第 3 章　DSP 芯片的开发环境

3.1　引　　言

DSP 芯片的开发过程就是在选定的 DSP 芯片上实现 DSP 系统功能的过程，例如在 TI 的 C55x 系列芯片上实现语音压缩编码算法、在 C64x 芯片上实现视频跟踪算法，等等。DSP 系统功能的实现，既需要 DSP 芯片及其外围电路的硬件支撑，也需要 DSP 芯片程序代码的软件支撑。因此，可编程 DSP 芯片的开发过程，是一个需要软件开发与硬件开发协调进行的过程。DSP 芯片的开发必须基于软件开发环境和硬件开发环境，并通过对 DSP 系统进行软硬件联合调试，实现最终的系统功能。图 3-1 给出了 DSP 芯片的基本开发流程。具体来说，第②、③、④、⑦、⑧阶段的开发需要相关软件开发环境的支撑，在第⑦、⑧阶段、有时在第④阶段还需要相关硬件开发环境的支撑。

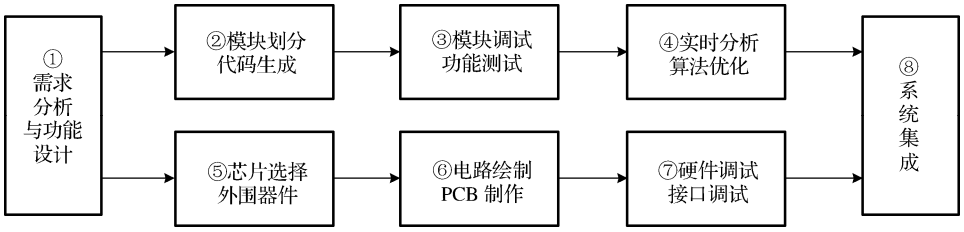

图 3-1　DSP 芯片的开发流程

通常，DSP 芯片的软件开发环境包括代码生成工具、代码调试工具、代码优化工具、代码分析工具四大类；硬件开发环境包括仿真器、硬件板卡和硬件调试环境。本章将分别对 DSP 芯片的软件开发环境和硬件开发环境进行介绍，其中的软件开发环境将重点介绍 TI 公司的 Code Composer Studio（以下简称 CCS）集成开发环境。

3.2　软件开发流程

DSP 芯片的软件开发过程，就是利用编程语言（C/C++或汇编语言），在选定芯片上实现所需要的功能模块的过程。DSP 芯片的软件开发一般包括以下几个阶段：

（1）代码编写：采用编程语言编写功能模块的算法代码。通常，我们为一个 DSP 开发工程设置一个 Project（以下也称为“工程”），然后在其中编写代码实现各功能模块。

（2）编译链接：对 Project 进行编译汇编，每个文件都生成一个目标文件*.obj；再按照链接命令文件*.cmd 的方案，将所有*.obj 文件链接生成可执行文件*.out（详见第 5 章）。注意，这里的*代表某文件的名称，下同。

（3）调试优化：将*.out 文件加载入芯片执行环境，对功能模块代码进行调试，并根据调试结果对代码进行优化，以提高模块的执行效率。

（4）代码固化：对调试结束的执行文件进行格式转换，形成的数据文件通过编程器固化到 DSP 系统的存储器中，以实现系统的脱机运行。

通过以上四个阶段，就能够完整实现在 DSP 芯片上的软件功能开发。其中第（1）、（2）阶段对应图 3-1 中的②，第（3）阶段对应图 3-1 中的③、④，第（4）阶段对应图 3-1 中的⑧。在第（3）阶段中，可执行文件*.out 的调试运行有三种实现方式：

（1）**Simulator 方式**：通过计算机模拟出 DSP 芯片的执行环境，*.out 文件在这个模拟环境中执行，仅需要安装有 CCS 的计算机，无需其他硬件设备的支持。

（2）**Emulator 方式**：计算机通过仿真器联接到 DSP 芯片，将可执行文件加载到 DSP 芯片执行环境中运行，计算机可以对运行过程进行控制，并查看运行结果。

（3）**脱机方式**：可执行文件代码保存在 DSP 系统的存储器中，DSP 系统加电后将程序自动搬移到 RAM 中直接运行，可脱离计算机和仿真器独立运行。

图 3-2 中给出了 DSP 芯片的软件开发流程。

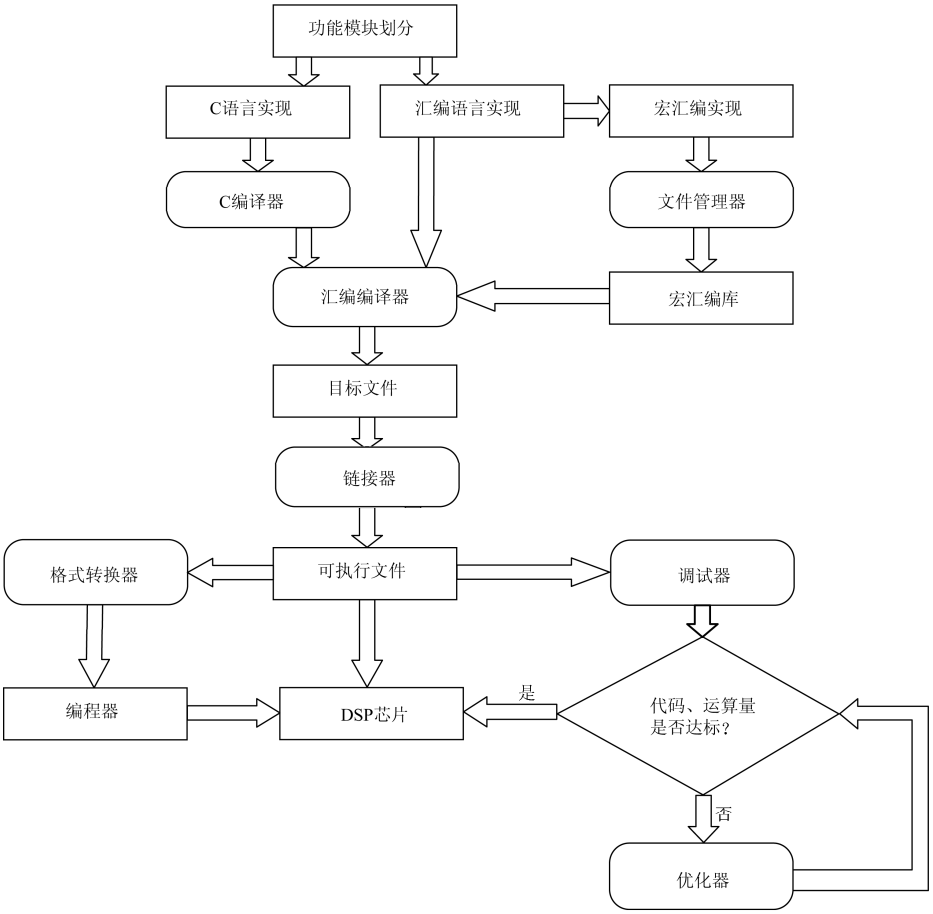

图 3-2　DSP 芯片软件开发流程

从图 3-2 中可以看出，在 DSP 芯片的软件开发过程中，需要在计算机上进行文件编辑、程序编译、代码仿真、性能优化等一系列步骤，这些步骤都可以在 DSP 芯片的软件 IDE *（Integrated Development Environment，集成开发环境）中完成。

目前，各 DSP 芯片生产商都推出了自己的 IDE，如 TI 公司的 CCS 等。开发人员可以借助集成开发环境方便、快捷地独自或合作完成开发流程。下面，我们将介绍 TI 公司的 DSP 芯片 IDE—CCS 的基本使用方法。

3.3 软件开发环境

1999 年以前，TI 提供的 DSP 开发工具是以 DOS 命令行的方式执行的。这些开发工具大多以单个程序的形式出现，数目众多，操作不便，中间数据分析困难，人机界面不够友好，导致 DSP 芯片的开发效率不高。表 3-1 中列出了 TMS320C5000 V3.50 版的部分工具名称。

表 3-1　TMS320C5000 V3.50 版工具程序

程序名	作　用	程序名	作　用
CL500.EXE	一步编译汇编链接程序 一步将.C 程序转换成.OUT 文件	AR500.EXE	文档管理程序 对目标文件库进行增加、删除、提取、替代等操作
AC500.EXE	C 文法分析程序 对.C 文件进行文法分析，生成.IF 中间文件	ASM500.EXE	COFF 汇编应用程序 将汇编语言程序转换为 COFF 目标文件.OBJ
OPT500.EXE	优化程序 对.IF 文件进行优化，生成.OPT 文件	HEX500.EXE	代码格式转换程序 将.OUT 文件转换为指定格式的文件
CG500.EXE	代码生成程序 将.IF 或.OPT 文件生成.ASM 文件	LNK500.EXE	链接程序 将目标文件链接成.OUT 文件
CLIST.EXE	交叉列表程序 对 CG500 生成的.ASM 文件进行交叉列表，生成.CL 文件	MK500.EXE	库生成应用程序

1999 年，TI 推出 CCS 集成开发环境，这是 DSP 软件开发环境的一次革命性的变化。CCS 的功能如图 3-3 所示，它采用图形界面，集编辑、编译、链接、软件仿真、硬件调试及实时跟踪等功能于一体。其中，集成的源代码编辑环境使程序的修改更为方便；集成的代码生成工具使开发人员不必在 DOS 窗口输入大量的命令及参数；集成的调试工具使程序调试一目了然，方便的观察窗口使程序调试得心应手。这些特性极大地方便了 DSP 程序的设计和开发。

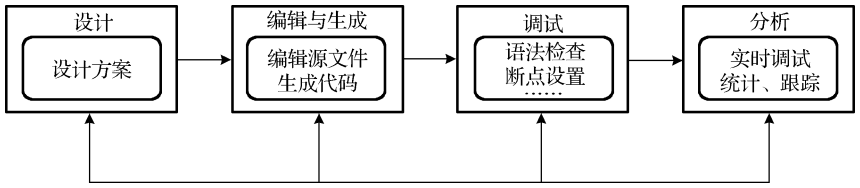

图 3-3　CCS 功能

更为重要的是，CCS 加速了实时、嵌入信号处理的开发过程，它提供配置、构造、调试、跟踪和分析程序的工具，在基本代码产生工具的基础上增加了调试和实时分析的功能。开发人员可在不中断程序运行的情况下检查算法的正确性，实现对硬件的实时跟踪调试，从而可大大缩短程序的开发时间。

2009 年 7 月，TI 推出了 CCS V4.0 版本。这个版本是基于 Eclipse V3.2（Callisto）开源软件框架开发的。它将 Eclipse 软件框架的优点和 TI 的嵌入式调试功能相结合，为用户提供了一个开放的、功能丰富的 IDE 环境。用户可以加入其他厂商的 Eclipse 插件或 TI 的工具，可以随时享受到 Eclipse 最新改进所带来的便利。

从 CCS V4.x 开始，CCS 不但能够支持 DSP 芯片的开发，还能够支持 TI 公司的所有其他嵌入式处理器（Embedded Processors）芯片的开发。这些芯片包括：基于 ARM 的处理器 Stellaris® Cortex-M3™、ARM9™ 系列，基于 Cortex-A8 处理器的微处理器（Sitara ARM MPU），DaVinci™ 视频处理器，OMAP™ 移动应用处理器，等等。

自 2010 年开始，TI 发布了 CCS V5.x，并提供了在 Linux 操作系统环境下使用的版本。CCS V5.x 是第一个基于最新 Eclipse 无修改版本（'Stock Eclipse'）的 CCS，能够让用户更方便地集成第三方的插件，并能够升级 CCS 的开源组件。

2014 年底，TI 发布了 CCS V6.0 版本。CCS V6.0 版本与 V5.0 版本看起来很类似，但与 CCS V5.0 之间还是有很多显著的不同，如 CCS V6.0 是基于 Eclipse 4.x 的，而 CCS V5.0 是基于 Eclipse 3.x 的；CCS V6.0 中增加了 App 中心，以便于用户及时知晓可用的附加软件；CCS V6.0 还停止了多个功能的支持：停止了对 Simulator 的支持、停止了对 C54x 系列的支持，以及在 SYS/BIOS 中停止了对 UART 实时分析的支持，等等。

本节将以 TI 公司推出的 DSP 芯片集成开发环境——CCSV5.5 为例，详细介绍 CCS 的基本环境和使用方法。3.3.1 节中将对 CCSV5.5 使用中涉及的一些基本概念进行介绍，有些概念与 CCS V3.x 的完全不同；3.3.2 节中将介绍软件开发环境界面，并对各界面上的菜单栏选项进行简介；3.3.3 节中将通过开发实例来对 CCS 的基本软件开发功能进行介绍；3.3.4 节将简要介绍 DSP/BIOS 和 SYS/BIOS 的情况。

CCS 的安装过程请参考附录 F。此外，我们假定本书的读者已经具备了 C/C++语言的编程能力，对 C/C++语言编译、链接、调试、执行等系列环节已有了一定的了解。本书对这一方面的内容不再作详细说明。

3.3.1 基本概念

在介绍 CCS 5.x 如何使用前，首先介绍 CCS 使用中的几个基本概念。

1．工程（Project）

CCS 中的任何开发都基于 Project，这是一个集合的概念，包含了一个开发任务的所有相关源代码、包含文件、配置文件等。通常在一个开发任务的开始，首先需要创建一个 Project，可以在其中新建、复制、链接与本次开发任务相关的各种文件。Project 必须被导入工作区后才可以打开。

2．工作区（Workspace）

在 CCS V5.x 中，工作区采用 Eclipse 中的 Workspace 概念，可以看成是用户文件夹，其中包含了被其定义的所有工程的管理信息。在一台计算机上，可以设置多个工作区，每个工作区中可以包含多个 Project，还可以包含一些能够被多个工程共享的其他资源。工作区的概念方便了用户对自己工作资源的管理。但需要注意的是，每个 CCS 工程只能在一个工作区中打开，同一工作区中也无法同时运行多个 CCS 工程。其与 CCS V3.x 中的*.wks 概念完全不同。

3．工作台（Workbench）

每次双击 CCS 图标，都将打开一个新的 CCS 开发环境界面，CCS 将这些界面称为工作台。这是 CCS4.0 版本以后引入的新概念。每次打开一个工作台，只能选择一个工作区进行

管理；而且同一时刻，只能对当前活跃的工程进行开发。可以在一台计算机上同时打开多个工作台，但多个工作台不能同时管理一个工作区。CCS 用统一的工作台模式来对工作区中的资源进行操作，这样就能够实现对工具的最优集成和对操作的最佳扩展。

4．透视图（Perspective）

“透视图”在概念上类似于 CCS V3.x 中的“工作区”（*.wks）。

在工作台上对工程进行开发时，一般都涉及代码编辑、代码调试等多个环节，CCS 将每个环节的相关操作集成在一个透视图中。大多数透视图中都包含一个编辑窗（Edit）和一个或多个不同用途的观察窗（Views），透视图对于工作台内这些窗口的初始设置和布局进行了定义。每个透视图都为某个特定类型任务的完成提供了一套功能组件。比如“CCS Edit”透视图用于工程开发，“CCS Debug”透视图用于工程调试，等等。用户可以创建自定义的透视图。

CCS 的界面有**简易**和**高级**两种模式。简易模式中，菜单选项和工具栏按钮会随着用户所在的透视图而自动简化，为用户的使用带来了极大的方便；高级模式下，将使用默认的 Eclipse 透视图，一般这种模式推荐给能够熟练使用 Eclipse 插件的用户使用。默认情况下，CCS 自动打开简易模式。用户也可以在两种模式间随意切换。本章介绍的是简易模式下的透视图。

图 3-4 是 CCS 一个工作台的上半部分。可以看到，在图中最上面标题栏的下方是菜单栏，其下紧接着的是工具栏。在工具栏同一排的最右面有一个标签部分，其中就是透视图的选择项。在本图中，CCS Edit 选项是高亮显示的，表示当前整个工作台显示的是 CCS Edit 透视图的界面。用鼠标单击不同的透视图选项，就可以在不同的透视图中转移。单击透视图标签选择栏中最左边的“  ”按钮，在下拉菜单中选择“Other…”选项，还可以在跳出的“Open Perspective”窗口中为标签增加更多的透视图选项。

图 3-4　CCS 工作台的上半部分

每个透视图针对的工程开发环节不同，因此所涉及的操作选项也是不同的。为了提高开发效率，降低用户的使用难度，CCS 针对不同的透视图提供了不同的菜单和工具选择；甚至在不同透视图中的同一菜单名下，其下属的选择项也有不同的增减组合。

鼠标右键单击图 3-4 标签中的透视图名称，在弹出的菜单中选择“Customize…”选项，可以对该透视图界面中的可见菜单、可见工具、可用命令集和可用快捷方式等进行选择。熟练的用户可以据此组合自己的常用工具。

在 3.3.3 小节中，将通过实例介绍 CCS 的编辑透视图和调试透视图的主要窗口。有关透视图的更多介绍，可以参看 CCS Help。

5．常用的文件类型

CCS 中常使用到以下几种类型的文件：

- *.c：　C 语言源文件；
- *.asm：　汇编语言源文件；
- *.h：　C 语言和 DSP/BIOS API 的头文件；
- *.lib：　库文件；
- *.gel：　初始化文件；
- *.obj：　编译或汇编产生的目标文件；
- *.out：　编译、汇编和链接后产生的执行程序文件，在 CCS 中可以加载并运行这个执行程序；
- *.ccxml：　目标配置文件，是定义设备与连接的 XML 文件。

除了以上文件外，在开发过程中，还会产生多种其他类型的文件。上面的几种类型文件在软件开发中承担着重要的角色，需要对其有清晰的了解。

3.3.2　CCS 开发环境

本节主要介绍两个常用的透视图以及几种常用的菜单选项。

1．CCS Edit 透视图

CCS Edit 的透视图如图 3-5 所示，用以进行代码编辑。在 CCS Edit 透视图中，通常显示的窗口有以下几种。

（1）工程浏览（Project Explorer）窗：位于左边的窗口 1，其中显示的是当前工作台中的资源目录。如图中的窗口中，显示了三个 Project 名称，其中的“hello”下的相关文件目录被显示出来。可以选中某个文件名称后右击鼠标，通过弹窗选项对该文件进行相应操作选择。双击对应文件名称，可以在右边的编辑窗口打开对应的文件进行编辑。

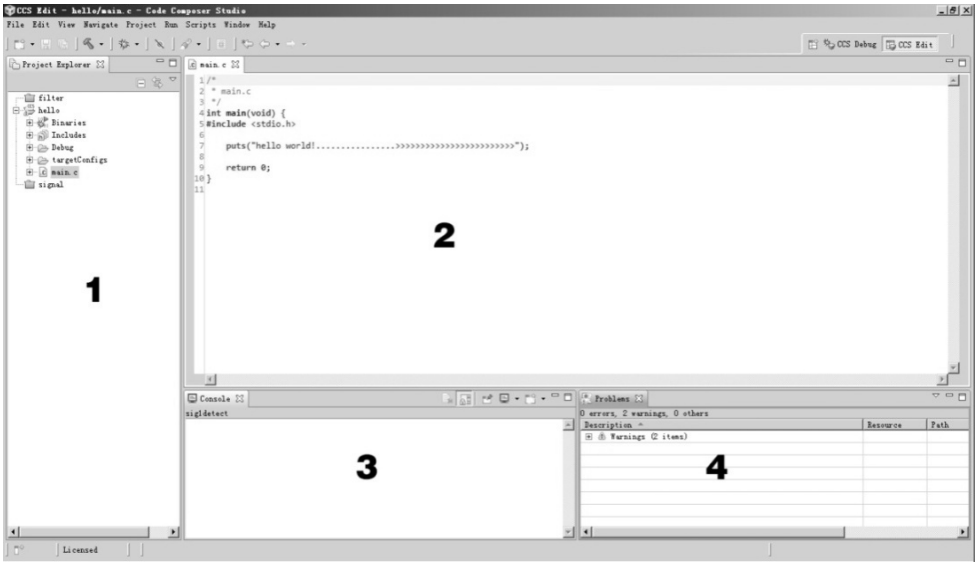

图 3-5　CCS Edit 透视图

（2）编辑（Edit）窗：位于 Project Edit 窗口右方上半部分的窗口 2，其中显示的是 Project Edit 窗口中被选定的 main.c 文件的内容，其标签上标出了所显示文件的名称。可以在该窗口中对文件内容进行文本编辑。

（3）控制台（Console）窗：位于编辑窗口左下方的窗口 3。有多种不同用途的控制台窗口，在 CCS Edit 透视图下的是构建（Build）控制台窗口，在对工程进行 Build 操作后出现，显示在工程进行 Build 操作中产生的各种信息。Build 操作中的问题信息会高亮显示，如果该信息与某文件的某行文本有关，双击该信息就可以直接打开显示该行文本。

（4）问题（Problem）窗：位于编辑窗口右下方的窗口 4。其中显示的是 Build 操作后产生的各种问题、错误等的列表信息。

除了上述窗口外，还可以单击透视图左下角的“ ”图标，选择其他的窗口进行显示，包括建议（Advice）窗、错误日志（Error Log）窗、提纲（Outline）窗、目标配置（Target Configurations）窗，等等。

2．CCS Debug 透视图

CCS Debug 透视图如图 3-6 所示，用以进行工程调试，一般包含 4 个区域。

区域 1：位于透视图左上部分，为调试（Debug）窗，其中用树状图的方式显示了目标调试信息。

区域 2：位于透视图右上部分，包含了多种调试参数中间信息的显示窗：

① 变量（Variables）窗：显示代码调试中各个变量的名称、类型、数值、位置等信息；

② 表达式（Expressions）窗：可以在代码调试中显示变量表达式的值；

③ 寄存器（Registers）窗：显示代码调试中目标芯片内部各寄存器的当前值；

④ 断点（Breakpoints）窗：用于显示和编辑代码调试断点的信息。

区域 3：位于透视图的中间部分，代码调试期间，在窗中显示的源代码行上，将通过标志、高亮等方式，同时显示断点位置、当前执行行等信息。

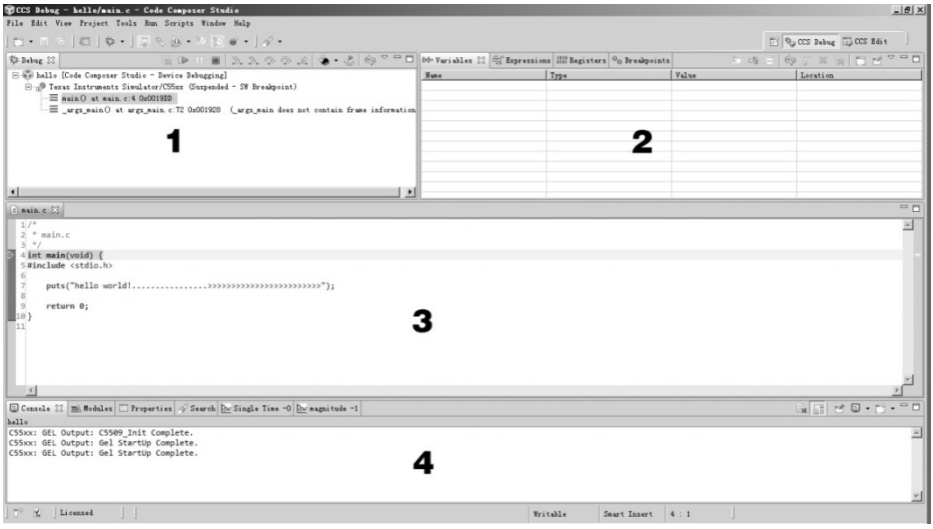

图 3-6　CCS Debug 透视图

区域 4：位于透视图的下方部分，其中包含了多种信息窗：

① 控制台（Console）窗：其中显示的是代码调试输入、输出、错误信息。

② 单元（Modules）窗：其中用树状图方式显示了目标可执行代码*.out 文件中涉及的各个文件、函数、类型、地址标记等信息。

③ 属性（Properties）窗：其中显示的是所选定资源的属性名称和对应值。

④ 搜索（Search）窗：显示搜索内容。

除了上述四种窗口，区域 4 中还可以显示图形窗，如时序图、频谱图等。

同样地，通过单击透视图左下角的“”按钮，可以在这个透视图中选择其他的窗口进行显示，包括反汇编代码（Disassembly）窗、存储区浏览（Memory Browser）窗、硬件跟踪（Trace）窗、目标配置（Target Configurations）窗，等等。

3．常用菜单选项

透视图中一般有多个菜单项，不同透视图中的菜单项组成不同，菜单项中的选项组成有时也会不同。这种特性大大减轻了用户操作中的选择压力。下面将介绍在 DSP 开发中常会用到的几个菜单的常用选项。

（1）File 菜单

File 菜单中提供了创建、保存、关闭、打印、输入、输出、退出等选项，如表 3-2 所示。

表 3-2　File 菜单常用选项

名称	快捷键	功　能
New	Shift+Alt+N	创建新的资源文件（如源文件、头文件等）。在工程开发开始，首先要创建一个 Project 以保存文件
Open File		打开待编辑的文件，可以打开不在当前工作区中的文件
Close	Ctrl+W	关闭当前活跃的编辑窗
Close All	Shift+Ctrl+W	关闭所有打开的编辑窗
Save	Ctrl+S	保存当前活跃的编辑窗中的内容
Save As		允许将当前的活跃编辑窗中内容用另一个名字保存到另一个目录
Save All	Shift+Ctrl+S	保存所有打开的编辑窗中的内容
Revert		在当前活跃编辑窗中，用以前保存的内容替代当前的内容
Move		将当前选择的资源移到另一个工程中
Rename	F2	更改当前选择的资源的名字
Refresh	F5	更新当前文件系统中的资源内容
Convert Line Delimiters To		改变所选文件的行分隔符。无需保存，立即生效
Print	Ctrl+P	打印当前活跃编辑窗的内容
Switch Workspace		从当前工作区切换到另一个工作区，工作台将重启
Import		打开 Import 向导，向工作台上加入新的资源
Export		打开 Export 向导，将工作台上的资源导出
Properties	Alt+Enter	打开当前所选资源的属性对话框，其中有 • 资源在文件系统中的路径； • 上次修改的日期； • 文件是可写还是可执行，以及其编码方式； • 工程中的资源是否继承了编码和行分隔符选择
Recent file list		含有最近进入的工作台中文件的列表
Exit		关闭并退出工作台

（2）Edit 菜单

这个菜单帮助用户在编辑窗中处理资源。在不同的透视图和不同的活跃窗状态下，这个菜单的下拉选项会不同，如表 3-3 所示。

表 3-3　Edit 菜单常用选项

名　称	功　能
Undo	撤销最近一次编辑行为
Redo	重做被撤销的最近一次编辑行为
Cut	将所选内容搬移到剪贴板上
Copy	将所选内容复制到剪贴板上
Paste	在当前活跃编辑窗或观察窗的光标位置放入剪贴板上的内容
Delete	删除所选内容
Select All	选择当前活跃编辑窗或观察窗内的所有文字或对象
Find/Replace	在当前活跃编辑窗内寻找某个符号，并可用另一个符号进行替代
Find Next	在当前活跃编辑窗内寻找当前搜索符号或最近一次搜索符号的下一个显示点
Find Previous	在当前活跃编辑窗内寻找当前搜索符号或最近一次搜索符号的前一个显示点
Incremental Find Next	允许在当前活跃编辑窗内进行表达式搜索，在输入表达式时，会自动增补表达式，并跳至相应的下一点。上下键可用来选择，左右、回车和空格键可用来终止搜索
Incremental Find Previous	允许在当前活跃编辑窗内进行表达式搜索，在输入表达式时，会自动增补表达式，并跳至相应的前一点。上下键可用来选择，左右、回车和空格键可用来终止搜索
Add Bookmark	在当前活跃文件的光标所在行增加一个书签
Add Task	在当前活跃文件的光标所在行增加一个任务
Word Completion	试图补完整当前活跃编辑窗中正输入的词
Set Encoding	展开编码对话框，允许对活跃编辑窗中文件的读写编码方式进行设置

（3）Navigate 菜单

这个菜单帮助用户在工作台中的资源和其他显示项中进行定位和导航。不同的活跃窗显示状态下，这个菜单的下拉选项会不同（见表 3-4），在某些透视图（如 CCS Debug）中不显示该菜单。

表 3-4　Navigate 菜单常用选项

Go Into	激活在这里所选的观察窗。可进行网页浏览器式的分级导航
Go To	• Back：类似于 HTML 的 Back 按键，回到当前显示的前一次显示层； • Forward：类似于 HTML 的 Forward 按键，回到当前显示的后一次显示层； • Up one level：显示当前显示最高层的上一层； • Resource：快速导航到资源
Open Hyperlink	打开当前位置的一个或多个超链接。若只有一个，则立即打开；否则在当前位置显示所有超链接
Open Resource	打开对话框，选择要在编辑窗中打开的工作区中任何资源
Show In	对于当前活跃资源，在后续子菜单中选中并激活其在另一个窗中的内容
Next	在活跃窗中移到当前列项或表格活跃条的下一条
Previous	在活跃窗中移到当前列项或表格活跃条的前一条
Last Edit Position	跳至上一次编辑位置
Go to Line	跳至当前活跃编辑窗的选定行
Back	类似于网页浏览器的 Back 键，回到编辑窗中前一次显示的资源
Forward	类似于网页浏览器的 Forward 键，撤销上一次 Back 的效果

（4）Project 菜单

这个菜单允许用户对工作台中的工程进行操作。不同的活跃窗显示状态下，这个菜单的下拉选项会不同，如表 3-5 所示。

表 3-5　Project 菜单常用选项

Open Project	打开当前选择的一个或多个 Project。所选的 Project 当前应该是关闭的
Close Project	关闭当前所选的一个或多个 Project，它（们）当前应该是打开的。关闭 Project 后，内存中的所有相关状态会被消除，但不会动硬盘上的相关内容
Build All	对自上一次编译以来有变化的当前工作台内所有工程进行编译。仅在 auto-build 关闭时有效。Auto-build 可通过当前菜单中 Build Automatically 菜单项或选项页面中的 General > Workspace preference 页关闭
Build Project	对自上一次增量编译以来有变化的所选工程进行增量编译。仅在 auto-build 关闭时有效。Auto-build 可通过当前菜单中 Build Automatically 菜单项或 Properties 选项页面中的 General > Workspace preference 页关闭
Build Working Set	对自上一次增量编译以来有变化的工作集中的工程进行增量编译。仅在 auto-build 关闭时有效。Auto-build 可通过当前菜单中 Build Automatically 菜单项或 Properties 选项页面中的 General > Workspace preference 页关闭
Clean	丢弃所有先前编译的结果。若 autobuild 有效，则可能带来一次完全编译
Build Automatically	打开/关闭 auto-build 选项。也可在当前菜单中 Properties 选项页中的 General > Workspace preference 页面中进行选择
Properties	打开所选工程或包含所选资源的工程的属性对话框

（5）Window 菜单

这个菜单允许用户对工作台中的各个视窗、透视图和功能窗进行显示、隐藏等操作，如表 3-6 所示。

表 3-6　Window 菜单常用选项

New Window	打开与当前工作台具有相同透视图的新工作台窗
New Editor	打开新的编辑窗，刚打开的时候与当前编辑窗具有相同的编辑类型和输入
Open Perspective	在当前工作台窗口中打开新的透视图。可在当前菜单中 Preference 选项页的 General > Perspectives preference 页面修改其可选项。快捷栏上会显示所有可选的透视图。从 Other...子菜单可以打开任何透视图
Show View	在当前透视图中打开所选的窗。可在当前菜单中 Preference 选项页的 General >Perspectives preference 页面配置如何打开窗。从 Other...子菜单可以打开任何窗。在 Show View 对话框中对窗进行了分类
Customize Perspective	可从菜单栏或工作台栏中选择定制透视图的功能项
Save Perspective As	保存当前透视图，创建用户定制的透视图。保存后，可通过 Window > Open Perspective > Other 打开该类型的透视图
Reset Perspective	将当前的透视图布局改回原始布局
Close Perspective	关闭当前活跃的透视图
Close All Perspectives	关闭工作台中所有打开的透视图
Navigation	该项中给出了在工作台中的观察窗、透视图和编辑窗之间进行导航的快捷键。 Show System Menu: 显示当前观察或编辑窗的系统菜单； Show View Menu: 显示当前活跃窗的工具栏上可用的下拉菜单； Maximize active view or editor:令活跃窗满屏显示;若其已满屏，则回到前一状态； Minimize active view or editor:最小化活跃窗； Activate Editor: 激活当前的编辑窗； Next Editor: 激活当前打开过的编辑窗列表中最近的下一个编辑窗； Previous Editor: 激活当前打开过的编辑窗列表中最近的前一个编辑窗； Switch to editor:显示允许切换去编辑窗的对话框； Quick switch editor: 给出可选的弹出栏，允许切换去新编辑窗； Next View: 激活当前打开过的观察窗列表中最近的下一个观察窗； Previous View: 激活当前打开过的观察窗列表中最近的前一个观察窗； Next Perspective: 激活当前打开过的透视图列表中最近的下一个透视图； Previous Perspective: 激活当前打开过的透视图列表中最近的前一个透视图
Working Sets	可对工作集进行选择和编辑
Preferences	用于进行工作台的选项确定

（6）Help 菜单

这个菜单能够帮助用户使用工作台，如表 3-7 所示。

表 3-7　Help 菜单常用选项

Welcome	打开 Welcome 内容
Help Contents	在 help 窗口或外部浏览器中显示 help 内容
Search	显示搜索页中打开的 help 窗
Dynamic Help	打开相关标题页的 help 窗
Key Assist ...	显示关键词对应的快捷键组合
Tips and Tricks ...	将打开你可能没有看过的产品特性
Cheat Sheets ...	打开边沿 help 选择对话框
Check for Updates	更新检查
Install New Software...	下载并安装新软件
About Eclipse SDK	显示产品、安装属性和可用插件的信息

3.3.3　软件开发功能

本节以 Simulator 方式为例，对 CCS V5.5 的基本功能进行详细介绍。CCS V5.5 可以构建一个模拟 DSP 芯片核心部分的软件环境，支持用户在该软件环境下对工程进行仿真调试，但是不支持工程代码中对应于芯片外围硬件部分的仿真调试。开发者一般通过 Simulator 方式来查看工程中信号处理算法的运行情况。

CCS 中软件开发的主要步骤为：

工程创建→文件编辑→编译链接→文件加载→代码执行→结果查看

当整个流程结束后，根据执行情况，可以回到第二步对代码修改，并再次调试。

本节将按照上述的实现步骤，通过软件开发的实例过程，对 CCS 中相应的 Simulator 功能进行介绍。但需要注意的是，CCS 的开发对象是 DSP 芯片而非计算机：后者一般有丰富的硬件资源，安装了操作系统，对显示器、键盘等输入输出设备有完善的支持，因此对于 printf() 等函数功能的支撑很方便；而 DSP 芯片的硬件资源有限，一般没有显示器、键盘这类标准的输入/输出设备，即使在 CCS 环境的支持下，也往往需要耗费大量的资源。因此，不建议在 DSP 开发中，尤其在脱机系统中运行时，大量采用 C 语言中的标准输入/输出语句。更多的相关说明可以参见本书第 6 章的内容。

1．工程创建

学习 C 语言大都开始于编写“HelloWorld”显示程序，该程序最终在显示器上输出一行文字“Hello World!”。这里我们也借用这个开始 CCS 的学习。

首先双击 CCS 图标，打开一个工作台，在菜单栏选择 Project→New CCS Project，或 File→New→CCS Project，弹出如图 3-7 所示对话框。

在 Project name 栏填入工程名称 HelloWorld。

在 Output type 栏中有三个选项：**可执行**（Executable）、**静态库**（Static Library）、**其他**，本工程选择为可执行项。

在“使用默认位置”选项前打钩，表明这个工程的开发是在默认工作区中进行的。默认工作区路径已经在安装部分设定。

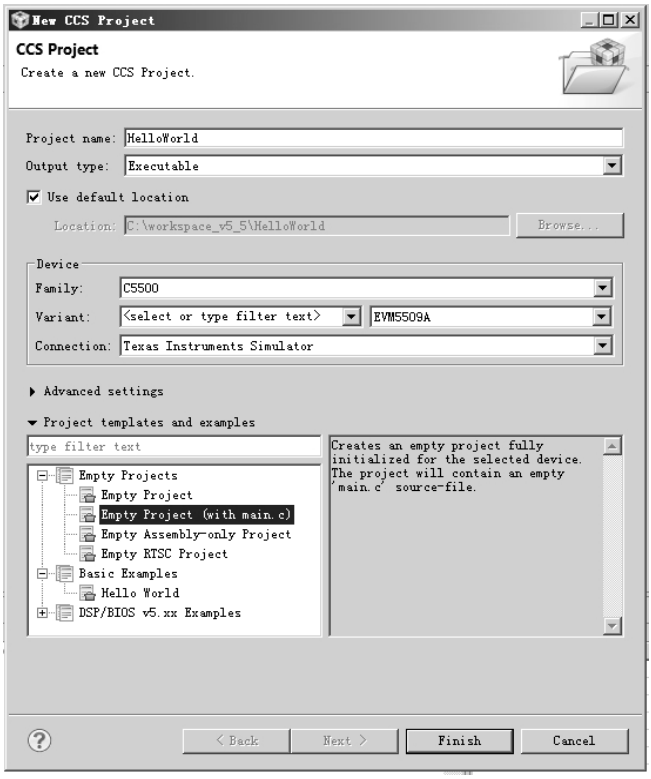

图 3-7　工程新建选项窗

在 Device 选项区中，Family 选项在右侧的下拉菜单中选择 C5500；Variant 选项在右侧的下拉菜单中选择 EVM5509A；Connection 选项在右侧下拉菜单中选择“Texas Instruments Simulator”。

在左下方的工程模版选区（Project templates and examples）中，选择空的工程（这里可以选择带 main.c 模版，如图 3-7 所示）。

单击窗口右下方的 Finish，自动进入 CCS Edit 透视图，工程创建完成。

创建成功后的透视图如图 3-8 所示。在左边的 Project Explorer 窗中出现了 Project 目录树，其中自动添加了 Includes、targetConfigs、Main.c 三个部分。

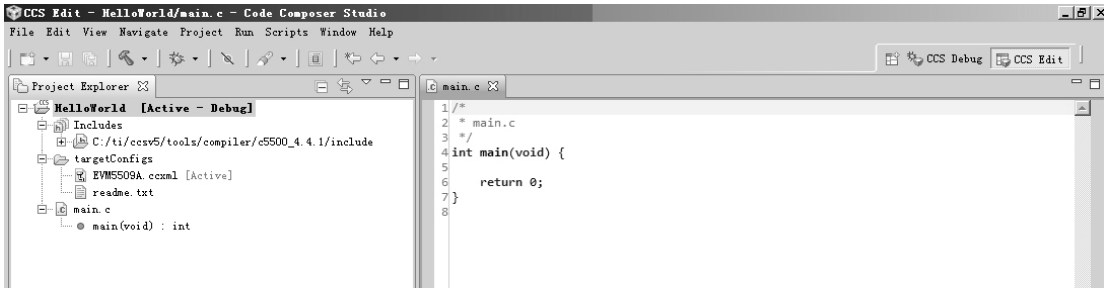

图 3-8　新建工程后的 CCS Edit 透视图

Includes 部分将其中显示的路径下所有的.h 文件都自动加入。

targetConfigs 部分中，根据我们在前面的目标选择，自动加载了 EVM5509A.ccxml 文件，其中保存了 EVM5509A 仿真开发板的配置信息。

如果需要对当前工程进行更改，可以在 Project Explorer 窗中的工程名称上单击鼠标右键，在跳出的弹窗中选择相应的操作命令，如创建新文件、更改工程名称、打开/关闭工程，等等。

例如，右键单击左侧 Project Explorer 窗中的工程名称，单击 Add Files…按钮，可以将工作区中已存在的文件加入到现在的工程中来。在选择文件后，会跳出新的窗口，询问所选择的文件是复制到本工程中，还是链接到本工程中。如果选择链接，那么这个文件在别的场合中的改动也会影响本工程中这个文件的内容。因此，一般选择复制方式。

上面给出的工程创建步骤中，当前 CCS 工作台中没有工程被激活。如果当前有一个工程正处于激活状态下，建议先关闭当前工程后，再创建新的工程，以避免造成管理的混乱。此时有以下两种方法：

方法一：在 Project Explorer 窗中，用鼠标右键单击当前工程名，在弹出菜单中选择 Delete 选项，注意在随后弹出的窗口中不勾选唯一的选项，以避免将当前工程的内容全部删除，一旦删除是无法恢复的！单击 OK 后，当前工程将被清除出当前工作台，在 Project Explorer 窗中工程名消失，但是所有的文件还保存在当前工作区中。

以后通过单击菜单栏的 Project → Import Existing CCS Eclipse Project，在弹出窗口选择相应工程，可以重新将工程加载到当前工作台中。

方法二：在 Project Explorer 窗中，用鼠标右键单击当前工程名，在弹出菜单中选择 Close Project 选项。此时，相应工程的名称仍然在 Project Explorer 窗中，但不再处于激活状态，文件目录树被关闭。

鼠标右击工程名称并选择 Open Project，可以再次激活工程，打开对应目录树。

2．文件编辑

工程创建后，下一步就是对工程中的文件进行编辑。双击 Project Explorer 窗中的任何文件名，都会在右边窗口出现该文件的内容，可以在这个窗口对相关内容进行编辑。

这里仍用 HelloWorld 工程示例。这个工程的 C 语言源代码很简单，如下所示：

```
#include  <stdio.h>
int main(void)
{
    puts("Hello World!!");
    return 0;
}
```

在本工程中，只需要对 C 语言源代码进行编辑。

双击 Project Explorer 窗中的 main.c 文件名，在右边的编辑窗中就出现了当前的 C 源代码内容。按照上面给出的源代码将 main 函数补充完整，然后单击上方工具栏中的“”图标保存文件。

3．编译链接

工程中的文件编辑完成后，进入编译、汇编、链接步骤。CCS 中，通过一个 Build 操作来完成上述全部三个步骤。这个操作可通过单击菜单中的 Project→Build All 选项完成，或者也可以单击工具栏中的“”图标进行。单击该图标的右侧向下键可以选择 Build 的类型，默认方式是“Debug”。这里我们采用了默认方式。工程 Build 的透视图如图 3-9 所示。

经过 Build 操作后，从 Project Explorer 窗中可以看到，工程下又增加了 Binaries 和 Debug 两个部分。

Binaries 中包含了一个.out 文件，这是工程生成的执行代码文件。在后面的调试阶段，需要将这个文件载入 CCS 中进行运行调试。

在 Debug 部分，除了目标文件（HelloWorld.obj）和可执行文件（HelloWorld.out）外，还出现了几种新的文件，其中比较重要的有：

- ccsObjs.opt 文件：保存了本工程的编译链接选项信息；
- HelloWorld_linkInfo.xml：保存了工程链接过程中的信息；
- HelloWorld.map：其中保存了工程链接后各代码块的位置信息，包括代码块的名称、归属、长度、起始地址等，对于代码的优化十分重要，将在第 5 章中介绍。

以上三类文件是编译链接过程中自动产生的，可以帮助我们对工程的编译链接过程进行更详细的了解，没有必要对其进行编辑。

此外，在透视图下方，Console 窗中给出了工程编译链接的相关过程信息，其中的黄色高亮部分显示的是警告信息，单击窗口上方的“ ”按钮可以清除本窗口内容；Problem 窗中列表显示了相应警告信息的详细情况。这里出现的两个警告不影响工程的执行。

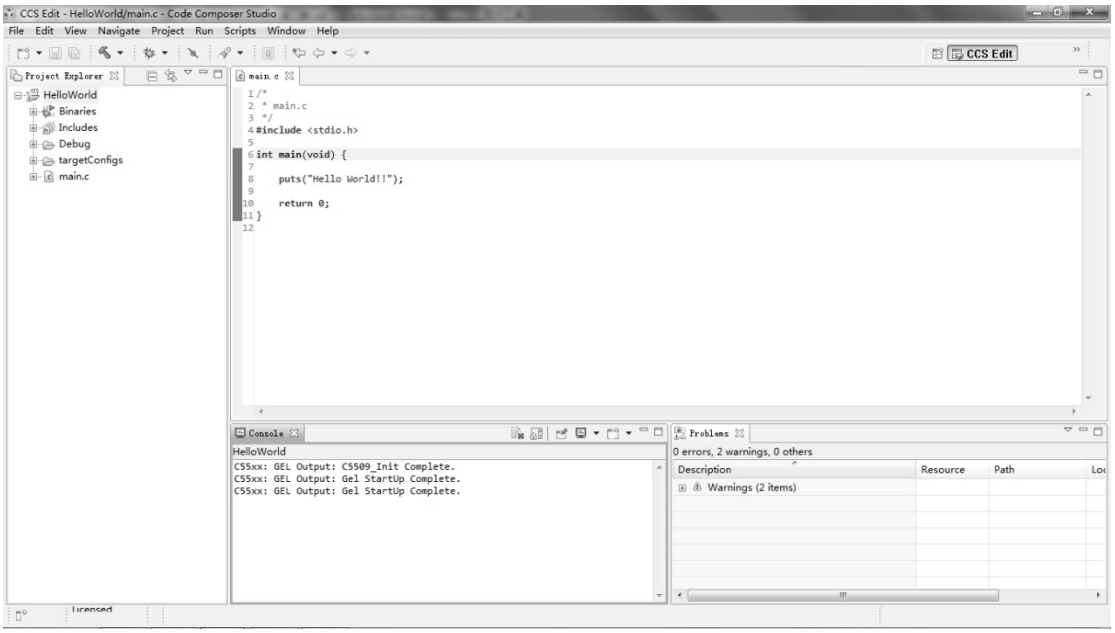

图 3-9　编译链接后的 CCS Edit 透视图

工程经过一次 Build 后，如果其中内容没有改动，再次执行 Build 操作后，CCS 将不会再次进行编译链接。用鼠标右键单击 Project Explorer 窗中的工程名，在出现的菜单中单击 Rebuild Project 选项，可以对工程再次进行编译链接。

4．调试执行

完成工程的编译链接后，就可以对生成的执行代码进行调试，以确定是否正确。因此，需要将上面生成的*.out 文件加载入 CCS 中并运行。这些工作全部通过 Debug 操作完成。

单击菜单栏中的 Run→Debug，或单击工具栏中的“”图标，或单击键盘的 F11 功能键，都可以执行这一操作。随后 CCS 进入 CCS Debug 透视图，如图 3-10 所示。

在中间代码窗口中，可以看到所显示的源文件第 6 行最左侧有一个箭头，同时该行被淡蓝色高亮显示，表明 CCS 当前执行到 main 函数的入口处，并暂停在这里。

在 Debug 窗口的右上方，排列着一排工具图标。这些图标对应的操作将帮助我们控制代码的执行：

：继续（Resume），单击后，将结束暂停状态，从当前高亮行开始，顺序向下执行。

：暂停（Suspend），单击后，将进入暂停状态，下一条等待执行的代码行高亮显示。

：结束（Terminate），单击后，将结束本次调试，CCS 跳转到 CCS Edit 透视图。

：单步执行（Step Over），单击后，将执行当前代码窗中高亮行代码后暂停。此时代码窗口左侧箭头指向下一条语句并高亮显示。工具栏中有两个不同颜色的这个图标：如果是黄色的图标，表明是针对 C 源代码进行相应操作；如果是绿色的图标，表明是针对汇编源代码进行相应操作。

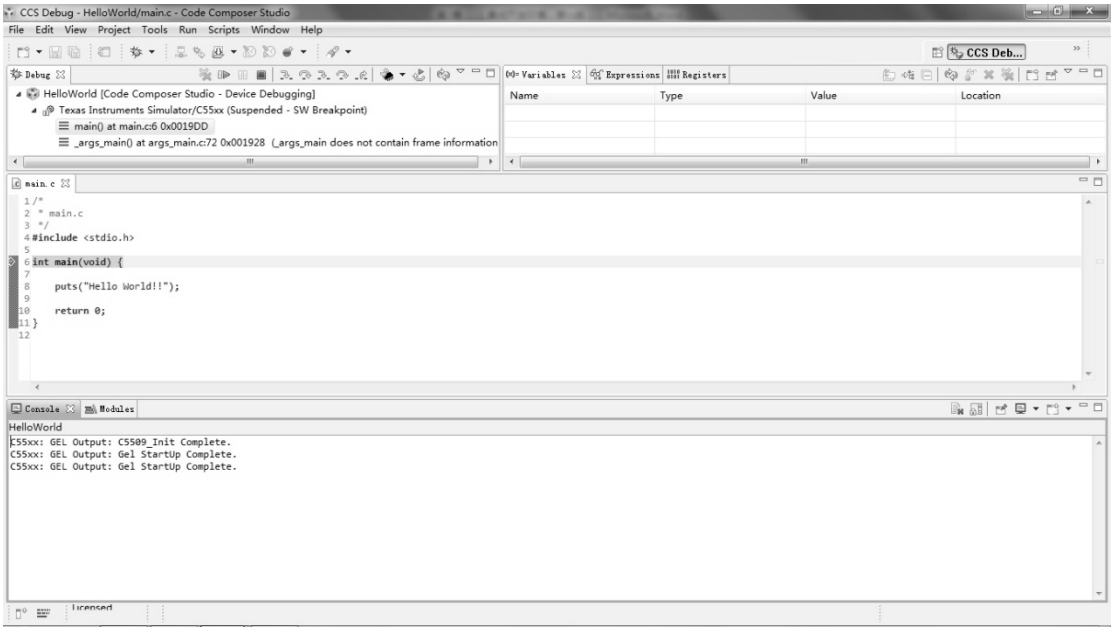

图 3-10　执行 Debug 操作后的 CCS Debug 透视图

：单步进入（Step Into），单击后，如果当前待执行语句中没有函数，则效果同上；否则，将跳转进入该行中第一个函数的对应源代码中并暂停。工具栏有两个不同颜色的这个图标，如果是黄色的图标，表明是针对 C 源代码进行相应操作；如果是绿色的图标，表明是针对汇编源代码进行相应操作。

：单步跳出（Step Return），如果当前语句所在的函数是正在被调用的，单击后，该函数剩余部分被执行，并暂停在对应调用语句的下一行上。在 main 函数中单击这个图标，将完成整个工程的运行。

：CPU 复位（CPU Reset），单击后，将对 CCS 当前的仿真环境进行复位，所有调试信息恢复到刚进入 CCS Debug 视图时的状态。

：重开始（Restart），单击后，CCS 仿真环境的参数不变，但 CCS 重新回到 main 函数第一行并暂停。此时，前面执行过程中对环境参数的改变部分都将保留。

上面这些图标对应的操作，也可以通过单击菜单栏 RUN 菜单，在弹出下拉菜单中选择相应命令来执行。

现在回到当前的工程，单击“”图标，工程源代码执行到结束。此时在左下方的 Console 窗中显示出工程执行结果：Hello World!!，如图 3-11 所示。

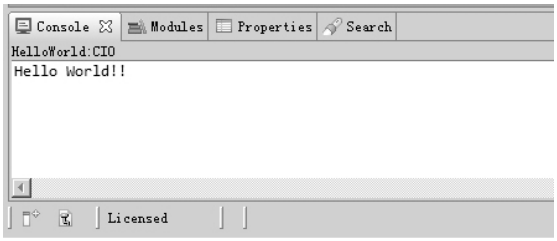

图 3-11　HelloWorld 的输出结果

5．中间结果观察

上一节中执行了整个工程后，输出结果显示在 Console 窗口中。但在前面我们提到，一般在 CCS 工程代码中不建议使用标准的输入/输出语句。此时，为查看代码执行步骤是否正确，或查看代码最终结果，可以利用设置断点、文件读/写等操作来观察代码执行的中间结果。

上面的 HelloWorld 例子比较简单，无法实现相应的要求。为此，下面我们将通过一个典型的信号产生例子，来介绍 CCS 开发中如何进行断点设置、数据输入/输出、结果观察。

在这个例子中，我们需要产生一个 500 点的数字信号序列，在 8000Hz 采样率下，其表现为 500Hz 和 2850Hz 两个单频信号的叠加，且这两个频率信号的起始相位都为 0。

这里，我们直接用 sin()函数产生单频信号，并将产生的两个信号相加得到最终结果。考虑到 TMS320VC5509A 的字长是 16 位，需要将信号幅值限制在 -32768～32767 之间，因此对相加后的信号乘以数值 16383。下面给出了 C 语言的实现代码。

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

int main(void)
{
    float a, b, x, y, pi;
    int i, z;
    short out[500];

    pi=3.14159;
    a = 500.0;              /*第一个信号的频率是 500Hz*/
    b = 2850.0;             /*第二个信号的频率是 2850Hz*/

    a = a/8000;
    b = b/8000;
    for(i=0;i<500;i++)
    {
        x=sin(i*a*2*pi);
        y=sin(i*b*2*pi);
        z = (short)(16383.0*(x+y));
        out[i] = z;
```

```
    }
    puts("finished!\n");
    return 0;
}
```

参照工程创建中的步骤建立一个 signal-produce 工程，其中除了工程名称外，其他所有选项与 3.3.1 小节中的相同。

参照上面给出的 C 源代码，编辑相应的 main 函数后，进行 Build 操作。

执行 Debug，进入 CCS Debug 透视图，如图 3-12 所示。

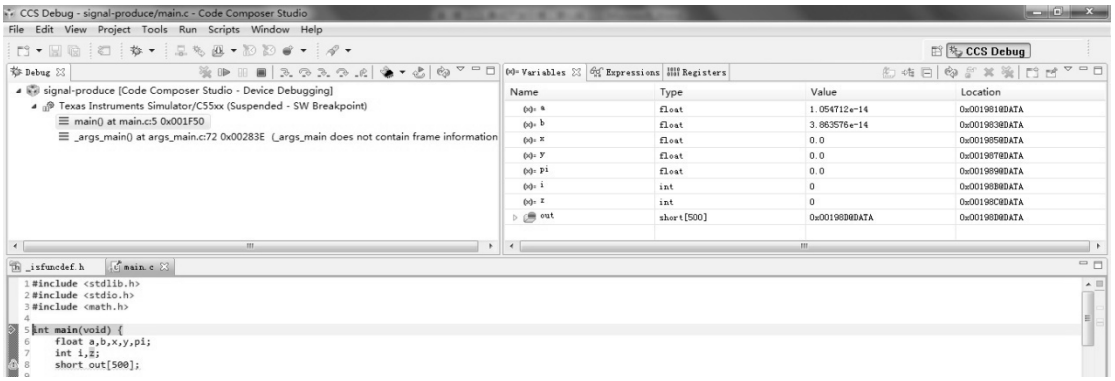

图 3-12　signal-produce 工程调试透视图

从图中可以看到，在右上方 Variables 窗的最左边列，列出了 main 函数中所定义的所有变量，在每一行中显示了对应变量的类型、数值、地址等信息。此时程序还没有执行到变量声明行，因此这里的数值、地址等内容是不确定的。

（1）断点设置

为了在代码执行的过程中，能够检查中间变量数值、输入输出变量等，可以在代码中的相关语句行设置断点。代码执行到断点后，CCS 将执行断点配置中所指定的任务。

将光标移到对应语句行上后，可以通过单击菜单的 Run→Toggle Breakpoint 选项来设置/清除断点，或通过双击对应语句行最左边灰蓝色部分来设置/清除断点。设置了断点的语句行在最左边的灰蓝色部分将出现一个蓝色的点“ ”，同时视图右上区域将出现 Breakpoint 窗口，其中用表格方式罗列了所设置所有断点的位置和配置。断点的配置可以在这个窗中进行，如图 3-13 所示。

双击 Breakpoint 窗中某个断点的左边方框，如果方框中出现“√”符号，则表示这个断点被使用；再次双击这个方框，可以去除“√”符号，表示这个断点被暂停使用。

选中窗中某断点行，右击鼠标，在跳出菜单中选择 Remove，可以去除这个断点。

在跳出菜单中选择 Break Propertise…选项，将出现一个断点配置窗，如图 3-14 所示。

单击右边窗中的 Action 选项，在其右边对应的 Values 行单击鼠标，将出现下拉按键。单击下拉按键，出现断点处的动作选项菜单。其中，默认的选项是 Remain Halted，当程序执行到断点处，将执行暂停，等待下一个操作。

为了在程序执行开始和结束时刻观察变量的值，我们在“pi =3.14159；”语句和“puts("finished!\n");”语句处分别设置了两个执行暂停的断点，以便于我们观察运算前和运算后的变量值。

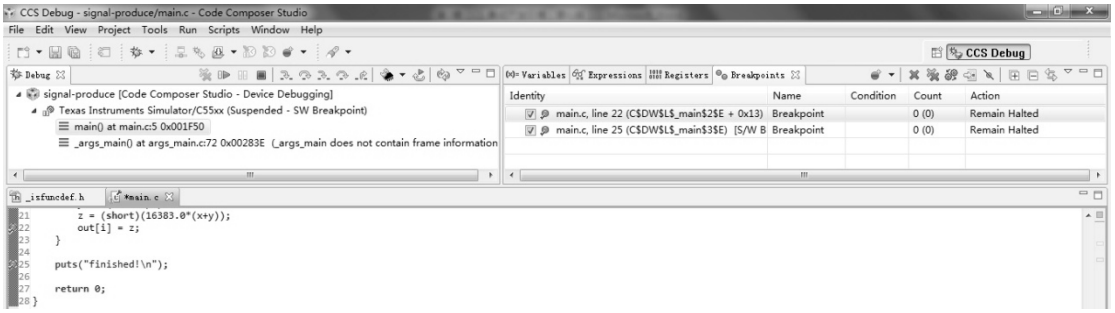

图 3-13　断点设置图

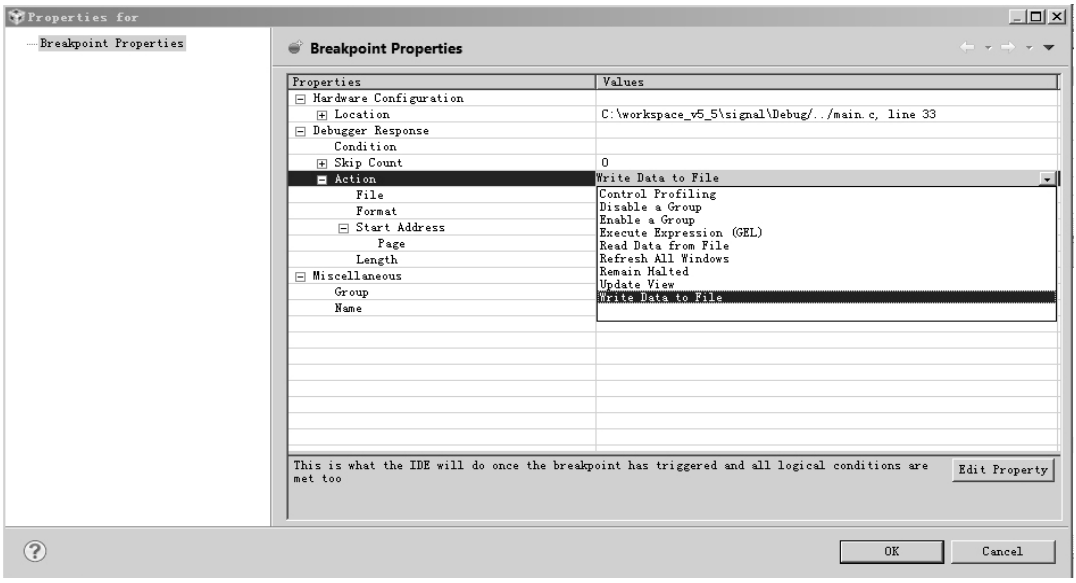

图 3-14　断点配置窗

（2）文件输入/输出

只在 Simulator 方式下，CCS 具有 Pin Connect 功能，允许用户利用文件输入方式模拟芯片某个管脚的输入，以便模拟外部中断信号；同时，CCS 也提供 Port Connect 功能，允许用户将存储区的某个部分与外部文件相关联，实现存储区数值从文件读入或向文件写出的操作。详细操作可以参看帮助文件。

下面介绍在当前工程中，如何通过断点配置方式将工程产生的波形数据写入外部文件中。

在“out[i] = z;”语句处设置断点，并将其执行的动作配置为文件输出。在图 3-14 的 Action 选项右边选择 Write Data to File 后，在其下方的配置设置如图 3-15 所示。其配置如下：

① File 栏指定了数值输出的文件名称，其后缀名为.dat。注意，如果文件是新建的，那么在文件名的后面一定要加上.dat 后缀。

② Format 栏了输出到文件中的数值的类型，有 6 个选项可以选择，分别是十六进制、整型、长整型、浮点、COFF 格式、可寻址单元。这里输出信号是 16 位整型数，因此选择 Integer 选项。

③ Start Address 栏了每次输出到文件中的数据的开始地址。这里设定将每次计算得出的信号幅值 z 输出到文件中，因此开始地址填为&z（也可以填写 z 在存储区中的十六进制地址）。该变量位于存储区的 DATA 区域，因此 Page 项会自动选为 DATA。