Quick Starting

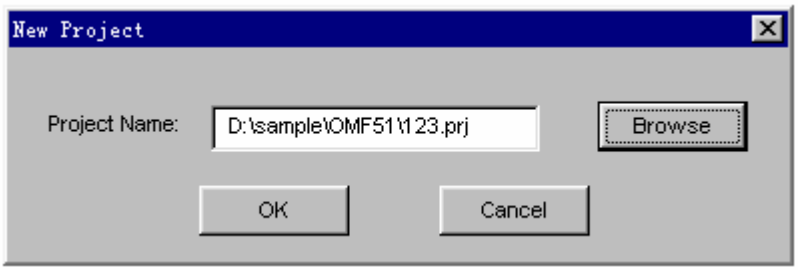
Summarization

This chapter introduces how to use the emulator of TOPICE quickly.

Compiling Source and Debugging

Creating a New Project

Select main menu “Project”, then choose the submenu “New” to create a new project. The project file has the name with extension “.prj”. The project file must be put at the same subdirectory as the source files. The path and file name should be DOS compatible.



(picture 1)

●Project Name

Type the name (with path) for the new project.

●Browse

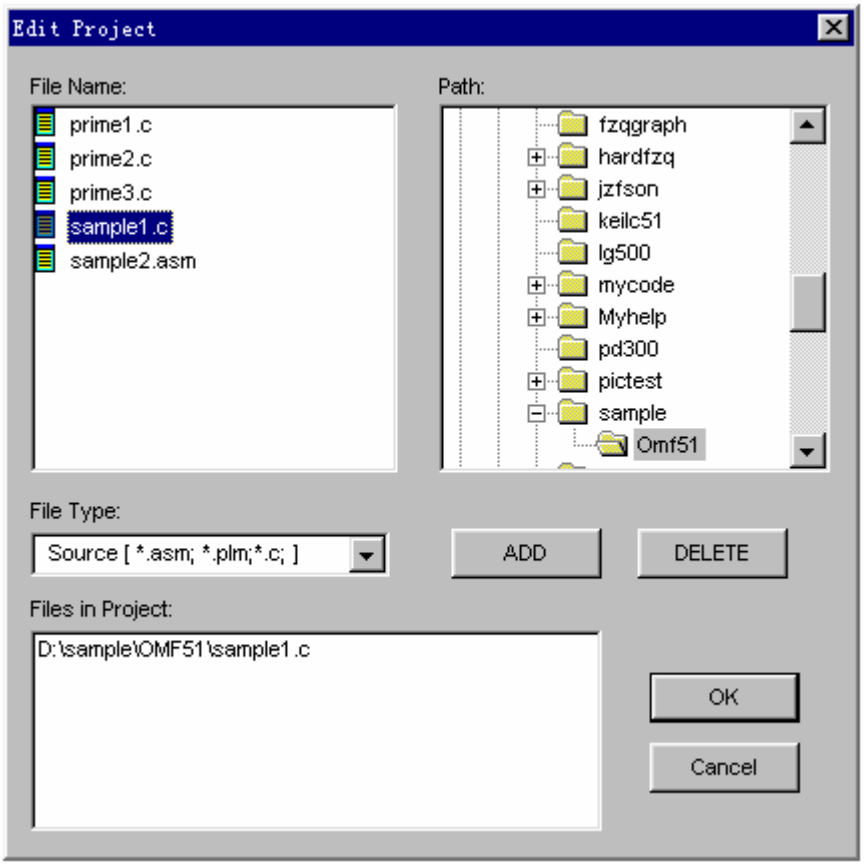
Display the “Browse Project” dialog box to select or change the name (with path).

●OK

Click to open the “Edit Project” dialog box to create the project files.

Edit a New Project

Select main menu “Project”, then choose the submenu “Edit” to edit a project.
The files in the project must be put at the same directory of the project file.(The C51 file has the name with extension “.c”, the assemble file with extension “.asm” and PL/M51 file with extension “.plm”)



(picture 2)

●File Name

List files with the filename extension selected in the “File Type” list box.

●Path

Select the path you want to contain in the project. After double-clicking on a path, all files with the filename extension selected in the “File Type” list box will be shown in the File Name box.

●File Type

Select the type of file you want to see in the File Name box.

●File in Project

List all files selected from File Name box

●Add

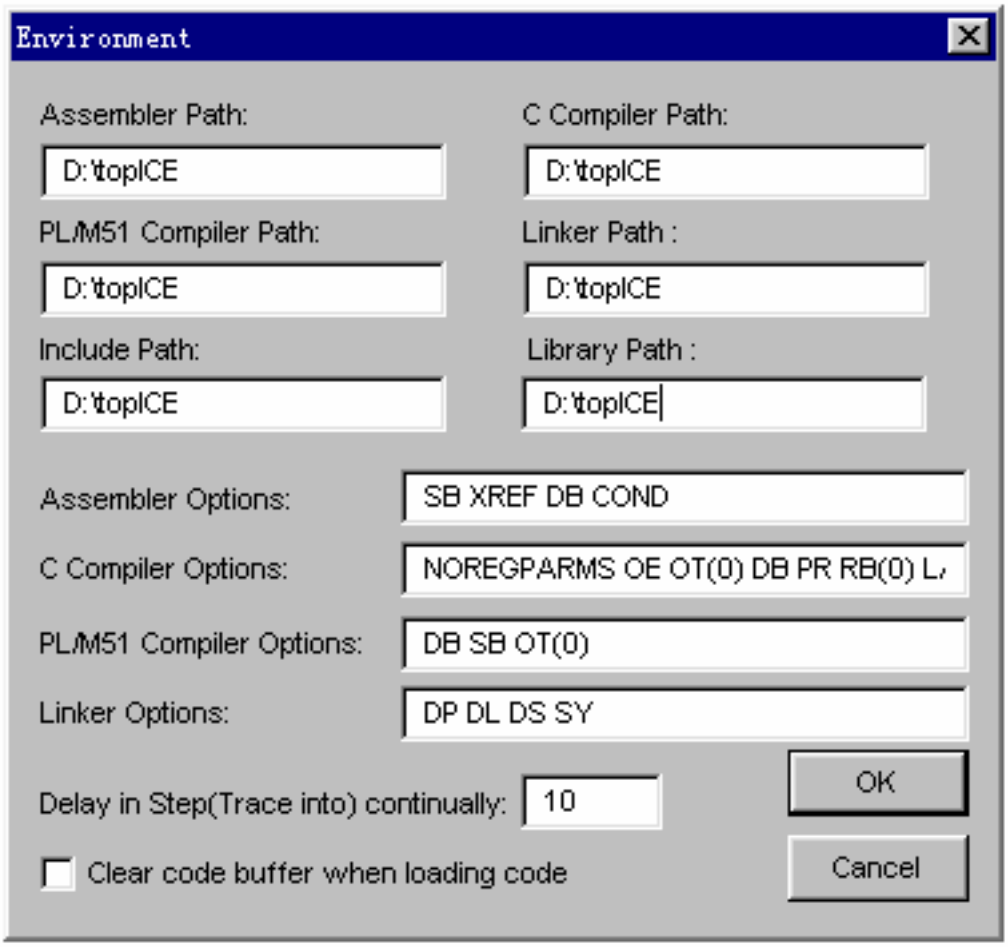
Add the highlighted item in the File Name box project box.

●Delete

Remove the highlighted file in Files in Project box from current project.

Setting Compiler Options

Select main menu “Option”, then choose the submenu “Environment” to set compiler options. The needed files must be copied to the special path, which include the Assembler named “A51.EXE”, C Compiler named “C51.EXE”, PL/M51 Compiler named “PLM51.EXE”, Linker named “L51.EXE”, etc.



(picture 3)

- **Assembler Path**

Type the path that contains the Assembler named “A51.EXE”.

- **C Compiler Path**

Type the path that contains the C Compiler named “C51.EXE”.

- **PL/M51 Compiler Path**

Type the path that contains the PL/M51 Compiler named “PLM51.EXE”.

- **Linker Path**

Type the path that contains the Linker named “L51.EXE”.

- **Include Path**

Type the path that contains include files that are provided with the standard library.

- **Library Path**

Type the path that contains the compiler library.

- **Assembler Options**

Type the Assembler options separated by space.

- **C Compiler Options**

Type the C Compiler options separated by space.

- **PL/M51 Compiler Options**

Type the PL/M51 Compiler options separated by space.

●**Linker Options**
Type the Linker options separated by space.
●**Delay**
Type the interval between two step when step continually.
●**Clear Code**
Clear the code buffer when load the program code.

Suggested Compilers
Franklin(Keil) tool chain C51(*.OMF) versions V3.20 or above
C Compiler Version 3.20 or above
Assembler Version 4.86 or above
PL/M51 Compiler Version 1.40 or above
Linker Version 3.11 or above

Build Project
Select main menu “Project”, then choose the submenu “Build Project” to compile project (according the options made in the previous session). If no error, ”Build” window displays the message “Build OK” and the system loads the code and downloads the code to the emulator. Otherwise, “Build” window displays error messages. You can locate the error line in the source window when double-click the error message in the “Build” window, modify the error and build project again.
Note: Please make sure that all used compiler files.

Load Code
Select main menu “File”, then choose the submenu “Load as”.

Source Level Debugging

Choose any submenu of main menu “Run” according to your need. Source Level Debugging works only with files in Intel OMF or Extended OMF format which contain necessary debugging messages including LINE numbers..
Debugging from Source Window
Select main menu “File”, then choose the submenu “Load File” to open source window. In source window you can view the source file, start and stop emulation, set breakpoints, step over and trace into, and watch variables. You can also modify the source file and rebuild the project.
NOTE: Programs start in assembly code and not in main() in C source level debugging. You can set breakpoints in source window and run to breakpoint.

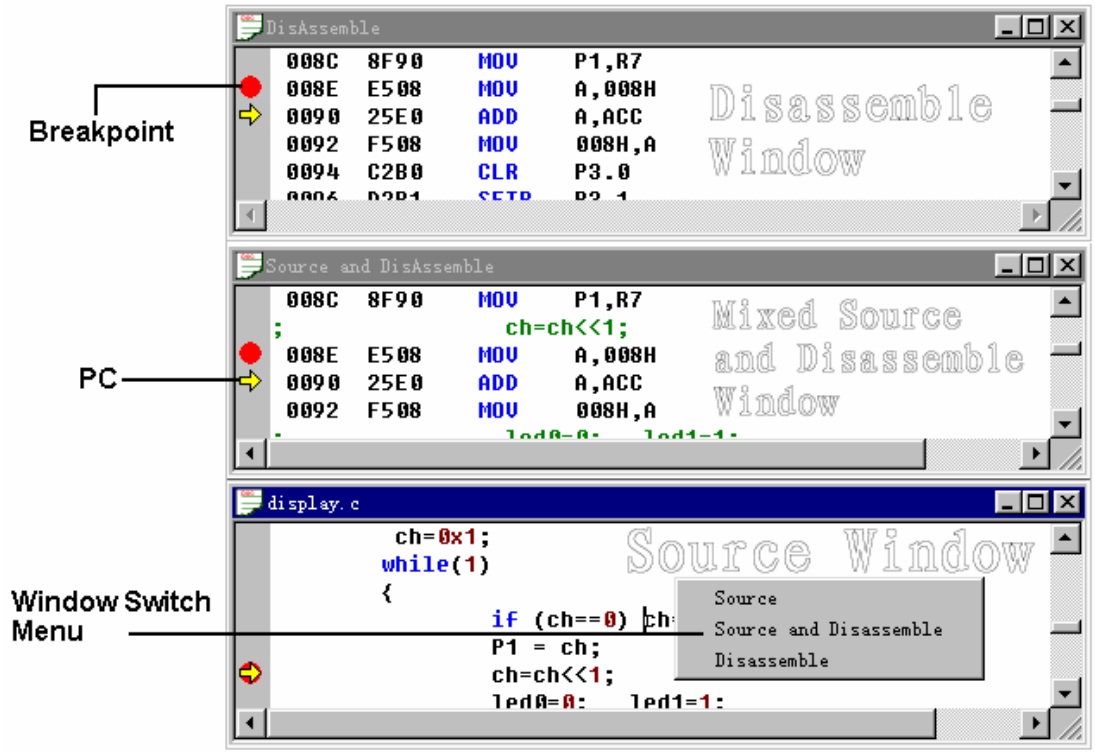
Mixed Source and Disassemble Window
Select main menu “View”, then choose the submenu “Source and Disassemble” to open the window that displays the program source

intermixed with the assembly language compiled for each source line.

Setting Breakpoint

Select main menu “Run”, then choose the submenu “Insert/Remove Breakpoint” or move the mouse to the left of the source line and double-clik the mouse. If a breakpoint is set, a breakpoint symbol will appear to the left margin of the source line.

Setting breakpoint must be done after loading code. If you try to set a breakpoint on a non-executable statement, an invalid breakpoint symbol appears.



(picture 4)

Emulating to Breakpoint

Select main menu “Run”, then choose the submenu “Go”. (When the breakpoint is reached, emulation halts before the instruction at the breakpoint address is executed). When emulating, you can open windows including every source window, disassemble window, mixed source and disassemble window. You can switch the window from one to another.

Stepping Source

This function allows single step of the program in source or assembly mode, including “trace into”, ”step over”, ”run to cursor”, “Go over return”, “trace into continually” and “step over continually”.

View Register

Select main menu “Window”, then choose the submenu “CPU” to open CPU window in which you can view and modify the register. You can also open the Peripheral Window and Bit Window.

View Memory
Memory includes internal Ram, XDATA (external data memory) and Code (program memory). Select main menu “Window”, then choose the submenu “Internal Ram” to open internal Ram window. The XDATA and Code are displayed in “Output” window.

View Variables
Select main menu “Window”, then choose the submenu “Variable” to open the window in which you can watch and modify the parameters and variables when emulating.

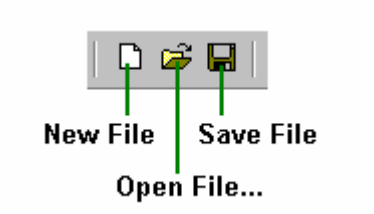
Discussion about Function

Summarization

The goal of this chapter is to introduce you to the TOPICE user interface that provides a quick and convenient way to create, edit a project, build a project, load the object file and download the code to the emulation.

Menu File

From the File menu, you can open source files:
New File
Create the source file saved as a file with extension name “.asm”,”.c” or “.plm”.
Open File...
Open the existing file.
Save File
Save the editing file.
Save File as...
Save the editing file with another name.



(picture 5)

You can also load object file:

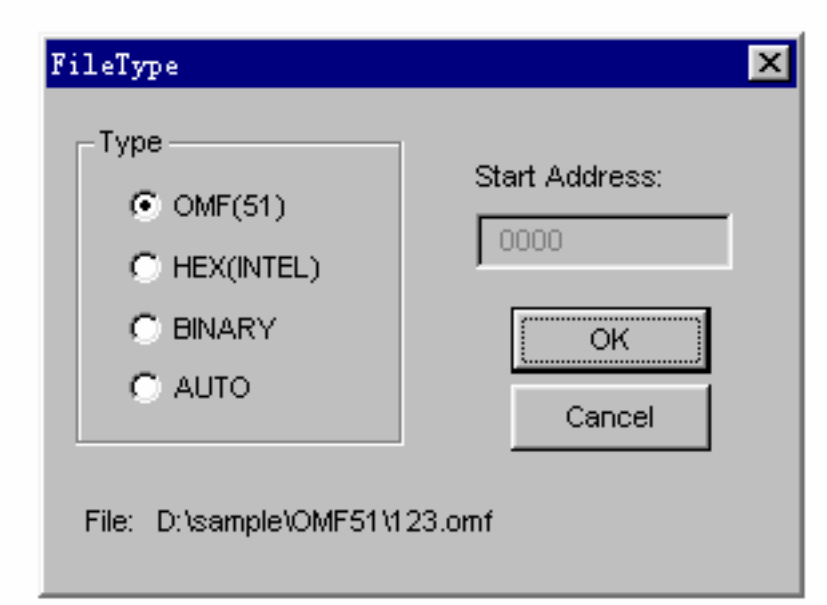
Load as

Load the object file with format:

Intel – OMF51 and Extension OMF51 (Franklin/Keil)

Intel – Hex

Binary



(picture 6)

•**Type**

Choose the format of file.

•**File**

Display the name of the selected file.

•**Start Address**

The start address in the buffer from which code will be loaded.

Save as

Save the code buffer into a file with format:

Intel – Hex

Binary

Exit

Exit the system.

Menu Edit

The Edit menu offers the following commands:

Undo

Reverse previous editing operation.

Redo

Repeats previously reversed editing operation.

Cut

Deletes data from the document and moves it to the clipboard.

Copy

Copies data from the document to the clipboard.

Paste

Pastes data from the clipboard into the document.

Delete

Deletes data from the document.

Select All

Selects the entire text within the active document.

Find

Finds the pattern within the active document.

Find Next

Finds the next occurrence of the pattern within the active document.

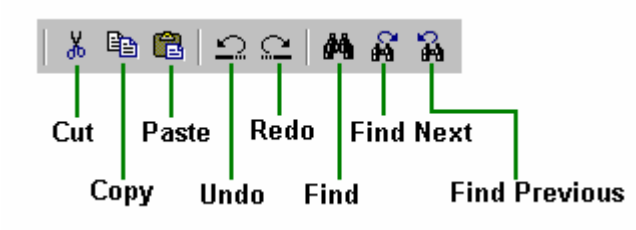
Find Previous

Finds the previous occurrence of the pattern within the active document.

Replace...

Replaces the pattern for other one within the active document.

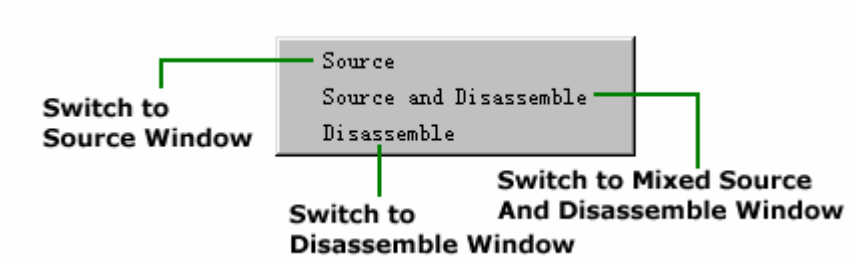
Note: When emulating, any editing operation is disallowed in source window.



(picture 7)

Menu Run

Choose commands in the Run menu to start and stop emulation or to step. You can switch the window between source window, disassemble window, source and disassemble window. When emulating, click the right mouse to pop up the menu to switch window.



(picture 8)

Start Emulation

Emulation is started with following commands followed as:

GO

Start the emulation and halt at a breakpoint address.

Trace into

This function allows single step of the program in source or disassemble window. If a step begins on a source statement containing a function call for which source is available, it steps into the source for that function and stops at the first executable line in the function.

Step over

This function allows single step of the program in source or disassemble window. If a step begins on a source statement containing a function call for which source is available, it steps over the source for that function and stops at the first executable line in the function.

Run to cursor

Start the emulation and halt at the executable line the cursor locates.

Go over Return

Execute from the current Program Counter and halt at the first executable statement or line (in the calling function) after a return.

Run

Start the emulation freely and ignore any breakpoint.

Trace into continually

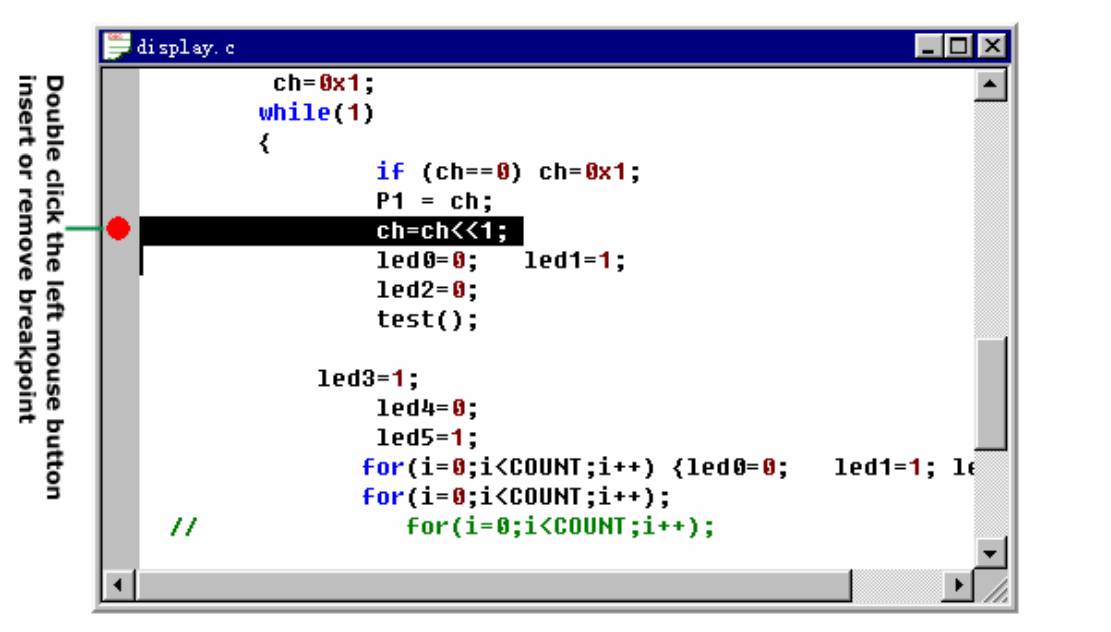
This function allows you to trace into continually the program in source or disassemble window.

Step over continually

This function allows you to step over continually the program in source or disassemble window.

Insert/Remove Breakpoint

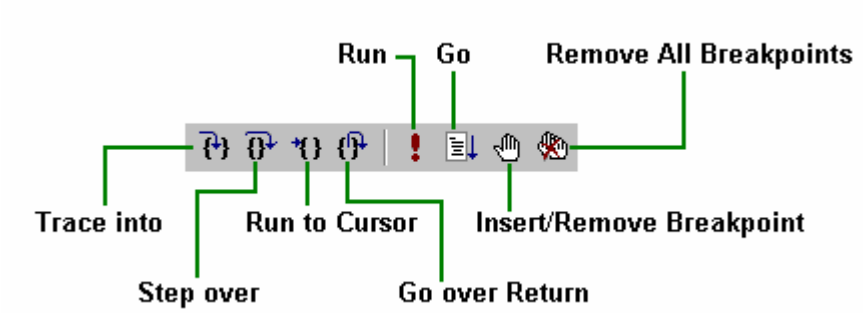
If the line where the cursor is blinking has been set as a breakpoint, this function allows to remove the breakpoint; otherwise, insert a breakpoint. You can also insert/remove a breakpoint with the mouse. Move the mouse to the left of the line and double-click.



(picture 9)

Remove All Breakpoints

Remove all program breakpoints and XDATA breakpoints. Refer the menu “option” and submenu “set break”.



(picture 10)

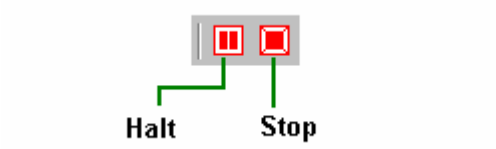
Halt and Stop Emulation

Halt

Halt the emulation and display the position, you can continue emulating from here.

Stop

This ends the emulation.



(picture 11)

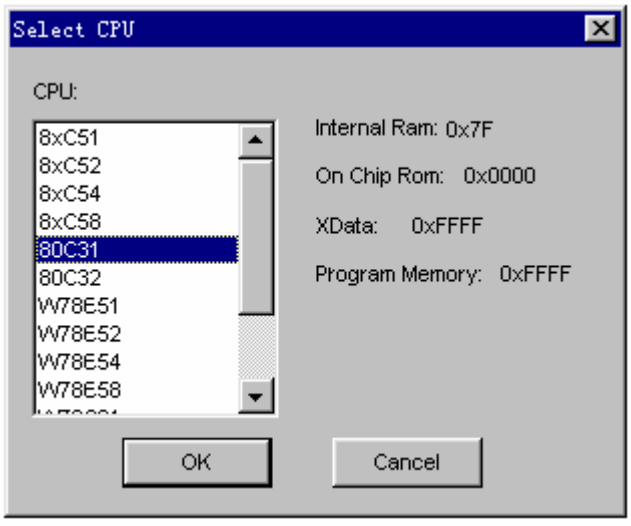
Menu Project

New
Opens the New Project dialog box to create new compiling project.
Open
Displays the Open Project dialog box to open an existing project.
Edit
Displays the Edit Project dialog box to add or remove files in current project.
Close
Close the project, stop the emulation, and close all opened windows.
Build Project
Compile the project according to options, generate a OMF-format file with extension ".omf". If no error, load this file and download the code to the emulator.
Download Code
Download the code to the emulator.

(picture 12)

Menu Option

This function allows you to make setting of software and hardware of emulator.
Select MCU
Select the type of MCU.

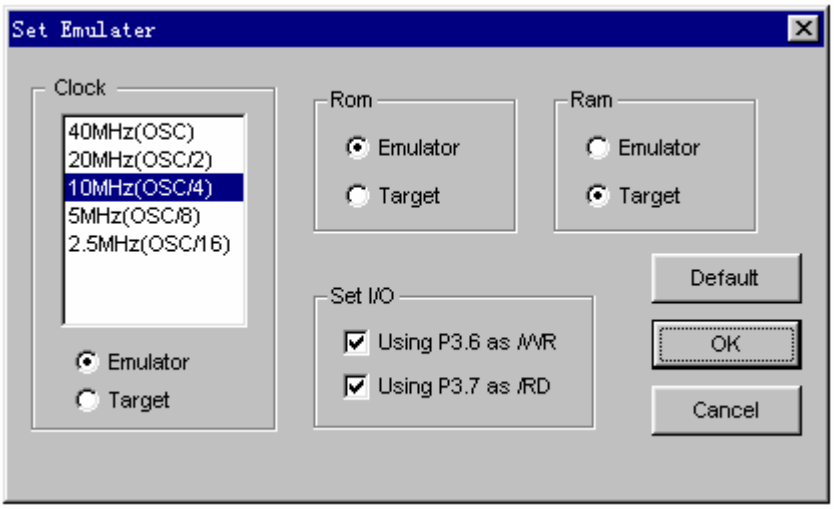


(picture 13)

If the MCU has internal ROM, the P0 and P2 are available as I/O when PC is in the internal program memory and serves as a multiplexed address/data bus when PC is in the external program memory.

Set Emulator

Set the option of emulator.



(picture 14)

●**Clock**

Select the emulator clock.

●**ROM**

Emulator: use the program memory in emulator.

Target: use the program memory in user board.

●**Internal Ram**

Emulator: use the data memory in emulator.

Target: use the data memory in user board.

●**Set I/O**

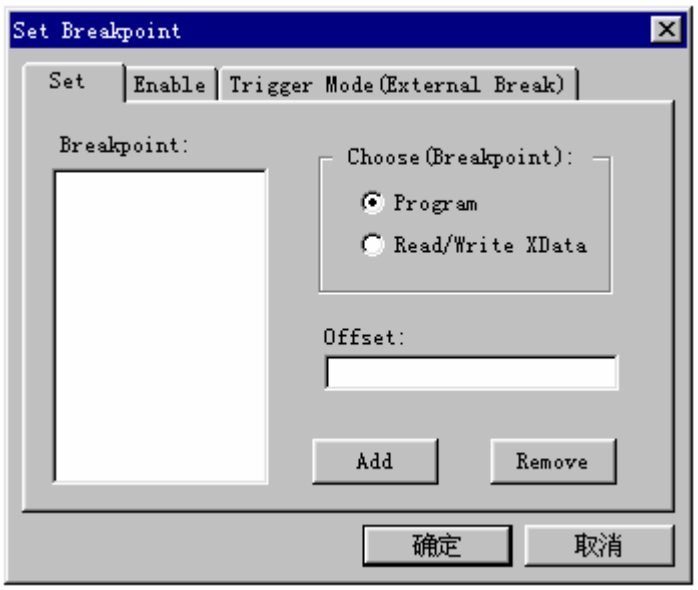
Set P3.6, P3.7

Set Break

Set the option of break

●***Page "Set"***

Add or delete breakpoints of program/XDATA. When the statement reads/writes the XDATA at the address where a breakpoint is set, the emulation halts.



(picture 15)

- **Breakpoint**

List the address of breakpoints.

- **Choose**

Program: list the program breakpoints

Read/Write XDATA: list the XDATA breakpoints

- **Offset**

Type the address

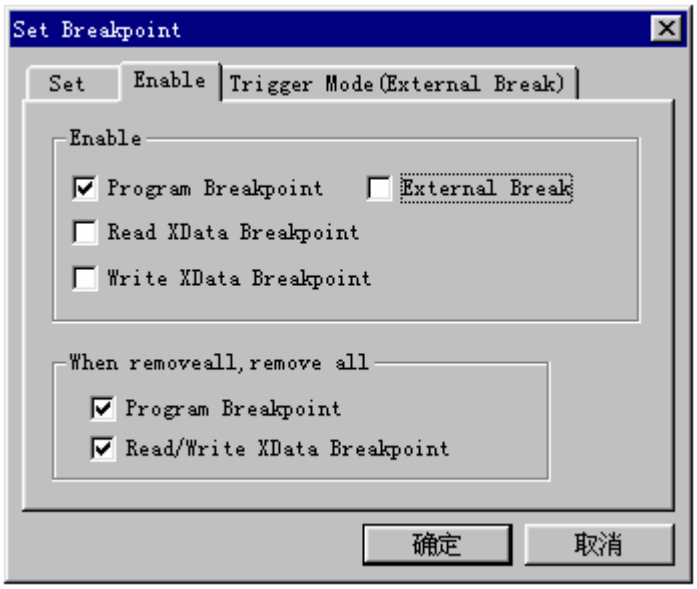
- **ADD**

Set breakpoint at the current address

- **Remove**

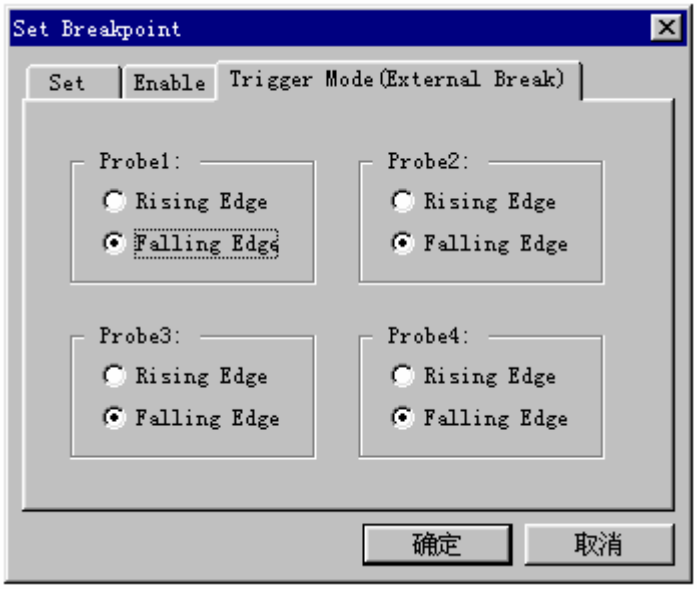
Remove the breakpoint at the highlighted address

- ***Page "Enable"***



(picture 16)

●***Page "Trigger Mode(External Break)"***

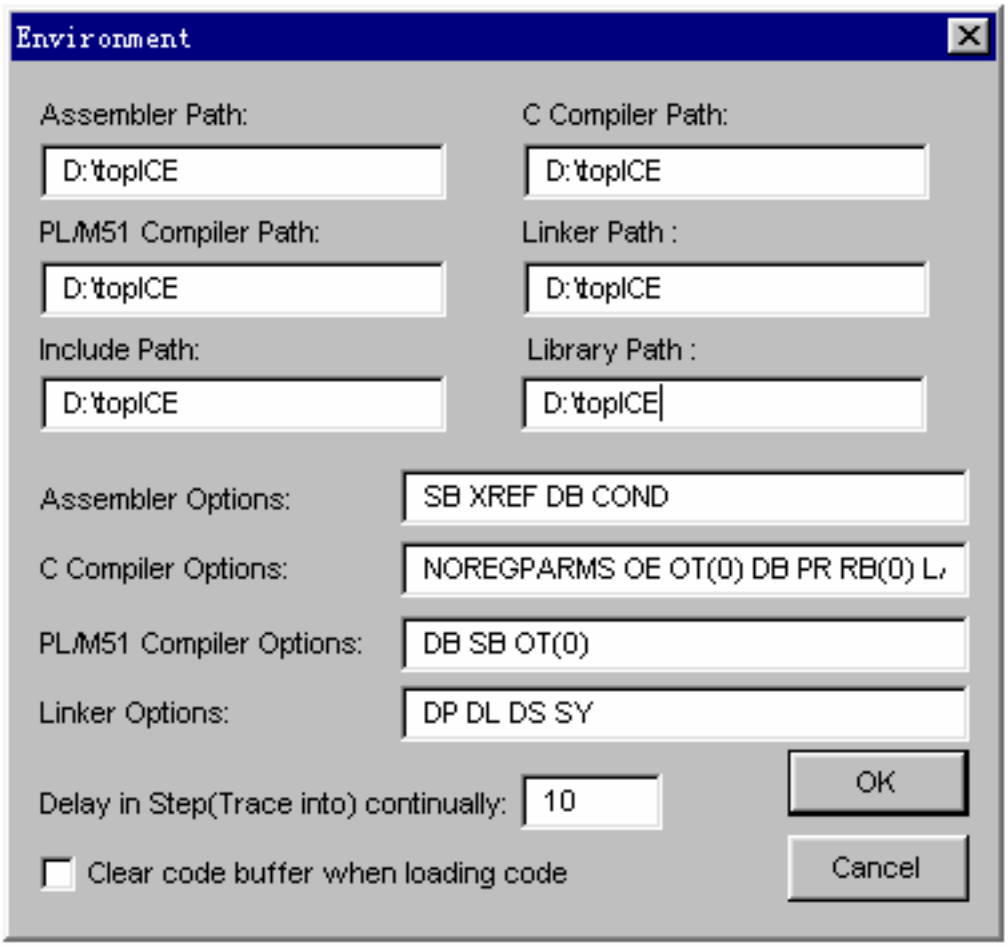


(picture 17)

The emulator has four external break probe. When trigger condition is satified , emulation halts.

Enviroment

Set option of compiler.



(picture 18)

- **Assembler Path**

Type the path that contains the Assembler named “A51.EXE”.

- **C Compiler Path**

Type the path that contains the C Compiler named “C51.EXE”.

- **PL/M51 Compiler Path**

Type the path that contains the PL/M51 Compiler named “PLM51.EXE”.

- **Linker Path**

Type the path that contains the Linker named “L51.EXE”.

- **Include Path**

Type the path that contains include files provided by the standard library.

- **Library Path**

Type the path that contains the compiler library.

- **Assembler Options**

Type the Assembler options separated by space.

- **C Compiler Options**

Type the C Compiler options separated by space.

- **PL/M51 Compiler Options**

Type the PL/M51 Compiler options separated by space.

- **Linker Options**

Type the Linker options separated by space.

●**Delay**

Type the interval between two steps.

●**Clear Code**

Clear the code buffer before loading code.

Refresh XDATA

No useful, you can ignore it.

(picture 19)

Load Target ROM

If target board has an external ROM with program code, you can use this function to load code and disassemble the code, then you can emulate target ROM in disassemble window.

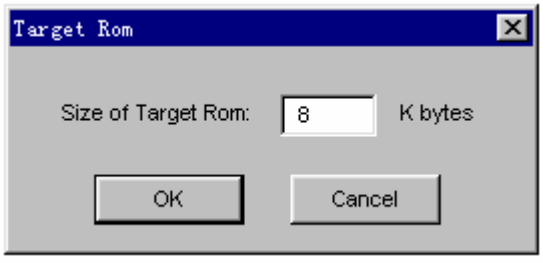
How to use?

Take 89C51 for example.

If user has 8K program code, 4K in the ROM of 89C51 and another 4K in the external ROM. Use this function to load external ROM before emulation,. When emulating, if PC is larger than 4K, you can view in the disassemble window. Follow as:

1. Select the type of CPU: 8xC51
2. Set the option of emulator, select ROM: Target
3. Set environment , uncheck the Box ”Clear code buffer when loading code”
4. If you want to debug the source level for 4K in the ROM of 89C51,you must load the OMF file.(create a project and build)
5. Before emulation, you first load the 4K code in the ROM of 89C51, then use this function to load 4K code in the external ROM.
6. Power on the target board.

Note: Load 4K external code, Size of Target ROM is 4+4=8K



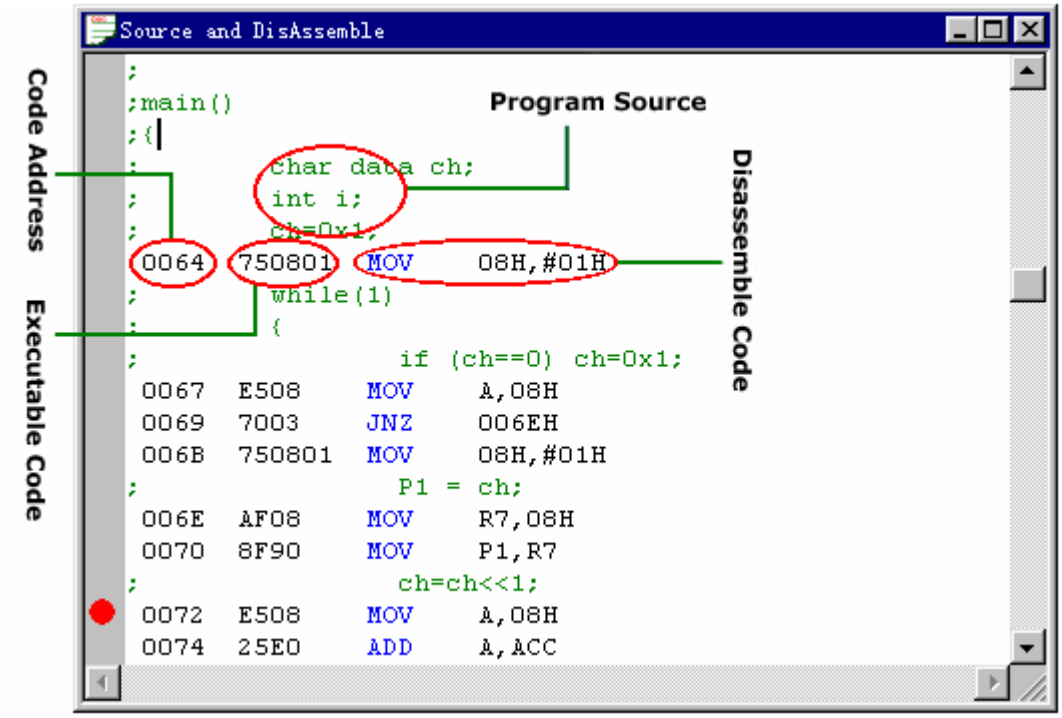
(picture 20)

Work Offline

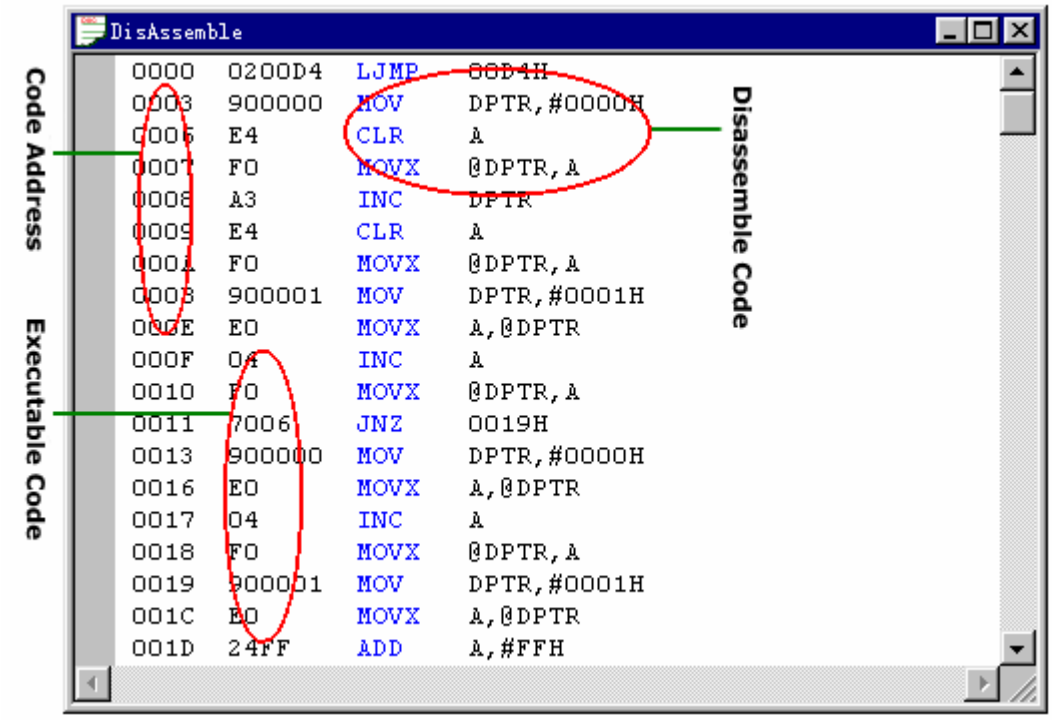
Software simulation only.

Menu View

ToolBar
Show/Hide the toolbar.
StatusBar
Show/Hide the statusbar.
Source and Disassemble
Open the Mixed Source and Disassemble window.



(picture 21)
Disassemble
Open the Disassemble window.

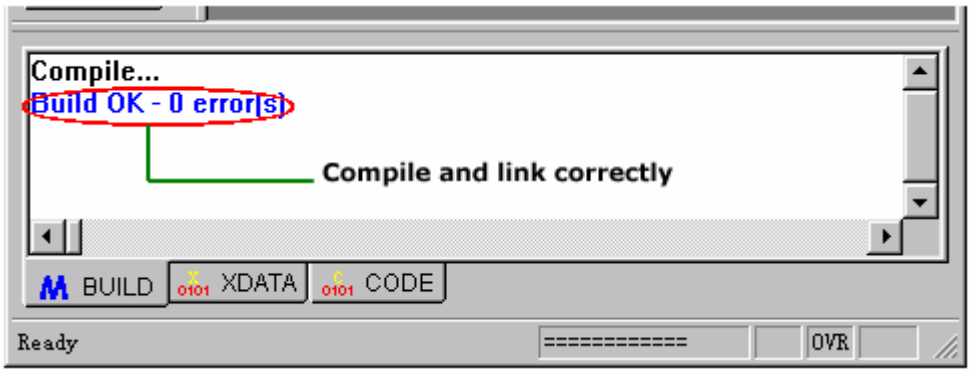


(picture 22)

Output

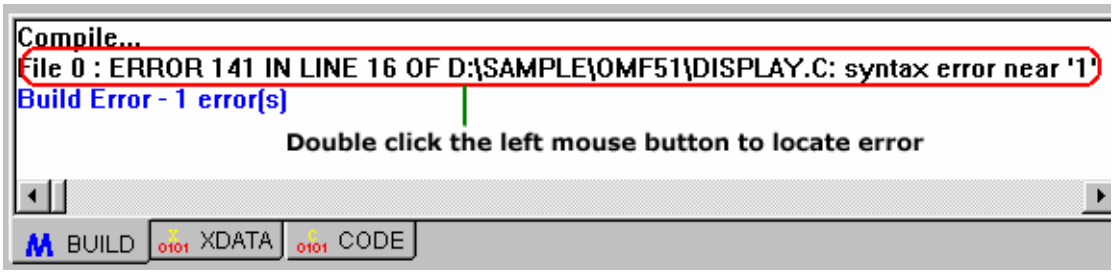
Show/Hide the “Output” window which is composed of Build, XDATA and Code subwindow.s

Build window shows the message of building a project. If no error, the window displays the correct compiling and linking messages. In the meantime, the system generates the OMF file and load it, then download code to emulation if working online.



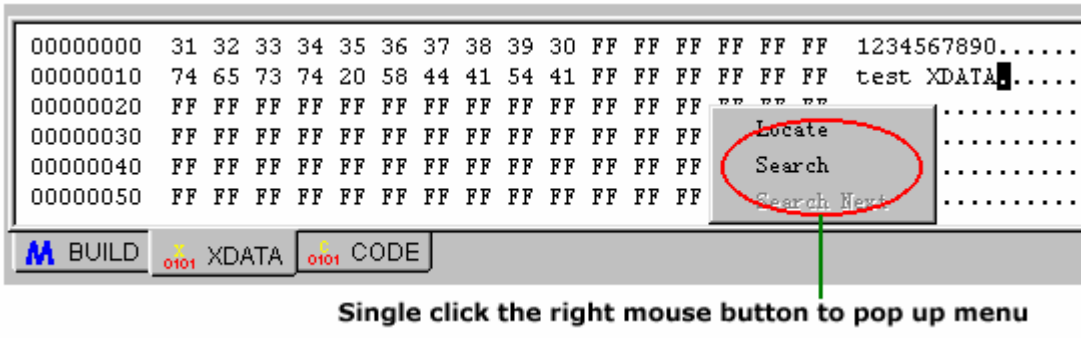
(picture 23)

If the compiled file has error, the window shows error messages and the lines where the error occurs.



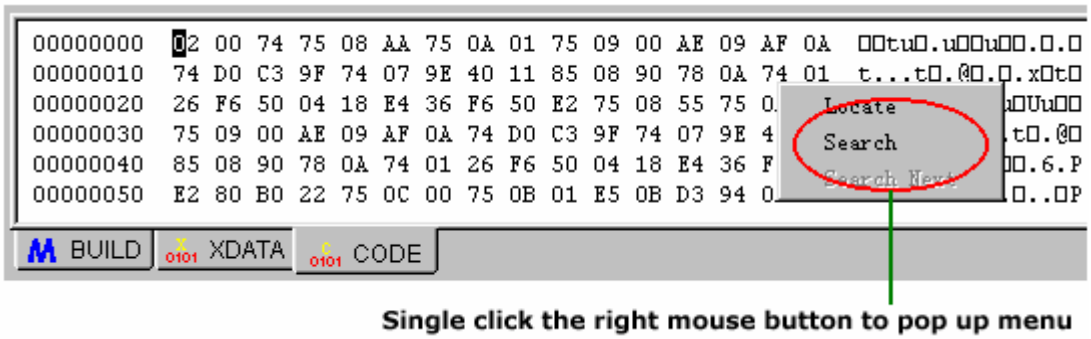
(picture 24)

XDATA window shows external RAM. After the emulation halts, the XDATA refreshes.



(picture 25)

Code window shows the executed code.



(picture 26)

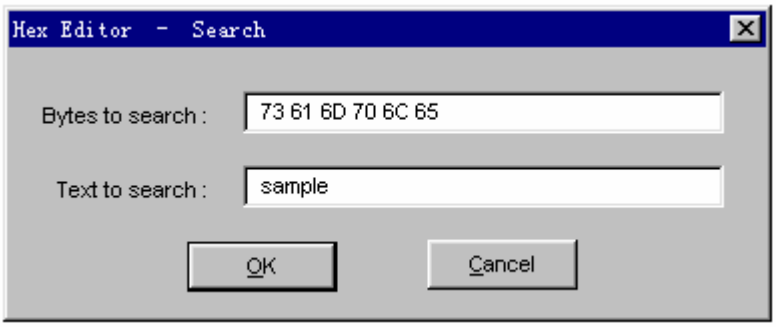
Locate

Locate to the address in the “Locate to Address” box.

(picture 27)

Search

Locate to the address where the string in the “Text to search” can be founded.



(picture 28)

Search Next

Go on searching the address which the string in the “Text to search” box can be founded.

Menu Window

Cascade

Arranges open debugging windows in an overlapping pattern so that the title bar of each window is visible.

Title Horizontally

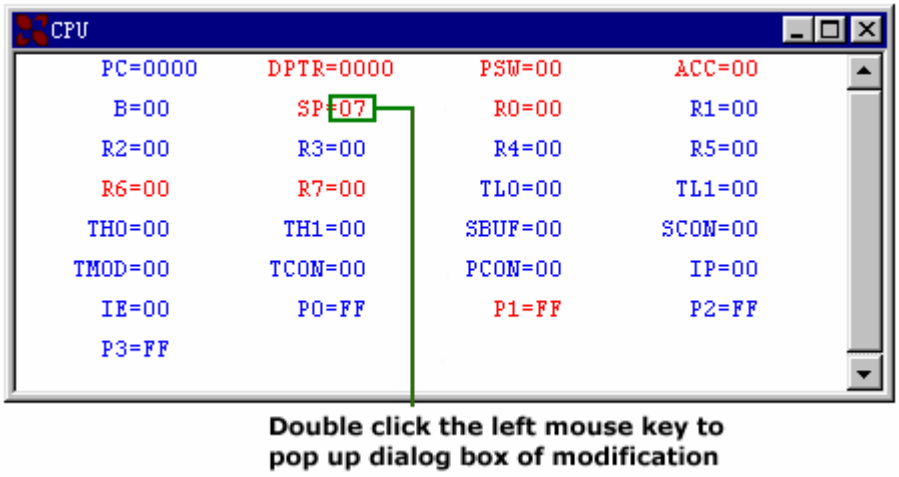
Arranges opened debugging windows side by side so that all windows are visible.

Title Vertically

Arranges opened debugging windows side by side so that all windows are visible.

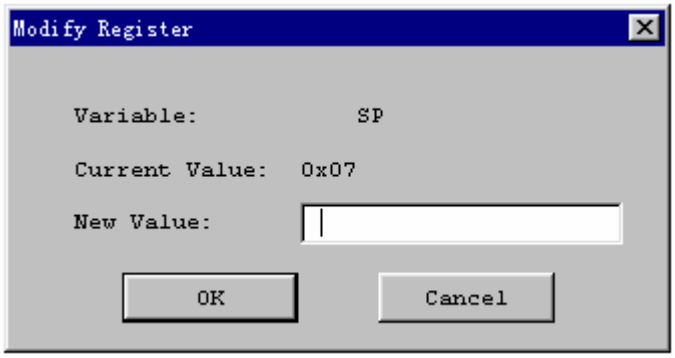
CPU

Open the CPU window in which value of all registers are displayed. Changed values are addressed with red color.



(picture 29)

Modify Dialog Box



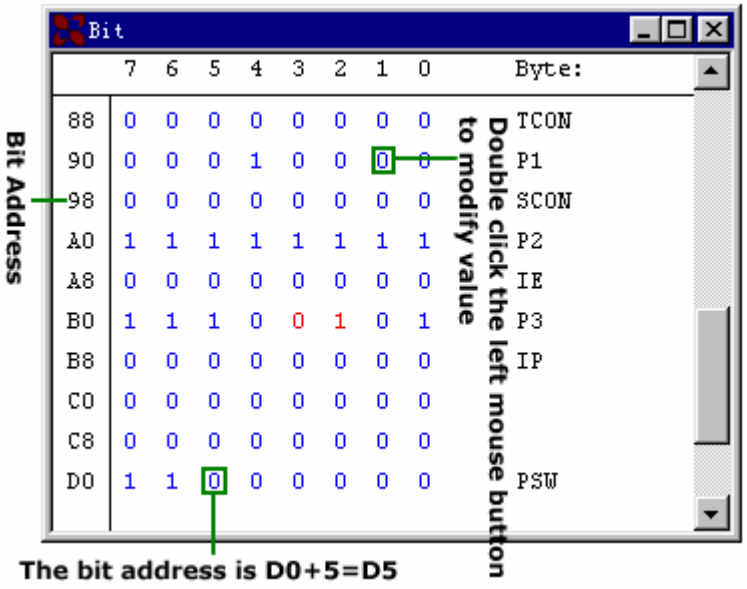
(picture 30)

•***New Value***

Type a new value.

Bit

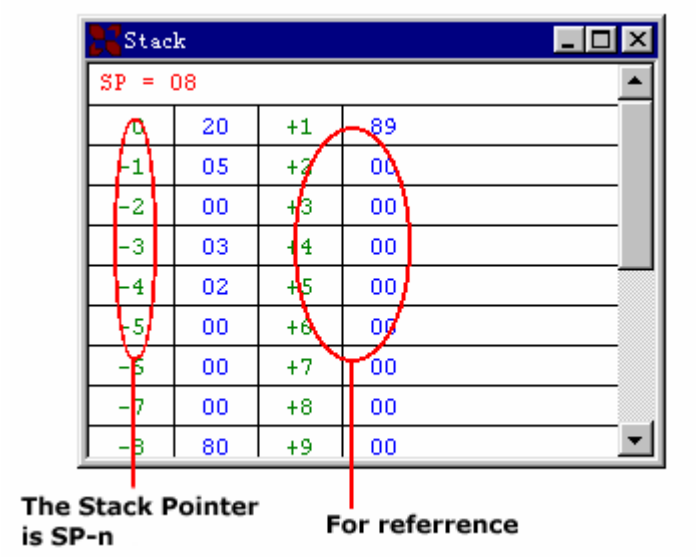
Open the Bit window that displays the value of bit-addressable locations.



(picture 31)

Stack

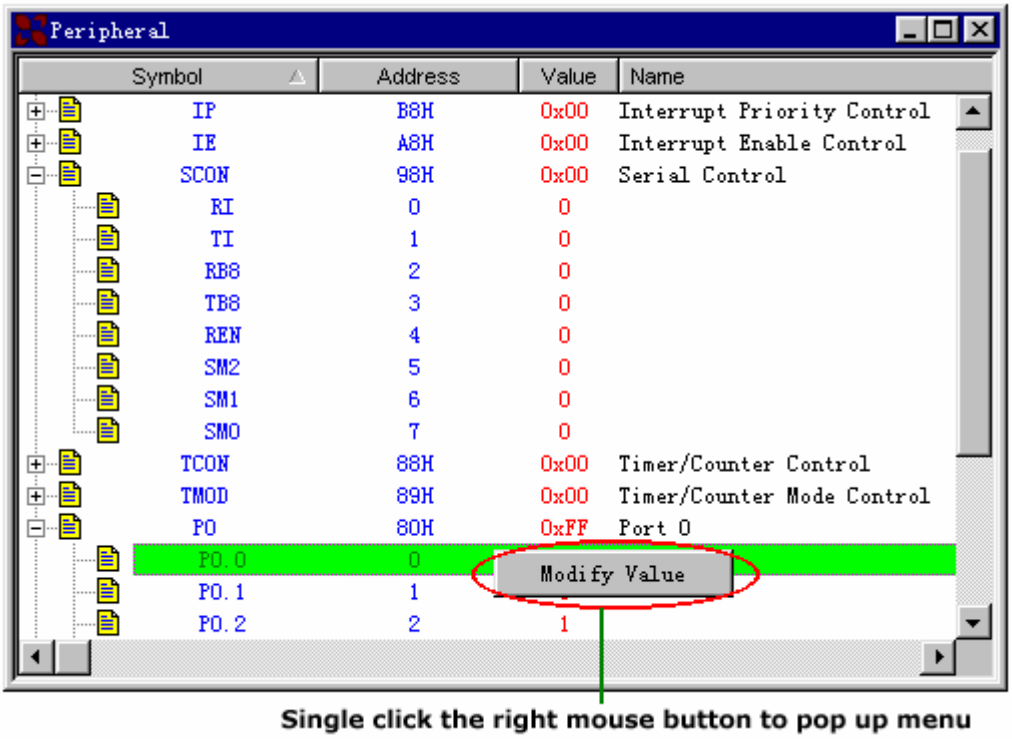
Open stack window.



(picture 32)

Peripheral

Opens the Peripheral window.



(picture 33)

Variable

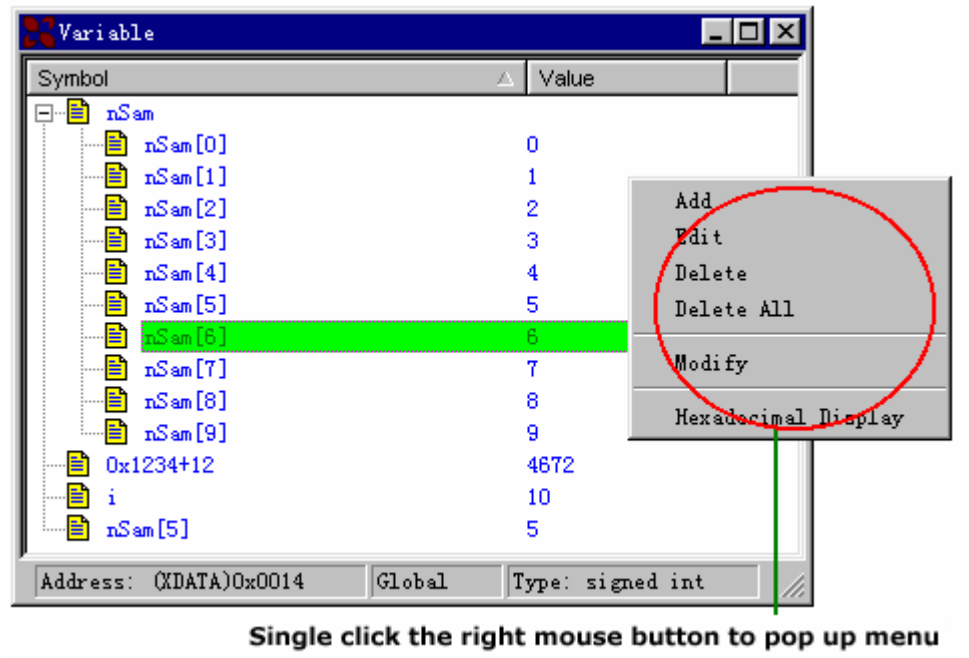
Open the variable window that can be used to inspect or edit variables.

If you want to watch the value of DATA, IDATA and XDATA, you can add an untyped variable as:

D：XXXX DATA Value in address “XXXX”

I：XXXX IDATA Value in address “XXXX”

X：XXXX XDATA Value in address “XXXX”



(picture 34)

- ***Add***

Add a variable which can be compose of C expression, struct or array.

- ***Edit***

Edit a variable. Use it to change the type of variable which is not defined.

- ***Delete***

Delete a variable.

- ***Delete All***

Delete all variables.

- ***Modify***

Modify the value of the variable.

- ***Hexadecimal Display***

The value is displayed in Hexadecimal.

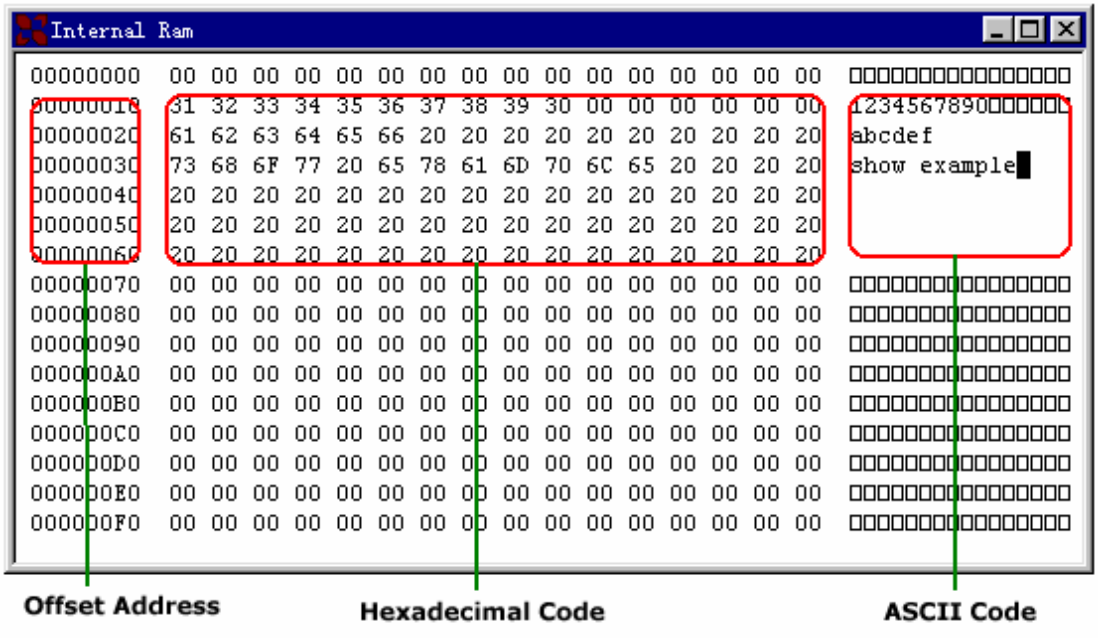
When debugging C51 file, if you want to display value of variable with type “pointer”, you can select the type “array”. For example, for variable “char* Name” you can use “Name[0]” to “Name[n]”, instead of “*Name”.

Internal RAM

Open the internal RAM window.

You can type the key “0-----9” and “A----F” in the HEX region.

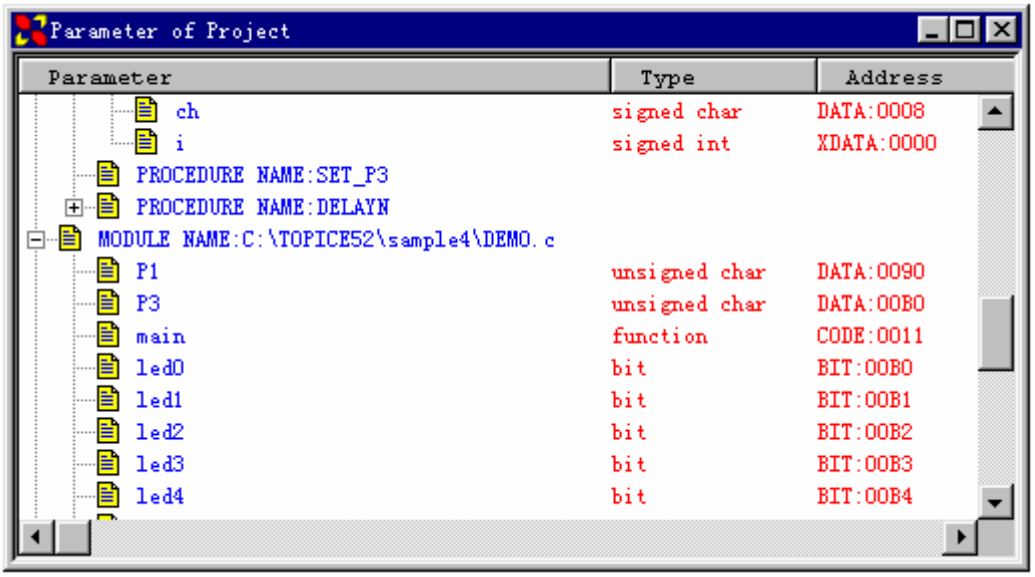
Using key “Tab” switches between two regions.



(picture 35)

Project Parameters

Open the internal Project Parameters window that shows the structure of OMF file. You can use “drag/drop” to add the variable into Variable window.



(picture 36)

Appendix

Error Message

1. Not a project file

Because TOPICE software creates a project file in special format, you must

close the project or quit system normally to save the project file, Otherwise, an error may occurs.

2. There is an error type file in project

There are three correct types of file in project, C51 file with extension name ".c" , assembly file with extension name ".asm" and PL/M51 file with extension name ".plm".

3. Not an OMF51 File

You can create a project including your files and build the project to generate a correct OMF51 file.

4. Invalid type value

Type the C type value. For example:

In Decimal, you can type: 1234, 45, 78, 09, etc.
In Hexadecimal, you can type: 0x1234, 0x6b, 0xabcd, etc.
Float: 12.345, 0.567, 2e-5, 3e+10
Long: 0x12345678L

5. Code size is too large

For example, if the code size is 4.5K while the MCU is AT89C51, this warning will happen when loading the OMF file.

6. Can not load all debug messages

If you debug a PL/M file, the name must be same as the file name.

Example: ***Pdemo***.plm , you must write as:

PDEMO: do;
. . . . . .
end ***PDEMO*;**