Features:

- ATMEL AT89S52 device.
- 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
- Interfacing of 8 bit ADC (AD0804) by connecting a variable voltage supply to the i/p of ADC and by reading the 8 bit output of ADC and display it on 16X2 LCD.
- 8 digit SMD LED interface.
- Interface DAC to generate different waveforms (like sine, square, triangular and ramp).
- Hex keypad / calculator interface.
- Interface stepper motor and control its speed and direction.
- Interface push button switch & 4 digit seven segment display, count the number (from 0 to 9999) of times the switch is pressed and display it on seven segment display.
- Interface 16x2 LCD modules.
- DC motor interface.
- Buzzer/Speaker interface for tone generation.
- USB Flash Programmer.
- RS 232 DB9 serial port.
- RTC (PCF8563) interface.
- EEPROM (24C04) interface.
- 12V/5V DC power supply through power adaptor.

General description

VT8051-2.2 provides a hardware platform for developing embedded system using ATMEL AT89S52 device.

These features make VT8051-2.2 board ideal for instrumentation, communication and other demanding application areas where flexibility and in-circuit hardware upgradeability is of paramount important.

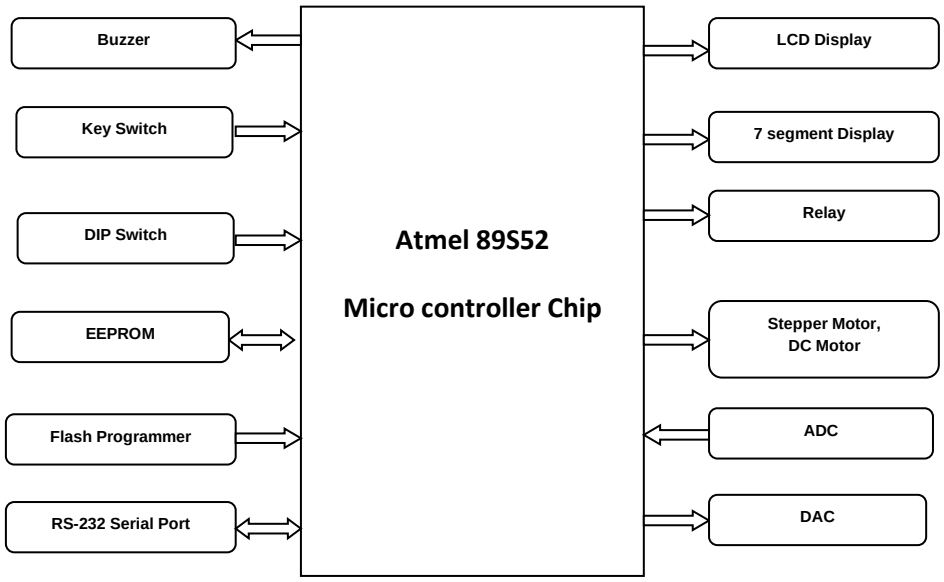
VT8051-2.2 board comes with complete drivers for windows98/Me/2000/XP/2007. 8051 code examples and windows DLL interface which can be used to interface it to most common programming language.

As it comes with complete with all Keil development software required and example code. it is also ideal as classroom training too for colleges and universities as well as engineers wanting to learn more about hardware development using 8051.

VT8051-2.2 board consists of the following.

- VT8051-2.2 board fitted in soft wooden box.
- 12v Dc power adaptor for Relay and stepper motor operation
- USB cable for HEX file downloading and 5V dc power for VT8051-2.2 board.
- Stepper motor.
- DC motor.
- Technical/operational Manual with hardware schematics and sample programs.
- *Software CD.*

Block Diagram of VT 8051-2.2 Microcontroller kit



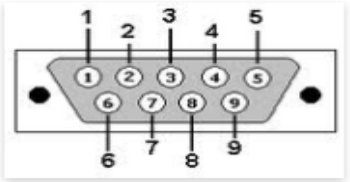
ATMEL AT 89S52 Device.

U9 is an ATMEL AT 89S52 Device in a 40 DIP package

Table1. ATMEL AT89S52 device features	
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	

Serial Port Connector connection details:

J10 is a standard DB9 serial connector. This connector is typically used for communication with host computer using a standard 9-pin serial cable connected to a Com port. Level shifting buffer U8 (MAX232) is used between computer and 89S52 device.VT8051-2.2 uses only the signals it needs, such as RXD and TXD. Fig (2) shows the pin connection between the serial connector and 89s52 device.
Table (2) shows DB9 connector pin



Fig(2)

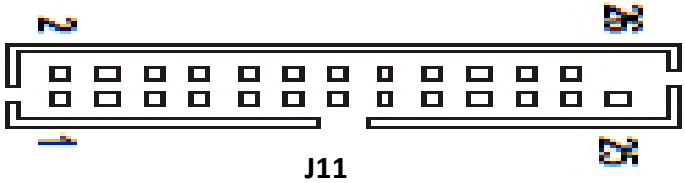
Table (2) DB9 connector pin Table.

DB9(J10) Connector Pin	1	2(TXD)	3(RXD)	4	5	6	7	8	9
89S52 Pin No	-	10	11	1	GND	-	-	-	-
89S52 Port No	-	P3.0	P3.1	-	-	-	-	-	-

Expansion Prototype connector

The power and I/O pins of the 89S52 device are brought out onto 26 pin Box header (J11).This can be used for external special daughter board card interfaces. Fig (3) and table (3) shows connections from Expansion connector.

Fig (3) Expansion connector.



J11

Table (3) Expansion connector pin Table.

***Expansion Prototype connector* J11 Pins**	**89S52 Pin No**	**89S52 Port No**
1	1	P1.0
2	2	P1.1
3	3	P1.2
4	4	P1.3
5	5	P1.4
6	6	P1.5
7	7	P1.6
8	8	P1.7
9	21	p2.0
10	22	P2.1
11	23	P2.2
12	24	P2.3
13	25	P2.4
14	26	P2.5
15	27	P2.6
16	28	P2.7
17	10	P3.0
18	11	P3.1
19	12	P3.2
20	13	P3.3
21	14	P3.4
22	15	P3.5
23	16	P3.6
24	17	P3.7

25	-	VCC
26	-	GND

RTC Pin Configuration

The Real Time Clock (RTC) is a set of counters for measuring time when system power is on, and optionally when it is off

U11 is a RTC(PCF8963) *device connected to AT89S52 Device and can be used for Real time clock applications.* On the 89S52, the RTC can be clocked by a separate 32.768 KHz oscillator. Also, the RTC is powered by its own power supply pin, Vdd, which can be connected to a battery or to the same 5 V supply used by the rest of the device.

Features.

- Measures the passage of time to maintain a calendar and clock.
- Ultra Low Power design to support battery powered systems.
- Provides Seconds, Minutes, Hours, Day of Month, Month, Year, Day of Week, and Day of Year.
- Dedicated 32 kHz oscillator.
- Dedicated power supply pin can be connected to a battery or to the main 5V.

NOTE: Switch on DSW14 and switch off all other dip switches.

Table (4) RTC pin connection Table.

PCF8563 Pin no	1	2	3	4	5	6	7	8
PCF8563 Pin Function	X1	X2	INT	VSS	SDA	SCL	CLK OUT	VDD
89S52 Pin No	-	-	-	GND	10	11	-	-
89S52 Port No	-	-	-	-	P3.0	P3.1	-	-

EEPROM Device:

U11 is an 2K E2PROM device connected to AT89S52 Device and can be used for external memory interface applications. Table (5) shows pin configuration.

Table (5) EEPROM Pin connection table.

24C04 Pin no	**24C04 Pin Function**	**89S52 Pin No**	**89S52 Port No**
1	A0	-	-
2	A1	-	-
3	A2	-	-
4	GND	-	-
5	SDA	10	P3.0
6	SCL	11	P3.1
7	WP	-	-
8	5V	-	-

NOTE: Switch on DSW13 and switch off all other dip switches.

BUZZER & SPEAKER Pin Configuration.

A **buzzer** or **beeper** is an audio signaling device, which may be mechanical, electromechanical, or piezoelectric. Typical uses of buzzers and beepers include alarm devices, timers and confirmation of user input such as a mouse click or keystroke. LS1 is a small 12mm round buzzer that operates around the audible 2kHz range. We drove it through Q5 which in turn connected to P3.0 via DIP switch DSW11.We need to send Logic '1' to generate beep.

A speaker is a device which converts electrical signal to audible signal.J9 is a two pin RMC where we are connecting external speaker. We drove it through Q5 which in turn connected to P3.0 via DIP switch DSW11.We need to send Logic '1' to generate Tone.

NOTE: Switch on DSW14 and switch off all other dip switches. Jumper JP2 should be shorted between 1and 2 for Buzzer and shorted between 2 and 3 for speaker. Table (6) shows speaker /Buzzer pin connection Table

Table (6) Speaker /Buzzer pin connection Table.

Speaker/ Buzzer Pin	89S52 Pin No	Port No
1	10	P3.0

DC Motor Pin Configuration

DC (direct current) motor rotates continuously. It has two terminals positive and negative. Connecting DC power supply to these terminals rotates motor in one direction and reversing the polarity of the power supply reverses the direction of rotation.

The speed of Dc motor can maintained at a constant speed for a given load by using "Pulse Width Modulation (PWM)" technique. By changing the width of the pulse of applied to dc motor, the power applied is varied thereby DC motor speed can be increased or decreased. Wider the pulse Faster is the Speed, Narrower is the Pulse, and Slower is the Speed

U5 is L293 driver IC used to drive the dc motor. It has two enable lines which are used to switch on the required DC motor. It is connected to P3.0 and P3.1 through Dip switch DSW8.Logic '1' enables the driver and logic '0' disables the driver.P2.0 and P2.1 which are connected through Dip switch DSW7 are used for Motor 1 direction and speed control.
Table (7) shows pin

NOTE: Switch on DSW7 ,DSW8 and switch off all other dip switches. Connect the DC Motor at J3

Table (7) DC Motor Pin connection table.

Motor Selection	Motor Direction	89S52 Pin No	89S52 Port No
Motor-1 P3.0	Clk	21&22	P2.0=1 & P2.1=0
	Aclk		P2.0=0 & P2.1=1

Motor-2	Clk	23 & 24	P2.2=1 & P2.3=0
P3.1	Aclk		P2.2=0 & P2.3=1

Stepper Motor Configuration:

A ***stepper motor*** is a special type of electric motor that moves in increments, or steps, rather than turning smoothly as a conventional motor does. Typical increments are 0.9 or 1.8 degrees, with 400 or 200 increments thus representing a full circle. The speed of the motor is determined by the time delay between each incremental movement.

U6 is a Driver Buffer (ULN2003) *device connected to AT89S52 Device and can be used for driving Stepper Motor.* On the 89S52, P2.0 to P2.3 are used to generate the pulse sequence required to run the stepper Motor. Also, the Stepper Motor is powered by its own power supply pin (COM), which is connected to a 12V supply. Table (8) shows connections for stepper Motor.

NOTE: Switch on DSW9 and switch off all other dip switches. Connect the stepper motor at J5

Table (8) Stepper Motor Pin connection table.

Stepper Motor Coil	89S52 Pin No	89S52 Port No
A	21	P2.0
B	22	P2.1
C	23	P2.2
D	24	P2.3

Relay Configurations:

A **relay** is an electrically operated switch. Many relays use an electromagnet to operate a switching mechanism mechanically, but other operating principles are also used. Relays are used where it is necessary to control a circuit by a low-power signal (with complete electrical isolation between control and controlled circuits), or where several circuits must be controlled by one signal. The first relays were used in long distance telegraph circuits, repeating the signal coming in from one circuit and re-transmitting it to another. Relays were used extensively in telephone exchanges and early computers to perform logical operations.

K1 is an electromagnetic relay which is connected to P3.0 through Dip switch DSW10.We need to send logic '1' to switch on relay.J6 is three pin PBT terminal used to connect external device to relay. Table (9) shows connections for Relay.

NOTE: Switch on DSW10 and switch off all other dip switches.

Table (9) Relay connection table.

Relay coil	89S52 Pin No	89S52 Port No
Relay 1	10	P3.0
Relay2	11	P3.1

HEX KEY PAD Pin configurations:

The hex keypad is a peripheral that is organized in rows and Columns. Hex key Pad 16 Keys arranged in a 4 by 4 grid, labeled with the hexadecimal digits 0 to F. An example of this can been seen in Figure 1, below. Internally, the structure of the hex keypad is very simple. Wires run in vertical columns (we call them C0 to C3) and in horizontal rows (called R0 to R3). These 8 wires are available externally, and will be connected to the lower 8 bits of the port. Each key on the keypad is essentially a switch that connects a row wire to a column wire. When a key is pressed, it makes an electrical connection between the row and column. Table (10) shows connections for HEX KEY Pad matrix.

NOTE: Switch on DSW6 and switch off all other dip switches.

Table (10) HEX KEY Pad matrix table.

ROW/COLOUMNS	ROW1	ROW2	ROW3	ROW4	COL1	COL2	COL3	COL4
89S52Pin No	1	2	3	4	5	6	7	8
89S52Port No	P1.0	P1.1	P1.2	P1.3	P1.4	P1.5	P1.6	P1.7

USER KEY PAD Pin configurations:

VT8051-2.2 board provides eight individual KEYS connected to 89S52 device through PORT1. S1 to S6,S11and S16 are connected to general purpose I/O pins on 89s52 device as shown in table(11). 89S52 device port pin drives to Logic '0' when the corresponding key is pressed. Table (11) shows connections for USER KEY Pad connection.

Table (11) USER KEY Pad connection table.

USER DEFINED KEYS	S1(K1)	S2(KY2)	S3(KY3)	S4(KY4)	S5(KY5)	S6(KY6)	S11(KY7)	S16(KY8)
89S52 Pin No	1	2	3	4	5	6	7	8
89S52 Port No	P1.0	P1.1	P1.2	P1.3	P1.4	P1.5	P1.6	P1.7

DAC Pin configurations:

U11 is a DAC(AD5330) enables the 89S52 to generate a variable analog output. The maximum DAC output voltage is the V_{REF} (5V) voltage.

Features:

- 8 bit digital to analog converter
- Resistor string architecture
- Buffered output
- Power-down mode
- Selectable speed vs. power

In 89S52 the digital inputs are converter to analog output voltage at pin 4(DACOUT). The total voltage provided by the DACOUT pin is function of binary number written on to pins 13(DB0) to pin 20(DB7) of U2 DAC (AD5330) and Pin 7 WRITE pin, we need to send a high to low pulse for each conversion. Table (12) shows DAC pin connection Table

NOTE: Switch on DSW4,DSW5 and switch off all other dip switches.Connect the CRO probe between TP3 and TP4

Table (12) DAC Pin connection table.

DAC Pin No	DAC Pin Functions	89S52 Pin No	89S52 Port No
1	BUF	-	-
2	NC	-	-
3	VREF	-	-
4	DACOUT	-	-
5	GND	-	-
6	$\overline{CS}$	-	-
7	$\overline{WR}$	10	P3.0
8	GAIN	-	-
9	$\overline{CLR}$	-	-
10	$\overline{LDAC}$	-	-
11	PD	-	-
12	VDD	-	-
13	DB0(LSB)	28	P2.7
14	DB1	27	P2.6
15	DB2	26	P2.5
16	DB3	25	P2.4
17	DB4	24	P2.3
18	DB5	23	P2.2
19	DB6	22	P2.1
20	DB7(MSB)	21	P2.0

ADC PIN Configuration:

Analog to digital converters are the most widely used devices for data acquisition. It is very much essential to convert the physical quantities like pressure, humidity, temperature and velocity in to digital data for processing with digital computers. First these physical quantities are converted into analog signals (either voltage or current) with the help of transducers or sensors. Then these analog signals are converted into digital data using Analog to digital converters (ADC), so that microcontrollers or processors can read digital data and process them. An ADC has n-bit resolution where n can be 8,10,12,16 or 24 bits. The higher resolution ADC can provide a smaller step size, where step size is the smallest change in analog data that an ADC can convert in to digital data.

U1 is an 8 bit **ADC (0804**) connected to 89S52 device through Dip switch DSW2 and DSW3. Table (13) shows the ADC pin connection table.

NOTE: Switch on DSW2,DSW3 and keep 1 on, 2 off in DSW1 .Switch off all other dip switches. Measure the applied analog voltage between P3 and P4.

Table (13) ADC Pin connection table.

ADC Pin No	ADC Pin Functions	89S52 Pin No	89S52 Port No
1	$\overline{CS}$	-	-
2	$\overline{RD}$	-	-
3	$\overline{WR}$	10	P3.0
4	CLK	-	-
5	$\overline{INTR}$	11	P3.1
6	VI+	-	-
7	VI-	-	-
8	AGND	-	-
9	VREF	-	-
10	GND	-	-
11	DB7(MSB)	28	P2.7
12	DB6	27	P2.6
13	DB5	26	P2.5
14	DB4	25	P2.4
15	DB3	24	P2.3
16	DB2	23	P2.2
17	DB1	22	P2.1
18	DB0(LSB)	21	P2.0

LED Pin Configuration

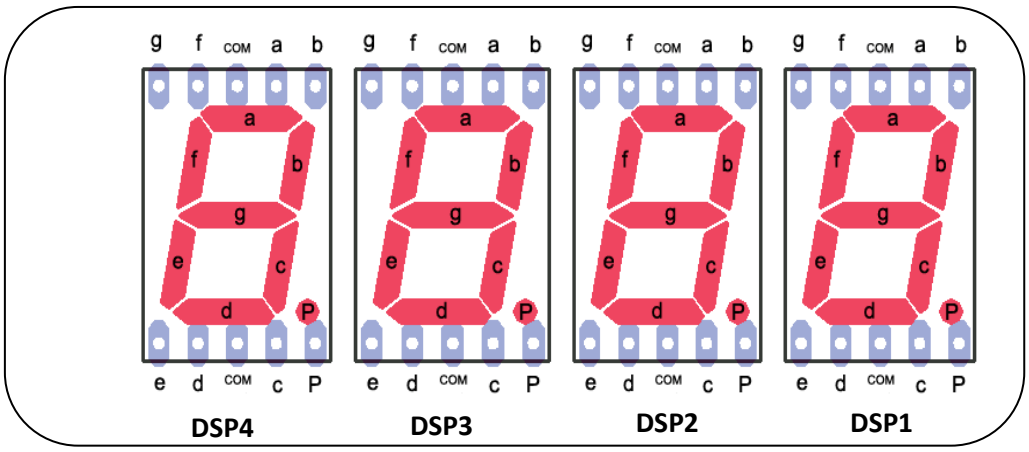
VT8051-2.2 board provides eight individual SMD led's connected to 89S52 device through 74HC151driver IC.D3 to D10 are connected to general purpose I/O pins on 89s52 device as shown in table(14) When 89S52 device drives Logic '1' the corresponding LED turns on.

Table (14) LED Pin connection table.

LED	*D3*	*D4*	*D5*	*D6*	*D7*	*D8*	*D9*	*D10*
89S52 Pin No	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*
89S52 Port No	*P2.0*	*P2.1*	*P2.2*	*P2.3*	*P2.4*	*P2.5*	*P2.6*	*P2.7*

SEVEN SEGMENT DIPLAY Pin Configuration:

D11, D12, D13 and D14 are Common Cathode segments connected to 89S52 device so that each segment is individually controlled by a general purpose I/O pin. When the 89S52 device drives logic '0' the corresponding segment turns on. See fig(3) and table(15) for more details.



fig(3)

Table (15) Seven Segment Pin connection table.

Seven Segment Data Lines	*g*	*f*	*a*	*b*	*p*	*c*	*d*	*e*
89S52 Pin No	*39*	*38*	*37*	*36*	*35*	*34*	*33*	*32*
89S52 Port No	*P0.0*	*P0.1*	*P0.2*	*P0.3*	*P0.4*	*P0.5*	*P0.6*	*P0.7*

SEVEN SEGMENT Selection Pin Configurations:

As VT8051-2.2 board comes with 4 digit seven segment unit. Displays connected to the microcontroller usually occupy a large number of valuable I/O pins, which can be a big problem especially if it is needed to display multi digit numbers. The problem is more than obvious if, for example, it is needed to display four digit numbers (a simple calculation shows that 32 output pins are needed in this case). The solution to this problem is called MULTIPLEXING. This is how an optical illusion based on the same operating principle as a film camera is made. Only one digit is active at a time, but they change their state so quickly making impression that all

digits of a number are simultaneously active. Each digit can made active using switching transistors Q1, Q2, Q3 and Q4 and these on switched on and off by selection lines which are in turn connected to 89S52 ports. Table (16) shows the details of seven segment selection lines.

Table(16) Seven Segment Selection table.

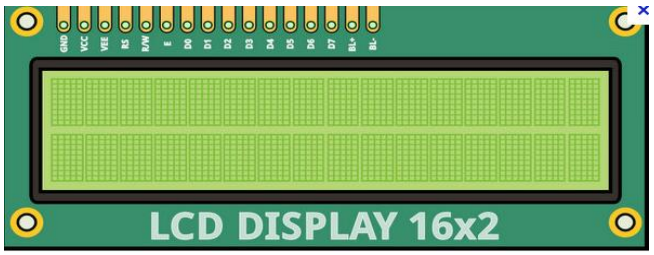
Seven Segment Selection Lines	Disp1 (DIG 1)	Disp2 (DIG 2)	Disp3 (DIG 3)	Disp4 (DIG 4)
89S52 Pin No	*14*	*15*	*16*	*17*
89S52 Port No	*P3.4*	*P3.5*	*P3.6*	*P3.7*

Display Encoding:

Digits to display	**Display Segments**								**Hex code**
	g	*f*	*a*	*b*	*p*	*c*	*d*	*e*	
0	1	0	0	0	1	0	0	0	88
1	1	1	1	0	1	0	1	1	EB
2	0	1	0	0	1	1	0	0	4C
3	0	1	0	0	1	0	0	1	49
4	0	0	1	0	1	0	1	1	2B
5	0	0	0	1	1	0	0	1	19
6	0	0	0	1	1	1	1	0	18
7	1	1	0	0	1	0	1	1	CB
8	0	0	0	0	1	0	0	0	08
9	0	0	0	0	1	0	0	1	09
A	0	0	0	0	1	0	1	0	0A
B	0	0	1	1	1	0	0	0	38
C	1	0	0	1	1	0	0	0	98
D	0	1	1	0	1	0	0	0	68
E	0	0	0	1	1	1	0	0	1C
F	0	0	0	1	1	1	1	0	1E

16 X 2 LCD Display

An LCD display is specifically manufactured to be used with microcontrollers, which means that it cannot be activated by standard IC circuits. It is used for displaying different mbessagesC on a miniature liquid crystal display. It displays all the letters of alphabet, Greek letters, punctuation marks, mathematical symbols etc. In addition, it is possible to display symbols made up by the user. Other useful features include automatic message shift (left and right), cursor appearance, LED backlight etc.



LCD Pin Configuration

There are pins along one side of a small printed board. These are used for connecting to the microcontroller. There are in total of 14 pins marked with numbers (16 if it has backlight). Their function is described in the table (17) below:

Table(17) LCD Pin connection table.

LCD Pin No	LCD Pin Functions	89S52 Pin No	89S52 Port No
1	GND	-	-
2	VCC	-	-
3	CONTRAST	-	-
4	RS	13	P3.3
5	R/W	GND	GND
6	EN	12	P3.2
7	DAT1	39	P0.0
8	DAT2	38	P0.1
9	DAT3	37	P0.2
10	DAT4	36	P0.3
11	DAT5	35	P0.4
12	DAT6	34	P0.5
13	DAT7	33	P0.6
14	DAT8	32	P0.7
15	VCC	-	-
16	GND	-	-

Create Application using KEIL

This chapter describes the Build Mode of µVision and explains the user interface; outlines the options for generating and maintaining projects, including output file options, and highlights configuration options for optimum code quality.

The required steps for creating application programs are listed below:

1. Create Project File explains how to create a µVision4 project file.
2. Select Device describes how tool settings, peripherals, and dialogs are customized.
3. Set Options for Target explains how to specify relevant parameters for your target.
4. Create Source File describes how to add new files to the project.
5. Configure a Target

Configure Micro Controller Target describes how to change the startup-code and how to use the library retarget file for Micro Controller devices.

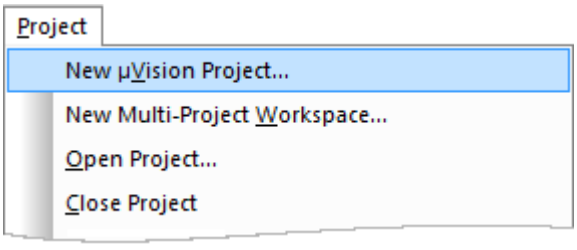
Configure Cortex-M Target describes how to change the startup-code and how to use the library retarget file for Cortex-M devices. In addition, this is a brief explanation of CMSIS.

6. Create File Group explains how to structure the code files for a better overview.
7. Specify Memory Layout describes how to change file and group attributes.
8. Build Project describes the build-options.

Create Project File

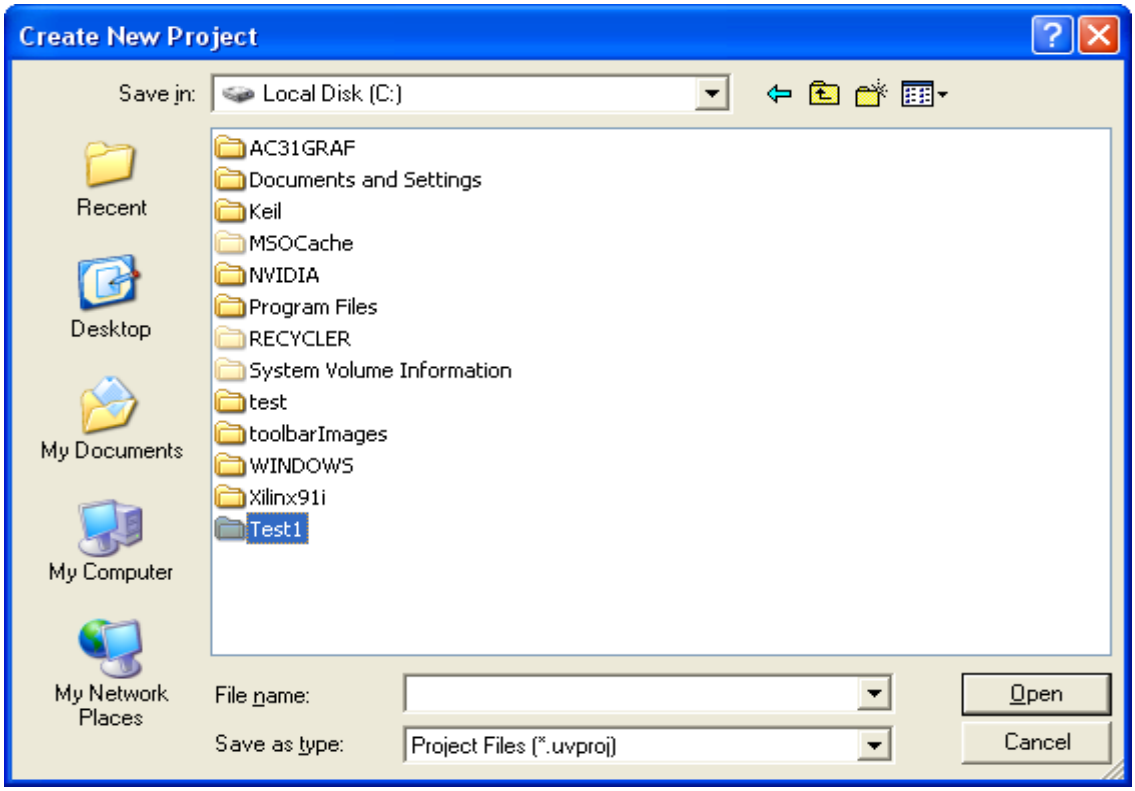
µVision4 maintains the files that belong to a project in one project file. It takes only a few steps to create a new project file with µVision4:

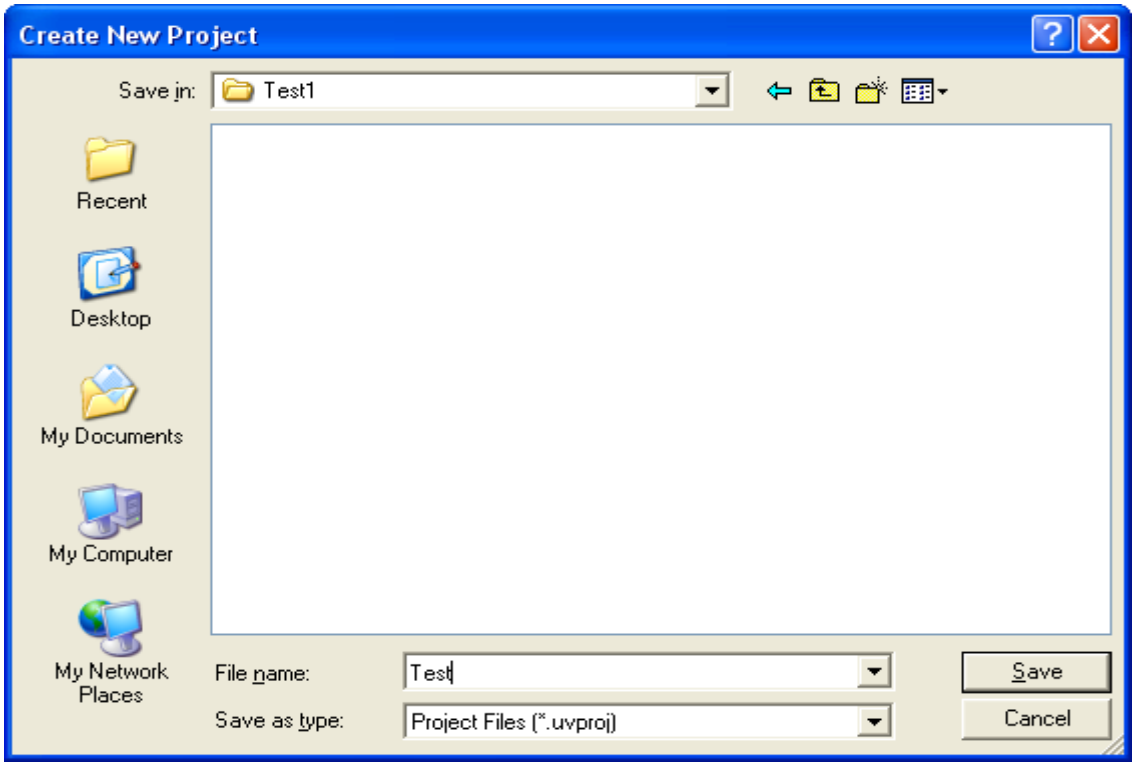
1. Select **Project - New Project** from the µVision4 menu.
   This opens a standard Windows dialog, which prompts you for the new project file name.



2. Create a new folder **Test1**.
3. Switch to the new folder and type the project name **Test**. µVision4 automatically adds the extension **.uvproj**.
4. Click **Save**.

Refer the below images.



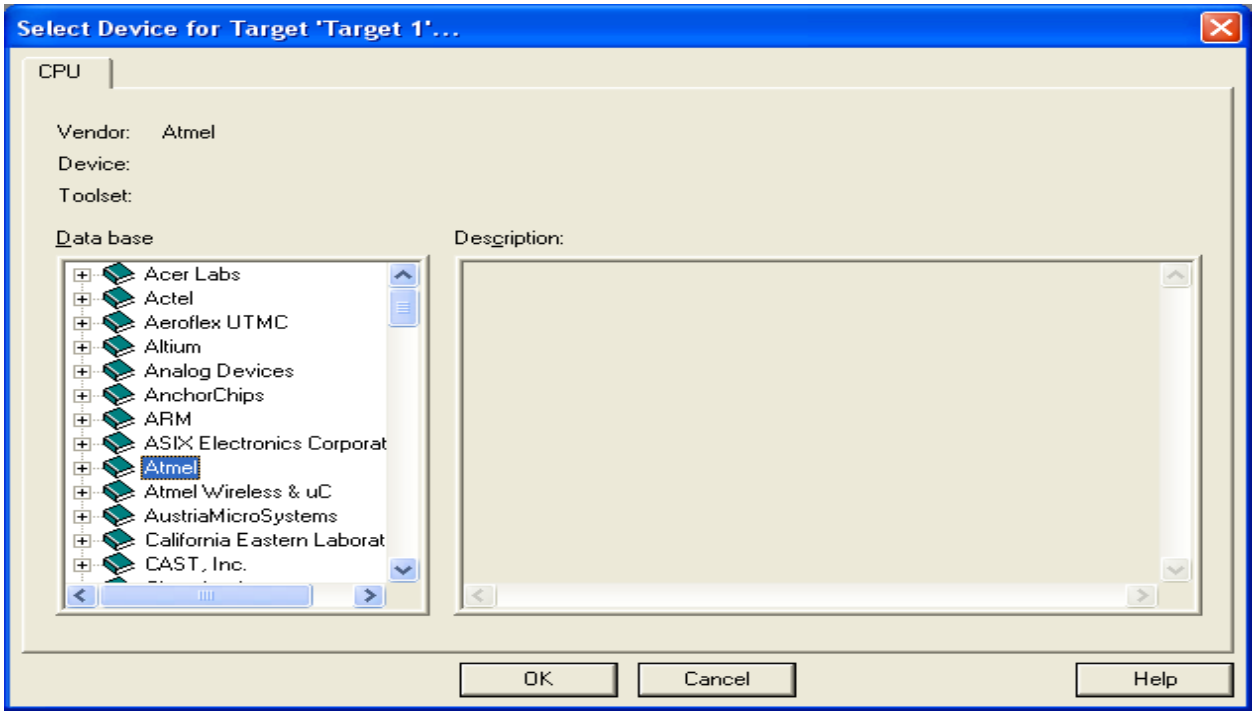


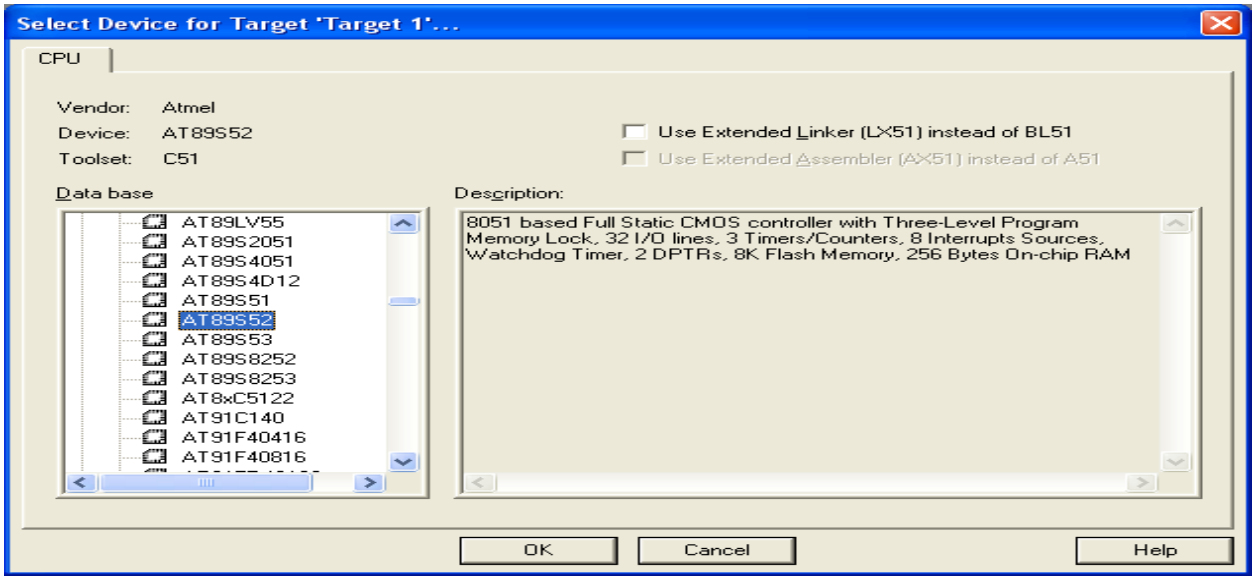
Note: It is good practice to use a separate folder for each project.

Select Device

When you create a new project, µVision4 asks you to select a microcontroller. This step customizes the µVision4 environment with Pre-Configured options and sets the tool options, Peripherals, and dialogs for that particular device. The Select Device for Target dialog shows the µVision Device Database.

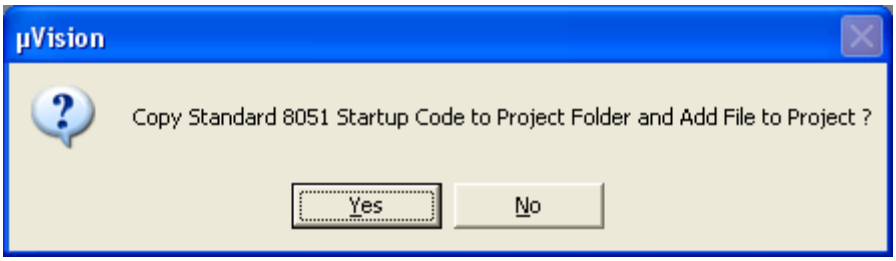
1. Select the microcontroller you use. For this example, choose the Atmel→AT89S52.



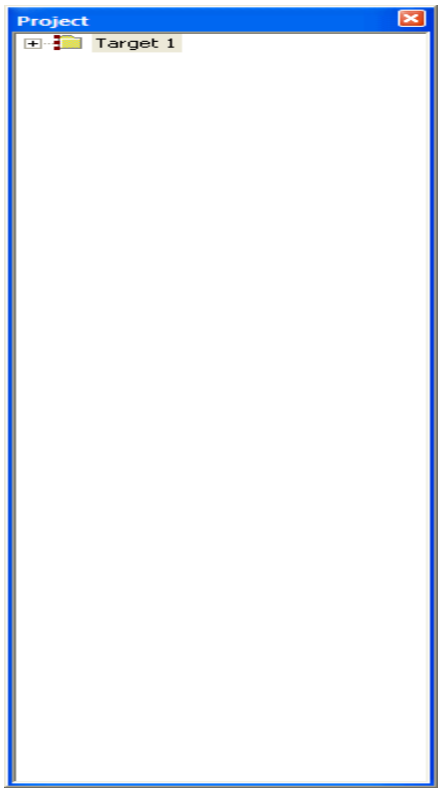


2. Click **OK.**

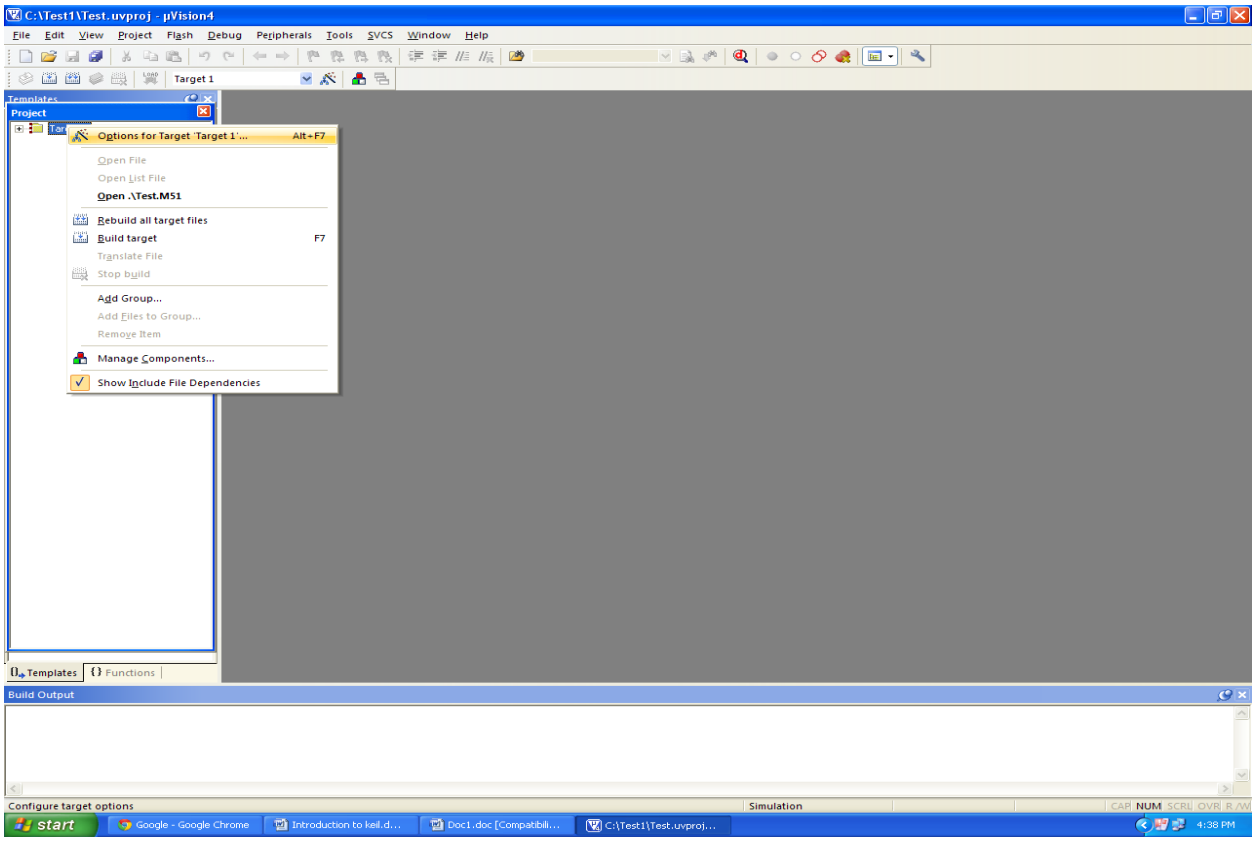
3. Click **YES** to add the Startup-Code, Which is provided for most devices. The startup-code Provides Configuration settings for the selected device.



Note: On some devices, additional parameters must be entered manually. Read carefully the information provided under **Description** of the **Select Device** dialog. It might have additional instructions for the device configuration.

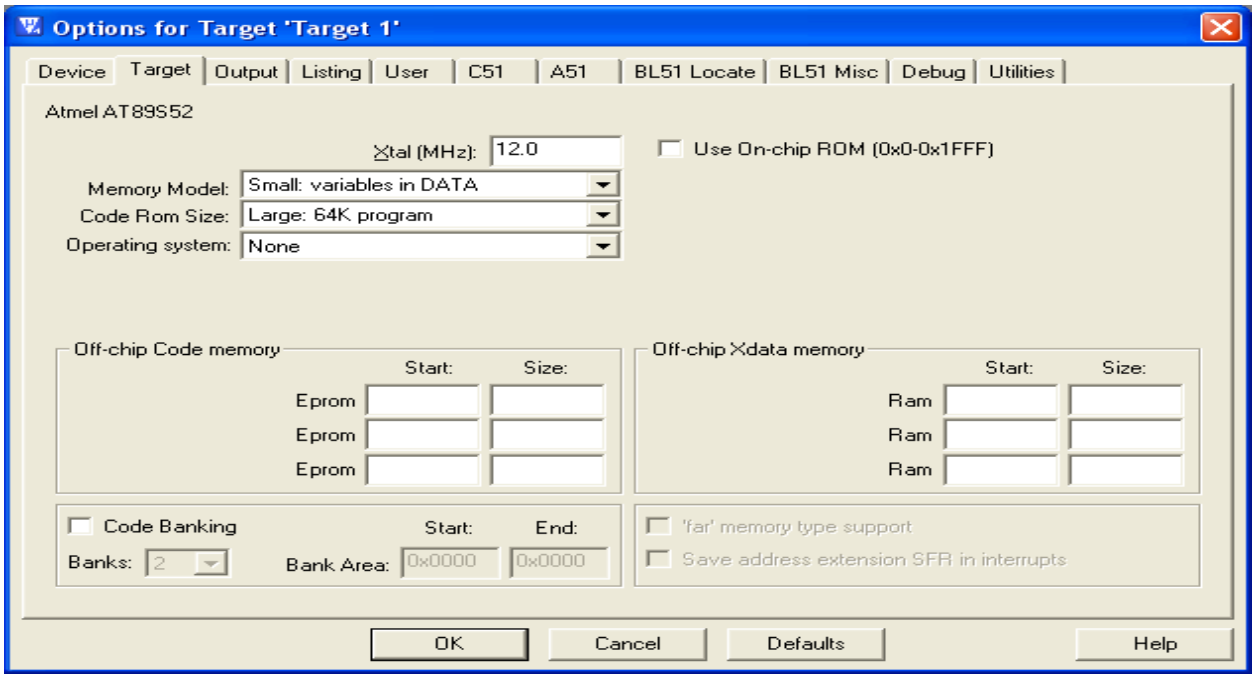


Right Click on the Target Folder Select the “Options for Target ‘Target1’… ”

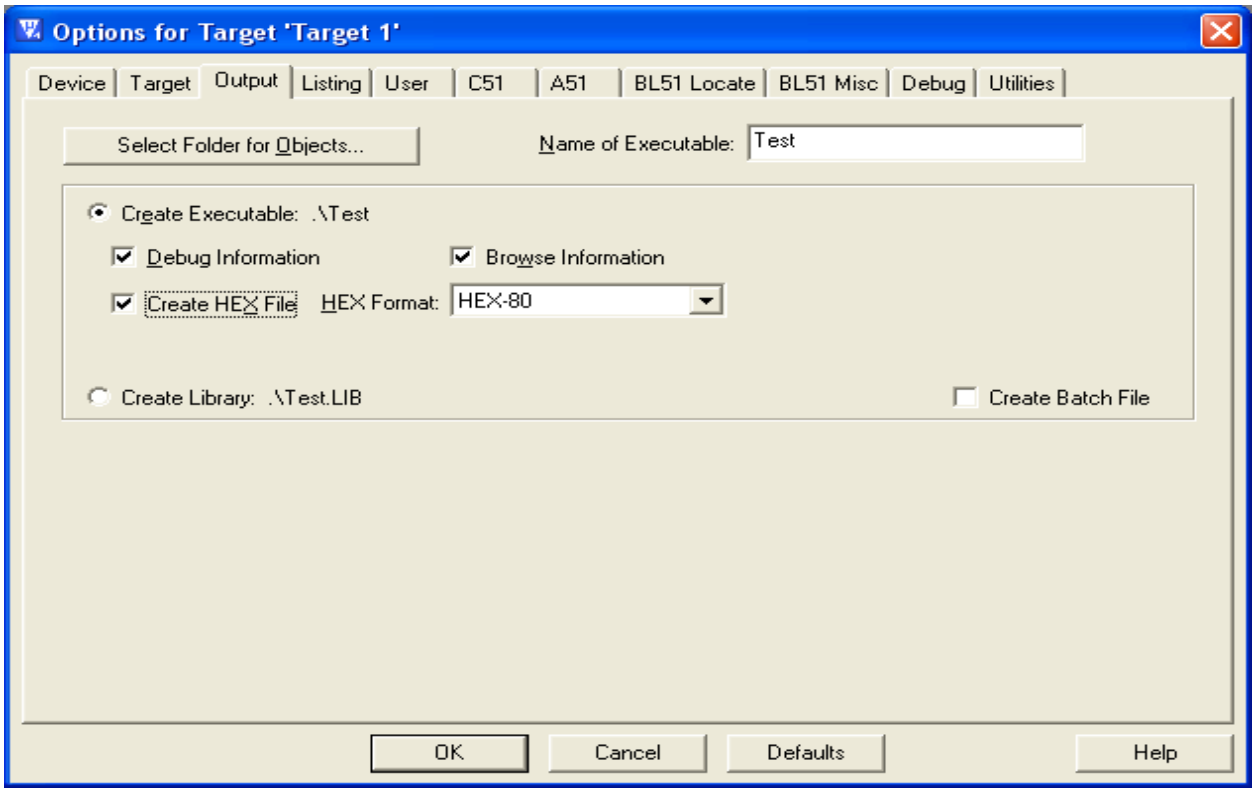


Set Options for Target

Set target parameters in the dialog **Options for Target - Target**.



Set the Target→Xtal frequency to 12.0MHz, Click on Output Tab selecting the Create HEXFile ☑ Check Box Options.

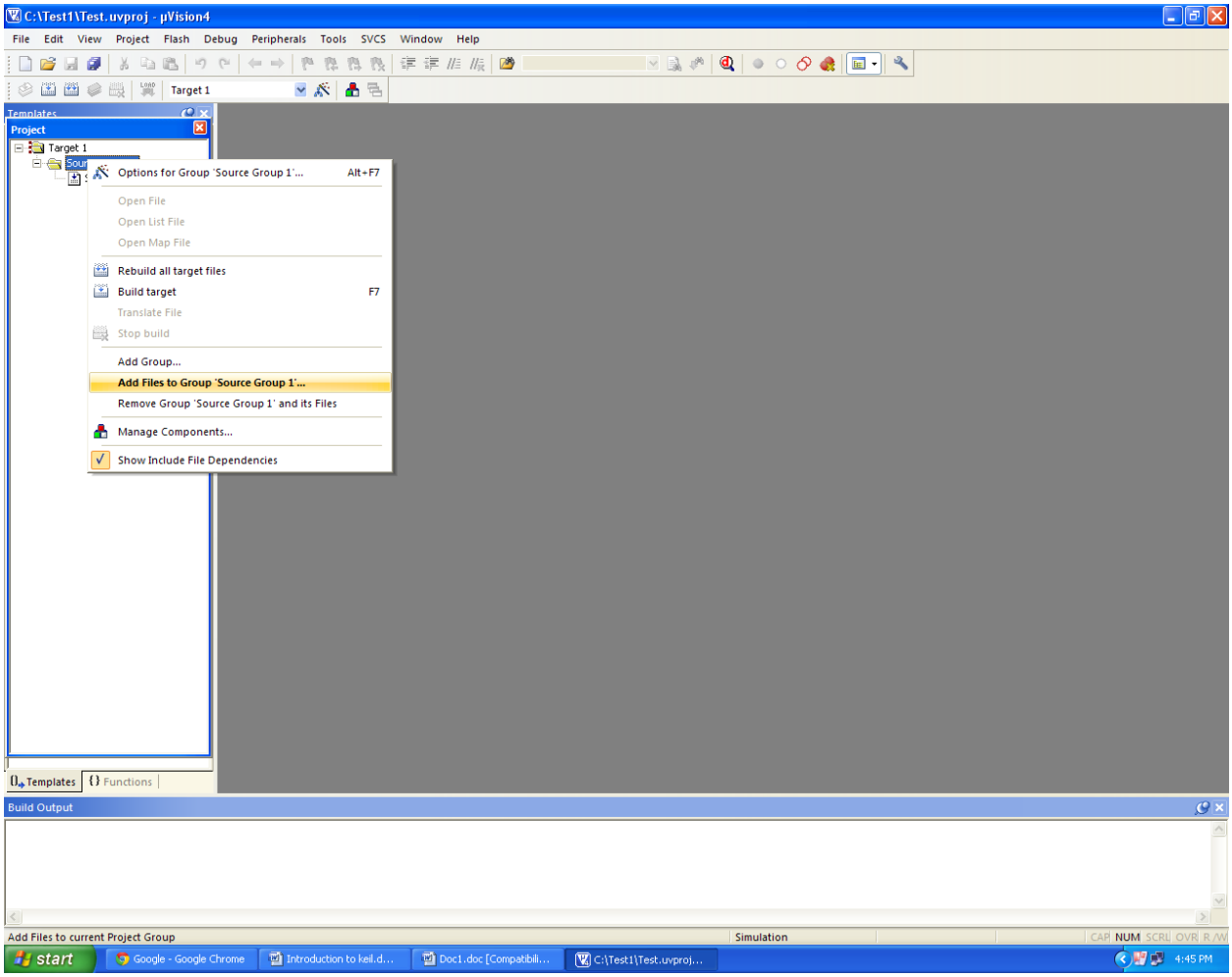


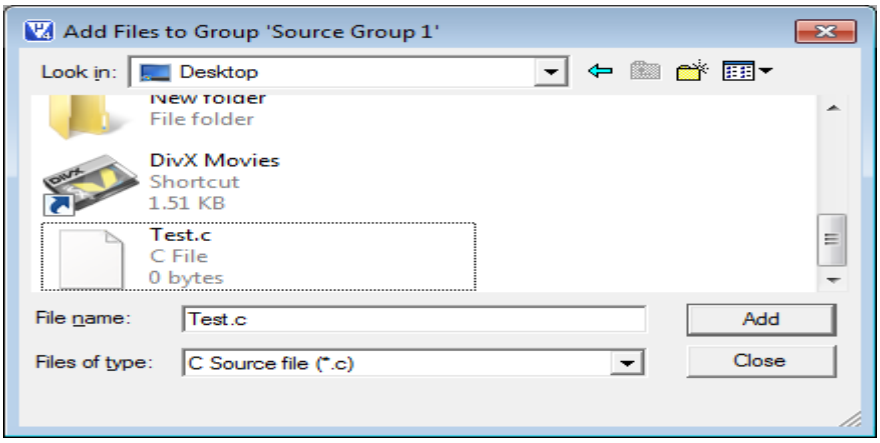
The following options can be set:

Item	Description
Xtal (MHz)	Specifies the XTAL frequency of the device; XTAL is used to configure the debugger and Flash programmer. XTAL reflects the main oscillator connected externally to the device.
Operating system	Choose the operating system. For example, the **RTX Kernel** adds the correct RTX library and enables kernel aware debugging.
Code Generation	
Use Cross-Module Optimization	Enables the linker feedback file, which allows further code optimizations.
Use MicroLIB	Changes the C run-time library-set to generate smallest application code. MicroLIB is not fully ANSI compatible, but sufficient for most small embedded applications.
Use Link-Time Code Generation	Instructs the compiler to create objects in an intermediate format to perform link-time code optimization.
Big Endian	Defines the method used to access multi-byte data.
Read/Only Memory Areas	
ROMx	Specify the Read Only Memory areas, typically located in ROM space.
IROMx	Internal Read Only Memory, typically configured by the startup-code.
Read/Write Memory Areas	
RAMx	Specify the ZI and RW areas, typically located in RAM space.
IRAMx	Specify the ZI and RW areas, typically configured through the startup-code.
default	The check box, in front of each entry, enables the area globally for the application. Individual modules may be assigned in the **Properties** dialog to specific areas.

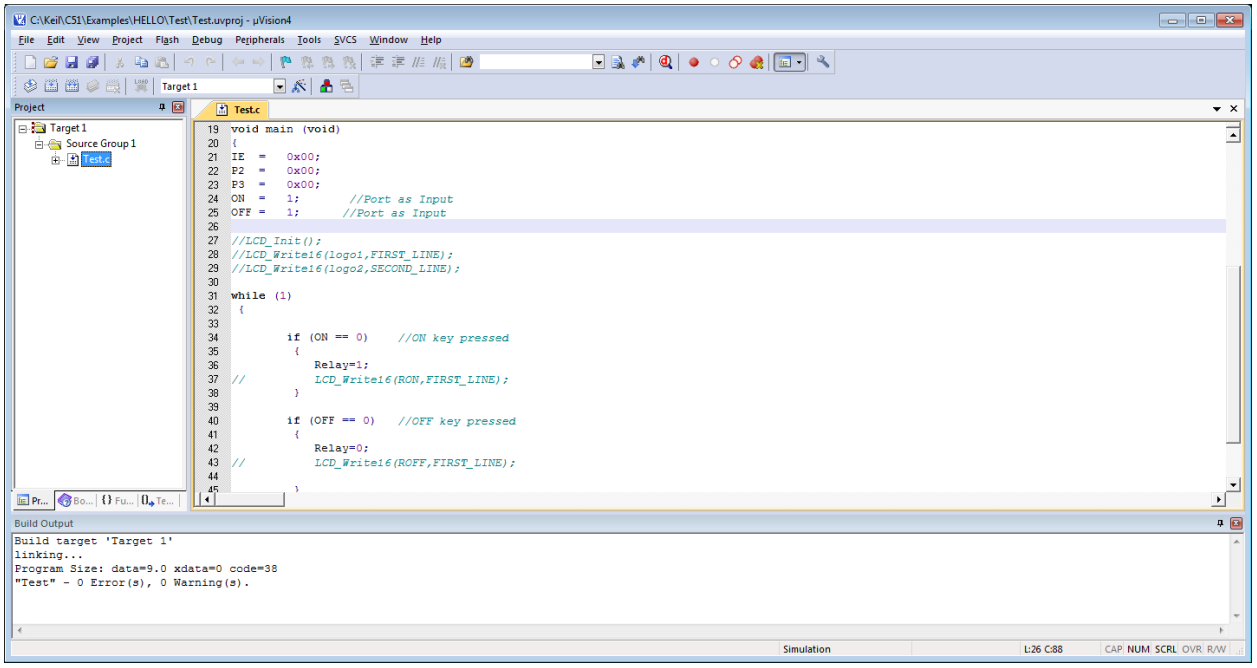
off-chip/on-chip	Specifies whether this is an internal or external ROM or RAM.
Start	Specifies the starting address of the area in HEX format.
Size	Specifies how large the area is. Also in HEX format.
NoInit	Specifies areas that should be excluded from zero initialization.

For adding files Right click on the project name in the Project Window and click on “Add Files to Group ‘Source Group1’





Coding can be done in a new window. Click Ctrl+N for new window.

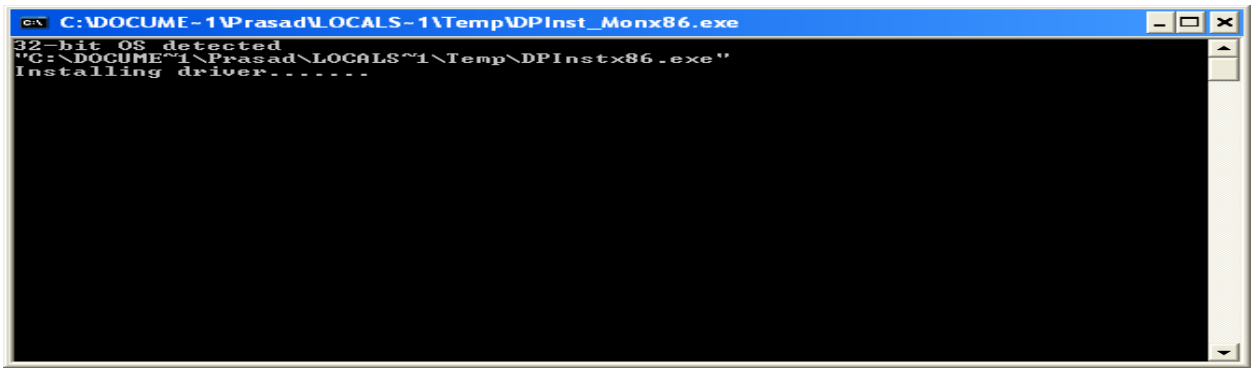


Build the Project and check for errors. Once the code is error free 0 Errors, 0 Warning is displayed in the Build Output Window.

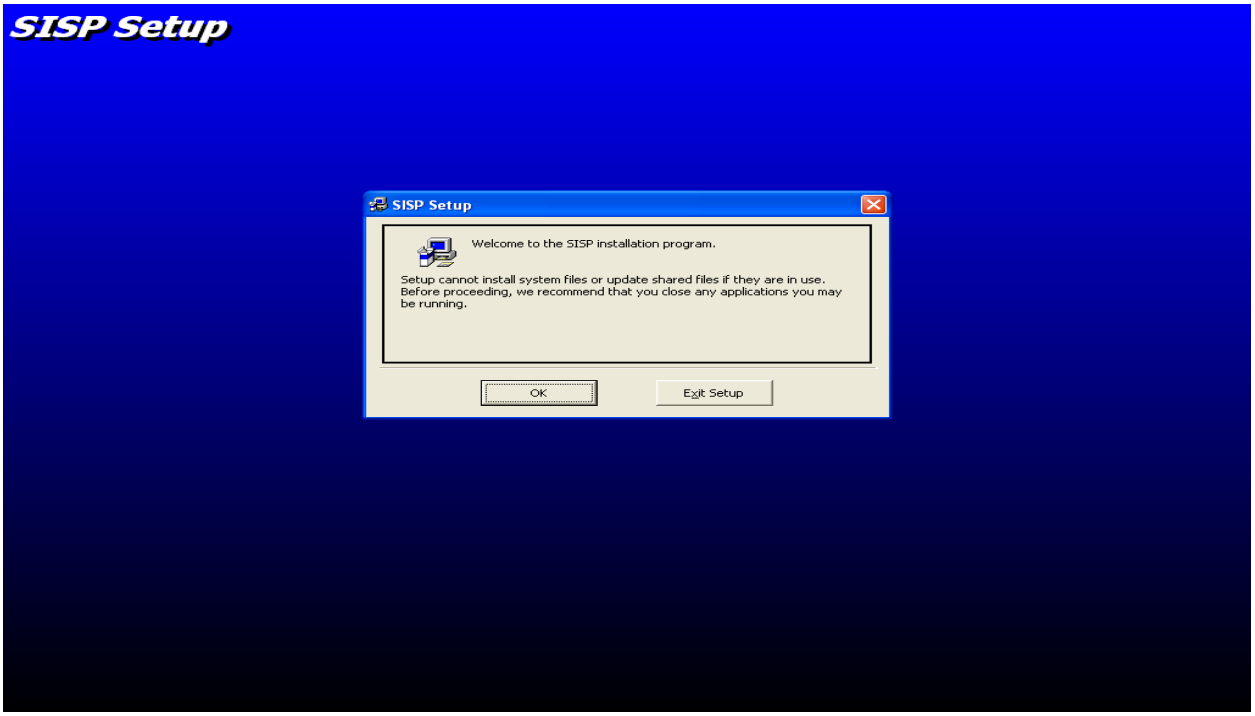
Procedure for Installing SISP Flash Programmer Software:

1. Open the SISP Setup Folder and click on the icon shown

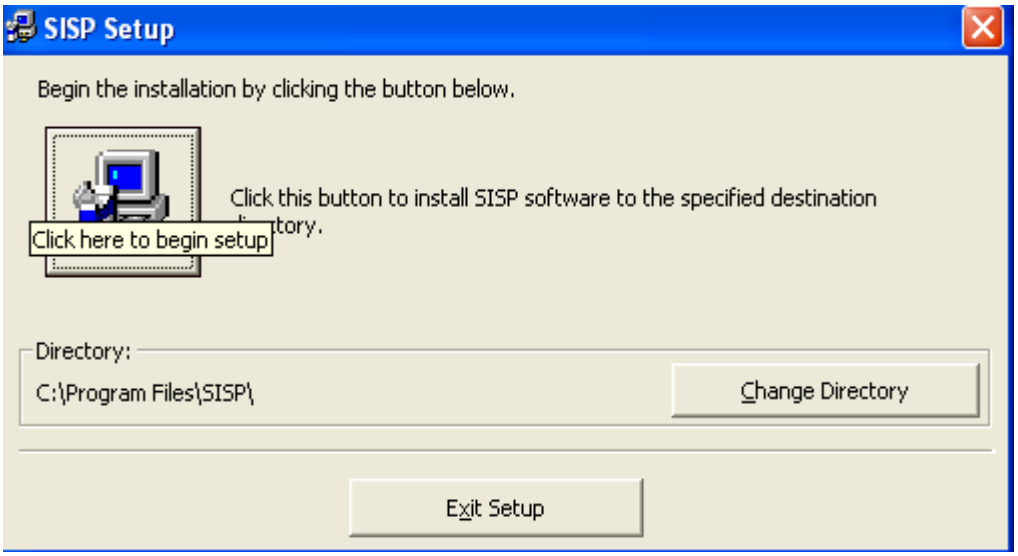
2. Click on the setup file to install the driver. Wait for the installation to complete.



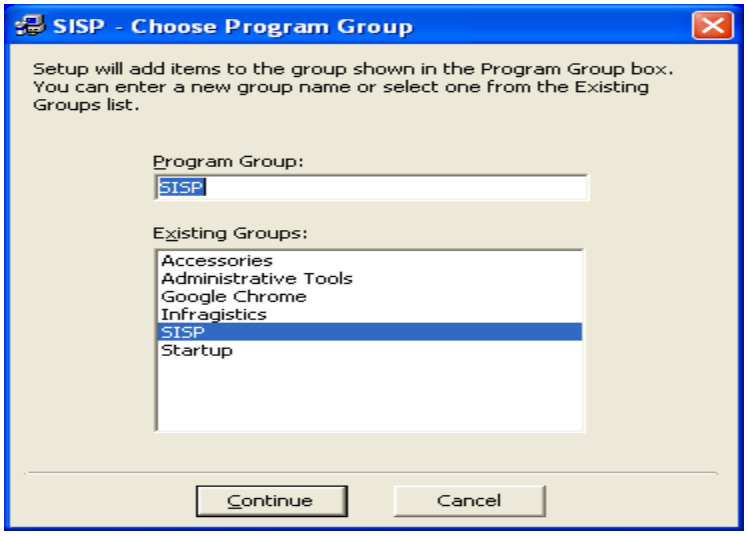
3. Click on the Setup Bootstrap.



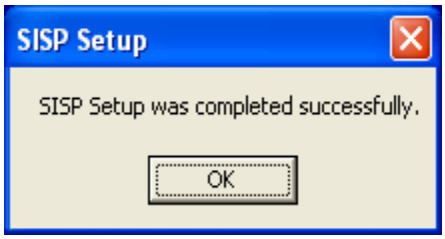
4. Click on OK button for continuing installation.



5. Click on the Desktop image as shown in the above figure.



6. Select SISP Group Name, this completes the Setup Process.

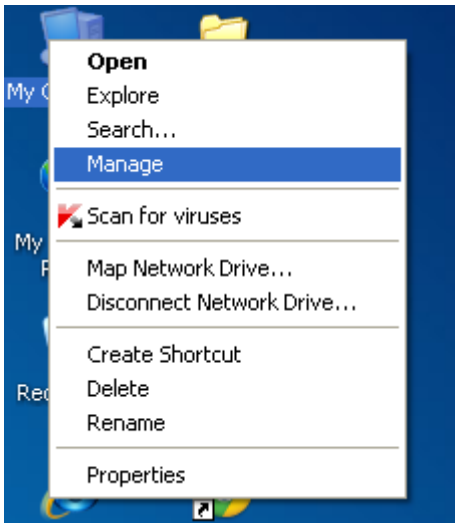


SISP Icon will be shown on the desktop screen, If Icon is not created than go for Programs→SISP, right click on SISP and select Create Shortcut option, This creates an additional name with the existing icon, drag and drop to the desktop,

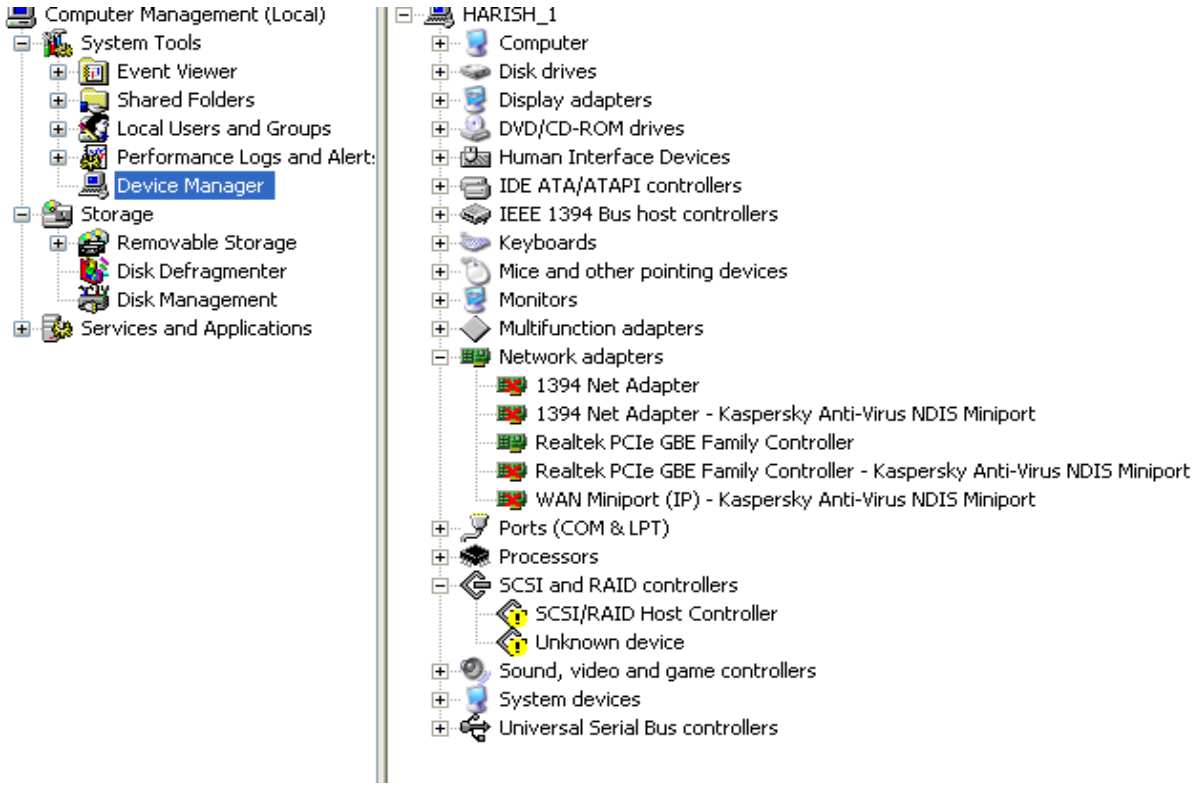
Shortcut image is seen on the desktop screen,

Steps for downloading Hex code to Micro Controller Kit.

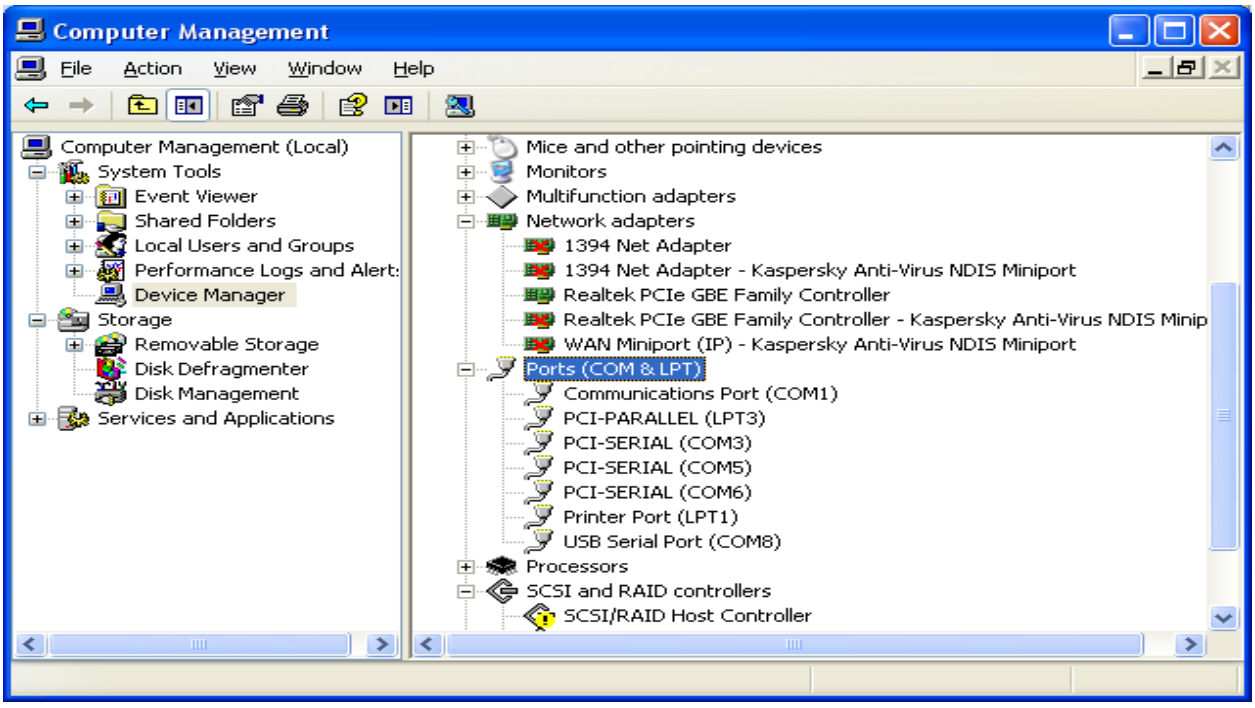
1. Insert one end of the USB cable to the USB Port and Other end to kit.
2. Right Click on My Computer option from the desktop and select Manage Option as shown in the below figure.



3. Click on Manage button and select the Device Manager Option as shown below

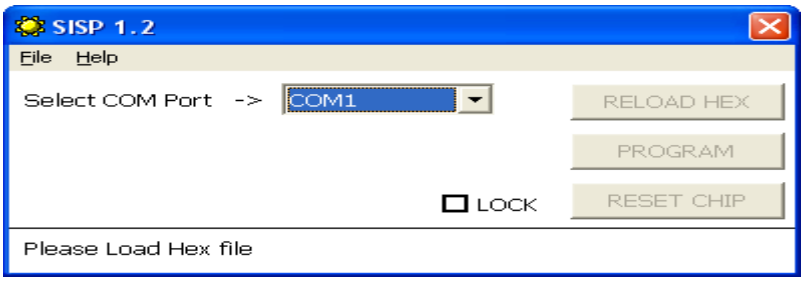


4. Expand Ports option on the right hand side to see the Serial port name of the USB Connected. For Example : **USB Serial Port(COM8)**



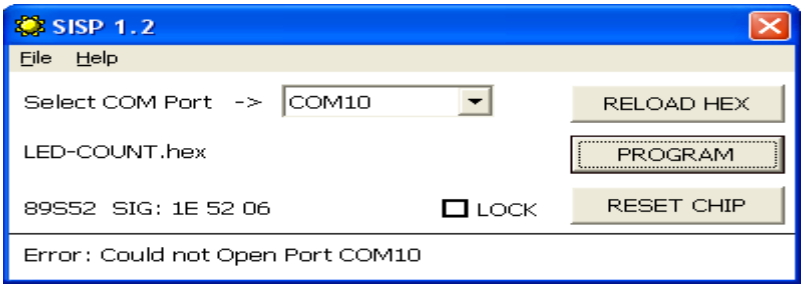
5. Minimize the window once the Port Name is known and select SISP option from desktop

6. After selecting the SISP Icon Window as is opened



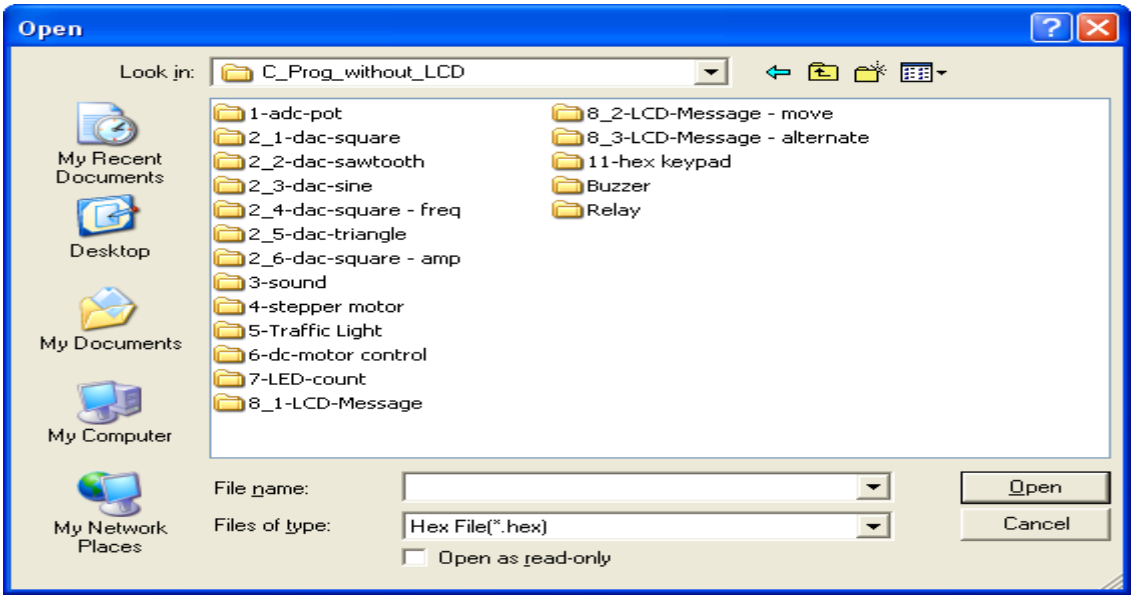
7. Select the COM Port name which is noticed(COM8) earlier

8. Selection of wrong Port Name will show an Error as shown

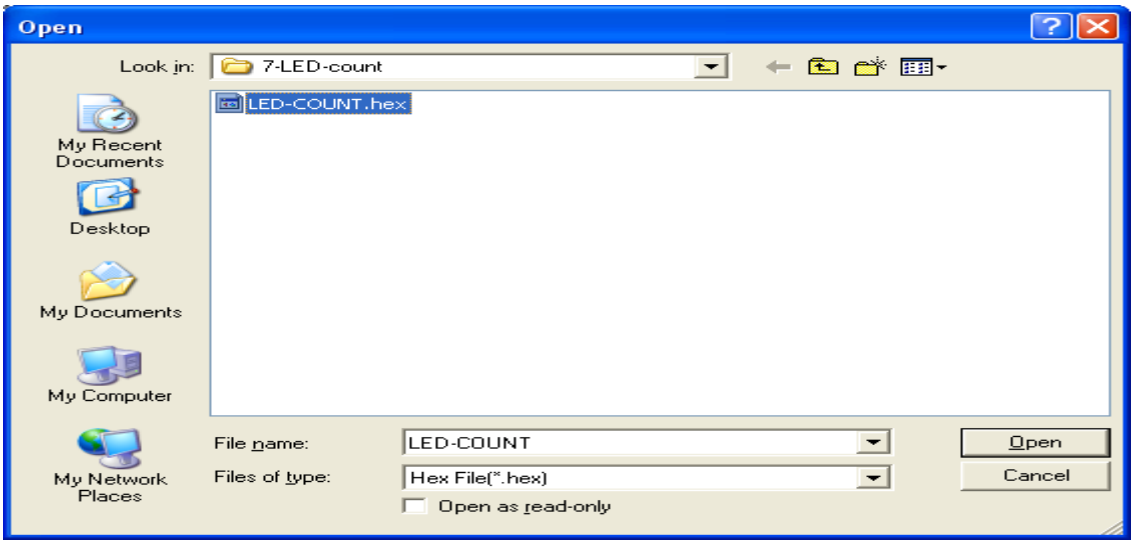


9. Select File Option→ Load Hex File

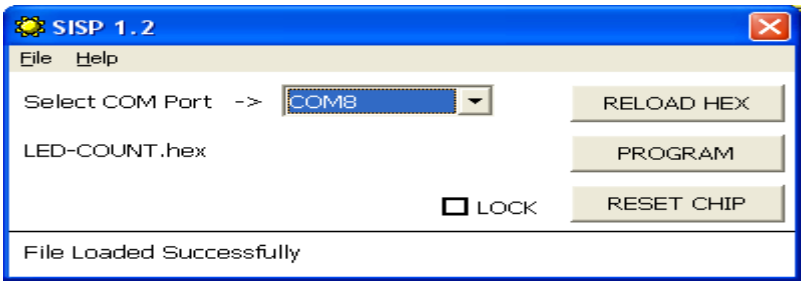
10. On click of Load Hex File will open a window for selecting the Code from the Computer



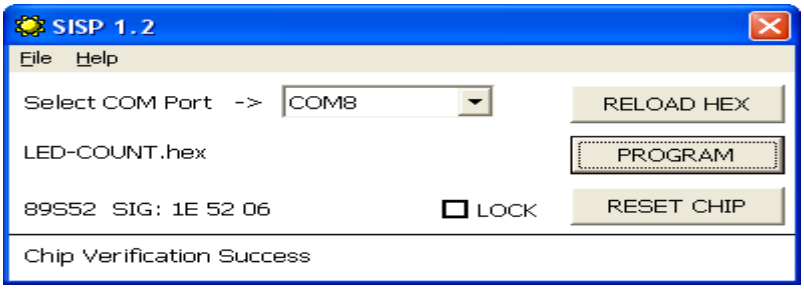
For Ex let us Load 7 Segment LED Code, Click on the respective file and Open



11. Once the Hex code is selected, File Loaded window is shown as below.



12. Click on the Program button for loading.



Chip Verification Success is shown once the code is loaded properly to the Chip. Reset Chip Button is used for resetting the IC & Reload HEX button is used to reload the same program which is selected earlier.

Check the Kit for desired Outputs by providing Valid Inputs.

Example programs

```
/* Program to read analog value from POT using ADC Crystal freq 11.0592Mz Processor: 89S52
Here we need to vary the ADC-POT(R4) and observe the LED output and reading on LCD.For each
0.02V change in analog voltage one bit digital valve will be generated*/
#include <reg52.h>
#include <intrins.h>
#include "lcd.h"

sbit ASTART = P3^0;                    // ADC Start Conversion Output
sbit AEOC  =  P3^1;                    // ADC End of Conversion Input
#define ADATA  P2                      // ADC Data Input

unsigned char code logo[] = "A TO D DEMO PROG";
unsigned char code adcv[] = "VOLTAGE =      v";

char code msb[] =
{00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,
00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,01,01,01,01,01,01,01,01,01,01,01,01,01,01,01,01,0
1,01,01,01,01,01,01,01,01,01,01,01,01,01,01,01,01,01,01,01,01,01,01,01,01,01,01,01,01,01,01,01,01,01,01,01,01
,02,02,02,02,02,02,02,02,02,02,02,02,02,02,02,02,02,02,02,02,02,02,02,02,02,02,02,02,02,02,02,02,02,02,02,
02,02,02,02,02,02,02,02,02,02,02,02,02,02,02,02,02,02,03,03,03,03,03,03,03,03,03,03,03,03,03,03,03,03,03,03,0
3,03,03,03,03,03,03,03,03,03,03,03,03,03,03,03,03,03,03,03,03,03,03,03,03,03,03,03,03,03,03,03,03,03,03,03,03
,04,04,04,04,04,04,04,04,04,04,04,04,04,04,04,04,04,04,04,04,04,04,04,04,04,04,04,04,04,04,04,04,04,04,04,
04,04,04,04,04,04,04,04,04,04,04,04,04,04,04,04,04,04,04,04,05,05,05,05,05,05,05,05,05,05,05,05,05,05,05,05,0
5,05,05,05,05,05,05,05,05,05,05,05,05,05,05,05,05,05,05,05,05,05,05,05,05,05,05,05,05,05,05,05,05,05,05,05,05
};

char code lsb[] =
{0x00,0x02,0x04,0x06,0x08,0x10,0x12,0x14,0x16,0x18,0x20,0x22,0x24,0x25,0x27,0x29,0x31,0x33,0x3
5,0x37,0x39,0x41,0x43,0x45,0x47,0x49,0x51,0x53,0x55,0x57,0x59,0x61,0x63,0x65,0x67,0x69,0x71,0x7
3,0x75,0x76,0x78,0x80,0x82,0x84,0x86,0x88,0x90,0x92,0x94,0x96,0x98,0x00,0x02,0x04,0x06,0x08,0x1
0,0x12,0x14,0x16,0x18,0x20,0x22,0x24,0x25,0x27,0x29,0x31,0x33,0x35,0x37,0x39,0x41,0x43,0x45,0x4
7,0x49,0x51,0x53,0x55,0x57,0x59,0x61,0x63,0x65,0x67,0x69,0x71,0x73,0x75,0x76,0x78,0x80,0x82,0x8
4,0x86,0x88,0x90,0x92,0x94,0x96,0x98,0x00,0x02,0x04,0x06,0x08,0x10,0x12,0x14,0x16,0x18,0x20,0x2
2,0x24,0x25,0x27,0x29,0x31,0x33,0x35,0x37,0x39,0x41,0x43,0x45,0x47,0x49,0x51,0x53,0x55,0x57,0x5
9,0x61,0x63,0x65,0x67,0x69,0x71,0x73,0x75,0x76,0x78,0x80,0x82,0x84,0x86,0x88,0x90,0x92,0x94,0x9
6,0x98,0x00,0x02,0x04,0x06,0x08,0x10,0x12,0x14,0x16,0x18,0x20,0x22,0x24,0x25,0x27,0x29,0x31,0x3
3,0x35,0x37,0x39,0x41,0x43,0x45,0x47,0x49,0x51,0x53,0x55,0x57,0x59,0x61,0x63,0x65,0x67,0x69,0x7
1,0x73,0x75,0x76,0x78,0x80,0x82,0x84,0x86,0x88,0x90,0x92,0x94,0x96,0x98,0x00,0x02,0x04,0x06,0x0
8,0x10,0x12,0x14,0x16,0x18,0x20,0x22,0x24,0x25,0x27,0x29,0x31,0x33,0x35,0x37,0x39,0x41,0x43,0x4
5,0x47,0x49,0x51,0x53,0x55,0x57,0x59,0x61,0x63,0x65,0x67,0x69,0x71,0x73,0x75,0x76,0x78,0x80,0x8
2,0x84,0x86,0x88,0x90,0x92,0x94,0x96,0x98,0x00};

/*-----------------------------------------------
Read ADC C function
-----------------------------------------------*/
unsigned char rdadc()
{
ASTART = 0;
_nop_();
_nop_();
```

```
_nop_();
   ASTART = 1;
_nop_();
_nop_();
_nop_();

while(AEOC == 1);
return(ADATA);
}
/*-----------------------------------------------
MAIN C function
-----------------------------------------------*/
void main (void)
{
        Unsigned char aval, hi, lo;
        Unsigned char txt [2];
        LCD_Init ();
        LCD_Write16 (logo, FIRST_LINE);
        LCD_Write16 (adcv, SECOND_LINE);
        IE=0x00;
         P3=0x00;
        ADATA=0xFF;
        ASTART=0;
        AEOC=1;

While (1)
        {
                 aval=rdadc ();
                 hi = msb[aval];
                 lo = lsb[aval];
                WriteLCD (0xCA, CONTROL_REG);
                 bytetoa(hi,txt);
                 WriteLCD (txt[0], DATA_REG);
                WriteLCD ('.', DATA_REG);
                bytetoa (lo,txt);
                WriteLCD (txt [1], DATA_REG);
                WriteLCD (txt [0], DATA_REG);
                 DelayS (1);
        }
}
```

```
/* Program to generate saw tooth wave using DAC Crystal freq 11.0592Mz Processor: 89S52.Down load
hex file and connect CRO probe between TP4 and TP3.You will get saw tooth wave*/

#include <reg52.h>
#include <intrins.h>

#define     DAC P2                          // DAC Data Output
sbit        DWR = P3^0;                     //DAC Write Pin

void delay(unsigned char tim)
{
        unsigned char i,j;
             for (j=0;j<tim;j++)
             for (i=0; i<10;i++);
}

/*-------------------------------------------------
MAIN C function
-------------------------------------------------*/
void main (void)
{
unsigned char dval;
          IE=0x00;
          P3=0X00;
          DAC = 0x00;
          DWR = 0;
          _nop_();
          DWR = 1;
          dval = 0xFF;
While (1)
      {
          DAC = dval;
          DWR = 0;
            _nop_();
            DWR = 1;
          delay(1); //varying this value will wary the freq of the waveform
          dval--;
       }
}
```

```
/*Program to generate SQUARE wave using DAC Crystal freq 11.0592Mz Processor: 89S52. Down load
hex file and connect CRO probe between TP4 and TP3.You will get square wave */

#include <reg52.h>
#include <intrins.h>

#define DAC    P2                          // DAC Data Output
sbit    DWR =                               P3^0; //DAC Write Pin

void delay(unsigned char tim)
{
unsigned char i,j;

                                           for (j=0;j<tim;j++)
                                           for (i=0; i<10;i++);
}

/*-------------------------------------------------
MAIN C function
-------------------------------------------------*/
void main (void)
{


IE=0x00;
DAC = 0x00;
P3=0X00;

DWR=1;

while (1)
 {
                DAC = 0x00;
                DWR = 0;
                  _nop_();
                 DWR = 1;
                delay(50);                  //varying this value will wary the freq of the waveform

                DAC = 0xFF;
                DWR = 0;
                  _nop_();
                 DWR = 1;
                delay(50);                  //varying this value will wary the freq of the waveform

 }
}
```

```
/*Program to generate SQUARE wave using DAC with Variable frequency Crystal freq 11.0592Mz
Processor: 89S52. Down load hex file and connect CRO probe between TP4 and TP3.You will get
square wave. Now press S1for increase in frequency and press S4 for decrease in frequency */
#include <reg52.h>
#include <intrins.h>
#define DAC  P2                          // DAC Data Output
sbit    DWR = P3^0;                       //DAC Write Pin
sbit    INC= P1^0;                       //Increase Button
sbit    DEC= P1^4;                       //Decrease Button
void delay(unsigned char tim)
{
unsigned char i,j;
   for (j=0;j<tim;j++)
  for (i=0; i<10;i++);
}
/*----------------------MAIN C function-----------------------------------------------*/
void main (void)
{
unsigned char del;
           P3=0X00;
           DAC = 0x00;
           DWR = 0;
            _nop_();
           DWR = 1;
           INC=1;                        //Port as Input
           DEC=1;                        //Port as Input
while (1)
      {
        if (INC == 0)                    //INC key pressed
          {
             if (del == 0xFF) goto wave;
            del++;
           goto wave;
          }
       if (DEC == 0)                     //INC key pressed
          {
            if (del == 0x00) goto wave;
           del--;
          }
wave:   DAC = 0x00;
           DWR = 0;
             _nop_ ();
              DWR = 1;
           delay (del);                  //varying this value will wary the freq of the waveform
           DAC = 0xFF;
           DWR = 0;
            _nop_();
             DWR = 1;
           delay (del);                  //varying this value will wary the freq of the waveform
          }
 }
```

```
/*Program to generate Sine wave using DAC Crystal freq 11.0592Mz Processor: 89S52 Down load hex
file and connect CRO probe between TP4 and TP3.You will get sine wave */

#include <reg52.h>
#include <intrins.h>

#define DAC  P2                          // DAC Data Output
sbit    DWR = P3^0;                       //DAC Write Pin


unsigned char code tab[] = { 128,131,134,137,140,144,147,150,153,156,159,162,165,168,171,174,177,
179,182,185,188,191,193,196,199,201,204,206,209,211,213,216,218,220,222,224,226,228,230,232,
234,235,237,239,240,241,243,244,245,246,248,249,250,250,251,252,253,253,254,254,254,255,255,
255,255,255,255,255,254,254,254,253,253,252,251,250,250,249,248,246,245,244,243,241,240,239,
237,235,234,232,230,228,226,224,222,220,218,216,213,211,209,206,204,201,199,196,193,191,188,
185,182,179,177,174,171,168,165,162,159,156,153,150,147,144,140,137,134,131,128,125,122,119,
116,112,109,106,103,100,97,94,91,88,85,82,79,77,74,71,68,65,63,60,57,55,52,50,47,45,43,40,38,36,
34,32,30,28,26,24,22,21,19,17,16,15,13,12,11,10,8,7,6,6,5,4,3,3,2,2,2,1,1,1,1,1,1,1,2,2,2,3,3,4,5,6,6,7,
8,10,11,12,13,15,16,17,19,21,22,24,26,28,30,32,34,36,38,40,43,45,47,50,52,55,57,60,63,65,68,71,74,
77,79,82,85,88,91,94,97,100,103,106,109,112,116,119,122,125
};
void delay(unsigned char tim)
{
unsigned char i,j;
          for (j=0;j<tim;j++)
          for (i=0; i<10;i++);
}
/*-------------------------------------------------
MAIN C function
-------------------------------------------------*/
void main (void)
{
unsigned char dval;
         P3= 0x00;
         DAC = 0x00;
         dval = 0x00;
while (1)
 {
           DAC = tab [dval];
           DWR = 0;
           _nop_();
          DWR = 1;
         delay(1); //varying this value will vary the freq of the wave
         dval++;
 }
}
```

```
/*Program to control Stepper Motor Speed and Direction Crystal freq 11.0592Mz Processor: 89S52.
Down load hex file and Press user defined keys S1 S4, S2, S5,S3 and S6 .observe the Stepper motor */

#include <reg52.h>
#include <intrins.h>

sbit START = P1^0;
sbit STOP  = P1^4;
sbit CLK   = P1^1;
sbit ACLK  = P1^5;
sbit INC   = P1^2;
sbit DEC   = P1^6;
#define  MOTOR P2
unsigned char run=1;
unsigned char dir=1;
unsigned char del=10;
/*-------------------------------------------------Stepper delay-------------------------------------------------*/
void stp_delay(unsigned char del)
{
unsigned int i,j;
        for (j = 0;j<del;j++)
       for(i = 0;i<1000;i++);
}
/*-------------------------------------------------Key functions-------------------------------------------------*/
void chk_key(void)
{
       if (START == 0)                    //Start key pressed
         {
            run = 1;
           if (dir == 1)
          goto rotate;
         }
     if (STOP == 0)                       //Stop key pressed
       {
          run = 0;
         goto rotate;
       }
  if (CLK == 0)                           //Clockwise direction key pressed
    {
       dir = 1;
      goto rotate;
    }
 if (ACLK == 0)                           //AClockwise direction key pressed
  {
     dir = 0;
    goto rotate;
     }
  if (INC == 0)                           //Increase Motor Speed
   {
     if (del == 1) goto rotate;
    del = del - 1;
```

```
    goto rotate;
   }
if (DEC == 0)                                   //Decrease Motor Speed
  {
    if (del == 100) goto rotate;
   del = del + 1;
 }
rotate:
_nop_();
}
/*-----------------------------------------------MAIN C function-----------------------------------------------*/
void main (void)
{
unsigned char stpval=1;
START = 1;                                      // all keys as inputs
STOP  = 1;
CLK   = 1;
ACLK  = 1;
INC   = 1;
DEC   = 1;
MOTOR = 0x00;
IE=0x00;
P3=0x00;
while (1)
 {
     chk_key();
     if (run == 1)
    {
    if (dir == 0)
      {
        if (stpval == 8)  //rotate step value bitwise left
       stpval = 1;
       else
       stpval <<= 1;
      }
else
   {
    if (stpval == 1)  //rotate step value bitwise right
   stpval = 8;
else
  stpval >>= 1;
    }
     MOTOR = stpval;
     stp_delay(del);
   }
 }
}
```

```
/*Program to control DC Motor Speed and Direction Crystal freq 11.0592Mz Processor: 89S52 Down
load hex file and Press user defined keys S1 S2, S3, S4,S5,S6,S7 and S8 .observe the DC motor */

#include <reg52.h>
#include <intrins.h>

sbit  DIR1  = P2^0;
sbit  DIR2  = P2^1;
sbit  PWM  = P3^0;

sbit     AKMIN = P1^0;                    //Anti Clockwise Minimum speed
sbit     AKAVG = P1^1;                    //Anti Clockwise Average speed
sbit     AKMED = P1^2;                    //Anti Clockwise Medium speed
sbit     AKMAX = P1^3;                    //Anti Clockwise Maximum speed
sbit    CKMIN = P1^4;                     //Clockwise Minimum speed
sbit    CKAVG = P1^5;                     //Clockwise Average speed
sbit    CKMED = P1^6;                     //Clockwise Medium speed
sbit    CKMAX = P1^7;                     //Clockwise Maximum speed

/*-------------------------------------------------
MAIN C function
-------------------------------------------------*/
void main (void)
{
unsigned char MOTOR;
unsigned char i;
IE = 0x00;
P3 = 0X00;
AKMIN = 1; // All Ports as Inputs
AKAVG = 1;
AKMED = 1;
AKMAX = 1;
CKMIN = 1;
CKAVG = 1;
CKMED = 1;
CKMAX = 1;
MOTOR=0;
DIR1=1; DIR2=1;                          //Motor stopped
PWM=1;

while (1)
 {
                if (AKMIN == 0) //Anti Clockwise Minimum speed
                 {
                        MOTOR = 10;
                        DIR1=1; DIR2=0;
                        goto quit;
                 }
                if (AKAVG == 0)//Anti Clockwise Average speed
                 {
                        MOTOR = 50;
```

```
                DIR1=1; DIR2=0;
                goto quit;
            }
            if (AKMED == 0)         //Anti Clockwise Medium speed
            {
                MOTOR = 100;
                DIR1=1; DIR2=0;
                goto quit;
            }
            if (AKMAX == 0)         //Anti Clockwise Maximum speed
            {
                MOTOR = 200;
                DIR1=1; DIR2=0;
            //  LCD_Write16(amax,SECOND_LINE);
                goto quit;
            }
            if (CKMIN == 0) //Clockwise Minimum speed
            {
                MOTOR = 10;
                DIR1=0; DIR2=1;
            //  LCD_Write16(cmin,SECOND_LINE);
                goto quit;
            }
            if (CKAVG == 0)         //Clockwise Average speed
            {
                MOTOR = 50;
                DIR1=0; DIR2=1;
            //  LCD_Write16(cavg,SECOND_LINE);
                goto quit;
            }
            if (CKMED == 0)         //Clockwise Medium speed
            {
                MOTOR = 100;
                DIR1=0; DIR2=1;
            //  LCD_Write16(cmed,SECOND_LINE);
                goto quit;
            }
            if (CKMAX == 0)         //Clockwise Maximum speed
            {
                MOTOR = 200;
                DIR1=0; DIR2=1;
                goto quit;
            }
quit:       PWM=0;
            for (i=0;i<MOTOR;i++)
              {
                _nop_(); _nop_(); _nop_();
              }
            PWM=1;
 }
}
```

```
/*Program to display Message on LCD Crystal freq 11.0592Mz Processor: 89S52. Down load hex file
and observe the LCD, Message "Vasundhara technologies" is displayed in two line. */
#include <reg52.h>
#include <intrins.h>
//LCD LINE ADDRESSES
#define FIRST_LINE 0x80
#define SECOND_LINE 0xC0
#define CONTROL_REG 0x00
#define DATA_REG    0x01
#define  LCD     P0
Sbit      LCD_EN  = P3^2;
Sbit      LCD_RS  = P3^3;
unsigned char code logo1[] = "    Vasundhara  ";
unsigned char code logo2[] = "  Technologies  ";
void delay10ms()                    //10 msec
{
        unsigned int m,n;
        for(n=0;n<4;n++)
        for(m=0;m<600;m++);
void delay50ms()
{
        unsigned int i;
        for(i = 0;i < 5;i++)
        delay10ms();
}
////// Write Data to LCD Display Device on a 8-Bit Bus///////
void WriteLCD(unsigned char ldat,unsigned char Reg)
{
unsigned char i;
                LCD = ldat;                     // write value
                LCD_RS = Reg;                   // select data register
                LCD_EN = 1;                     // give operation start signal
                for(i = 0;i < 50;i++);          // wait
                LCD_EN = 0;
                for(i = 0;i < 250;i++);         // wait
}
//////////////////////////LCD Initialization//////////////////////////
void LCD_Init()
{
                delay50ms();
                delay50ms();
                delay50ms();
                delay50ms();
                WriteLCD(0x38,CONTROL_REG);
                WriteLCD(0x38,CONTROL_REG);
                WriteLCD(0x38,CONTROL_REG);
                WriteLCD(0x10,CONTROL_REG);
                WriteLCD(0x0C,CONTROL_REG);
                WriteLCD(0x06,CONTROL_REG);
                WriteLCD(0x01,CONTROL_REG);
}
```

```
//////LCD Write for 16 Characters i.e one Line//////////////////

void LCD_Write16(unsigned char *String,unsigned char Linenum)

{
                    unsigned char i;
                    WriteLCD(Linenum,CONTROL_REG);
                    for(i = 0;i<16;i++)
                    {
                              WriteLCD(String[i],DATA_REG) ;
                    }
}

/*---------------------------------------------
MAIN C function
---------------------------------------------*/

void main (void)
{

IE=0x00;
LCD_Init();

LCD_Write16(logo1,FIRST_LINE);
LCD_Write16(logo2,SECOND_LINE);

while(1);
}
```

```
/*Program to Switch on and off a Buzzer Crystal freq 11.0592Mz, Processor: 89S52. Down load hex file
and Press S1 to switch on Buzzer and S4 to switch off the Buzzer*/

#include <reg52.h>
#include <intrins.h>

sbit    Buzzer = P3^0;                  //BUZZER CONTROL Pin
sbit   ON      = P1^0;                  //ON KEY
sbit  OFF      = P1^4;                  //OFF KEY

/*-------------------------------------------------
MAIN C function
-------------------------------------------------*/
void main (void)
{
IE = 0x00;
P2 = 0x00;
P3 = 0x00;
ON = 1;                                 //Port as Input
OFF = 1;                                //Port as Input

while (1)
 {

                if (ON == 0)       //ON key pressed
                 {
                        Buzzer=0;
                 }

                if (OFF == 0)     //OFF key pressed
                 {
                        Buzzer=1;
                 }

 }
}
```

```
/*Program to display Count on the 7-Seg Display increment or decrement through keys Crystal freq
11.0592Mz Processor: 89S52. Press S1 to UP Count and S4 for Down count */

#include <reg52.h>
#include <intrins.h>

sbit    INC    = P1^0;                  //Increase Button
sbit    DEC    = P1^4;                  //Decrease Button
sbit    DSP1 = P3^5;                    //display 1 enable port line
sbit    DSP2 = P3^4;                    //display 2 enable port line
sbit    DSP3 = P3^6;                    //display 3 enable port line
sbit    DSP4 = P3^7;                    //display 4 enable port line

#define DIGDAT      P0                  //common digit lines

unsigned char code dig[] = { 0x88,0xeb,0x4c,0x49,0x2b,0x19,0x18,0xcb,0x8,0x9,0xa,0x38,0x98,
                             0x68,0x1c,0x1e };



/*-------------------------------------------------
MAIN C function
-------------------------------------------------*/
void main (void)
{
unsigned char cnt = 0;
unsigned char qot, rem;
unsigned char j;
unsigned int i;

`//AUXR = 0x01;                       //disable ALE
INC = 1;                              //Port as Input
DEC = 1;                              //Port as Input
DIGDAT = 0x00;

IE=0x00;
P2=0x00;

while(1)
 {
            if (INC == 0)      //INC key pressed
             {
                  if (cnt == 99) goto disp;
                  cnt++;
                  goto disp;
             }

            if (DEC == 0)     //DEC key pressed
             {
                  if (cnt == 0x00) goto disp;
                  cnt--;
```

```
                }

disp:
                qot = cnt / 10;
                rem = cnt % 10;

                for (i=0; i< 1000; i++)                 //change to inc/dec sensitivity of key press
                {
                        DIGDAT = dig[qot];
                        DSP2 = 1;
                        for (j=0;j<10;j++);             //change to inc/dec brightness of display
                        DSP2 = 0;

                        DIGDAT = dig[rem];
                        DSP1 = 1;
                        for (j=0;j<10;j++);             //change to inc/dec brightness of display
                        DSP1 = 0;

                        DIGDAT = dig[0];
                        DSP3 = 1;
                        for (j=0;j<10;j++);             //change to inc/dec brightness of display
                        DSP3 = 0;

                        DIGDAT = dig[0];
                        DSP4 = 1;
                        for (j=0;j<10;j++);             //change to inc/dec brightness of display
                        DSP4 = 0;
                }

 }
}
```

```
/*Program to demonstrate HEX Keypad Crystal freq 11.0592Mz Processor: 89S52. Switch ON Dip switch
DSW6 .Press S1 to switch on Relay and S4 to switch off the relay */

//////////////////////////////////////////
// Matrix Keypad Scanning Routine
//
// COL1 COL2 COL3 COL4
// 0    1    2    3    ROW 1
// 4    5    6    7    ROW 2
// 8    9    A    B    ROW 3
// C    D    E    F    ROW 4
//////////////////////////////////////////


#include <reg52.h>
#include <intrins.h>
#include "lcd.h"

#define ROWPORT  P1                              //P1.0 - P1.3 are Inputs
Sbit     COL1 = P1^4;
Sbit     COL2 = P1^5;
Sbit     COL3 = P1^6;
Sbit     COL4 = P1^7;

unsigned char code logo1[] = "HEX KEY-PAD DEMO";
unsigned char code logo2[] = "KEY PRESSED = N ";

/*-------------------------------------------------
MAIN C function
-------------------------------------------------*/
void main (void)
{
unsigned char key, i;
unsigned char rval[] = {0xFE,0xFD,0xFB,0xF7,0x0};
unsigned char keyPadMatrix[] =
{

   'F','B','7','3',
   'E','A','6','2',
   'D','9','5','1',
   'C','8','4','0'
};
 /* we scan the keypad by turning on the row outputs and / reading the columns*/
   COL1 = 1;
   COL2 = 1;
   COL3 = 1;
   COL4 = 1;

IE=0x00;
P3=0x00;
LCD_Init();
```

```
LCD_Write16(logo1,FIRST_LINE);
LCD_Write16(logo2,SECOND_LINE);

while (1)
 {
   key = 0;
   for( i = 0; i < 4; i++ )
   {
      // turn on row output
              ROWPORT = rval[i];
     // read colums - break when key press detected
      if( COL1 == 0)
         break;
      key++;
      if( COL2 == 0)
         break;
      key++;
      if( COL3 == 0)
         break;
      key++;
      if( COL4 == 0)
         break;
      key++;
   }
   if (key == 0x10)
     LCD_Write16(logo2,SECOND_LINE);
else
    {
     WriteLCD(0xCE,CONTROL_REG);
    WriteLCD(keyPadMatrix[key],DATA_REG);
   }
 }
}
```

```
/* Program to Switch on and off a Relay Crystal freq 11.0592Mz,Processor: 89S52 Press S1 to switch on
Relay and S4 to switch off the relay */

#include <reg52.h>
#include <intrins.h>
sbit    Relay  = P3^0;                          //Relay control line
sbit   ON      = P1^0;                          //ON KEY
sbit  OFF     = P1^4;                           //OFF KEY
/*-------------------------------------------------
MAIN C function
-------------------------------------------------*/
void main (void)
{
        IE   = 0x00;
        P2 = 0x00;
        P3 = 0x00;
        ON= 1;                                  //Port as Input
        OFF = 1;                                //Port as Input

while (1)
 {
          if (ON == 0)    //ON key pressed
         {
                Relay=1;
         }
         if (OFF == 0)     //OFF key pressed
         {
                Relay=0;
         }
  }
}
```