



**CSU-IDE V5.0 用户手册**  
Rev 1.0.5 (2019.04)

通讯地址：深圳市南山区蛇口南海大道 1079 号花园城数码大厦 A 座 9 楼  
邮政编码：518067  
公司电话：+(86 755)86169257  
传 真：+(86 755)86169057  
公司网站：[www.chipsea.com](http://www.chipsea.com)  
微 信 号：芯海科技  
微信二维码：



---

---

## 目录

<b>1 欢迎使用 CSU-IDE V5.0</b> .....	<b>6</b>
1.1 CSU-IDE V5.0 功能总述 .....	6
1.2 表达约定 .....	6
<b>2 安装环境</b> .....	<b>7</b>
2.1 系统环境 .....	7
2.2 安装和运行 .....	7
2.3 仿真小版 .....	7
<b>3 主界面</b> .....	<b>8</b>
3.1 主界面 .....	8
3.2 设置窗口状态 .....	11
<b>4 快速入门</b> .....	<b>13</b>
<b>5 工程管理</b> .....	<b>17</b>
5.1 关于解决方案 .....	17
5.2 创建解决方案 .....	18
5.3 添加工程 .....	20
5.4 添加文件 .....	21
5.5 添加文件夹 .....	22
5.6 解决方案管理器 (Solution Explorer) .....	23
5.7 工程设置 (Project Settings) .....	26
5.8 编译解决方案或工程 (Build) .....	34
5.9 输出窗口 (Output Window) .....	35
5.10 .lik 文件 .....	35
5.11 Class View 窗口 .....	36
5.12 导出反汇编文件 .....	37
5.13 Include/Exclude in Target Build .....	38
<b>6 文本编辑</b> .....	<b>39</b>
6.1 编辑区窗口 .....	39
6.1.1 关键字 .....	39
6.1.2 更改文字大小 .....	40
6.1.3 文件布局 .....	40
6.1.4 文件保存 .....	40
6.1.5 编辑区窗口功能 .....	40
6.1.6 选中行功能 .....	43

6.1.7 同名的头文件和源文件相互切换功能 .....	43
6.2 书签功能 (Bookmark) .....	43
6.3 文本查找和替换 .....	44
6.3.1 查找 .....	44
6.3.2 替换 .....	44
6.3.3 在文件中查找/替换 .....	45
6.3.4 Find result 窗口 .....	46
<b>7 调试器 .....</b>	<b>47</b>
7.1 下载 .....	47
7.2 调试操作 .....	47
7.3 调试窗口 (Debug Window) .....	48
7.3.1 反汇编窗口 (Disassembler) .....	49
7.3.2 寄存器窗口 (Register) .....	50
7.3.3 RAM\E2PROM\ROM 窗口 (RAM(Page0\1\2\3)、E2PROM、ROM) .....	51
7.3.4 变量窗口 (Watch) .....	52
7.3.5 断点窗口 (Breakpoint) .....	53
7.3.6 Trace Buffer 和 Call Stack 窗口 (Trace Buffer) .....	56
7.4 调用图窗口 (Call Graph Window) .....	59
<b>8 LCD 模拟器 .....</b>	<b>61</b>
8.1 LCD 模拟器的图形界面 .....	61
8.2 LCD 模拟器图形界面的配置 .....	61
8.3 LCD 模拟器映射关系设置 .....	62
8.4 LCD 模拟器的仿真 .....	64
8.5 LCD 模拟器配置的自动保存 .....	65
8.6 导入/导出功能 .....	65
<b>9 蓝牙 .....</b>	<b>67</b>
9.1 导入蓝牙包 .....	67
9.2 蓝牙设置 .....	68
<b>10 用户自定义 .....</b>	<b>69</b>
10.1 文本编辑设置 (TextEditor) .....	70
10.1.1 常用设置 (General) .....	70
10.1.2 字体和颜色设置 (Fonts and colors) .....	71
10.1.3 格式定义 (Formatting) .....	72
10.2 调试设置 (Debugging) .....	73
10.3 工具快捷菜单设置 (Custom Tools) .....	73

---

---

<b>11 菜单项介绍</b> .....	<b>75</b>
11.1 文件 (File) .....	75
11.2 编辑 (Edit) .....	76
11.3 视图 (View) .....	77
11.4 工程 (Project) .....	78
11.5 构建 (Build) .....	79
11.6 调试 (Debug) .....	79
11.7 工具 (Tools) .....	80
1.8 窗口 (Window) .....	81
11.9 帮助 (Help) .....	81
<b>12 工具栏介绍</b> .....	<b>83</b>
12.1 标准工具条 (Standard) .....	83
12.2 文本编辑条 (TextEditor) .....	83
12.3 构建工具条 (Build) .....	84
12.4 调试工具条 (Debug) .....	85
<b>13 USB 驱动安装</b> .....	<b>86</b>
<b>14 检测 IDE 和 ICE 最新版本</b> .....	<b>89</b>
<b>15 LibMaker</b> .....	<b>90</b>
15.1 界面.....	90
15.2 基本操作步骤.....	91
<b>16 Pack Installer</b> .....	<b>92</b>
16.1 界面.....	92
<b>17 SDK</b> .....	<b>93</b>
17.1 界面.....	93
<b>18 ICD</b> .....	<b>94</b>
18.1 设备.....	94
18.2 Setting 设置.....	95
18.3 调试.....	95
18.3.1 断点 .....	96
18.3.2 Trace Buffer 窗口 .....	98
18.3.3 ADC 窗口.....	98
18.3.4 RAM 窗口.....	99

**版本说明:**

版本	版本说明	发布日期
V1.0.5	1、 增加 ICD 说明	2019-04
V1.0.4	1、 增加 LCD 导出、导入*.elcd 功能 2、 增加 Generate Disassembly File 功能 3、 增加 Rom、Call graph 窗口 4、 支持编辑窗口鼠标左键快速单击 3 次选中行功能 5、 增加 Tools-》Custom Tools 功能 6、 增加"Include in Target Build" & "Exclusive in Target Build" 功能 7、 增加目标文件增加输出*.lib 文件功能 8、 增加头文件和源文件相互切换功能 9、 增加工程设置快捷图标	2018-10
V1.0.3	1、 支持 Setting-》General-》Target File Name 编辑功能 2、 增加 Setting-》Assembler-》Enable case sensitivity 功能 3、 增加 Setting-》General-》Watchdog reset not re-initialize sram 功能 4、 增加 Tools-》Pack Installer 功能 5、 增加 Tools-》Options-》Debugging 功能 6、 增加 Help-》Contact Me ， 提供联系方式	2018-06
V1.0.2	1、 适用于 V5.1.0 及以上版本 2、 增加 Trace Buffer、Call Track、蓝牙、Lib Maker、Class View 等窗口说明	2018-01
V1.0.1	1、支持 ASM 工程类型，支持 14bit IC 的 c 工程类型 2、支持代码选项快捷功能 3、支持 XP 操作系统 4、支持 Check New Version 功能	2017-06
V1.0.0	初稿	2017-04

---

---

# 1 欢迎使用 CSU-IDE V5.0

---

---

欢迎使用 CSU-IDE V5.0。CSU-IDE V5.0 是由芯海科技（深圳）股份有限公司开发的一套 IDE。

CSU-IDE V5.0 替换了 CSU-IDE V4.0 中的编译器、汇编器、链接器、调试器、以及新增 lib 制作工具，增强了 IDE 编译和调试过程的稳定性和准确性。它支持本公司混合信号处理器 CSU 系列产品的开发，并支持 CSU C 和 ASM 编程语言。CSU-IDE V5.0 提供了强大的工程管理、编辑、设置、查找以及调试等功能，以及更加友好的界面。

## 1.1 CSU-IDE V5.0 功能总述

- 1、引入解决方案的概念，提供了强大的项目管理功能和编辑、搜索功能。
- 2、支持 C89 和 ASM 编程语言。
  - 1) C 语言规格请点击 Help-> CSU C Programming Guide;
  - 2) Asm 语言规范请点击 Help-> CSU ASM Programming Guide;
- 3、加强编辑器功能，提供智能感应和自动完成功能。
- 4、加强调试器功能，优化所有调试窗口，并提供 Mixed View 功能。
- 5、增加断点窗口、书签窗口，并能持久化断点、书签的设置。
- 6、增加工具链的设置选项，使编译过程更符合用户使用需求。
- 7、支持制作 lib 文件，以及工程链接 lib 文件。

注：V5.0.3 及以后的版本已支持 C 和 ASM 工程类型。

## 1.2 表达约定

在使用手册中，有如下约定的表达方式：

➤ 菜单操作

Project | Open Solution: 表示从工程菜单栏 Project 中选择 Open Solution 命令。

➤ 鼠标操作

单击一般指单击鼠标左键，双击一般指双击鼠标左键。

➤ 窗口和对话框

“打开”对话框代表该对话框的名字为“打开”。

---

---

## 2 安装环境

---

---

本章将介绍如何安装和运行 CSU-IDE V5.0。

### 2.1 系统环境

支持的操作系统：XP、WIN7、WIN8、WIN10。

### 2.2 安装和运行

➤ 安装：双击安装程序 CSU\_IDE.msi，按照安装程序的指示进行安装。

如果登陆操作系统的为非管理员权限用户，在安装过程中会提示需提升为管理员权限才能继续安装。

说明：不支持安装在网络盘

➤ 运行：选择桌面快捷方式运行；也可选择【开始】→【程序】→【Chipsea】→【CSU-IDE 版本号】运行 CSU-IDE。

### 2.3 仿真小版

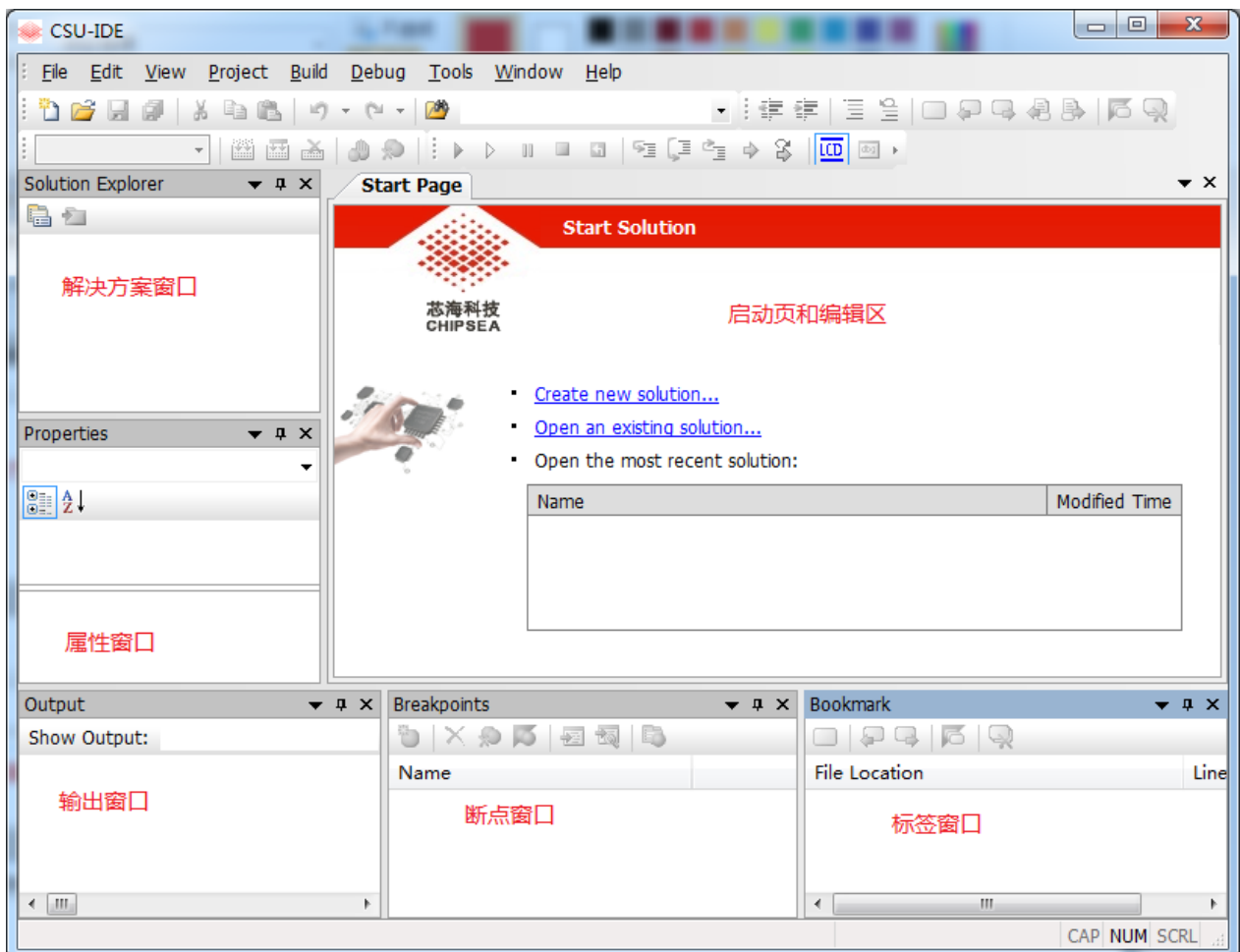
请参考简易仿真器及仿真小板说明书。

## 3 主界面

本章将帮助您初步了解 CSU-IDE 的主界面。CSU-IDE 的界面使用了集成环境开发的经典布局，可以根据个人喜好，随意拖动并组合各个窗口、菜单栏、工具栏等。

### 3.1 主界面

第一次开启 IDE，可以看到如下界面，包括启动页面（start page）、解决方案窗口、属性窗口、编辑区、输出窗口等。

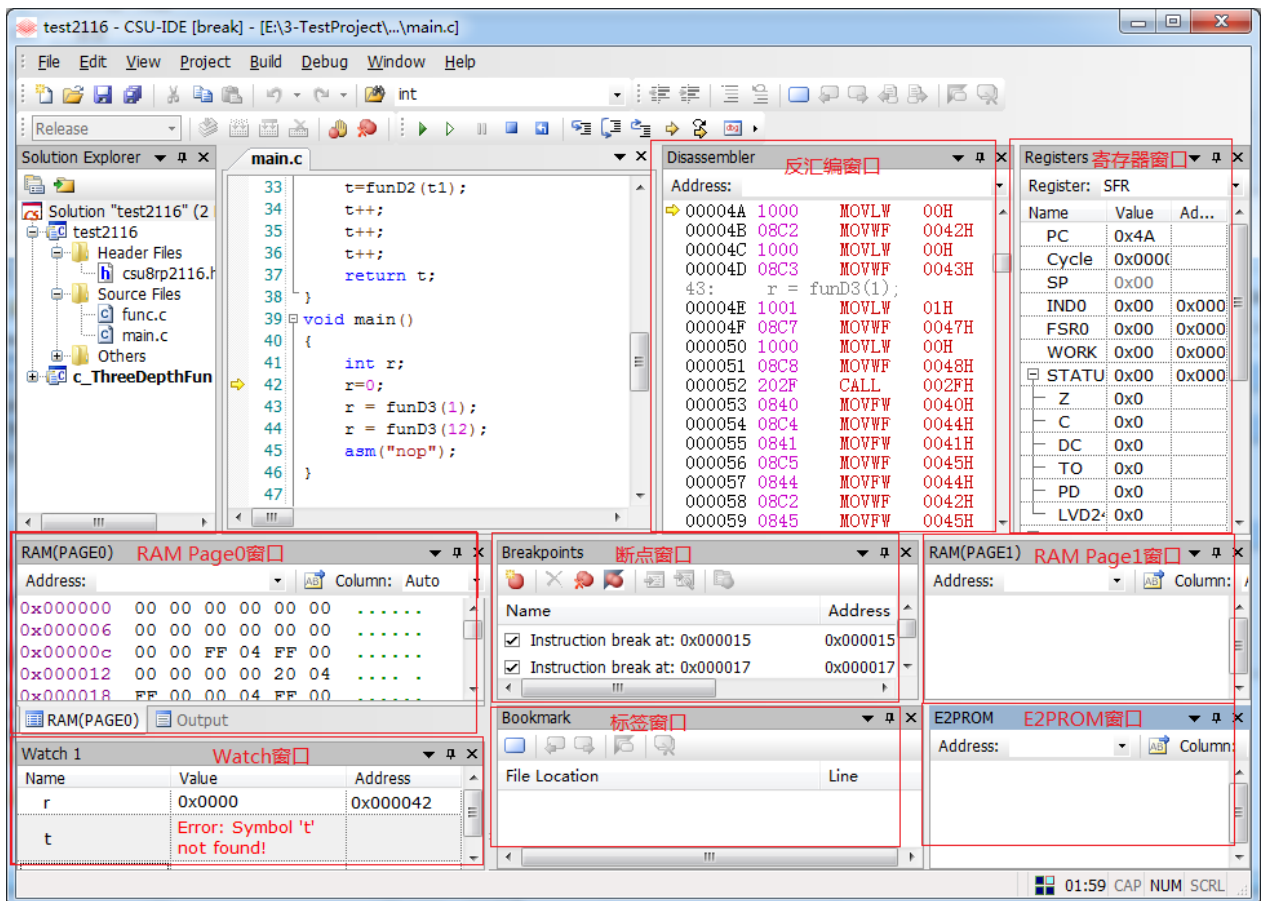


窗口	作用
<b>Solution Explorer</b>	显示解决方案、工程以及文件，对所包含的内容进行管理
<b>Properties</b>	显示在 Solution Explorer 窗口中选中的解决方案、工程以及文件的属性
<b>Start Page</b>	提供快捷方式，可快速新建工程和打开工程



<b>Editor</b>	查看、编写代码区域
<b>Output</b>	显示编译或调试过程中生成的信息
<b>Bookmark</b>	显示书签相关信息
<b>Find Results</b>	显示查找结果
<b>BreakPoints</b>	显示硬件断点信息，及进行断点设置等操作
<b>LCD Simulator</b>	设置 COM、SEG 窗口（详见 LCD Simulator 说明）
<b>Class View</b>	显示定义的函数和全局变量（不包括 SysRegDefine.c 中定义的），默认不显示 Class view 窗口，可通过菜单 View->Class View 打开此窗口。

调试模式下，可查看调试窗口。



窗口	作用
<b>Disassembler</b>	对下载的目标文件进行反汇编操作并显示，显示源程序中每行有效源码对应的汇编指令
<b>Register</b>	显示通用寄存器，CPU 状态寄存器和一些功能寄存器的值

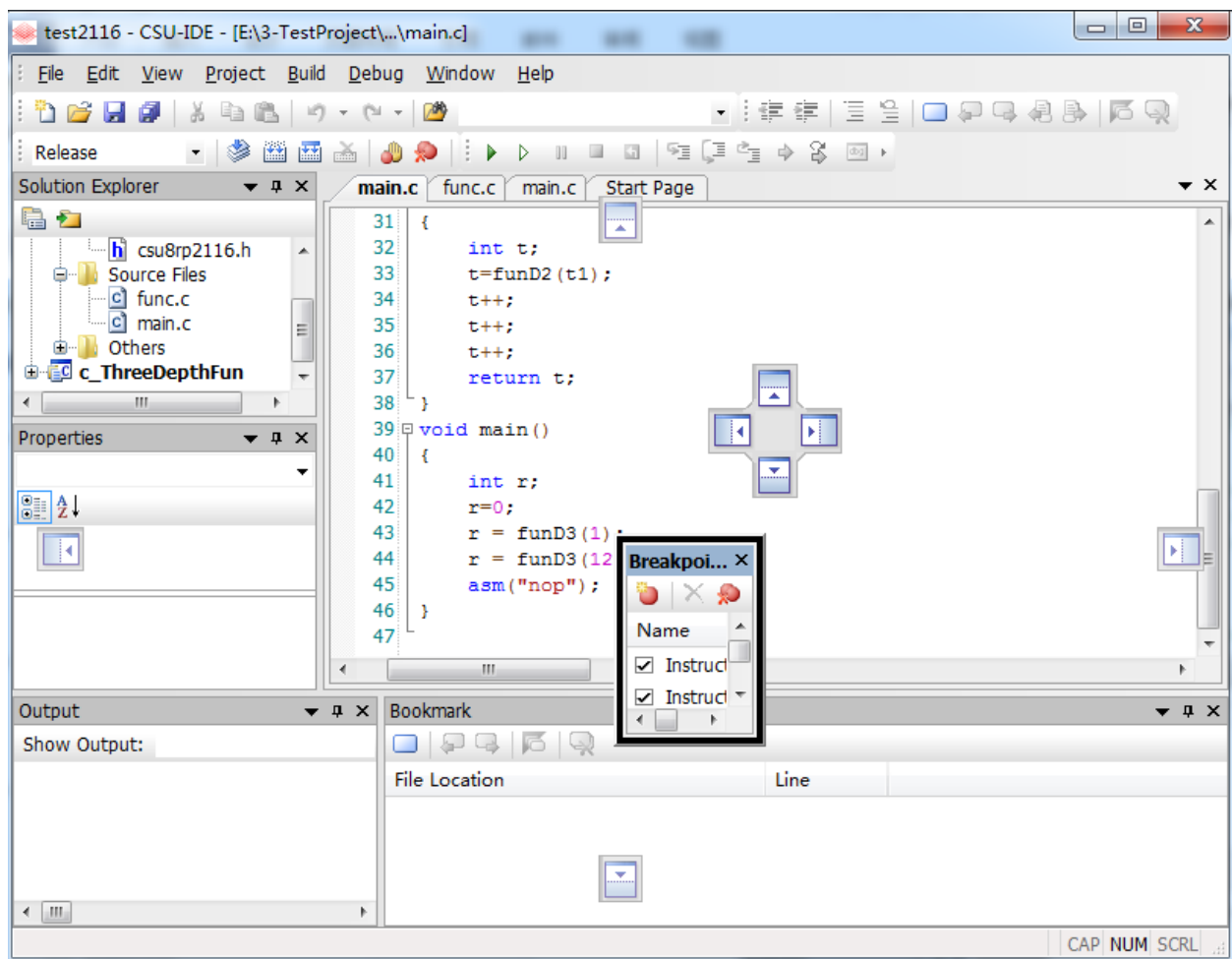
<b>RAM(Page0)</b>	显示当前 RAM(Page0)内存中的内容
<b>RAM(Page1)</b>	显示当前 RAM(Page1)内存中的内容
<b>RAM(Page2)</b>	显示当前 RAM(Page2)内存中的内容
<b>RAM(Page3)</b>	显示当前 RAM(Page3)内存中的内容
<b>E2PROM</b>	显示 E2PROM 中的内容
<b>Watch</b>	显示变量名、变量值、类型或地址
<b>LCD Simulator</b>	显示 LCD 窗口
<b>Trace Buffer</b>	显示 Trace Buffer 窗口, 默认不显示 Class view 窗口, 可通过菜单 View->Debug->Trace Buffer 打开此窗口。
<b>Call Stack</b>	显示 Call Stack 窗口, 默认不显示 Class view 窗口, 可通过菜单 View->Debug->Call Stack 打开此窗口。

## 3.2 设置窗口状态

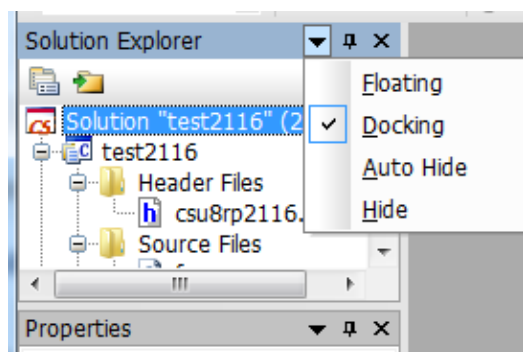
IDE 提供了灵活的窗口布局功能，每个窗口状态可以随意更改：悬浮、停靠、自动隐藏或隐藏等

### ➤ 拖动窗口


可选中一个窗口进行拖动，拖动过程中会有指示箭头，根据指示箭头停靠在任意位置。参考下图：



在每个窗口的右上角，会有一个向下箭头，点击该箭头，会弹出窗口状态供选择，参考下图：



- Floating: 将窗口置为悬浮状态，该状态下可随意拖动；选中窗口的标题栏直接拖动，也可以将窗口置为悬浮状态。
- Docking: 将窗口停靠在某个位置。

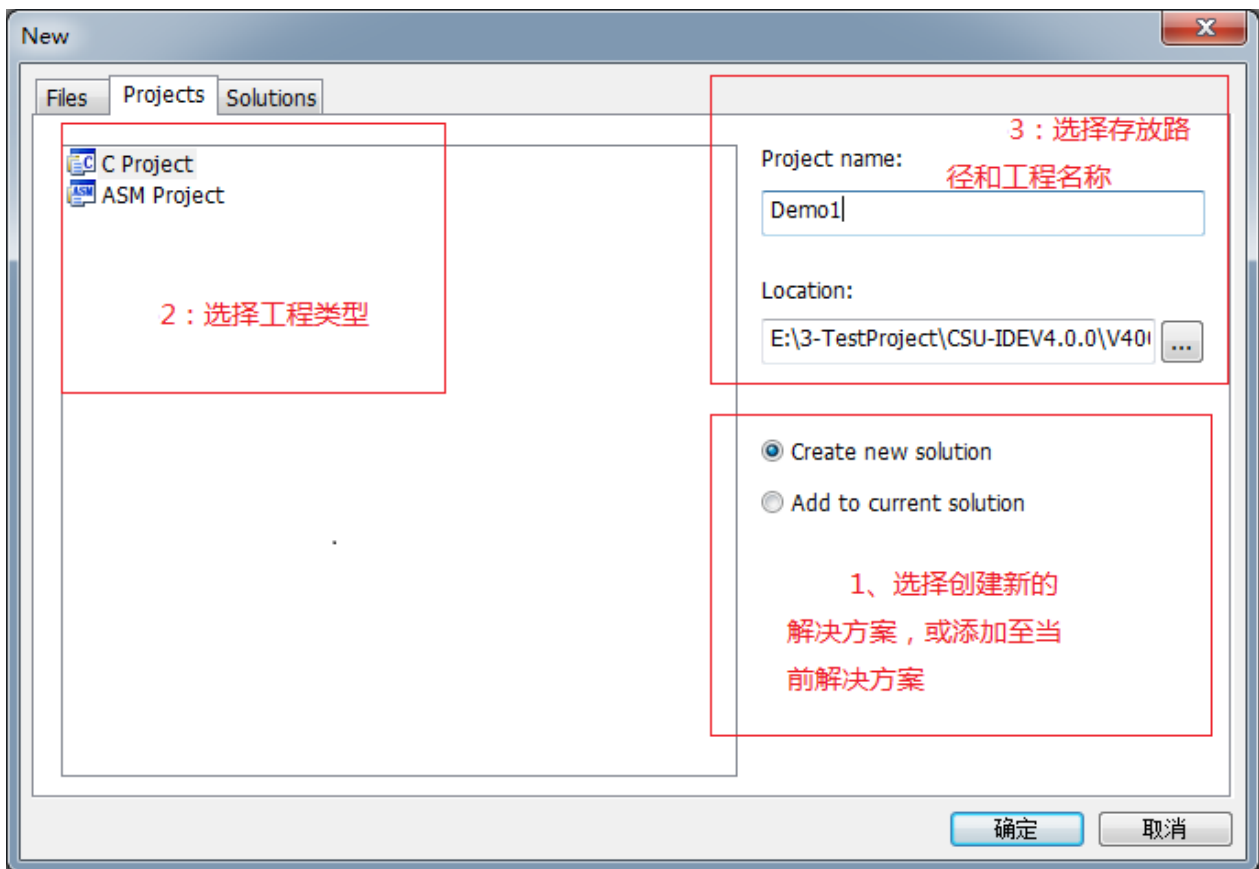
- **Auto Hide:** 自动隐藏。该状态下，不使用时，该窗口会自动隐藏，使用时点击对应标题即可。也可点击右上角 Auto Hide 按钮来实现自动隐藏。
- **Hide:** 将该窗口隐藏，之后只有在菜单栏 View 中才能打开。

## 4 快速入门

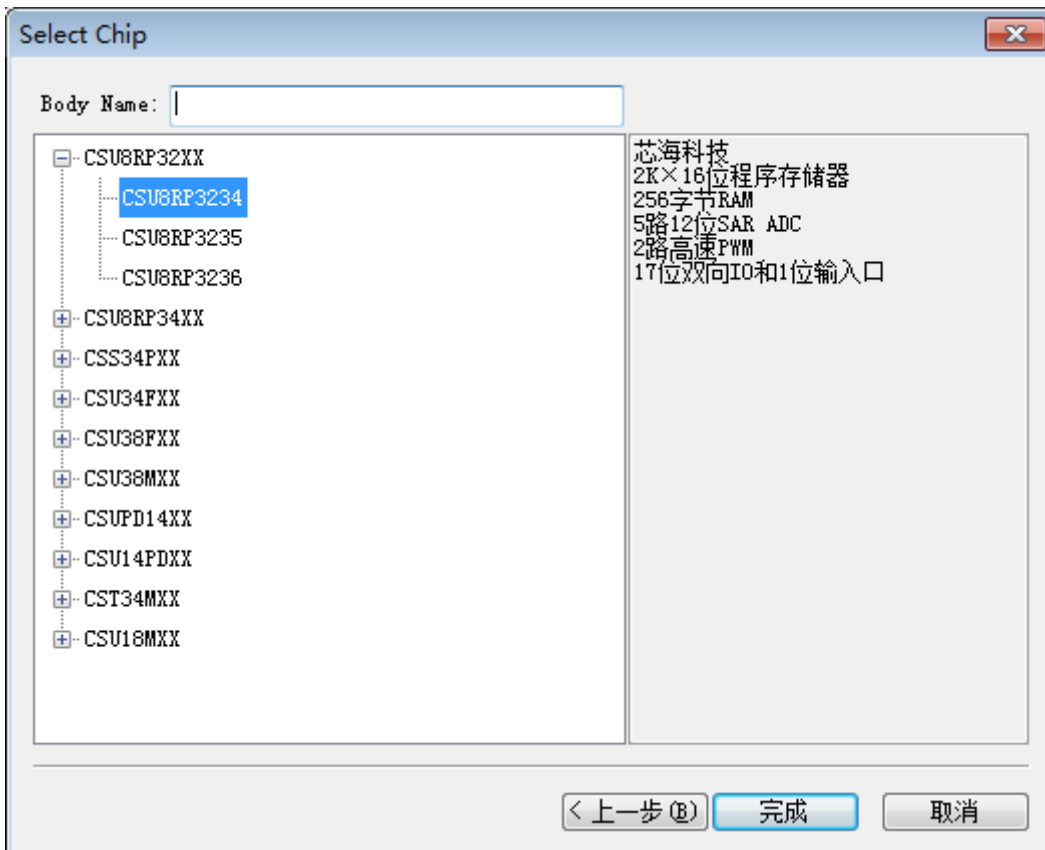
本章我们将用一个简单的工程，介绍主要的操作步骤，比如创建工程、设置工程、编译工程、运行工程以及调试工程。您可以通过查询相关文档了解更多详细功能。

**Step1**、运行 CSU-IDE。

**Step2**、新建工程：选择 File | Open Project，可以开启已经存在的工程；或选择 File | New | Project，根据向导新建一个工程。如下图所示：



执行上述三个步骤，点击确定，进入芯片类型选择界面，如下图所示：

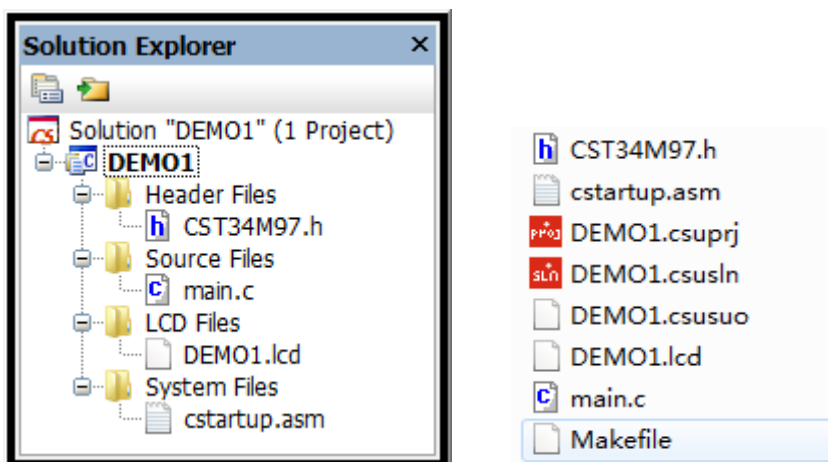


左边区域显示 IDE 支持的芯片类型，右边区域会同步显示选中芯片类型的参数信息。

先选择所需 IC 的类型，再点击完成，一个新的工程创建完毕。

注：可在 Body name 输入框中输入芯片型号，实现快速查找。

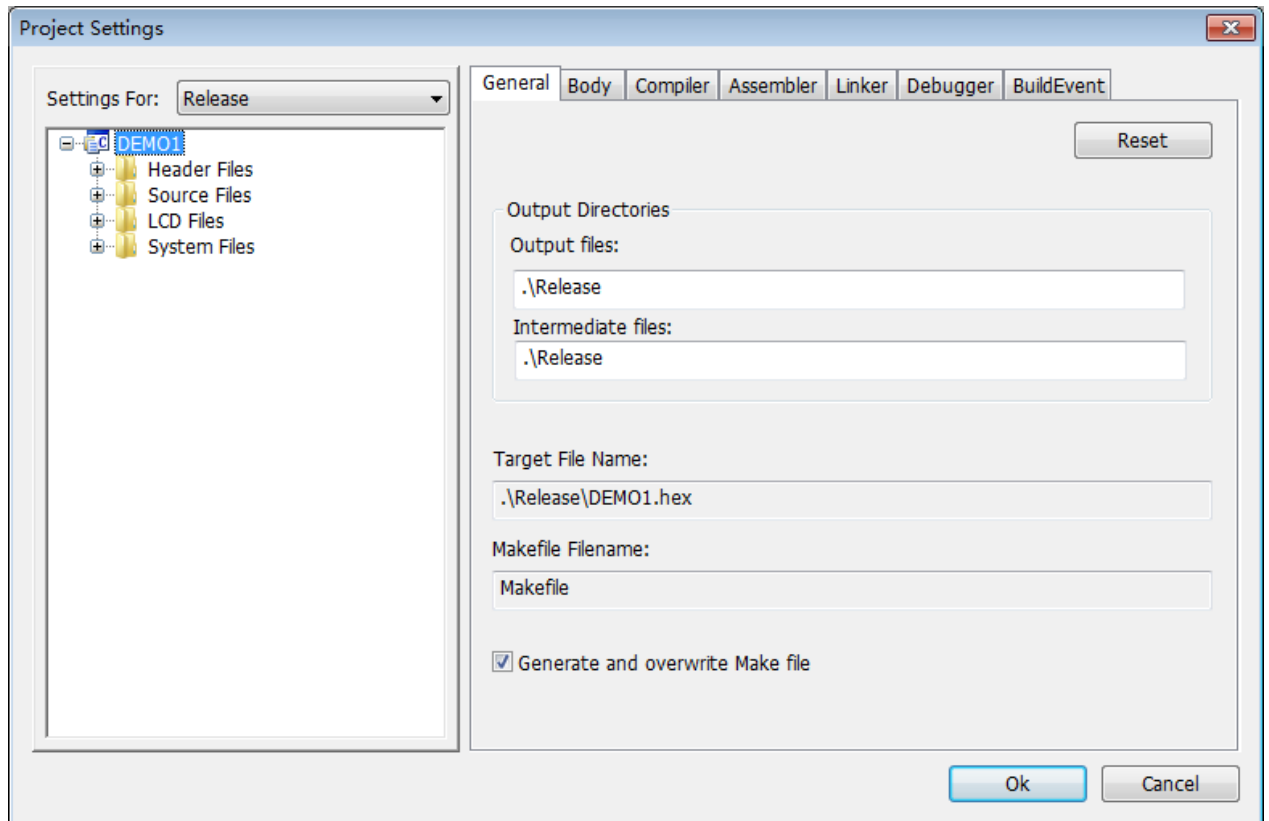
**Step3、编辑文件：**IDE 会自动开启刚新建完成的工程，在 Solution Explorer 窗口默认位于主界面的左边，当前活动工程名称会呈黑体加粗显示（如下面左图所示），双击 Source Files 目录，可看到 main.c 的文件，双击将会在编辑器中打开该文件，您可在编辑器中对文件进行编辑。



在工程路径下可以看到新建工程产生的文件，如上面右图所示：

- \*.csuprjx: 工程文件
- \*.csuslnx: 解决方案文件

**Step4**、修改 Setting: 选择 Project | Settings, 将会弹出工程设置对话框, 您可在不同的 Tab 间进行切换并进行相对应的设置。

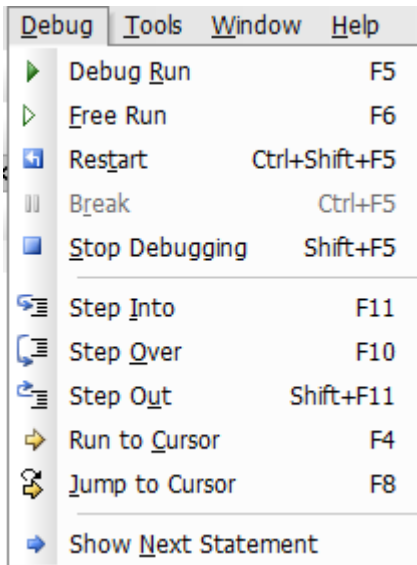


**Step5**、生成 Hex 档: 点击 Build | Rebuild All 编译工程, 如果编译成功, 将会在工程输出路径下 (默认是 Release 文件夹) 的输出\*.hex 档。

- cstartup.obj
- DEMO1.dat
- DEMO1.hex
- DEMO1.map
- DEMO1.sbms
- main.asm
- main.obj
- Script.lik
- sys.lst

**Step6**、下载：点击 Debug | Debug run 下载 code，然后使用 Debug 菜单项中的调试命令开始执行或调试程序。

**Step7**、调试：使用 Debug 菜单项中的调试命令开始执行或调试工程





## 5 工程管理

### 5.1 关于解决方案

#### 1、什么是解决方案？


在 CSU- IDE 中，Solution 被我们称为解决方案。解决方案可理解为用来管理所有工程项目的“管家”。一个解决方案中可以包含多个工程，可以是不同工程类型。使用解决方案，将大大方便开发过程中的相关工程的管理。

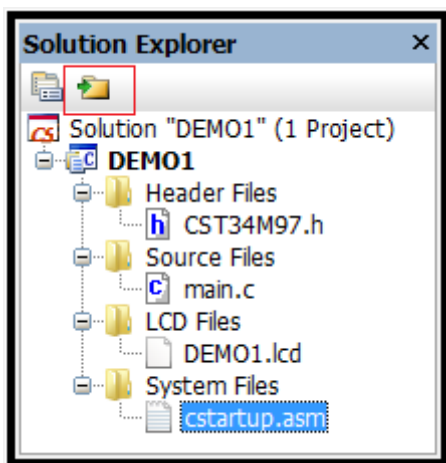
#### 2、解决方案相关文件

新建一个解决方案后，会生成两个文件：\*.csuprjx 和\*.csuslnx。

#### 3、解决方案管理器

解决方案管理器（Solution Explorer）主要用来管理工程以及工程下的文件。您可以创建不同的文件夹，然后将相应的文件添加到对应的文件夹中，这样将大大方便您对文件的查找。但是请注意，解决方案管理器中的文件结构只是一种逻辑关系，和文件的实际存储结构可能没有任何关联。

在解决方案管理器中，提供了一个十分方便而且好用的功能，目录导航，您只需点击 ，可打开所选中的工程或者文件所在文件夹，如下图所示：



特别说明：在 V5.0.X 版本中不能正确打开在 V4.0.X 版本中创建的工程，请用户注意。

#### 4、活动工程（Active Project）

一个解决方案，有且只有一个活动工程。

可通过以下方式设置其它工程为活动工程：

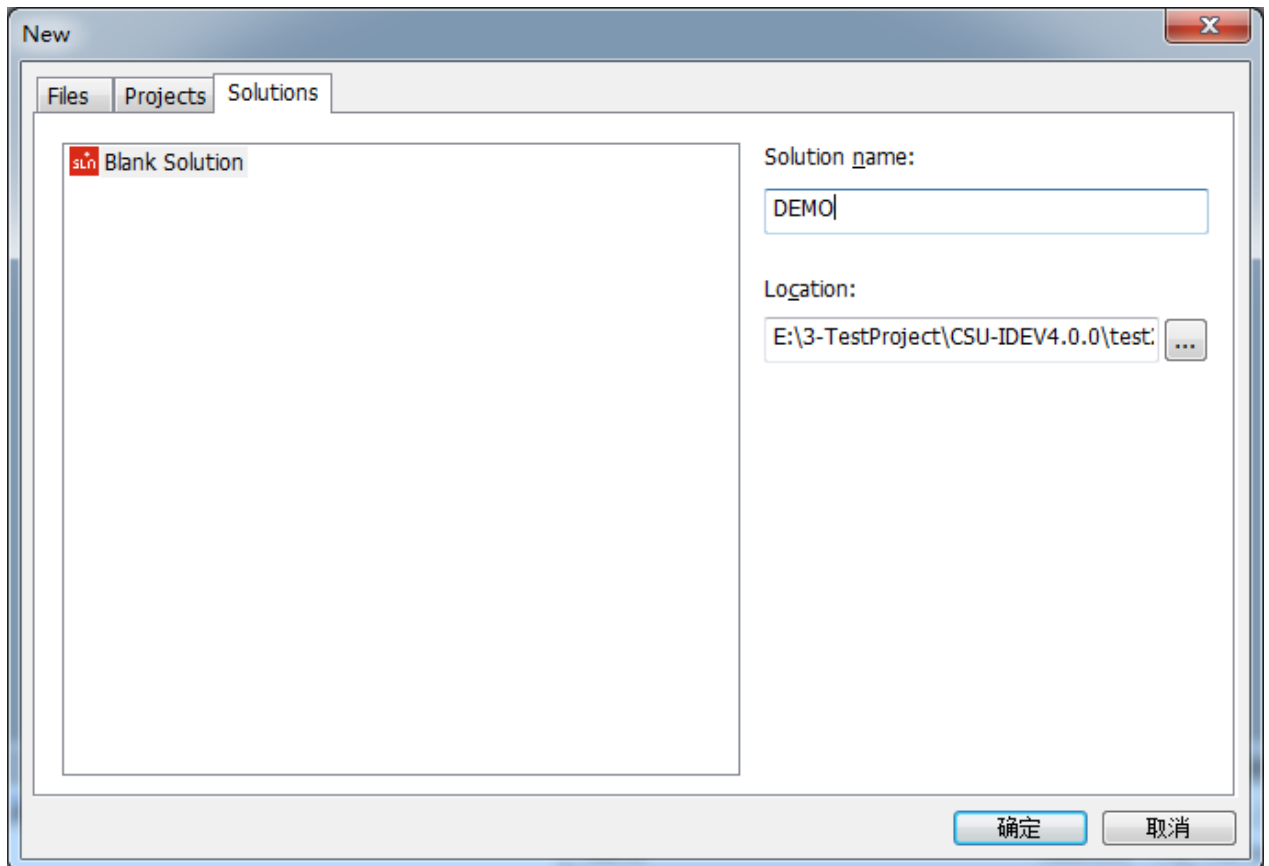
- 1、选择 Project | Set Active Project，将选中工程设置为活动工程。
- 2、在 Solution Explorer 窗口中，选中工程名并选择右键快捷方式 Set as active project。

特别说明：执行 Build 和 Debug 菜单功能，均是对当前活动工程进行操作的。

## 5.2 创建解决方案

### 1、创建一个空的解决方案

选择 File | New | Blank Solution..., 弹出“New”对话框。如下图

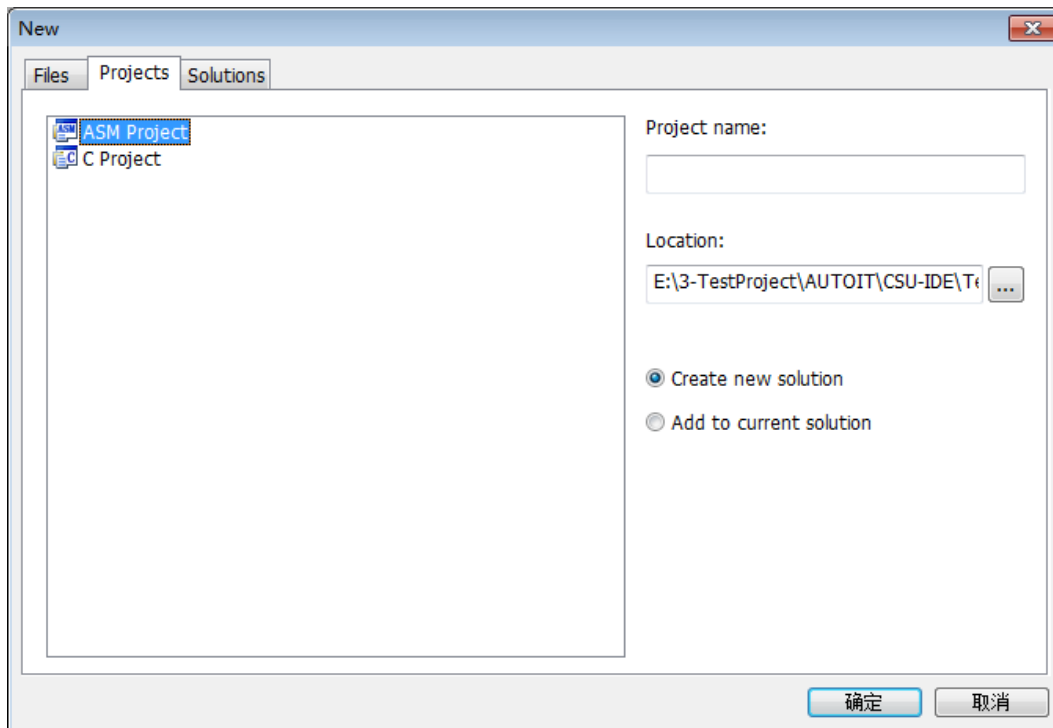


点击确定，将会创建一个空的解决方案。

可再通过菜单 Project | Insert Project into Solution 添加已经存在的工程到解决方案中。

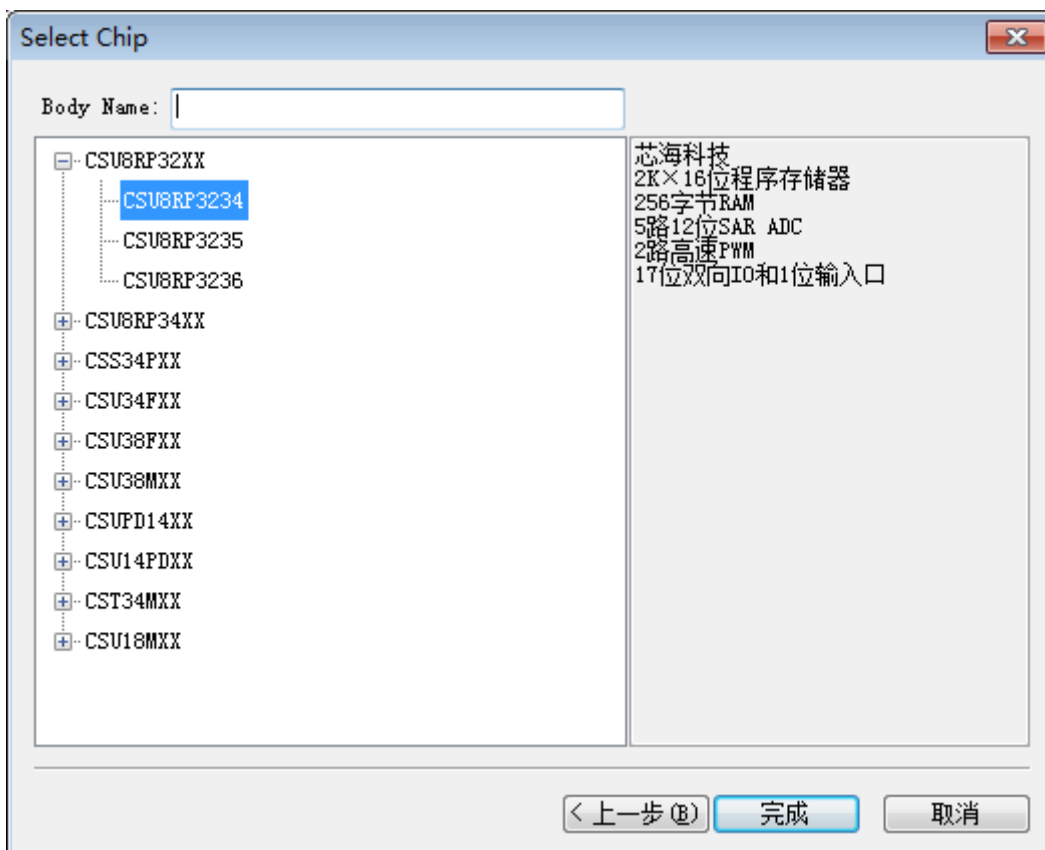
### 2、创建一个解决方案，包含一个同名工程

选择 File | New | Project, 弹出“New”对话框。如下图：



选择 Create new solution，点击确定，将进入 IC 类型选择界面。

注：当前 IDE 已打开一个解决方案时，Add to current solution 控件会显示为有效，选择该控件，将会创建一个新工程，并将新建工程添加至当前的解决方案中。



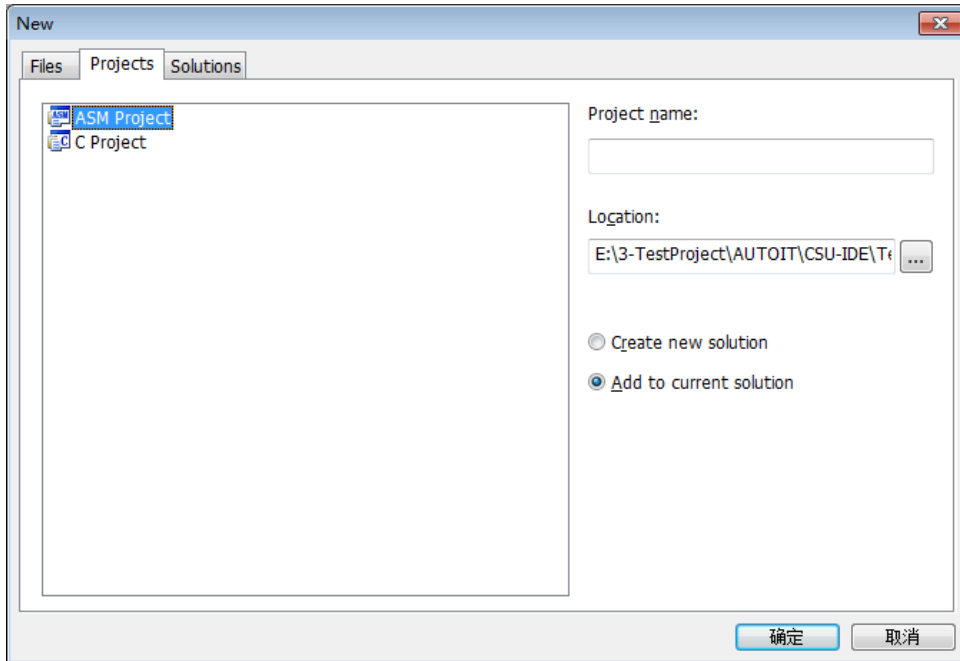
选择您所需的类型，再点击完成，会创建一个新建解决方案，以及一个同名的新工程。

### 5.3 添加工程

在解决方案中，可添加新的工程，也可以添加已经存在的工程。

#### 1、添加新工程

IDE 开启一个解决方案，再点击选择 File | New | Project，会弹出“New”对话框，如下图：

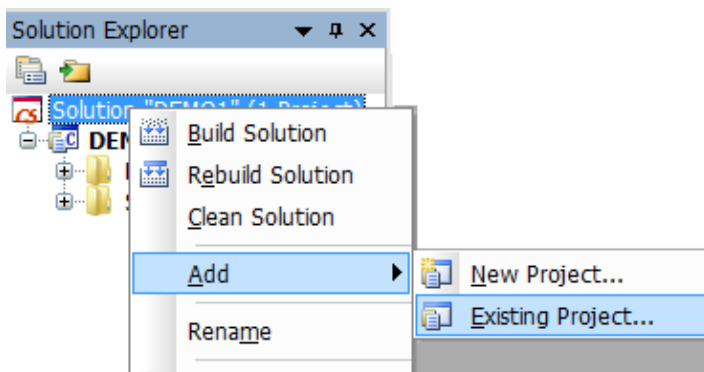


选择 Add to current solution 控件，点击确定，将会创建一个新工程，并将新建工程添加至当前的解决方案中。

#### 2、添加已存在的工程

点击 Project | Insert Project into Solution，弹出“Open”对话框，选择后缀为“csuprjx”的工程文件，点击“OK”，工程即被加入到当前解决方案中。

也可以直接在解决方案管理器窗口中，选中解决方案，单击右键，Add | Existing Project。

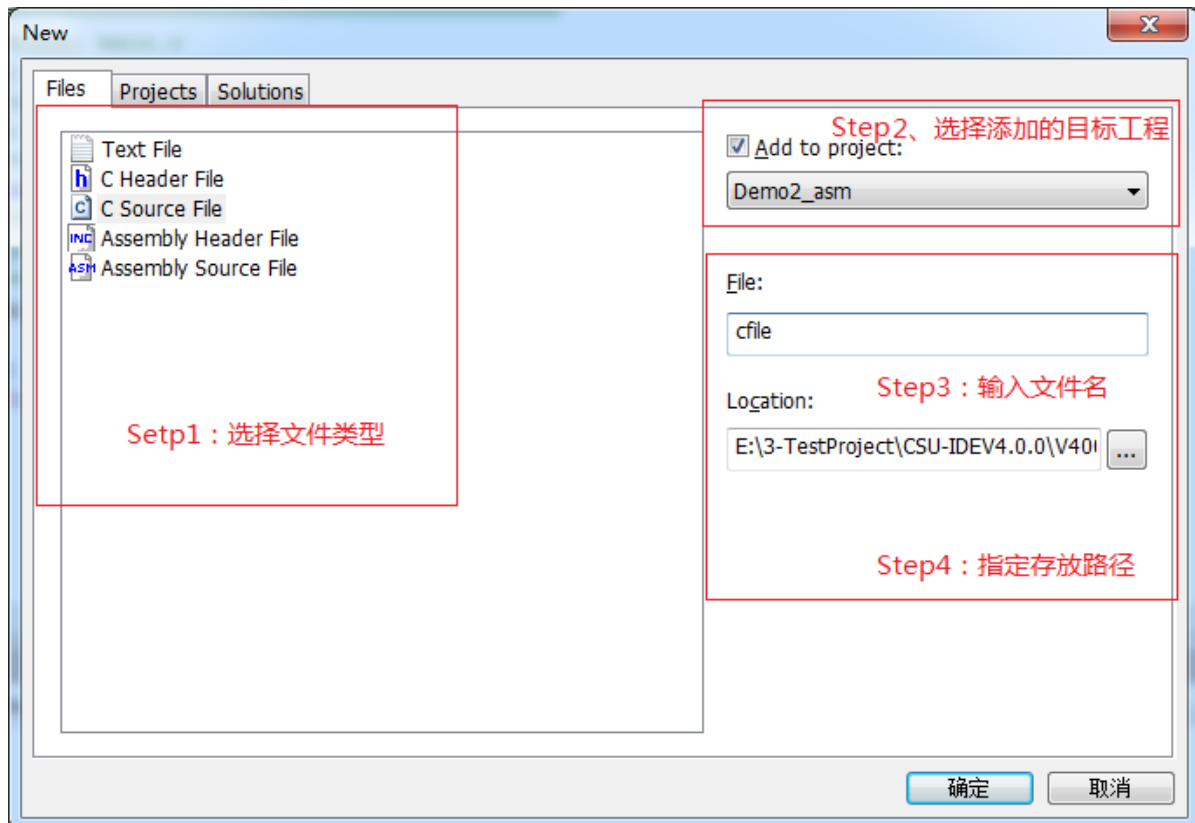


## 5.4 添加文件

在工程中，可添加新文件和已经存在的文件。

### 1、创建新文件

选择 File | New | File，弹出“New”对话框，按照下面步骤操作，将会创建一个新文件至指定的目标工程。

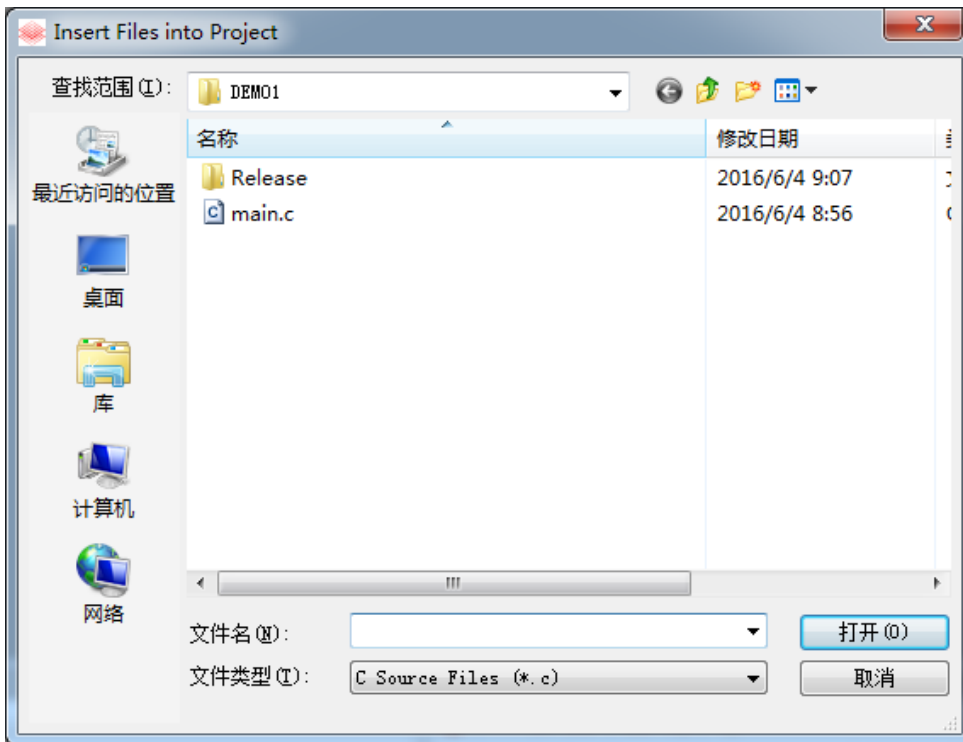


其它方法：

➤ 选择 Project | Add To Project | New File，弹出“New”对话框，其余操作步骤与上面一致。

### 2、添加已存在的文件

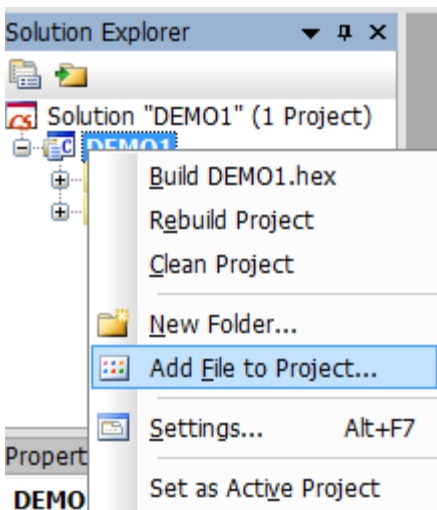
选择 Project | Add To Project | Exist Files，弹出“Insert Files into Project”对话框。



选择要添加的文件，点击【Open】，所选择文件将根据其后缀自动添加到当前工程的对应文件夹中。

其它方法：

- Solution Explorer 窗口的右键快捷方式 Add File to Project。



## 5.5 添加文件夹

在解决方案管理器中可以看到多个文件夹，文件夹主要用于文件管理，把某一个类型或其相关类型的多个文件放到同一个文件夹中，方便文件查找。IDE 对不同的工程类型默认提供了不同的文件夹，在新建工程后，可在解决方案管理器窗口中看到工程下的默认文件夹。您也可以选中工程，点击右键，通过“New Folder”命令创建文件夹，或选中某个文

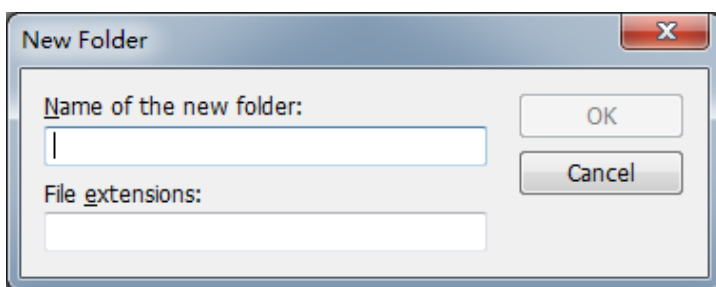
文件夹，点击右键，通过“New Folder”在已有文件夹中创建子文件夹。

- “Source Files”：源文件夹。添加到该目录通常都是一些源文件，如\*.asm，\*.c文件等。
- “Header Files”：头文件夹。用于添加源文件所包含的头文件。

添加文件夹至工程：

Setp1: 选择 Project | Add To Project | New Folder，弹出“New Folder”对话框。

Solution Explorer 窗口也提供了 New Folder 右键快捷方式。



Setp2: 编辑文件名和文件后缀类型，点击【OK】即可。

## 5.6 解决方案管理器（Solution Explorer）

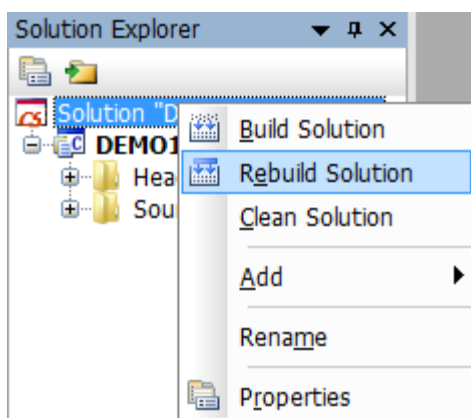
“Solution Explorer”即解决方案管理器，主要用来管理工程以及工程下的文件。您可以创建不同的文件夹，然后将相应的文件添加到对应的文件夹中，这样将大大方便您对文件的查找。

请注意：解决方案管理器中的文件结构只是一种逻辑关系；在解决方案管理器中添加、修改、移除、重命名等操作，不会对文件的实际存储结构产生影响。

右键单击解决方案名称或工程名称、文件夹名、文件，均会弹出相应的快捷菜单，方便用户使用。

### 1、解决方案

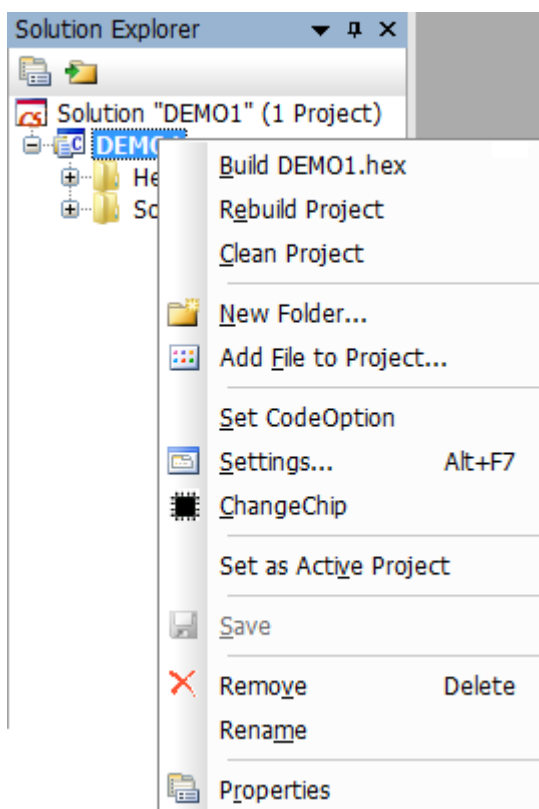
选中解决方案的右键快捷菜单，如下图所示：



<b>Build Solution</b>	构建解决方案，会对解决方案中的所有工程进行构建。
<b>Rebuild Solution</b>	重新构建解决方案中所有的工程。
<b>Clean Solution</b>	清除解决方案中所有工程的输出文件
<b>Add</b>	在解决方案中，添加新工程或已经存在的工程。
<b>Rename</b>	为解决方案重新命名。
<b>Properties</b>	切换至属性窗口，并显示解决方案的各种属性。

## 2、工程

选中工程的右键快捷菜单，如下图所示：



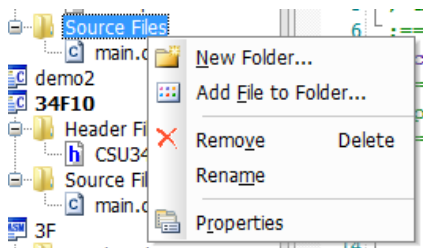
<b>Build *.hex</b>	增量编译，只对修改的源文件重新编译，生成 HEX 档。
<b>Rebuild Project</b>	重新构建该工程，生成 HEX 档。
<b>Clean Project</b>	清除该的输出文件。
<b>New Folder...</b>	在该工程中，新建一个文件夹。
<b>Add File to Project...</b>	在该工程中，添加 1 个或多个文件。



<b>Set CodeOption</b>	设置该工程的代码选项。
<b>Change Chip</b>	更改该工程的芯片型号。
<b>Settings...</b>	打开该工程的工程设置窗口。
<b>Set as Active Project</b>	将该工程设置为活动工程。
<b>Save</b>	保存对该工程的修改。
<b>Rename</b>	为该工程重新命名。
<b>Properties</b>	切换至属性窗口，并显示该工程的各种属性。

### 3、文件夹

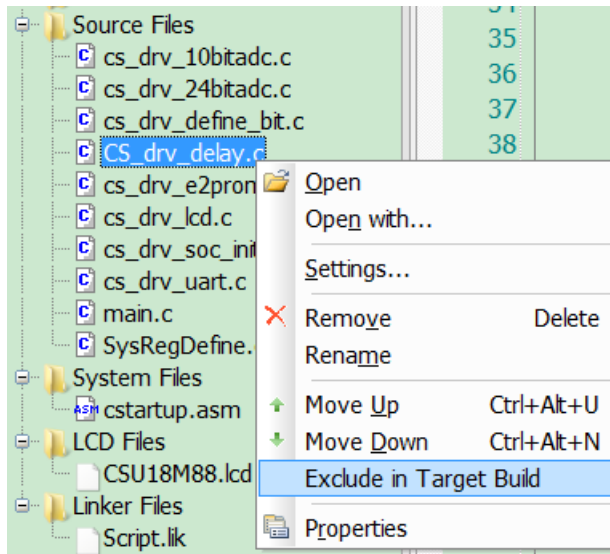
选中文件夹的右键快捷菜单，如下图所示：



<b>New Folder...</b>	在该工程中，新建一个文件夹。
<b>Add File to Project...</b>	在该工程中，添加 1 个或多个文件。
<b>Remove</b>	移除该文件夹和文件。
<b>Rename</b>	为该文件夹重新命名。
<b>Properties</b>	切换至属性窗口，并显示该文件夹的各种属性。

### 4、文件

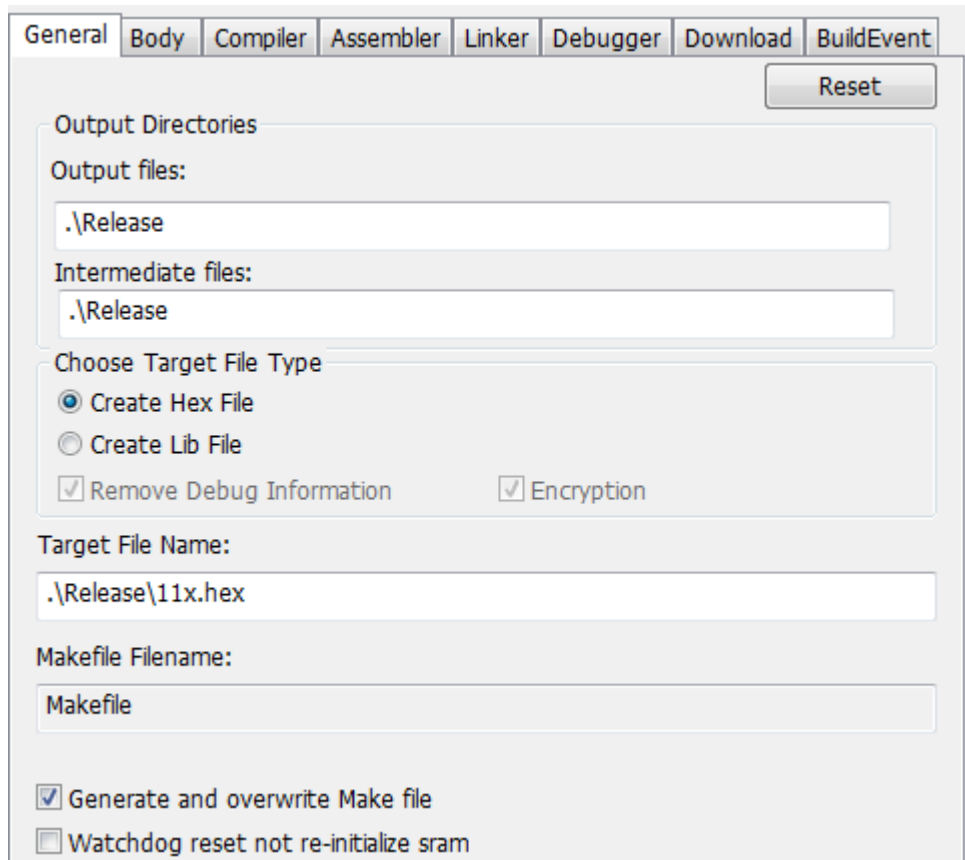
选中文件的右键快捷菜单，如下图所示：



<b>Open</b>	会在编辑区打开该文件
<b>Open with...</b>	会弹出打开方式对话框，再选择想用来打开此文件的程序即可
<b>Settings...</b>	打开该文件设置对话框
<b>Remove</b>	移除该文件
<b>Rename</b>	为该文件重新命名
<b>Move Up</b>	向上移动
<b>Move Down</b>	向下移动
<b>Exclude/Include in Target Build</b>	不包含/包含在构建中
<b>Build</b>	
<b>Properties</b>	切换至属性窗口，并显示该文件的各种属性

## 5.7 工程设置（Project Settings）

此功能是针对单个 Project 选项的设置, 包括 General、Body、Compiler、Assembler、Linker、Debugger、Download、BuildEvent 等页面。



➤ General

该用来设置和查看输出目录、中间文档目录、目标文档名、Makefile 名。

**Output Files:** 指定工程的 Hex 输出目录，工程编译时会将 Hex 输出在此目录。执行 Clean 操作，会清除该输出目录的文件。

**Intermediate Files:** 指定工程的中间文档的输出目录，工程编译时会将中间输出在此目录。执行 Clean 操作，会清除该输出目录的所有输出文件。

**Choose Target File Type:** 可选择生成 hex 文件或 lib 文件。只有选择生成 hex 文件时，才支持调试。

**Remove Debug Information:** 生成的 lib 是否移除调试信息。生成的 obj 文件，有可能会带调试信息，勾选此项将会使生成的 lib 文件将不再带调试信息。

注：obj 是否包含调试信息，可在 setting->Compiler/Assembler 页面中通过 Generate Debug info File 选项控制

**Encryption:** 生成的 lib 是否加密。已加密后的文件，将不能再导入。

**Target File Name:** 指定 HEX 文件名，可依据需求修改输出路径和文件名。

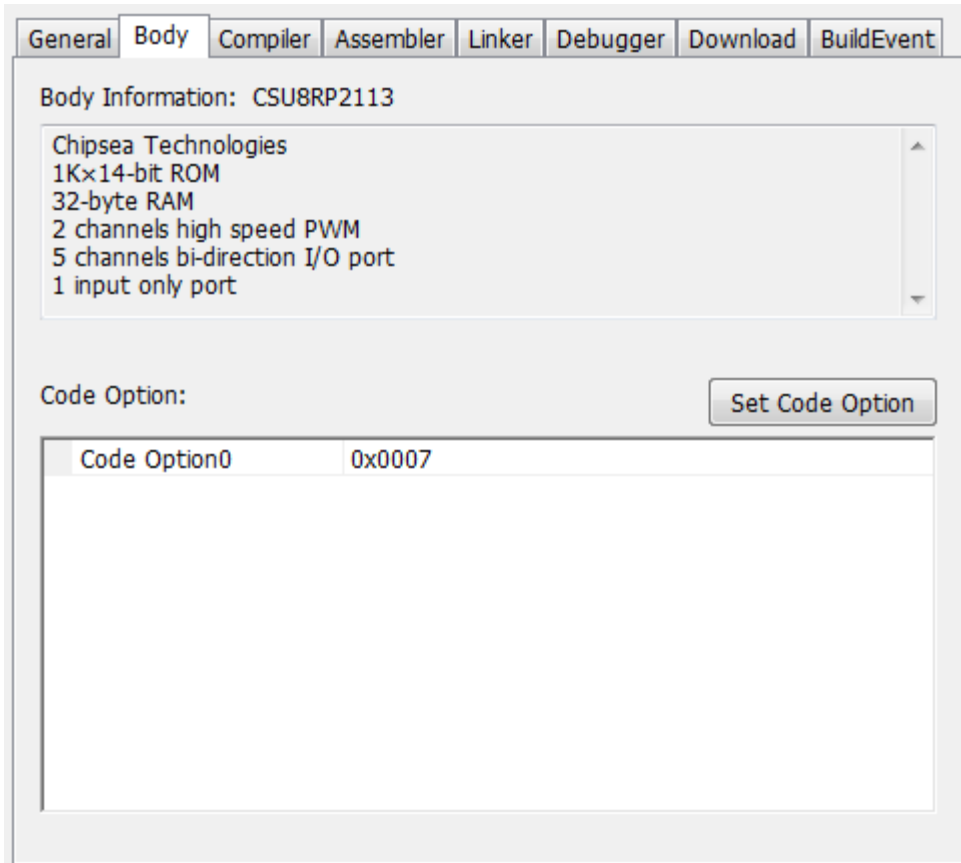
**Makefile FileName:** 指定 MakeFile 文件名，暂不提供编辑功能。

**Generate and overwrite Make file:** 是否在编译时重新生成和替换已有的 Make file。

Watchdog reset not re-initialize sram: 是否在复位时对 sram 进行初始化，勾选了该选项则在复位时不对 sram 进行，不勾选该选项则在复位时仍进行初始化。

➤ Body

在该页面显示 IC 类型，可设置 code option 的值。



Body Information: 显示 IC 类型和参数信息;

Code Option: 在列表中显示 Code option 值，该值可通过点击 Set Code Option 进行修改，但暂不支持在列表中直接编辑。

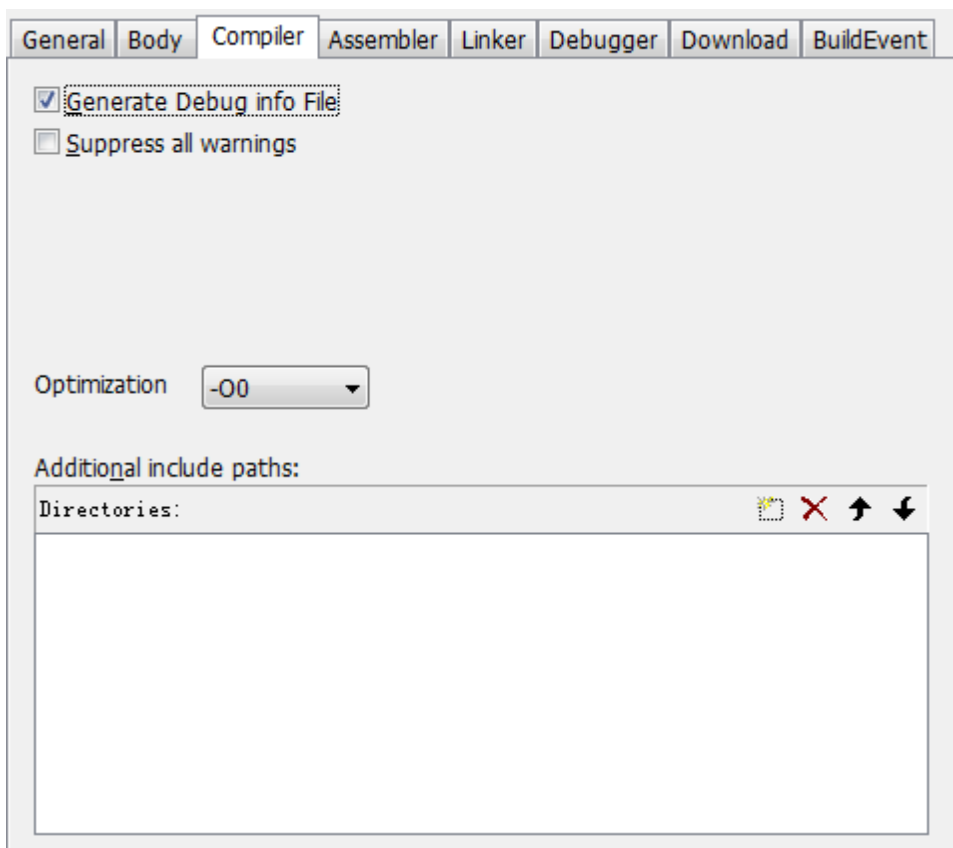
Set Code Option: 提供 Code Option 设置对话框，修改后的值会存放在 Code Option 列表中。

Clock: 只有部分 IC 支持该选项，请参考各 IC 型号的使用手册。

注: Set Code Option 对话框中参数的详细说明，请参考各 IC 型号的使用手册。

➤ Compiler

在该页面可设置编译器的相关参数及行为。



**Generate Debug Info File:** 是否在指定的输出目录中会生成调试信息（后缀为 c 的源文件对应的调试信息）。

**Suppress all warnings:** 在编译时是否显示警告信息。勾选该选项，在编译时在 Output 窗口不会显示警告信息。

**Additional include paths:** 添加 include 文件所搜寻的目录。

操作步骤如下：

Step1: 点击 ，在 Additional include paths 列表中弹出一个编辑区

Step2: 点击 ，弹出路径选择框

Step3: 选择需要添加的路径，就完成了，如下图所示



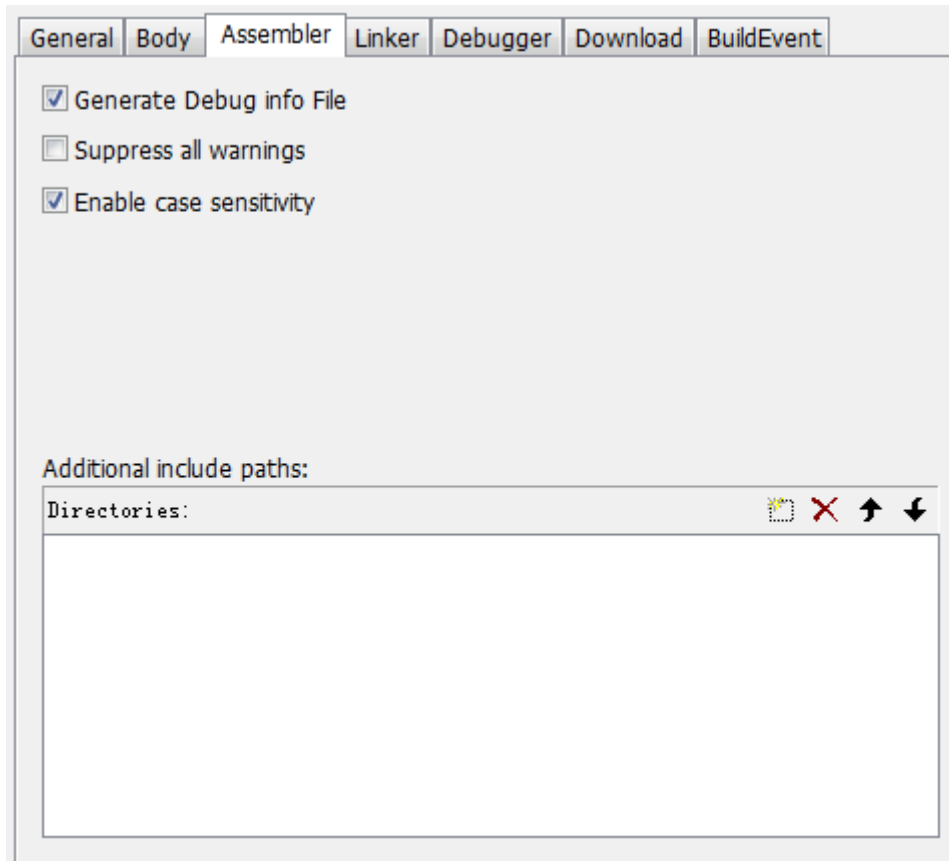
Step4: 在源代码中，就可以使用 include 命令来关联该目录中的文件了。

**Optimization:** 可选择-O0、-O1、-O2、-Oz、-Os 五种优化级别，暂时支持-O0 优化级别。

注：C 工程才有该 Compiler 页面。

#### ➤ Assembler

在该页面可设置汇编器的相关参数及行为。




**Generate Debug Info File:** 是否在指定的输出目录中会生成调试信息（后缀为 asm 的源文件对应的调试信息）。

**Suppress all warnings:** 在编译时是否显示警告信息。勾选该选项，在编译时在 Output 窗口不会显示警告信息。

**Enable case sensitivity:** 是否区分大小写。勾选该选项，对汇编工程中自定义的标号等符号的区分大小写（关键字仍不区分大小写）；不勾选该选择，则不区分大小写。

**Additional include paths:** 添加 include 文件所搜寻的目录。

操作步骤如下：

Step1: 点击 ，在 Additional include paths 列表中弹出一个编辑区

Step2: 点击 ，弹出路径选择框

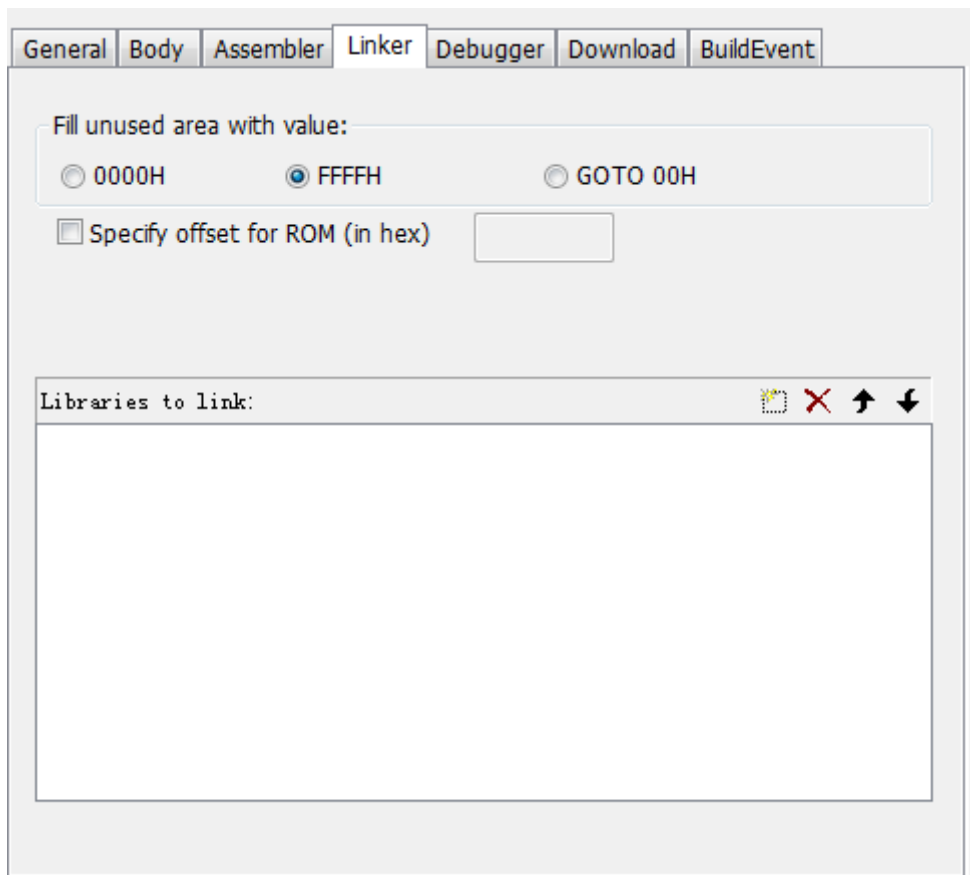
Step3: 选择需要添加的路径，就完成了，如下图所示



Step4: 在源代码中，就可以使用 include 命令来关联该目录中的文件了。

## ➤ Linker

在该页面可设置链接器的相关参数及行为。




**Fill unused area with value:** 选择未使用区域的默认填充值；会填充至生成的 HEX 档。（注：14bit IC 选择 FFFFH 时填充的是 3FFFH）

**Specify offset for ROM(in hex):** 指定代码在 HEX 中的偏移值，单位是 words。

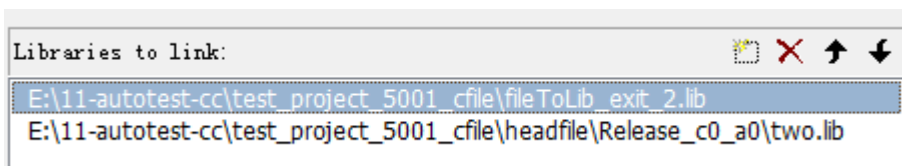
**Libraries to link:** 添加 lib 文件路径。

操作步骤如下：

Step1: 点击 ，在 Additional include paths 列表中弹出一个编辑区

Step2: 点击 ，弹出路径选择框

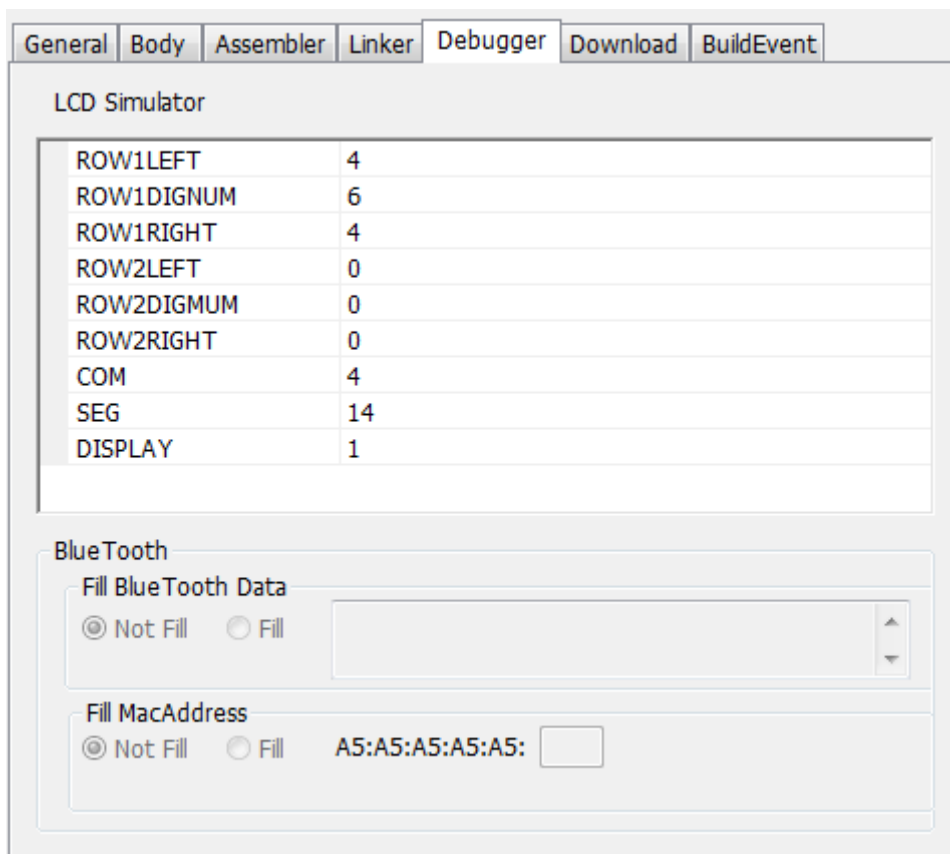
Step3: 选择需要添加的 lib 文件就完成了，如下图所示



**需注意：**不同指令长度的 IC(14bit /16bit )，需使用对应指令长度的 lib 文件。

## ➤ Debugger

在该页面可设置调试器的相关参数及行为。



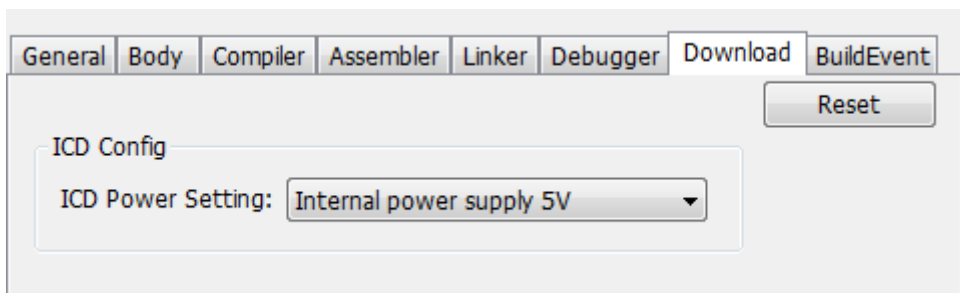
本页面是对 LCD Simulator 和蓝牙进行设置。

LCD Simulator 请详见第 8 章 LCD 模拟器。

蓝牙请详见第 9 章蓝牙。

➤ Download

在该页面可设置下载相关参数及行为。



Only Download Setting 功能与工具栏中的 Download hex file 配合使用。



Download hex file 功能：只下载，不进入调试状态。

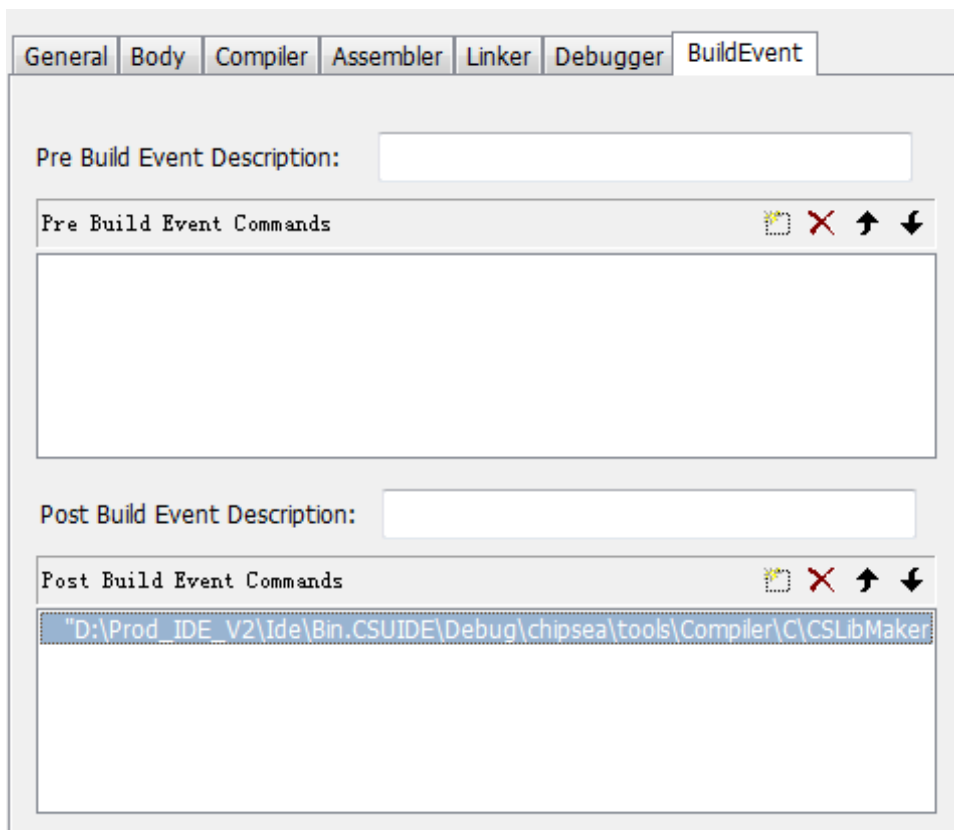


- 勾选 Download after run 后，点击 Download hex file 后会先下载 hex 文件再自动运行。
- 勾选 Download after Pause 后，点击 Download hex file 后会先下载 hex 文件不会自动运行。该功能是否有效与具体型号有关。

ICD Power Setting 功能：可选择不同的供电方式，只对 ICD 有效。

#### ➤ Build Event

在该页面可添加一个或多个脚本命令。



Pre Build Event Description: 填写描述信息。

Pre Build Event Commands: 可添加一个或多个命令，这些命令会在编译之前执行。

Post Build Event Description: 填写描述信息。

Post Build Event Commands: 可添加一个或多个命令，这些命令会在编译结束后执行。

以制作 lib 为例：

在 Post Build Event Commands 中添加以下命令

```
"D:\chipsea\tools\Compiler\C\CSLibMaker.exe" -lib "C:\LibTest\Release\Added.lib" -o
"\LibTest\Release\Added.obj" -o "\LibTest\Release\Nop.obj"
```

说明：在编译过程中会生成 Added.obj 和 Nop.obj 文件，在编译结束执行该命令，就可以生成 Added.lib。

## 5.8 编译解决方案或工程（Build）

IDE 可编译单个工程，也可编译一个解决方案下的所有工程。编译成功，将会在工程输出路径下的输出\*.hex 档

### 1、构建活动工程

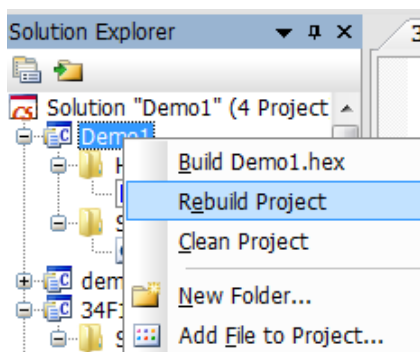
选择 **Build | Rebuild All**，将会对当前解决方案的活动工程进行编译。会先执行 **clean** 操作，再对工程所有文件重新编译，生成 HEX 档。

选择 **Build | Build**，会对修改过的源文件进行重新编译，重新生成 HEX 档。

建议：工程文件较小时，修改源文件后，尽量使用 **Rebuild All**，重新构建工程。

### 2、构建选中的工程

需要构建非活动的工程，需在 **Solution Explorer** 窗口中，选中一个工程名，选择右键快捷菜单 **Rebuild Project**。



注：V5.0.3 以及更高版本，会将 **CodeOption** 的值写入 HEX 文件中。

### 3、编译所有工程

在 **Solution Explorer** 窗口中，选中解决方案，选择右键快捷菜单 **Rebuild Solution**。

### 4、编译单个文件

点击 **Build | Compile**，可以编译编辑器窗口中当前活动文档，也可以在解决方案管理器中选中某个文件，单击右键，执行 **Compile** 命令。

**Compile** 只允许编译工程中的单个源文件，IDE 会根据是否是当前活动工程的文件以及文件的扩展名自动判断当前文件是否可以 **Compile** 动作。

注：编译完毕，可在 **Output** 窗口中查看结果。

**特别说明:修改头文件后,必需选择 Rebuild all 才会对头文件重新编译。**

## 5.9 输出窗口 (Output Window)

编译过程中工具链的输出信息都会在输出窗口中显示，可以通过输出窗口查看编译的状态及产生的信息，编译过程中产生的错误和警告也将在该窗口显示。编译结束后，查看统计结果，如果错误为 error 0，则表示编译成功。

在输出窗口中，按住 Ctrl 键，同时滚动鼠标，可以放大或缩小其显示字体。

```

Output
Show Output: Build
-----
"C:\Program Files (x86)\chipsea\CSU-IDE V5.0.0.2\chipsea\tools\Compiler\C\csasm.exe" -filetype=obj -arch=csc -B
//
"C:\Program Files (x86)\chipsea\CSU-IDE V5.0.0.2\chipsea\tools\Compiler\C\cslinker.exe" -s "E:\3-TestProject\de
CSLinker - Version 1.0.0.0
link library: C:\Program Files (x86)\chipsea\CSU-IDE V5.0.0.2\chipsea\tools\Compiler\C\libcsc.lib
//
Bank Statistics:
      BANK      Used Size      Total Size
      0          0 bytes      128 bytes
      1         141 bytes      256 bytes
      2           0 bytes      104 bytes
//
Memory Statistics:
Code Offset = 0 [0x0000] words
Total ROM   = 4096 [0x1000] words
Total RAM   = 488 [0x01E8] bytes
ROM Used    = 119 [0x0077] words (%2.9)
RAM Used    = 141 [0x008D] bytes (%28.9)
Free ROM    = 3977 [0x0F89] words (%97.1)
Free RAM    = 347 [0x015B] bytes (%71.1)
//
ChipName    : CSU34F10
Checksum    : 0x854D
CodeOption  : CodeOption0:0x0540;
//
34f10.hex - 0 error(s), 0 warning(s)
    
```

注：存在 Warning 信息不影响调试，但说明程序存在问题，建议用户完善代码。

## 5.10 .lik 文件

新建工程，会自动生成 Script.lik 文件。在 Script.lik 文件中可指定段排列的地址，以及段排列的先后顺序。

```

Solution Explorer
Solution "xxa" (6 Project)
  xxa
    3111
      Header Files
        csu8rf3111.h
      Source Files
        main.c
        SysRegDefine.c
      System Files
      Linker Files
        Script.lik
    
```

```

Script.lik
1 ;example begin
2 ;locate: CSCC_SECTION_1 at 0x80
3 ;locate: CSCC_SECTION_2 after CSCC_INIT_1
4 ;example end
    
```

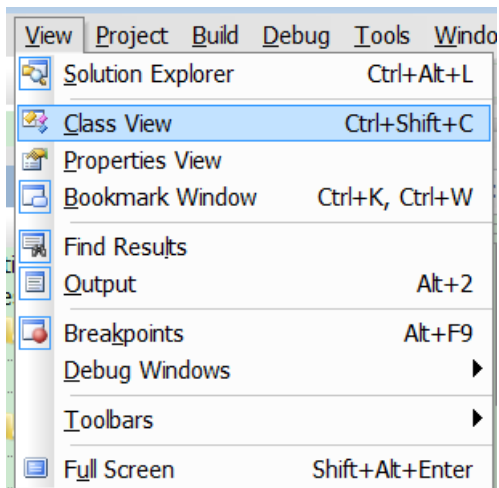
**at:** 指定段的地址；

**after:** 指定段的先后顺序。

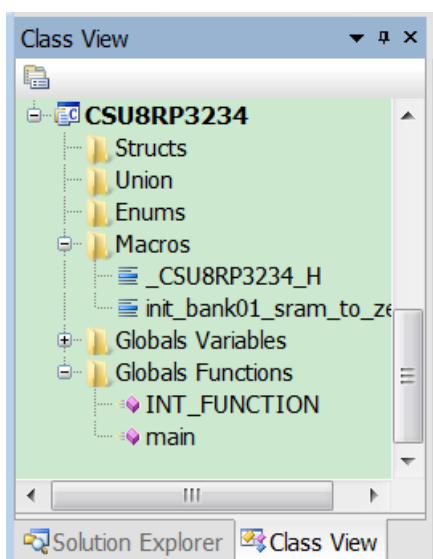
## 5.11 Class View 窗口

源代码导航窗口是针对用户定义的全局变量、函数等在 **Class View** 视图中显示出来，方便用户查看并进行定位，在变量、函数使用的地方右键选择跳转到声明、定义处即可跳转到声明、定义的地方。暂不包含 **SysRegDefine.c** 中定义的变量。

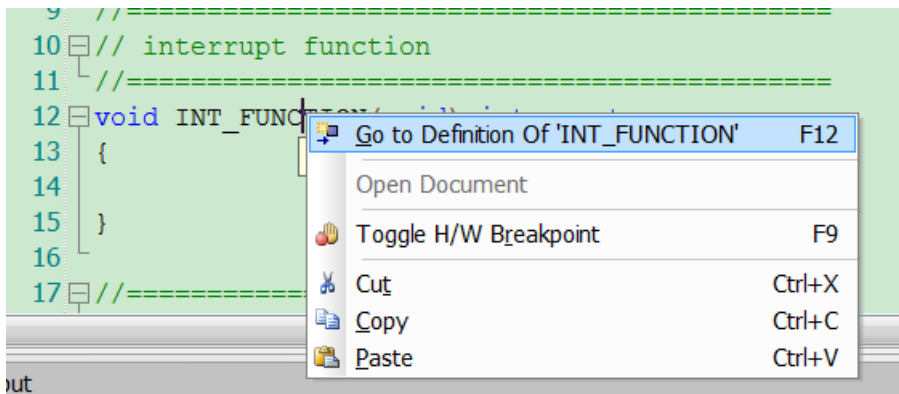
在 IDE 中的菜单



点击【View】->【Class View】，弹出 Class View 窗口



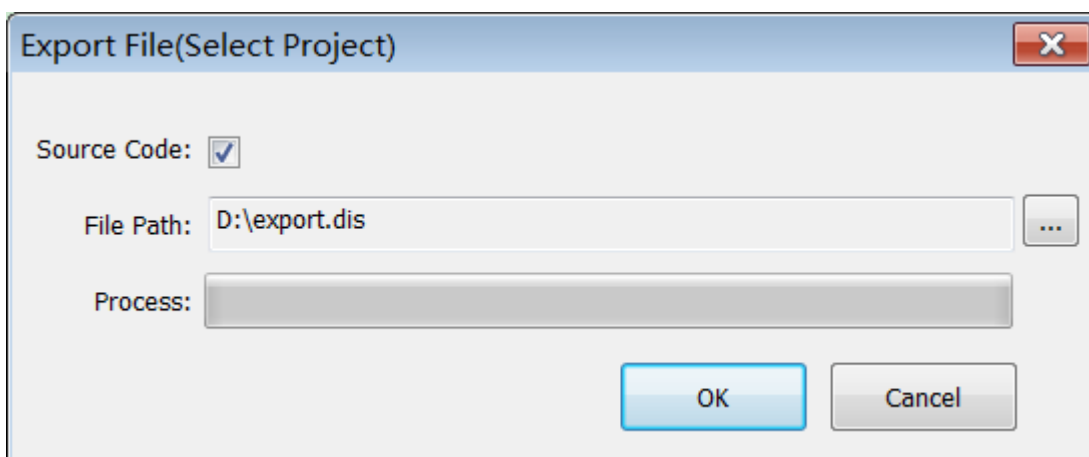
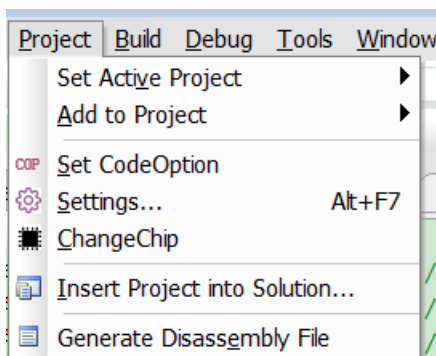
在编辑区中，右键菜单“Go to Defintion of ‘...’”，将跳转至定义行。



## 5.12 导出反汇编文件

可将编译通过的工程导出所有的反汇编文件。

点击 Project-> Generate Disassembly File，弹出 Export File 对话框。



Source Code: 勾选，导出的\*.dis 文件中将包含源代码

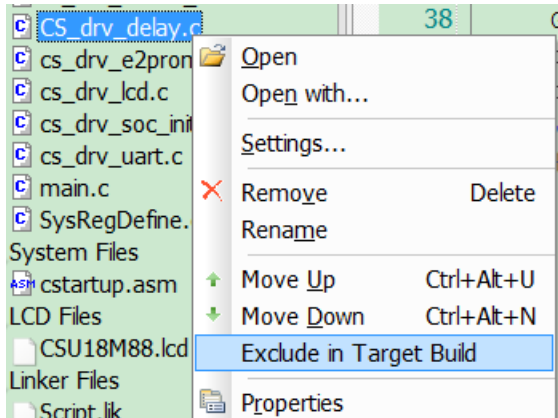
File Path: 点击  选择存储路径。


点击 OK，完成导出功能。

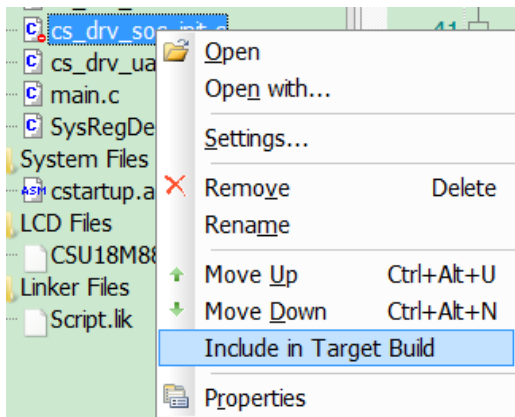
Select Project: 导出的反汇编文件，是依据 Solution Explorer 中选中工程。

## 5.13 Include/Exclude in Target Build

在 Solution Explorer 窗口，选中文件时的右键快捷菜单中包含 Exclude in Target Build:



已设置为 Exclude 的文件，文件图标会多一个红色的标识 ，右键菜单为 Include in Target Build:



执行 Build/Rebuild 时，不会编译已设置为 Exclude 的文件。

## 6 文本编辑

### 6.1 编辑区窗口

IDE 采用多文档视图的架构，您可同时在编辑器窗口中打开任意多的文档。按住 **Ctrl**，滚动滑鼠，可放大或缩小其字体，按“**Ctrl+F6**”或 **Tab** 键可以在各文档之间进行切换。

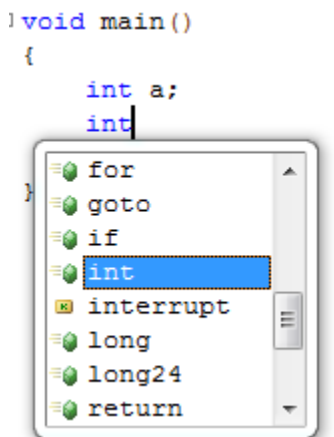
编辑区窗口主要功能是用来编辑代码，IDE 还提供了一些辅助编辑的功能。

- 关键字着色
- 关键字的智能感应和自动完成功能
- 代码段的缩放
- 更改编辑区的文字大小

注：其它功能请参考菜单和工具栏的详细介绍。

#### 6.1.1 关键字

在编辑区，输入关键字，会智能感应和自动完成功能，如下图所示：



着色表如下：

项目	颜色
关键字	蓝色
字符串	褐色
数字	紫色
注释	绿色
其它	黑色

## 6.1.2 更改文字大小

可依照个人习惯，更改编辑区的文字大小。

光标定位在编辑区，按住 CTRL 键，并同时滚动鼠标滑轮，即可放大或缩小编辑区的文字了。

## 6.1.3 文件布局

点击 View | Full Screen，可以进行全屏显示模式。在该模式下，所有工具栏和状态栏会自动隐藏。按 Esc 键或者点击“Close Full Screen”可退出全屏模式。

点击 Window | New Window，将在编辑器中创建一个与当前活动文件完全相同的视图，点击多次，则创建多个，并且所有这些文件，将同步改变。

点击 Window | Rearrange Window，将重置整个 IDE 界面为默认布局。

## 6.1.4 文件保存

在编辑器中，当文件的标题位置有“\*”显示，表示文件被改动过且未保存。点击“Save”可保存文件，保存后“\*”将自动消失。如果文件未保存状态下直接关闭，将会提示您是否保存。

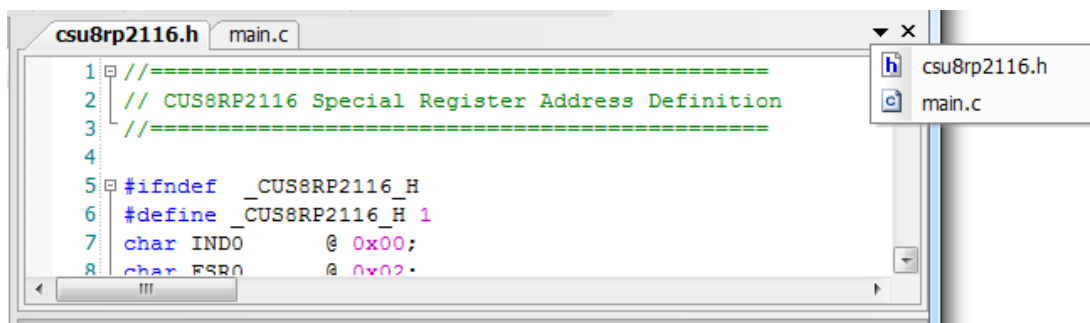
点击 File | Save As，可保存文件为其它文件名；

点击 File | Save All，保存所有打开文件。

## 6.1.5 编辑区窗口功能

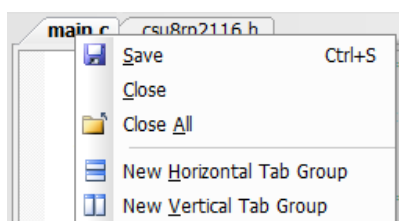
### 1、切换当前编辑文件

在编辑区窗口右上角，提供一个快速切换当前编辑文件的功能，点击  即可显示当前文件名，如下所示：



### 5、标题右键快捷菜单

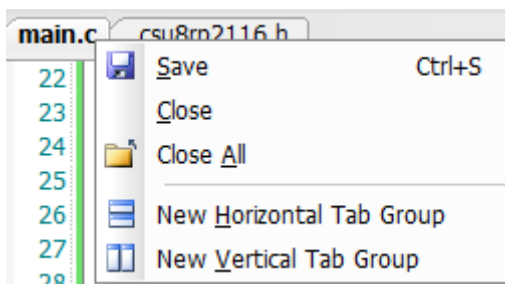
在编辑区选中一个文件标题，右键快捷菜单如下：





<b>Save</b>	保存该文件。
<b>Close</b>	关闭该文件。
<b>Close All</b>	关闭编辑区所有打开的文件。
<b>New Horizontal Tab Group</b>	新建水平选项卡组。
<b>New Vertical Tab Group</b>	新建垂直选项卡组。

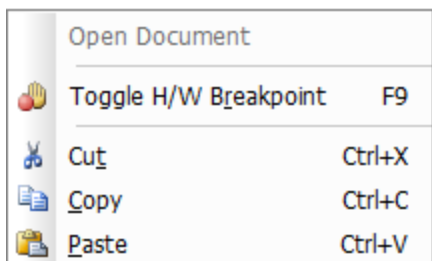
有多个选项卡组时，选中一个文件标题，右键快捷菜单如下：







**Move to Next Tab Group:** 移动至下一个选项卡组。

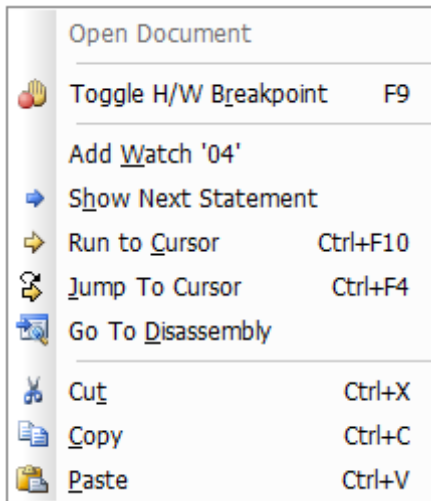
#### 6、编辑区右键快捷菜单

未进入 Debug 状态，在编辑区域内的右键快捷菜单功能如下：



<b>Open Document</b>	选中包含的头文件时，该菜单会亮显，点击该菜单，将会在编辑器中打开该文件。
<b>Toggle H/W Breakpoint</b> 	插入或删除一个硬件断点。
<b>Cut</b> 	剪切选中内容。
<b>Copy</b> 	复制
<b>Paste</b> 	粘贴

进入 Debug 状态，在在编辑区域内的右键快捷菜单功能如下：




<b>Open Document</b>	选中包含的头文件时, 该菜单会亮显, 点击该菜单, 将会在编辑器中打开该文件。
<b>Toggle H/W Breakpoint</b>	插入或删除一个硬件断点。
<b>Add Watch '****'</b>	添加 '****' 至 Watch 窗口
<b>Show Next Statement</b>	显示程序的下一条语句。可快速定位到 PC 指针所在行。
<b>Run to Cursor</b>	运行程序到光标位置。相当于在光标位置设置了一个临时断点。
<b>Jump to Cursor</b>	直接将 PC 指针跳到新的位置, 跳过一段程序后继续进行调试。因为跳过一段程序, 执行结果可能和正常调试结果不一致, 建议在十分清楚程序逻辑的时候使用。
<b>Go to Disassembly</b>	打开反汇编窗口, 并定位到断点对应的反汇编程序
<b>Cut</b>	剪切选中内容。
<b>Copy</b>	复制
<b>Paste</b>	粘贴

### 6.1.6 选中行功能

在编辑窗口，鼠标左键快速单击 3 次将会选中整行。

### 6.1.7 同名的头文件和源文件相互切换功能

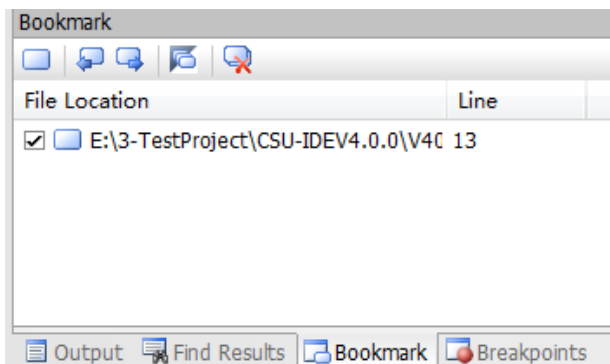
为了方便客户快速查看同名的头文件或源文件，提供了“Open corresponding file”功能。

在编辑区中打开某文件 h/c（例如：A.h），点击工具栏中的  图标，可快速打开同名的 c/h 文件（例如：A.c）。


注：只支持 c 工程中的 h 和 c 文件之间相互切换。

## 6.2 书签功能（Bookmark）


可以在文本中添加书签，所有书签将会在“Bookmark”窗口中显示，双击书签即可以帮助您快速定位。




设置书签：

将光标移动到目标行，在工具栏“Text Editor”，或者“Bookmark”窗口中点击“Toggle Bookmark”  即可。

在编辑器窗口中，设置为书签的行在左侧会有一个蓝色的标记。

“Previous Bookmark” ：指向“Bookmark”窗口中上一条书签，选中的书签在“Bookmark”窗口中会黑体显示，且会自动关联到源程序。

“Next Bookmark” ：指向“Bookmark”窗口中下一条书签。

“Disable all Bookmark” ：禁用所有书签。

“Clear all Bookmark” ：删除所有书签。

“Previous Bookmark in Current document” ：指向当前文档的上一条书签。

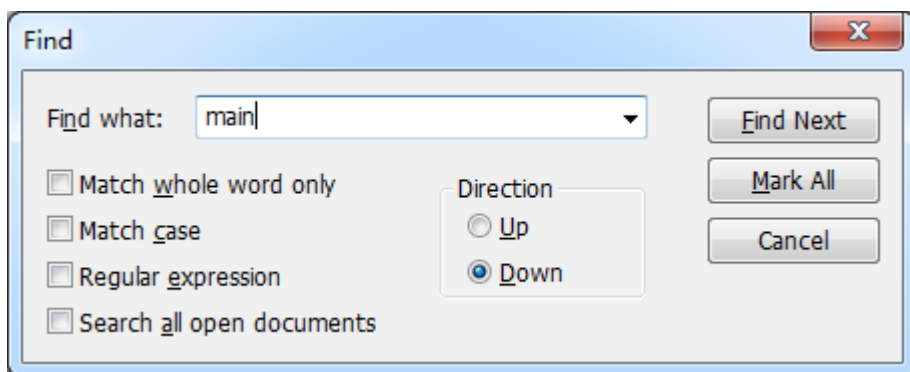
“Next Bookmark in Current document” ：指向当前文档的下一条书签。

## 6.3 文本查找和替换

IDE 提供了查找和替换功能，支持在指定的文件中查找和替换。

### 6.3.1 查找

点击 Edit | Find...，弹出“Find”对话框：



“Find what”：输入所需要查找的文本，支持正则表达式。如果之前有过查找记录，也可直接下下拉列表中选择。

“Match Whole Word Only”：是否仅匹配整个单词。

“Match Case”：是否匹配大小写。

“Regular expression”：是否进行正则表达式的匹配。

“Search all open document”：是否在所有打开文档中进行查找。

上述四项为复选框，可进行多项的组合。但“Regular expression”和“Match whole word only”不能同时选择。

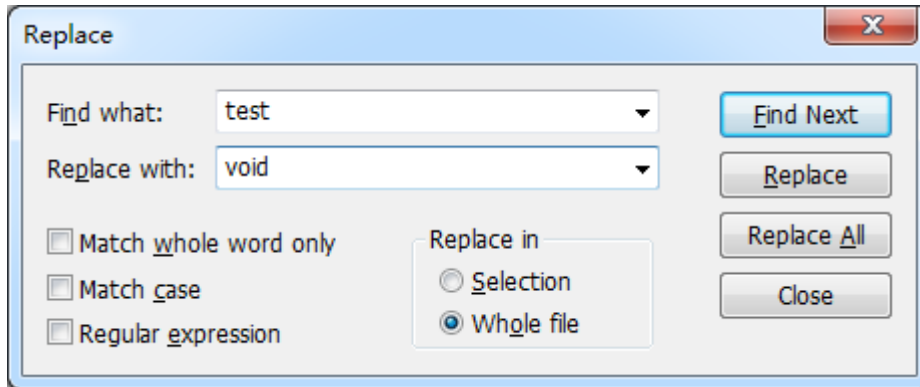
“Up”和“Down”为单选框，用来定义搜索时查找的方向，从当前位置开始向前查找或向后查找。

设置好查找参数和条件后，点击“Find Next”，开始查找匹配的字符串。或使用快捷键 F3，将会使您的查找变得更加方便。

“Mark All”：将所有符合查找条件的行添加书签，并在“Bookmark”窗口中显示。

### 6.3.2 替换

点击 Edit | Replace...，弹出“Replace”对话框：



“Find what”、“Match whole word only”、“Match case”、“Regular expression”、“Find Next”，使用方法请参考“查找”章节。

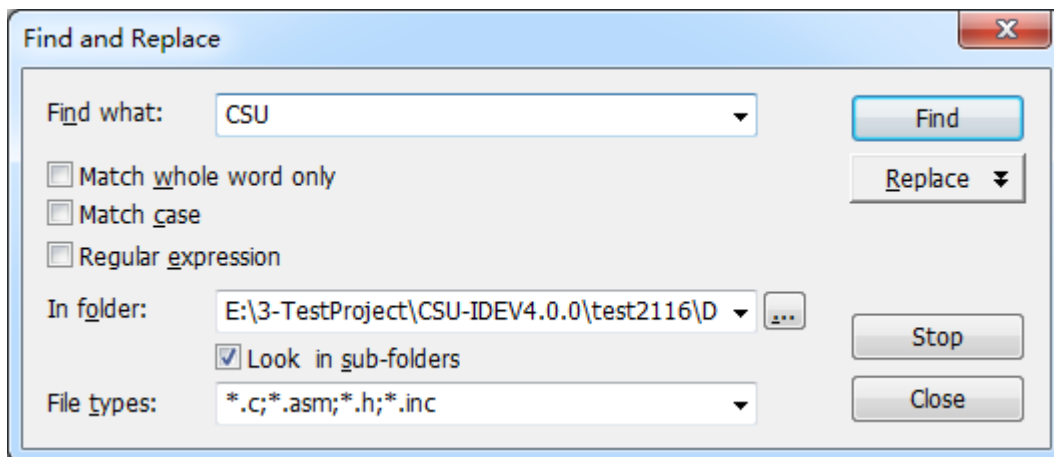
“Replace with”：替换文本编辑框。

“Replace”：进行替换操作。

“Selection”和“Whole file”为单选框，用来设置替换范围，在使用“Replace All”命令时，是对选中区域还是整个文件进行替换。

### 6.3.3 在文件中查找/替换

点击 Edit | Find/Replace In Files...，弹出“Find and Replace”对话框。



“Find what”、“Match whole word only”、“Match case”、“Regular expression”使用方法请参考“查找”章节。

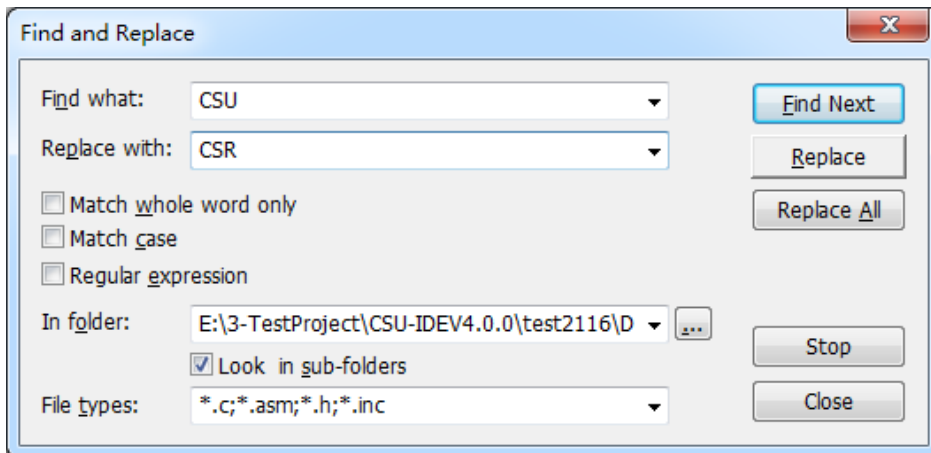
“In folder”：用来设置文件夹的位置，默认为当前工程所在目录。可以更改需要查找的路径，还可以选择在整个解决方案（Entire Solution）、当前活动工程(Active Project)、选择工程(Selected Project)中查找。

“Look in sub-folder”：是否在子文件夹中搜索。

“File types”：设置文件的查找类型，设置后，仅在符合类型的文件中进行查找。您可以在下拉框中选择您需要的文件类型，或者直接编辑，编辑格式请参考下拉列表中的内容。

“Find”：开始查找，并将所有结果输出到“Find Result”窗口中。

点击“Replace”，“Find and Replace”对话框将会多出替换相关项目，如下图：



“Replace”：仅替换当前文本。

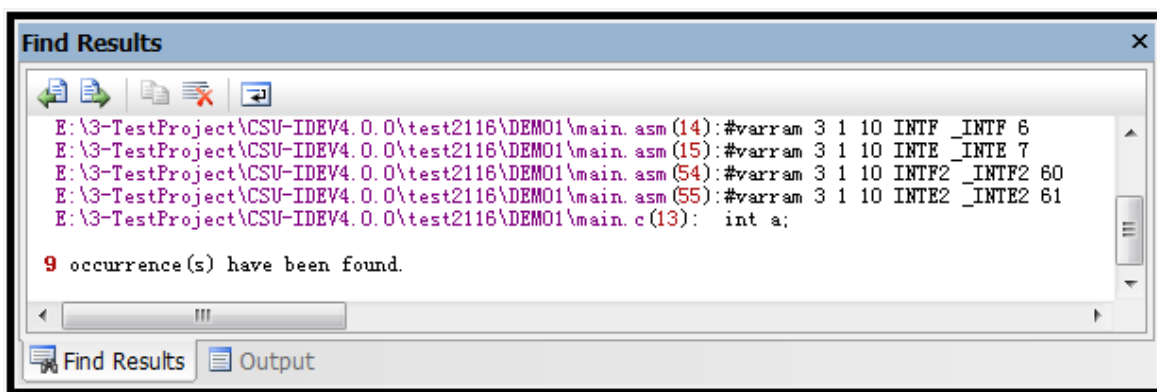
“Replace All”：替换文件夹中所有符合条件的文本。

“Stop”：查找过程中点击“Stop”，提示搜索。

“Close”：关闭对话框。

### 6.3.4 Find result 窗口

查找的结果会输出到“Find Result”窗口中，如下图所示：



“Go to Previous Message” ：指向上一个查找结果

“Go to Next Message” ：指向下一个查找结果

“Copy Line” ：复制“Find Result”窗口中选中的内容

“Clear All” ：清空“Find Result”窗口的内容

“Toggle Word Wrap” ：切换自动换行。

注：在“Find Result”窗口中，双击某条查找结果，在编辑区会快速定位该查找结果。

## 7 调试器

本章将介绍一下与调试相关的操作与窗口功能。

### 7.1 下载

在进行调试之前，需先将编译通过的程序代码下载到仿真版。

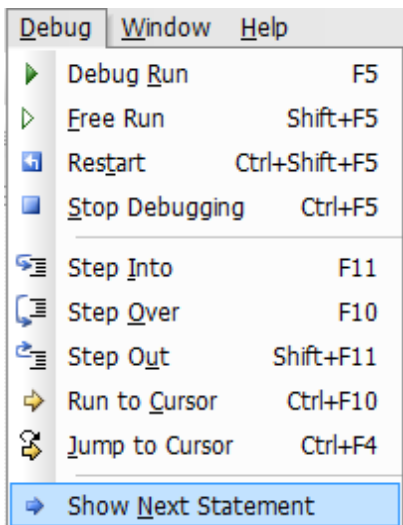
点击 Debug| Debug Run，会执行下载的步骤。

下载完毕，黄色光标会停在程序起始地址 0x00 或 main 函数的起始位置。

### 7.2 调试操作

下载完成后，即可开始进行调试。

先介绍一下 Debug 菜单功能，如下图所示：

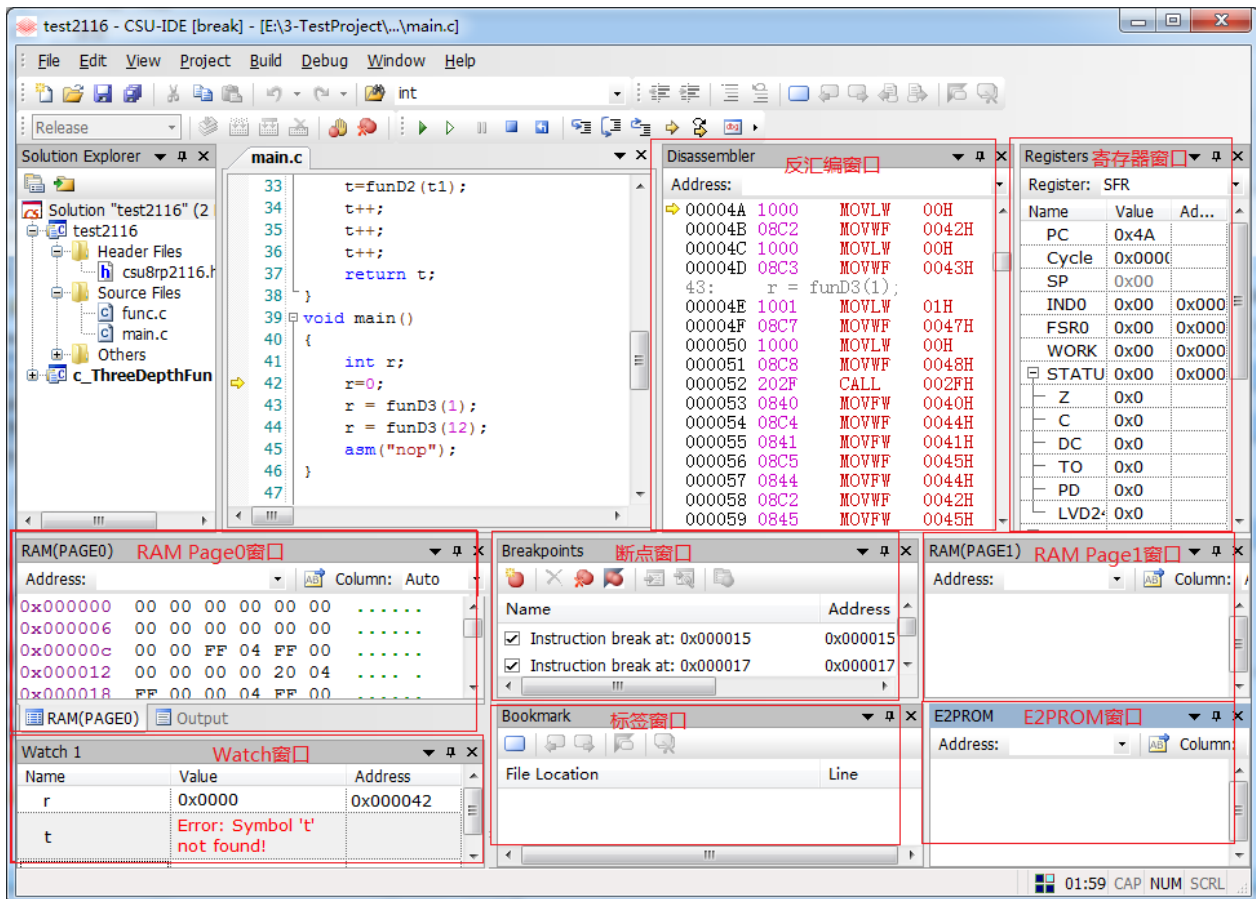


<b>Debug Run</b>	下载程序；如果程序已经下载，则会运行程序，并遇到断点停下来。
<b>Free Run</b>	全速运行程序，遇到断点不会停下来。
<b>Restart</b>	重新运行程序，之前的运行结果将不被保留。
<b>Stop Debugging</b>	退出调试模式，转为编辑模式。
<b>Step Into</b>	单步步入，可进入子程序
<b>Step Over</b>	单步步过，不进入子程序

<b>Step Out</b>	单步步出，可跳出子程序。
<b>Run to Cursor</b>	程序全速运行到光标处停止。
<b>Jump to Cursor</b>	直接将 PC 指针跳到新的位置，跳过一段程序后继续进行调试。因为跳过一段程序，执行结果可能和正常调试结果不一致，建议在十分清楚程序逻辑的时候使用。
<b>Show Next Statement</b>	显示程序的下一条语句。可快速定位到 PC 指针所在行。

### 7.3 调试窗口 (Debug Window)

调试窗口主要用来显示或修改当前监视到的变量、内存、寄存器的值以及反汇编结果等。调试状态下，可以在“Debug”工具栏或 View | Debug Windows 菜单中打开或关闭各个调试窗口。




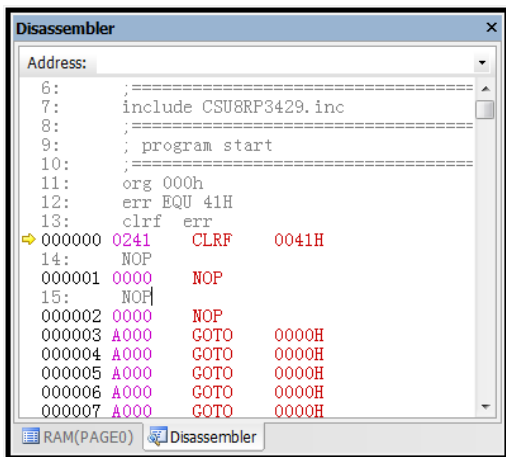
窗口	作用
<b>Disassembler</b>	对下载的目标文件进行反汇编操作并显示，显示的为汇编级别语言
<b>Register</b>	显示通用寄存器，CPU 状态寄存器和一些功能寄存器的值
<b>RAM(Page0)</b>	显示当前 RAM(Page0)内存中的内容



<b>RAM(Page1)</b>	显示当前 RAM(Page1)内存中的内容
<b>E2PROM</b>	显示 E2PROM 中的内容
<b>Watch</b>	显示变量名、变量值、类型或地址
<b>Breakpoint</b>	显示软、硬件断点信息，及进行断点设置等操作
<b>ROM</b>	显示代码区
<b>Call Graph</b>	显示 c 函数调用图

### 7.3.1 反汇编窗口 (Disassembler)

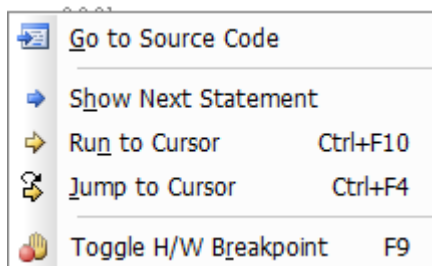
点击 View | Debug Windows | Disassembler，或点击“Debug”工具栏的即可打开反汇编窗口。





在反汇编窗口中会显示源代码、反汇编代码。

在工具栏的“Address”  输入框中输入地址值，按回车将会直接定位到该地址所在行。

右键快捷菜单如下图所示：



**Go To Source Code** : 打开编辑窗口，并定位到对应的源程序。


**Show Next Statement** : 显示程序的下一条语句。可快速定位到 PC 指针所在行。

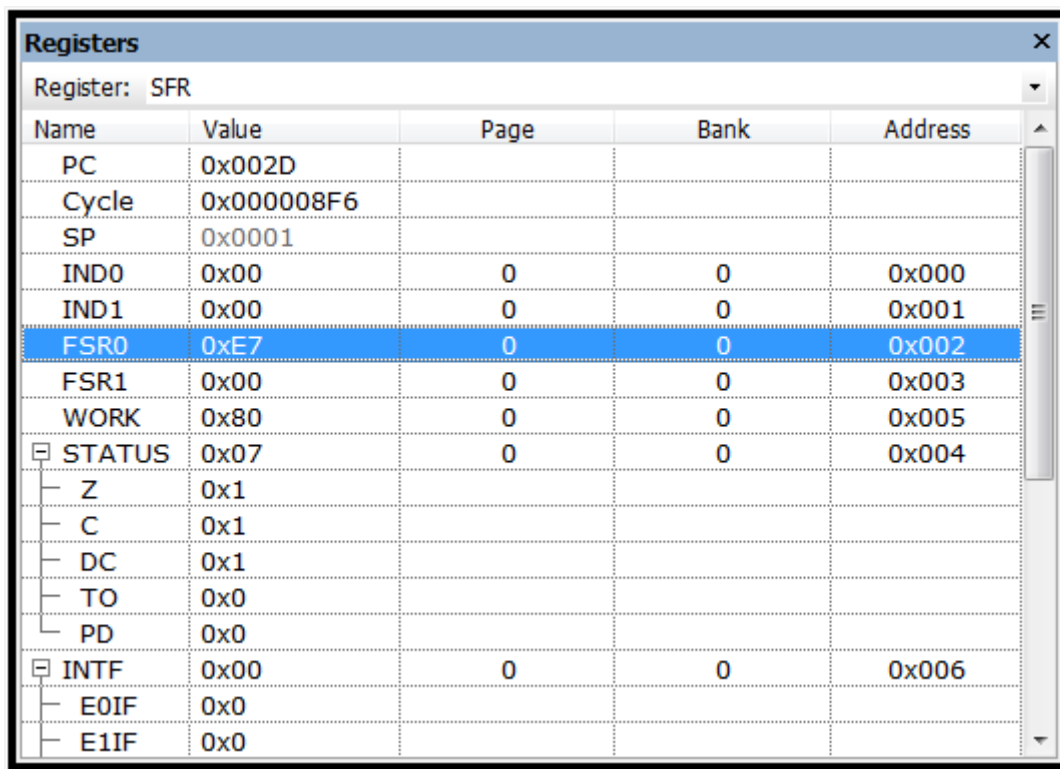
**Run To Cursor** : 运行程序到光标位置。相当于在光标位置设置了一个临时断点。

**Jump to Cursor** :

直接将 PC 指针跳到新的位置，跳过一段程序后继续进行调试。因为跳过一段程序，执行结果可能和正常调试结果不一致，建议在十分清楚程序逻辑的时候使用。

### 7.3.2 寄存器窗口 (Register)

点击 View | Debug Windows | Register，或点击“Debug”工具栏的  即可开启寄存器窗口。



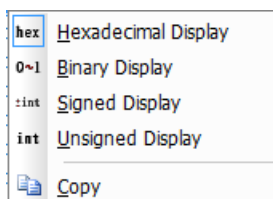
Name	Value	Page	Bank	Address
PC	0x002D			
Cycle	0x000008F6			
SP	0x0001			
IND0	0x00	0	0	0x000
IND1	0x00	0	0	0x001
FSR0	0xE7	0	0	0x002
FSR1	0x00	0	0	0x003
WORK	0x80	0	0	0x005
<input type="checkbox"/> STATUS	0x07	0	0	0x004
<input type="checkbox"/> Z	0x1			
<input type="checkbox"/> C	0x1			
<input type="checkbox"/> DC	0x1			
<input type="checkbox"/> TO	0x0			
<input type="checkbox"/> PD	0x0			
<input type="checkbox"/> INTF	0x00	0	0	0x006
<input type="checkbox"/> E0IF	0x0			
<input type="checkbox"/> E1IF	0x0			


寄存器窗口会来显示 PC、Cycle、状态寄存器和其它功能寄存器的值。






双击 Value 可编辑寄存器的值(寄存器的值是否可以修改，由寄存器的属性决定)。修改寄存器的值后，仿真版也会相应变化。

<b>PC</b>	即将运行的程序地址
<b>SP</b>	栈指针寄存器
<b>Cycle</b>	统计已执行指令的周期
<b>STATUS</b>	状态寄存器

右键快捷菜单如下图所示：



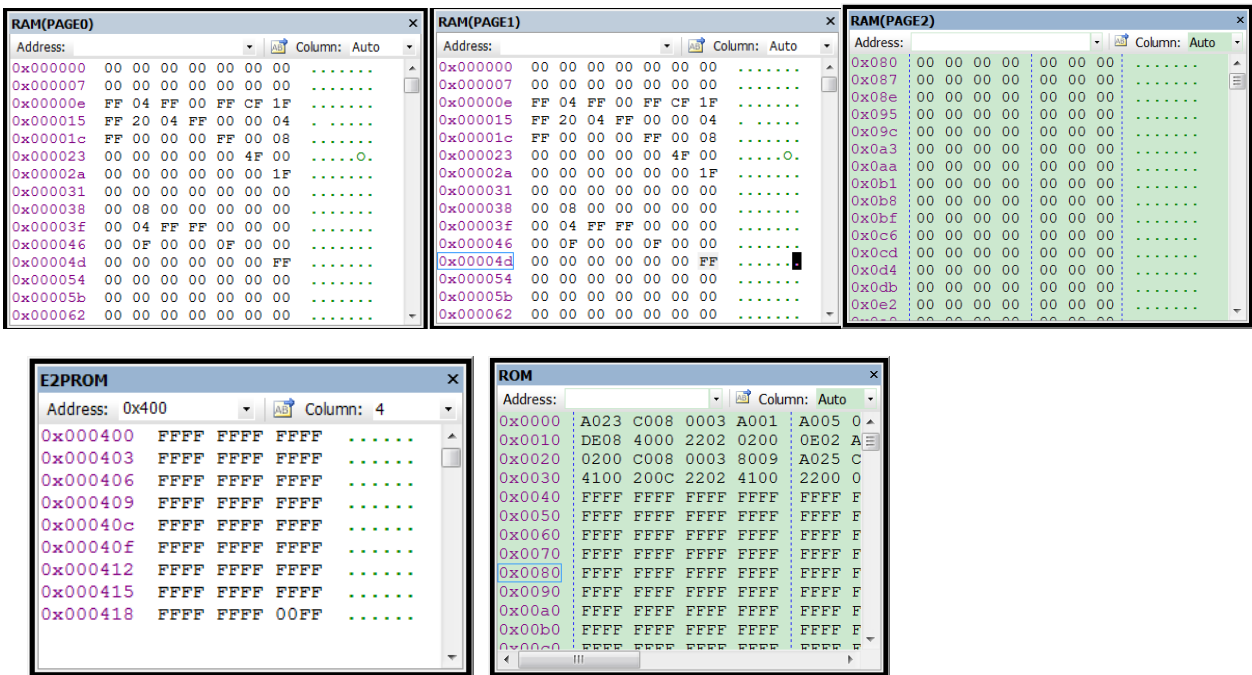
hex	Hexadecimal Display
0~1	Binary Display
:int	Signed Display
int	Unsigned Display
	Copy


hex 	使用十六进制数显示，其它窗口如 watch、Memory 窗口也会相应变化
0~1 	二进制数显示
sint 	有符号的十进制数显示
int 	无符号的十进制数显示
Copy 	复制

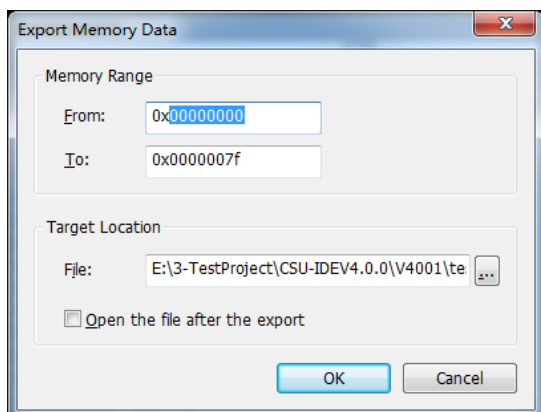
点击“Register”的下拉框 ，可以查看或修改其它寄存器地址。

### 7.3.3 RAM(E2PROM)ROM 窗口 (RAM(Page01\2\3)、E2PROM、ROM)

在 View | Debug Windows 中可选择 RAM(Page0)、RAM(Page1)、RAM(Page1)、E2PROM、ROM 窗口。

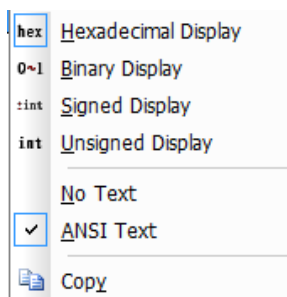


- 在工具栏的“Address”  输入框中输入地址值，按回车将会直接定位到该地址所在行。
- 在工具栏“Column”  下拉列表框中选择每行显示的 byte 个数。
- “Export Memory Data”  用于导出“Memory”空间的一段内容，并存储到文件中。在“Memory range”输入开始和结束地址，“Target”中设置文件路径及文件名，点击【OK】即可。



“Open the file after the export”：导出完毕之后文件将会在编辑器区域打开该文档。

右键快捷菜单，如下：



提供了不同的数制显示方式，包括十六进制（Hexadecimal Display）、二进制（Binary Display）、有符号（Signed Display）、无符号 Unsigned Display）；文本显示方式、排列方式，以及复制、编辑功能等。

hex		使用十六进制数显示，其它窗口如 watch、Memory 窗口也会相应变化
0~1		二进制数显示
sint		有符号的十进制数显示
int		无符号的十进制数显示
No Text		不显示文本
ANSI Text		文本用 ANSI 方式显示
Copy		复制

注：E2PROM、ROM 窗口中的值禁止编辑。

### 7.3.4 变量窗口（Watch）

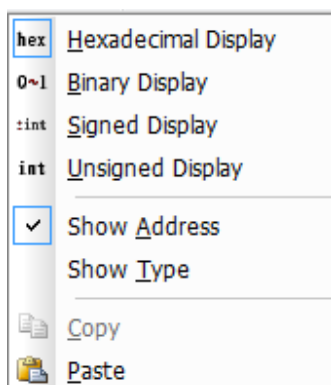
点击 View | Debug Windows | Watch，或点击“Debug”工具栏的 即可开启变量窗口。

Name	Value	Page	Bank	Address
PC	0x002D			
work	0x80	0	0	0x005
r1	0x0000	0	1	0x006
r2	0x0000	1	2	0x080

“Watch”窗口可用来监视变量、寄存器的值，类型或地址。可以在“Name”列输入需要查看的变量或符号，也可直接从其它文本拖至“Name”中，“Watch”窗口会自动显示对应的值。可以在“Value”列中修改变量的值，修改后“RAM”窗口和仿真版上的值会同步变化。

程序运行过程中，如果监视到的值有被改变，将会红色显示。

右键快捷菜单如下图所示：

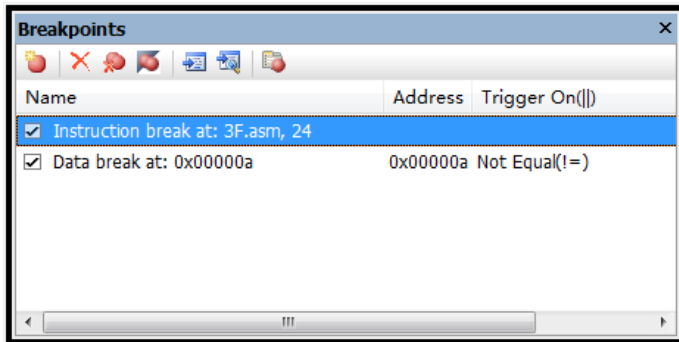


hex		使用十六进制数显示，其它窗口如 watch、Memory 窗口也会相应变化
0~1		二进制数显示
sint		有符合的十进制数显示
int		无符号的十进制数显示
Show Address		显示数据地址
Show Type		显示数据类型
Copy		复制
Paste		粘贴

### 7.3.5 断点窗口 (Breakpoint)

“Breakpoint”窗口，可设置硬件断点，包括指令地址中断和数据中断，最多支持 5 个硬件指令地址中断和 3 个数据

中断。



**Name:** 显示断点类型、文件信息、行号信息、地址信息等

**Address:** 断点地址信息

**Trigger On:** 所有数据断点的逻辑关系，包含与和或，点击 **Trigger On(&&)** 可相互切换。

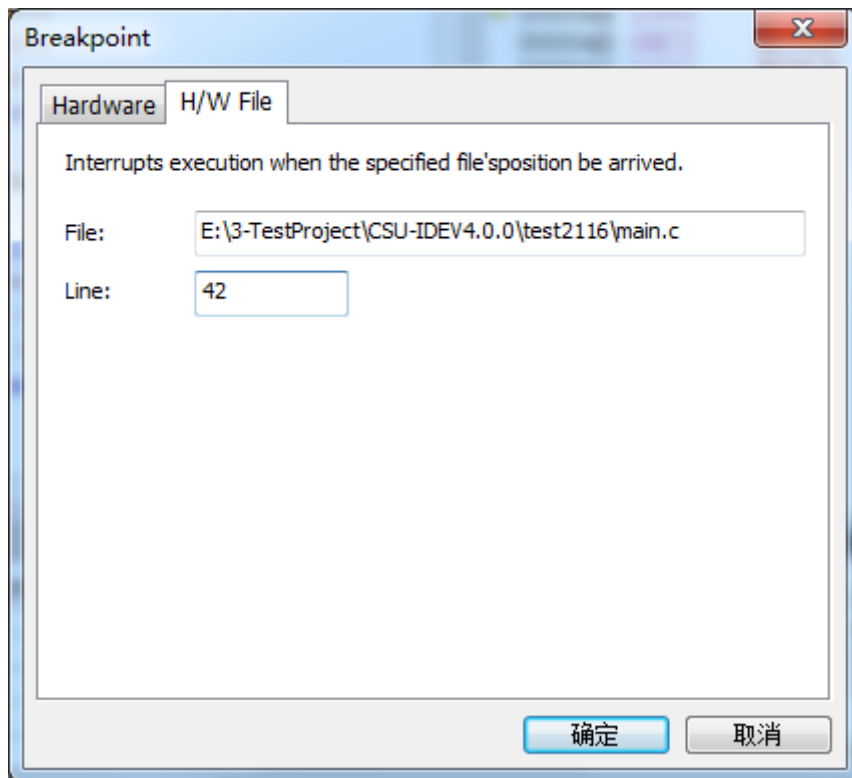
**||:** 所有数据断点只要有一个数据断点满足触发条件，则触发数据断点。


**&&:** 所有数据断点全部满足触发条件，才会触发数据断点。

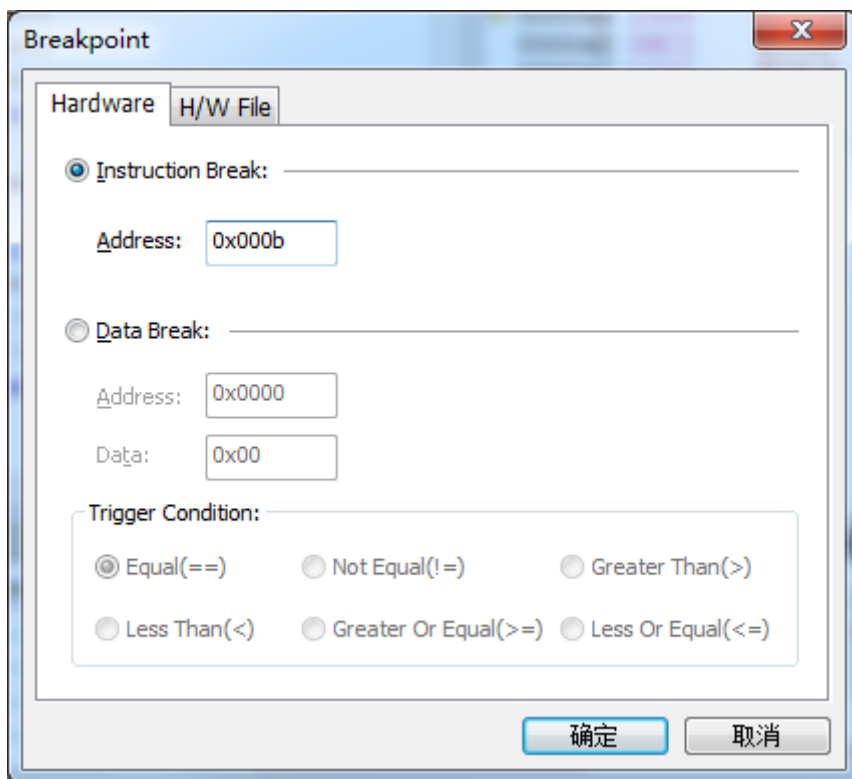
<b>New</b>	新建一个断点
<b>Delete</b>	删除选中的断点
<b>Delete All</b>	删除所有断点
<b>Disable All</b>	禁用所有断点，禁用后，程序将不会在断点位置产生中断。
<b>Go To Source Code</b>	定位到断点对应的源文件
<b>Go To Disassembly</b>	定位到断点对应的反汇编程序
<b>Edit</b>	编辑断点信息

断点设置方式

- 1、在编辑器窗口或反汇编窗口中的指令行按“F9”
- 2、在编辑器窗口或反汇编窗口中的指令行，点击“Build”工具栏上图标 ；
- 3、在断点窗口点击新建按钮 ，在“H/W File”编辑框中输入文件的绝对路径，在“Line”编辑框中输入行号，点击“确定”；



- 4、在断点窗口点击新建按钮, 切换到“Hardware”标签页, 参考下图



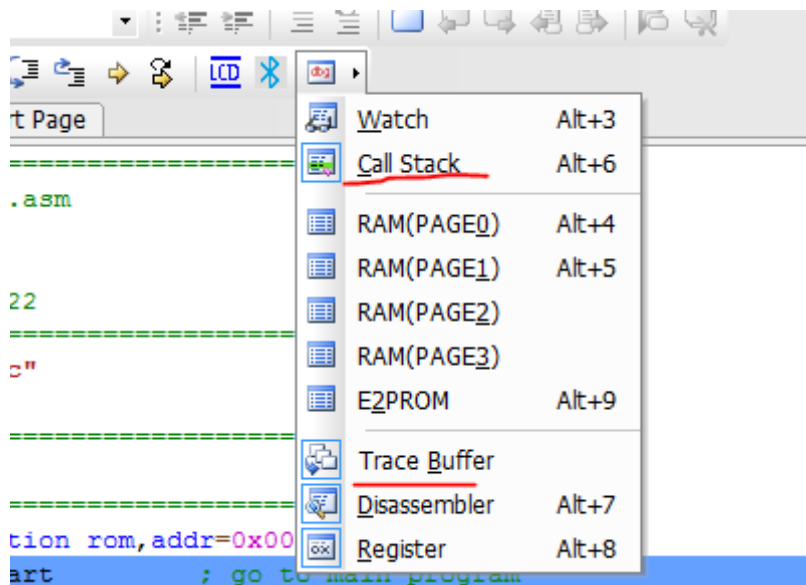
下面分别介绍每一项的作用：

**Instruction Break** 设置指令中断

<b>Address</b>	设置要中断的地址
<b>Data Break</b>	设置数据断点
<b>Address</b>	设置被监视的数据的地址
<b>Data</b>	从指定地址读取的值 data 符合 Trigger Condition 时，将产生数据中断
<b>Trigger Condition</b>	选择触发条件：==、!=、>、<、>=、<=。

### 7.3.6 Trace Buffer 和 Call Stack 窗口 (Trace Buffer)

可通过 IDE 菜单栏 View->Debug->View ，或通过工具条直接调出 trace buffer 和 call stack 功能窗口。



#### (1) “Trace Buffer” 窗口

当仿真停止时，Trace buffer 提供最新的 100 条仿真 PC 值运行记录，用户可通过记录追踪代码是否运行正常。当运行超过 100 条指令时最开始运行的记录将会被覆盖，仅保留最新的 100 条仿真记录。

仅在仿真复位后才将记录清空，当暂停仿真后又继续开始仿真则继续记录。IDE V5.1.0 使用 C 工程时因自带启动代码，因此每次复位后 trace buffer 会记录启动代码运行记录，并非清空状态。

使用简介：

如下图所示，Trace buffer 中记录了程序运行轨迹，第 1 行为当前 PC 值和相关指令（该行准备执行），第 2 行为上一 PC 值运行的指令相关，以此类推。双击其中某一行，会跳转到该行对应的反汇编位置处（非源文件位置）。



Index	PC	Code Bytes	Instruction
1	0x0018	0000	NOP
2	0x001E	0003	RETURN
3	0x001D	0000	NOP
4	0x0021	0003	RETURN
5	0x0020	0000	NOP
6	0x001F	0000	NOP
7	0x001C	801F	CALL 01FH
8	0x001B	0000	NOP
9	0x001A	0000	NOP
10	0x0017	801A	CALL 01AH
11	0x0016	0000	NOP
12	0x0015	0000	NOP
13	0x0012	8015	CALL 015H
14	0x0011	0000	NOP
15	0x0010	0000	NOP
16	0x0009	8010	CALL 010H
17	0x000F	0003	RETURN
18	0x000E	0000	NOP
19	0x000D	0000	NOP
20	0x0008	800D	CALL 00DH
21	0x0007	0000	NOP
22	0x0006	0000	NOP
23	0x0005	0000	NOP
24	0x0000	A005	GOTO 005H

(2) “Call Stack” 窗口

当仿真程序有压栈操作时（如 call 调用子函数、进入中断等），call stack 提供进入子函数的入口地址。当使用 return、retfie 等出栈后，call stack 会自动删除该记录。用户可使用 call stack 功能清楚了解函数的调用关系。

使用简介：

如下图所示，程序从 39 行调用子函数 s2，在子函数 s2 中调用子函数 s3，在子函数 s3 中调用子函数 s4。运行到 s4 时，call stack 窗口分别记录 s2、s3、s4 子函数的入口位置与当前 PC 的位置（入口位置含 PC 值、文件名、所在行数）。

第一栏箭头指向当前 PC 所在位置（该行准备执行），第 2 行为 s4 子函数的入口位置：PC=17，在 asm\_test.asm 文件 59 行处调用；第 3 行为 s3 子函数的入口位置：PC=12，在 asm\_test.asm 文件 52 行处调用。双击 call stack 窗口第 3 行，光标移到 s3 子函数入口位置高亮显示。通过 call stack 可定位所有堆栈入口，方便调试。

The screenshot displays the CSU-IDE interface with two main panels. The left panel, titled 'asm\_test.asm', shows assembly code with line numbers 46 through 70. The code includes several subroutines: 's2' (lines 49-55), 's3' (lines 56-62), 's4' (lines 63-69), and 's5' (line 70). The instruction 'call s3' at line 52 is highlighted in blue. A yellow arrow on the left margin points to line 65. The right panel, titled 'Call Stack', shows a list of stack frames: '0x001B asm\_test.asm line:65', '0x0017 asm\_test.asm line:59', '0x0012 asm\_test.asm line:52', and '0x0009 asm\_test.asm line:39'. A green arrow points to the top frame, and a yellow arrow points to the second frame.

```
asm_test.asm
46     nop
47     return
48
49 s2:
50     nop
51     nop
52     call    s3
53     nop
54     return
55
56 s3:
57     nop
58     nop
59     call    s4
60     nop
61     return
62
63 s4:
64     nop
65     nop
66     call    s5
67     nop
68     return
69
70 s5:
```

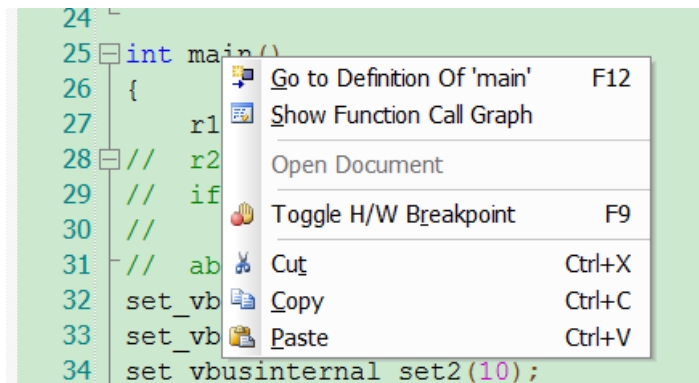
Call Stack

- 0x001B asm\_test.asm line:65
- 0x0017 asm\_test.asm line:59
- 0x0012 asm\_test.asm line:52
- 0x0009 asm\_test.asm line:39

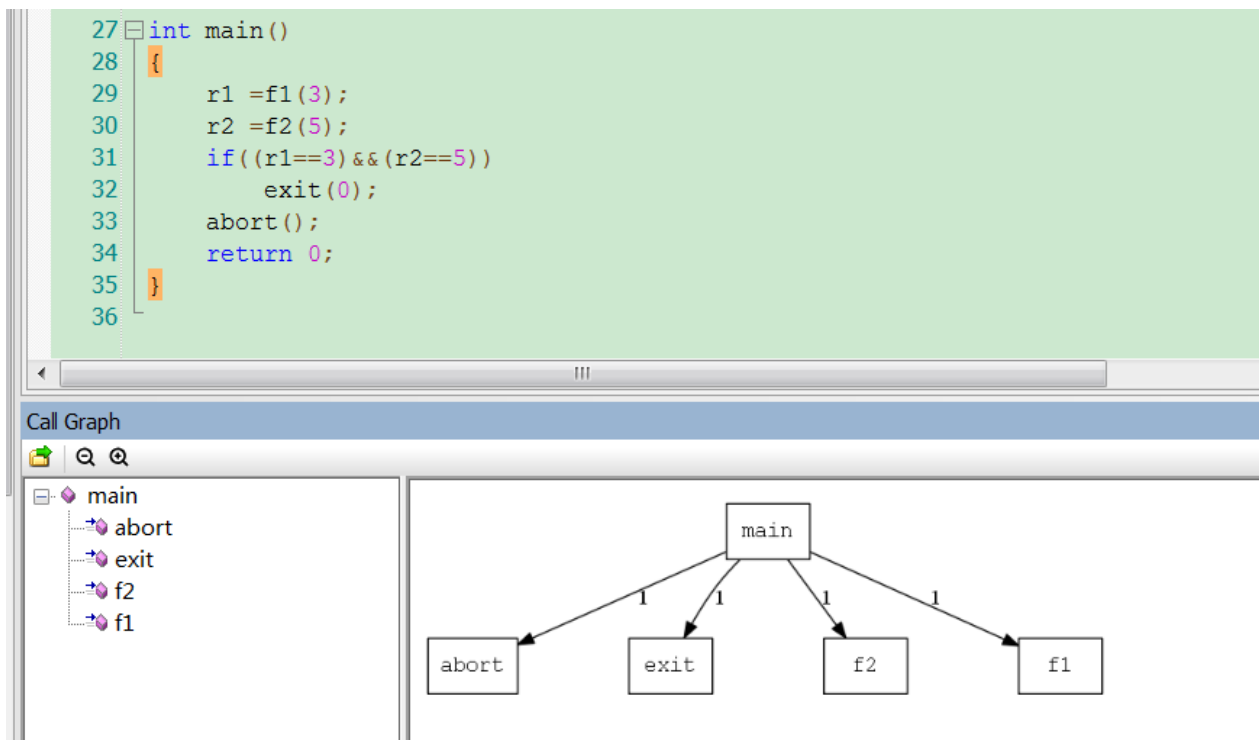
## 7.4 调用图窗口 (Call Graph Window)

CSU\_IDE 对编译通过的 C 工程（只支持 C 工程），通过 View -> Function Call Graph 功能，可以打开 Call Graph 窗口。

在编辑区选中函数，选择右键快捷菜单“Show Function Call Graph”，



将在 Call Graph 窗口显示该函数的调用图关系。



注：

- 1、函数调用图，显示的是在 solution Explorer 窗口中选中的工程中的函数。
- 2、调用汇编函数，显示的是 section 名称。

例：显示的是 demoSec1、demoSec2

The screenshot displays the CSU-IDE V5.0 interface with three code editors and a call graph.

**cfun.c**

```

1 extern bank1 int f1(int a);
2 extern bank2 int f2(int a);
3 bank1 r1;
4 bank2 r2;
5 unsigned int iicdata[3];
6 void test_asm1()
7 {
8     asm("call fun1");
9     asm("call fun2");
10    asm("call fun3");
11    asm("call fun4");
12 }
13
14 void test_asm2()
15 {
16    asm("call demo1");
17    asm("call demo2");
18    asm("call demo3");
19    asm("goto demo4");
20 }

```

**asm.asm**

```

1 demoSec2 .section rom
2 demo1:
3     return;
4 demo2:
5     return;
6 demo3:
7     return;
8 demo4:
9     return;
10 .ends

```

**cstartup.asm**

```

275 ; goto $
276 ;.endif
277 ;.if CPU_HALT
278 ; halt
279 ;.endif
280 .ends
281
282
283 demoSec1 .section rom
284 fun1:
285     return;
286 fun2:
287     return;
288 fun3:
289     goto demo3
290     return;
291 fun4:
292     call demo4
293     return;
294 .ends

```

**Call Graph**

The call graph shows the execution flow:

```

graph TD
    test_asm1 -- 1 --> demoSec1
    demoSec1 -- 2 --> demoSec2

```

## 8 LCD 模拟器

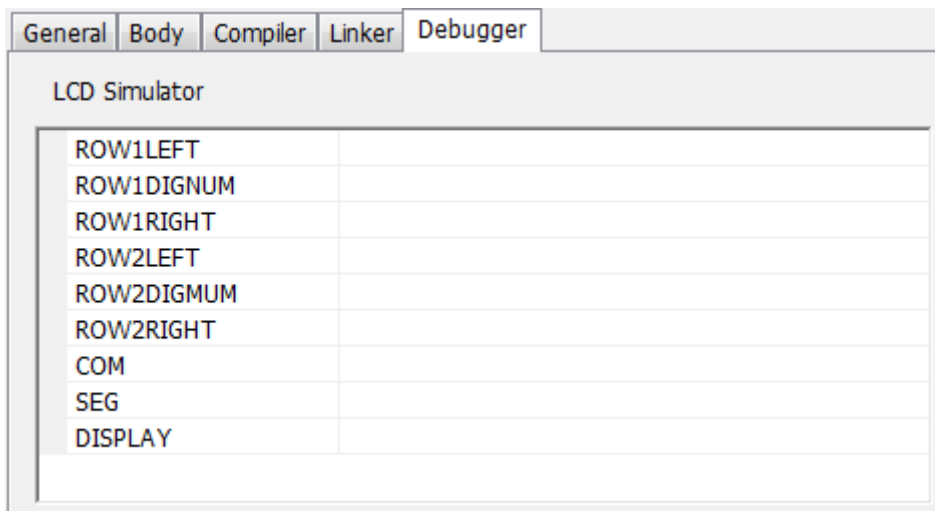
CSU-IDE 的 LCD 模拟器，提供了 LCD 驱动器的软件仿真和在线仿真。在 LCD 模拟器上，使用者可根据自己配置的图形和控制程序，来实时显示这些图形。这在尚未取得真实的 LCD 硬件面板之前，LCD 模拟器将给开发带来便利。

### 8.1 LCD 模拟器的图形界面



### 8.2 LCD 模拟器图形界面的配置

在新建或打开一个工程后，在【Project】菜单中点击【Setting】，会弹出调试设置窗口，如下图所示：



该列表用来配置 LCD 模拟器。

ROW1、ROW2：代表 LCD 的第一行和第二行。

LEFT：表示 LCD 某一行的左边四个红点；

RIGHT：表示 LCD 某一行的右边四个红点。红点一般用来给使用者设置自己定义的符号，如 kg，g 等,可填写在红点旁的文本框中。

LEFT 和 RIGHT 列数值设置范围为 0-4（数字代表红点可用的个数）。

DIGNUM：表示数码管；DIGNUM 列数值设置范围为 0—12（数字代表数码管可用的个数）。

COM：表示配置 COM 的数量；

SEG：表示配置 SEG 的数量。

DisplayEnable：表示 LCD 显示的使能电平。如果输入 0，表示低电平 LCD 进行亮显示；输入 1，表示高电平 LCD 进行亮显示。

### 8.3 LCD 模拟器映射关系设置

映射关系字符约定：

L：代表 LCD 某行左边的四个红点。

R：代表 LCD 某行右边的四个红点。

T：代表 LCD 第一行。

D：代表 LCD 第二行。

S：代表数码管。

利用这些字符进行组合：

LT：LCD 第一行左边的四个红点。

LD：LCD 第二行左边的四个红点。

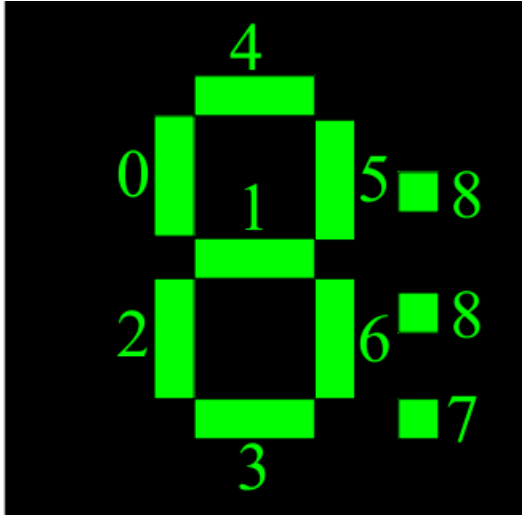
RT：LCD 第一行右边的四个红点。

RD：LCD 第二行右边的四个红点。

ST: LCD 第一行的数码管。

SD: LCD 第二行的数码管。

其中数码管的段号进行如下约定：



再加上一些数字，就可以指定四个红点具体的某一个；或是第几个数码管的第几段。

对于红点，只需在 LT、LD、RT、RD 后面加上 0 到 3 的数字，红点从上往下数，分别对应 0 到 3。例如：

RT2 则表示 LCD 第一行右边的第 3 个红点（从上往下数）。对于数码管，在 ST 和 SD 后加数字表示第几个数码管（从左往右数），之后加下划线“\_”，再加数字（0 到 8）。其中数字表示这个数码管对应的段号。例如：

ST4\_5 则表示 LCD 第一行数码管的第四个数码管的第 5 段。

**注意：** LT、LD、RT、RD、ST、SD 后加的数字受 LCD 配置的参数约束，如果超出约束则称为无效映射关系字符。如在 LCD Setting 中把 LCD 第一行数码管的个数配为 6，那么之后 ST 后的数字范围为 1 到 6；把 LCD 第二行左边的红点设为 3，则 LD 后的数字范围为 0 到 2。

映射关系字符在书写中忽略大小写。

LCD 映射关系设置有两种方法：

1) 通过 LCD Simulator 窗口中的 com/seg 栏输入映射关系字符

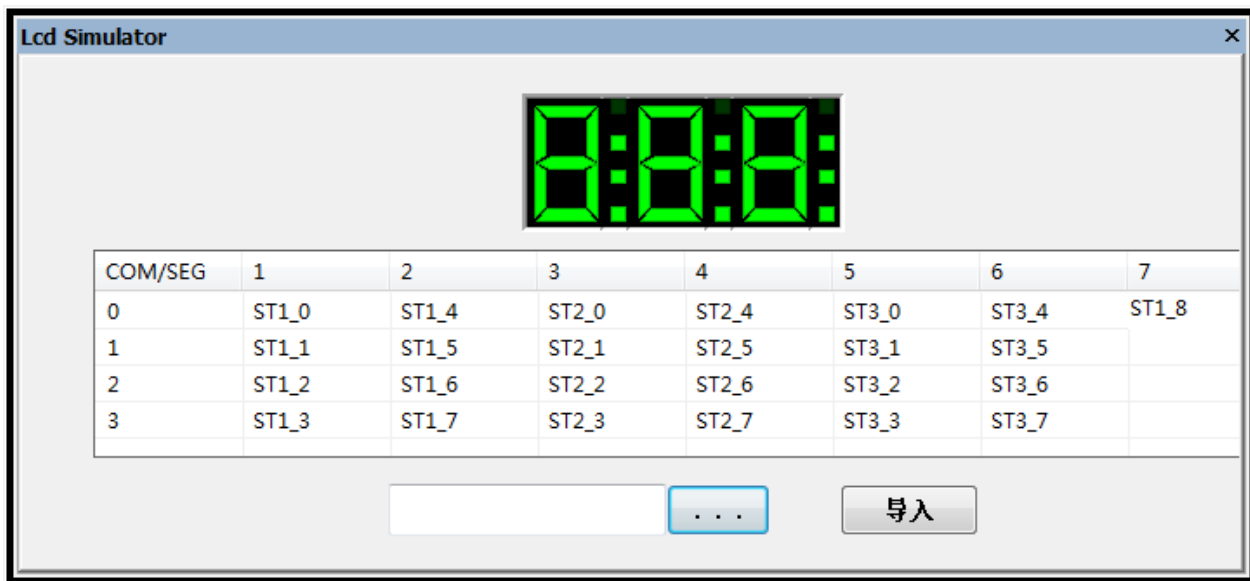
点击工具栏的 ，弹出 LCD 模拟器窗口。之后在 com/seg 栏中填入的映射关系字符。

例如：将 seg1[2]映射为 LCD 第一行第二个数码管的第 5 段，应在 seg1 和 com2 对应的地方输入 ST2\_5。S 表示数码管，T 表示 LCD 第一行，2 表示这行的第 2 个数码管，5 表示这个数码管的第 5 段。

又如：将 seg5[0]映射为 LCD 第二行右边红点的第一个（最上面的那个红点），则应在 seg5 和 com0 对应的地方输入 RD0。R 表示右边的红点，D 表示 LCD 第二行，0 表示第一个红点。

注意：如果书写错误或无效，在所填的地方的字符会自动清除。

下面是某个程序的 LCD 映射关系：



## 2) 通过 lcd 格式文件导入映射关系

新建一个工程，软件会自动生成一个名为 `map.lcd` 的文件，里面的映射关系对应于 CSU\_ICE 上的 LCD，用户也可以自己另外新建一个 `lcd` 格式的映射关系文件。在 `lcd` 格式的文件中填写好映射关系并保存后，只需在 LCD Simulator 窗口，点击 选择 `lcd` 格式映射关系文件，之后点击 把文件的映射关系导入 `com/seg` 栏中。

为了简化映射关系字符的书写，此书写格式与前面的有一些不同。这里主要采用数组的形式。例如：将 `seg1[2]` 映射为 LCD 第一行第二个数码管的第 5 段，只需在 `lcd` 格式文件输入 `seg1[2]=ST2[5];`。记住要有分号，分号表示这一语句的结束标志。采用了数组的形式，就容易同时对多位进行映射。如：将 `seg4` 低四位对应 LCD 第一行左边的四个红点，其中 `seg4` 的最低位对应第四个红点。只需在 `lcd` 格式文件输入 `seg4[3:0]=lt[0:3];`。

注意：如果某语句书写错误，在导入 LCD 映射关系时，错误语句及错误语句以后的映射关系将不能导入。

`seg3[3:0]=st2[3:0];`

`seg4[3:0]=st2[7:4];`

`seg5[3:0]=st3[3:0];`

`seg6[3:0]=st3[7:4];`

`seg7[0]=st1[8];`

## 8.4 LCD 模拟器的仿真

LCD Simulator 不能独立进行仿真，要配合软件仿真或在线仿真一起使用。在软件仿真或在线仿真状态下，打开 LCD Simulator 窗口，之后运行，LCD 模拟器会实时显示图形。

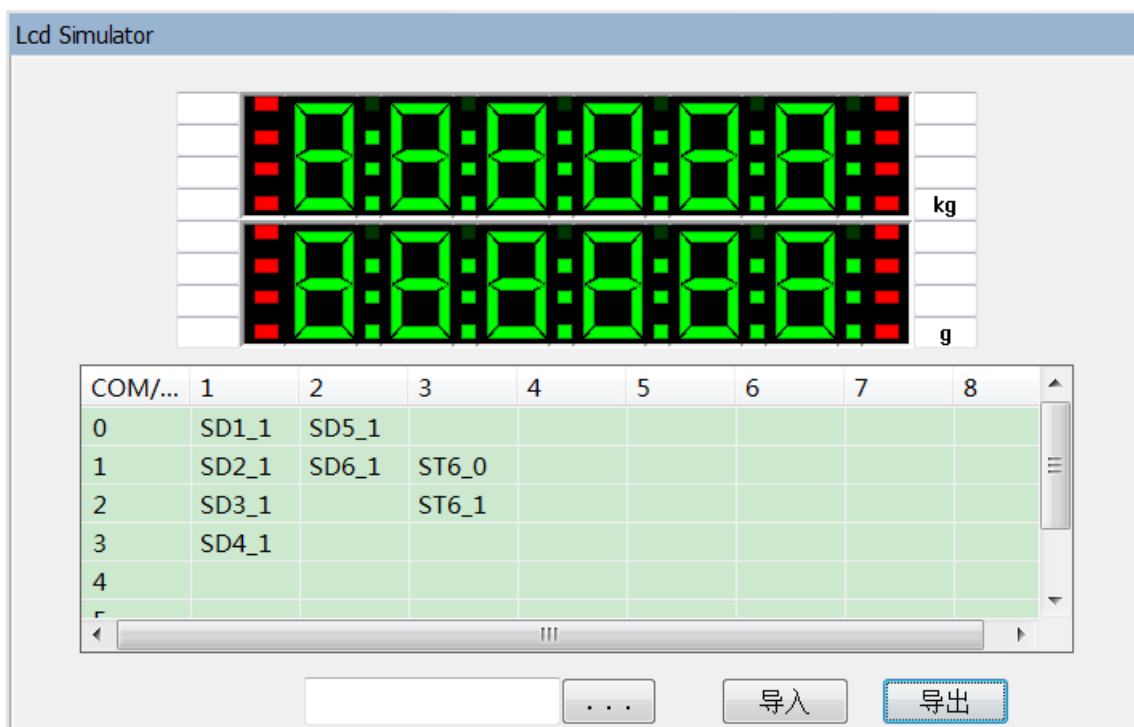


## 8.5 LCD 模拟器配置的自动保存

在关闭 LCD Simulator 窗口或软件时，LCD Simulator 能自动保存 LCD 图形界面参数和映射关系。之后再打开软件或 LCD Simulator 窗口，LCD Simulator 能自动读取保存的数据。


## 8.6 导入/导出功能

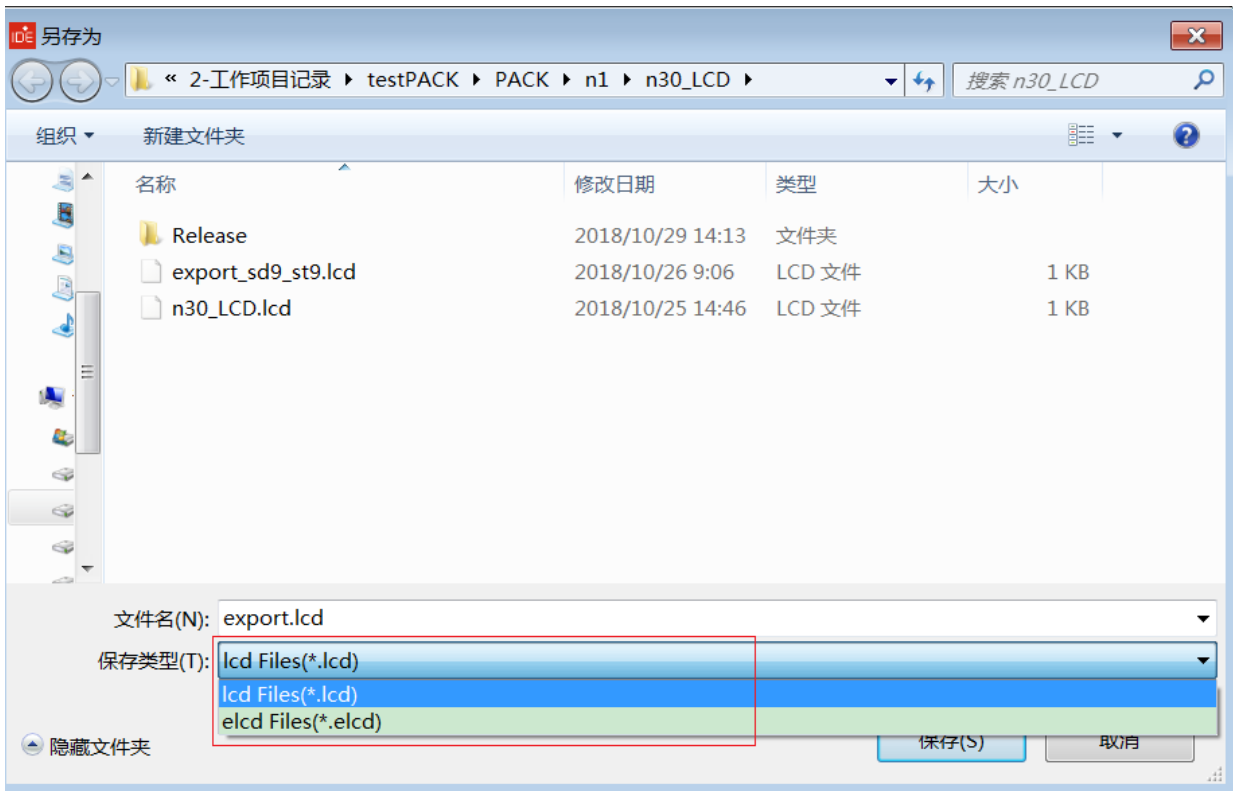
在 LCD Simulator 窗口，支持导出、导入 COM/SEG 的设置值，以及红点的设置值，使用不同的文件进行保存。




\*.lcd: 保存 COM/SEG 之间的映射关系。

\*.elcd: 保存 LCD 第一排和第二排左、右红点的数据。

点击  弹出对话框，可选择文件类型，保存相关内容。




点击 ，在弹出的对话框中选择文件类型，并导入相关文件。

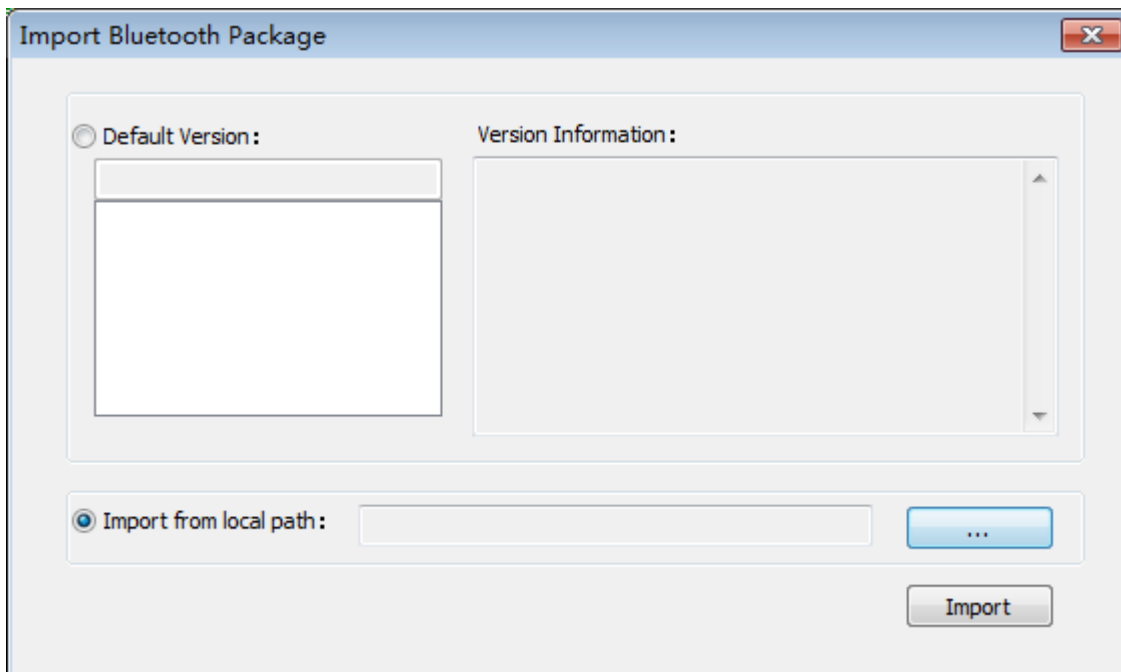
## 9 蓝牙

本章将介绍蓝牙相关的设置，以及如何导入蓝牙包。

### 9.1 导入蓝牙包

对于需要蓝牙文件的工程，需先导入蓝牙包。

点击工具条中图标，将会弹出导入蓝牙包对话框，如下所示：



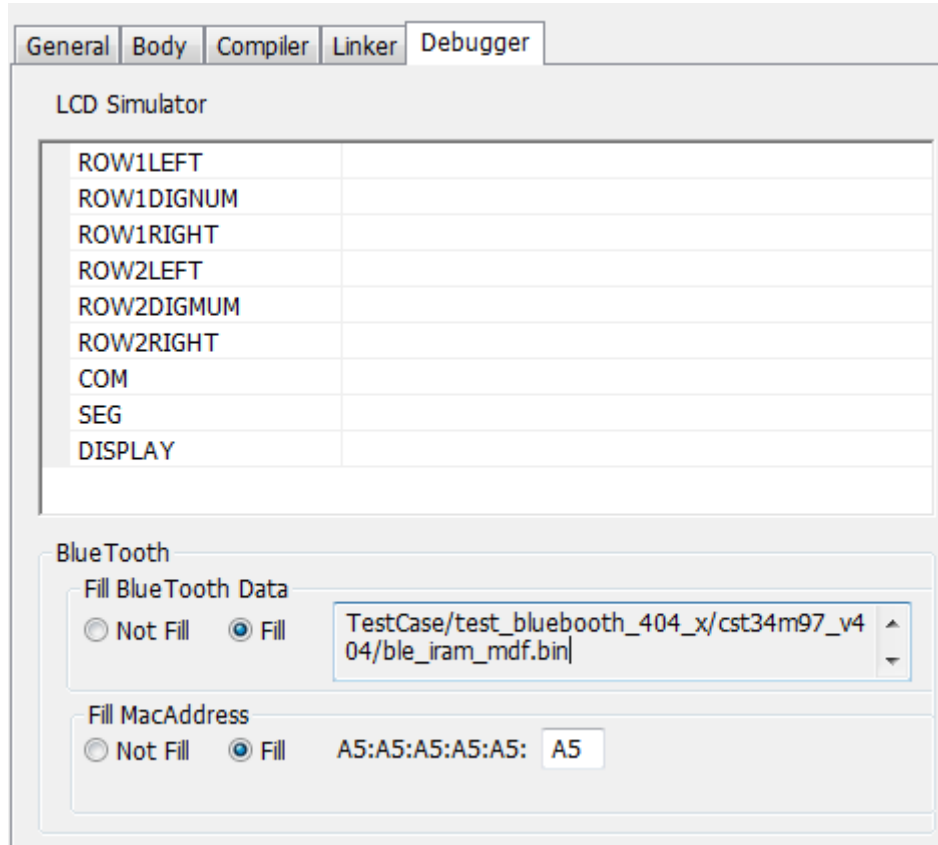
用户可依据个人需求选择导入的蓝牙包，可从 **Default Version** 中选择 IDE 提供的蓝牙包（暂不提供），也可以从本地路径中选择蓝牙包文件（后缀名为：**btpkg**）。

- **Default Version:** 显示 IDE 所提供的蓝牙包，在下拉列表中显示蓝牙版本名称，在右侧的 **Version Information** 中显示具体的版本信息。
- **Import from local path:** 从本地路径中选择需要导入的蓝牙包。

**注意：** 请注意蓝牙包格式；在导入时若提示蓝牙文件格式不正确或导入失败，请向相关人员确认蓝牙包格式的正确性。

## 9.2 蓝牙设置

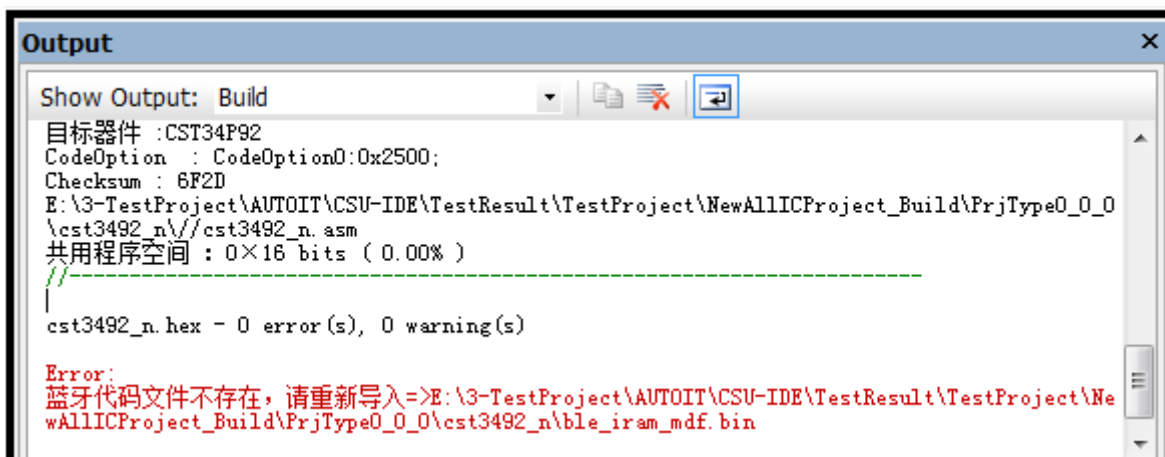
在 ProjectSetting 中提供了蓝牙设置功能，界面如下：



**Fill Blue Tooth Data:** 选择是否将指定的蓝牙文件填充至 HEX 档。右侧显示蓝牙文件存放路径，禁止编辑。

**Fill MacAddress:** 选择是否填充 Mac 地址；右边编辑框可输入 mac 地址，需输入十六进制数据。

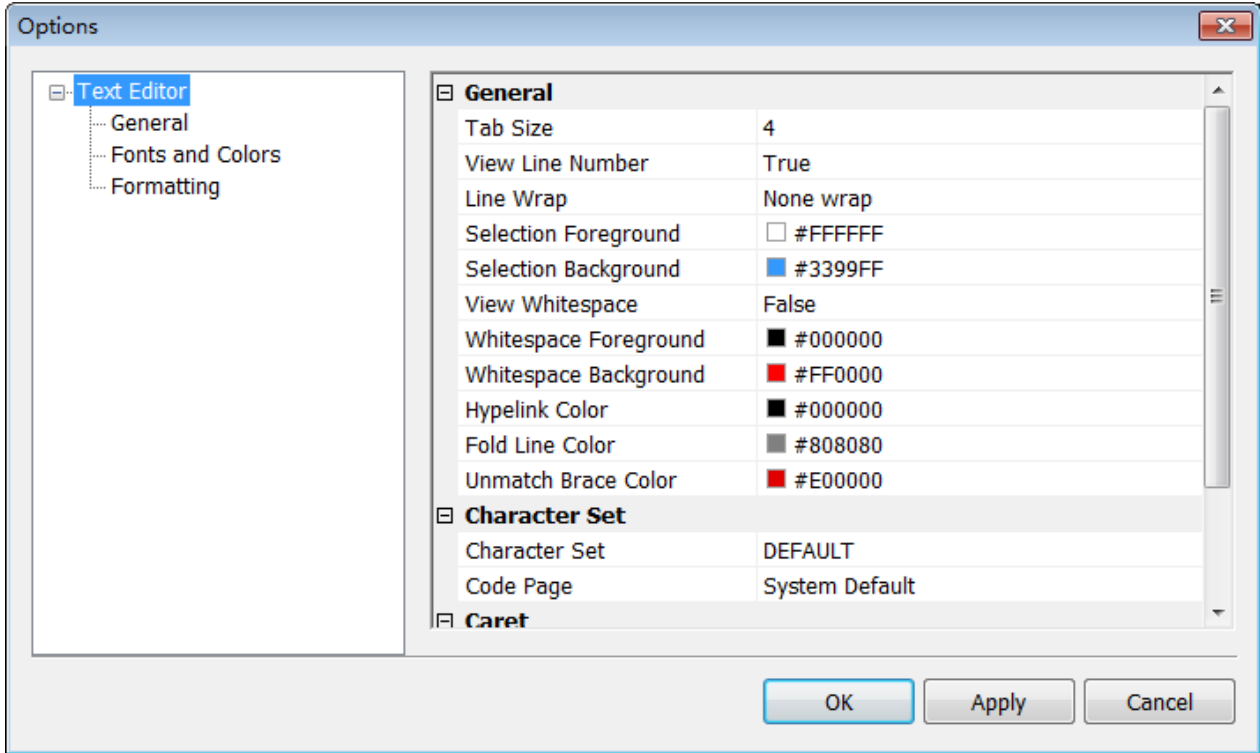
注：如果蓝牙 bin 文件不存在，执行编译时，将会在 output 窗口中弹出 error 信息，如下所示：



## 10 用户自定义

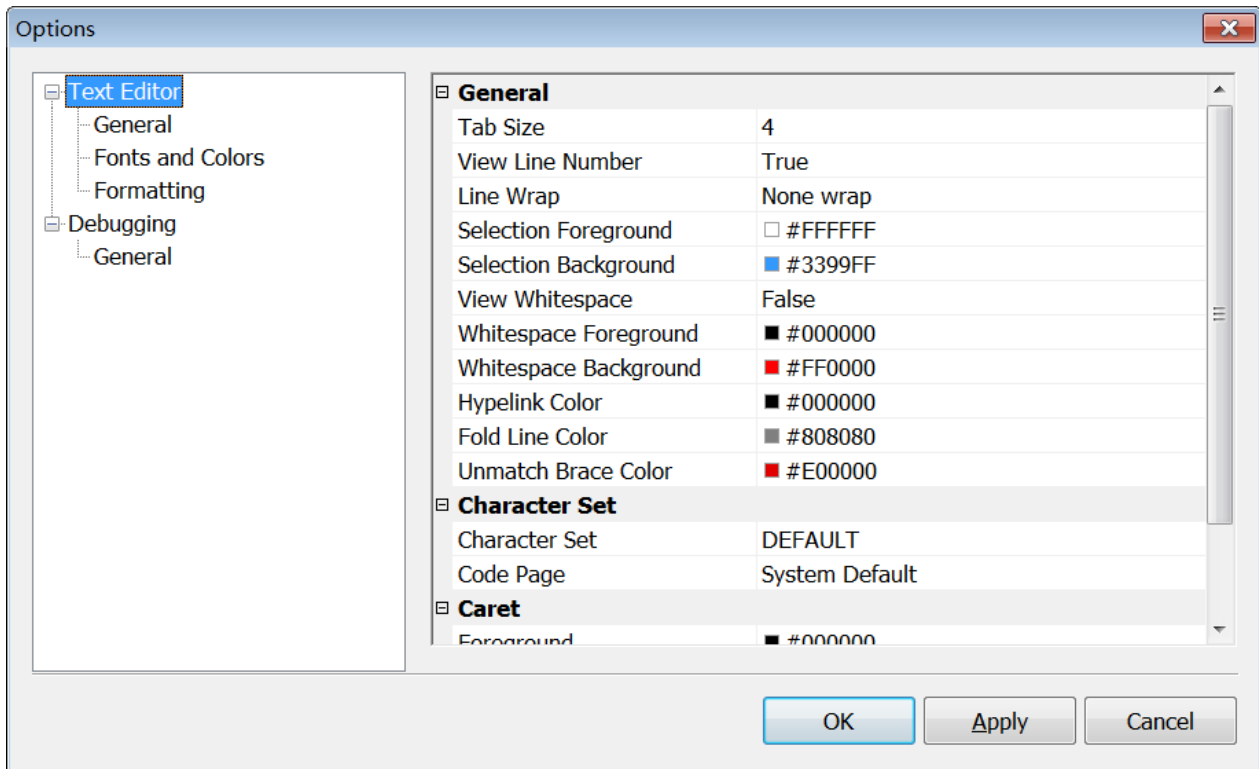
提供了用户自定义功能，用户可根据个人习惯来定义 IDE 的用户环境。

点击 **Tools | Options** 将会弹出“Options”对话框。



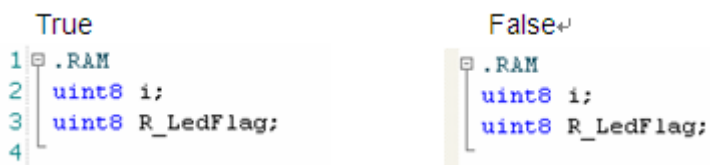
## 10.1 文本编辑设置 (TextEditor)

### 10.1.1 常用设置 (General)



Tab Size: 按“Tab”，将会跳几格，设置范围为 1~8。

View Line Number: 设为“True”，将会有行号显示。



Line Wrap

None wrap: 当一行的内容超过编辑器框后，不会换行。

According to word wrap: 当一行的内容超过编辑器框后，会从当前单词开始换行。

According to character wrap: 当一行的内容超过编辑器框后，从当前字符开始换行。

Foreground: 编辑器中选中文本的前景色。

Background: 编辑器中选中文本的背景色。

For example

Selection Foreground	#FFC0CB	nop
Selection Background	#FFFF00	goto \$-1

View Whitespace: 显示空格和“Tab”键的位置。

False	True
11 org 000h	11 org 000h
12 nop	12 nop
13 nop	13 nop
14 goto \$-1	14 goto \$-1
15	15
16 end	16 end

Whitespace Foreground: 空格和“Tab”键头的前景色。

Whitespace Background: 在空格和“Tab”键位置的背景色，参考上图。

Hypelink Color: 设置超链接的下划线颜色，如下图：

```
http = "http://www.chipsea.com"
```

Fold Line Color: 设置代码折叠线的颜色

```
1 |-----|
2 | filename: PrjType1_16_4.asm
3 | chip : CSU18M83
4 | author :
5 | date : 2016-07-07
6 | http = "http://www.chipsea.com"
7 |-----|
```

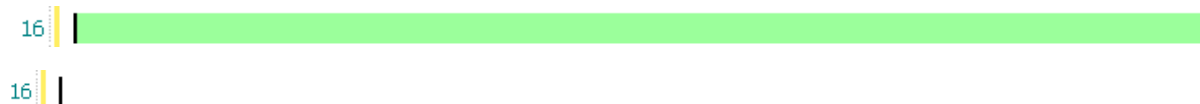
Unmatch Brace Color: 设置括号不匹配时，所显示的颜色

Character Set: 用来设置字符集

Caret

Foreground: 设置光标的颜色

Line Hilight: “True”时，光标所在行将高亮显示，对比下图。

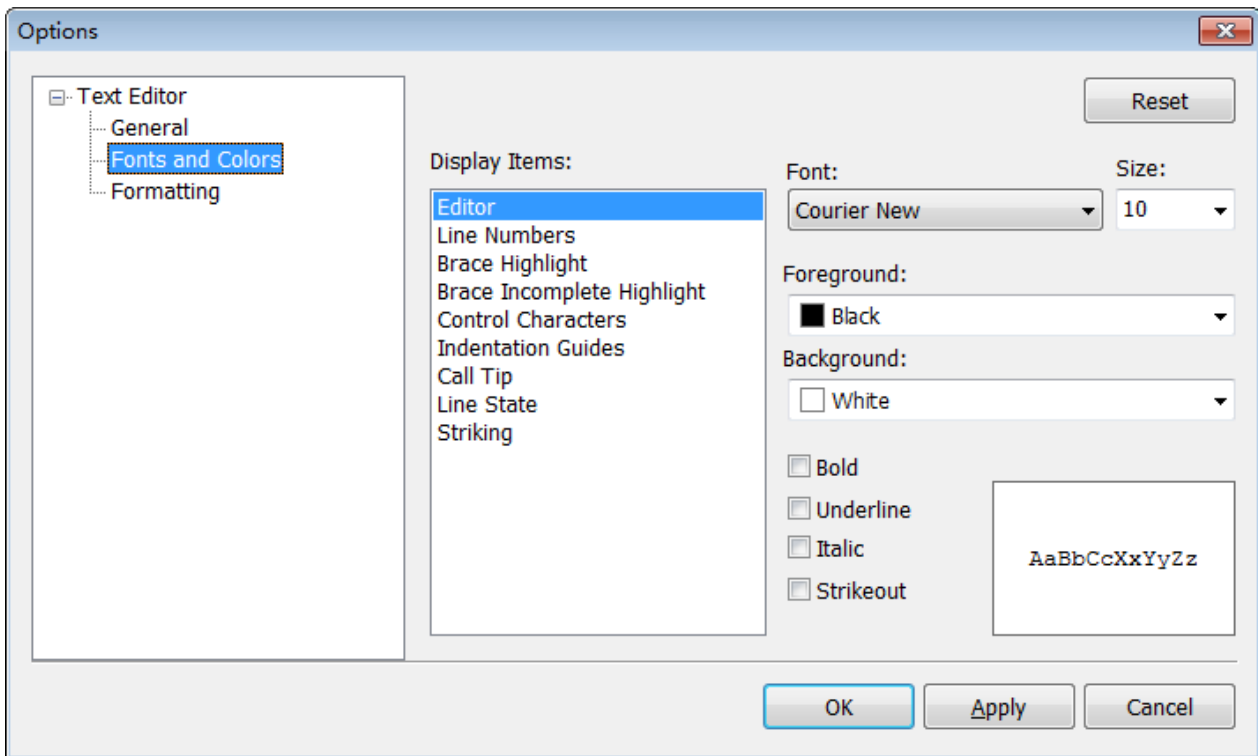


Line Background: 设置光标高亮时的背景色。

Line Background Alpha: 设置光标高亮时的透明度。

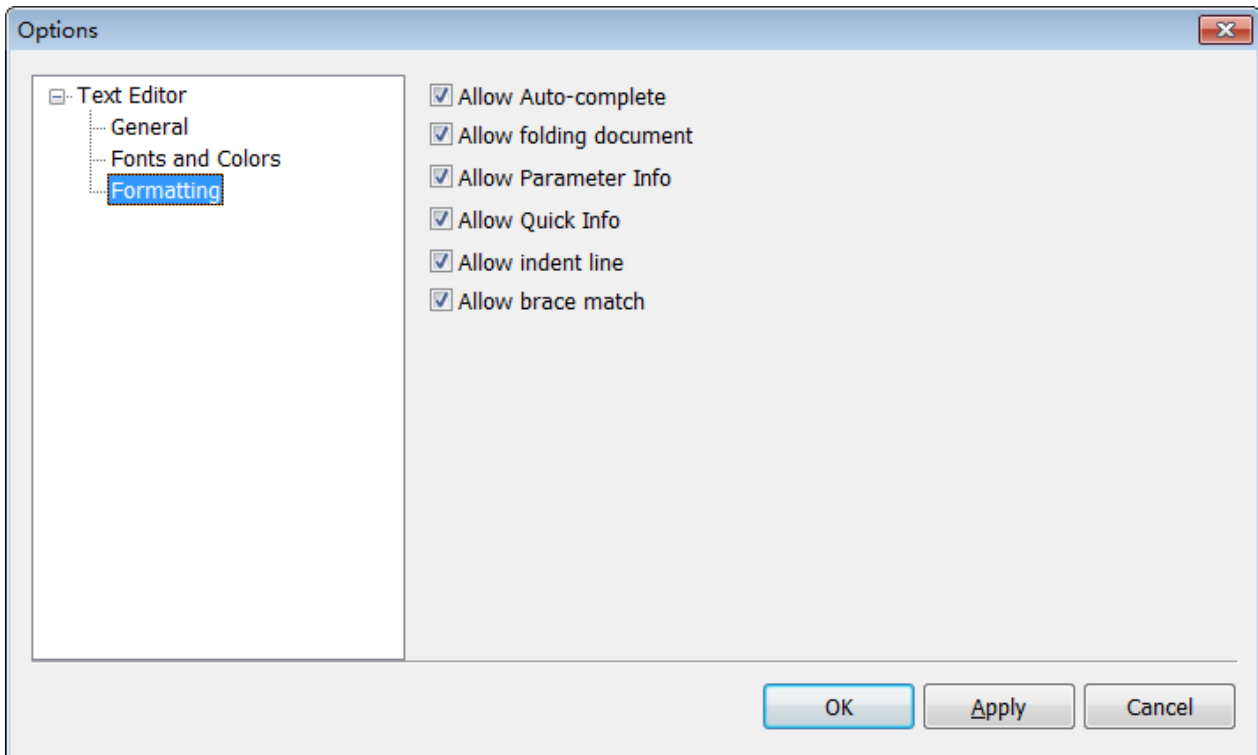
## 10.1.2 字体和颜色设置 (Fonts and colors)

提供了不同的字体和颜色设置方式。



### 10.1.3 格式定义 (Formatting)

提供了不同的字体和颜色设置方式。



Allow Auto-complete: 是否允许自动完成

Allow folding document: 是否允许折叠文档



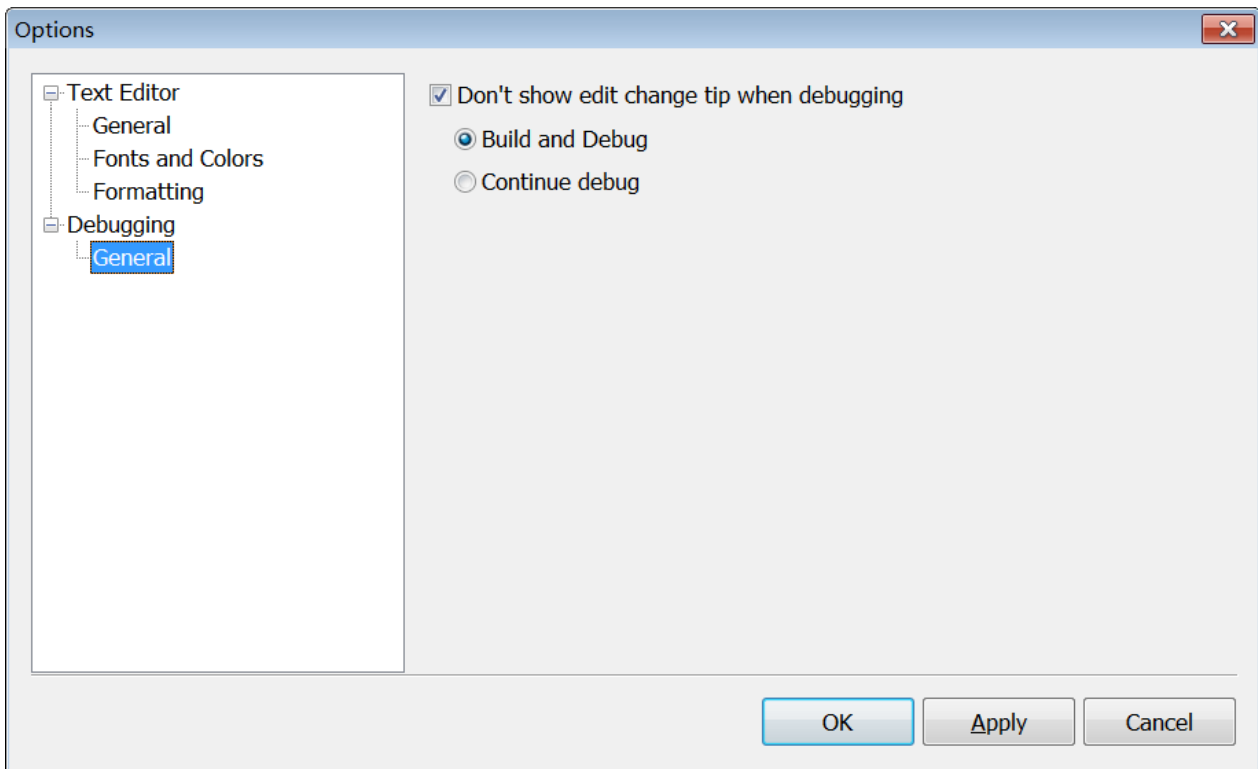
Allow Parameter Info: 是否允许显示参数信息

Allow Quick info: 是否允许快速显示关键字消息

Allow indent line: 是否允许行缩进

Allow brace match: 是否允许括号匹配

## 10.2 调试设置 (Debugging)



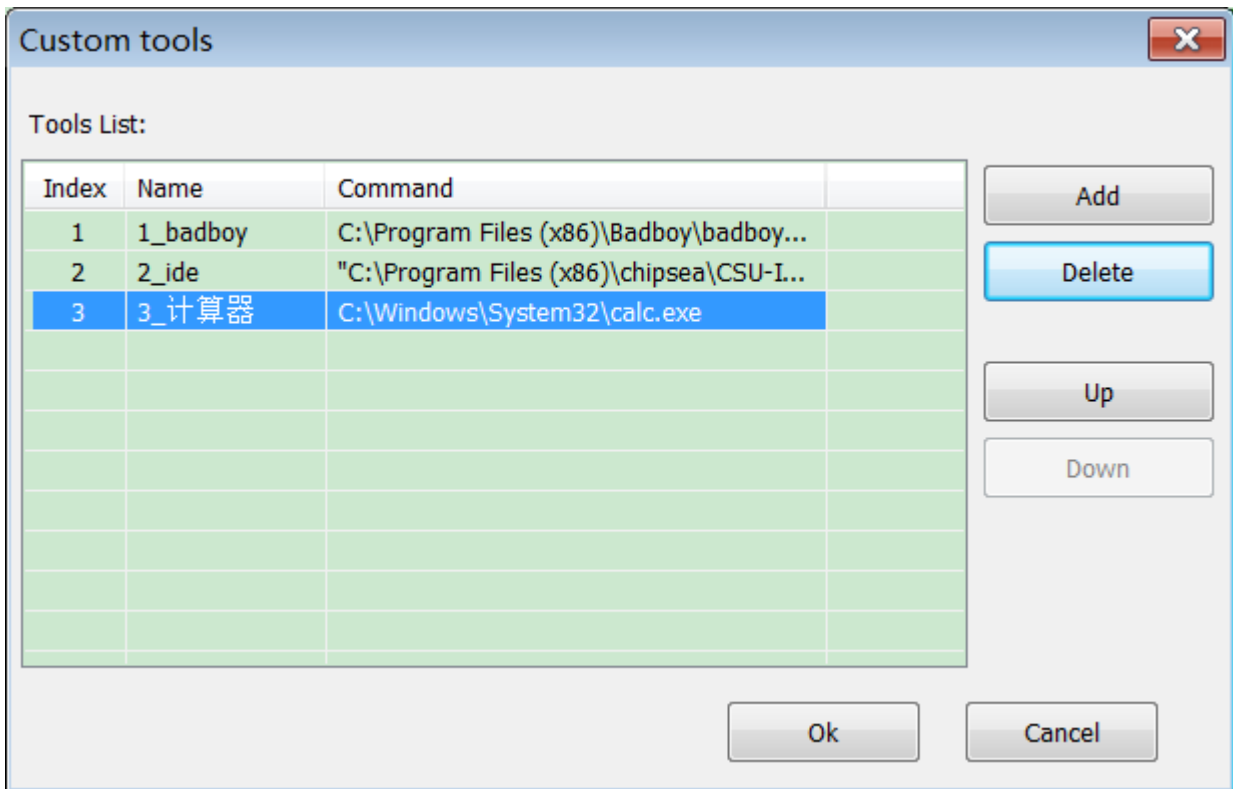
**Don't show edit change tip when debugging:** 在调试状态时，修改工程文件内容，是否需要弹出提示。不勾选该选项，则保持和以前版本一致，会继续弹出提示信息；勾选则不弹出提示，并依据选项【Build and Debug】或【Continue debug】，执行操作。

**Build and Debug:** 修改工程内容后，执行调试操作，会先执行 build 再进入 debug。

**Continue debug:** 修改工程内容后，执行调试操作，会继续调试。

## 10.3 工具快捷菜单设置 (Custom Tools)

为了方便客户快速使用其他工具，提供了 Custom Tools 功能，用户可自定义菜单。点击 Tools-》Custom ，会弹出 Custom Tools 对话框。



Add: 点击 Add 添加菜单名及 exe 路径。

Delete: 删除已添加菜单。

Up、Down: 调整菜单列表位置。

## 11 菜单项介绍

菜单项包括“File, Edit, View, Project, Build, Debug, Window, Help”。下面简要描述了每个项目的功能。

<b>File</b>	新建、打开、保存、关闭解决方案或工程或文件等相关操作。
<b>Edit</b>	编辑和查找相关操作。
<b>View</b>	控制各个窗口的显示。
<b>Project</b>	工程管理等相关设置。
<b>Build</b>	工程编译相关设置。
<b>Debug</b>	下载和调试相关操作。
<b>Window</b>	对编辑窗口的操作，排列等。
<b>Help</b>	使用手册、版本信息等。

### 11.1 文件（File）

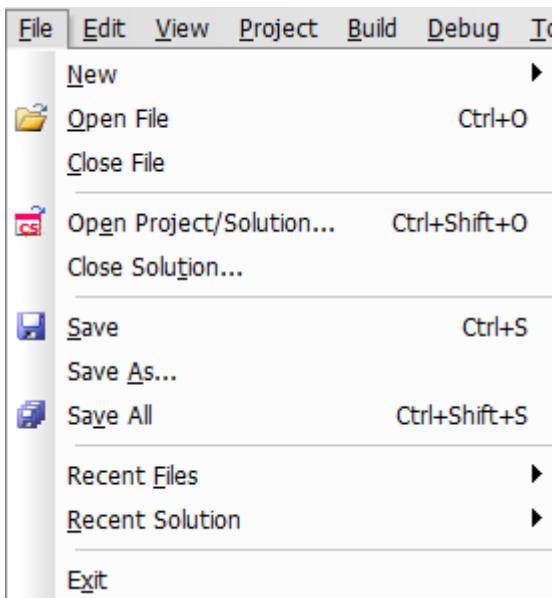


表 1 File 菜单

内容	作用	默认热键
New	(Project)新建工程	Ctrl+Shift+N Ctrl+N

	(File)新建文件	
	(Blank Solution)新建空的解决方案	
Open File	打开已经存在的文档	Ctrl+O
Close File	关闭编辑窗口中当前活动文档	
Open Project Solution...	打开已有解决方案	Ctrl+Shift+O
Close Solution...	关闭当前解决方案	
Save	保存编辑窗口中当前活动文档	Ctrl+S
Save As...	将编辑窗口中当前文档另存为	
Save All	保存所有文档及工程配置	Ctrl+Shift+S
Recent Files	列出最近打开文件，最多记忆 8 个	
Recent Solution	列出最近打开解决方案，最多记忆 8 个	
Exit	关闭 IDE	

## 11.2 编辑 (Edit)

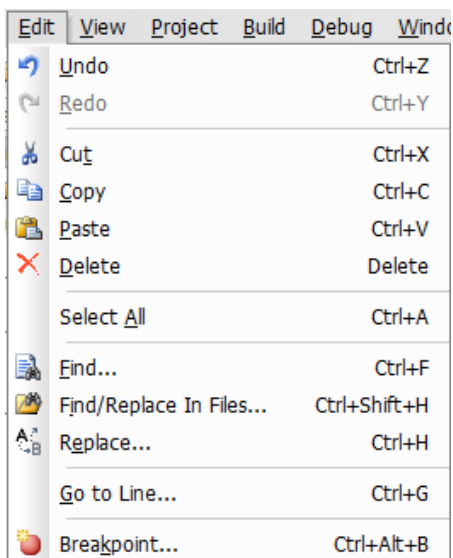


表 2 Edit 菜单

内容	作用	默认热键
Undo	撤销上次操作	Ctrl+Z
Redo	恢复被撤销的操作	Ctrl+Y
Cut	剪切	Ctrl+X
Copy	复制	Ctrl+C

Paste	粘贴	Ctrl+V
Delete	删除	Delete
Select All	选择整个文件内容	Ctrl+A
Find...	文本查找	Ctrl+F
Find/Replace In Files...	在指定的目录下进行文本查找和替换	Ctrl+Shift+H
Replace...	文本替换	Ctrl+H
Go to Line...	定位到指定行	Ctrl+G
Breakpoint...	设置断点	Ctrl+Alt+B

### 11.3 视图 (View)

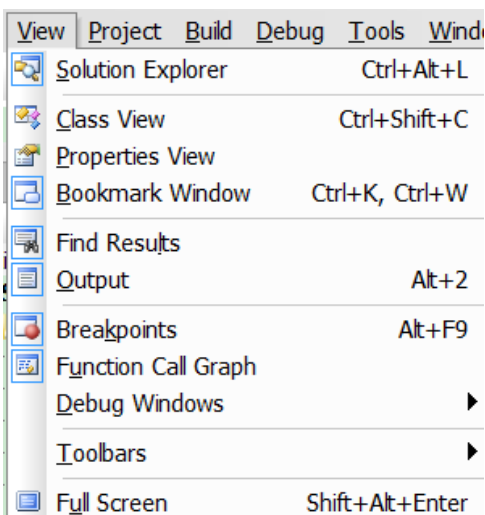


表 3 View 菜单

内容	作用	默认热键
Solution Explorer	显示解决方案管理器窗口	Ctrl+Alt+L
Properties View	显示属性窗口	F4
Bookmark Window	显示标签窗口	Ctrl+K, Ctrl+W
Find Results	显示查找结果窗口	
Output	显示输出窗口	Alt+2
Breakpoints	显示断点窗口	Alt+F9
Function Call Graph	显示函数调用图	
Debug Windows	显示调试窗口，包括：	

	(1)变量窗口	Alt+3
	(2) RAM(PAGE0)窗口	Alt+4
	(3) RAM(PAGE1)窗口	Alt+5
	(4)E2PROM 窗口	Alt+6
	(5)反汇编窗口	Alt+7
	(6)寄存器窗口	Alt+8
	(7)Rom 窗口	
Toolbars	显示或隐藏工具条。包括：	
	标准工具条	
	文本编辑工具条	
	调试工具条	
	编译工具条	
Full Screen	全屏显示	Shift+Alt+Enter

## 11.4 工程 (Project)

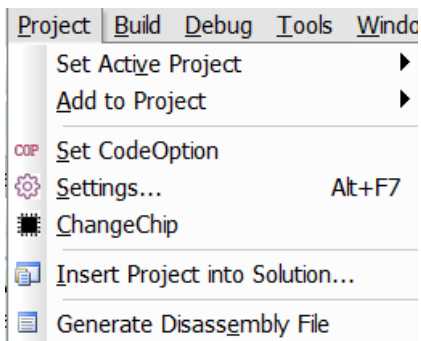


表 4 Project 菜单

内容	作用	默认热键
Set Active Project	设置选中工程为活动工作	
Add to Project	添加新文件、新文件夹或已存在的文件到工程中	
Set CodeOption	设置该工程的代码选项	
Settings...	工程设置	Alt+F7

Change Chip	更改该工程的芯片型号
Insert Project into Solution...	在当前解决方案中加入一个已存在的工程
Generate Disassembly File	生成反汇编文件

## 11.5 构建 (Build)

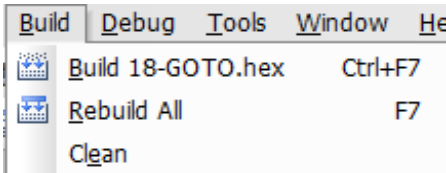


表 5 Build 菜单

内容	作用	默认热键
Build *.hex	构建工程	Ctrl+F7
Stop Build	停止构建	
Rebuild All	清除编译或重构过程中产生的文件，然后重新构建工程	F7
Clean	清除编译或重构过程中产生的文件	

## 11.6 调试 (Debug)

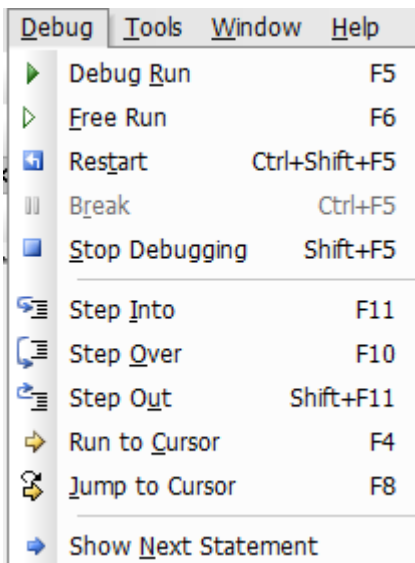
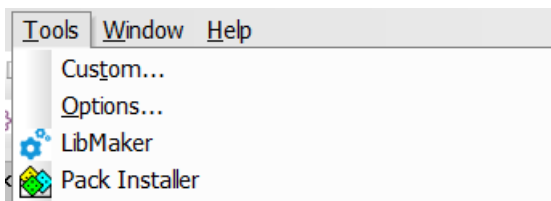


表 6 Debug 菜单

内容	功能	默认热键
Debug Run	下载程序：如果程序已经下载，则会运行程序，并遇到断点停下来。	F5
Free Run	全速运行程序，遇到断点不会停下来。	Shift+F5
Restart	重新运行程序，之前的运行结果将不被保留。	Ctrl+ Shift+F5
Stop Debugging	退出调试模式，转为编辑模式。	Ctrl +F5
Step Into	单步步入，可进入子程序	F11
Step Over	单步步过，不进入子程序	F10
Step Out	单步步出，可跳出子程序。	Shift+F11
Run to Cursor	程序全速运行到光标处停止。	F4
Jump to Cursor	直接将 PC 指针跳到新的位置，跳过一段程序后继续进行调试。因为跳过一段程序，执行结果可能和正常调试结果不一致，建议在十分清楚程序逻辑的时候使用。	F8
Show Next Statement	定位到 PC 指针所在行	

## 11.7 工具 (Tools)



内容	作用	默认热键
Custom	客户自定义功能菜单	
Options	定制开发环境，如热键的设置等	
Libmaker	制作 lib 库文件	
Pack Installer	安装 IC 型号安装包	



## 1.8 窗口 (Window)

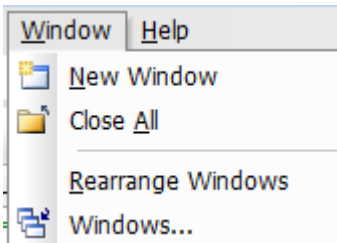


表 7 Window 菜单

内容	作用	默认热键
New Window	在编辑器中创建一个与当前活动文件完全相同的新文件	
Close All	关闭编辑器中所有打开的文件	
Rearrange Windows	重新排列窗口布局	
Windows	进行 windows 相关的设置	

## 11.9 帮助 (Help)

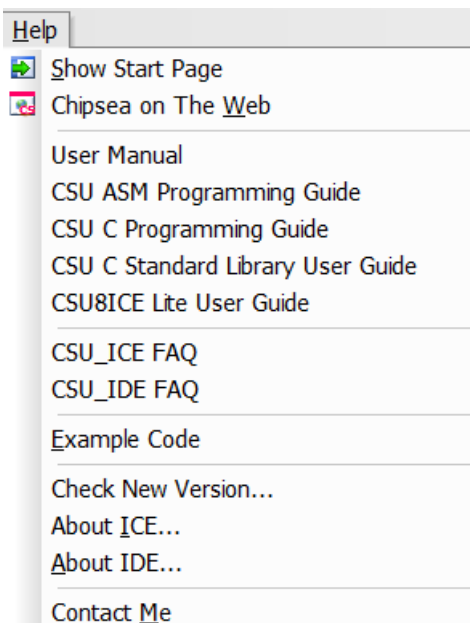


表 8 Help 菜单

内容	作用	默认热键
Show Start Page	在编辑器中显示起始页。	

Chipsea on the Web	在编辑器中打开公司网页。
User Manual	IDE 使用手册。
CSU ASM Programming Guide	CSU ASM 编程手册。
CSU C Programming Guide	CSU C 编程手册。
CSU C Standard Library Use Guide	C 标准函数库的使用手册
CSU8ICE Lite User Guide	仿真版使用手册。
CSU_ICE FAQ	CSU_ICE 快速问答手册
CSU_IDE FAQ	CSU_IDE 快速问答手册
Example Code	编程范例
Check New Version	检测官网是否有 IDE 和 ICE 的最新版本
About ICE	显示仿真程序版本信息。
About IDE	显示当前版本信息。
Contact Me	提供联系方式，可联系我们

## 12 工具栏介绍

IDE 在工具栏提供了 4 个工具条，分别是标准、文本编辑、调试和构建。每个工具栏存在三种状态：隐藏、停靠或者悬浮。

### 12.1 标准工具条 (Standard)



表 1 标准工具条



图标	内容	功能
	New File	新建文本
	Open	打开已有文本
	Save	保存当前文本
	Save All	保存所有文本及工程设置
	Cut	剪切
	Copy	复制
	Paste	粘贴
	Undo	撤销上次操作
	Redo	恢复被撤销的操作
	Find in Files	在指定的目录下进行文本查找和替换

### 12.2 文本编辑条 (TextEditor)



表 2 文本编辑工具条

图标	内容	功能
----	----	----



	Decrease Indent	减小缩进
	Increase Indent	增加缩进
	Comment Line	添加行注释
	Uncomment Line	删除行注释
	Toggle Bookmark	创建/删除一条书签
	Previous Bookmark	指向书签窗口中的上一条书签
	Next Bookmark	指向书签窗口中的下一条书签
	Previous Bookmark in Current Document	指向当前文本中的上一条书签
	Next Bookmark in Current Document	指向当前文本中的下一条书签
	Disable all Bookmark	禁用所有书签
	Clear all Bookmark	删除所有书签

## 12.3 构建工具条 (Build)



表 3 构建工具条

图标	内容	作用
Release	Select Active Configuration	点击下拉框选择所需的工程配置
	Compile	编译当前文件
	Build	构建工程
	Rebuild All	清除编译或重构过程中产生的文件，然后重新构建工程
	Stop Build	停止构建
	Insert/Remove HardWare Breakpoint	添加/删除硬件断点
	Remove All Breakpoint	删除所有断点
	Setting	打开选中工程的 Setting 对话框

	Set CodeOption	打开选中工程的代码选项框
	Select SDK function	打开 SDK 选择框

## 12.4 调试工具条 (Debug)



表 4 调试工具条

图标	内容	作用
	DebugRun	下载程序; 如果程序已经下载, 则会运行程序, 并遇到断点停下来。
	FreeRun	全速运行程序, 遇到断点不会停下来。
	Break	暂停程序运行
	Stop Debugging	退出调试模式, 转为编辑模式。
	Restart	重新运行程序, 之前的运行结果将不被保留。
	Step Into	单步步入, 可进入子程序
	Step Over	单步步过, 不进入子程序
	Step Out	单步步出, 可跳出子程序。
	Run to Cursor	程序全速运行到光标处停止。
	Jump to Cursor	直接将 PC 指针跳到新的位置, 跳过一段程序后继续进行调试。因为跳过一段程序, 执行结果可能和正常调试结果不一致, 建议在十分清楚程序逻辑的时候使用。
	LCD Simulator	显示 LCD 模拟器窗口
	Set Bluetooth Package	显示蓝牙设置对话框
	Debug Windows	点击弹出调试窗口
	Open corresponding file	在编辑区切换同名的 h 和 c 文件。

## 13 USB 驱动安装

需要用到 FTDI USB 驱动的,可按照下面两种方式安装。

● 方法一:

- (1) 如果 PC 没有安装 FTDI USB 驱动,当插上仿真板时,会弹出<找到新的硬件向导>对话框。选中<从列表或指令位置安装(高级)(S)>,并执行下一步。

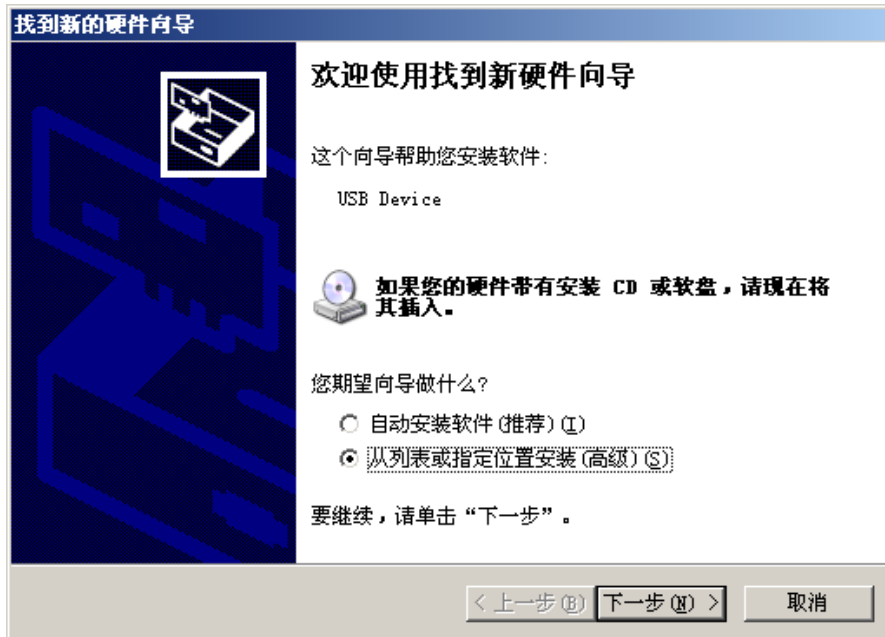


图 1.16 FTDI USB 安装图片 A

- (2) 在弹出来的窗口中,指定驱动程序的位置。点击<浏览>,选择合适 PC 机的 FTDI USB 驱动安装文件,并执行下一步完成 FTDI USB 驱动的安装。

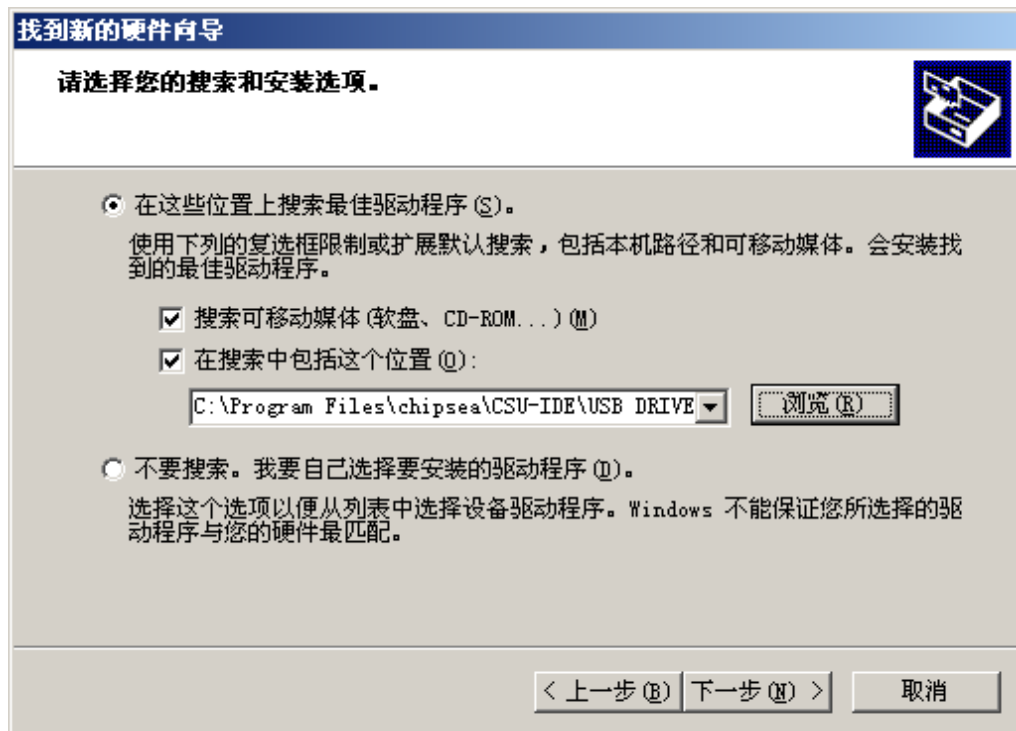


图 1.17 FTDI USB 安装图片 B



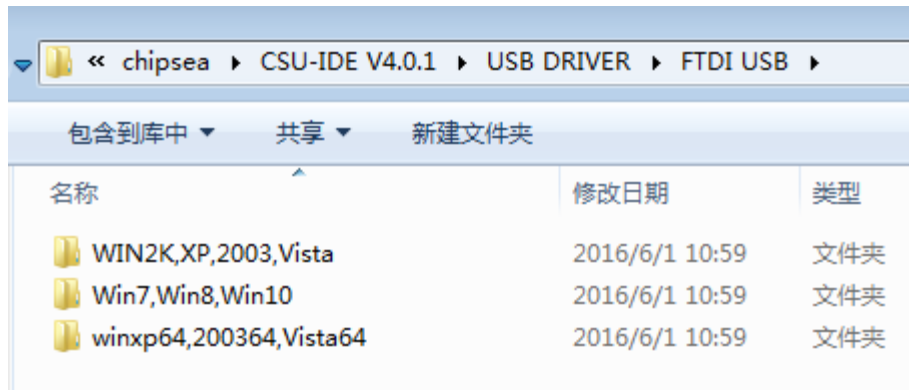
图 1.18 FTDI USB 安装图片 C

- 方法二:

- 1) 直接打开安装目录中的 USB DRIVER 文件夹

例如: C:\Program Files (x86)\chipsea\CSU-IDE V4.0.1\USB DRIVER\FTDI USB

- 2) 再依据当前操作系统, 选择匹配的驱动安装即可。

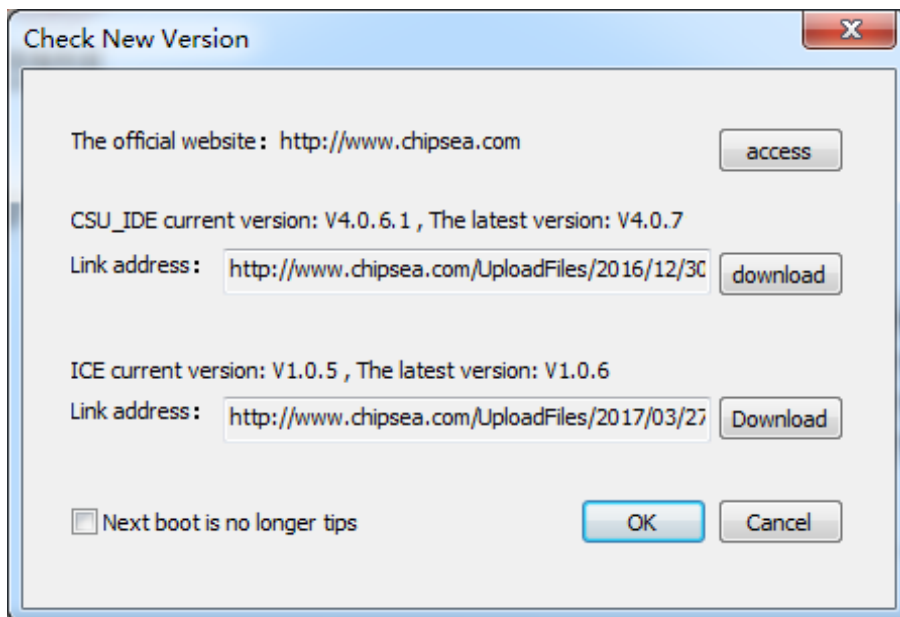




## 14 检测 IDE 和 ICE 最新版本

在互联网时，可检测在官网中是否有更新 IDE 和 ICE 的版本。

启动 IDE 时会自动检测官网中是否有高于当前 IDE 和 ICE 的版本，如果有，会弹出如下类似 Check New Version 对话框。



Check New Version 对话框也可通过菜单[Help]- Check New Version 打开。

Download:

- Download 为亮色，说明检测到官网中 IDE 或 ICE 的版本高于当前 IDE 或 ICE 的版本。点击 download 可下载 IDE 或 ICE 最新版本至指定目录。
- Download 为灰色，说明未检测到有新版本或网络未连接。

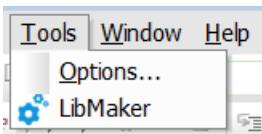
Next boot is no longer tips:

- 不勾选。在 IDE 启动时会自动检测官网中是否有高于当前 IDE 和 ICE 的版本。如果有更高版本，就会弹出 Check New Version 对话框。
- 勾选。在 IDE 启动时将不会自动检测，有更高版本也不会弹出提示；用户如需检测新版本，可通过菜单[Help]- Check New Version 来实现。

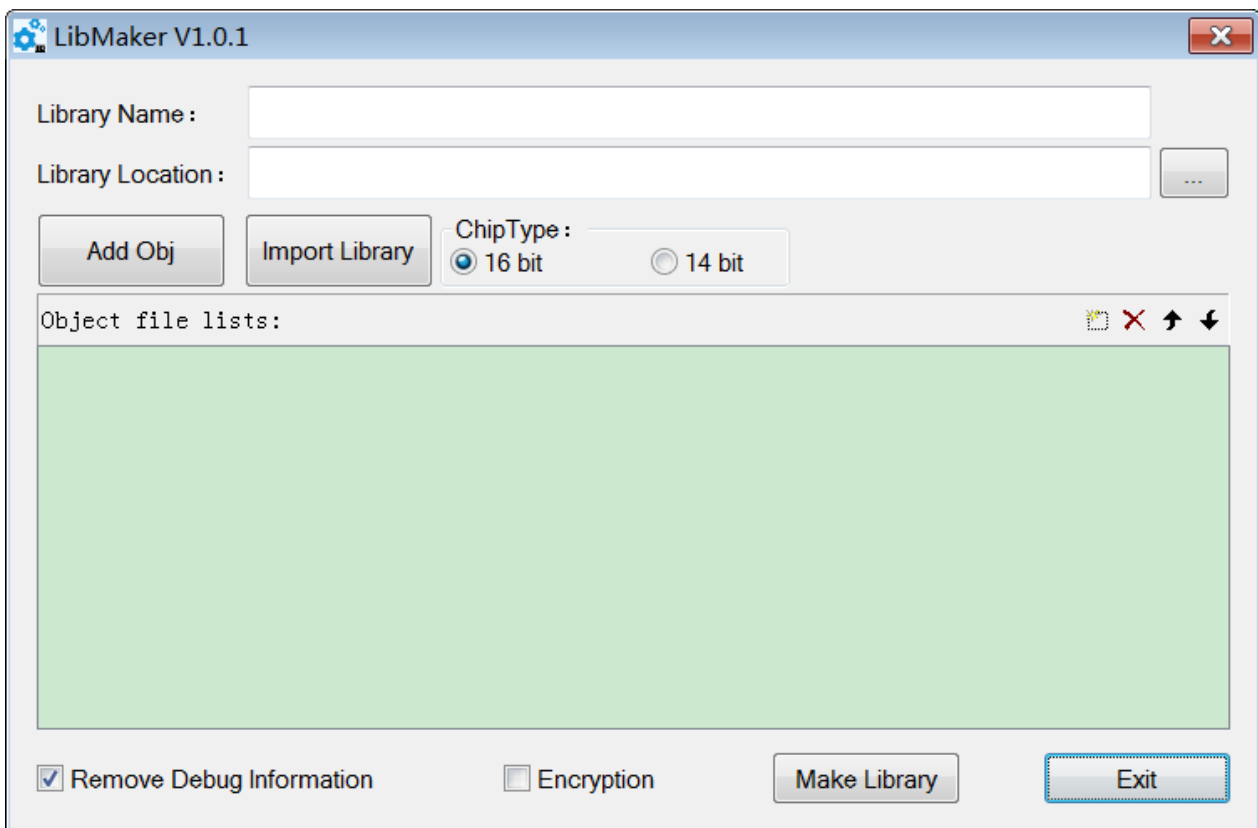
## 15 LibMaker

### 15.1 界面


点击【Tools】-【LibMaker】，如下图：



会弹出 LibMaker 对话框，如下图所示：



**Library Name:** 在输入框中输入需要创建的 lib 的名字。

**Library Location:** 可在输入框中输入 lib 放置的路径，也可通过  选择所需要的路径。

---

---

**Add Obj:** 可选择一个或多个 obj 文件（需选择与 ChipType 匹配的 obj 文件，否则会报错），将 Obj 文件添加至 Object file lists。

**Import Library:** 导入\*.lib 库，导入的库与 Chip Type 中选择的 bit 位需要保持一致，否则会提示 bit 位不一致。

**ChipType:** 选择生成的 lib 的 bit 类型。依据 IC 的类型分 14bit 和 16bit；在加入 obj 和 lib，及生成 lib 时，均要保持一致。

**Object file lists:** 显示文件列表信息。

**Remove Debug Information:** 生成的 lib 是否移除调试信息。生成的 obj 文件，有可能会带调试信息，勾选此项将会使生成的 lib 文件将不再带调试信息。

注：obj 是否包含调试信息，可在 setting->Compiler/Assembler 页面中通过 Generate Debug info File 选项控制

**Encryption:** 生成的 lib 是否加密。已加密后的文件，将不能再导入。

**Make Library:** 生成 lib 文件。

**Exit:** 关闭对话框。

## 15.2 基本操作步骤

**Step1:** 在 **Libray Name** 中自定义一个 lib 名字，建议使用字母、数字、下划线。

**Step2:** 在 **Library Location** 中指定 lib 存放路径。

**Setp3:** 选择匹配的 **Chip Type** 类型。

**Step4:** 点击 **Add obj**，在选择文件对话框中选择需要添加的 **obj** 文件。

**Step5:** 勾选 **Remove Debug Information**、**Encryption**

**Step6:** 点击 **Make Library** 生成 lib 文件，

---

---

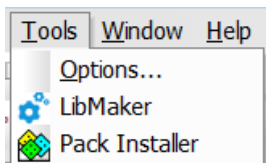
## 16 Pack Installer

---

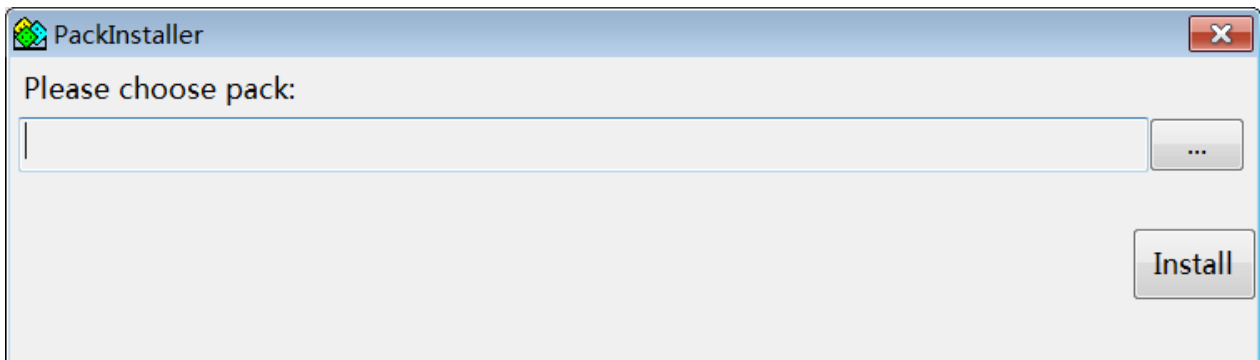
---

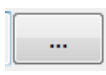
### 16.1 界面

点击【Tools】-【Pack Installer】，如下图：



会弹出 Pack Installer 对话框，如下图所示：



: 选择 IC 安装包路径，文件格式后缀为\*.csu8Pack

: 点击 Install，执行安装。

备注：只能安装版本号高的版本，已安装的版本不能重复安装。

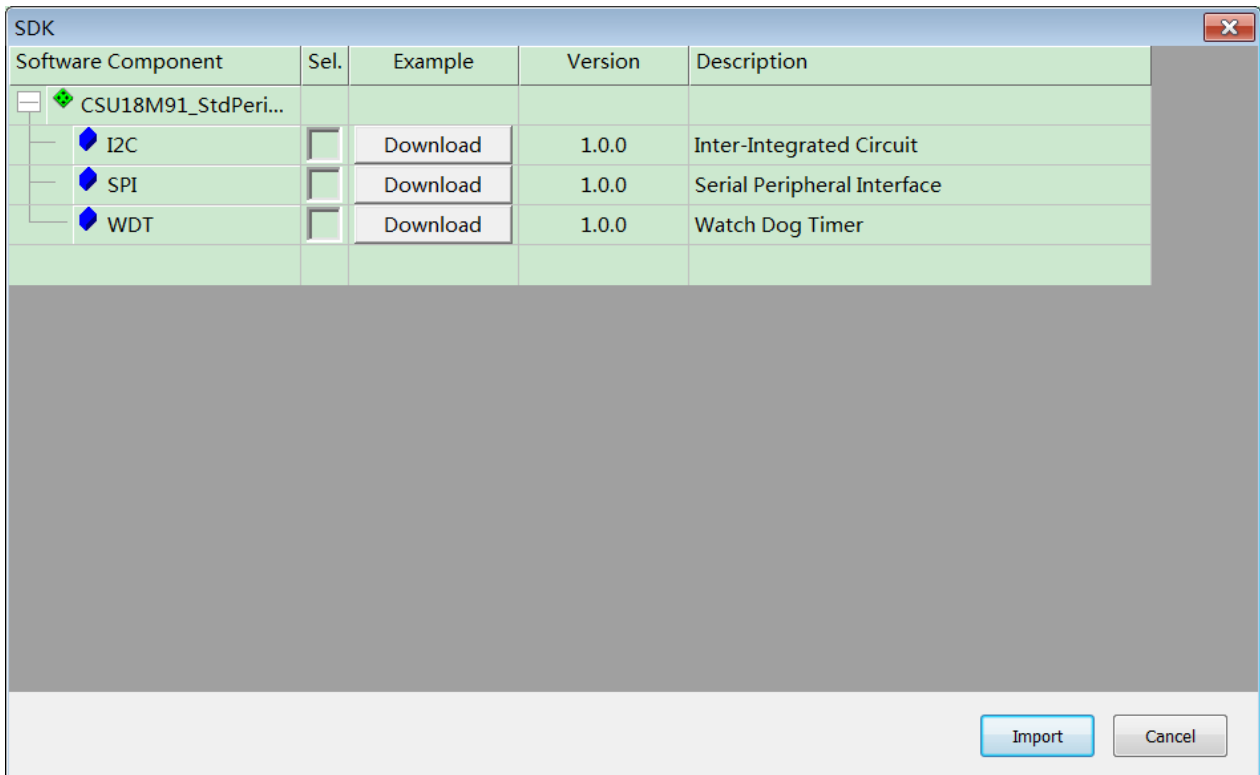
## 17 SDK

### 17.1 界面

点击 Build 工具栏中选择 SDK，如下图：



点击 SDK 图标，会弹出 SDK 选择对话框：



CSU18M91\_StdPerPeripherals: 支持 CSU18M91 的外围设备模块

Sel: 勾选后，并点击 Import，会将该模块的文件（模块名.c 和模块名.h）导入选择的工程内。

## 18 ICD

CSU\_IDE 在 V5.3.1 版本之后，可以支持 ICD 的芯片型号。

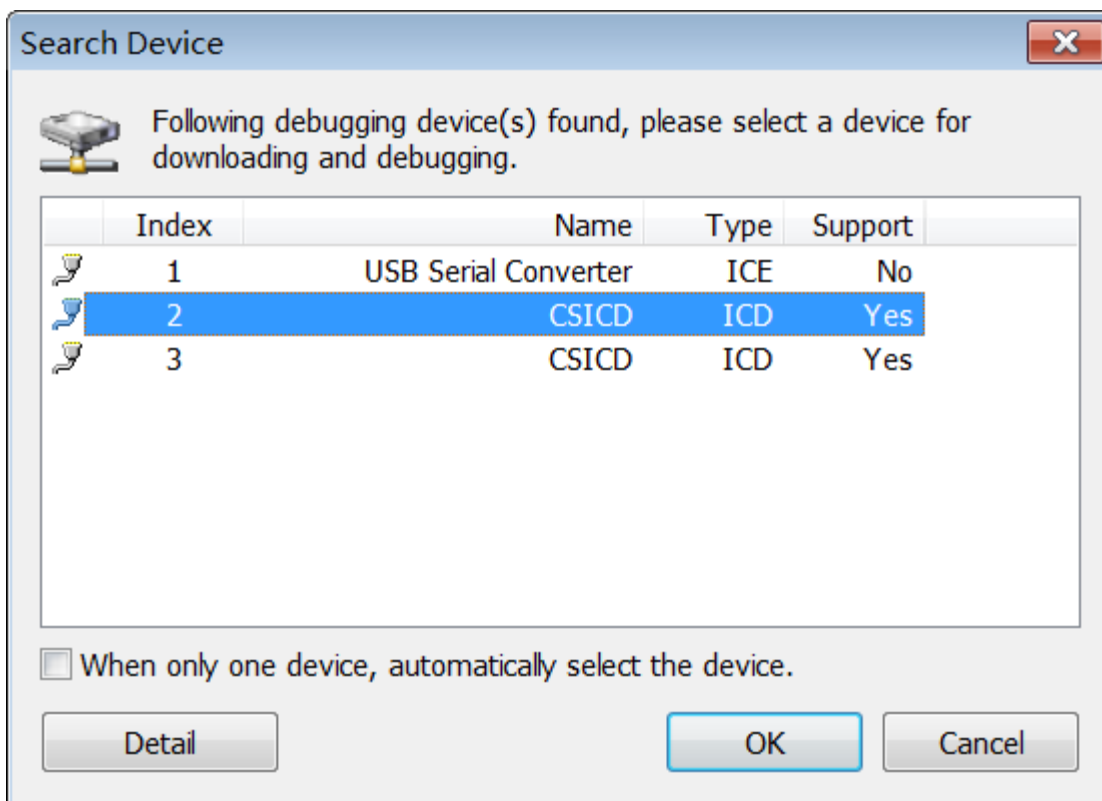
ICE 和 ICD 在 CSU\_IDE 中有以下几点不同，用户使用时需区分。

### 18.1 设备

IDE 中开启 ICD 的芯片型号的工程，连接 ICD 设备，点击下载后会列出设备信息，如下所示：

已连接 1 台 ICE 、2 台 ICD 的设备。需要使用 ICD 设备，则选择 CSICD。

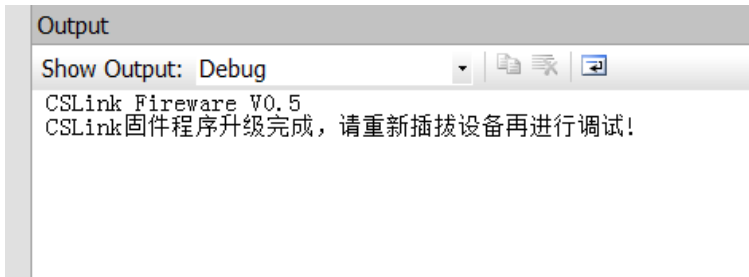
使用 ICD 设备，无需安装驱动。



#### 自动升级

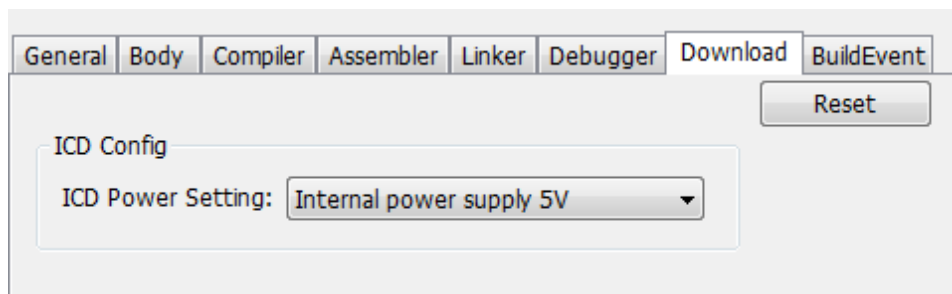
IDE 中提供的 ICD 版本号高于 CSLinker 设备的版本号，并且 CSLinker 设备固件程序为 V0.5 及以上版本，在 IDE 中进入调试时，会自动进行固件升级，升级完成后请重新插拔设备再进行调试。

输出窗口信息如下：



## 18.2 Setting 设置

在 Setting 中的 download 页面中：



Only Download Setting 功能与工具栏中的 Download hex file 配合使用。



Download hex file 功能：只下载，不进入调试状态。

ICD Power Setting 功能：可选择不同的供电方式，该功能是否有效与具体型号有关。

## 18.3 调试

在 IDE 中，因 ICD 调试和 ICE 调试使用的设备原理不同，提供的调试信息数据也会不同。

有以下几点区别：

- 1、在断点的支持个数不同。

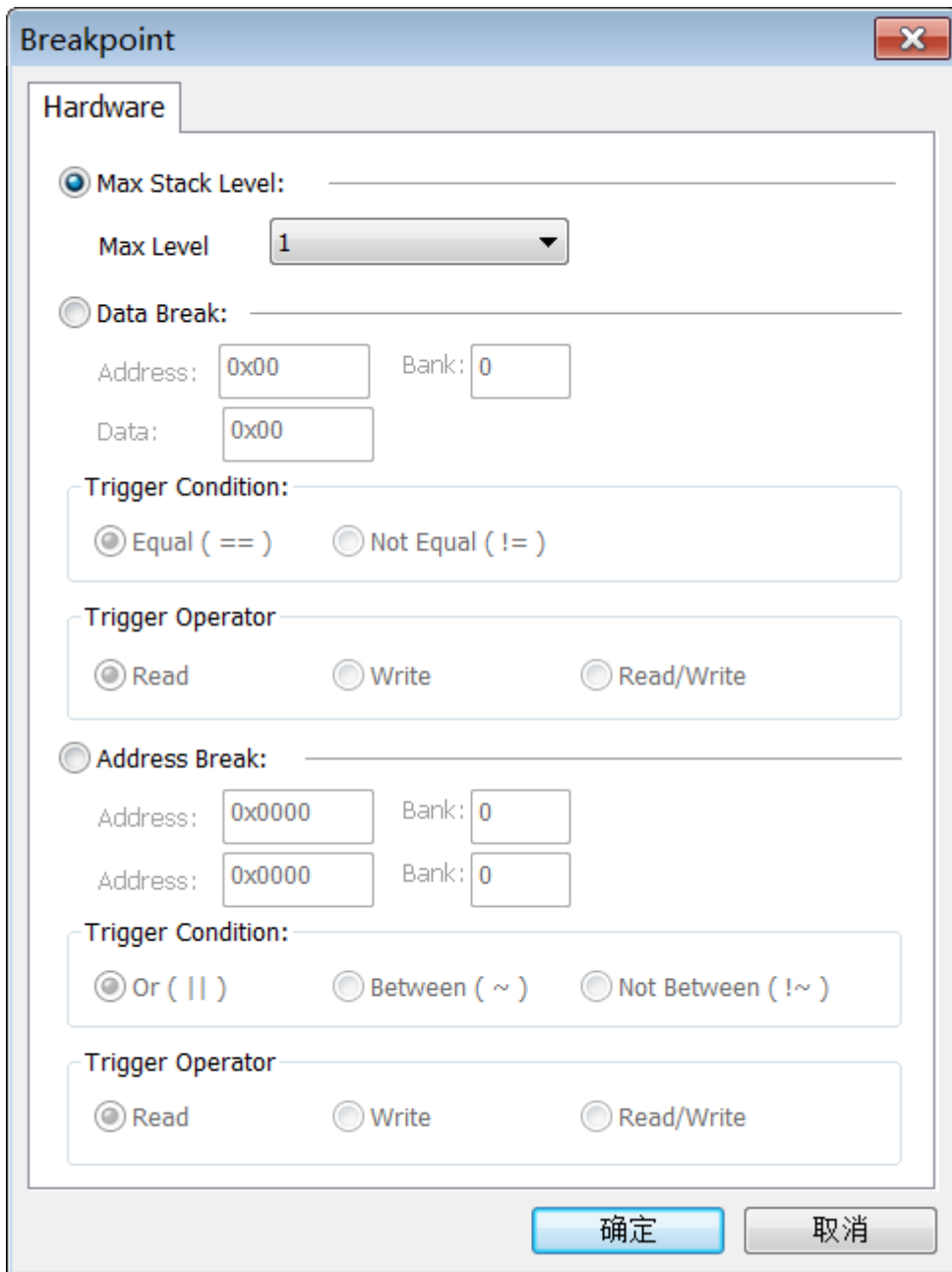
- 
- 2、 Trace Buffer 显示历史数据个数不同。
  - 3、 ICD 支持 ADC 模拟显示。
  - 4、 ICD 不支持在 RAM 窗口修改系统寄存器的值（SFR 地址 00-08h，但不包括 INTE/INTF 寄存器，INTE 中 GIE 不能修改），而 ICE 可以修改。

### 18.3.1 断点

ICD 支持 4 个指令断点、1 个数据断点。

数据断点窗口如下：





下面分别介绍每一项的作用：

**Instruction Break**：设置指令中断，在 Address 输入有效的指令地址。

**Max Stack Level**：可设置 1~8 级的栈，在当前调试函数的栈级别与设置的 Max Stack Level 相同时，断点有效。

**Data Break**：设置数据断点。在 Bank 和 Address 指定地址满足 Trigger Operation（Read、Write、Read/Write）条件时，并将值与 Data 中的值进行比较，符合 Trigger Condition（Equal、Not Equal）时，将产生数据中断。

**Address Break**：同时监视一段范围内的地址值。在 Address 和 Bank 中指定的地址，在 Trigger Condition（Or、Between、not Between）指定的范围内，满足 Trigger Operation（Read、Write、Read/Write）操作时，将产生数据中断。

### 18.3.2 Trace Buffer 窗口

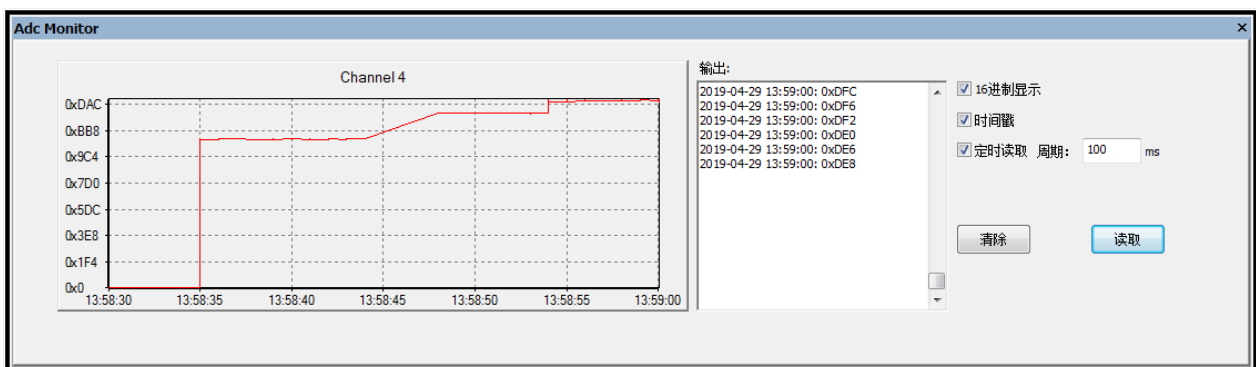
使用 ICD，在调试时，在 Trace buffer 窗口，可以查看最新的 1 条 PC 值运行记录，用户可通过记录追踪代码是否运行正常。

Index	PC	Code Bytes	Instruction
1	0x07E1	A521	GOTO 521H

### 18.3.3 ADC 窗口

在 ICD 中支持使用 ADC 实时波形查看功能，可将 ADC 寄存器数据映射到 ADC 监视区，以便于实时查看 ADC 数据。

在 Debug 工具栏中，点击 ADC，将弹出 ADC 窗口。



读取数据的方式：

- 1) 手动读取，每次点击读取按钮后获取一次 ADC 通道、码值信息和时间戳。
- 2) 定时读取，勾选【定时读取】，并设置周期时间（1-1000ms），点击读取后将定时获取一次通道、码值信息和时间戳。

截。

**显示内容:**

显示 ADC 波形，每次获取 ADC 码值后，添加当前码值到波形中，X 轴为时间，Y 轴为码值，码值有正负。选择部分图形区域可局部放大或缩放，便于用户查看。

显示 ADC 当前码值，码值可选择 16 进制显示和 10 进制显示。

**18.3.4 RAM 窗口**

ICD 不支持在 RAM 窗口修改系统寄存器的值（SFR 地址 00-08h，但不包括 INTE/INTF 寄存器，INTE 中 GIE 不能修改），而 ICE 可以修改。

