



十速科技股份有限公司
tenx technology inc.

TICE99IDE

User Manual

Tenx reserves the right to change or discontinue the manual and online documentation to this product herein to improve reliability, function or design without further notice. Tenx does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. Tenx products are not designed, intended, or authorized for use in life support appliances, devices, or systems. If Buyer purchases or uses tenx products for any such unintended or unauthorized application, Buyer shall indemnify and hold tenx and its officers, employees, subsidiaries, affiliates and distributors harmless against all claims, cost, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use even if such claim alleges that tenx was negligent regarding the design or manufacture of the part.

tenx technology inc.

Rev 1.2, 2020/11/16

AMENDMENT HISTORY

Version	Date	Description
V1.0	Aug, 2011	New release
V1.1	Dec, 2012	<ol style="list-style-type: none"> 1. Add eeprom 2. Upgrade the screen shot 3. Modify the typo 4. Add memory occupied information 5. Add delete watch description 6. Add list debug in program memory 7. Modify make library and add concept flow chart 8. Change 2.7.5 title to "Hex File Converter", and add chip list 9. Add C compiler/assembler user manual 10. Modify the screen shot of new project 11. Modify the table of supported IC list 12. Modify project setting and screen shot, section 4.8 delete the other debug screen shot 13. Modify the description of debug function
V1.2	Nov, 2020	<ol style="list-style-type: none"> 1. Add TM55/56 Assembly project and C/Assembly project create library file instructions. 2. Add TM55/56 Assembly Project and C/Assembly Project creation instructions, Interrupt Function creation instructions, chip search function instruction. 3. Add "Create a New Library" 4. Project Settings add Build tool instructions. 5. Add Appendix B IC List

CONTENTS

AMENDMENT HISTORY	2
1. Introduction	6
1.1 Main menu	7
1.2 Toolbar	9
1.3 Project manager	11
1.4 Function Manager	12
1.5 Memory	13
1.6 Register	14
1.7 Output	15
1.8 Search result	16
1.9 Variables	17
1.10 Watch	18
1.11 ICE Status Stack	19
1.12 Program Memory	20
1.13 Editor	21
1.14 Status Bar	22
2. Main Menu	23
2.1 File	23
2.1.1 New File, New Project, Open File, Open Project	23
2.1.2 Save, Save As, Save Project, Save Project As, Save All, Close, Close All	23
2.1.3 Recent File, Recent Project	23
2.1.4 Exit	24
2.2 Edit	25
2.2.1 Undo, Redo	25
2.2.2 Copy, Cut, Past, Delete, Select All	25
2.2.3 Find, Find Previous, Find Next, Find in Project, Find and Replace, Go to	25
2.3 View	26
2.3.1 Project Manager, Function Viewer, Memory Window	26
2.3.2 Registers Viewer	26
2.3.3 Watch, ICE Status Stack, Program Memory	26
2.3.4 C Variables, ASM F-Plane1, ASM F-Plane2, ASM R-Plane	26
2.3.5 Output, Find and Search Result	27
2.3.6 LCD Editor	27
2.3.7 Toolbar	27
2.4 Project	28
2.4.1 Add New	28

2.4.2 Add Exist File	28
2.4.3 Remove	28
2.4.4 Project Setting	29
2.5 Build	30
2.5.1 Build, Rebuild	30
2.5.2 Build and Download to ICE.....	30
2.5.3 Compile File.....	30
2.5.4 Configuration Debug, Release	30
2.5.5 Make Library.....	31
2.6 Debug	33
2.6.1 Pause	33
2.6.2 Run to Cursor	33
2.6.3 Go, Free Run	33
2.6.4 Single Step, Auto Single Step	33
2.6.5 Step Over, Auto Step Over.....	33
2.6.6 Reset ICE	33
2.6.7 Initialize ICE board	33
2.6.8 Insert / Remove Break Point, Remove All Break Point.....	34
2.6.9 Debug with list file, debug with source file	34
2.7 Tools	35
2.7.1 Language English.....	35
2.7.2 Language Simplified Chinese	35
2.7.3 Language Traditional Chinese	35
2.7.4 QTP Creator	35
2.7.5 Hex File Converter.....	35
2.7.6 Options	36
2.8 Help	37
2.8.1 IDE User Manual	37
2.8.2 C Compiler User Manual	37
2.8.3 Assembler User Manual	37
2.8.4 About.....	37
2.8.5 Update / New Version Check.....	37
3. Toolbar	38
4. Build a Project	39
4.1 Create a New Project.....	39
4.2 Open an Existed Project.....	44
4.3 Create a New Library.....	46
4.4 Project Setting	49
4.5 Editor Options.....	56
4.6 Bookmarks.....	60
4.7 Searching	61

4.7.1Find	61
4.7.2Find in Project.....	62
4.7.3Find and Replace.....	64
4.8 Build a Project.....	65
4.8.1Build File.....	65
4.8.2Build and Download to ICE.....	67
4.8.3Make Library.....	69
4.9 Debug with Project	70
4.10 Switch Language Interface	72
4.11 QTP Creator.....	73
4.12 Hex File Converter	74
4.13 Help.....	76
5. Debug.....	77
6. LCD Editor	79
6.1 LCD Pattern Editor	80
6.2 LCD Attributes	81
6.3 LCD Panel Editor	82
7. Shortcuts	83
Appendix A: Meaning of LED on ICE board	84
Appendix B: IC List.....	85

1. Introduction

The TICE99IDE is a development tool of tenx technology inc., we provide an Integrated Development Environment (IDE) for users to develop their code for tenx chips. It supports TM57 serial chips and TMU serial chips for now. The IDE contains 4 components. They are editor, project manager, compiler and In-Circuit Emulator (ICE) debugger. TICE99IDE provides editor which allows users to edit, modify the code. Project manager manages all files inside the project, including the C and ASM files. Compiler will compile source code that users entered. ICE can simulate ROM program execution and output the result. Before the simulation process, it must connect PC with TICE99 hardware. The following we will focus on these four components of the application window in the IDE, and providing a more detailed explanation.

The main component of IDE window can be divided into four parts to illustrate. The first part is located on the upper left corner of the IDE window, this window contains three functions: project manager, function manager and memory. The upper right part contains only register tab page (F-Plane, R-Plane). The middle of the area is an editor where you can edit and modify the program code on here. On the bottom of the IDE, it can be divided into three parts. The lower left corner is the compiler, linker program which will display messages, which contains compiled outputs, debug results and search results. Second part of the bottom is variables, which contains C variables, and ASM variables. The ASM variables can be divided into four parts, F-Plane1, F-Plane2, R-Plane and EEPROM. If user begins to debug, the contents of all variable of C and ASM will be updated automatically. The lower right corner of debugging message contains three tab pages: watch C variable, ICE status stack, and program memory (Figure 1.1).

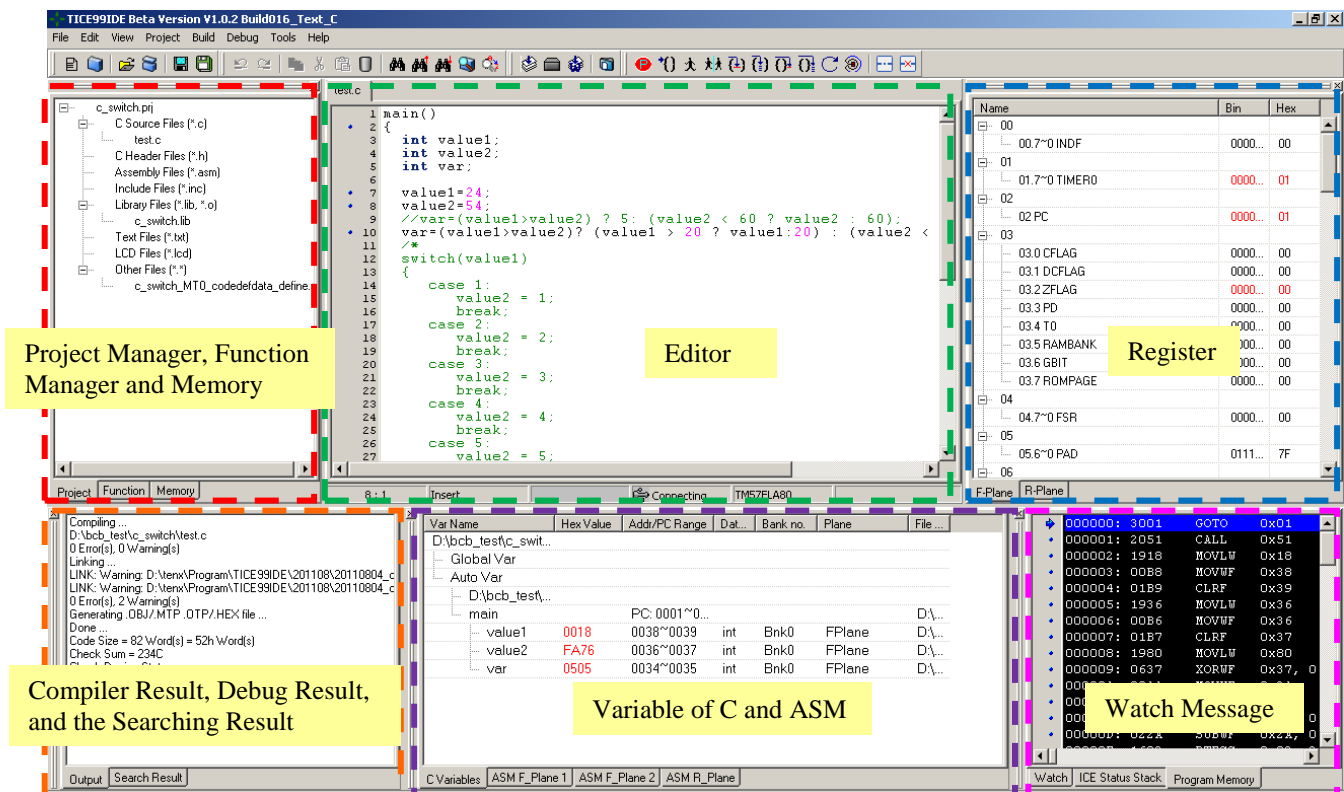


Figure 1.1 TICE99IDE main window

1.1 Main menu

When first executing TICE99IDE, the thing that catches your eyes is the main toolbar which is provided by the IDE - the main menu. Main menu offers all the functions of the IDE. The first level of main menu contains "File", "Edit", "View", "Project", "Build", "Debug", "Tools" and "Help."

In the "File" menu, you can create a new or open an existed project or file. After editing files, you can save then close it; it is the same way when dealing with project. Also, you can close the TICE99IDE by pressing the exit button. You can even open file or project by "recently opened files" and "recently opened project", in the file/project list that are the file/project which you have ever opened before.

"Edit" menu provides the functions which editor requires, including basic functions: copy, paste, cut, delete, select all, undo, redo, and advance functions such as find, find and replace, search in project or opened files, and goto (Figure 1.2). In the "View" menu, you can choose to switch the window to view, including project manager, function viewer, watch, ICE stack status, registers monitoring, output or search result, LCD editor and toolbar option (Figure 1-2).

"Project" menu includes adding file to project, removing file from project and project setting. "Build" is to compile source file which will translate source files to obj files (*.o files) and execution files (*.bin files). User can decide whether it is necessary to download the compile result to ICE (Figure 1-3). "Debug" is a tool to do source code tracing, and to find out where the problem is if it has bug in it. In "Tools", you can switch the language interface dynamically, QTP creator, hex file converter, and set the editor option. In "Help", you can find out user manual including IDE, C compiler and Assembler, about information and update software.

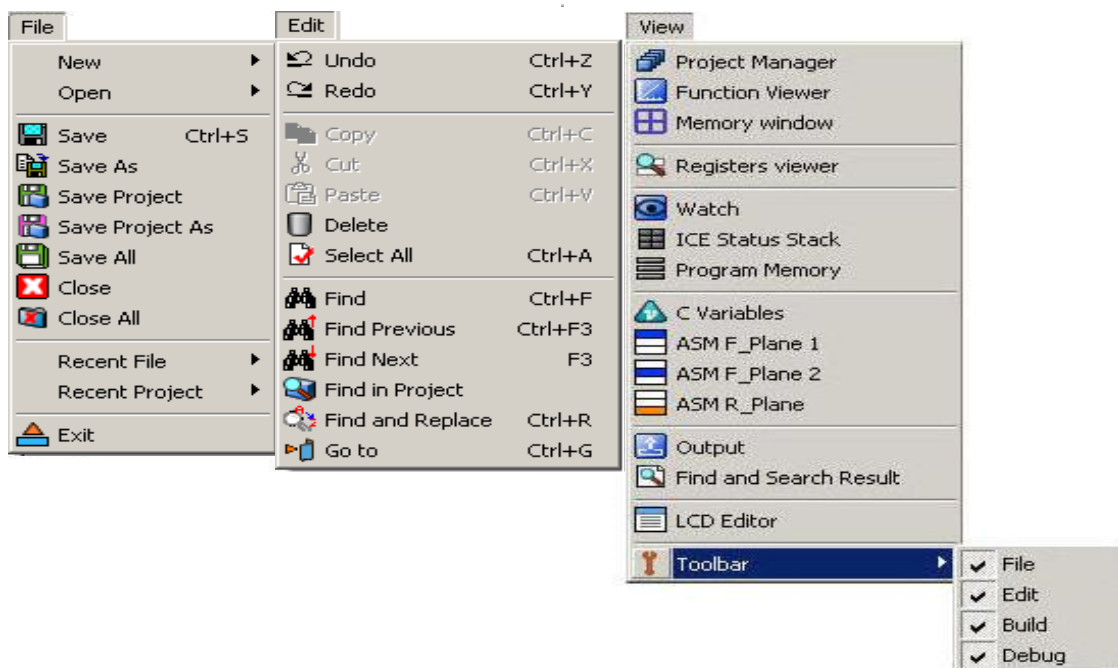


Figure 1.2 main menu -1

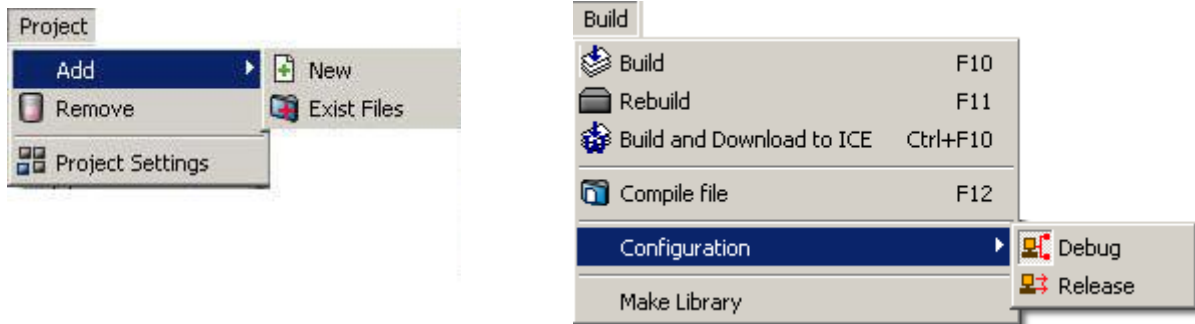


Figure 1.3 main menu -2

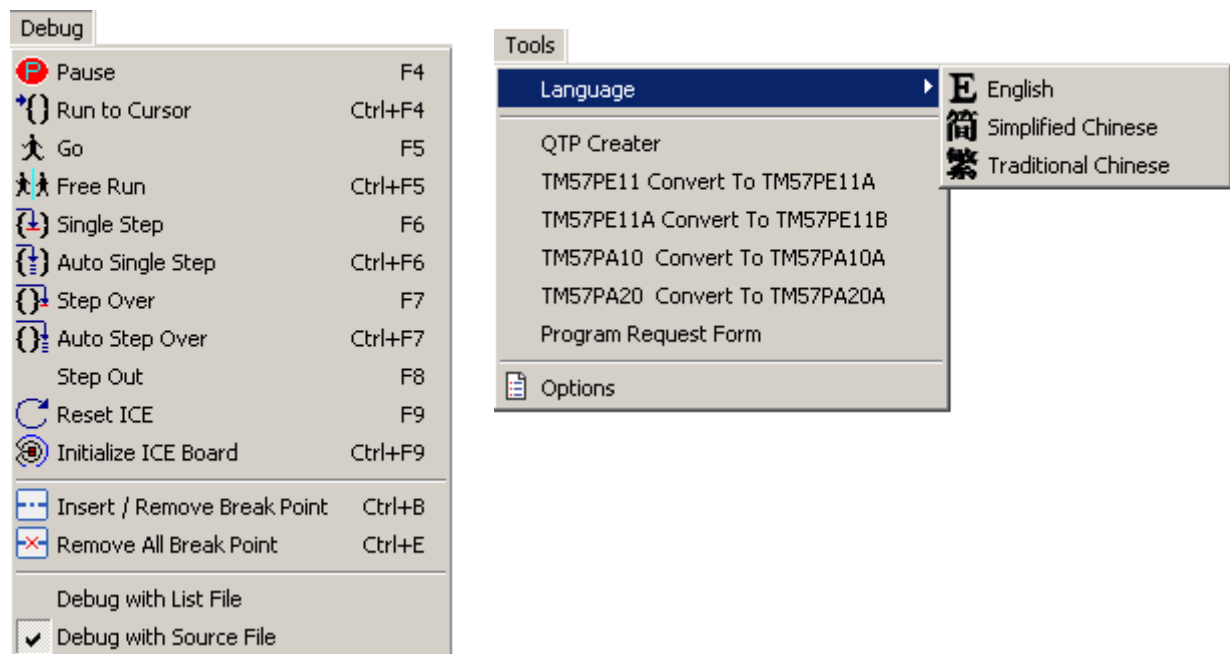


Figure 1.4 main menu -3

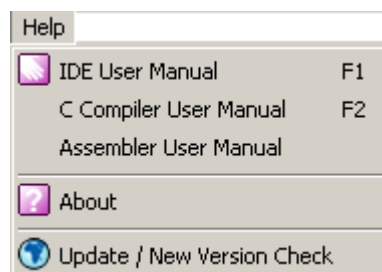








Figure 1.5 main menu -4












1.2 Toolbar





Toolbar is a shortcut to execute the function in main menu, and it contains almost most of the functions of the main menu. User can press button in toolbar rather than select it in main menu.















Figure 1.6 toolbar

File	
	Create a new file.
	Create a new project.
	Open an existed file.
	Open an existed project.
	Save current editing file.
	Save all files.

Edit	
	Undo
	Redo
	Copy the selection text
	Cut the selection text.
	Paste the coping text.
	Delete the selection text.
	Search specific text in current editing file.
	Search specific text from the current position to the beginning of the file.
	Search specific text from the current position to the end of the file.
	Search specific text and display each occurrence in a project at output window.
	Search specific text and replace it with new text.

Build	
	Compile all files in the current project.
	Delete all output files before compiling all files in the current project.
	Compile all files in the current project, and download the output files to ICE.
	Compile current editing file.

Debug	
	Pause debug process.
	Run to the cursor , your program is executed at full speed, then pauses and places the execution point on the line of code containing the cursor.
	Run , run your program, but it will ignore break-point.
	Auto Run , run your program, then pauses on the location of break-point if it has.
	Single Step , the debugger walks through your code one statement or instruction at a time.
	Auto Single Step , the debugger walks automatically through your code one statement or instruction at a time.
	Step over , same as “Single Step”, except that when the execution point is on a function call, the debugger executes the function at full speed and then pauses on the line of code or instruction following the function call.
	Auto step over , execute step over automatically.
	Reset ICE program counter.
	Initial ICE board.
	Add/remove break-point.
	Remove all break-point.

1.3 Project manager

Users can use project management to centrally manage chip properties file, the relevant library files and source files. These files are divided into several groups. They are source code files (.c files), assembly files (.asm files), header files (.h files), include files (.inc files), the library files, text files (.txt files), LCD files (.lcd files) and other files. You can double click the file name, the IDE will switch to corresponding file if it has already opened, or IDE will open the file automatically.

Programmer can create a new file to edit, which will be named as Unit*, * means 1~n serial number. When saving the new file, the “save as dialog” will appear, and let user to save as a .c or other type of file. After saving the file, project manager will **not** classify the file you save into corresponding categories. You must add an existing file to the opened project, so that you can save time in programming and reuse the code as needed.

In the area of project tag, right-click the mouse to show a popup menu dialog. In the popup menu, you can add existing files to a project, remove file from project, set the project and set main file. Main file only supports assembly project. When one file is main file, there will be a ✓ selection icon in front of the file name, which is shown in Figure 1.7.

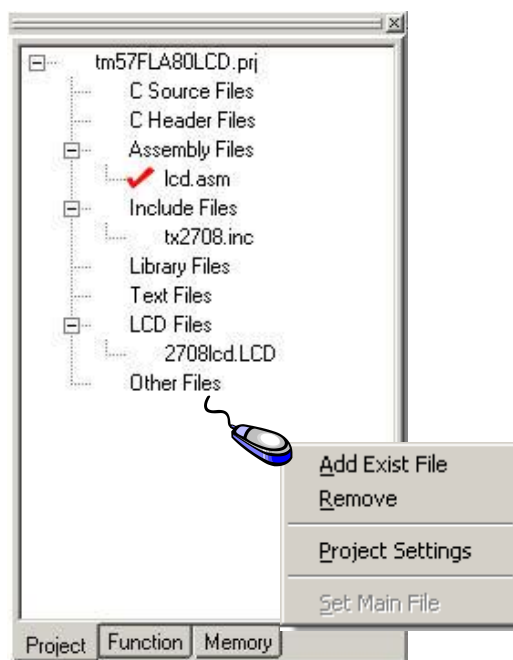


Figure 1.7 Project Manager

1.4 Function Manager

“Function manager” only supports .c files in your project. After you compile a project successfully, the program administrator shall automatically list all the functions which are defined in all .c files of the project. You can double click the function prototype line; it will jump to corresponding line of source code file when the file is opened, or open it and jump to corresponding line if the file is not opened. We now only support the word start with int, char, float, double, void and there’s pointer version. We will support user define structure in future version.

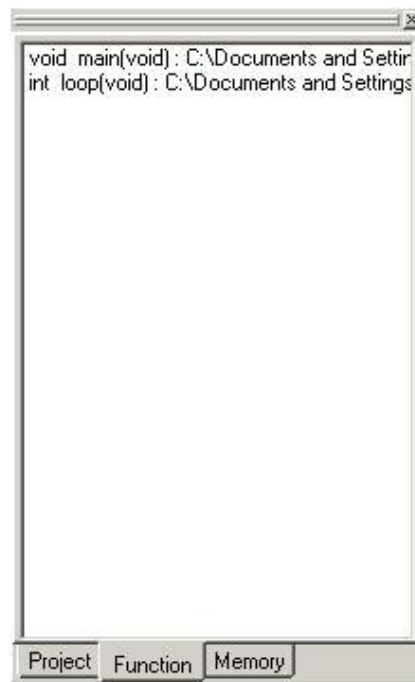


Figure 1.8 Function Manager

1.5 Memory

Memory panel consists of F-plane1, F-plane2, R-plane and EEPROM areas. It will show content of 4 memory areas while debugging. The color of memory content depends on the current step and previous step. If the execution result between two steps is different, the color will be red, otherwise the color is black. The area will show “-“ when the memory space is not used. When you are in debug mode and the ICE is stopped, you can double click the memory address to modify the value of memory directly if the memory address is allowed to be modified. Once you change the value, the value will be updated immediately on the corresponding address in register.

F_Plane 1 --bank0								
00	00	01	01	00	00	FF	FF	00
08	00	00	EA	FE	00	04	00	00
10	00	07	FF	FF	FF	3F	00	00
18	80	00	00	20	00	00	00	00
20	61	62	63	64	36	E0	22	85
28	61	00	80	00	20	03	8A	C7
F_Plane 2 --bank1								
30	68	29	EA	E5	7C	58	AE	07
38	FA	B0	4F	9F	93	E3	04	20
40	FB	28	2C	20	02	1E	91	9D
48	BB	1E	6D	C5	C3	00	7E	4A
50	BB	AF	97	50	63	4C	A4	00
58	CE	40	40	90	5D	0F	48	40
R_Plane								
00	00	00	00	00	00	00	00	FF
08	7F	FF	0F	03	00	00	00	00
10	00	7F	00	00	00	00	3F	00
18	00	34	00	00	00	00	00	00
20	61	62	63	64	67	67	67	00
28	00	20	80	00	00	40	08	10
EEPROM								
00	--	--	--	--	--	--	--	--
08	--	--	--	--	--	--	--	--
10	--	--	--	--	--	--	--	--

Figure 1.9 Memory

1.6 Register

Register panel shows the contents of registers. Register address, name, and the value are listed in tree-style in the window. Similar to memory display, it will be colored with red when content is changed otherwise the color of content is black when it is the same. You can modify the register content during project debug process (if the register is modifiable).

There are three types of presentations to show content of registers. Clicking right mouse to show the popup menu, you can select binary column, hexadecimal column and value column to show. You can choose what types of demonstration you want the register presents. You can collapse or expand the tree node if you only want to see the whole value of the register. Once you collapse the tree node; the whole value will be only shown in hexadecimal column.

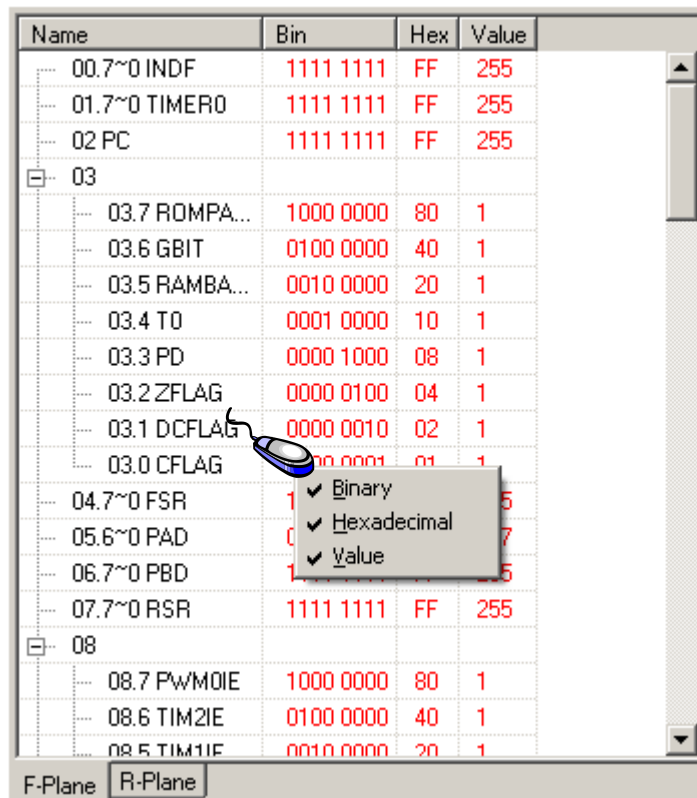


Figure 1.10 Register

1.7 Output

Output panel shows information while compiling, assembling and linking source code. The information could include the information of warning and error. You can double click the warning line or error line; it will jump to the line of corresponding source code.

Memory occupied information for C project

```

LINK: Assignment the start address of RAM variable is 0x24
0 Error(s), 0 Warning(s)
Generating .OBJ/.MTP/.OTP/.HEX file ...
Done ...
Code Size = 88 Word(s) = 58h Word(s)
Check Sum = 16D9
Check Device Status ...
Writing Config Value (0x01...)
Downloading ...
Device Reset Success!

-----
FPlane size: 256 (0x0100)
-----
FPlane Bank0 Used: 53 Bytes (0x0035) Usage: 41%
FPlane Bank0 Remains: 75 Bytes (0x004B)
-----
FPlane Bank1 Used: 36 Bytes (0x0024) Usage: 28%
FPlane Bank1 Remains: 92 Bytes (0x005C)
-----
Detail memory occupied information please check file:D:\bcb_test\2012-12-17_
-----
Download Success!
Detail memory occupied information please check file:D:\bcb_test\2012-12-17 fla80\DEBUGFILE\2012-12-17 fla80.mem
C Variables ASM F_Plane Bank0 ASM F_Plane Bank1 ASM H_Plane
    
```

Var name	Hex Value	Addr/PC Range
D:\bcb_test\2012-...		
Global Var		
Auto Var		
D:\bcb_test\...		
main		PC: 0001~00:
array		0025~0034
[0]	2BBBB8FC	0025~0028
[1]	C243919D	0029~002c
[2]	CD478A03	002d~0030
[3]	05247539	0031~0034
i	76 V	0024~0024

Double Click

```

D:\bcb_test\2012-12-17 fla80\DEBUGFILE\2012-12-17 fla80.mem
1 === Fplane Bank0 ===
2 0 1 2 3 4 5 6 7
3
4 00| V V V V V V V V
5 08| V V V V V V V V
6 10| V V V V V V V V
7 18| V V V V V V V V
8 20| V V V V V V V V
9 28| V V V V V V V V
10 30| V V V V V - - -
11 38| - - - - - - - -
12 40| - - - - - - - -
13 48| - - - - - - - -
14 50| - - - - - - - -
15 58| - - - - - - - -
16 60| - - - - - - - -
17 68| - - - - - - - -
18 70| - - - - - - - -
19 78| - - - - - - - -
20
21
22
23 === Fplane Bank1 ===
24 0 1 2 3 4 5 6 7
25
26 00| V V V V V V V V
27 08| V V V V V V V V
28 10| V V V V V V V V
29 18| V V V V V V V V
30 20| V V V V - - - -
31 28| - - - - - - - -
32 30| - - - - - - - -
33 38| - - - - - - - -
34 40| - - - - - - - -
35 48| - - - - - - - -
36 50| - - - - - - - -
37 58| - - - - - - - -
38 60| - - - - - - - -
39 68| - - - - - - - -
40 70| - - - - - - - -
41 78| - - - - - - - -
42
    
```

For C project, it provides variable memory occupied information which is shown in output panel. Programmer can double click to open the file in the editor. This file lists the detail memory occupied information in the format of 2 dimension matrix.

As shown in Figure 1.11 Output, sign “V” means this register address has been used; sign “-” means free. User can define “-” register address in C program.

Programmer can refer to this file to know which register space has been occupied in the process of linking.

Figure 1.11 Output

1.8 Search result

You can see the result of search in search result panel. There are two kinds of search, search current open files and search files in current project. You can double click the result line to jump to the line of corresponding source code.

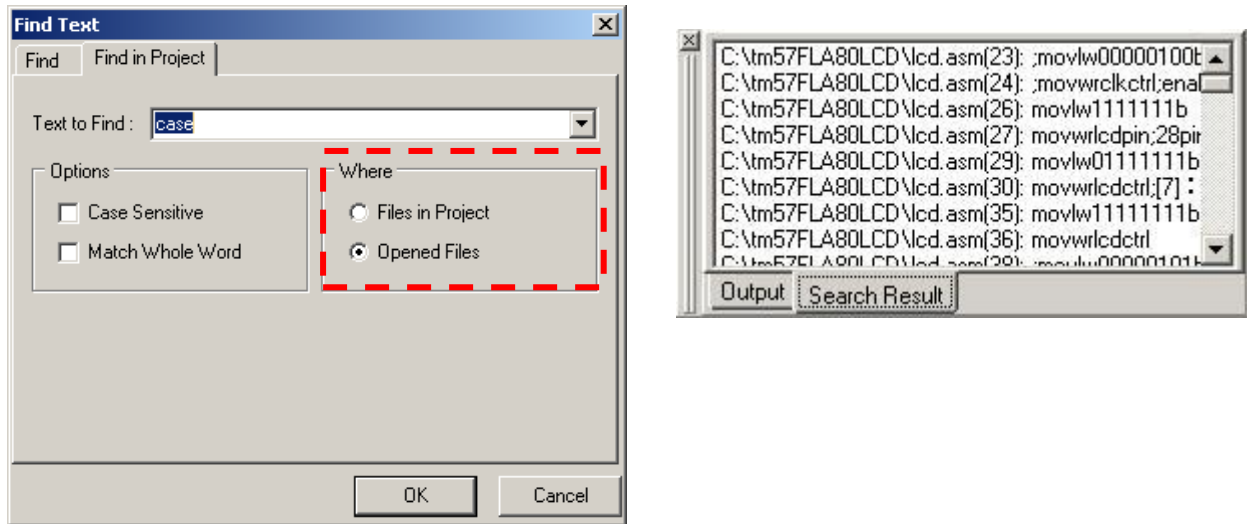


Figure 1.12 Search

1.9 Variables

Variables list all the variables in program, including C and ASM variables. In ASM variable, you can double click the window to select what variables that you want to keep. If you want to see all variables after you selected, just press the button of “Add All”.

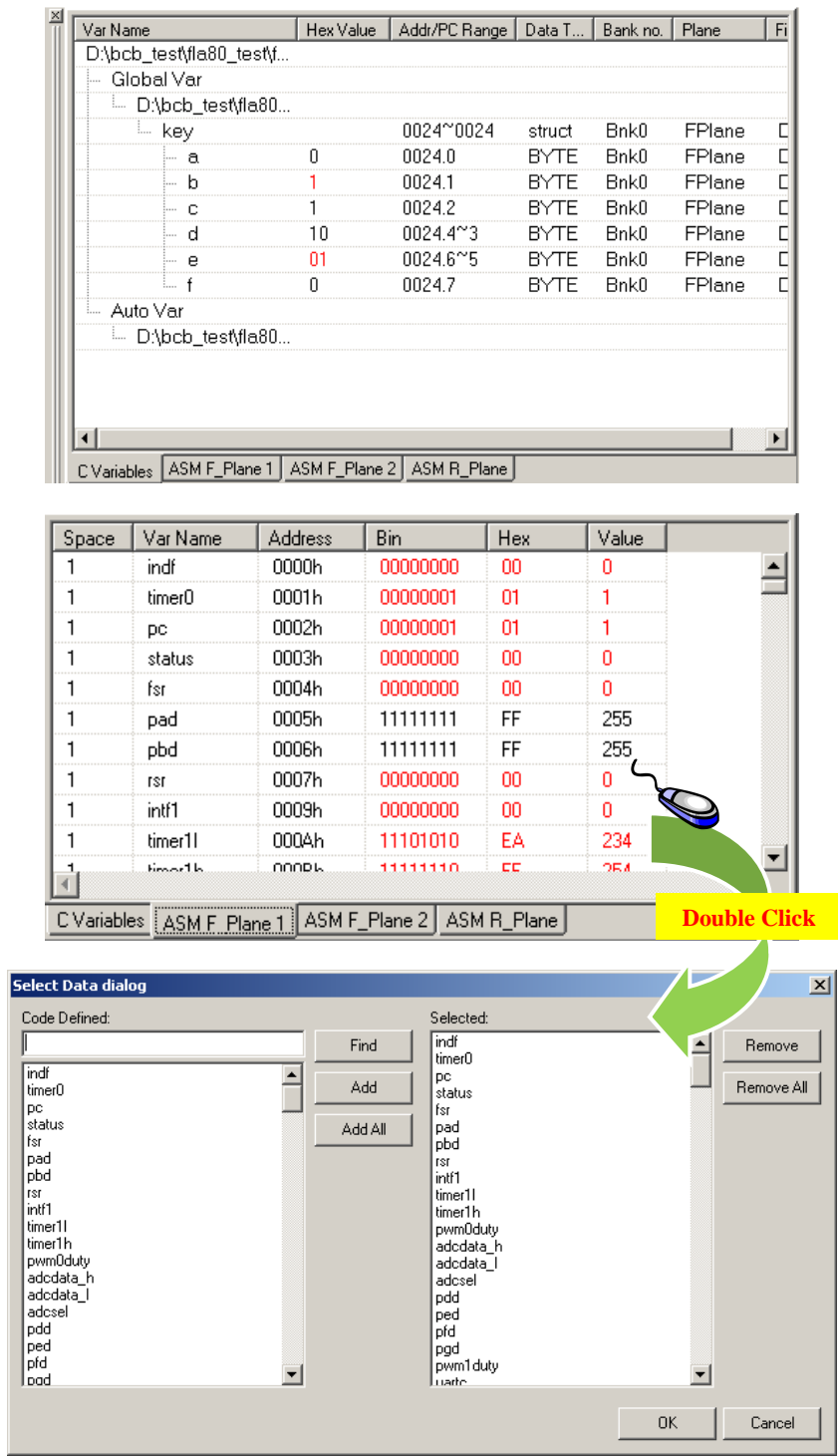


Figure 1.13 Variables

1.10 Watch

The following 3 sections will help you to trace your code while debugging. Watch will show the variables which you want to trace. Especially when you want to monitor the value of a variable or expression while you debug your code, add a watch to the Watch List Window. The Watch List window displays the current value of the watch expression based on the scope of the execution point. Each time your program is executed, the debugger evaluates all the items listed in the Watch List Window and updates their displayed values.

You can set a watch expression in the following ways:

- The easiest way to set a watch is using mouse to drag a term from the Code editor.
- Right-click Watch List Window, IDE will pop-up a watch properties dialog box and users can choose the maintenance action:
 - (1) **Edit Watch:** modify watch.
 - (2) **Add Watch:** add a watch.
 - (3) **Insert Watch:** insert a watch above the line of focused line.
 - (4) **Delete Watch:** delete a watch of focused line.

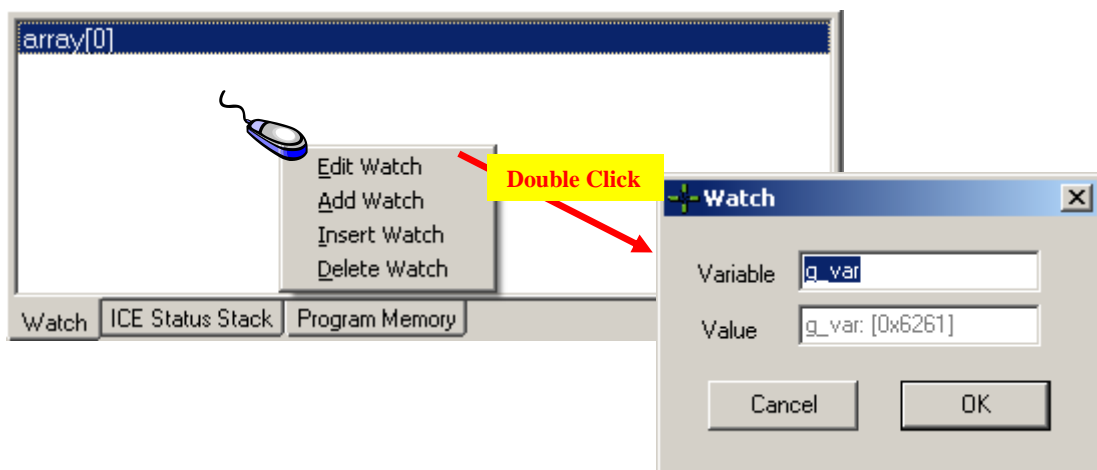


Figure 1.14 Watch

1.11 ICE Status Stack

Stack page shows the subroutine or program information when they have been called. Whenever you call a subroutine or program, the next address of execution point will be added to the stack record before execute the callee. When callee has been return, the address will be pop-up from the stack. Also, it provides the information about ICE status, such as W (Working register), PC (Program Counter), stack over flow and stack under flow.

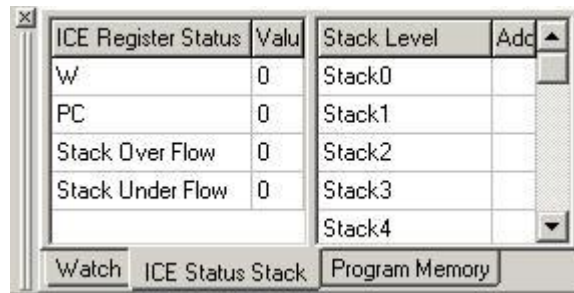


Figure 1.15 ICE Status Stack

In the following example of C code, when the main() program calls subroutine subFoo, the program counter (i.e. next execution address after subFoo: 0x20) will be pushed into the stack (here, Stack0 address is 0x0020). When subFoo is finished and returns to main, it will pop-up the address from the stack (i.e. Stack0 address 0x0020) to facilitate the implementation of the program to continue down.

```

3 main()
4 {
5   int value1;
6   int value2;
7   int var;
8
9   value1=24;
10  value2=54;
11  subFoo(value1);
12
13  value1=33;
14  value2=44;
15
16 }
    
```

0x07

```

void subFoo(int x)
{
  x = x+1;
  return;
}
    
```

Stack Level	Address
Stack0	0020
Stack1	1080
Stack2	0000
Stack3	0000
Stack4	0000
Stack5	0000
Stack6	0000
Stack7	0000

- 00001F: 2007 CALL 0x07
- 000020: 1921 MOVLW 0x21

1.12 Program Memory

Program memory will list the content of ROM code. While compiling process is success, program memory will list: program counter, machine code, mnemonic code and the parameter of operation code.

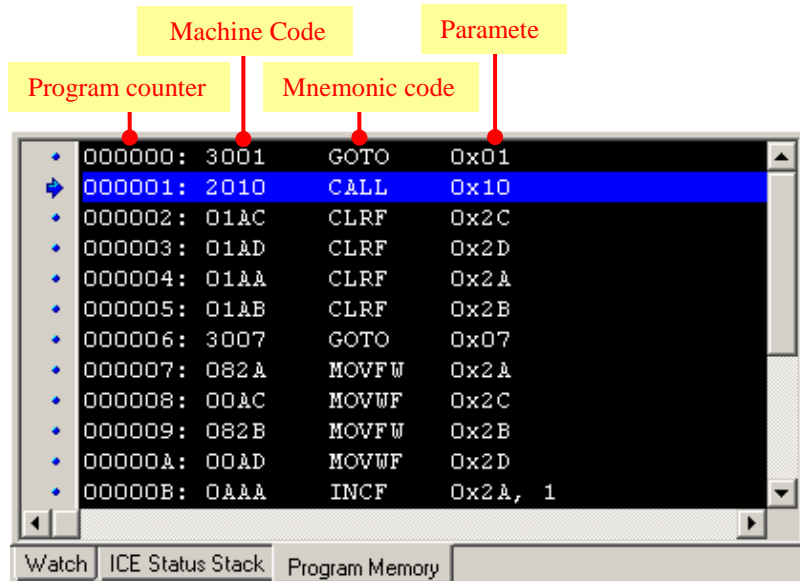
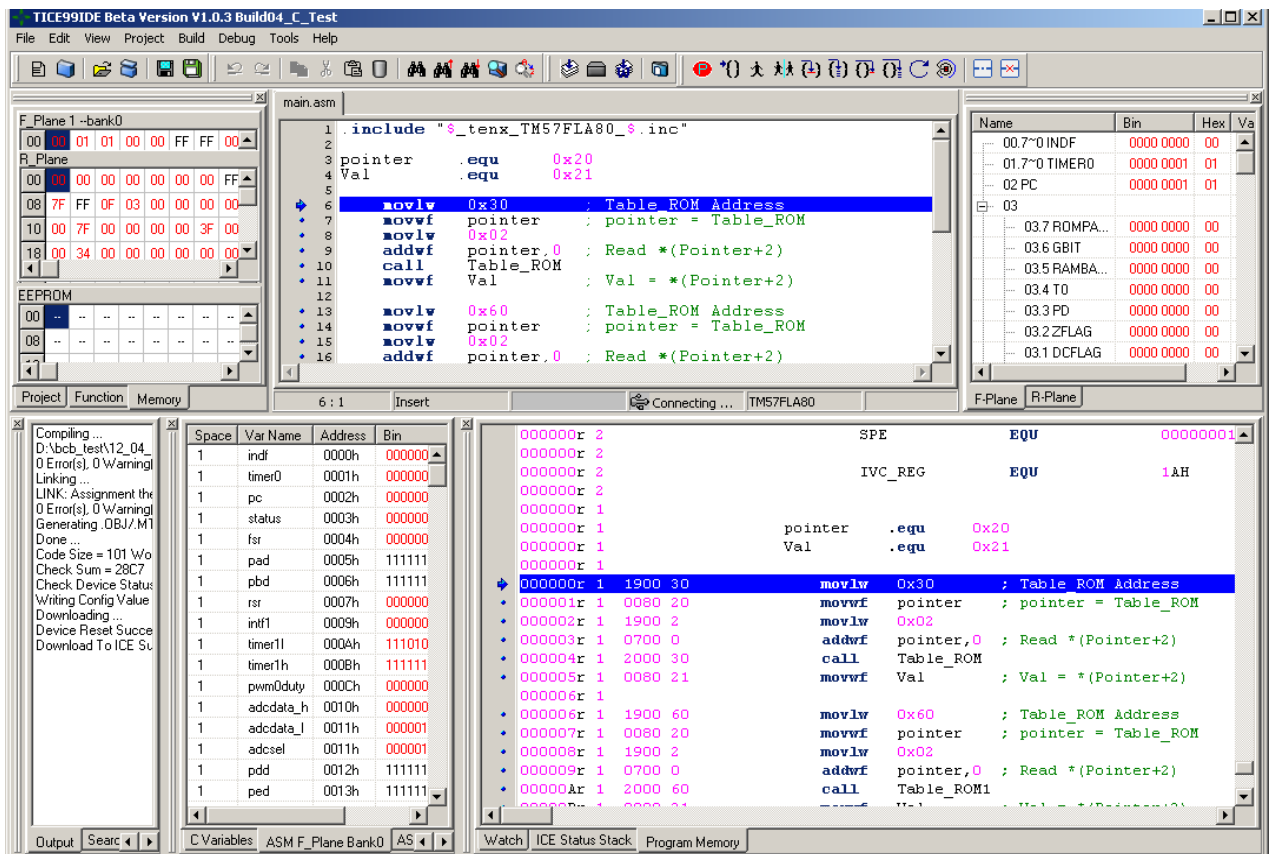


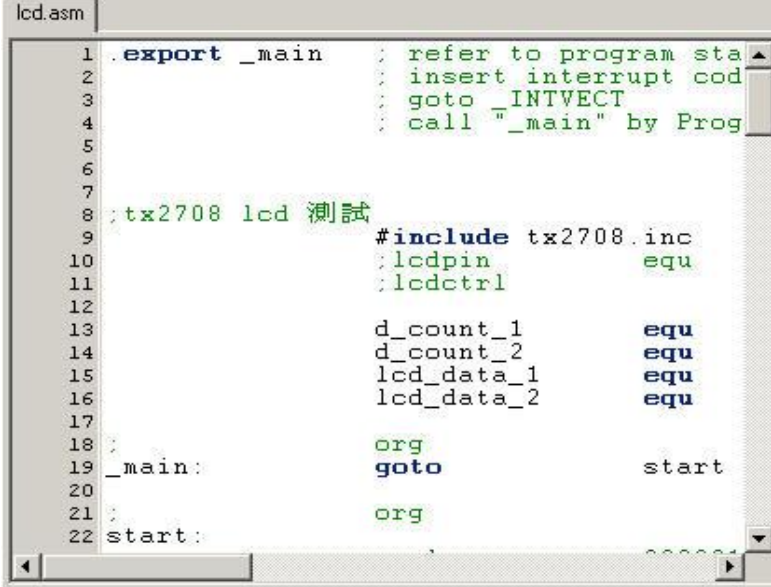
Figure 1.16 Program Memory

But, if there just has one .asm file in assembly project, program memory will show the content of list file as below.



1.13 Editor

Editor provides users with a source code and editor area, the C program file, ASM program file or header file will be input file to the editor.



```
lcd.asm
1 .export _main      : refer to program sta
2                   : insert interrupt cod
3                   : goto _INTVECT
4                   : call "_main" by Prog
5
6
7
8 ;tx2708 lcd 測試
9                   #include tx2708.inc
10                  :lcdpin      equ
11                  :lcdctrl
12
13                  d_count_1    equ
14                  d_count_2    equ
15                  lcd_data_1   equ
16                  lcd_data_2   equ
17
18 ;
19 _main:            goto        start
20
21 ;
22 start:           org
```

Figure 1.17 Editor

1.14 Status Bar

Status bar shows information about editor, debugger and ICE type. The editor information is about caret position and editor mode which is insert or overwrite mode. It also shows debug mode while debugging, the ICE connecting status and chip type.



Figure 1.18 Status Bar

2. Main Menu

Main menu provides a serial of functions which will be useful and convenient for you to use the IDE. It's including File process, editor process, view panel, project management, build a project, debug a project, option of IDE, and help.

2.1 File

In "File" menu, you can create a new file or a project, close a file or a project, and exit program.

2.1.1 New File, New Project, Open File, Open Project

Let's start from File.

"New File" lets developers create a new empty document for editing. The new file is not collected to current project, it is individual. You have to do it by yourself. When you select File | New | project to create a new project, IDE will show a new project window, you can select a specify chip for the new project and assign where the project will be located.

"Open File" can open an existed file and show it on a new editor. "Open Project" opens an existed project and you can easily visualize how all your project files are related in the Project Manager page. If there are some files in project are opened then project is closed, these file will be reopened automatically when opening the project next time.

2.1.2 Save, Save As, Save Project, Save Project As, Save All, Close, Close All

"Save" can save current editing file. If the opening filename is not assigned, like `Unit*`, the "Save As" dialog will be shown and then you can type a filename you want and press yes button to save it. "Save As" let user save current opening file to another filename. However, if the file is in project, it will not change the filename in the project; it only saves a copy file to disk. "Save Project" is saving current project that you opened. If you want to change to another project name, you can use "Save Project As". "Save All" will save all opened files, including the file which is not in project. "Close" can close current opened file. If you want to close the project, please press "Close All" button then it will close current project, including all the files which have been opened.

2.1.3 Recent File, Recent Project

"Recent File" lists the recent opened 4 files at most; "Recent Project" stores the recent 10 projects at most. By clicking recent item, it will open corresponding file if the file is not opened or switch to an opened file. You can also open project that you recently opened from "Recent Project", which will be the same as "Open Project".

2.1.4 Exit

Exit the IDE program.

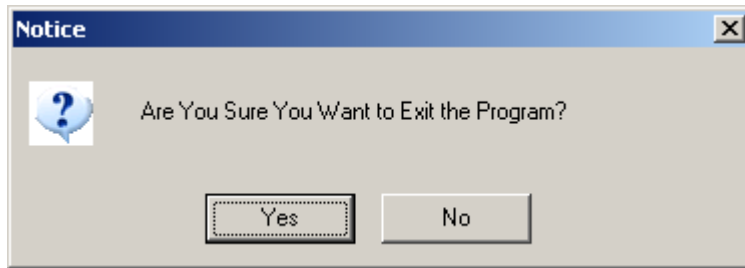


Figure 2.1 Exit notice dialog

2.2 Edit

In “Edit” menu, you can edit your code more quickly by edit functions, they are *Copy, Cut, Paste, Delete* as well as *Select All*. If you want to recover back from a typo, you can press “Undo”. Once you change your mind from undo, you can redo it. “Edit” menu also provides a serial of search function, including *Find, Find Previous, Find Next, Find in Project, and Search then replace*.

2.2.1 Undo, Redo

When you encounter a typo, you can use “Undo” to recover back the step that you typed. Once you change your mind, you can press “Redo” to do the action again.

2.2.2 Copy, Cut, Past, Delete, Select All

Copy, Cut, Past, Delete and Select All can let programmers type their code more quickly.

2.2.3 Find, Find Previous, Find Next, Find in Project, Find and Replace, Go to

You can use “Find” when you want to search a word or a letter in a file. After you find one in you searching, you can use “Find Previous” or “Find Next” to reach previous or next one target word directly. If you want to search word or letter in a project, please use “Find in Project”. “Find and Replace” let you find out the word that you want to replace. “Go to” will move cursor to the line you specified.

2.3 View

There are many panels in TICE99, including Project Manager, Function Viewer, Memory Window, Registers Viewer, Output, Find and Search Result, C Variables, ASM F-Plane1, ASM F-Plane2, ASM R-Plane, Watch for C code, ICE Status stack, Program memory, and toolbars. Those are helpful for programmer to get information more clearly.

2.3.1 Project Manager, Function Viewer, Memory Window

Project Manager, Function Viewer and Memory are in the same panel. To switch which tab page which you want to see, click the menu of project manager, function viewer or memory. With the “Project Manager”, you can easily visualize how all your project files are related. Also, you can select any file to display, right-click, and perform various project management tasks, such as adding or removing files; “Function Viewer” lists all function declarations in all C files; “Memory Window” contains the content for F-Planes, R-Plane and EEPROM.

2.3.2 Registers Viewer

Registers viewer contains two tab pages. One is F-Plane, and the other is R-Plane. You can switch tab to see the contents. If registers viewer is disappeared, click the menu item View|Registers Viewer to show the panel. The contents of register viewer will be shown when source code has been built successfully and the output is downloaded to ICE automatically.

2.3.3 Watch, ICE Status Stack, Program Memory

“Watch” will list C variables which user selected from C Variable tab or drag from source code editor. “ICE Status Stack” displays ICE information, including Working register, Program Counter (PC), stack over flow, stack under flow, and contents of stack. “Program Memory” shows up the list of machine code. To view this panel, click the menu item.

2.3.4 C Variables, ASM F-Plane1, ASM F-Plane2, ASM R-Plane

User can see the detail information of C Variables, ASM F-Plane1, ASM F-Plane2 and ASM R-Plane which list all variables of source code. When successfully compiled the source code and start debugging, this window's content value of each variable will be updated automatically. The color of variable text will be changed when the result is changed between the two actions. If it does not show up, click the menu View|C Variables or View|ASM X_Plane to let this visible.

2.3.5 Output, Find and Search Result

“Output” shows the result of compiling, assembling, and linking process. If there are any error or warning messages, it will be listed on the output panel, you can double click the line, IDE will jump to the corresponding line in editor. “Output” panel shows the information about memory occupied of variable in C project. “Find and Search Result” jumps to corresponding line when the keyword you search has been found. Click the menu to let the panel visible.

2.3.6 LCD Editor

LCD Editor is only for the chip which supports LCD function. The IDE will enable this menu item when chip supports LCD, and disable when does not support.

2.3.7 Toolbar

Toolbar controls 4 sub-toolbars to show up or not. They are *File*, *Edit*, *Build* and *Debug*. When the item is checked, the toolbar is visible. Or you can uncheck it to hide the toolbar.

2.4 Project

"Project" provides a centralized file management mechanism. It performs various project management tasks, such as opening, adding or removing files from a project.

2.4.1 Add New

When programmer clicks this menu item Project|Add|New, the IDE will create an empty document and Unit*(1~n) will be named to it. This is one way to add file to project. When you save the file, it will show a "save as dialog" to let you save another filename for this file, and the project manager will be changed too.

2.4.2 Add Exist File

The other way to add files to a project is adding an existed file. You can click this item to choice the existed file and project manager will add the file you selected in project. When you click this menu item Project|Add|Exist Files, IDE will open an open dialog to let you select the file you want to add in project.

2.4.3 Remove

There are two ways to remove files from the project: (1) tap out on the Project|Remove menu item and IDE will pop-up a project files window (Figure 2-2), allows users to remove files, or (2) within the tree of "project management", click the file item which will be removed, then right-click, in the pop-up menu, select "Remove" menu item. It only removes the specified files from the project tree, but the file is not actually deleted from the file storage directory.

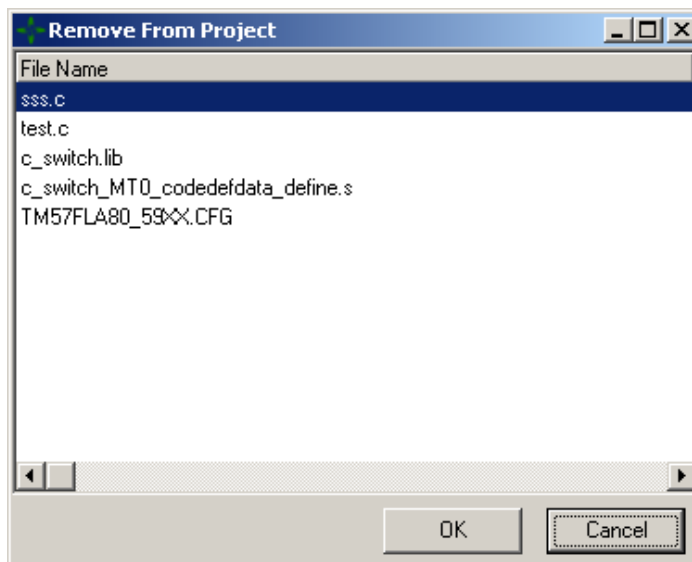


Figure 2.2 Remove from project dialog

2.4.4 Project Setting

There are two ways to set project properties: (1) Click on the menu bar Project|Project Settings menu item, or (2) within the tree of "project management", then right-click in the area and in the pop-up menu, select "Project Settings". Project settings include IC configuration settings, select the library directory and EV settings. The configuration settings will be downloaded and write to the ICE before the program codes does. The configuration settings can be set as different types of projects with ICE. The configuration tab will show the configuring value when you change the setting of configuration. Another "Project Settings" page is a "directory" to specify the directory path that can be found in the compilation and assembler which uses the directory path to find the header files or library files. EV option contains two main functions, one is to calculate the best IRCF for chip, and the other is to switch the ICE voltage.

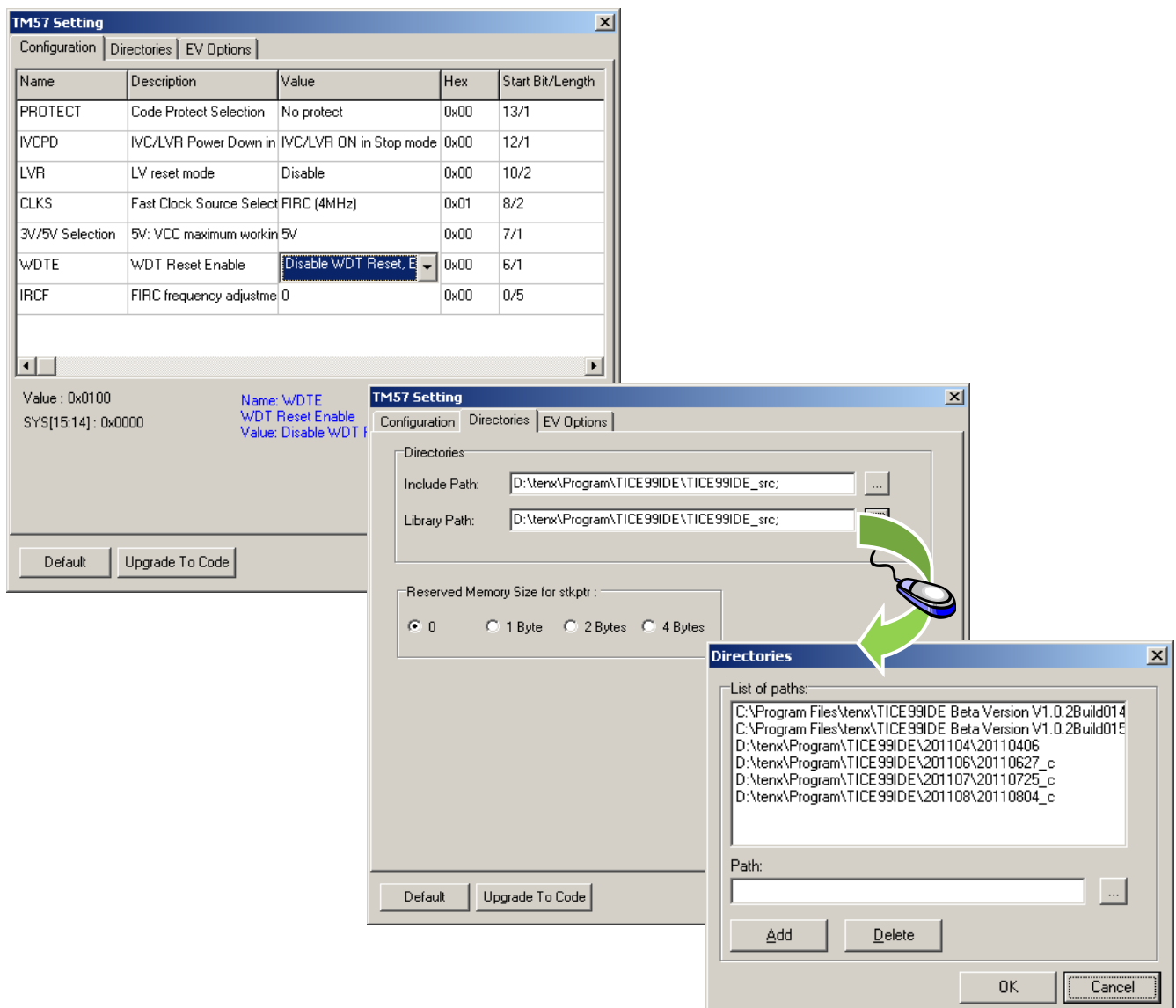


Figure 2.3 Project Setting

2.5 Build

"Build" can check the code you written whether it is correct or not, and compile the code into an executable file (.bin file) or an object file (.o file). It can also download an executable file to the ICE for online debugging.

2.5.1 Build, Rebuild

"Build" only compiles files in a project; however it will not download code to ICE. You can use this menu item to check the code you written to determine whether it is correct or not. If there is any error, the message will be displayed in output panel. Double click the error line; the caret will go to the place of editor where the error occurs. "Rebuild" will clear the .o files, .bin files before build the project.

2.5.2 Build and Download to ICE

Different from "Build", "Build and Download to ICE" will download code to ICE after a successful building.

2.5.3 Compile File

Compile file only compiles current file you are editing. You can use this function instead of "Build" to check only one file.

2.5.4 Configuration Debug, Release

These will produce debugging information whiling compiling. If release is checked, *.o file will not contain debug information.

2.5.5 Make Library

TICE99IDE provides tools let user can create own library in C and assembly language. The methods and concepts are shown in Figure 2.4.

Two stages:

1. Create object file

Programmer uses compiler, assembler to translate source code files into object files, that object files extensions of TM55/ 56 and TM57 are different, **the file of TM55/56 is *.obj and *.lpp file. The file of TM57 is *.o files.**

2. Create library

After create object files, user can run library maker to select object files and combine functions into a library file. The file of TM55/56 and file of TM57 are different library file. **The file of TM55/56 is *.o and *.lpp. The file of TM57 is *.lib.**

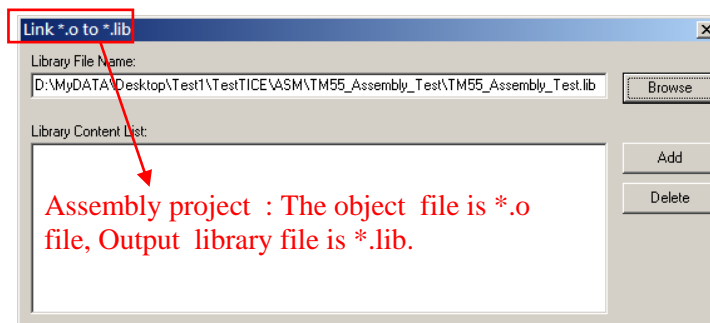


Figure 2.4.1 Assembly Language-Make Library

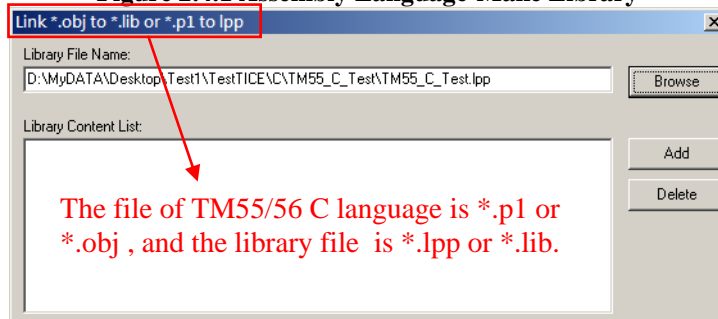


Figure 2.4.1 TM55/56 C Language - Make Library

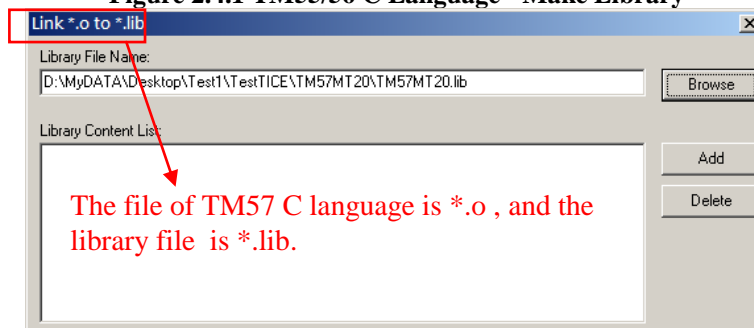


Figure 2.4.1 TM57 C language - Make Library

Concept:

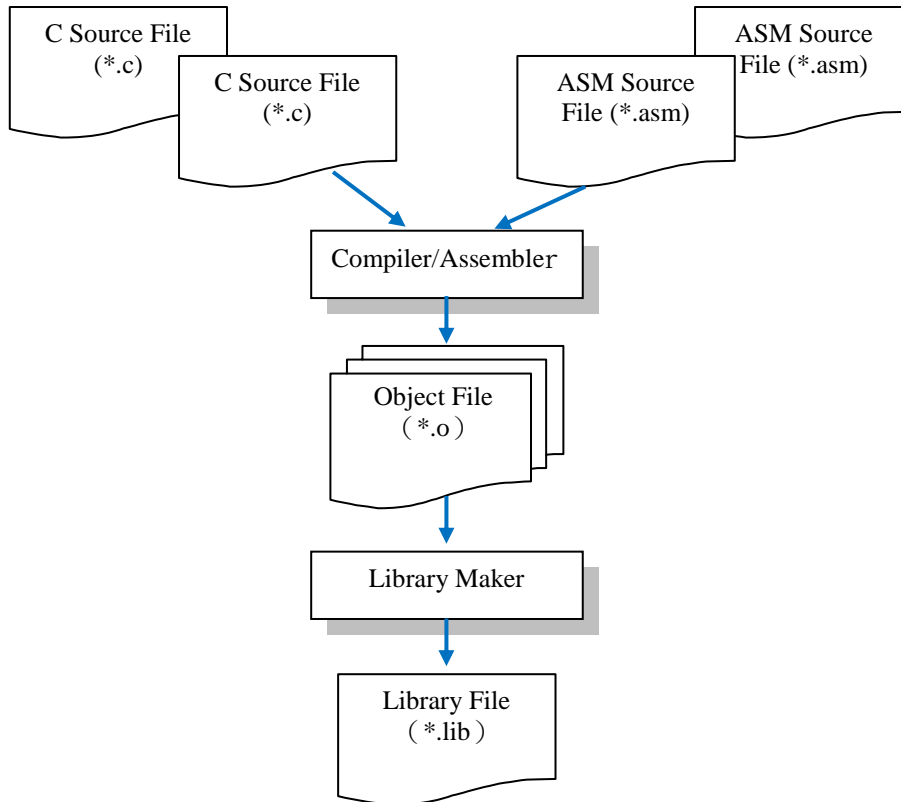


Figure 2.5 Concept of Library Maker

2.6 Debug

2.6.1 Pause

“Pause” stops all debug steps. When the ICE is in *Go, Free Run, Auto Single Step or Auto Step Over mode*, press “Pause” to stop debugging. Others it will be no action.

2.6.2 Run to Cursor

“Run to Cursor” will run to the line where the caret is located and wait for next debugging instruction.

2.6.3 Go, Free Run

“Go” and “Free Run” both will execute the code which was downloaded to ICE. The difference between “Go” and “Free Run” is “Free Run” will not stop when encountering break point, but “Go” will stop at the breakpoint in place if breakpoints are set.

2.6.4 Single Step, Auto Single Step

“Single Step” during debugging, it will execute code line by line, or execute one machine code when debugging with list file. When debugging ASM file, one line source code will correspond to one list file line. One line source code will correspond to multiple list file line when debugging C file. "Auto Step" is that IDE will automatically and periodically implement "Single Step" which is according to time setting. The timer of Auto Single Step setting is set in Tools|Option|Debug Option. The timer unit is millisecond.

2.6.5 Step Over, Auto Step Over

“Step Over” is a special kind of step. It will execute a program one line at a time, stepping over functions while executing them as a single unit. “Step Over” will execute one statement when it is not a function call.

2.6.6 Reset ICE

This will reset ICE’s program counter value to 0 and set the debug mode to be “single step” mode.

2.6.7 Initialize ICE board

“Initialize ICE board” will download the file that FPGA needed, confirm ICE connecting status and power status, test RAM and ROM and reset ICE.

2.6.8 Insert / Remove Break Point, Remove All Break Point

"Add Breakpoint" will allow the debugging steps to stop at the breakpoint. To interrupt the running program debugging line, you can double-click on the code line to set the breakpoint. To cancel a breakpoint, just double-click on the code line which has been set a breakpoint, then you can switch a non-interrupted state. "Remove all breakpoints" removes all breakpoint settings.

2.6.9 Debug with list file, debug with source file

There are two ways to debug. One is with list file, and the other is with source code. List file debugging will run one step according to the mnemonic code. However, source file will run one statement according to ASM or C file.

2.7 Tools

In “Tools”, you can switch language interface dynamically. We provide three languages now; they are English, Simplified Chinese and Traditional Chinese. You can decide what language interface you want to use, and then select it. The other function of “Tools” is “Option”. In “Option”, you can set text font and size for editor.

2.7.1 Language English

The language interface is switched to English.

2.7.2 Language Simplified Chinese

The language interface is switched to Simplified Chinese.

2.7.3 Language Traditional Chinese

The language interface is switched to Traditional Chinese.

2.7.4 QTP Creator

Output “Coding Request Form” of current project, it includes hex file name, checksum, system configuration and other information.

2.7.5 Hex File Converter

Hex file converter translates the source chip hex file to destination chip hex file. Table 2.1. shows the supported list.

Source Chip	Destination Chip
TM57PE11	TM57PE11A
TM57PE11A	TM57PE11B
TM57PA10	TM57PA10A
TM57PA20	TM57PA20A

Table 2.1 Supported List

2.7.6 Options

Option provides three kinds of setting:

- Editor:
 1. Set the text font, font size and tab stop for editor.
 2. Determine whether the new IDE automatically open the project which has been opened before.
 3. When user checks "Code Completion", then in code editing process, the Editor will display a resizable prompt window, which lists the valid elements for the user to select and add to your code.
 4. Set the color of keywords, numbers, colors, comment text, etc...
- Debug: Set "Auto step" of the time setting, and the timer unit is millisecond.
- Key Assignments: set the keystrokes for each item of menu.

2.8 Help

“Help” collects information about the IDE, including usage of IDE, the version number of compiler, assembler and linker which are executed by IDE, and update function of the IDE.

2.8.1 IDE User Manual

This will open an IDE user manual .pdf file which corresponding to the language. If you are in English language interface, it will open English version user manual, and so on.

2.8.2 C Compiler User Manual

This will open a C compiler user manual .pdf file which corresponding to the language.

2.8.3 Assembler User Manual

This will open an Assembler user manual .pdf file which corresponding to the language.

2.8.4 About

“About” window displays copyright information, versions of IDE, C compiler, Assembler, and Linker.

2.8.5 Update / New Version Check

User can check new version via Update | New Version Check. If there is no any new version, it will show the newest version window, otherwise update the newer part of IDE.

3. Toolbar

Toolbar collects the common usage of menu. The toolbar can be divided into four parts; they are “File”, “Edit”, “Build” and “Debug”.

- In “**File**” toolbar, there are 6 buttons on it. They are *New File, New Project, Open File, Open Project, Save and Save All*. The functions are the same as “File” menu.
- The second toolbar is “**Edit**”. Most of functions in menu are listed in here, including *Undo, Redo, Copy, Cut, Paste, Delete, Find, Find Previous, Find Next, Find in Project*. The last button of “Edit” toolbar is *Find and Replace*.
- Then, *Build, Rebuild, Build and Download to ICE and Compile File* is in the “**Build**” toolbar.
- The last toolbar is “**Debug**”, it includes *Go, Free Run, Single Step, Auto Single Step, Pause, Step Over, Auto Step Over, Run to Cursor, Reset ICE, Initial ICE Board, Set/Remove Break Point, and Remove All Break Point*.

4. Build a Project

TICE99IDE provides C and inline assembly compiler, it also provides pure assembly compiler. TICE99IDE services two types of project; they are C/Assembly project and assembly project. However, C/Assembly project may include both C and assembler program or contains only C programs alone. Programmer can select project type. Then following the step below, step by step to create a project, coding, build it, download it to ICE and test it.

4.1 Create a New Project

When you want to create a new project, press the File|New|Project in toolbar.

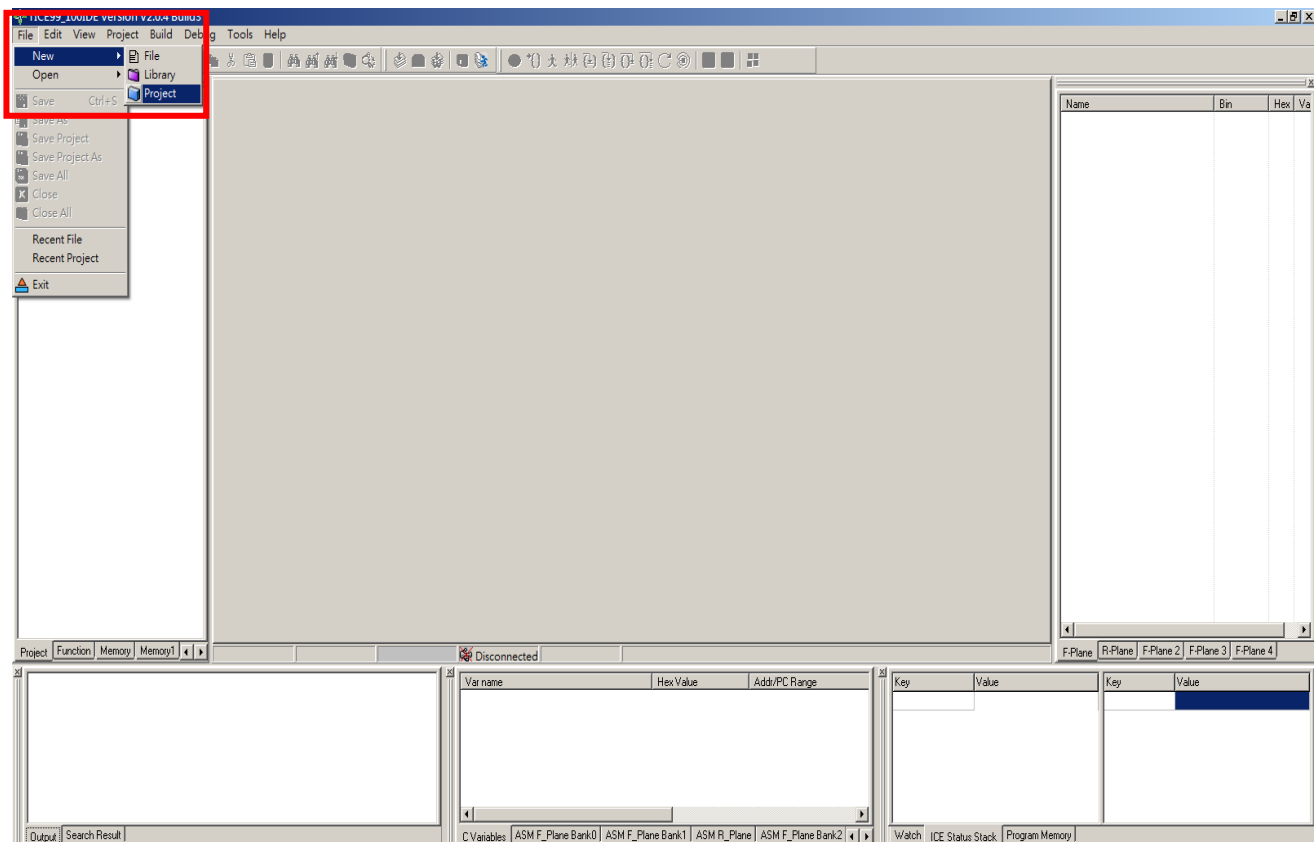


Figure 4.1 Create a New Project -1

The new project window will be displayed. Select your target chip type in chip list (See Appendix B). Next, choose your project type, assembly or C/Assembly. Give a name to the new project and decide which file folder to store this project.

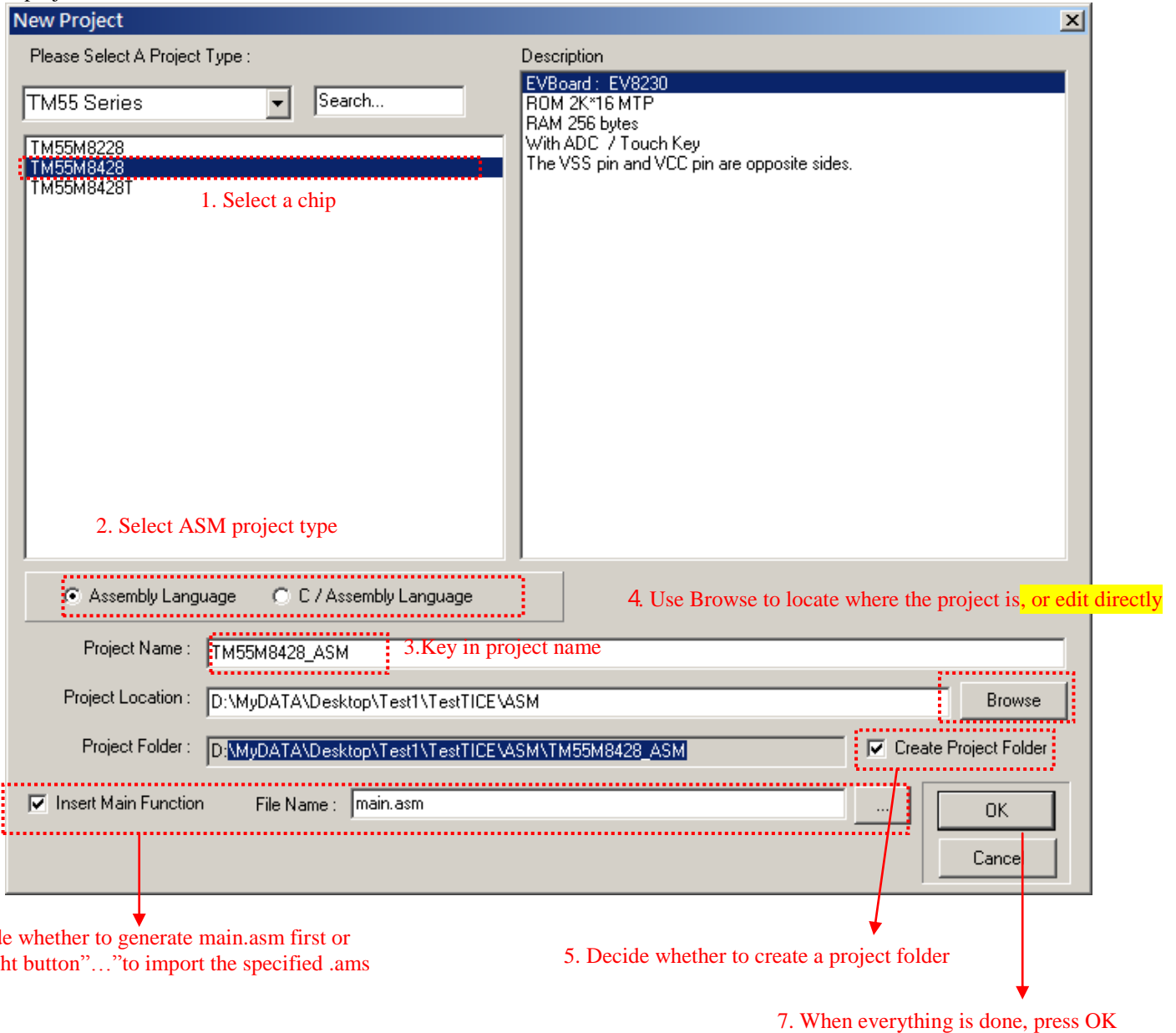
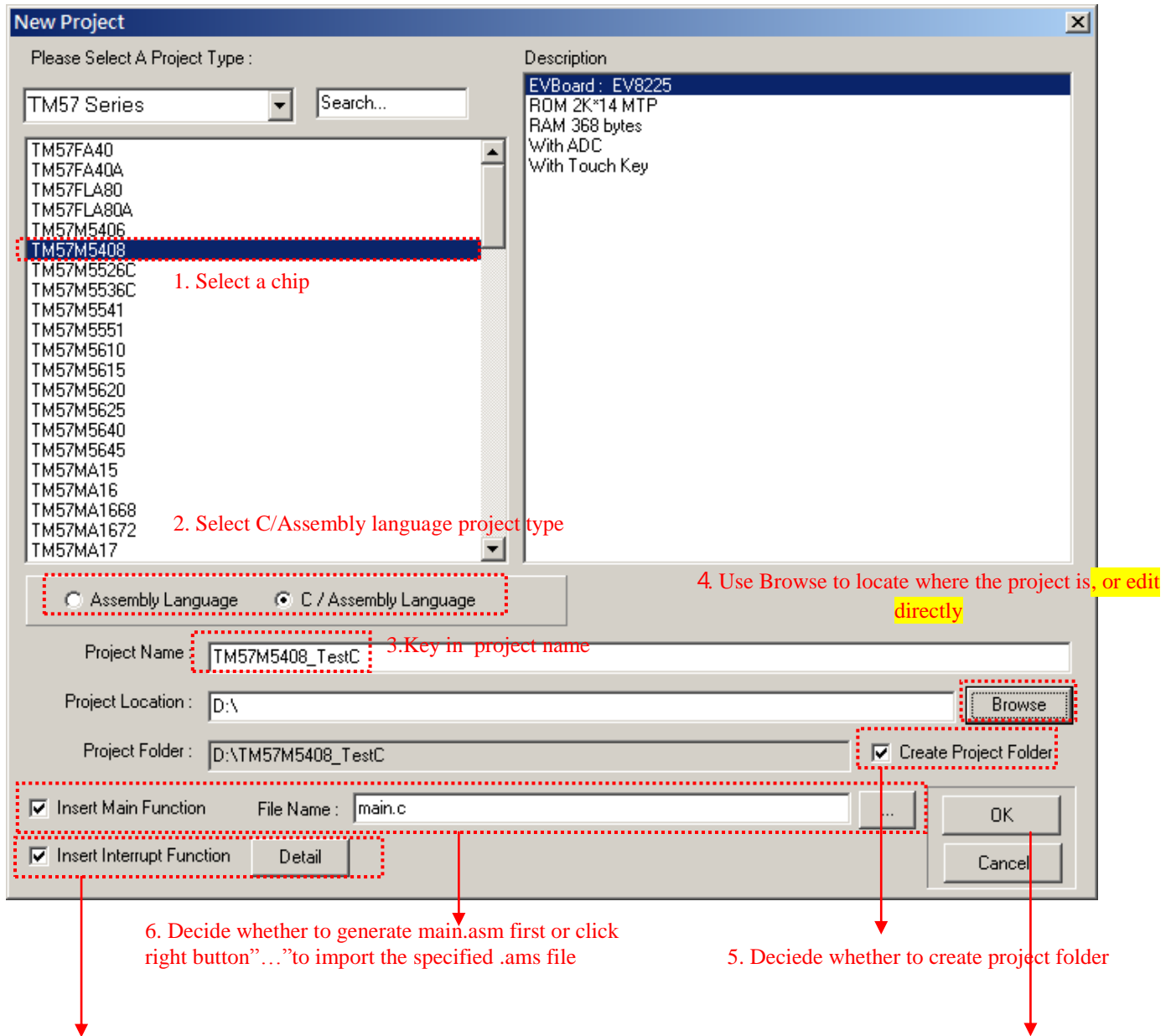


Figure 4.2 Create a New Project -1 –Create ASM project



TM57 Series C language project supports the interrupt functions, if necessary,click right button “Detail” to create.

7. When everything is done, press OK

Figure 4.2.1 Create a New Project – Create C/Assembly Language project

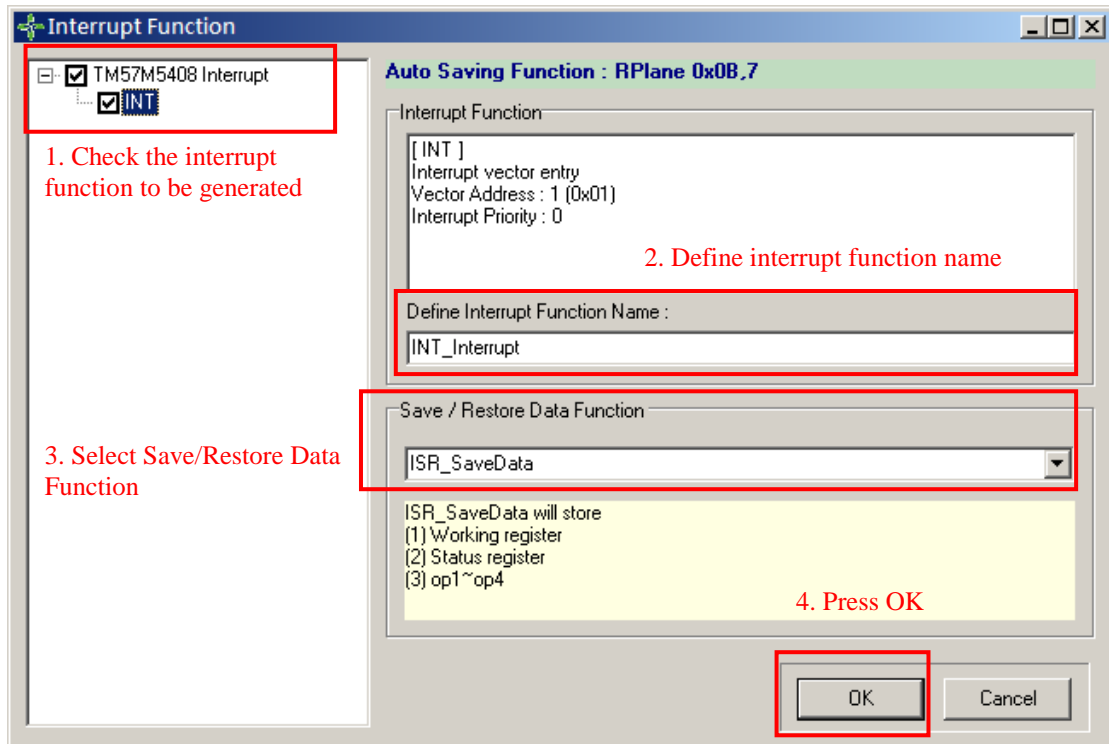
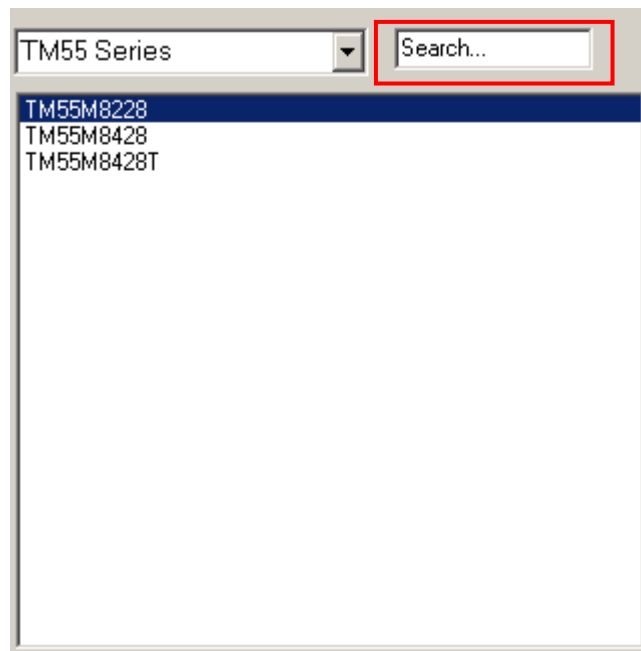
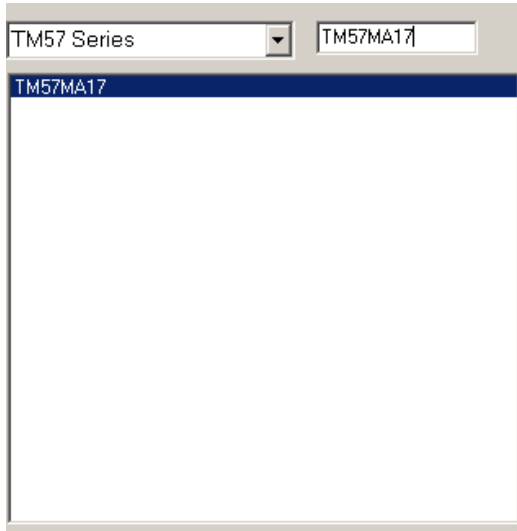


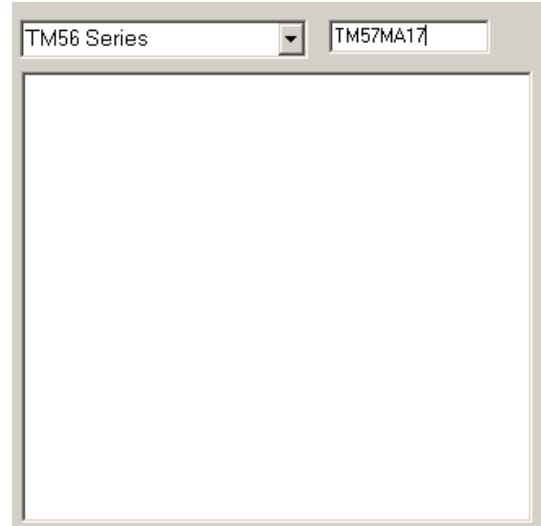
Figure 4.3 Create a New Project – Create the Interrupt Function



※Search Function: Provide users to quickly find the specified chip, according to the left Series, and enter corresponding chip model, you can quickly find the chip, but input the chip isn't in the series on left, it will not be found.

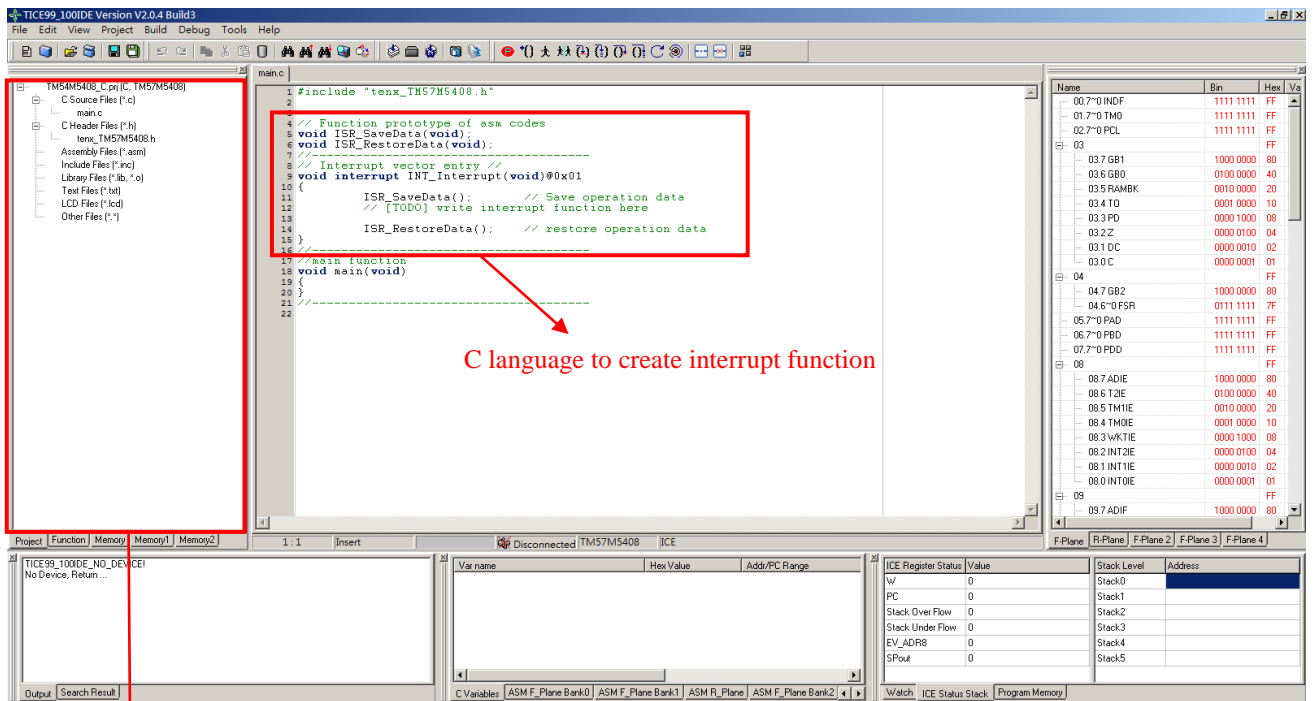


※TM57 Series has this chip ,so it's displayed.



There is no such chip in TM56 Series ,so it no shown.

After filling in the above required content, click OK and the project is created.



The project Manager will be different depending on the choice of assembly language or C language

Figure 4.4 Create a New Project – Create completed

4.2 Open an Existed Project

If you have an existed project and you want to modify it. You can use “Open an Existed Project” to do it.

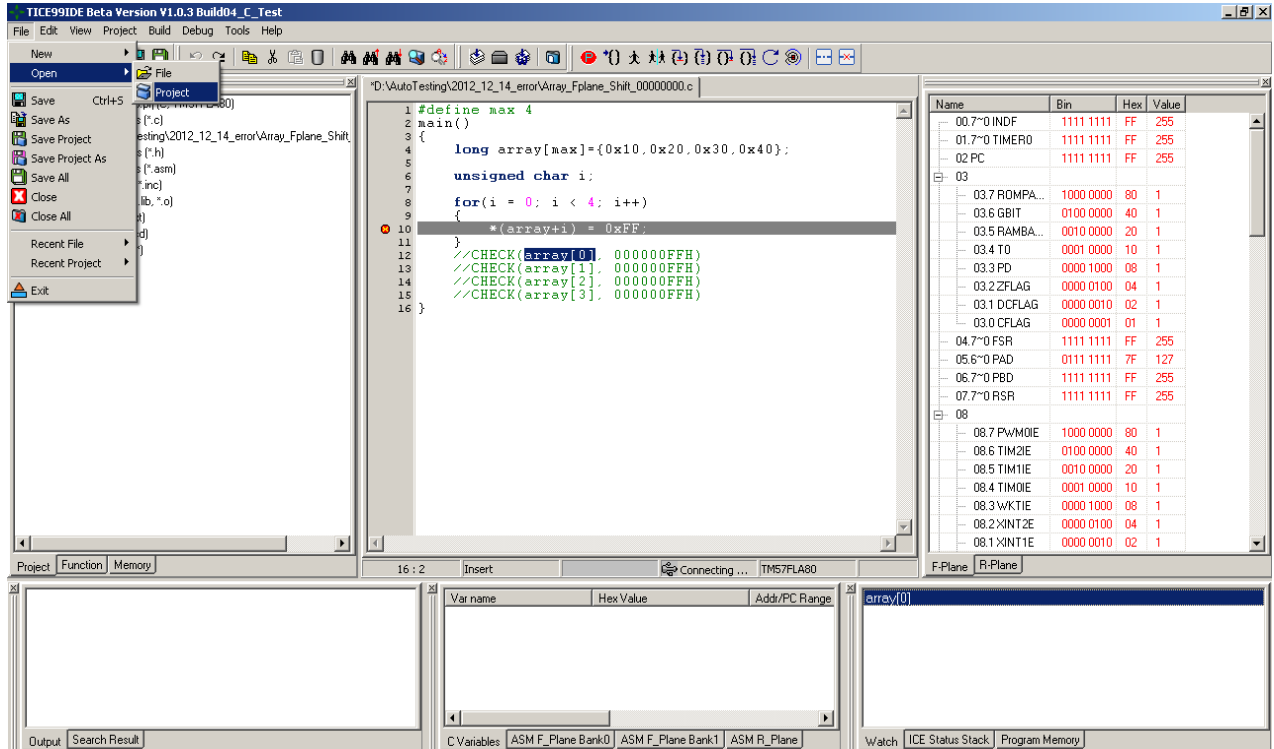


Figure 4.5 Open an Existed Project -1

Press File|Open|Project, the “Open File” dialog will appear. In the dialog, you can find out where your project is. The extended file name of project is .prj. Select the project and push the open button to open the existed project.

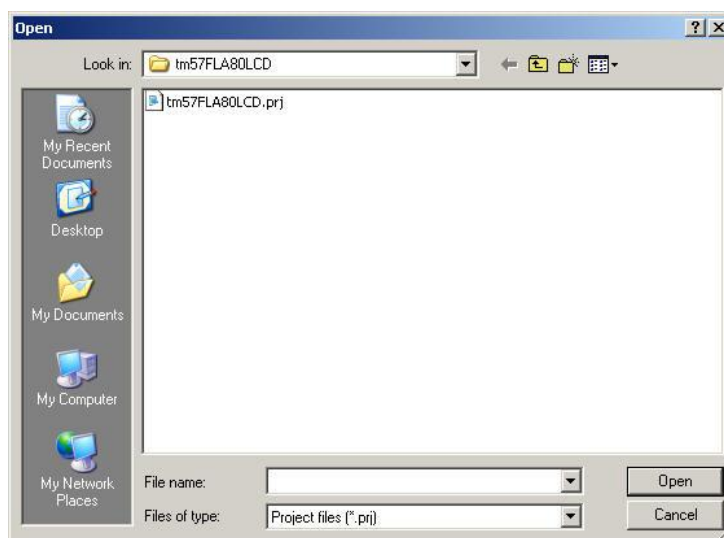


Figure 4.6 Open an Existed Project -2

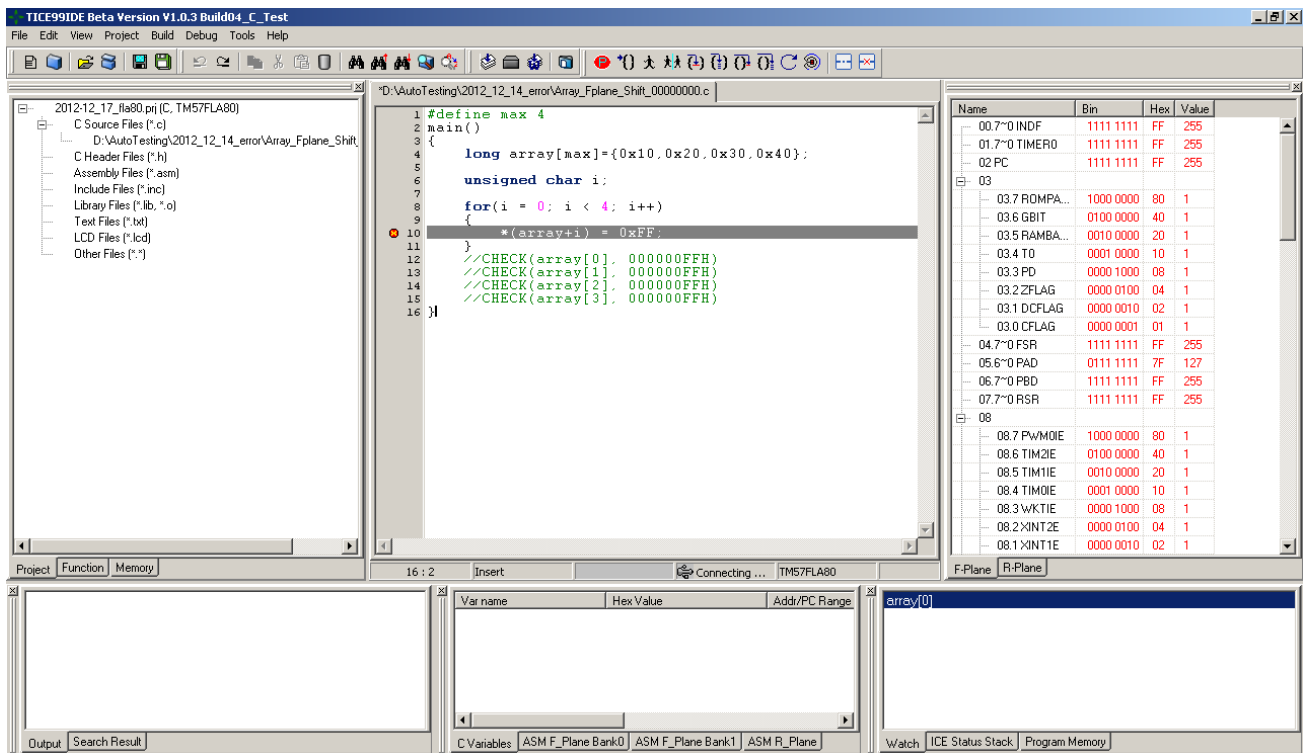
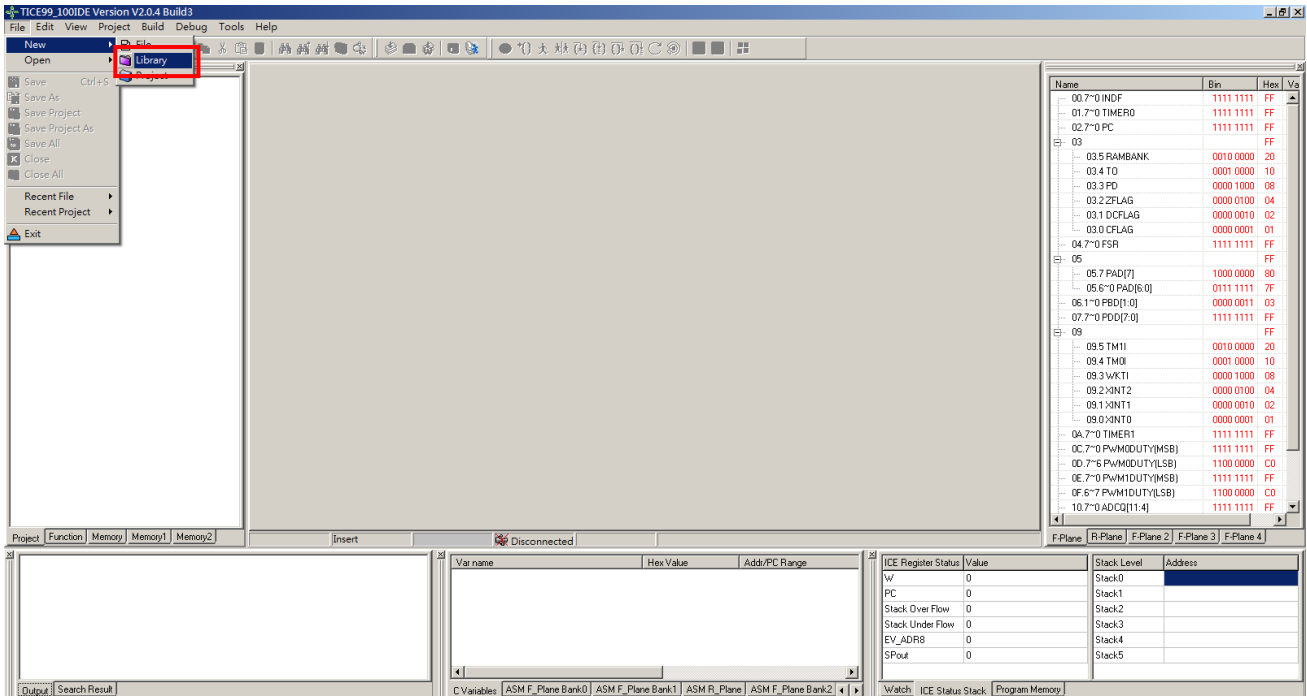


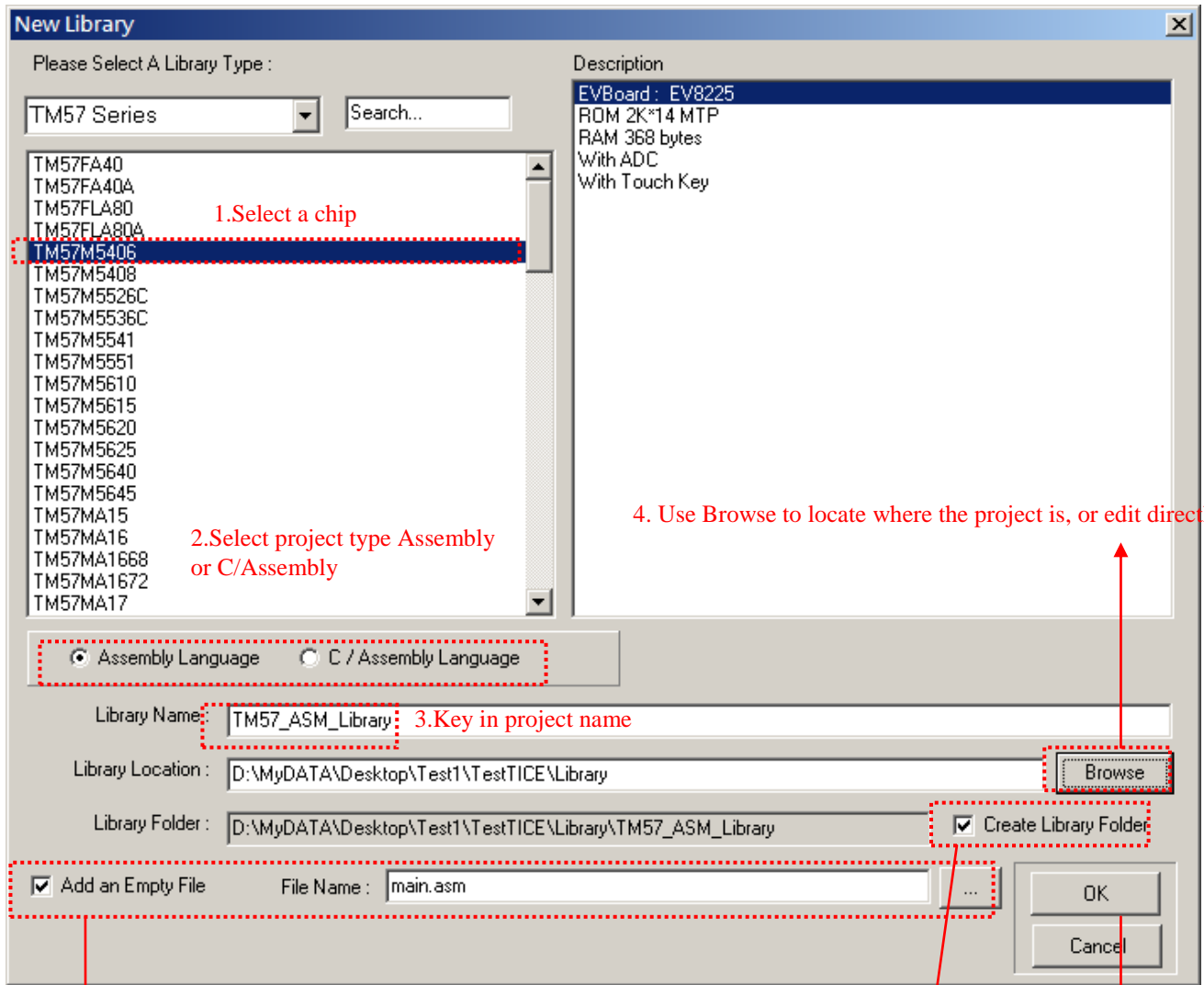
Figure 4.7 Open an Existed Project -3

4.3 Create a New Library

Press File ->New->Library to create a library



In the Library windows, select the target chip in the chip list according to the actual application. Next, select project type as Assembly or C/Assembly, then name the library and determine path to save library.



6. Decide whether to generate an empty file or click right button "..." to import the generated file

5. Decided whether to Create Library Folder

7. When everything is done, press OK

Figure 4.8 Create a new library

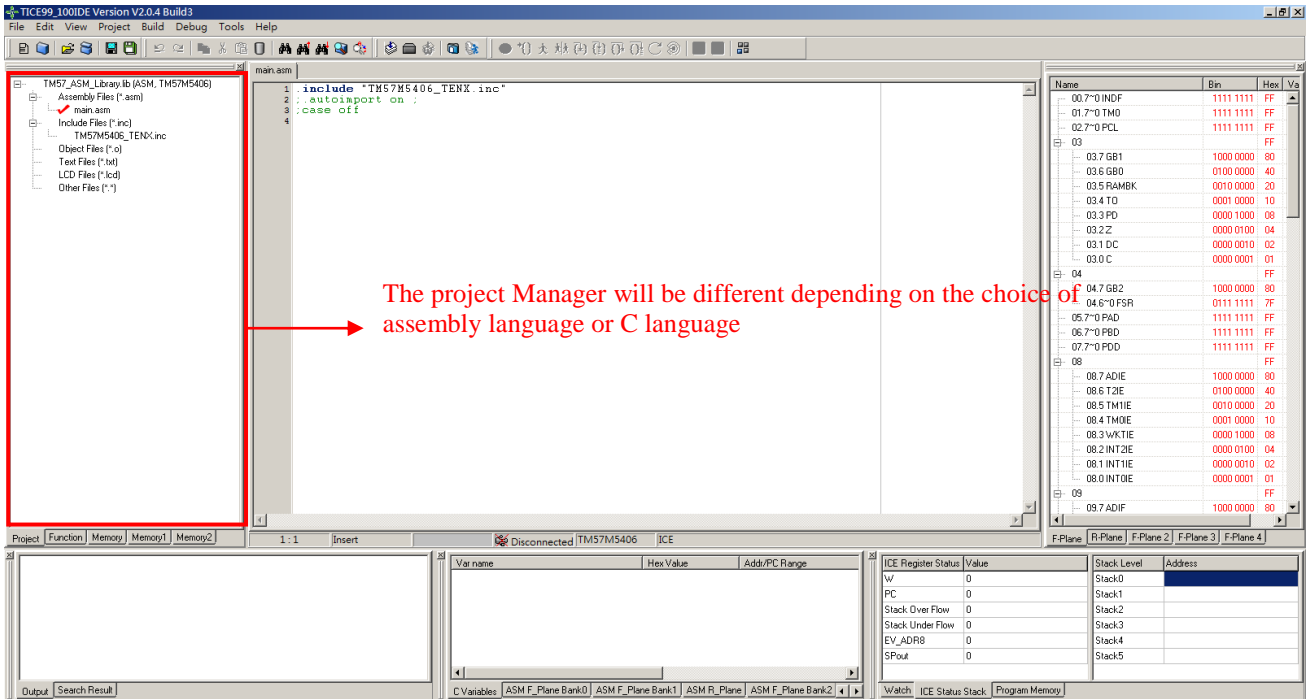


Figure 4.9.1 Create a new library completed

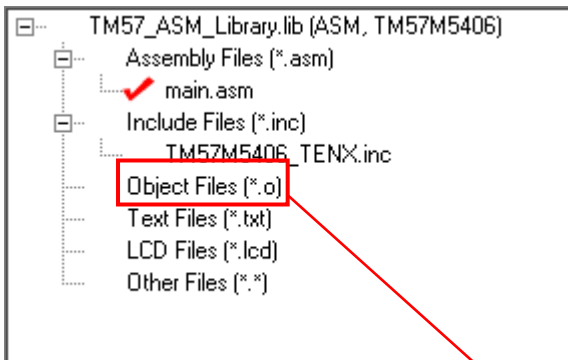


Figure 4.9.2 Assembly language project list

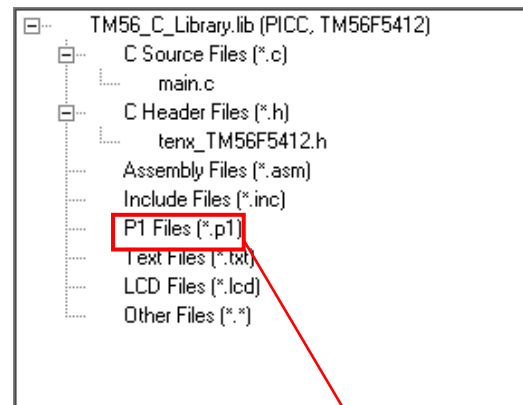


Figure 4.9.3 TM55/56 C language project list

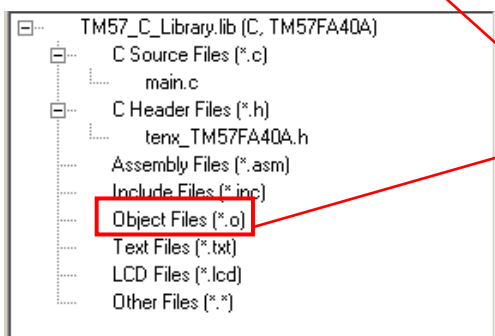


Figure 4.9.4 TM57 C/Assembly language project list

The loaded Object File is an .o file

The loads P1 File(.P1)

4.4 Project Setting

When you want to modify configuration parameters for ICE, you can press Project|Project Setting|Configuration to change value of parameters. The type of setting items of configuration is displayed according to the ICE types. For example, they are IRCF, WDTE, 3V/5V Selection, CLKS, LVR, ICVPD as well as PROTECT.

After you choose one configuration item, the corresponding information will be shown at the bottom part of the panel. It includes name, item description, selection description and hex value. All you can do is clicking combo box of value to choose the one that you want. At the same time, the value will be changed to respond the configuration setting value.

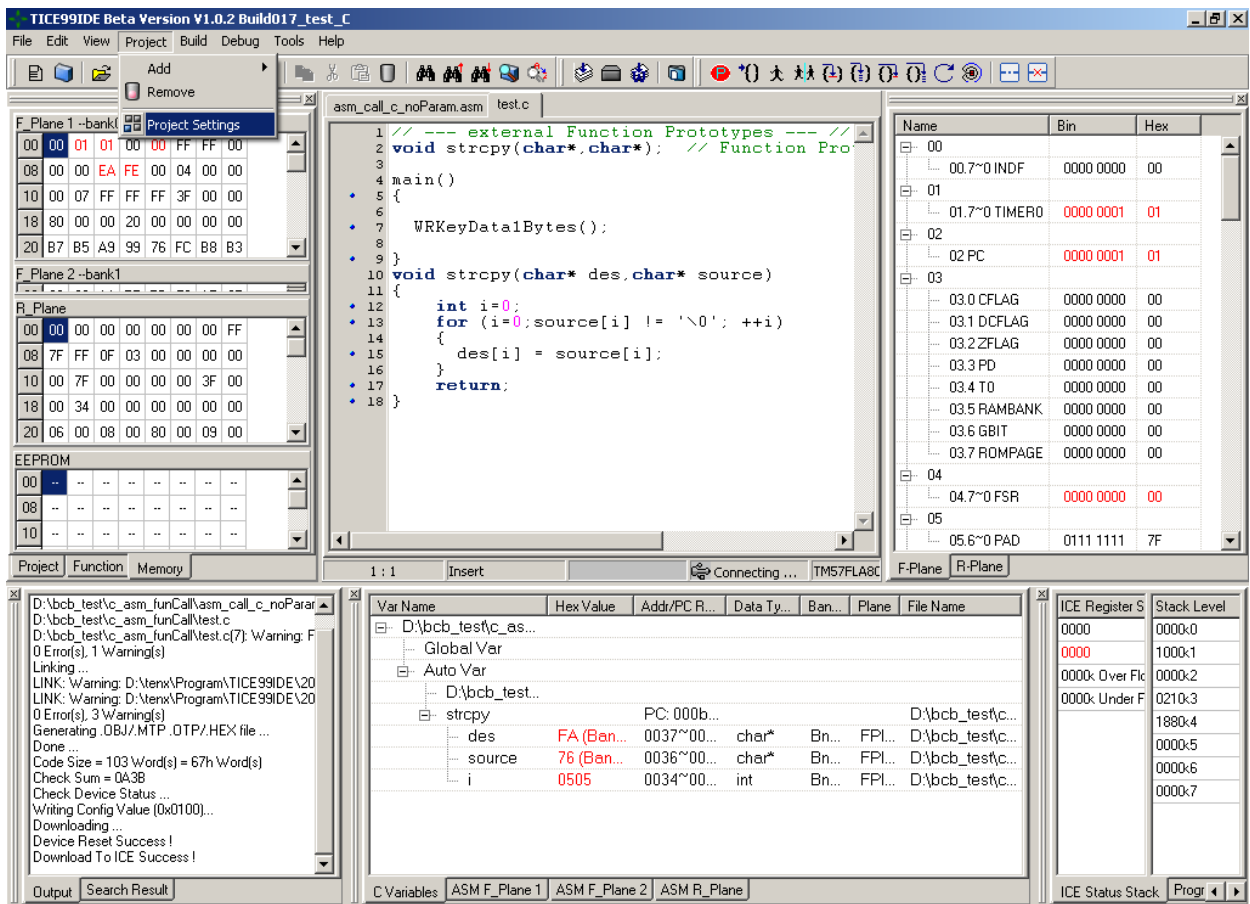


Figure 4.10 Project Setting -1

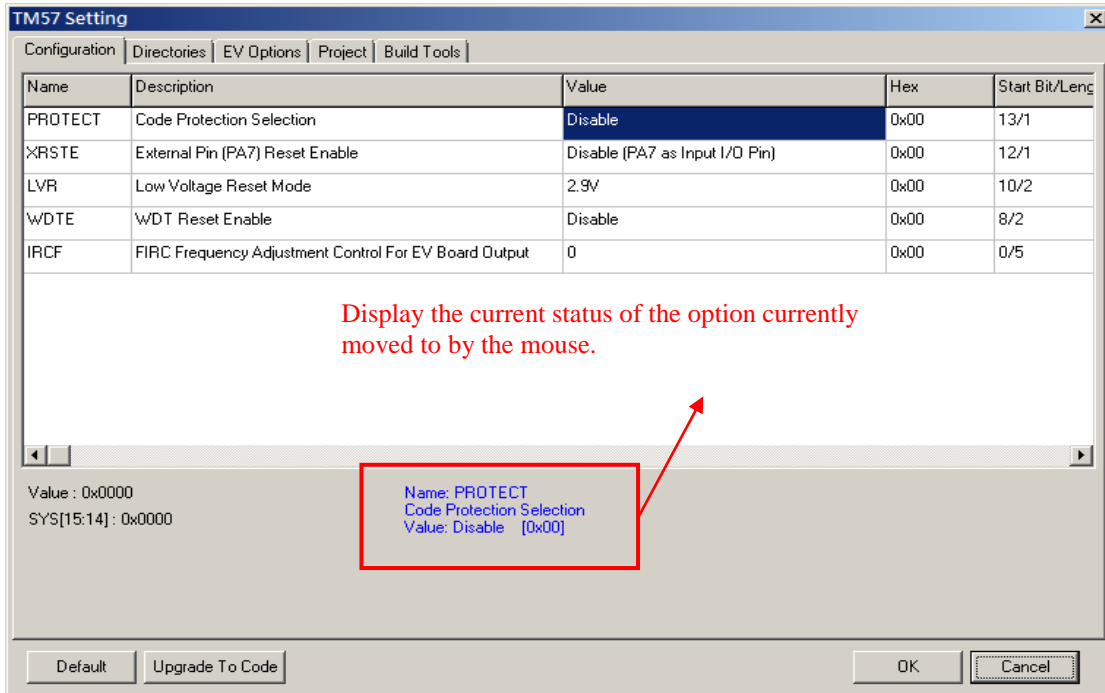


Figure 4.11.1 Project Setting – Configuration

The Directories menu is to record the location on include file and the .lib Library.

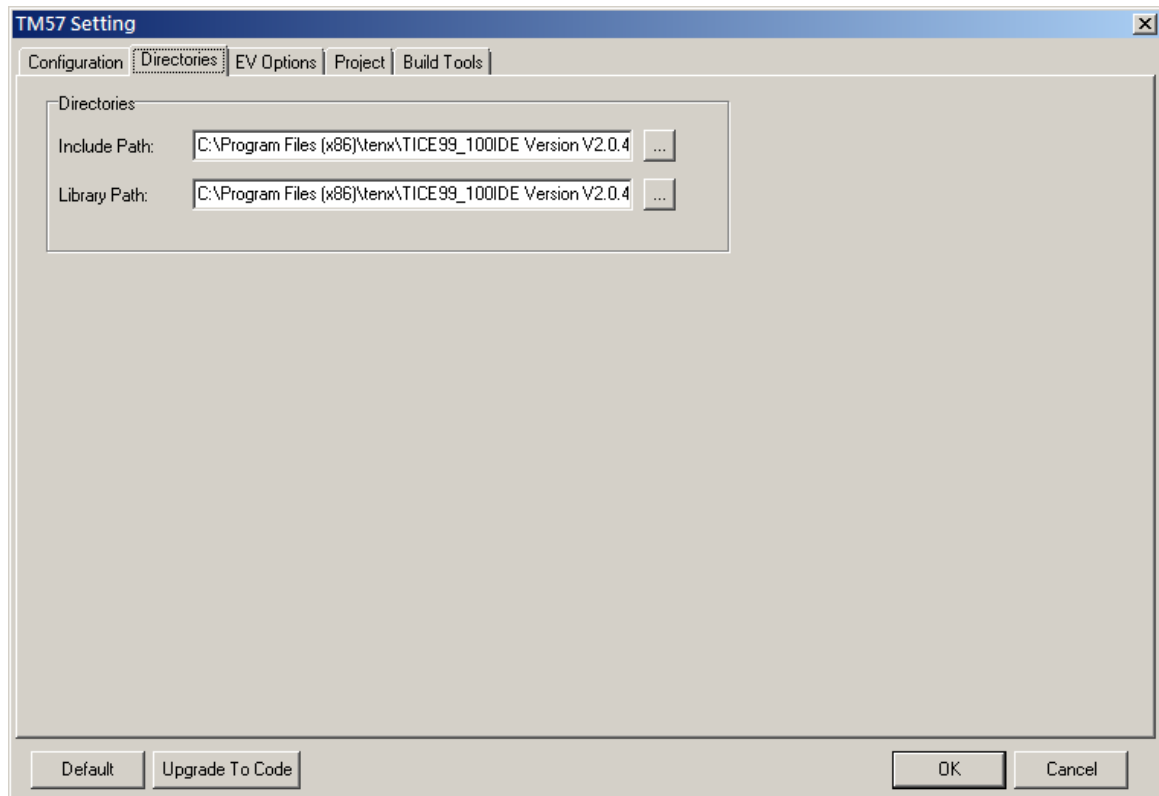


Figure 4.11.2 Project Setting - Directories

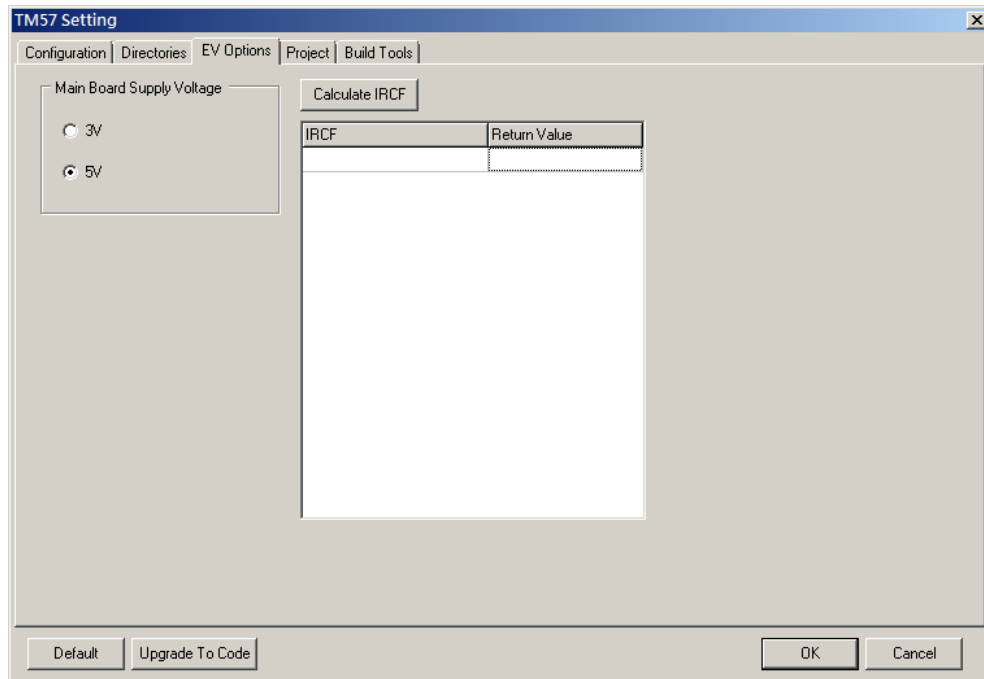


Figure 4.11.3 Project Setting – EV options

The Project page provides some options for project settings, and TM57 Series C project, there will be an option for reserved memory Size for STKPTR for user to choose. The instructions are as follows:

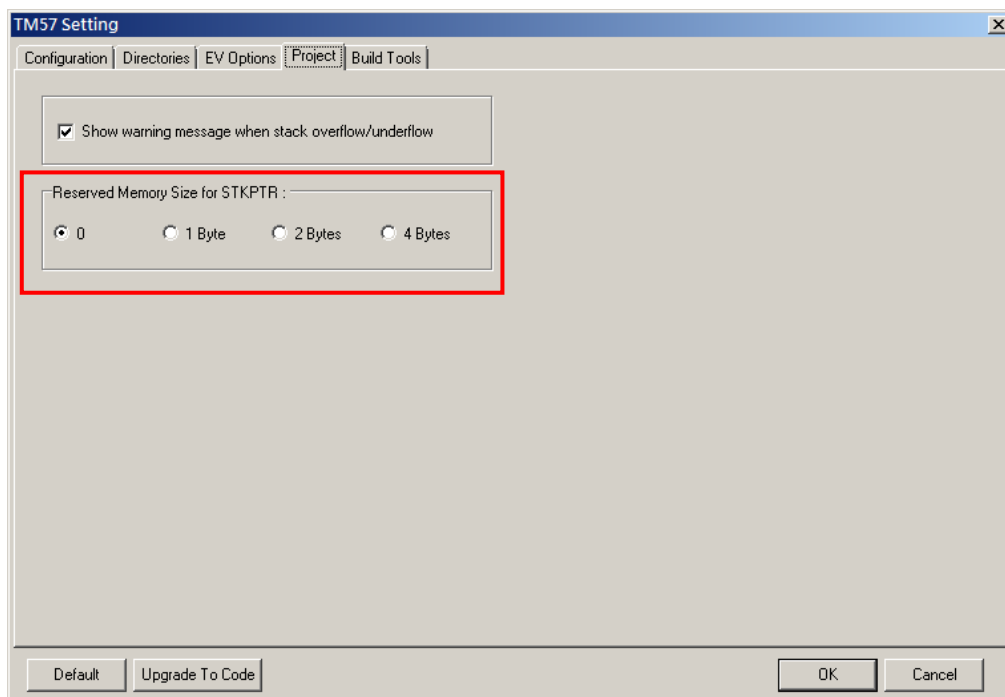


Figure 4.11.4 Project Setting – Project

For C project, in the tab page of "Project", there will show a selection: "Reserved Memory Size for stkptr". It will reserve an additional RAM space to capture and save the contents of STKPTR, and restore the data previously saved. Programmer has 4 selections of reserved size: 0, 1, 2, 4 byte(s). The default setting is 0 byte (non-reserved).

STKPTR is a reserved area for register variables in C compiler. It holds temporary variables and results during the execution of a program. If user selects outside 0, C compiler will allocate RAM space to store STKPTR and add extra instructions within function in order to save and restore the content of STKPTR.

We provide some situations needed to set "Reserved Memory Size for STKPTR" to avoid error occurs.

1. One function invokes another function call and callee modifies the data of STKPTR in process. This situation will overwrite the STKPTR saved by caller previously. Once callee returns, caller will retrieve the overwritten STKPTR.
2. The left side of assignment statement is array (or pointer) and right side is function call (ex. array[i] = func()). Before call function, the data of array (or pointer) will be saved to STKPTR.
3. Function expression (ex. retVal = fun1() + fun2()). Before calling the second function, the result of the first function will be saved to STKPTR.
4. The more complexity of the programming, the more internal interactions will be affected. Need to save STKPTR during the execution.

We strongly recommend the following program coding style to avoid the above situations. To state more particularly, using simple programming statement instead a complex one.

Original Code	The Best Way
array[i] = func()	retVal = func(); array[i] = retVal;
retVal = fun1() + fun2()	retVal = fun1(); retVal = retVal + fun2();

In order to illustrate how this works, two kinds of case will be described: one is reserved memory is not set, the other is setting 1 byte reserved. We will point out the differences in *.s file.

```

1 int R_Var[4];
2
3 int i;
4
5
6 int f1()
7 {
8     i = 1;
9     R_Var[i] = i + 3;
10    return R_Var[i];
11 }
12
13 main()
14 {
15
16     i = 0;
17     R_Var[i] = f1();
18 }
19

```

No setting reserved memory to backup STKPTR	Reserve 1 byte memory to backup STKPTR
<pre> main() ----- : int main () :----- : : .segment "CODE";;2 : .proc _main : .segment "CODE";;3 : { : .dbg line, "D:\nbc_test\20130107_fla80\Pointer_Set_Array : CALL initialize_defdata : : i = 0; : .dbg line, "D:\nbc_test\20130107_fla80\Pointer_Set_Array : MAC_STORE_IMMED _i+0,\$0.2 ;;val=0,Flags=1001 : : R_Var[i] = f1(); : .dbg line, "D:\nbc_test\20130107_fla80\Pointer_Set_Array : MAC_STORE_FR _i+0,op1,1,0 ;adjust of scale type : MAC_SHL_IMMED op1,1,1 ;3,pop=1,_istkptr=0,flags=1,DT=1 : MOVWLW _R_Var ;1001,1,0,idel=1,stkptr : ADDWF op1,0 ;340C,addaddr,static : MAC_STORE -1,stkptr+0,1 ;push,id=1345,flags=1009 : CALL f1 ;Flags=1 stk=1 id=0 inop=1 stkptr returnFlags=1 : MAC_STORE_MEMtoPTR op2,0,stkptr+0,0,2,0,0,0 ;func->ptr : : } : : .dbg line, "D:\nbc_test\20130107_fla80\Pointer_Set_Array : RET : .dbg line : : .endproc </pre>	<pre> main() ----- : int main () :----- : : .segment "CODE";;2 : .proc _main : .segment "CODE";;3 : { : .dbg line, "D:\nbc_test\20130107_fla80\Pointer_Set_Array_RI : CALL initialize_defdata : : i = 0; : .dbg line, "D:\nbc_test\20130107_fla80\Pointer_Set_Array_RI : MAC_STORE_IMMED _i+0,\$0.2 ;;val=0,Flags=1001 : : R_Var[i] = f1(); : .dbg line, "D:\nbc_test\20130107_fla80\Pointer_Set_Array_RI : MAC_STORE_FR _i+0,op1,1,0 ;adjust of scale type : MAC_SHL_IMMED op1,1,1 ;3,pop=1,_istkptr=0,flags=1,DT=1 : MOVWLW _R_Var ;1001,1,0,idel=1,stkptr : ADDWF op1,0 ;340C,addaddr,static : MAC_STORE -1,stkptr+0,1 ;push,id=1345,flags=1009 : CALL f1 ;Flags=1 stk=1 id=0 inop=1 stkptr returnFlags=1 : MAC_STORE_MEMtoPTR op2,0,stkptr+0,0,2,0,0,0 ;func->ptr : : } : : .dbg line, "D:\nbc_test\20130107_fla80\Pointer_Set_Array_RI : RET : .dbg line : : .endproc </pre>
<p>Save the temporary variables and results to STKPTR</p>	

```

f1()
.proc _f1
.segment "CODE":;:3
i = 1;
.dbg line, "D:\bcb_test\20130107_fla80\Pointer_Set_Array_RPLANE
MAC_STORE_IMMED _i+0,$1,2 ;;val=1,Flags=1001
R_Var[i] = i + 3;
.dbg line, "D:\bcb_test\20130107_fla80\Pointer_Set_Array_RPLANE
MAC_STORE_FR _i+0.op1,1,0 ;adjust of scale type
MAC_SHL_IMMED op1,1,1 ;3.pop=1,_istkptr=0,flags=1,DT=1001
MOVWLW _R_Var ;1001,1,0,idel=1,stkptr
ADDWF op1,0 ;340C,addaddr,static
MAC_STORE _-1,stkptr+0,1 ;push,id=1345,flags=1009
MAC_STORE_FR _i+0.op1,2,0 ;V-98,1,133,137F6C,'i',1,opid=1,_iQues
MAC_ADD_IMMED op1,$3,2 ;val=3 <<<ADD_IMMED 2>>,1,callname=runti
MAC_STORE_MEMtoPTR op1,0,stkptr+0,0,2,0,0,0 ;MP2,icallsb=1241,2
return R_Var[i];
.dbg line, "D:\bcb_test\20130107_fla80\Pointer_Set_Array_RPLAN
MAC_STORE_FR _i+0.op1,1,0 ;adjust of scale type
MAC_SHL_IMMED op1,1,1 ;3.pop=1,_istkptr=0,flags=1,DT=1001,
MOVWLW _R_Var ;1001,1,0,idel=1,stkptr
ADDWF op1,0 ;340C,addaddr,static
MAC_STORE op1,op2,2 ;1
MAC_STORE_PTRtoMEM op1,0,op2,0,2,0 ;1
GOTO L0003
L0003: .dbg line, "D:\bcb_test\20130107_fla80\Pointer_Set_Array_RPLANE
RET
.dbg line
.endproc

f1()
.proc _f1
.segment "CODE":;:3
{
}
.dbg line, "D:\bcb_test\20130107_fla80\Pointer_Set_Array_RPLANE_0000C
MAC_STORE stkptr,f1_PARAM-1,1 ;0
i = 1;
.dbg line, "D:\bcb_test\20130107_fla80\Pointer_Set_Array_RPLANE_0000C
MAC_STORE_IMMED _i+0,$1,2 ;;val=1,Flags=1001
R_Var[i] = i + 3;
.dbg line, "D:\bcb_test\20130107_fla80\Pointer_Set_Array_RPLANE_0000C
MAC_STORE_FR _i+0.op1,1,0 ;adjust of scale type
MAC_SHL_IMMED op1,1,1 ;3.pop=1,_istkptr=0,flags=1,DT=1001,id=2,i
MOVWLW _R_Var ;1001,1,0,idel=1,stkptr
ADDWF op1,0 ;340C,addaddr,static
MAC_STORE _-1,stkptr+0,1 ;push,id=1345,flags=1009
MAC_STORE_FR _i+0.op1,2,0 ;V-98,1,133,137F6C,'i',1,opid=1,_iQues
MAC_ADD_IMMED op1,$3,2 ;val=3 <<<ADD_IMMED 2>>,1,callname=runti
MAC_STORE_MEMtoPTR op1,0,stkptr+0,0,2,0,0,0 ;MP2,icallsb=1241,2
return R_Var[i];
.dbg line, "D:\bcb_test\20130107_fla80\Pointer_Set_Array_RPLANE_0000C
MAC_STORE_FR _i+0.op1,1,0 ;adjust of scale type
MAC_SHL_IMMED op1,1,1 ;3.pop=1,_istkptr=0,flags=1,DT=1001,id=2,i
MOVWLW _R_Var ;1001,1,0,idel=1,stkptr
ADDWF op1,0 ;340C,addaddr,static
MAC_STORE op1,op2,2 ;1
MAC_STORE_PTRtoMEM op1,0,op2,0,2,0 ;1
GOTO L0003
L0003: .dbg line, "D:\bcb_test\20130107_fla80\Pointer_Set_Array_RPLANE_00000
RET
.dbg line
.endproc
    
```

Overwrite the data of STKPTR

Backup the contents of STKPTR

Restore the data of previously-saved STKPTR

Note: although setting "Reserved Memory Size for stkptr" can avoid overwriting the data of STKPTR, but it will consume more RAM memory to backup the data of STKPTR and more ROM program memory to save and restore the content of STKPTR within the function. We strongly recommend reducing programming complexity instead of setting "Reserved Memory Size for stkptr".

Build Tools will only appear in TM55/56 C/Assembly language project, it provide for optimization when compiling standard C projects.

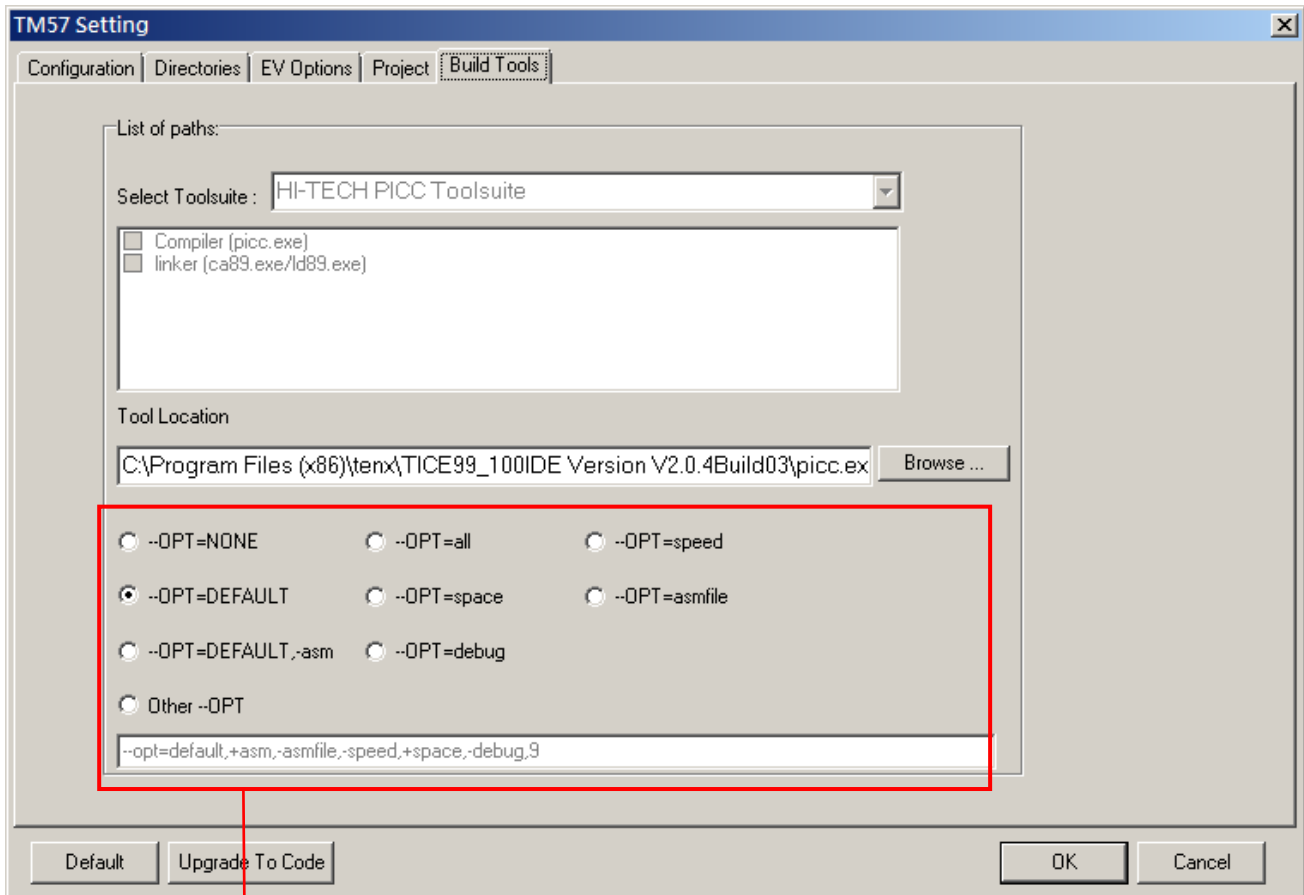


Figure 4.11.5 Project Setting – Build Tools

User can choose the degree of optimization for compiling standard C projects ,”None” is not optimized ,”Other –OPT” can manually enter the content to be optimized.

4.5 Editor Options

Editor setting helps programmer to change the text font style and text size for editor. Users can press Tools|Options to open option window. In editor options, you can change the font name, size and tab stops. By the way, users can change keyword color of C and ASM to fit own habit.

In Other Options, IDE will auto-open recent project when option “Auto open recent project when start IDE” is checked. The editor will popup code completion proposal, when “Code Completion” is checked.

Key Assignments provides function that customizes short cut for all menus, user just chooses the short cut at “Keystrokes”, and then presses “Assign New Key”.

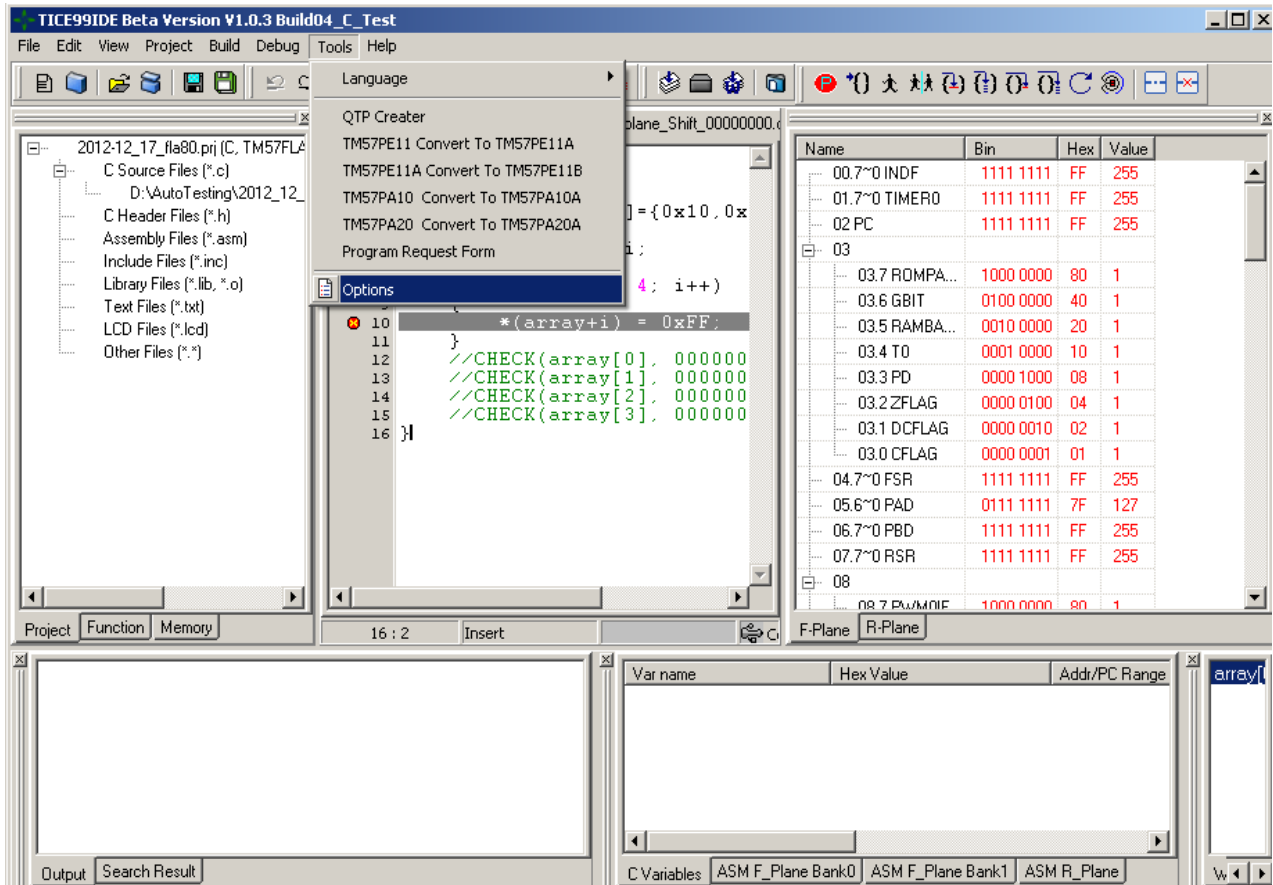


Figure 4.12 Editor Options -1

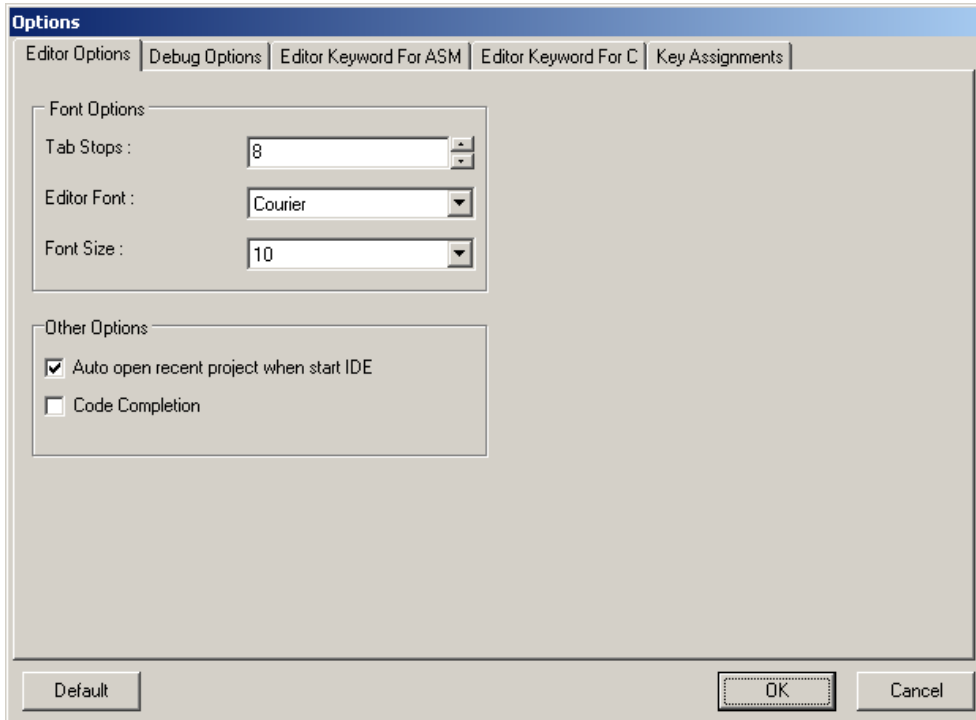


Figure 4.13 Editor Options -2

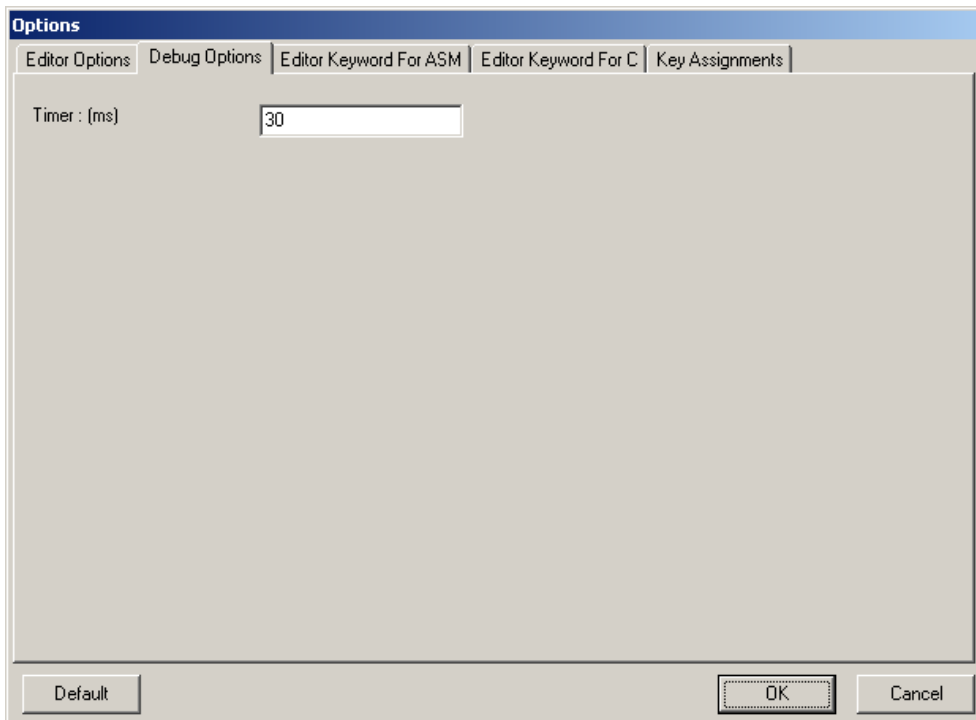


Figure 4.14 Editor Options -3

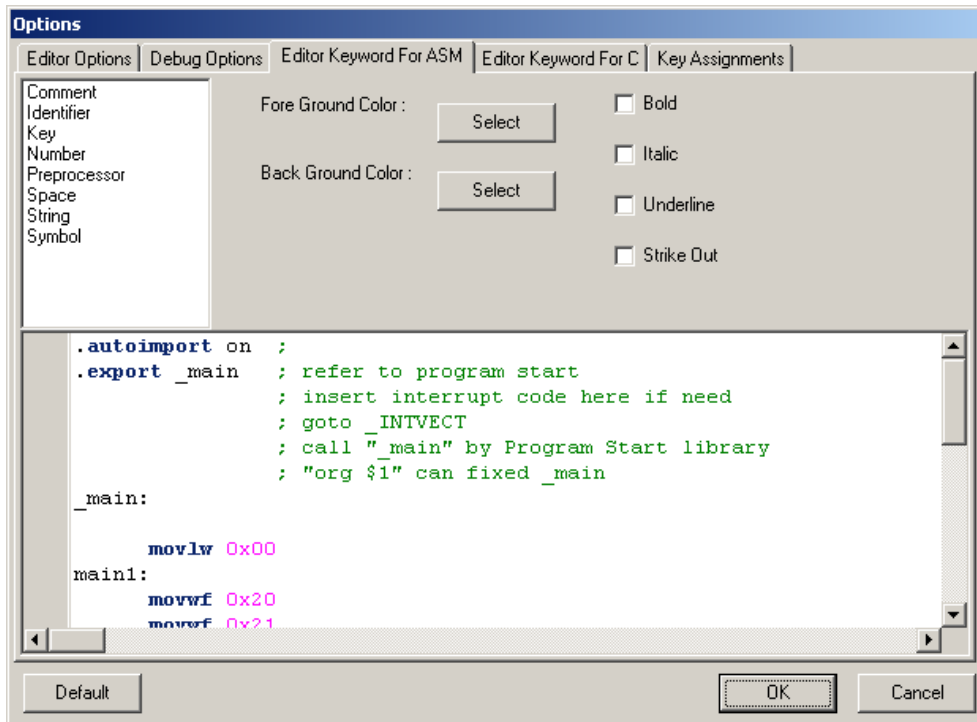


Figure 4.15 Editor Setting -4

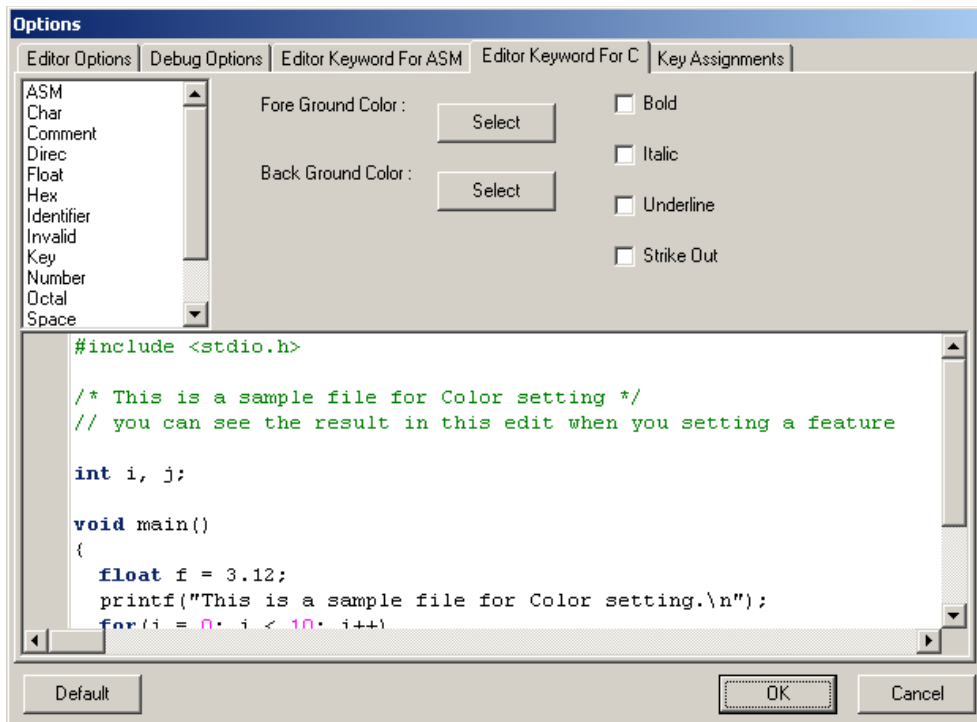


Figure 4.16 Editor Setting -5

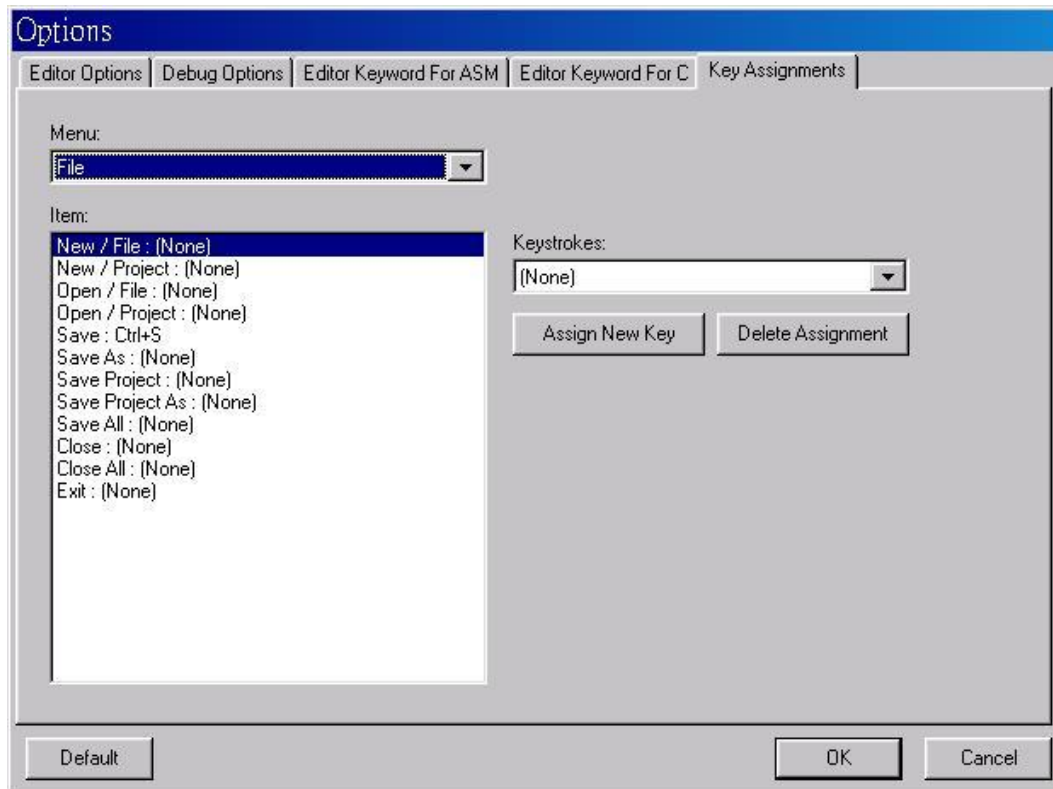



Figure 4.17 Editor Setting -6

4.6 Bookmarks

It is used to look for a certain period of record among the dataset, either to move from one record to another record of the sequential search for a specific keyword. A bookmark marks a location in a dataset so that an application can easily return to that location quickly. In Editor Area, user can right-click to use the pop-up menu and then use bookmarks function. Each bookmark records the file name and line number, and will show a bookmark icon  at the left side of the editor. Up to 10 bookmarks can be marked. Using “Goto Bookmarks” to return to the location of the bookmark.

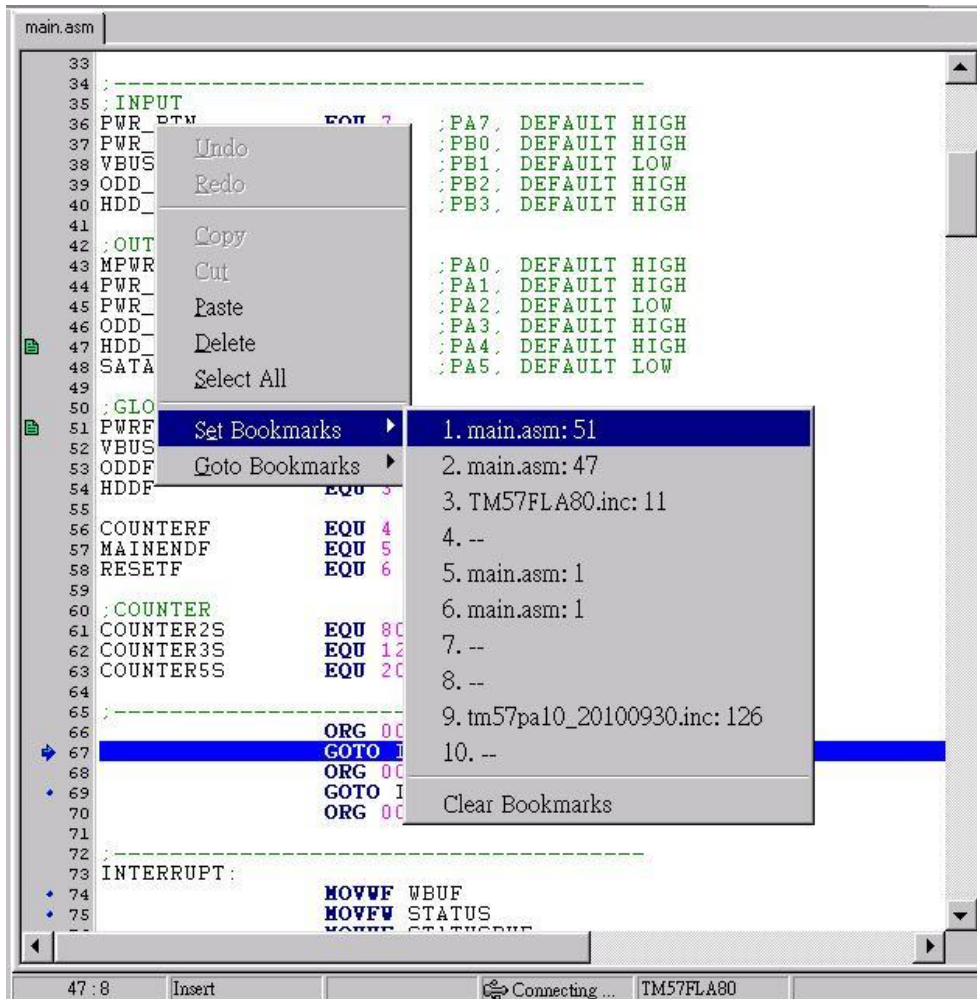


Figure 4.18 Bookmarks

4.7 Searching

With the search function, programmer can quickly search keywords in the program code area. Searching function helps user coding more efficiently. Use searching function to find out keywords to replace or modify. Short cut which is a fast way to open find window is Ctrl + F. Once there is a string in the search edit, you can press F3 to find out next one, or Ctrl + F3 to search previous one. For shortcut key combination settings, please refer to the menu Tools|Options page of the "Key Assignment".

4.7.1 Find

“Find” can search a specified string or words in current active editor page. You have to key in a search string let IDE search it for you. The caret will jump to corresponding line if it finds the search string. If there is no any match string, the end of file or begin of file dialog will be shown. Once you have a search string, you can press F3 or Ctrl + F3 to search next or previous match string.

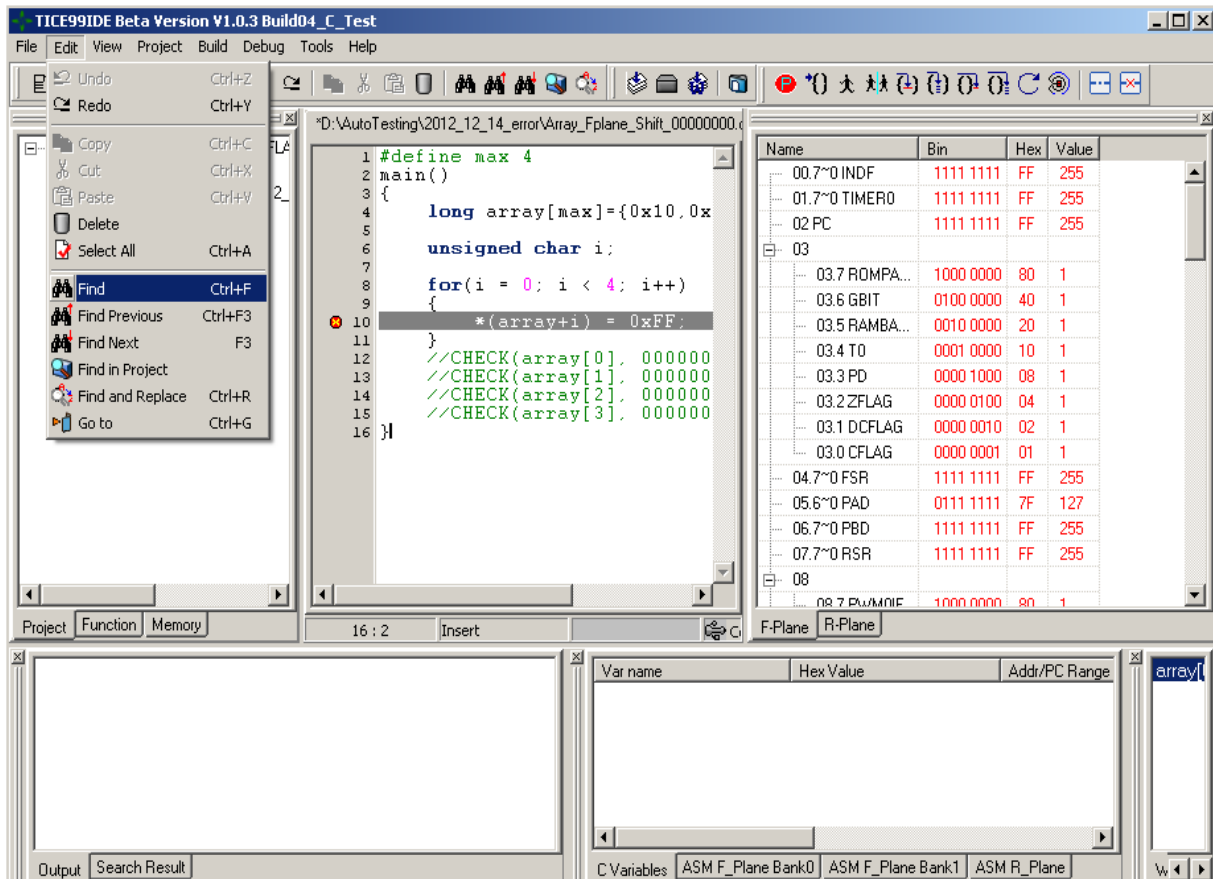


Figure 4.19 Find -1

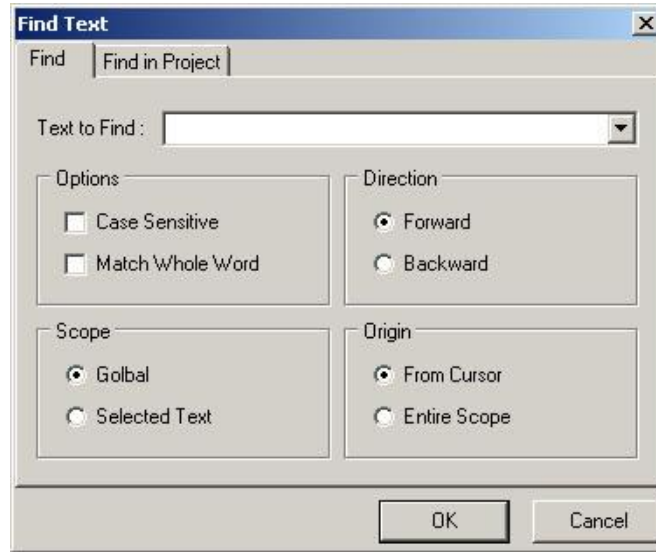


Figure 4.20 Find -2

4.7.2 Find in Project

“Find in Project” will find out all the match string in project or opened files and list the searching result in search result window. User can double-click one item in the list to go to the record line in the editor.

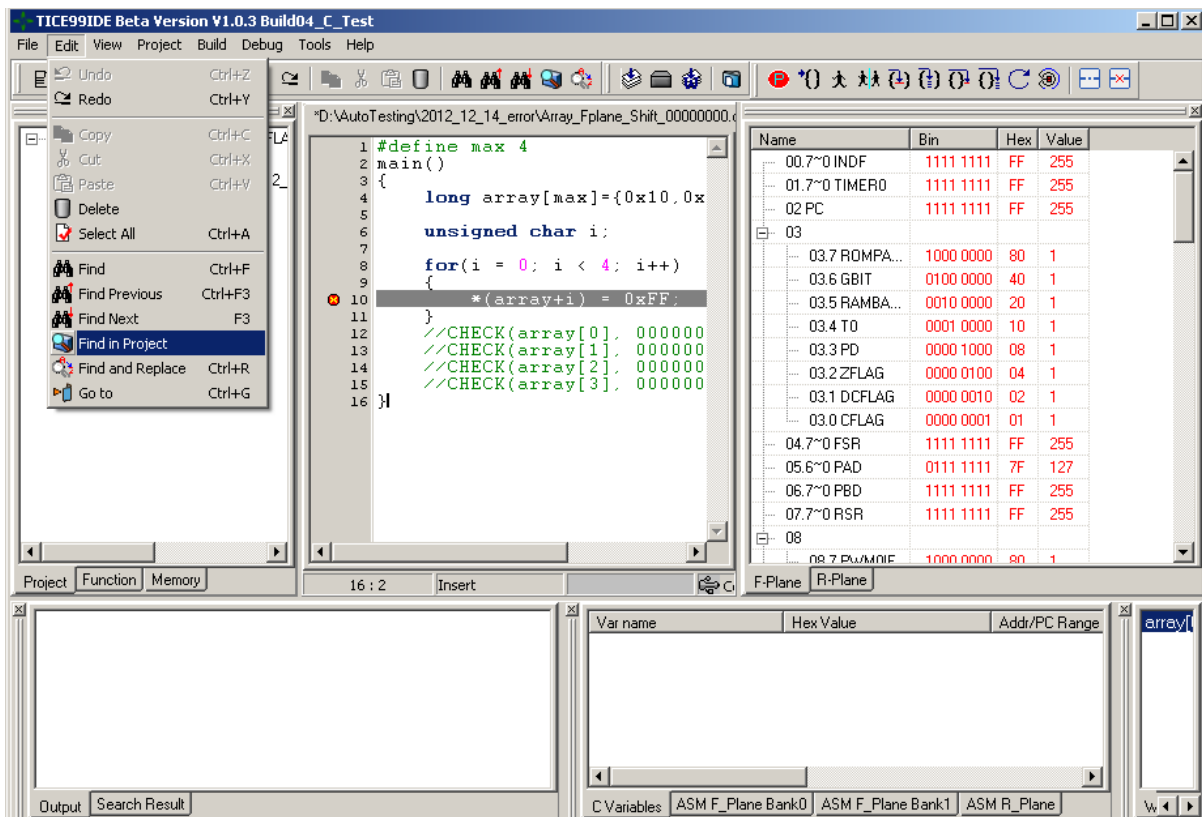


Figure 4.21 Find in Project -1

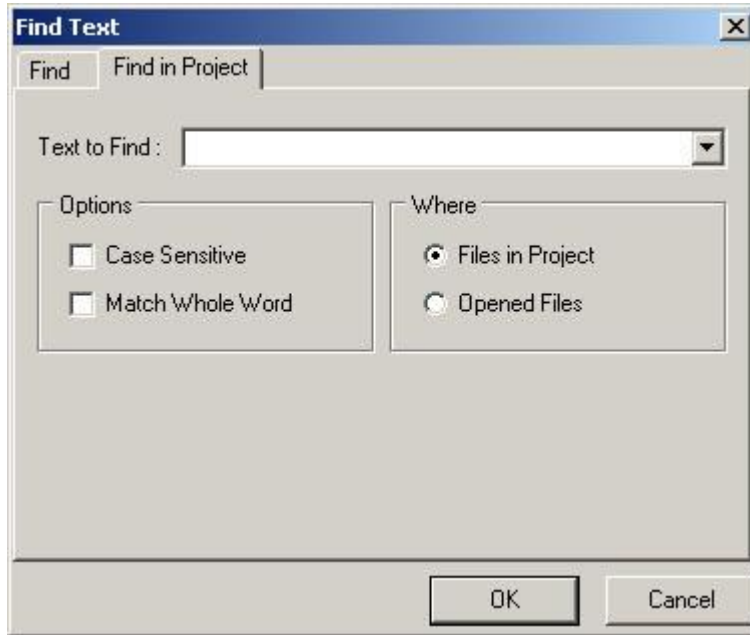


Figure 4.22 Find in Project -2

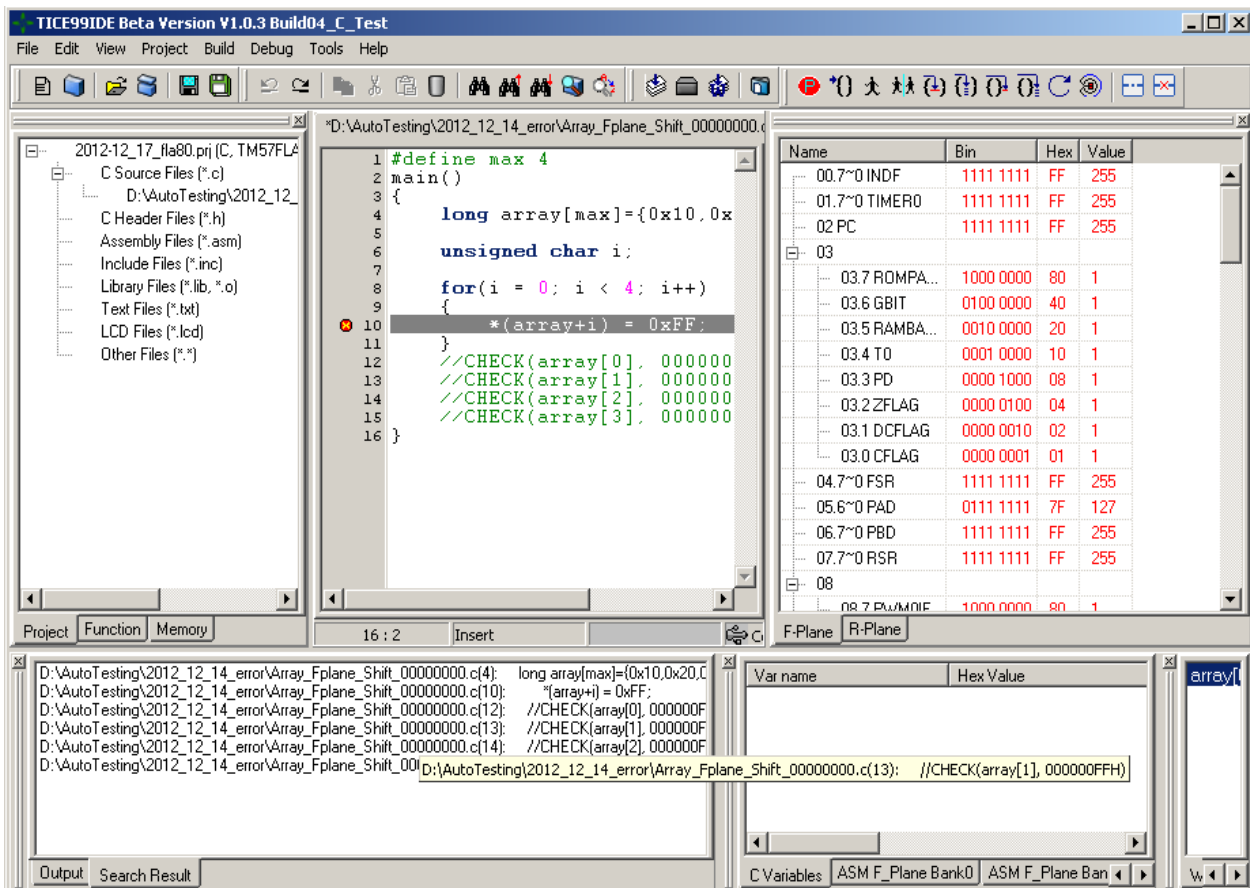


Figure 4.23 Find in Project -3

4.7.3 Find and Replace

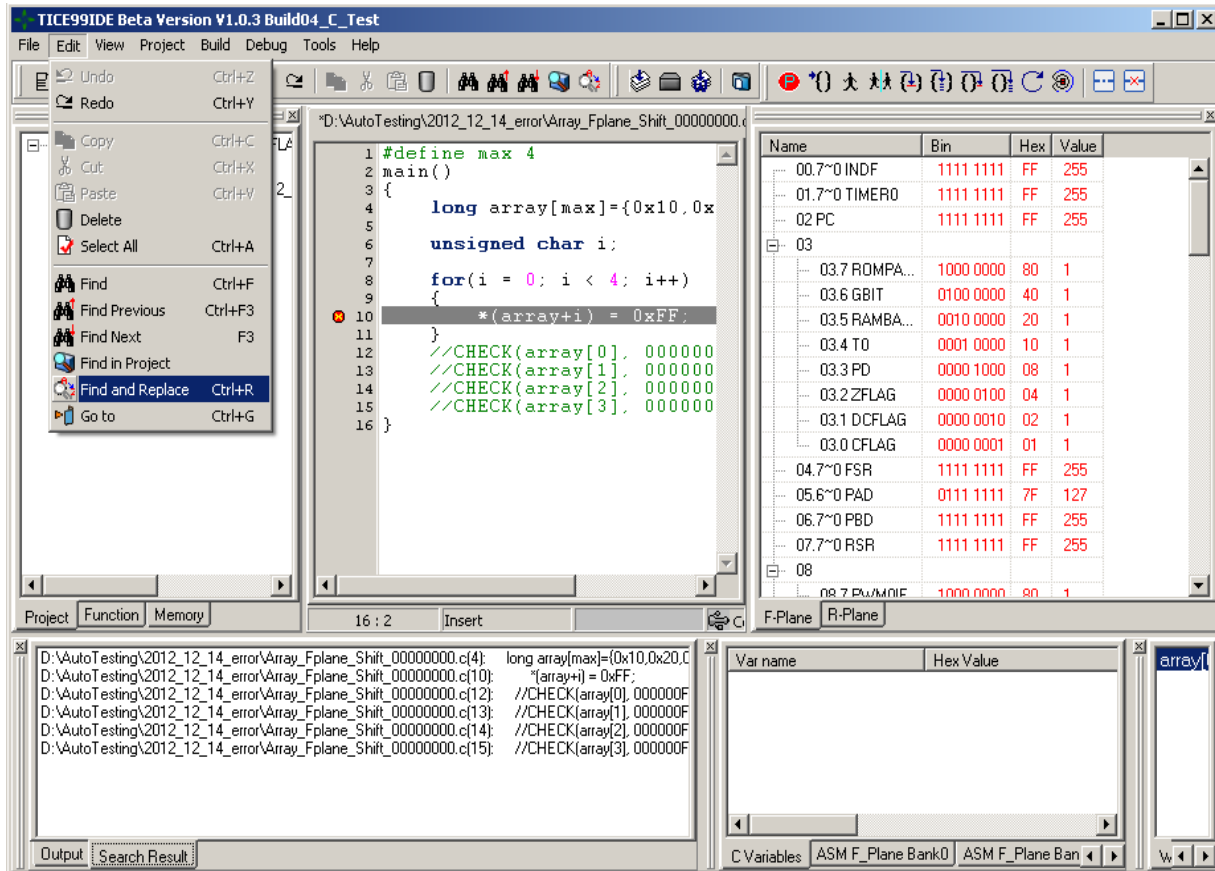


Figure 4.24 Find and Replace -1

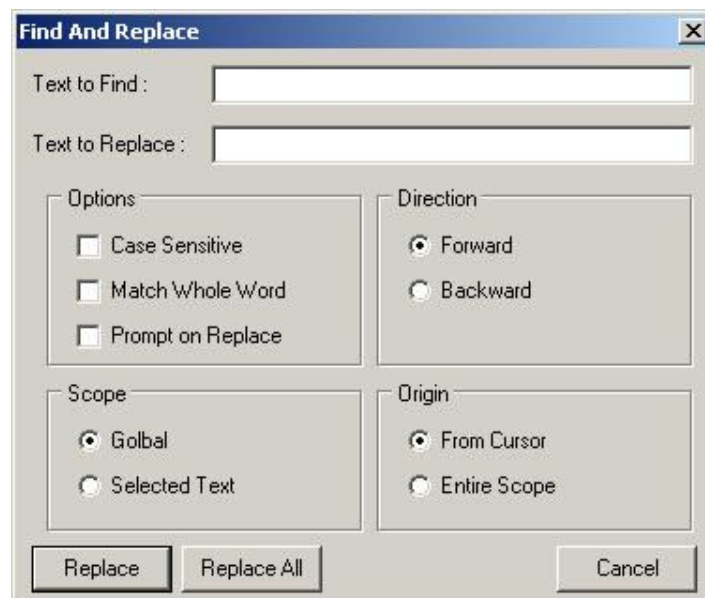


Figure 4.25 Find and Replace -2

4.8 Build a Project

Once you finish the coding, then it is the time to compile the code. Press Build|Build to test the code whether it is correct or not. If there has any error or warning, the output window will show all the error or warning messages. Use the output window to correct the code. When there is no any error in the code, press Build|Build and Download to ICE to prepare to simulate the program.

4.8.1 Build File

“Build File” helps users to check C codes or assembly codes to determine whether they are correct or not. Users can verify if there is any error in code, and then correct it.

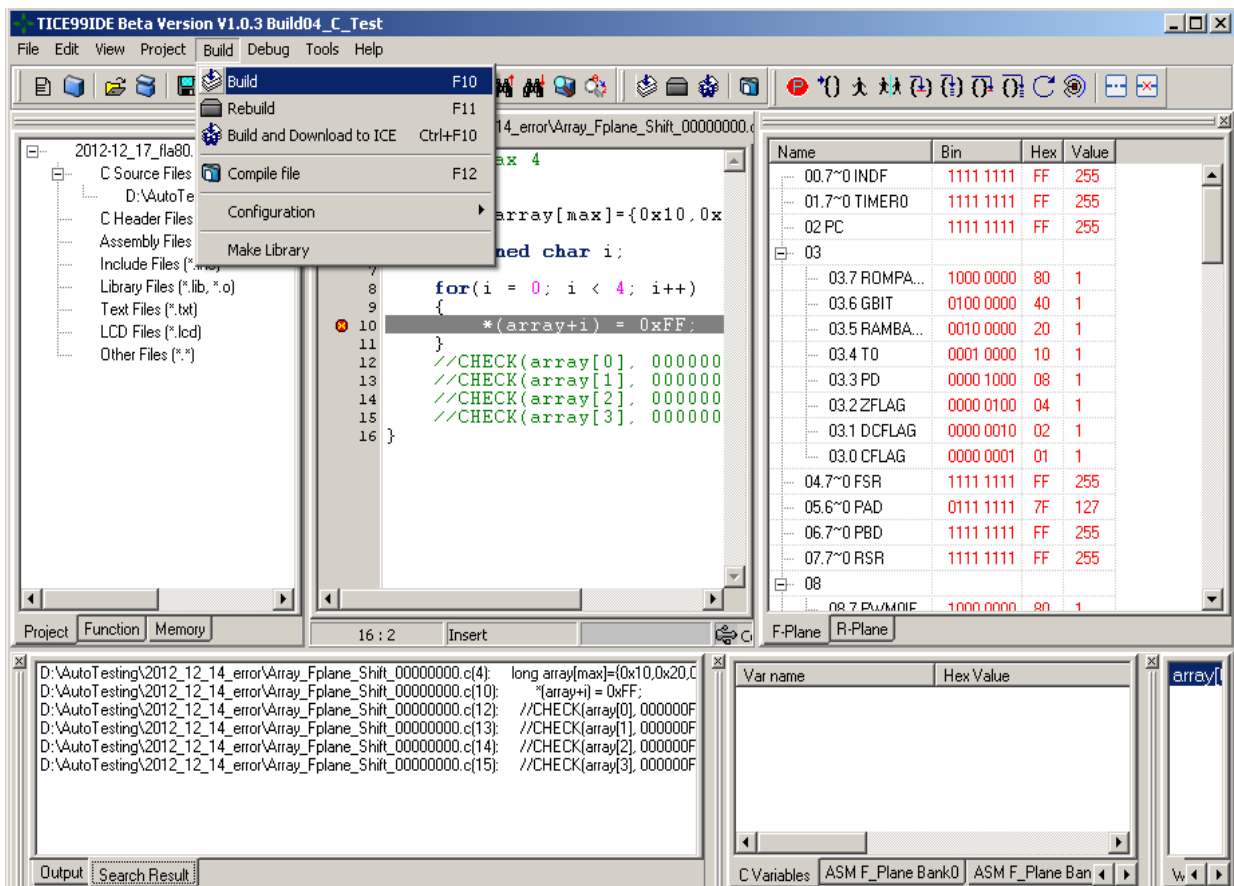


Figure 4.26 Build Project -1

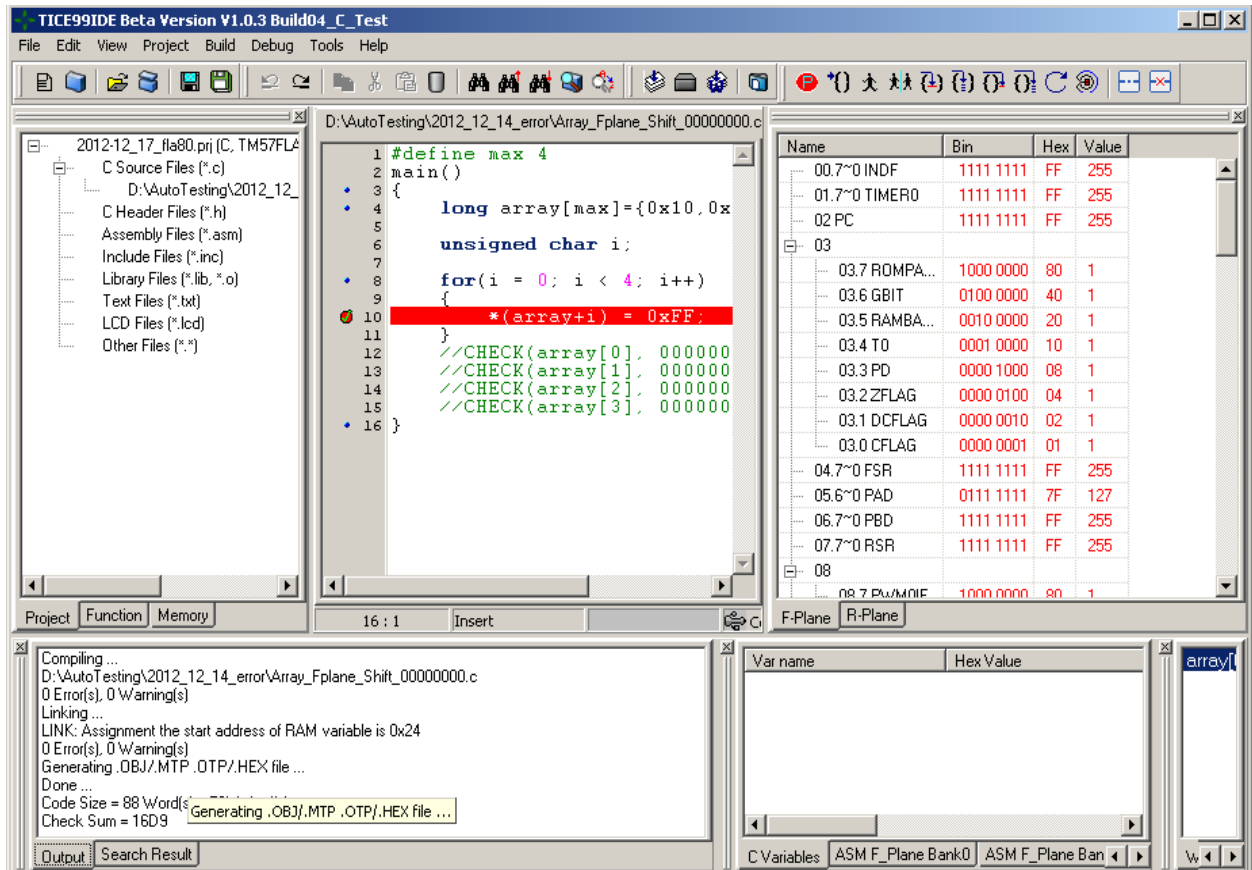


Figure 4.27 Build Project -2

4.8.2 Build and Download to ICE

Once all the codes in one project are correct, you can download the codes to ICE to simulate the action of the codes. Use “Build and Download to ICE” to download the binary execution file to ICE.

Note: Before you download the code to ICE, you have to plug the ICE to PC and turn on the ICE board. Make sure the connection to PC is success.

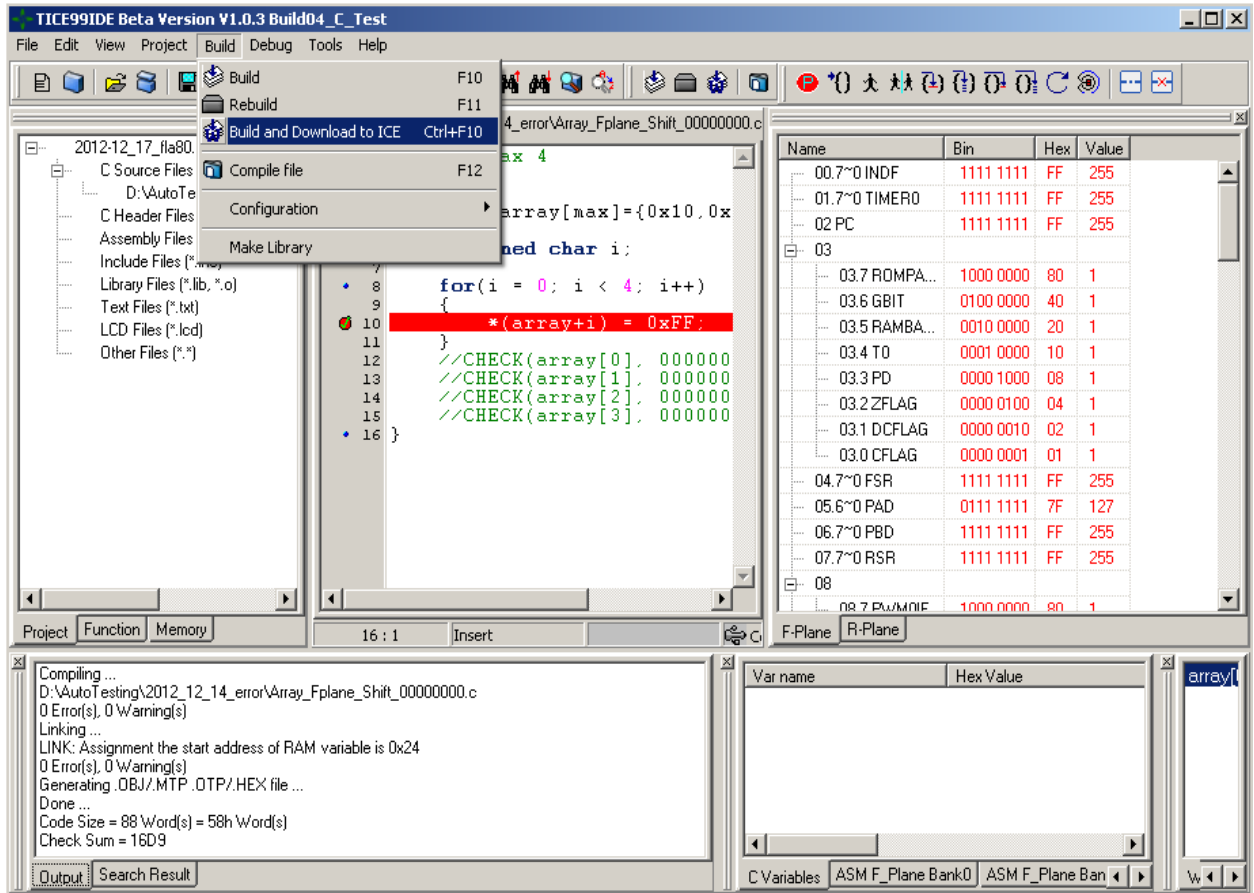


Figure 4.28 Build and Download to ICE -1

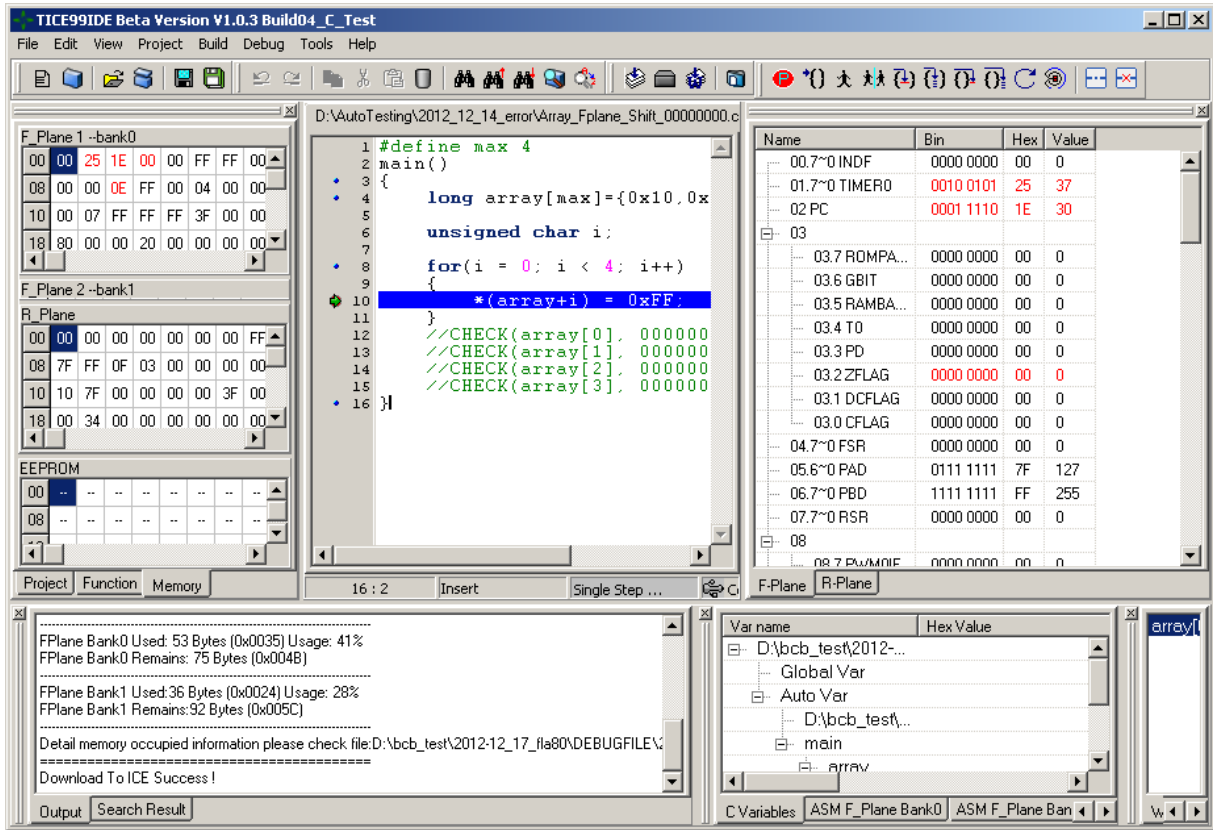


Figure 4.29 Build and Download to ICE -2

4.8.3 Make Library

Libraries are no different than regular source files. All they do is to break the code up into smaller parts so that they can be maintained better and easier to read. TICE99IDE provides a tool to make a library easily. User can link .o file to build a customized library (*.lib). First, the user needs to select an existing or input a new library file name at the field of the "Library File Name". And then press "Add" to select .o file or press "Delete" to delete selected .o file. When pressing "Add" or "Delete", library will prompt message and update immediately.

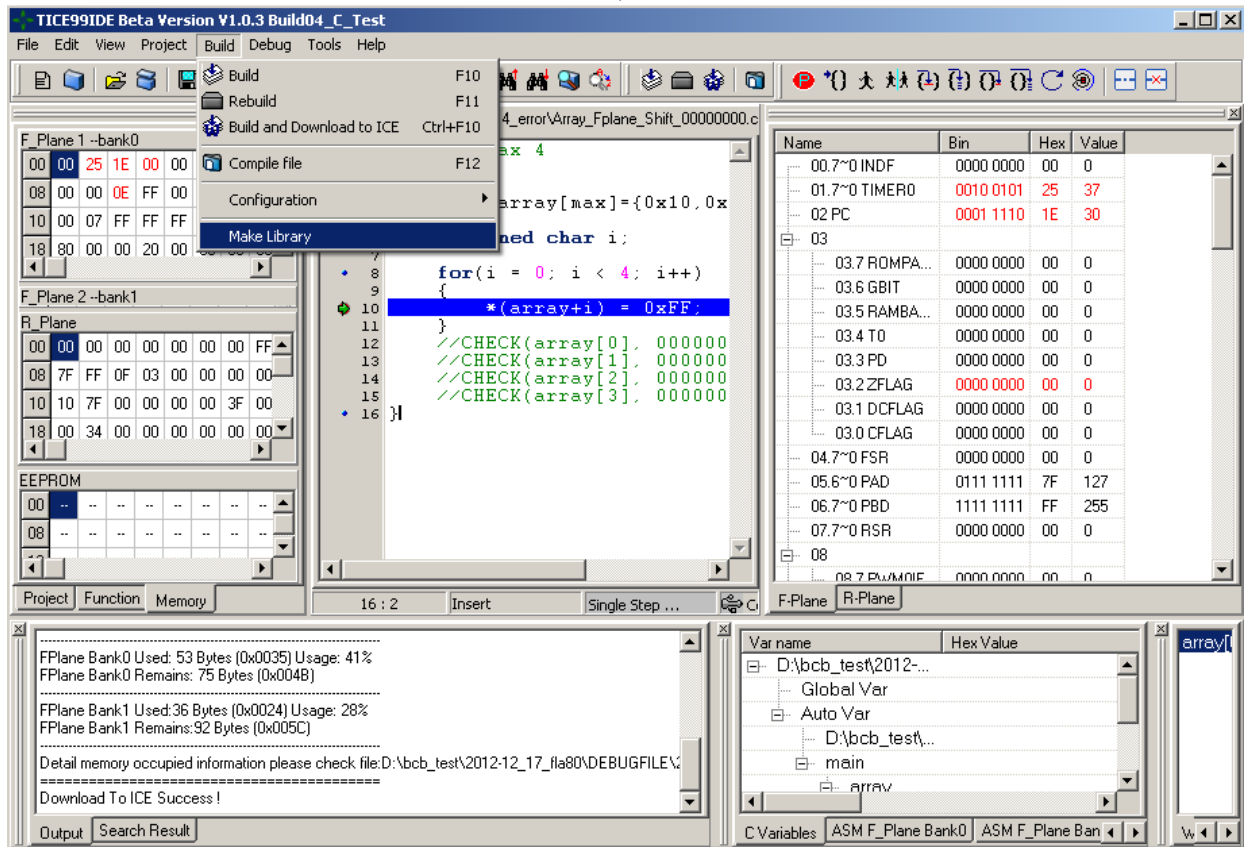


Figure 4.30 Make Library -1

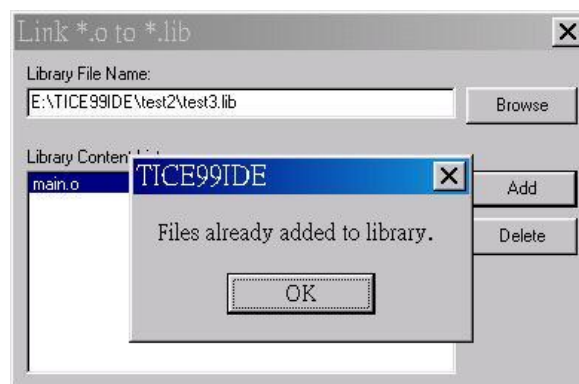


Figure 4.31 Make Library -2

4.9 Debug with Project

TICE99IDE has several kinds of debug, including *Pause*, *Run to Cursor*, *Go*, *Free Run*, *Single Step* and *Step Over*. “Auto Single Step” and “Auto Step Over” are extended from *Single Step* and *Step Over*. It will run “Single Step” or “Step Over” when timer is triggered. Timer can be set in Tools|Options|Debug Options. The Unit of Timer is millisecond.

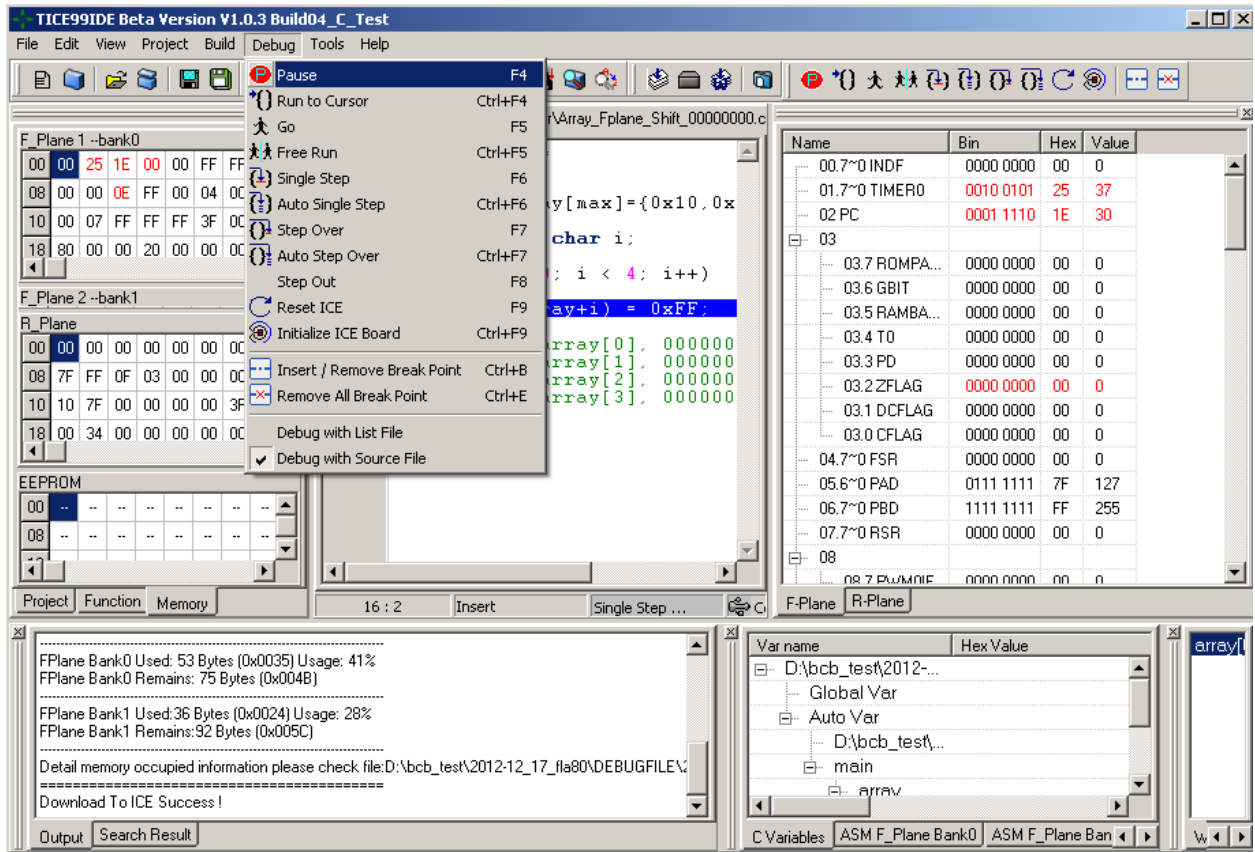
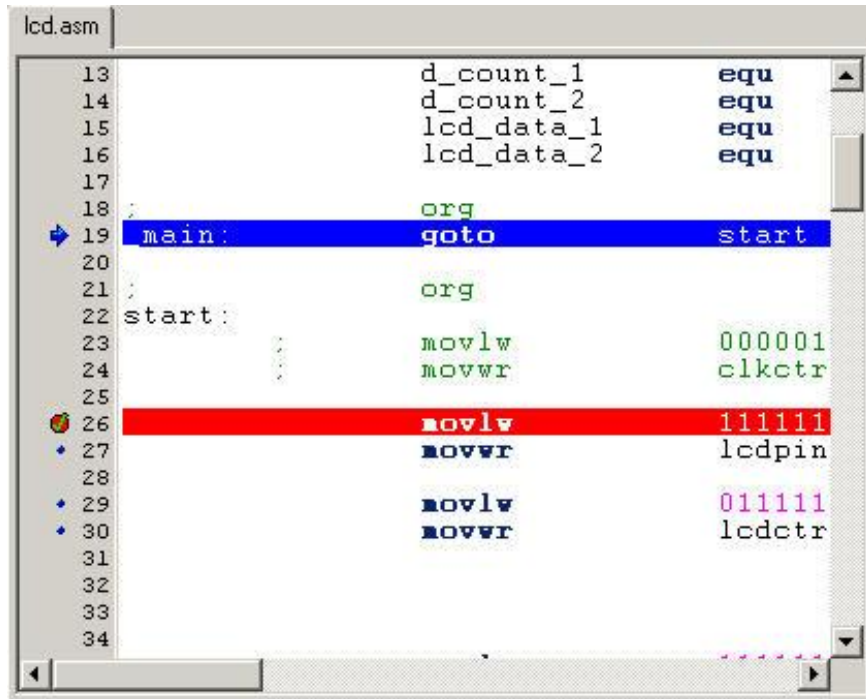


Figure 4.32 Debug function with Project



```
lcd.asm
13      d_count_1      equ
14      d_count_2      equ
15      lcd_data_1     equ
16      lcd_data_2     equ
17
18      org
19 main:  goto         start
20
21      org
22 start:
23      movlw          000001
24      movwr          clkctr
25
26      movlw          111111
27      movwr          lcdpin
28
29      movlw          011111
30      movwr          lcdctr
31
32
33
34
```

Figure 4.33 Debug with Project - Break point

4.10 Switch Language Interface

TICE99IDE now provides three kinds of language interface. They are English, Simplified Chinese and Traditional Chinese. Click Tools|Language to switch the language if needed.

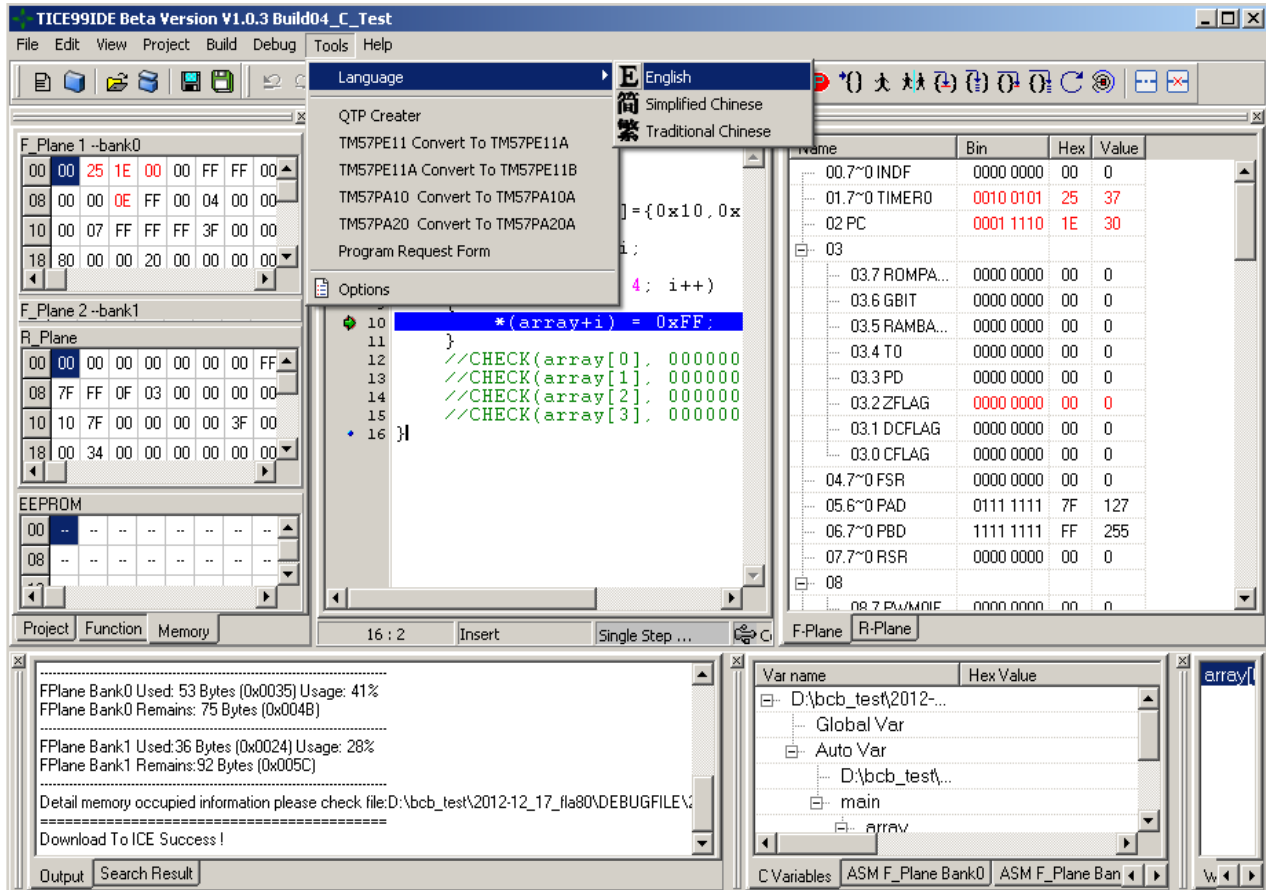


Figure 4.34 Switch Language Interface

4.11 QTP Creater

“QTP Creater” is for output “Coding Request Form” of current project, it includes information that hex file name, checksum, system configuration, total code size and usage rate.

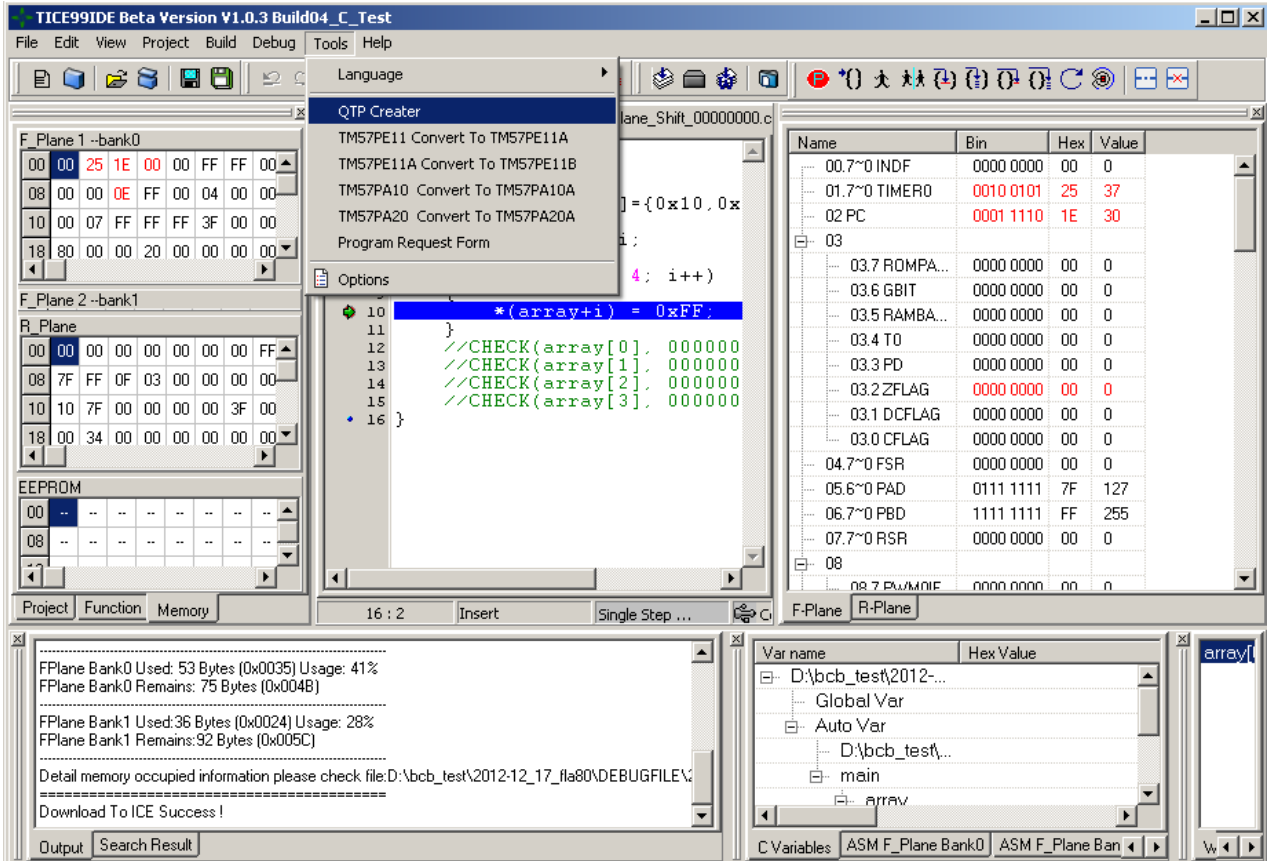


Figure 4.35 QTP Creater

4.12 Hex File Converter

TICE99IDE provides a tool to convert a hex file from one chip type to another type for programming. It will pop up a confirmation dialog after selecting a hex file. Even the conversion is successful, it still has something to re-verify, please pay attention to see the detail.

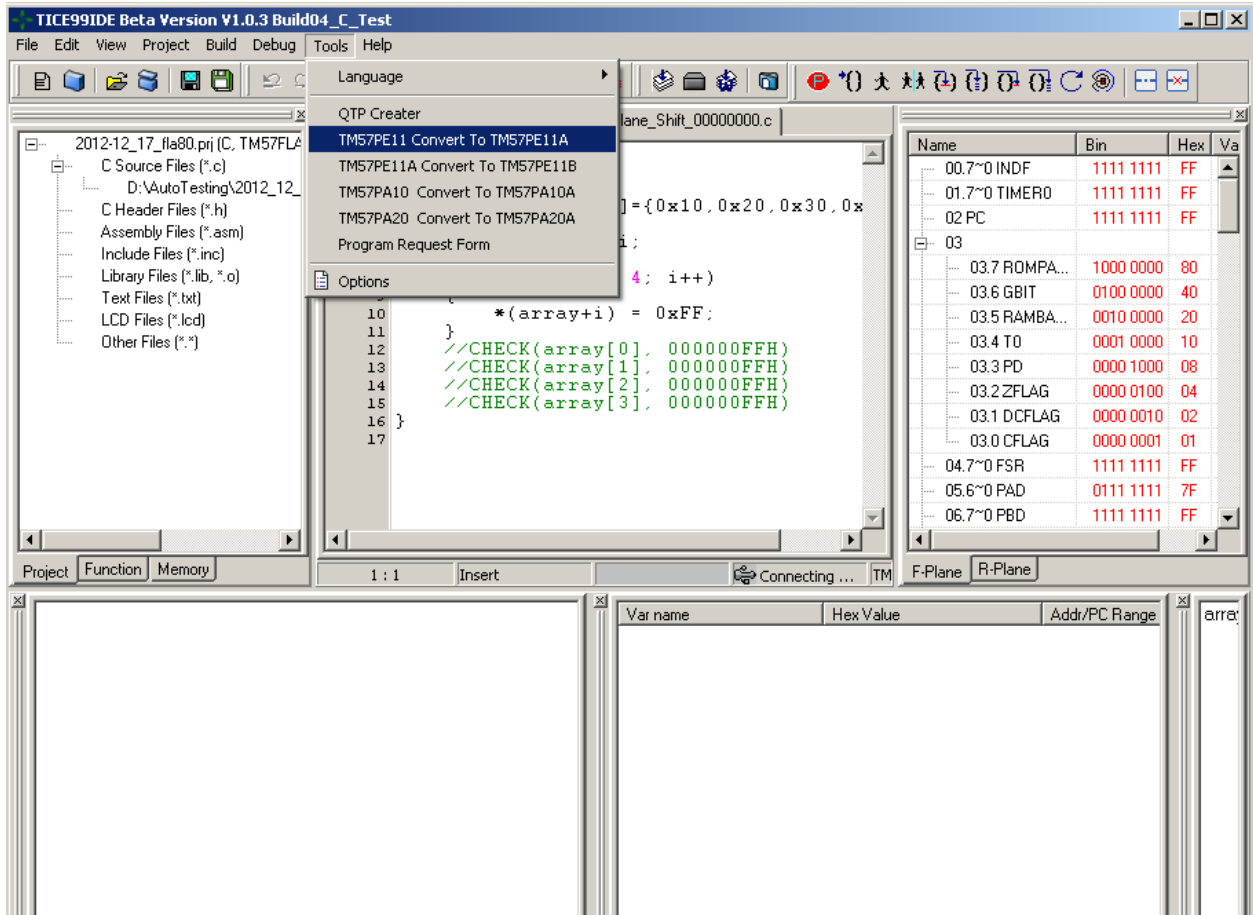
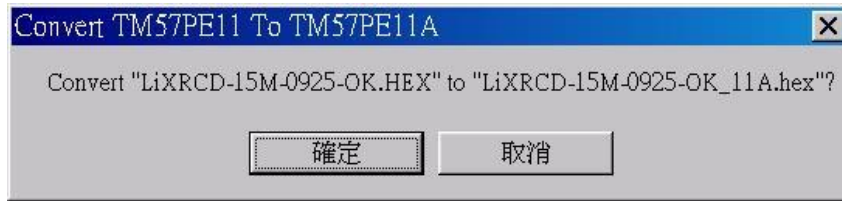


Figure 4.36 TM57PE11 Convert To TM57PE11A -1



4.37 TM57PE11 Convert To TM57PE11A -2

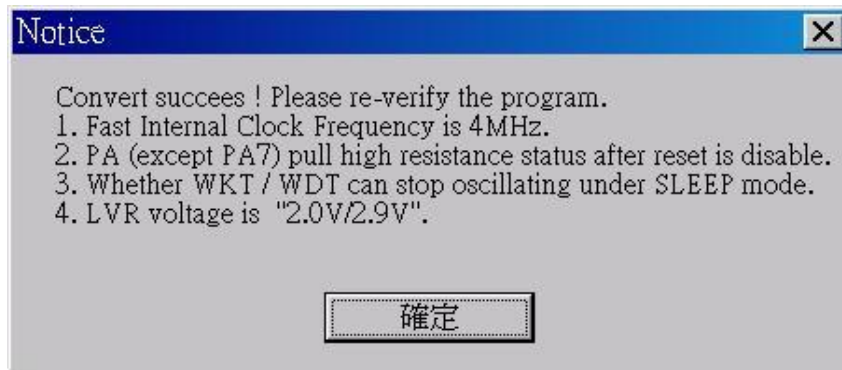


Figure 4.38 TM57PE11 Convert To TM57PE11A -3

4.13 Help

This user manual only introduces how to use the IDE, C compiler and Assembler. If you want to know how to write code in IDE, please refer to the C, Assembler programming guide.

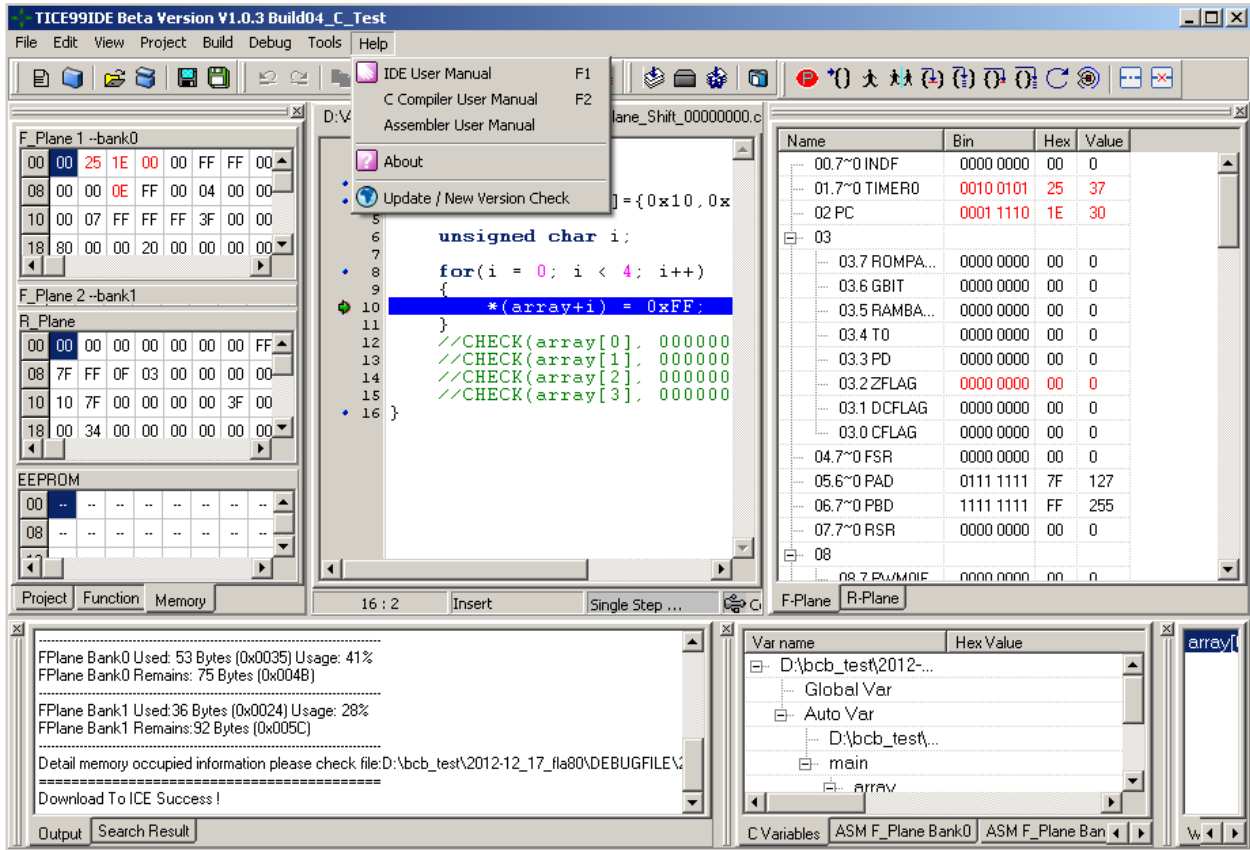


Figure 4.39 Help

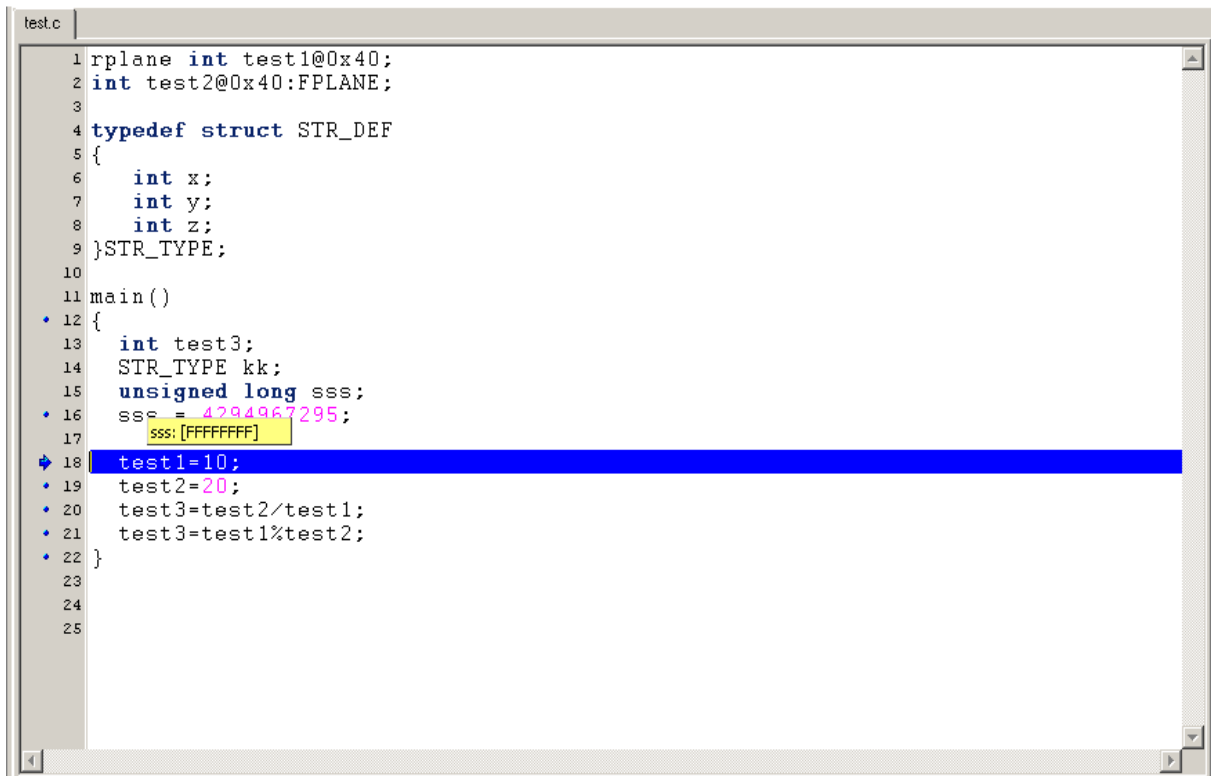
5. Debug

ICE can simulate the status of chip via debug. The registers of ICE will be shown on IDE, which can be divided into memory, registers, ICE stack status and mnemonic code. In the following, we will describe in what situation using the appropriate debug function:

- **Single step / Auto Single Step:** If you want to trace each source code to find out some problems, then using “Single Step”. Single step will run just one program counter step then it will stop. All current statuses or register values are shown on IDE. User can use this information to determine the problem.

If you want to press the button of single step many times, another choice is using “Auto Single Step” to save the times you press the button. Once you want to stop auto single step, press “Pause” to stop it.

- **Step Over / Auto Step Over:** If you do not want to step into a function call, press “Step Over”. Same as “Auto Single Step”, you can use “Auto Step Over” instead of “Step Over” continually. If you want to stop “Auto Step Over”, press “Pause”.
- **Run to Cursor:** You can use “Run to Cursor” to pass the code that you have tested.
- **Reset ICE:** When you want to start debug from beginning, “Reset ICE” will help you to achieve.
- **Go:** You can set break point to stop the ICE when you use “Go” function. Using “Insert or Remove Break Point” to maintain the setting of break point function. One thing should be mentioned is all EV has break point limitation now. The maximum number of break point setting depends on the chip you selected.
- **Debug by source file / debug by list file:** There are two ways to debug the code. One is by source file; the other is by list file. One line in list file means one program counter. However in C program, one line in source will map a range of program counter. If you want to trace source code more detail, our suggestion is using list file. Otherwise, choose source file to debug. Pick one way you feel free.
- **View the variable value when debugging:** When you are debugging and you stop debugging, you can use mouse to point to a variable, and IDE will show a hint message box to display the current value result of this variable. IDE provides a convenient way to verify the value of a variable at any time when debugging.



```
test.c
1 rplane int test1@0x40;
2 int test2@0x40:FPLANE;
3
4 typedef struct STR_DEF
5 {
6     int x;
7     int y;
8     int z;
9 }STR_TYPE;
10
11 main()
12 {
13     int test3;
14     STR_TYPE kk;
15     unsigned long sss;
16     sss = 4294967295;
17     sss: [FFFFFFFF]
18 test1=10;
19 test2=20;
20 test3=test2/test1;
21 test3=test1%test2;
22 }
23
24
25
```

Figure 5.1 Tooltip expression evaluation

6. LCD Editor

LCD provided by the IDE editor is an independent function. LCD editor edits the content which will be displayed in LCD. Some chips support the LCD function. The function of LCD Editor contains three parts: LCD Panel Editor, LCD Attributes as well as LCD Pattern Editor. LCD Panel Editor is main window. However, let's start with LCD Pattern Editor.

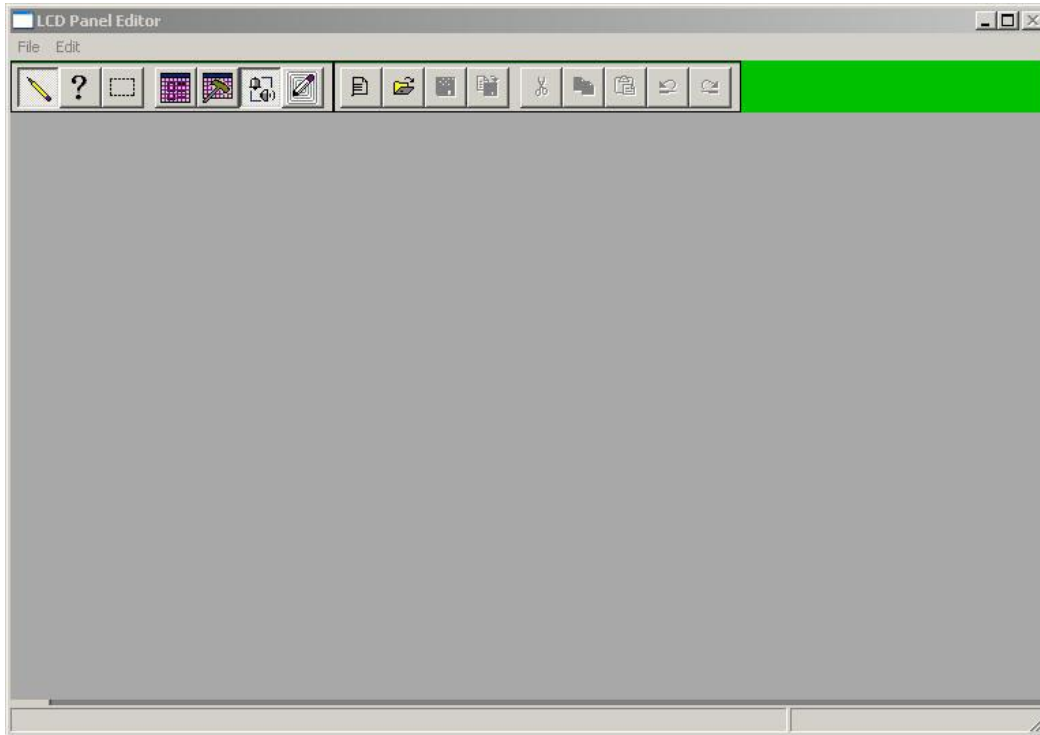


Figure 6.1 LCD Panel Editor

6.1 LCD Pattern Editor

“LCD Pattern Editor” window is an editor which can be used to design some patterns for reuse and save it as a file, and whose default extended file name is .pat. The size range of pattern is from 2 by 2 to 32 by 32. First, you have to decide the size of pattern. Then, use draw tool to place black dot on the pattern you decided in first step. If you place in an error position, you can use eraser to erase it. Other draw tools are ellipse, line, rectangle, and font. Font is convenient tool to let you draw text more easily. By the way, there are five edit functions. They are cut, copy, paste, undo and redo.

After you have finished a pattern, you can use “Save to File” to keep what you draw. The pattern file is a serial image; you can use “Add Panel to Tail” or “Insert Panel” to enrich your patterns or remove current pattern by “Delete”. You can redefine the size of pattern by LCD pattern definition. Developers can use scroll bar to change to the panel that you want to modify or use indicator to input a pattern number to jump to pattern directly.

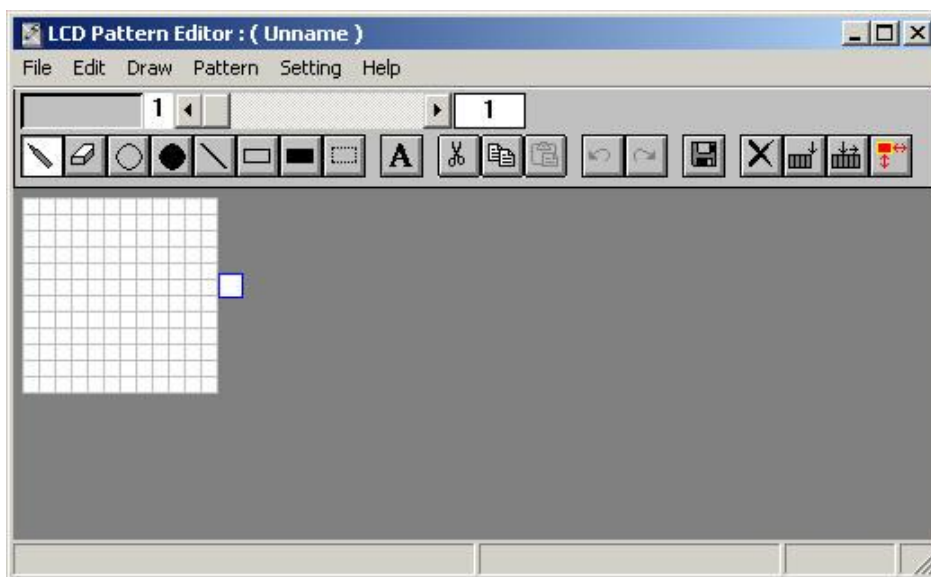


Figure 6.2 LCD Pattern Editor

6.2 LCD Attributes

Once you create a pattern file, you can use “LCD Attributes” window to load the pattern file. Use “Open File” to load the pattern, and it will be shown on patterns list. Select a pattern and switch to LCD Panel Editor window, then you can place the pattern to “LCD Panel Editor” or load external bitmap file to place on LCD Panel Editor. Except this, you can set the value of pattern, they are Segment and Common. The Segment value always represents column; on the other hand, the Common value always represents row. LCD data shows content of LCD RAM.

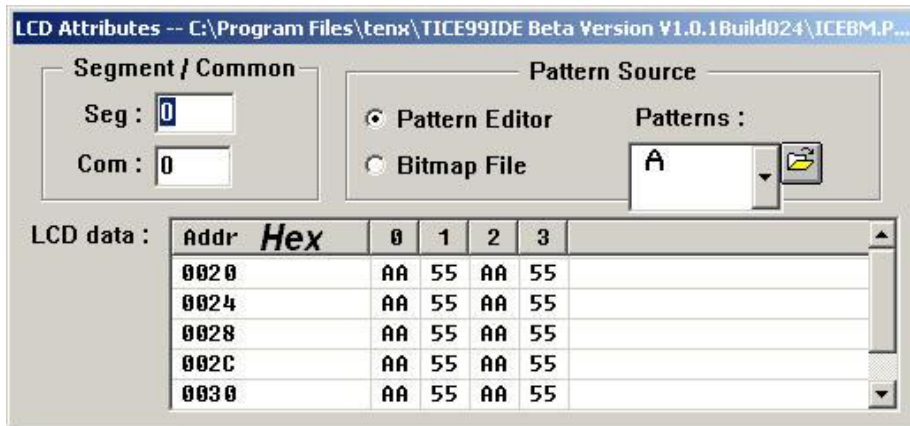






Figure 6.3 LCD Attributes

6.3 LCD Panel Editor

In “LCD Panel Editor”, you can layout the image which will be displayed on the LCD panel connected with you chip. You can use grid  as a ruler to arrange you image. At the same time, you have to set the size of grid . That includes width and height, from 1 to 32. This depends on the pattern size that you draw in LCD Pattern Editor. LCD Panel Editor places the image according to the Pattern or bitmap file according to the switch you selected, pattern editor or bitmap file. If you place the pattern on a wrong position, you can use select pattern  to move to correct place. Select pattern can choose more than one pattern. The other way to move pattern is query attribute. It provides Segment and Common information and you can move the pattern at the same time. Once you finish, use save file  to save LCD file, the extended file name is .LCD. This file can be added into project.

7. Shortcuts

<i>Function</i>	<i>Shortcut</i>
<i>File</i>	
<i>Save</i>	<i>Ctrl + S</i>
<i>Edit</i>	
<i>Undo</i>	<i>Ctrl + Z</i>
<i>Redo</i>	<i>Ctrl + Y</i>
<i>Copy</i>	<i>Ctrl + C</i>
<i>Cut</i>	<i>Ctrl + X</i>
<i>Paste</i>	<i>Ctrl + V</i>
<i>Select All</i>	<i>Ctrl + A</i>
<i>Find</i>	<i>Ctrl + F</i>
<i>Find Previous</i>	<i>Ctrl + F3</i>
<i>Find Next</i>	<i>F3</i>
<i>Find and Replace</i>	<i>Ctrl + R</i>
<i>Go To</i>	<i>Ctrl + G</i>
<i>Build</i>	
<i>Build</i>	<i>F10</i>
<i>Rebuild</i>	<i>F11</i>
<i>Compile File</i>	<i>F12</i>
<i>Build and Download to ICE</i>	<i>Ctrl + F10</i>
<i>Debug</i>	
<i>Pause</i>	<i>F4</i>
<i>Run to Cursor</i>	<i>Ctrl + F4</i>
<i>Go</i>	<i>F5</i>
<i>Free Run</i>	<i>Ctrl + F5</i>
<i>Single Step</i>	<i>F6</i>
<i>Auto Single Step</i>	<i>Ctrl + F6</i>
<i>Step Over</i>	<i>F7</i>
<i>Auto Step Over</i>	<i>Ctrl + F7</i>
<i>Step Out</i>	<i>F8 (RFU)</i>
<i>Auto Step Out</i>	<i>Ctrl + F8 (RFU)</i>
<i>Reset ICE</i>	<i>F9</i>
<i>Initialize ICE</i>	<i>Ctrl + F9</i>
<i>Insert / Remove Break Point</i>	<i>Ctrl + B</i>
<i>Function</i>	<i>Short cut</i>
<i>Help</i>	<i>F1</i>

Table 7.1 Shortcuts

Appendix A: Meaning of LED on ICE board

- V5.0: Power from AC Adapter.
- DONE: FPGA download file success, ICE can work properly.

Appendix B: IC List

EV Board	Chip	ROM	RAM(byte)
EV1655	TMU32FA80	8K*14 Flash	160+192
EV1694	TMU3132MS	4K*14 MASK	160*8+128*4
EV1697, EV1694	TMU3130	8K*14 Flash	160*8+128*8
	TMU3132	4K*14 MTP	160*8+128*4
	TMU3131	6K*14 MTP	160*8+128*4
	TMU3132LV	4K*14 MTP	160*8+128*4
	TMU3132F	4K*14 MTP	160*8+128*4
EV2767	TM57PA28	2K*14 OTP	176
EV2768	TM57PA45	4Kx14 OTP, or 2Kx14 TTP	184
	TM57PT45	4Kx14 OTP, or 2Kx14 TTP	184
	TM57PA45C	4Kx14 OTP, or 2Kx14 TTP	184
	TM57PT45C	4Kx14 OTP, or 2Kx14 TTP	184
EV2769	TM57MT20	2K*14 MTP	184
EV2771	TM57ME16	1K*14 MTP	48
	TM57ME18	1K*14 MTP	48
	TM57ME16AS	1K*14 MTP	48
EV2773	TM57PA15	1K*14 OTP	64
EV2774	TM57PE20A	2K*14 OTP	184
	TM57PT20A	2K*14 OTP	184
	TM57PE20B	2K*14 OTP	184
	TM57PT20B	2K*14 OTP	184
EV2777	TM57PA11	1K*14 OTP	48
EV2780	TM57PE11BS	1K*14 OTP	48
	TM57PE11B	1K*14 OTP	48
	TM57PE11C	1K*14 OTP	48
	TM57PE11CS	1K*14 OTP	48
	TM57P11CU	1K*14 OTP	48
	TM57P11C	1K*14 OTP	48
EV Board	Chip	ROM	RAM(byte)
EV2781	TM57MA21B	2K*14 MTP	184
	TM57MA25	2K*14 MTP	184
	TM57MA28	2K*14 MTP	96
	TM57MA28B	2K*14 MTP	96
	TM57MA28MB	2K*14 MTP	96
	TM57MA29	2K*14 MTP	96
	TM57MA29C	2K*14 MTP	96
EV2783	TM57MR10	1K*14 MTP	48

EV2784	TM57PA25B	2K*14 OTP	176
EV2785	TM57ML40	4K*14 MTP	368
EV2786	TM57PE12	1K*14 OTP	48
	TM57PE12AS	1K*14 OTP	48
	TM57P12	1K*14 OTP	48
	TM57RE12	1K*14 MASK	48
	TP6717	1K*14 OTP	48
	TM57PE16	1K*14 OTP	48
	TM57PE12D	1K*14 OTP	48
EV2786, EV2786B	TM57PE10	1K*14 OTP	48
EV2786B	TM57RE12A	1K*14 MASK	48
	TM57PE15A	1K*14 OTP	48
	TM57PE15AS	1K*14 OTP	48
	TM57PE15C	1K*14 OTP	48
	TM57PE15CS	1K*14 OTP	48
	TM57P15AU	1K*14 OTP	48
EV2787	TM57PE40	4K * 14 OTP or 2K * 14 TTP(Two Time Programmable)	176
EV2788	TM57ME20	2K*14 MTP	96
	TM57ME20A	2K*14 MTP	96
EV Board	Chip	ROM	RAM(byte)
EV2793	TM57PA10	1K*14 OTP	64
	TM57PA10A	1K*14 OTP	64
	TM57PE11A	1K*14 OTP	48
	TM57P11	1K*14 OTP	48
EV2795	TM57FA40	4K*14 Flash	184
	TM57FA40A	4K*14 Flash	184
	TM57PA20	2K*14 OTP	184
	TM57PA40	4K*14 OTP	184
	TM57PA20A	2K*14 OTP	184
	TM57PA20B	2K*14 OTP	184
	TM57PA20E	2K*14 OTP	184
	TM57PA40E	4K*14 OTP	184
EV2796	TM57FLA80	8K*14 Flash	368
	TM57FLA80A	8K*14 Flash	368

EV Board	Chip	ROM	RAM(byte)
EV8201	TM57PA16	1K*14 OTP	48
	TM57PT16	1K*14 OTP	48
	TM57PA16B	1K*14 OTP	48
	TM57PT16B	1K*14 OTP	48
	TM57PA16AS	1K*14 OTP	48
	TM57PT16AS	1K*14 OTP	48
EV8205	TM57MA15	1K*14 MTP	96
	TM57MA33	1K*14 MTP	96
	TM57MA16	1K*14 MTP	96
	TM57MA1668	1K*14 MTP	96
	TM57MA1672	1K*14 MTP	96
EV8206	TM57MT21	2K*14 MTP	184
	TM57ME21	2K*14 MTP	184
EV8209	TM57M5526C	2K*14 MTP	176
	TM57M5536C	2K*14 MTP	176
EV8212	TM57M5541	4K*14 MTP	368
	TM57M5551	4K*14 MTP	368
EV8214	TM57MA45	4K*14 MTP	184
	TM57MA46	4K*14 MTP	184
EV8215	TM57MA17	1K*14 MTP	96
	TM57MA18	1K*14 MTP	96
EV8216	TM57MT21A	2K*14 MTP	184
EV8217	TM57ME15B	1K*14 MTP	48
	TM57ME15CG	1K*14 MTP	48
EV8222	TM57M5615	1K*14 MTP	64
	TM57M5610	1K*14 MTP	64
EV8225	TM57M5406	2K*14 MTP	368
	TM57M5408	2K*14 MTP	368

EV Board	Chip	ROM	RAM(byte)
EV8226	TM56F5412	4K*16 MTP + EEPROM 128	256
	TM56F5416	4K*16 MTP + EEPROM 128	256
EV8228	TM57M5645	4K*14 MTP	336
	TM57M5640	4K*14 MTP	336
	TM57M5625	2K*14 MTP	176
	TM57M5620	2K*14 MTP	176
EV8230	TM55M8228	2K*16 MTP	256
	TM55M8428	2K*16 MTP	256
	TM55M8428T	2K*16 MTP	256
EV8235	TM56F8225	2K*16 MTP + EEPROM 128	176