Topview Simulator

Software User Guide

FRONTLINE
ELECTRONICS

For Technical or Customer Support

You can reach Frontline Electronics Pvt, Ltd for the technical support and application assistance in following ways:

Email questions to: **feplslm@frontlinemail.com**

Send questions by mail to:

Frontline Electronics Pvt Ltd.,
1/255C - Thathagounder St,
Kumaran Nagar,
Alagapuram,
Salem - 636 016,
Tamilnadu,
India.

Phone	:	0427 244 9238 / 243 1312.
Fax	:	0427 244 9010.

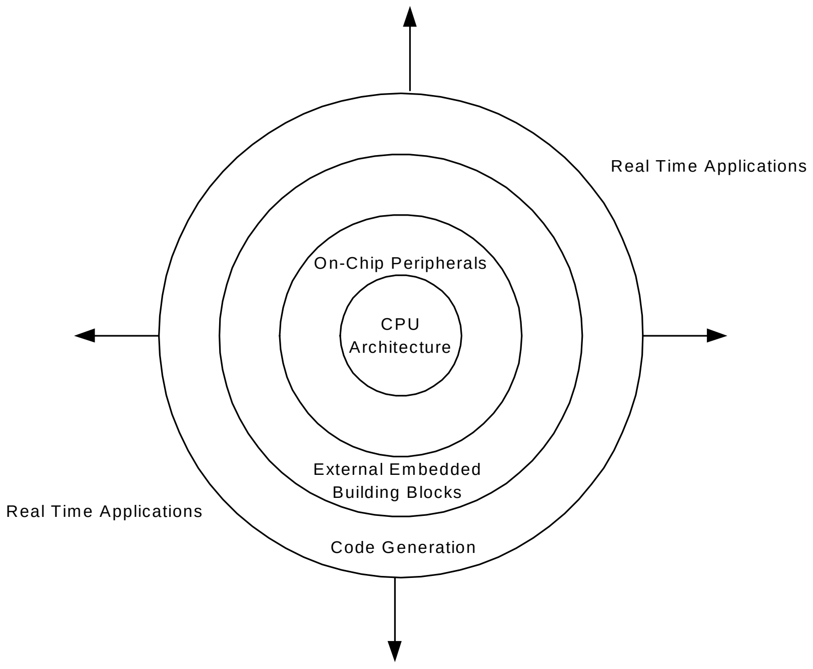
Web site : **www.Frontline-Electronics.com**

Contents

1.1 - Introduction

Topview Simulator gives an excellent simulation environment for the Industry's most popular 8 bit microcontroller family, MCS 51. It gives all the required facilities to enable the system designers to start projects right from the scratch and finish them with ease and confidence.

The following figure indicates the facilities available in the simulation environment that give you required development power to handle your next real time embedded system design applications.



Topview Simulator is the total solution giving many state of art features meeting the needs of the designers possessing different levels of expertise. If you are a beginner, then you can easily learn about 8031 based embedded solutions without any hardware. If you are an experienced designer, you may find most of the required facilities built in the simulator that enable you to complete your next project without waiting for the target hardware.

The simulator is designed by the active feedback from the demanding designers and when you use this in your next 8031 project, you are assured of definite savings in time and increase in productivity.

The features of the simulator are briefly tabulated here.

Device Selection

A wide range of device selection, including generic 8031 devices and Atmel's AT89CXX series 8031 microcontrollers.

Program Editing

Powerful editing feature for generating your programs and the facility to call an external assembler to process input programs.

ClearView GUI Environment

ClearView GUI facility gives all the internal architectural details in the strategically placed windows. Information about the Program, Data Memory, Registers, Peripherals, SFR Bits, Memory Bits are clearly presented in many windows to make you understand the program flow very easily.

Program Execution

A variety of program execution options include Single Stroke full speed execution, SingleStep, StepOver and BreakPoint execution modes give you total control over the target program.

ClearView updates all the windows with the correct and latest data and it is a convenient help during your debugging operations.

You may find how this Topview Simulator simplifies the most difficult operation of the program development, debugging, into a most simple task.

Simulation Facilities

Powerful simulation facilities are incorporated for I/O lines, interrupt lines and the clock sources meant for Timers/Counters.

Many external embedded building blocks can be simulated:

- Range of Plain Point LED's and Seven Segment LED Display options.
- LCD modules in many configurations.
- Momentary ON keys, Toggle Switches.
- A variety of keypads upto 4 X 8 key matrix.
- All modes of on-chip serial port communication facility.
- IIC components including RTC, EEPROMs.
- SPI Bus based EEPROM devices.

Code Generation Facilities

Powerful and versatile code generating facility enables you to generate the exact and compact assembly code for many possible application oriented interfacing options.

You can simply define your exact needs and get the target assembly code at a press of button at anywhere in your program flow. The code gets embedded into your application program automatically.

You are assured trouble free working of final code in the real time.

- All modes of the serial port.
- Interfacing IIC, SPI Bus devices.
- Range of keypads.
- Many LED/LCD interfacing possibilities.

1.2 - System Requirements

The Topview Simulator operates in windows 95/98 environments. The required hard disk space is about 5MB.

Computer with colour monitor, 800 X 600 pixel resolution setting is preferred because contents of many windows are displayed in different colours for your visual convenience.

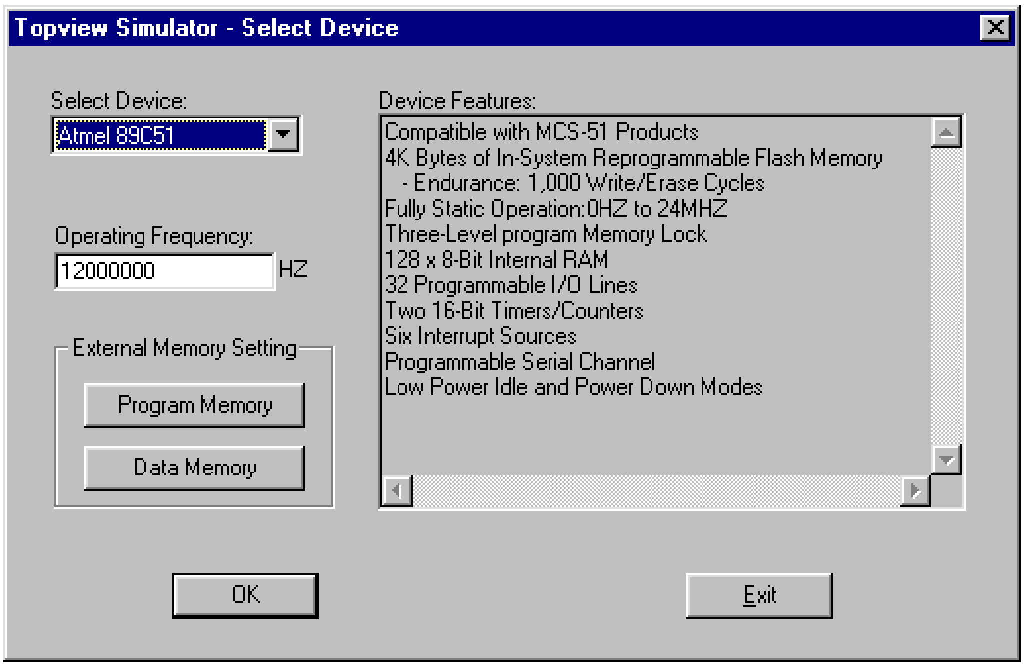
2.1 - Introduction

This chapter gives information on how to start Topview Simulator for your applications and other relevant operations. In case, if you need more assitance in using the simulator, check online help.

2.2 - Device Selection

Whenever the simulator is activated, an opening screen comes alive in the monitor asking you to select the device, operating frequency and memory options as shown here:

Select any one device from the combo box with the name **Select Device**. The accompanied box displays the features of the selected device for your reference. You can make sure that you selected a right microcontroller for the target application.


Topview Simulator - Select Device

Select Device: Atmel 89C51

Operating Frequency: 12000000 HZ

Device Features:
Compatible with MCS-51 Products
4K Bytes of In-System Reprogrammable Flash Memory
- Endurance: 1,000 Write/Erase Cycles
Fully Static Operation:0HZ to 24MHZ
Three-Level program Memory Lock
128 x 8-Bit Internal RAM
32 Programmable I/O Lines
Two 16-Bit Timers/Counters
Six Interrupt Sources
Programmable Serial Channel
Low Power Idle and Power Down Modes

After the device selection, enter the operating frequency. This frequency value is taken as the master reference for all the operations of the simulator. Enter a value that is less than or equal to the maximum operating frequency as specified by the device manufacturers. Otherwise you will get an error message.

The possible microcontroller selections are :

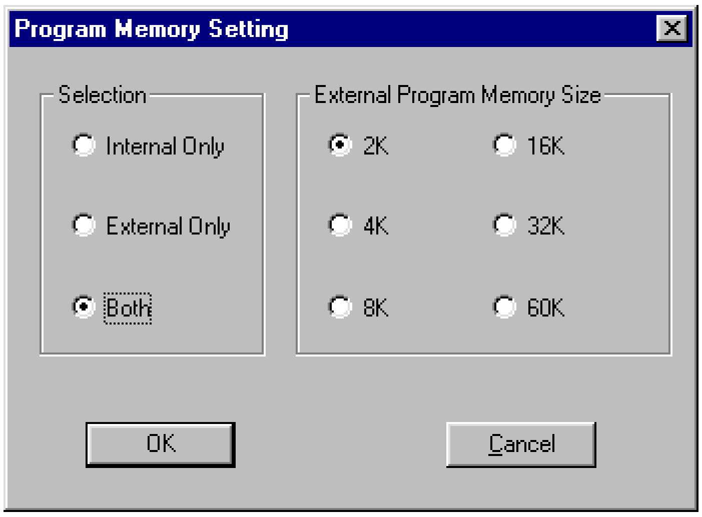
- Generic 8031 microcontrollers:
  8031, 8051, 8032, and 8052.

- Atmel's AT89CXX microcontrollers :
  89C1051, 89C2051, 89C4051, 89C51, 89C52, 89C55, 89S53, 89S8252.

Apart from this, there is a facility to simulate 8031 in single chip and expanded configuration. Provision is there to set address range of both external Program and Data Memories.

Now, set the amount of program memory you want by clicking over **Program Memory** button under **External Memory Setting**.

When this button is activated, a dialog box appears as shown here :

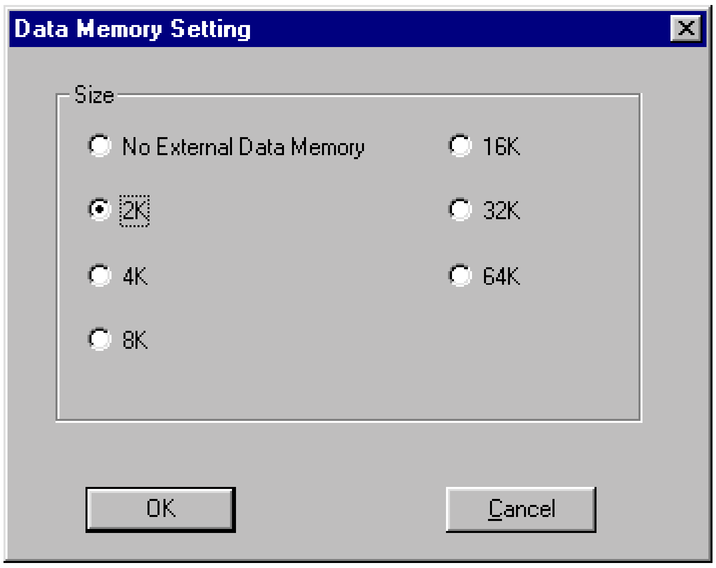

Program Memory Setting

Selection: Internal Only / External Only / [✓ Both]

External Program Memory Size: [✓ 2K] / 16K / 4K / 32K / 8K / 60K

First select any one in the heading, **Selection**. Then choose amount of memory and click **OK** button.

Similarly to set the amount of external data memory, click over **Data Memory** button. This will open another dialog box :



Size: ( ) No External Data Memory [✓ 2K] ( ) 4K ( ) 8K ( ) 16K ( ) 32K ( ) 64K

You can choose any option out of possible seven and click **OK** button.

Press **OK** button to close the **Select Device** dialog box and the simulator will be set according to the selected device. All the generic features like, internal program memory size, clock speed will be assigned with the selected values. Also all the peripheral functions of the device are enabled by default.

Now the simulator is ready to get into action.

Some settings and status are displayed in the status bar : Current Register Bank, Current BreakPoint Status and the Total Machine cycles count.

2.3 - Simulator Operations

Apart from above mentioned, simulator has plenty of operations :

- File operations.
- Memory operations.
- Window operations.
- Program Execution operations.
- BreakPoint, SingleStep operations.
- Activating Internal/External interfacing options.

You can know more information on these in the next chapters.

3.1 - Introduction

Topview Simulator sports an extensive and user friendly GUI environment using many windows to present the information in a more productive way. There are shortcut ways to combine many operations in single steps to enable you implement repeated tasks in less time.

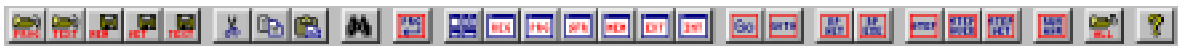
The details are given in the subsequent chapters and now this chapter introduces this GUI environment to you.

3.2 - Simulator Toolbar

Topview Simulator uses many user friendly windows to implement an effective GUI interface. When you activate this simulator, an opening screen prompts you to select the target microcontroller.

Once you select the microcontroller, the dialog box disappears leaving behind a blank screen with the Tool Bar and a Menu Bar.

For your convenience, the tool bar is given below and a short description on each command buttons. The simulator gives you same explanation about each of these buttons whenever you keep mouse cursor over that button.



This command loads the program from the disk into the program memory of the simulator. The file may be either in Hex or Binary format.

It is same as activating **Load Program** from the **File** menu.

This button loads a text file into the Built-in Editor. This is same as **Load Text File** from **File** menu.

Simulator Toolbar

This button saves the memory contents of the simulator in the disk.

It is similar to **Save Memory** command of **File** menu.

Makes the computer remember the current windows position and size for future reference.

This is equivalent to **Save Setting** of **File** menu.

This button saves the current editor contents to disk.

It is similar to **Save Text File** command from **File** menu.

Enables you to key in your program into the program memory of the simulator using built-in single line assembler.

It is same as activating **Enter Program** command of **Memory** menu.

By clicking this button, you can make the ClearView window structure alive from the NormalView.

It is same as activating **ClearView** command from **View** menu.

This button switches the simulator window to Normal windows mode from the ClearView Windows arrangement.

It is equivalent to activating **NormalView** in **View** menu.

This button opens the Register window or sets the Register window as active if it is already opened in the background.

It is same as activating **Register Window** in **View** menu

This opens the Program Window or sets the Program Window as active, If it is hidden in the background.

It is equivalent to activating **Program Window** from the **View** menu.

This button opens the SFR Window or makes the window active if it is in the background.

It is same as clicking over **SFR Bit Status Window** from **View** menu.

This button is responsible for making Memory Bit Status Window active. If this window is not available in the background, a fresh window will be opened.

It is same as activating the **Memory Bit Status Window** from **View** menu.

This button opens the External Data Memory. If the window is hidden at the background, it is brought to front again.

It is equivalent to activating **External Memory Window** from **View** menu.

This button is responsible to make the Internal Data Memory as active.

You can use **Internal Data Memory** command from **View** menu for doing this.

This button executes your program in full speed taking PC contents or user defined address (already set) as the starting address.

You can activate this command by **Go** from **Run** menu.

Executes your program in full speed after getting the starting address using a dialog box.

It is same as activating **Goto** from **Run** menu.

It enables you to define the BreakPoint.

It is equivalent to **Setting** from **BreakPoint** menu.

Executes your program in BreakPoint mode taking the PC contents as starting address.

It is equivalent to **Execution** command of **BreakPoint** menu.

Executes your program in SingleStep mode taking PC contents as starting address.

It is same as activating **SingleStep** command from the **SingleStep** menu.

Executes the program in StepOver mode taking PC contents as starting address.

It is also available by clicking **StepOver** command in **SingleStep** menu.

Activate the SingleStep setting command.

It is same as **Setting** command of **SingleStep** menu.

This is About command.

Same as **Help** menu's **About** command.

Cuts the selected text in the Editor.

Copies the selected text into the Clipboard.

Same as activating **Copy** command in **Edit** menu.

Pastes the text kept in the Clipboard into the Editor.

Equivalent to **Paste** in **Edit** menu.

Find command meant for the Text Editor.

Runs the external Assembler.

It is equivalent to activating **Run Assembler** from the **Command** menu.

Closes all the opened windows in a single command.

Same as **Close All** of **Window** menu.

Stops the program execution.

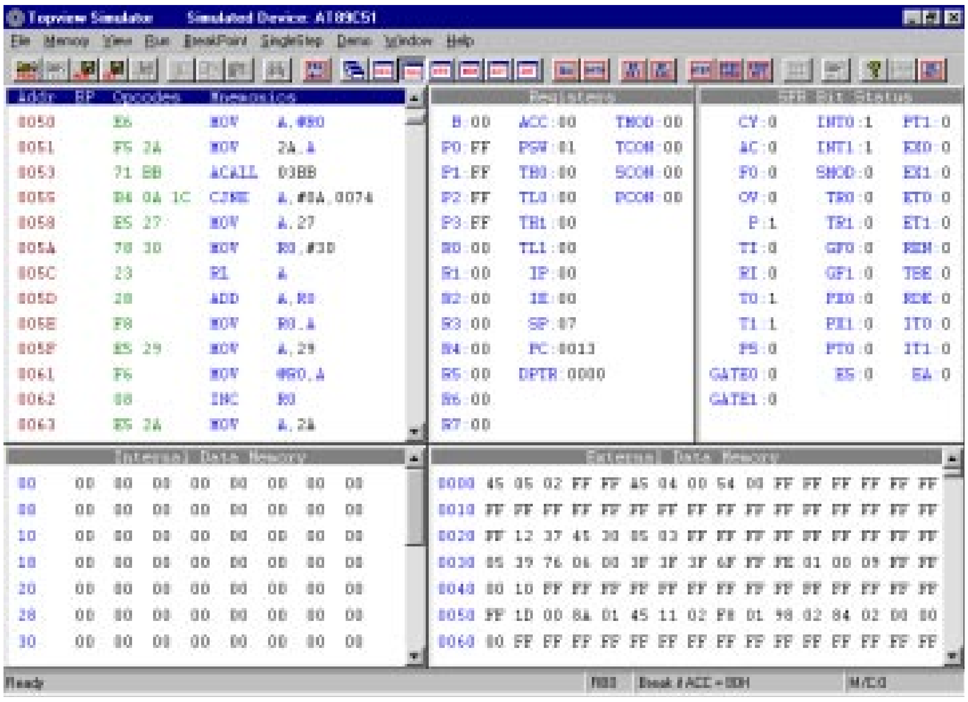
Same as **Stop Execution** command from the **Run** menu.

Gives a Reset signal to simulator.

Same as **Reset CPU** command from **Run** menu.

3.3 - ClearView Window Structure

This is an optimized arrangement where windows are strategically placed in the display. Total display area of the monitor is divided into 5 windows. Windows meant for Program, Register, Internal Data Memory, External Data Memory and SFR Bit Status are placed in the ClearView.


Topview Simulator   Simulated Device: AT89C51
File Memory View Run BreakPoint SingleStep Demo Window Help

```
Addr  BP  Opcodes    Mnemonics
0050      E6         MOV    A,@R0
0051      F5 2A      MOV    2A,A
0053      71 BB      ACALL  03BB
0055      B4 0A 1C   CJNE   A,#0A,0074
0058      E5 27      MOV    A,27
005A      78 30      MOV    R0,#30
005C      23         RL     A
005D      28         ADD    A,R0
005E      F8         MOV    R0,A
005F      E5 29      MOV    A,29
0061      F6         MOV    @R0,A
0062      08         INC    R0
0063      E5 2A      MOV    A,2A
```

```
Registers
 B:00    ACC:00    TMOD:00
P0:FF    PSW:01    TCON:00
P1:FF    TH0:00    SCON:00
P2:FF    TL0:00    PCON:00
P3:FF    TH1:00
R0:00    TL1:00
R1:00     IP:00
R2:00     IE:00
R3:00     SP:07
R4:00     PC:0013
R5:00   DPTR:0000
R6:00
R7:00
```

```
SFR Bit Status
   CY:0   INT0:1   PT1:0
   AC:0   INT1:1   EX0:0
   F0:0   SMOD:0   EX1:0
   OV:0    TR0:0   ET0:0
    P:1    TR1:0   ET1:0
   TI:0    GF0:0   REN:0
   RI:0    GF1:0   TBE:0
   T0:1    PX0:0   RDE:0
   T1:1    PX1:0   IT0:0
   PS:0    PT0:0   IT1:0
GATE0:0     ES:0    EA:0
GATE1:0
```

```
Internal Data Memory
00    00 00 00 00 00 00 00 00
08    00 00 00 00 00 00 00 00
10    00 00 00 00 00 00 00 00
18    00 00 00 00 00 00 00 00
20    00 00 00 00 00 00 00 00
28    00 00 00 00 00 00 00 00
30    00 00 00 00 00 00 00 00
```

```
External Data Memory
0000 45 05 02 FF FF A5 04 00 54 00 FF FF FF FF FF FF
0010 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0020 FF 12 37 45 30 85 03 FF FF FF FF FF FF FF FF FF
0030 85 39 76 06 00 3F 3F 3F 6F FF FE 01 00 09 FF FF
0040 00 10 FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0050 FF 1D 00 8A 01 45 11 02 F8 01 98 02 84 02 00 00
0060 00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

```
Ready                RB0   Break if ACC = 00H    M/C:0
```

Size and position of the windows can’t be changed. Scrolling facility is available wherever it is required.

This window structure can be activated or all these windows can also be viewed in NormalView by clicking suitable tool buttons.

This ClearView gives you complete picture on the internal architecture in a single screen when you debug your target program code.

3.4 - NormalView Window Structure

In this arrangement, you can activate as many windows as required during debugging process. All these windows are tiled in many ways.

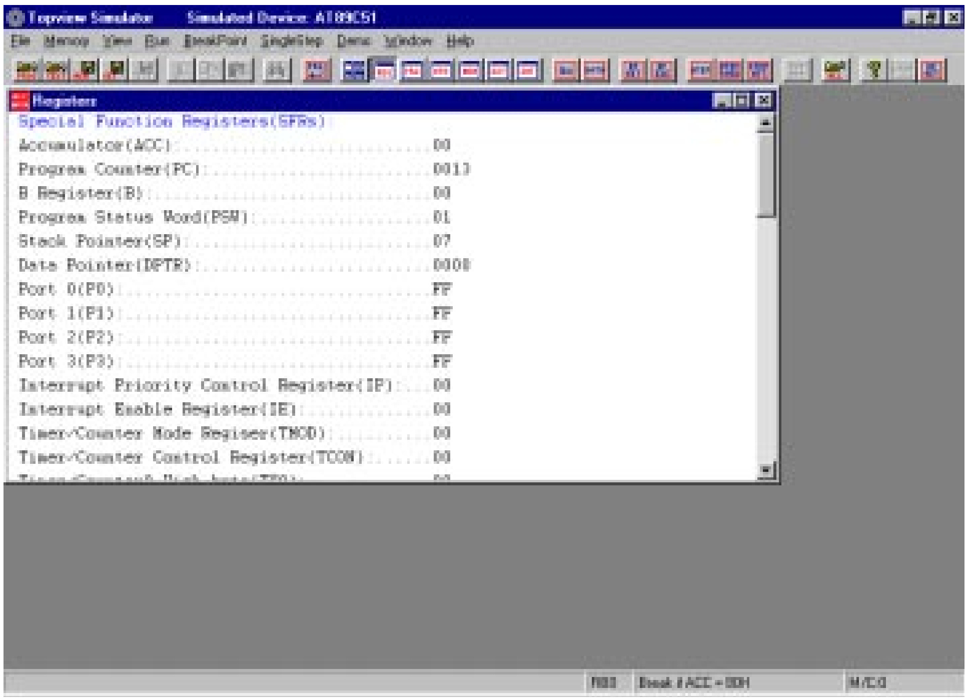
You can activate NormalView window structure at any time at the press of a button.

3.5 - Register Window

Click **Register Window** in the **View** menu or click the relevant button of the toolbar to open or activate this window.

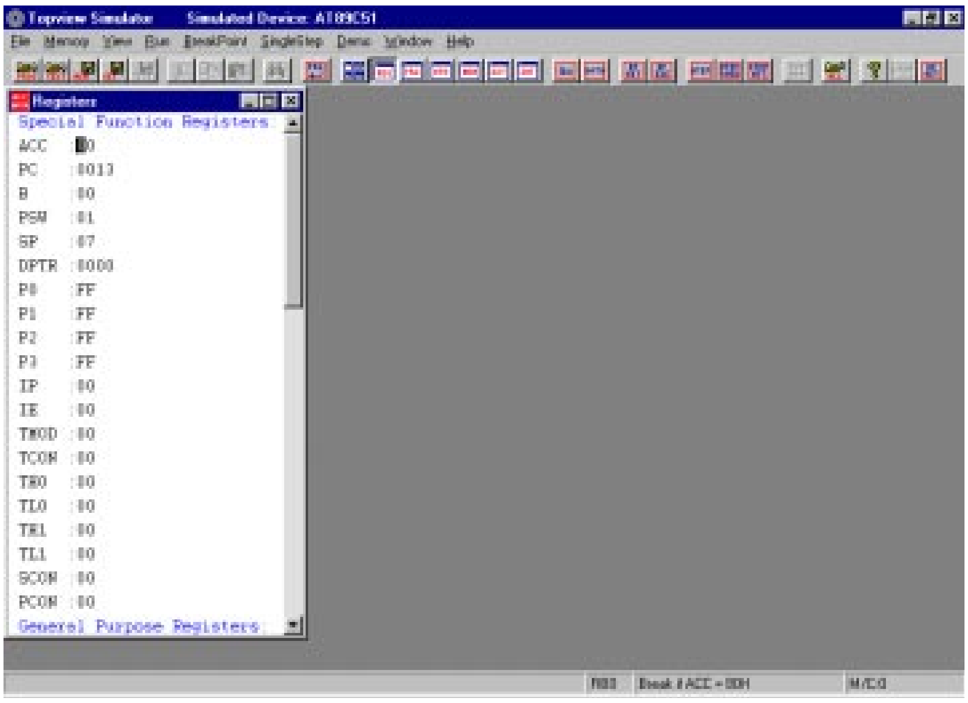
This Register Window indicates all the available internal registers and their contents. You can see the complete name of the register and you can edit the contents by clicking over the register contents.

The cursor starts blinking wherever you click. You can modify the register contents at that place. When you enter any data, the same will be automatically transferred to that particular location.


Topview Simulator   Simulated Device: AT89C51
File Memory View Run BreakPoint SingleStep Demo Window Help
Registers
RB0
Break if ACC = 00H
M/C 0

```
Special Function Registers(SFRs)
Accumulator(ACC):.........................00
Program Counter(PC):......................0013
B Register(B):............................00
Program Status Word(PSW):.................01
Stack Pointer(SP):........................07
Data Pointer(DPTR):.......................0000
Port 0(P0):...............................FF
Port 1(P1):...............................FF
Port 2(P2):...............................FF
Port 3(P3):...............................FF
Interrupt Priority Control Register(IP):..00
Interrupt Enable Register(IE):............00
Timer/Counter Mode Regiser(TMOD):.........00
Timer/Counter Control Register(TCON):.....00
```

If you reduce the size of this Register Window, the window will reconfigure itself as shown here:


Topview Simulator Simulated Device: AT89C51
File Memory View Run BreakPoint SingleStep Demo Window Help
Registers
R03
Break # ACC = 00H
M/C:0

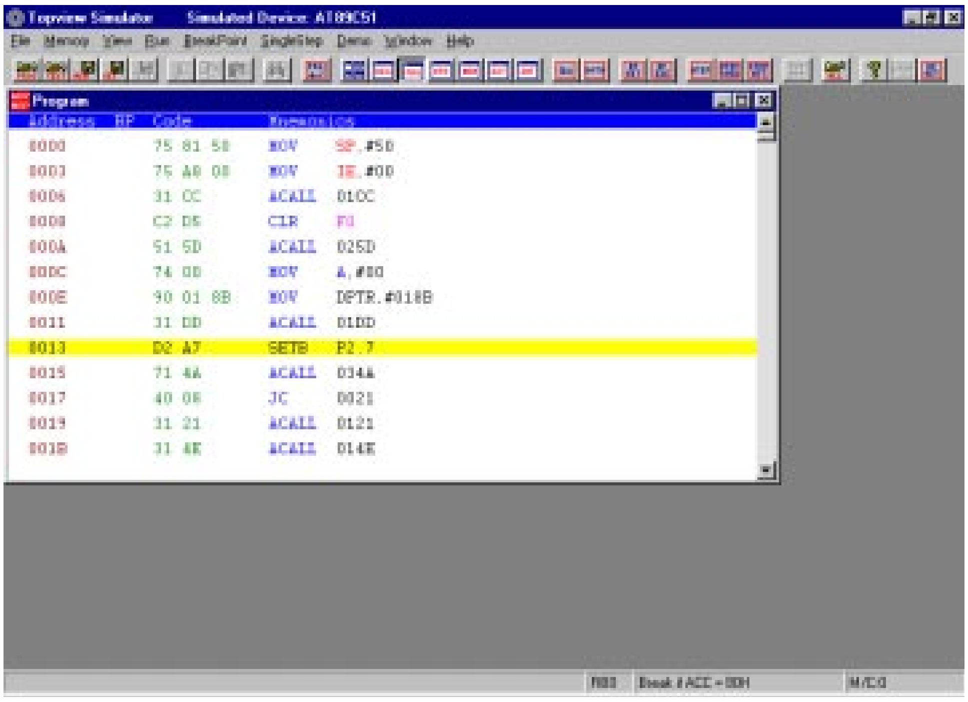
```
Special Function Registers
ACC   :00
PC    :0013
B     :00
PSW   :01
SP    :07
DPTR  :0000
P0    :FF
P1    :FF
P2    :FF
P3    :FF
IP    :00
IE    :00
TMOD  :00
TCON  :00
TH0   :00
TL0   :00
TH1   :00
TL1   :00
SCON  :00
PCON  :00
General Purpose Registers
```

3.6 - Program Window

You can open this window by clicking **Program Window** in the **View** menu. If the window is in the background, it now becomes active and comes to the front.

This window disassembles the program memory contents as shown in the figure.

Program Window

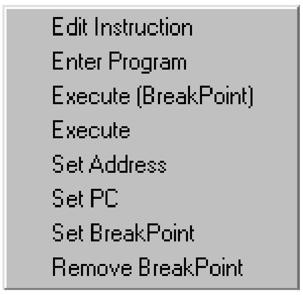

Topview Simulator
Simulated Device: AT89C51
File Memory View Run BreakPoint SingleStep Demo Window Help
Program
R00
Break # ACC = 00H
M/C:0

```
Address  BP  Code        Mnemonics
0000         75 81 50    MOV    SP,#50
0003         75 A8 00    MOV    IE,#00
0006         31 CC       ACALL  01CC
0008         C2 D5       CLR    F0
000A         51 5D       ACALL  025D
000C         74 00       MOV    A,#00
000E         90 01 8B    MOV    DPTR,#018B
0011         31 DD       ACALL  01DD
0013         D2 A7       SETB   P2.7
0015         71 4A       ACALL  034A
0017         40 08       JC     0021
0019         31 21       ACALL  0121
001B         31 4E       ACALL  014E
```

Addresses, Opcodes and Mnemonics are displayed in different colours for your visual convenience.

Keeping this window active, you can enter your program line by line using built-in Single line assembler. You can know more information on this in the chapter 10.

You can use mouse to do many useful tasks when you are in this program window. Just right click your mouse anywhere in this window and you will get a pop up window that gives a list of operations you can perform as shown:



You can proceed with any of these commands by left clicking over the name of the commands.

- Edit Instruction
- Enter Program
- Execute (BreakPoint)
- Execute
- Set Address
- Set PC
- Set BreakPoint
- Remove BreakPoint

3.6.1 - Edit Instruction

Using this command, you can edit the disassembled program you are viewing at that time of mouse click in the program window.

As you know already, this program window displays the raw program in the assembly language. Most of the time, you may use this window to read the target program from the memory of the microcontroller.

But if you need to develop or key in your target program right from the scratch, you have to use the single line assembler or the external assembler. You can know more of this in the coming chapters.

So, this Edit Instruction is suitable for making small changes or editing only few instructions in the program window.

You can change a 3 byte instruction into a 2 byte or 1 byte instructions. You can’t change a 1 byte to 2 / 3 byte instructions for obvious reasons. When you change a 3 byte instruction into 1 / 2 byte instructions the remaining locations are filled with NOP (00H) instruction.

You can also activate this **Edit Instruction** command by double clicking left mouse button over the exact instruction in the window.

3.6.2 - Enter Program

Activating this command enables you to enter your target program using built-in single line assembler.

Detailed information is available in the chapter 10.

3.6.3 - Execute BreakPoint

This command takes you to the BreakPoint execution. Details are available in chapter 13.

3.6.4 - Execute

This command is for executing your target program in the microcontroller and you can get the complete picture in chapter 13.

3.6.5 - Set Address

You can set the starting address of the program list in the program window. You can also activate this command by double clicking left button of your mouse over the address portion displayed in the window.

A dialog box comes up to get the starting address.

3.6.6 - Set PC

By this command, you can set the PC with the address where you click the mouse.

The initialization of the PC is indicated by the yellow highlight at that instruction.

3.6.7 - Set BreakPoint

You can set any instruction as the PC BreakPoint by this command.

You can get more information about this BreakPoint operations in the chapter 14.

3.6.8 - Remove BreakPoint

If you activate this command at any specific instruction, which is already set for the PC BreakPoint, then that PC BreakPoint condition is removed from that instruction.

3.6.9 - Clear Program Memory

This is another simple command to clear Program Memory area and the total space is initialized with the data FFH.

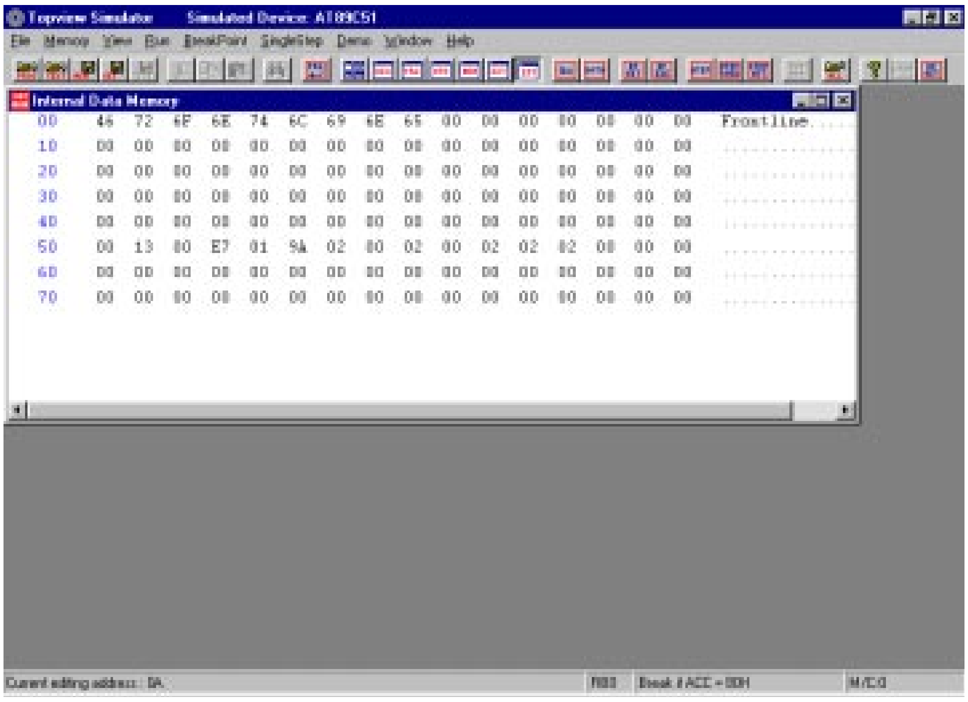
You can activate this command by **Memory → Program Memory → Clear**.

3.7 - Internal Data Memory Window

Activate this window by clicking **Internal Data Memory Window** from the **View** menu.

This window displays the memory addresses and the contents. The data are displayed in both Hex and ASCII formats. When you are using the ClearView, you can see the contents of this window in either one of these formats. To change the contents from one format to another, right click your mouse anywhere in the window and select the suitable option from the pop up window.

You can also edit the contents in this window.



Topview Simulator
Simulated Device: AT89C51
File Memory View Run BreakPoint SingleStep Demo Window Help
Internal Data Memory

```
00   46 72 6F 6E 74 6C 69 6E 65 00 00 00 00 00 00 00   Frontline.......
10   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
20   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
30   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
40   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
50   00 13 00 E7 01 9A 02 00 02 00 02 02 02 00 00 00   ................
60   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
70   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

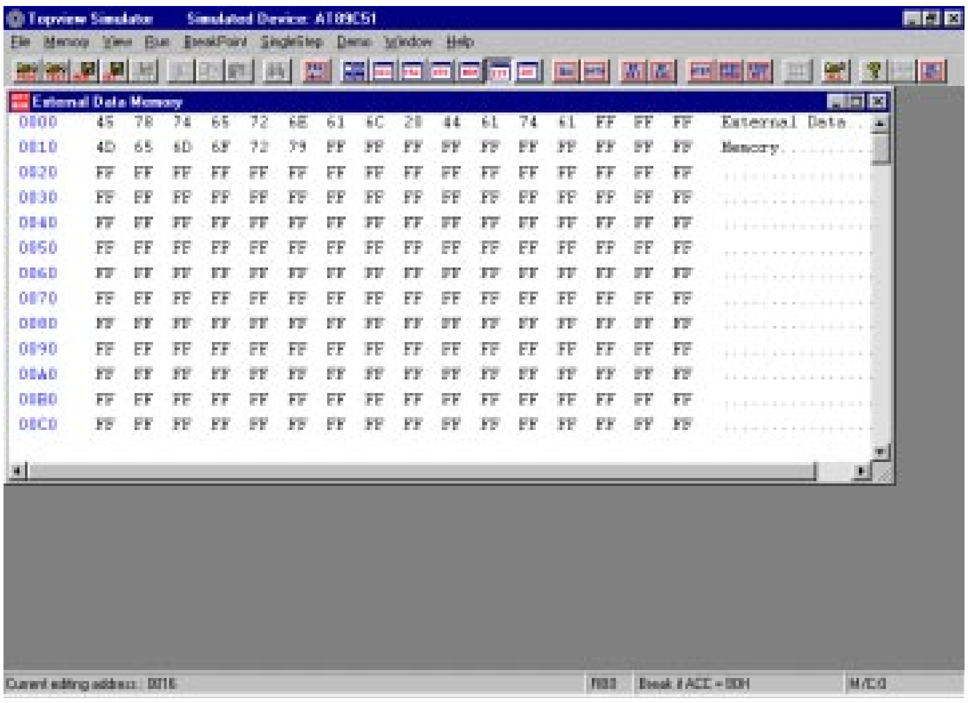
```
Current editing address : 5A     RB0     Break if ACC = 00H     M/C:0
```

3.8 - External Data Memory Window

To open the external data memory window or set external data memory window as current active window, this command can be used.

To activate this command, click **External Data Memory Window** from **View** menu.

When you click this command, the external data memory window gets opened or becomes active if it is in the background. This window displays the contents of the external memory along with the address. The data are displayed in both Hex and ASCII formats.



```
0000    45 78 74 65 72 6E 61 6C 20 44 61 74 61 FF FF FF   External Data...
0010    4D 65 6D 6F 72 79 FF FF FF FF FF FF FF FF FF FF   Memory..........
0020    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
0030    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
0040    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
0050    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
0060    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
0070    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
0080    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
0090    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00A0    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00B0    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00C0    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
```

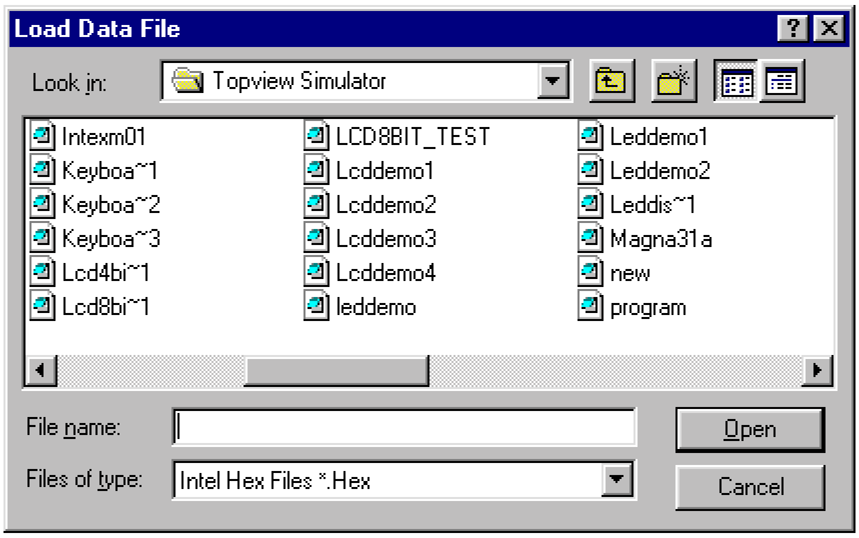
Current editing address : 0016   RB0   Break if ACC = 00H   M/C:0

3.8.1 - Load Data Into Data Memory

This command is meant for loading a Hex or Binary file into Data Memory of the simulator from the disk/hard disk.

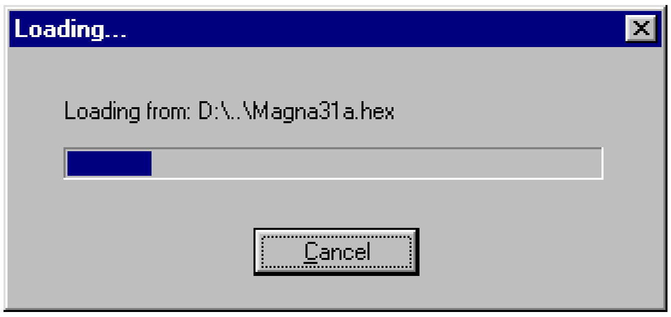
Click **File → Load Data** to activate this command. Short cut key is **Ctrl+D**.

You should see a dialog box coming up asking you to select the target file.



Select the file and click **Open** button.

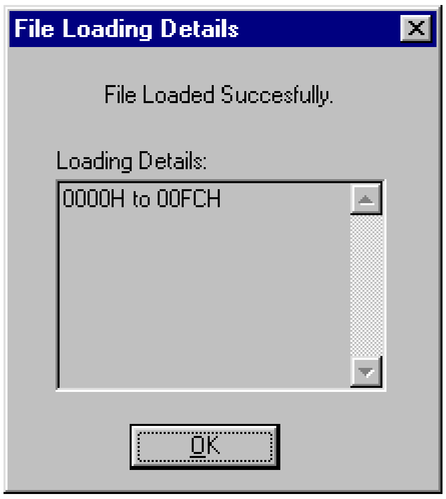
Another small window pops up indicating program loading using a progressive display.



Loading from: D:\..\Magna31a.hex

At the end of this command execution, you can notice the summary of data loaded into the memory.

Load Data Into Data Memory



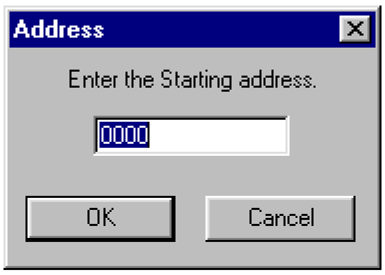
File Loaded Succesfully.

Loading Details:

0000H to 00FCH

Pressing **OK** button completes this command execution.

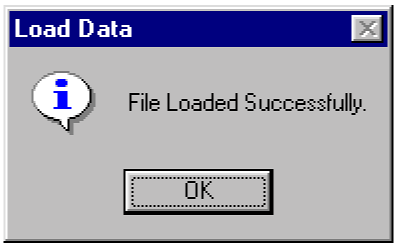
When you select a Binary file, you need to key in the starting address in a dialog window.



Enter the Starting address.

0000

Press **OK** button after entering the starting address. The status of this binary file loading is indicated by another window.



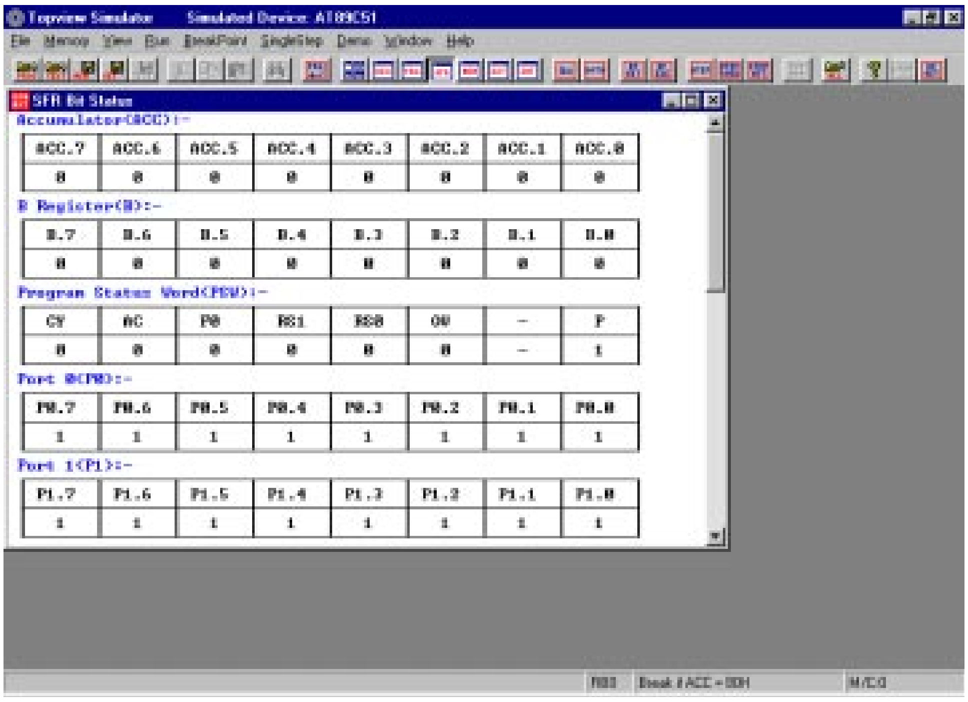
File Loaded Successfully.

If the entered address exceeds the available data memory capacity, the error condition is indicated.

3.9 - SFR Bit Status Window

Activate this command, by clicking **SFR Bit Status Window** in **View** menu. You can see this window comes into action indicating SFR contents in bit wise along with their names.

You can also edit the contents in the window.


Topview Simulator   Simulated Device: AT89C51
File Memory View Run BreakPoint SingleStep Demo Window Help
SFR Bit Status

Accumulator(ACC):-

ACC.7	ACC.6	ACC.5	ACC.4	ACC.3	ACC.2	ACC.1	ACC.0
0	0	0	0	0	0	0	0

B Register(B):-

B.7	B.6	B.5	B.4	B.3	B.2	B.1	B.0
0	0	0	0	0	0	0	0

Program Status Word(PSW):-

CY	AC	F0	RS1	RS0	OV	-	P
0	0	0	0	0	0	-	1

Port 0(P0):-

P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
1	1	1	1	1	1	1	1

Port 1(P1):-

P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
1	1	1	1	1	1	1	1

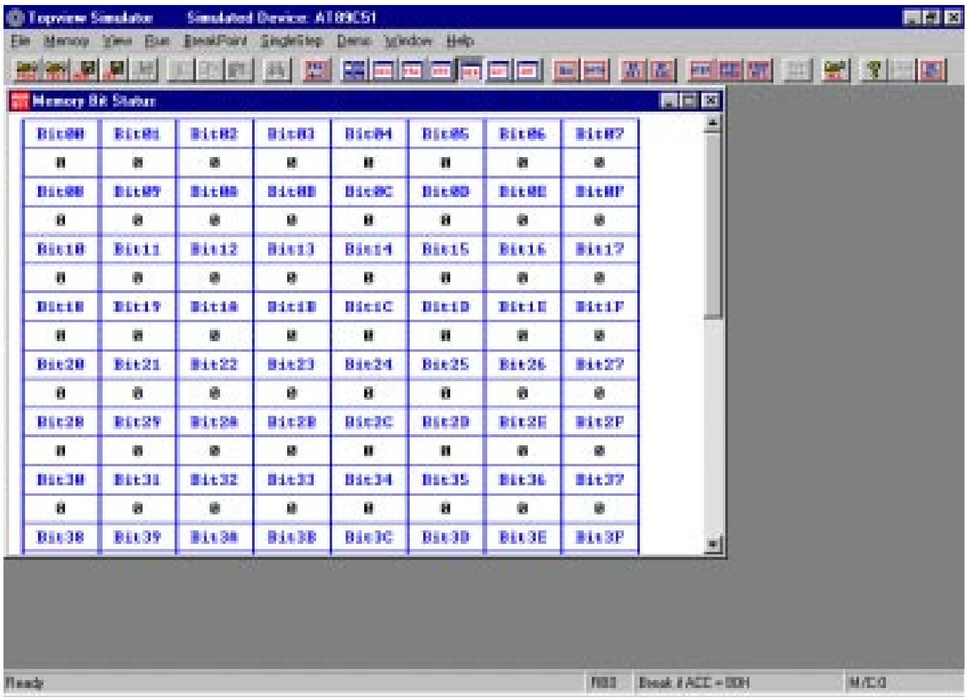
RB0   Break # ACC = 00H   M/C:0

3.10 - Memory Bit Status Window

In this window, the bit addressable Internal Data Memory (20H - 2FH) contents are displayed in bit wise along with their bit addresses.

You can also change the content of any bit when this window is in active condition.

Memory Bit Status Window


Topview Simulator   Simulated Device: AT89C51
File Memory View Run BreakPoint SingleStep Demo Window Help
Memory Bit Status
Ready
R00
Break / ACC = 00H
M/C:0

Bit00	Bit01	Bit02	Bit03	Bit04	Bit05	Bit06	Bit07
0	0	0	0	0	0	0	0
Bit08	Bit09	Bit0A	Bit0B	Bit0C	Bit0D	Bit0E	Bit0F
0	0	0	0	0	0	0	0
Bit10	Bit11	Bit12	Bit13	Bit14	Bit15	Bit16	Bit17
0	0	0	0	0	0	0	0
Bit18	Bit19	Bit1A	Bit1B	Bit1C	Bit1D	Bit1E	Bit1F
0	0	0	0	0	0	0	0
Bit20	Bit21	Bit22	Bit23	Bit24	Bit25	Bit26	Bit27
0	0	0	0	0	0	0	0
Bit28	Bit29	Bit2A	Bit2B	Bit2C	Bit2D	Bit2E	Bit2F
0	0	0	0	0	0	0	0
Bit30	Bit31	Bit32	Bit33	Bit34	Bit35	Bit36	Bit37
0	0	0	0	0	0	0	0
Bit38	Bit39	Bit3A	Bit3B	Bit3C	Bit3D	Bit3E	Bit3F

So far you have been informed about the basic GUI environment of the Topview Simulator. Apart from these, there is a facility to visualize all the internal peripherals of the selected microcontroller in the suitable windows.

You can also simulate external embedded building blocks meant for LEDs, LCD, serial buses like IIC, SPI, keyboard, RTC and EEPROMs.

You know more of these internal and external modules in the next chapters.

3.11 - Load and Save Screen Setting

You can save status of windows (opened or closed) and the location of the same for your future reference. This information is stored as a file with **.SST** extension.

To save the current window status, click **File → Save Screen Setting**. If you need to name this file, use **Save as Screen Setting** command.

When you want to load a previous file, click **File → Load Screen Setting** and select the file in the dialog window. Or press the shortcut key : **Ctrl+L**.

4.1 - Introduction

This Chapter gives more information about the simulation facilities available for all the peripherals of the 8031 microcontroller. All I/O port lines, Timers/Counters, serial communication port of the controller are simulated in all the possible ways. Similarly SPI port and the internal EEPROM of the device AT89S8252 and SPI port of AT89S53 are simulated.

All these simulation facilities are complete in all respects and you can use all the facilities of all the peripherals in your applications without any restriction.

4.2 - I/O Window

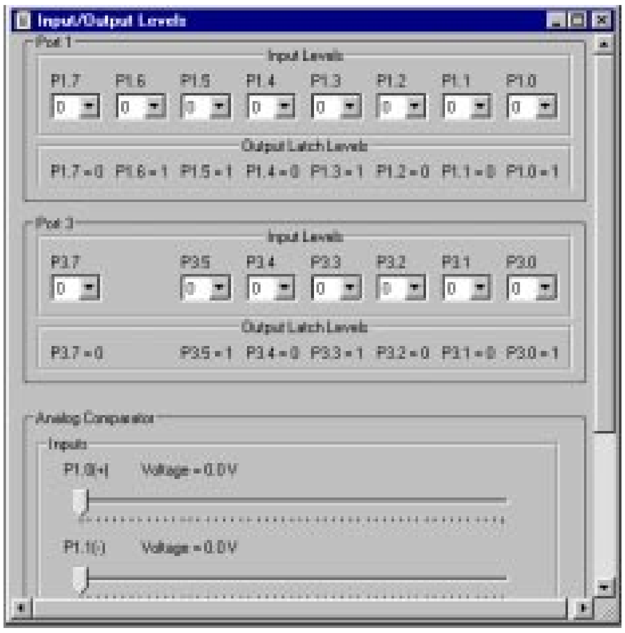
There are two I/O windows available for your applications. If you select any of the 20 pin controllers, you may get a smaller window with less I/O lines. But you can get all the features including the Analog Comparator (available in AT89C1051). You can also watch the analog comparator output at the corresponding output pin.

A much bigger I/O window is available for the 40 pin devices. This window comes with all the 4 ports of I/O lines with facility to simulate timers/counters at the respective I/O pins. Interrupt conditions can be simulated at your will at any time.

During program execution, you can make this I/O window active and using your mouse, you can simulate any I/O condition and watch for the expected results at the relevant I/O pins.

Activate this I/O window by clicking **I/O Window** from the **View** menu.

Depends upon the device selection, right I/O window comes up in the monitor for your usage.



Input/Output Levels

Port 1

Input Levels

P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
0	0	0	0	0	0	0	0

Output Latch Levels

P1.7 = 0 P1.6 = 1 P1.5 = 1 P1.4 = 0 P1.3 = 1 P1.2 = 0 P1.1 = 0 P1.0 = 1

Port 3

Input Levels

P3.7	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0
0	0	0	0	0	0	0

Output Latch Levels

P3.7 = 0 P3.5 = 1 P3.4 = 0 P3.3 = 1 P3.2 = 0 P3.1 = 0 P3.0 = 1

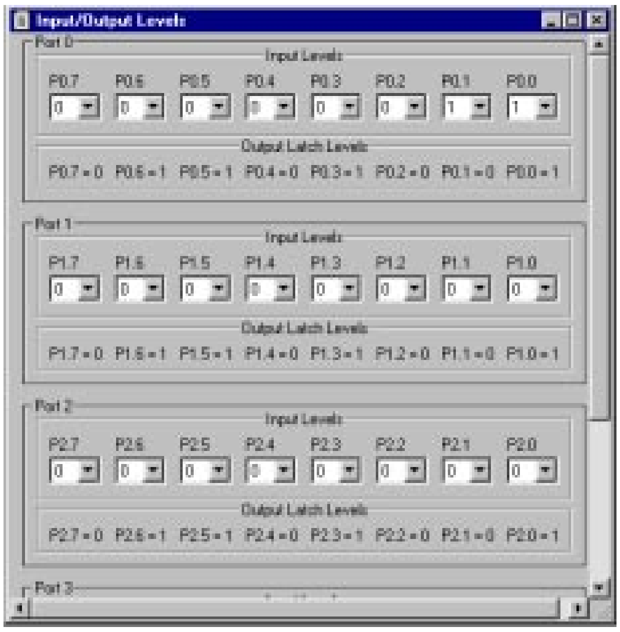
Analog Comparator

Inputs

P1.0(+) Voltage = 0.0V

P1.1(-) Voltage = 0.0V

All I/O pins have three options: Level **0,** Level **1,** and **NO CONNECTION**. You can leave any I/O pin open as you do in real time. Depending upon your output selection, all output conditions are promptly indicated in the window. If you leave any I/O pin in NC, then corresponding input represents the output latch level.



Input/Output Levels

Port 0

Input Levels

P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
0	0	0	0	0	0	1	1

Output Latch Levels

P0.7 = 0 P0.6 = 1 P0.5 = 1 P0.4 = 0 P0.3 = 1 P0.2 = 0 P0.1 = 0 P0.0 = 1

Port 1

Input Levels

P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
0	0	0	0	0	0	0	0

Output Latch Levels

P1.7 = 0 P1.6 = 1 P1.5 = 1 P1.4 = 0 P1.3 = 1 P1.2 = 0 P1.1 = 0 P1.0 = 1

Port 2

Input Levels

P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0
0	0	0	0	0	0	0	0

Output Latch Levels

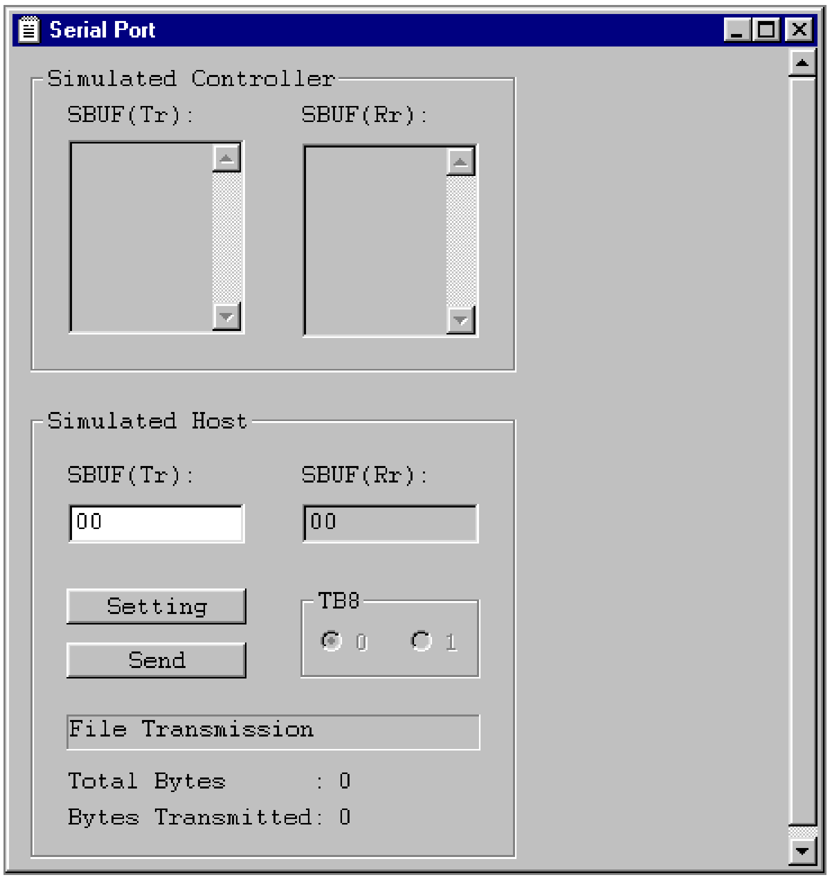
P2.7 = 0 P2.6 = 1 P2.5 = 1 P2.4 = 0 P2.3 = 1 P2.2 = 0 P2.1 = 0 P2.0 = 1

Port 3

You can notice the availability of special functions at T2EX(P1.1), T0(P3.4), T1(P3.5), T2(P1.0), few selected I/O functions like Timer/Clock Inputs, interrupt lines INT0(P3.2), INT1(P3.3) at their respective I/O pins. The drop list at these pins give these special functions. So, you can manipulate different I/O conditions, Timers/Clocks, Interrupt options in this I/O window.

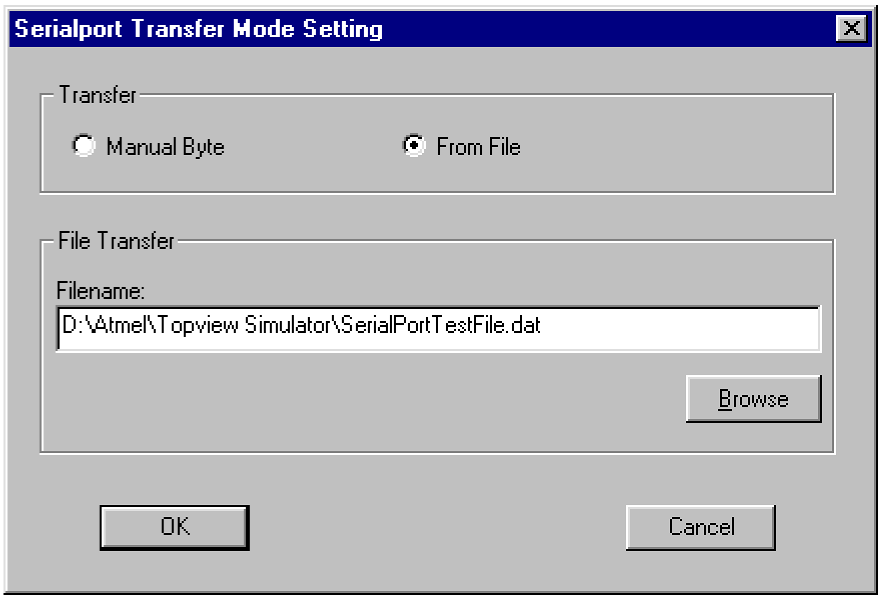
4.3 - Serial Port Window

You can enable this serial port window by clicking **Serial Port Window** from the **View** menu. The serial port window gets opened or becomes active if it is hidden in the background.



SBUF(Tr): 00
SBUF(Rr): 00
TB8: [✓ 0] 1
Total Bytes : 0
Bytes Transmitted: 0

You can also notice the presence of a simulated host in the serial port window. This one is an useful facility to check proper working of your serial port in the external environment. This simulated host may function as external computer's serial port or act as a serial port of another 8031 microcontroller in a multiprocessor communication mode.



Transfer: Manual Byte [✓ From File]

Filename: D:\Atmel\Topview Simulator\SerialPortTestFile.dat

There is a provision to send and receive test data between the microcontroller and this simulated host. Either you can manually key in test data byte by byte or send the same from a file kept in your hard disk.

For your convenience, separate buffers of 255 bytes are maintained for both transmitter and the receiver. These buffers keep 255 bytes last transmitted and received.

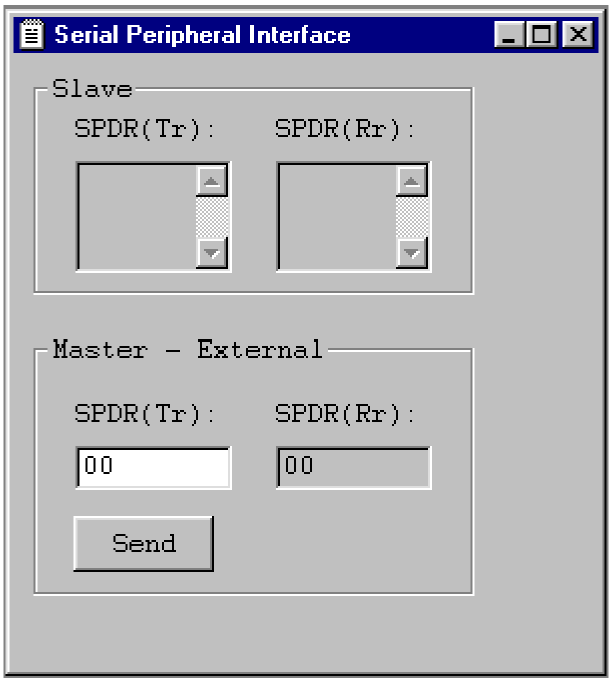
Also keep in mind that this external host serial port also operates at the same baud rate of controller's serial port. So it saves you from worrying about defining characteristics of the external serial port.

4.4 - Serial Peripheral Interface Window

This window is meant for studying or interfacing SPI port facility available in the devices, AT89S8252 and AT89S53. Just like serial port, here also you can use an external SPI port as a master to initialize communication with the target controller's SPI port.

You can start this facility by clicking, **Serial Peripheral Interface Window** from the **View** menu.

Now you should see this window coming into action.



You can observe the external SPI configured as a master. You can key in the test data in the SPI's transmitter and observe the same coming into controller's receive buffer. Just enter your data in the blank space and press the send button.

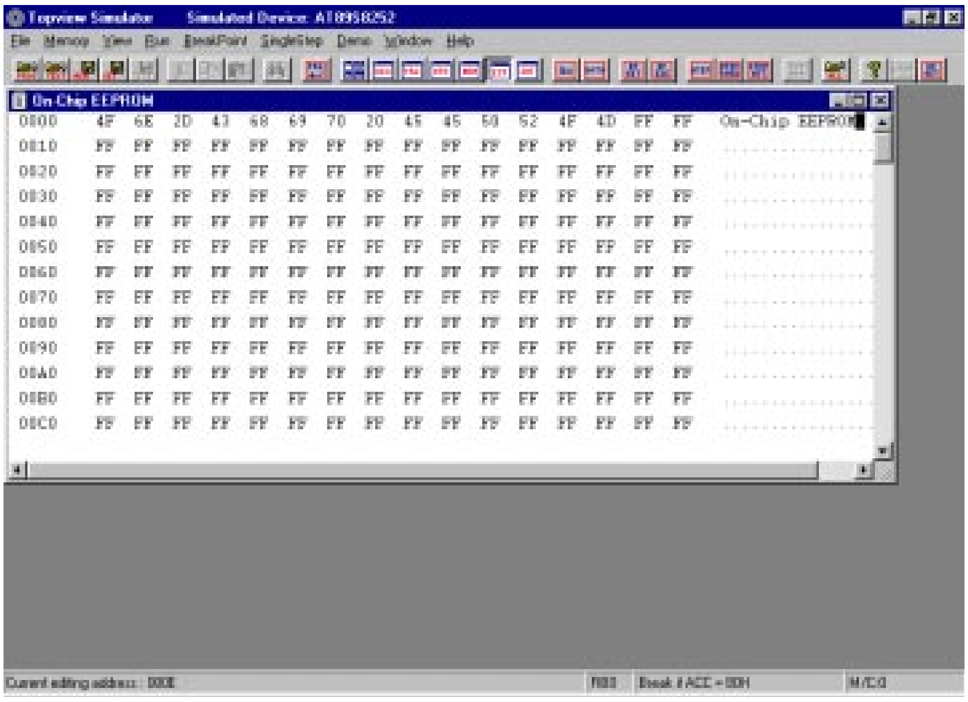
Last 255 bytes transmitted or received at the controller's SPI port are available for your verification.

4.5 - On-Chip EEPROM Window

An internal EEPROM of capacity 2KB is available in the microcontroller, AT 89S8252. You can simulate this memory space and define the memory contents. You can make your target program read from or write into this EEPROM and you can observe those changes in the respective locations in this EEPROM window.

Activate this window by clicking, **On-Chip EEPROM Window** from the **View** menu.

You should see this window coming into view indicating contents of the memory space.



```
0000   4F 6E 2D 43 68 69 70 20 45 45 50 52 4F 4D FF FF   On-Chip EEPROM
0010   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
0020   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
0030   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
0040   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
0050   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
0060   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
0070   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
0080   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
0090   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00A0   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00B0   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00C0   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
```

```
Current editing address : 000E        R00   Break if ACC = 00H        M/C:0
```

The contents of this memory area are displayed along with the address. The data are displayed in both Hex and ASCII format.

Just like other memory, you can also edit the contents of this EEPROM space.

5.1 - Introduction

Topview Simulator comes with facility to simulate a variety of external embedded building blocks apart from the microcontroller and its on-chip peripherals. As a result, you can start and complete your total applications in the software. So, simulating these external embedded building blocks will save good amount of your project development time.

You have facility to simulate IIC standard EEPROM, SPI bus EEPROM, IIC bus RTC, Point LEDs, Seven Segment LED Displays, or a range of keypads upto 4 X 8 matrix and LCD modules. So, without waiting for the target hardware, you can finish your project using Topview Simulator.

5.2 - Activating Building Blocks

Embedding these building blocks in your target design is very simple and you need not do anything specific for this.

Just complete your design as if you are using external building blocks like switches, LEDs, LCD modules, etc.

Also complete the initialization for all the selected blocks. This initialization can be done in a graphical environment. And keep all the windows of the predefined modules open.

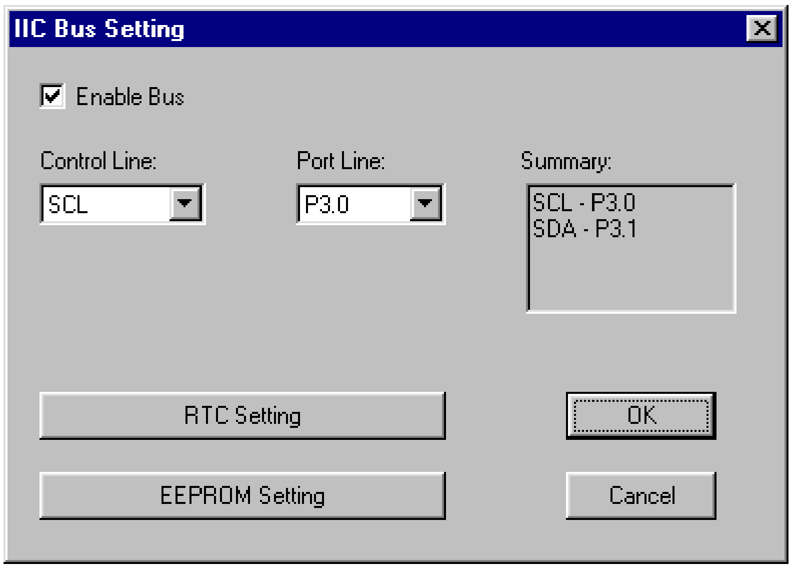
Then execute your target program in the simulator. Now you can see all the selected embedded building blocks spring into action and you can see the working of your target application in the simulator itself!

5.3 - IIC Devices

Topview simulates most common IIC devices, RTC from Philips, PCF8583 and the Atmel's EEPROM devices, AT24C01, AT24C02, AT24C04, AT24C08, AT24C16.

To start initialization, select **File → External Module Setting → IIC Bus.**

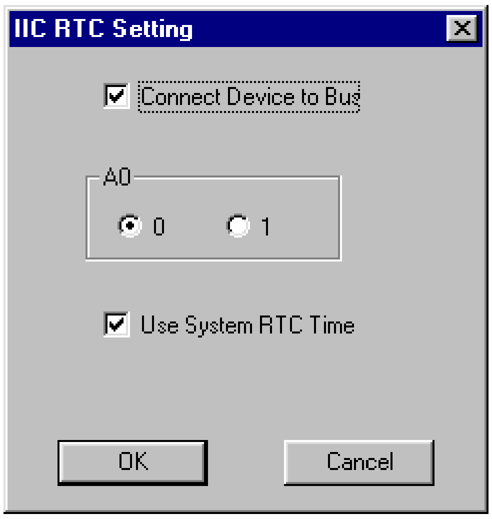
Now you should see the following dialog box.



Control Line: SCL
Port Line: P3.0
Summary:
SCL - P3.0
SDA - P3.1

First, check the **Enable Bus** to activate IIC bus. Then select the port lines for the IIC bus signals, SCL and SDA. You can notice other two buttons, RTC Setting and EEPROM Setting to start configuring both RTC and EEPROM devices.

When you click **RTC Setting**, you get a dialog box in the screen prompting you to configure the RTC device.


IIC RTC Setting

☑ Connect Device to Bus

A0: [✓ 0] 1

☑ Use System RTC Time

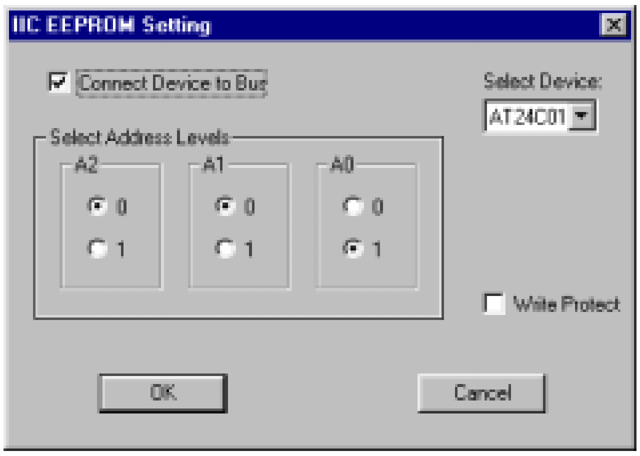
Now click the box **Connect Device to Bus** to include this device in the IIC bus.

Then set the address line A0 to either as 0 or 1 level. You can even select the PC system clock as RTC reference. It may help you in getting the RTC with current time indicating facility.

Press **OK** button to include the selected IIC device into the design.

When you need to have IIC based EEPROM, click **EEPROM Setting** in the first dialog box, **IIC Bus Setting**.

You should be greeted with another dialog box as shown here:


IIC EEPROM Setting

☑ Connect Device to Bus

Select Device: AT24C01

Select Address Levels

A2: [✓ 0] 1

A1: [✓ 0] 1

A0: 0 [✓ 1]

☐ Write Protect

You need to include this IIC EEPROM device into the bus by enabling the **Connect Device to Bus** check box.

Then select any one of the five possible EEPROM devices from the right side drop list box.

Also select the addressing for the lines A0, A1 and A2. Press **OK** button to complete initialization of the selected EEPROM devices.

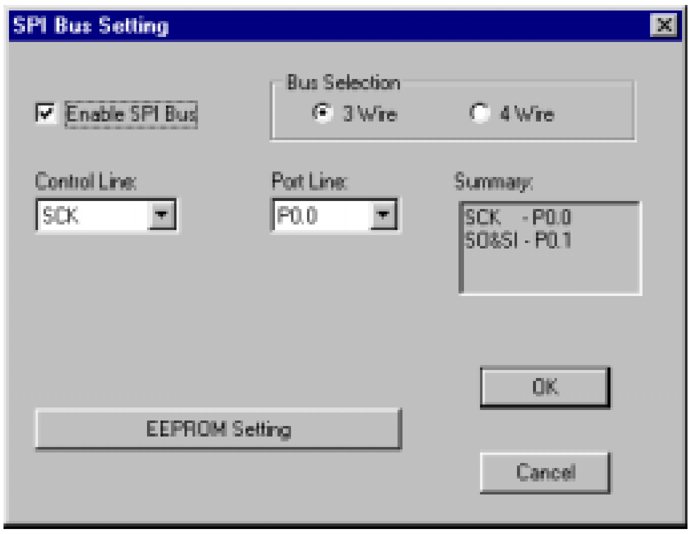
If you want to protect the memory device against unwanted writing, activate the **Write Protection** option in the window.

5.4 - SPI Devices

Topview also simulates SPI bus based EEPROM memory devices from the Atmel range, AT25C010, AT25C020 and AT25C040. These devices can also be configured for either 3 wire or 4 wire interfacing.

Activate this command by this sequence : **File →External Modules →SPI Bus**.

As usual, you get a dialog box at the center of the screen to start initialization.



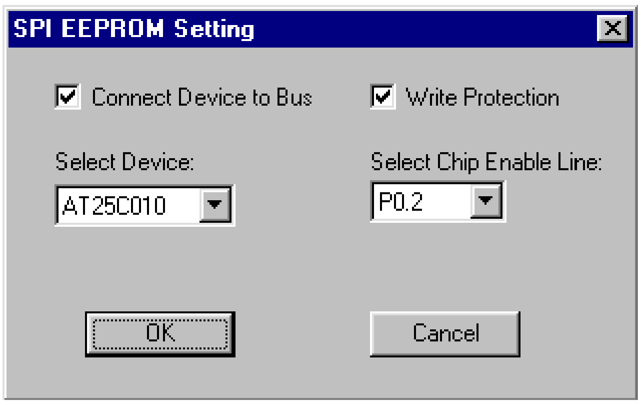
SCK - P0.0
SO&SI - P0.1

First enable SPI bus by clicking **Enable SPI Bus** check box.

Then decide whether you need 3 wire or 4 wire interfacing option.

Allot port lines for the SPI control lines, SCK, SO and SI.

Press **EEPROM Setting** button to proceed with device configuration.


SPI EEPROM Setting

☑ Connect Device to Bus ☑ Write Protection

Select Device: AT25C010

Select Chip Enable Line: P0.2

In this new dialog box, you need to check **Connect Device to Bus** box to include the EEPROM memory device into the bus.

Select the required device in the drop list, **Select Device**

Then allot the port line to chip select line of the EEPROM device.

There is an option to include write protection for the device simulation.

Press **OK** button to complete the device configuration meant for simulating device in your target design.

5.5 - LED Modules

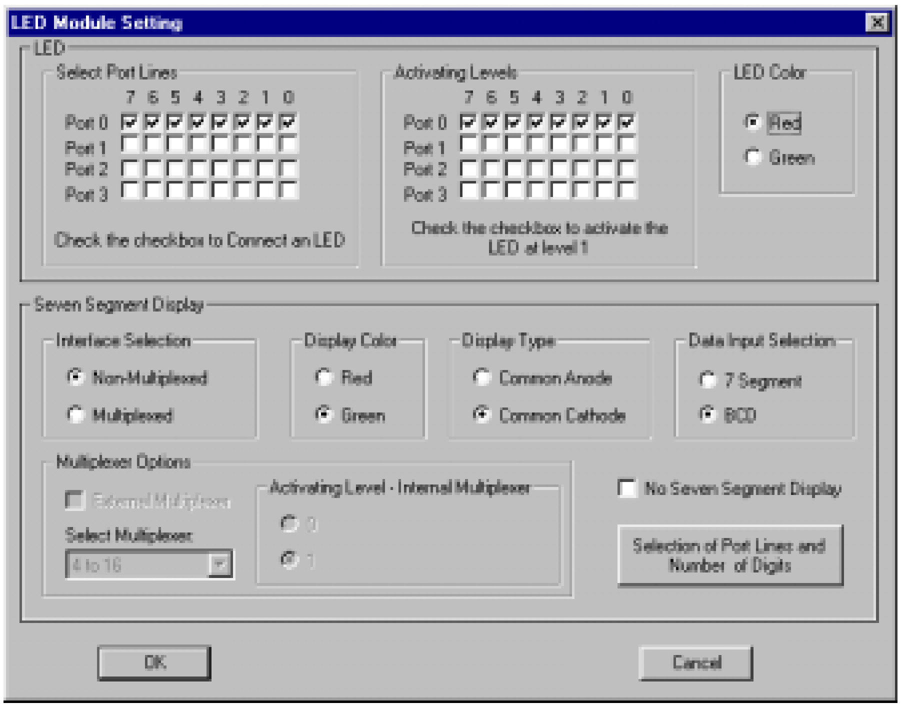
You can get a variety of options covering all the possibilities when using these LED modules. You can interface point LEDs, Seven segment displays, both multiplexed and non-multiplexed displays, choice of inputs like BCD or seven segment data and even colour of displays.

You can just embed these display modules within the target design and activate this LED module window to see the working of these modules with few mouse clicks.

In total, you can get a maximum of 32 point LEDs or 16 digits of seven segment displays. Just initialize these display modules in the given window to interface them with the selected microcontroller.

To start initialization, select **File → External Module Setting → LED**.

Now you should see a dialog box comes into action to receive your configuration data.


LED Module Setting

LED

Select Port Lines

	7	6	5	4	3	2	1	0
Port 0	✓	✓	✓	✓	✓	✓	✓	✓
Port 1	☐	☐	☐	☐	☐	☐	☐	☐
Port 2	☐	☐	☐	☐	☐	☐	☐	☐
Port 3	☐	☐	☐	☐	☐	☐	☐	☐

Check the checkbox to Connect an LED

Activating Levels

	7	6	5	4	3	2	1	0
Port 0	✓	✓	✓	✓	✓	✓	✓	✓
Port 1	☐	☐	☐	☐	☐	☐	☐	☐
Port 2	☐	☐	☐	☐	☐	☐	☐	☐
Port 3	☐	☐	☐	☐	☐	☐	☐	☐

Check the checkbox to activate the LED at level 1

LED Color: [✓ Red] Green

Seven Segment Display

Interface Selection: [✓ Non-Multiplexed] Multiplexed

Display Color: Red [✓ Green]

Display Type: Common Anode [✓ Common Cathode]

Data Input Selection: 7 Segment [✓ BCD]

Multiplexer Options

☐ External Multiplexer

Select Multiplexer: 4 to 16

Activating Level - Internal Multiplexer: 0 [✓ 1]

☐ No Seven Segment Display

FRONTLINE ELECTRONICS

5.5.1 - Point LEDs

A maximum of 32 green or red coloured round shape point LEDs are simulated assuming all the port lines are available for these point LEDs.

Now you can set the port lines meant for the LEDs, activation level and the colour of the LEDs.

Just click in the respective check boxes to connect the point LEDs at the selected port lines.

Then select the required activation level, either 1 or 0 level for those LEDs.

Last choice is the colour of these point LEDs. Note that only a single colour, either red or green is available for all the defined LEDs.

So in three simple steps, you can configure your target point LEDs at the respective port lines at right signal levels.

5.5.2 - Seven Segment LED Displays

In the seven segment LED displays, following options are available :

- Non multiplexed displays with BCD input.
- Non multiplexed displays with 7 segment input.
- Multiplexed BCD input displays with internal multiplexer.
- Multiplexed Seven Segment input displays with internal multiplexer.
- Multiplexed BCD input displays with external multiplexer.
- Multiplexed Seven Segment input displays with external multiplexer.

In this, it is assumed that you simulate the internal multiplexer using program and the port lines. When you simulate the multiplexer in software, you can get more number of displays.

You can also use an external multiplexer for this purpose.

Now let us start configuring these seven segment displays. First, select the display interface type: Multiplexed or Non Multiplexed one. You check the correct box in the interface selection area.

When the multiplexed display is required, you can select the multiplexer in drop list of **Select Multiplexer**. Available options are : 2 to 4; 3 to 8 and 4 to 16.

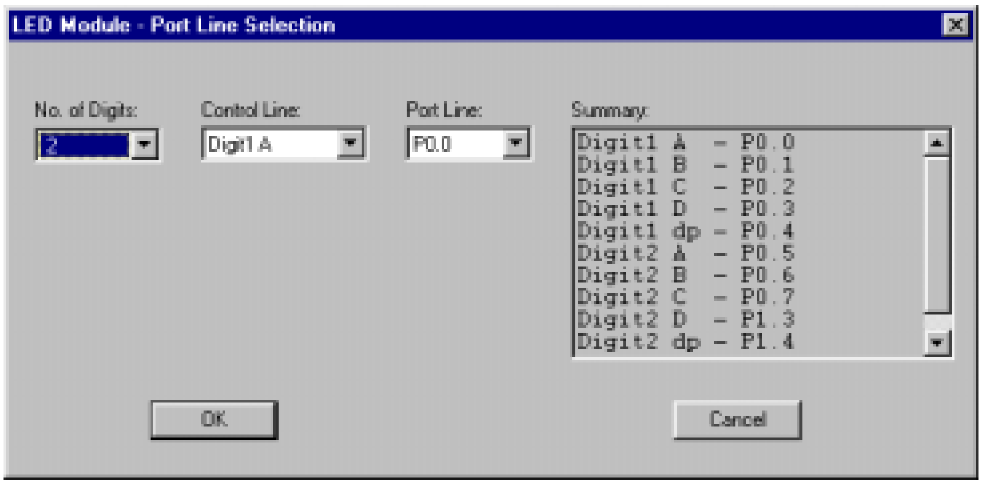
Then select the colour of the display, red or green and the display type, common anode or common cathode.

Also decide upon type of data input meant for the displays: 7 Segment or BCD.

You need to configure port lines in correct way to get these multiplexed display pattern.

To proceed with this setting, click the button **Selection of Port Lines and Number of Digits.**

Now another window pops up to enable you to complete this initialization task.



No. of Digits: 2

Control Line: Digit1 A

Port Line: P0.0

Summary:

```
Digit1 A  - P0.0
Digit1 B  - P0.1
Digit1 C  - P0.2
Digit1 D  - P0.3
Digit1 dp - P0.4
Digit2 A  - P0.5
Digit2 B  - P0.6
Digit2 C  - P0.7
Digit2 D  - P1.3
Digit2 dp - P1.4
```

Select the number of digits and then go ahead with defining the port lines meant for this display configuration. You can verify this configuration data in the summary box at the right side.

After completing all the initialization press the **OK** button to embed this Seven Segment LED module into your design. When you keep the LED module window in active state, you should see the displays working as per your design requirements during simulation.

5.6 - Keyboard Modules

Keyboard modules can be used to define, simulate, interface Momentary keys, Toggle switches and Matrix keyboards.

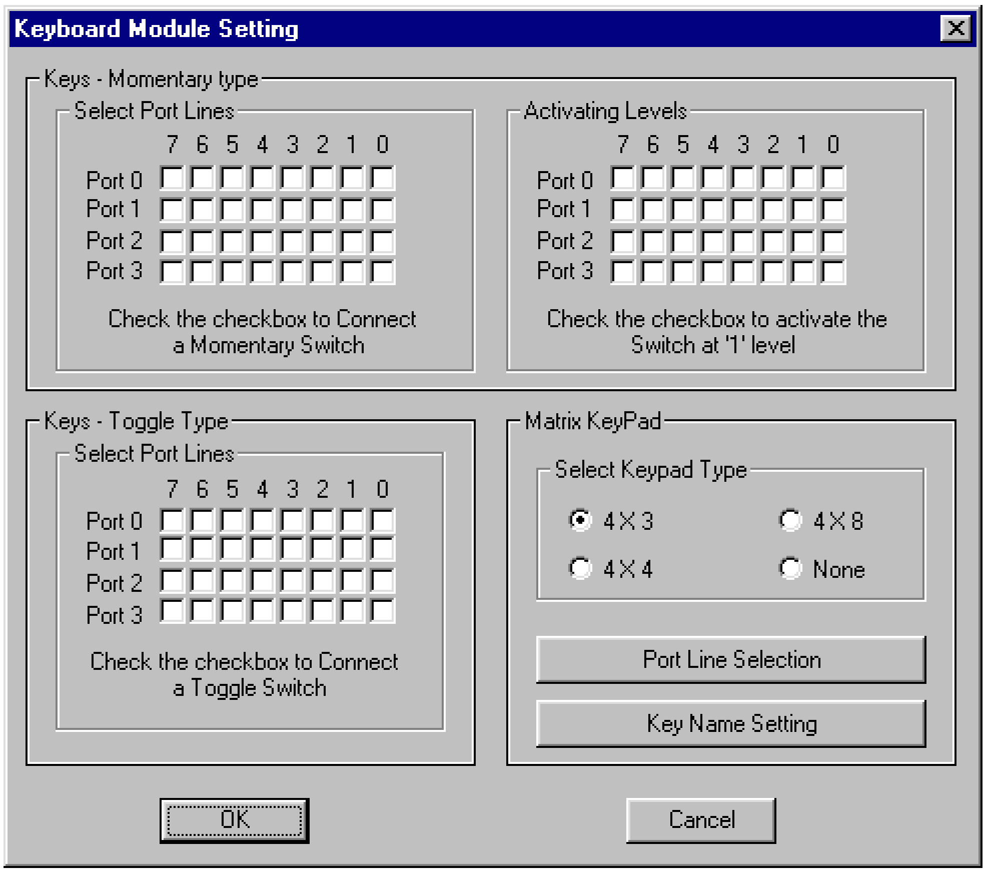
Following are possible in the keyboard module:

- A maximum of 32 momentary keys can be connected to the ports. You can even select the activation state for the key closure either as an 1 or 0 level.
- A maximum of 32 Toggle switches.
- Three types of keypads are possible.
  - 3 X 4 matrix keypad - 12 keys.
  - 4 X 4 matrix keypad - 16 keys.
  - 8 X 4 matrix keypad - 32 keys.

As you can understand, embedding these keys, switches, keypads depends upon the available port lines in your target design.

To start this configuration, select **File →External Module Setting → Keyboard**.

You can see a dialog window comes up at the screen to enable you to proceed further.



To get the momentary keys interfaced with any specific port lines, check the corresponding boxes. Also define the activation level at the right side.

Similarly you can initialize the required toggle switches at the required port lines.

When the keypad is required, select the right key matrix and then press button, **Port line Selection**.

This leads to another configuration window in which you need to set the port lines to the key matrix.

Select the proper key matrix control lines and set them to the port lines as per your design requirement. Once it is done, press the **OK** button to complete the keypad initialization.

Another interesting and useful feature is the availability of naming facility for the individual keys of the keypad.

To get into this facility, press **Key Name Setting** button in the main window.

Here also you are presented with another dialog window. In the window, select the key and define the required name in the opposite side.

Once the naming ceremony is over, then press **OK** button to complete the task.

5.7 - LCD Module

Topview Simulator supports popular Dot matrix LCD modules and by few mouse clicks you can embed them into your target design.

The following LCD modules are supported:

- 1 Line X 16 Characters.
- 2 Lines X 16 Characters.
- 4 Lines X 16 Characters.
- 4 Lines X 20 Characters.

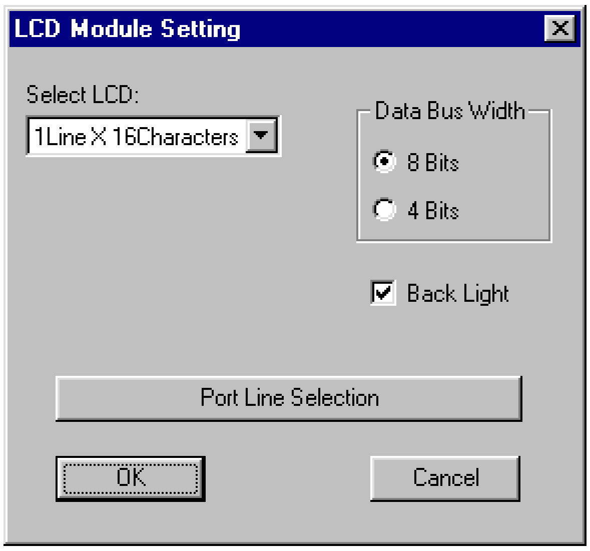
You can interface any of these modules with the microcontroller either in 8 bit or in 4 bit bus.

To complete the picture, you are given the facility to enable the back light LED of the LCD Module.

So, you can completely simulate the required LCD module in your target design.

To start, click, **File → External Module Setting → LCD.**

You are greeted with a dialog box as shown:



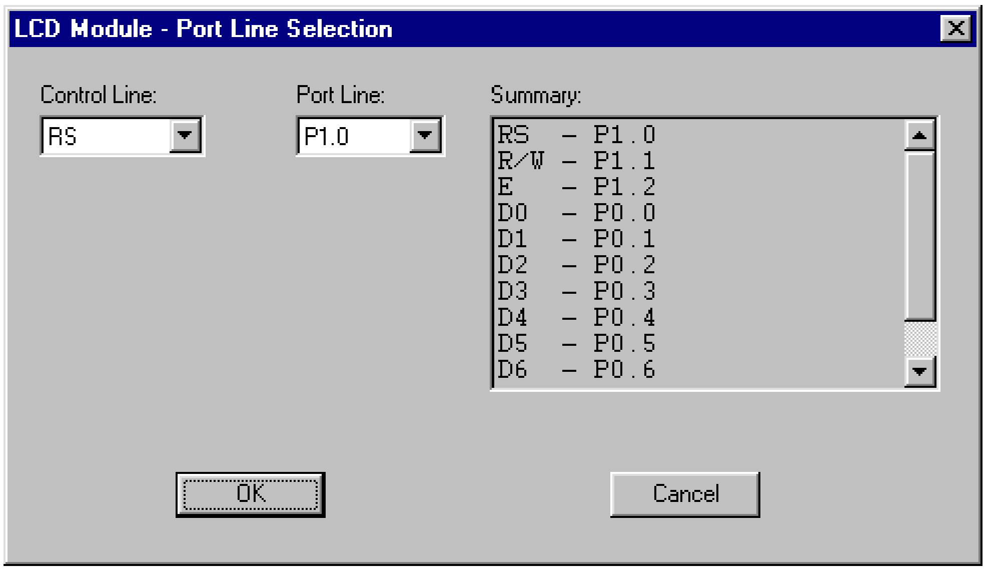
First select the LCD type from the drop list.

Then finalize the bus width you have planned to use for interfacing LCD with the microcontroller.

If it is required, enable the back light LED by checking **Back Light** box.

Now you have to define the port lines to connect with the LCD module.

Press the button, **Port Line Selection** to do this. Initialize the port lines in the new dialog box that springs into screen.



Control Line: RS

Port Line: P1.0

Summary:

```
RS  - P1.0
R/W - P1.1
E   - P1.2
D0  - P0.0
D1  - P0.1
D2  - P0.2
D3  - P0.3
D4  - P0.4
D5  - P0.5
D6  - P0.6
```

First select the LCD control line from the left side drop list and select the suitable port line at the right side.

The **Summary** box indicates the summary of your port line initialization at the right side.

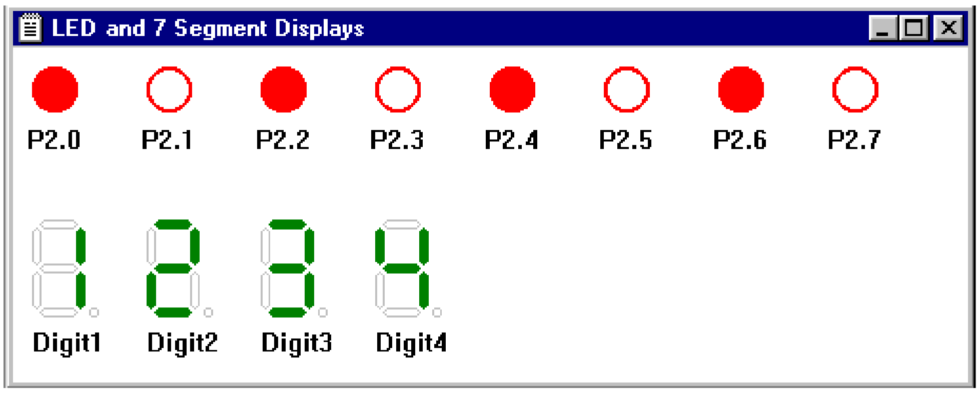
Press **OK** button to complete the port initialization. Also press **OK** button in the LCD module setting window to complete LCD interfacing task.

5.8 - Activating External Embedded Modules

After initializing external modules, you need to keep required modules active to visualize the working in your target design. You can notice all the changes happening in these modules.

5.8.1 - LED Window

In this window, you can see the point LEDs and the seven segment displays in action.



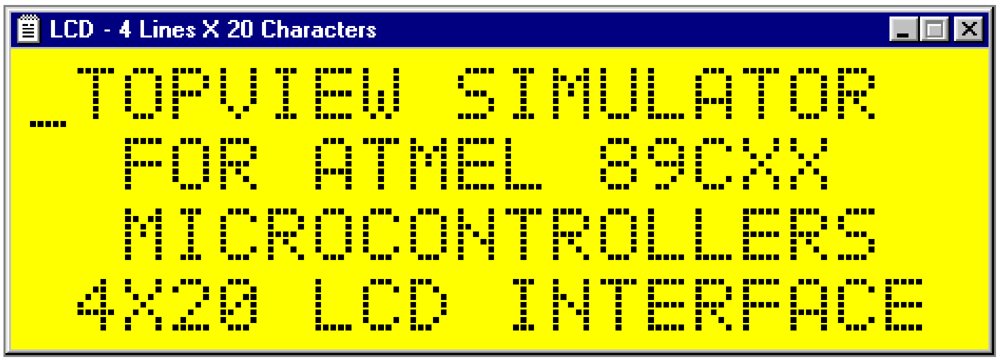
To activate this window, click **View → External Modules → LED**.

When you click on this command, the LED window gets activated and starts showing working of LEDs and Seven segment displays in your target application.

5.8.2 - LCD Window

Click **View → External Modules → LCD**.

Now you should see the LCD window coming to the front and starts showing the working of the defined LCD module in your application.

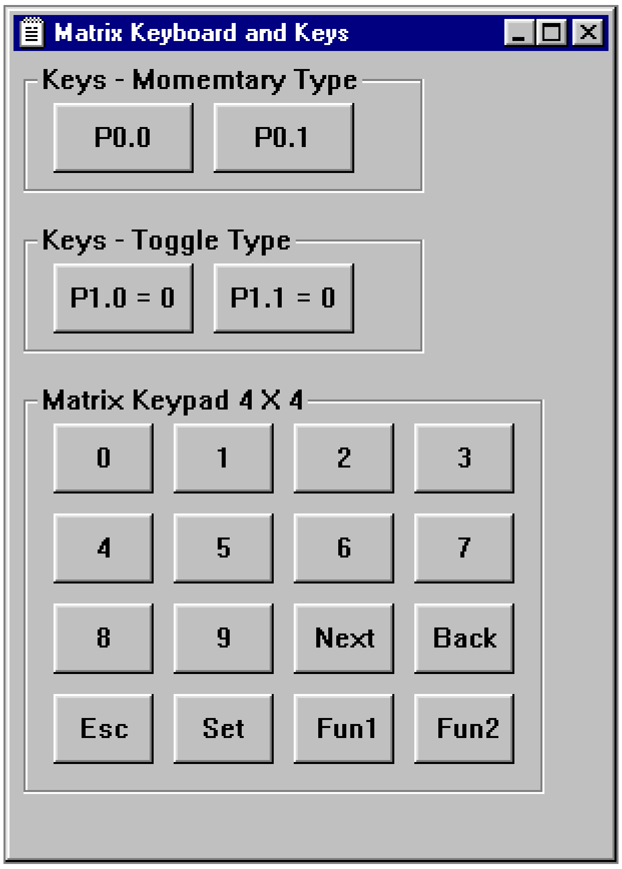

LCD - 4 Lines X 20 Characters

```
_TOPVIEW SIMULATOR
  FOR ATMEL 89CXX
  MICROCONTROLLERS
 4X20 LCD INTERFACE
```

5.8.3 - Keyboard Window

To get your defined keyboard working, click **View → External Modules → Keyboard.**

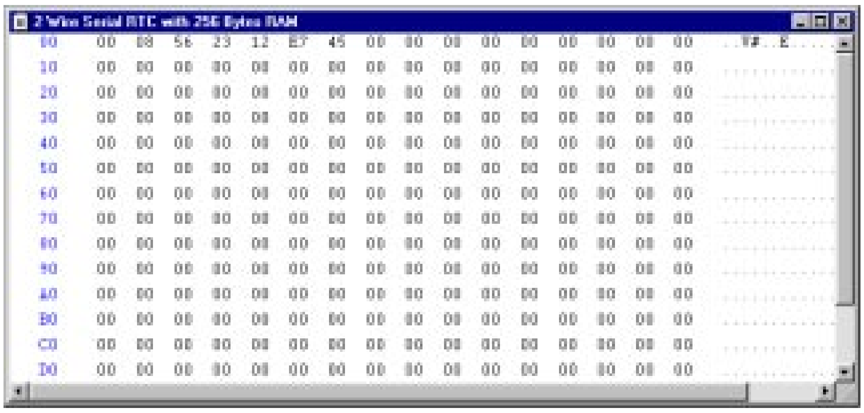
Now you can get the keyboard in the monitor enabling you to carry out simulation task. Also you should see the keys with the names you allotted during initialization.



5.8.4 - IIC RTC Window

Click **View → External Modules → IIC → RTC**.

Now the IIC RTC function is ready for the simulation. This window displays the Real Time Clock and 256 Data bytes of RAM. These data bytes are displayed in both Hex and ASCII format.

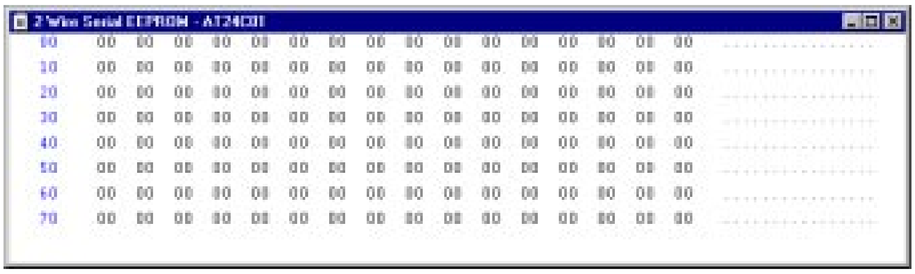


```
00    00 08 56 23 12 E7 45 00 00 00 00 00 00 00 00 00   ..V#..E.........
10    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
20    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
30    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
40    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
50    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
60    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
70    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
80    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
90    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
A0    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
B0    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
C0    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
D0    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

5.8.5 - IIC EEPROM Window

Similarly you can simulate the IIC bus based EEPROM. To activate this, press **View → External Modules → IIC → EEPROM**.

This window can simulate any of the following : AT24C01, AT24C02, AT24C04, AT24C08, AT24C16. If the write protection is enabled, the contents are displayed in green colour otherwise they are in black colour.

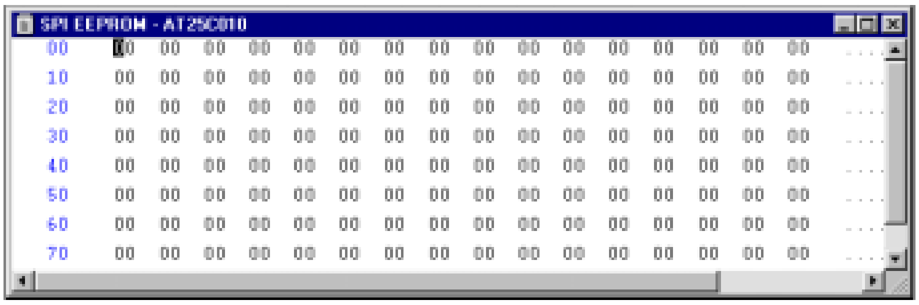


```
00    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
10    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
20    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
30    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
40    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
50    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
60    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
70    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

5.8.6 - SPI EEPROM Window

Windows meant for SPI bus based EEPROM devices, AT25C010, AT25C020 and AT25C040 can be simulated by activating the command, **View → External Modules → SPI → EEPROM.**

Now the SPI EEPROM window gets opened or becomes active if it is already in opened state.



SPI EEPROM - AT25C010

```
00   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ....
10   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ....
20   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ....
30   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ....
40   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ....
50   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ....
60   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ....
70   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ....
```

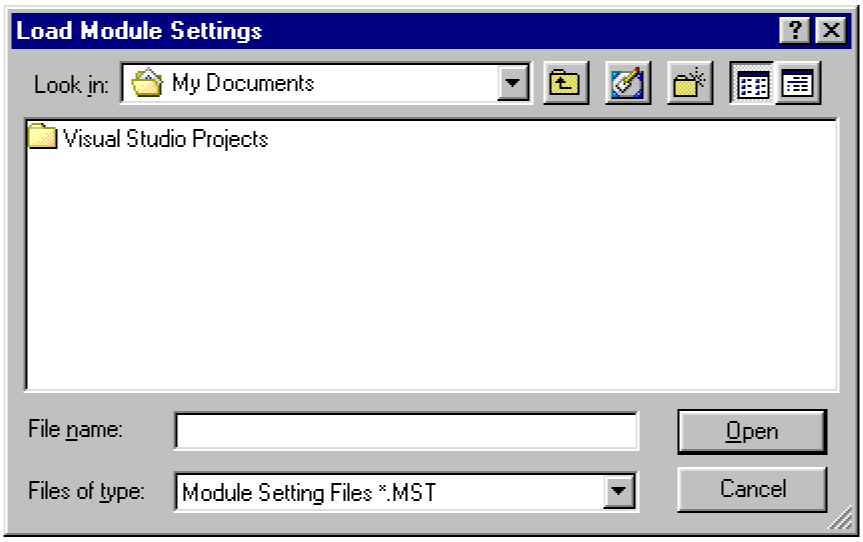
You can notice the display of contents the EEPROM devices in both Hex and ASCII formats.

If the write protection is enabled, the contents will be displayed in green colour otherwise they are displayed in black colour.

5.9 - Load and Save Module Setting

You can keep the record of various module setting information like port line allocation, peripheral initialization, external module configuration in a file. The file has **.MST** extension.

You can save all the module setting information by clicking **File → Save Module Setting**. You have to select a file name or generate a new file in the dialog box that comes up for this purpose.



If you already have a reference file in the disk, you can load that into simulator by using the **File → Load Module Setting**

6.1 - Introduction

This is one of the most useful and most desired facility required by all system designers. This is a foolproof feature to generate required code snippets in the assembly language of 8031 microcontroller. The whole task becomes very easy and simple just using the mouse and even if you are not an experienced designer, you can get the required code to embed many building blocks.

Code generation facility is available for the microcontroller's on-chip serial port and a range of external embedded building blocks.

Apart from easy and effective way of getting the code, you are assured of very compact and tight program code meant for these building blocks. And you can use this code in any place of your program flow without any additional effort.

6.2 - Internal Peripheral Functions

Topview Simulator sports the required code generating facility to use the on-chip serial communication port in all possible modes.

When generating code for the external embedded building blocks, other on-chip peripherals like I/O ports, Timers/Counters are automatically considered (in implementing these building blocks).

Getting the required target code is very simple. Just keep the program editor in open and keep the cursor at the right place in your program flow.

Then activate the relevant code generation dialog box and define your exact requirements.

Press **Generate Code** button to get the target application assembly code at the cursor defined place.

6.2.1 - Serial Port

You can get the required routines meant for serial port communication.

- Initializing serial port.
- Transmitting data using serial port.
- Receiving data at the serial port.

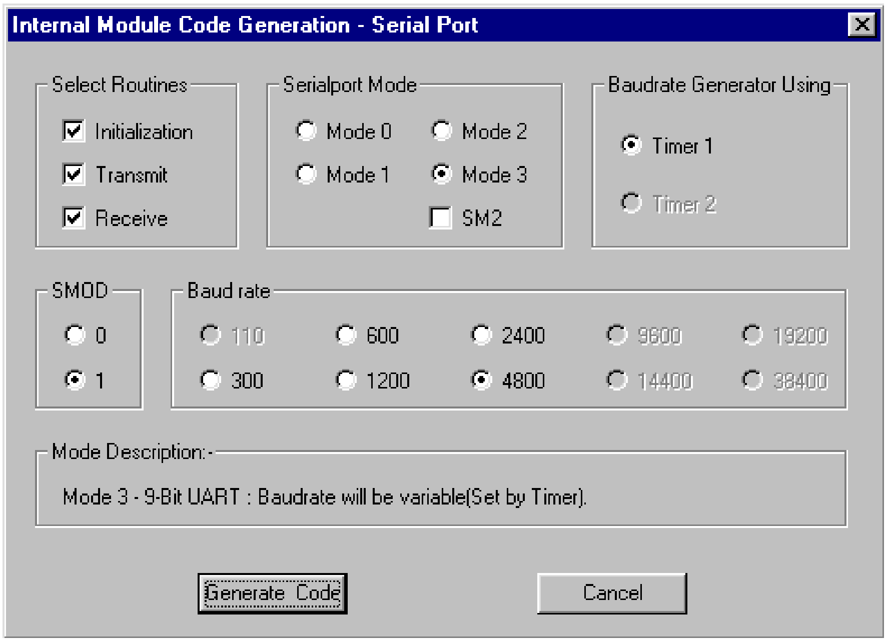
You can use this serial port in all possible modes. Also you can employ either Timer 1 or Timer 2 for generating different baud rates.

To start code generation facility, first either open an existing program file or create a new program text file by clicking **File → Load Text File**.

Place your mouse cursor in the program text editor where you want to insert the generated assembly program code.

Select and Click **Code Generation → Internal Modules → Serial Port.**

When this command is activated, you can notice the presence of a dialog box enabling you to define your requirements.



Select Routines: [✓ Initialization] [✓ Transmit] [✓ Receive]

Serialport Mode: Mode 0, Mode 1, Mode 2, [✓ Mode 3], SM2

Baudrate Generator Using: [✓ Timer 1], Timer 2

SMOD: 0, [✓ 1]

Baud rate: 110, 300, 600, 1200, 2400, [✓ 4800], 9600, 14400, 19200, 38400

Mode Description:-

Mode 3 - 9-Bit UART : Baudrate will be variable(Set by Timer).

Serial Port

This window gives all possible choices available for the serial port communications.

Just click and select the required options and press **Generate Code** button to get the required program code.

As you can observe here, the available baud rates are automatically adjusted depending upon the mode selection, Timer1/Timer2 selection and the SMOD and SM2 bits.

You are also presented with mode description facility to aid you in selecting right choice of mode.

Now click **Generate Code** button to generate the required routine at the cursor defined location.

6.3 - External Embedded Building Blocks

The simulator generates the required code for a variety of external embedded building blocks covering a wide range of applications.

- LED Display functions.
- LCD Module Selection.
- Keyboard Interfacing.
- IIC Bus.
- SPI Bus.

All these external modules can be interfaced with any I/O line of the microcontroller as you do in real design. There is absolutely no restriction in using these modules with any I/O line of the controller.

6.3.1 - LED Display Functions

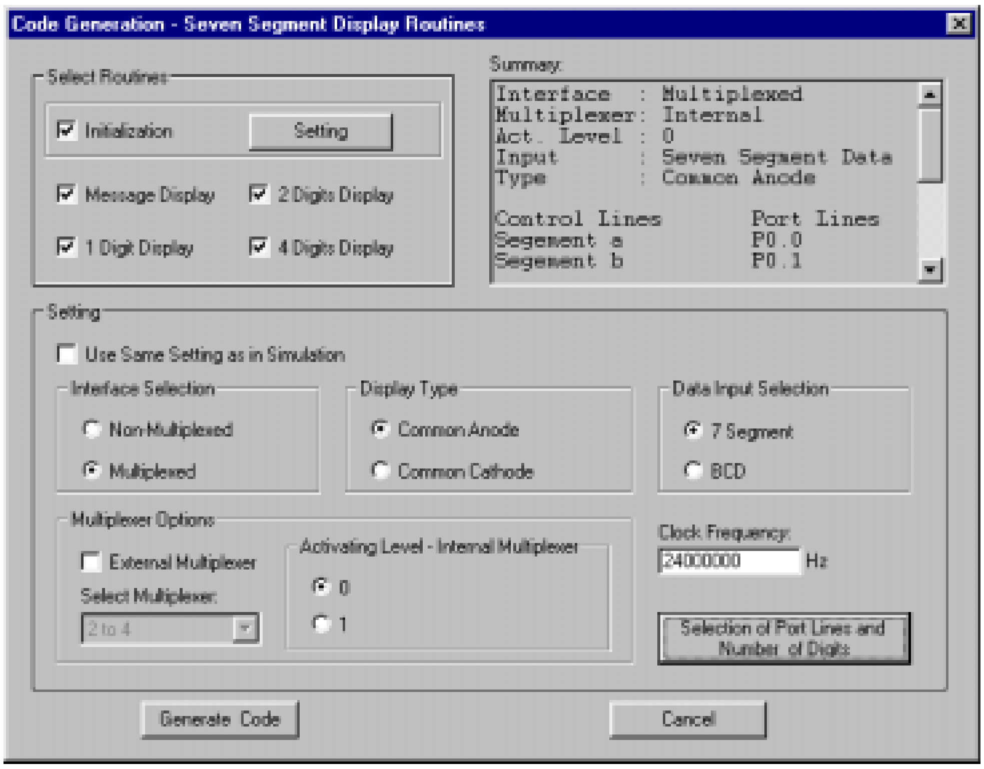
For interfacing external LED displays, many functions are available:

- Initialization.
- Message Display.
- 1 Digit Display.
- 2 Digits Display.
- 4 Digits Display.

As usual, activate program text editor and place the mouse cursor at the exact location where LED display function is required.

Select **Code Generation → External Module → LED → 7 Segment Display**.

When you execute this command, a dialog box appears on the screen to prompt you to define your selection.



Code Generation - Seven Segment Display Routines

Select Routines

[✓] Initialization Setting

[✓] Message Display [✓] 2 Digits Display

[✓] 1 Digit Display [✓] 4 Digits Display

Summary:

```
Interface  : Multiplexed
Multiplexer: Internal
Act. Level : 0
Input      : Seven Segment Data
Type       : Common Anode

Control Lines         Port Lines
Segement a            P0.0
Segement b            P0.1
```

Setting

[ ] Use Same Setting as in Simulation

Interface Selection: Non-Multiplexed [✓ Multiplexed]

Display Type: [✓ Common Anode] Common Cathode

Data Input Selection: [✓ 7 Segment] BCD

Multiplexer Options: [ ] External Multiplexer; Select Multiplexer: 2 to 4

Activating Level - Internal Multiplexer: [✓ 0] 1

Clock Frequency: 24000000 Hz

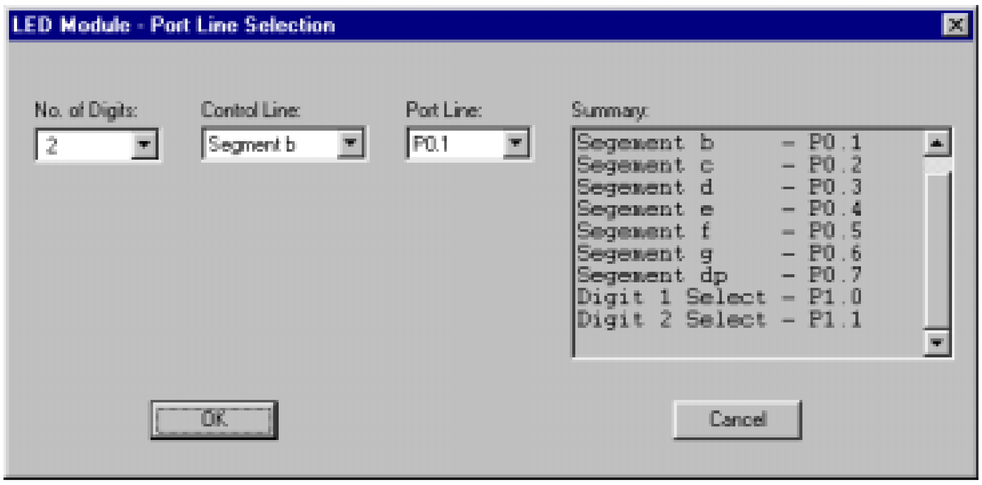
Carefully select the required display function by checking various options.

There is another provision made available in the dialog box to use the same setting as you have already defined in the simulator.

Then click the button, **Selection of Port Lines and Number of Digits** to configure the controller's I/O lines for the selected display operation.

Now another dialog box opens up to enable you to complete this configuration.

This window also provides a summary box to give you the complete initialization information.

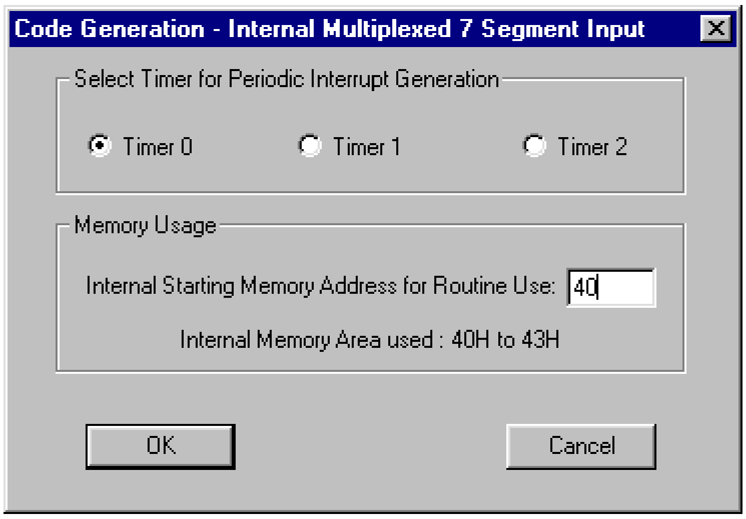


```
Segement b      - P0.1
Segement c      - P0.2
Segement d      - P0.3
Segement e      - P0.4
Segement f      - P0.5
Segement g      - P0.6
Segement dp     - P0.7
Digit 1 Select  - P1.0
Digit 2 Select  - P1.1
```

Once you complete this task, press **OK** button to proceed further.

After selecting all the required display functions, press **Setting** button to start setting task.

Now you should get a window giving you all the options that you need to consider for the display functions.


Code Generation - Internal Multiplexed 7 Segment Input

Select Timer for Periodic Interrupt Generation

[✓ Timer 0] Timer 1 Timer 2

Memory Usage

Internal Starting Memory Address for Routine Use: 40

Internal Memory Area used : 40H to 43H

In this window you can select any of the three available timers and also the memory location where temporary data will be kept.

Press **OK** to finish this setting.

When you return to the parent window, press **Generate Code** button to generate the program code at the defined place.

6.3.2 - LCD Module Selection

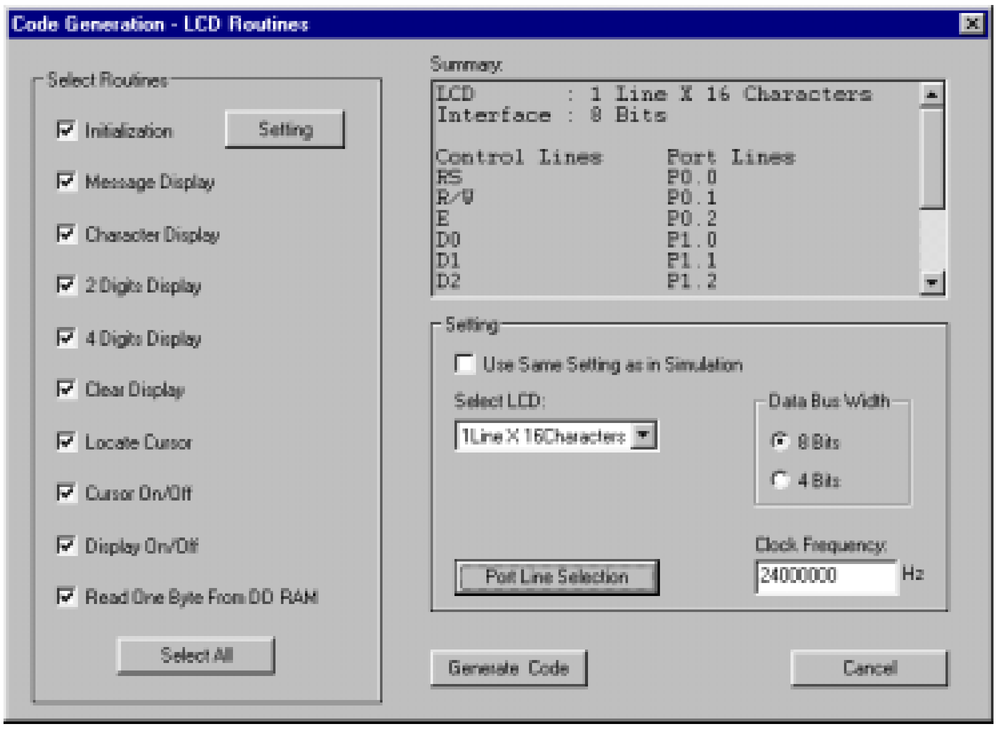
Topview supports all possible LCD interfacing options and you can exactly define your requirements. Following choices are available:

- **Display Module Types**
  - 1 Line X 16 Characters.
  - 2 Lines X 16 Characters.
  - 4 Lines X 16 Characters.
  - 4 Lines X 20 Characters.

- **Available Routines for LCM**
  - Initialization
  - Message Display.
  - Character Display.
  - 2 Digits Display.
  - 4 Digits Display.
  - Clear Display.
  - Locate Cursor.
  - Cursor On/Off.
  - Display On/Off.
  - Read one Byte form DD RAM.

Now, to start code generation for LCD modules, place the cursor in program text editor at the exact location.

Select **Code Generation → External Modules → LCD.**


Code Generation - LCD Routines

Select Routines
[✓] Initialization
[✓] Message Display
[✓] Character Display
[✓] 2 Digits Display
[✓] 4 Digits Display
[✓] Clear Display
[✓] Locate Cursor
[✓] Cursor On/Off
[✓] Display On/Off
[✓] Read One Byte From DD RAM

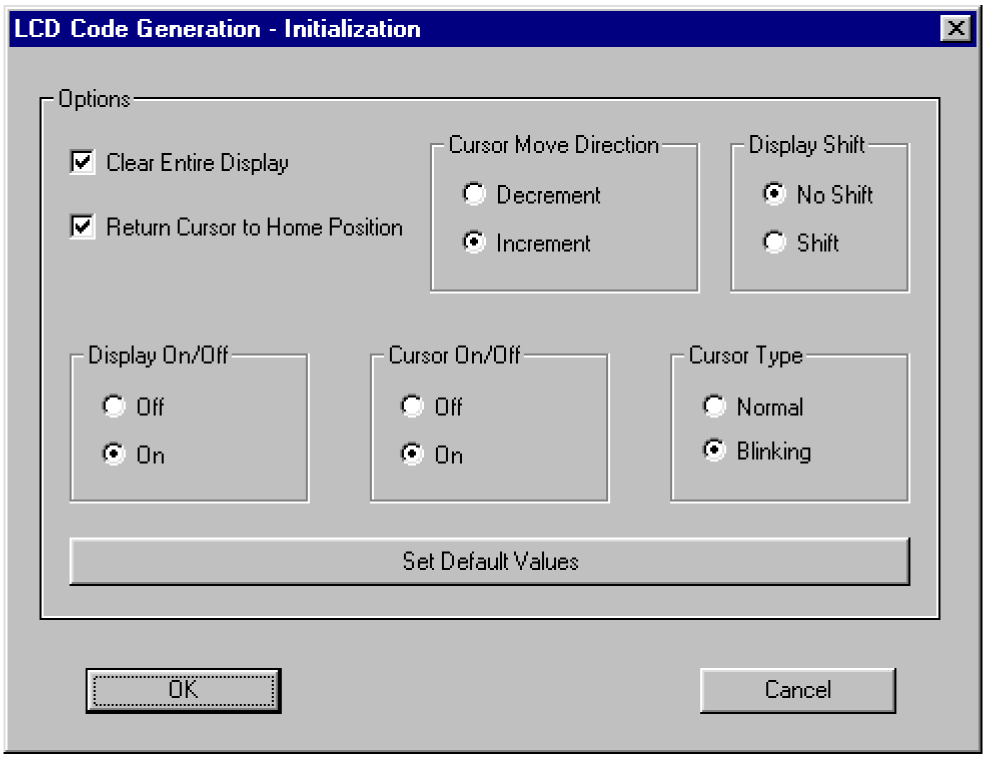
Summary:
```
LCD       : 1 Line X 16 Characters
Interface : 8 Bits

Control Lines      Port Lines
RS                 P0.0
R/W                P0.1
E                  P0.2
D0                 P1.0
D1                 P1.1
D2                 P1.2
```

Setting
[ ] Use Same Setting as in Simulation
Select LCD: 1Line X 16Characters
Data Bus Width: [✓ 8 Bits] 4 Bits
Clock Frequency: 24000000 Hz

A dialog box comes up to get your selection. You can also include all these routines by pressing **Select All** button.

You can press **Setting** button to define other display functions.

You are presented with dialog box in which you can select other secondary display functions.


LCD Code Generation - Initialization

Options

[✓] Clear Entire Display

[✓] Return Cursor to Home Position

Cursor Move Direction: Decrement [✓ Increment]

Display Shift: [✓ No Shift] Shift

Display On/Off: Off [✓ On]

Cursor On/Off: Off [✓ On]

Cursor Type: Normal [✓ Blinking]

You can even set default values by pressing the button **Set Default Values**. The default values are:

In Options Frame, Selects **Clear Entire Display**,
Return Cursor to Home Position.
In Cursor Move Direction - **Increment**,
Display Shift: **No Shift,**

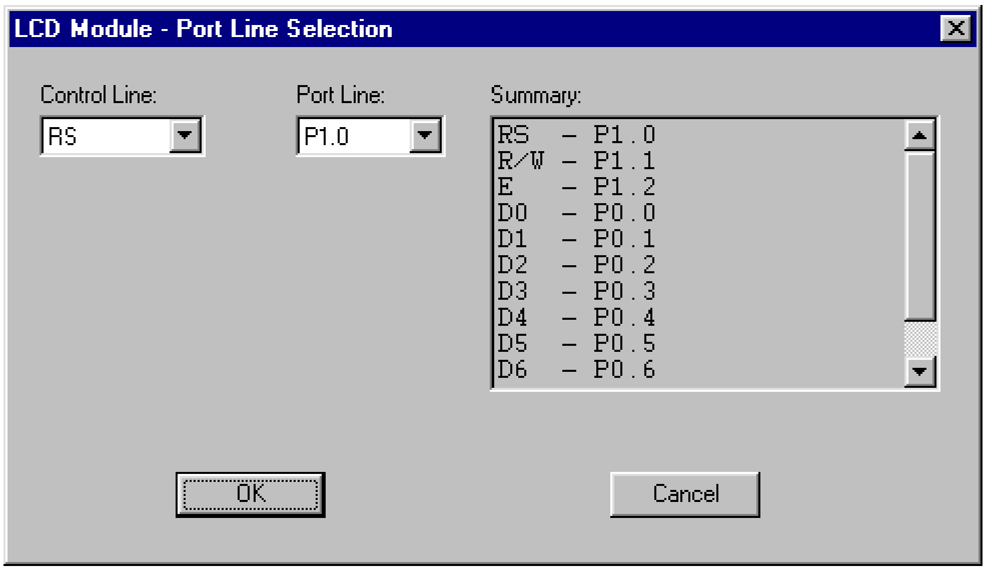
Display On/Off : **On,**
Cursor On/Off : **On** and Cursor Type : **Blinking.**

Then you can click **OK** button to return to main code generation window.

In this main window you have the option to use simulator settings as your initialization. For this click **Use Same Setting as in Simulation**.

Now you need to select the display type from the drop list and also the nature of data bus width. You can also select port lines to interface with the selected display type.

Press **Port Line Selection** button in the main window.



```
RS   - P1.0
R/W  - P1.1
E    - P1.2
D0   - P0.0
D1   - P0.1
D2   - P0.2
D3   - P0.3
D4   - P0.4
D5   - P0.5
D6   - P0.6
```

In this window select the required port lines from the drop list and you can verify your selection in the accompanied summary space.

Click **OK** to return to the main window.

Now press **Generate Code** to get the new assembly code meant for the required LCD functions.

6.3.3 - Keyboard Interfacing

You can generate the required assembly code to read the pressed key of matrix key pad. You can define following key matrices.

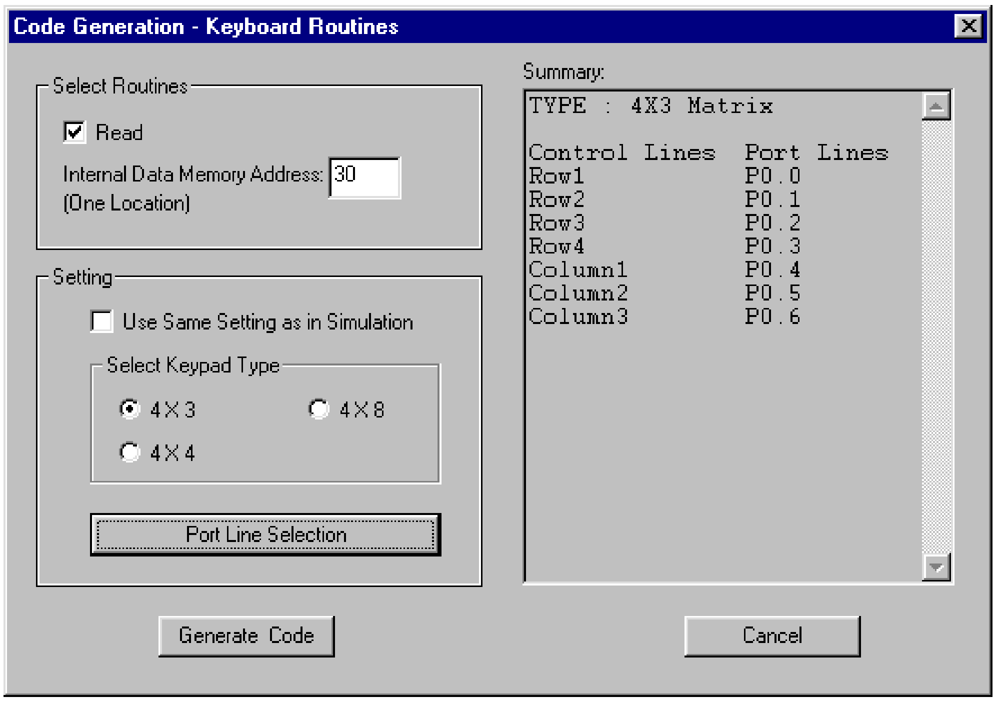
- 12 keys arranged in 4 X 3 matrix
- 16 keys arranged in 4 X 4 matrix
- 32 keys arranged in 4 X 8 matrix

You can also define the port lines that can be used to interface this matrix keypad.

You can even define the memory location where you want to get the value of the pressed key.

As usual, keep your mouse cursor at the required place in the program editor where you want to get the target assembly code. Select **Code Generation → External Modules → Matrix Keyboard.**

Now you should see a dialog window comes up to get your inputs.



Read [✓]
Internal Data Memory Address: 30
(One Location)

Use Same Setting as in Simulation [ ]

Select Keypad Type: [✓ 4X3] 4X8 4X4

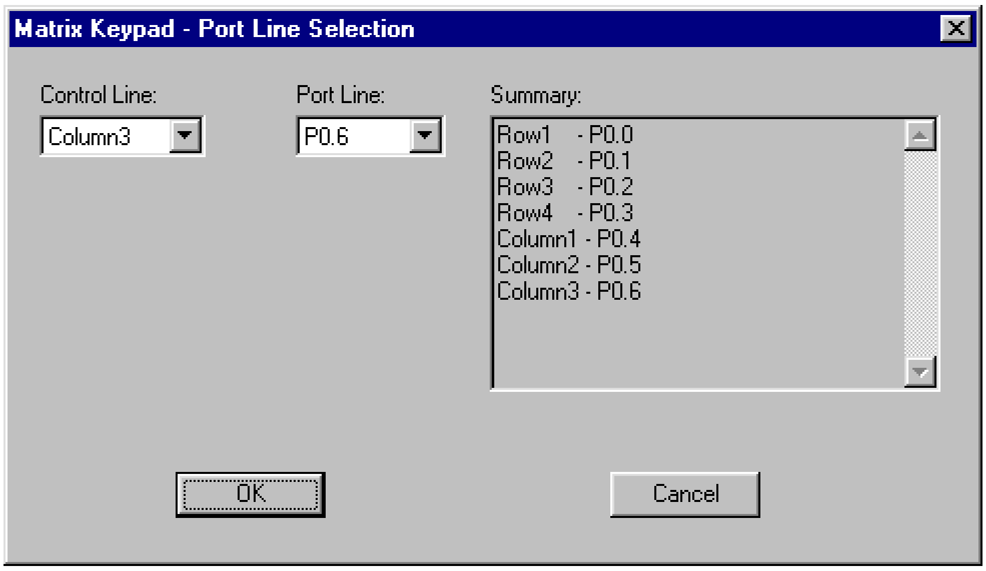
Summary:

```
TYPE : 4X3 Matrix

Control Lines   Port Lines
Row1            P0.0
Row2            P0.1
Row3            P0.2
Row4            P0.3
Column1         P0.4
Column2         P0.5
Column3         P0.6
```

When you check the box, **Use Same Setting as in Simulation,** key matrix defined already during simulation will be retained as such. If you want to redefine a new key matrix, leave this box unchecked.

Select the required key matrix and press **Port line Selection** to define the port I/O lines for that keyboard.

You can see another dialog box pops up to get your port configuration meant for that key matrix.



Control Line: Column3

Port Line: P0.6

Summary:

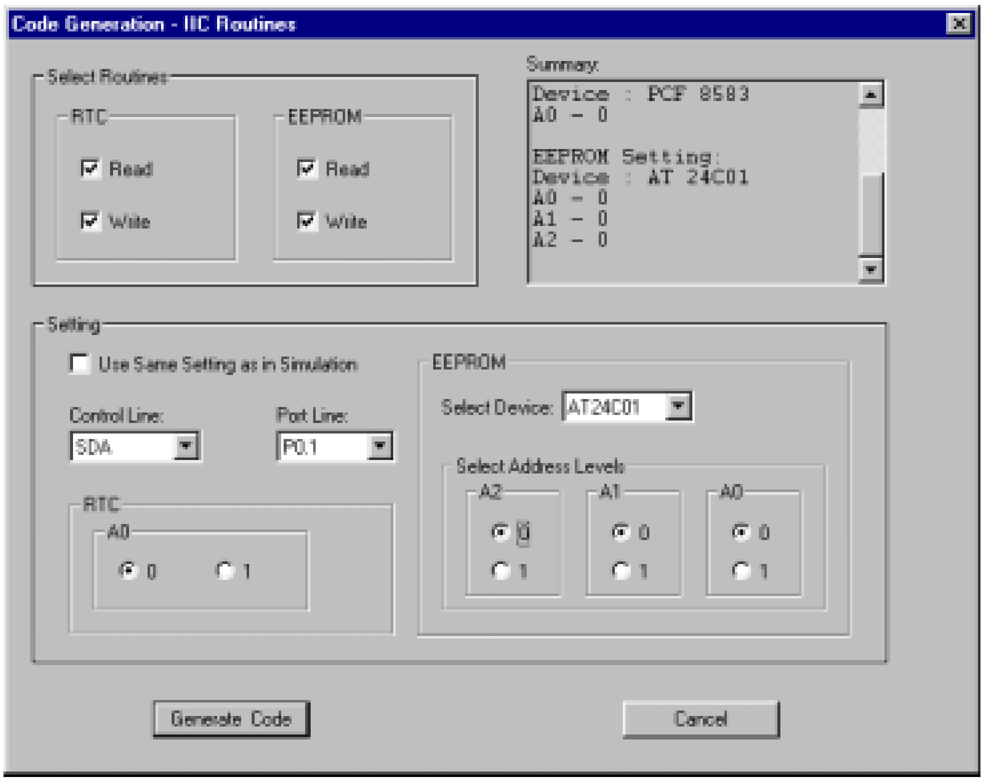
```
Row1    - P0.0
Row2    - P0.1
Row3    - P0.2
Row4    - P0.3
Column1 - P0.4
Column2 - P0.5
Column3 - P0.6
```

Select the port lines and click **OK** button. Now you return to the main window and you need to press **Generate Code** button to get the required and optimized assembly code to read the pressed key of that key matrix.

6.3.4 - IIC Bus

You can generate assembly code meant for reading from IIC device and writing to IIC device. The IIC may be a RTC or an EEPROM.

Keep the mouse cursor in the program text editor at the correct place and select **Code Generation → External Modules → IIC Bus.**



Code Generation - IIC Routines

Select Routines

RTC: [✓] Read [✓] Write

EEPROM: [✓] Read [✓] Write

Summary:

```
Device : PCF 8583
A0 - 0

EEPROM Setting:
Device : AT 24C01
A0 - 0
A1 - 0
A2 - 0
```

Setting

[ ] Use Same Setting as in Simulation

Control Line: SDA

Port Line: P0.1

RTC — A0: [✓ 0] 1

EEPROM — Select Device: AT24C01

Select Address Levels

A2: [✓ 0] 1

A1: [✓ 0] 1

A0: [✓ 0] 1

You can generate relevant code meant for following devices:

- IIC RTC, PCF 8583
- IIC EEPROM :   AT24C01
  AT24C02
  AT24C04
  AT24C08
  AT24C16

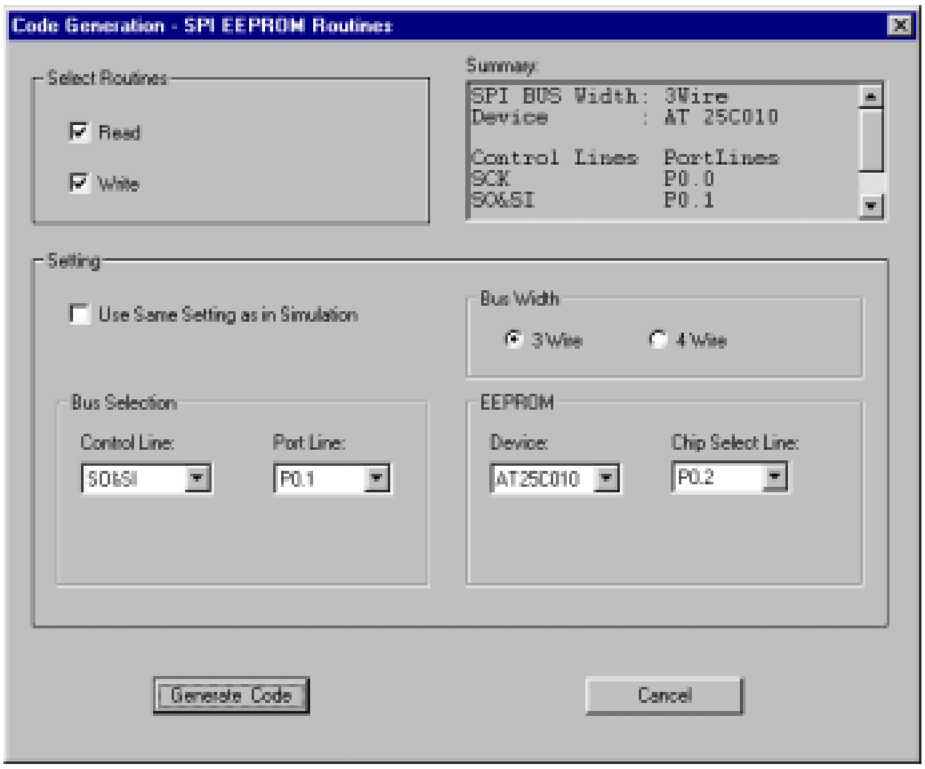
Check suitable boxes and press **Generate Code** button to get the assembly code at the required place.

6.3.5 - SPI Bus

Here you can generate assembly routines to read SPI bus EEPROM device and also to write into same device.

Keeping the cursor at the exact place, select **Code Generation → External Modules → SPI EEPROM.**

You can define your selection in the dialog window that appears in the front.



Code Generation - SPI EEPROM Routines

Select Routines
☑ Read
☑ Write

Summary:
```
SPI BUS Width: 3Wire
Device       : AT 25C010

Control Lines  PortLines
SCK            P0.0
SO&SI          P0.1
```

Setting
☐ Use Same Setting as in Simulation

Bus Width: (● 3 Wire) ○ 4 Wire

Bus Selection
Control Line: SO&SI
Port Line: P0.1

EEPROM
Device: AT25C010
Chip Select Line: P0.2

A press of **Generate Code** button at the end of your selection pastes the required code at the place where you kept the mouse cursor.

7.1 - Introduction

Topview Simulator gives you facility to develop your programs right from the scratch. Simulator’s built-in text editor takes care of program entry operations. You can simply key in your target code line by line. You can also download any input program from the disk.

Then you can make use of any external assembler software to assemble the keyed in program code lines.

7.2 - Developing Target Program Code

You can call a third party external assembler package to assemble your input program lines. The simulator captures the output file coming out of the assembler and displays the same in a separate window for your convenience. You can activate this only when using **NormalView.**

When you get assembly errors, you can keep both Text Editor Window and the Assembler output window side by side to analyze the output file. It is a convenient feature helping you in debugging process.

We have tested the simulator with freeware cross assembler supplied by the Atmel, ASM51. The assembler is made available in the accompanied CD ROM.

Actually this ASM51 is a DOS program. But when you call this assembler through simulator, you need not open DOS window separately. Assembling and data capturing tasks are handled by the simulator itself. You really don’t know that everything is happening in the DOS environment. Then you can even load the assembled program straight into the simulator memory.

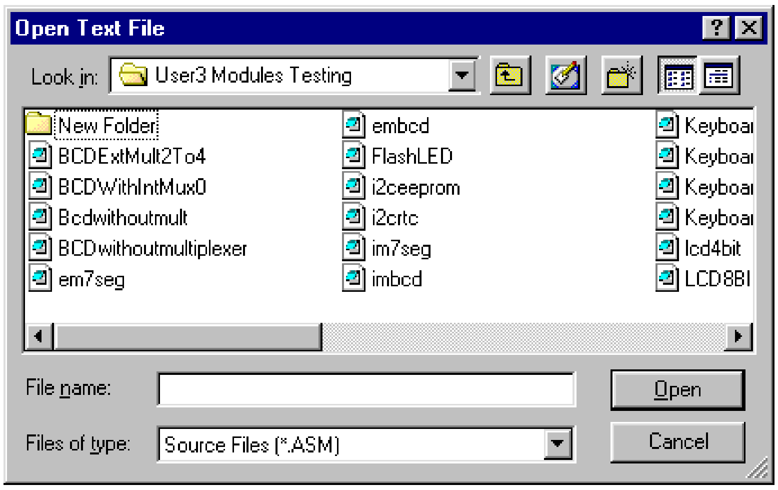
You can configure the whole process of developing the program and then loading the same into simulator memory in a single step.

This is an important and time saving feature that helps you save time during repeated debugging operations.

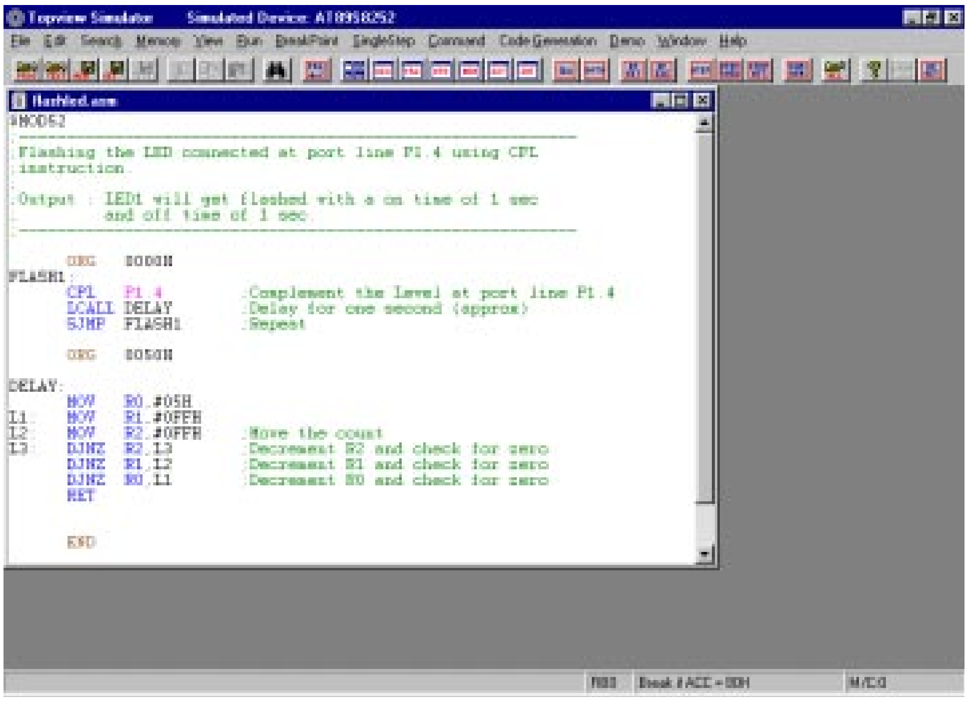
Developing Target Program Code

You should start the program entry by opening the Text Editor. Click **Load Text File** in the **File** menu for this.

A file open dialog box will appear on the screen to prompt you to select an existing file (available in the disk) or enter a new file name and then press **Open** button.



Now the Text Editor comes alive as shown here:


Topview Simulator
Simulated Device: AT89S8252
File Edit Search Memory View Run BreakPoint SingleStep Command Code Generation Demo Window Help
flashled.asm

```
$MOD52
;-------------------------------------------------------------
;Flashing the LED connected at port line P1.4 using CPL
;instruction.
;
;Output  :  LED1 will get flashed with a on time of 1 sec
;           and off time of 1 sec
;-------------------------------------------------------------

        ORG     0000H
FLASH1:
        CPL     P1.4            ;Complement the level at port line P1.4
        LCALL   DELAY           ;Delay for one second (approx)
        SJMP    FLASH1          ;Repeat

        ORG     0050H

DELAY:
        MOV     R0,#05H
L1:     MOV     R1,#0FFH
L2:     MOV     R2,#0FFH        ;Move the count
L3:     DJNZ    R2,L3           ;Decrement R2 and check for zero
        DJNZ    R1,L2           ;Decrement R1 and check for zero
        DJNZ    R0,L1           ;Decrement R0 and check for zero
        RET


        END
```

You can store the contents of the current editor into the disk using **Save Text File** command of **File** menu. From keyboard, use **Ctrl+S** to activate this command.

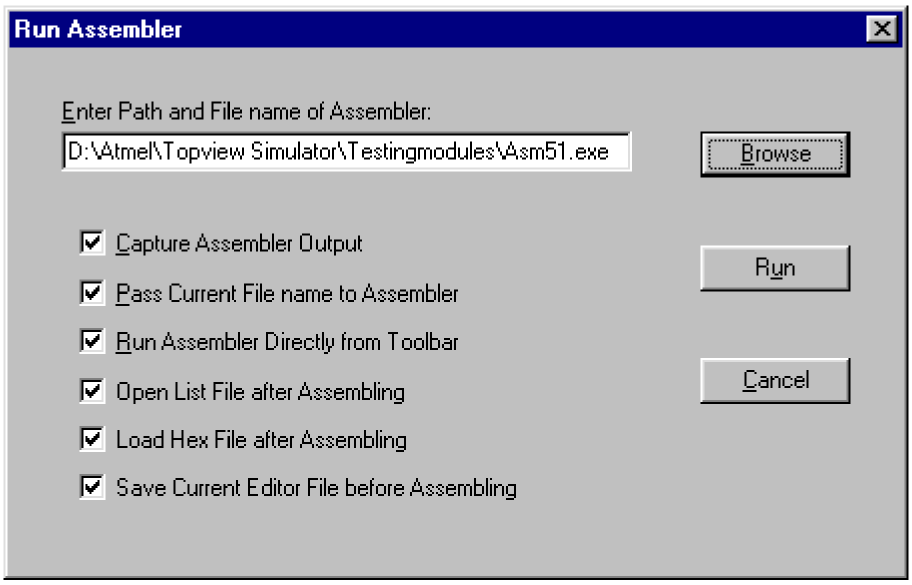
Suppose, if you want to store these contents in a new name, select **Save Text File as** command from **File** menu.

Then another window pops up to get your file name. Either create a new file or select an existing file and then press **Save** button.

You can enter the program code line by line. The editor supports colour syntax facility. Because of this, you can visualize different parts of the program, address, mnemonic, data and comments in different colours.

At this stage, you have to configure the activation of many tasks associated with the external assembler.

Now you select **Run Assembler** from **Command** menu and a dialog box comes up inviting you to select different options.



Enter Path and File name of Assembler: D:\Atmel\Topview Simulator\Testingmodules\Asm51.exe

- ☑ Capture Assembler Output
- ☑ Pass Current File name to Assembler
- ☑ Run Assembler Directly from Toolbar
- ☑ Open List File after Assembling
- ☑ Load Hex File after Assembling
- ☑ Save Current Editor File before Assembling

If you want to capture the assembler output, check the **Capture Assembler Output** box.

To pass the current text editor file name to assembler as command line parameter, check the **Pass Current File name to Assembler** box.

Check the **Run Assembler Directly from Toolbar** box to run this command from the toolbar without opening this dialog box every time.

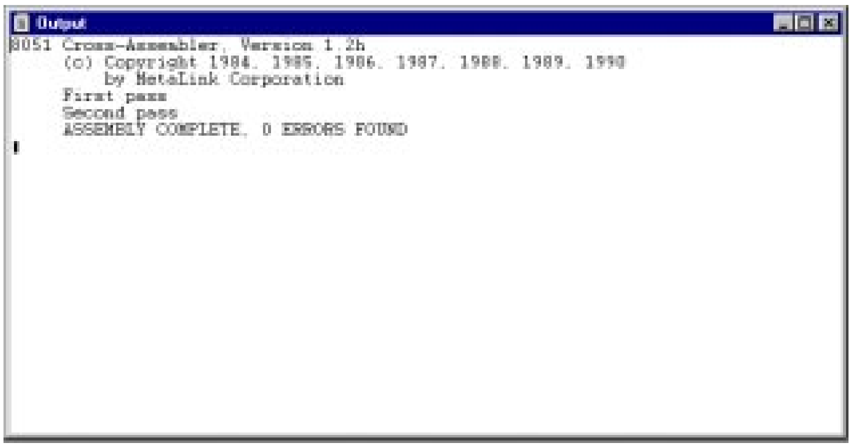
To open the program list file coming out of the assembler, check the **Open List File after Assembling** box.

To load the assembled program into the simulator memory, check the box **Load Hex File after Assembling.**

To save the current editor file before passing it to the assembler, check the **Save Current Editor File before Assembling** box.

Apart from selecting all the above mentioned, you have to define the path of the external assembler using the **Browse** button.

Now click **Run** button to run the assembler. When this command is activated, if the capture assembler output is already selected, then the assembler output file is captured and displayed as shown below in a separate window.



```
8051 Cross-Assembler, Version 1.2h
     (c) Copyright 1984, 1985, 1986, 1987, 1988, 1989, 1990
         by MetaLink Corporation
     First pass
     Second pass
     ASSEMBLY COMPLETE, 0 ERRORS FOUND
```

8.1 - Introduction

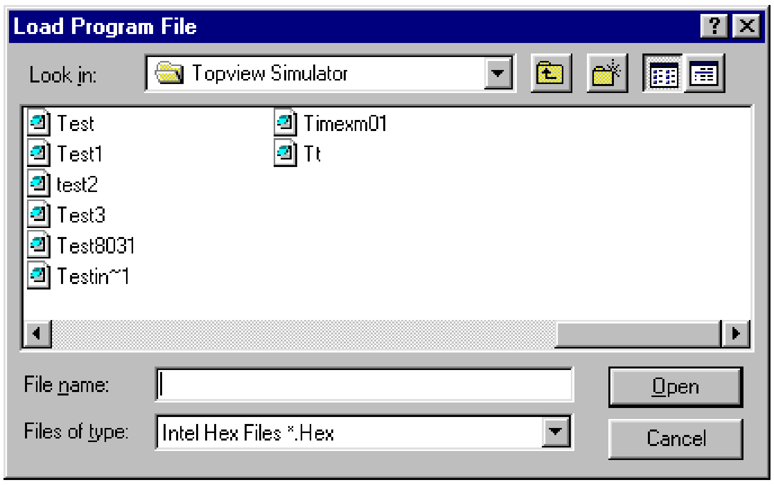
This chapter gives you an idea about loading a file from the disk into the program memory of the simulator.

You can import both Hex and Binary files into the simulator.

8.2 - Loading a Program from Disk to the Simulator Memory

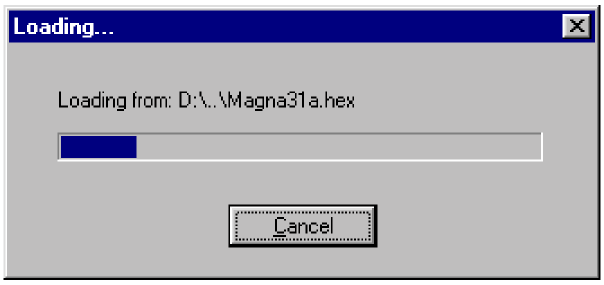
Activate this command by clicking over the **Load Program** in the **File** menu.

Now you should see a dialog box coming up asking you to select the target file or enter a file name.



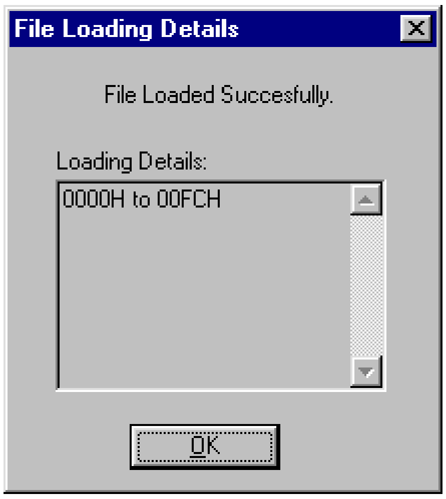
Select the file and click over **Open** button.

Another small window comes up indicating program loading using a progressive display.



Loading a Program from Disk to the Simulator Memory

At the end of the command execution, a summary of addresses where the program is loaded is displayed as shown here.


File Loading Details

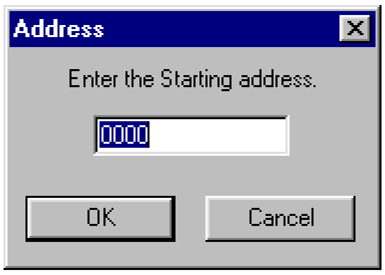
File Loaded Succesfully.

Loading Details:

0000H to 00FCH

Now click **OK** button to finish the load program command.

For a binary file, a small dialog box may appear to get the starting address.

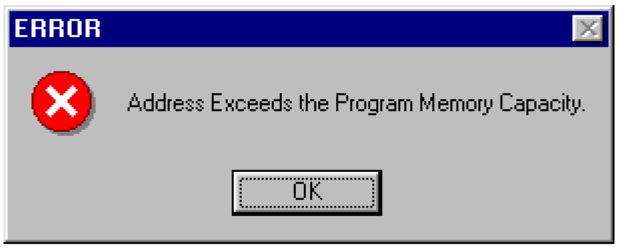

Address

Enter the Starting address.

0000

Key in the starting address and then click **OK** button or press **Enter** key to start loading. The status of program loading is displayed in a separate message box.

If the entered address exceeds the available program memory capacity, the error condition is indicated.


ERROR

Address Exceeds the Program Memory Capacity.

9.1 - Introduction

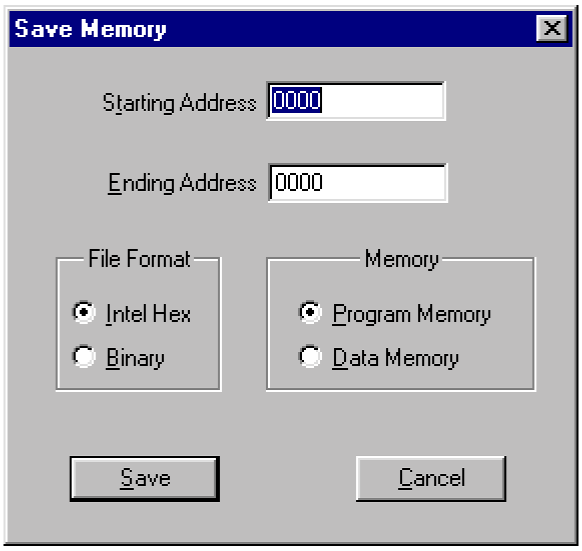
Here you will know about the way to store the memory contents of the simulator in the disk.

You can save this memory block in both Hex and Binary formats.

9.2 - Storing the Memory Contents to Disk

Activate this save command by clicking **Save Memory** in **File** menu. You can see a dialog box coming up in the screen asking the following information.

- Starting Address.
- Ending Address.
- File Format (Either Intel Hex or Binary)
- Memory (Either Program or External Data Memory)


Save Memory

Starting Address: 0000

Ending Address: 0000

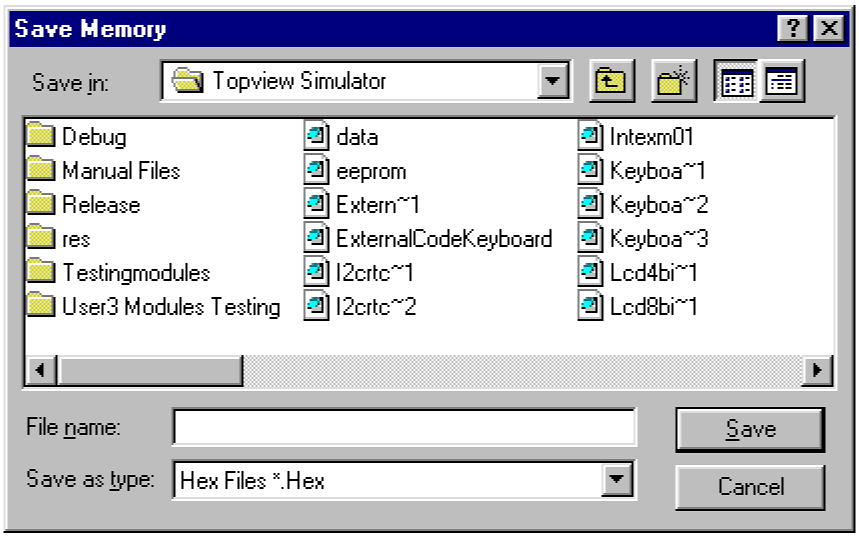
File Format: [✓ Intel Hex] Binary

Memory: [✓ Program Memory] Data Memory

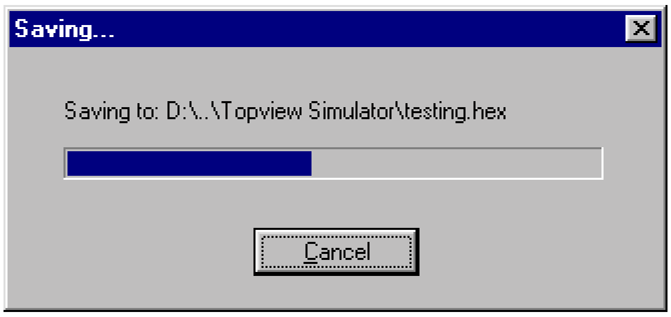
Key in and select suitable format and press **Save** button to proceed further.

Now another window pops up to get your file name. Either create a new file or select an existing file and then press **Save** button.

Storing the Memory Contents to Disk

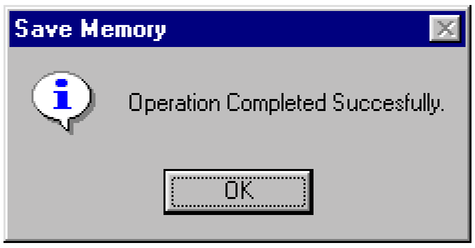


Then a small window comes alive giving a progressive display of save operation and the completion of the operation is indicated by another window.



Saving to: D:\..\Topview Simulator\testing.hex

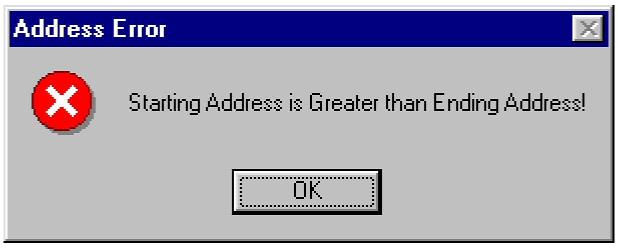
At the end, the completion of the operation will be indicated by a message as shown here:



Operation Completed Succesfully.

Error Message:

If you key in a starting address greater than the ending address, then **Address Error** message is displayed.



Starting Address is Greater than Ending Address!

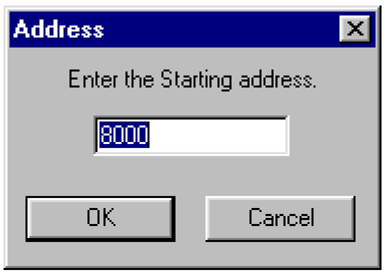
10.1 - Introduction

You can start entering your target assembly language programs in two ways. Either you can call the external assembler once you entered the raw code in the Text Editor. In this case, your complete target code is assembled as a whole by the external assembler or you can use single line assembler facility of the Program Window.

This chapter gives more information on how to use this single line assembler while keying in your application code.

10.2 - Using Single Line Assembler

Activate the command, by clicking **Enter Program** in the **Memory** menu. Then you should see a small dialog box coming up in the screen asking for the starting address.

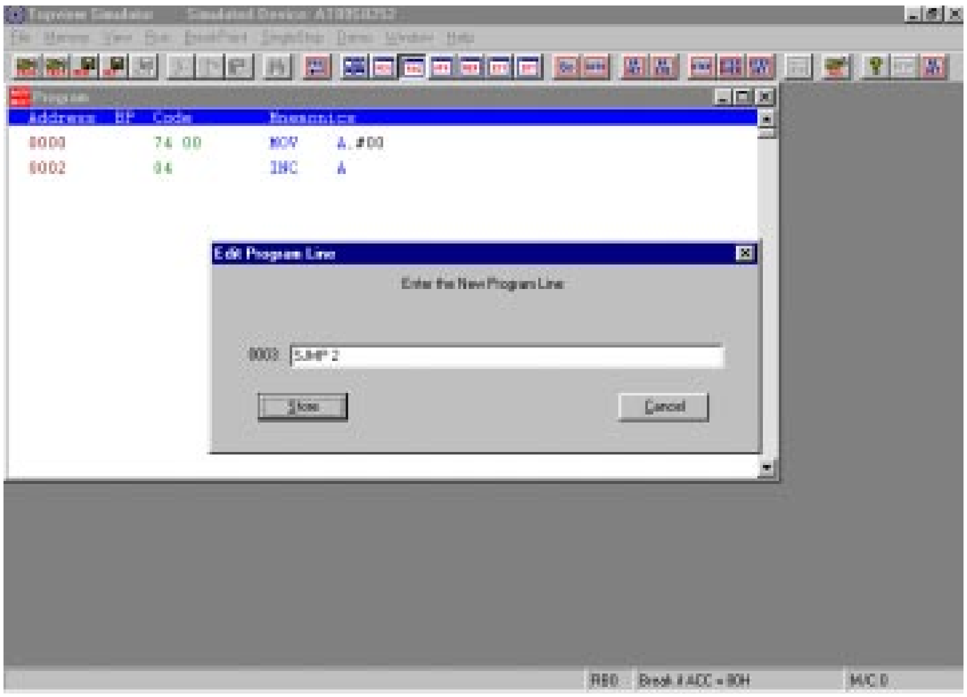


Enter the starting address of the program and press **Enter** key or click **OK** button.

Now the blank Program window gets opened.

Another dialog box will appear at the centre of the screen for keying in program instructions as shown.

Using Single Line Assembler



```
Address  BP  Code    Mnemonics
8000         74 00   MOV   A,#00
8002         04      INC   A
```

Edit Program Line

Enter the New Program Line

8003 SJMP 2

Start entering your target program line by line and keep on pressing **Enter** key or click over **Store** button to store program in the memory.

At the end of the program entry, press **Esc** key or click **Cancel** to complete this operation.

Note: This single line assembler doesn't support any label assignment.

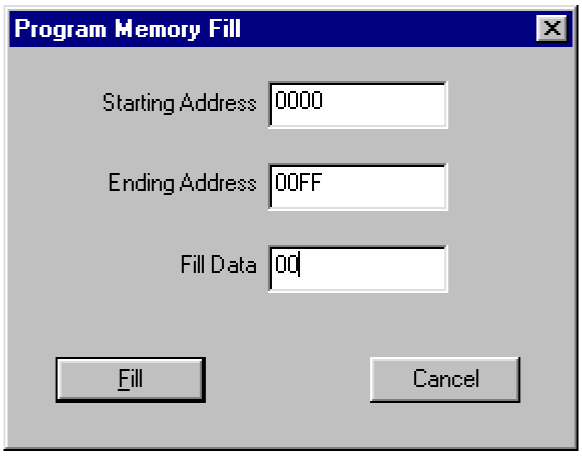
11.1 - Introduction

This chapter gives more information on filling a fixed data in a block of memory spaces in both internal/external data/program memory areas. Separate commands are available to fill both internal and external memories. Filling operation is same for both memory areas.

11.2 - Filling a Fixed Data in Internal/External Program Memory

Activate this command by selecting **Memory → Program Memory → Fill.**

When this command is activated, a dialog box will appear to get the starting address, ending address and the data meant for the fill operation.

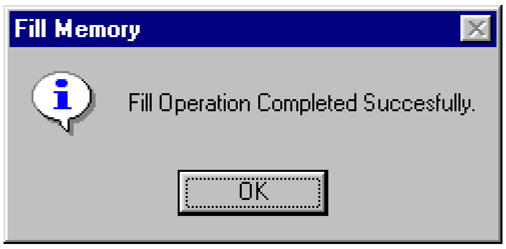

Program Memory Fill

Starting Address 0000

Ending Address 00FF

Fill Data 00

Key in the required information and press fill button to start the operation. The completion of the operation is indicated by this message.


Fill Memory

Fill Operation Completed Succesfully.

Similarly you can fill a fixed data in data memory areas.

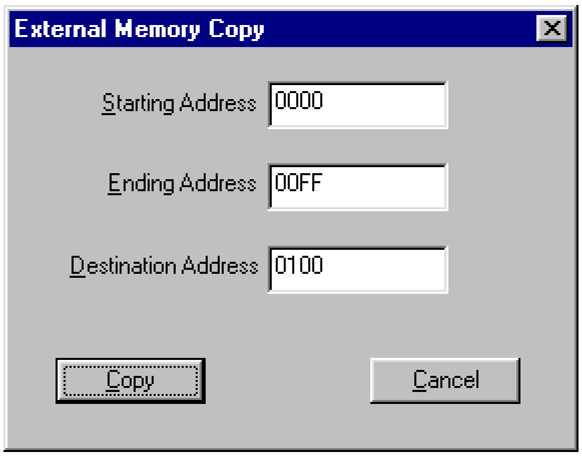
12.1 - Introduction

This chapter gives you an idea on how to copy a block of data from one location to other place in both internal and external memory areas. Separate commands are available to copy data in internal memory and external memory. The details are given here.

12.2 - Copying a Block of Data from One Location to Other Place

You can activate this command by selecting **Memory → External Data Memory → Copy.**

Now you may see a dialog box pops up at the center of the screen prompting you for the address of the data blocks.

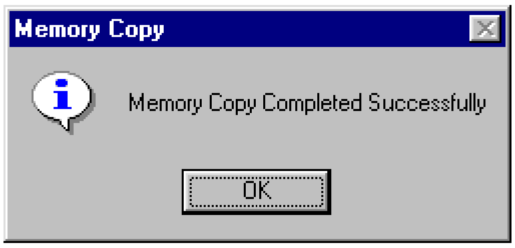

External Memory Copy

Starting Address 0000

Ending Address 00FF

Destination Address 0100

Enter the required information and press **Copy** command to start the operation. The completion of this copy operation is indicated by this window.


Memory Copy

Memory Copy Completed Successfully

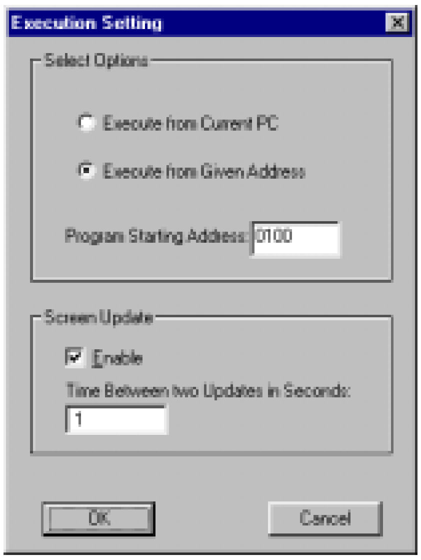
13.1 - Introduction

This command is one of the most used in the debugging process. Topview Simulator using right GUI features gives more information during the execution of this command enabling you to get the complete picture. This chapter gives more details about this most important and used command.

13.2 - Program Execution in Full Speed

Topview Simulator gives you total control on the program execution. You can execute your target program in many ways. You can execute the program in a single shot indicating the starting address of the program. You can make the simulator to execute the program upto a BreakPoint and then analyze the contents of various registers or the desired memory locations. May be you can execute the program line by line using SingleStep command or execute subroutines in a single step by the StepOver command. Simulator gives you the required flexibility in controlling the application code execution and now we are going to see how to execute the program in single shot in full speed. **Run** menu gives commands meant for full speed program execution : **Go** and **Goto**.

Run menu also carries setting option to define address selection required by the **Go** command. Before using **Go** command, click **Setting** and define your choice in the address of Program Counter.


Execution Setting

Select Options

Execute from Current PC [✓ Execute from Given Address]

Program Starting Address: 0100

Screen Update

[✓ Enable]

Time Between two Updates in Seconds: 1

If you enable the **Execute from Current PC** choice, the current PC value will be considered as the starting address for the program execution. Otherwise, the program starting address will always be taken from the user defined PC value in the **Setting** option.

If the first option, **Execute from Current PC** has been enabled, then each time you have to set the PC value.

Program execution from the current PC option is helpful when using BreakPoints in your development. When the debugger stops the program flow at the BreakPoint value, you can continue program execution without redefining PC contents again. There are two ways to set the PC value.

When right clicking your mouse over the correct address in the Program Window you get a pop up window giving few options. You select **Set PC** to initialize PC with that particular address. Now that instruction is highlighted with yellow colour to indicate this step.

The second way is to enter the right address into PC in the Register Window.

If the second option of the **Run** menu setting, **Execute from Given Address** is enabled, the starting address for the PC when using the **Go** command will always be taken from the user provided information of the setting. This second method is very helpful in the occasions when you are debugging a program module (from the fixed address) in which case you need not define PC for every execution.

During repeated program debugging, just press **Go** button to start the program execution.

Using this setting, the screen update timing can also be set.

Program Execution in Full Speed

To get an idea on this **Go** command, enter the following program.

```
0000  CPL        P1.4
0002  ACALL      0050
0004  SJMP       0000
0050  MOV        R7,#05
0052  MOV        R6,#7F
0054  MOV        R5,#FF
0056  DJNZ       R5,0056
0058  DJNZ       R6,0054
005A  DJNZ       R7,0052
005C  RET
```

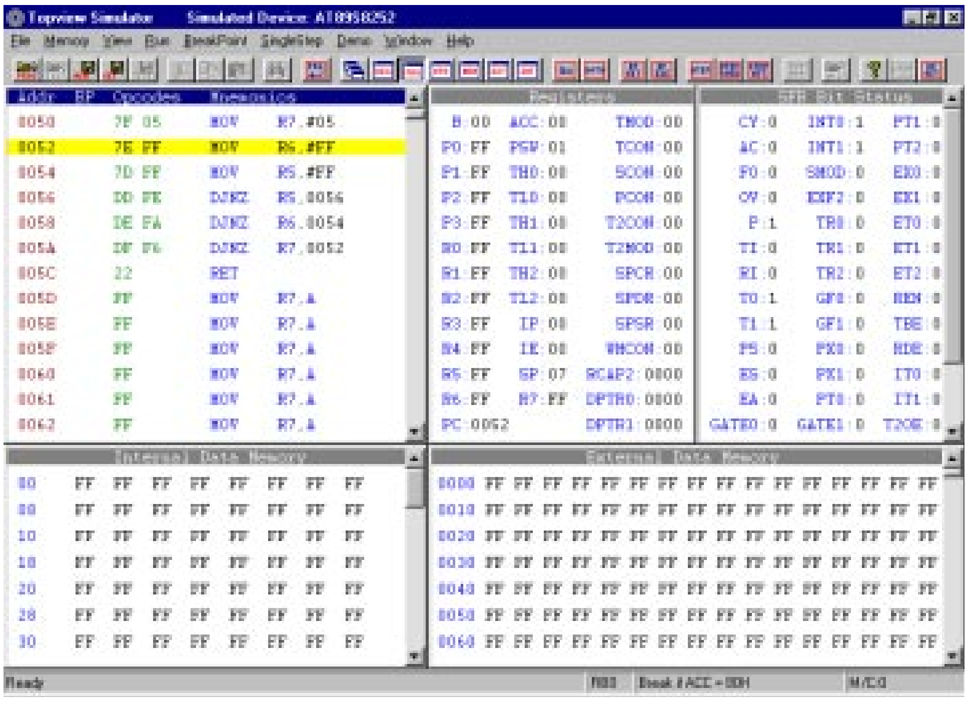
This program will flash the LED connected at port line P1.4

Now initialize the PC with the starting address of the program. Then click **Go** form **Run** menu or click over **Go** button in the toolbar.

You can notice the LED connected with the port line starts flashing at an approximate interval of one second.

The program execution can be stopped by clicking over **Stop Execution** button or pressing **Esc** key.

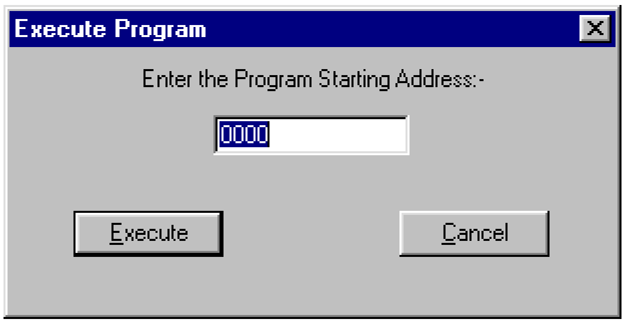
When you stop any execution, all the windows are refreshed and the PC at that moment of execution of stop command is highlighted as shown here:


Topview Simulator   Simulated Device: AT89S8252
File Memory View Run BreakPoint SingleStep Demo Window Help
Addr BP Opcodes Mnemonics
Registers
SFR Bit Status
Internal Data Memory
External Data Memory
Ready

```
0050    7F 05    MOV    R7,#05
0052    7E FF    MOV    R6,#FF
0054    7D FF    MOV    R5,#FF
0056    DD FE    DJNZ   R5,0056
0058    DE FA    DJNZ   R6,0054
005A    DF F6    DJNZ   R7,0052
005C    22       RET
005D    FF       MOV    R7,A
005E    FF       MOV    R7,A
005F    FF       MOV    R7,A
0060    FF       MOV    R7,A
0061    FF       MOV    R7,A
0062    FF       MOV    R7,A

 B:00   ACC:00     TMOD:00      CY:0    INT0:1   PT1:0
P0:FF   PSW:01     TCON:00      AC:0    INT1:1   PT2:0
P1:FF   TH0:00     SCON:00      F0:0    SMOD:0   EX0:0
P2:FF   TL0:00     PCON:00      OV:0    EXF2:0   EX1:0
P3:FF   TH1:00    T2CON:00       P:1     TR0:0   ET0:0
R0:FF   TL1:00    T2MOD:00      TI:0     TR1:0   ET1:0
R1:FF   TH2:00     SPCR:00      RI:0     TR2:0   ET2:0
R2:FF   TL2:00     SPDR:00      T0:1     GF0:0   REN:0
R3:FF    IP:00     SPSR:00      T1:1     GF1:0   TBE:0
R4:FF    IE:00    WMCON:00      PS:0     PX0:0   RDE:0
R5:FF    SP:07   RCAP2:0000     ES:0     PX1:0   IT0:0
R6:FF    R7:FF   DPTR0:0000     EA:0     PT0:0   IT1:0
PC:0052          DPTR1:0000  GATE0:0   GATE1:0  T2OE:0

00   FF FF FF FF FF FF FF FF
08   FF FF FF FF FF FF FF FF
10   FF FF FF FF FF FF FF FF
18   FF FF FF FF FF FF FF FF
20   FF FF FF FF FF FF FF FF
28   FF FF FF FF FF FF FF FF
30   FF FF FF FF FF FF FF FF

0000 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0010 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0020 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0030 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0040 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0050 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0060 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

R00   Break if ACC = 00H   M/C:0
```

13.3 - Goto Command

This is almost similar to **Go** command. But you can use this as a short cut during program executions. Whenever you activate this command, it may ask for a starting address of your program.



Enter the Program Starting Address:-

0000

You can use this command as a short cut in following conditions:

Goto Command

Suppose you are debugging a lengthy program and you keep the ClearView configuration as active. You have to either scroll up or down to define the PC with the starting address.

Here just click **Goto** command in the toolbar and enter your target address to execute the code.

Also assume you have different program modules at different places of your memory and PC has to be loaded with right address every time.

Here also you can initialize the PC with the correct address in just one step.

Other operations of this **Goto** command are similar to **Go** command.

Now you can appreciate various full speed execution possibilities when using Topview Simulator that save your valuable development time.

13.4 - Reset CPU Command

Reset command can be used to initialize the simulated device with pre-defined conditions.

Click **Run → Reset CPU** to activate this command during simulation, when the Reset command is executed, following initialization is activated:

SFR Name	Reset Value
PC	0000H
ACC	00H
B	00H
PSW	00H
SP	07H
DPTR	0000H
P0-P3	FFH
IP	XXX00000B
IP	XX000000B
IE	0XX00000B
IE	0X000000B
TMOD	00H
T2MOD	XXXXXX00B
TCON	00H
T2CON	00H
TH0	00H
TL0	00H
TH1	00H
TL1	00H
TH2	00H
TL2	00H
RCAP2H	00H
RCAP2L	00H
SCON	00H
SBUF	Indeterminate
PCON	0XXX0000B

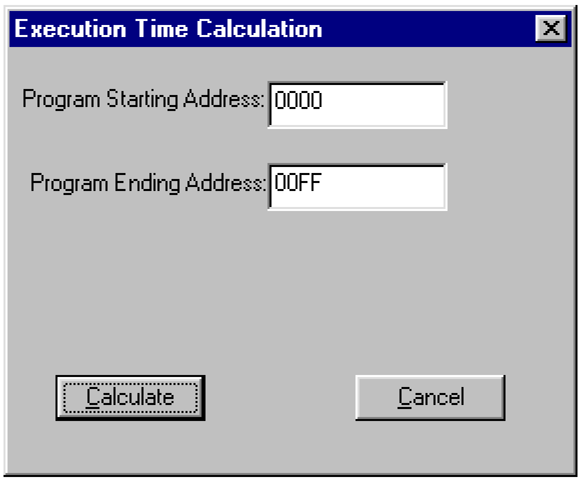
13.5 - Calculate Execution Time

The simulator has facility to calculate the execution time of the program taking operating clock into account.

You need to key in both starting and ending address of the program to get the correct execution time. This is useful to calculate the exact delay times whenever it is required.

Click **Run → Calculate Execution Time**.

A dialog box pops up prompting you to key in both Starting and Ending address of the program.



Program Starting Address: 0000

Program Ending Address: 00FF

Click **Calculate** button to start the process.

Now another small window indicates the execution time and also the number of machine cycles.

Press **OK** to quit this command.

13.6 - Initialize Machine Cycle

This command can be used to initialize the counter that indicates total machine cycles to zero.

Click **Run → Initialize Machine Cycle**.

You should see the total machine cycle counter initialized to 0 at the right corner of status bar.

14.1 - Introduction

This type of program control is another way to keep track of your programs' behavior during execution. You can make the simulator watch for a specific data in any memory location/register when the target program is executed. When the desired value is reached in the target location/register, simulator stops the program flow and refreshes all the windows with the current information to enable you to check for the desired results.

14.2 - Program Execution Using BreakPoints

You can set a BreakPoint in many places thanks to the Simulator's flexible defining facility.

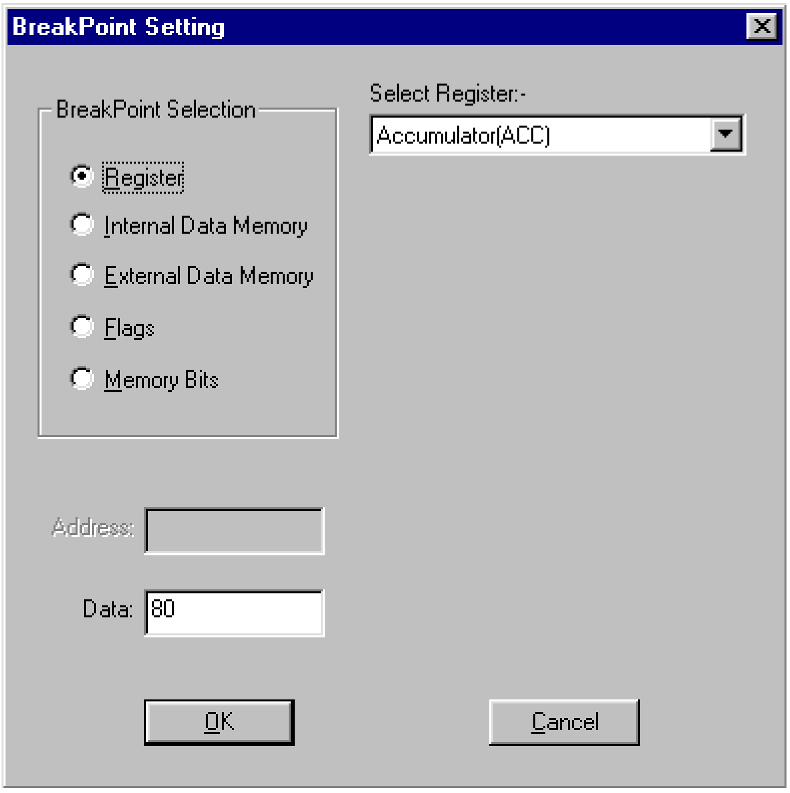
- Program Counter.
- Any of these registers:
  ACC, B, DPH, DPL, DPTR, R0, R1, R2, R3, R4, R5, R6, R7 and SP.
- Internal Memory Location.
- External Memory Location.
- Any of these flags : Carry (C); Auxiliary Cary (AC), Flag 0, Overflow Flag (OV) and the Parity Flag (P).
- Any of the memory bit between addresses, 00H and 7FH.

As you can see, this range of options make your BreakPoint selection as very useful one during debugging.

To activate this command, click **Setting** of **BreakPoint** menu.

A dialog box pops up at the center of the screen prompting you to define the BreakPoint.

Program Execution Using BreakPoints

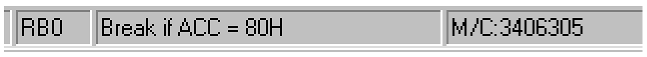


Select Register:- Accumulator(ACC)

Address:

Data: 80

Make your selection and initialize the location/Register with the BreakPoint data and click over **OK** button or press **Enter** key. Now the BreakPoint is set for your next operation and the same is indicated in the status bar as shown:



RB0 | Break if ACC = 80H | M/C:3406305

Now the simulator starts waiting for the BreakPoint execution command. PC contents are considered as the program starting address when using the **BreakPoint Execution** command.

So before executing the program, initialize the PC with the correct starting address.

You can set the PC BreakPoint in an easy way. As you can see, the Program Window has a **BP** indication that points to the PC BreakPoint assignment at that specific address.

Program Execution Using BreakPoints

Just double click at the required address under **BP** indication to define that particular address as PC BreakPoint. Simulator accepts this assignment by displaying the indication **B** at that address location.

You can also set this PC BreakPoint by right clicking any instruction or its address in the Program window. Select the **Set BreakPoint** from the pop up window.

You can remove any BreakPoint that has been already defined by the same way. Just right click at that BreakPoint and select **Remove BreakPoint** from the pop up window.

To give you more clear picture of this BreakPoint execution, an example program is given here:

```
0000  MOV A, #00
0002  INC A
0003  SJMP  0002
```

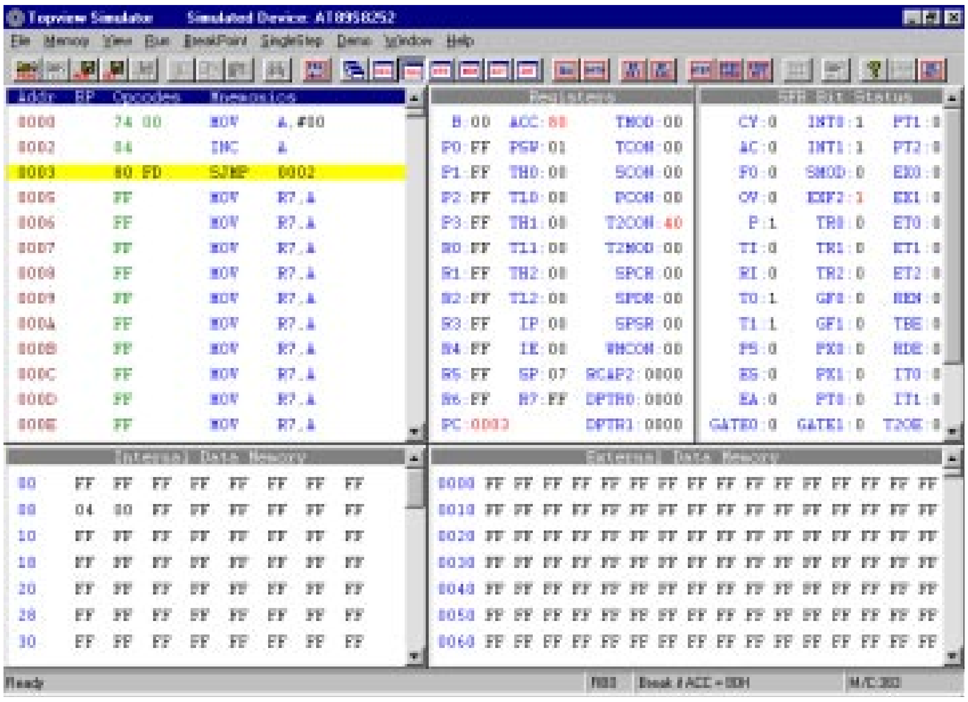
This small program increments the Acc from 00H to FFH continuously.

Set a BreakPoint to break the program flow when ACC gets the value 80H.

Activate BreakPoint Execution by clicking **Execution** in **BreakPoint** menu.

The program execution is stopped when the BreakPoint condition is met and all the windows are refreshed and you can see the changed Registers/Memory locations with highlights.

Program Execution Using BreakPoints



```
Topview Simulator      Simulated Device: AT89S8252
File  Memory  View  Run  BreakPoint  SingleStep  Demo  Window  Help

Addr  BP  Opcodes    Mnemonics
0000      74 80      MOV   A,#80
0002      04         INC   A
0003      80 FD      SJMP  0002
0005      FF         MOV   R7,A
0006      FF         MOV   R7,A
0007      FF         MOV   R7,A
0008      FF         MOV   R7,A
0009      FF         MOV   R7,A
000A      FF         MOV   R7,A
000B      FF         MOV   R7,A
000C      FF         MOV   R7,A
000D      FF         MOV   R7,A
000E      FF         MOV   R7,A

Registers
 B:00   ACC:80     TMOD:00
P0:FF   PSW:01     TCON:00
P1:FF   TH0:00     SCON:00
P2:FF   TL0:00     PCON:00
P3:FF   TH1:00    T2CON:40
R0:FF   TL1:00    T2MOD:00
R1:FF   TH2:00     SPCR:00
R2:FF   TL2:00     SPDR:00
R3:FF    IP:00     SPSR:00
R4:FF    IE:00    WMCON:00
R5:FF    SP:07   RCAP2:0000
R6:FF    R7:FF   DPTR0:0000
PC:0003          DPTR1:0000

SFR Bit Status
   CY:0    INT0:1    PT1:0
   AC:0    INT1:1    PT2:0
   F0:0    SMOD:0    EX0:0
   OV:0    EXF2:1    EX1:0
    P:1     TR0:0    ET0:0
   TI:0     TR1:0    ET1:0
   RI:0     TR2:0    ET2:0
   T0:1     GF0:0    REN:0
   T1:1     GF1:0    TBE:0
   PS:0     PX0:0    RDE:0
   ES:0     PX1:0    IT0:0
   EA:0     PT0:0    IT1:0
GATE0:0   GATE1:0   T2OE:0

Internal Data Memory
00    FF FF FF FF FF FF FF FF
08    04 80 FF FF FF FF FF FF
10    FF FF FF FF FF FF FF FF
18    FF FF FF FF FF FF FF FF
20    FF FF FF FF FF FF FF FF
28    FF FF FF FF FF FF FF FF
30    FF FF FF FF FF FF FF FF

External Data Memory
0000 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0010 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0020 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0030 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0040 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0050 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0060 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

Ready                        RB0   Break if ACC = 80H      M/C:303
```

For the example, you can notice the ACC contents as the BreakPoint value, 80H.

15.1 - Introduction

You can use this SingleStep command to analyze the target program line by line in complicated situations. It enables you to get the total picture of changes happened in Registers, Internal/External memory for every program line execution. The GUI of ClearView promptly indicates these changes in highlights to draw your attention.

15.2 - Program Execution Using SingleStep Command

This facility makes you understand the behavior of the target code when you develop complex applications.

Activate this command by this sequence :**SingleStep → SingleStep**.

The current PC contents are taken as the starting address of the program. So keep the program address in PC before activating SingleStep command.

When you activate this command, first instruction gets executed and the simulator refreshes all the windows with the current information and stops the execution and starts waiting for the next command.

For this command, the shortcut key is F7. So press this to execute the next instruction or click over **SingleStep** in **SingleStep** menu.

The following example helps you to understand fully this SingleStep command.

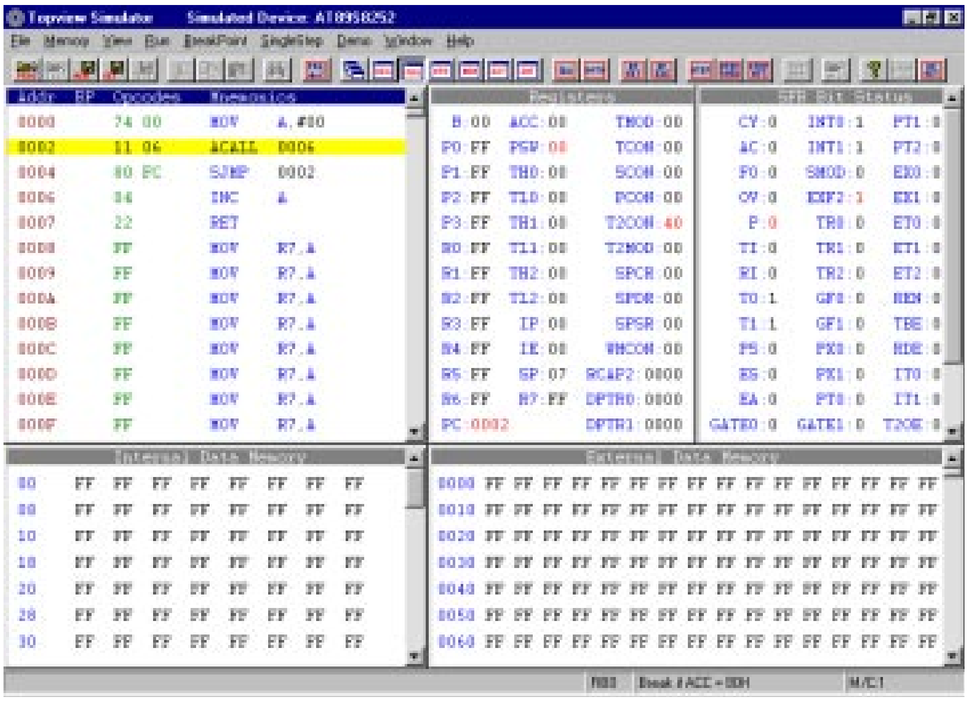
```
0000  MOV A,#00
0002  ACALL 0006
0004  SJMP  0002
0006  INC A
0007  RET
```

Program Execution Using SingleStep Command

This simple program will increment the ACC from 00H to FFH. It calls one subroutine to increment ACC contents.

After entering the program in memory, initialize the PC with the address, 0000H.

Activate the **SingleStep** command either from menu bar or tool bar. The first instruction, MOV A, #00 moves the data 00H to ACC and the program is made to stop. PC value becomes 0002H as shown in the fig.


Topview Simulator   Simulated Device: AT89S8252
File Memory View Run BreakPoint SingleStep Demo Window Help
Addr BP Opcodes Mnemonics
Registers
SFR Bit Status
Internal Data Memory
External Data Memory

```
0000    74 00   MOV    A,#00
0002    11 06   ACALL  0006
0004    80 FC   SJMP   0002
0006    04      INC    A
0007    22      RET
0008    FF      MOV    R7,A
0009    FF      MOV    R7,A
000A    FF      MOV    R7,A
000B    FF      MOV    R7,A
000C    FF      MOV    R7,A
000D    FF      MOV    R7,A
000E    FF      MOV    R7,A
000F    FF      MOV    R7,A
```

```
 B:00  ACC:00    TMOD:00      CY:0   INT0:1   PT1:0
P0:FF  PSW:00    TCON:00      AC:0   INT1:1   PT2:0
P1:FF  TH0:00    SCON:00      F0:0   SMOD:0   EX0:0
P2:FF  TL0:00    PCON:00      OV:0   EXF2:1   EX1:0
P3:FF  TH1:00   T2CON:40       P:0    TR0:0   ET0:0
R0:FF  TL1:00   T2MOD:00      TI:0    TR1:0   ET1:0
R1:FF  TH2:00    SPCR:00      RI:0    TR2:0   ET2:0
R2:FF  TL2:00    SPDR:00      T0:1    GF0:0   REN:0
R3:FF   IP:00    SPSR:00      T1:1    GF1:0   TBE:0
R4:FF   IE:00   WMCON:00      PS:0    PX0:0   RDE:0
R5:FF   SP:07   RCAP2:0000    ES:0    PX1:0   IT0:0
R6:FF   R7:FF  DPTR0:0000     EA:0    PT0:0   IT1:0
PC:0002        DPTR1:0000  GATE0:0  GATE1:0  T2OE:0
```

```
00   FF FF FF FF FF FF FF FF
08   FF FF FF FF FF FF FF FF
10   FF FF FF FF FF FF FF FF
18   FF FF FF FF FF FF FF FF
20   FF FF FF FF FF FF FF FF
28   FF FF FF FF FF FF FF FF
30   FF FF FF FF FF FF FF FF
```

```
0000 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0010 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0020 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0030 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0040 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0050 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0060 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

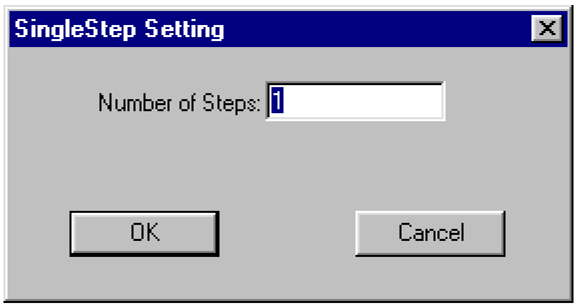
```
RB0   Break if ACC = 00H   M/C:1
```

To execute the next instruction, you can activate **SingleStep** command again. You can notice that all the windows are refreshed with the current information in highlights using different colours to grab your attention.

15.3 - SingleStep Setting

The number of steps (i.e. number of instructions to be executed) for the SingleStep and StepOver commands can be set using this command.

To activate the command, select **Setting** from **SingleStep** menu.



When the command is activated, a dialog box will appear on screen prompting the number of steps (program lines). Enter the number of steps and press **Enter** key or click **OK** button to set the value.

16.1 - Introduction

This is another useful command meant for debugging your complicated target code. This is similar to the SingleStep command in operation but it takes routine calls as a single instruction. You need not execute all the instructions of the routine line by line. The complete routine is executed in a single shot. Other instruction lines are executed line by line like SingleStep execution.

16.2- Program Execution Using StepOver Command

To activate this command, select **StepOver** from the **SingleStep** menu. This command executes your program line by line and call routines by ACALL and LCALL are treated as single line. During the program flow, this command executes these routines in a single shot.

As usual all the windows are promptly refreshed after every program line/routine execution.

Key in and execute the following program using this command:

```
0000  MOV    A,#00
0002  ACALL  0006
0004  SJMP   0002
0006  INC    A
0007  RET
```

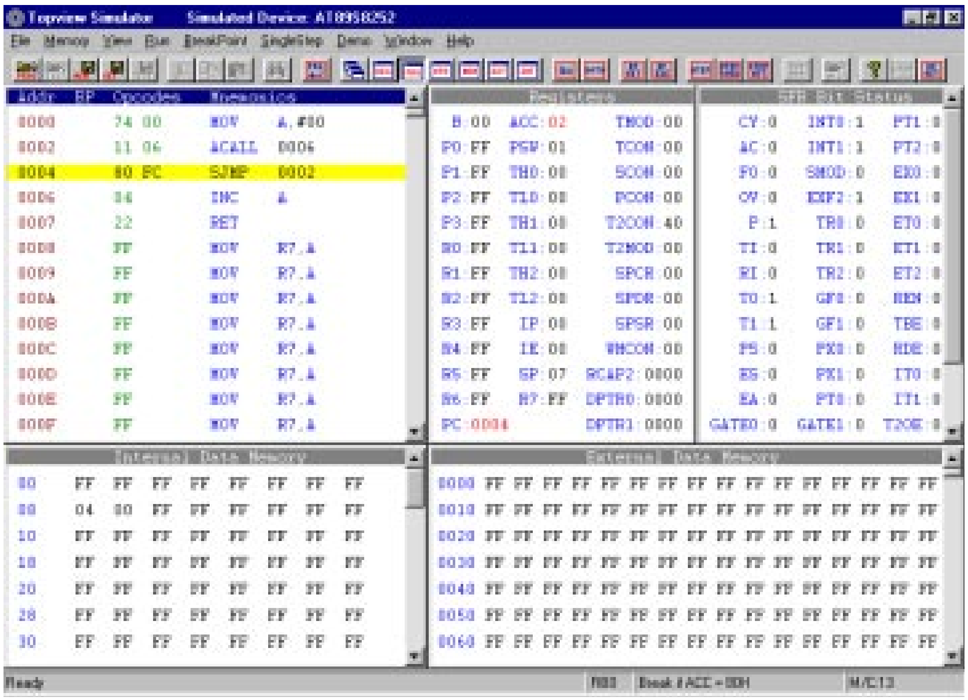
This program increments ACC from 00H to FFH. It calls a subroutine to increment ACC contents.

Initialize the PC with the address, 0000H.

Activate the StepOver command from either menu bar or toolbar. The first instruction moves data 00H to ACC and the program flow gets stopped. Now PC becomes 0002H. Press **StepOver** button again to execute current

Program Execution Using StepOver Command

instruction. ACALL is executed in one step and PC points to the next instruction at 0004H.


Topview Simulator   Simulated Device: AT89S8252
File Memory View Run BreakPoint SingleStep Demo Window Help

```
Addr  BP  Opcodes   Mnemonics
0000      74 00     MOV    A,#00
0002      11 06     ACALL  0006
0004      80 FC     SJMP   0002
0006      04        INC    A
0007      22        RET
0008      FF        MOV    R7,A
0009      FF        MOV    R7,A
000A      FF        MOV    R7,A
000B      FF        MOV    R7,A
000C      FF        MOV    R7,A
000D      FF        MOV    R7,A
000E      FF        MOV    R7,A
000F      FF        MOV    R7,A
```

```
Registers
 B:00   ACC:02      TMOD:00
P0:FF   PSW:01      TCON:00
P1:FF   TH0:00      SCON:00
P2:FF   TL0:00      PCON:00
P3:FF   TH1:00     T2CON:40
R0:FF   TL1:00     T2MOD:00
R1:FF   TH2:00      SPCR:00
R2:FF   TL2:00      SPDR:00
R3:FF    IP:00      SPSR:00
R4:FF    IE:00     WMCON:00
R5:FF    SP:07    RCAP2:0000
R6:FF    R7:FF    DPTR0:0000
PC:0004           DPTR1:0000
```

```
SFR Bit Status
   CY:0    INT0:1    PT1:0
   AC:0    INT1:1    PT2:0
   F0:0    SMOD:0    EX0:0
   OV:0    EXF2:1    EX1:0
    P:1     TR0:0    ET0:0
   TI:0     TR1:0    ET1:0
   RI:0     TR2:0    ET2:0
   T0:1     GF0:0    REN:0
   T1:1     GF1:0    TBE:0
   PS:0     PX0:0    RDE:0
   ES:0     PX1:0    IT0:0
   EA:0     PT0:0    IT1:0
GATE0:0   GATE1:0   T2OE:0
```

```
Internal Data Memory
00    FF  FF  FF  FF  FF  FF  FF  FF
08    04  00  FF  FF  FF  FF  FF  FF
10    FF  FF  FF  FF  FF  FF  FF  FF
18    FF  FF  FF  FF  FF  FF  FF  FF
20    FF  FF  FF  FF  FF  FF  FF  FF
28    FF  FF  FF  FF  FF  FF  FF  FF
30    FF  FF  FF  FF  FF  FF  FF  FF
```

```
External Data Memory
0000 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0010 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0020 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0030 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0040 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0050 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0060 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

```
Ready                    RB0   Break if ACC = 00H     M/C:13
```

Trigger **StepOver** command again. The ACC content is incremented by calling the routine at 0006H in one step. Now PC becomes 0004 instead of 0006.

Keep clicking **StepOver** button till you finish debugging.

16.3 - Finish Subroutine in SingleStep

You can use this command to execute a total subroutine in a Single step.

Click **SingleStep → Finish Subroutine**.

When this command is activated, your target program is executed from the current PC value and execution continuous upto RET instruction. Because of this, you can notice the execution of subroutines in a single shot enabling you to debug bigger programs.

17.1 - Introduction

Topview Simulator is an easy and versatile development tool to design and complete 8031 based real time applications in less time. This tutorial gives a complete picture of 'How - to Start - and Complete - My Project' using this simulator.

17.2 - How to start from the scratch?

You can start and complete your real time applications in few easy steps:

Now consider a simple application where you are expected to drive a point LED On and Off at a predefined interval. The procedure is the same even if you venture into bigger and complex applications.

Step 1 : Open the Text Editor and key in the target code.

Step 2 : Assemble target code using an assembler.

Step 3 : Load the Hex file into the simulator memory.

Step 4 : Set the LED module active by initializing a point LED at P1.0 line.

Step 5 : Keep all the required windows open. Here you need to keep program window and LED module window active.

Step 6 : Execute the program in full speed and check for the desired results.

Step 7 : If required, execute the target program code in SingleStep or StepOver mode to observe the changes happening at various program points.

17.3 - Step 1 : Open Text Editor and Key in Target Program

Activate the simulator software, select the device, operating frequency, define the amount of program and data memory as per project requirement.

Click **Load Text File** from **File** menu to open the text editor.

Now a dialog box comes up on the screen prompting you for the file name. Enter the file name for the new file or select one from the list shown.

Key in this program in the editor.

```
$MOD51

           ORG   0000H
           SJMP  START

           ORG   0050H
START:
           CPL   P1.0
           ACALL DELAY
           SJMP  START

DELAY:
           MOV   R0,#07FH
LOOP:      MOV   R1,#0FFH
           DJNZ  R1,$
           DJNZ  R0,LOOP
           RET

           END
```

The above code is written for the ASM51 assembler. Save the entered code by selecting **Save Text File** from **File** menu.

Now your program is ready for further action.

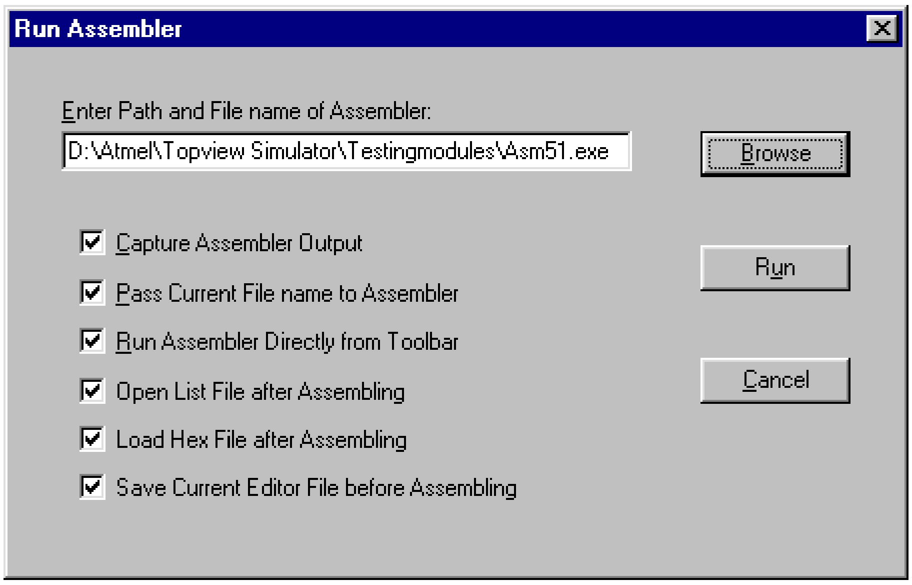
17.4 - Step 2 : Assemble the Program Using an Assembler

Now you know that Topview Simulator uses an external assembler to assemble raw keyed in program code. If the program length is small, you can even use simulator's built-in single line assembler.

Here it is assumed that we use the external assembler to assemble the program lines.

So, to assemble the contents of current editor file, click **Run Assembler** from **Command** menu.

A dialog box pops up to prompt you to decide on few operations.


Run Assembler

Enter Path and File name of Assembler:
D:\Atmel\Topview Simulator\Testingmodules\Asm51.exe

- ✓ Capture Assembler Output
- ✓ Pass Current File name to Assembler
- ✓ Run Assembler Directly from Toolbar
- ✓ Open List File after Assembling
- ✓ Load Hex File after Assembling
- ✓ Save Current Editor File before Assembling

Set the path meant for the external assembler using **Browse** button.

To capture the assembler output, check the **Capture Assembler Output** box.

To pass the current editor file name to assembler as command line parameter, check the **Pass Current File name to Assembler** box.

Check the **Run Assembler Directly from Toolbar** to run this command from the toolbar without opening this dialog box next time.

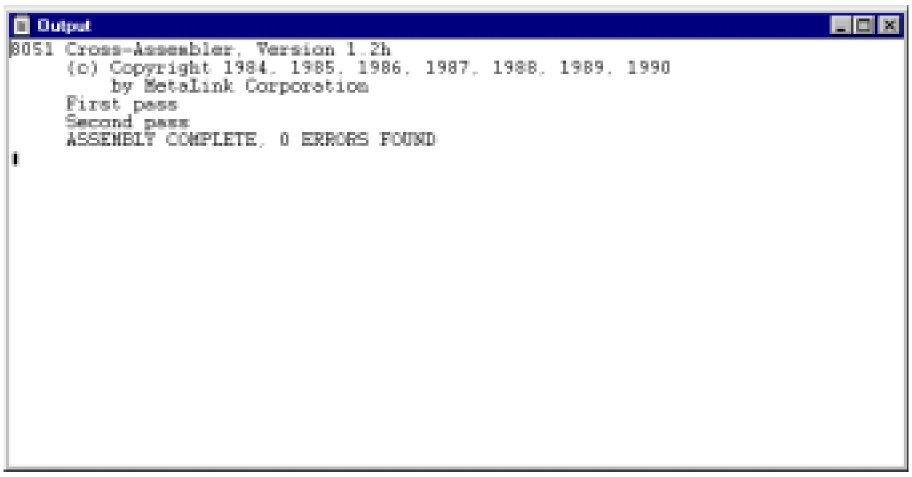
To open the list file after assembling, check **Open List File after Assembling** box.

To load the assembled program into the simulator, check **Load Hex File after Assembling** box.

To save the current editor file before passing to assembler, check the **Save Current Editor File before Assembling** box.

Click **Run** button to start assembling.

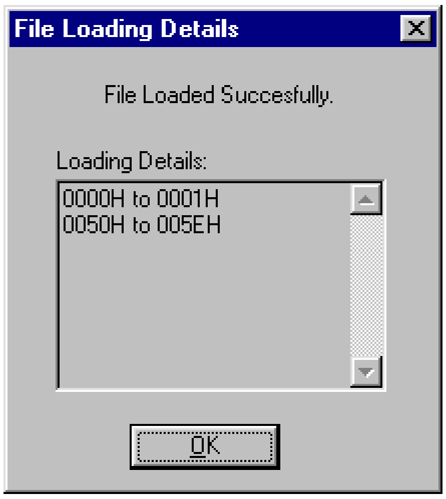
If the capture assembler output option is enabled then the assembler output is captured and displayed as shown here:



```
8051 Cross-Assembler, Version 1.2h
     (c) Copyright 1984, 1985, 1986, 1987, 1988, 1989, 1990
         by MetaLink Corporation
     First pass
     Second pass
     ASSEMBLY COMPLETE, 0 ERRORS FOUND
```

If there are any error, you can view the list file and correct the same.

When the **Load Hex File after Assembling** option is enabled, then the target program is automatically loaded into simulator's memory and is indicated by this summary window.



File Loaded Succesfully.

Loading Details:

```
0000H to 0001H
0050H to 005EH
```

17.5 - Step 3 : Load the Hex File Into Simulator Memory

Now you need to load the assembled Hex file into simulator memory for execution. In the previous step, if you have already enabled **Load Hex File after Assembling** option, the target program file will be automatically loaded into simulator memory.

Otherwise, suppose if you have a fully assembled Hex file ready for the simulation, then you need to use **Load Program** command from **File** menu.

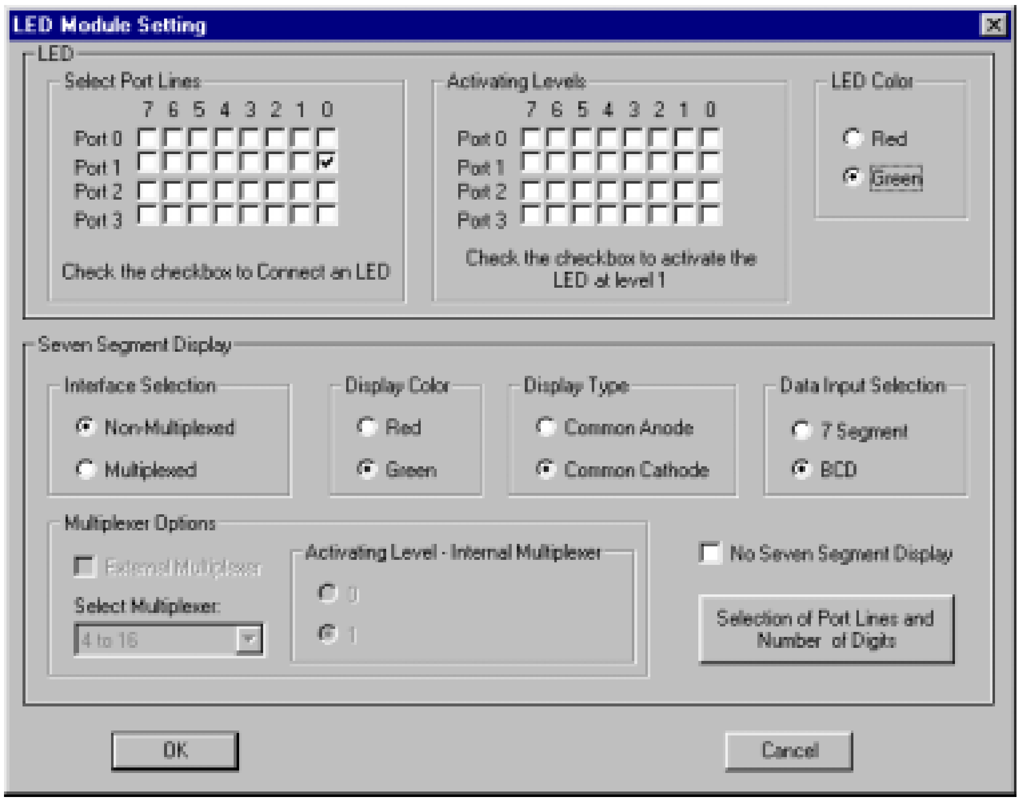
A dialog box may appear to get the file name. Enter or select the file name and click **Open** button.

The progress of the file loading is indicated in a window along with a summary.

17.6 - Step 4 : Set the LED Module Active by Initializing a Point LED at P1.0 line

To activate LED module click **File → External Modules → LED.**

A dialog box comes up to enable you to initialize different LED display options.



LED Module Setting

LED

Select Port Lines

	7	6	5	4	3	2	1	0
Port 0								
Port 1								✓
Port 2								
Port 3								

Check the checkbox to Connect an LED

Activating Levels

	7	6	5	4	3	2	1	0
Port 0								
Port 1								
Port 2								
Port 3								

Check the checkbox to activate the LED at level 1

LED Color: Red [✓ Green]

Seven Segment Display

Interface Selection: [✓ Non-Multiplexed] Multiplexed

Display Color: Red [✓ Green]

Display Type: Common Anode [✓ Common Cathode]

Data Input Selection: 7 Segment [✓ BCD]

Multiplexer Options: External Multiplexer

Select Multiplexer: 4 to 16

Activating Level - Internal Multiplexer: 0 [✓ 1]

No Seven Segment Display

Selection of Port Lines and Number of Digits

OK Cancel

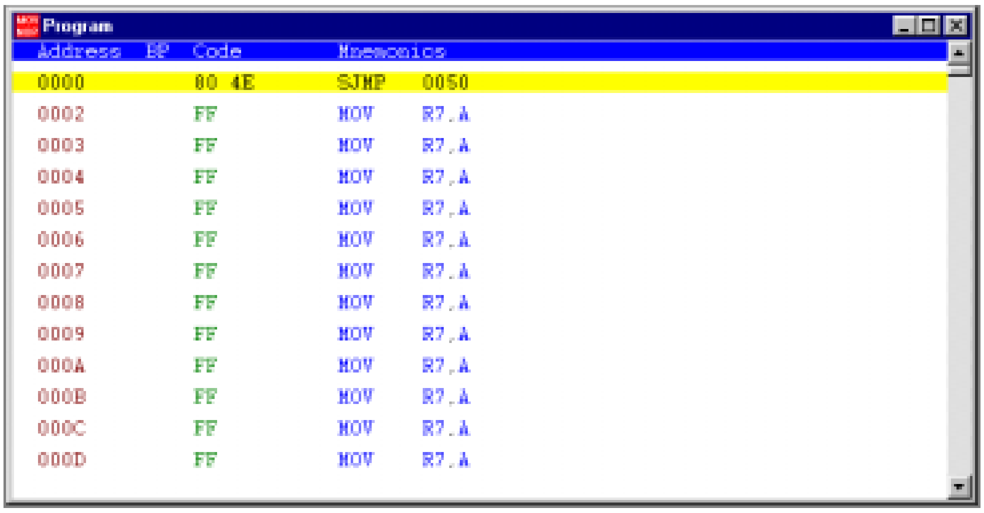
Here check the box corresponding LED box at P1.0 port line and also select the activating level.

Click **OK** button to complete the task.

17.7 - Step 5 : Keep Required Windows Active

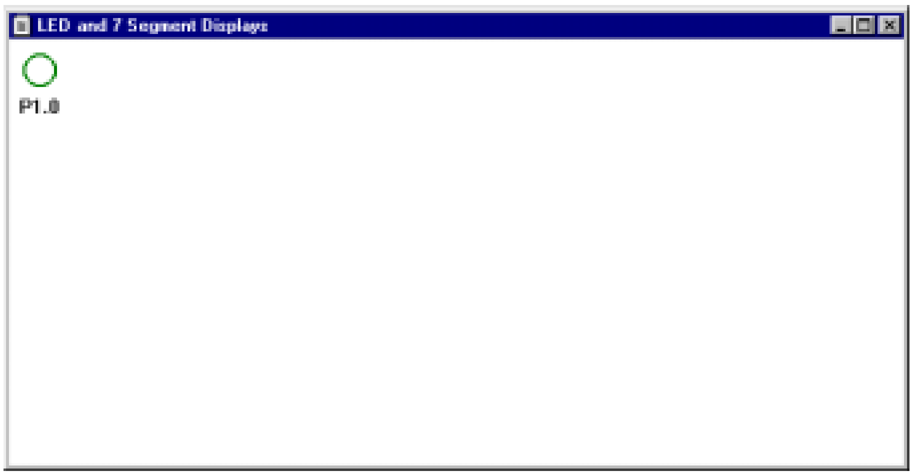
You need to keep all relevant windows active to visualize the working of these embedded building blocks.

For this tutorial, you have to keep Program Window and the LED Window in active condition. Click **View → Program Window** to get program window in open.



```
Address  BP  Code      Mnemonics
0000         80 4E     SJMP   0050
0002         FF        MOV    R7,A
0003         FF        MOV    R7,A
0004         FF        MOV    R7,A
0005         FF        MOV    R7,A
0006         FF        MOV    R7,A
0007         FF        MOV    R7,A
0008         FF        MOV    R7,A
0009         FF        MOV    R7,A
000A         FF        MOV    R7,A
000B         FF        MOV    R7,A
000C         FF        MOV    R7,A
000D         FF        MOV    R7,A
```

Select LED Window by clicking **View → External Modules → LED**



Another active window comes up to indicate the selected LED.

17.8 - Step 6 : Execute the Program in Full Speed

To execute your target program, there are two ways. First one is set the PC with the starting address of your program in the Register Window.

Then select **Run → Go** Command to start execution.

Second way is execute **Run → Goto**. This will prompt you to enter the starting address in a small window.

Key in the right address and press **OK** button to start execution.

17.9 - Step 7 : Execute the Target Program in SingleStep or StepOver Mode

You can control the program execution in few other ways to get more insight on how program behaves during execution. You can command the simulator execute your target program line by line or execute program lines in steps. You can define number of lines for one step of execution.

First, set the PC with the correct starting address by modifying the PC in the Register Window.

Now click **SingleStep → SingleStep** command to execute the first line of the program.

You can even set the number of program lines to be executed for every step when executing StepOver mode.

Set this here : **SingleStep → Setting**

When you execute this command, you can see the changes happened in memory area, registers, flags in all the respective windows. All the windows are refreshed promptly and changed locations are highlighted to grab your attention.