

## 3.1 CCS 使用实验

### 3.1.1 CCS 入门实验 1（CCS 使用）

#### 3.1.1.1 实验目的：

1. 熟悉 CCS 集成开发环境，掌握工程的生成方法；
2. 熟悉 SEED-DEC6713 实验环境；
3. 掌握 CCS 集成开发环境的调试方法。

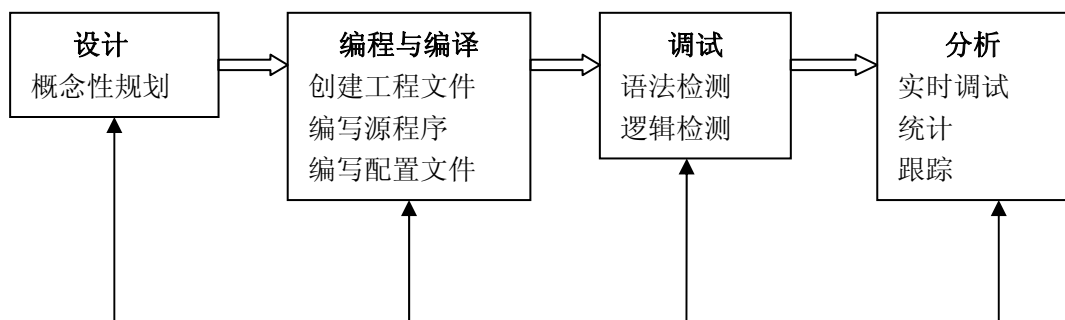
#### 3.1.1.2 实验内容：

1. DSP 源文件的建立；
2. DSP 程序工程文件的建立；
3. 学习使用 CCS 集成开发工具的调试工具。

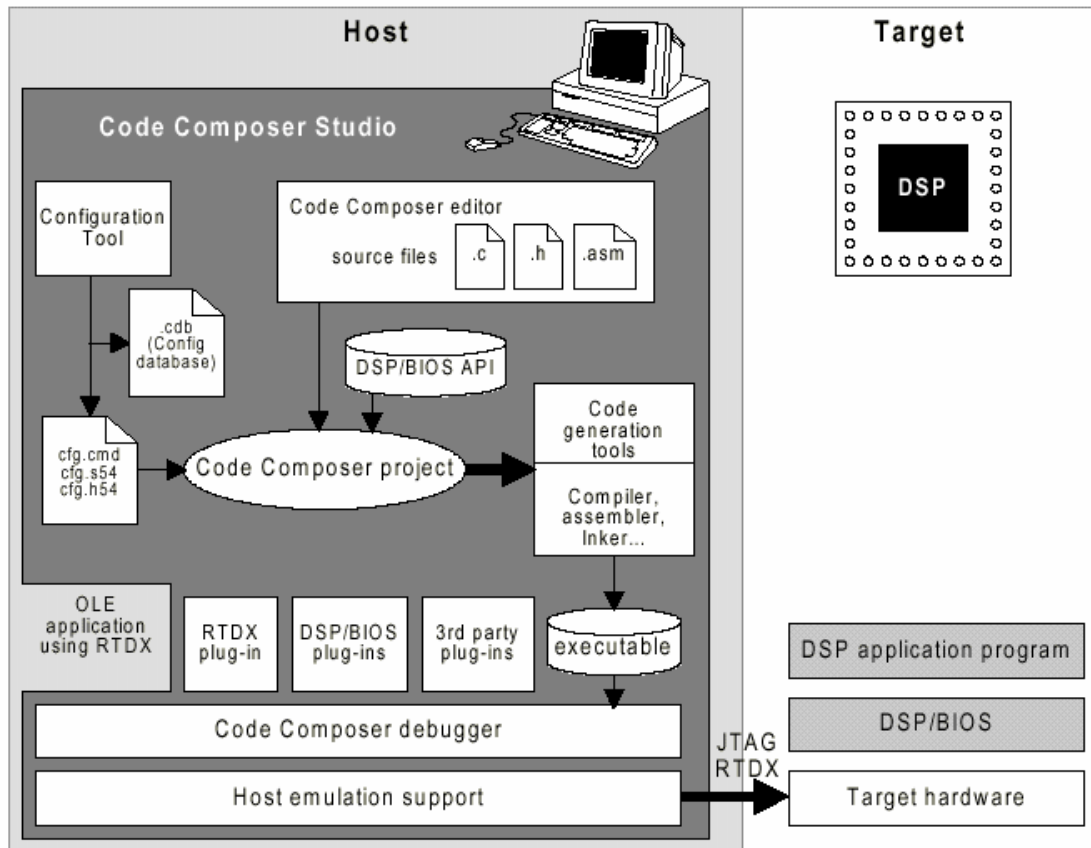
#### 3.1.1.3 实验背景知识：

##### 3.1.1.3.1 CCS 简介

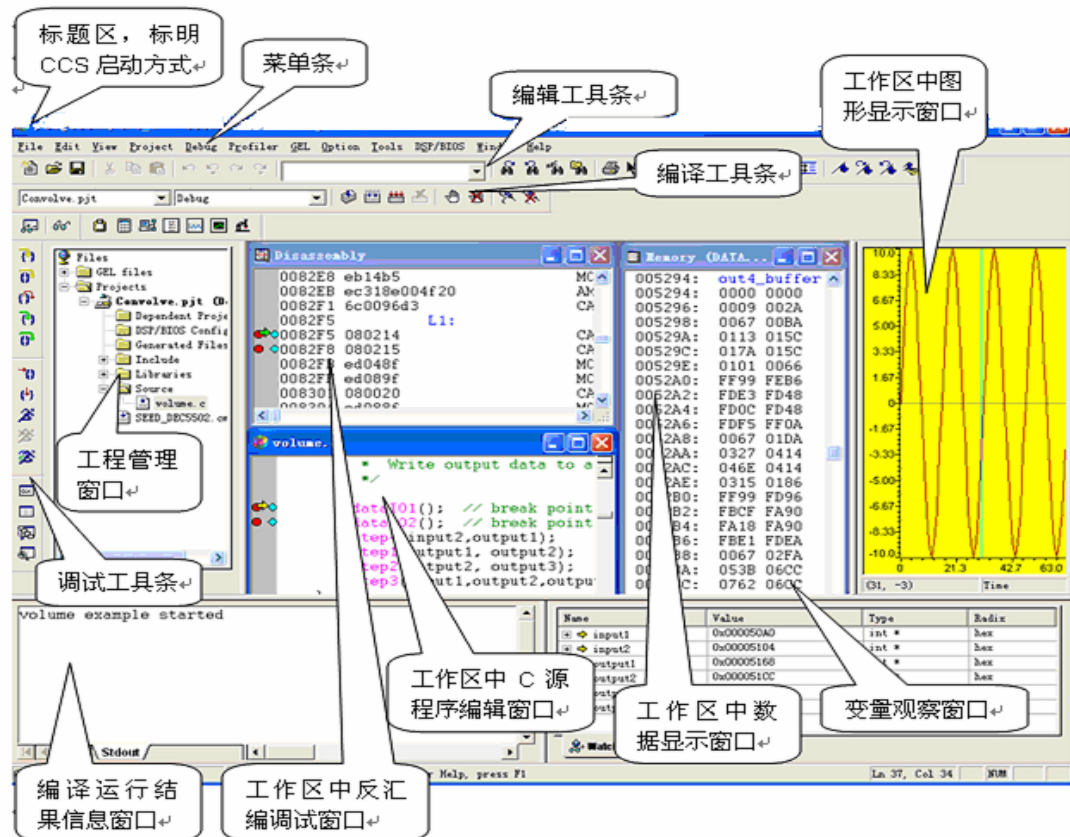
CCS 提供了配置、建立、调试、跟踪和分析程序的工具，它便于实时、嵌入式信号处理程序的编制和测试，它能够加速开发进程，提高工作效率。CCS 提供了基本的代码生成工具，它们具有一系列的调试、分析能力。CCS 支持如下所示的开发周期的所有阶段



CCS 构成及接口见下图



CCS窗口介绍:



### 3.1.1.3.2 使用 CCS 常遇见文件简介:

1. program.c: C 程序源文件
2. program.asm: 汇编程序源文件
3. filename.h: C 程序的头文件, 包含 DSP/BIOS API 模块的头文件
4. filename.lib: 库文件
5. project.cmd: 连接命令文件
6. program.obj: 由源文件编译或汇编而得的目标文件
7. program.out: 经完整的编译、汇编以及连接后生成可执行文件
8. program.map: 经完整的编译、汇编以及连接后生成空间分配文件
9. project.wkst: 存储环境设置信息的工作区文件

保存配置文件时将产生下列文件:

1. programcfg.cmd: 连接器命令文件
2. programcfg.h54: 汇编头文件
3. programcfg.s54: 汇编源文件

#### (一) CMD 文件简介

cmd 文件用于 DSP 代码的定位。由 3 部分组成:

1. 输入 / 输出定义:
  - .obj 文件: 链接器要链接的目标文件。
  - .lib 文件: 链接器要链接的库文件。
  - .map 文件: 链接器生成的交叉索引文件。
  - .out 文件: 链接器生成的可执行代码;链接器选项。
2. MEMORY 命令: 描述系统实际的硬件资源。
3. SECTIONS 命令: 描述"段"如何定位。

下面例子则可说明其基本格式:

```
-o sample.out
-m sample.map
-stack 100
sample.obj meminit.obj
-l rts.lib
MEMORY
{
    PAGE 0: VECT: origin = 0xff80, length 0x80
    PAGE 0: PROG: origin = 0x2000, length 0x400
    PAGE 1: DATA: origin = 0x800, length 0x400
}
SECTIONS
{
    .vectors : {} >PROG PAGE 0
    .text : {} >PROG PAGE 0
    .data : {} >PROG PAGE 0
    .cinit : {} >PROG PAGE 0
    .bss : {} >DATA PAGE 1
}
```

下面介绍一下 CMD 文件中常用的程序段名与含义

1. `.cinit` 存放 C 程序中的变量初值和常量;
2. `.const` 存放 C 程序中的字符常量、浮点常量和用 `const` 声明的常量;
3. `.text` 存放 C 程序的代码;
4. `.bss` 为 C 程序中的全局和静态变量保留存储空间;
5. `.far` 为 C 程序中用 `far` 声明的全局和静态变量保留空间;
6. `.stack` 为 C 程序系统堆栈保留存储空间, 用于保存返回地址、函数间的参数传递、存储局部变量和保存中间结果;

`.system` 用于 C 程序中 `malloc`、`calloc` 和 `realloc` 函数动态分配存储空间。

## (二) `vecs.asm` 文件简介

`vecs.asm` 是 DSP 的中断向量表文件。中断服务程序的地址(中断向量)要装载到存储器的合适区域。一般中断向量表文件是采用汇编语言编写; 在文件中一般汇编指令 `.sect` 来生成一个表。这个表包含中断向量的地址和跳转指令。因为中断读物的标志符在汇编语言模块外部使用, 所以标志符用 `.ref` 或 `.global`。

## (三) GEL 文件简介

GEL 文件的功能同 `cmd` 文件的功能基本相同, 用于初始化 DSP。但它的功能比 `cmd` 文件的功能有所增强, GEL 在 CCS 下有一个菜单, 可以根据 DSP 的对象不同, 设置不同的初始化程序。以下面的例子介绍一下 GEL 文件的构成。

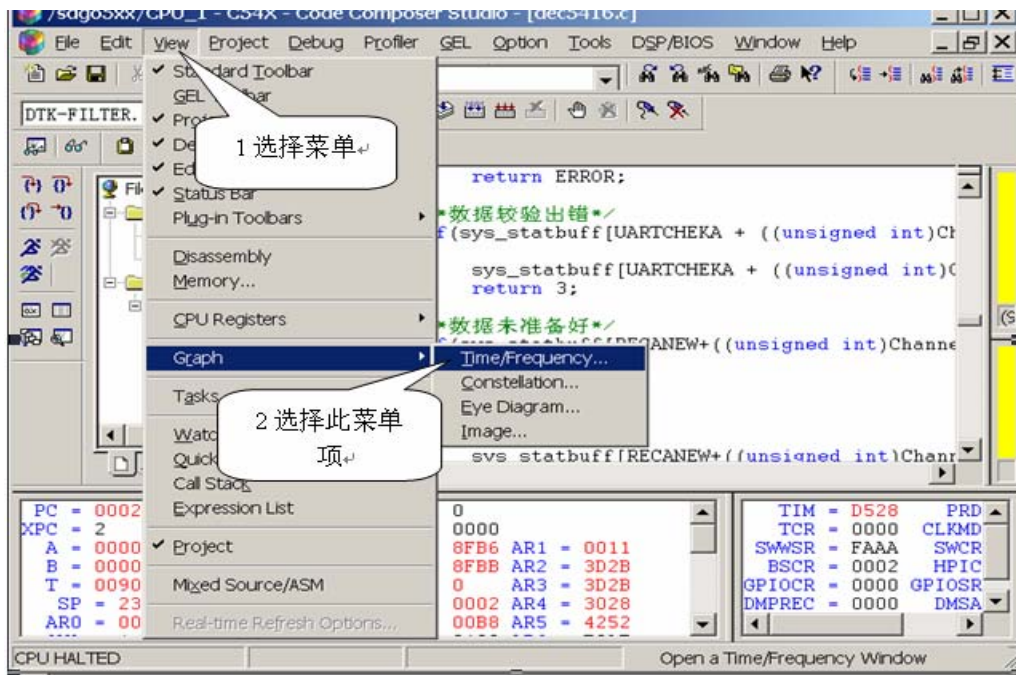
例:

```
#define DEC6713_CTL 0x60000 //定义 DEC6713_CTL 寄存器
#define DEC6713_INT 0x60001 //定义 DEC6713_INT 寄存器
#define DEC6713_STA 0x60002 //定义 DEC6713_STA 寄存器
StartUp(); 开始函数
{
    GEL_MapReset(); ; 存储空间复位
    GEL_MapAdd(0x0000,0,0x7fff,1,1); 定义程序空间 0000—7fff 可读写
    GEL_MapAdd(0x8000,0,0x7000,1,1); 定义程序空间 8000—f000 可读写
    GEL_MapAdd(0x0000,1,0x1000,1,1); 定义数据空间 0000—f000 可读写
    GEL_MapAdd(0xffff,2,1,1,1); 定义 I/O 空间 0xffff 可读写
    GEL_MapOn(); 存储空间打开
    GEL_MemoryFill(0xffff,2,1,0x40); 在 I/O 空间添入数值 40h
}
```

### 3.1.1.3.3 CCS 常用指令简介

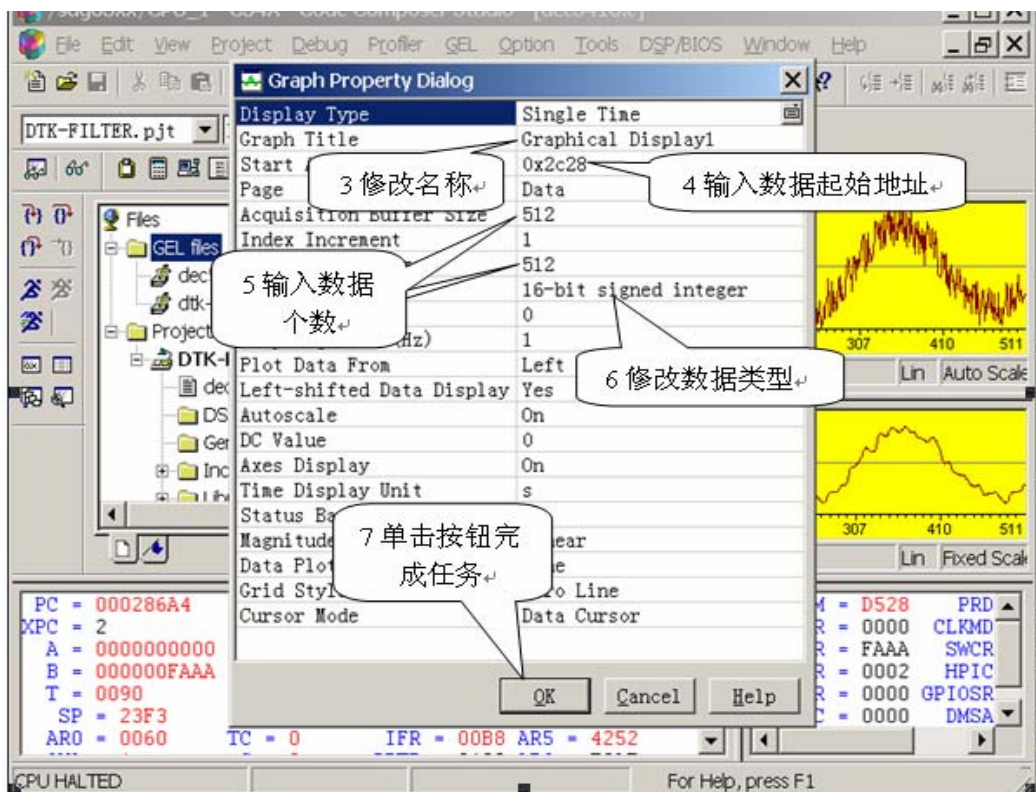
1. 设置断点。  
将光标放置在需要设置断点的程序行前, 选择 `Debug`→`Breakpoints`, 即完成可一个断点的设置。
2. CCS 提供 3 种方法复位目标板
  - 1) `Reset DSP`: `Debug`→`Reset D`, 初始化所有的寄存器内容并暂停运行中的程序。使用此命令后, 要重新装载 `.out` 文件后, 再执行程序。
  - 2) `Restart`: `Debug`→`Restart`, 将 PC 值恢复到当前载入程序的入口地址。
  - 3) `Go main`: `Debug`→`Go main`, 将程序运行到主程序的入口处暂停。
3. CCS 提供 4 种执行操作

- 1) 执行执行: Debug → Run , 程序运行直到遇到断点为止。
  - 2) 暂停执行: Debug → Halt , 程序停止运行。
  - 3) 动画执行: Debug → Animate, 用户反复运行程序, 直到遇到断点为止。
  - 4) 自由执行: Debug → Run Free , 禁止所有断点运行程序。
4. CCS 提供 4 种单步执行操作
- 1) 单步进入: 快捷键 F8, Debug → step into , 当调试语句不是基本的汇编指令时, 此操作进入语句内部。
  - 2) 单步执行: Debug → step Over , 此命令将函数或子函数当作一条语句执行, 不进入内部调试。
  - 3) 单步跳出: Debug → step Out , 此命令作用为从子程序中跳出
  - 4) 执行到光标处: 快捷键 ctrl+F10, Debug → Run to Cursor, 此命令作用为将程序运行到光标处。
5. 内存、寄存器与变量的操作
- 1) 查看变量: 使用 view → Watch Window 命令
  - 2) 查看寄存器: 使用 view → Registers → CPU Registers 命令
  - 3) 查看内存: 使用 view → memory 命令
6. Graph 的设置即图形显示
- 1) 选择 View → Graph → Time/Frequency。

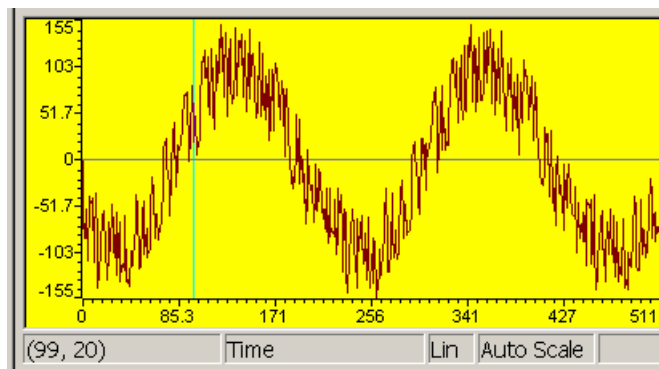


- 2) 在弹出的 Graph Property Dialog 对话框中, 将 Graph Title, Start Address, Acquisition Buffer Size, Display Data Size, DSP Data Type 等的属性可改变为如下图所示 (也可根据具体需要设置属性)。向下滚动右侧的滚动条或调整 dialog 框的大小可看到所有的属性。

示例(A) 将起始地址为 0x2c28 的数组进行图形显示



- 3) 点击 OK，将出现所设的图形窗口。如：在滤波实验中，用以上方法设定的图形窗口，在运行滤波程序后，最终的显示结果如下图所示



- 4) 可以在图形上单击右键，选择“Float In Main Table”，这时图形将浮现在主窗口中，以便观察。

#### 3.1.1.4 实验准备：

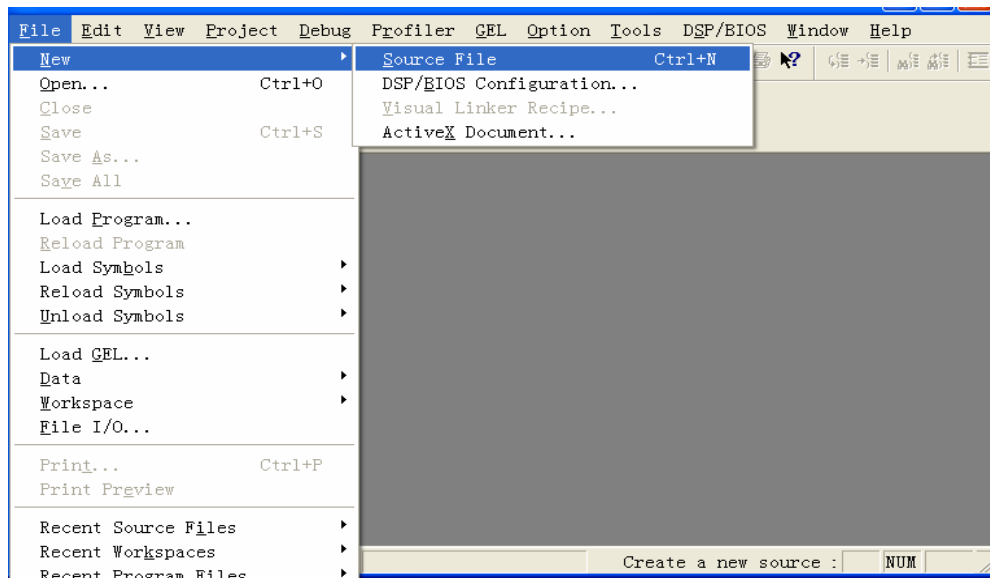
1. 将 DSP 仿真器与计算机连接好；
2. 将 DSP 仿真器的 JTAG 插头与 SEED-DEC6713 单元的 J2 相连接；
3. 启动计算机，当计算机启动后，打开 SEED-DTK6713 的电源。观察 SEED-DTK\_MBoard 单元的 +5V，+3.3V，+15V，-15V 的电源指示灯以及 SEED-DEC6713 的电源指示灯 D1、D3 是否均亮；若有不亮的，请断开电源，检查电源。

下面按照原文件、工程文件以及编译条件的设置来分别介绍一下 CCS 的使用。

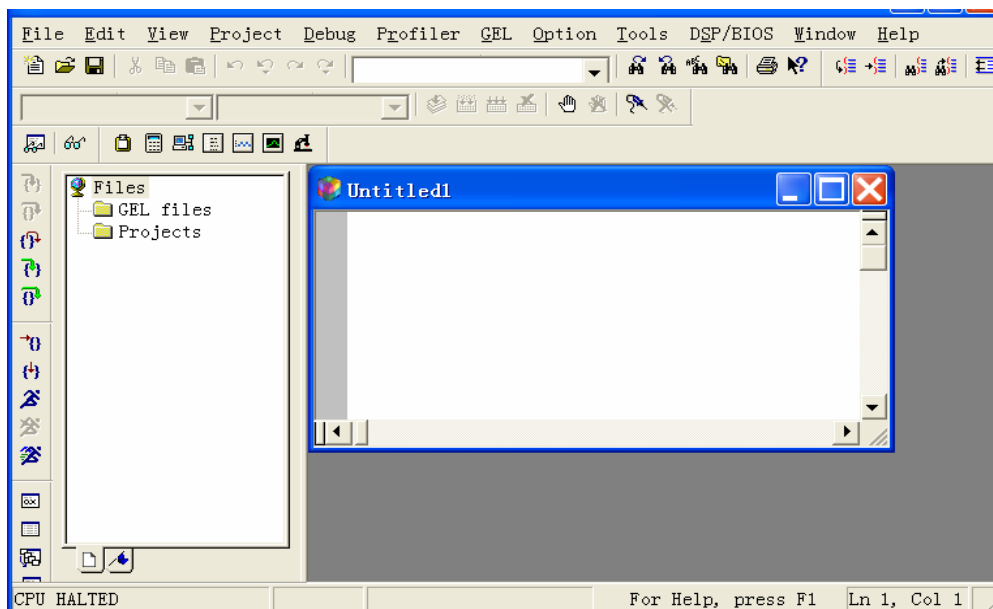
### 3.1.1.5 实验步骤:

#### 3.1.1.5.1 创建源文件

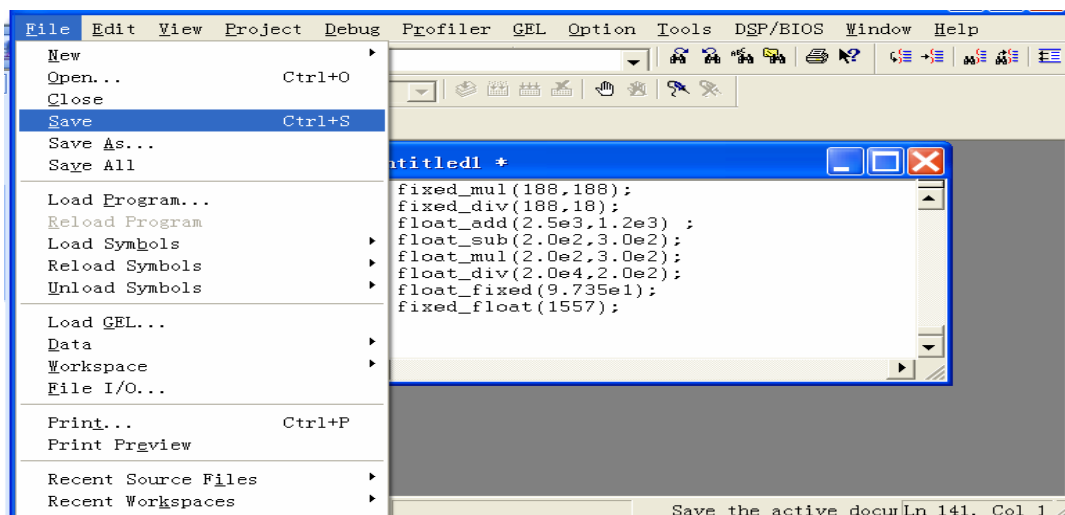
1. 双击  图标进入 CCS 环境。
2. 打开 CCS 选择 File → New → Source File 命令



3. 编写源代码并保存



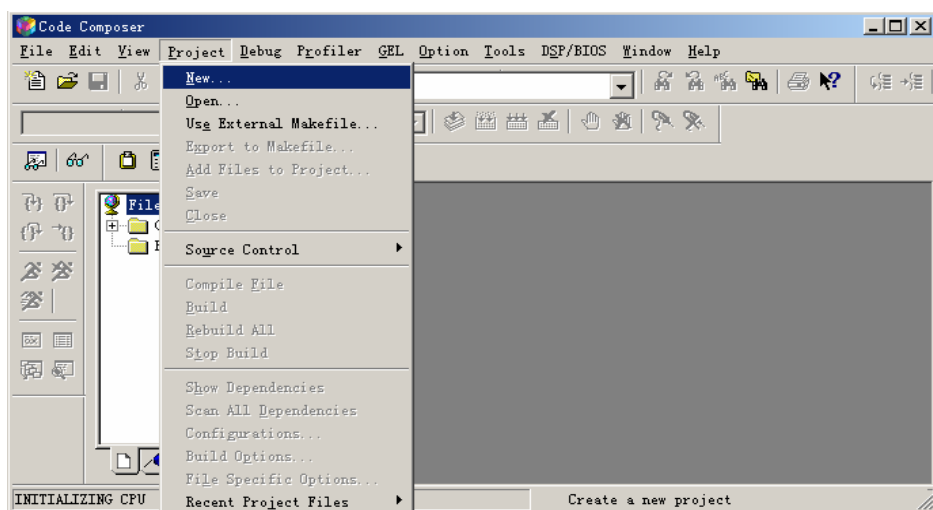
4. 保存源程序名为 math.c, 选择 File → Save



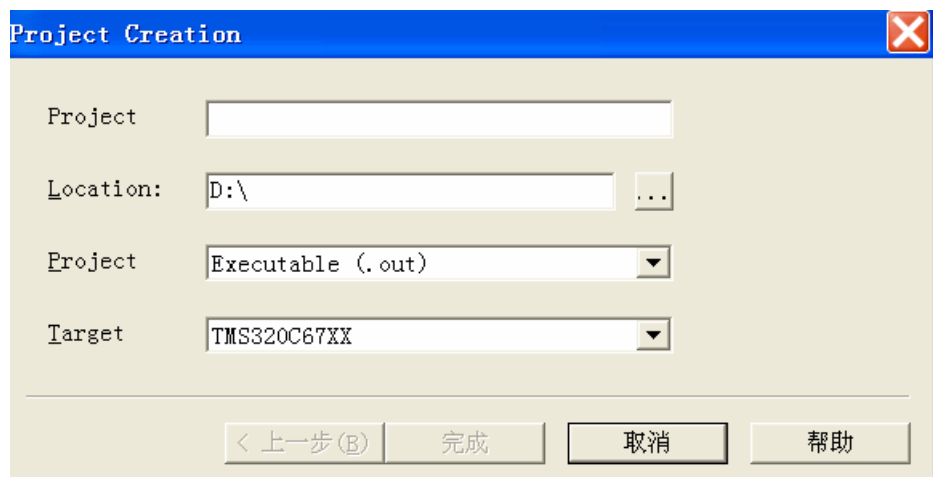
5. 创建其他源程序（如.cmd）可重复上述步骤。

### 3.1.1.5.2 创建工程文件

1. 打开 CCS，点击 Project-->New,创建一个新工程，其中工程名及路径可任意指定



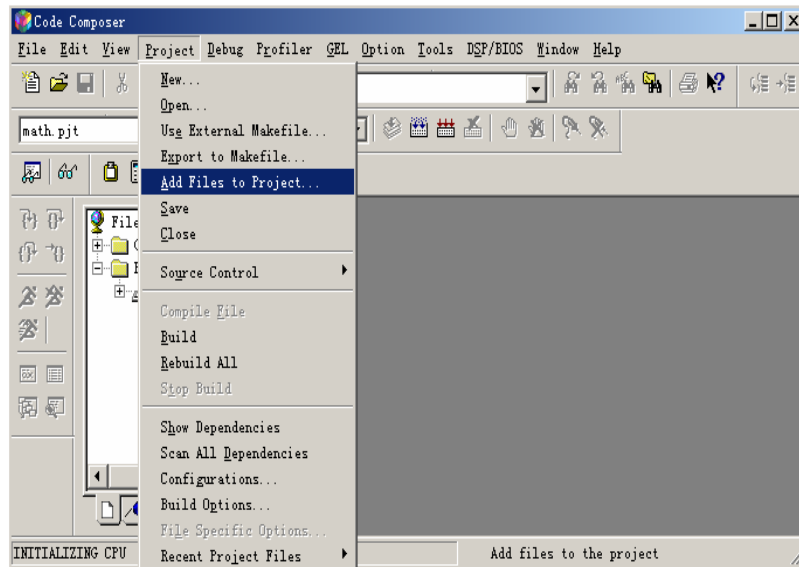
弹出如下对话框：



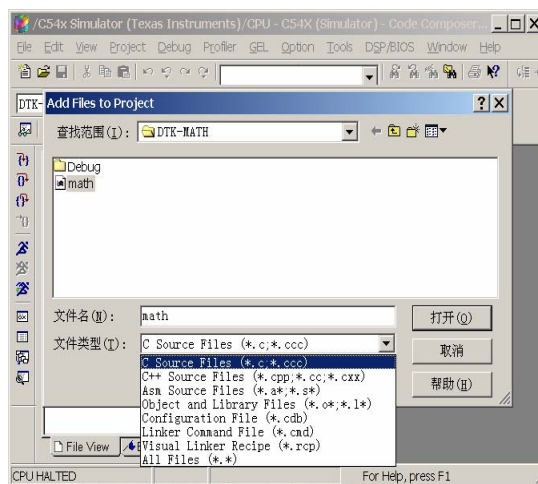
2. 在 Project 中填入工程名，Location 中输入工程路径；其余按照默认选项，点击完

成即可完成工程创建；

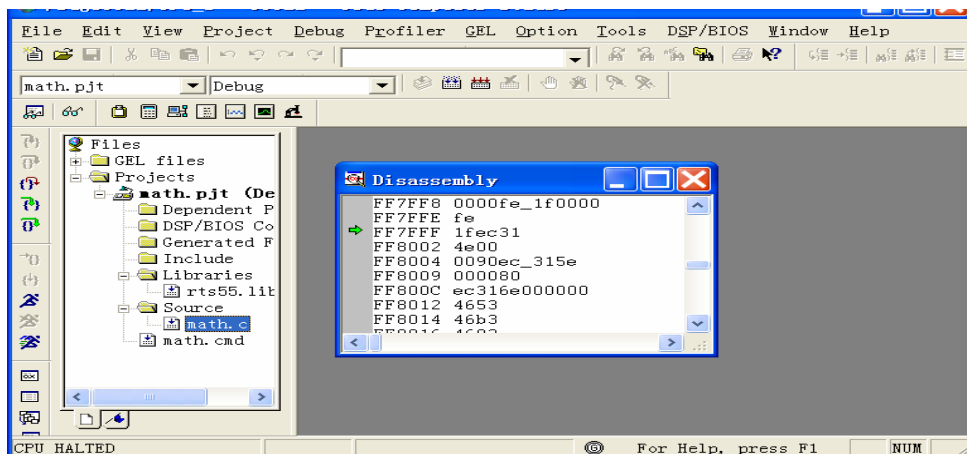
3. 点击 **Project** 选择 **add files to project**，添加工程所需文件；



4. 在弹出的对话框中的下拉菜单中分别选择 **.c** 点击打开，即可添加源程序 **math.c** 添加到工程中；

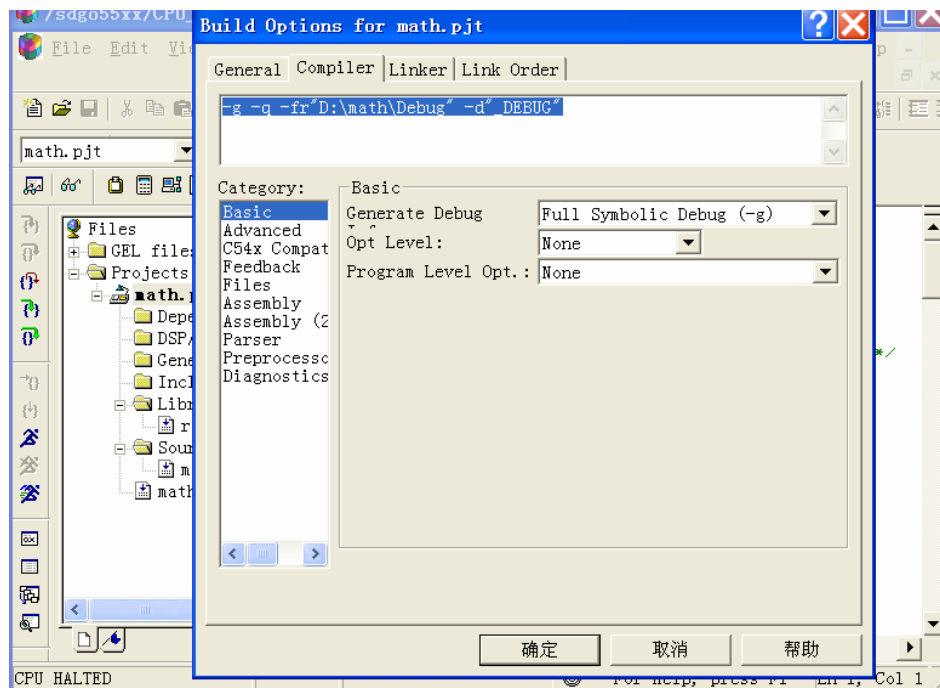


5. 同样的方法可以添加文件 **math.cmd**、**rts.lib** 到工程中；在下面窗口中可以看到 **math.c**、**math.cmd**、**rts.lib** 文件已经加到工程文件中。

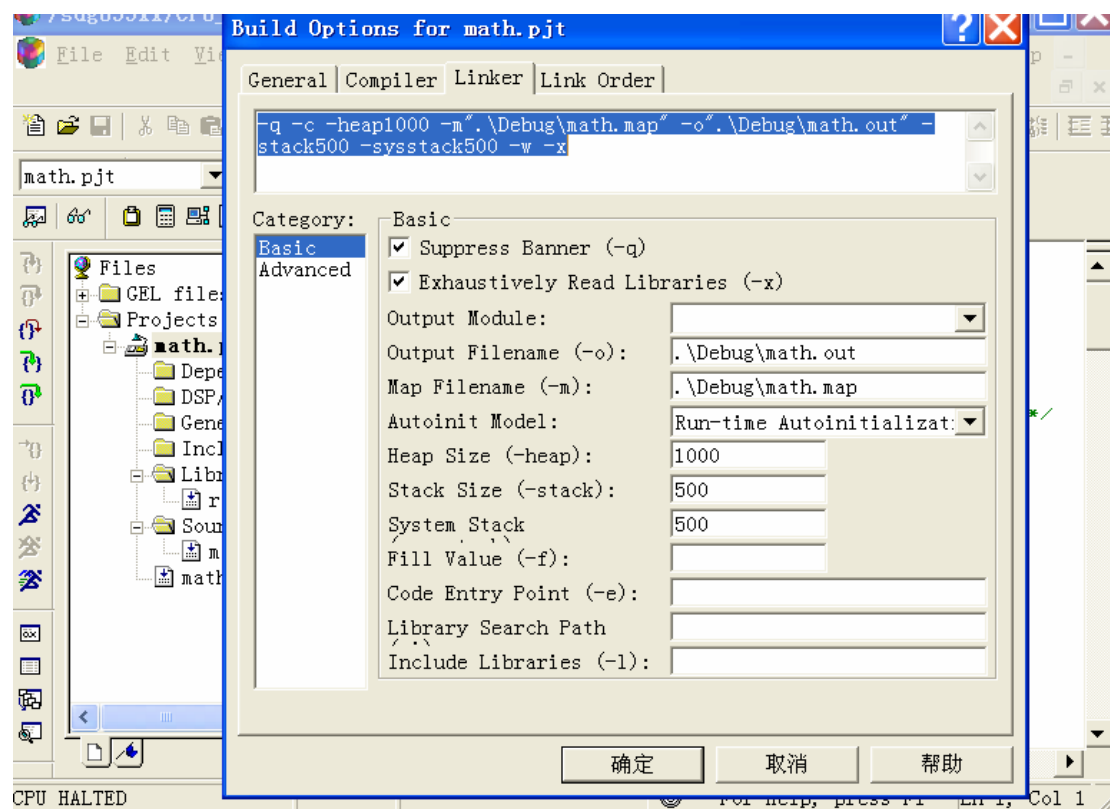


### 3.1.1.5.3 设置编译与连接选项

1. 点击 Project 选择 Build Options;
2. 在弹出的对话框中设置相应的编译参数，一般情况下，按默认值就可以；

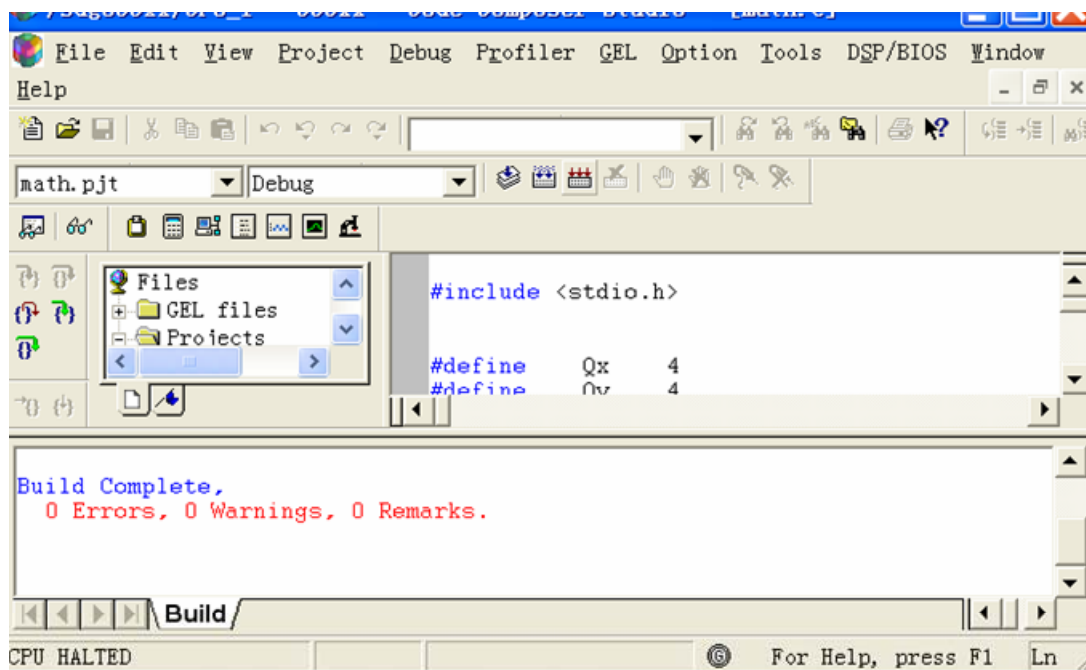


3. 在弹出的对话框中选择连接的参数设置，设置输出文件名（可执行文件与空间分配文件），堆栈的大小以及初始化的方式。

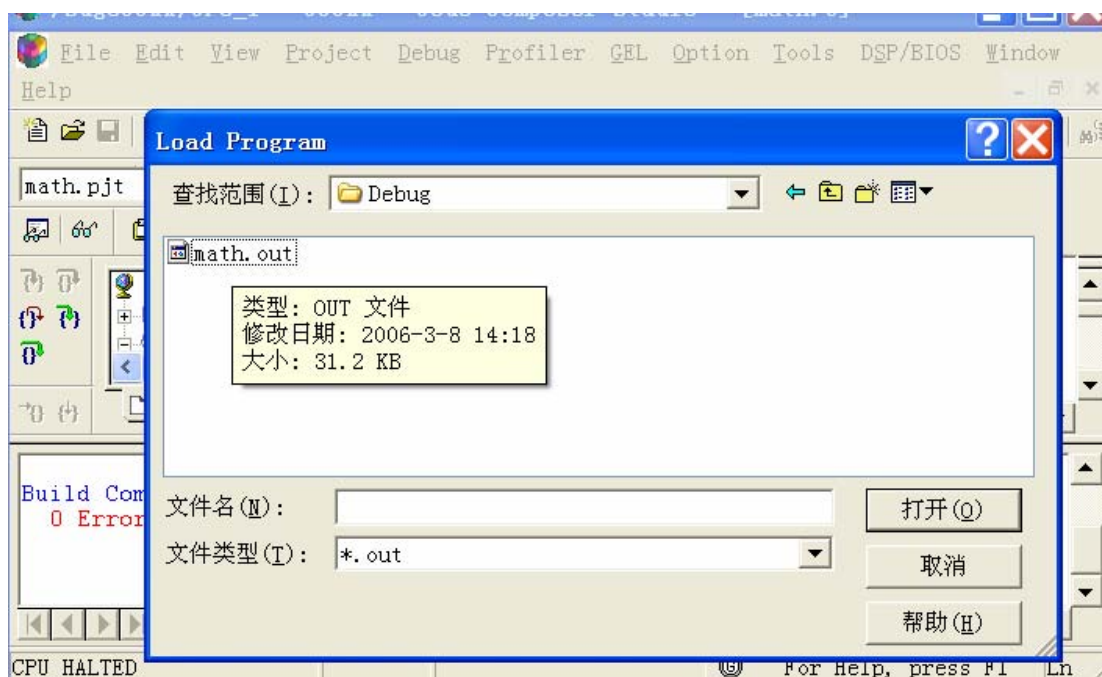


#### 3.1.1.5.4 工程编译与调试

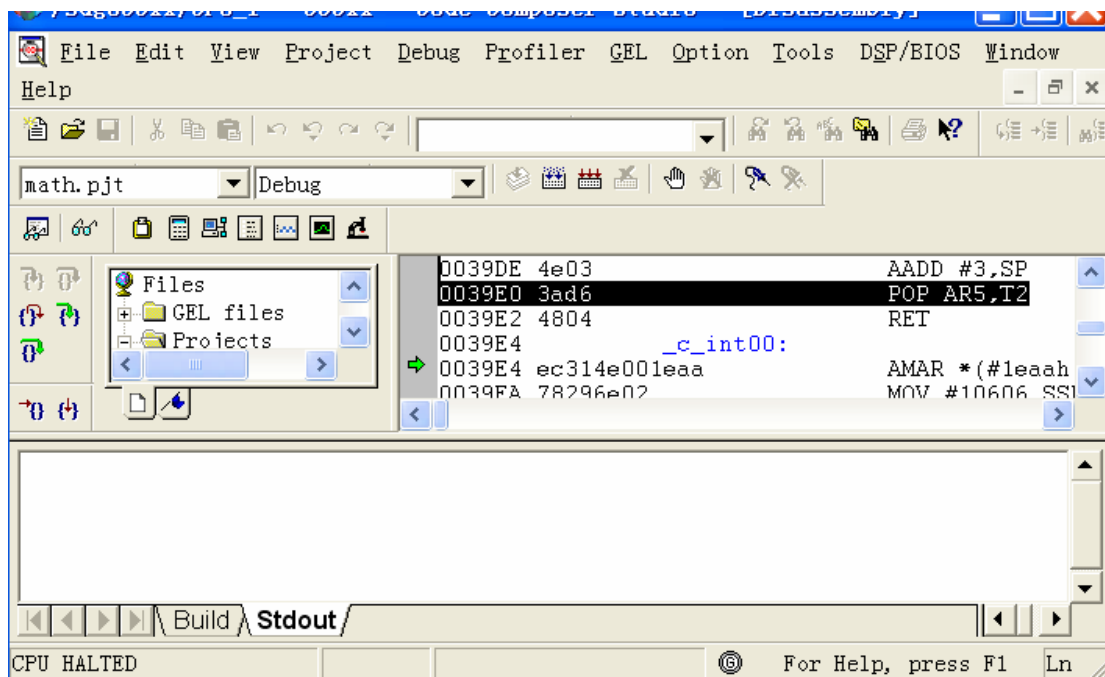
1. 点击 Project → Build all, 对工程进行编译, 如正确则生成 out 文件; 若是修改程序, 可以使用 Project → Build 命令, 进行编译连接, 它只对修改部分做编译连接工作。可节省编译与连接的时间。编译通过, 生成.out 文件;



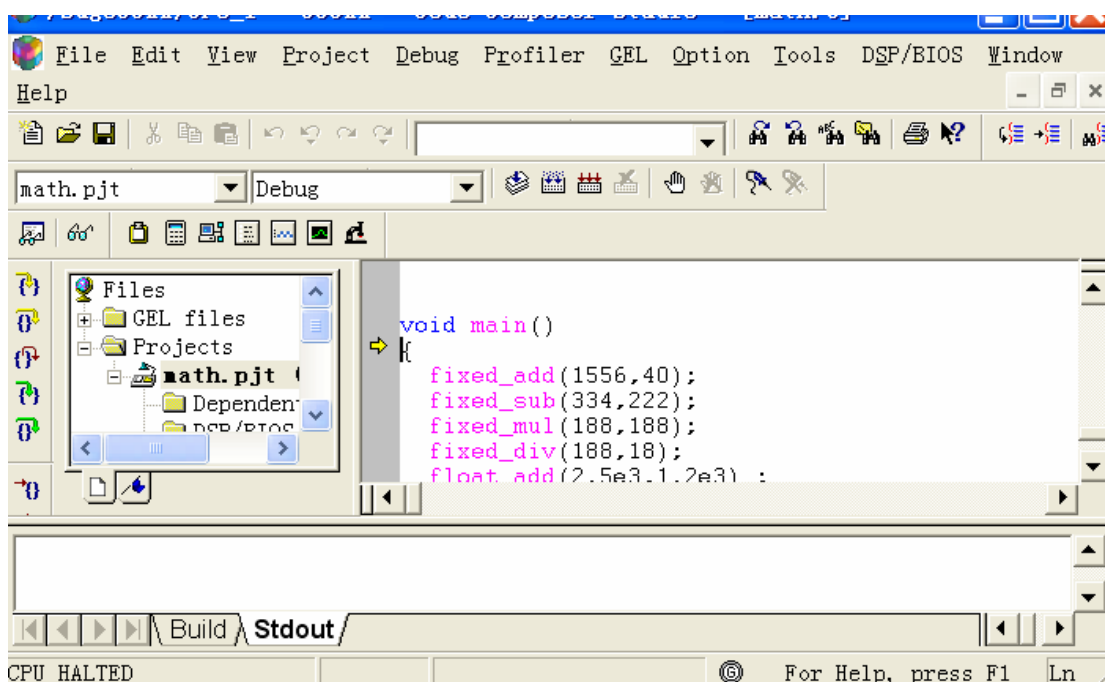
2. 点击 File → load program, 在弹出的对话框中载入 debug 文件夹下的.out 可执行文件;



3. 装载完毕;



4. 点击 debug →Go Main 回到 C 程序的入口;



5. 打开 File →Workspace →Save Workspace 保存调试环境，以便下次调试时不需要重新进行设置。只要 File →Workspace →Load Workspace 即可恢复当前设置。