高等学校电工电子实践系列

DSP 系统设计与实践

雷　勇　主编

電子工業出版社
Publishing House of Electronics Industry
北京・BEIJING

内 容 简 介

本书是根据高等院校工科本科生“DSP 原理及应用”、“DSP 技术”、“DSP 设计与实践”等课程的基本要求编写的。

书中实验及实践内容丰富，主要介绍了以美国得克萨斯州仪器公司（TI）TMS320C54X 系统为基础的 DSP 系统，在 CCS 入门实验、算法实验、DSP 接口实验的基础上，还加入了大量 DSP 实时操作系统的 DSP/BIOS 实验，并对 DSP/BIOS 做了详细说明。

本书可作为高等学校电类和部分非电类本科生（专科生）的实验教材，也可作为电视大学、职业大学、业余大学及远程教育、网络教育的实验教学用书。

图书在版编目（CIP）数据

DSP 系统设计与实践/雷勇主编. —北京：电子工业出版社，2004.8
（高等学校电工电子实践系列）
ISBN 7-121-00194-2

Ⅰ.D… Ⅱ. 雷… Ⅲ. 数字信号—信号处理—系统设计—高等学校—教材 Ⅳ. TN911.72

中国版本图书馆 CIP 数据核字（2004）第 077808 号

策划编辑：章海涛
责任编辑：章海涛 王羽佳
印 刷：
出版发行：电子工业出版社
北京市海淀区万寿路 173 信箱 邮编 100036
经 销：各地新华书店
开 本：787×980 1/16 印张：14 字数：312 千字
印 次：2004 年 8 月第 1 次印刷
印 数：5 000 册 定价：20.00 元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系。联系电话：（010）68279077。质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

前　言

数字信号处理器（DSP）问世以来，由于其结构独特并具有快速实现各种数字信号处理算法的优点，因此在多个领域得到了广泛的应用，发展十分迅速。为适应这一发展趋势，我国许多高校都开设了 DSP 的相关课程和实验。在吸取其他 DSP 相关教材经验的基础上，根据近几年四川大学“ DSP 原理及应用 ”课程的教学经验与当前教学改革的要求，我们编写了本书。通过对该门课程的学习，学生能由浅入深地掌握 TMS320C54X 系列 DSP 的基本原理、系统组成和软、硬件开发，将理论知识和实际操作相结合，在学习过程中逐步培养和提高自身的实验能力、实际操作能力、设计能力、独立分析和解决问题的能力，以及创新思维能力和理论联系实际的能力。

本书实验内容丰富，并且遵从循序渐进的原则，由浅入深，从 TMS320C54X 系列 DSP 入手，分为 CCS 入门基本操作实验、DSP 算法实验、DSP 接口实验和 DSP/BIOS 实验。其中，入门基本操作实验包括 CCS 软件使用，算法实验包括定点算术运算实验、浮点算术运算实验、数字滤波器实验、FFT 实验，接口实验包括定时器实验、C54X 串口通信与 A/D、D/A 接口实验。

本书的一大特点就是理论与实践相结合，着重于实践，针对 DSP 的难点，由浅入深地进行教学。特别是对于相对较难的 DSP/BIOS 这一环节，本书从原理和实践两方面进行了通俗易懂的阐述，并配有大量 DSP/BIOS 源程序，使得 DSP/BIOS 部分更容易得到理解，使读者通过本书的学习基本掌握 DSP 的应用。

本书由雷勇主编。其中实验一由李雷编写，实验二由雷勇编写，实验三、六、七由雷勇、徐雪梅共同编写，实验四、五由雷勇、涂国强共同编写，技术准备、实验八至十三由张行编写，附录 A 由雷勇、涂国强、李雷共同编写，附录 B 由张行、徐雪梅共同编写。

本书由四川大学电气信息学院宁元中教授主审，参加本书审稿工作的还有四川大学电气信息学院周群副教授、吴志红副教授、刘婕副教授和杨刚工程师，在此表示衷心感谢!

本书在编写过程中得到四川大学电工电子基础教学实验中心的同志们的大力支持和帮助，同时还获得四川大学电气信息学院各位领导的鼓励和支持，以及北京闻亭公司、成都思莱特公司的技术支持，在此一并表示感谢!

本书不足之处，恳请读者批评指正。

编　者

2004 年 5 月于四川大学

目　录

上　篇

基 础 实 验

实验一　CCS 基本操作实验

一、实验目的

熟悉 CCS 软件的基本使用方法，重点掌握 CCS 软件的调试工具和技巧。

二、实验内容

1. 创建新项目文件

本次练习中将使用 Code Composer Studio 程序（以下简称 CCS）创建一个新的项目文件（project），然后将源代码文件（source code files）和库文件（libraries）加入到该项目文件中。

（1）如果 CSS 安装在“ C:\ti ”下，则在“ C:\ti\myprojects ”目录下创建一个名为 volume1 的目录；如果 CSS 安装在其他目录下，则将 volume1 目录创建在相应位置的 myprojects 目录下。

（2）将“ C:\ti\tutorial\target\volume1 ”目录内所有文件复制到新建目录下。

（3）如果未启动 Code Composer Studio 2（C5000）程序，则在【开始】菜单中选择【程序】→ Texas Instruments→ Code Composer Studio→ Code Composer Studio，或双击桌面上的 Code Composer Studio 图标，则 CSS 启动，界面如图 1-1 所示。

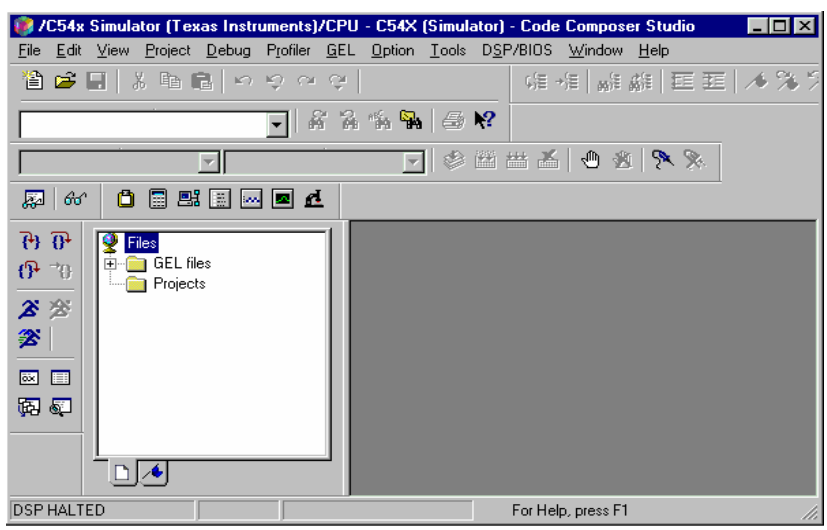

图 1-1　Code Composer Studio 界面

（4）在 Project 菜单中选择 New 选项，显示如图 1-2 所示的项目创建菜单。

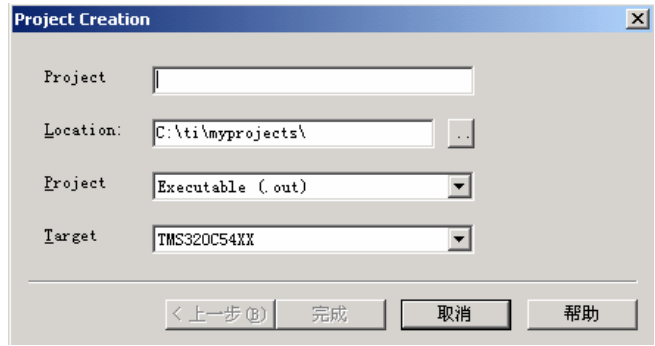

图 1-2　项目创建菜单

（5）在 Project Name 文本框中填入“volume1”。

（6）在 Location 文本框中单击…按钮（browse），选择在步骤（1）中创建的工作目录。

（7）在 Project Type 下拉框中选择 Executable (.out)。

（8）在 Target 下拉框中选择在 CCS 配置设备中使用的目标板（或仿真目标板），然后单击【完成】按钮。

至此，CSS 已经创建了一个文件名为 volume1.pjt 的项目文件，该文件存储了项目的设置和与项目中使用的各种文件的关联。

2. 向项目中加入文件

（1）选择 Project→Add Files to Project，选择 volume.c 并单击 Open，也可通过以下方式将文件加入到项目中：在项目文件察看窗口（Project View）中的 Project 按钮上单击右键，并从弹出菜单中选择 Add Files，将文件加入项目，或将所需文件拖入项目文件察看窗口。

（2）选择 Project→Add Files to Project，在文件类型下拉框（Files of type box）中选择 asm 源文件（*.a*，*.s*），选择 vectors.asm 和 load.asm 并单击 Open。这两个文件包含了设置程序从复位中断（0FF80H）处转向 C 语言入口 c_int00 的汇编指令。对于更复杂的程序，可以在 vectors.asm 中定义附加中断向量，或使用 DSP/BIOS 软件模块自动定义所有中断向量。

（3）选择 Project→Add Files to Project，在文件类型下拉框中选择链接命令文件 volume.cmd 并单击 Open。该文件将源程序中定义的程序段、数据段与堆栈段等映射到内存中。

（4）选择 Project→Add Files to Project，转到编译库文件存放目录“C:\ti\c5400\cgtools\lib”。在文件类型下拉框中选择目标文件和库文件（*.o*，*.lib），选择 rts.lib（该库

文件为 DSP 目标板提供了实时执行支持）并单击 Open。但对某些目标板而言，实时执行库有其他特殊的名称，如 rts_ext.lib。此时双击项目文件察看窗口的 volume.pjt（如图 1-3 所示），则可见到加入的文件。

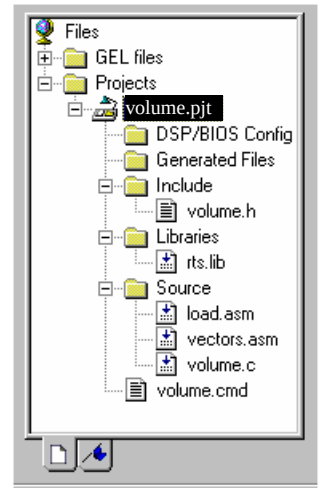

图 1-3　项目文件察看窗口

注意：开启项目文件察看窗口时，如果看不到项目文件察看窗口，则选择 View→Project；如果书签按钮被选中，则单击文件察看窗口下方的文件按钮 Files。

因为 CSS 能自动搜寻在编译过程中所需的支持文件，所以不需要手动将 Include 文件加入项目文件中。当编译项目文件时，在项目文件察看窗口中会自动出现 Include 文件。

如果需要从项目文件察看窗口中删除某个文件，只需在项目文件察看窗口中选中文件，然后单击右键并在弹出菜单中选择 Remove from project 即可。

编译程序时，CSS 按下列路径次序为项目文件搜寻所需文件：

- 包含源文件的目录。
- 编译和汇编菜单中 Include 文件搜索路径中列出的目录（从左到右）。

3. 检查源程序

在项目文件察看窗口中双击 volume.c 文件，源程序代码会出现在 CCS 窗口的右半窗口。在此 C 程序中应注意以下部分：

- 主函数 main 输出第一条信息后，就进入一个无限循环。在该循环内，主函数调用了函数 dataIO 和 processing。
- 函数 processing 将输入缓冲区中取出的每一个值乘以增益（gain），然后将结果送入输出缓冲区。它也调用了汇编加载程序，这在变量调入的处理过程中会消耗大量指令周期。
- 本程序中的函数 dataIO 完成返回（return）操作，使用探针从文件中将数据读入缓冲区，其效率大大超过了用 C 代码来完成 I/O 操作。

volume.c 程序清单如下：

```
#include <stdio.h>
#include "volume.h"
/* Global declarations */
int inp_buffer[BUFSIZE]; /* processing data buffers */
int out_buffer[BUFSIZE];
int gain = MINGAIN; /* volume control variable */
```

```
unsigned int processingLoad = BASELOAD; /* processing load */
struct PARMS str =
{
   2934,
   9432,
   213,
   9432,
   &str
};
/* Functions */
extern void load(unsigned int loadValue);
static int processing(int *input, int *output);
static void dataIO(void);
/* ======== main ======== */
void main()
{
   int *input = &inp_buffer[0];
   int *output = &out_buffer[0];
   puts("volume example started\n");
   /* loop forever */
   while(TRUE)
   {
      /* Read using a Probe Point connected to a host file.     */
      /* Write output to a graph connected through a probe-point. */
      dataIO();
      #ifdef FILEIO
      puts("begin processing");
      #endif
      /* apply gain */
      processing(input, output);
   }
}
/* ======== processing ======== *
* FUNCTION: apply signal processing transform to input signal.
* PARAMETERS: address of input and output buffers.
* RETURN VALUE: TRUE. */
```

```
static int processing(int *input, int *output)
{
    int size = BUFSIZE;
    while(size--){
        *output++ = *input++ * gain;
    }
    /* additional processing load */
    load(processingLoad);
    return(TRUE);
}
/* ======== dataIO ======== *
* FUNCTION: read input signal and write output signal.
* PARAMETERS: none.
* RETURN VALUE: none. */
static void dataIO()
{
    /* do data I/O */
    return;
}
```

4. 编译、运行程序

CSS 能在改变项目文件时自动存储，正如刚建立它时一样。如果在进行某操作时意外退出 CSS，可再次启动 CSS 并选择 Project→Open，则返回退出前的中断处。

注意：要重新设置 DSP 目标板。如果 CSS 显示出一条错误信息，该信息指明程序不能初始化 DSP 目标板，选择 Debug→Reset DSP 菜单选项。如果仍不能纠正这一错误，就应运行目标板配套的复位（reset）应用程序来完成。

编译、运行程序，可按以下步骤完成：

（1）选择 Project→Rebuild All 或单击 （Rebuild All）按钮，CSS 将重新编译、组织、链接项目中的所有文件。整个过程的信息将在窗口下方的信息框内显示。

（2）默认方式编译过程中，在当前项目文件存放目录下将生成一个 Debug 子目录（如图 1-4 所示），编译生成的 out 文件将存放在该目录下。也可使用 CCS 工具栏选择其他位置存放该文件。

图 1-4　编译生成 Debug 子目录

（3）选择 File→Load Program，选中刚编译成功的程序文件 volume1.out，然后单击 Open（该文件将在“C:\ti\myprojects\volume1\Debug\”目录下，除非目标文件存放在其他位置），CSS 将目标程序加载到 DSP 目标板（或仿真目标板），并打开一个显示程序编译、链接后生成汇编指令代码的窗口。此时，CSS 将在主窗口下方打开一个显示程序输出结果的小窗口。

（4）选择 View→Mixed Source/ASM，这样可以同时察看 C 源程序和最终的汇编代码。

（5）在混合显示窗口中单击汇编指令（单击指令本身，而非该指令的地址或指令所在的区域），按 F1 键，CSS 寻找该指令的帮助文件。这是一个熟悉未知汇编指令的好办法。

（6）选择 Debug→Go Main，使程序从主函数 main 处开始执行，这一进程将在 main 处暂停显示“”符号。

（7）选择 Debug→Run 或单击工具条中的 （Run）。

（8）选择 Debug→Halt，使程序退出运行状态。

（9）在 View 菜单中选择 Mixed Source/ASM，将只显示 C 代码，不再显示汇编指令。

5. 修改程序设置与更正语法错误

因为 FILEIO 还未定义，前面被预处理程序（#ifdef and #endif）附加的部分并未执行。接下来，我们用 CSS 设置预处理选项，并寻找和更正一处语法错误。

（1）选择 Project→Build Options，显示如图 1-5 所示的窗口。

（2）在编译窗口的编译器选择栏的列表选项中选择 Preprocessor，在 Define Symbols 栏中输入 FILEIO，并按下 Tab 键。

注意：此时窗口上方的编译命令包括 define Symbols[-d]选项（见图 1-5）。当重新编译程序时，#ifdef FILEIO 后的代码将被包括在其中（其他选项将会随所使用的目标板不同而不同）。

（3）单击 OK，保存新的设置选项。

（4）选择 Project→Rebuild All 或单击 （Rebuild All）按钮（项目文件改变时需要重新编译所有文件）。

（5）一个编译信息显示文件中包括编译错误，单击 Build 栏并移动 Build 栏中的滚动条，将会看到出现了一个语法错误，如图 1-6 所示。

（6）双击红色文本行，该行指明了语法错误的位置（line 68）。此时 volume.c 源文件被打开了，并且光标停留在下面这一行上：

```
processing(input, output);
```

（7）更正光标所在行的上面一行的语法错误（缺少分号），则该行显示如下：

```
puts("begin processing");
```

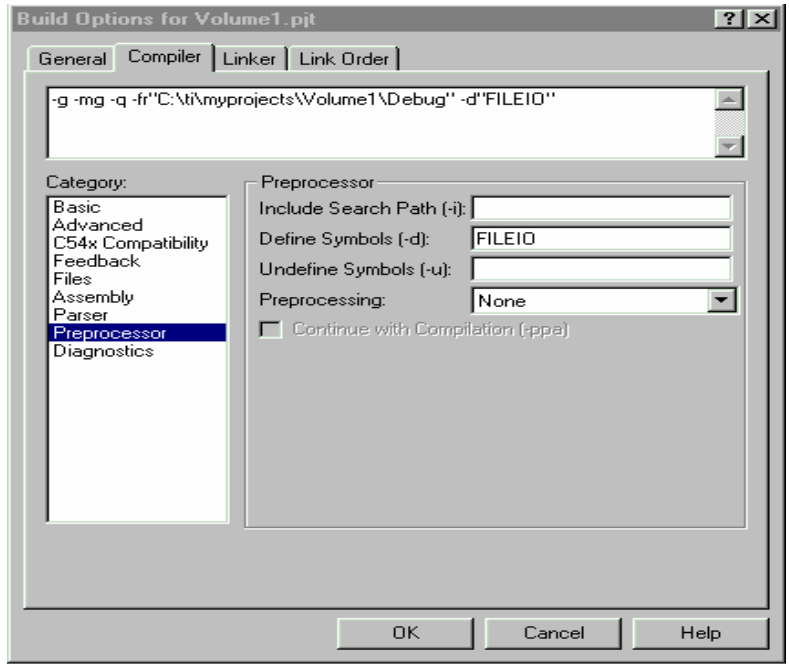

图 1-5　Build Options **窗口**

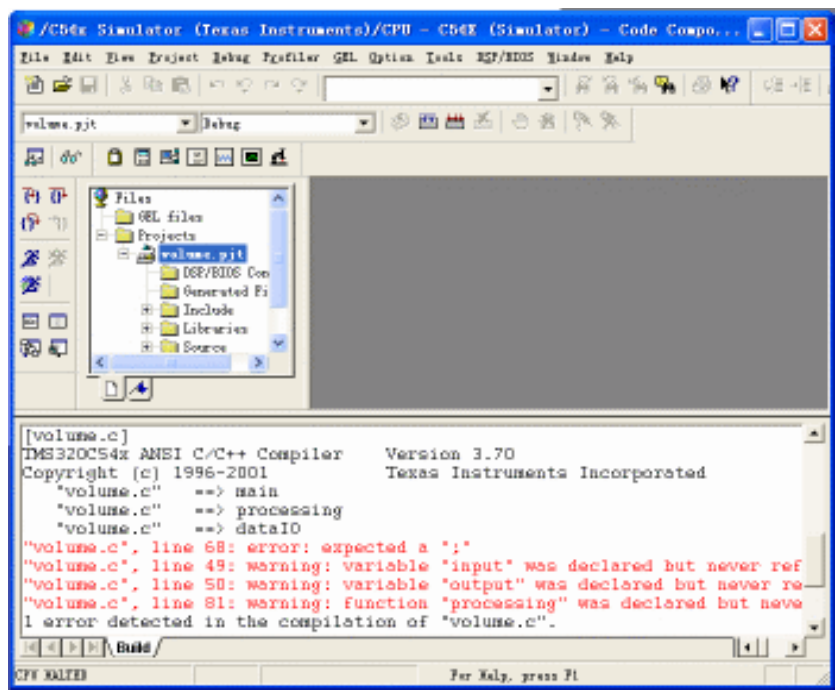

图 1-6　**编译信息显示窗口**

（8）此时在编辑窗口标题栏的文件名右边会出现一个星号（*），这表明源文件已被更改，当文件存盘后星号会自动消失。

（9）选择 File→Save 或按 Ctrl+S 保存更正后的 volume.c 文件。

（10）选择 Project→Build 或单击 （Incremental Build）按钮，CSS 将新重编译替换后的文件。

（11）选择 File→Load Program 并选中 volume1.out，可以通过定制（customizing）项目环境，以便每次在编译后自动调入程序 选择 Option→Customize，并单击 Program Load Options 选项，然后选择 Load Program After Build Option 选项。

（12）选择 Debug→Go Main，使程序从主函数 main 处开始执行，这一进程将在 main 处暂停并由 键确定。

（13）选择 Debug→Run 或单击工具条中的 （Run），显示如图 1-7 所示的程序运行窗口。

（14）选择 Debug→Halt，使程序退出运行状态。

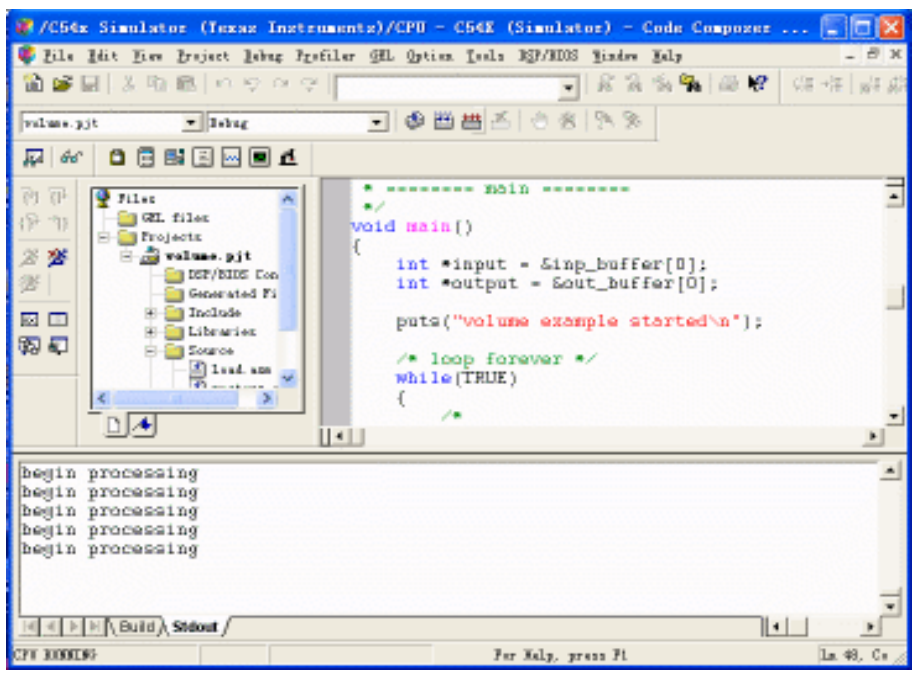

图 1-7　程序运行窗口

6. 使用断点（Breakpoints）与观察窗口（Watch Window）

当开发和测试一个程序时，经常需要在程序运行过程中检查一个变量的值。这里使用断点（Breakpoints）与观察窗口（Watch Window）来观察这些变量，也可在到达断点后使用 step 命令。

（1）选择 File→Reload Program。

（2）在项目文件察看窗口中双击 volume.c 文件，应尽量放大窗口以便一次看到更多的源程序指令。

（3）将光标移到下面这行：

```
dataIO();
```

（4）单击（Toggle Breakpoint）或按 F9 键，空白选择区（selection margin）将指明一个断点已经被设置（红色图标）。如果禁止空白选择区（selection margin），该行将显示为紫红色高亮条（可以通过 Option→Customize→Color 来更改颜色）。

（5）选择 View→Watch Window，一个单独的窗口将出现在 CCS 窗口的右下方，如图 1-8 所示。在程序运行中，该区域将显示出所观察的变量的值。在默认方式下，首先选中的是 Watch Locals 选项，在函数执行时，局部变量被显示出来。

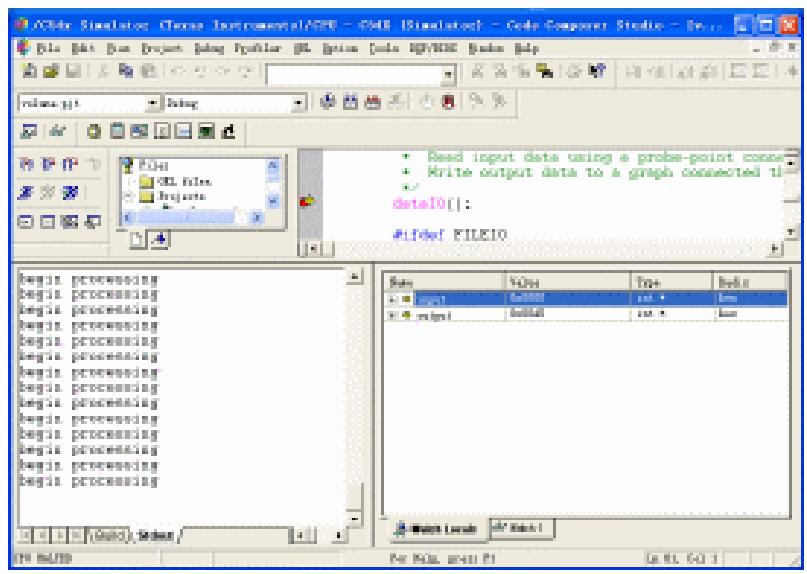

图 1-8　变量观察窗口（一）

（6）程序如果未回到 main 处，选择 Debug→Go Main。

（7）选择 Debug→Run，或按 F5 键，或单击工具条中的（Run）。

（8）选择 Watch1 选项。

（9）在 Name 栏单击表达式按钮，并将 dataIO 作为变量名输入。

（10）单击观察窗口的空白处保存设置，变量值会立即显示出来，如图 1-9 所示。

（11）单击（Step Over）或按 F10 键单步调用执行指令 dataIO( )。

验证 CSS 提供的 step 命令如下：

- Step Into (F8)

- Step Over (F10)
- Step Out (Shift +F7)
- Run to Cursor (Ctrl +F10)

（12）结束本练习后，在进行下一个练习之前单击（Remove All Breakpoints）按钮，清除所有断点。

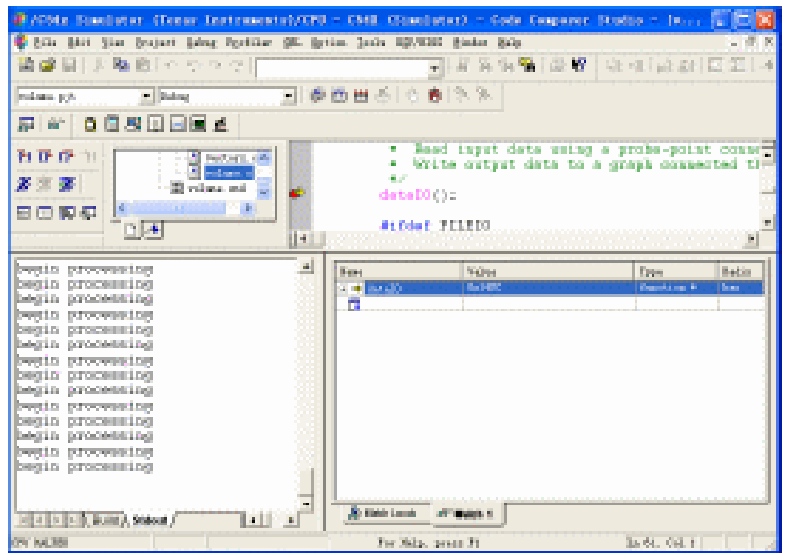

图 1-9　变量观察窗口（二）

7. 使用观察窗口查看结构体变量

除查看简单变量值外，还可利用观察窗口查看结构体元素的变量值。

（1）选择 Watch1 选项。

（2）在 Name 栏单击表达式按钮，并将 str 作为变量名输入，如图 1-10 所示。

（3）单击观察窗口的空白处保存设置，变量值会立即显示出来（如图 1-11 所示）。

（4）重新查看源程序会发现一个 PARMS 类型的结构体变量已经在 volume.c 中被初始化并定义为全局变量。该结构体类型是在 volume.h 中定义的。

（5）单击 str 旁的“ + ”，CSS 将扩展该行并列出该结构体中的所有元素及其值（显示出的链接可能有所不同），通过双击其中元素值可更改选中的元素值。

（6）在观察窗口的 Value 列中，改变一个变量的值。注意此时观察窗口的值已经改变，同时该变量的颜色变为红色，表明已经改变了该变量的值。

（7）窗口中选择 str 变量并按 Delete 键，并对观察窗口中的所有表达式重复该步骤。

（8）选择 Debug→Breakpoints，在 Breakpoints 栏中单击 Delete All 并单击 OK。

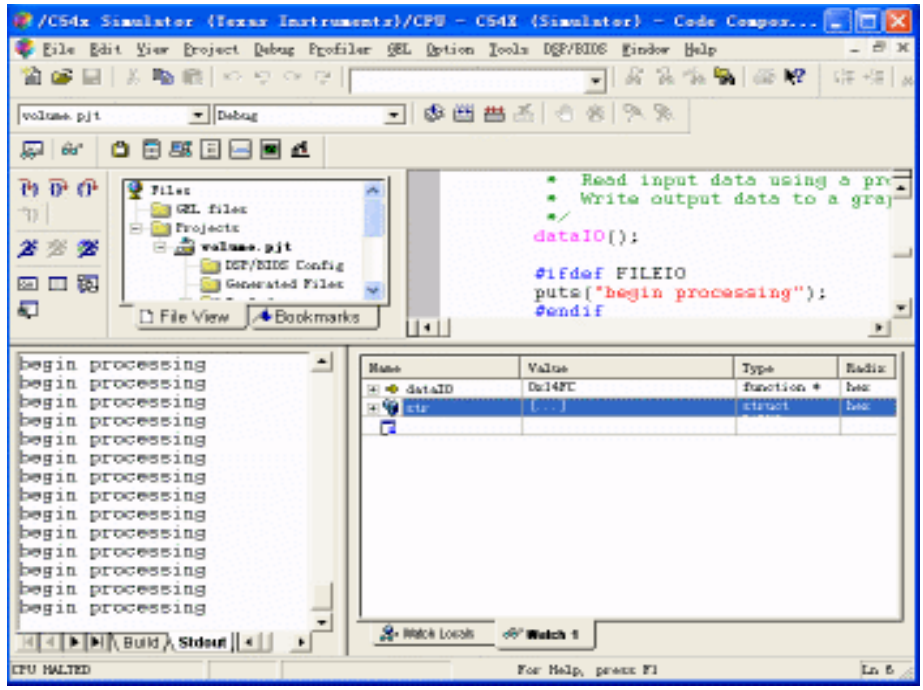

图 1-10　变量观察窗口（三）

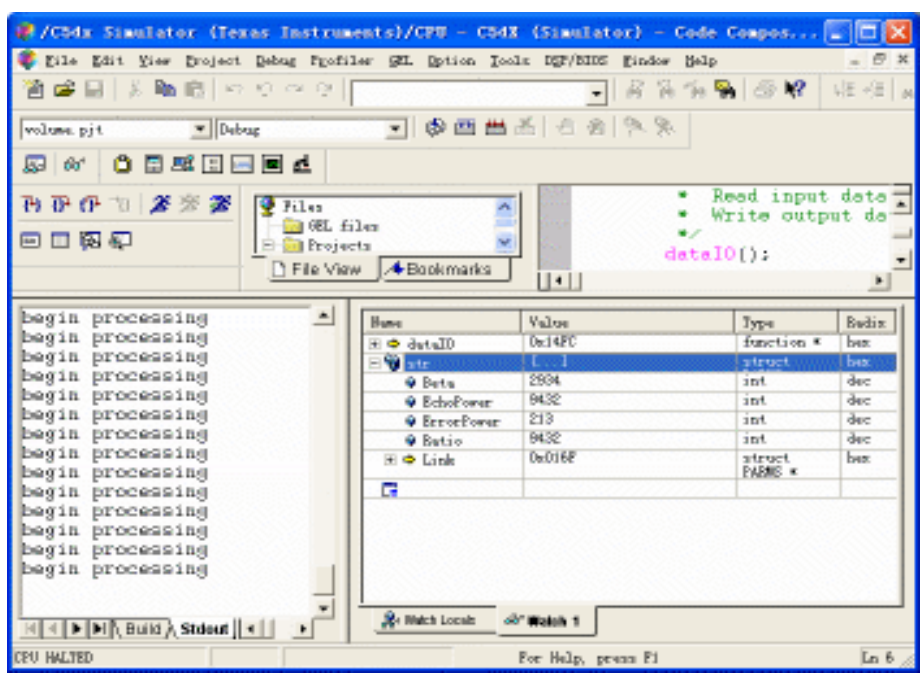

图 1-11　变量观察窗口（四）

8. 加入文件输入/输出探针（Probe Point）

下面的内容中加入了一个能从计算机文件中读入数据的探针。探针是开发运算程序的有用工具，可以通过下列方式使用：

- 从主机文件中将算法所需的输入数据传送到目标板缓冲区。
- 从目标板缓冲区输出数据到一个计算机文件以用于分析。
- 使用数据更新窗口，如图形。

在暂停目标程序运行方面使用探针和中断很相似。但是，探针与中断在以下方面是有所不同的：

- 探针立即停止程序运行，执行一个单独的任务后，程序继续运行。
- 中断暂停 CPU 运行，直到手工使其继续运行，并刷新所有窗口。
- 探针允许在运行过程中自动进行文件输入或输出操作，这是中断无法完成的。

本练习显示如何使用探针从计算机文件中读入目标所需的测试数据。当探针到达时，也可使用中断来刷新所有窗口。

（1）选择 File→Load Program，然后选择 volume.out 文件，再单击 Open。

（2）在项目文件察看窗口中双击 volume.c 文件。

（3）把光标放在主程序（main function）的下面这行：

```
dataIO();
```

函数 dataIO 是作为占符位使用的，在稍后加入。目前，放置探针是从计算机文件中读入数据的很方便的方法。

（4）单击  （Toggle Probe Point）按钮，在空白区域（selection margin）处显示一个探针已被设置（蓝色图标）。如果禁止空白区域（Option→Customize→Editor Properties），该行则以高亮蓝色显示（可以通过 Option→Customize→Color 改变颜色）。

（5）在 File 菜单中选择 File I/O，这时 File I/O（文件输入输出）对话框显示出来，如图 1-12 所示，从中选择输入或输出文件。

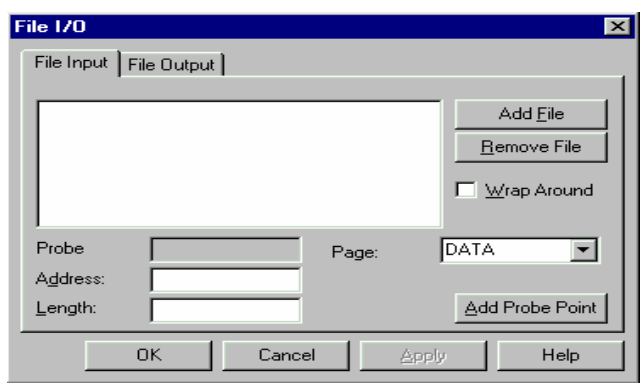

图 1-12　File I/O 对话框