

基于 Keil C51 高级语言的

TKS 系列仿真器使用指南

广州致远电子有限公司

TKS 为您提供这样的开发工具：

更准确

更便宜

更方便

TKS 系列仿真器开发组

目录

第一章	TKS 系列仿真器简介
1.1	公司简介
1.2	TKS 系列仿真器的技术优势
第二章	TKS 系列仿真器选型指南
1.1	TKS 系列仿真器的型号分类
2.2	TKS 系列仿真器的选购
2.3	TKS 系列仿真器的仿真芯片
2.4	仿真芯片的适用范围
2.5	仿真其它公司芯片的解决方案
第三章	TKS 系列仿真器的技术支持
第四章	使用 TKS 系列仿真器前需要掌握的知识
第五章	Keil IDE 环境的快速入门
5.1	Keil 开发软件的介绍
5.2	Keil 软件开发的流程
5.3	软件的安装
5.4	Keil 软件的工作环境
5.5	Hello 测试程序
第六章	TKS 在 Keil 环境下仿真的详细范例
6.1	建立工程
6.2	为工程选择目标器件
6.3	建立并编辑程序源文件
6.4	为工程添加文件
6.5	对工程进行设置
6.6	对仿真器进行设置
6.7	对工程进行编译/连接
6.8	进入硬件仿真前的准备工作
6.9	进入硬件仿真
6.10	运行程序
6.11	程序复位
6.12	断点操作
6.13	空间资源的查看和修改
6.14	变量的查看和修改
6.15	在 Keil 调试中的使用技巧
6.16	退出仿真

警告：《TKS 系列仿真器用户使用手册》的内容会在以后发生改变，用户需要以随机提供的电子文档为标准。

第一章 TKS 系列仿真器简介

1.1 公司简介

广州致远电子有限公司是一家专业从事开发工具设计、制造的高科技公司，主要为客户提供完整的嵌入式系统解决方案，包括器件选择方案、原型设计方案、开发工具方案、功能验证方案等，同时发展自有品牌的产品。

在公司成长和发展的过程中，我们深深地感到开发工具对公司和用户的重要性。我们有很多国外优秀的产品和技术希望介绍给国内的用户，并希望形成有前途的商业应用。但是国内无法解决开发工具的问题，可能是因为技术的问题，或是因为没有利润。长期以来，面对太多用户的咨询开发工具的解决方案，我们给用户的唯一答案就是国外从几万元到十几万元的方案。我们知道国外的开发工具性能是优异的，但是价格也是昂贵的。面对大多数客户无法承受的工具价格，很多用户放弃了自己原本优秀的方案，改用其它的折中手段。多年的技术服务经验使我们明白，很多折中的方案就是平庸的设计方案。我们决心用我们的经验进入开发工具行业，设计生产出**更准确，更便宜，更方便**的开发工具。

1.2 TKS 系列仿真器的技术优势

TKS 系列仿真器是广州致远电子有限公司推出的高性能实时在线仿真器，可以实时在线仿真 PHILIPS 半导体公司的 80C51 系列单片机的同时，还能够实时在线仿真 ATMEL、HY、Winbond 等公司的兼容 MCS-51 内核的标准 80C51 单片机，彻底解决 80C51 开发中仿真工具的瓶颈问题。

TKS 系列仿真器在硬件上采用了 PHILIPS 授权的 HOOKS/Bondout 仿真技术并加以改进，几乎支持所有的 80C51 系列单片机的实时仿真，能保证用户更加方便的操作和更加真实的仿真效果，并能直接或通过简单的升级支持仿真 PHILIPS 陆续推出的全系列 80C51 内核单片机。每种型号的仿真器均能支持多种不同型号 80C51 单片机的仿真。我们将能得到世界第一大 80C51 内核单片机供应商 PHILIPS 半导体公司详尽的技术支持，并为用户提供准确可靠真实的仿真性能。

购买正版软件请与德国 Keil 公司签约商业合作伙伴广州周立功单片机发展有限公司联系，与此同时 Keil 公司将全面支持我们的仿真器开发。TKS 系列仿真器除使用自主知识产权的调试环境和编译软件外，还可以同 Keil 调试软件无缝嵌接并能得到 Keil 公司的强力技术援助，具备强大调试功能 Keil 的 IDE 环境将全面提升 TKS 系列仿真器的操作性能。

第二章 TKS 系列仿真器选型和选购

采用 PHILIPS 授权 HOOKS/Bondout 仿真技术的 TKS 系列仿真器目前的型号见表 1，广州致远电子有限公司将陆续推出采用其它型号的仿真器，全面支持 PHILIPS 和其它公司的 80C51 芯片仿真。

2.1 TKS 系列仿真器的型号分类

表 1 TKS-HOOKS 系列仿真器型号

型号	TKS-58	TKS-61	TKS-RD2	TKS-668
仿真技术	HOOKS	HOOKS	HOOKS	HOOKS
仿真程序空间	内部 64K	内部 64K	内部 64K	内部 64K
仿真数据空间	外部 64K	内部或外部 64K	内部或外部 64K	内部或外部 64K
任意位置 64K 断点	√	√	√	√
外部电源输入支持	2.7-5.5V	2.7-5.5V	2.7-5.5V	2.7-5.5V
外部复位输入支持	√	√	√	√
仿真芯片可更换性	√	√	√	√
实时仿真频率	0-24MHz	0-24MHz	0-24MHz	0-24MHz
支持仿真范围	仿真标准 51	仿真 61 以下	仿真 RD2 以下	仿真型号无限制
后续芯片支持特性	√	√	√	√
出内置厂仿真芯片	P87C52X2	P89C61X2	P89C51RD2	P89C668

续表 1

型号	TKS-554	TKS-591	TKS-764	TKS-932
仿真技术	HOOKS	HOOKS	BondOut	BondOut
仿真程序空间	内部 64K	内部 64K	4K	8K
仿真数据空间	内部或外部 64K	内部或外部 64K		
任意位置 64K 断点	√	√	√ (4K)	√ (4K)
外部电源输入支持	√	√	√	√
外部复位输入支持	√	√	√	√
仿真芯片可更换性	√	*	*	*
实时仿真频率	0-24MHz	0-24MHz	0-20MHz	0-12MHz
支持仿真范围	87C554 资源以内	8XC591 资源以内	51LPC 系列	* LPC900 系列
后续芯片支持特性	√	*	*	*
出内置厂仿真芯片	P87C554	P87C591	P87LPC768 或 769	P89LPC932

注：每一种型号的 TKS 仿真器均可向下兼容仿真几乎所有的单片机，用户仅需自行更换厂家随机提供的多种仿真 CPU，即可仿真对应型号的单片机：

2.2 TKS 系列仿真器的选购

- **TKS—58:** 经济型的标准 80C51 单片机仿真器, 仿真标准的 P87C5x、P89C5x、P87C5xX2、P89C5xX2, 同时支持其它公司兼容 80C51 单片机的仿真。如果用户只仿真标准型号的 80C51 单片机, 包括 PHILIPS 的 2 倍频单片机, 建议选购 TKS-58 型号仿真器。它支持 2.7V-5.5V 的宽电压范围仿真, 内部 64K 的 ROM 仿真范围, 支持外部复位信号和用户板的时钟和用户板的晶振。TKS—58 仿真器也可以通过有偿的技术支持升级到 TKS—61/RD2/668 仿真器的功能。

- **TKS—61:** 增强型的 80C51 仿真器, 除兼容仿真 TKS-58 仿真器所支持的所有型号外, 还支持 P89C60X2、P89C61X2 单片机的仿真。TKS—61 仿真器也可以通过有偿的技术支持升级到 TKS—RD2/668 仿真器的功能。

- **TKS—RD+:** 增强型的 80C51 仿真器, 除兼容仿真 TKS—61 仿真器的所有型号外, 还支持 P89C51Rx+ 单片机的仿真。如果用户需要仿真增强性能的 80C51 系列单片机, 如内部附加 RAM, PCA 阵列, 可以选购 TKS—RD+ 仿真器。TKS—RD+ 也可以通过技术支持升级仿真 TKS—RD2/668 支持的型号, 但是需要升级费用。

- **TKS—RD2:** 增强型的 80C51 仿真器, 除兼容仿真 TKS—61/TKS—RD+ 仿真器的所有型号外, 还支持 P89C51Rx+、P89C51Rx2 单片机的仿真。如果用户需要仿真增强性能的 80C51 系列单片机, 如内部附加 RAM, PCA 阵列, 可以选购 TKS—RD2 仿真器。TKS-RD2 也可以通过技术支持升级仿真 TKS—668 支持的型号, 但是需要升级费用。

- **TKS—668:** 高级型的 80C51 仿真器, 除兼容仿真 TKS—RD2 的所有型号外, 还支持 P89C66x 单片机的仿真。如果用户大范围的使用增强型号的 PHILIPS 的 80C51 系列单片机, 建议选用 TKS—668 仿真器。它支持仿真 PHILIPS 目前几乎全部的 80C51 单片机 (40 脚) 的仿真, 而且能支持 PHILIPS 陆续推出的新型号单片机。

- **TKS—591:** 专用型的仿真器, 支持 P87C591 的仿真, 支持 P87C591 资源内的 80C51 单片机的仿真。如果用户需要仿真内部带 CAN 总线控制器的 P8xC591, 可以选用 TKS—591 仿真器。

- **TKS—554:** 专用型的仿真器, 支持 P8xC552/554 的仿真, 支持 P8xC552/554 资源内的 80C51 单片机的仿真。如果用户需要仿真内部带 8 路 10 位 A/D 转换器的 P8xC552/554 单片机, 可以选用 TKS—554 仿真器。

- **TKS—764:** 专用型仿真器, 支持仿真 PHILIPS 的 51LPC 系列, 目前可以支持的芯片为 P87LPC759/760/761/762/764/767/768/769。如果用户仿真 51LPC 系列单片机, 可以选用 TKS—764 仿真器。

- **TKS—932:** 专用型仿真器, 支持仿真 PHILIPS 的 LPC900 系列。如果用户需要仿真 LPC900 系列, 可以选用 TKS—932。

2.3 TKS 系列仿真器的仿真芯片

TKS 系列仿真器具有众多的仿真芯片供用户选择。用户在购买仿真器时，如果仿真器内部预置的仿真芯片不能完全满足您当前或以后的需求，可以单独购买指定的仿真芯片。TKS 除以下列出的仿真芯片外，仿真芯片的种类还在不断扩充，用户请留意网站上的更新。

TKS 之所以选择更换仿真芯片而不是更换仿真头，主要是考虑采用这种方式的可靠性非常高，另外更换仿真芯片的费用是相当低廉的。仿真芯片的更换要由 TKS 仿真器的专业人员或用户在其指导下进行。

表 2 TKS 系列仿真器的仿真芯片

TKS 仿真芯片	仿真芯片性能	仿真电压	适应机型
P87C51 P87C52	标准 80C52,128/256 字节 RAM,内部 64K 程序空间,双 DPTR,ALE 可关闭,3 个定时器,掉电可唤醒.	2.7-5.5V	TKS-58 TKS-61 TKS-RD2 TKS-668
P87C51X2 P87C52X2	双倍速 80C51,128/256 字节 RAM,内部 64K 程序空间,双 DPTR,ALE 可关闭,3 个定时器,掉电可唤醒,时钟模式可以切换,.	2.7-5.5V	TKS-58 TKS-61 TKS-RD2 TKS-668
P89C51 P89C52	标准 80C52,128/256 字节 RAM,内部 64K 程序空间,双 DPTR,ALE 可关闭,3 个定时器,掉电可唤醒.	4.5-5.5V	TKS-58 TKS-61 TKS-RD2 TKS-668
P89C51X2 P89C52X2	双倍速 80C51,128/256 字节 RAM,内部 64K 程序空间,双 DPTR,ALE 可关闭,3 个定时器,掉电可唤醒,时钟模式可以切换.	2.7-5.5V	TKS-58 TKS-61 TKS-RD2 TKS-668
P89C60X2 P89C61X2	双倍速 80C51,256 字节 RAM,内部 64K 程序空间,双 DPTR,ALE 可关闭,3 个定时器,掉电可唤醒,时钟模式可以切换,256/768 字节附加 XRAM.	4.5-5.5V	TKS-58 TKS-61 TKS-RD2 TKS-668
P87C51RA+ P87C51RB+ P87C51RC+ P87C51RD+	增强型 80C51,256 字节 RAM,内部 64K 程序,双 DPTR,ALE 可关闭,3 个定时器,掉电可唤醒,256/768 字节附加 XRAM,PCA 阵列,看门狗.	2.7-5.5V	TKS-58(需升级) TKS-61(需升级) TKS-RD2 TKS-668
P89C51RA+ P89C51RB+ P89C51RC+ P89C51RD+	增强型 80C51,256 字节 RAM,内部 64K 程序,双 DPTR,ALE 可关闭,3 个定时器,掉电可唤醒,256/768 字节附加 XRAM,PCA 阵列,看门狗.	4.5-5.5V	TKS-58(需升级) TKS-61(需升级) TKS-RD2 TKS-668
P89C51RA2 P89C51RB2 P89C51RC2 P89C51RD2	增强型双倍速 80C51,256 字节 RAM,内部 64K 程序,双 DPTR,ALE 可关闭,3 个定时器,掉电可唤醒,256/768 字节附加 XRAM,PCA 阵列,看门狗.	4.5-5.5V	TKS-58(需升级) TKS-61(需升级) TKS-RD2 TKS-668
P89C660 P89C662	增强型双倍速 80C51,256 字节 RAM,内部 64K 程序,双 DPTR,ALE 可关闭, 256/768/1792/8K 字节附加 XRAM,PCA	4.5-5.5V	TKS-58(需升级) TKS-61(需升级)

P89C664 P89C668	阵列,看门狗,I ² C.		TKS-RD2(需升级) TKS-668
P87C552	增强型 80C51, 256 字节 RAM,内部 64K 程序,双 DPTR,ALE 可关闭, 68 管脚封装,8 路 10 位 A/D,2 路 8 位精度 PWM,5 路 8 位 I/O 口,看门狗,I2C, 软件复位,上电检测等.	2.7-5.5V	TKS-554
P87C554	增强型双倍速 80C51, 256 字节 RAM,内部 64K 程序,双 DPTR,ALE 可关闭, 256 字节附加 XRAM,68 管脚封装,8 路 10 位 A/D,2 路 8 位精度 PWM,5 路 8 位 I/O 口,看门狗,I2C, 软件复位,上电检测等.	2.7-5.5V	TKS-554
P87C591	内部带 CAN 总线控制器, 256 字节 RAM,内部 64K 程序,双 DPTR,ALE 可关闭, 256 字节附加 XRAM, 6 路 10 位 A/D,2 路 8 位 PWM,I2C, 软件复位等	4.5-5.5V	TKS-591
P87LPC768 P87LPC769	51LPC 系列专用仿真器	3.3-5.5V	TKS-764
P89LPC932	LPC 系列专用仿真器	*3.3-3.6V	TKS-932

2.4 仿真芯片的适用范围

TKS 的仿真芯片不但能精确仿真 PHILIPS 的全系列 80C51 单片机, 而且也能直接或完全精确仿真其它公司的 80C51 单片机。这是由于 PHILIPS 是全球最大的 80C51 单片机供应商, 80C51 的产品线能覆盖绝大多数的应用领域。在仿真其它公司的 80C51 单片机时, 除了特殊型号或用途的单片机, 一般在 PHILIPS 的 80C51 单片机中都能找到完全相同或基本类似的仿真型号。

我们建议用户在系统设计期间尽量选择 PHILIPS 的 80C51 单片机系列, 而不要选择一些 80C51 产品线很短的系列。这是因为 PHILIPS 的各种型号的 80C51 单片机都有仿真器支持, 这在设计大系统中是非常关键的。在大系统设计验证后期, 系统任何一个问题的排除都变得非常复杂。如果没有类似仿真器的工具支持, 用户将耗费大量的精力和物力去排除一个细小的问题。

如果用户确实想使用 TKS 仿真器来仿真其它公司的 80C51 单片机型号, 请按照下面的步骤:

1. 如果确信选中的器件型号是 PHILIPS 公司的型号或兼容型号, 则可以直接从仿真芯片表中选取。
2. 从下面的**仿真其它公司芯片的解决方案**中选取我们建议的仿真芯片。如果在我们的解决方案中没有找到对应的型号, 请仔细阅读你所选中器件的数据手册, 然后选中基本相近的仿真芯片。如果你所选中的器件属于该公司的特殊型号, 请致电我们的技术支持, 我们可以根据您的实际要求为您提供相应的 PHILIPS 单片机型号以及开发工具的解决方案。
3. 选中仿真芯片后, 你可以从仿真芯片列表中找到对应的仿真器型号。
4. 致电我们的销售部门或代理, 申请购买。

2.5 仿真其它公司芯片的解决方案

如果你选用其它公司的通用芯片,一般都可以在我们的仿真芯片库中找到可以完全精确仿真的仿真芯片。下面,我们列举了世界知名的几大 80C51 生产商的常用型号,并给出在 TKS 系列仿真器中的解决方案。

PHILIPS

PHILIPS 的全线 80C51 产品都可以直接仿真。

Winbond

芯片型号	芯片特点	TKS 仿真芯片	说明
W78E51 W78E52 W78E54 W78E58 W78E516	标准 80C51	P87C51/P89C51 P87C51/P89C52 P87C51/P89C52 P87C51/P89C52 P87C51/P89C52	
W78E51B W78E52B W78E54B W78E58B W78E516B	看门狗 看门狗 看门狗 看门狗,附加 RAM 看门狗,附加 RAM	P89C51RA+到 RD+ P89C51RA+到 RD+ P89C51RA+到 RD+ P89C51RA+到 RD+ P89C51RA+到 RD+	看门狗用法有区别
W78LE51 W78LE52 W78LE54 W78LE58 W78LE516	看门狗,低电压 看门狗,低电压 看门狗,低电压 看门狗,低电压,附加 XRAM 看门狗,低电压,附加 XRAM	P87C51RA+到 RD+ P87C51RA+到 RD+ P87C51RA+到 RD+ P87C51RA+到 RD+ P87C51RA+到 RD+	

Atmel

芯片型号	芯片特点	TKS 仿真芯片	说明
AT87F51 AT87F52 AT87F55 AT87F51RC	标准 80C51 标准 80C52 双 DPTR,看门狗 双 DPTR,看门狗,附加 XRAM	P87C51/P89C51 P87C52/P89C52 P87C51RA+到 RD+ P87C51RA+到 RD+	
AT87LV51 AT87LV52 AT87LV55	标准 80C51 标准 80C52 双 DPTR,看门狗,附加 XRAM	P87C51RA+到 RD+ P87C51RA+到 RD+ P87C51RA+到 RD+	看门狗用法有区别
TS87C52X2 TS87C54X2 TS87C58X2	双 DPTR,6CLK 时钟 双 DPTR,6CLK 时钟 双 DPTR,6CLK 时钟	P87C52X2 P87C52X2 P87C52X2	
TS87C51RB2 TS87C51RC2 TS87C51RD2	增强型 80C51,256 字节 RAM,内部 64K 程序,双 DPTR,ALE 可关闭,3 个定时器,掉电可唤	P87C51RB2 P87C51RC2 P87C51RD2	

	醒 ,256/768 字 节 附 加 XRAM,PCA 阵列,看门狗		
AT89C51	标准 80C51	P87C51/P89C51	
AT89C52	标准 80C52	P87C52/P89C52	
AT89C55	双 DPTR,看门狗	P87C51RA+到 RD+	
AT89C51RC	双 DPTR,看门狗,附加 XRAM	P87C51RA+到 RD+	
AT89LV51	标准 80C51	P87C51RA+到 RD+	看 门 狗 用 法有区别
AT89LV52	标准 80C52	P87C51RA+到 RD+	
AT89LV55	双 DPTR,看门狗,附加 XRAM	P87C51RA+到 RD+	
AT89S51	128 字节 RAM,双 DPTR,看门狗	P89C51RB+到 RD+	
AT89S52	256 字节 RAM,双 DPTR,看门狗		
AT89S53	256 字节 RAM,双 DPTR,看门狗		
AT89LS51	128 字节 RAM,双 DPTR,看门狗	P89C51RB+到 RD+	
AT89LS52	256 字节 RAM,双 DPTR,看门狗		
AT89LS53	256 字节 RAM,双 DPTR,看门狗		
TS89C51RB2	增强型 80C51,256 字节 RAM,内 部 64K 程序,双 DPTR,ALE 可关 闭 ,3 个 定 时 器 ,掉 电 可 唤 醒 ,256/768 字 节 附 加 XRAM,PCA 阵列,看门狗.	P89C51RB2	
TS89C51RC2		P89C51RC2	
TS89C51RD2		P89C51RD2	
AT89S8252	双 DPTR,看门狗	P89C51RB+到 RD+ P87C51RB+到 RD+	EEROM 不 能仿真,并 有 其 它 操 作差异.
AT89LS8252	双 DPTR,看门狗		
AT89C1051	小管脚标准 80C51	P87C51	需要仿真 转换插座
AT89C2051			
AT89C4051			

SST

芯片型号	芯片特点	TKS 仿真芯片	说明
SST89C54	看门狗	P87C51RB+到 RD+	看 门 狗 用 法有区别
SST89C58			
SST89E554RD	双 DPTR,看 门 狗 ,1K 附 加 XRAM,PCA,ALE 可关闭, 6/12 CLK 可选,	P87C51RD+(仿低电压) P89C51RD2(仿 6CLK)	有操作差 异.
SST89V554RD			
SST89E564RD			
SST89V564RD			

其它公司

除了以上列出的 80C51 单片机生产厂商外,还有其它一些生产厂商。这些厂商一般生产标准 80C51 产品,这些类型用户可以按标准的 80C51 选择仿真芯片进行仿真。

第三章 TKS 系列仿真器的技术支持

我们愿意以可能的任意方式为您提供技术支持，在仿真器/开发工具或辐射的其它方面。为了尽快地解决用户的问题，我们建议用户主动采用下面的方式来获取我们的技术支持。

1. 电话(传真)

使用电话可以最快的速度解决您的实际问题。请您在拨通电话前，先整理好您的问题，将仿真器放置在计算机和电话机的旁边并打开，我们的技术支持人员就能很快地根据您的操作来分析您提出的问题。

我们的联系电话：(020) 85520995 85539796 85547386 85541621 85541773 85571663

2. 电子邮件

如果使用电子邮件您可以详细地说明您遇到的具体问题，最好能提供相应的测试程序、电路原理图和 PCB 板图。

我们的 email: mcutools@zlgmcu.com

3. BBS

TKS 系列仿真器建有专门的 BBS 以解决用户遇到的所有问题，在我们的主页上有仿真器 BBS 的连接。对于用户提出的全部问题，可以得到我们详尽的技术解答，同时也能得到其他有经验朋友的回答。

如果您遇到关于 80C51 系列单片机的任何问题，也可以立即寻求我们的技术支持，也可以访问我们的主页 <http://www.zlgmcu.com>。从产品的设计思想、原器件的选型、开发工具的选择、调试的方法以及可靠性的设计，我们的技术支持工程师都能为您提供全面的解决方案或良好的建议。

4. 维修服务

如果您在使用过程中遇到仿真器损坏的现象，请直接将仿真器寄给广州致远电子有限公司维修部，由生产厂商直接、快捷地为您服务，详细的联系方式如下：

广州致远电子有限公司

地址：广州市车陂路黄洲工业区 11 栋 6 楼

电话：(020) 85529859

传真：(020) 85529940

email: mcutools@zlgmcu.com

第四章 TKS 系列仿真器使用前需要掌握的知识

4.1 需要的基本背景知识

使用 TKS 系列仿真器前需要掌握一定的背景知识，一般来讲，如果您有下列的背景知识就可以尝试使用 TKS 系列仿真器。

- 熟悉 Windows 的系统操作以及一般应用软件的操作技能；
- 了解标准 80C51 系列单片机的基本原理；
- 了解一般的电子电路知识。

如果您有以上的背景知识，您可能就会较快地熟悉 TKS 系列仿真器的使用并应用到您的设计中；如果对于上述的背景知识，您还有一定的欠缺或不足，您可能会在熟悉 TKS 系列仿真器的过程中遇到一些问题，建议您在掌握了以后再重新开始学习使用 TKS 系列仿真器。但不能否认的是，在实践中学习也是一种很好的办法。因此如果您虽然对以上的背景知识没有信心，但是有很强的实践能力，您也可以开始尝试使用 TKS 系列仿真器。

4.2 熟悉 TKS 的硬件构成和作用

在开始使用 TKS 系列仿真器前，您首先应该熟悉一下 TKS 系列仿真器的硬件。在您购买回来的仿真器套件中，有几样东西是必须使用的。

- **仿真器主机：**完成仿真功能的最主要部分；
- **仿真器电源：**为仿真器主机提供工作电源。它的输入电源为交流 220V；
- **仿真头：**仿真头带有针状（或其它形状）插座，可以插入用户目标板的插座中。它的另外一端通过仿真电缆连接到仿真主机，这样把仿真主机和用户目标板连接起来；
- **通讯电缆：**用于连接仿真器主机和您的计算机。根据仿真器型号的不同，通讯电缆可能是 RS-232 串行电缆，并口电缆或者是 USB 电缆。

4.3 TKS 系列仿真器对计算机的要求

计算机也是使用 TKS 系列仿真器所必须的工具，您的计算机最低的配置要求是：

- Pentium、Pentium-II 或兼容处理器的 PC；
- Windows 95 以上的操作系统。
- 至少 16MB 的 RAM；
- 20M 以上的多余硬盘；
- 一个 3.5”的软盘驱动器；
- 至少带有一个串行接口（或一个并口，或一个 USB 口）。

4.4 了解 Keil 的 μ Vision2 环境

使用 TKS 系列仿真器前还需熟悉的是 Keil IDE 集成调试环境（ μ Vision2），全部 TKS 系列仿真器系列都可以同 Keil 的 IDE 环境无缝连接，调试功能强大的 Keil IDE 集成调试环境将全面提升 TKS 系列仿真器的操作性能。在第五/六章里我们将介绍 Keil 的使用方法和技巧，以及如何与 TKS 系列仿真器进行连接。

4.5 经常使用的专业词汇

仿真:	对用户编写的程序进行运行检查, 验证运行逻辑, 运行流程等;
软件仿真:	用软件仿真用户编写的程序, 缺点是无法挂接硬件;
硬件仿真:	使用硬件仿真器对用户编写的程序进行验证, 也是必须的验证手段;
Keil :	德国 Keil Software/Keil Elektronik 公司, 专门生产微处理器编译软件;
μVision2:	Keil 的 80C51 系列编译/仿真环境, 也是 TKS 的工作平台, 我们使用的是 V6.23 版本。

第五章 Keil IDE 环境（ μ Vision2）的快速入门

5.1 Keil 开发软件的介绍

Keil IDE (μ Vision2) 集成开发环境是 Keil Software Inc/Keil Elektronik GmbH 开发的基于 80C51 内核的微处理器软件开发平台，内嵌多种符合当前工业标准的开发工具。可以完成从工程建立和管理，编译，连接，目标代码的生成，软件仿真，硬件仿真等完整的开发流程。尤其 C 编译工具在产生代码的准确性和效率方面达到了较高的水平，而且可以附加灵活的控制选项，在开发大型项目时非常理想。由于 Keil 本身是一个纯软件的东西，还不能直接进行硬件仿真，必须挂接类似 TKS 系列仿真器的硬件才可以进行仿真。通过下面介绍可以看到 TKS 系列硬件仿真器如何无缝连接到 Keil 的调试环境中。

- **μ Vision2 IDE**

μ Vision2 IDE 包括一个工程管理器、一个功能丰富并有交互式错误提示的编辑器、选项设置、生成工具以及在线帮助。您可以使用 μ Vision2 创建源文件，并组成应用工程加以管理。 μ Vision2 可以自动完成编译、汇编、链接程序的操作，使您可以只专注开发工作的效果。

- **C51 编译器和 A51 汇编器**

由 μ Vision2 IDE 创建的源文件，可以被 C51 编译器或 A51 汇编器处理，生成可重定位的 object 文件。Keil C51 编译器遵照 ANSI C 语言标准，支持 C 语言的所有标准特性。另外，还增加了几个可以直接支持 80C51 结构的特性。Keil A51 宏汇编器支持 80C51 及其派生系列的所有指令集。

- **LIB51 库管理器**

LIB51 库管理器可以从由汇编器和编译器创建的目标文件建立目标库。这些库是按规定格式排列的目标模块，可在以后被链接器所使用。当链接器处理一个库时，仅仅使用了库中程序使用的目标模块而不是全部加以引用。

- **BL51 链接器 / 定位器**

BL51 链接器使用从库中提取出来的目标模块和由编译器、汇编器生成的目标模块，创建一个绝对地址目标模块。绝对地址目标文件或模块包括不可重定位的代码和数据。所有的代码和数据都被固定在具体的存储器单元中。绝对地址目标文件可以用于：

- 编程 EPROM 或其它存储器设备；
- 由 μ Vision2 调试器对目标进行调试和模拟；
- 使用在线仿真器进行程序测试。

- **μ Vision2 软件调试器**

μ Vision2 软件调试器能十分理想地进行快速、可靠的程序调试。调试器包括一个高速模拟器，您可以使用它模拟整个 80C51 系统，包括片上外围器件和外部硬件。当您从器件数据库选择器件时，这个器件的属性会被自动配置。

- **μ Vision2 硬件调试器**

μ Vision2 调试器向您提供了几种在实际目标硬件上测试程序的方法。

- 安装 MON51 目标监控器到您的目标系统，并通过 Monitor-51 接口下载您的程序；
- 使用高级 GDI 接口，将 μ Vision2 调试器同类似于 TKS 系列仿真器的硬件系统相连接，通过 μ Vision2 的人机交互环境指挥连接的硬件完成仿真操作。

- **RTX51 实时操作系统**

RTX51 实时操作系统是针对 80C51 微控制器系列的一个多任务内核。RTX51 实时内核简化了需要对实时事件进行反应的复杂应用的系统设计、编程和调试。这个内核完全集成在 C51 编译器中，使用非常简单。任务描述表和操作系统的一致性由 BL51 链接器 / 定位器自动进行控制。

5.2 Keil 软件开发的流程

对于刚刚使用 Keil 的用户来讲，一般是按照下面的流程来完成开发任务的：

- 建立工程。
- 为工程选择目标器件（例如选择 PHILIPS 的 P89C58）。
- 设置工程的配置参数。
- 打开/建立程序文件。
- 编译和连接工程。
- 纠正程序中的书写和语法错误，并重新编译连接。
- 对程序中某些纯软件的部分使用软件仿真验证。
- 使用 TKS 硬件仿真器对应用程序进行硬件仿真。
- 将生成的 Hex 文件烧写到 ROM 中运行测试。

上面的流程只是一个标准的开发流程，实际中用户可能反复重复一个或几个步骤。

5.3 Keil 软件的安装

本章将解释如何设置操作环境以及如何将软件安装到您的硬盘上。

- 系统要求

必须满足最小的硬件和软件要求，才能确保编译器以及其它程序功能正常。您必须具有：

- Pentium、Pentium-II 或兼容处理器的 PC；
- Windows95、Windows98、Windows NT4.0；
- 至少 16MB RAM；
- 至少 20MB 硬盘空间。

- 安装详细说明

所有的 Keil 产品都自带一个安装程序和安装说明，非常易于安装。根据您得到的软件途径不同，软件的存放格式可能不同。

5.4 Keil 软件的工作环境

安装完成后，用户可以点击运行图标进入 IDE 环境。

μVision2 软件有菜单栏、可以快速选择命令按钮的工具栏、一些源代码文件窗口、对话框窗口、信息显示窗口。μVision2 允许同时打开几个源程序文件。

菜单栏命令、工具栏和快捷方式

菜单栏为您提供了各种操作菜单，比如：编辑器操作、工程维护、开发工具选项设置、程序调试、窗体选择和操作、在线帮助。工具栏按钮可以快速执行 μVision2 命令。快捷键（您可以自己配置）也可以执行 μVision2 命令。下表列出了 μVision2 的菜单项和命令、工具栏图标、默认快捷键以及它们的说明。

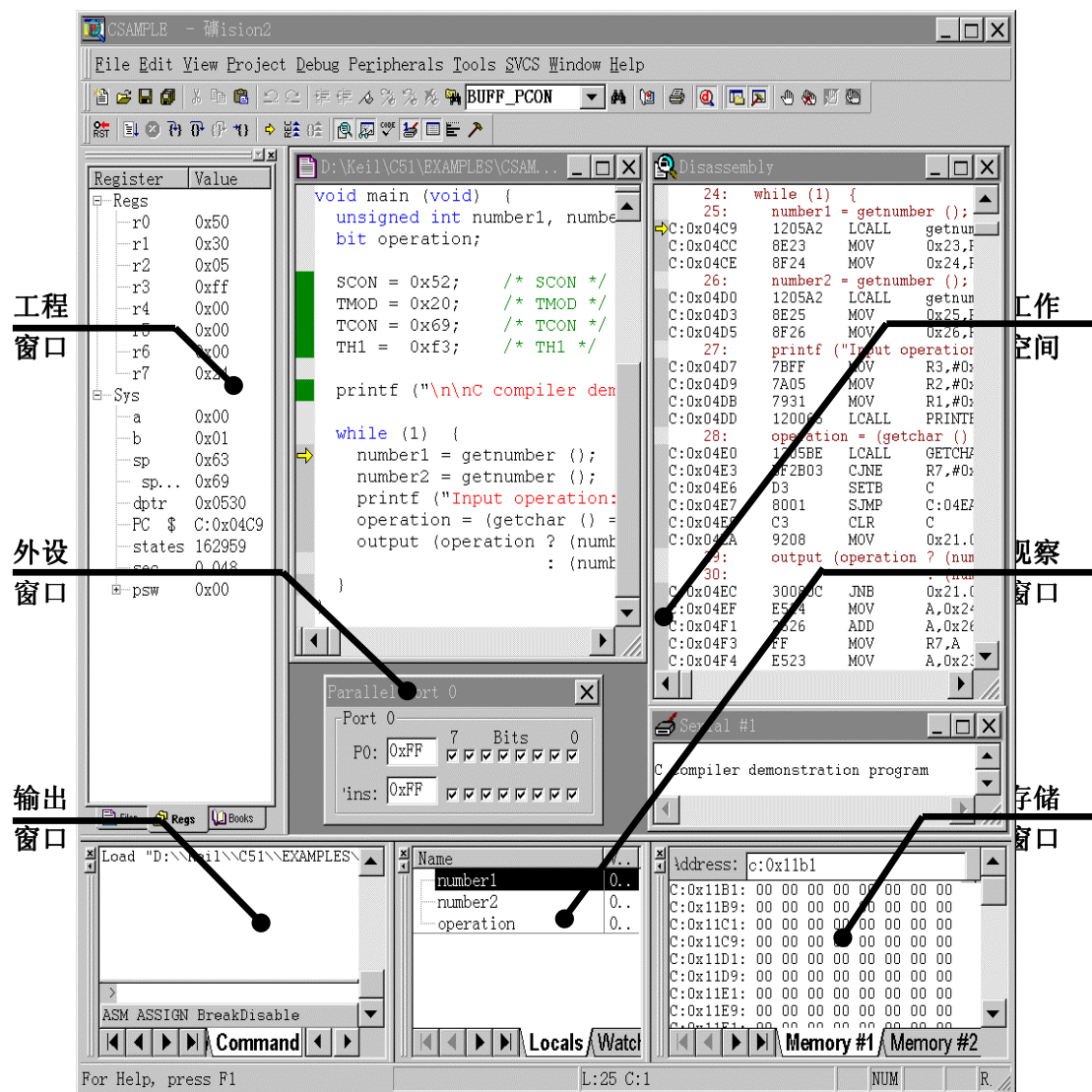
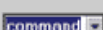


图 5.4.1 μVision2 操作界面

文件菜单和文件命令(File)

File 菜单	工具栏	快捷键	描述
New		Ctrl+N	创建一个新的源文件或文本文件
Open		Ctrl+O	打开已有的文件
Close			关闭当前的文件
Save		Ctrl+S	保存当前的文件
			保存所有打开的源文件和文本文件
Save as...			保存并重新命名当前的文件
Device Database			维护μVision2 器件数据库
Print Setup...			设置打印机
Print		Ctrl+P	打印当前的文件
Print Preview			打印预览
1-9			打开最近使用的源文件或文本文件
Exit			退出μVision2, 并提示保存文件

编辑菜单和编辑器命令(Edit)

Edit 菜单	工具栏	快捷键	描述
		Home	将光标移到行的开始处
		End	将光标移到行的结尾处
		Ctrl+Home	将光标移到文件的开始处
		Ctrl+End	将光标移到文件的结尾处
		Ctrl+←	将光标移到上一个单词
		Ctrl+→	将光标移到下一个单词
		Ctrl+A	选中当前文件中的所有文字
Undo		Ctrl+Z	撤销上一次操作
Redo		Ctrl+Shift+Z	重做上一次撤销的命令
Cut		Ctrl+X	将选中的文字剪切到剪贴板
		Ctrl+Y	将当前行的文字剪切到剪贴板
Copy		Ctrl+C	将选中的文字复制到剪贴板
Paste		Ctrl+V	粘贴剪贴板的文字
Indent Selected Text			将选中的文字向右缩进一个制表符位
Unindent Selected Text			将选中的文字向左缩进一个制表符位
Toggle Bookmark		Ctrl+F2	在当前行放置书签
Goto Next Bookmark		F2	将光标移到下一个书签
Goto Previous Bookmark		Shift+F2	将光标移到上一个书签
Clear All Bookmarks			清除当前文件中的所有书签
Find		Ctrl+F	在当前文件中查找文字
		F3	继续向前查找文字
		Shift+F3	继续向后查找文字
		Ctrl+F3	查找光标处（选中）的单词
		Ctrl+]	查找匹配的花括号、圆括号、方括号（使用这个命令时请将光标移到一个花括号、圆括号或方括号的前面）
Replace		Ctrl+H	替换特定的文字
Find in Files...			在几个文件中查找文字

选择文本命令

在µVision 中，您可以按下 Shift 键和相应的光标键来选择文字。例如，Ctrl+→是将光标移到下一个单词，而 Ctrl+Shift+→是选中从光标的位置到下一个单词开始前的文字，您也可以使用鼠标选择文字。

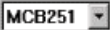
选择...	鼠标要...
任何数量的文字	在文字上拖动
一个单词	双击这个单词
一行文字	将鼠标移到行的左边，直到它变成一个指向右的箭头，然后点击
多行文字	将鼠标移到行的左边，直到它变成一个指向右的箭头，然后向上或向下拖动鼠标
垂直的一块文字	按着 Alt 键，然后拖动

视图菜单(View)

View 菜单	工具栏	快捷键	描述
Status Bar			显示或隐藏状态栏
File Toolbar			显示或隐藏文件工具栏
Build Toolbar			显示或隐藏编译工具栏
Debug Toolbar			显示或隐藏调试工具栏
Project Window			显示或隐藏工程窗口
Output Window			显示或隐藏输出窗口
Source Browser			打开源（文件）浏览器窗口
Disassembly Window			显示或隐藏反汇编窗口
Watch & Call			显示或隐藏观察和堆栈窗口
Stack Window			
Memory Window			显示或隐藏存储器窗口
Code Coverage Window			显示或隐藏代码覆盖窗口
Performance Analyzer Window			显示或隐藏性能分析窗口
Symbol Window			显示或隐藏符号变量窗口
Serial Window #1			显示或隐藏串行窗口 1
Serial Window #2			显示或隐藏串行窗口 2
Toolbox			显示或隐藏工具箱
Periodic Window Update			在运行程序时，周期刷新调试窗口
Workbook Mode			显示或隐藏工作簿窗口的标签
Options...			设置颜色、字体、快捷键和编辑器选项

工程菜单和工程命令(Project)

Project 菜单	工具栏	快捷键	描述
New Project...			创建一个新的工程
Import μ Vision1 Project...			输入一个 μ Vision1 工程文件
Open Project...			打开一个已有的工程

Close Project...			关闭当前的工程
Target Environment			定义工具系列、包含文件、库文件的路径
Targets, Groups, Files			维护工程的对象、文件组和文件
Select Device for Target			从器件数据库选择一个 CPU
Remove...			从工程中删去一个组或文件
Options...		Alt+F7	设置对象、组或文件的工具选项
			设置当前目标的选项
			选择当前目标
File Extensions			选择文件的扩展名以区别不同的文件类型
Build Target		F7	转换修改过的文件并编译成应用
Rebuild Target			重新转换所有的源文件并编译成应用
Translate...		Ctrl+F7	转换当前的文件
Stop Build			停止当前的编译进程
1-9			打开最近使用的工程文件

调试菜单和调试命令(Debug)

Debug 菜单	工具栏	快捷键	描述
Start/Stop		Ctrl+F5	启动或停止µVision2 调试模式
Debugging			
Go		F5	运行（执行），直到下一个有效的断点
Step		F11	跟踪运行程序
Step Over		F10	单步运行程序
Step out of current function		Ctrl+F11	执行到当前函数的程序
Stop Running		ESC	停止程序运行
Breakpoints...			打开断点对话框
Insert/Remove Breakpoint			在当前行设置 / 清除断点
Enable / Disable Breakpoint			使能 / 禁能当前行的断点
Disable All Breakpoints			禁能程序中所有断点
Kill All Breakpoints			清除程序中所有断点
Show Next Statement			显示下一条执行的语句 / 指令
Enable/Disable Trace Recording			使能跟踪记录，可以显示程序运行轨迹
View Trace Records			显示以前执行的指令
Memory Map...			打开存储器空间配置对话框

Performance	打开性能分析器的设置对话框
Analyzer...	
Inline Assembly...	对某一行重新汇编，可以修改汇编代码
Function Editor	编辑调试函数和调试配置文件

外围器件菜单(Peripherals)

Peripheral 菜单	工具栏	快捷键	描述
Reset CPU			复位 CPU
Interrupt, I/O-Ports, Serial, Timer, A/D Converter, D/A Converter, I2C Controller, CAN Controller, Watchdog			打开在片外围器件的对话框。对话框的列表和内容由您在器件数据库中选择 CPU 决定，不同的 CPU 会有所不同。

工具菜单(Tools)

通过工具菜单，可以配置和运行 Gimpel PC-Lint、Siemens Easy-Case 和用户程序。执行 Customize Tools Menu...，可以将用户程序添加到菜单中。

Tools 菜单	工具栏	快捷键	描述
Setup PC-Lint...			配置 Gimpel Software 公司的 PC-Lint
Lint			在当前的编辑文件中运行 PC-Lint
Lint all C Source Files			在工程的 C 源代码文件中运行 PC-Lint
Setup Easy-Case...			配置 Siemens Easy-Case
Start/Stop Easy-Case			启动或停止 Siemens Easy-Case
Show File (Line)			在当前编辑的文件中运行 Easy-Case
Customize Tools Menu...			将用户程序加入工具菜单

软件版本控制系统菜单(SVCS)

这个菜单可以配置和添加软件版本控制系统(Software Version Control System)命令。

SVCS 菜单	工具栏	快捷键	描述
Configure Version Control...			配置您的软件版本控制系统命令

视窗菜单(Window)

Window 菜单	工具栏	快捷键	描述
Cascade			层叠所有窗口
Tile Horizontally			横向排列窗口（不层叠）
Tile Vertically			纵向排列窗口（不层叠）
Arrange Icons			在窗口的下方排列图标
Split			将激活的窗口拆分成几个窗格
1-9			激活选中的窗口对象

帮助菜单(Help)

Help 菜单	工具栏	快捷键	描述
Help topics			打开在线帮助
About μ Vision			显示 μ Vision 的版本号和许可信息

5.5 Hello 测试程序

HELLO: 您的第一个 80C51 C 程序

HELLO 示范程序位于缺省目录 C:\KEIL\C51\EXAMPLES\HELLO\中。这个程序只是将文本“Hello World”输出到串行口中。整个程序只包含一个源文件 HELLO.C。这个小型的应用程序帮助您确定工具软件可以编译、链接和调试一个应用程序。

HELLO 的硬件是标准的 80C51 CPU。使用的唯一在片外围器件是串行口。您不需要一个实际的目标 CPU，因为 μ Vision2 可以模拟程序所需要的硬件。

打开 HELLO 工程文件

在 μ Vision 中，应用程序都包含在工程中。HELLO 的工程文件已经创建好，选择 Project 菜单的 Open Project，从文件夹...\C51\EXAMPLES\HELLO 打开 HELLO.UV2，载入工程文件（见图 5.5.1）。

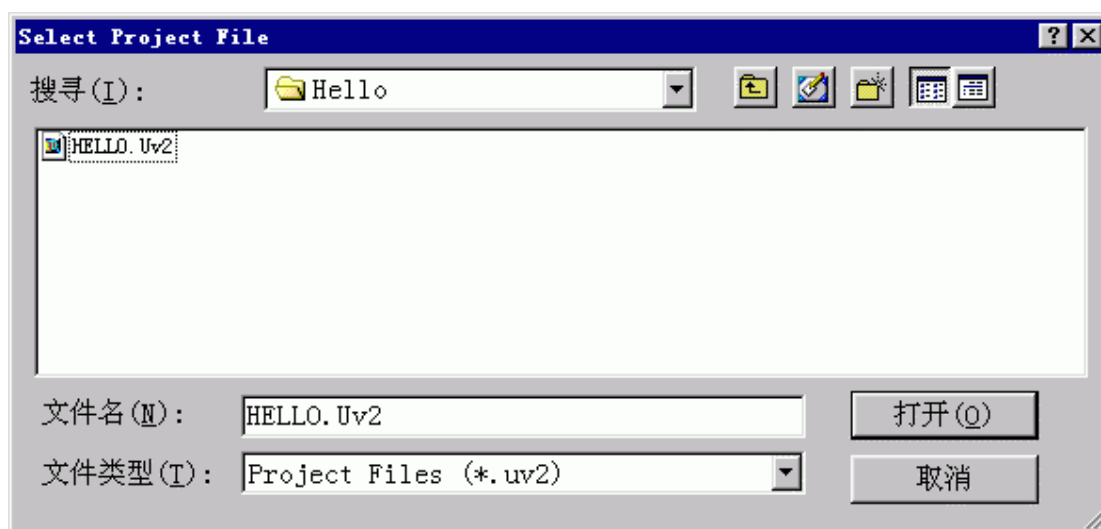


图 5.5.1 打开 HELLO 工程文件

编辑 HELLO.C

双击 Project Window 页中的 HELLO.C，现在就可以开始编辑 HELLO.C 了。 μ Vision2 在编辑器窗口载入和显示 HELLO.C 的内容。

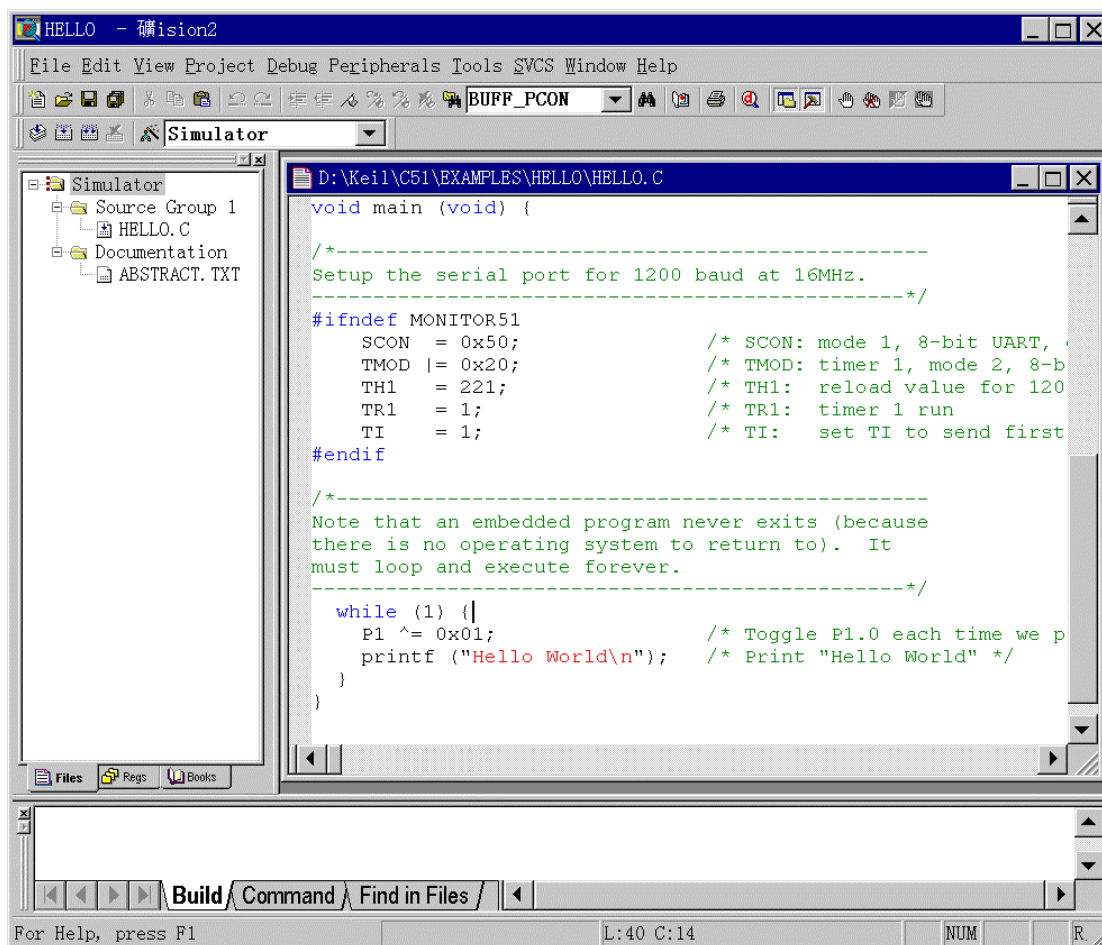


图 5.5.2 Hello 工程的编辑/编译界面

编译和链接 HELLO

用 Project 菜单或工具栏按钮的 Build Target 命令编译和链接工程。μVision2 开始编译和链接源文件，并创建一个可以载入到 μVision2 调试器调试的绝对目标模块。编译过程的结果列在 Output Window 的 Build 页上（见图 5.5.3）。

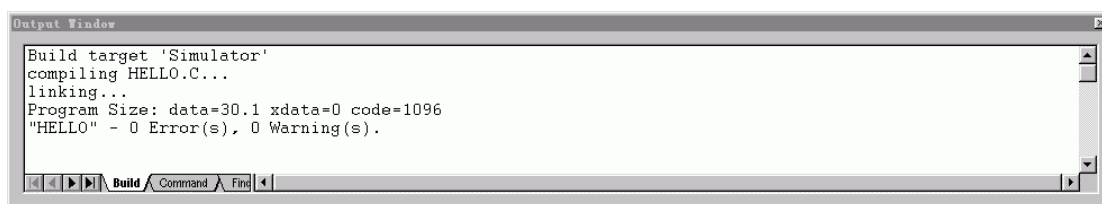


图 5.5.3 Hello 工程的编译结果输出

注意：在 μVision2 使用提供的示范程序不应该会有错误。

测试 HELLO

HELLO 程序被编译和链接后，您可以用 μVision2 调试器对它进行测试。在 μVision2 Debug 菜单或工具栏按钮用 Start/Stop Debug Session 命令可以开始测试。μVision2 初始化调试器并启动运行程序直到 main 函数。下面是显示的屏幕：

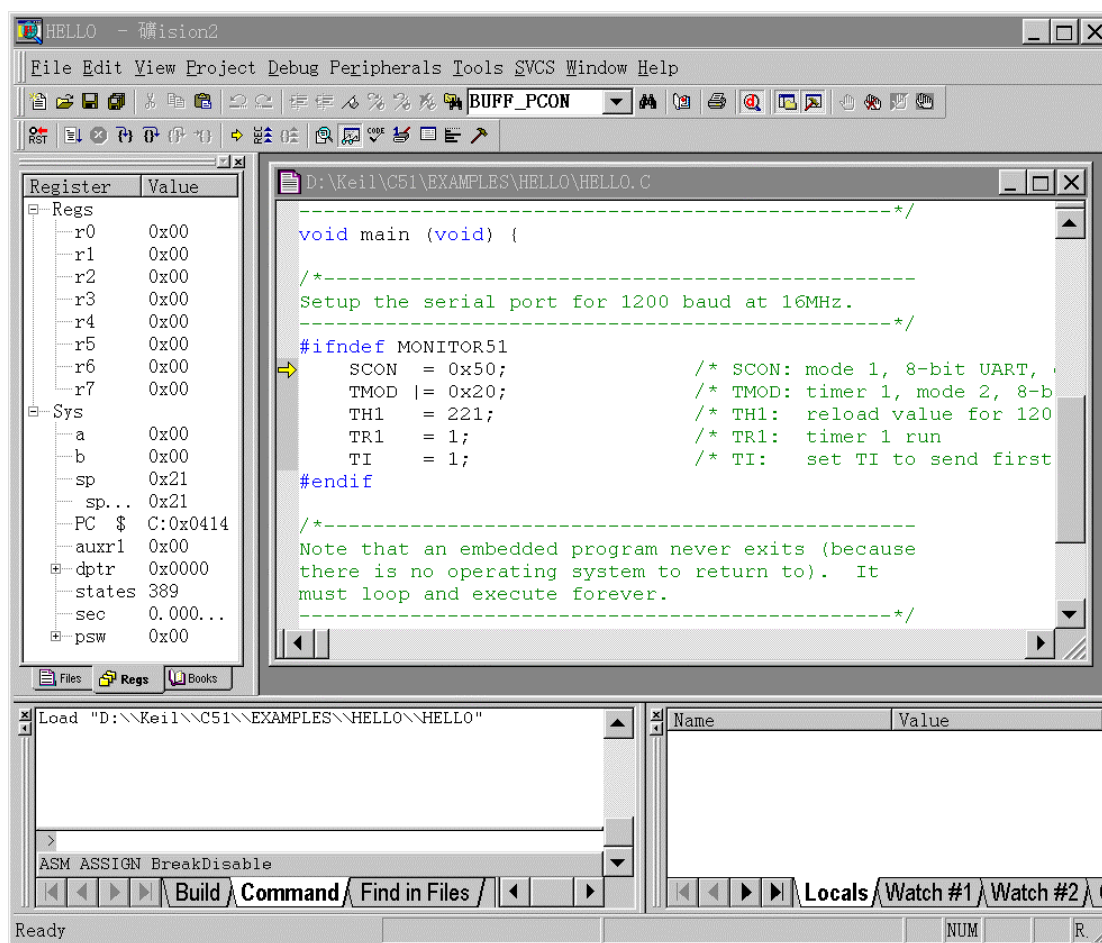


图 5.5.4 Hello 工程的调试界面 1



打开 Serial Window #1，用 View 菜单或 Debug 工具栏的 Serial Window #1 命令显示应用程序的串行（Serial）输出。



用 Debug 菜单或工具栏的 Go 命令运行（Run）HELLO 程序。HELLO 程序执行后在 Serial 窗口显示文字“Hello World”。在 HELLO 输出“Hello World,”后，它开始执行一个无限的循环。



用 Debug 菜单或工具栏的 Halt 命令停止运行（Stop Running）HELLO 程序。您也可以在 Output 窗口的 Command 页输入 ESC 停止运行。

调试过程中， μ Vision2 会显示下面的输出：

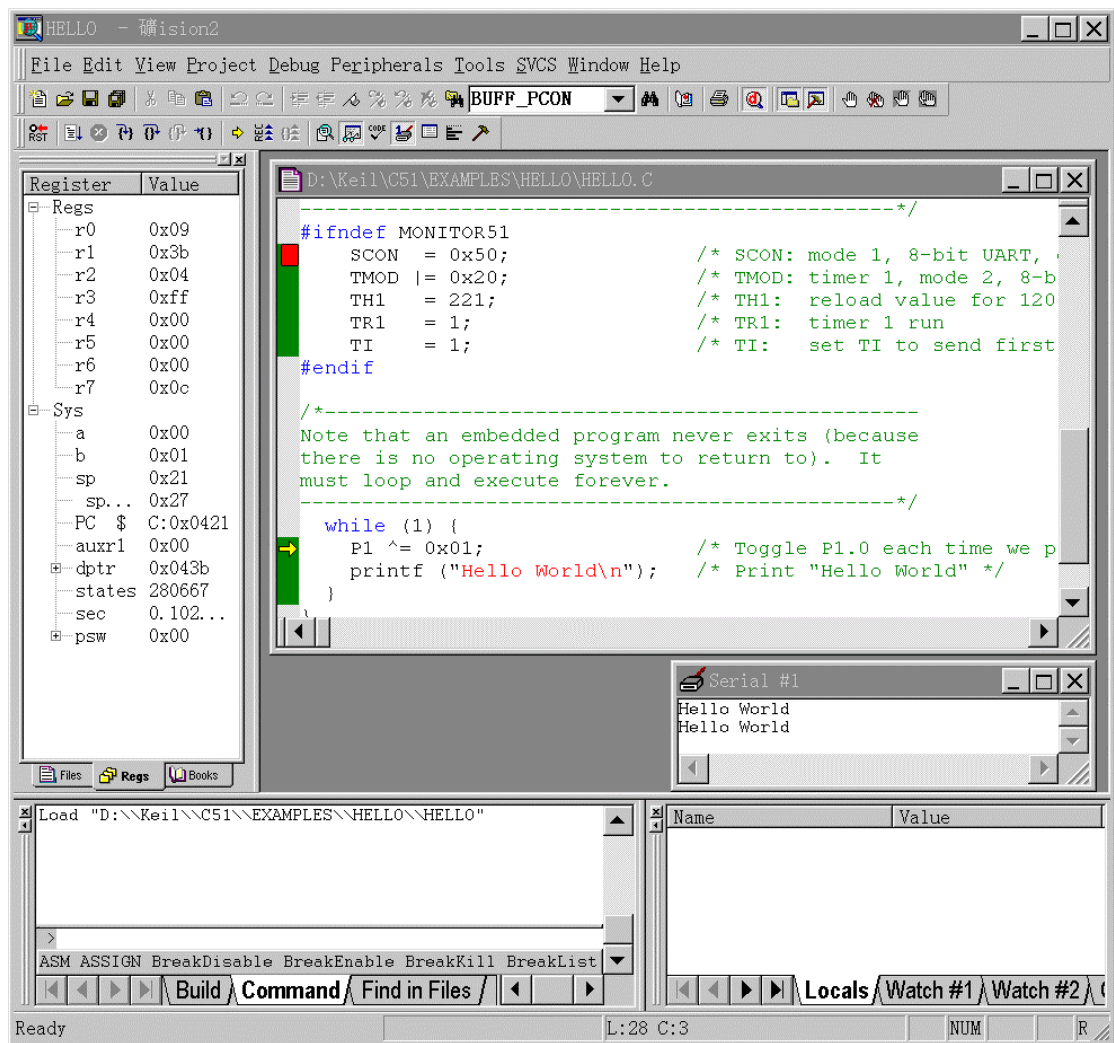


图 5.5.5 Hello 工程的调试界面 2

单步和断点



用工具栏或鼠标右键打开的 local 编辑器菜单的 Insert/Remove Breakpoints 命令在 main 函数的开始处设置一个断点。



用 Debug 菜单或工具栏的 Reset CPU 命令。如果 HELLO 停止运行，Run 命令可以使它继续执行。μVision2 会在断点处停止程序。



用 Debug 工具栏的 Step 按钮可以单步执行 HELLO 程序。当前的指令用黄色箭头标出。每执行一步箭头都会移动。



将鼠标移到一个变量上可以看到它们的值。



任何时间都可以用 Start/Stop Debug Session 命令停止调试。

第六章 TKS 在 Keil 环境下仿真的详细范例

在第五章中讲述了使用 Keil 处理 Hello 工程的实例。Hello 工程是一个已经建立好的工程，工程的配置/器件的选择等操作并没有提到，并且使用的是软件仿真。在这一章中，我们将从建立工程开始详细描述一个工程的处理过程，特别描述 Keil 同 TKS 硬件仿真器的配合。

6.1 建立工程

为什么要使用工程？

在 Keil 中文件的管理使用的工程的方法，而不是单一文件的模式。您的所有的文件包括源程序（包括 C 程序，汇编程序），头文件，甚至说明性的技术文档都可以放在工程里统一管理。在使用 Keil 前，您应该习惯这种工程的管理方式。

如何建立 μ Vision2 的工程？

启动 μ Vision2 后， μ Vision2 总是打开用户前一次处理的工程，您可以点击菜单 Project-> Close Project 关闭。要建立一个新工程，您可以点击 Project-> New,出现对话框（见图 6.1.1）。



图 6.1.1 新建工程对话框

在这里，您需要做的事情是：

- 为您的工程选一个名称。
- 选择工程存放的路径。建议为每个工程单独建立一个目录，并且工程中需要的所有文件都放在这个目录下。
- 在选择了工程目录和名称后，点击保存 返回。

6.2 为工程选择目标器件

在工程建立完毕以后， μ Vision2 会立即弹出器件选择窗口(见图 6.2.1)。器件选择的目的是告诉 μ Vision2 您最终 80C51 芯片的型号是哪一个公司的哪一个型号，因为不同型号的 51 芯片内部的资源是不同的， μ Vision2 可以根据您的选择为您进行 SFR 的预定义，在软硬件仿真中提供易于操作的外设浮动窗口等。

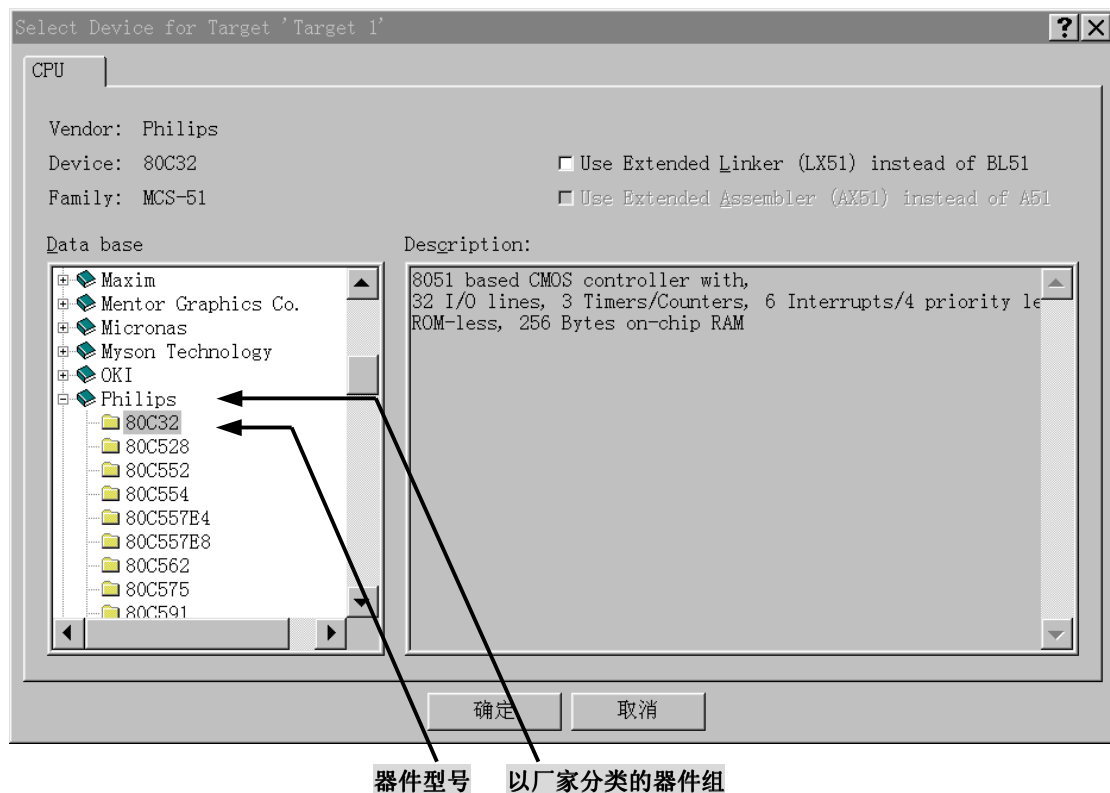


图 6.2.1 目标器件选择窗口

在图 6.2.1 中， μ Vision2 支持的所有型号根据生产厂家形成器件组。用户可以点开相应的器件组并选择相应的器件型号。

另外，如果用户在选择完目标器件后想重新改变目标器件，可点击菜单 Project->Select Device for..出现图 6.2.1 的对话框后重新加以选择。由于不同厂家的许多型号性能相同或相近，因此如果用户的目标器件型号在 μ Vision2 中找不到，用户可以选择其它公司的相近型号。

6.3 建立/编辑程序文件

到现在用户已经建立了一个空白的工程文件，并为工程选择好了目标器件，但是这个工程里没有任何程序文件。程序文件的添加必须人工进行，如果在程序文件在添加前还没有创立，用户还必须建立它。

建立程序文件

点击菜单 File->New 后在文件窗口会出现 Text1 的新文件窗口，如果多次点击 File->New 则会出现 Text2, Text3... 等多个新文件窗口。现在我们在 Keil 中有了一个名字为 Text1 新文件框架，我们还需要把它保存起来，并为它起一个正式的名字。

点击菜单 File->Save As ，出现图 6.3.1 的对话框。在文件名栏输入文件的正式名称，这里我们

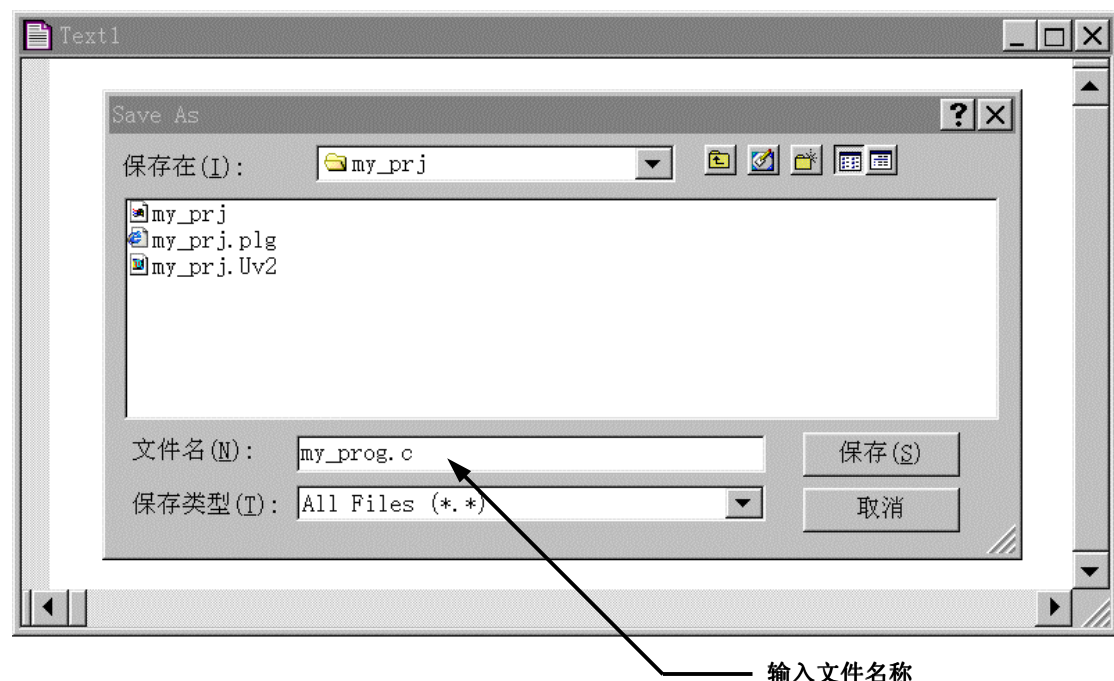


图 6.3.1 保存新建文件

把 Text1 保存成 my_prog.c。注意文件的后缀，因为 μ Vision2 要根据后缀判断文件的类型，从而自动进行处理。my_prog.c 是一个 c 语言程序，如果用户想建立的是一个汇编程序，则输入文件名称 my_prog.asm。唯一需要注意的是文件要保存在同一的工程目录 my_prj 中，而不要放置在其它的目录中，否则容易造成工程管理混乱。

编辑程序文件

上面我们建立了一个叫 my_prog.c 的 C 语言程序文件，但是这个文件是一个空文件，我们还必须编辑它，写入我们的程序代码。在 μ Vision2 中，文件的编辑方法同其它文本编辑器是一样的，用户执行可以输入，删除，选择，拷贝，粘贴等基本功能。

μ Vision2 不完全支持汉字的输入和编辑，因此如果用户需要编辑汉字最好使用外部的文本编辑器来编辑。编辑完毕后保存到磁盘中， μ Vision2 中有文件变化感知功能，提示您外部编辑器改变了该文件，是否需要把 μ Vision2 中的该文件刷新。您应该选择是，然后您就可以看到 μ Vision2 中文件的刷新。

为了以后说明的方便，我们在这里给出一个程序范例，并给出详细的程序说明。

```

/*****
/* 程序说明:
/* 依次点亮在 P1 口外部的 LED,并无限循环
*****/

#include <REG52.H> /* 声明头文件, 定义标准 51 的特殊功能寄存器 */

void Delay(unsigned int DelayTime ) /* 延时子程序 */
{ if( DelayTime == 0 )return; /* 如果延时为 0 则立即返回 */
  while( DelayTime != 0)
  { DelayTime--;} /* 利用减 1 的执行时间达到延时目的 */
}

void mian() /* 主运行程序 */
{ data unsigned char LedBuff; /* 保存当前 LED 的状态 */

  for(;;) /* 连续运转 */
  { LedBuff = 0X01; P1 = LedBuff; /* 点亮 P1.0 脚接的 LED */
    Delay(10000); /* 延时,数值用户可以尝试改变 */
    LedBuff = 0X02; P1 = LedBuff; /* 点亮 P1.1 脚接的 LED */
    Delay(10000); /* 延时,数值用户可以尝试改变 */
    LedBuff = 0X04; P1 = LedBuff; /* 点亮 P1.2 脚接的 LED */
    Delay(10000); /* 延时,数值用户可以尝试改变 */
    LedBuff = 0X08; P1 = LedBuff; /* 点亮 P1.3 脚接的 LED */
    Delay(10000); /* 延时,数值用户可以尝试改变 */
    LedBuff = 0X10; P1 = LedBuff; /* 点亮 P1.4 脚接的 LED */
    Delay(10000); /* 延时,数值用户可以尝试改变 */
    LedBuff = 0X20; P1 = LedBuff; /* 点亮 P1.5 脚接的 LED */
    Delay(10000); /* 延时,数值用户可以尝试改变 */
    LedBuff = 0X40; P1 = LedBuff; /* 点亮 P1.6 脚接的 LED */
    Delay(10000); /* 延时,数值用户可以尝试改变 */
    LedBuff = 0X80; P1 = LedBuff; /* 点亮 P1.7 脚接的 LED */
    Delay(10000); /* 延时,数值用户可以尝试改变 */
  }
}
*****/

```

用户可以把上面的程序拷贝到 my_prog.c 文件中,并使用菜单 File->Save 加以保存。利用这种建立编辑程序文件的方法, 您可以同样建立起其它的程序文件。

6.4 为工程添加文件

我们在章节 6.3 中建立了一个程序文件，并初步把它编辑成我们需要的格式。需要特别提出的是，这个程序文件我们仅仅是建立了它，除了它保存在\my_prj 的目录中外，my_prj.c 文件到现在为止跟 my_prj 工程还没有建立起任何关系。我们现在要做的工作就是要把 my_prog.c 添加到 my_prj 工程中。

要求添加文件

用鼠标的右键点击 Source Group1，弹出的菜单（见图 6.4.1 添加工程文件）。

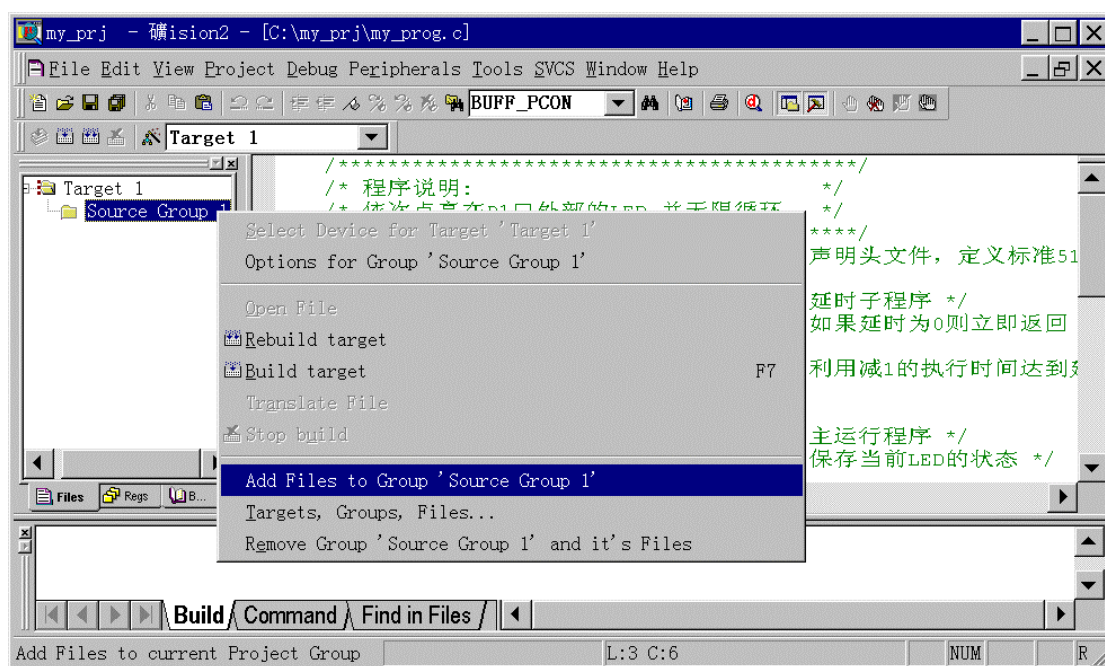


图 6.4.1 添加工程文件

添加文件定位

在菜单中选择 Add files to Group “Source Group1”（向工程中添加程序文件）后，弹出文件选择窗口（见图 6.4.2 选择要添加的工程文件）。

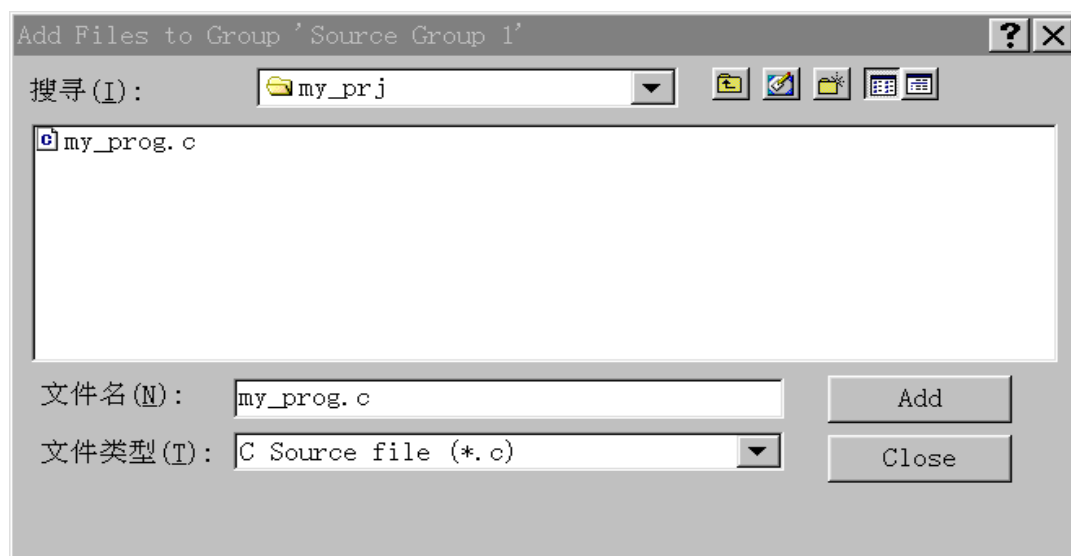


图 6.4.2 选择要添加的工程文件

加入工程文件

在图 6.4.2 所示的窗体中，用户可以根据要加入的文件类型显示出所有符合的文件列表。图 6.4.2 中只有一个文件，用鼠标点击 my_prj.c 选中，然后点击按钮 **Add**，就将 my_prj.c 加入到工程中。在图 6.4.3 中可以看到在工程 Target 1 下的组 Source Group1 中已经加入了文件 my_prj.c。

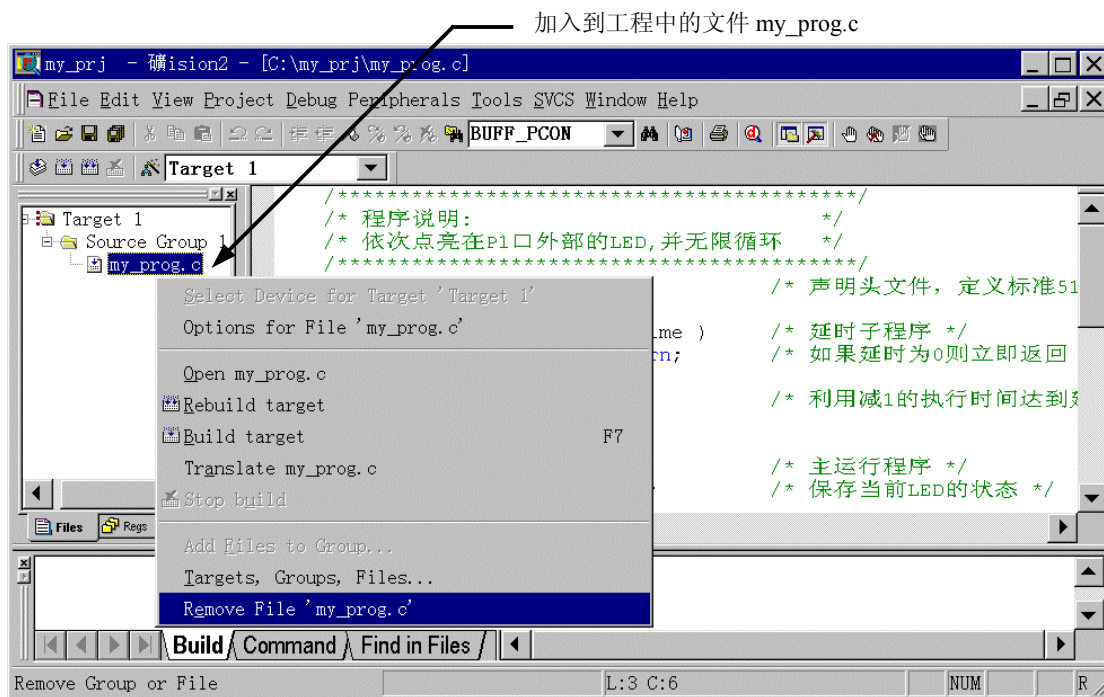


图 6.4.3 删除工程中的文件

删除工程文件

如果用户想删除已经加入的工程文件，可以用鼠标右键点击该文件，弹出选择菜单（见图 6.4.3），选择 **Remove File...**，可以将文件从工程中删除，但该文件并没有从磁盘上删除，仍旧保留在原目录下。

6.5 对工程进行设置

在工程建立以后，还需要对工程进行设置。工程的设置分软件设置和硬件设置。硬件设置主要针对仿真器，用于硬件仿真时使用；软件设置主要用于程序的编译和连接，也有一些参数用于软件仿真。对于软件和硬件的设置，用户都应该仔细选择，不恰当的配制会使用户的一些操作无法完成。

使用鼠标右键点击工程名 Target 1，出现选择菜单（见图 6.5.1）。

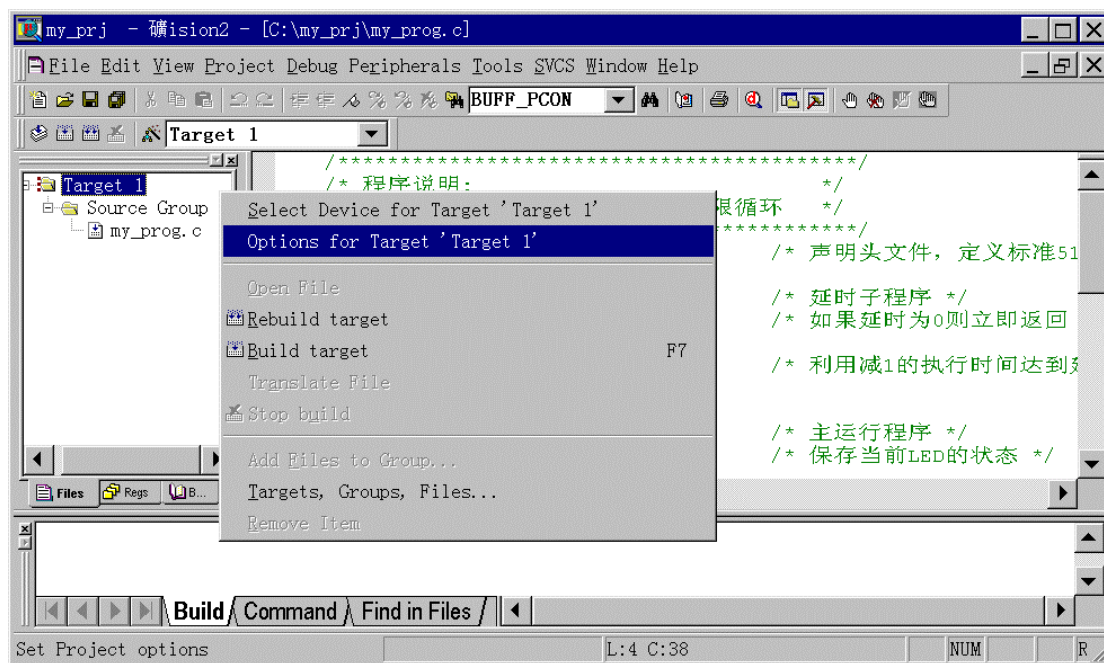


图 6.5.1 选择工程设置

选择菜单上的 Option for Target ‘Target 1’ 后，出现工程的配置窗口（见图 6.6.1）。一个工程的配置分成 8 部分：

- Target : 用户最终系统的工作模式的设定，它决定用户系统的最终框架。
- Output : 工程输出文件的设定。例如是否输出最终的 Hex 文件以及格式设定。
- List : 列表文件的输出格式设定。
- C51 : 使用 C51 处理的一些设定。
- A51 : 使用 A51 处理的一些设定。
- BL51 Location: 连接时用户资源的物理定位。
- BL51 MISC : BL51 的一些附加设定。
- Debug : 硬件和软件仿真的设定。

在工程的 8 种设定中，Target，C51 和 Debug 最为重要，其余的设定在一般的工程设计中不需要特别改动，使用 μ Vision2 的缺省设定就可以了。至于全面的设置规范，请 TKS 系列仿真器用户参考随机携带的其它资料。对于一般使用的用户，可以只关心 Target，C51 和 Debug 设定，其余的设定使用缺省设定。

6.5.1 Target 设置

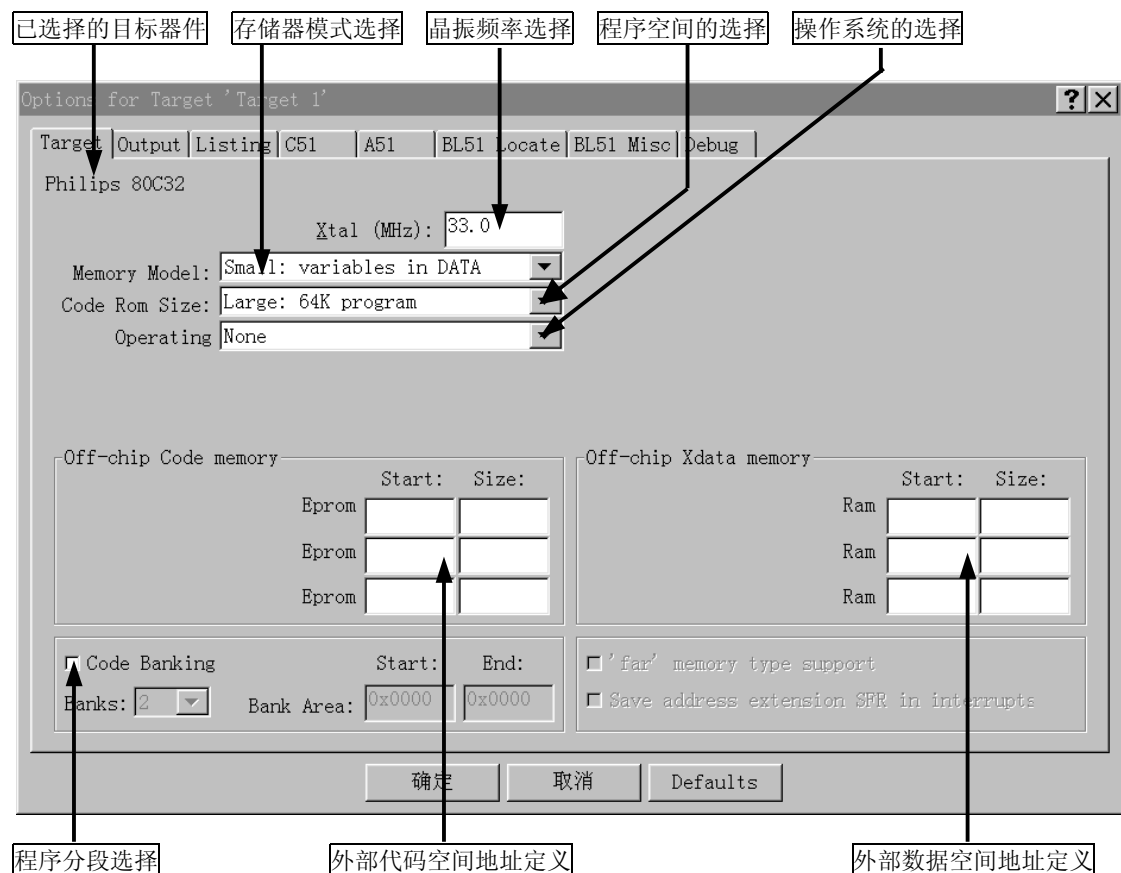


图 6.5.2 Target 设置窗口

1. **已选择的目標器件:**是用户在建立工程时选择的目標器件型号，这里不能更改。如用户需要更改，可关闭当前窗体，在工程窗口中点击工程名 Target 1，出现浮动菜单后选择 Select Device for Target ‘Target 1’。

2. **存储器模式选择:**存储器模式有 3 种可以选择：

- **Small:** 没有指定区域的变量缺省放置在 data 区域内。
- **Compact:** 没有指定区域的变量缺省放置在 pdata 区域内。
- **Larger:** 没有指定区域的变量缺省放置在 xdata 区域内。

存储器模式的作用主要是对这样的变量定义起作用，例如一个变量声明：`unsigned char Temp1`，根据用户设置存储器模式的不同，编译器会在 data(80C51 内部可直接寻址数据空间)或 pdata(外部一个 256 字节的 xdata 页)或 xdata（外部数据空间）。但是如果用户在变量声明时指定了空间类型，例如：`data unsigned char Temp1`，则存储器模式的选择将 Temp 变量没有约束作用，Temp 总被安排在 data 空间。

3. **晶振频率选择:**晶振的选择主要是在软件仿真时起作用， μ Vision2 将根据您的输入频率来决定软件仿真时系统运行的时间和时序。这个设置在硬件仿真时完全没有作用，请用户注意！

4. **程序空间的选择:** 选择用户程序空间的大小。

5. **操作系统的选择:** 是否选用操作系统。

6. 程序分段选择： 是否选用程序分段，这种功能一般用户不会使用到。

7. 外部代码空间地址定义： 这个选项主要是用在用户使用了外部程序空间，但在物理空间上又并不是连续的。通过这个选项最多有 3 个 起始地址和结束地址的输入， μ Vision2 在连接定位时将把程序安排在有效的程序空间内。这个选项一般只用于外部扩展的程序，因为 MCU 的内部程序空间多数都是连续的。

8. 外部数据空间地址定义： 这个选项类似于 7，但是应用与外部数据空间的定义。

6.5.2 C51 设置

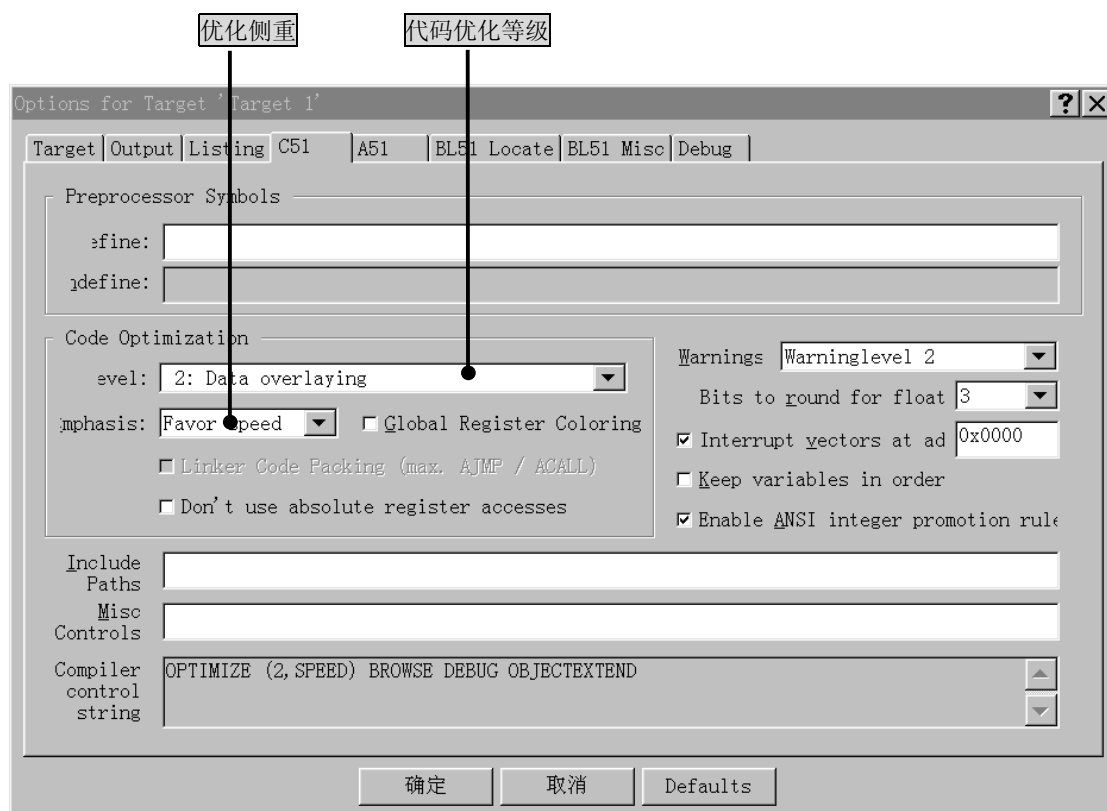


图 6.5.3 C51 设置窗口

代码优化等级

C51 在处理用户的 c 语言程序时能自动对您的程序做出优化，用于减少代码量或提高速度。经验证明，调试初期选择优化等级 2（Data overlaying）是比较明智的，因为根据您的程序不同，选择高级别的优化等级有时候会出现错误。

注意：在我们的 my_prj 工程中,请选择优化等级 2。在用户程序调试成功后再提高优化级别改善您的程序代码。

优化侧重

用户优化的侧重，分成：

- **Favor speed:** 优化时侧重优化速度。
- **Favor size:** 优化时侧重优化代码大小。
- **Default:** 不规定，使用缺省优化。

6.5.3 Debug 设置

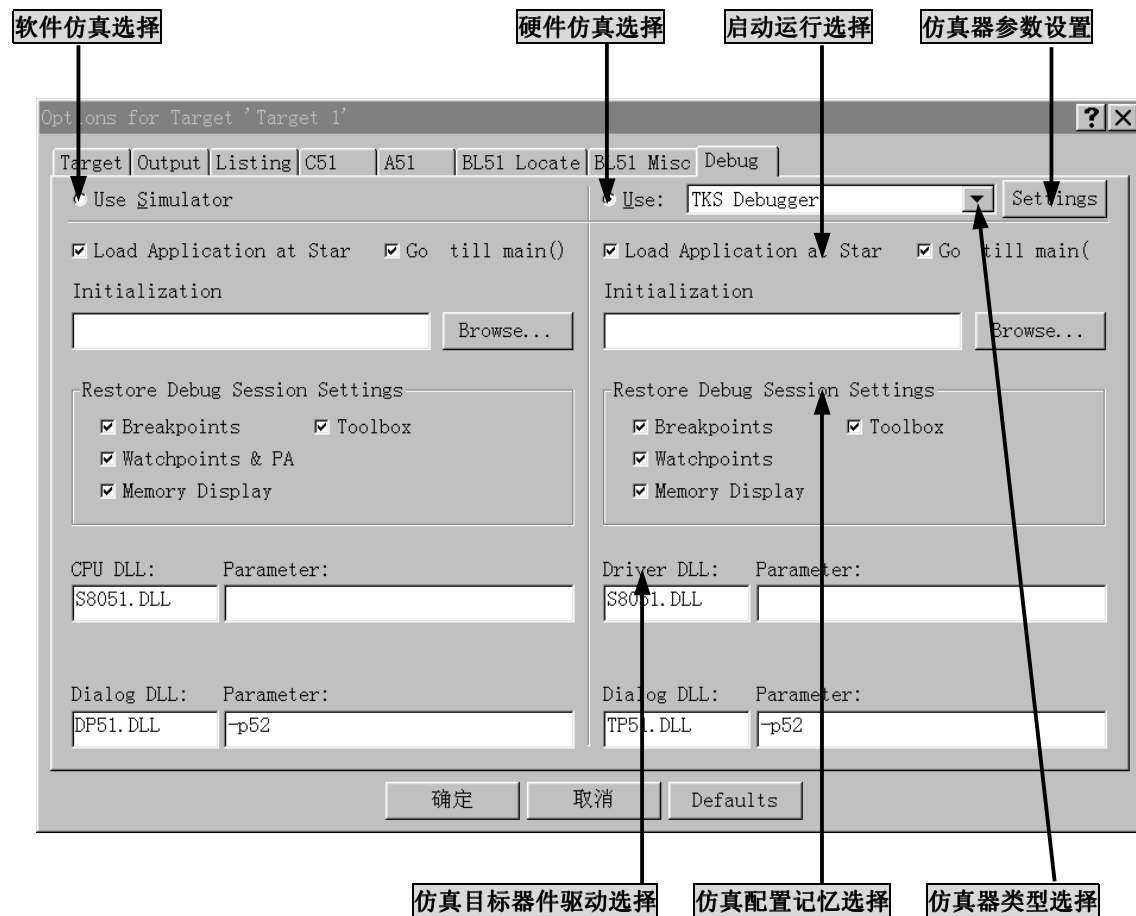


图 6.5.4 工程设置 Debug 窗口

Debug 设置窗口分成两部分：软件仿真设置（左边）和硬件仿真设置（右边）。软件仿真和硬件仿真的设置基本一样，只是硬件仿真设置增加了**仿真器参数设置**。在下面我们只介绍硬件仿真的设置。

软件仿真选择 **硬件仿真选择**

选择当前仿真的模式。软件仿真，使用计算机来模拟程序的运行。用户不需要建立硬件平台，就可以快速的得到某些运行结果。但是在仿真某些依赖于硬件的程序时，软件仿真则无法实现。硬件仿真是最准确的仿真方法，因为它必须建立起硬件平台，通过 PC<->硬件仿真器<->用户目标系统进行系统调试。适当的结合两种仿真方法，可以快速的对程序进行验证。

启动运行选择

选择在进入仿真环境中的启动操作。

- Load Application at Start: 进入仿真后将用户程序代码下载到仿真器。
- Go till main: 在使用 C 语言设计时,下载完代码则直接运行到 main（）函数位置。

仿真配置记忆选择

对用户仿真时的操作进行记忆。

- Breakpoints: 选中后记忆当前设置的断点，下次进入仿真后该断点设置存在并有效；
- Watchpoints: 选中后记忆当前设置的观察项目，下次进入仿真仍有效；
- Memory Display: 选中后记忆当前存储器区域的显示，下次进入仿真仍有效；
- Toolbar: 选中后记忆当前的工具条设置，下次进入后仍有效。

仿真目标器件驱动程序选择

如果用户在目标器件选择中选择了相应的器件，Keil 将自动选择相应的仿真目标器件驱动程序。在图 6.5.4 中，Keil 选择了标准 S80C51.DLL 驱动文件，Dialog .DLL 选择 p591。Keil 根据不同的器件选择不同的仿真驱动 DLL，这样在仿真时就会有该器件相应的外设菜单。用户进入仿真（硬件仿真和软件仿真）后，在 Peripherals 菜单中会添加该器件的外设观察菜单，用户点击后会出现浮动的观察窗口，方便用户观察和修改。

仿真器类型选择

用于选择当前 Keil 可以使用的硬件仿真设备。

任何可以挂接 Keil 仿真环境的硬件都必须提供驱动程序，驱动程序是 dll 文件。TKS 系列仿真器也提供 Keil 下的驱动程序，并根据仿真器设备不同而不同，驱动程序和详细说明都在随机提供的光盘中提供。

当用户得到驱动程序 dll 后，还必须在 Keil 的配置文件中声明，才能在**仿真器类型选择**中找到该硬件设备。不同的 TKS 系列仿真器可能使用不同的驱动 dll，具体使用的 dll 参考相应型号的说明。

仿真器驱动程序声明：

注意：这是 TKS-58/61/RD+/RD2/668/591/554 的声明，其它型号参考相应的使用手册！

在配置仿真器前，请将随机提供的 TKS 系列仿真器驱动文件 TKS_DEB.DLL 拷贝到 Keil 的安装目录 C51\bin 下。例如，如果您的 Keil 的安装目录为 C:\Keil，则将 TKS_DEB.DLL 拷贝到 C:\Keil\C51\bin 目录下。然后打开 C:\Keil 目录下的 Tools.ini 文件，在几个分类中找到[C51]，加入下列描述：

TDRV3=C:\Keil\C51\bin\TKS_DEB.DLL ("TKS Debugger")

其中：TDRV3 是驱动 DLL 的序号，如果您前面已经安装了多个驱动 DLL 以致占用了 TDRV3，则将 TKS 的驱动程序序号向后顺延，例如 TDRV5。在随机软件中，有一个示范 Tools.ini 文件，可以用来参考如何加入驱动程序。需要注意的是该文件中驱动程序说明行所指出的路径可能和您的不同，这要根据您的 Keil 软件安装在硬盘中的位置。“C:\Keil\C51\bin\TKS_DEB.DLL”是驱动程序放置的路径，该路径下必须有 TKS_DEB.DLL 文件。

6.6 对仿真器进行设置

在图 6.5.4 工程设置 Debug 窗口中, 点击 Settings 后出现配置窗体, 进入仿真器参数设置。注意, 不同的仿真器型号可能配置的窗体有所不同, 详细资料参考相应的使用手册。在设置中, 请用户仔细按照下面的要求选择, 不正确的选择将造成仿真的困难和不准确, 甚至是难以理解的现象!

注意: 这是 TKS-58/61/RD+/RD2/668/591/554 的设置, 其它型号参考相应的使用手册。

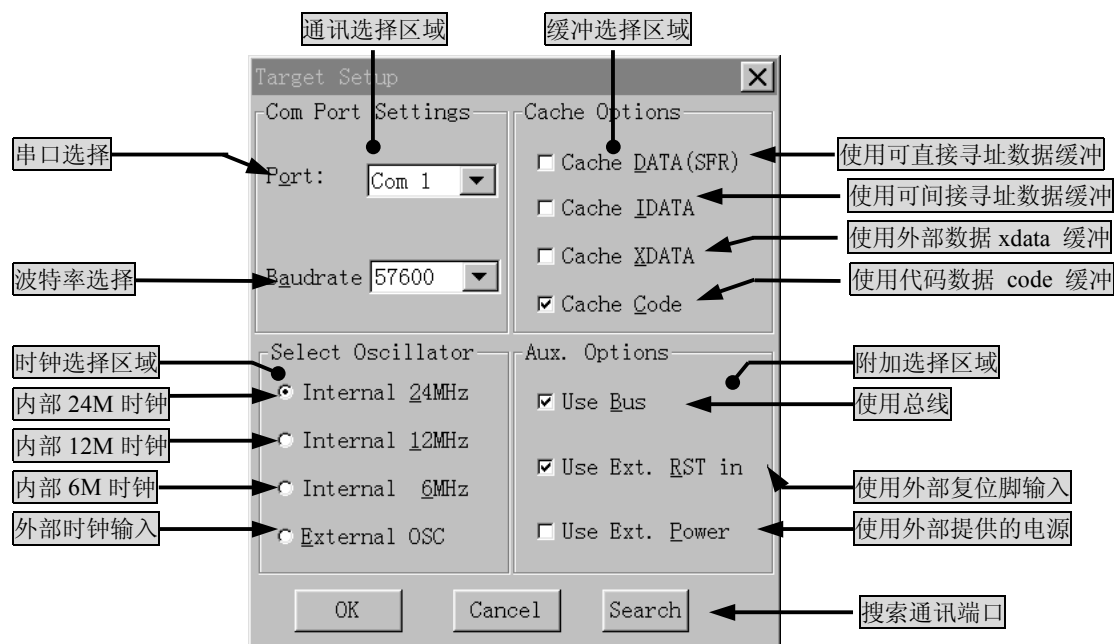


图 6.6.1 仿真硬件配置界面

Com Port Settings 串口设置区域

- **Port:** 选择硬件仿真器使用的串口。TKS 使用串口同 Keil 进行通讯。
- **Baudrate:** 串口使用的波特率。TKS 使用的是自动适应波特率, 也就是该选项在一定选择范围内 TKS 都能正常工作。选择高的波特率能加快运行速度, 但是低的波特率在环境恶劣的条件下更可靠。TKS 适应的波特率范围在 1200-57600 之间, 但根据用户的实际情况略有差异。

注意: 选定波特率后, 必须先接通 TKS 系列仿真器电源, 仿真器主机面板上的指示灯闪烁结束后, 才可以进入 Keil 的硬件仿真程序。在硬件仿真环境中用户可以在 Peripherals 菜单中打开 Target Settings 窗口 (图 6.6.1) 来更改波特率, 更改后整个系统重新建立连接; 也可以更改串口, 但是用户必须在更改后的串口上连接仿真器。

Catch Option 缓存设置区域

使用存储器缓冲区域, 这样在一般的操作中仿真软件不用频繁的读取仿真器中的内容, 而使用缓冲区域。使用缓冲可以大大加快仿真速度, 建议用户将该区域选项全部选中! 使用缓冲的缺点是屏幕的数据刷新慢, 但是在单步或运行后所有在屏幕上显示的信息将全部刷新一次。

Select Oscillator 时钟选择区域

选择仿真时使用的时钟。用户可以选择仿真器内部提供的 3 个时钟频率选择, 也可以选择外部时钟。

选择仿真内部时钟系统的性能最稳定，但是只提供 24M/12M/6M 3 个频点给用户使用；选择外部时钟用户可以根据自己的要求来选择频率，只需要在仿真头上插入希望的晶振或使用目标系统板上的晶振。但是由于时钟信号从外部引入到仿真器内部，中间要经过仿真电缆和接插件，因此稳定性和抗干扰性能不如使用内部时钟。

Aux. 附加选择区域

- **Use Bus 使用总线选择**

用户要根据自己的实际情况选择。如果用户系统中使用了总线（即 P0 P2 口作为总线而不是作为 I/O），请选择该项，这样仿真器才能打开总线并完成总线的读写功能；如果用户没有使用总线，即 P0 P2 口单纯作为 I/O 使用，这时用户不要选择该项。如果用户选择该项，则在仿真某些型号的单片机的 MOVX 操作会引起总线动作而影响 I/O 的状态。

- **Use Ext. RST in 使用外部复位输入选择**

如果用户选择该项，则仿真运行动作（单步，全速，全速断点）会根据外部复位信号的电平来动作。如果在开始运行前发现有效的外部复位电平，仿真器将提示用户，并停止运行，等待用户解除外部复位；如果在运行当中发现有外部复位信号，系统也将提示用户，并复位系统，复位信号解除后重新运行。

- **Use Ext. Power 使用外部电源选择**

使用外部电源作为仿真电源。在不使用外部电源时，使用的是内部的+5V 电源，要求用户板上的电源也必须是+5V，但是允许有 0.7V 的上下偏差。如果用户需要仿真非+5V 供电的系统，则可以选择该项。使用的外部电源电压必须在 2.7V-5.5V 之间。外部输入电源从用户板得到，经过仿真头电源脚进入仿真器内部。外部输入电源最低电压为 2.7V，小于 2.7 伏内部电压将维持在 2.7 伏不变；外部输入电压最大为 5.5V，超过 5.5V 将引起内部保护装置的启动。需要注意的是，如果您要仿真的单片机不支持低电压运行，请不要选择该项。TKS 系列仿真器内部有检查装置，如果仿真器件不支持低电压运行，将不会使用外部电源。

6.7 对工程进行编译/连接

在μVision2 环境中，用户程序编写完毕后还需要编译和连接才能够进行软件和硬件仿真。在程序的编译/连接中，如果用户程序出现错误，还需要修正错误后重新编译/连接。

程序的编译/连接

点击菜单 Project 出现选择菜单（见图 6.7.1），选择 Rebuild all target files。

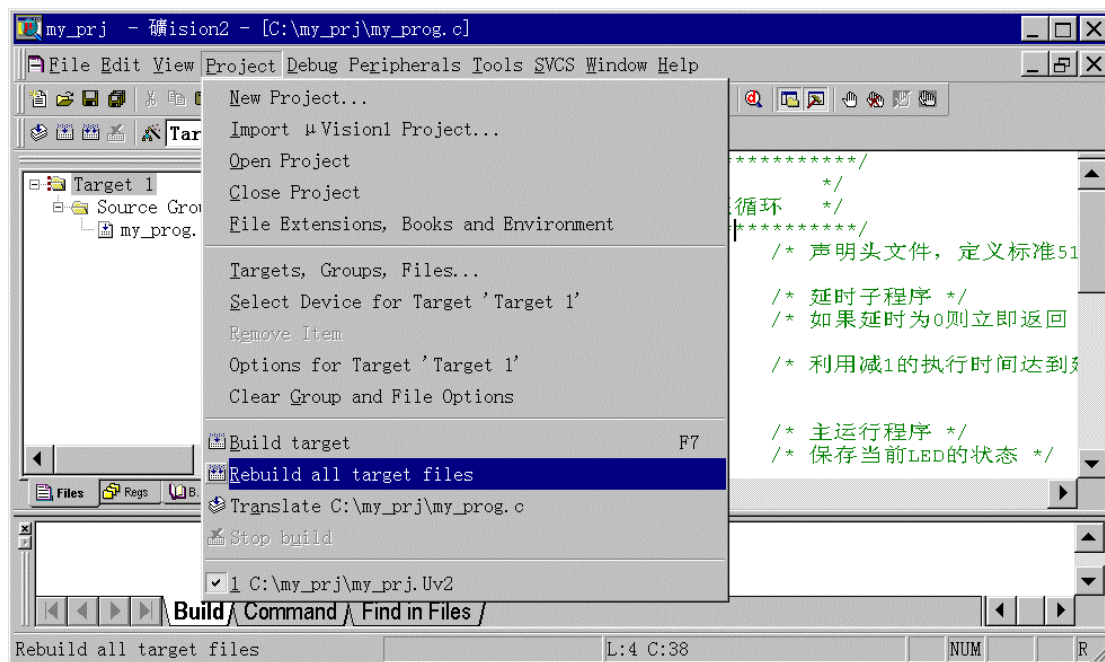


图 6.7.1 请求编译/连接工程

如果用户程序和工程设置没有错误，编译和连接将能顺利完成。操作信息在信息输出窗口提示用户。

程序的排错

我们现在给出的程序是经过验证没有错误的，但是用户在开始编写程序时容易出现错误。现在我们把 my_prj.c 人为的改错一下，LedBuff 写成 LLedBuff:

```
for(;;)                                /* 连续运转 */
{ LLedBuff = 0X01; P1 = LedBuff;        /* 点亮 P10 脚接的 LED */
...

```

现在我们再次进行编译和连接，编译/连接后的结果见图 6.7.2。

编译出现错误！μVision2 有详细的错误提示，使用户容易理解错误的原因。这次错误原因是：

MY_PROG.C (17): error C202: 'LLedBuff': undefined identifier

在文件 MY_PROG.C 的 17 行,出现 C202 类型的错误: LLedBuff 没有定义。μVision2 中有错误定位功能，在输出窗口用鼠标双击错误提示行，MY_PROG.C 文件中的错误所在行被明显的作了标记，并以当前文件的方式显示出来。

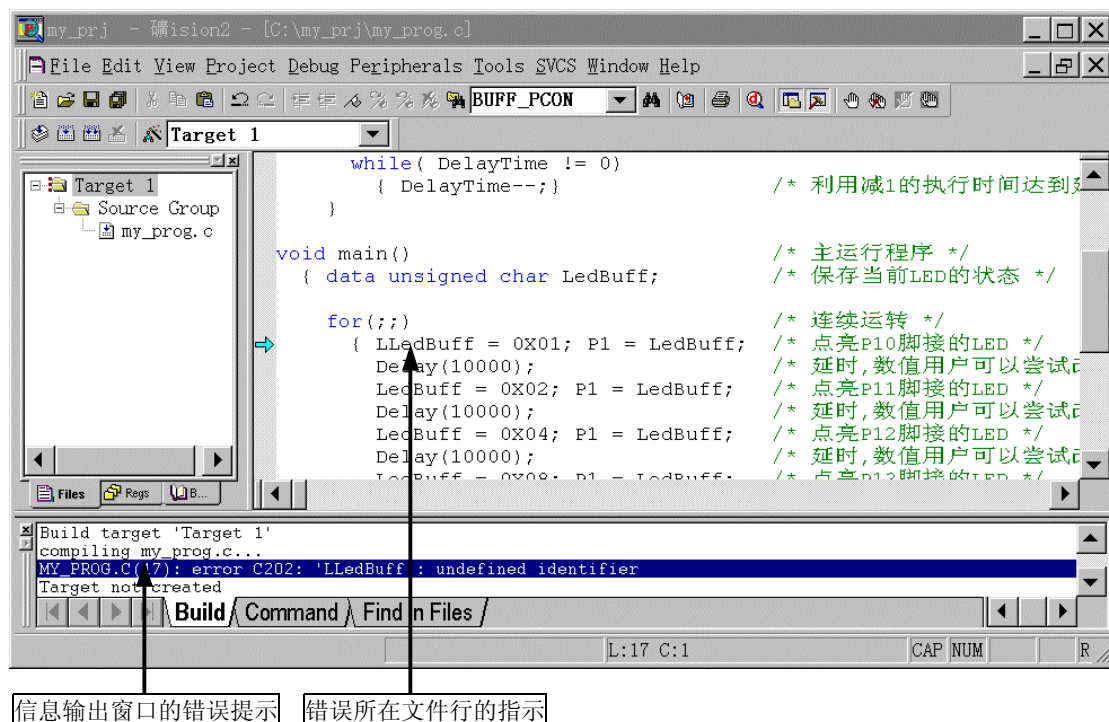


图 6.7.2 编译后的错误定位

程序的重新编译/连接

经过排错后，要重新进行编译和连接，直到编译和连接成功。

6.8 进入硬件仿真前的准备工作

在进入硬件仿真前，还需要将 TKS 系列仿真器准备好，用户需要以下步骤：

- 将 TKS 系列仿真器安放在稳定，安全，平整，操作方便的桌面上。
- 用仿真电缆把仿真头组件连接到仿真器上。
- 用通讯电缆将仿真器同计算机连接起来，并拧紧固定螺丝。
- 将随机提供的电源适配器插到 220V 交流插座中，输出插头接入到 TKS 系列仿真器。
- 如果用户需要接入目标板，在 TKS 自检完成后，将仿真头插入用户目标板插座，然后开启用户目标板电源。

在用户准备仿真器的操作中，应该绝对注意以下事项：

- 要使用随机提供的仿真电缆/仿真头/电源等附件，不允许使用替代品。
- 避免仿真器工作在强烈震动，高温度，高湿度或其它恶劣的环境中。
- 对于允许使用外部电源供电的仿真器，外部输入电源绝对不允许超过直流 5.5V。
- 除非在 TKS 生产厂商的专业人员指导下，用户不要打开 TKS 系列仿真器外壳。
- 在将仿真头插入用户目标板前，应仔细检查用户目标板存在的问题，例如管脚之间短路，管脚同电源或地之间的短路等，避免造成仿真器损坏。

6.9 进入硬件仿真

在图 6.6.3 Debug 设置中点击**硬件仿真选择**，选择硬件仿真。点击菜单 Debug->Start/Stop Debug Session，进入 μ Vision2 的硬件仿真环境。

如果出现通讯失败将无法进入硬件仿真，出现这种现象的原因一般是：

- 仿真器同 PC 的通讯电缆连接不良。
- 仿真器没有电源。
- 错误的选择了硬件驱动程序。
- 串口选择错误，或者该串口已经被其它设备占用。
- 用户串口已经损坏。

排除故障后，重新进入。

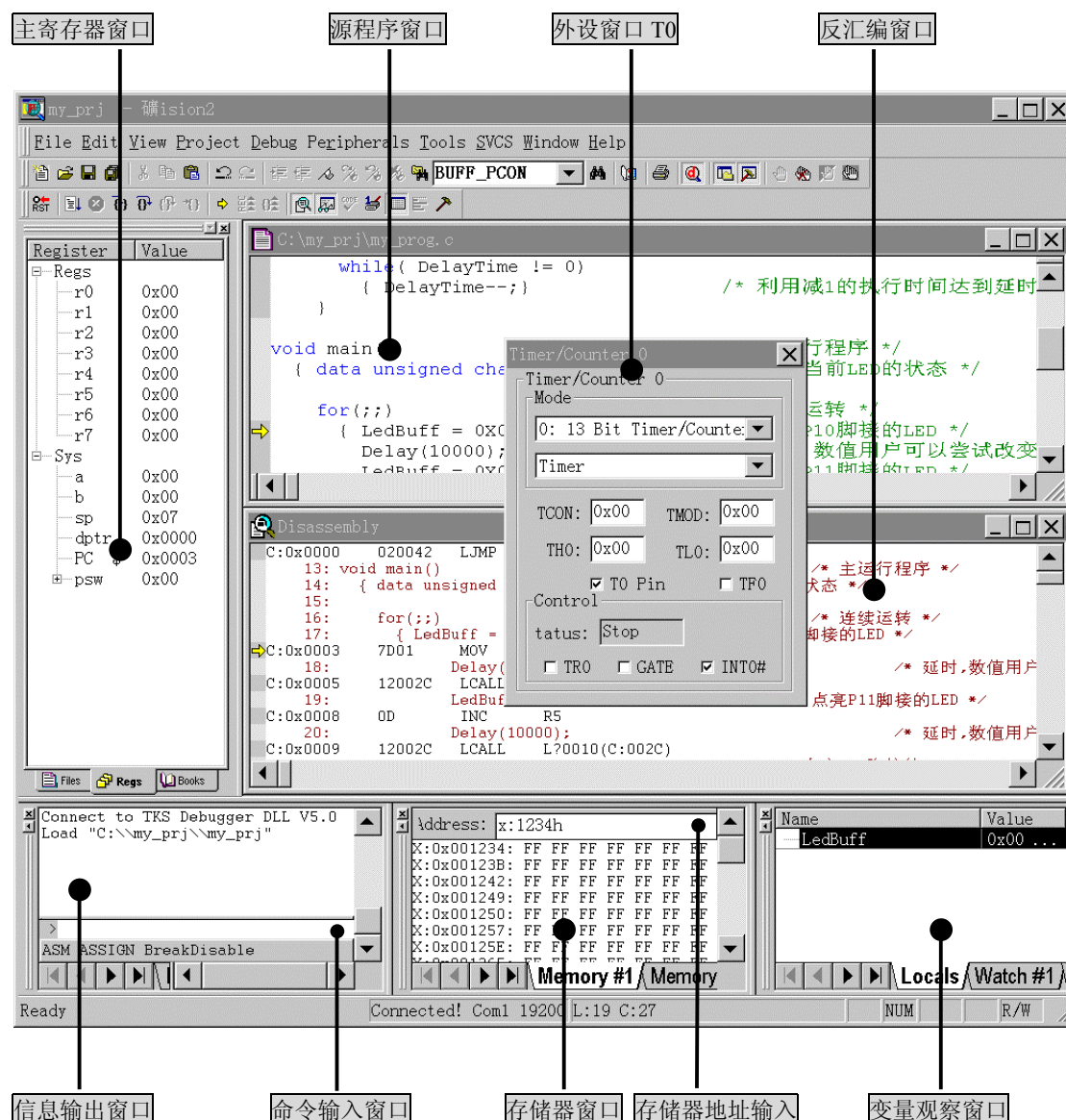


图 6.9.1 硬件仿真环境

进入仿真环境后，在 c 源程序窗口运行指针处于程序行：

```
{ LedBuff= 0X01; P1 = LedBuff; /* 点亮 P10 脚接的 LED */
```

这是由于我们在章节 **5.4.1 Debug 设置** **启动运行选择**中选择了 **Go till main**，而该程序行正是 main()函数下的第一个有效的执行程序行。如果该参数没有选择，程序会停在 0X0000 位置，也就是 80C51 复位后的状态。

μVision2 的窗口

主寄存器窗口	：显示（修改）主寄存器内容，例如 A，B，SP 等特殊功能寄存器。该窗口可以切换成工程文件管理窗口或资料窗口。
源程序窗口	：显示用户编写的源程序，μVision2 支持源程序调试。
外设窗口 T0	：显示或修改外设数据，例如 T0，串口，中断等内容。用户还可以打开其它外设。
反汇编窗口	：显示或修改最终形成的汇编代码形式。
信息输出窗口	：μVision2 给用户的提示信息输出。
命令输入窗口	：用户可以输入控制命令，包括显示资源，设置断点，单步运行等。
存储器窗口	：批量显示或修改 80C51 存储器资源，例如程序存储器，数据存储器等。
存储器地址输入	：规定存储器窗口显示的存储器种类和地址范围。
变量观察窗口	：显示或修改当前工程中定义的所有变量。

μVision2 窗口的启动或关闭

在μVision2 中，任何窗口都可以显示或关闭。对于已经显示的窗口，用户可以用鼠标点击该窗口上的 **Windows 标准关闭图**来关闭该窗口。对于没有显示的窗口，用户可以点击菜单 **View**，里面的不同选择用于开启或关闭不同的显示内容：

- **Project Windows** 开启或关闭工程管理窗口（主寄存器窗口，资料窗口）；
- **Output Windows** 开启或关闭信息输出窗口；
- **Disassembly Windows** 开启或关闭反汇编窗口；
- **Memory Windows** 开启或关闭存储器窗口；
- **Watch and Call Stack Windows** 开启或关闭变量观察窗口。

μVision2 外设窗口的启动或关闭：

在μVision2 中，还可以开启一些外设窗口，外设窗口的内容与您选择的目标器件有关系。例如，您如果选择 89C51，外设中就没有 T2 这个选项；只有选择了 89C52，才可能出现 T2 外设选项。外设窗口只有在进入仿真环境中才有可能打开或关闭。

在仿真环境中，点击菜单 **Peripherals** 就可以看到可以使用的外设或外设组。点击对应的外设就可以打开或关闭该外设窗口。

外设窗口以简单的形式告诉用户现在外设的状态。例如在图 6.15.1 中的**外设窗口 T0** 中，可以很容易地看出现在 T0 处于 13 位定时器状态，还没有启动。

μVision2 中用户可以完成的操作：

- 查看内部所有的资源。对于 80C51 内部的所有资源，在μVision2 中都可以直接查看。

- 内部可以直接寻址的数据存储器 (data)。
 - 内部只能间接寻址的数据存储器 (idata)。
 - 特殊功能寄存器 (SFR, 名称必须定义)。
 - 外部数据存储器 (xdata)。
 - 程序代码空间 (code)。
- 运行用户程序, 包括:
 - Step into 单步跟踪。
 - Step over 单步运行。
 - Run till cursor 运行到光标处。
 - Run 全速运行。
 - 设置运行断点。

6.10 运行程序

在µVision2 中, 有以下的几种程序运行方式:

- (1) **单步跟踪 Step into** : 是尽最大的可能跟踪当前程序的最小运行单位。

在 C 语言下的单步跟踪

单步跟踪在 c 语言环境调试下最小的运行单位是一条 c 语句, 因此单步跟踪 每次最少要运行一条 c 语句。如果一条 c 语句只对应一条汇编指令, 则单步跟踪 一次可以运行 c 语句对应一条汇编指令; 如果一条 c 语句对应多条汇编指令, 则一次单步跟踪 要运行完对应的所有汇编指令。

单步跟踪可以使用 **F11** 快捷键来启动, 也可以使用菜单 Debug->Step, 另外按动 Step 图标  可以启动单步跟踪, 为了以后的方便我们一律使用 F11。

现在我们开始运行一下我们的范例程序 my_prj, 为了在 c 语言下调试, 您必须点击 my_prog.c 的窗口, 把它变为当前的焦点 Windows 窗口。如果您的程序运行光标指在程序行:

➡ { LedBuff= 0X01; P1 = LedBuff; /* 点亮 P10 脚接的 LED */

说明它是进入仿真后的状态; 如果因为您的操作而发生了变化, 退出仿真环境后重新进入。点击菜单 Project->Start/Stop Debug Session 可以从仿真环境中退出。

按动 F11, 我们可以看到黄色的程序运行光标指在了下一个程序语句行:

➡ Delay(10000); /* 延时,数值用户可以尝试改变 */

通过 F11 我们完成了一个程序行的执行。现在我们要执行的是一个函数 Delay()的调用, 单步跟踪会如何执行呢? 按动 F11, 我们看到程序运行光标跳到了 Delay()函数内部的程序行:

➡ { if(DelayTime == 0)return; /* 如果延时为 0 则立即返回 */

出现这种现象的原因是因为**单步跟踪**要跟踪可能最小的程序行。

在汇编语言下的单步跟踪

现在我们退出仿真环境, 再在汇编下运行程序。重新进入仿真环境后, 我们把反汇编窗口作为当前窗口。如果您无法找到反汇编窗口, 可以点击菜单 Windows 观察是否有 Disassemble 窗口。如果有, 点击它使之成为当前窗口。如果没有, 说明汇编窗口没有打开。点击菜单 View->Disassemble, 打开反汇编窗口。在反汇编窗口中, 我们看到程序指针指在:

➡ C:0X0003 750801 MOV 0X08, #0X0 // 对应的 c 程序为 LedBuff= 0X01

按动 F11 后，程序指针应该指在：

➡ C:0X0006 850890 MOV P1(0X90), 0X08 // 对应的 c 程序为 P1 = LedBuff

到现在为止,按动 F11 每次都只运行一句汇编指令.现在我们连续按动 F11,当程序指针指在:

➡ C:0X000D 12006E LCALL Delay(C:0066)

现在程序准备调用子程序 Delay，按动 F11 执行单步跟踪操作，可以看到程序指针指在 C: 0X006E 的位置，跟踪进入了 Delay 子程序，也就是说程序跟踪每一个汇编指令的执行。与 Step into 不同，在下面的 Step over 的操作中，我们可以看到，在汇编窗口下执行单步运行后将整个 LCALL 指令执行完毕，并不跟踪到子程序内部。

结论：单步跟踪是尽最大的可能跟踪程序的运行。在 c 语言环境下调试，可以跟踪到每一个 c 程序行；在汇编语言调试下，可以跟踪到每一个汇编指令的执行。



图 6.10.1 运行程序的反汇编窗口

(2) 单步运行 Step over：是尽最大的可能执行完当前的程序行。

在 C 语言下的单步运行

单步运行在 c 语言环境调试下最小的运行单位是一条 c 语句，因此单步跟踪 每次最少要运行一个 c 语句。如果一条 c 语句只对应一条汇编指令，则单步跟踪 一次可以运行 c 语句对应一条汇编指令；如果一条 c 语句对应多条汇编指令，则一次单步跟踪 要运行完对应的所有汇编指令。

单步运行可以使用 **F10** 快捷键来启动，也可以使用菜单 Debug->Step，另外按动 Step over 图标  也可以启动单步跟踪。为了以后的方便我们一律使用 F10。

现在我们开始运行一下我们的范例程序 my_prj，为了在 c 语言下调试，您必须点击 my_prog.c 的窗口，把它变为当前的焦点 Windows 窗口。如果您的程序运行光标指在程序行：

➡ { LedBuff= 0X01; P1 = LedBuff; /* 点亮 P10 脚接的 LED */

说明它是进入仿真后的状态；如果因为您的操作而发生了变化，退出仿真环境后重新进入。点击菜单 Project->Start/Stop Debug Session 可以从仿真环境中退出。

按动 F10, 我们可以看到黄色的程序运行光标指在了下一个程序语句行:

→ Delay(10000); /* 延时,数值用户可以尝试改变 */

通过 F10 我们完成了一个程序行的执行,跟 F11 的执行完全一致。现在我们要执行的是一个函数 Delay() 的调用,单步运行后会如何执行呢?按动 F10,我们看到程序运行光标指在了下一条 c 语句位置:

→ LedBuff = 0X02; P1 = LedBuff; /* 点亮 P11 脚接的 LED */

出现这种现象的原因是因为执行了单步运行,它执行完了 Delay(10000)这条语句,而不是跟踪进入 Delay() 函数内部,跟 F11 的执行不一致。

在汇编语言下的单步运行

现在我们退出仿真环境,再在汇编下运行程序。重新进入仿真环境后,我们把反汇编窗口作为当前窗口。如果您无法找到反汇编窗口,可以点击菜单 Windows 观察是否有 Disassemble 窗口。如果有,点击它使之成为当前窗口。如果没有,说明汇编窗口没有打开。点击菜单 View->Disassemble,打开反汇编窗口。在反汇编窗口中,我们看到程序指针指在:

→ C:0X0003 750801 MOV 0X08, #0X0 // 对应的 c 程序为 LedBuff = 0X01

按动 F10 后,程序指针应该指在:

→ C:0X0006 850890 MOV P1(0X900), 0X08 // 对应的 c 程序为 P1 = LedBuff

到现在为止,按动 F10 每次都只运行一句汇编指令,跟 F11 的执行完全一致。现在我们连续按动 F10,当程序指针指在:

→ C:0X000D 12006E LCALL Delay(C:0006)

现在程序准备调用子程序 Delay,按动 F10 执行单步运行操作,可以看到程序指针指在 C: 0X0010 的位置,也就是说程序执行完毕了 LCALL Delay,而不是跟踪进入 Delay 子程序,跟 F11 的执行不一致。

结论:单步运行是尽最大的可能运行完当前的程序行。

(3) Run till cursor 运行到光标处

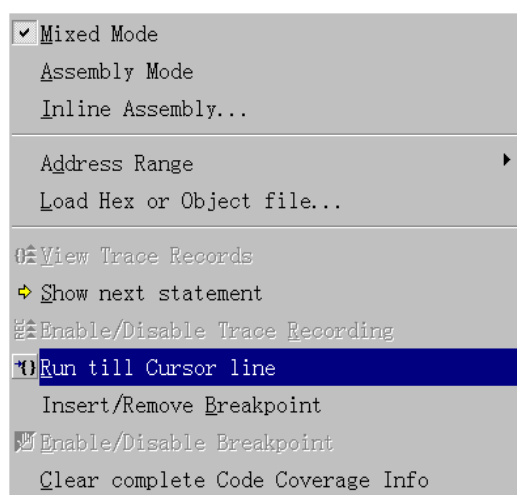


图 6.10.2 Run till cursor 浮动菜单

退出仿真环境后重新进入,把 my_prog 设置为当前的焦点窗口,程序指针指在程序行:

→ { LedBuff = 0X01; P1 = LedBuff; /* 点亮 P10 脚接的 LED */

如果您想程序运行到程序行:

LedBuff = 0X20; P1 = LedBuff; /* 点亮 P15 脚接的 LED */

用鼠标的左键点击程序行：

```
LedBuff = 0X20; P1 = LedBuff; /* 点亮 P15 脚接的 LED */
```

看到光标停留在该行后，按动鼠标的右键出现浮动菜单，选择 **Run to Cursor line**。运行停止后，我们会发现程序运行光标已经停留在程序行：

➡ `LedBuff = 0X20; P1 = LedBuff; /* 点亮 P15 脚接的 LED */`

(4) 全速运行

TKS 系列仿真器在全速运行期间将实时仿真 MCU 的特性。全速运行的启动有三种方法：

- 按动 F5 按键。
- 点击图标 。
- 点击菜单 Debug->Go。

当 μ Vision2 处于全速运行期间， μ Vision2 不允许任何资源的查看，也不接受其它的命令。

如果用户想终止程序的运行，可以：

- 点击菜单 Debug->Stop Running。
- 点击图标 。

在我们的范例程序 my_prj 中，如果仿真器接到用户板，且用户板上的 P1 口接有 LED 发光二极管，进入全速运行后会看到 LED 交替连续的闪亮。

6.11 程序复位

如果用户想重新开始运行用户程序，可以对仿真器的用户程序进行复位。由于 80C51 芯片复位后程序计数器将从 0000H 重新开始，因此 TKS 系列仿真器复位后程序计数器（PC 指针）将复位成 0000H。另外，一些内部特殊功能寄存器在复位期间也将重新赋值，例如 A 将变为 00H，DPTR 变为 0000H，SP 变为 07H，I/O 口变为 0FFH。

程序的复位有以下几种办法实现：

- 点击图标 。
- 点击菜单 Peripherals->Reset CPU。
- 在命令输入窗口输入 “RESET”。

6.12 断点操作

当我们需要程序全速运行到某个程序位置停止时，我们就可以使用断点。

断点的设置/取消

在 μ Vision2 中，在 C 源程序窗口/汇编源程序窗口/反汇编窗口都可以在任何有效位置设置断点，断点的设置/取消操作也非常简单。在三中窗口中，如果您想在某一行设置断点，只需要用鼠标左键双击该行，就可以在该行左边框发现红色的断点标志。断点的取消也是同样的操作，如果该行已经设置为断点行，双

击该行将取消该行的断点。

带断点的全速运行

在章节 6.10 (3) 运行到光标处操作中，使用的是 Run till cursor（运行到光标处）功能，现在我们使用断点的方法。

- 退出仿真环境后重新进入。
- 双击行 `LedBuff = 0X20; P1 = LedBuff; /* 点亮 P15 脚接的 LED */` 设置断点。
- 按动 F5 启动全速运行。
- 观察程序运行光标是否停留在断点设置行。

断点的管理

如果用户设置了很多的断点，可能存在着断点管理的问题。例如，我想取消全部的断点全速运行，如果逐个的取消将是非常烦琐的事情。在 μ Vision2 中有断点管理器。点击 Debug->Breaks, 出现断点管理窗口：

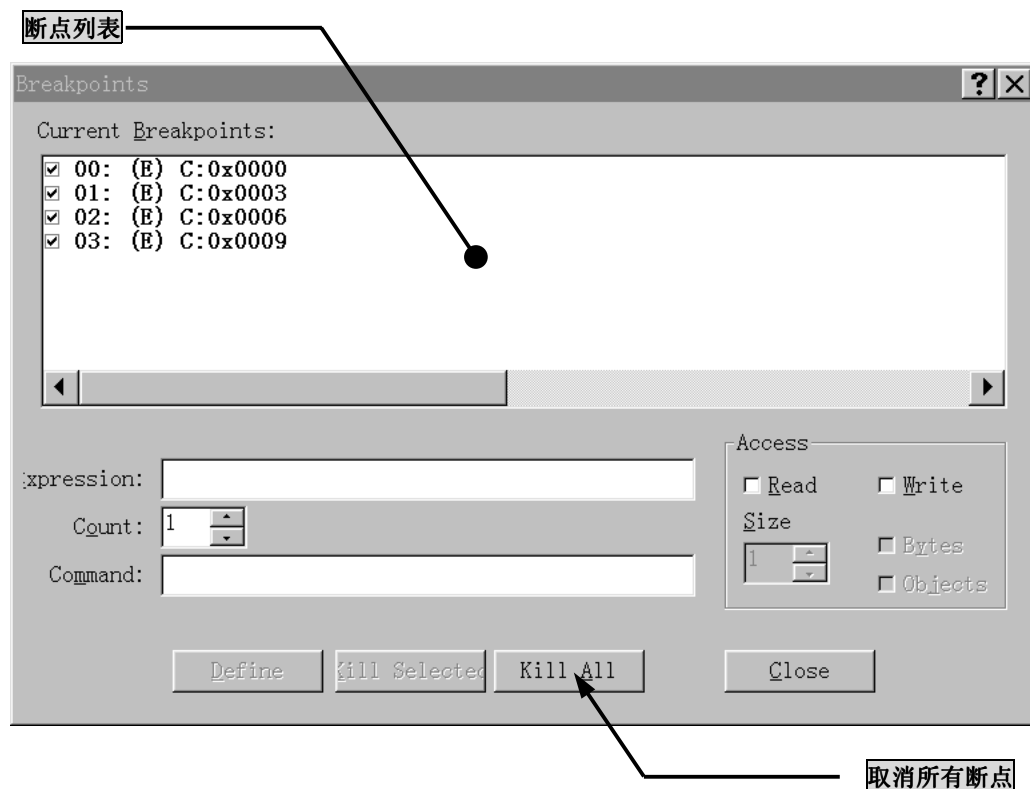


图 6.12.1 断点管理

取消所有断点 可以取消所有已经设置的断点。

6.13 空间资源的查看和修改

在μVision2 的仿真环境中，标准 80C51 的所有有效空间资源都可以查看和修改。μVision2 把空间资源分成几种类型加以管理：

6.13.1 内部可直接寻址 RAM（或 SFR）（类型 data，简称 d）

在标准 80C51 中，可直接寻址空间为 0-0X7F 范围内的 RAM 和 0X80-0XFF 的 SFR（特殊功能寄存器）。在μVision2 中把它们组合成空间连续的可直接寻址的 data 空间。data 空间有下列几种查看和修改方法：

● 存储器窗口（Memory Windows）

存储器窗口的打开

点击 View->Memory Windows 可以打开存储器窗口(见图 6.13.1).如果窗口已打开,该窗口则会关闭。

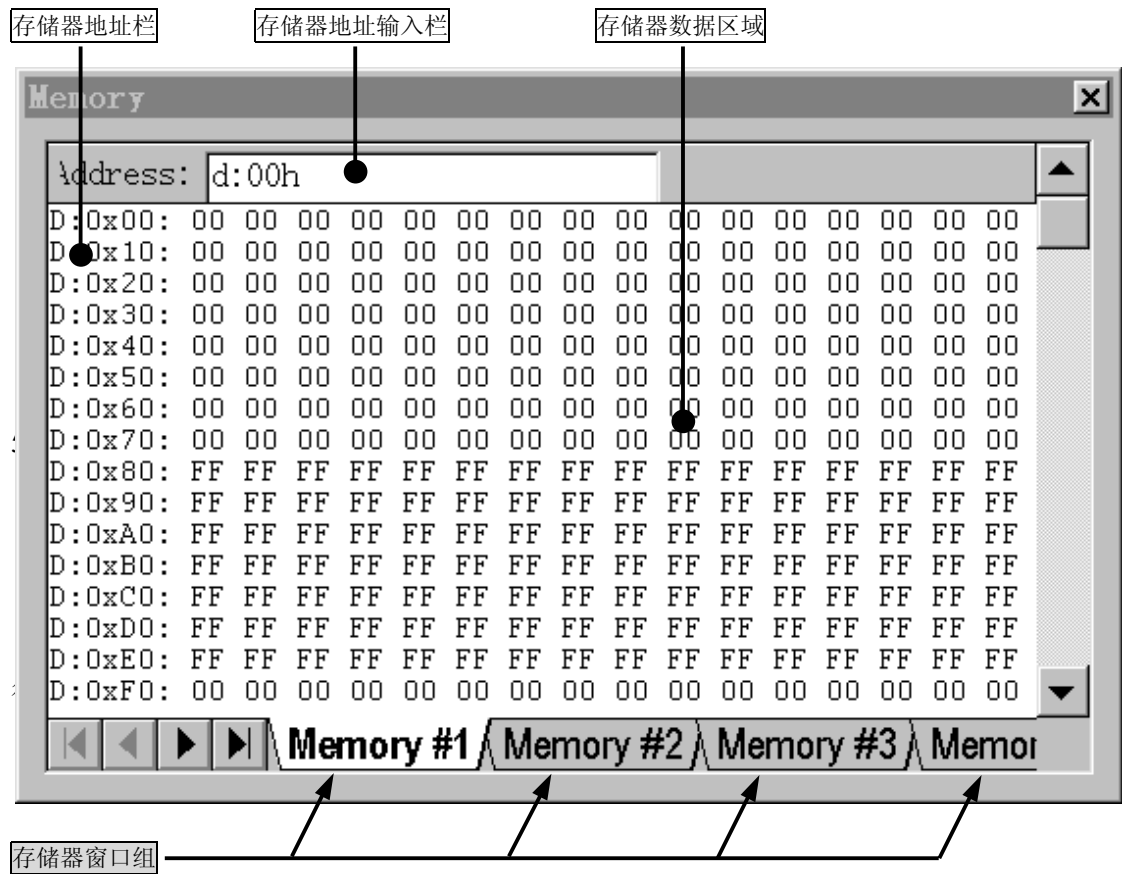


图 6.13.1 存储器窗口的显示

打开存储器窗口后，可以看到：

存储器地址输入栏：用于输入空间类型和起始地址，在上图中，d 表示 data 区域，0 表示显示 起始地址。

存储器地址栏：显示每一行的起始地址，便于观察和修改。data 区域的最大地址为 0XFF。

存储器数据区域：数据显示区域，显示格式可以改变。

存储器窗口组： 分成独立的 4 个组，每个组可以单独定义空间类型和起始地址，点击组图标可以切换。

存储器窗口中数据的修改

存储器窗口 中的数据可以方便的修改。修改的方法是：把鼠标移动到该数据的显示位置，按动鼠标的右键，出现浮动菜单（见图 6.13.2）。

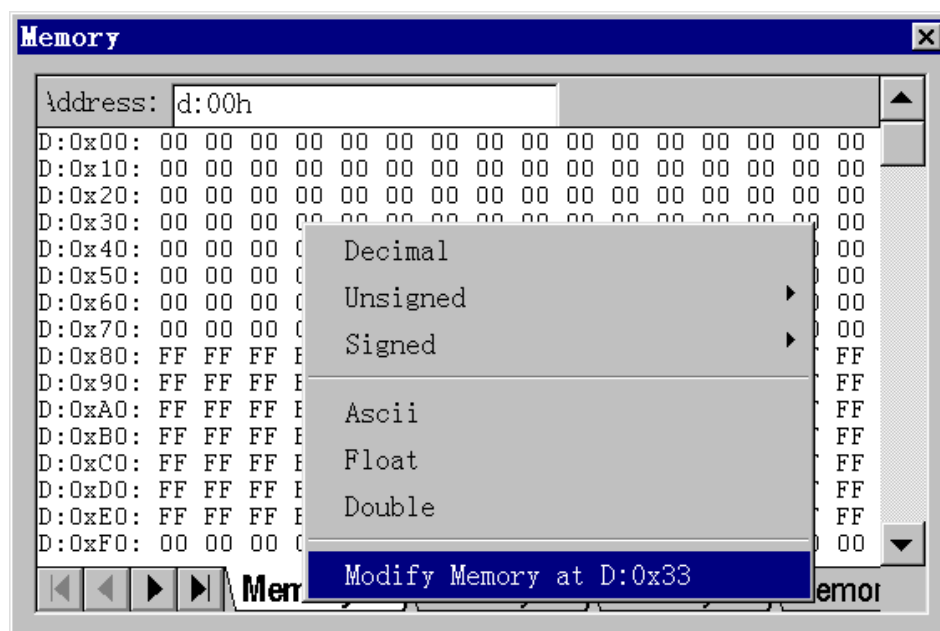


图 6.13.2 存储器窗口的修改

选择 **Modify Memory at D: 0X33**，表示要改动 data 区域 0X33 地址的数据内容。选择后系统会出现输入栏，输入新的数值后点击 **OK** 返回。需要注意的是，在使用 TKS 系列仿真器时， μ Vision2 提供了改动的手段，但是改动并不一定能完成。例如，如果您改动 0xFF 位置的内容，但是 80C51 在这个位置没有可操作的单元。改动并不能完成，操作完成后地址 0xFF 单元显示的不一定是您想改动的数据。

● 使用命令行

在**命令输入窗口** 通过输入符合格式的命令，可以查看和修改 data 空间的数据。

通过命令行对数据的查看

例如，我们想查看 data 空间从 0x23 到 0x56 区间的内容，可以在**命令输入窗口**输入：

“d d:0x23,56H”，回车即可完成。d 表示 data 空间，0x23 表示起始地址，56H 表示结束地址。注意两种 16 进制的表示方法在这里都可以接受。 μ Vision2 把输出结果在**信息输出窗口**中显示出来（见图 6.13.3）。

通过命令行对数据的修改

例如，我们想把 d: 0X23 开始的 3 个数据修改成 0x01 0x02 0x03，可以在**命令输入窗口**输入：

E char d:0x23=0x01,0x02,0x03 回车即可完成。char 表示修改数据类型，d 表示 data 空间，“=” 后面是输入的修改数据。

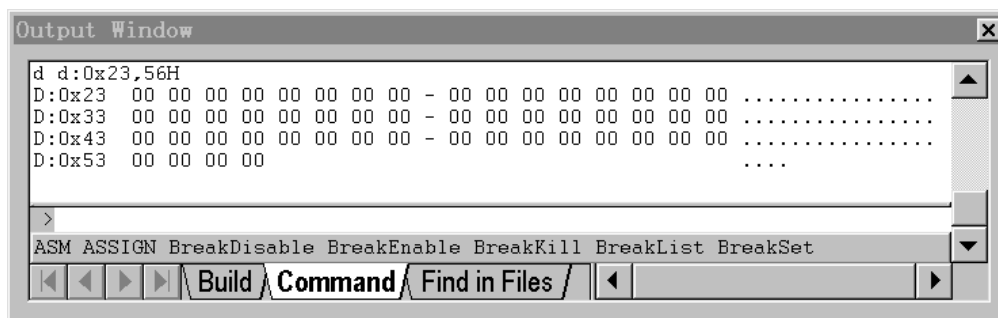


图 6.13.3 命令窗口的数据显示

6.13.2 内部可间接寻址 RAM（类型 idata，简称 i）

在标准 51 中，可间接寻址空间为 0-0xFF 范围内的 RAM。其中，地址范围 0x00-0x7F 的 RAM 既可间接寻址，也可直接寻址；地址范围 0x80-0xFF 的 RAM 只能间接寻址。在 μ Vision2 中把它们组合成空间连续的可间接寻址的 idata 空间。idata 空间有下列几种查看和修改方法：

- 存储器窗口（Memory Windows）

内部可间接寻址 RAM（idata）在存储器窗口（Memory Windows）中的操作方法同内部可直接寻址 RAM（或 SFR）完全相同，只是需要用户在存储器地址输入栏输入的空间信息要变为 i。例如，要在存储器窗口显示起始地址为 0x65 的 idata 数据，只要在存储器地址输入栏输入：“i: 0x65”。

- 使用命令行

其它详细的操作方法请用户参考 5.12.1 章节的内容，只是把表示可直接寻址空间的“d”换成表示可间接寻址空间的“i”。

6.13.3 外部数据空间 XRAM（类型 xdata，简称 x）

在标准 80C51 中，外部可间接寻址 64K 地址范围的数据存储器，其读取的指令是 MOVX，间接寻址寄存器是 DPTR 或 Ri。在 μ Vision2 中把它们组合成空间连续的可间接寻址的 xdata 空间。xdata 空间数据有下列几种查看和修改方法。

- 存储器窗口（Memory Windows）

- 使用命令行

其它详细的操作方法请用户参考章节 6.13.1 的内容，只是把表示可直接寻址空间的“d”换成表示外部间接寻址数据空间的“x”。

6.13.4 程序空间 code（类型 code，简称 c）

在标准 80C51 中，程序空间有 64K 的地址范围。程序存储器的数据按用途可分为程序代码（用于程序执行），程序数据（程序使用的固定参数）。在 μ Vision2 中把它们组合成空间连续的可间接寻址的 code 空间。code 空间数据有下列几种查看和修改方法：

- 存储器窗口（Memory Windows）
- 使用命令行

其它详细的操作方法请用户参考 6.13.1 章节的内容，只是把表示可直接寻址空间的“d”换成表示外部间接寻址数据空间的“c”。

6.14 变量的查看和修改

在我们的工程 my_prj 中,我们定义了一些变量,例如 `data unsigned char LedBuff`，告诉编译器这是一个无符号的字节变量，要求放在 data 空间内。至于放在 data 空间的位置，用户一般不必关心，只是关心它的数值就可以了。在 μ Vision2 中可以直接观察和修改变量，方法为以下几种：

6.14.1 使用观察窗口（Watch Windows）

点击 View->Memory Windows 可以打开观察窗口（见图 6.14.1），如果窗口已经打开,该窗口则会关闭。

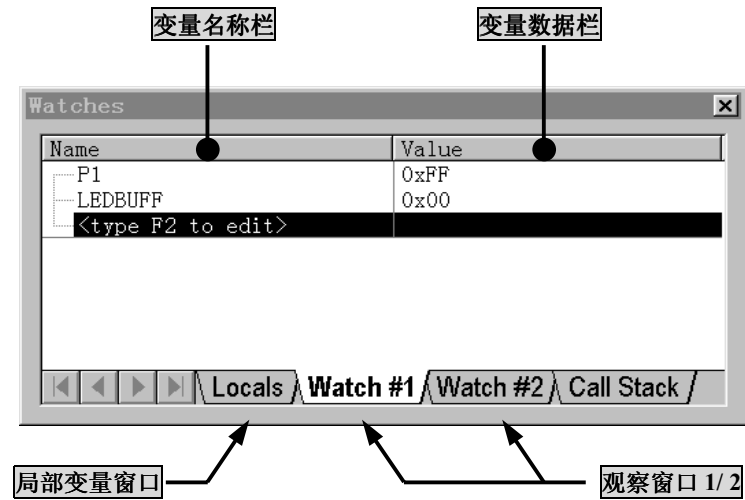


图 6.14.1 观察（Watches）窗口的数据显示

观察窗口 1/2

观察窗口分#1 和#2，用户可以根据分类把变量添加到#1 或#2 中。

变量名称栏

用于输入变量的名称。输入方法是：用鼠标左键点击准备添加行（选择该行）的变量名称栏，然后按 F2 键，出现文本输入栏后输入变量的名称，确认正确后按回车键。输入的变量名称必须是文件中已经定义

的。在图 6.14.1 中，LedBuff 是我们自己定义的，而 P1 是头文件 REG52.H 中定义的。

变量数据栏

用于显示变量的数值。用户可以修改变量的数值，方法是：用鼠标左键点击该行的变量数据栏，然后按 F2 键，出现文本输入栏后输入修改的数据，确认正确后按回车键。

局部变量窗口

自动显示当前正在使用的局部变量，不需要用户自己添加。

6.14.2 使用命令行

使用命令行可以全局变量和当前使用的全局变量进行读写。

- 全局变量的解释

全局变量是在所有函数外部定义的变量，所有的函数都可以对它进行操作。在工程 my_prj 中,P1 就是全局变量,它定义在头文件 REG52.H 中。现在我们尝试使用命令行对 P1 进行读写操作。

- 全局变量的操作

在进入硬件仿真的任何时刻，在命令输入窗口输入 P1，按回车键，在信息输出窗口则会输出从 TKS 系列仿真器中读回的 P1 数值，例如 0x02。如果用户需要修改把 P1 的数值从 0x02 修改成 0x03，则可在命令输入窗口输入 P1=0x03，按回车键即可。

- 局部变量的解释

局部变量是在一个函数内部定义的变量，只有该函数可以对它进行操作。在工程 my_prj 中,DelayTime 就是一个局部变量,它定义在函数 Delay()中,只有在该函数中才能进行操作。

- 局部变量的无效的操作

现在我们尝试使用命令行对 DelayTime 进行读写操作。在退出硬件仿真环境后重新进入，在 my_prog.c 窗口中，程序运行指针指在程序行：

```
{ LedBuff = 0X01; P1 = LedBuff; /* 点亮 P10 脚接的 LED */
```

这时，我们在命令输入窗口输入 “DelayTime”，按回车键，要求读取数据。由于 DelayTime 是局部变量，未进入 Delay()函数此变量无效，所以在信息输出窗口出现 “error 34: undefined identifier”，表示该函数还未定义。

- 局部变量的有效操作

现在我们准备进入函数 Delay()中，连续按动单步跟踪按键 F11，进入 Delay()函数内部。这时我们在命令输入窗口输入 “DelayTime”，按回车键，可以发现信息输出窗口出现 “0x2710”，说明正确读出了

DelayTime 的数值。

现在我们尝试修改 DelayTime，在命令输入窗口输入“DelayTime = 0x1234”，按回车键。现在重新读取 DelayTime 进行检查，在命令输入窗口输入“DelayTime”，按回车键，信息输出窗口出现“0x2710”，说明前面成功地进行了写操作。

6.15 在 Keil 调试中的使用技巧

- 使用外设菜单 Peripherals 增加调试效率

在进入硬件仿真环境后，根据您为工程选择的器件型号，在菜单 Peripherals 中会有一些该型号内部的外设名，点击一个外设名后将会弹出一个浮动窗口。现在我们点击菜单 Peripherals->Timer->Timer 0，出现 Timer0 的浮动窗口（见图 6.15.1）。

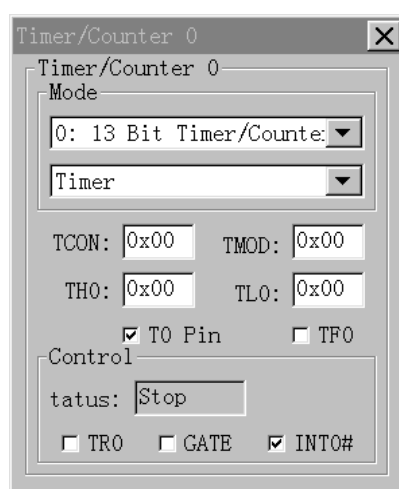


图 6.15.1 T0 的外设窗口

从图 6.15.1 可以看出，相关 T0 的大部分资源在它的外设窗口中都可以直观显示并能进行修改。

- 现在我们在退出仿真环境后重新进入，点击菜单 Windows，打开后观察是否有 Disassemble。如果有，点击它使之成为当前窗口；如果没有，说明汇编窗口没有打开，点击菜单 View->Disassembly Windows 打开汇编窗口。点击菜单 Peripherals->Reset CPU,复位,现在可以看到程序指针指在 C:0000H 位置。

- 现在我们点击图 6.15.1 中的 TR0，启动 T0。然后我们在反汇编窗口进行单步运行，按动 F11，图中的 TL0 将发生变化。根据指令的机器周期数，TL0 一次增加 1 或 2 或 4 个数。

- 再次点击图中的 TR0，关闭 T0，以后单步运行 TL0 将不会发生变化。

从上面的操作我们可以看出，通过图形化的外设菜单，我们可以轻松的实现某些功能的调试。

- 使用 Show next state 找到当前光标

在调试过程中，我们可能在查看中发现当前的程序运行指针不再我们的视野中，也可能在另外的一个窗口中。我们可以一点一点的寻找，但有时候这样做非常的麻烦，尤其在工程比较庞大，文件比较多的时候。我们可以使用 Next Statement 快速找到当前的程序运行指针。

- 点击菜单 Debug->Show Next Statement。

- 点击工具条上的图标。

在完成上述的任何一条操作后，我们可以看到程序指针所在的汇编和源程序窗口变为最上面的两个窗口，并且程序指针指在了窗口内的位置。

● 将变量快速加到观察窗口中

在章节 5.12.1 中讲到，在观察窗口中通过输入变量名可以在 Watch 窗口观察或修改该变量。但是如果变量名比较长或要添加的变量比较多，这样操作也比较烦琐。通过下面提供的操作，可以直观快速的将一个变量添加到 Watch 窗口中。

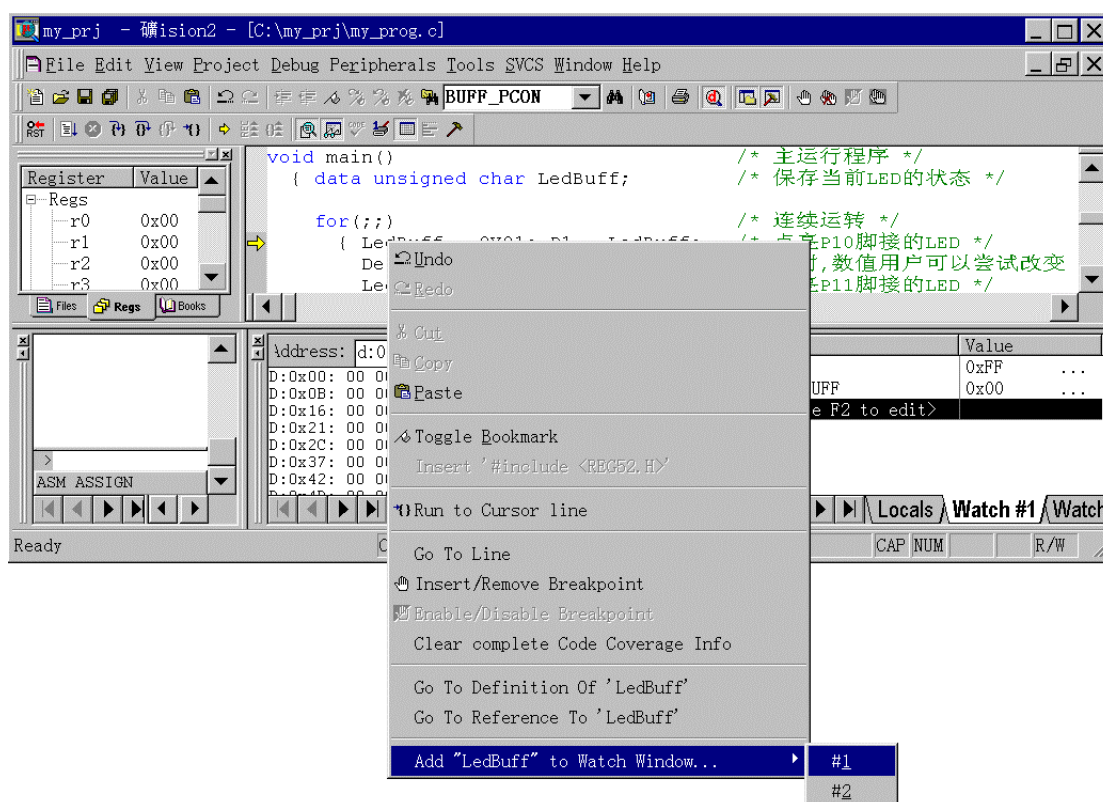


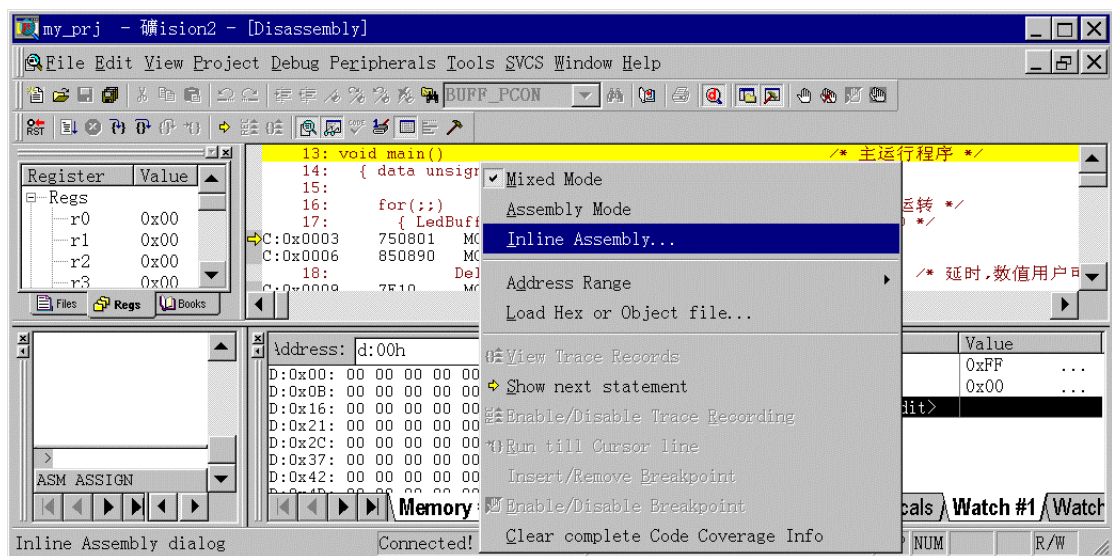
图 6.15.2 添加变量到 watch 窗口

在硬件仿真环境中，将鼠标移动到一个变量名的上面，点击鼠标右键，出现浮动菜单。如果变量是有效的，在菜单中将会有 Add “变量名” to Watch Windows 的选项，子层还会有 Watch 1 或 2 的选项。点击后该变量就会加到对应的 Watch 窗口去。

● 在调试中改变程序的汇编代码

在调试已经编译好的程序时，可以在硬件仿真环境中直接修改程序代码，这样就不用返回到编译环境中重新编译。在大程序的调试中，这种技巧很有用。μVision2 支持在调试中的在线汇编。

在汇编窗口中，如果想改动一条汇编指令，可用鼠标左键点击该行，然后点击鼠标右键，出现浮动菜单后选择 Inline Assembly。在μVision2 弹出输入窗口后，输入汇编指令，例如 LJMP 0080H，回车。



6.15.3 在线汇编界面

6.16 退出仿真

步骤 1: 退出μVision2 仿真环境

如果用户想退出μVision2 的硬件仿真环境，可以使用下列方法退出。

- 点击  图标；
- 点击菜单 Debug -> Start/Stop Debug Session;
- 在命令输入窗口输入“exit”。

步骤 2: 关闭 TKS 系列仿真器

如果用户想停止使用 TKS 系列仿真器，按照下列顺序操作：

- 关闭用户目标板的电源；
- 拔下 TKS 系列仿真器的电源；
- 从用户目标板上拔下仿真头；
- 如果暂时不再进行硬件仿真，卸下 TKS 同计算机的通讯电缆；
- 将仿真器主机，仿真电缆，仿真头，电源放到适当的地方保存。

请按照上面的步骤顺序退出仿真，这样能保证本次操作的一些设置完全得到保存并在下次得到完全的使用。不要在μVision2 退出仿真环境以前切断仿真器的电源，这样会影响 TKS 系列仿真器同计算机通讯参数的记录，不建议用户这样操作。