

# 基于 Proteus 的 51 系列单片机 设计与仿真 (第2版)

■ 陈忠平 编著



电子工业出版社

PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

<http://www.phei.com.cn>

■ 聚焦 EDA

# 基于 Proteus 的 51 系列单片机 设计与仿真 (第2版)

■ 陈忠平 编著

电子工业出版社

Publishing House of Electronics Industry

北京 · BEIJING

## 内 容 简 介

本书以目前流行的软硬件仿真软件 Proteus 为核心，采用现代教学方法，从实验、实践、实用的角度出发，通过丰富的实例详细讲述了 Proteus 软件在 51 单片机课程教学和单片机应用产品开发过程中的应用。

本书以夯实基础，面向应用，理论与实践紧密结合为原则，采用汇编及 C 语言作为系统软件开发平台。全书共 9 章，主要包括 80C51 单片机系统设计相关软件的使用、Proteus7.8 入门、51 系列单片机软件程序设计、51 系列通用 I/O 控制、LED 数码管与键盘的应用、D/A 转换器和 A/D 转换器的应用、显示器的应用、电动机控制，以及综合应用设计。

本书适合从事单片机应用研发的科技人员自学使用，也可作为高等学校单片机课程的教学用书。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有，侵权必究。

### 图书在版编目（CIP）数据

基于 Proteus 的 51 系列单片机设计与仿真 / 陈忠平编著. —2 版. —北京：电子工业出版社，2012.5  
(聚焦 EDA)

ISBN 978-7-121-16910-6

I. ①基… II. ①陈… III. ①单片微型计算机—系统设计—应用软件 ②单片微型计算机—系统仿真—应用软件 IV. ①TP368.1

中国版本图书馆 CIP 数据核字（2012）第 084936 号

责任编辑：张 剑 (zhang@phei.com.cn)

印 刷：三河市双峰印刷装订有限公司  
装 订：

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

开 本：787×1 092 1/16 印张：26 字数：666 千字  
印 次：2012 年 5 月第 1 次印刷  
印 数：4 000 册 定价：59.80 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，  
联系及邮购电话：(010) 88254888。

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。

服务热线：(010) 88258888。

## 第2版前言

单片机又称为单片微处理器，其实质是将一个简单的计算机系统集成在一个芯片上。单片机具有体积小、质量小、价格便宜、控制功能强等特点，在工业控制、智能仪表、家用电器和军事装置等领域都得到了广泛的应用。

本书第1版于2008年9月出版，已被许多学校或培训机构作为单片机课程的实践教材使用，受到众多教师、学生和读者的欢迎。

在坚持第1版“由浅入深，循序渐进”、“软硬结合，虚拟仿真”、“C语言与汇编语言并存”、“淡化原理，注重实用”的编著原则基础上，本书第2版根据使用教师和读者的建议进行了修订与补充。

与第1版相比，本书第2版主要在以下方面进行了修订。

- 删除了第1章中的80C51单片机应用的设计部分，增加了STC89系列单片机下载内容。
- 第2章增加了Keil C51与Proteus的联机调试部分内容。
- 第3章至第8章采用了汇编语言与C语言两种语言编写程序。
- 第4章增加了单片机与PC的通信部分内容。
- 删除了音乐的应用这一章的内容。
- 将共阴极LED和共阳极LED的应用这一节的内容，改为一位共阳极LED数码管静态显示。
- 在D/A转换器和A/D转换器的应用这一章中增加了DAC0832和ADC0832的应用。
- 在显示器的应用这一章中，删除了两个8×8点阵滚动显示、两个16×16点阵汉字显示、两个16×16点阵汉字分批显示；增加了字符式LCD静态显示、汉字式LCD滚动显示。
- 将工业控制这一章的内容划分为电动机控制和综合应用设计两章。其中电动机控制这一章中，讲解了步进电动机和直流电动机的基本知识；综合应用设计这一章中，讲解了数字电子钟、篮球计分器、DS1302可调时钟、24C04开启次数统计、DS18B20测量温度、按键选播电子音乐等内容。

参加本书修订工作的有湖南工程职业技术学院陈忠平、徐刚强、龚亮、李锐敏、龙晓庆，湖南航天局7801研究所刘琼，湖南涉外经济学院侯玉宝、高金定，湖南科技职业技术学院高见芳，湖南三一重工集团王汉其等，全书由湖南工程职业技术学院陈建忠教授主审。在本书的修订过程中，许睿等同志给予了大力支持及帮助，在此深表感谢。

由于作者水平有限，书中难免有错漏之处，恳请读者予以指正或提出修改意见。

编著者

2012年3月

# 目 录

<b>第 1 章 80C51 单片机系统设计相关软件的使用</b>	1
1.1 Keil C51 的使用	1
1.2 仿真器	11
1.3 编程器	12
1.4 ISP 下载	14
1.5 串行调试软件	16
<b>第 2 章 Proteus7.8 入门</b>	18
2.1 Proteus ISIS 的操作及电路原理图设计	18
2.1.1 Proteus ISIS 编辑环境及参数设置	18
2.1.2 Proteus ISIS 原理图设计	31
2.1.3 Proteus ISIS 元器件制作	39
2.2 Proteus VSM 虚拟系统模型	41
2.2.1 激励源	41
2.2.2 Proteus VSM 虚拟仪器的使用	42
2.2.3 Keil C51 与 Proteus 的联机调试	54
2.3 Proteus ARES 的 PCB 设计	56
2.3.1 Proteus ARES 简介	56
2.3.2 Proteus ARES 参数设置	61
2.3.3 Proteus ARES 中 PCB 制作实例	63
<b>第 3 章 51 系列单片机软件程序设计</b>	68
3.1 清零与置数位程序的设计	68
3.1.1 片内清零程序的设计	68
3.1.2 置数程序的设计	71
3.2 拼字与拆字程序的设计	73
3.2.1 片内拼字程序的设计	73
3.2.2 片内拆字程序的设计	76
3.3 数据块传送与排序程序的设计	78
3.3.1 数据块传送程序的设计	78
3.3.2 数据排序程序的设计	81
<b>第 4 章 51 系列通用 I/O 控制</b>	85
4.1 P1 端口的应用（一）	85
4.2 P1 端口的应用（二）	88

4.3	闪烁灯 .....	93
4.4	流水灯 .....	97
4.5	花样灯（一） .....	100
4.6	花样灯（二） .....	104
4.7	模拟交通灯 .....	109
4.8	定时/计数器的应用（一） .....	113
4.9	定时/计数器的应用（二） .....	116
4.10	中断系统的应用（一） .....	118
4.11	中断系统的应用（二） .....	122
4.12	两个单片机之间的串行通信 .....	127
4.13	串行口扩展应用 .....	132
4.14	单片机与 PC 之间的通信 .....	135
<b>第 5 章</b>	<b>LED 数码管与键盘的应用 .....</b>	<b>143</b>
5.1	LED 数码管应用 .....	143
5.1.1	LED 数码管的结构及分类 .....	143
5.1.2	LED 数码管的显示方式 .....	144
5.1.3	一位共阳极 LED 数码管静态显示 .....	145
5.1.4	串行口驱动一位共阴极 LED 数码管显示 .....	148
5.1.5	0~99 计数器的设计 .....	152
5.1.6	59s 计时器的设计 .....	156
5.1.7	8 位 LED 数码管移位显示 .....	160
5.1.8	MAX7219 串行驱动 LED 数码管 .....	166
5.2	键盘的应用 .....	177
5.2.1	键盘工作原理 .....	177
5.2.2	查询式键盘设计 .....	178
5.2.3	矩阵式键盘的识别（一） .....	182
5.2.4	矩阵式键盘的识别（二） .....	190
<b>第 6 章</b>	<b>D/A 转换器和 A/D 转换器的应用 .....</b>	<b>196</b>
6.1	DAC0832 D/A 转换器的应用 .....	196
6.1.1	DAC0832 输出方波 .....	197
6.1.2	DAC0832 输出三角波 .....	200
6.1.3	DAC0832 输出正弦波 .....	202
6.2	TLC5615 D/A 转换器的应用 .....	205
6.2.1	TLC5615 输出锯齿波 .....	207
6.2.2	TLC5615 输出阶梯波 .....	212
6.3	A/D 转换器的应用 .....	216
6.3.1	ADC0808 数字电压表的设计 .....	217

6.4 ADC0832 A/D 转换器的应用 .....	224
6.4.1 ADC0832 数字电压表的设计 .....	225
<b>第 7 章 显示器的应用 .....</b>	<b>233</b>
7.1 点阵 LED 的应用 .....	233
7.1.1 一个 5×7 点阵字符显示 .....	234
7.1.2 一个 8×8 点阵字符显示 .....	237
7.1.3 两个 8×8 点阵字符显示 .....	241
7.1.4 16×16 点阵汉字显示 .....	247
7.2 LCD 液晶显示器的应用 .....	254
7.2.1 字符式 LCD 字符串显示 .....	260
7.2.2 字符式 LCD 静态显示 .....	265
7.2.3 汉字式 LCD 静态显示 .....	271
7.2.4 汉字式 LCD 滚动显示 .....	291
<b>第 8 章 电动机控制 .....</b>	<b>303</b>
8.1 步进电动机控制 .....	303
8.1.1 步进电动机正转控制 .....	304
8.1.2 步进电动机的起动、停止控制 .....	307
8.1.3 步进电动机正、反转控制 .....	310
8.1.4 步进电动机转速控制 .....	314
8.2 直流电动机控制 .....	317
8.2.1 直流电动机的起动、停止控制 .....	319
8.2.2 直流电动机正、反转控制 .....	322
8.2.3 直流电动机转速控制 .....	326
8.2.4 直流电动机多动控制 .....	329
<b>第 9 章 综合应用设计 .....</b>	<b>333</b>
9.1 数字电子钟的设计 .....	333
9.2 篮球计分器的设计 .....	337
9.3 DS1302 可调时钟的设计 .....	346
9.4 24C04 开启次数统计 .....	368
9.5 DS18B20 测量温度 .....	379
9.6 按键选播电子音乐 .....	389
<b>附录 A 汇编指令速查表 .....</b>	<b>396</b>
<b>附录 B C51 库函数 .....</b>	<b>400</b>
<b>附录 C Proteus 常用快捷键 .....</b>	<b>405</b>
<b>参考文献 .....</b>	<b>406</b>

# 第1章 80C51 单片机系统设计相关软件的使用

单片机应用系统是以单片机为核心，同时配以相应的外围电路及软件来完成某些功能的系统。它包括硬件和软件两部分，硬件是系统的躯体，软件是系统的灵魂。本章主要讲述相关软件的使用。

## 1.1 Keil C51 的使用

单片机的源程序是在哪里进行编写的？是在哪里将其调试并生成.HEX 文件的？其实这些工作在单片机的一些编译软件中就可以完成。单片机程序的编译调试软件比较多，如 51 汇编集成开发环境、伟福仿真软件、Keil 单片机开发系统等。

Keil C51 是当前使用最广泛的基于 80C51 单片机内核的软件开发平台之一，由德国 Keil Software 公司推出。 $\mu$ Vision4 是 Keil Software 公司推出的关于 51 系列单片机的开发工具。 $\mu$ Vision4 集成开发环境 IDE 是一个基于 Windows 的软件开发平台，集编辑、编译、仿真于一体，支持汇编语言和 C 语言的程序设计。一般来说，Keil 51 和 $\mu$ Vision4 指的是 $\mu$ Vision4 集成开发环境。

Keil C51 可以从相关网站下载并安装。安装好后，双击桌面上的快捷图标，或者在“开始”菜单中选择“Keil  $\mu$ Vision4”，即可启动 Keil  $\mu$ Vision4 集成开发环境，如图 1-1 所示。

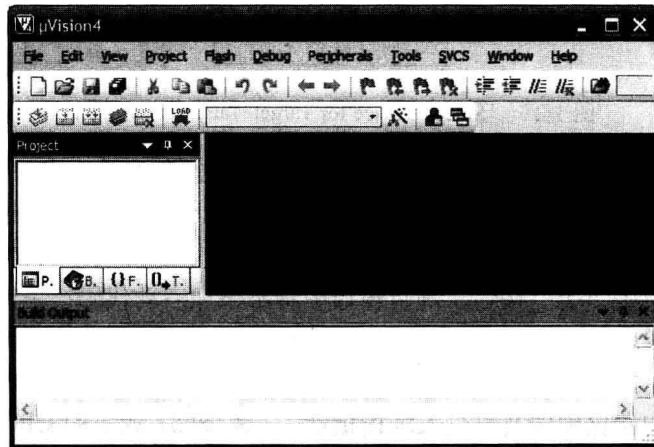


图 1-1 启动 Keil  $\mu$ Vision4 后画面

### 1. 创建项目

Keil  $\mu$ Vision4 中有一个项目管理器，它包含了程序的环境变量和编辑有关的全部信息，为单片机程序的管理带来了很大的方便。

### 【创建新项目操作步骤】

- (1) 启动μVision4，创建一个项目文件，并从元器件数据库中选择一款合适的CPU。
- (2) 创建一个新的源程序文件，并把这个源程序文件添加到项目中。
- (3) 设置工具选项，使之适合目标硬件。
- (4) 编译项目，并生成一个可供PROM编程的HEX文件。

1) 启动μVision4 并创建一个项目文件 μVision4 是一个标准的 Windows 应用程序，直接在桌面上双击图标即可启动它。在μVision4 中执行菜单命令“Project”→“New Project”，弹出“Create New Project”对话框，在此可以输入项目名称。建议每个项目使用一个独立的文件夹。

输入新建项目名后，单击“确定”按钮，弹出如图 1-2 所示的“Select Device for Target ‘Target 1’”对话框。在此对话框中根据需要选择合适的单片机型号。执行菜单命令“Project”→“Select Device for Target”也会弹出图 1-2 所示的对话框。

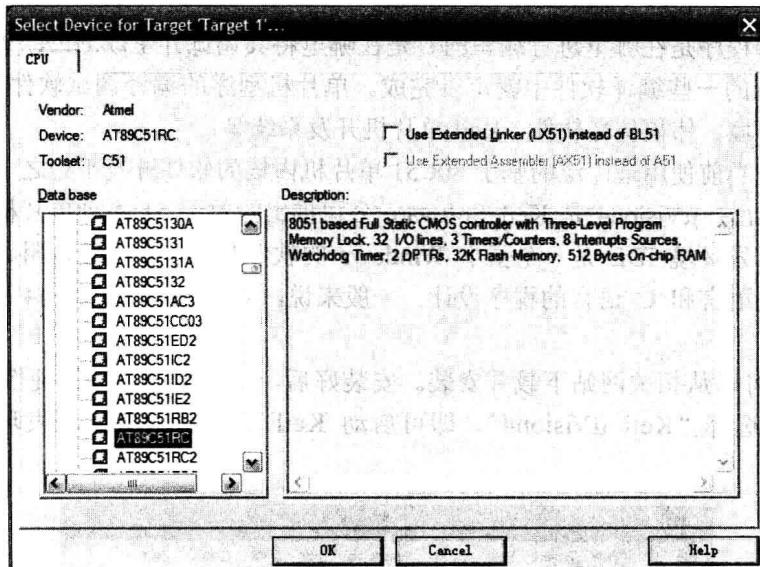


图 1-2 “Select Device for Target ‘Target 1’” 对话框

在图 1-2 中，左边“Data base”栏列出了各厂商名及其产品，右边“Description”栏是对该选中单片机的说明。选择了目标器件后，单击“OK”按钮，将弹出图 1-3 所示的对话框。在此对话框中，询问用户是否将标准的 8051 启动代码复制到项目文件夹并将该文件添加到项目中。在此单击“否”按钮，项目窗口中将不添加启动代码；单击“是”按钮，项目窗口中将添加启动代码。二者的区别如图 1-4 所示。

startup.a51 文件是大部分 8051CPU 及其派生产品的启动程序，启动程序的操作包括清除数据存储器内容、初始化硬件及可重入堆栈指针。一些 8051 派生的 CPU 需要初始化代码以使配置符合硬件上的设计。例如，Philips 的 8051RD+ 片内 xdata RAM 需通过在启动程序中的设置才能使用。应按照目标硬件的要求来创建相应的 startup.a51 文件，或者直接将它从安装路径的\CS1\LIB 文件夹中复制到项目文件中，并根据需要进行更改。

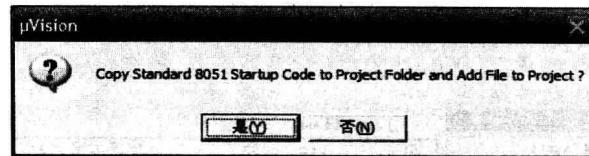


图 1-3 询问是否添加启动代码对话框

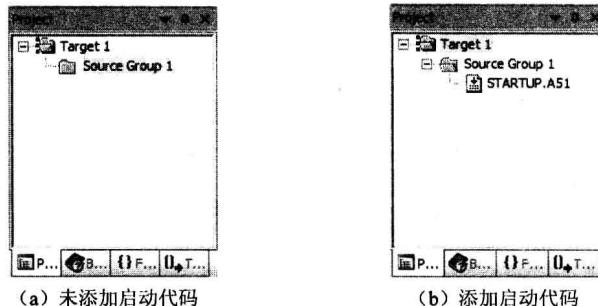


图 1-4 是否添加启动代码的区别

2) 创建新的源程序文件 单击图标<sup>新建</sup>或执行菜单命令“File”→“NEW”，就可以创建一个源程序文件。该命令会打开一个空的编辑器窗口，在编辑窗口中输入源代码，如图 1-5 所示。源代码可以用汇编语言或单片机 C 语言进行书写，源代码输入完成后，执行菜单命令“File”→“Save as...”或“Save”，即可对源程序进行保存。在保存时，文件名只能由字符、字母或数字组成，并且一定要带扩展名（使用汇编语言编写的源程序的扩展名为.A51 或.ASM；使用单片机 C 语言编写的源程序的扩展名为.C）。源程序保存好后，源程序窗口中的关键字呈彩色高亮度显示。

01 org 00h ;程序上电从00h开始
02 ajmp main ;跳转到主程序
03 org 0030h ;主程序起始地址
04 main: mov a,#0feh ;给A赋值成11111110
05 loop: mov p0,a ;将a送到p0口,发光二极管低电平点亮
06 lcall delay ;调用延时子程序
07 rl a ;累加器A循环左移一位
08 ajmp loop ;重新送P1显示
09 delay:mov r3,#20 ;软件延时
10 d1: mov r4,#80
11 d2: mov r5,#248
12 djnz r5,\$ ;d1
13 djnz r4,d2
14 djnz r3,d1
15 ret
16 end

图 1-5 源程序编辑窗口

源程序文件创建好后，可以把这个文件添加到项目中。在μVision4 中，添加的方法有多种。如图 1-6 所示，在“Source Group 1”上单击鼠标右键，在弹出的菜单中选择“Add Files to Group 'Source Group 1'”，在弹出的“Add Files to Group 'Source Group 1'" 对话框中选择刚才创建的源程序文件即可将其添加到项目中。

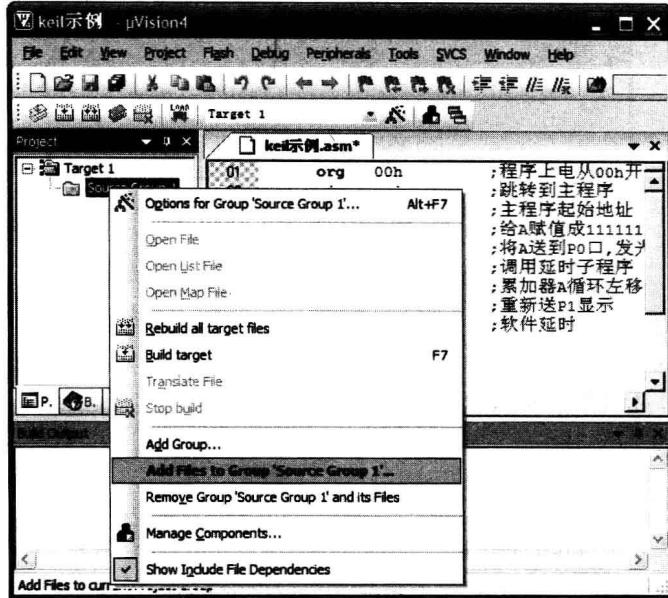


图 1-6 在项目中添加源程序文件

3) 为 目 标 设 定 工 具 选 项 单击图标 或 执 行 菜 单 命 令 “Project” → “Options for Target”，将 会 出 现 “Options for Target 'Target 1'" 对 话 框，如 图 1-7 所 示。在 “Target” 选 项 卡 中 可 以 对 目 标 硬 件 及 所 选 器 件 片 内 部 件 进 行 参数 设 定。表 1-1 描 述 了 “Target” 选 项 卡 的 选 项 说 明。

表 1-1 “Target” 选 项 卡 的 选 项 说 明

选 项	说 明
Xtal	指 定 器 件 的 CPU 时 钟 频 率，多 数 情 况 下，它 的 值 与 XTAL 的 频 率 相 同
Use On-chip ROM	使 用 片 上 自 带 的 ROM 作 为 程 序 存 储 器
Memory Model	指 定 C51 编 译 器 的 存 储 模 式，在 开 始 编 译 新 应 用 时，默 认 为 Small
Code Rom Size	指 定 ROM 存 储 器 的 大 小
Operating system	操 作 系 统 的 选 择
Off-chip Code memory	指 定 目 标 硬 件 上 所 有 外 部 地 址 存 储 器 的 地 址 范 围
Off-chip Xdata memory	指 定 目 标 硬 件 上 所 有 外 部 数据 存 储 器 的 地 址 范 围
Code Banking	指 定 Code Banking 块 数

标 准 的 80C51 的 程 序 存 储 器 空 间 为 64KB，若 程 序 存 储 器 空 间 超 过 64KB 时，可 在 “Target” 选 项 卡 中 对 “Code Banking” 栏 进 行 设 置。Code Banking 为 地 址 复 用，可 以 扩 展 现 有 的 CPU 程 序 存 储 器 寻 址 空 间。复 选 “Code Banking” 栏 后，用 户 根 据 需 求 在 “Banks” 中 选 择 合 适 的 块 数。在 Keil C51 中，用 户 最 多 能 使 用 32 块 64KB 的 程 序 存 储 空 间，即 2MB 的 空 间。

4) 编 译 项 目 并 创 建 HEX 文 件 在 “Target” 选 项 卡 中 设 置 好 参 数 后，就 可 对 源 程 序 进 行 编 译。单 击 图 标 或 执 行 菜 单 命 令 “Project” → “Build Target”，可 以 编 译 源 程 序 并 产 生 应 用。当 所 编 译 的 程 序 有 语 法 错 误 时，μVision4 将 会 在 “Build Output” 窗 口 中 显 示 错 误 和 警 告 信 息，如 图 1-8 所 示。双 击 某 一 条 信 息，光 标 将 会 停 留 在 μVision4 文 本 编 辑 窗 口 中 出 现 该 错 误 或 警 告 的 源 程 序 位 置 上。

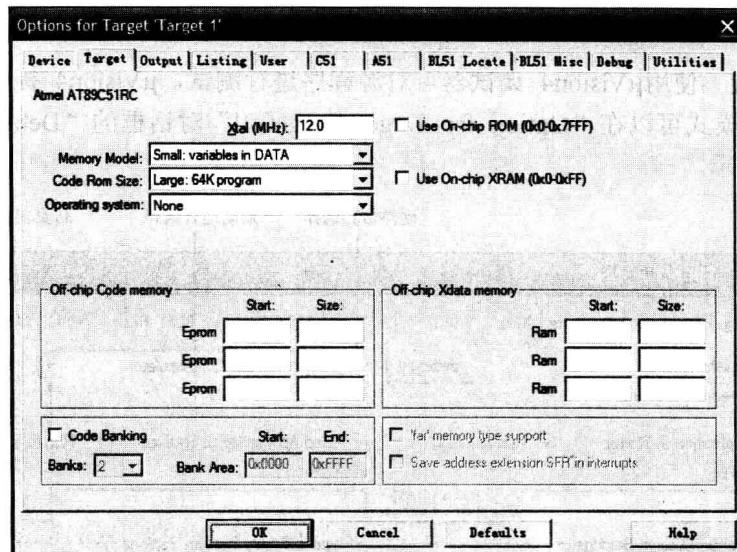


图 1-7 “Options for Target 'Target 1'” 对话框

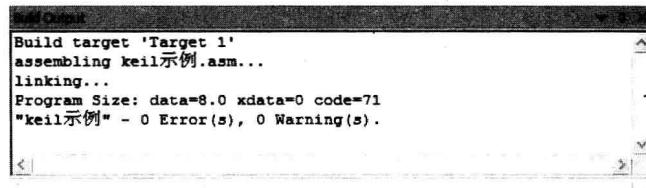


图 1-8 错误和警告信息

若成功创建并编译了应用程序，就可以开始调试。当程序调试好后，要求创建一个 HEX 文件，生成的 HEX 文件可以下载到 EPROM 编程器或模拟器中。

若要创建 HEX 文件，必须将“Options for Target 'Target 1'”对话框中的“Output”选项卡下的“Create HEX File”复选框选中，如图 1-9 所示。

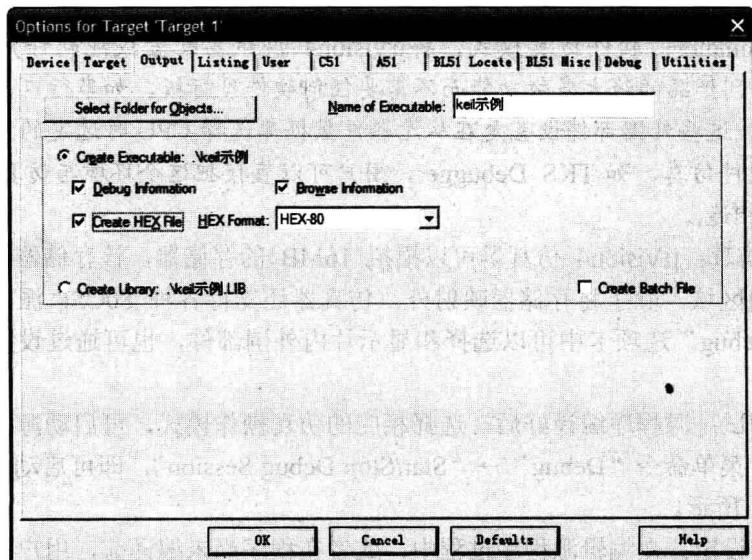


图 1-9 选中“Create HEX File”

## 2. 仿真设置及窗口介绍

1) 仿真设置 使用μVision4 调试器可对源程序进行测试, μVision4 提供了两种操作工作模式, 这两种模式可以在 “Option for Target ‘Target 1’” 对话框的 “Debug” 选项卡中选择, 如图 1-10 所示。

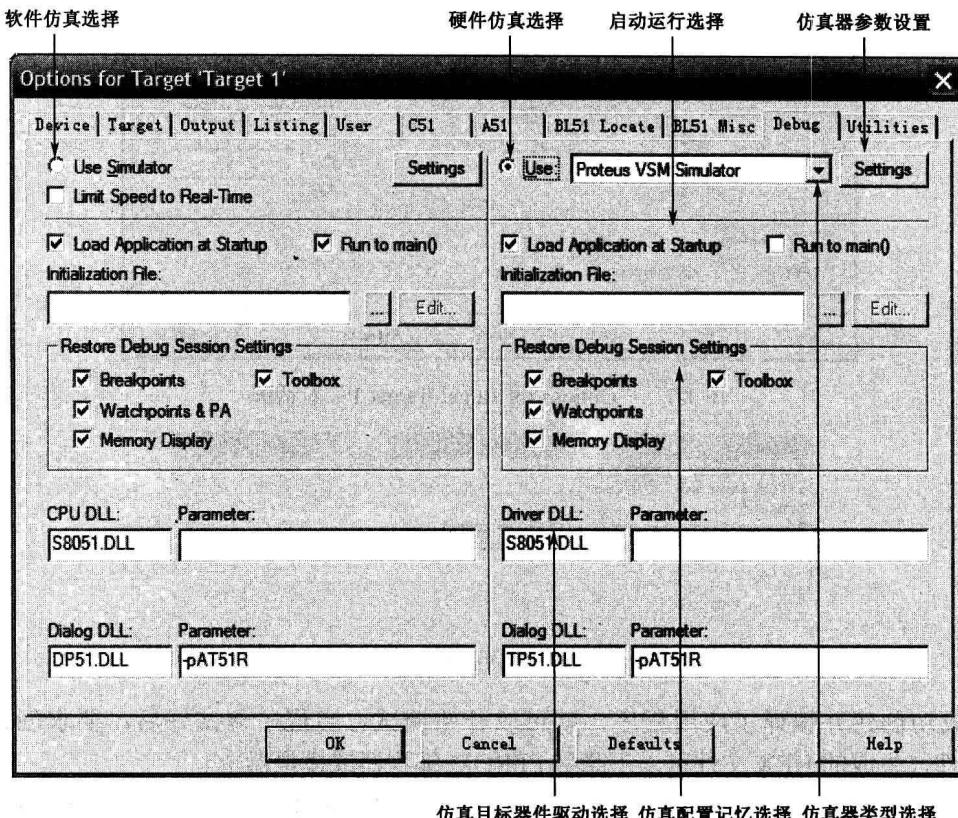


图 1-10 仿真设置

- Use Simulator: 软件仿真模式, 将μVision4 调试器配置成纯软件产品, 能够仿真 8051 系列产品的绝大多数功能而不需要任何硬件目标板, 如串行口、外部 I/O 和定时器等, 这些外围部件设置是在从元器件数据库选择 CPU 时选定的。
- Use: 硬件仿真, 如 TKS Debugger, 用户可以直接把这个环境与仿真程序或 Keil 监控程序相连。

(1) CPU 仿真: μVision4 仿真器可以模拟 16MB 的存储器, 该存储器被映射为读、写或代码执行访问区域。除了将存储器映射外, 仿真器还支持各种 80C51 派生产品的集成外围器件。在 “Debug” 选项卡中可以选择和显示片内外围部件, 也可通过设置其内容来改变各种外设的值。

(2) 启动调试: 源程序编译好后, 选择相应的仿真操作模式, 可启动源程序的调试。单击图标 或执行菜单命令 “Debug” → “Start/Stop Debug Session”, 即可启动μVision4 的调试模式, 如图 1-11 所示。

(3) 断点的设定: 在编辑源程序过程中, 或者在程序尚未编译前, 用户可以设置执行断点。μVision4 中可用不同的方法来定义断点。

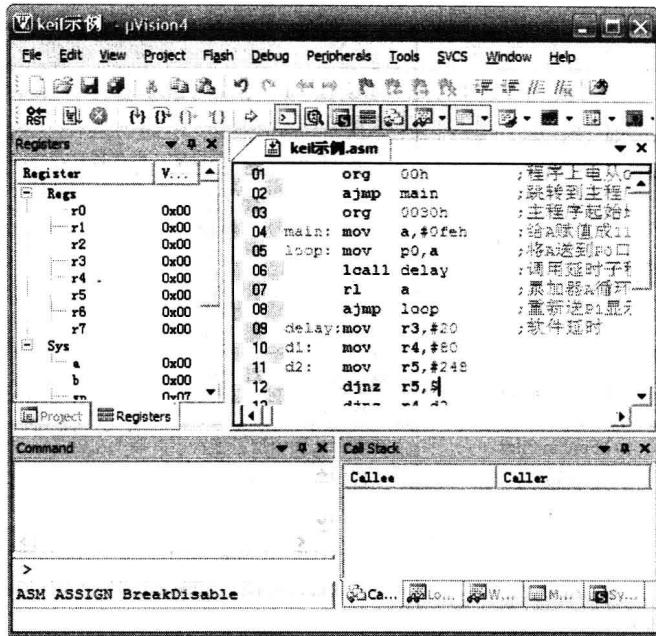


图 1-11 调试界面

- 在文本编辑框中或反汇编窗口中选定所在行，然后单击“File Toolbar”按钮或单击图标。
  - 在文本编辑窗口或反汇编窗口本地菜单上单击鼠标右键，打开快捷菜单进行断点设置。
  - 执行菜单命令“Debug”→“Breakpoint”，打开“Breakpoint”对话框，在这个对话框中可以查看定义或更改断点的设置。
  - 在“Command”窗口中可以使用 BreakSet、BreakKill、BreakList、BreakEnable 和 BreakDisable 等命令。
- (4) 目标程序的执行：目标程序的执行可使用以下方法操作：
- 执行菜单命令“Debug”→“Run”，或者直接单击图标。
  - 在文本编辑窗口或反汇编窗口单击鼠标右键，在弹出的快捷菜单上选择“Run till Cursor line”命令。
  - 在“Command”窗口中可以使用 Go、Ostep、Pstep、Tsetp 命令。

2) 主要窗口介绍 在μVision4 中有许多的工作窗口，这些窗口有些是在编译状态下有效，有些是在调试状态下有效，但多数窗口只在调试状态下有效。表 1-2 列出了μVision4 中的主要工作窗口。

表 1-2 μVision4 中的主要工作窗口

窗 口 名	英 文 名	功 能 说 明
程序编辑/调试窗口	<文件名>	显示源程序，进行各种编辑操作和调试操作
源程序浏览窗口	Source Browser	显示源文件及调试信息（如当前程序的执行位置、断点等），要求在项目制作时要选中产生
反汇编窗口	Disassembly	显示反汇编代码及调试信息，并可进行在线汇编



续表

窗口名	英文名	功能说明
项目窗口	项目页 Project	显示项目构成（项目、文件组、文件），可修改项目、文件组名，增加或删除其中的文件
	书籍页 Books	显示联机的参考资料目录，可选择打开阅读
	函数页 Functions	显示程序中的函数头和主函数中的块语句头，帮助用户快速查找定位
	模板页 Templates	定义、插入文本模板，可加快文本的录入速度。模板中的文本用“ ”界定
	寄存器页 Registers	可显示、修改寄存器的值，获得程序的运行信息（包括已执行的机器周期数的以秒为单位的时间）
制作输出窗口		显示项目处理过程中产生的信息，如警告、出错、数据/代码规模等。双击错误/警告信息可快速定位到它的源
批量文件查找窗口		在多个文件中搜索字符串并显示在该窗口
命令窗口		支持用户输入命令行，并显示系统已执行的命令
存储器窗口	存储器#1 Memory#1	显示、修改 ROM/RAM 存储器中的数据。使用 D:/i/b/x/c: 前缀可分别寻址 data/idata/bdata/xdata/code 中的数据，如 d:0x30,x:0x0100,c:0x0005
	存储器#2 Memory#2	
	存储器#3 Memory#3	
	存储器#4 Memory#4	
监视窗口	局部变量 Local	显示、修改局部变量
	监视#1 Watch#1	用户可设置所要监视的变量和表达式，随程序运行观察其变化
	监视#2 Watch#2	
串行口窗口	串行口#1 UART#1	显示经串行口输出的字符数据和要输入串行口的字符数据，在使用标准 I/O 函数时，需打开该窗口来进行 I/O 操作。软件模拟时，要求对单片机串行口/定时器进行初始化；硬件仿真时，可直接使用标准 I/O 函数
	串行口#2 UART#2	
	串行口#3 UART#3	
	调试（打印） Debug (print) Viewer	
分析窗口	逻辑分析 Logic Analyzer	对选中信号进行逻辑分析，即记录其逻辑状态的变化
	性能分析 Performance Analyzer	显示所要观察的各函数/模块占用 CPU 工作时的比例
	代码覆盖 Code Coverage	显示各函数/模块被 CPU 执行的几率
外围设备窗口	中断 Peripherals: Interrupt	显示、操作各种外围设备，包括 P0~P3 并口、定时/计数器、中断系统和串行口控制等。其中，中断可显示/设置与各中断源有关的数据；I/O 端口可显示/设置 P0~P3 各端口之值；串行口可显示/设置与串行口有关的数据；定时器可显示/设置与各定时/计数器有关的数据
	I/O 端口 Peripherals: I/O-Port	
	串行口 Peripherals: Serial	
	定时器 Peripherals: Timer	

(1) 反汇编窗口：在进行程序调试及分析时，经常会用到反汇编。反汇编窗口同时显示目标程序、编译的汇编程序和二进制文件，如图 1-12 所示。

在程序调试状态下，执行菜单命令“View”→“Disassembly Window”，即可打开反汇编窗口。当反汇编窗口作为当前活动窗口时，若单步执行指令，所有的程序将按照 CPU 指令（即汇编）来单步执行，而不是 C 语言的单步执行。

(2) CPU 寄存器窗口：在程序调试状态下，执行菜单命令“View”→“Registers Window”，将打开 CPU 寄存器窗口，在此窗口中将显示 CPU 寄存器相关内容，如图 1-13 所示。

(3) 存储器窗口：在程序调试状态下，执行菜单命令“View”→“Memory Window”→“Memory #1”，将打开存储器窗口。存储器窗口最多可以通过 4 个不同的页观察 4 个不同的存储区，每页都能显示存储器中的内容，如图 1-14 所示。

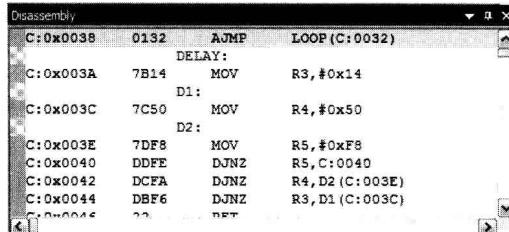


图 1-12 反汇编窗口

Register	Value
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x07
sp_max	0x07
dptr	0x0000
PC	C:0x0000
states	0
sec	0.00000000
psw	0x00

图 1-13 CPU 寄存器窗口

Memory 1
Address: C:00h
C:0x0000: 01 30 00
C:0x0011: 00
C:0x0022: 00 74 FE F5
C:0x0033: 80 12 00 3A 23 01 32 7B 14 7C 50 7D F8 DD FE DC FA
C:0x0044: DB F6 22 00
C:0x0055: 00
C:0x0066: 00
C:0x0077: 00
C:0x0088: 00
C:0x0099: 00
C:0x00AA: 00
C:0x00BB: 00
C:0x00CC: 00
C:0x00DD: 00
C:0x00EE: 00

图 1-14 存储器窗口

在“Address”栏中输入地址值后，显示区域直接该地址的内容。若要更改地址中的内容，只需在该地址上双击鼠标左键，并输入新的内容即可。

(4) 串行窗口：μVision4 提供了 4 个专门用于串行调试输入和输出的窗口，被模拟仿真的 CPU 串行口数据输出将在该窗口进行显示，输入串行窗口中的字符将被输入到模拟的 CPU 中。

在程序调试状态下，执行菜单命令“View”→“Serial Window”→“UART #1”，即可打开串行调试窗口。

### 3. Keil 程序调试与分析

前面讲述了如何在 Keil 中建立、编译、连接项目，并获得目标代码，但是做到这一步仅代表源程序没有语法错误，至于源程序中存在的其他错误，必须通过调试才能发现并解决。事实上，除了极简单的程序外，绝大多数的程序都要通过反复调试才能得到正确的结果，因此，调试是软件开发中的一个重要环节。

1) 寄存器和存储器窗口分析 进入调试状态后，执行菜单命令“Debug”→“Run”，或者单击图标，全部运行源程序。执行菜单命令“Debug”→“Step”，或者单击图标，单步运行源程序。源程序运行过程中，项目工作区（Project Workspace）“Registers”选项卡中显示相关寄存器当前的内容。若在调试状态下未显示此窗口，可执行菜单命令“View”→“Project Window”将其打开。

在源程序运行过程中，可以通过存储器窗口（Memory Window）来查看存储区中的数据。在存储器窗口的上部，有供用户输入存储器类型的起始地址的文本输入栏，用于设置关注对象所在的存储区域和起始地址，如“D:0x30”。其中，前缀表示存储区域，冒号后为要

观察的存储单元的起始地址。常用的存储区前缀有“d”或“D”(表示内部 RAM 的直接寻址区)、“i”或“I”(表示内部 RAM 的间接寻址区)、“x”或“X”(表示外部 RAM 区)、“c”或“C”(表示 ROM 区)。由于 P0 端口属于 SFR (特殊功能寄存器)，片内 RAM 字节地址为 80H，所以在存储器窗口的上部输入“d:80h”时，可查看 P0 端口的当前运行状态为 FE，如图 1-15 所示。

**2) 延时子程序的调试与分析** 在源程序编辑状态下，执行菜单命令“Project”→“Options for Target 'Target 1'”，或者在工具栏中单击图标 $\text{A}$ ，再在弹出的对话框中选择“Target”选项卡。在“Target”选项卡的“Xtal (MHz) :”栏中输入 12，即设置单片机的晶振频率为 12MHz。然后在工具栏中单击图标 $\text{B}$ ，对源程序再次进行编译。

执行菜单命令“Debug”→“Start/Stop Debug Session”，或者在工具栏中单击图标 $\text{C}$ ，进入调试状态。在调试状态下，单击图标 $\text{D}$ ，使光标首次指向 LCALL DELAY 后，项目工作区“Registers”选项卡的 Sys 项中 sec 为 0.00000400，如图 1-16 所示，表示进入首次运行到 LCALL DELAY 时花费了 0.00000400s。再次单击图标 $\text{D}$ ，光标指向“RL A”，Sys 项的 sec 为 0.79846900，如图 1-17 所示。因此，DELAY 的延时时间为二者之差，即 0.79846500s，也就是说延时约为 0.8s。

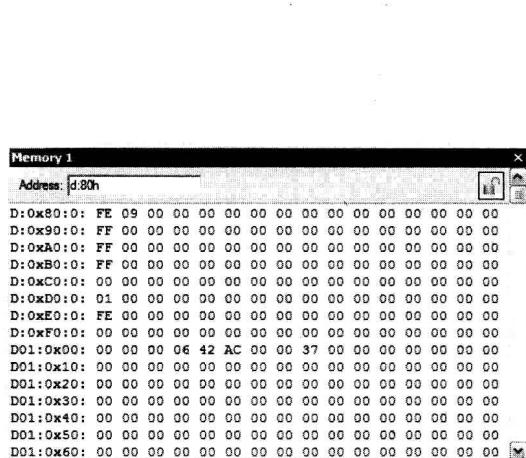


图 1-15 存储器窗口

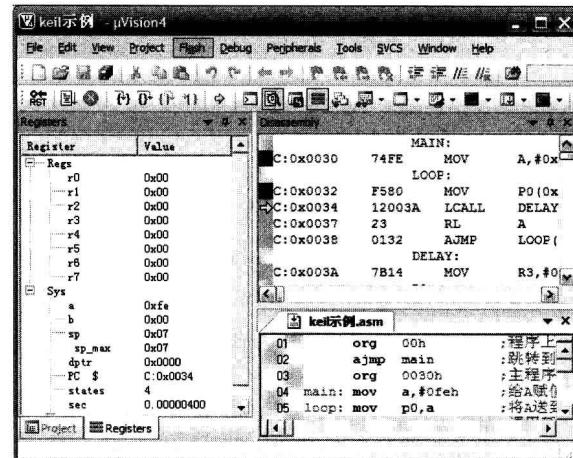


图 1-16 光标首次指向 LCALL DELAY

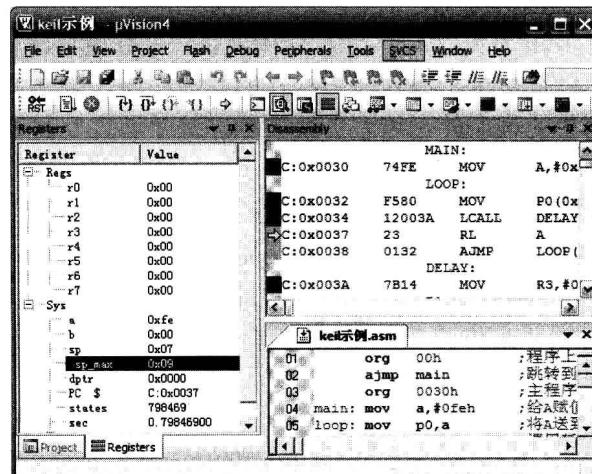


图 1-17 光标首次指向“RL A”