十速科技股份有限公司
tenx technology inc.

TICE99IDE

使用手册

tenx technology inc.

Rev 1.1, 2013/01/28

修改记录

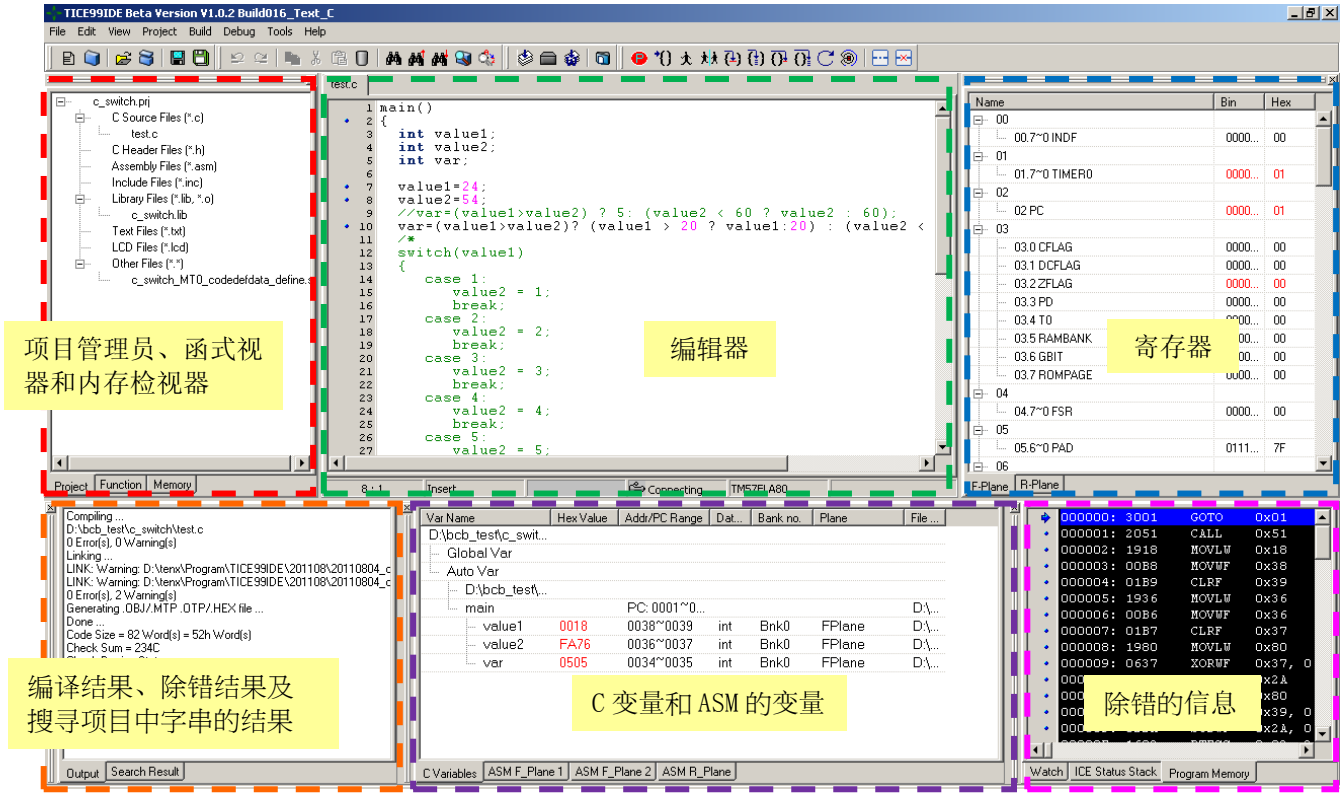
版本	日期	说明
V1.0	08.2011	新颁.
V1.1	12.2012	1. 增加 eeprom 2. 更新程序画面 3. 修改用词 4. 增加程序内存使用信息 5. 增加删除查找的说明 6. 增加了程序内存的列表调试 7. 修改产生函式库和增加概念流程图 8. 修改 2.7.5 标题为"16 进制档案转换器", 和增加芯片列表 9. 增加 C 编译器/汇编器使用手册 10. 修改新项目的程序画面 11. 修改有支持的芯片列表 12. 修改项目设定和程序画面

目录

1. 简介

TICE99IDE 为十速科技所提供的一整合开发环境（IDE），利用此开发工具可让使用者快速开发十速科技芯片应用程序。目前 TICE99IDE 开发软件只支持 TM57 系列芯片。IDE 包含四个组件：编辑器、项目管理员、编译器、仿真电路的调试器。TICE99IDE 所提供的编辑器可让使用者编辑、修改程序代码。项目管理员用以管理项目里面的所有文件，包含 C 和 ASM 文件。编译器对使用者输入的程序代码进行编译程序。ICE 可以仿真 ROM 程序执行的结果。在仿真的过程中，计算机设备必须与 TICE99 硬件连结。以下我们将针对此四组件在 IDE 窗口的应用，提供更详细的说明。

IDE 窗口的主要组件可区分为四个部分来说明。第一部分是 IDE 窗口的左上角，这包含三项功能窗口：项目管理员、函数视图器和内存视图器。右上角的部分只包含寄存器(F-Plane，R-Plane)的页面。中间的区域是编辑器部分，可供您编辑及修改程序代码。IDE 的下方，可区分为三个部分。左下方是编译(compiler)，连结(linker)程序显示讯息的部分，包含编译结果、调试结果及查找项目中字符串的结果。中间部分是变量的区域，包含 C 变量和 ASM 的变量。ASM 变数可分为三个部分：F-Plane1、F-Plane2 及 R-Plane。这部分的实时内容值只有当使用者开始进行调试时，才会自动更新显示寄存器的内容或 C 变量值。右下方部份是调试的信息，包含三个页面：watch（变量查看）、ICE status stack（ICE 堆栈状态）、程序内存(图表 1)。



图表 1 TICE99IDE 主窗口

1.1 主菜单

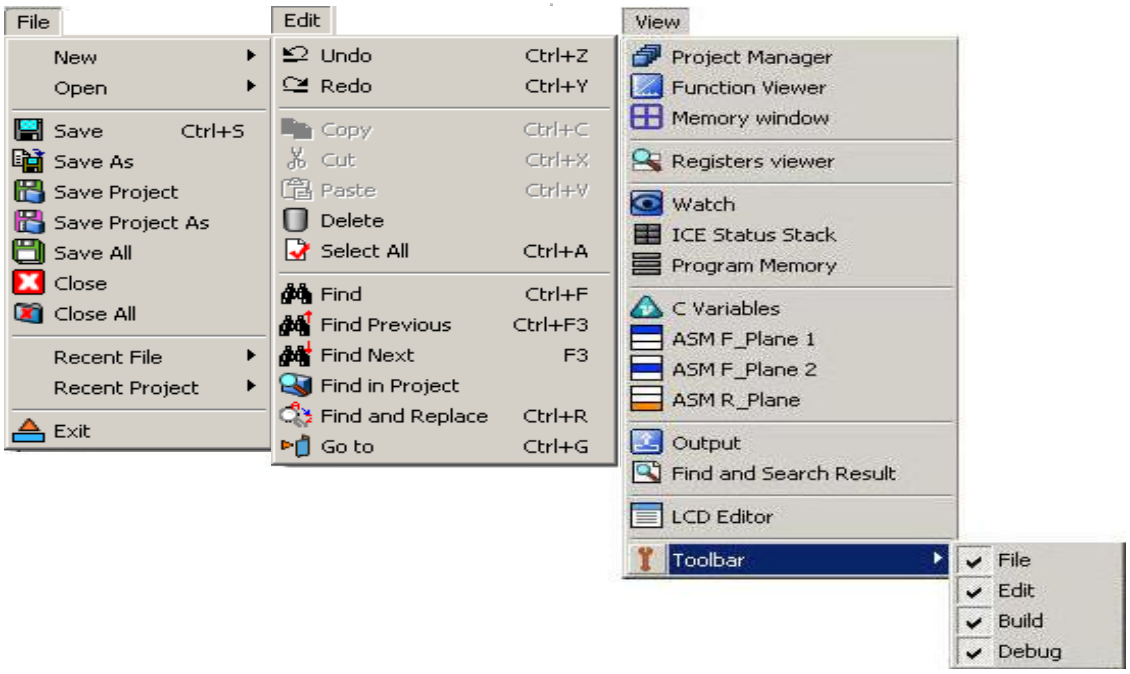
当首次执行 TICE99IDE 时，映入您眼帘的是 IDE 所提供的主要工具列 - 主菜单。主菜单提供了所有 IDE 的功能。第一层主菜单包含「文件」、「编辑」、「视图」、「项目」、「建立」、「调试」、「工具」和「帮助」。

在「文件」菜单，您可以建立新的或开启已存在的项目或文件。关闭文件或关闭项目后，您可以在各个存盘中再度开启文件或项目。甚至您可以藉由「最近开启文件」及「最近开启项目」所列示您之前所开启过的文件或项目，来开启文件或项目。您也可以点选「退出」的按键来关闭 TICE99IDE；并由跳出的询问窗口来决定是否要保存所修改的项目。

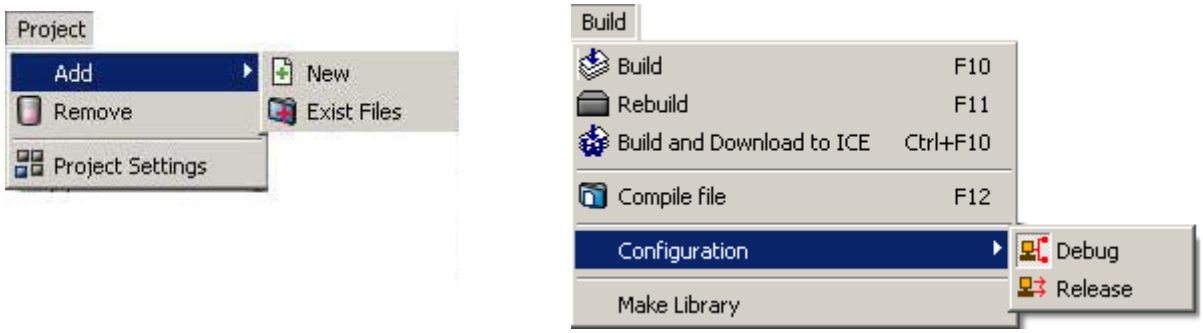
「编辑」菜单提供了编辑器的基本功能和高级功能。基本功能包括：复制、粘贴、剪切、删除、全选、复原、重做。高级功能包括：查找、替换、在项目中搜索和直接跳转到所指定的编辑器列号(图表 2)。

在「视图」菜单，您可以选择切换所要视图的窗口，包含项目管理员，函数视图器、内存窗口、寄存器窗口、ICE 堆栈状态、输出、查看、LCD 编辑器及工具栏(图表 2)。

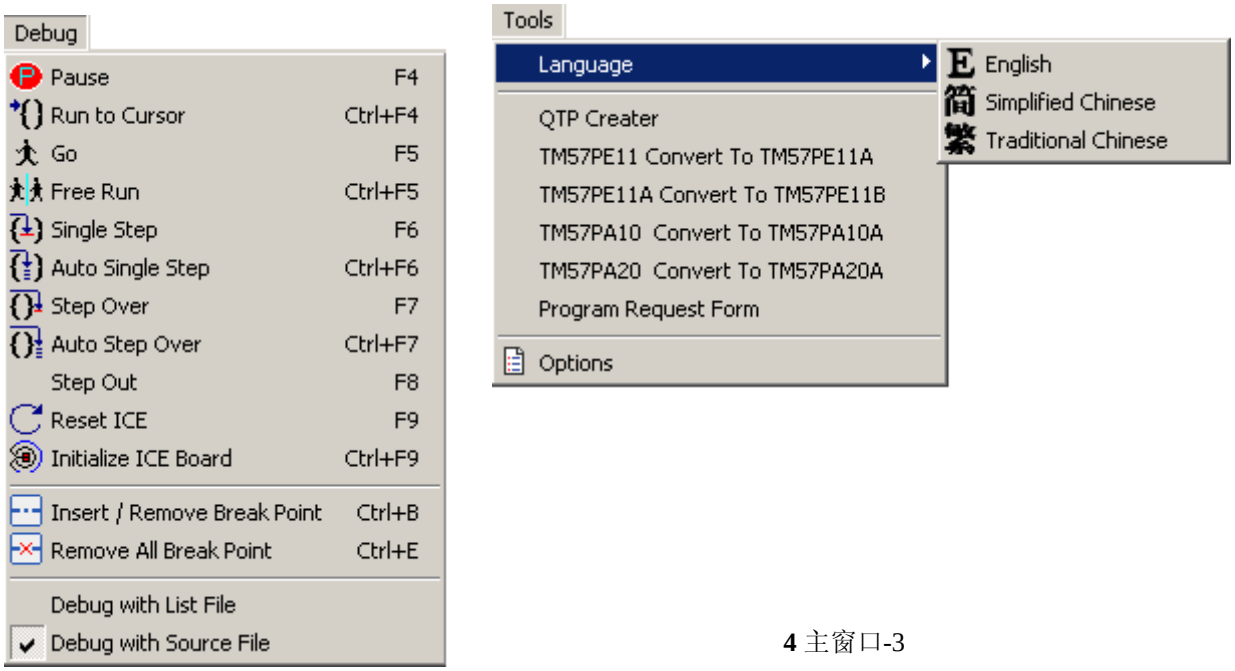
「项目」包含新建、删除和项目设置。「建立」是执行编译器或汇编器将原始程序文件编译成对象文件(.o 文件)及执行档(.bin 文件)并选择是否要下载传入 ICE(图表 3)。「调试」可用来追踪原始程序，并试着找出错误的问题点(图表 4)。在「工具」，您可以动态切换语言界面、产生 QTP 文件、将 TM57PE11 程序代码转换为 TM57PE11A 程序格式及设定编辑器的选项。在「帮助」菜单，可找到使用手册包含 IDE，C 编译器和汇编器、关于 IDE 软件相关的主要执行文件之版本序号和在线实时检查及更新最新软件。(图表 5)



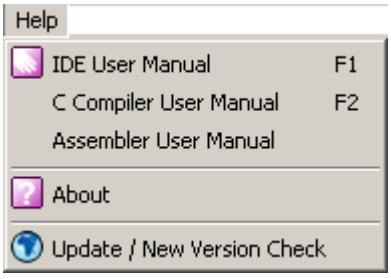
图表 2 主窗口-1



图表 3 主窗口-2



图表 4 主窗口-3



图表 5 主窗口-4

1.2 工具列

工具列提供一快捷方式以快速执行主菜单各个功能，其几乎包含大部份主菜单的功能。使用者除了可以在主菜单的下拉式菜单中选择其所想要执行的功能，也可以直接点选工具列上的按钮。

图表 6 工具列

文件 (File)	
	新建一空白文件。
	新建一项目。
	开启一个已存在的原始程序文件。
	开启一个已存在的项目。
	保存目前正在编辑的文件。
	保存所有文件。

编辑 (Edit)	
	返回上一次的修改。
	重作上一次被返回的修改。
	复制所选取的文字。
	剪切所选取的文字。
	粘贴所复制的文字。
	删除所选取的文字。
	查找目前正在编辑中的字符串。
	依上回查找字符串结果，向上查找。
	依上回查找字符串结果，向下查找。
	在所在的项目下，查找所有文件中所指定的字符串。
	查找目前正在编辑文件中的字符串并以指定字符串替代。

建立(Build)	
	编译项目中的原始程序文件。
	删除所有建立的输出档，并重新建立项目中的原始程序文件。
	编译当前项目的所有文件，并将所产生出的bin或o档下载到ICE。
	编译目前正在编辑的原始程序文件。

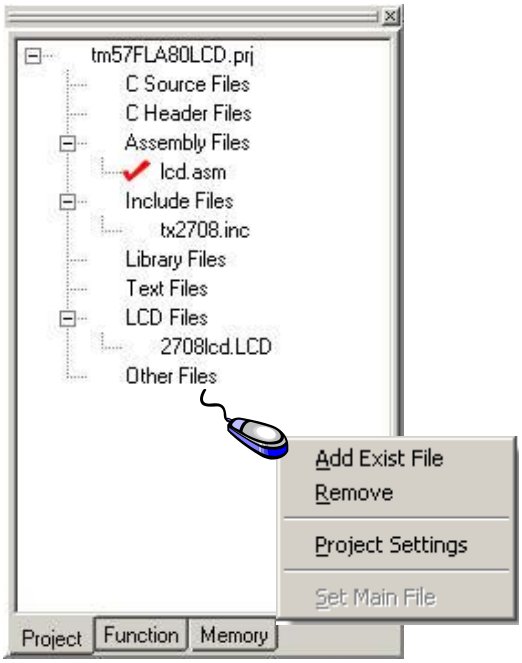
调试 (Debug)	
	暂停所有调试步骤。
	将调试程序运行到光标所指位置。
	运行，运行下载到ICE的程序代码，但不理会断点。
	自由运行，运行下载到ICE的程序代码，若有设定断点则停在该点上。
	调试时，单步运行一个机器码(.asm檔)或一条程序代码(.c檔)。
	自动单步调试，由IDE依使用者所设定的执行速率自动执行「单步执行」。
	单步不进入函数，将函数视为一完整单元。
	自动单步不进入函数。
	重设ICE程序计数器(Program counter，PC)。
	对ICE板进行初始化程序。
	新增/移除断点。
	移除所有已设定的断点。

1.3 项目管理

使用者可利用项目管理来集中管理其所选择的芯片属性文件、相关链接库文件和原始程序文件。项目管理的文件类型可区分为以下组别：原始文件（.c 文件）、汇编语言文件（.asm 文件）、表头文件（.h 文件）、包含文件（.inc 文件）、链接库文件(.lib 文件)、文字文件（.txt 文件）、LCD 文件和其它的文件。您可以鼠标双击文件名以开启该文件，若文件已开启，IDE 将自动切换窗口到对应的文件。

当程序设计者新建一文件且在未保存文件前，其文件名将暂时被命名为 Unit*，*意指 1~n 流水号。在编辑完程序并保存该文件时，「另存新文件」的对话窗口将自动显示，让使用者为该文件存为.c 或其它类型的文件格式并给予一有意义的文件名称。存盘后，项目管理员并不会分类您所保存的文件到相对的类别。您需手动增加已存在的文件到已开启的项目，以便节省编写程序的时间和重复利用程序代码。

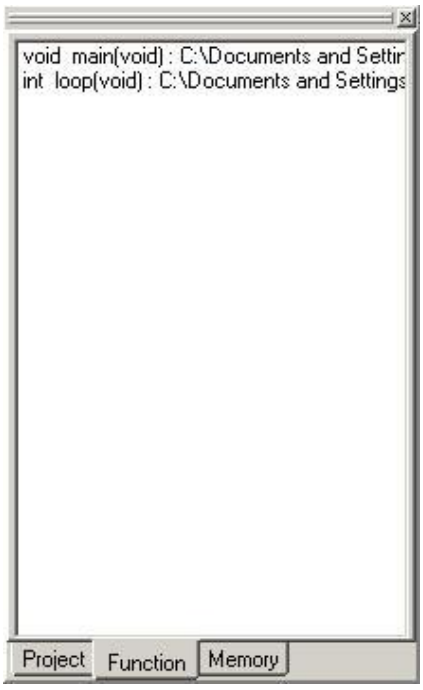
在项目管理页面范围内，单击鼠标右键将触发一功能窗口。利用此快捷功能窗口，使用者可新增已存在的文件到一项目、移除项目中某一选取的文件、设定项目属性或设定主要文件(只针对汇编类型的项目支持此功能)。当其一文件被设定为主要文件，其文件名左侧将显示✔图标以识别(图表 7)。



图表 7 项目管理员

1.4 函数视图器

函数视图器只支持您项目里面的.c 文件。当您编译(build)该项目，程序管理员将自动条列出，在该项目里所有的.c 文件中其所定义的程序模块。您可以鼠标左键双击某一程序模块，它会跳转到该程序模块定义所在的原始程序代码文件之编辑行列；或该文件尚未开启，系统将自动将该文件编辑器开启后，再跳转到对应的行列。我们现在只支持句子开头为 int、char、float、double、void 和指针版本的函数定义句子。我们将在下一个版本支持使用者定义的结构。

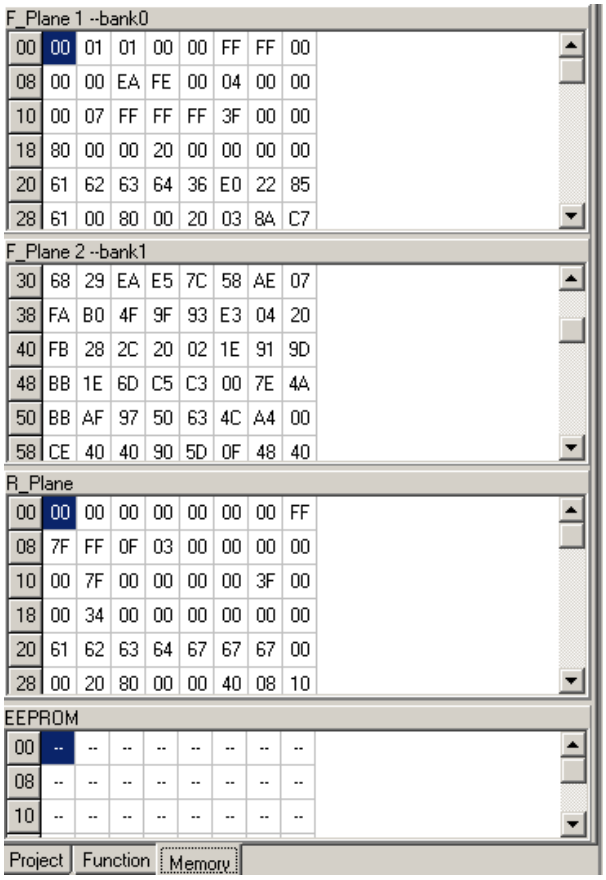


图表 8 函数视图器

1.5 内存

内存的版面显示 R-Plane、F-Plane1、F-Plane2 和 EEPROM 区域之所有内存内容值。当对汇编程序进行调试时，将同时显示以上四个内存区域的内容值。内存内容值的颜色可以显示目前步骤和前一步骤的执行结果值是否相同。若两步骤间的结果值不同，颜色会是红色，否则，维持为黑色。当某些内存空间在该芯片类型是不可使用的，则此区域将显示为“--”。

在进行调试中且 ICE 为暂停的状态时，使用者可以鼠标左键双击某一内存地址，以对此一地址内容值做直接修改(若该地址的内容值是允许被修改的)。一旦完成修改，其新修改的内容值将马上反映到相对的内存地址上。

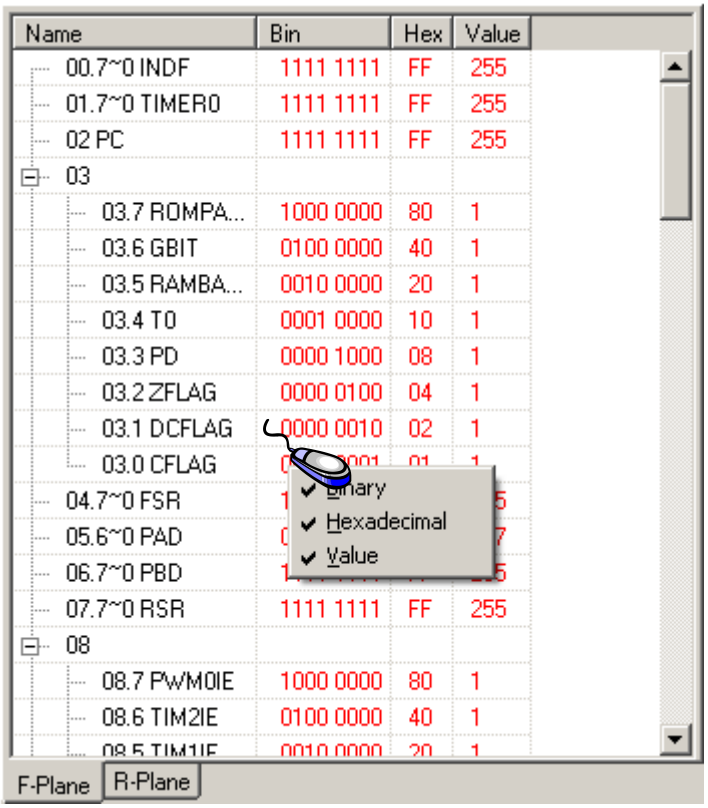


图表 9 内存

1.6 寄存器

寄存器的版面可以显示寄存器的内容。在窗口内以树状形式列出寄存器地址、名称和其包含值。如同内存的显示一般，当内容值被修改时，颜色会被改为红色，反之相同时，其颜色为黑色。您可以在进行项目调试的过程中修改寄存器的内容(若该寄存器内容值是允许被修改时)。

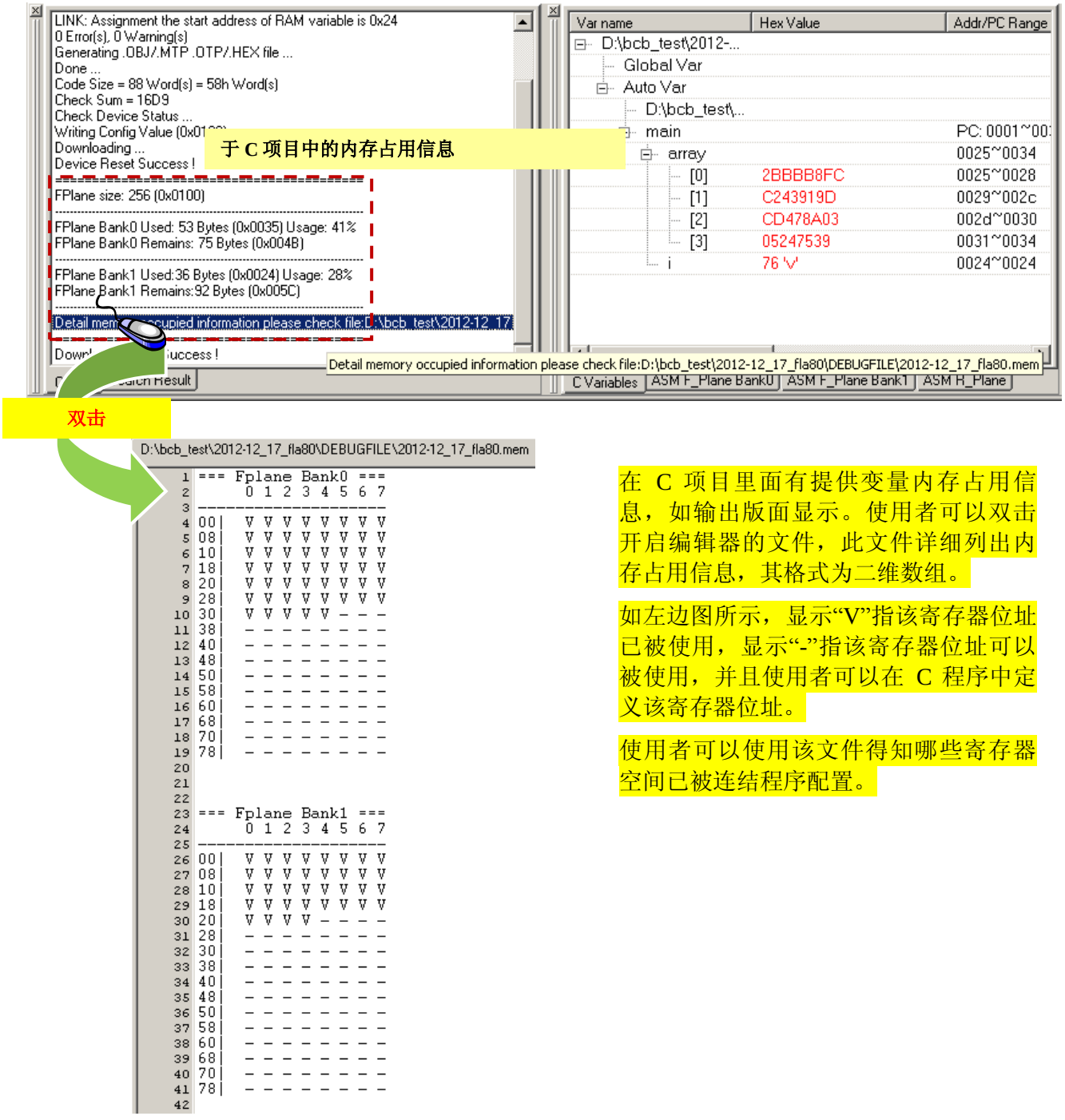
在寄存器页面范围内，单击鼠标右键将触发一功能窗口。利用此快捷功能窗口，使用者可选择性决定要显示的字段：二进制、十六进制或数值字段。使用者可依显示需求对树状中的各节点进行折叠或展开动作。一旦折叠树节点时，该范围的寄存器值将只显示在十六进制字段中。



图表 10 寄存器

1.7 输出

输出的版面显示：编译、汇编及连结原始程序代码所输出的结果信息。信息种类包含警告和错误信息。使用者可双击某警告或错误的行列，系统将自动跳转到所对应原始程序代码的该问题程序行。



在 C 项目里面有提供变量内存占用信息，如输出版面显示。使用者可以双击开启编辑器的文件，此文件详细列出内存占用信息，其格式为二维数组。

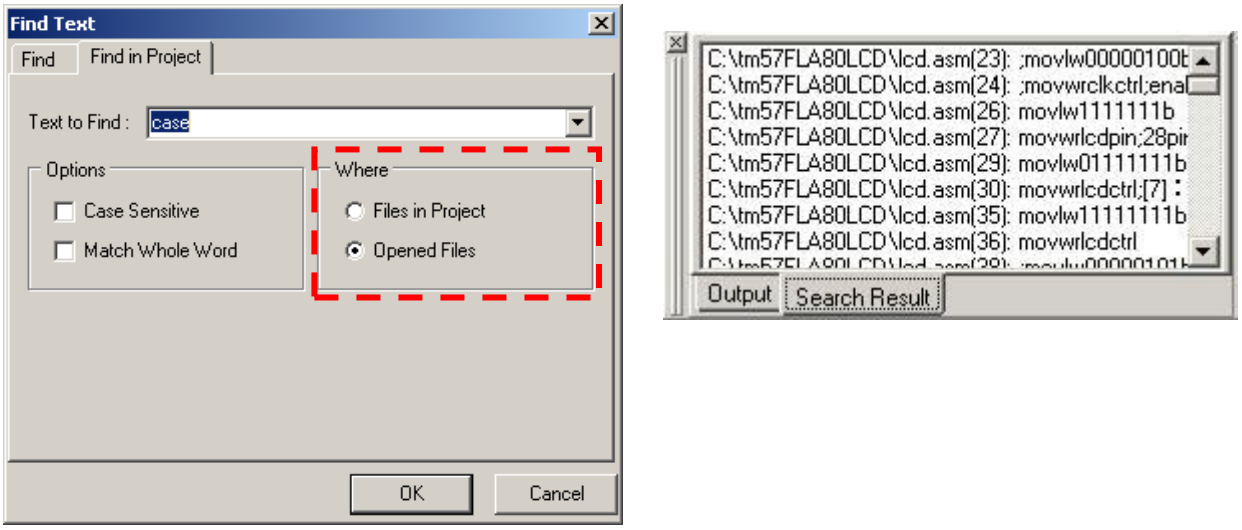
如左边图所示，显示“V”指该寄存器位址已被使用，显示“-”指该寄存器位址可以被使用，并且使用者可以在 C 程序中定义该寄存器位址。

使用者可以使用该文件得知哪些寄存器空间已被连结程序配置。

图表 11 输出

1.8 查找结果

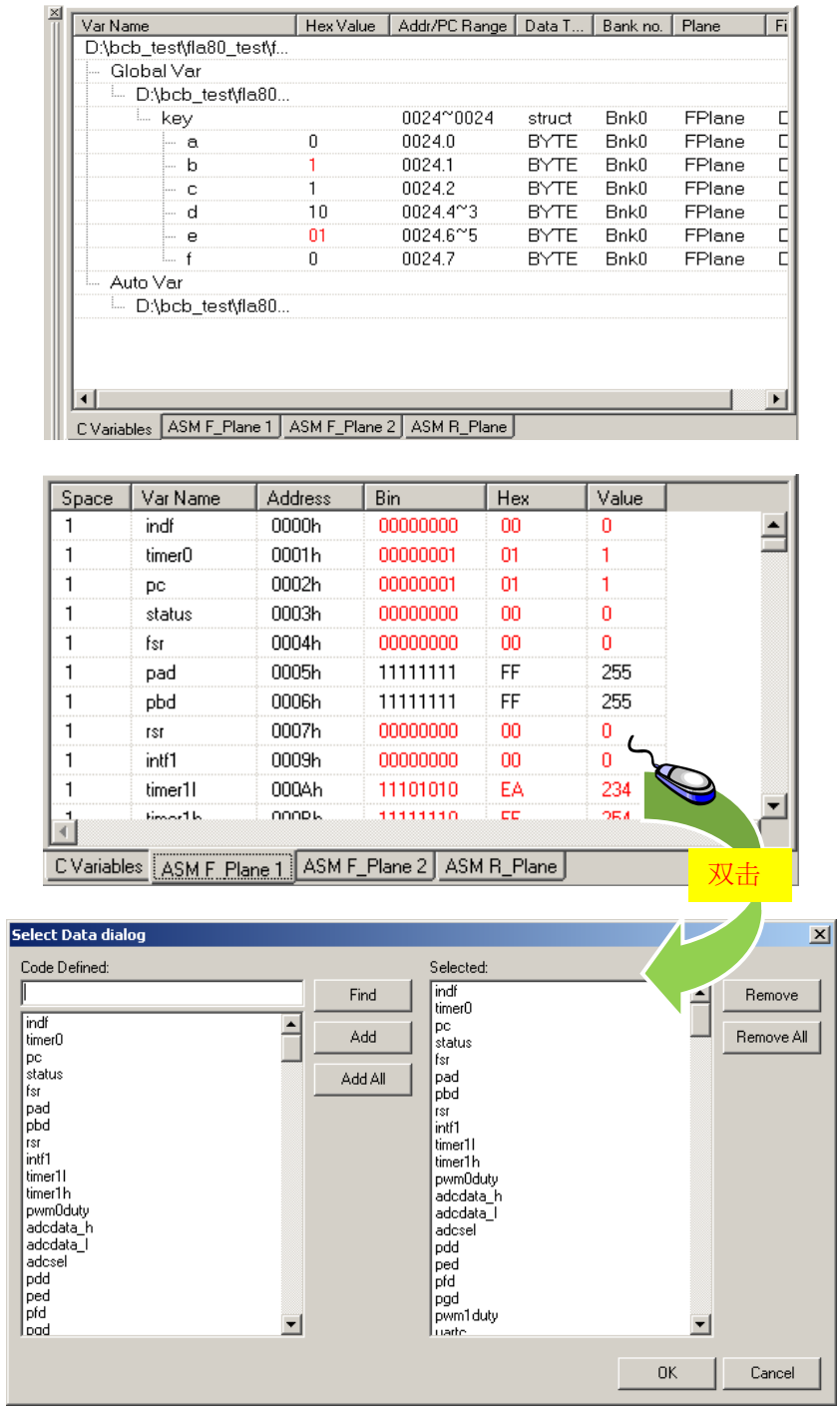
您可以在查找结果窗口中看到查找的结果。IDE 提供两种文字查找方式：查找在目前编辑器中开启的，或是查找在该项目中的所有文件。在查找结果窗口中所列示的结果中，使用者可以鼠标左键双击所感兴趣的程序行，系统将自动跳转到该文字所在的原始程序代码。



图表 12 查找

1.9 变量

变量窗口将列出项目所有程序里的所有变量，包含 C 和 ASM 变量。在 ASM 变量中，您可鼠标左键双击窗口并由所显示的选择窗口中，自由选择或调整您所想要观看的变量。若您想看所有的变量内容，可直接按取”Add All ”按键以选取全部的变量。



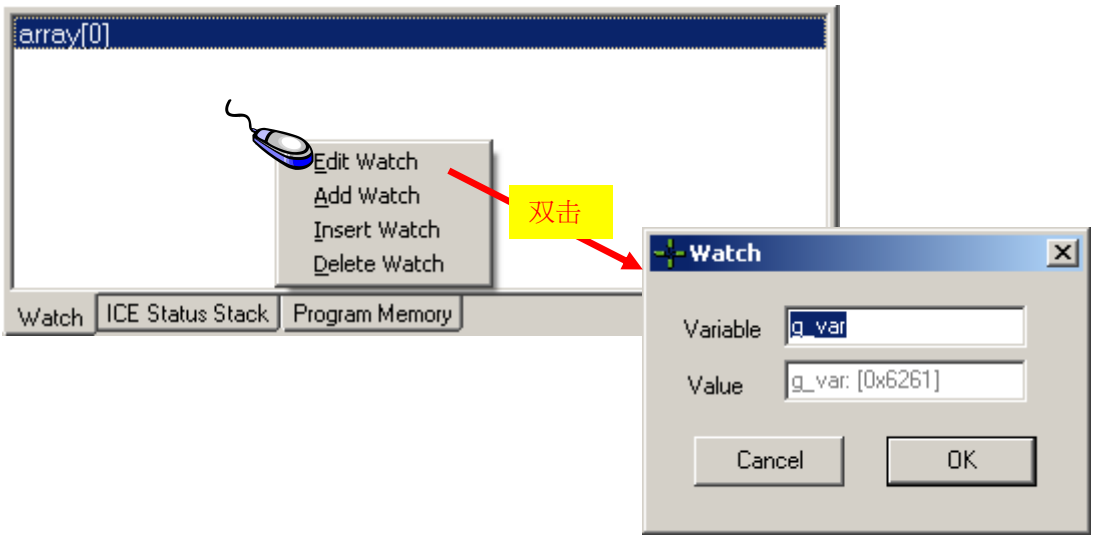
图表 13 参数

1.10 变量监看

在以下 3 个章节将说明 TICE99IDE 在进行调试时所提供的工具，以帮助您方便及有效地追踪变量值的运算变化。变量监看窗口显示变量的目前值是根基在执行点的范围上。所以若是执行点已移出该变量表达式的范围(out of scope)，则整个表达式将成为未定义的(undefined)。若执行点重回到监看变量表达式的算出值的区域时(即，若是执行点重回变量表达式的范围(scope))，变量监看窗口将会重新显示参数表达式的目前值。

以下为设定变量监看窗口的变量或表达式之方式

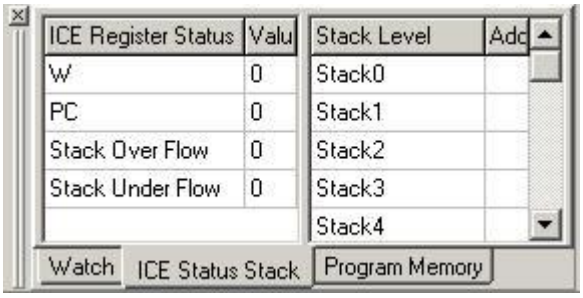
- 直接在程序代码编辑区域内，以鼠标拖放(drag and drop)的方式来新增监看之变量。
- 单击鼠标右键，在弹出的功能菜单中维护所要监看的变量：
  (1) 编辑监看(Edit Watch)： 开启监看特性对话框，以维护变量名称。
  (2) 新增监看(Add Watch)： 开启监看特性对话框，以新增一监看变量。
  (3) 插入监看(Insert Watch)： 开启监看特性对话框，在光标所在列之上方插入一监看变量。
  (4) 删除监看：删除一项监看



图表 14 变量监看

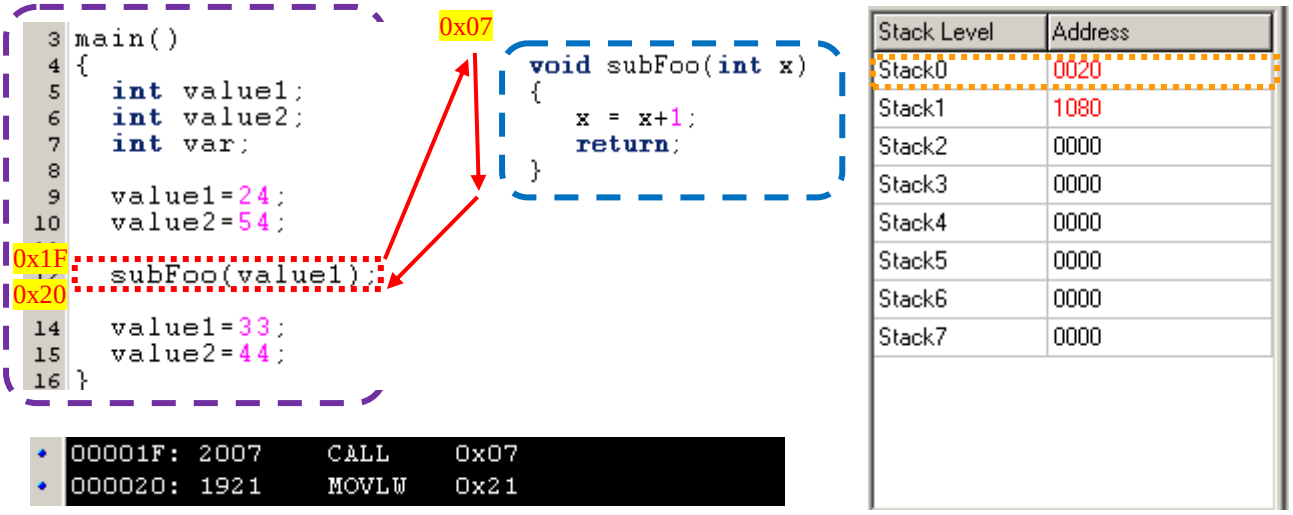
1.11 ICE 堆栈状态

堆栈版面显示关于您所调用的子程序信息。每当您调用一个子程序或者程序，调用子程序或者程序的下一个执行地址，执行被调用者之前.将会被加到堆栈中；且当该程序结束时，会从堆栈跳出来。此版面也提供 ICE 的状态显示，如 W(working register，工作寄存器)、PC (program counter，程序计数器)、堆栈上溢(stack over flow)和堆栈下溢(stack under flow)。



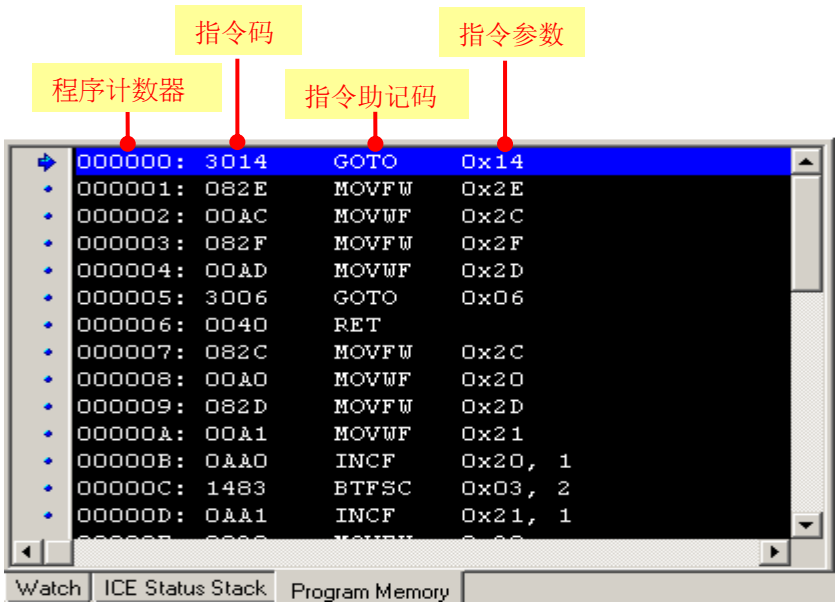
图表 15 ICE 堆栈状态

以下例子为 C code，在 main()主程序转移到子程序 subFoo 时，会将目前的程序计数器(program counter)，即 main 程序中 subFoo 的下一个执行地址：ox20 推入堆栈中(如 Stack0 的地址 0x0020)，则当 subFoo 执行完毕返回 main 主程序时，会从堆栈中弹出该地址(即 Stack0 的地址 0x0020)以便于程序继续往下执行。



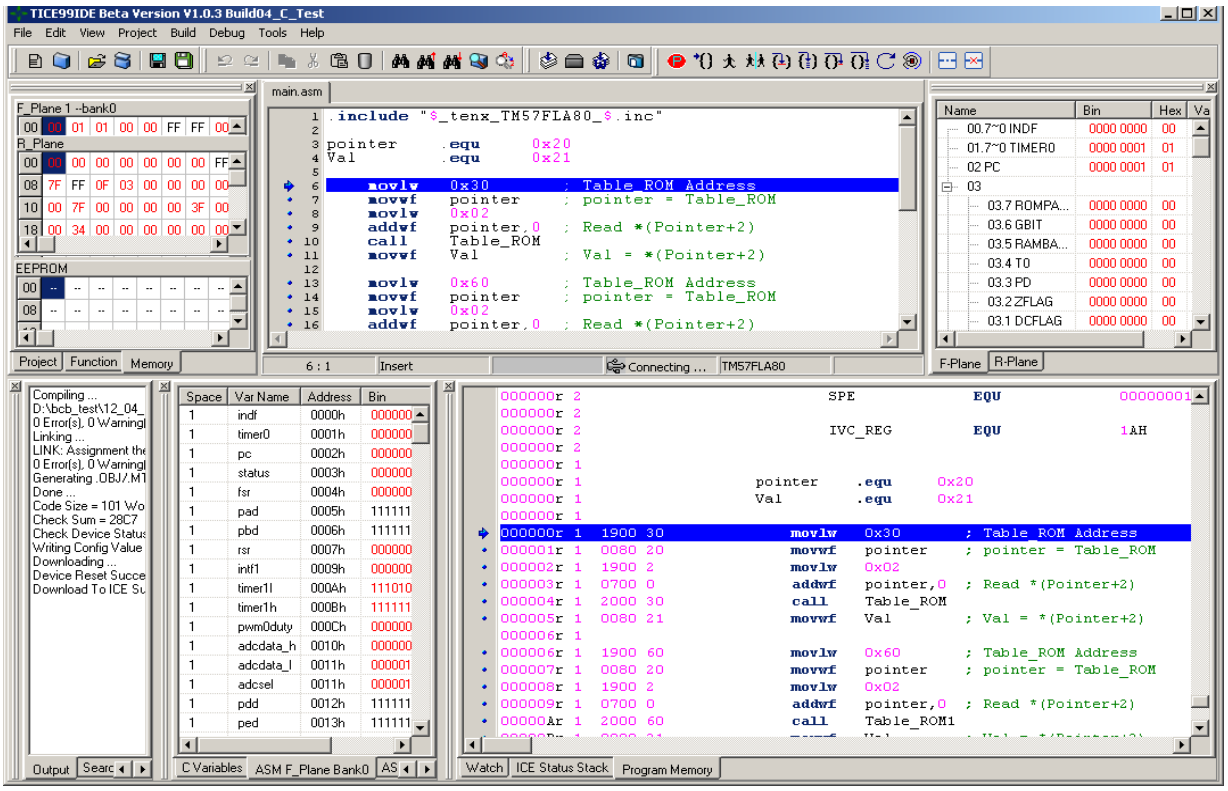
1.12 程序内存

程序内存窗口显示 ROM 程序代码的相关内容。在编译完原始程序后，此窗口将显示 ROM 的程序计数器、机器代码、助记码、和参数的操作码..等信息。



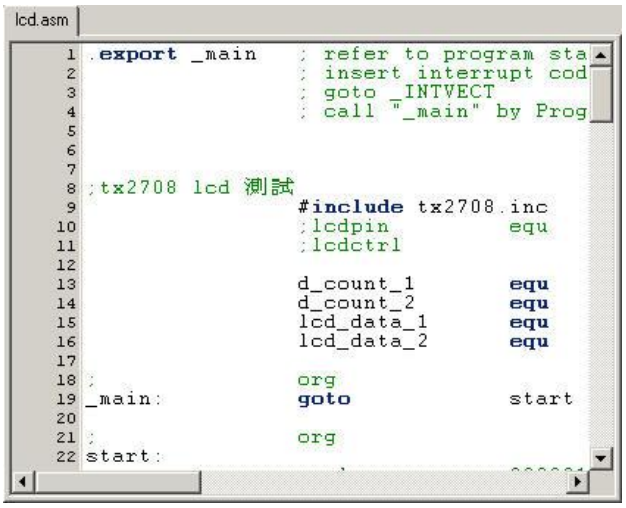
图表 16 程序内存

然而，若**汇编语言项目中只有一个. asm 档案**，程序内存将显示文件列表内容如下：



1.13 编辑器

编辑器提供使用者一个编辑原始码的区域，即将一个 C 程序文件、ASM 程序文件或表头文件，当作输入文件在编辑器中编辑及执行。



图表 17 编辑器

1.14 状态列

状态列显示编辑器、调试器与 ICE 型号等信息。编辑器的信息是显示光标的位置和编辑模式是否为插入或者复写。在进行程序调试时，状态列亦会显示目前的调试模式以及 ICE 联机状态。

图表 18 状态列

2. 主菜单

主菜单提供一系列可帮助您方便使用 IDE 的功能。这包含文件处理、编辑处理、版面显示、项目管理、项目建立、项目调试、IDE 的选项和说明。

2.1 文件

藉由选择文件下拉式菜单选项，您可以建立新文件或项目、关闭文件或项目、保存文件及关闭 IDE 应用程序。

2.1.1 新建文件、项目，开启文件、项目

让我们从文件开始：

「新建文件」让程序开发者建立一个新的空白文件以编辑程序或文件。新建立的文件并不会自动加入到项目里，而是各别独立的。使用者可依项目特性，手动加入至适宜的项目之中以便集中管理。当您点选「新建项目」按键，IDE 将显示一个新建项目窗口，您可以选择新项目的芯片类型并指定此项目名称、文件保存位置及选择是否要创建一新目录。

「开启文件」可以开启一个已存在的文件且将内容显示在新的编辑器。「开启项目」将开启已存在的项目且在项目管理员窗口中，以树状方式显示该项目所包含的各种文件。若在项目中有多个已开启的文件，则在关闭项目后，下次再开启此项目时，这些原先被开启的文件将重新被自动开启。

2.1.2 保存、另存文件，保存、另存项目，全部保存、关闭、关闭所有文件

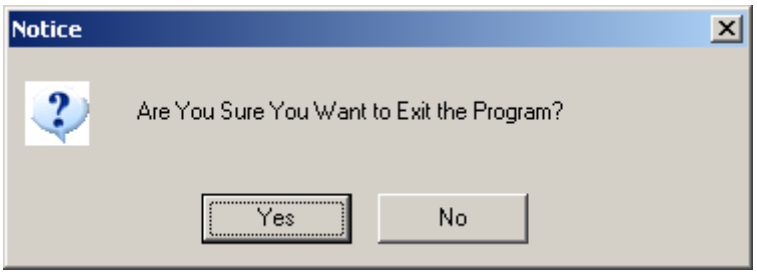
「保存」将保存目前编辑的文件。若保存的文件其档名尚未被指定，(如 Unit*)，则 IDE 将跳出「另存新档」的对话框，以便使用者可输入一个档名及文件类型，在按下「确认」键后以保存文件。「另存盘案」让使用者以其它档名来另存目前开启的文件。然而，若原文件是包含在项目里，则不会改变项目里面的档名，只会另存一份副本到硬盘里。「保存项目」将保存您目前开启的项目。若您欲改变成其它的项目名称，则可选择「另存项目」。「全部保存」将保存所有已开启的文件及项目所包含的所有文件。「关闭」可以关闭目前开启的文件。若您要关闭项目，请按「关闭所有文件」按键。这样可以同时关闭所有目前开启的文件及项目本身。

2.1.3 开启最近文件、项目

「开启最近文件」将列出至多四个最近曾被开启过的文件；「开启最近项目」存放至多十笔最近曾被开启过的项目。点选「开启最近文件」的下拉选项其一时，将开启所点选的文件或者切换到在编辑器中已开启的文件。您也可以从「开启最近项目」开启您最近曾开启过的项目。

2.1.4 退出

退出此 IDE 应用系统。



图表 19 退出提示对话框

2.2 编辑

在编辑菜单中，提供很多功能让使用者更方便与快速的编辑程序代码：如复制、剪切、粘贴、删除和全选。若您想取消之前的编辑动作，例如输入错误时，您可以点选「返回」以回复至原先之编辑状态。若您对「返回」改变主意，您可以点选「重做」以取消之前的回复动作。编辑菜单也提供一系列的文字查找功能，包含「查找」、「找上一个」、「找下一个」、「在项目中查找」和「查找并替代」。

2.2.1 返回，重做

当发生输入的内容有误时，您可以使用「返回」取消输入以重回到您输入前的编辑状态。当您改变主意，您可以点选「重做」再重做一次刚才被返回的输入动作。

2.2.2 复制，剪切，粘贴，删除，全选

复制、剪切、粘贴、删除及全选可让使用者更快速的编辑程序代码。

2.2.3 查找，找上一个、找下一个，在项目中查找，查找并替代，跳转

当您要在一个文件里面找一个词或文字时，您可使用「查找」。查找的过程中，若在查找的文件里有多个目标文字时，您可以使用「找上一个」或「找下一个」直接到达上一个或下一个的目标文字。若您想要在项目或目前开启的所有文件里面，查找词或文字，请用「在项目中查找」。「查找并替代」可让您找到您想替代的词或文字，并以设定的词或文字替代之。「跳转」会将鼠标指针直接移到您指定的行数。

2.3 视图

TICE99 里面有很多版面窗口，包含项目管理员、函数视图器、内存窗口、缓存器视图、输出信息框、查找结果、C 变量、在 F-Plane1、 F-Plane2、 R-Plane 的汇编码变量、监看 C 变量、ICE 堆栈状态、程序内存和工具列。这些会帮助程序设计者依需求取得更详细的信息。

2.3.1 项目管理员、函数视图器、内存窗口

项目管理员、函数视图器和内存窗口是在同一个版面。要切换至您想看的信息，可直接点选项目管理员、函数视图器或内存的标签页面。例如，若您点选项目管理员，将会切换到项目的页面，若该版面为隐藏的话，将会自动显示出来。当开启一个项目，项目管理员以树形图来显示与管理该项目所包含的程序文件、链接库、LCD、文字文件等..。当以编译器编译 C 程序后，函数显示器将一一条列出项目中 C 程序文件中所定义的函数。内存窗口显示：包含 F-Plane、R-Plane 和 EEPROM 各个内存地址之实时内容值。

2.3.2 缓存器视图

缓存器视图包含两个页面，一为 F-Plane，另一个为 R-Plane。您可以切换标签页面来看里面的内容。若缓存器视图消失，请点选主菜单「视图」|「缓存器视图」 来显示其版面。当程序编译成功且已下载到 ICE 时，其版面内容会显示出来。

2.3.3 变量监看、ICE 堆栈状态、程序内存

「变量监看」提供使用者监看要调试之 C 变量。「ICE 堆栈状态」显示 ICE 信息，包含工作寄存器(working register，W)、程序计数器（program counter，PC）、堆栈上溢、堆栈下溢和堆栈的内容。「程序内存」显示汇编机器码的窗体。

2.3.4 C 变量 、 ASM F-Plane1、ASM F-Plane2、ASM R-Plane

使用者可利用 C 变量(包含局部变量及全局变量)、ASM F-Plane1、ASM F-Plane2 和 ASM R-Plane 窗口来一一视图项目中原始程序文件中所定义的所有变量。当编译程序代码成功后及开始进行调试时，这窗口内的各个变量的内容值将随着程序的执行结果自动更新，并以颜色区别是否在二个执行动作间被更动。若无法显示该窗口，请点选主菜单的「视图」|「C 语言变量」或「视图」|「汇编语言变量」来显示。

2.3.5 输出、查找、搜索结果

「输出」页面将显示编译、汇编及连结程序的执行结果。若有任何错误或警告将显示在输出版面上，并且您可鼠标双击该信息行，IDE 会跳到编辑器对应的行。当您设定查找在项目或开启文件中的关键词时(即选择「编辑」|「在项目中查找」)，搜索结果列示在「搜索结果」版面中，鼠标双击某一数据行，IDE 也会跳到编辑器中该文件对应的行。这也将显示在 C 项目中变量占用内存相关信息。

2.3.6 LCD 编辑器

LCD 编辑器只能应用在有支持 LCD 功能的芯片上。IDE 会启动此菜单项目，若芯片不支持 LCD，则此功能将会被取消。

2.3.7 工具列

点选「视图」|「工具列」，可选择性控制四个主要工具列是否要显示或隐藏。它们包含「文件」、「编辑」、「编译」、「调试」。意即，当该项目被启动，工具列是可显示的，或者您也可以取消显示工具列。

2.4 项目

「项目」提供一集中管理与维护相关文件的机制。针对项目所提供的功能选项，包含新增项目的文件、增加已存在的文件到项目、从项目移除文件和设定项目。

2.4.1 新增文件到项目

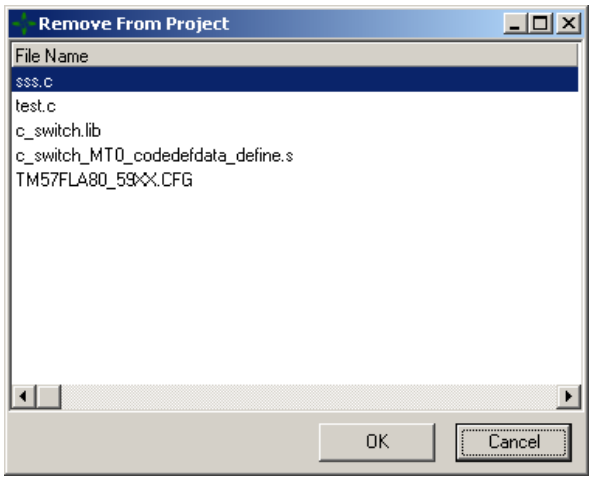
当程序设计者点选功能列上之「项目」|「加入」|「新文件」 菜单项目，IDE 会建立一个空的文件且会用 Unit*来命名，其中*表示 1~n 流水号。当您编辑完文件并存盘时，IDE 将显示另存新文件的对话框让您用其它档名来储存这个文件，并增加该文件到项目管理员的树状结构之中。

2.4.2 增加已存在的文件到项目

另外一个在项目中加入文件的方式是增加现存的文件。您可以点选功能列上之「项目」|「加入」|「已存在文件」 菜单项目，或是在「项目管理」的页面范围内点击鼠标右键，在弹出菜单中选择「加入现存文件」 菜单项目，来选择欲加入项目的现存文件，如此 IDE 将显示一开启对话框让您选择您要加入到项目的文件。则项目管理员会把您选的文件增加到项目之树状文件结构里面。

2.4.3 移除

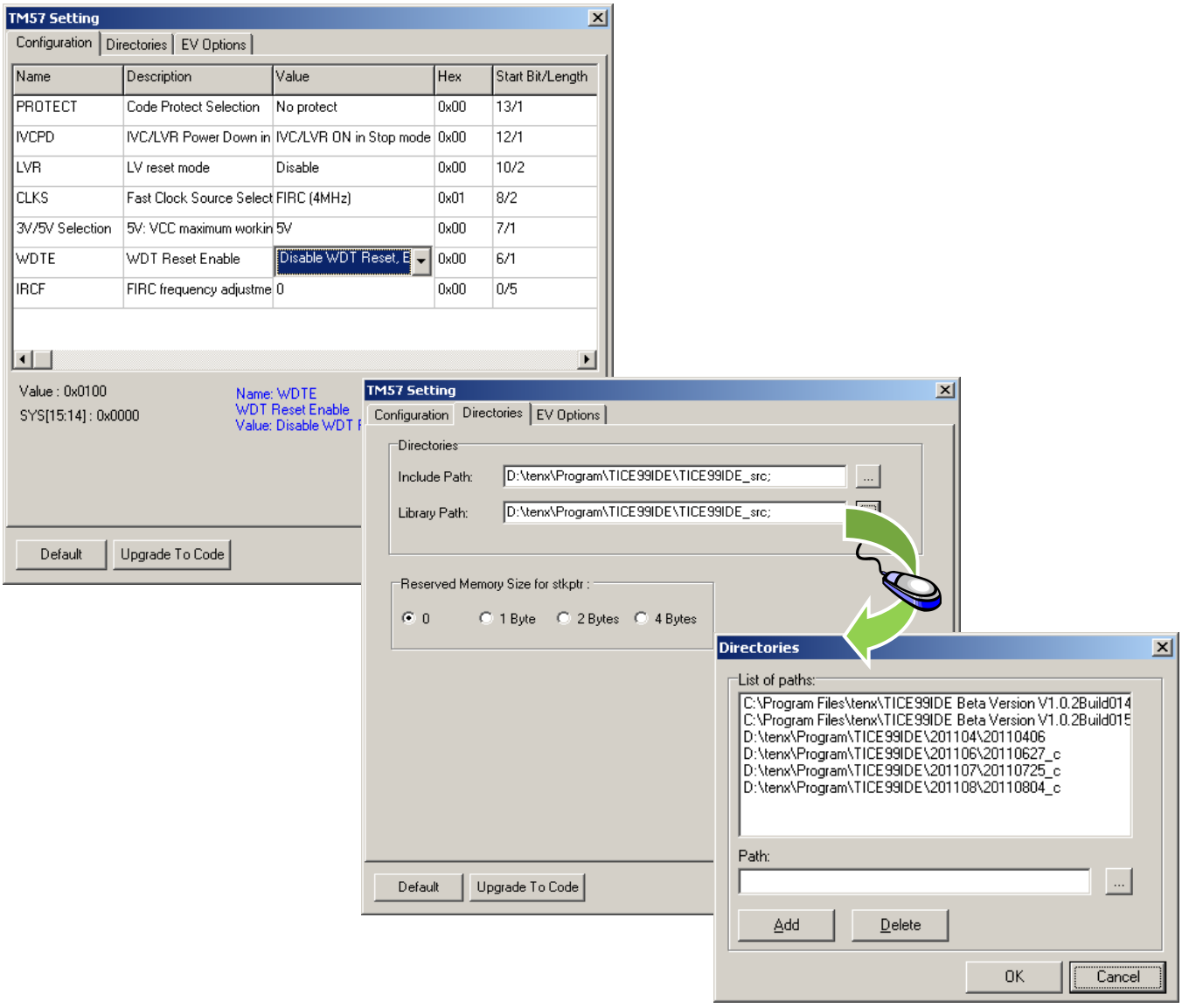
自项目中移除文件的方式有二个：(1)点选功能列上之「项目」|「移除」菜单项目，将显示一项目文件明细窗口 (图表 20)，让使用者决择要移除的文件，或(2)是先在「项目管理」的树状范围内，点选欲移除的文件并点击鼠标右键，在弹出菜单中选择「移除」 菜单项目。移除只是将文件从项目树状结构中移除掉，但该文件并不会真正自储存目录中删除此文件。



图表 20 移除项目中文件之对话框

2.4.4 项目设定

设定项目属性的方式有二：(1)点选功能列上之「项目」|「项目设定」菜单项目，或(2)在「项目管理」的树状范围内，点击鼠标右键并在弹出菜单中选择「项目设定」 。项目设定包含 IC 配置设定、目录和 EV 选择(图表 21)。所设定的配置设定值将在程序代码下载到 ICE 前，预先写入 ICE 之中。其可设定的配置设定项目随着 ICE 类型而不同。若您有修改配置设定值，设定页面也会显示修改过的设定值。另一「项目设定」的页面是「目录」，用以指定在那些目录路径可以找到编译及汇编时会用到的头文件或是链接库文件。EV 选择包含两个主要功能，其一是计算芯片的最佳的 IRCF，其二为切换 ICE 的电压。



图表 21 项目设定

2.5 建立

建立可以检查您写的程序代码是否正确，并且编译成执行档(.bin 文件)或物件文件(.o 檔)。亦可下载执行档至 ICE 中，以进行在线调试功能。

2.5.1 建立、重新建立

「建立」只有编译项目里面的文件。此功能不会自动将编译成功的执行档下载到 ICE 中。您可藉由此菜单项目来检查您编写的程序代码是否正确。若有任何错误，输出版面将显示错误或警示信息。鼠标左键双击错误信息，IDE 会把您带到错误的地方。不同于「建立」，「重新建立」在编译程序前会先清除*.o 文件、.bin 文件..等，再进行编译、汇编及连结项目内的程序文件。

2.5.2 建立且下载到 ICE

不同于「建立」，「建立且下载到 ICE」会在编译成功后，将执行档自动下载到 ICE。

2.5.3 编译文件

「编译文件」只会编译您目前编辑的单一文件。故而，若只需检查一个文件，不需使用「建立」，另一选择为使用此功能。

2.5.4 编译设定

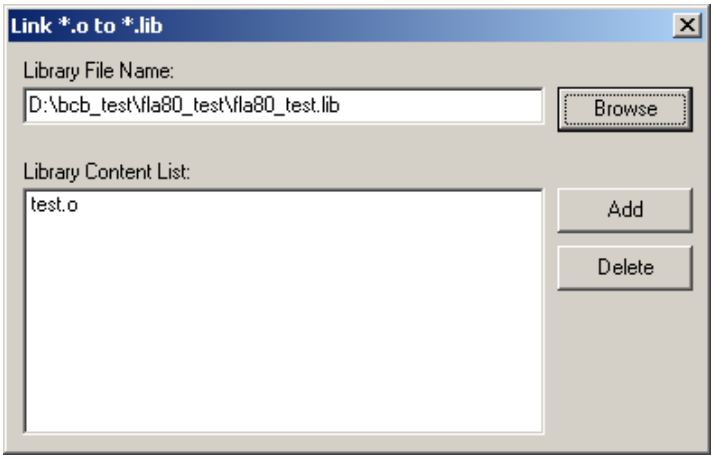
编译设定有二个选项：「调试模式」与「发布模式」。二者的不同为，选择「调试模式」将产生编译过程中的调试信息以利于执行调试功能。若选择「发布模式」，*.o 文件不会包含调试信息。

2.5.5 建立链接库

C 语言或汇编函数皆可藉由 TICE99 IDE 所提供之工具以建立函数库文档，其方法语概念如下图：

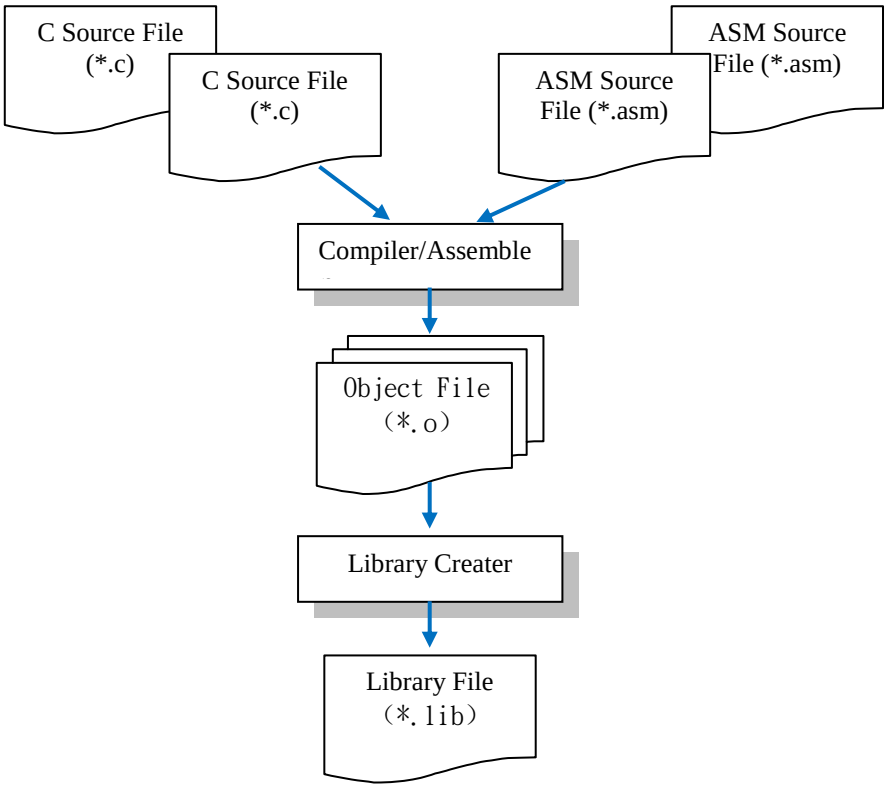
方法分为二阶段：

1. 产生目标档，将包含数个函数模组的单一 C 语言或汇编程序档，经过编译器、汇编器处理，以产生目标档（*.o）。
2. 建立函数库文档，利用函数库产生器（library maker），选择性决定要将哪些目标档，集中建立为单一之函数库文档（*.lib）。



图表 22 建立链接库

概念：



2.6 调试

2.6.1 暂停

暂停所有调试步骤。当 ICE 是在「运行」、「自由运行」、「单步运行」或「自动单步」模式，按取「暂停」会将以上调试模式的步骤暂时停止。请注意在其它模式中「暂停」是无法发挥任何作用的。

2.6.2 运行到光标位置

「运行到光标位置」将程序运行到目前光标所指的位置，并且等待下一个调试的指令。

2.6.3 运行，自由运行

「运行」和「自由运行」皆会运行下载到 ICE 的程序代码。两者之间的差异是运行过程中，「自由运行」不会停在设有断点的地方，而「运行」会停在设有断点的地方，若有设断点的话。

2.6.4 单步运行，自动单步

「单步运行」在进行调试时，将一行一行地运行程序代码；或者当用文件列(list file)调试时会一次运行一个机器码。当针对汇编程序(.asm)文件进行调试时，一行原始程序代码将对应到一个机器码；然而针对 C 程序文件调试时，一行原始程序代码则可能会对应到一至多个机器码。「自动运行」是 IDE 自动依计时设定的时间，周期性地运行「单步运行」。「自动单步」的计时设定是在 「工具」|「选项」|「调试」 选项。定时器的单位为毫秒（ms）。

2.6.5 单步不进入函数，自动单步不进入函数

「单步不进入函数」是一种特殊的调试步骤。此运行步骤不会跳入函数里面逐步运行。这意味着，它将一次运行完整个函数。在没有碰到函数转移 (CALL) 的情况下，「单步不进入函数」将运行一个程序代码，如同单步运行一般。

2.6.6 重置 ICE

这将清零 ICE 程序计数器(即 PC=0)且将状态设定为等待调试指令。

2.6.7 初始化 ICE 板

「初始化 ICE 板」将下载 FPGA 所需要的文件，确认 ICE 联机和电源状态，测试 RAM 和 ROM 和重新设定 ICE。

2.6.8 加入|删除断点，移除所有断点

这些项目是关于断点的设定，「加入断点」将会让调试运行步骤停在断点。在欲中断调试运行的程序行，以鼠标左键双击即可设定断点；若要取消断点，同样以鼠标左键双击已设定断点的程序行，即可切换为非中断的状态。「移除所有断点」将移除所有已设定的断点。

2.6.9 以 List 档方式调试，以原始档方式调试

进行调试的方式有两种：一种是使用 List 文件，另一种是使用原始程序文件来进行调试。使用 List 文件调试会根据机器码来运行单步调试。而使用原始檔则会根据汇编(.asm)或 C 的原始程序代码来逐行运行程序。

2.7 工具

在「工具」，您可以动态切换语言。我们目前提供三种语言，英文、简体中文和繁体中文。您可以自行决定您要使用哪种语言。「工具」的另外一个功能是「选项」。在「选项」，您可以设定编辑器的字型。

2.7.1 英文

将语言切换到英文。

2.7.2 简体中文

将语言切换到简体中文。

2.7.3 繁体中文

将语言切换到繁体中文。

2.7.4QTP 产生器

输出项目的编码申请表(Coding request form)，其包含 hex 文件档名(*.hex)、检查和(checksum)、系统配置及其它相关信息。

2.7.516 进制文件转换器

16 进制文件转换器将原始 IC 的 16 进制文件转译至目标 IC 的 16 进制文件，有以下选择：

原始芯片	目标芯片
TM57PE11	TM57PE11A
TM57PE11A	TM57PE11B
TM57PA10	TM57PA10A
TM57PA20	TM57PA20A

2.7.6 选项

「选项」提供以下三种类型设定

- 编辑器：
  1. 编辑文字的字型、字体大小、TAB 键的停顿位置。
  2. 当新开启 IDE 时是否自动开启之前项目
  3. 当勾选代码完成，在程序代码编辑过程中，编辑将显示一个可调整大小的提示窗口，其列出了有效的元素，供使用者选择并添加到您的代码中。
  4. 改变编辑器的关键词、数字、注释文字…等等的颜色。

- 调试：设定「自动运行」之计时设定，定时器的单位为毫秒（ms）。
- 菜单：各个功能下拉式菜单之快捷组合键。

2.8 说明

「说明」收集了有关 IDE 的信息，包含 IDE 使用方法、IDE 中所执行的编译程序、汇编程序及连结程序之版号和 IDE 的更新功能。

2.8.1IDE 使用手册

这将开启一个相对语言的 IDE 使用手册.pdf 文件。若您是在英文下执行 IDE，将开启英文版的使用手册，以此类推。

2.8.2C 编译器使用手册

这将开启一个相对语言的 C 编译器使用手册.pdf 文件。

2.8.3 汇编器使用手册

这将开启一个相对语言的汇编器使用手册.pdf 文件。

2.8.4 关于

「关于」窗口显示版权信息、IDE、C 编译器、汇编器和连结器的版本。

2.8.5 更新|新版本检查

使用者可以透过「更新|新版本检查」检查是否有最新版本。若没有任何新版本，将显示最新的版本窗口，否则会将 IDE 更新到最新的版本。

3. 工具列

工具列收集一般常会用到的菜单。工具列可以分为四个部分：文件、编辑、建立和调试。

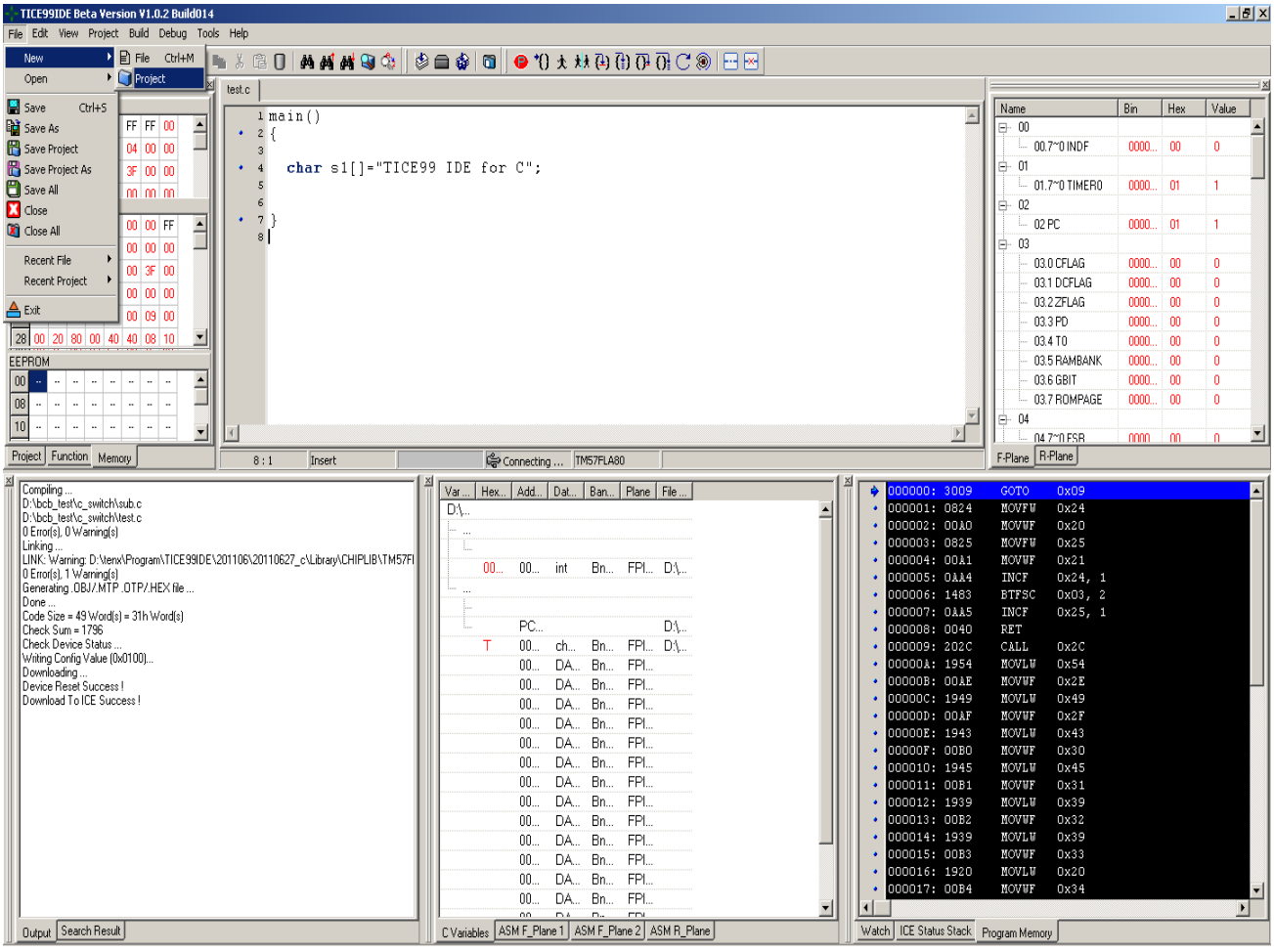
- 文件工具列，有六个按键，分别为新增文件、新增项目、开启文件、开启项目、保存和全部保存。其功能和「文件」菜单是相同的。
- 编辑工具列，大部分在「编辑」菜单的功能都列在这里，包含返回、重做、复制、剪切、粘贴、删除、查找、找上一个、找下一个、在项目中搜索。编辑工具列的最后按钮为搜索并替代。
- 然后建立、重新建立、建立且下载至 ICE 和编译文件，这些都是在建立工具列。
- 调试工具列，包含运行、自由运行、单步运行、自动单步、暂停、单步不进入函数、自动单步不进入函数、运行到光标位置、重置 ICE、初始化 ICE 板、加入/删除断点和移除所有断点。

4. 建立一个项目

TICE99IDE 提供两种项目服务：C/汇编项目和汇编项目。汇编项目只能包含汇编程序，但 C/汇编项目可同时包含 C 及汇编程序或单纯只包含 C 程序。设计者可依应用特性选择项目类型，然后按照以下的步骤，建立一个项目、编写、编译程序代码、下载到 ICE 和做调试测试。

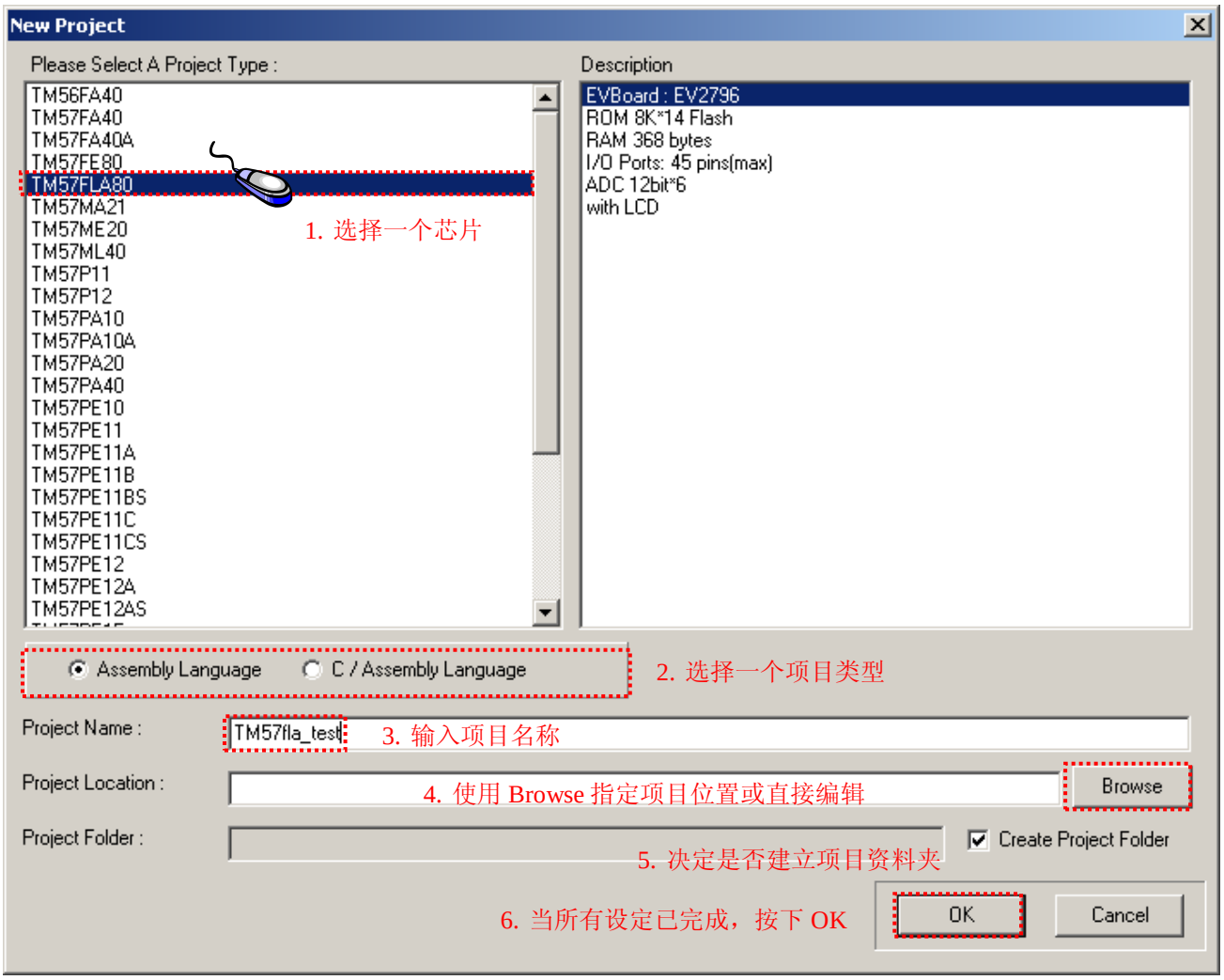
4.1 产生一个新的项目

当您要建立一个新的项目，请点选功能列之 「文件」|「新增」|「项目」，



图表 23 产生一个新的项目-1

在项目窗口中，依实际应用在芯片列表中，选择目标芯片类别。接下来，选择您的项目类别为汇编语言或 C 语言，然后帮项目命名并决定保存项目之目录位置。

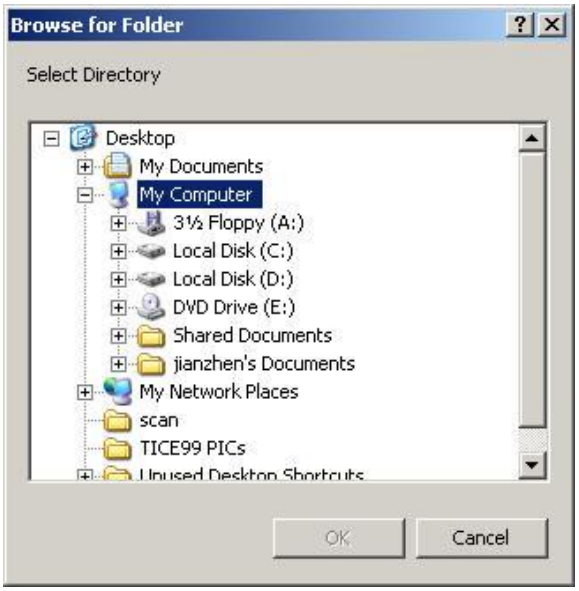


图表 24 产生一个新的项目-2

当选择目标芯片类型，若我们点击该芯片，左边对话框将显示芯片相关的参考信息。以下表格为 TICE99 IDE 所支持的 IC 列表：

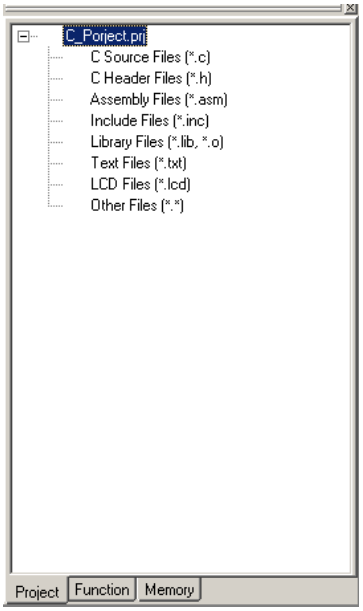
EVBoard	Chip	ROM	RAM (bytes)	Pin	ADC	LCD	Future support
EV1697 / EV1694	TMU3130	8K*14 (Flash)	160*8 + 128*8	48	--		
	TMU3132	4K*14 (MTP)	160*8 + 128*4	20	--		
	TMU3131	6K*14 (MTP)	160*8 + 128*4	28(max)	--		
	TM57FE80	8K*14 (Flash)	160*8 + 128*8	48	--		
EV2780	TM57PE11B	1K*14 (OTP)	48	6 (max)	--		
	TM57PE11BS	1K*14 (OTP)	48	6 (max)	--		
	TM57PE11C	1K*14 (OTP)	48	6 (max)	--		
	TM57PE11CS	1K*14 (OTP)	48	6 (max)	--		
EV2781	TM57MA21	2K*14 (MTP)	184	18(max)	12bit*11		
EV2785	TM57ML40	4K*14 (MTP)	368	34 (max)	--	V (8*28)	V
EV2786	TM57PE10	1K*14 (OTP)	48	16 (max)	--		
	TM57PE12	1K*14 (OTP)	48	12 (max)	--		
	TM57PE12AS	1K*14 (OTP)	48	12 (max)	--		
	TM57P12	1K*14 (OTP)	48	12 (max)	--		
	TM57RE12	1K*14 (MASK)	48	12 (max)	--		
	TP6717	1K*14 (OTP)	48	16 (max)	--		
EV2786B	TM57PE15	1K*14 (OTP)	48	12 (max)	--		
	TM57PE15A	1K*14 (OTP)	48	12 (max)	--		
	TM57PE15AS	1K*14 (OTP)	48	12 (max)	--		
	TM57RE12A	1K*14 (MASK)	48	12 (max)	--		
EV2787	TM57PE40	1K*14 (OTP)	48	12 (max)	--		V
EV2788	TM57ME20	2K*14 (MTP)	96	18 (max)	--		
EV2793	TM57P11	1K*14 (OTP)	48	6 (max)	--		
	TM57PA10	1K*14 (OTP)	64	14 (max)	12bit*6		
	TM57PA10A	1K*14 (OTP)	64	14 (max)	12bit*6		
	TM57PE11	1K*14 (OTP)	48	6 (max)	--		
	TM57PE11A	1K*14 (OTP)	48	6 (max)	--		
	TM57PE12	1K*14 (OTP)	48	12 (max)	--		
EV2795	TM57FA40	4K*14 (Flash)	184	18 (max)	12bit*8		
	TM57FA40A	4K*14 (Flash)	184	18 (max)	12bit*8		
	TM57PA20	2K*14 (OTP)	184	18 (max)	12bit*8		
	TM57PA40	4K*14 (OTP)	184	18(max)	12bit*8		
EV2796	TM57FLA80	8K*14 (Flash)	368	45(max)	12bit*6	V	

利用浏览按键指定产生项目的位置。



图表 25 产生一个新的项目-3

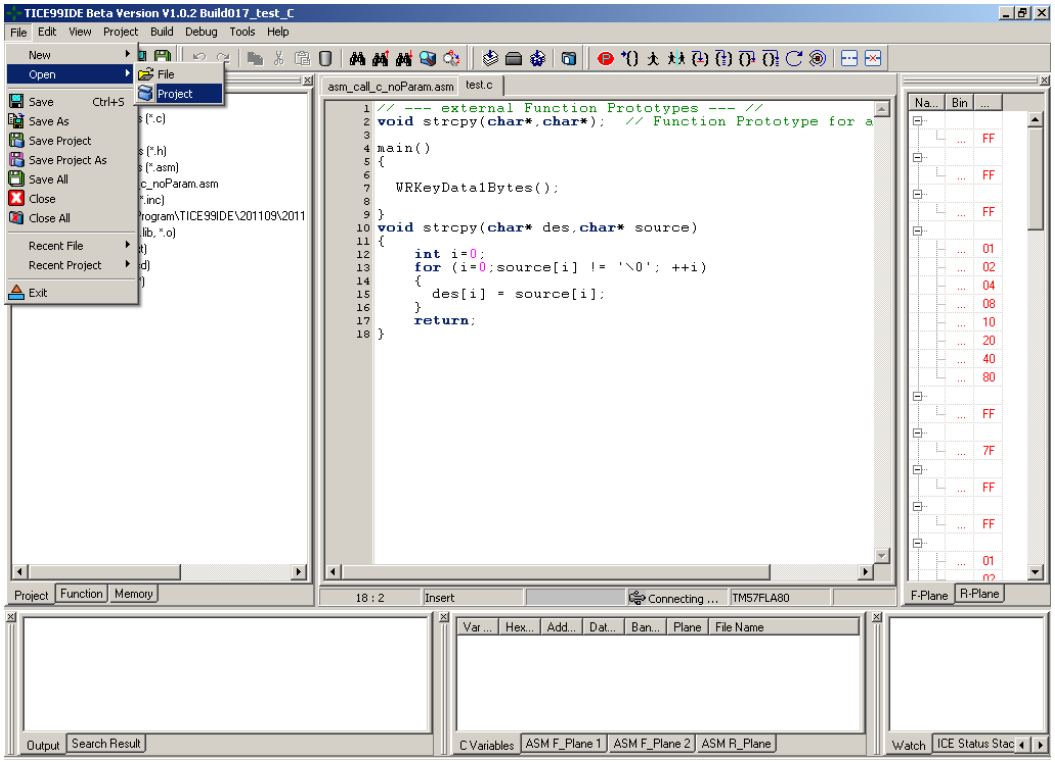
当您输入完项目所需要的信息，按下确认按键。新项目就会产生。



图表 26 产生一个新的项目-4

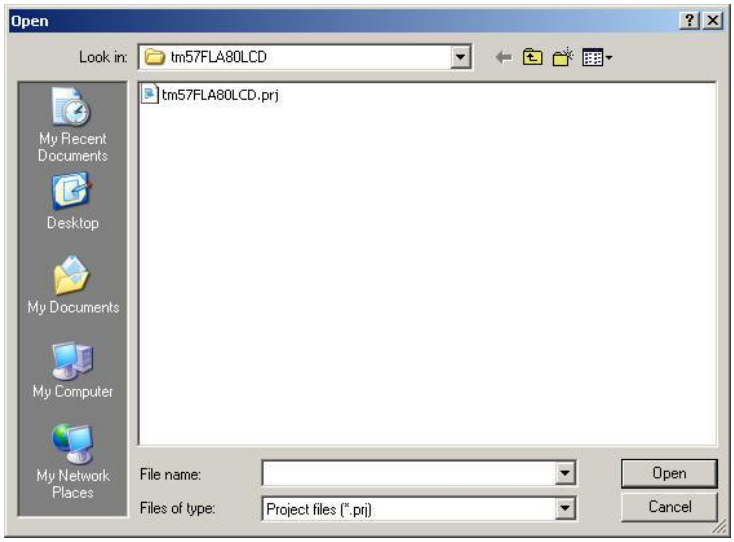
4.2 开启已存在的项目

若您已有现存的项目且要对它做修改，您可以使用「开启已存在的项目」来执行。

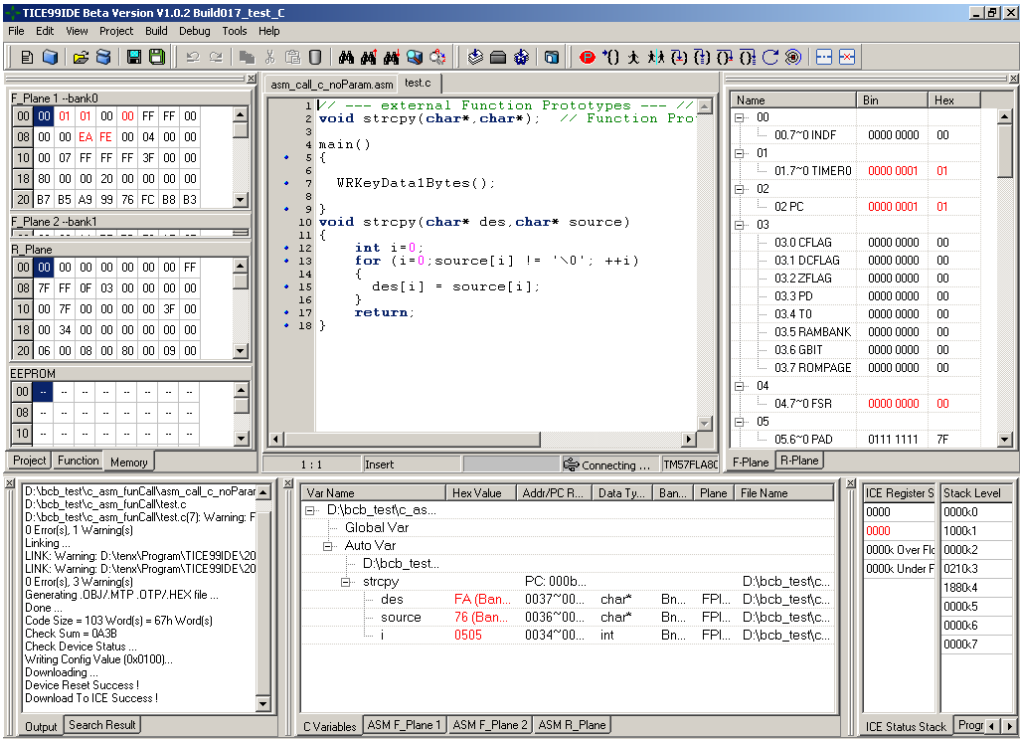


图表 27 开启已存在的项目-1

点选功能列 「文件」|「开启」|「项目」，会出现开启文件的对话框，在对话框中，浏览文件目录选择欲开启的项目。项目的扩展名为.prj。选择项目且点选开启按键即可开启已存在的项目。



图表 28 开启已存在的项目-2

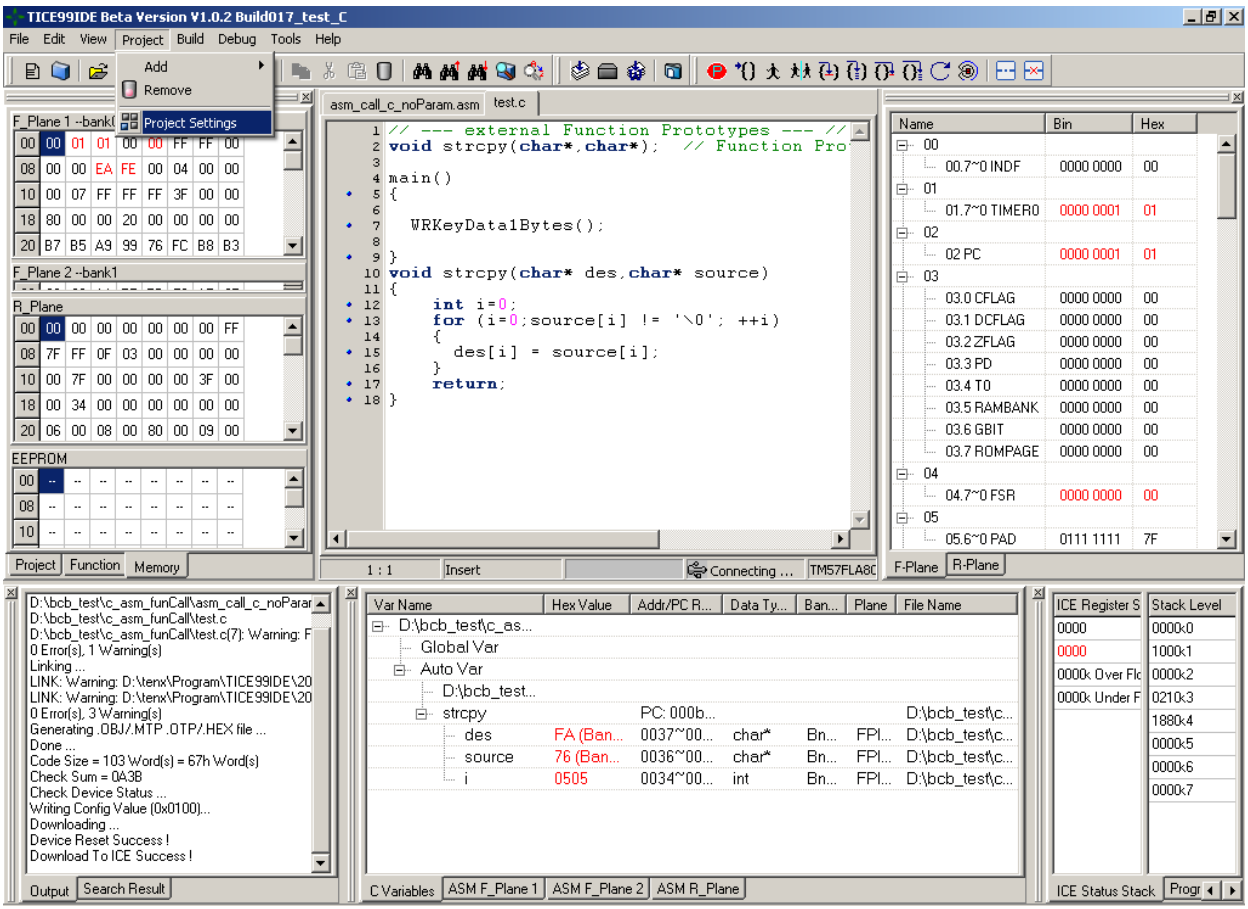


图表 29 开启已存在的项目-3

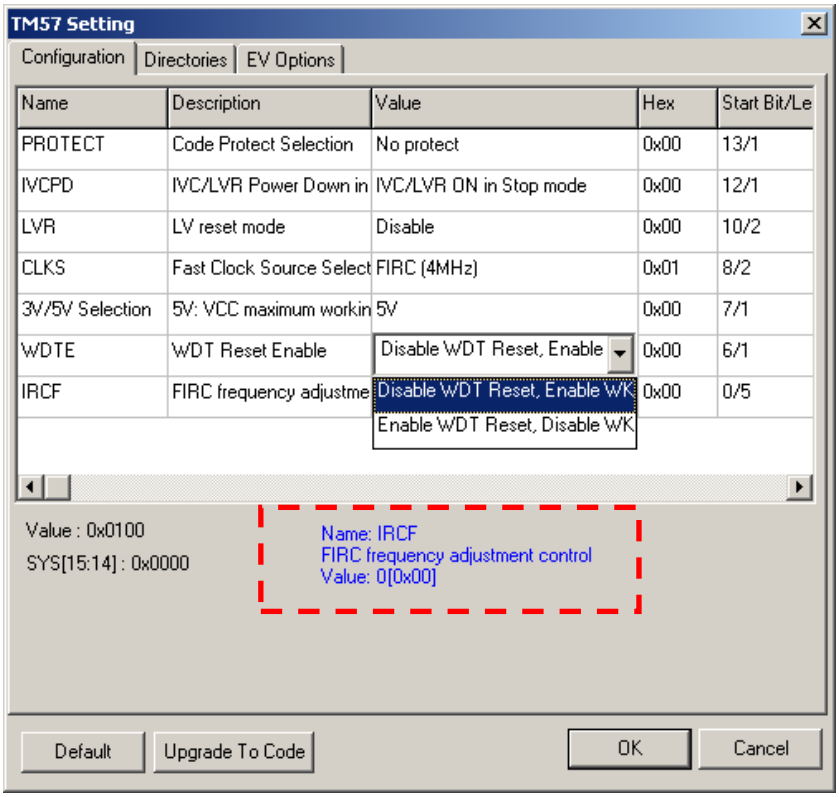
4.3 项目设定

当您要修改 ICE 的配置参数，点选功能列 「项目」|「项目设定」 并点选 配置页面 来修改参数值。所展出的 ICE 参数项目依 ICE 类型而不同，例如：FIRCF、WDTE、3V/5V 选项、CLKS、LVR、ICVPD 和 PROTECT。

当使用者选取一个配置项目，对应的信息将显示于版面的下方。这包含名称，项目说明，选择说明和 16 进制值。使用者可以点击该值的下拉式方块选择一个需要的。同时，该值将被改变响应配置设定的值。



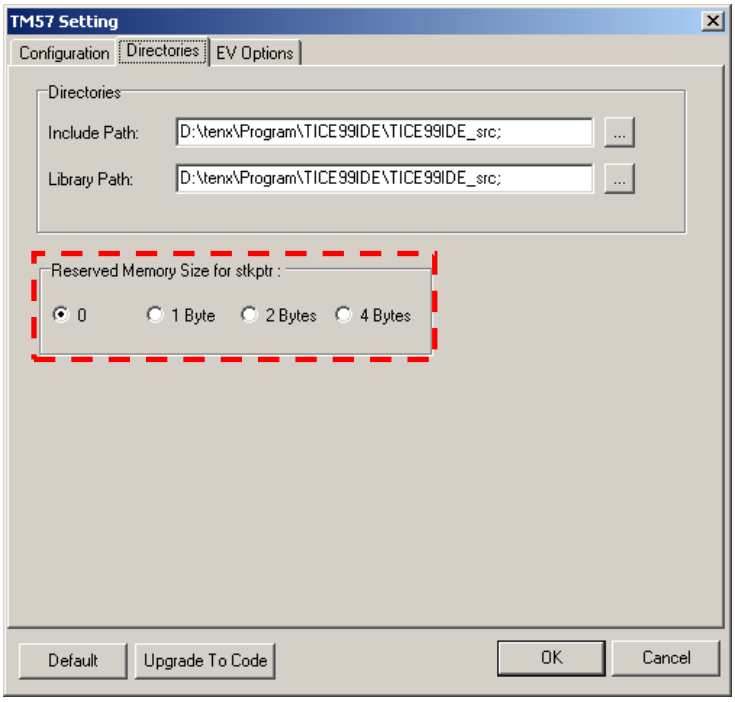
图表 30 项目设定-1



图表 31 项目设定-2

在 C 项目，"Directories"的选单，会显示一个选项："Reserved Memory Size for stkptr"。它将预留一个额外的 RAM 内存空间抓取并且储存 STKPTR 的内容，再把之前储存的 STKPTR 内容恢复。预留内存的空间大小有分四种选项：0，1 字节，2 字节，4 字节，预设为 0（无预留）。

STKPTR 是 C 编译器预留的寄存器变量，且暂时保留变量和运行程序当中的结果。当选项至少为 1 字节，C 编译器将配置 RAM 内存储存 STKPTR 且增加更多的指令储存和恢复在函数中 STKPTR 的内容。



图表 32 项目设定**-3**

我们提供几个需要设定"Reserved Memory Size for stkptr"的情况避免错误发生。

1. 一个函数调用其他函数且被调用的函数将修改 STKPTR 的资料。它将覆盖之前调用者存的 STKPTR 资料。当被调用者回传，调用者将存取错误的 STKPTR 值。
2. 赋值语句的左边是数组（或指针）且右边是函数调用（例如：array[i] = func()）。调用函数之前，数组（或指针）的资料将被储存于 STKPTR。
3. 函数表示（如：retVal = fun1() + fun2()）。调用第二个函数之前，第一个函数的结果将被储存于 STKPTR。
4. 复杂度越高的程序，越多内部互动受到影响。执行中需要储存 STKPTR。

我们强烈推荐以下程序编码方式避免以上的情况。更明确的说，降低程序复杂度避免错误发生。

原始码	最好的方式
array[i] = func()	retVal = func(); array[i] = retVal;
retVal = fun1() + fun2()	retVal = fun1(); retVal = retVal + fun2();

为了说明如何运作，将利用两种情况做说明：第一个没有设定预留内存，另一个是设定预留 1 字节内存。我们将于*.s 档案指出其差异。

```
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
```

无设定预留内存备份STKPTR	预留1字节内存备份STKPTR

main()

```
; ------------------------------------------------------------------
; int main ()
; ------------------------------------------------------------------

.segment        "CODE";;2

.proc   _main

.segment        "CODE";;;3

;
; {
;
        .dbg    line, "D:\bcb_test\20130107_fla80\Pointer_Set_Array
        CALL    initialize_defdata
;
; i = 0;
;
        .dbg    line, "D:\bcb_test\20130107_fla80\Pointer_Set_Array
                MAC_STORE_IMMED _i+0,$0,2 ;;val=0,Flags=1001
;
; R_Var[i] = f1();
;
        .dbg    line, "D:\bcb_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,D
        MOVLW   _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=a1,stk=1,_id=0,ipop=1,stkptr,returnFlags
                MAC_STORE_MEMtoPTR op2,0,stkptr+0,0,2,0,0,0 ;func->
;
; }
;
        .dbg    line, "D:\bcb_test\20130107_fla80\Pointer_Set_Array
        RET
        .dbg    line

.endproc
```

main()

```
; ------------------------------------------------------------------
; int main ()
; ------------------------------------------------------------------

.segment        "CODE";;2

.proc   _main

.segment        "CODE";;;3

;
; {
;
        .dbg    line, "D:\bcb_test\20130107_fla80\Pointer_Set_Array_RE
        CALL    initialize_defdata
;
; i = 0;
;
        .dbg    line, "D:\bcb_test\20130107_fla80\Pointer_Set_Array_RE
                MAC_STORE_IMMED _i+0,$0,2 ;;val=0,Flags=1001
;
; R_Var[i] = f1();
;
        .dbg    line, "D:\bcb_test\20130107_fla80\Pointer_Set_Array_RE
                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
        MOVLW   _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=a1,stk=1,_id=0,ipop=1,stkptr,returnFlags=1
                MAC_STORE_MEMtoPTR op2,0,stkptr+0,0,2,0,0,0 ;func->ptr
;
; }
;
        .dbg    line, "D:\bcb_test\20130107_fla80\Pointer_Set_Array_RE
        RET
        .dbg    line

.endproc
```

储存暂时的变量和结果于 STKPTR

f1()

```
.proc   _f1

.segment        "CODE";;;3

;
; i = 1;
;
        .dbg    line, "D:\bcb_test\20130107_f1a80\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_f1a80\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=100
        MOVLW   _R_Var ;1001,1,0,idel=1,stkptr
        ADDWF   op1,0 ;340C,addaddr_static
                MAC_STORE -1,stkptr+0,1 ;push,id=1345,flags=100
                MAC_STORE_FR _i+0,op1,2,0 ;V-98,1,133,
                MAC_ADD_IMMED op1,$3,2 ;val=3,<<<ADD_IMMED
                MAC_STORE_MEMtoPTR op1,0,stkptr+0,0,2,0,0,0
;
; return R_Var[i];
;
        .dbg    line, "D:\bcb_test\20130107_f1a80\Pointer_Set_Array_RPLA
                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=100.
        MOVLW   _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
;
; }
;
        .dbg    line, "D:\bcb_test\20130107_f1a80\Pointer_Set_Array_RPLANE
L0003:  RET
        .dbg    line

.endproc
```

f1()

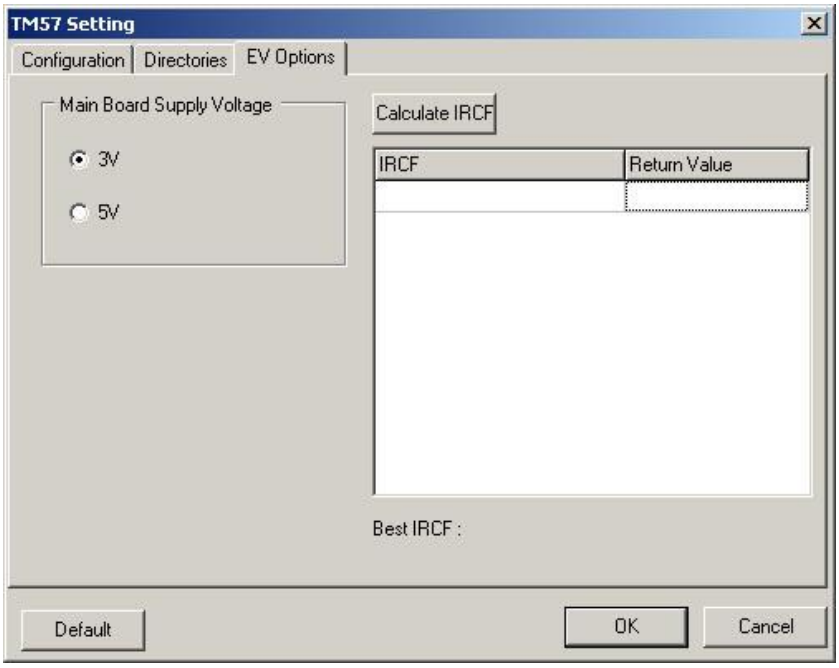
```
.proc   _f1

.segment        "CODE";;;3

;
; {
;
        .dbg    line, "D:\bcb_test\20130107_f1a80\Pointer_Set_Array_RPLANE_00000
                MAC_STORE stkptr,f1_PARAM-1,1 ;0
;
; i = 1;
;
        .dbg    line, "D:\bcb_test\20130107_f1a80\Pointer_Set_Array_RPLANE_00000
                MAC_STORE_IMMED _i+0,$1,2 ;;val=1,Flags=1001
;
;  3;
                "D:\bcb_test\20130107_f1a80\Pointer_Set_Array_RPLANE_00000
                _STORE_FR _i+0,op1,1,0 ;adjust of scale type
                AC_SHL_IMMED op1,1,1 ;3,pop=1,_istkptr=0,flags=1,DT=1001,id=2,i
                :1001,1,0,idel=1,stkptr
                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,icallsub=1241,2
;
; return R_Var[i];
;
        .dbg    line, "D:\bcb_test\20130107_f1a80\Pointer_Set_Array_RPLANE_00000
                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
        MOVLW   _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
;
; }
;
        .dbg    line, "D:\bcb_test\20130107_f1a80\Pointer_Set_Array_RPLANE_00000
L0003:  MAC_STORE f1_PARAM-1,stkptr,1
        RET
        .dbg    line

.endproc
```

备份 STKPTR 的内

覆盖 STKPTR 的资料

恢复之前储存的 STKPTR 资料

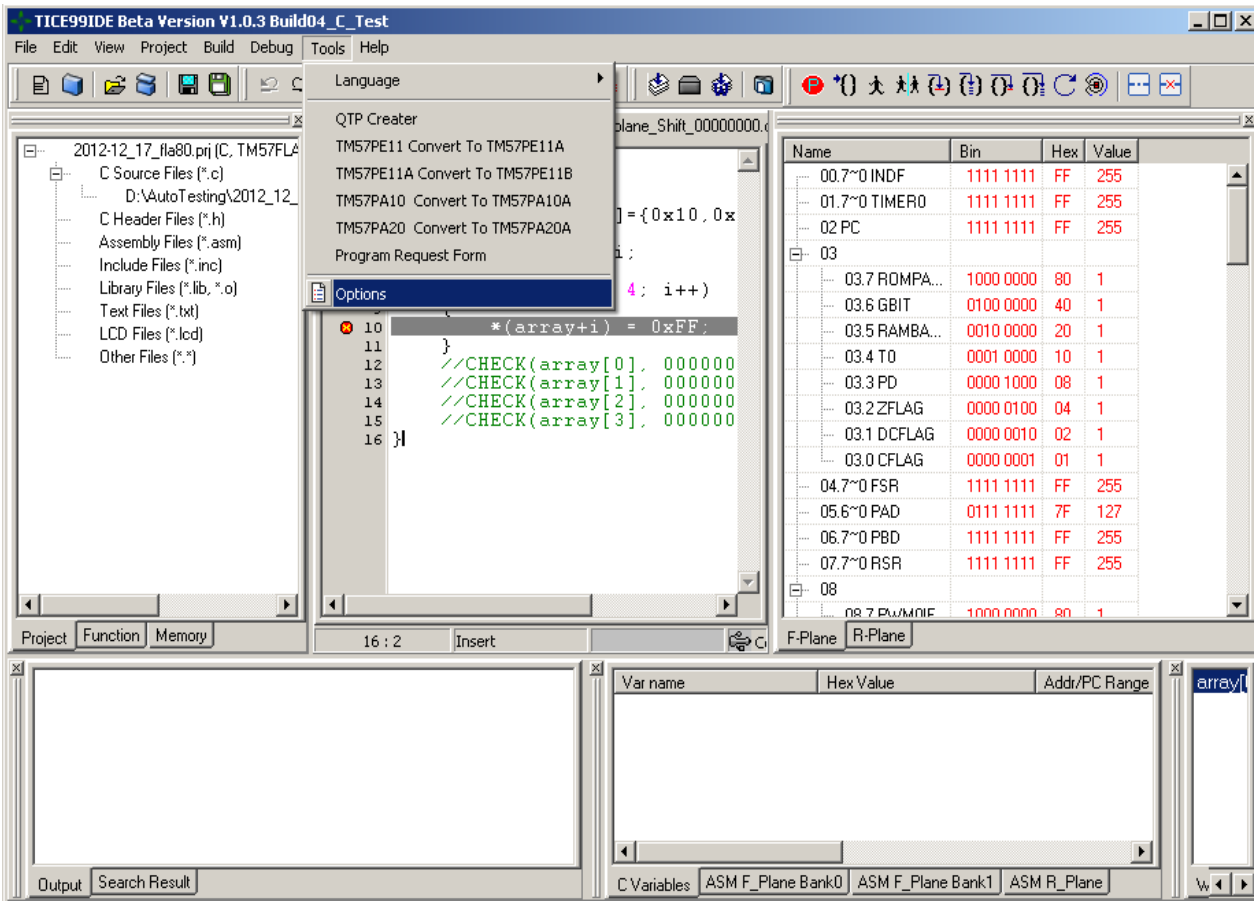
备注：虽然设定"Reserved Memory Size for stkptr"可以避免 STKPTR 被覆盖，可是它将消耗更多的 RAM 内存来备份 STKPTR 资料且更多的 ROM 内存来储存和恢复函数中的 STKPTR 内容。与其设定"Reserved Memory Size for stkptr"，我们强烈推荐降低程序复杂度。


TM57 Setting
Configuration
Directories
EV Options
Main Board Supply Voltage
3V
5V
Calculate IRCF
IRCF
Return Value
Best IRCF :
Default
OK
Cancel

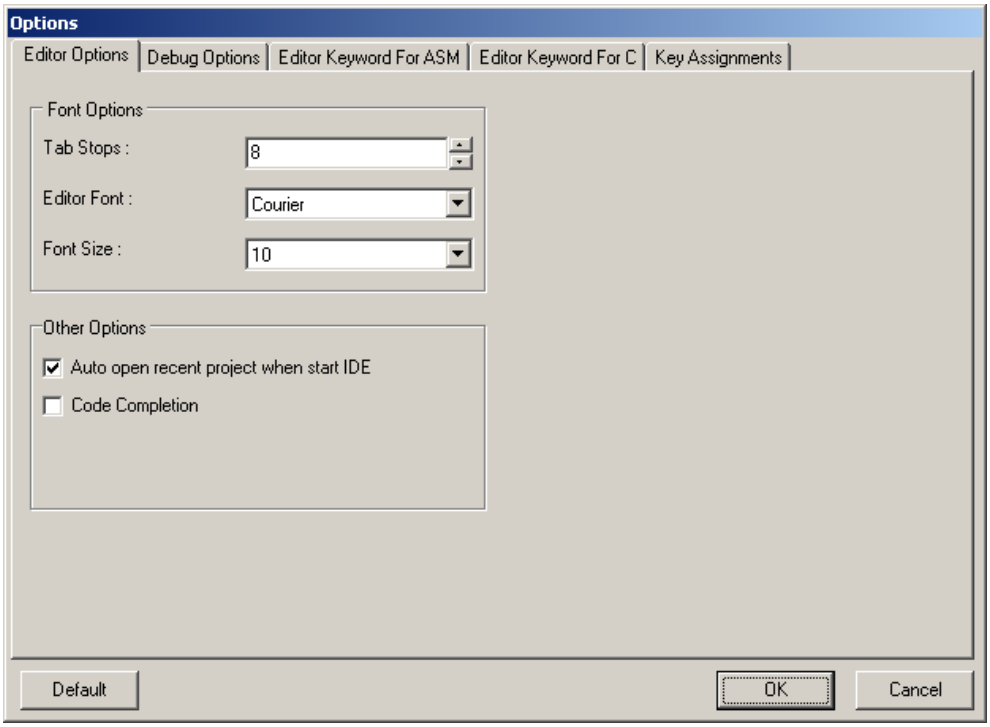
图表 32 项目设定-4

4.4 编辑器选项

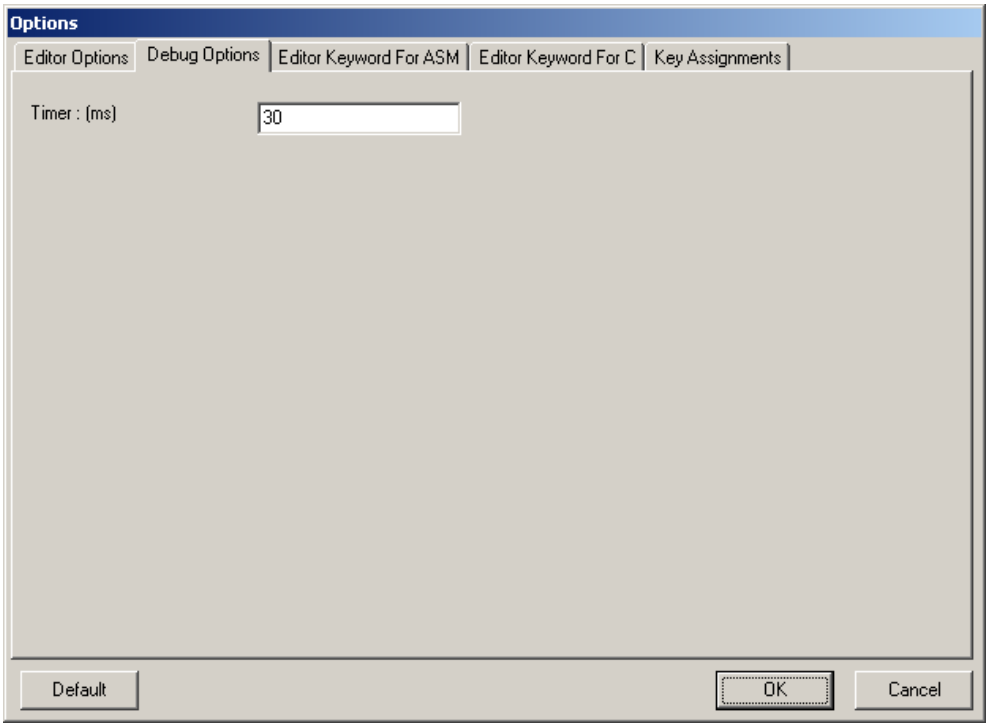
藉由编辑器设定，程序设计者可依个人的喜好或习惯改变编辑器的显示与外观。使用者可以点选「工具」|「选项」 以开启选项窗口。在编辑器中，您可以改变字型、字体大小和 Tab 停靠位置。使用者也可以改变 C 和 ASM 关键词的颜色来配合自己的习惯。



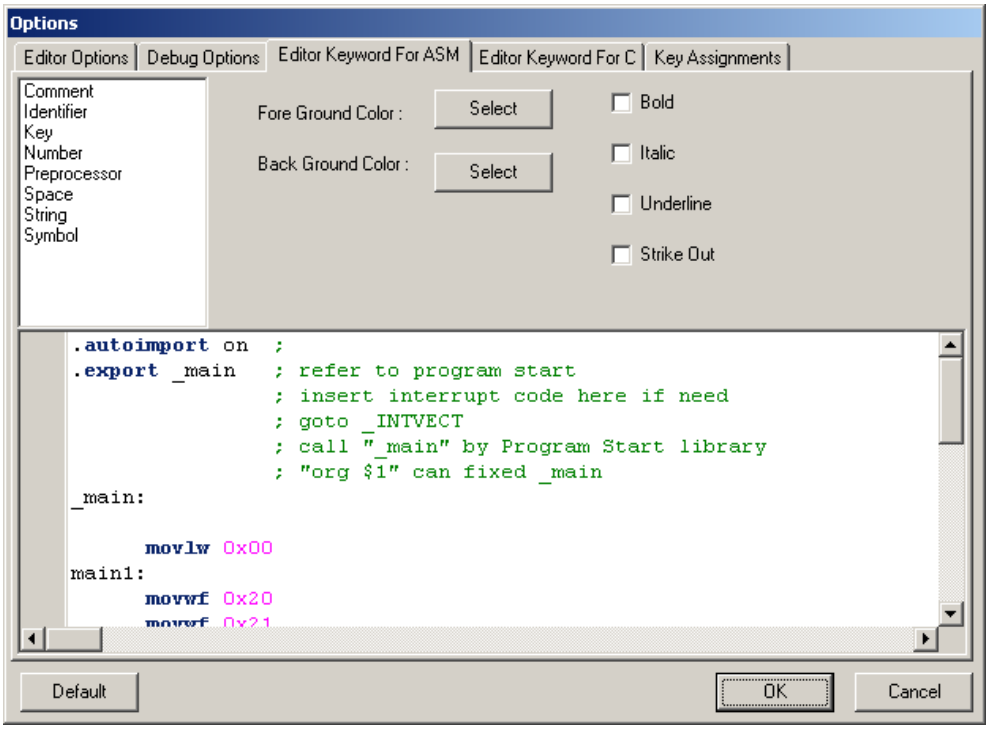
图表 33 编辑器选项-1



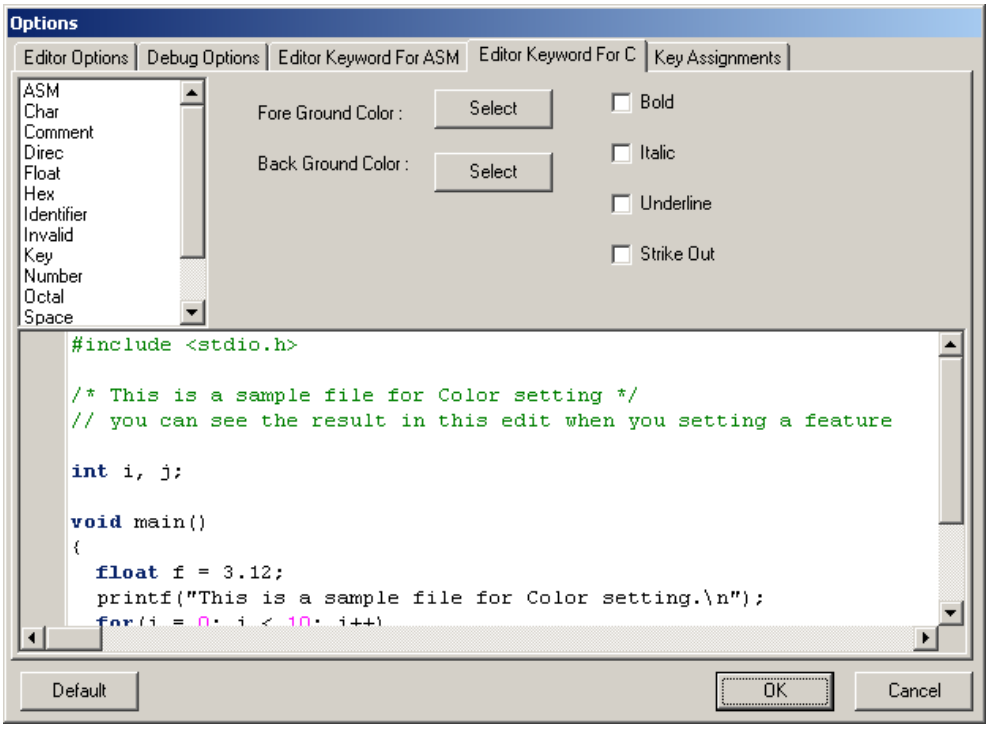
图表 34 编辑器选项-2



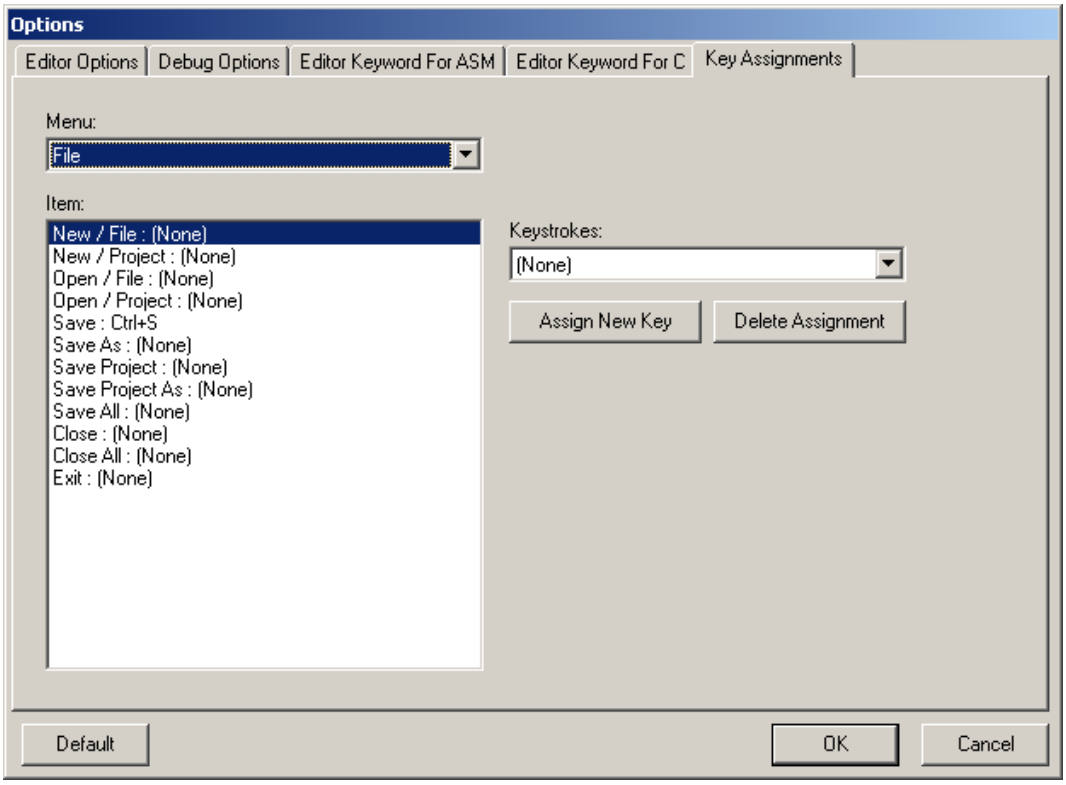
图表 35 编辑器选项-3



图表 36 编辑器选项-4



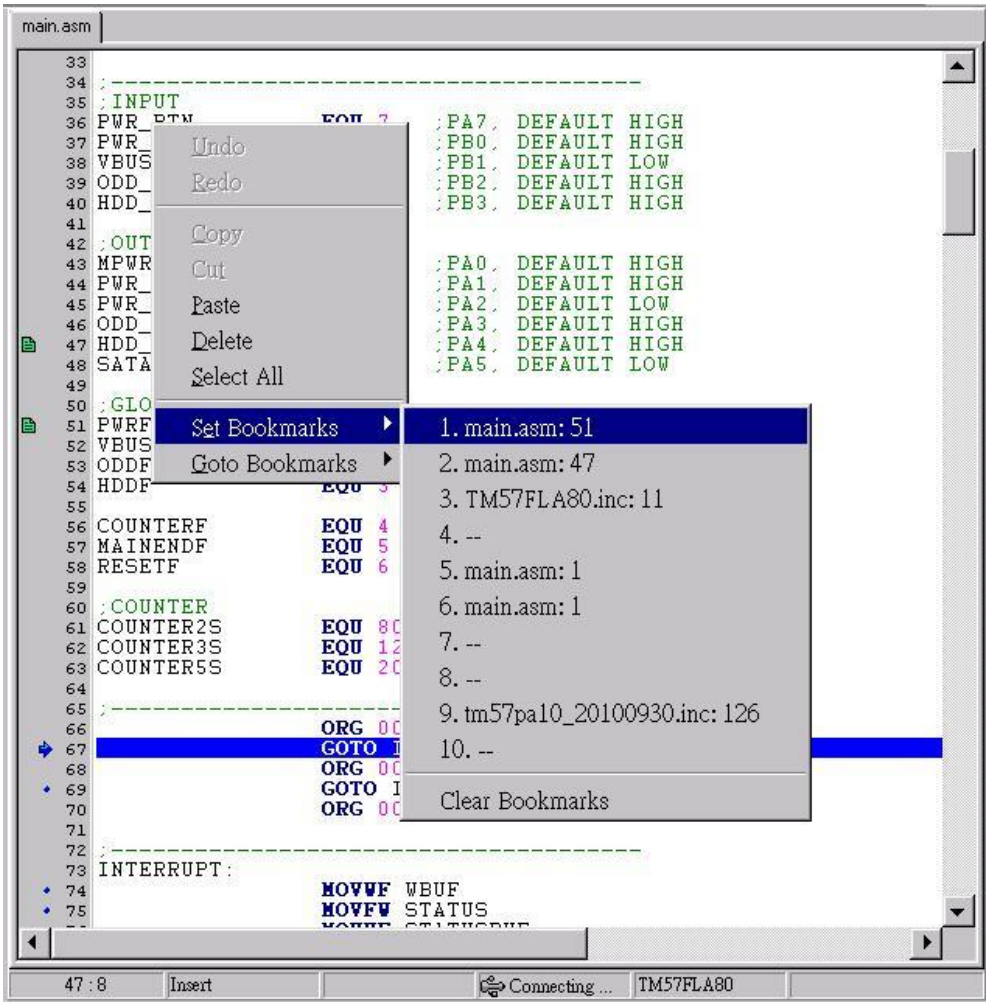
图表 37 编辑器选项-5


Options
Editor Options
Debug Options
Editor Keyword For ASM
Editor Keyword For C
Key Assignments
Menu:
File
Item:
New / File : (None)
New / Project : (None)
Open / File : (None)
Open / Project : (None)
Save : Ctrl+S
Save As : (None)
Save Project : (None)
Save Project As : (None)
Save All : (None)
Close : (None)
Close All : (None)
Exit : (None)
Keystrokes:
(None)
Assign New Key
Delete Assignment
Default
OK
Cancel

图表 39 编辑器选项-6

4.5 书签

在众多的数据集中寻找某一段记录，除了从一个记录移动到另一个特定的记录之循序寻找外。书签提供一个快捷的方式，藉由在数据集中某一特定位置设定一标记，以便需要时可以快速返回标记之位置，此即为书签功能。使用者可在编辑器区域内，点击鼠标右键在弹出的下拉菜单中使用书签功能。所设定的每一笔书签中记录了文件名称及列号，编辑器区域的左侧也会显示一图标以标记书签，至多可标记 10 个书签。点选 ”转到书签”，并在下拉菜单中选择已设定的书签标记号码。



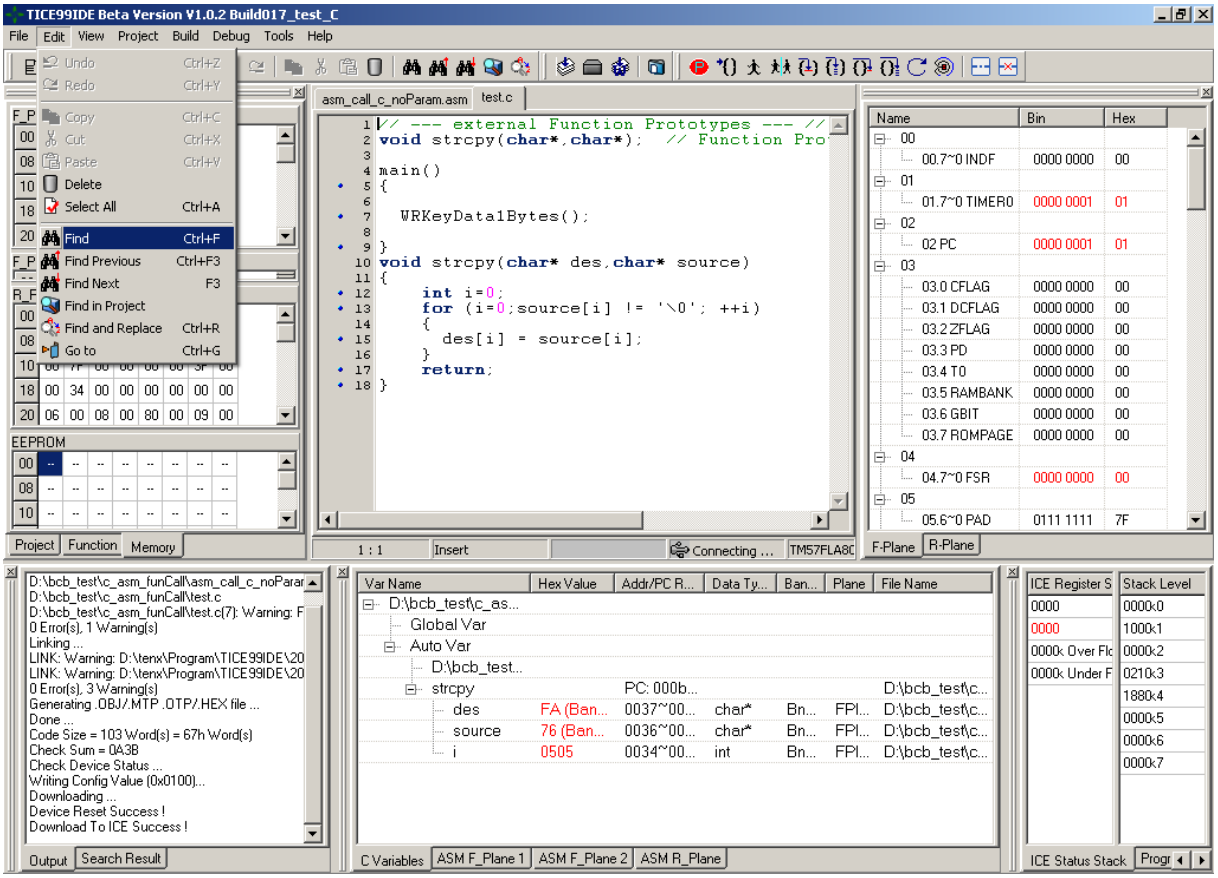
图表 40 书签

4.6 查找

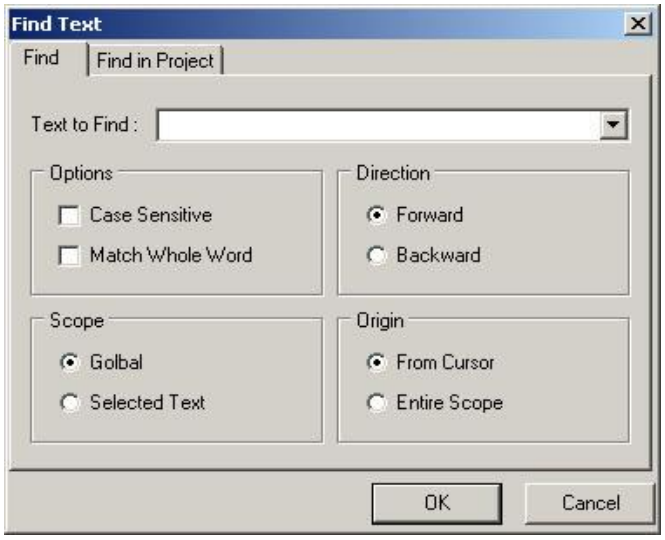
藉由查找功能，程序设计者可以在程序代码区域快速查找关键词。查找功能可帮助使用者更有效与快速的维护程序代码。除此之外，使用查找功能可找到关键词并作批次替代或修改。快速开启「查找」窗口的快速键是 Ctrl+F。一旦一个字符串有被找到，您可直接按 F3 来找到下一个或者 Ctrl+F3 来找前一个。快捷组合键的设定，请参考功能列 「工具」|「选项」 之快捷键页面。

4.6.1 寻找

「寻找」可以在目前正在用的编辑器页面里找文字或字词。您必须输入欲查找的字符串让 IDE 找给您。若字符串有被找到，输入光标会跳转到相对应的列行。若没有符合的字符串，会显示『已到文件结尾』。一旦您找到一个字符串，您可按 F3 或 Ctrl+F3 来找下一个或者前一个字符串。



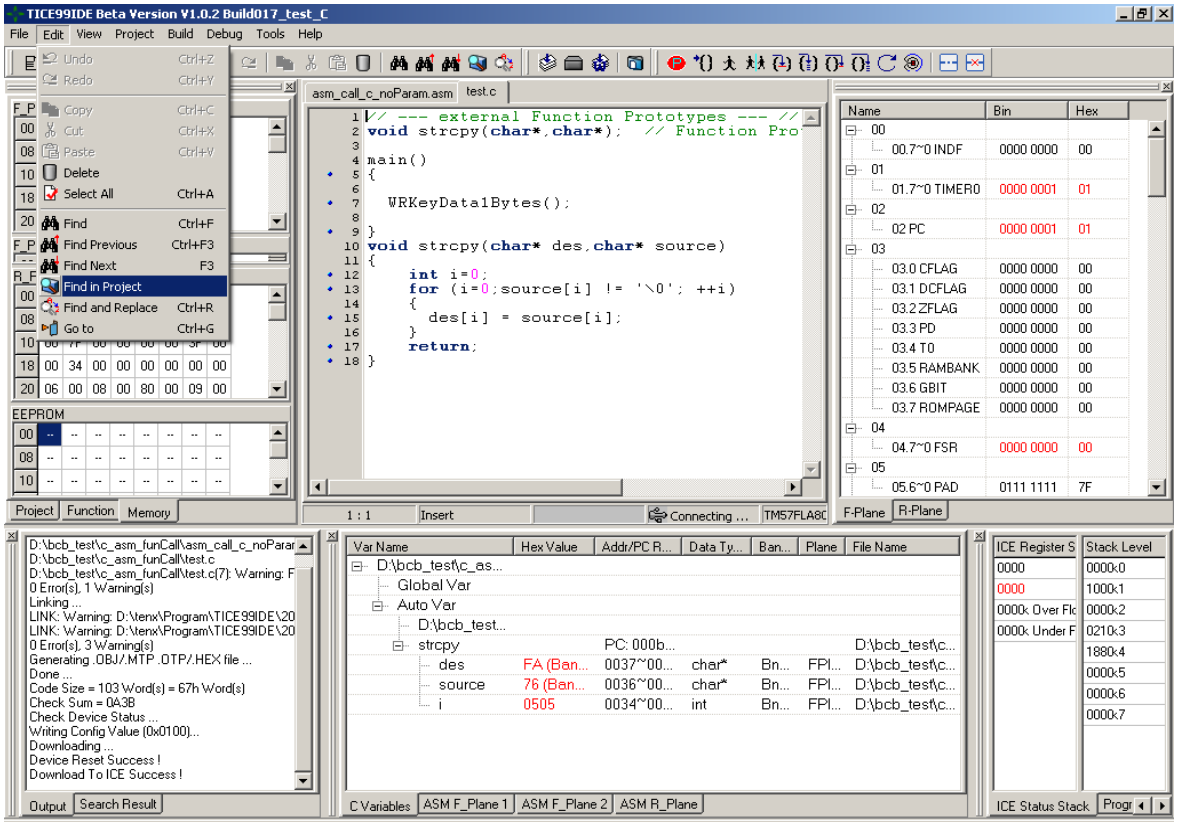
图表 41 寻找-1



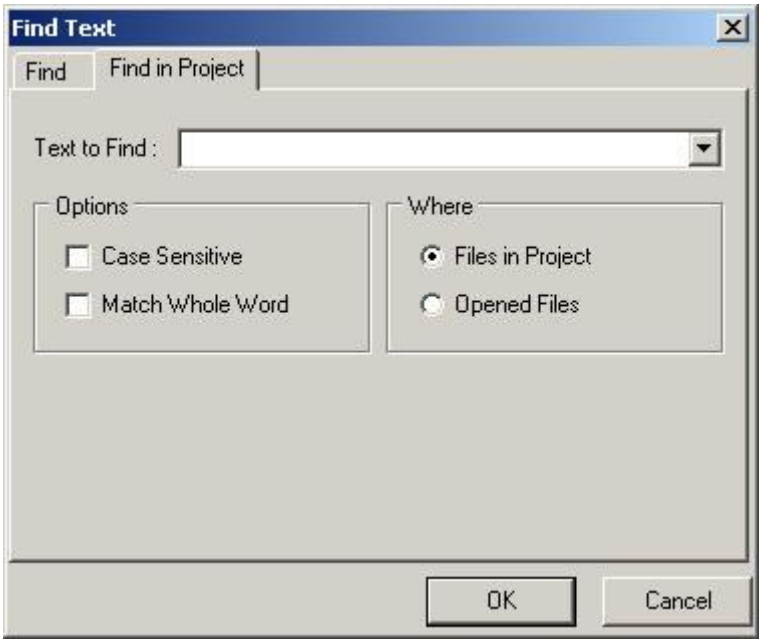
图表 42 寻找-2

4.6.2 在项目中寻找

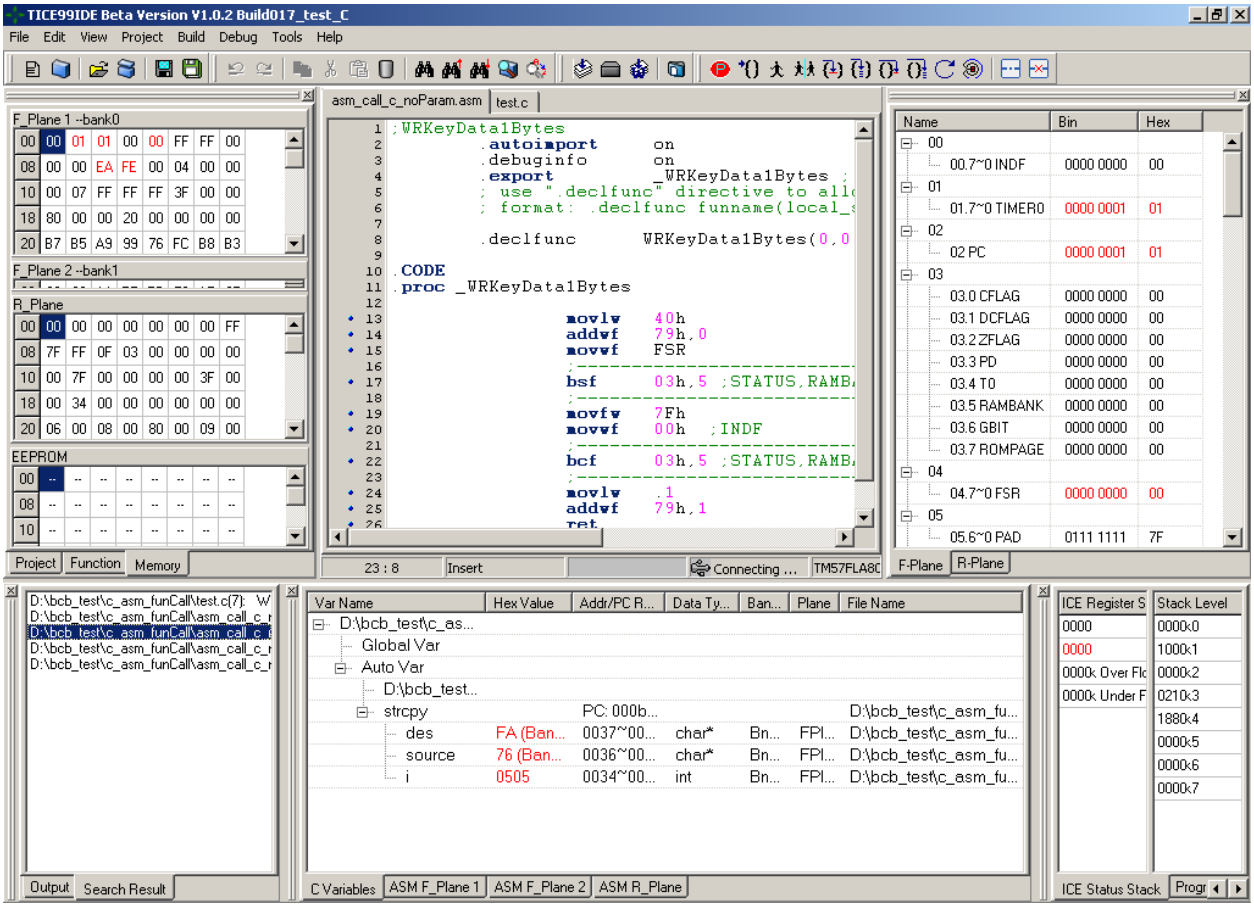
「在项目中寻找」将找出在项目或开启文件中，所有符合查找关键词的字符串。其查找的多笔结果将一一条列在「查找结果」的窗口中。使用者在「查找结果」的窗口中，可直接鼠标左键双击某一项查找结果，则系统会自动跳转到在相对应的文件中。



图表 43 在项目中寻找-1

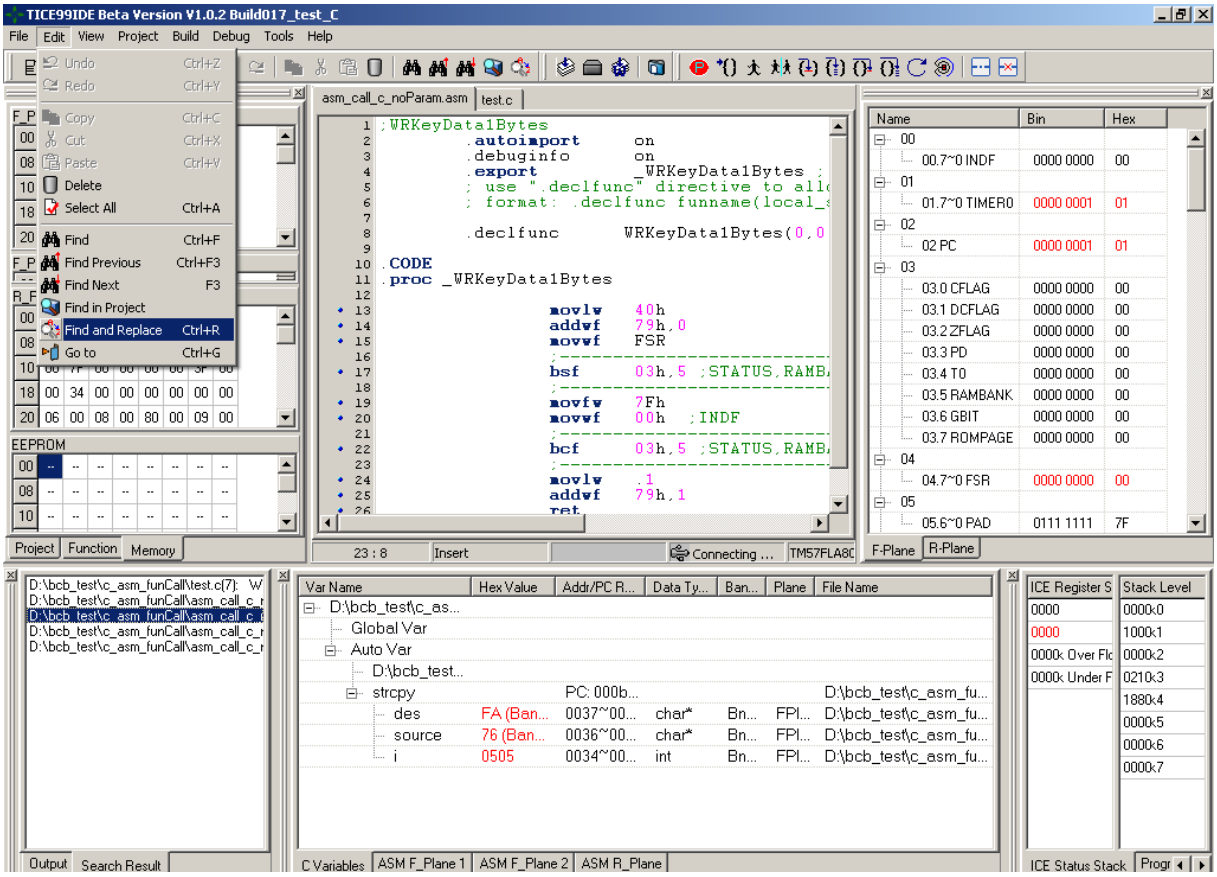


图表 44 在项目中寻找-2

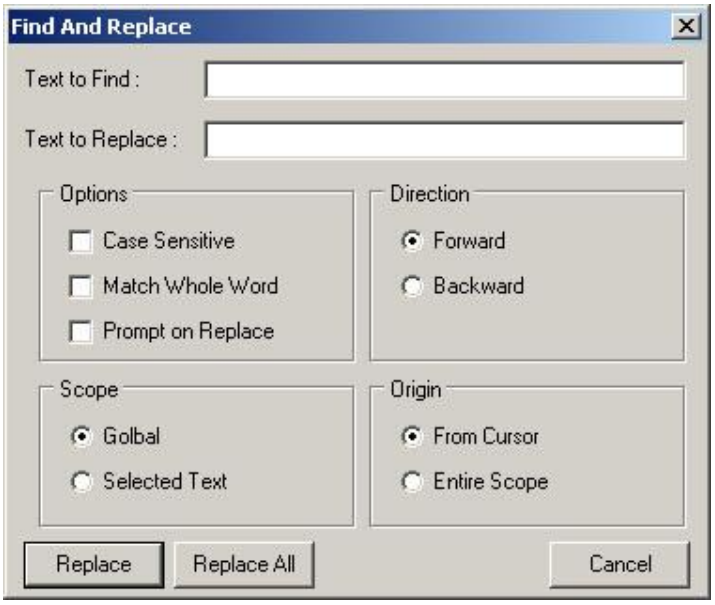


图表 45 在项目中寻找-3

4.6.3 寻找并替代



图表 46 寻找并取代-1



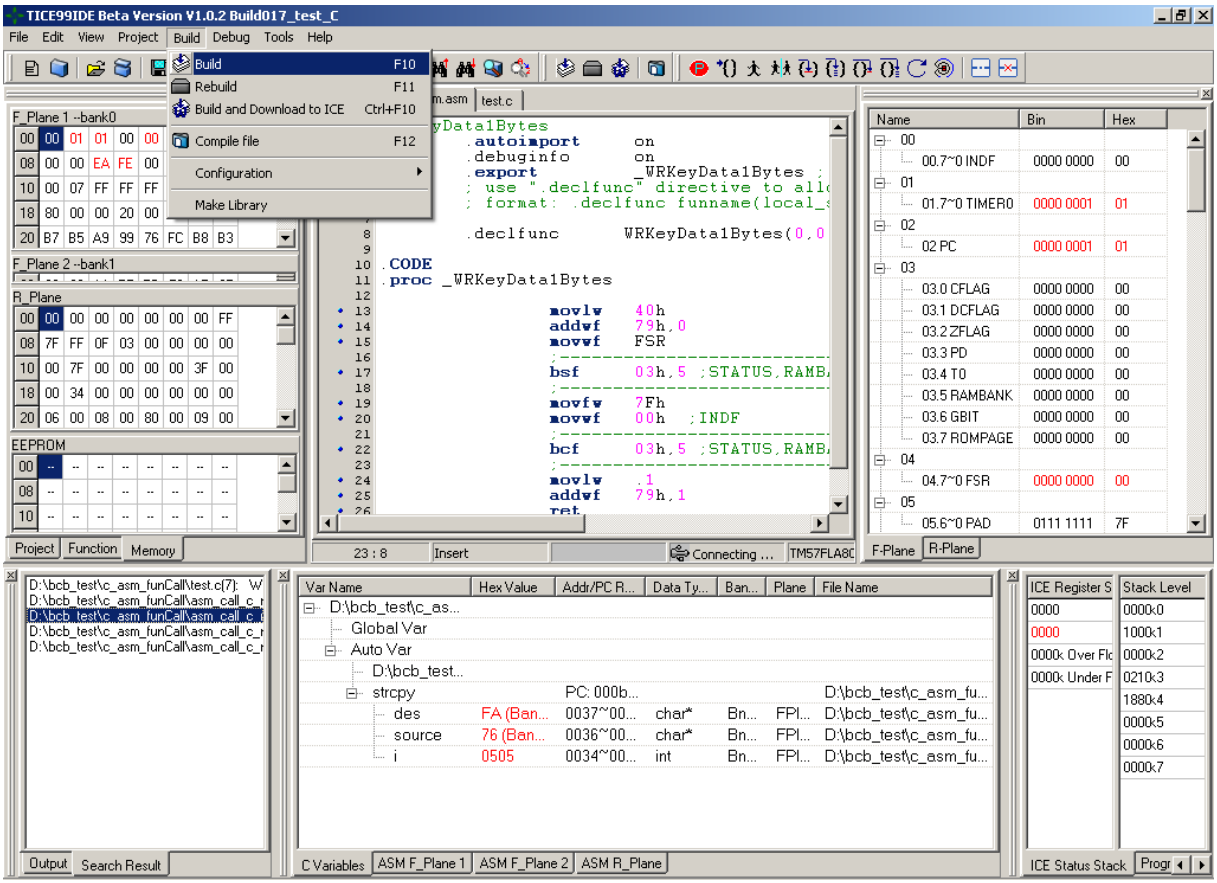
图表 47 寻找并取代-2

4.7 建立一个项目

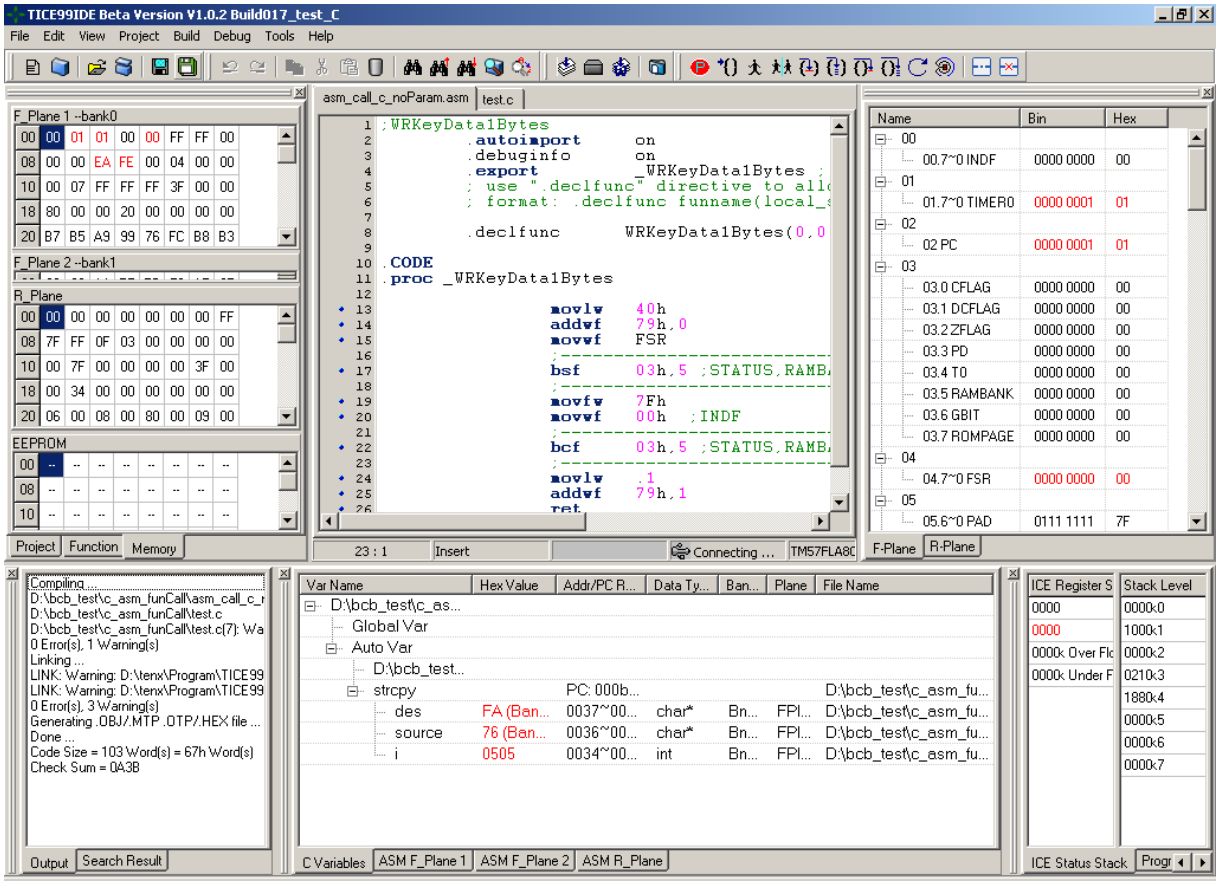
一旦完成程序代码的编写，您就可开始编译程序代码了。请点选功能列中「建立」|「产生目标档」 来检测程序代码是否有误。若有任何错误或警示，输出窗口将会显示其错误或警示信息。在输出窗口中鼠标双击某一错误或警示信息，IDE 将快速跳转到编辑文件中的错误列，以增加修改程序代码的效率。当程序代码没有错误的部分，请点选 「建立」|「建立并下载」到 ICE，来准备模拟您输入到 ICE 的程序代码并可进行调试程序。

4.7.1 建立文件

建立文件帮助使用者检查单一个 C 语言或汇编程序文件是否正确。若有任何错误，使用者可以依其错误信息来修改程序代码。



图表 48 建立项目-1

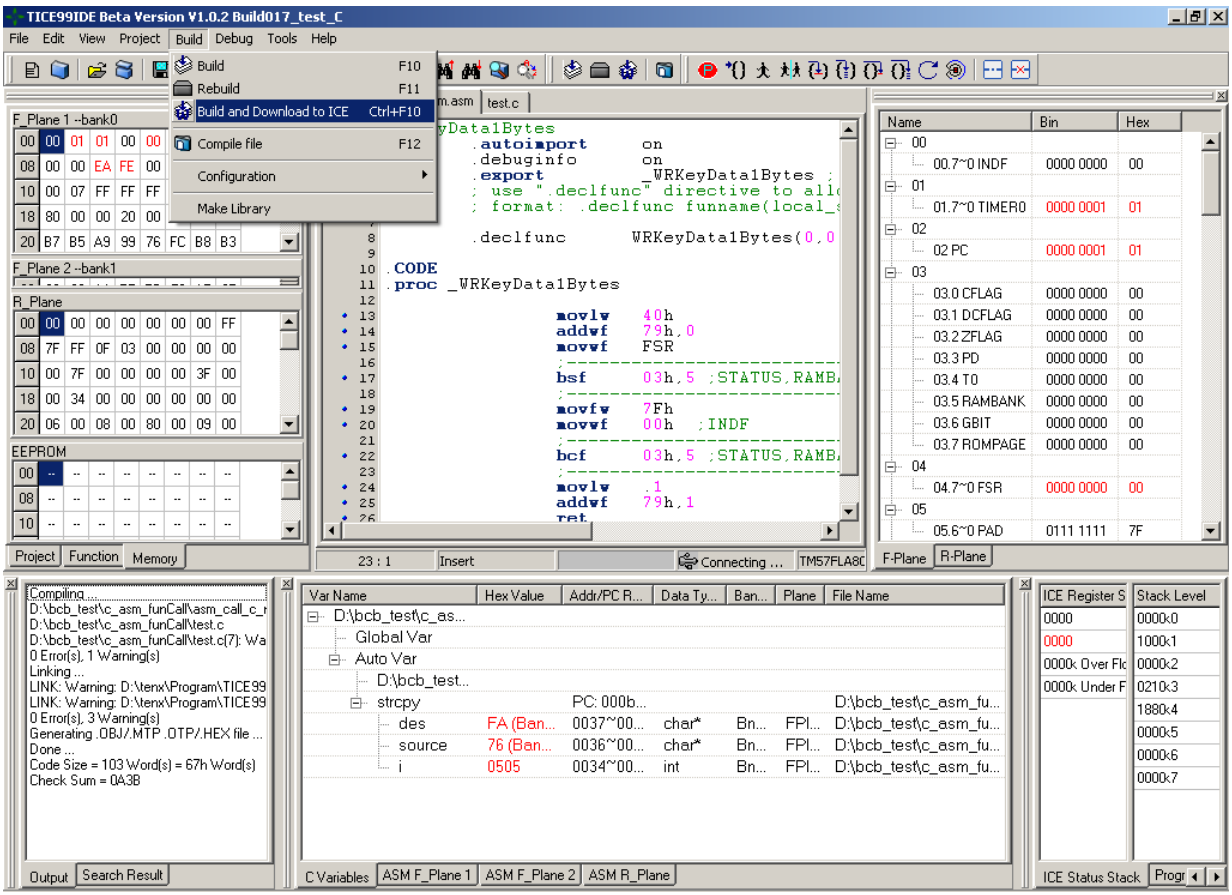

TICE99IDE Beta Version V1.0.2 Build017_test_C
File Edit View Project Build Debug Tools Help
asm_call_c_noParam.asm
test.c
F_Plane 1 --bank0
F_Plane 2 --bank1
R_Plane
EEPROM
Project Function Memory
Name Bin Hex
F-Plane R-Plane
Compiling ...
Output Search Result
Var Name Hex Value Addr/PC R... Data Ty... Ban... Plane File Name
C Variables ASM F_Plane 1 ASM F_Plane 2 ASM R_Plane
ICE Register S Stack Level
ICE Status Stack

图表 49 建立项目-2

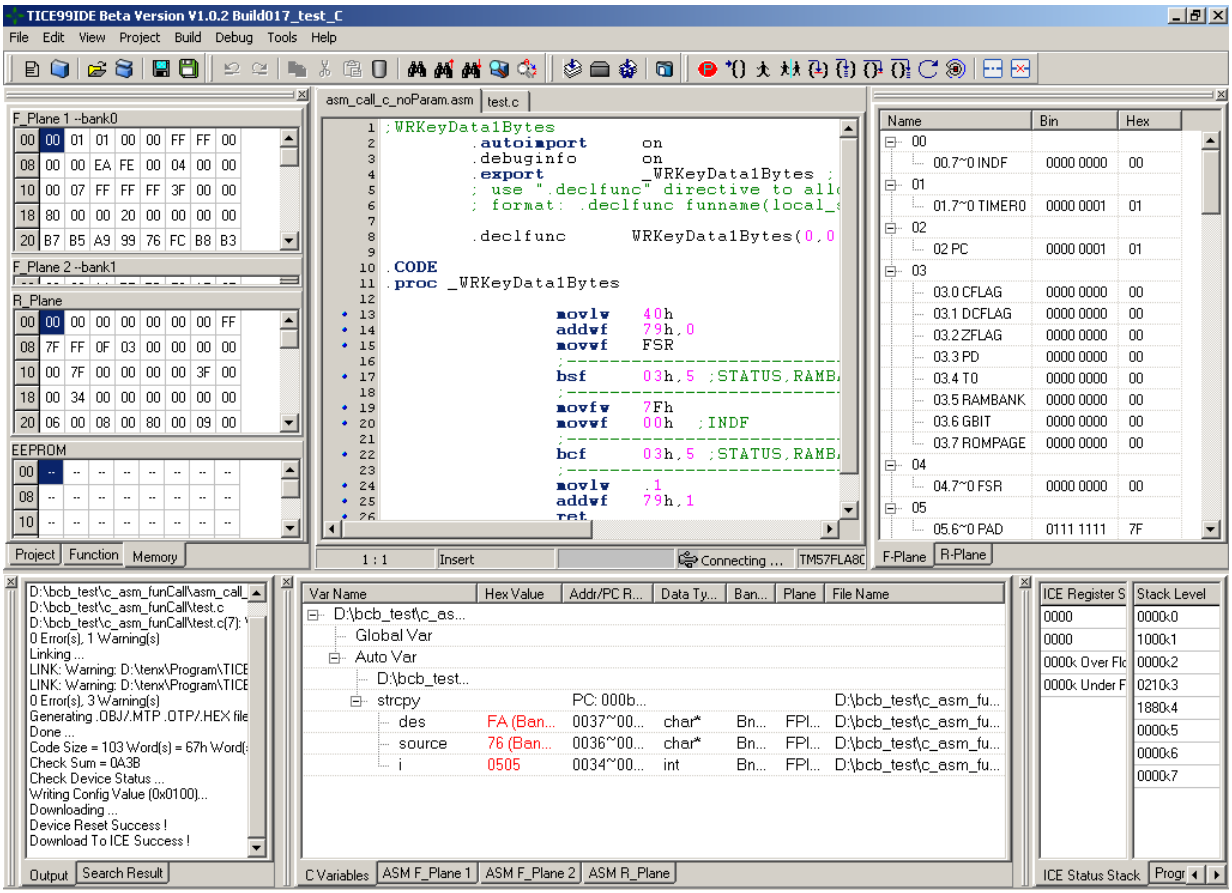
4.7.2 建立且下载到 ICE

一旦项目里面的程序代码没问题，您可以下载程序代码到 ICE 中，来模拟程序代码的动作。用「建立且下载到 ICE」把二进制的执行文件下载到 ICE。

注意：下载程序代码到 ICE 之前，您必须把 ICE 接到 PC 且开启 ICE 电路板，并且确认是成功联机到 PC。



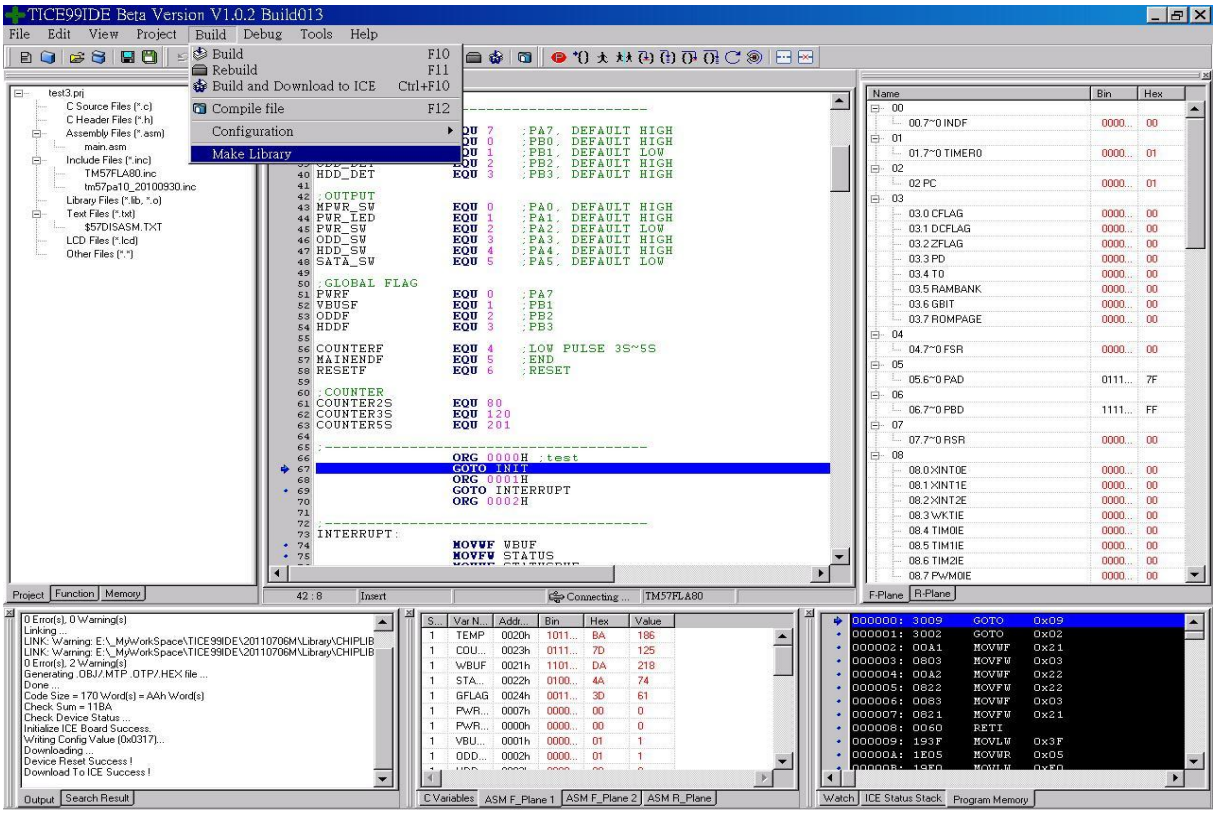
图表 50 建立且下载到 ICE-1



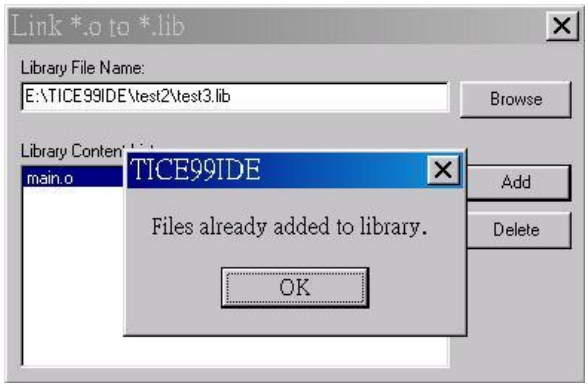
图表 51 建立且下载到 ICE-2

4.7.3 函数库

函数库与一般原始程序文件并没有什么不同，函数库所做的事是将程序代码区分出数个更小的程序单元，可以更好维护程序与更易于阅读。TICE99 IDE 提供使用者一建置函数库更方便的方式。藉由连结现存的.o 档以建立一个自定义的函数库(*.lib)。点选功能列之 「建立」|「建置函数库」。在建置函数库的窗口中，首先选择现存或新输入一函数库档名，接着按取”新增”按键以文件浏览要连结入函数库的.o 文件，或是按取”删除”按键以删除所选择的.o 文件，并且 IDE 皆会实时显示建置的结果信息。



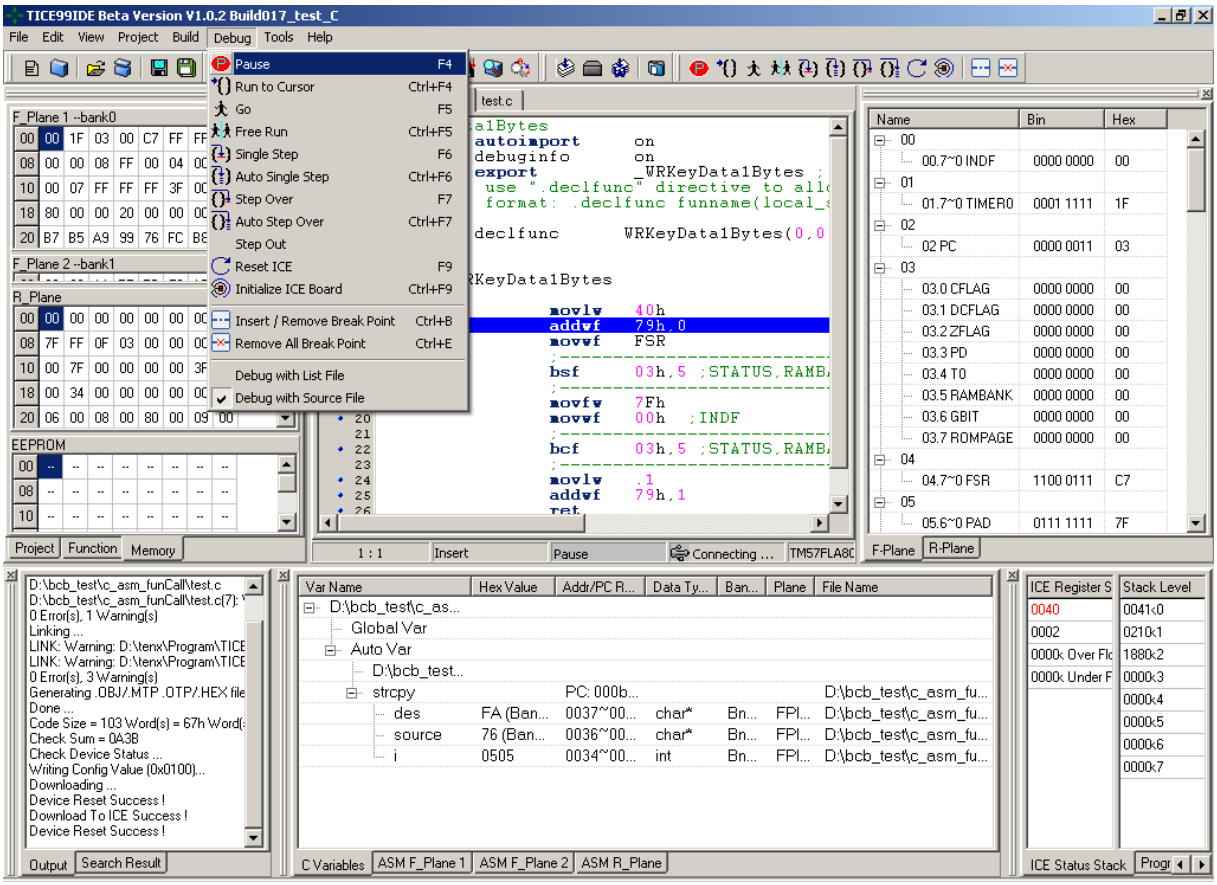
图表 52 建置函数库



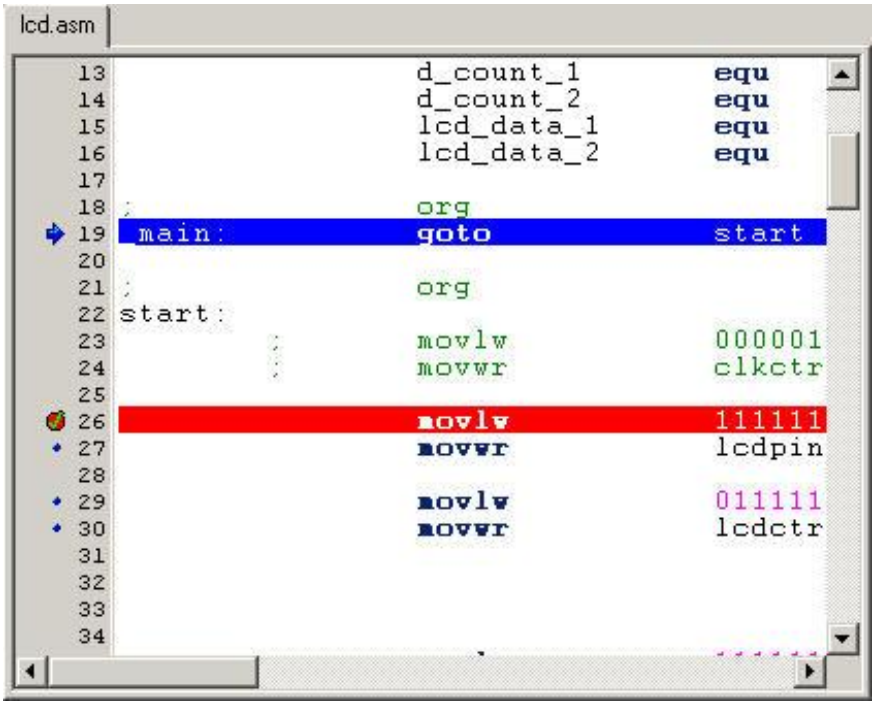
图表 53 建置函数库-提示讯息

4.8 以项目调试

TICE99IDE 提供许多方便的调试功能，包含「暂停」、「运行到光标位置」、「运行」、「自由运行」、「单步运行」和「单步不进入函数」。「自动单步」和「自动单步不进入函数」是从「单步运行」和「单步不进入函数」延伸出来的。当定时器被触发时，就会周期性执行「单步运行」或「单步不进入函数」。定时器可在「工具」|「选项」 的调试选项里面设定。定时器的单位为毫秒。



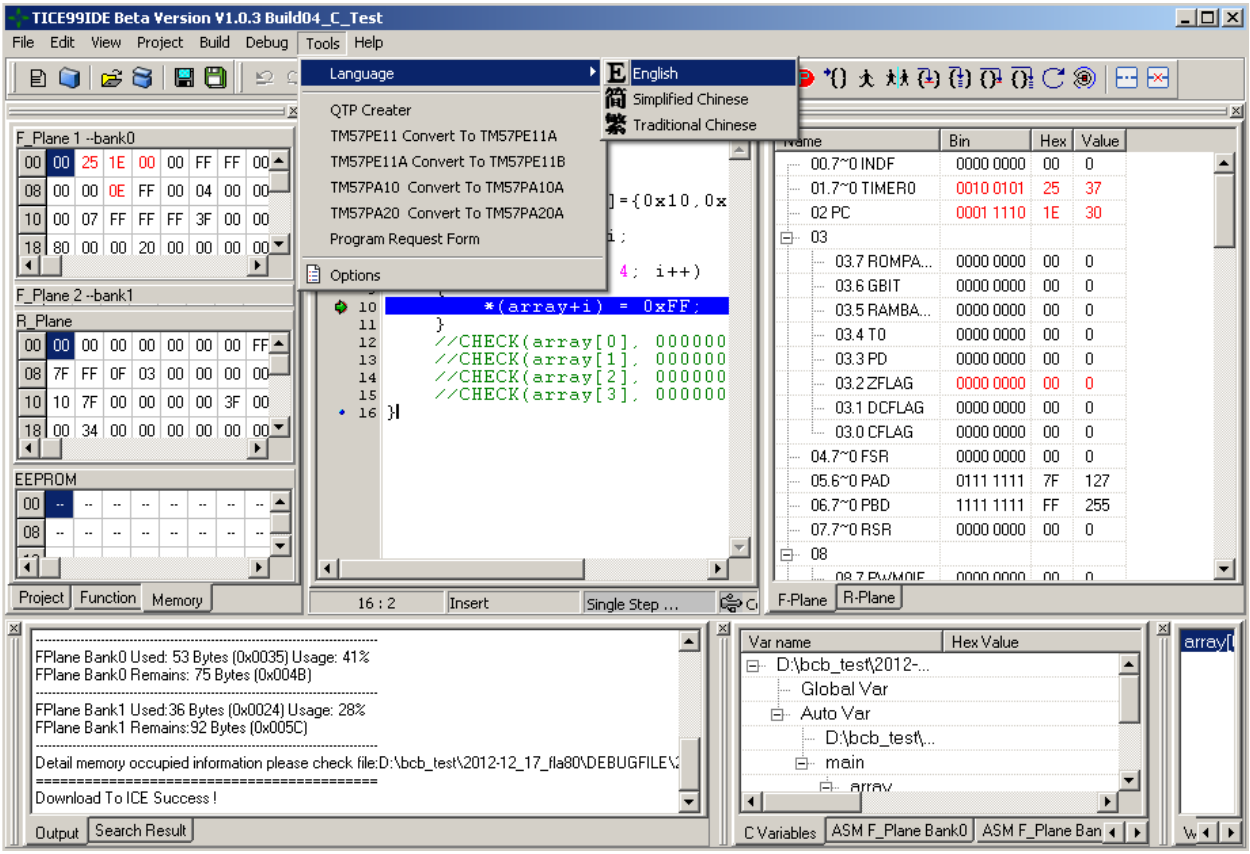
图表 54 以项目调试



图表 55 以项目调试-9 断点

4.9 切換语言界面

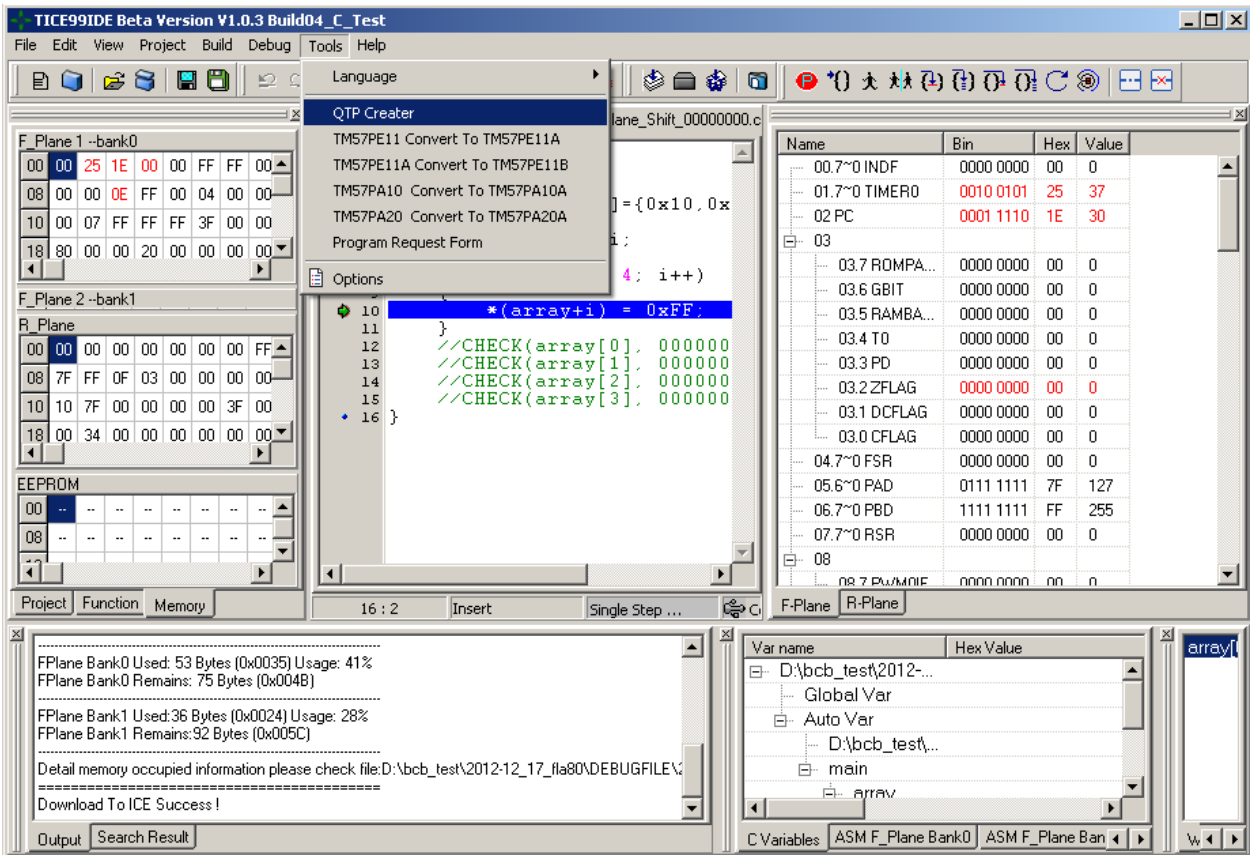
目前 TICE99IDE 提供三种不同的语言：英文、简体中文及繁体中文。点选功能列 「工具」|「语言」来切换到所需的语言。



图表 56 切换语言界面

4.10 QTP 产生器

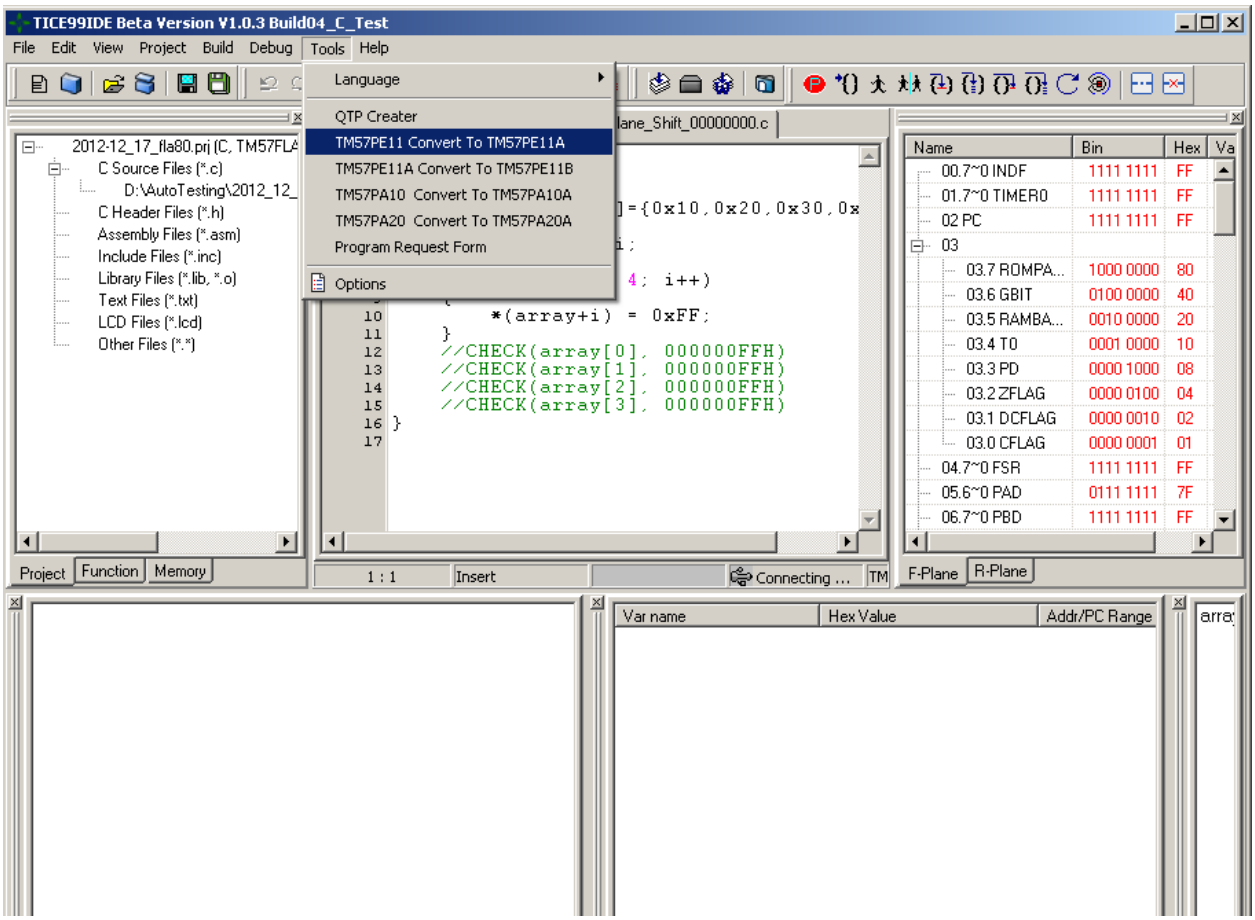
输出项目的编码申请表(Coding request form)，其包含 hex 文件档名(*.hex)、检查和(checksum)、系统配置、程序代码的大小及使用频率。



图表 57 QTP 产生器

4.11 16 进制文件转换

TICE99IDE 提供一个工具转换一个 16 进制档案由一个芯片类别至另一种类别并编程。在选择完所要转换的 hex 文件后，系统将显示一确认对话框以再次确认。即使转换成功，仍然有些需要使用者再次验证程序的提示内容，烦请耐心看完此重要提示信息。



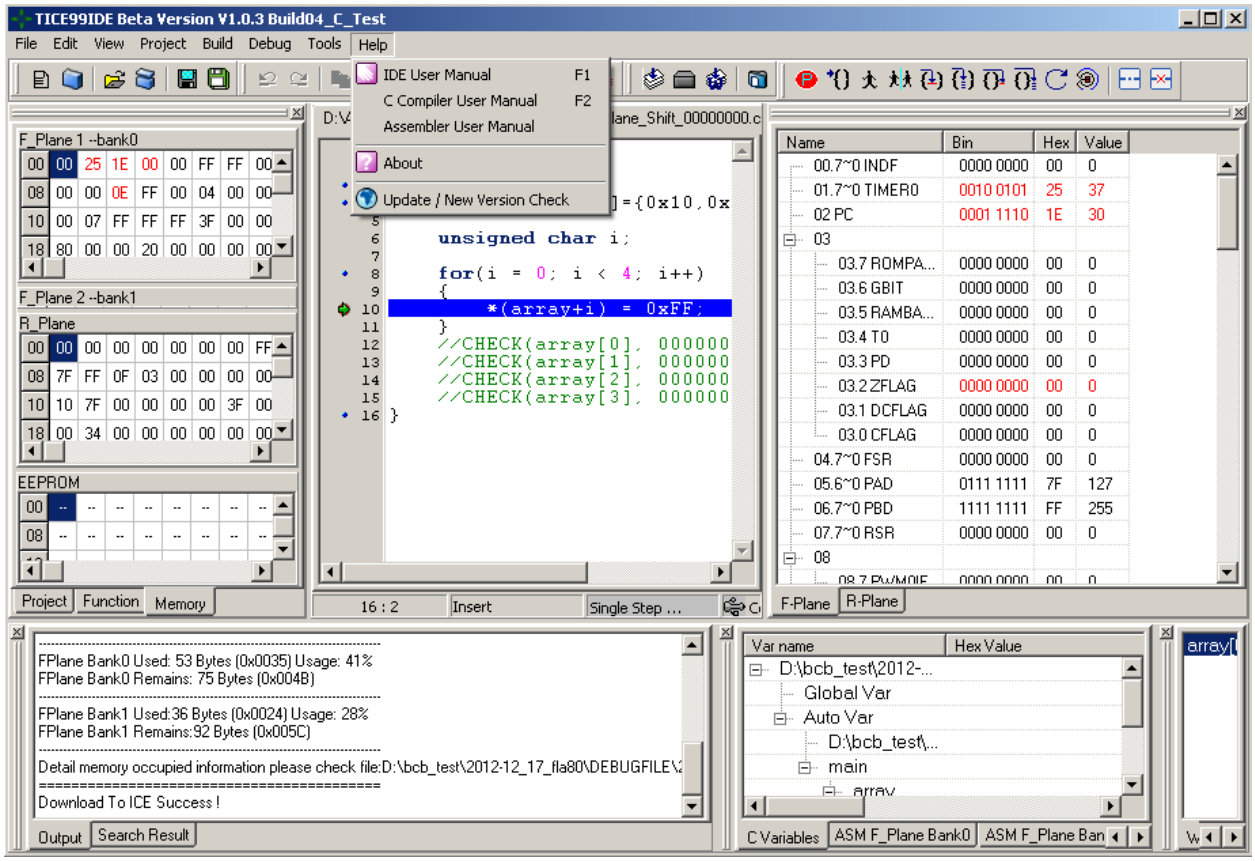
图表 58 转换 TM57PE11 为 TM57PE11A 格式

图表 59 转换提示信息

图表 60 转换完成-需使用者再次确认相关信息

4.12 使用说明

此使用说明只有介绍如何使用 IDE, C 编译器和汇编器。若您要知道如何在 IDE，C 编译器和汇编器编写程序代码，请参考程序编写指南。



图表 61 说明

5. 调试

ICE 可以通过调试来仿真程序在芯片中的运行状态。IDE 所显示的 ICE 缓存器可分为：内存、寄存器、ICE 堆栈状态和汇编助记码。以下我们将说明在各种情况下使用匹配的调试方法：

- **单步运行 / 自动单步：**若使用者欲追踪每个原始程序码发现一些问题，可以使用『单步运行』。「单步运行」将在运行完一个程序计数器步骤后即停止，而所有当前状态或寄存器的值皆会在 IDE 上显示。使用者可以利用这些信息判别问题。

  若您预期需点选多次「单步运行」来调试，另一有效的替代方式为使用「自动单步」来节省点选按键的时间。当您要停止「自动单步」，点选「暂停」就可以停住了。

- **单步不进入函数 / 自动单步不进入函数 ：**若您不想转移到函数的内部程序，可点选「单步不进入函数」。相同地，若您要连续使用「单步不进入函数」，您可以使用「自动单步不进入函数」。若您要停止「自动单步不进入函数」，可点选「暂停」。

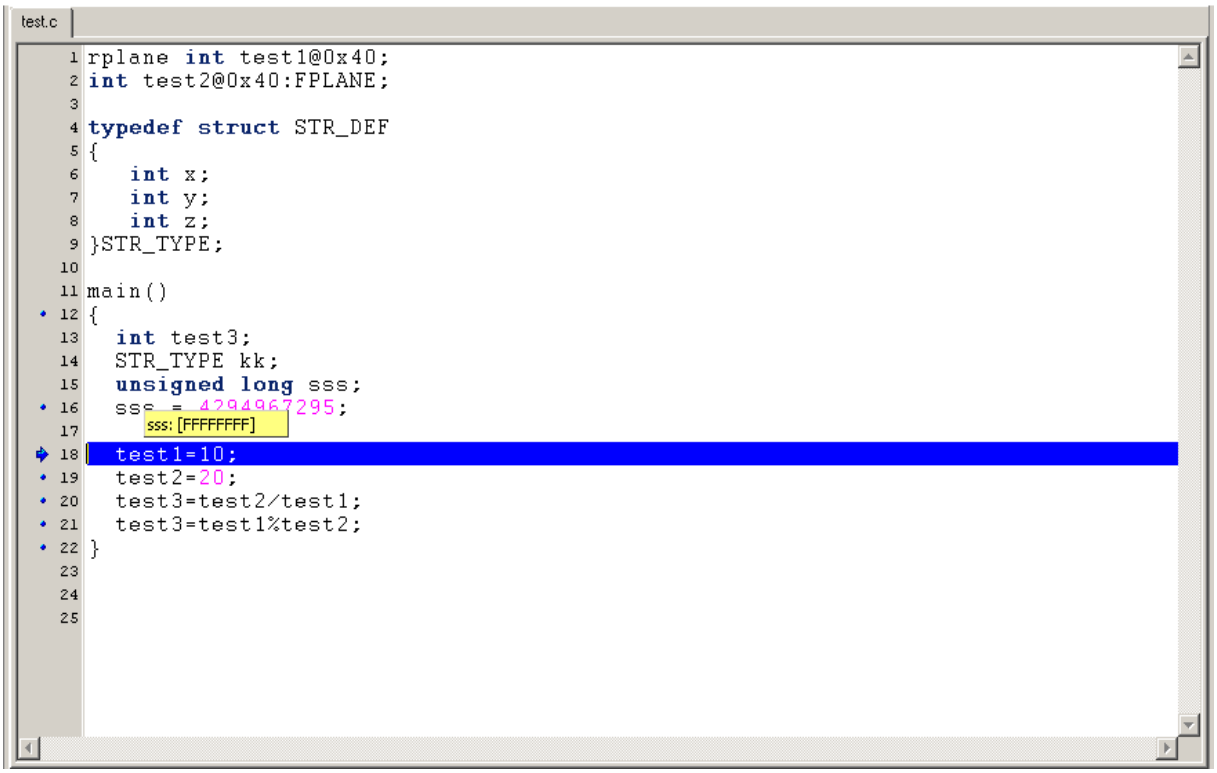
- **运行到光标位置 ：**您可以使用「运行到光标位置」来通过您测试的程序代码，并将调试步骤停在光标位置。

- **重置 ICE ：**当您想要从头开始，「重置 ICE」将帮助您达到此目的。

- **运行：**您也可以使用「运行」功能并搭配断点的设定来停止 ICE。请以「加入|删除断点」来维护断点的设定。另外，必须提到的一件事是，所有 EV 皆有可设定断点数目上限的限制，这取决于您选取的芯片类型，请详细参阅相关规格文件。

- **以原始文件调试/以 List 文件调试：**有两种方式进行程序代码调试：第一为原始来源档，另一种是 List 文件，您可选择对您最方便的方式。List 文件里面的一行表示一个程序计数器。然而在 C 程序，在原始来源档的一行将对应到一个范围的程序计数器。若您要仔细地追踪程序代码的运行，我们的建议是使用 List 文件。否则，可以选择原始来源文件来调试。

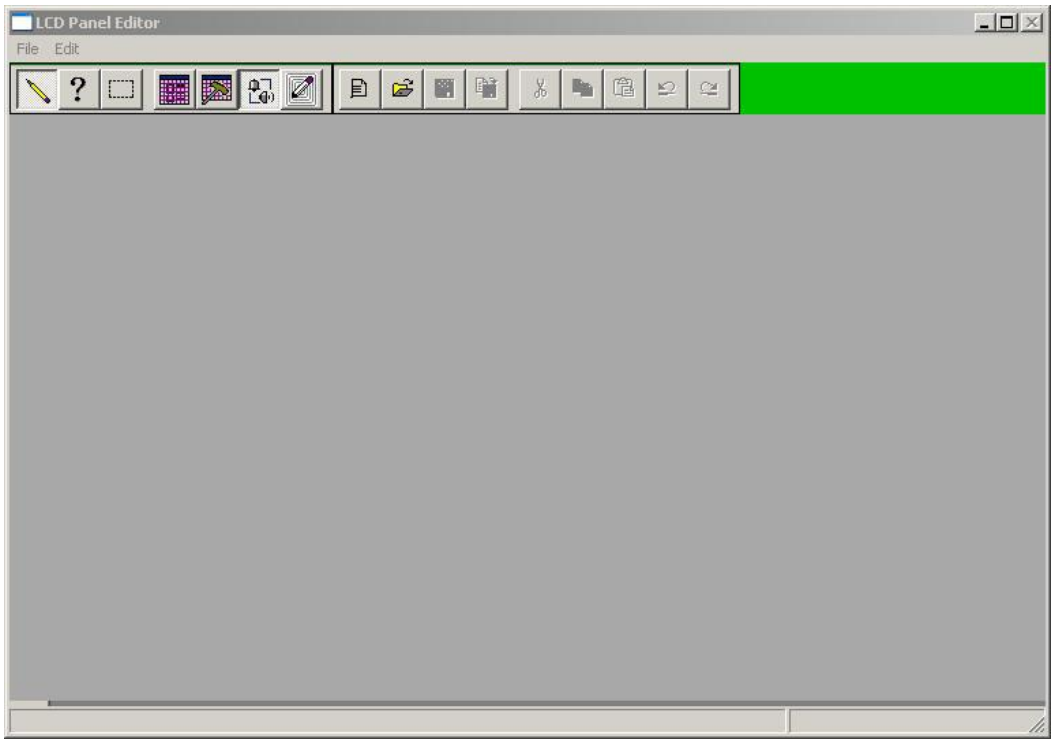
- **调试过程中显示变量值：**当已正在运行调试且在暂停状态时，您可以鼠标移到在执行点范围内之各个变量，IDE 将显示小提示信息框，以供使用者查看该变量之实时数值。这在调试时，让使用者以更便捷的方式来验证变量值。



图表 62 调试-变量提示信息

6. LCD 编辑器

LCD 编辑器是 IDE 所提供的一个独立功能。LCD 编辑器用以编辑 LCD 所要显示之内容， LCD 功能视连接的芯片是否有支持！LCD 编辑器分为三个部分：LCD 版面编辑器、LCD 属性和 LCD 样式编辑器。LCD 版面编辑器是主窗口。现在我们先从 LCD 样式编辑器开始说明。

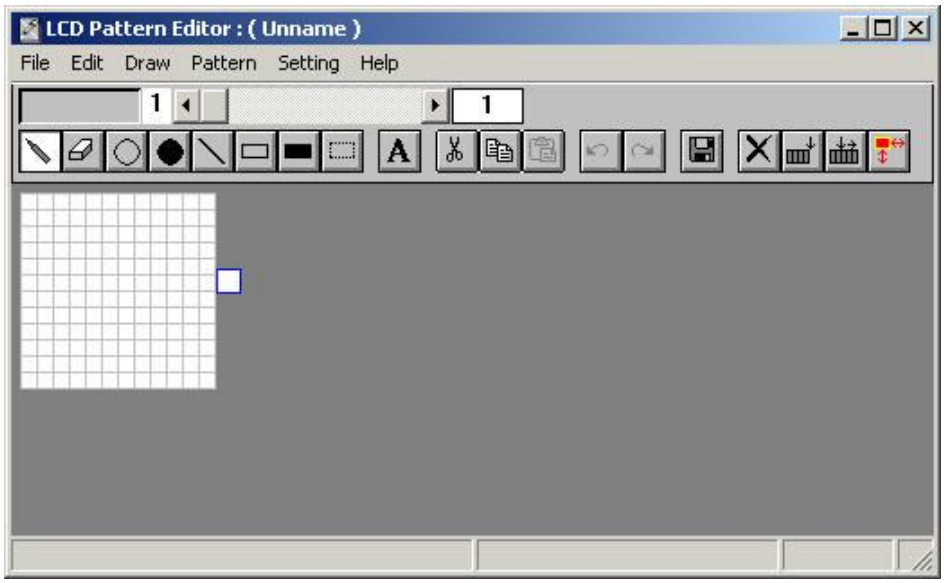


图表 63 LCD 编辑器版面

6.1 LCD 样式编辑器

使用者可利用 LCD 样式编辑器窗口来设计一些 LCD 样式以供重复使用，并且另储存为一扩展名为 .pat 之文件。样式尺寸的长与宽的设定范围为从 2 到 32。在开始编辑前，您须事先决定样式的尺寸，然后使用绘图工具来逐一摆放黑点、图形或文字到您前一个步骤所决定的样式。若您错放黑点的位置，您可以使用橡皮擦来擦掉错误。其它绘图工具有椭圆、线条、长方型、文字。文字是一个很方便的工具可让您更简单地画出文字。另外，还有五项编辑功能：剪切、复制、粘贴、返回及重做。

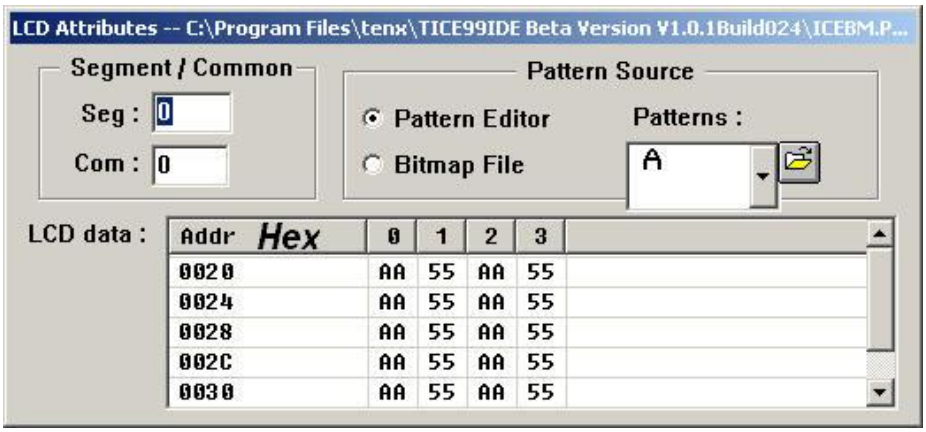
当您完成样式编辑，可以储存为一个文件来保存您所绘制的样式。样式文件是一系列的图案，您可以用「增加版面」增加在后面或「插入样式」来装饰您的样式。或者使用「移除」来移除目前的样式。您可以用「定义 LCD 样式」重新定义样式的尺寸。程序开发者可使用横滚动条的左右键来改变显示的版面以便于编辑，或用「指示器」来指定一样式编号以直接跳到该样式的编辑画面。



图表 64 LCD 样式编辑器

6.2 LCD 属性

当您建立一个样式文件，您可使用「LCD 属性」窗口来加载现存的样式文件。用「开启文件」加载样式文件，并且会显示在样式列表上面。选择一个样式然后切换到 LCD 版面编辑窗口，您就可以把样式放到 LCD 版面编辑器或者加载外部的位图文件(*.bmp)到 LCD 版面编辑器。除此之外，您可以设定样式的值，分为分段或一般。分段值永远代表字段，一般值永远代表列。「LCD 数据」显示 LCD RAM 的内容。



图表 65 LCD 属性

6.3 LCD 版面编辑器

在 LCD 版面编辑器，您可以设计将显示在与芯片联机的 LCD 版面的影像。您可选择使用  格子当标尺来安排影像。同时，您必须设定格子的尺寸  ，这包含宽与高，从 1 到 32。这取决于您想画在 LCD 样式编辑器的样式尺寸。LCD 版面编辑会根据样式或位图文件(*.bmp)摆放影像。若您放样式在错误的位置，您可使用「选择样式」  圈选样式并移动到正确的位置。「选择样式」可以选择超过一个以上的样式。其它移动样式的方式是属性询问。这部分提供分段或者一般的信息并且您可以同时地移动样式。当您结束编辑，可使用「储存文件」  来储存 LCD 文件，扩展名为.LCD。该文件可以被加进去项目里面以集中管理。

7. 快捷键

功能	快捷键
文件	
保存	Ctrl + S
编辑	
返回	Ctrl + Z
重做	Ctrl + Y
复制	Ctrl + C
剪切	Ctrl + X
粘贴	Ctrl + V
全选	Ctrl + A
寻找	Ctrl + F
找上一个	Ctrl + F3
找下一个	F3
寻找并替代	Ctrl + R
到	Ctrl + G
建立	
产生目标档	F10
重新产生目标档	F11
编译文件	F12
建置且下载到 ICE	Ctrl + F10
调试	
暂停	F4
运行到光标位置	Ctrl + F4
运行	F5
自由运行	Ctrl + F5
单步运行	F6
自动单步	Ctrl + F6
单步不进入函数	F7
自动单步不进入函数	Ctrl + F7
从函数跳出来	F8 (RFU)
自动从函数跳出来	Ctrl + F8 (RFU)
重置 ICE	F9
初始化 ICE	Ctrl + F9
加入/删除断点	Ctrl + B
说明	
说明	F1

表格 7 快速键

附录 A： ICE 板子上 LED 的意义

- V5.0：从交流电转接器的电源
- DONE：FPGA 下载文件成功，ICE 可正常运作