Enable bit.** Clear to disable multiprocessor communication feature. Set to enable multiprocessor communication feature in mode 2 and 3, and eventually mode 1. This bit should be cleared in mode 0.
4	REN	**Reception Enable bit.** Clear to disable serial reception. Set to enable serial reception.
3	TB8	**Transmitter Bit 8 / Ninth bit to transmit in modes 2 and 3.** Clear to transmit logic 0 in the 9th bit. Set to transmit a logic 1 in the 9th bit.
2	RB8	**Receiver Bit 8 / Ninth bit received in modes 2 and 3.** Cleared by hardware if 9th bit received is a logic 0. Set by hardware if 9th bit received is a logic 1. In mode 1, if SM2 = 0, RB8 is the received stop bit. In mode 0 RB8 is not used.
1	TI	**Transmit Interrupt flag.** Clear to acknowledge interrupt. Set by hardware at the end of the 8th bit time in mode 0 or at the beginning of the stop bit in the other modes.
0	RI	**Receive Interrupt flag.** Clear to acknowledge interrupt. Set by hardware at the end of the 8th bit time in mode 0

2.4 Relay Circuit

Relay is an electromagnetic device which is used to isolate two circuits electrically and connect them magnetically. They are very useful devices and allow one circuit to switch another one while they are completely separate. They are often used to interface an electronic circuit (working at a low voltage) to an electrical circuit which works at very high voltage. For example, a relay can make a 5V DC battery circuit to switch a 220V AC mains circuit. Thus a small sensor circuit can drive, say, a fan or an electric bulb.

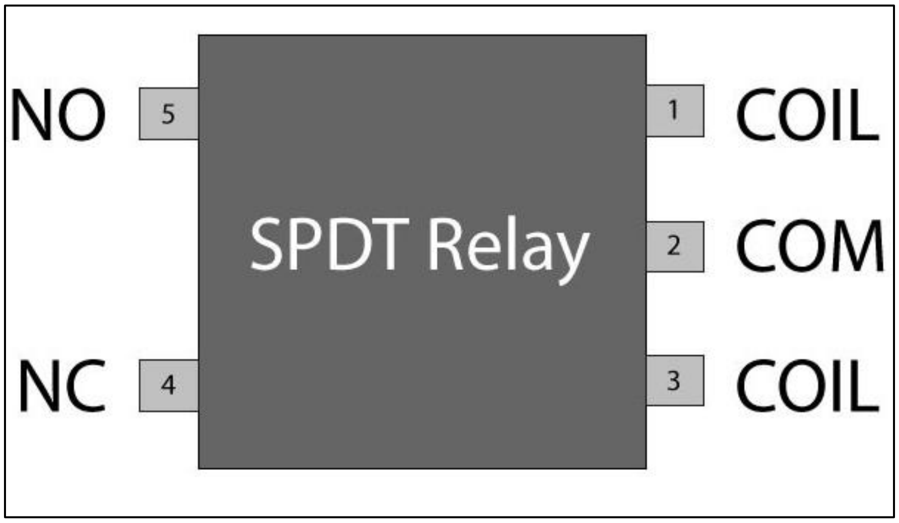


Figure – 11 Pin diagram of relay

The relay’s switch connections are usually labelled COM (POLE), NC and NO:

COM/POLE= Common, NC and NO always connect to this; it is the moving part of the switch.

NC = Normally Closed, COM/POLE is connected to this when the relay coil is not magnetized.

NO = Normally Open, COM/POLE is connected to this when the relay coil is MAGNETIZED and vice versa.

A **relay switch** can be divided into two parts: input and output. The input section has a coil which generates magnetic field when a small voltage from an electronic circuit is applied to it. This voltage is called the operating voltage.

Commonly used relays are available in different configuration of operating voltages like 6V, 9V, 12V, 24V etc. The output section consists of contactors which connect or disconnect mechanically. In a basic relay there are three contactors: normally open (NO), normally closed (NC) and common (COM).

At no input state, the COM is connected to NC. When the operating voltage is applied the relay coil gets energized and the COM changes contact to NO. By using proper combination of contactors, the electrical circuit can be switched on and off. Get inner details about structure of a Relay switch.

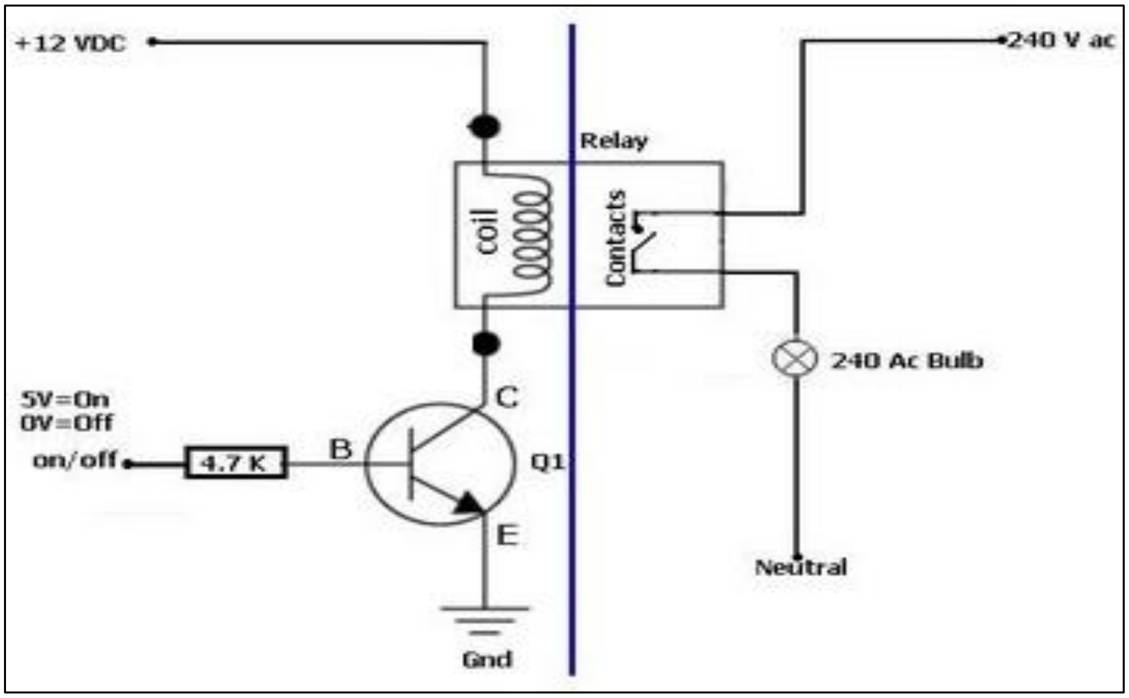


Figure – 12 internal circuit diagram of relay

In the example above a low voltage (12Vdc) is used to turn on a relay to switch on a 240V ac main circuit. Please note that there is no electrical connection inside the relay between the two circuits, the link is only magnetic and mechanical.

If there is Zero (0) volt at the Base (B) of transistor Q1, the transistor is open and therefore current is not able to flow from the 12V rail to the Ground and hence the relay contacts are open and the 240V ac Bulb is off.

Now if a Voltage is applied at the Base (B) of transistor Q1. The transistor will close (ON) and the current will start flowing from the 12V rail through the relay coil to the Collector of the Q1 down to the Ground.

When the current flow through the coil a magnetic field is created around the wire and this magnet (electro) will attract the contacts of the relay. Once the contacts closes then the other circuit is switched on and the Bulb lights.

CHAPTER 3

SOFTWARE DESCRIPTION

3.1 VRbotGUI

This software can be used to easily configure your EasyVR module connected to your PC through an adapter board, or by using the microcontroller host board with the provided “bridge” program.

Getting Started

Connect the adapter board or a microcontroller host board with a running “bridge” program1 to your PC, and then check that all devices are properly turned on and start the EasyVR Commander.

Select the serial port to use from the toolbar or the “File” menu, and then go with the “Connect” command.

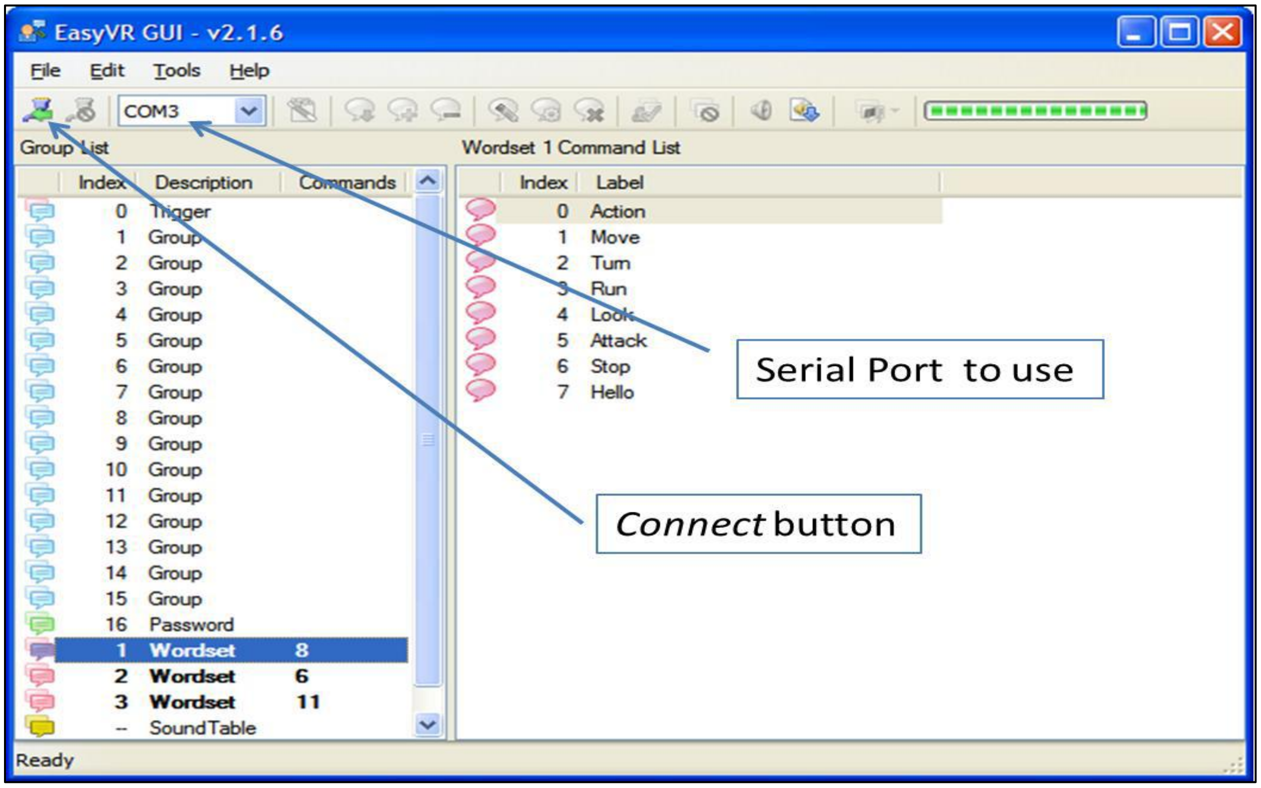
There are four kinds of commands in the software (see Figure 13):

Trigger - is a special group where you have the built-in SI trigger word "Robot" and you may add one user-defined SD trigger word. Trigger words are used to start the recognition process

Group - where you may add user-defined SD commands

Password - a special group for "vocal passwords" (up to five), using Speaker Verification (SV) technology.

Word set - built-in set of SI commands (for instance in Figure 13 below, the Word set 1 is selected)



Group List

Index	Description	Commands
0	Trigger	
1	Group	
2	Group	
3	Group	
4	Group	
5	Group	
6	Group	
7	Group	
8	Group	
9	Group	
10	Group	
11	Group	
12	Group	
13	Group	
14	Group	
15	Group	
16	Password	
1	Wordset	8
2	Wordset	6
3	Wordset	11
--	SoundTable	

Wordset 1 Command List

Index	Label
0	Action
1	Move
2	Turn
3	Run
4	Look
5	Attack
6	Stop
7	Hello

Figure – 13 Main application windows

It's a graphical user interface used by EasyVR to implement or store various voice commands. There are 15 groups in which we can store various commands. We can replace the stored command with new one at later stage.

3.2 KEIL µVision

It is the industry-standard tool chain for all 8051-compatible devices, it supports classic 8051, Dallas 390, NXP MX, extended 8051 variants, and C251 devices. The µVision IDE/Debugger integrates complete device simulation, interfaces too many target debug adapters, and provides various monitor debug solutions.

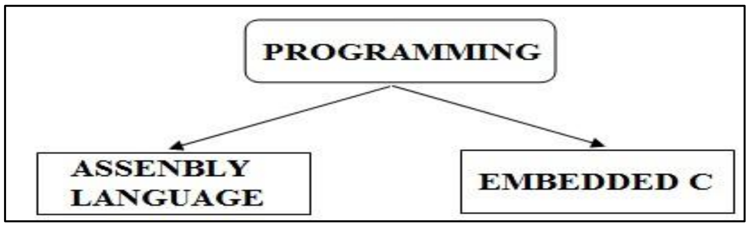
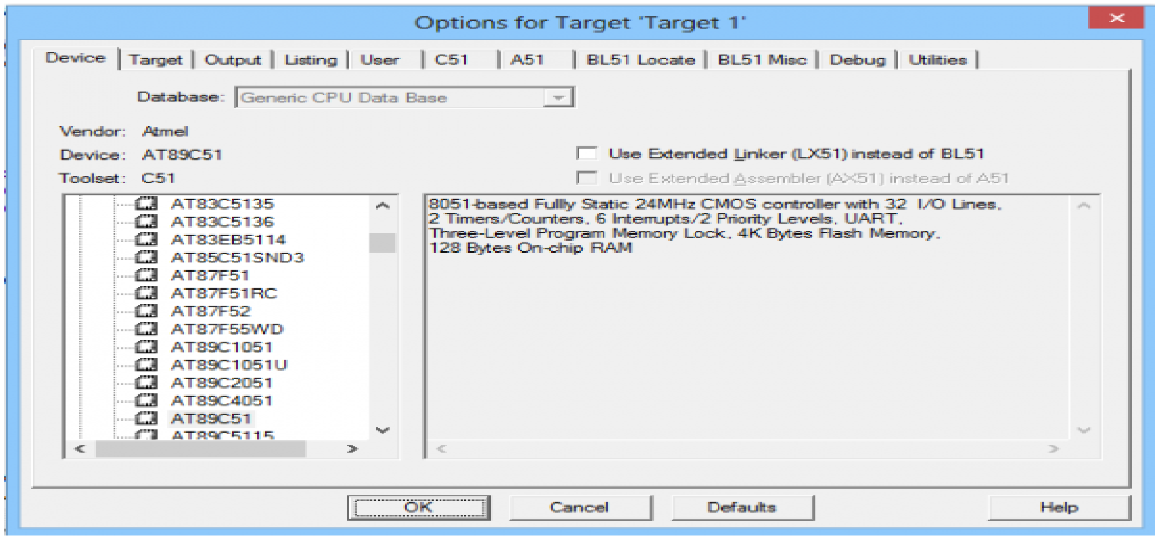


Figure – 14 Types of programming language



Database: Generic CPU Data Base

Vendor: Atmel
Device: AT89C51
Toolset: C51

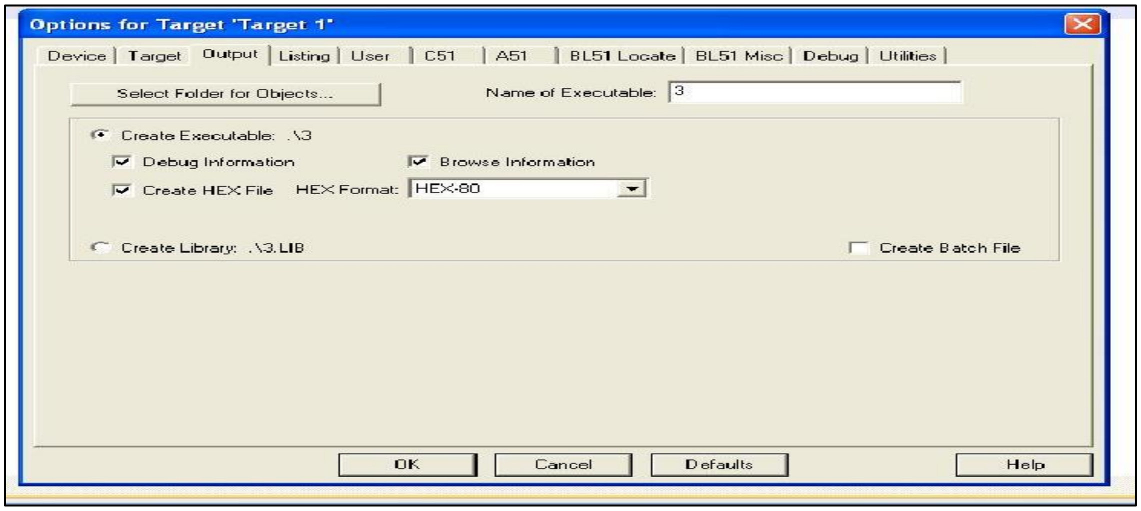
☐ Use Extended Linker (LX51) instead of BL51
☐ Use Extended Assembler (AX51) instead of A51

AT83C5135
AT83C5136
AT83EB5114
AT85C51SND3
AT87F51
AT87F51RC
AT87F52
AT87F55WD
AT89C1051
AT89C1051U
AT89C2051
AT89C4051
AT89C51
AT89C5115

8051-based Fullly Static 24MHz CMOS controller with 32 I/O Lines, 2 Timers/Counters, 6 Interrupts/2 Priority Levels, UART, Three-Level Program Memory Lock, 4K Bytes Flash Memory, 128 Bytes On-chip RAM

Figure – 15 Windows for Selecting Device

This software will convert codes into HEX file which is burnt into the microcontroller using a burner. This compiler is used to convert High Level Language into Machine Language.



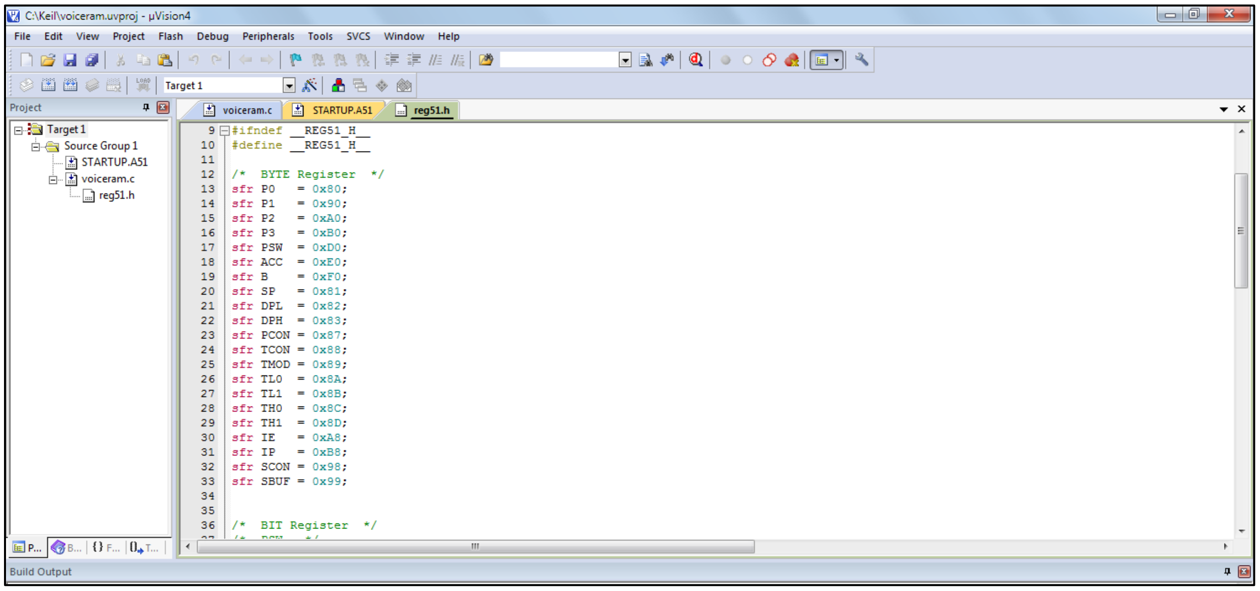
Name of Executable: 3

● Create Executable: .\3
☑ Debug Information ☑ Browse Information
☑ Create HEX File HEX Format: HEX-80

○ Create Library: .\3.LIB ☐ Create Batch File

Figure – 16 Creating HEX files

The software flowchart programmed in the microcontroller of the transmitter section is shown in figure 14. It is written in assembly language and compiled using Keil software to generate the HEX code. The HEX program can be burnt into the AT89S52 microcontroller. The software program is designed to accept the input from user as well as control the devices.



```
#ifndef __REG51_H__
#define __REG51_H__

/*  BYTE Register  */
sfr P0   = 0x80;
sfr P1   = 0x90;
sfr P2   = 0xA0;
sfr P3   = 0xB0;
sfr PSW  = 0xD0;
sfr ACC  = 0xE0;
sfr B    = 0xF0;
sfr SP   = 0x81;
sfr DPL  = 0x82;
sfr DPH  = 0x83;
sfr PCON = 0x87;
sfr TCON = 0x88;
sfr TMOD = 0x89;
sfr TL0  = 0x8A;
sfr TL1  = 0x8B;
sfr TH0  = 0x8C;
sfr TH1  = 0x8D;
sfr IE   = 0xA8;
sfr IP   = 0xB8;
sfr SCON = 0x98;
sfr SBUF = 0x99;


/*  BIT Register  */
```

Figure – 17 HEX code using Keil

<u>3.3 Proteus8.1</u>

Proteus 8 is a new application framework lets you view modules of Proteus as tabs in a single window or, via drag and drop, as separate windows for a side-by-side view. We first simulated our hardware virtually on the software then we made it on project board.

CHAPTER 4

INTERFACING OF MICROCONTROLLERS

4.1 Interfacing with LEDS

The LED is a two terminal device. We can therefore characterize it according to two quantities: the voltage across it, and the current through it. To a (fairly good) first order, the light output of the LED, either in photons per second or in milli watts, is linearly proportional to the current through it. This means that it is useful to think of the LED as a current-operated device. Here I will be explaining how to interface a LED to a Microcontroller & a sample code for LED flashing. The adjoining figure shows how to interface the LED to 8051 microcontroller. As you can see the Anode is connected through a resistor to V_{cc} & the Cathode is connected to the Microcontroller pin. So when the Port Pin is HIGH the LED is OFF & when the Port Pin is LOW the LED **is** turned ON [6].

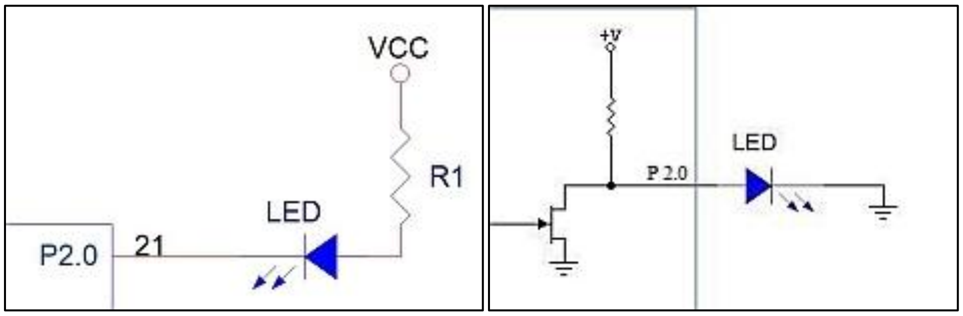


Figure – 18 LED interfacing

8051 has an internal pull-up resistor of 10kΩ. So now when the port Pin is HIGH the Anode is positive with respect to the Cathode so the LED should turn ON right? But the internal pull-up resistor comes in series with the resistor thus limiting the current flowing through the LED. This current is not sufficient enough to Turn ON the LED.

Now the block diagram to interface eight led’s with the 8051 is:

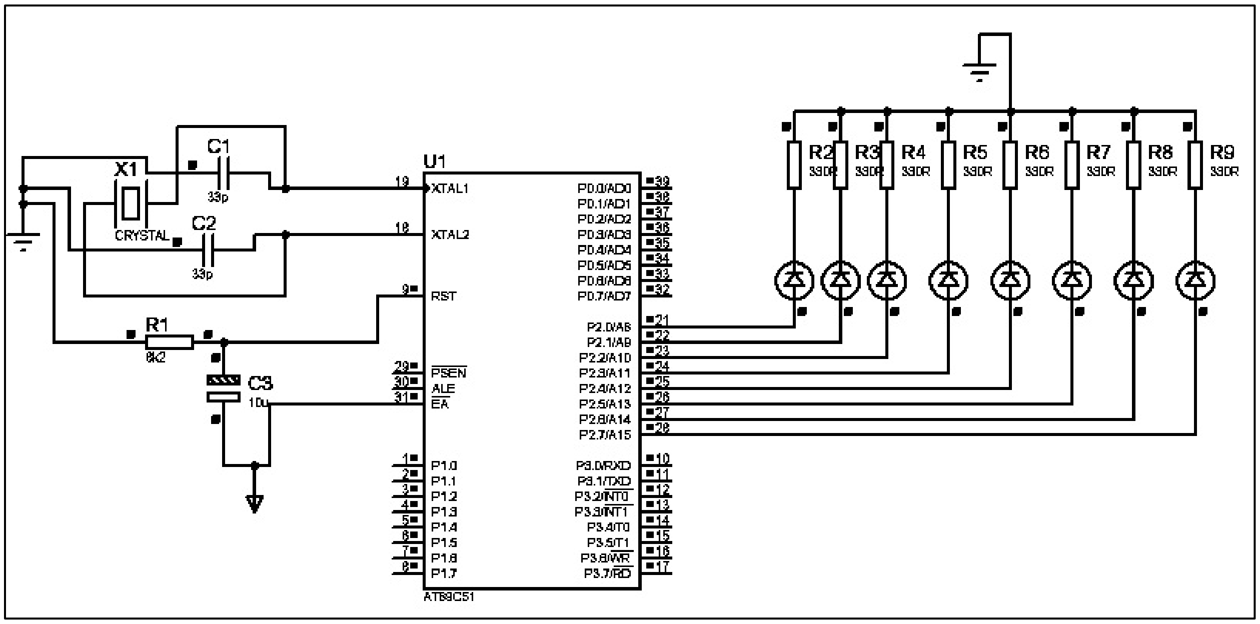


Figure – 19 Eight LED interfacing

4.2 Interfacing with Seven Segment

7 segment displays are basically 7 LED's. It will be much easier to understand if you first read Interfacing LED's to Microcontroller.

Basically there are two types of 7-Seg displays:

1. **Common Cathode**: where all the segments share the same Cathode.
2. **Common Anode**: where all Segments share the same Anode.

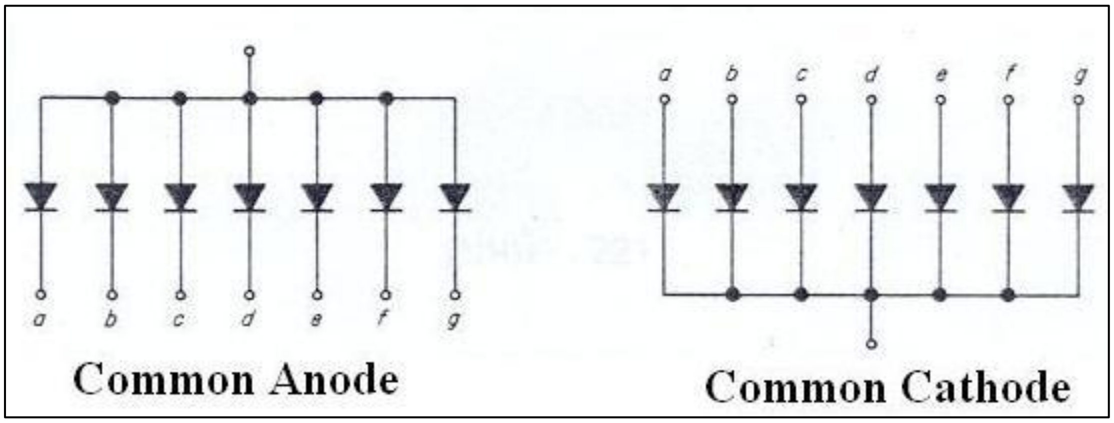


Figure – 20 Types of seven segment displays

Here we will be only discussing the Common Anode type. In common Anode in order to turn ON a segment the corresponding pin must be set to 0. And to turn it OFF it is set to 1.

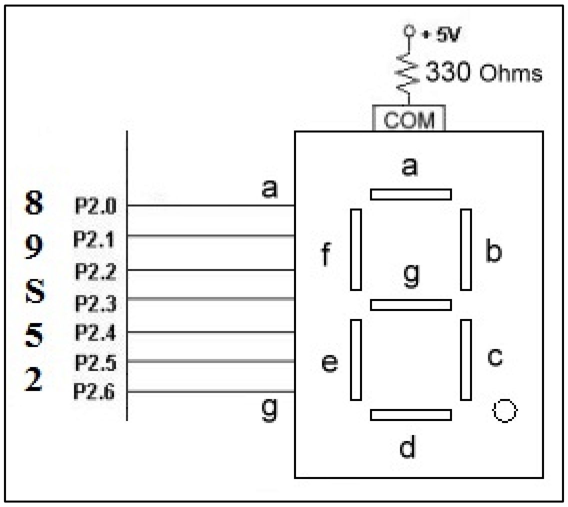


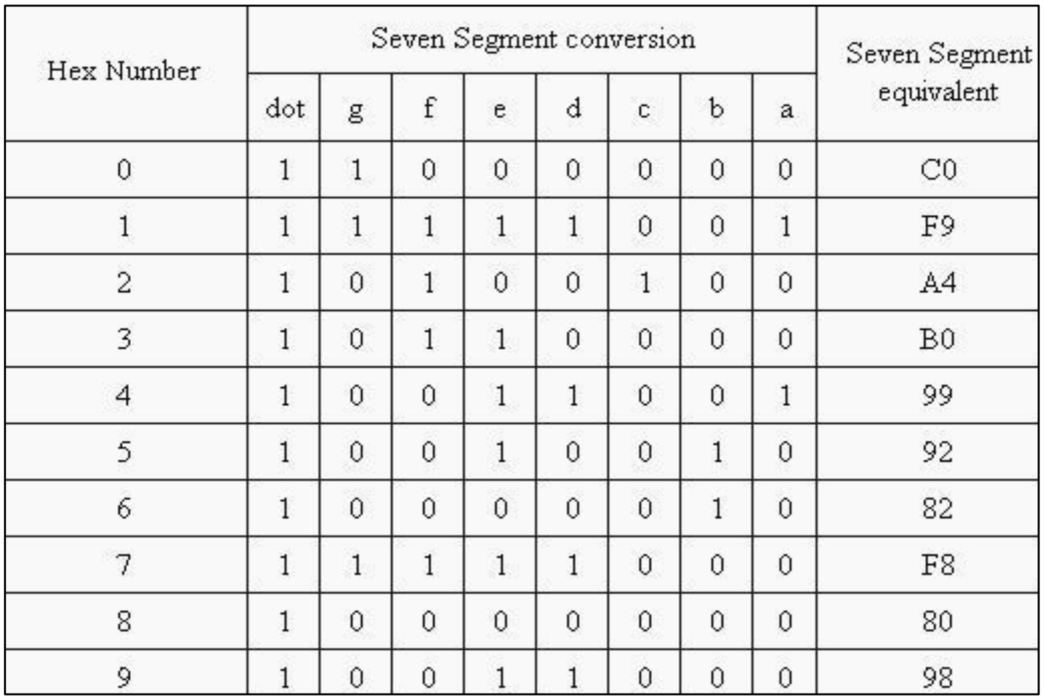
Figure – 21 Interfacing with seven segment display

Now we create a lookup table containing the seven segment pattern to display the corresponding hex digits. e.g. consider we have to display '1' from the above figure we come to know that turning ON segment B & C will show '1' on the 7-seg display so P2.1 & P2.2 should be LOGIC 0 whereas rest of the pins should be LOGIC 1. The different codes to show the different numbers on the seven segments are given below in table.

We can now interface a single 7-Seg to the microcontroller but for interfacing multiple 7-seg's we use Scanning Principle where one 7-seg is displayed after another but this process is very fast hence the flickering cannot be seen by human eye. Figure 3 shows the circuit for interfacing two 7 segment displays.

When interfacing more than one 7-seg display the segment's (A-G) of all displays are connected together whereas their ANODE (Cathode in case of CC displays) are switched ON one after another. Consider we have to display '31' on the above 7-seg display so we TURN ON the first transistor by setting its corresponding pin to 1 & then give the 7-seg equivalent code for '3' which is 4fh.

Table – 9 seven segment conversion table

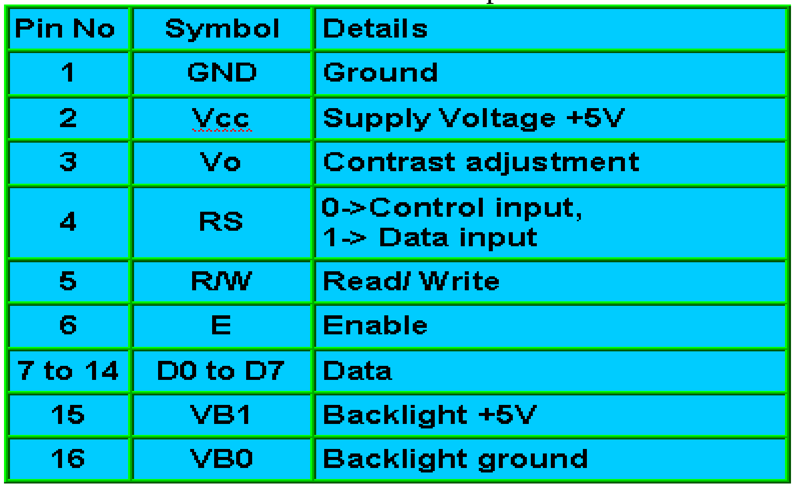


Hex Number	Seven Segment conversion								Seven Segment equivalent
	dot	g	f	e	d	c	b	a	
0	1	1	0	0	0	0	0	0	C0
1	1	1	1	1	1	0	0	1	F9
2	1	0	1	0	0	1	0	0	A4
3	1	0	1	1	0	0	0	0	B0
4	1	0	0	1	1	0	0	1	99
5	1	0	0	1	0	0	1	0	92
6	1	0	0	0	0	0	1	0	82
7	1	1	1	1	1	0	0	0	F8
8	1	0	0	0	0	0	0	0	80
9	1	0	0	1	1	0	0	0	98

Then we TURN OFF the first transistor & TURN ON the second & output its corresponding 7-seg equivalent code of '1' i.e. 06h.Then we again go back to display '3' this is a never ending loop.

4.3 Interfacing with LCD

Table – 10 Pin description of LCD



Pin No	Symbol	Details
1	GND	Ground
2	Vcc	Supply Voltage +5V
3	Vo	Contrast adjustment
4	RS	0->Control input, 1-> Data input
5	R/W	Read/ Write
6	E	Enable
7 to 14	D0 to D7	Data
15	VB1	Backlight +5V
16	VB0	Backlight ground

LCD Initialization:

This is the pit fall for beginners. Proper working of LCD depend on the how the LCD is initialized. We have to send few command bytes to initialize the LCD. Simple steps to initialize the LCD

1. Specify Function Set:

Send **38H** for 8-bit, double line and 5x7 dot character format.

2. Display On-Off Control:

Send **0FH** for display and blink cursor on.

3. Entry Mode Set:

Send **06H** for cursor in increment position and shift is invisible.

4. Clear Display:

Send **01H** to clear display and return cursor to home position.

Algorithm to Send Data to LCD:

1. Make R/W low

2. Make RS=0; if data byte is command

RS=1; if data byte is data (ASCII value)

3. Place data byte on data register

4. Pulse E (HIGH to LOW)

5. Repeat the steps to send another data byte

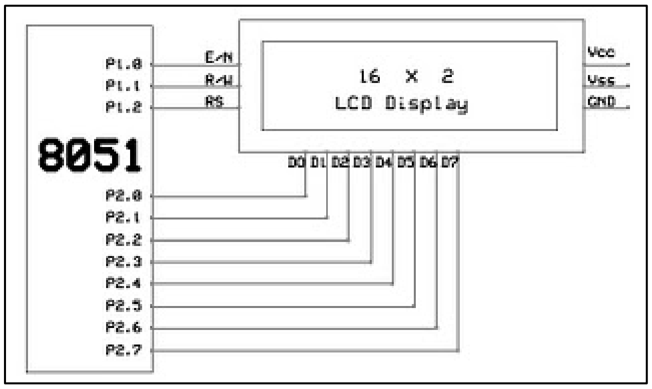


Figure – 22 LCD interfacing with 89S52

Future Scope and Application

Our smart home system is a simulation product for the future life. The purpose of it is to make people's lives more convenient. To replace turn on or turn off on switches by hand for current product, our system is controlled by voice. That should be a trend for the future 10 years which we believe that it is coming to real product soon. When you read the book in bed at night and you feel sleepy and turn off light without leaving bed, so this system become more useful and helps to control lights or fan by voice. This system provides increased quality of life for elderly and disabled persons who might otherwise require caregivers or institutional care.

Applications:

1. **Heating, Ventilation and Air conditioning (HVAC)**
   Due to the wireless network many devices execute simple tasks like lighting, access control, air controlling, all this application can be developed using low-end, low-cost microcontrollers.
2. **Lighting, fan and music systems**
   This system can be used to control various home appliances such as lights, fan and music system etc.
3. **Control and integration of security systems**
   With this system, the consumer can select and watch cameras live from an internet source to their home or business. Security cameras can be controlled, allowing the user to observe activity around a house or business right from a monitor or touch panel. Security systems can include motions sensors that will detect any kind of unauthorized movement and notify the user through the security system or via cell phone.
4. **Other systems**
   Using special hardware, almost any household appliances can be monitored and controlled automatically or remotely, including
   - Coffee maker
   - Garage door
   - Pet feeding and watering
   - Plant watering

References

1. Jackson Muhirwe, “AUTOMATIC SPEECH RECOGNITION: HUMAN COMPUTER INTERFACE FOR KINYARWANDA LANGUAGE”. August 2005
2. Thomas Jorgensen, HW Manager, Niels T. Johansen, VP R&D Zensys A/S Denmark, “Z-Wave™ as Home Control RF Platform”.
3. Jurafsky D., Martin J. (2000). Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition. Delhi, India: Pearson Education.
4. Subrata Ghoshal, “8051 Microcontroller: Internals, Instructions, Programming and Interfacing”, 1st Edition, Publisher: DORLING KINDERSLEY (RS).
5. Deshmukh, N., Ganapathiraju, A, Picone J., (1999), Hierarchical Search for Large Vocabulary Conversational Speech Recognition. IEEE Signal Processing Magazine, 1(5):84-107.
6. Mane, A., Boyce,S., Karis,D.,Yankelovich,N., (1996) “Designing the User Interface for Speech Recognition Applications” SIGCHI Bulletin 28(4):29-34.

Appendix A
Source code

Code on AT89S52

```
#include<reg51.h>
sbit sw0=P2^0;
sbit sw1=P2^2;
sbit sw2=P2^4;
sbit rs=P2^5;
sbit rw=P2^6;
sbit en=P2^7;
sbit busy=P0^7;
sbit relay=P3^7;
#define lcd P0
void transmit(unsigned char *a);
void dataa(unsigned char );
void busycheck();
void cmd(unsigned int);
unsigned int ab,cd,led6,led7;
void delay(unsigned int );
unsigned int recieve,n,abc=0;
unsigned long int timer,pl=0;
void display( unsigned char *);
void transmit_1( unsigned char a);
void serial (void)interrupt 4
{
        if(TI==1)
{
TI=0;
cd=0;
}
if(RI==1)
{
RI=0;
        recieve=SBUF;
        abc=1;

}

}
code unsigned char
array[107]={0x62,0x62,0x78,0x20,0x62,0x62,0x78,0x20,0x6F,0x46,0x6D,0x20,0x20,0x20,
0x20,0x20,0x20,0x20,0x20,0x63,0x49,0x20,0x70,0x49,0x41,0x20,0x20,0x20,0x20,0x20,0x20,0
x70,0x49,0x42,0x20,0x20,0x20,0x20,0x20,
0x63,0x4A,0x20,0x63,0x4B,0x20,0x63,0x4C,0x20,0x63,0x4D,0x20,0x63,0x4E,0x20,0x63,0x4F
,0x20,0x63,0x50,0x20,0x63,0x51,0x20,
0x70,0x51,0x41,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x63,0x41,0x20,0x70,0x41,0x41,0x20,0
x20,0x20,0x20,0x20,0x63,0x42,0x20,
0x70,0x42,0x41,0x20,0x20,0x20,0x20,0x20,0x63,0x48,0x20,0x70,0x48,0x41,0x20,0x20,0x20,0
x20,0x20,0x00};


void main()
{
        TMOD=0X20;
SCON=0X50;
IE=0X92;
TH1=0XFd;
TR1=1;
        relay=0;

        busycheck();

        cmd(0x38);
                cmd(0x0c);
        start:
                cmd(0x01);
        cmd(0x80);
                display(" VOICE CONTROL  ");
        recieve=0;
        do
        {
        transmit_1(0xf0);
        transmit_1(0xff);
delay(500);
}       while(recieve!=0x76);

cmd(0xc0);
        display(" COMM SUCCESSFULL ");


        delay(100);

transmit(&array[0]);
        delay(1000);
        recieve=0;
transmit_1(0x64);
transmit_1(0x49);
        timer=200000;
while((--timer)&&(recieve==0));
if(timer==0)
{
```

```
          goto start;
}
if((recieve==0x76)||(recieve==0x72)||(recieve==0x74)||(recieve==0x65)||(recieve==0x6f))
{
;
}
else
{
          goto start;
}

cmd(0xc0);
          display("READY TO LISTEN ");
while(1)
{
                    recieve=0;
          transmit_1(0x64);
transmit_1(0x49);
timer=200000;//9sec
while((--timer)&&(recieve==0));
if(timer==0)
{
          cmd(0x80);
          display(" MODEM  FAILED  ");
}
else if(recieve==0x72)
{
recieve=0;
transmit_1(0x20);
timer=20000;
while((--timer)&&(recieve==0));
if((recieve==0x41)||(recieve==0x43)||((recieve==0x45))||((recieve==0x47)))
          {
                    relay=1;
                    cmd(0x80);
          display("    BULB ON    ");
}
if((recieve==0x42)||(recieve==0x44)||((recieve==0x46))||((recieve==0x48)))
          {
                              relay=0;
          cmd(0x80);
          display("   BULB OFF    ");
}
}
else if(recieve==0x65)
{
```

```
cmd(0xc0);
          display("  RECOG. FAILED ");
}
else if(recieve==0x74)
{
cmd(0xc0);
          display("TIME OUT EXPIRED ");
                    delay(100);
}
}
}
void delay(unsigned int kk)
{
unsigned int yy,jj;
for(yy=0;yy<kk;yy++)
{
for(jj=0;jj<111;jj++);
}
}
void cmd(unsigned int h)
{
          rs=0;
          rw=0;
          lcd=h;
          en=1;
          delay(1);
          en=0;
}
void dataa(unsigned char h)
{
          rs=1;
          rw=0;
          lcd=h;
          en=1;
          delay(1);
          en=0;
}
void transmit( unsigned char *a)
{

          while(*a!='\0')
          {
                    cd=1;
                    SBUF=*a;
                    while(cd==1);
                    a++;
```

```
	}

}
void transmit_1( unsigned char a)
{


		cd=1;
		SBUF=a;
		while(cd==1);
}
void busycheck()
{
	rs=0;
	rw=1;
	do
		{
			en=0;
			delay(1);
			en=1;

	}
	while(busy==1);
}
void display( unsigned char *c)
{
	while(*c!='\0')
	{
	dataa(*c);
		c++;

	}
}
```

Appendix B
Components

B.1 Transformer

A transformer is a device that transfers electrical energy from one circuit to another through inductively coupled conductors—the transformer's coils. If a load is connected to the secondary, an electric current will flow in the secondary winding and electrical energy will be transferred from the primary circuit through the transformer to the load. In an ideal transformer, the induced voltage in the secondary winding (Vs) is in proportion to the primary voltage (Vp), and is given by the ratio of the number of turns in the secondary (Ns) to the number of turns in the primary (Np) as follows:

$$\frac{V_S}{V_P} = \frac{N_S}{N_P}$$

By appropriate selection of the ratio of turns, a transformer thus allows an alternating current (AC) voltage to be "stepped up" by making Ns greater than Np, or "stepped down" by making Ns less than Np.

In the vast majority of transformers, the windings are coils wound around a ferromagnetic core, air-core transformers being a notable exception.
Here we use a 12-0-12 transformer. 12-0-12 means that the voltage or the potential difference (P.D.) between each of the end terminals of the secondary winding and the mid-point of the secondary winding of the transformer is 12V. And, between the two ends of the secondary winding, you will get 12 + 12 = 24V. 500mA means the current delivery capability of the secondary winding of the transformer. Normally it is said in VA. In your case it would be 25 x 0.5 = 12VA. The ratings are arrived at based on the requirements of the loads that are to be connected to the transformer. The limiting criteria are the winding wire thickness and the insulation of the winding.

B.2 Transistor

Transistors amplify current, for example they can be used to amplify the small output current from a logic IC so that it can operate a lamp, relay or other high current device. In many circuits a resistor is used to convert the changing current to a changing voltage, so the transistor is being used to amplify voltage.
A transistor may be used as a switch (either fully on with maximum current, or fully off with no current) and as an amplifier (always partly on).The amount of current amplification is called the current gain, symbol hFE.

B.3 Connectors

DB9 connector

U.S.A. PATENT 5199906 GENDER CHANGER
U.S.A. PATENT 5199906 GENDER CHANGER

The term "DB9" refers to a common connector type, one of the D-Subminiature or D-Sub types of connectors. DB9 has the smallest "footprint" of the D-Subminiature connectors, and houses 9 pins (for the male connector) or 9 holes (for the female connector).
DB9 connectors were once very common on PCs and servers. DB9 connectors are designed to work with the EIA/TIA 232 serial interface standard, which determined the function of all nine pins as a standard, so that multiple companies could design them into their products. DB9 connectors were commonly used for serial peripheral devices like keyboards, mice, joysticks, etc. Also they are used on DB9 cable assemblies for data connectivity.

Today, the DB9 has mostly been replaced by more modern interfaces such as USB, PS/2, Fire wire, and others. However, there are still many legacy devices that use the DB9 interface for serial communication.

Crocodile clips

The ‘standard’ crocodile clip has no cover and a screw contact. However, miniature insulated crocodile clips are more suitable for many purposes including test leads. They have a solder contact and lugs which hold down to grip the cable’s insulations, increasing the strength of the joint. Remember to feed the cable through the plastic cover before soldering! Add and remove the cover by fully opening the chip, a piece of wood can be used to hold the jaws open.

B.4 Tools required for Electronics

Soldering iron and stand

A soldering iron is a hand tool used in soldering. There are many soldering irons available on the market. They come in a variety of sizes and shapes. Which soldering iron to choose for yourself depends on the soldering projects you are planning to do, as well as how often you are planning on using it. This instructable will cover choosing a soldering iron that will be used for projects in electronics for soldering and de-soldering work on the circuit boards.

The four main factors to consider when choosing a soldering iron are:
1) Wattage
2) Type of the soldering iron
3) Temperature control
4) Tip size and shape

Wattage
The wattage of the soldering iron is one of the most important factor of a soldering iron. Most of soldering irons used in the electronics are in range 20 – 60 Watts. Soldering iron with wattage 50W is very common these days and it will provide sufficient heat for most of soldering projects on the circuit boards. Soldering irons with higher wattage (40W -60W) are better. It does not mean that soldering irons with higher wattage apply more heat to the solder joint- it means that soldering irons with higher wattage has more power available. Since most of soldering stations comes with knob on power station for setting iron's temperature, it is possible to regulate amount of the heat on iron tip. On the other hand, soldering iron with low wattage (20W - 30W) can lose heat faster than it can re-heat itself - this results in bad solder joints.

Types of soldering irons
Generally, there are 4 different types of soldering irons:
-Soldering pencil
-Soldering station
-Soldering systems (rework /repair stations)
-Soldering guns

Desoldering pump
(solder sucker):

Tool for removing solder when desoldering a joint to correct a mistake or replace a component.

Solder remover wick (copper braid):

This is the alternative to the desoldering pump shown above.

Solder wire

The best size for electronics is 22 SWG (SWG = standard wire gauge).

Side cutters

For trimming component leads close to the circuit board.

Wire strippers

Most designs include a cutter as well, but they are not suitable for trimming component leads.

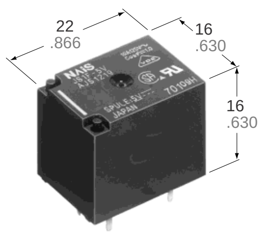
Small pliers
Usually called ‘snipe nose’ pliers, these are for bending component leads etc. If you put a strong rubber band across the handles the pliers make a convenient holder for parts such as switches while you solder the contacts.

Small flat-blade screwdriver

For scraping away excess flux and dirt between tracks, as well as driving screws!

NAiS

Relay Data sheet



mm inch

FEATURES

- **Ultra-miniature size with universal terminal footprint**
- **High contact capacity: 10 A**
- **Class B coil insulation type available**
- **TV-5 type available**
  1 Form A type $\longrightarrow$ TV-5
  1 Form C type $\longrightarrow$ TV-5 (N.O. side only)
- **VDE, TÜV also approved**
- **Sealed construction for automatic cleaning**

SPECIFICATIONS

Contact

Arrangement		1 Form A, 1 Form C
Initial contact resistance, max. (By voltage drop 6 V DC 1 A)		100 m&
Contact material		Silver alloy
Rating (resistive load)	Nominal switching capacity	10 A 250 V AC 10 A 125 V AC 6 A 277 V AC
	Max. switching power	2,500 VA
	Max. switching voltage	250 V AC, 100 V DC
	Max. switching current	10 A (AC), 5 A (DC)
Expected life (min.ope.)	Mechanical (at 180 cpm)	10^7
	Electrical at 10 A 125 V AC, 6 A 277 V AC resistive (at 20 cpm)	10^5
	10 A 250 V AC resistive (at 20 cpm)	5 - 10^4

Coil

Nominal operating power	360 mW

Remarks

* Specifications will vary with foreign standards certification ratings.
*[1] Detection current: 10mA
*[2] Excluding contact bounce time
*[3] Half-wave pulse of sine wave: 11ms; detection time: 10µs
*[4] Half-wave pulse of sine wave: 6ms
*[5] Detection time: 10µs
*[6] Refer to 5. Conditions for operation, transport and storage mentioned in AMBIENT ENVIRONMENT (Page 24).

Characteristics

Max. operating speed		20 cpm
Initial insulation resistance		Min. 100 M& (at 500 V DC)
Initial breakdown voltage*[1]	Between open contacts	750 Vrms for 1 min.
	Between contacts and coil	1,500 Vrms for 1 min.
Operate time*[2] (at nominal voltage)		Approx. 10 ms
Release time(without diode)*[2] (at nominal voltage)		Approx. 10 ms
Temperature rise (at nominal voltage)		Max. 35°C
Shock resistance	Functional*[3]	Min. 98 m/s^2 {10 G}
	Destructive*[4]	Min. 980 m/s^2 {100 G}
Vibration resistance	Functional*[5]	Approx. 98 m/s^2 {10 G}, 10 to 55 Hz at double amplitude of 1.6 mm
	Destructive	Approx. 117.6 m/s^2 {12 G}, 10 to 55 Hz at double amplitude of 2 mm
Conditions for operation, transport and storage*[6] (Not freezing and condensing at low temperature)	Ambient temp.*[7]	–40°C to +85°C –40°F to +185°F
	Humidity	5 to 85% R.H.
Unit weight		Approx.12 g .423 oz

*[7] When using relays in a high ambient temperature, consider the pick-up voltage rise due to the high temperature (a rise of approx. 0.4% V for each 1°C 33.8°F with 20°C 68°F as a reference) and use a coil impressed voltage that is within the maximum allowable voltage range

TYPICAL APPLICATIONS

1. Home appliances
   Air conditioner, heater, etc.
2. Automotive
   Power-window, car antenna, door-lock, etc.

ORDERING INFORMATION

Ex. JS [1a] — [F] — [B] — [12V]

Contact arrangement	Protective construction	Coil insulation class	Coil voltage (DC)
1: 1 Form C 1a: 1 Form A	Nil: Sealed type F: Flux-resistant type	Nil: Class E insulation B: Class B insulation	5, 6, 9, 12, 18, 24, 48 V

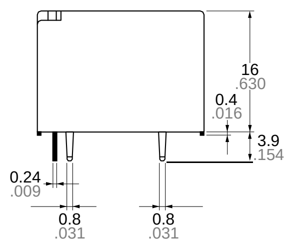
UL/CSA, VDE, TÜV approved type is standard.
Notes: 1. Standard packing: Carton: 100 pcs. Case: 500 pcs.
2. When ordering TV rated (TV-5) types, add suffix -TV.

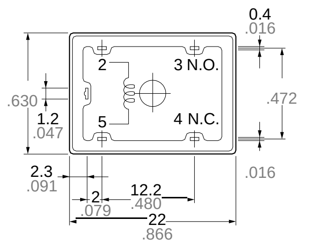
COIL DATA

Part No.				Nominal voltage, V DC	Pick-up voltage, V DC (max.) (at 20°C68°F)	Drop-out voltage, V DC (min.) (at 20°C68°F)	Coil resistance, & (±10%) (at 20°C68°F)	Nominal operating current, mA (±10%) (at 20°C68°F)	Nominal operating power, mW (at 20°C68°F)	Max. allowable voltage (at 85°C185°F)
Sealed type		Flux-resistant type								
1 Form A	1 Form C	1 Form A	1 Form C							
JS1a-5V	JS1-5V	JS1aF-5V	JS1F-5V	5	3.5	0.5	69.4	72	360	130%V of nominal voltage
JS1a-6V	JS1-6V	JS1aF-6V	JS1F-6V	6	4.2	0.6	100	60		
JS1a-9V	JS1-9V	JS1aF-9V	JS1F-9V	9	6.3	0.9	225	40		
JS1a-12V	JS1-12V	JS1aF-12V	JS1F-12V	12	8.4	1.2	400	30		
JS1a-18V	JS1-18V	JS1aF-18V	JS1F-18V	18	12.6	1.8	900	20		
JS1a-24V	JS1-24V	JS1aF-24V	JS1F-24V	24	16.8	2.4	1,600	15		
JS1a-48V	JS1-48V	JS1aF-48V	JS1F-48V	48	33.6	4.8	6,400	7.5		

DIMENSIONS

mm inch

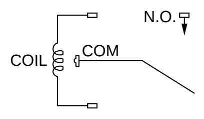




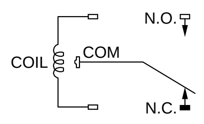
Note: Terminal No. 4 is only for 1 Form C type
General tolerance: ±0.3 ±.012

Schematic (Bottom view)

1a

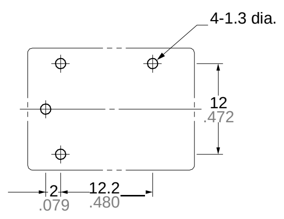


1c

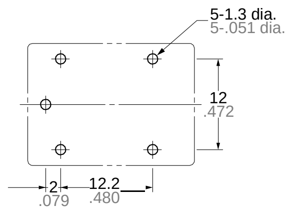


PC board pattern (Copper-side view)

1a



1c



Tolerance: ±0.1 ±.004