C51系列微控制器的开发工具

uVision2入门教程

使用指南

Keil Software 声明

本文档所述信息不属于我公司的承诺范围其内容的变化也不会另行通知本文档所述软件的出售必须经过授权或签订特别协议本文档所述软件的使用必须遵循协议约定在协议约定以外的任何媒体上复制本软件将触犯法律购买者可以备份为目的而做一份拷贝在未经书面许可之前本手册的任何一部分都不允许为了购买者个人使用以外的目的而以任何形式和任何手段(电子的机械的)进行复制或传播

注意
本手册假定你已经熟悉微软操作系统和8051系列产品的硬件和指令集

我们尽全力去做来保证这本手册的正确从而保证我们个人公司和在此提及的商标的形象

前言

这本手册是Keil Software 公司关于8051系列MCU的开发工具的介绍它向新用户和有兴趣的读者介绍本公司的产品这本使用指南包含下列各章

第 1 章 简介概述并描述了Keil Software 为8051系列MCU提供的不同产品

第 2 章 安装描述了该如何安装软件以及如何设置工具的操作环境

第 3 章 开发工具描述了集成有调试器C编译器汇编器的uVision2 IDE的主要特性和用途

第 4 章 建立应用描述该如何建立项目编辑源文件编译并报告语法错误 产生运行代码

第 5 章 测试程序描述了如何使用Vision2 debugger模拟并测试你的整个应用

第 6 章 调试功能讨论了扩展uVision2 debugger功能的各种函数

第 7 章 示例程序提供几个示例程序以说明该如何使用Keil 8051开发工具

第 8 章 实时操作系统讨论了RTX-51 Tiny版和RTX-51 Full版并提供一个示例程序

第 9 章 使用片上外围设备描述了如何使用C51编译器访问片上外围设备本章也包括几个应用注意事项

第 10 章 CPU和程序启动代码描述了如何为你的应用设置8051CPU

第 11 章 使用Monitor-51讨论该如何初始化Monitor并把它安装到你的目标板上

第 12 章 命令参考简单地介绍了Keil 8051开发工具的命令和控制

本文档中使用如下约定:

举例	描述
README.TXT	黑粗体用来表示执行文件数据文件源文件环境变量和你在命令提示行键入的命令 这些文字往往表示你必须按照字面的字符键入如**CLS DIR BL51.EXE**
`Courier`	这种形式的字体用来表示在屏幕或打印机上出现的信息.
Variables	斜体字表示必须由你提供的信息如在语法字符串中的"*projectfile*"表示你必须提供实际的项目名称少数情况下斜体字也用来表示强调
Elements that Repeat	省略号表示一个你可以替换的内容

Omitted code	垂直的省略号用来在源程序列表中表示一段被忽略的程序如 Void main (void) {
	while (1);
	}

[*Optional Items*]	方括号表示命令行或输入域中的可选项如 C51 TEST.C PRINT [(filename)]
{ *opt1* | *opt2* }	包括在大括号中的被'|'分开的文字表示一组选项必须从中选一
Keys	以sans serif字体出现的字符表示键盘上实际的键,如: "Press **Enter** to continue." 中的**Enter**表示键盘上的回车键.
Point	移动鼠标直到光标直到期望的条目上
Click	单击鼠标.
Drag	鼠标拖动操作.
Double-Click	双击鼠标.

目录

第1章 简介

感谢您允许Keil Software为您提供8051系列单片机的软件开发工具利用本工具您可以开发所有8051系列单片机的嵌入式应用

注意

尽管我们在本手册中称它为*8051*开发工具其实它支持所有的由*8051*家族派生而来的类型

Keil Software 的8051开发工具提供以下程序你可以用它们来编译你的C源码汇编你的汇编源程序连接和重定位你的目标文件和库文件创建HEX文件调试你的目标程序从21页开始的第三章开发工具一章中将对每一个程序进行详细描述

Windows应用程序uVision2是一个集成开发环境它把项目管理源代码编辑程序调试等集成到一个功能强大的环境中
C51美国标准优化C交叉编译器从你的C源代码产生可重定位的目标文件
A51宏汇编器从你的8051汇编源代码产生可重定位的目标文件
BL51连接/重定位器组合你的由C51和A51产生的可重定位的目标文件生成绝对目标文件
LIB51库管理器组合你的目标文件生成可以被连接器使用的库文件
OH51目标文件到HEX格式的转换器从绝对目标文件创建Intel HEX 格式的文件
RTX-51实时操作系统简化了复杂和对时间要求敏感的软件项目

在16页的产品一览中将对由这些工具组成的开发套件进行描述它们是为专业开发人员而设计的但所有层次的编程人员都可以用它们来获得8051微控制器的绝大部分应用

手册主题

本手册讨论的主题有

怎样为你的应用选择最好的工具包参照16页的产品一览
怎样在你的系统上安装本软件参照16页安装
本开发工具的特征页
怎样用uVision2 IDE创建一个完整的应用57页
怎样调试程序怎样用uVision2调试器模拟你的目标硬件93页
在C51编译器中该如何访问片上外围设备和8051派生系列产品的特殊功能114页
怎样运行示例程序149页

注意
为了立即开始请参照第二章安装软件然后参照第七章运行示例程序

本文档的最后改动

本软件和手册最后一刻的变化和修改在**RELEASE.TXT**中位于**\KEIL\UV2**和**\KEIL\C51\HLP**文件夹中
花点时间读一下这些文件看看这些变化和修改是否对安装产生影响

测试版工具包和产品工具包

Keil Software把软件分成两种类型测试版和正式版

测试版包括8051工具的测试版本和本用户手册你可以用它们产生目标代码小于2K字节的应用此套件主要是让你测试我们产品的效力并产生小的应用

正式版在16页讨论包括没有限制的8051工具和全套手册(含本手册)正式版套件包含1年的免费技术支持和产品升级升级通过**www.keil.com**提供

用户类型

本手册针对三种用户测试用户新用户有经验的用户

测试用户是那些还没有购买本软件但已经要求使用测试开发包以进一步了解本工具和本工具的性能的用户测试开发包包括有2K字节目标代码限制的工具和几个为8051MCU系列产品而创建的应用即使你是一个测试用户你最好也花点时间阅读本手册它解释了怎样安装本软件为你提供本开发工具的初步信息并介绍了示例程序

新用户是那些第一次购买本开发工具的用户你所购买的软件为你提供最新的开发工具技术手册和示例程序如果你对8051或本工具比较生疏花点时间学习本手册中描述的示例程序它们为新用户和没有经验的用户快速起步提供了一个指南和帮助

有经验的用户是指那些以前已经用过Keil 8051开发工具现在升级到最新版本的用户升级软件产品包含最新的开发工具和示例程序

请求援助

Keil Software 的全体员工专注于为您提供最好的开发工具和文档资料如果你对本手册有建议的话请跟我们联系如果你认为你发现了一个软件上问题请在联系技术支持中心前按下面的步骤做

1阅读与你试图完成的工作或任务相关的章节
2确定你所用的是最新的版本到**www.keil.com**核对升级内容以确定你使用的是最新版本
3分析所发现的问题确定它是汇编器的问题还是编译器连接器库管理器或其他的开发工具的问题
4进一步通过减少你的代码到几行使问题更明确

如果你在经过上述步骤后问题仍然存在请你向我们技术支持中心报告请包含你的产品序列号和版本号我们倾向于你通过E-mail的方式发送如果你通过FAX联系请确定包含我们可以与你联系上的你的名字和电话号码(电话和传真)

请尽可能详细地描述你所遇到的问题你描述的越详细我们就能越快地找到解决办法如果你能用仅仅一页的代码描述你遇到的问题请把它E-mail给我们如果可能请确定你的问题能够在开发工具上

重复出现请避免发送整个应用代码或很长的代码给我们以免延误我们对你的答复

注意

你总是可以从**www.keil.com/support**获得技术支持,产品升级,应用笔记和示例程序

软件开发流程

当你使用Keil Software工具时你的项目开发流程和其它软件开发项目的流程极其相似

1创建一个项目从器件库中选择目标器件配置工具设置

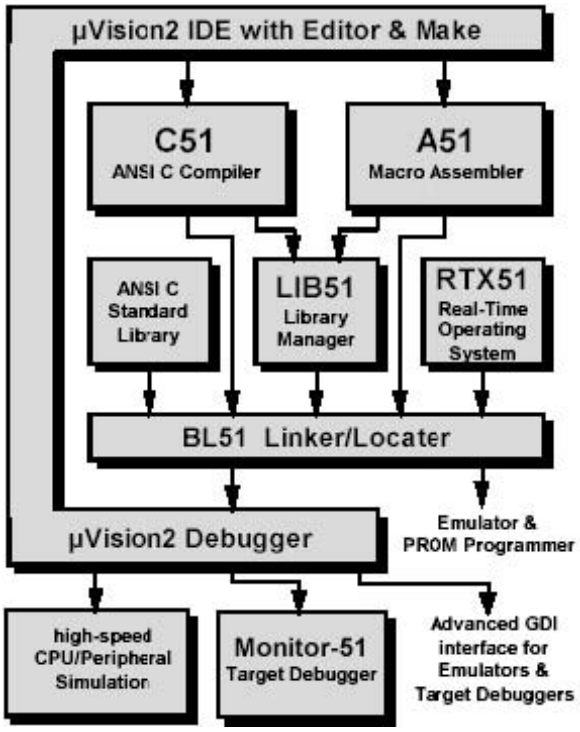
2用C语言或汇编语言创建源程序

3用项目管理器生成你的应用

4修改源程序中的错误

5测试连接应用

一个完整的8051工具集的框图可以最好地表述此开发流程每一个组件在下面详细描述



uVision2 IDE

uVision2 集成开发环境集成了一个项目管理器一个功能丰富有错误提示的编辑器以及设置选项生成工具在线帮助利用uVision2创建你的源代码并把它们组织到一个能确定你的目标应用的项目中去uVision2自动编译汇编连接你的嵌入式应用并为你的开发提供一个单一的焦点

C51编译器和A51汇编器

源代码由uVision2 IDE创建并被C51编译或A51汇编编译器和汇编器从源代码生成可重定位的目标文件

Keil C51编译器完全遵照ANSI C语言标准支持C语言的所有标准特性另外直接支持8051结构的几个特性被添加到里面

Keil A51宏汇编器支持8051及其派生系列的全部指令集

LIB51 库管理器

LIB51库管理器允许你从由编译器或汇编器生成的目标文件创建目标库库是一种被特别地组织过并在以后可以被连接重用的对象模块当连接器处理一个库时仅仅那些被使用的目标模块才被真正使用

BL51 连接器/定位器

BL51 连接器/定位器利用从库中提取的目标模块和由编译器或汇编器生成的目标模块创建一个绝对地址的目标模块一个绝对地址目标模块或文件包含不可重定位的代码和数据所有的代码和数据被安置在固定的存储器单元中此绝对地址目标文件可以用来

写入EPROM或其它存储器件

由uVision2调试器使用来模拟和调试

由仿真器用来测试程序

uVision2 调试器

uVision2源代码级调试器是一个理想地快速可靠的程序调试器此调试器包含一个高速模拟器能够让你模拟整个8051系统包括片上外围器件和外部硬件当你从器件库中选择器件时这个器件的特性将自动配置

uVision2调试器为你在实际目标板上测试你的程序提供了几种方法

安装MON51目标监控器到你的目标系统并且通过Monitor-51接口下载你的程序

利用高级的GDIAGDI接口把uVision2调试器绑定到你的目标系统

Monitor-51

uVision2调试器支持用Monitor-51进行目标板调试此监控程序驻留在你的目标板的 存储器里它利

用串口和uVision2调试器进行通信利用Monitor-51uVision2调试器 可以对你的目标硬件实行源代码级的调试

RTX51实时操作系统

RTX51实时操作系统是一个针对8051系列的多任务核RTX51实时内核从本质上简化了对实时事件反应速度要求高的复杂应用系统的设计编程和调试RTX51实时内核是完全集成到C51编译器中的从而方便使用任务描述表和操作系统的连接由BL51连接器/定位器自动控制

产品一览

Keil Software提供第一流的8051系列开发工具我们把我们的开发工具捆绑到不同的开发包或工具套件17页的对照表说明了整个Keil Software 8051开发工具每一个套件及其内容描述如下

PK51 专业开发套件

PK51专业开发套件包括了所有专业开发人员创建和调试复杂8051嵌入式应用系统所要用到的一切工具PK51专业开发套件可以针对所有的8051及其派生系列进行配置使用

DK51开发套件

DK51开发套件是PK51专业开发套件的精简版本它不包括小型RTX51实时操作系统此套件可以针对所有的8051及其派生系列进行配置使用

CA51编译套件

CA51编译套件是那些需要C编译器而不需要调试系统的开发人员的最好选择CA51开发包仅仅包含uVision2 IDEuVision2调试器不包括在内此套件可以针对所有的8051及其派生系列进行配置使用

A51汇编套件

A51汇编套件包括一个汇编器和你创建嵌入式应用所需要的所有功能此套件可以针对所有的8051

及其派生系列进行配置使用

RTX51 实时操作系统FR51

RTX51实时操作系统是一个8051系列MCU的实时内核RTX51 FULL提供RTX51 TINY的所有功能和一些扩展功能并且包括CAN通信协议接口

开发套件和工具的对照表

利用此表选择你所需要的开发套件.

Components	PK51	DK51†	CA51	A51	FR51
µVision2 Project Management & Editor	✓	✓	✓	✓	
A51 Assembler	✓	✓	✓	✓	
C51 Compiler	✓	✓	✓		
BL51 Linker/Locator	✓	✓	✓	✓	
LIB51 Library Manager	✓	✓	✓	✓	
µVision2 Debugger/Simulator	✓	✓			
RTX51 Tiny	✓				
RTX51 Full					✓

第2章 安装

本章解释如何设置操作环境以及如何在你的硬盘上安装本软件在开始安装程序之前请

确认你的计算机系统符合最小的需求

制作一份安装盘的副本

系统需求

为了取得比较好的运行效果最低的硬件和软件配置必须满足

具有奔腾奔腾II或兼容的处理器的个人计算机

操作系统为WIN95WIN98WINNT4.0或更高

RAM大于16MB

20MB 的硬盘空余空间

安装详细说明

所有的Keil产品都带有一个安装程序安装方便8051开发工具的安装步骤如下

插入Keil开发工具光盘

从CD浏览界面选择安装软件

跟随提示进行安装操作

注意

当你插入CD时你的计算机可能会自动浏览CD如果没有运行***\KEIL\SETUP\SETUP.EXE***安装软件

文件夹组织结构

安装程序复制开发工具到基本目录的各个子目录中默认的基本目录是C:\KEIL下表列出的文件夹结构是包括所有8051开发工具的全部安装信息你的安装信息由你购买的开发套件决定

文件夹	描述
C:\KEIL\C51\ASM	汇编SFR定义文件和模板源程序文件
C:\KEIL\C51\BIN	8051工具的执行文件
C:\KEIL\C51\EXAMPLES	示例应用
C:\KEIL\C51\RTX51	完全实时操作系统文件
C:\KEIL\C51\RTX_TINY	小型实时操作系统文件
C:\KEIL\C51\INC	C编译器包含文件
C:\KEIL\C51\LIB	C编译器库文件启动代码和常规I/O资源
C:\KEIL\C51\MONITOR	目标监控文件和用户硬件的监控配置
C:\KEIL\UV2	普通uVision2文件

在本使用指南中我们假定用户采用默认的文件夹结构如果你安装你的软件到一个不同的文件夹你

必须调整路径名以适应你的安装

第3章 开发工具

Keil 8051开发工具提供数个十分有用的特性可以帮助你快速地成功开发嵌入式应用这些工具使用简单并保证你达到你的设计目的

uVision2 集成开发环境

uVision2 IDE 是一个基于Window的开发平台包含一个高效的编辑器一个项目管理器和一个MAKE工具

uVision2支持所有的KEIL 8051工具包括C编译器宏汇编器连接/定位器目标代码到HEX的转换器uVision2通过以下特性加速你的嵌入式系统的开发过程

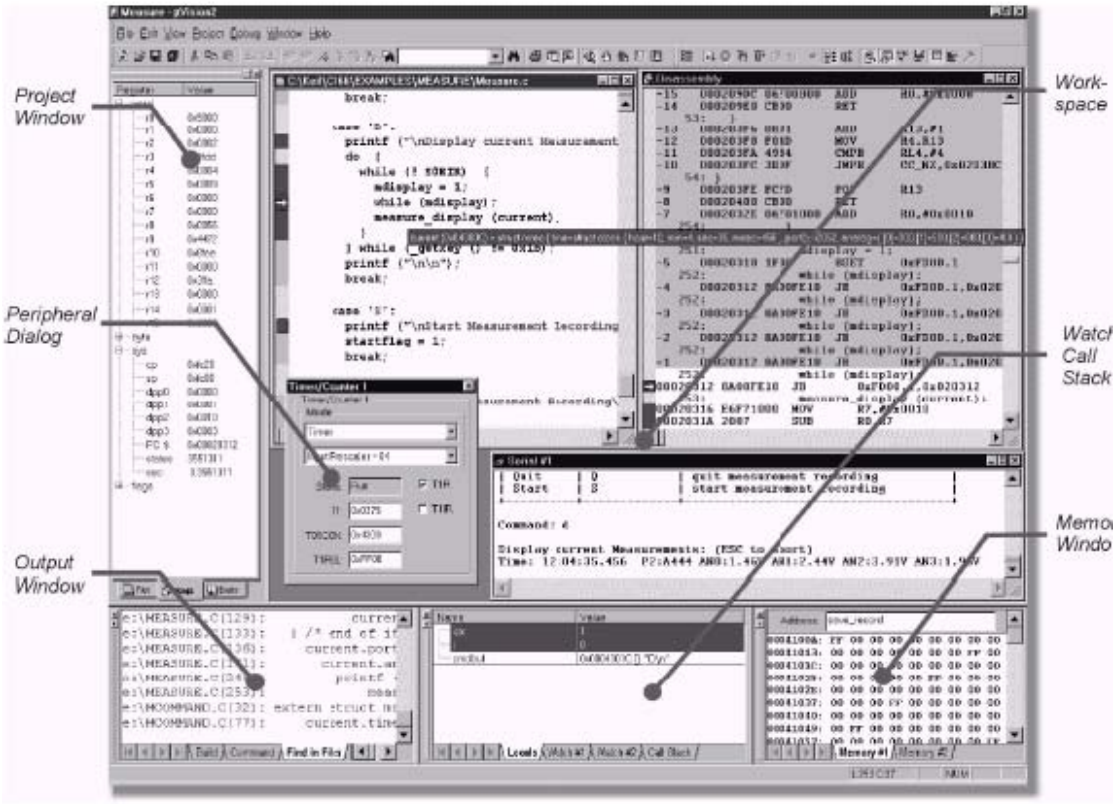
全功能的源代码编辑器
器件库用来配置开发工具设置
项目管理器用来创建和维护你的项目
集成的MAKE工具可以汇编编译和连接你的嵌入式应用
所有开发工具的设置都是对话框形式的
真正的源代码级的对CPU和外围器件的调试器
高级GDIAGDI接口用来在目标硬件上进行软件调试以及和Monitor-51进行通信
与开发工具手册和器件数据手册和用户指南有直接的链接

注意

*uVision2*调试器的特性只有***PK51***和***DK51***套件具备

关于开发环境

uVision2 界面提供一个菜单一个工具条以便你快速选择命令按钮另外还有源代码的显示窗口对话框和信息显示uVision2允许同时打开浏览多个源文件



菜单条工具条和快捷键

菜单条提供各种操作菜单如编辑操作项目维护开发工具选项设置调试程序 窗口选择和处理在线帮助工具条按钮允许你快速地执行uVision2命令键盘快捷键你自己可以配置允许你执行uVision2命令下面的表格列出了uVision2菜单项命令工具条图标默认的快捷键以及他们的描述

文件菜单和命令**File**

菜单	工具条	快捷键	描述
New		Ctrl+N	创建新文件
Open		Ctrl+O	打开已经存在的文件
Close			关闭当前文件
Save		Ctrl+S	保存当前文件
Save all			保存所有文件
Save as			另外取名保存
Device Database			维护器件库
Print Setup			设置打印机
Print		Ctrl+P	打印当前文件
Print Preview			打印预览

1-9 打开最近用过的文件
Exit 退出uVision2提示是否保存文件

编辑菜单和编辑器命令**(Edit)**

菜单 工具条 快捷键 描述
Home 移动光标到本行的开始
End 移动光标到本行的末尾
Ctrl+Home 移动光标到文件的开始
Ctrl+End 移动光标到文件的结束
Ctrl+<- 移动光标到词的左边
Ctrl+-> 移动光标到词的右边
Ctrl+A 选择当前文件的所有文本内容
Undo Ctrl+Z 取消上次操作

Redo Ctrl+Shift+Z 重复上次操作

Cut Ctrl+X 剪切所选文本

Ctrl+Y 剪切当前行的所有文本
Copy Ctrl+C 复制所选文本

Paste Ctrl+V 粘贴

Indent 将所选文本右移一个制表键的距离

Selected Text
Unindent 将所选文本左移一个制表键的距离

Selected Text
Toggle Bookmark Ctrl+F2 设置/取消当前行的标签

Goto Next Bookmark F2 移动光标到下一个标签处

Goto Previous Bookmark Shift+F2 移动光标到上一个标签处

Clear All Bookmarks 清除当前文件的所有标签

Find Ctrl+F 在当前文件中查找文本


F3 向前重复查找
Shift+F3 向后重复查找
Ctrl+F3 查找光标处的单词
Ctrl+] 寻找匹配的大括号圆括号方括号
用此命令将光标放到大括号圆括号或方括号的前面
Replace Ctrl+H 替换特定的字符
Find in Files 在多个文件中查找

/*****************************译者注-开始*************************************/

Ctrl+]命令在我的*uVision2 2.20a*中好象没有作用另外我的*uVision2*的***Edit***菜单中还有一个***Goto Matching brace*** 命令在最后功能是选择匹配的一对大括号圆括号或方括号中的内容但是在操作之前你必须把光标置于其中一个括号的旁边前或后都可以但是要注意必须紧靠

/*****************************译者注-结束*************************************/

选择文本命令

在uVision2中你可以通过按住**Shift**键和相应的光标操作键来选择文本如**Ctrl+->** 是移动光标到下一个词那么**Ctrl+Shift+->** 就是选择当前光标位置到下一个词的开始位置间的文本

当然你也可以用鼠标来选择文本操作如下

要选择　　鼠标操作

任意数量的文本　　在你要选择的文本上拖动鼠标
一个词　　双击此词
　一行文本　移动鼠标到此行的最左边直到鼠标变成右指向的箭头然后单击
　多行文本　移动鼠标到此行的最左边直到鼠标变成右指向的箭头然后相应拖动
一个距形框中的文本　按住**Alt**键然后相应拖动鼠标

视图菜单**View**

菜单　　工具条　快捷键　描述

Status Bar　　显示/隐藏状态条
File Toolbar　　显示/隐藏文件菜单条
Build Toolbar　　显示/隐藏编译菜单条
Debug Toolbar　　显示/隐藏调试菜单条
Project Window　　显示/隐藏项目窗口

Output Window　　显示/隐藏输出窗口

Source Browser　　打开资源浏览器

Disassembly Window　　显示/隐藏反汇编窗口

Watch & Call　　显示/隐藏观察和堆栈窗口

Stack Window
Memory Window　　显示/隐藏存储器窗口

Code Coverage Window　　显示/隐藏代码报告窗口

Performance
Analyzer Window 显示/隐藏性能分析窗口

Symbol Window 显示/隐藏字符变量窗口
Serial Window #1 显示/隐藏串口1的观察窗口

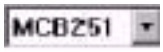
Serial Window #2 显示/隐藏串口2的观察窗口
Toolbox 显示/隐藏自定义工具条

Periodic Window Update 程序运行时刷新调试窗口
Workbook Mode 显示/隐藏窗口框架模式
Options 设置颜色字体快捷键和编辑器的选项

项目菜单和项目命令**Project**

菜单 工具条 快捷键 描述

New Project 创建新项目
Import Vision1
Project 转化uVision1的项目
Open Project 打开一个已经存在的项目
Close Project 关闭当前的项目
Target Environment 定义工具包含文件和库的路径
Targets, Groups,Files 维护一个项目的对象文件组和文件
Select Device for Target 选择对象的CPU
Remove 从项目中移走一个组或文件.
Options Alt+F7 设置对象组或文件的工具选项



File Extensions 选择不同文件类型的扩展名
Build Target F7 编译修改过的文件并生成应用

Rebuild Target 重新编译所有的文件并生成应用

Translate Ctrl+F7 编译当前文件

Stop Build 停止生成应用的过程

1-9 打开最近打开过的项目

调试菜单和调试命令**Debug**

菜单 工具条 快捷键 描述

Start/Stop Ctrl+F5 开始/停止调试模式

Debugging

Go F5 运行程序直到遇到一个中断

Step F11 单步执行程序遇到子程序则进入

Step over F10 单步执行程序跳过子程序

Step out of Ctrl+F11 执行到当前函数的结束

Current function

Stop Running ESC 停止程序运行

Breakpoints 打开断点对话框

Insert/Remove 设置/取消当前行的断点

Breakpoint

Enable/Disable 使能/禁止当前行的断点

Breakpoint

Disable All 禁止所有的断点

Breakpoints

Kill All 取消所有的断点

Breakpoints

Show Next 显示下一条指令

Statement

Enable/Disable 使能/禁止程序运行轨迹的标识

Trace Recording

View Trace 显示程序运行过的指令

Records

Memory Map 打开存储器空间配置对话框

Performance 打开设置性能分析的窗口

Analyzer

Inline Assembly 对某一个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**

菜单 工具条 快捷键 描述

Reset CPU 复位CPU

Interrupt, 打开所上外围器件的设置对话框 I/O-Ports, 对话框的种类及内容依赖于你选择的CPU

Serial,

Timer,
A/D Converter,
D/A Converter,
I2C Controller,
CAN Controller,
Watchdog

工具菜单**Tool**

利用工具菜单你可以配置运行Gimpel PC-LintSiemens Easy-Case和用户程序通过Customize Tools Menu菜单你可以添加你想要添加的程序更详细的信息请参考72页的Using the Tools Menu

菜单 工具条 快捷键 描述

Setup PC-Lint 配置Gimpel Software的PC-Lint程序
Lint 用PC-Lint处理当前编辑的文件
Lint all C Source Files 用PC-Lint处理你项目中所有的C源代码文件
Setup Easy-Case 配置Siemens的Easy-Case程序
Start/Stop Easy-Case 运行/停止Siemens的Easy-Case程序
Show File (Line) 用Easy-Case处理当前编辑的文件
Customize Tools Menu 添加用户程序到工具菜单中

软件版本控制系统菜单**SVCS**

用此菜单来配置和添加软件版本控制系统的命令更详细的信息参见76页的Using the SVCS Menu

菜单 工具条 快捷键 描述

Configure 配置软件版本控制系统的命令
Version Control

视窗菜单**Window**

菜单 工具条 快捷键 描述

Cascade 以互相重叠的形式排列文件窗口
Tile Horizontally 以不互相重叠的形式水平排列文件窗口
Tile Vertically 以不互相重叠的形式垂直排列文件窗口
Arrange Icons 排列主框架底部的图标
Split 把当前的文件窗口分割为几个
1-9 激活指定的窗口对象

帮助菜单**Help**

菜单 工具条 快捷键 描述

Help topics 打开在线帮助

About Vision　　显示版本信息和许可证信息

uVision2 有两种操作模式

创建模式让你编译应用中所有的文件以产生执行程序此模式的特性在57页Creating Applications中描述
调试模式提供一个非常强劲的调试器你可以用它来调试你的程序此模式的特性在93页Testing Programs中描述

在两种模式下你都可以用源文件编辑器来编辑你的源代码

C51优化的C语言交叉编译器

Keil C51交叉编译器是一个基于ANSI C标准的针对8051系列MCU的C编译器生成的可执行代码快速紧凑在运行效率和速度上可以和汇编程序得到的代码相媲美

和汇编语言相比用C语言这样的高级语言有很多优势比如
对处理器的指令集不必了解8051 CPU的基本结构可以了解但不是必须的
寄存器的分配以及各种变量和数据的寻址都由编译器完成
程序拥有了正式的结构由C语言带来的并且能被分成多个单独的子函数这使整个应用系统的结构变得清晰同时让源代码变得可重复使用
选择特定的操作符来操作变量的能力提高了源代码的可读性
可以运用和人的思维很接近的词汇和算法表达式
编写程序和调试程序的时间得到很大程度的缩短
C运行连接库包含一些标准的子程序如格式化输出数字转换浮点运算
由于程序的模块结构技术使得现有的程序段可以很容易的包含到新的程序中去
ANSI 标准的C语言是一种非常方便的获得广泛应用的在绝大部分系统中都能够很容易得到的语言

因此如果需要现有的程序可以很快地移植到其他的处理器上节省投资

C51 语言的扩展

虽然C51是一个兼容ANSI的编译器但为了支持8051系列MCU还是加入了一些扩展的内容C51编译器的扩展内容包括

数据类型
存储器类型
指针
重入函数
中断服务程序
实时操作系统

和PL/M及A51源程序的接口

以下各节简单地描述了上述的扩展特性

数据类型

本C51编译器支持下表列出的各种规格的数据类型.除了这些数据类型以外变量可以组合成结构联合及数组除非特别说明这些变量都可以用指针存取

Data Type	Bits	Bytes	Value Range
bit †	1		0 to 1
signed char	8	1	-128 to +127
unsigned char	8	1	0 to 255
enum	16	2	-32768 to +32767
signed short	16	2	-32768 to +32767
unsigned short	16	2	0 to 65535
signed int	16	2	-32768 to +32767
unsigned int	16	2	0 to 65535
signed long	32	4	-2147483648 to 2147483647
unsigned long	32	4	0 to 4294967295
float	32	4	±1.175494E-38 to ±3.402823E+38
sbit †	1		0 to 1
sfr †	8	1	0 to 255
sfr16 †	16	2	0 to 65535

注* bit, sbit sfr,和sfr16为8051硬件和C51及C251编译器所特有它们不是ANSI C 的一部分也不能用指针对它们进行存取

这些**sbit sfr**和**sfr16**类型的数据使你能够操作8051MCU所提供的特殊功能寄存器例如下面的表达式

```
sfr P0 = 0x80; /* Define 8051 P0 SFR */
```

声明了一个变量**P0**并且把它和位于0x808051的端口0处的特殊功能寄存器联系在一起

当结果的数据类型和源数据类型不同时C51编译器在数据类型间自动进行转换例如一个bit变量赋值给一个interger变量时将会被转换为integer当然你可以用类型表示进行强制转换数据转换时要注意有符号变量的转换其符号是自动扩展的

存储器类型

本C51编译器支持8051及其派生类型的结构能够访问8051的所有存储器空间具有下表列出的存储器类型的变量都可以被分配到某个特定的存储器空间

存储器类型 描述

code 程序空间64 Kbytes通过 MOVC @A+DPTR 访问

data 直接访问的内部数据存储器访问速度最快128 bytes

idata 间接访问的内部数据存储器可以访问所有的内部存储器空间256 bytes
bdata 可位寻址的内部数据存储器可以字节方式也可以位方式访问16 bytes
xdata 外部数据存储器64 Kbytes通过MOVX @DPTR访问
pdata 分页的外部数据存储器256 bytes通过 MOVX @Rn 访问

访问内部数据存储器将比访问外部数据存储器快的多由于这个原因你应该把频繁使用的变量放置在内部数据存储器中把很少使用的变量放在外部数据存储器中这通过使用SMALL模式将很容易就做到通过定义变量时包括存储器类型你可以定义此变量存储在你想要的存储器中

在变量的声明中你可以包括存储器类型和**signed**或**unsigned**属性

```
char data var1;
char code text[] = "ENTER PARAMETER";
unsigned long xdata array[100];
float idata x,y,z;
unsigned int pdata dimension;
unsigned char xdata vector[10][4][4];
char bdata flags;
```

如果在变量的定义中没有包括存储器类型将自动选用默认或暗示的存储器类型暗示的存储器类型适用于所有的全局变量和静态变量还有不能分配在寄存器中的函数参数和局部变量默认的存储器类型由编译器的参数SMALLCOMPACT及LARGE决定这些参数定义了编译时使用的存储模式

存储模式

存储模式决定了默认的存储器类型此存储器类型将应用于函数参数局部变量和定义时未包含存储器类型的变量你可以在命令行用SMALLCOMPACT和LARGE参数定义存储模式定义变量时使用存储器类型显式定义将屏蔽默认存储器类型

小**(SMALL)**模式

所有变量都默认在8051的内部数据存储器中这和用data显式定义变量起到相同的作用在此模式下变量访问是非常快速的然而所有数据对象包括堆栈都必须放在内部RAM中堆栈空间面临溢出因为堆栈所占用多少空间依赖于各个子程序的调用嵌套深度在典型应用中如果具有代码分段功能的BL51连接/定位器被配置成覆盖内部数据存储器中的变量时此SMALL模式是最好的选择

紧凑**COMPACT**模式

此模式中所有变量都默认在8051的外部数据存储器的一页中地址的高字节往往通过Port 2输出其值必须由你在启动代码中设置编译器不会为你设置这和用pdata显式定义变量起到相同的作用此模式最多只能提供256字节的变量这种限制来自于间接寻址所使用的R0,R1MOVX @R0/R1这种模式不如SMALL模式高效所以变量的访问不够快不过它比LARGE模式要快

大**LARGE**模式

在大模式下所有的变量都默认在外部存储器中xdata这和用xdata显式定义变量起到相同的作用数据指针DPTR用来寻址通过DPTR进行存储器的访问的效率很低特别是在对一个大于一个字节的变量进行操作时尤为明显此数据访问类型比SMALL和COMPACT模式需要更多的代码

注意
你或许应该一直使用小模式它产生最快最紧凑效率最高的代码
你最好显式定义你的变量的存储器类型只有当你的应用不能在*SMALL*模式下操作时你才需要往上增加你的存储模式
的，你可能要在上增加你的存储模式。

指针

C51编译器支持用星号*进行指针声明你可以用指针完成在标准C语言中有的所有操作
另外由于8051及其派生系列所具有的独特结构C51编译器支持两种不同类型的指针存储器指针和通用指针

通用指针

通用或未定型的指针的声明和标准C语言中一样如

```
char *s; /* string ptr */
int *numptr; /* int ptr */
long *state; /* long ptr */
```

通用指针总是需要三个字节来存储第一个字节是用来表示存储器类型第二个字节是指针的高字节第三字节是指针的低字节

通用指针可以用来访问所有类型的变量而不管变量存储在哪个存储空间中因而许多库函数都使用通用指针通过使用通用指针一个函数可以访问数据而不用考虑它存储在什么存储器中

通用指针很方便但是也很慢在所指向目标的存储空间不明确的情况下它们用的最多

存储器指针

存储器指针或类型确定的指针在定义时包括一个存储器类型说明并且总是指向此 说明的特定存储器空间例如

```
char data *str; /* ptr to string in data */
int xdata *numtab; /* ptr to int(s) in xdata */
long code *powtab; /* ptr to long(s) in code */
```

正是由于存储器类型在编译时已经确定通用指针中用来表示存储器类型的字节就不再需要了

指向idatadatabdata和pdata的存储器指针用一个字节保存指向code和xdata的存储器指针用两个字节保存使用存储器指针比通用指针效率要高速度要快当然存储器指针的使用不是很方便在所指向目标的存储空间明确并不会变化的情况下它们用的最多

存储器指针和通用指针的比较

使用存储器指针可以显著的提高8051 C程序的运行速度

下面的示例程序说明了使用不同的指针在代码长度占用数据空间和运行时间上的不同

Description	Idata Pointer	Xdata Pointer	Generic Pointer
C源程序	idata *ip; char val; val = *ip;	xdata *xp; har val; val = *xp;	char *p; char val; val = *xp;
编译后的代码	MOV R0,ip MOV val,@R0	MOV DPL,xp +1 MOV DPH,xp MOV A,@DPTR MOV val,A	MOV R1,p + 2 MOV R2,p + 1 MOV R3,p CALL CLDPTR
指针大小	1 byte	2 byte	3 byte
代码长度	4 bytes	9 bytes	11 bytes + library call
执行时间	4 cycles	7 cycles	13 cycles

重入函数

多个进程可以同时使用一个重入函数当一个重入函数被调用运行时另外的一个进程可能中断此运行过程然后再次调用此重入函数通常情况下C51函数不能被递归调用也不能应用导致递归调用的结构有此限制是由于函数参数和局部变量是存储在固定的地址单元中重入函数特性允许你声明一个重入函数即可以被递归调用的函数如

```
int calc (char i, int b) reentrant
{
int x;
x = table [i];
return (x * b);
}
```

重入函数可以被递归调用也可以同时被两个或更多的进程调用重入函数在实时应用中及中断服务程序代码和非中断程序代码必须共用一个函数的场合中经常用到

对每一个重入函数来说根据存储模式重入堆栈被安置在内部或外部单元中

注意

在一个基本函数的基础上添加**reentrant**说明从而使它具有重入特性

需要注意的是,你可以选择哪些必须的函数为重入函数而不需将全部程序声明为重入函数

把全部程序声明为重入函数将增加目标代码的长度并减慢运行速度

中断服务程序

C51编译器允许你用C语言创建中断服务程序你仅仅需要关心中断号和寄存器组的选择编译器自动产生中断向量和程序的入栈及出栈代码在函数声明时包括**interrupt**将把所声明的函数定义为一个中断服务程序另外你可以用**using**定义此中断服务程序所使用的寄存器组

```
unsigned int interruptcnt;
unsigned char second;
void timer0 (void) interrupt 1 using 2
{
 if (++interruptcnt == 4000)
 {   /* count to 4000 */
  second++; /* second counter */
  interruptcnt = 0; /* clear int counter */
 }
}
```

参数传递

C51编译器能在CPU寄存器中传递最多三个参数由于不用从存储器中读出和写入参数从而显著提高了系统性能参数传递由REGPARMS和NOREGPARMS编译参数所控制下表列出了不同的参数和数据类型所占用的寄存器

参数	char,	int,	long,	generic
数目	1-byte pointer	2-byte pointer	float	pointer
1	R7	R6 & R7	R4 R7	R1 R3
2	R5	R4 & R5		
3	R3	R2 & R3		

如果没有CPU寄存器供参数传递所用或太多的参数需要传递时地址固定的存储器将用来存储这些额外的参数

函数返回值

函数返回值总是通过CPU寄存器进行下表列出了返回各种数据时所用的CPU寄存器

返回数据类型	寄存器	描述
bit	Carry Flag	
char,unsigned char,1-byte pointer	R7	
int, unsigned int, 2-byte pointer	R6 & R7	MSB in R6, LSB in R7
long, unsigned long	R4 R7	MSB in R4, LSB in R7
float	R4 R7	32-Bit IEEE format
generic pointer	R1 R3	Memory type in R3, MSB R2, LSB R1

寄存器优化

根据程序前后的联系C51编译器分配最多7个寄存器来存储寄存器变量C51编译器能分析每个程序模块中对寄存器的修改连接程序产生一个全局的项目级的寄存器文件此文件包含被外部程序改变的所有寄存器的信息因而C51编译器知道整个应用中每个函数所使用的寄存器并能为每个C函数优化分配CPU寄存器

对实时操作系统的支持

C51编译器很好地集成了RTX-51 Full和RTX-51 Tiny 多任务实时操作系统任务描述表在连接过程中控制和产生关于RTX实时操作系统的详细信息请参考169页开始的 "RTX-51 Real-Time Operating System"一章

和汇编语言的接口

你可以很容易在C程序中调用汇编程序反之依然函数参数通过CPU寄存器传递或使用NOREGPARMS参数指示编译器通过固定的存储器传递从函数返回的值总是通过CPU寄存器传递除了直接产生目标代码外你还可以用SRC编译参数指示编译器产生汇编源代码文件供A51 汇编器使用例如下面的C语言源代码

```
unsigned int asmfunc1 (unsigned int arg)
{
  return (1 + arg);
}
```

用SRC指示C51编译器编译时产生以下汇编文件

```
?PR?_asmfunc1?ASM1 SEGMENT CODE
PUBLIC        asmfunc1
    RSEG ?PR?_asmfunc1?ASM1
    USING 0
  asmfunc1:
;---- Variable 'arg?00' assigned to Register 'R6/R7' ----
    MOV A,R7      ; load LSB of the int
    ADD A,#01H      ; add 1
    MOV R7,A      ; put it back into R7
    CLR A
    ADDC A,R6      ; add carry & R6
    MOV R6,A
 ?C0001:
    RET        ; return result in R6/R7
```

你可以用**#pragma asm** 和 **#pragma endasm** 预处理指示器来在你的C语言程序中插入汇编指令

和**PL/M-51**的接口

Intel的PL/M-51是一种流行的编程语言在很多方面和C语言类似你很容易就可以将C程序和PL/M-51程序联接起来

在你用alien声明PL/M-51函数后你就可以从C语言中调用它们所有在PL/M-51模块中定义的全局变量都可以在C语言程序中使用例如

extern alien char plm_func (int, char);

PL/M-51编译器和Keil Software工具都产生OMF51格式的目标文件连接程序使用OMF51文件来处理外部字符变量而不管它们在什么地方声明和使用

代码优化

C51是一个杰出的优化编译器它通过很多步骤以确保产生的代码是最有效率的最小和/或最快编译器通过分析初步的代码 产生最终的最有效率的代码序列以此来保证你的C语言程序占用最少空间的同时运行的快而有效

C51编译器提供9个优化级别每个高一级的优化级别都包括比它低的所有优化级别的优化内容以下列出的是目前C51编译器提供的所有优化级别的内容

常量折叠在表达式及寻址过程中出现的常量被综合为一个单个的常量
跳转优化采用反转跳转或直接指向最终目的的跳转从而提升了程序的效率
哑码消除永远不可能执行到的代码将自动从程序中剔除
寄存器变量只要可能局部变量和函数参数被放在CPU寄存器中不需要为这些变量再分配存储器空间

通过寄存器传递参数最多三个参数通过寄存器传递
消除全局公用的子表达式只要可能程序中多次出现的相同的子表达式或地址计算表达式将只计算一次
合并相同代码利用跳转指令相同的代码块被合并
重复使用入口代码需要多次使用的共同代码被移到子程序的前面以缩减代码长度
公共块子程序需要重复使用的多条指令被提取组成子程序指令被重新安排以最大化一个共用子程序的长度

/**********译者注-开始**********/
对于操作硬件有时序要求的应用,慎用第9级优化.
因为它将可能调整你的指令顺序.
/**********译者注-结束**********/

对8051的特殊优化

窥孔优化当能够缩小代码空间或执行时间时复杂的操作被简单的操作代替
访问优化常量和变量被计算后直接包含在操作中
扩展访问优化用DPTR做存储器指针来增加代码的密度
数据覆盖一个函数的数据和位变量空间是可覆盖的BL51连接器将采用覆盖技术来分配变量空间
Case/Switch优化根据使用的数字序列和位置用跳转表或一连串的跳转指令来优化switch及case结构

代码生成选项

空间优化公共C操作被子程序代替以程序执行速度的降低来换取程序代码空间的缩减
时间优化公共C操作被嵌入到程序中以程序代码空间的增加来换取程序执行速度的提高
不用绝对寄存器不用绝对寄存器地址访问程序代码依赖于寄存器的分段
不用寄存器传递参数用局部数据段来传递参数而不用寄存器这是为了兼容早期版本的C51编译器PL/M-51编译器和ASM-51汇编器

调试

C51编译器使用Intel目标格式OMF51来产生目标文件和全部的字符变量信息而且编译器还包含所有必要的信息如变量名函数名行数以及uVision2调试器或任何兼容Intel格式的仿真器用来逐条彻底地调试和分析程序所需要的信息
另外使用OBJECTEXTEND参数可以指示编译器产生附加的变量类型信息到目标文件中这样利用相应的仿真器就可以显示变量和结构的数据信息你可以向你的仿真器供应商询问是否支持Intel OMF51格式和Keil软件生成的目标模块

库函数

C51编译器包含有ANSI标准的7个不同的的编译库从而满足不同功能的需要

库文件	描述
C51S.LIB	小模式库不支持浮点运算
C51FPS.LIB	小模式库支持浮点运算
C51C.LIB	紧凑模式库不支持浮点运算
C51FPC.LIB	紧凑模式库支持浮点运算
C51L.LIB	大模式库不支持浮点运算
C51FPL.LIB	大模式库支持浮点运算
80C751.LIB	Philips 8xC751及其派生系列使用的库

和硬件相联系的输入/输出操作的库函数模块的源代码文件位于\KEIL\C51\LIB文件夹中你可以利用这些文件来修改你的库以适应你目标板上的任何器件的输入/输出操作

内连的库函数

本编译器的库中包含一定数量的函数是内连函数内连函数不产生ACALL或LCALL指令来执行库函数以下的内连函数产生内连的代码因而它比一个调用函数要快而有效

内连函数	描述
crol	字节左移
cror	字节右移
irol	整数左移
iror	整数右移
lrol	长整数左移
lror	长整数右移
nop	空操作
testbit	判断并清除8051 JBC 指令

编译器的调用

通常情况下当你创建你的项目时C51编译器由uVision2 IDE调用当然你也可以在DOS方式在命令行键入C51来运行你的C源程序文件名必须和编译控制参数一起在命令行输入如

```
>C51 MODULE.C COMPACT PRINT (E:M.LST) DEBUG SYMBOLS
C51 COMPILER V6.00
C51 COMPILATION COMPLETE. 0 WARNING(S), 0 ERROR(S)
```

编译器控制参数可以在命令行输入也可以在文件中的开头用#pragma 定义要想知道全部的编译器控制参数请参考213页 的"C51/C251 Compiler"

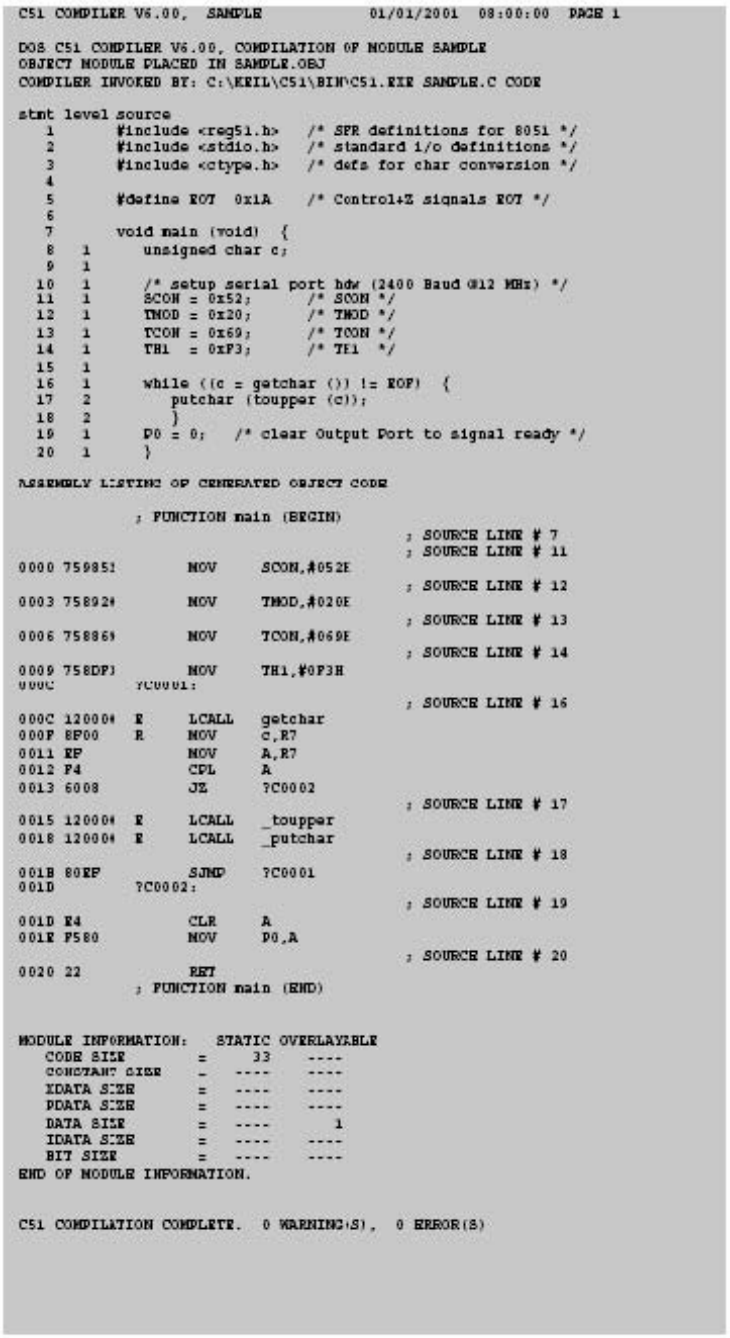
示例程序

下面的程序显示了C51编译器的一些功能特性本C51编译器根据编译控制参数产生相应的OMF51格式的目标文件

编译器报告所有必要的信息如变量名函数名行数以及uVision2调试器或其它仿真器用来详细调试和分析程序所需要的信息

编译后C51编译器产生一个列表文件文件中包含源代码指示信息汇编清单和字符表

下页中展示了一个C51编译器生成的示例列表文件



```
C51 COMPILER V6.00,  SAMPLE              01/01/2001  08:00:00  PAGE 1

DOS C51 COMPILER V6.00, COMPILATION OF MODULE SAMPLE
OBJECT MODULE PLACED IN SAMPLE.OBJ
COMPILER INVOKED BY: C:\KEIL\C51\BIN\C51.EXE SAMPLE.C CODE

stmt level source
   1       #include <reg51.h>   /* SFR definitions for 8051 */
   2       #include <stdio.h>   /* standard i/o definitions */
   3       #include <ctype.h>   /* defs for char conversion */
   4
   5       #define EOT  0x1A    /* Control+Z signals EOT */
   6
   7       void main (void)  {
   8   1     unsigned char c;
   9   1
  10   1     /* setup serial port hdw (2400 Baud @12 MHz) */
  11   1     SCON = 0x52;       /* SCON */
  12   1     TMOD = 0x20;       /* TMOD */
  13   1     TCON = 0x69;       /* TCON */
  14   1     TH1  = 0xF3;       /* TH1  */
  15   1
  16   1     while ((c = getchar ()) != EOF)  {
  17   2       putchar (toupper (c));
  18   2       }
  19   1     P0 = 0;   /* clear Output Port to signal ready */
  20   1     }

ASSEMBLY LISTING OF GENERATED OBJECT CODE

             ; FUNCTION main (BEGIN)
                                          ; SOURCE LINE # 7
                                          ; SOURCE LINE # 11
0000 759852        MOV     SCON,#052H
                                          ; SOURCE LINE # 12
0003 758920        MOV     TMOD,#020H
                                          ; SOURCE LINE # 13
0006 758869        MOV     TCON,#069H
                                          ; SOURCE LINE # 14
0009 758DF3        MOV     TH1,#0F3H
000C             ?C0001:
                                          ; SOURCE LINE # 16
000C 120000  E     LCALL   getchar
000F 8F00    R     MOV     c,R7
0011 EF            MOV     A,R7
0012 F4            CPL     A
0013 6008          JZ      ?C0002
                                          ; SOURCE LINE # 17
0015 120000  E     LCALL   _toupper
0018 120000  E     LCALL   _putchar
                                          ; SOURCE LINE # 18
001B 80EF          SJMP    ?C0001
001D             ?C0002:
                                          ; SOURCE LINE # 19
001D E4            CLR     A
001E F580          MOV     P0,A
                                          ; SOURCE LINE # 20
0020 22            RET
             ; FUNCTION main (END)


MODULE INFORMATION:   STATIC OVERLAYABLE
   CODE SIZE        =     33    ----
   CONSTANT SIZE    =   ----    ----
   XDATA SIZE       =   ----    ----
   PDATA SIZE       =   ----    ----
   DATA SIZE        =   ----       1
   IDATA SIZE       =   ----    ----
   BIT SIZE         =   ----    ----
END OF MODULE INFORMATION.


C51 COMPILATION COMPLETE.  0 WARNING(S),  0 ERROR(S)
```

The C51 Compiler produces a listing file with line numbers and the time and date of compilation.

Information about compiler invocation and the object file generated is printed.

The listing contains a line number before each source line and the instruction nesting ( ) level.

If errors or possible sources of errors exist an error or warning message is displayed.

Enable under µVision2 Options for Target – Listing - Assembly Code the C51 CODE directive. This gives you an assembly listing file with embedded source line numbers.

A memory overview provides information about the occupied 8051 memory areas.

The number of errors and warnings in the program are included at the end of the listing.

C51编译器产生行号编译时的时间和日期

编译器的运行和产生的目标文件的信息被记录在案

列表文件在每个源代码前面包含行号和{}的嵌套层数

如果错误或可能错误的代码存在一个错误或告警信息将显示出来

选择在Vision2-Options for Target – Listing 中的Assembly Code 代码指示选项将在列表文件的汇编代码处加入原代码所在的行号

存储器一览表提供了8051存储器占用信息

程序中的错误和告警总数包括在文件的结尾处.

A51宏汇编器

本A51是一个8051MCU系列的宏汇编器它把汇编语言翻译成机器代码本A51汇编器允许你定义你

程序中的每一个指令在需要极快的运行速度很小的代码空间精确的硬件控制时使用本汇编器的宏特性让公共代码只需要开发一次从而节约了开发和维护的时间

源码级调试

本A51汇编器在生成的目标文件中包含全部的行号字符及其类型的信息这让你的调试器能够精确显示程序变量行号是为了uVision2调试器或第三方仿真器源代码级调试你汇编过的程序时使用的

功能一览

本A51汇编器翻译汇编源程序为可重定位的目标代码它产生一个列表文件其中可以包含或不包含字符表及交叉参考信息

本A51汇编器支持两种宏处理

标准的宏处理

这是一个比较容易使用的宏处理它允许你在你的8051汇编代码中定义和使用宏它标准的宏语法和其它许多汇编器中使用的相同

宏处理语言**(MPL)**

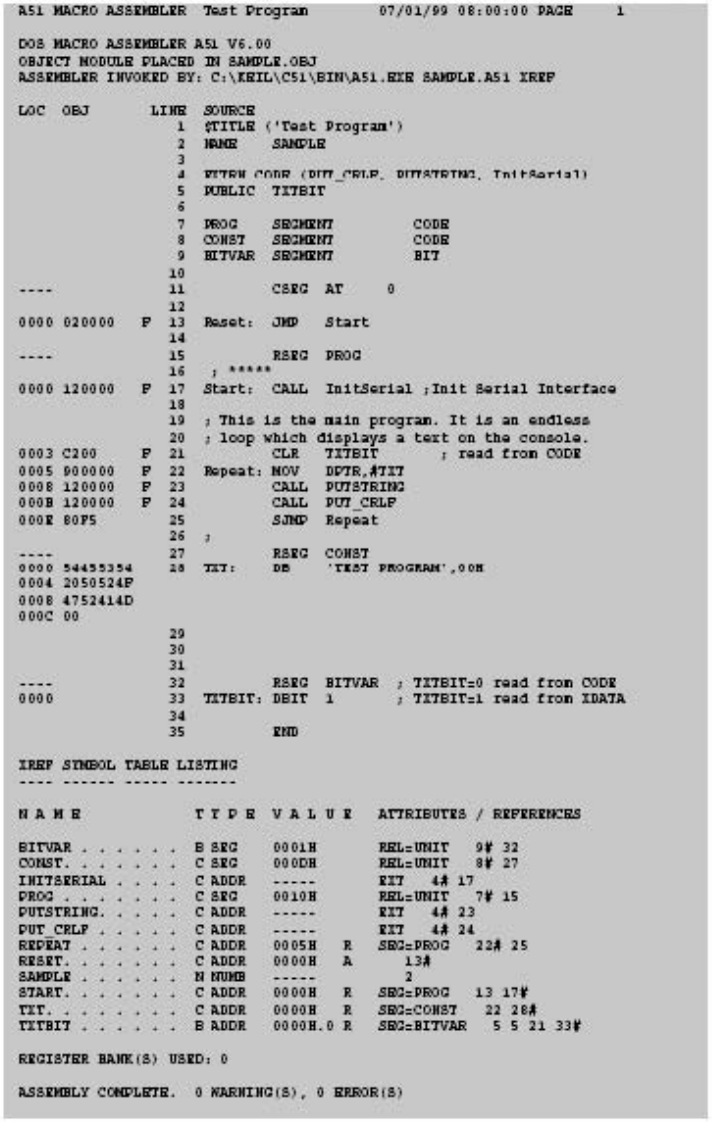
是一个和Intel ASM51宏处理兼容的字符串替换工具MPL有几个预先定义好的宏处理功能来执行一些有用的操作如字符串处理或数字处理

本A51汇编器宏处理的另一个有用的特性是根据命令行参数或汇编符号进行条件汇编代码段的条件汇编能帮助你实现最紧凑的代码

它也让你可以从一个汇编源代码文件产生不同的应用

列表文件

下面是汇编器产生的列表文件的例子



```
A51 MACRO ASSEMBLER  Test Program        07/01/99 08:00:00 PAGE     1

DOS MACRO ASSEMBLER A51 V6.00
OBJECT MODULE PLACED IN SAMPLE.OBJ
ASSEMBLER INVOKED BY: C:\KEIL\C51\BIN\A51.EXE SAMPLE.A51 XREF

LOC  OBJ       LINE  SOURCE
                  1  $TITLE ('Test Program')
                  2  NAME    SAMPLE
                  3
                  4  EXTRN CODE (PUT_CRLF, PUTSTRING2, InitSerial)
                  5  PUBLIC  TXTBIT
                  6
                  7  PROG    SEGMENT         CODE
                  8  CONST   SEGMENT         CODE
                  9  BITVAR  SEGMENT         BIT
                 10
----              11          CSEG  AT     0
                 12
0000 020000   F  13  Reset:  JMP   Start
                 14
----             15          RSEG  PROG
                 16   ; *****
0000 120000   F  17  Start:  CALL  InitSerial ;Init Serial Interface
                 18
                 19  ; This is the main program. It is an endless
                 20  ; loop which displays a text on the console.
0003 C200     F  21          CLR   TXTBIT        ; read from CODE
0005 900000   F  22  Repeat: MOV   DPTR,#TXT
0008 120000   F  23          CALL  PUTSTRING2
000B 120000   F  24          CALL  PUT_CRLF
000E 80F5        25          SJMP  Repeat
                 26  ;
----             27          RSEG  CONST
0000 54455354    28  TXT:    DB    'TEST PROGRAM',0DH
0004 2050524F
0008 4752414D
000C 0D
                 29
                 30
                 31
----             32          RSEG  BITVAR  ; TXTBIT=0 read from CODE
0000             33  TXTBIT: DBIT  1       ; TXTBIT=1 read from IDATA
                 34
                 35          END

IREF SYMBOL TABLE LISTING
---- ------ ----- -------

N A M E             T Y P E  V A L U E   ATTRIBUTES / REFERENCES

BITVAR . . . . . .  B SEG    0001H       REL=UNIT   9# 32
CONST. . . . . .  C SEG    000DH       REL=UNIT   8# 27
INITSERIAL . . . .  C ADDR   -----       EXT   4# 17
PROG . . . . . .  C SEG    0010H       REL=UNIT   7# 15
PUTSTRING. . . . .  C ADDR   -----       EXT   4# 23
PUT_CRLF . . . . .  C ADDR   -----       EXT   4# 24
REPEAT . . . . . .  C ADDR   0005H   R   SEG=PROG   22# 25
RESET. . . . . .  C ADDR   0000H   A     13#
SAMPLE . . . . . .  N NUMB   -----       2
START. . . . . .  C ADDR   0000H   R   SEG=PROG   13 17#
TXT. . . . . .  C ADDR   0000H   R   SEG=CONST   22 28#
TXTBIT . . . . . .  B ADDR   0000H.0 R   SEG=BITVAR   5 5 21 33#

REGISTER BANK(S) USED: 0

ASSEMBLY COMPLETE.  0 WARNING(S), 0 ERROR(S)
```

A51 produces a listing file with line numbers as well as the time and date of the translation. Information about assembler invocation and the object file generated is printed.

Typical programs start with EXTERN, PUBLIC, and SEGMENT directives.

The listing contains a source line number and the object code generated by each source line.

Error messages and warning messages are included in the listing file. The position of each error is clearly marked.

Enable under µVision2 Options for Target – Listing – Cross Reference to get a detailed listing of all symbols used in the assembler source file.

The register banks used, and the total number of warnings and errors are at the end of the listing file.

A51汇编器产生一个列表文件包括行号汇编时的时间和日期关于汇编器运行和目标文件产生的信息被记录下来

通常情况下程序从EXTERNPUBLIC and SEGMENT 指示器开始列表文件包含了每个源代码的行号及每行产生的代码

列表文件包含了错误和告警信息错误和告警的位置被明显的标识出来

选择在Vision2-Options for Target – Listing 中的Cross Reference 选项将在列表文件列出源程序中所用到的所有字符

存储器组的占用信息和程序中的错误和告警总数包括在文件的结尾处.

BL51具有代码分段功能的连接/重定位器

本BL51是具有代码分段功能的连接/重定位器它组合一个或多个目标模块成一个8051的执行程序此连接器处理外部和全局数据并将可重定位的段分配到固定的地址上

本BL51连接器处理由Keil C51编译器A51汇编器和Intel PL/M-51编译器ASM-51汇编器产生的目标模块连接器自动选择适当的运行库并连接那些用到的模块

你也可以在命令行上输入相应的目标模块的名字的组合来运行本连接器BL51连接器默认的控制

参数是经过仔细选择的因而不需要定义附加的控制参数就可以适应大多数应用当然你可以很容易为你的应用作特定的设置

数据地址管理

本连接器通过覆盖那些不会同时使用的函数变量的技术来管理8051有限的内部存储器资源这极大地降低了大多数应用对存储器的需求本BL51连接器分析函数间的引用以决定存储的覆盖策略你可以用OVERLAY指示器来人为控制函数间的引用这些引用被连接器用来确定哪些存储器单元是独占的NOOVERLAY指示器让BL51不进行覆盖连接这在使用间接调用的函数时或为了调试而禁止覆盖时比较有用.

代码分段

本BL51连接器支持创建程序空间大于64KB的应用既然8051不能直接操作大于64KB的代码地址空间就必须由外部硬件来交换代码段完成此功能的硬件必须要在8051中运行的程序的控制中这就是大家所知的段(块)切换

本BL51连接器让你管理一个公共的区域和32个最大64KB空间的块从而达到总共2MB的分段程序空间支持外部硬件块切换的软件包括的一个汇编程序可以由你来编辑以适应你应用中的特定硬件平台

本BL51连接器让你定义哪个段装载哪个特定的程序模块通过仔细考虑把各个函数分配到不同的段中来创建一个非常大而有效的应用

公共段

段切换程序中的公共段是一块在任何时候在所有的段中都可以访问的存储器此公共段在物理上就不能切换出局或变换地址空间

在公共段中的代码可以复制到每个段中如果切换整个程序空间或驻留在一个独立的地址空间或器件中公共段不用切换

公共段包含那些必须在所有时候都要用到的程序段和常量它还可以包括经常使用的代码默认情况下以下的代码内容将自动分配在公共段中

复位和中断向量
代码常量
C51 中断服务进程
分段开关跳转表
一些C51实时运行库函数

执行其它段中的程序

分段代码空间是通过附加的由软件控制的地址线控制的这些地址线可以是由8051的I/O口或位于存储器空间的锁存器来模拟本BL51连接器为位于其他段中的函数生成一个跳转表当你用C语言调用一个位于不同的段中的函数时它要先切换段再跳到目标程序运行完成后再回到调用的那个段中去并继续往下执行这种段切换处理需要附加的50个CPU指令周期和占用2Bytes堆栈空间

如果你把相关的函数分配在相同的段中将显著的提高系统的性能那些需要从多个段中经常调用的函数应该位于公共段中

映象文件

下面是BL51产生的一个例子文件:

```
BL51 BANKED LINKER/LOCATER V4.00        07/01/99  08:00:00  PAGE 1

MS-DOS BL51 BANKED LINKER/LOCATER V4.00, INVOKED BY:
C:\KEIL\C51\BIN\BL51.EXE SAMPLE.OBJ

MEMORY MODEL: SMALL

INPUT MODULES INCLUDED:
  SAMPLE.OBJ (SAMPLE)
  C:\C51\LIB\C51S.LIB (?C_STARTUP)
  C:\C51\LIB\C51S.LIB (PUTCHAR)
  C:\C51\LIB\C51S.LIB (GETCHAR)
  C:\C51\LIB\C51S.LIB (TOUPPER)
  C:\C51\LIB\C51S.LIB (_GETKEY)


LINK MAP OF MODULE:  SAMPLE (SAMPLE)

     TYPE    BASE      LENGTH    RELOCATION   SEGMENT NAME
     -----------------------------------------------------

     *******   D A T A   M E M O R Y   *******
     REG     0000H     0008H     ABSOLUTE     "REG BANK 0"
     DATA    0008H     0001H     UNIT         ?DT?GETCHAR
     DATA    0009H     0001H     UNIT           DATA GROUP
             000AH     0016H                  *** GAP ***
     BIT     0020H.0   0000H.1   UNIT         ?BI?GETCHAR
             0020H.1   0000H.7                *** GAP ***
     IDATA   0021H     0001H     UNIT         ?STACK

     *******   C O D E   M E M O R Y   *******
     CODE    0000H     0003H     ABSOLUTE
     CODE    0003H     0021H     UNIT         ?PR?MAIN?SAMPLE
     CODE    0024H     0006H     UNIT         ?C_C51STARTUP
     CODE    002AH     002DH     UNIT         ?PR?PUTCHAR?PUTCHAR
     CODE    0057H     0011H     UNIT         ?PR?GETCHAR?GETCHAR
     CODE    0068H     0018H     UNIT         ?PR?_TOUPPER?TOUPPER
     CODE    0080H     000AH     UNIT         ?PR?_GETKEY?_GETKEY


OVERLAY MAP OF MODULE:   SAMPLE (SAMPLE)

SEGMENT                            DATA GROUP
  +--> CALLED SEGMENT            START    LENGTH
-----------------------------------------------
?C_C51STARTUP                    -----    -----
  +--> ?PR?MAIN?SAMPLE

?PR?MAIN?SAMPLE                  0009H    0001H
  +--> ?PR?GETCHAR?GETCHAR
  +--> ?PR?_TOUPPER?TOUPPER
  +--> ?PR?PUTCHAR?PUTCHAR

?PR?GETCHAR?GETCHAR              -----    -----
  +--> ?PR?_GETKEY?_GETKEY
  +--> ?PR?PUTCHAR?PUTCHAR


LINK/LOCATE RUN COMPLETE.  0 WARNING(S),  0 ERROR(S)
```

BL51 produces a MAP file (extension .M51) with date and time of the link/locate run.

BL51 displays the invocation line and the memory model.

Each input module and the library modules included in the application are listed.

The memory map contains the usage of the physical 8051 memory.

The overlay-map displays the structure of the program and the location of the bit and data segments of each function.

Warning messages and error messages are listed at the end of the MAP file. These may point to possible problems encountered during the link/locate run.

BL51产生一个包含连接时的时间和日期的映象文件(*.M51)
BL51 显示调用的命令和存储模式
应用中包含的每个模块和库模块被列出来
存储器映象文件包含8051实际存储器的使用信息
覆盖映象图显示了程序结构和每个函数的数据和位段
错误和告警总数包括在文件的结尾处这些映象图指出在连接定位时可能面临的问题

LIB51库管理器

本库管理器让你建立和维护库文件一个库文件是格式化的目标模块由编译器或汇编器产生的集合库文件提供了一个方便的方法来组合和使用大量的连接程序可能用到的目标模块利用uVision2 项目管理器的Options for Target– Output – Create Library选项可以建造一个库你也可以从命令行运行LIB51程序命令行参数参照218页"LIB51 / L251 Library Manager Commands

使用库有一系列优点安全高速和减少磁盘空间仅仅是使用库的一小部分原因另外库提供了一个好的分发大量函数而不用分发大量函数源代码的手段例如ANSI C的库是作为一套库文件提供的

uVision2 项目C:\KEIL\C51\RTX_TINY\RTX_TINY.UV2允许你修改和创建RTX51小型实时操作系统库你很容易创建你自己的库用来包括象串行I/OCAN和闪存操作这样一些你自己一次又一次要用到的流程一旦这些流程调试无误后你就可以把它们转换成库由于库只包含目标模块不用在每个项目中重新编译这些模块所以生成应用的时间将缩短

连接定位程序连接最终应用是用到的库文件库中的模块仅仅在需要的时候才被提取加到程序中没有被你的应用调用的库函数不会出现在你的最终结果中连接器把从库中提取出来的模块和其他目标模块做同样的处理

OC51 分段目标文件转换器

此OC51转换器为一个分段目标模块中的每一个代码段创建绝对的目标模块分段目标模块是你生成一个分段代码切换应用时由BL51创建的字符变量的调试信息被拷贝到转换后的绝对目标模块中以便给uVision2调试器和其他仿真器使用

你可以从命令行用OC51去为你分段目标模块中的每一个代码段创建绝对目标模块
然后你还可以用OH51目标代码到HEX文件的转换器为每一个绝对目标模块产生相应的Intel HEX格式的文件

OH51 目标代码到HEX文件的转换器

此转换器为绝对目标模块创建Intel HEX格式的文件绝对目标模块可以由BL51或OC51产生 Intel HEX文件是ASCII文件它用十六进制的数表示你的应用系统的目标模块它们可以很容易的下载到编程器以便写入EPROMS器件

第4章　创建应用

我们提供了一个测试版本测试版本中包含示例程序和我们工具的受限使用版本从而使你很容易的评估熟悉我们的系列产品测试版本中的示例程序同样包含在正式版本中

注意
Keil C51测试版本在功能和你能够创建的应用的代码长度上都有限制关于这点的详细信息请参考版本发行说明要创建大的应用系统你必须购买其中一个开发包关于开发包套件的详细描述参考P16的产品一览

本章描述了uVision2的创建模式并向你演示如何使用它来创建一个简单的程序(译者注原文为示例程序)以及生成和维护项目的一些选项包括文件输出选项C51编译器的关于代码优化的配置uVision2项目管理器的特性等

创建项目

uVision2包括一个项目管理器它可以使你的8051应用系统设计变得简单要创建一个应用你需要按下列步骤进行操作

启动uVision2新建一个项目文件并从器件库中选择一个器件
新建一个源文件并把它加入到项目中
增加并配置你选择的器件的启动代码
针对目标硬件设置工具选项
编译项目并生成可以编程PROM的HEX文件

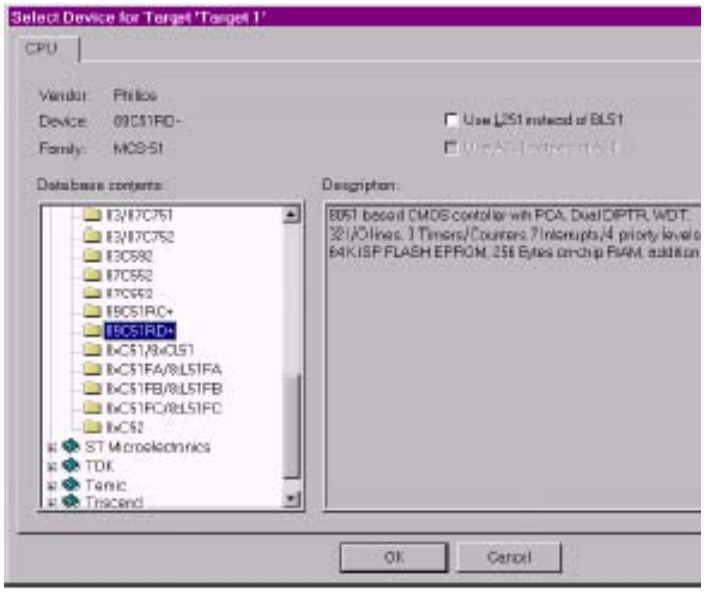
这里将一步一步的进行描述从而指引你如何去创建一个简单的uVision2项目

启动uVision2并创建一个项目

uVision2是一个标准Windows应用程序直接点击程序图标就可以启动它要新建一个项目文件从uVision2的Project菜单中选择New Project这将打开一个标准的Windows对话框此对话框要求你输入项目文件名

我们建议你为每个项目建一个单独的文件夹你可以在弹出的对话框中点击新建文件夹的图标来得到一个空的文件夹然后选择子文件夹并键入项目的名称如Project1uVision2将创建一个文件名为Project1.UV2的新项目文件新的项目文件包含了一个以默认的文件名命名的目标和文件组你可以在项**Project Window – Files**看到这些名字

现在从菜单Project – Select Device for Target为你的项目选择一个CPU弹出的对话框中显示的是器件数据库你只要选择所需要的MCU就可以了在例子程序中我们选择Philips 80C51RD + CPU该选择就为80C51RD+器件设置了工具选项这种方式简化了工具的配置



注意:

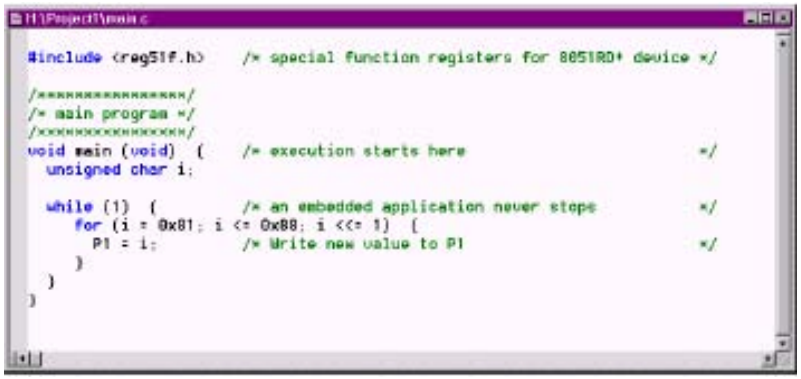
对于一些器件uVision2环境需要你手工输入一些附加的参数仔细阅读此选择器件对话框中Description下面的信息它可能提供了你所选择器件的一些附加的说明

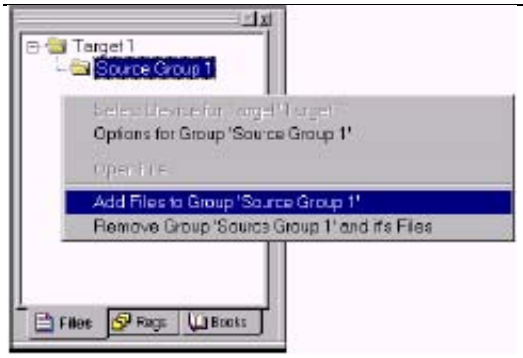
一旦你从器件库中选择了一个CPU你就可以在项目窗口的Books页打开此CPU的用户手册这些用户手册是Keil开发工具光盘中的一部分

新建一个源文件

你可以用菜单选项File-New来新建一个源文件这将打开一个空的编辑窗口让你输入你的源代码当你把此文件另存为*.C的文件后uVision2将高亮显示C语言语法字符我们把我们的例子程序保存为MAIN.C

一旦你创建了源文件你就可以把它加入到你的项目中uVision2提供了几种手段让你把源文件加入到项目中例如你可以右击Project窗口– Files页中的文件组来弹出快捷菜单菜单中的Add Files选项打开一个标准的文件对话框从对话框中选择你刚刚生成的文件MAIN.C



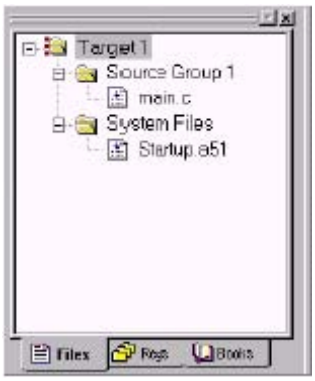


增加和配置启动代码.

文件STARTUP.A51是大多数不同的8051CPU准备的启动代码启动代码清除数据存储器并初始化硬件和重入函数堆栈指针另外一些8051派生产品要求初始化CPU来迎合你设计中的相应的硬件例如,Philips 8051RD+提供的作业xdata RAM应该在启动代码中启用假如你需要修改启动文件来迎合你的目标硬件你应该把文件STARTUP.A51复制一份到你的项目文件夹中

为你选择的CPU的配置文件创建一个文件组是一个良好的习惯用菜单**Project – Targets, Groups, Files** 打开对话框来添加一个名为System Files的文件组到你的目标中也在此对话框中你用**Add Files to Group**按钮把文件STARTUP.A51添加到你的项目中

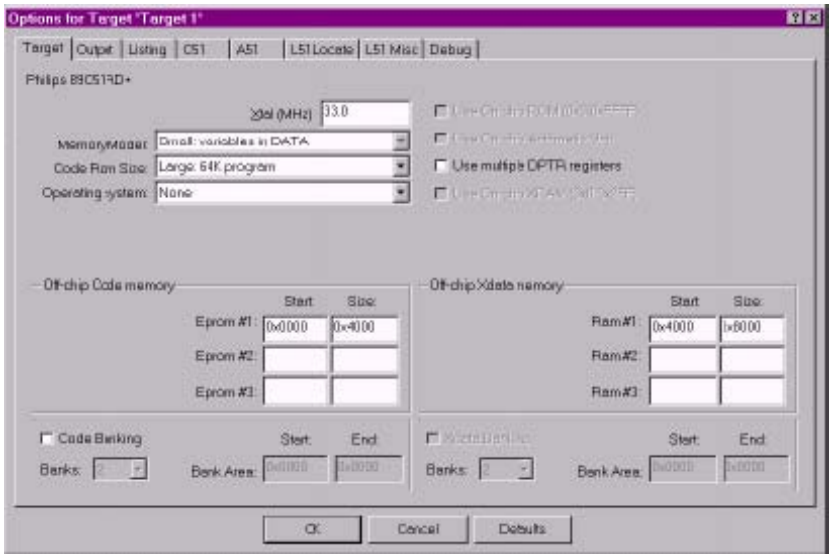
项目窗口的文件页列出了你项目的所有条目.



现在,你的uVision2的**Project Window – Files**应该显示上图中的文件结构在项目窗口中双击文件名STARTUP.A51你就可以把它在编辑器中打开然后按照P197页的第十章(CPU and C Startup Code)的描述来配置启动代码 如果你使用你所选择器件的片上RAM在启动代码中的设置必须匹配**Options – Target**对话框中的设置下面来讨论**Options – Target**对话框

为目标设置工具选项

uVision2 允许你为你的目标硬件设置选项Options for Target对话框可以通过工具条图标打开在目标的各个页中你可以定义和你的目标硬件及你所选器件的片上元件相关的所有参数下图显示了我们例子的设置



下表描述了目标对话框的一些选项

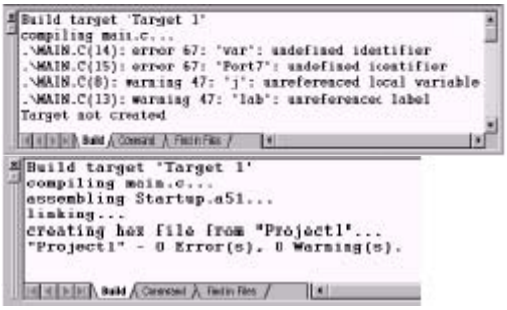
对话框条目	描述
Xtal	定义CPU时钟对于大多数应用中和实际的XTAL频率相同
Memory Model	定义编译器的存储模式对于一个新的应用默认的是SMALL模式参照P78 存储模式和存储器类型的讨论
Allocate On-chip Use multiple DPTR registers	定义在启动代码中使能的片上元器件的使用如果你使用片上xdata RAM那你应该在文件STARTUP.A51中使能XRAM的访问
Off-chip Memory	在此定义你目标硬件上所有的外部存储器区域
Code Banking Xdata Banking	为代码和数据的分段Banking定义参数详细信息参照P67 代码分段Code Banking

注意

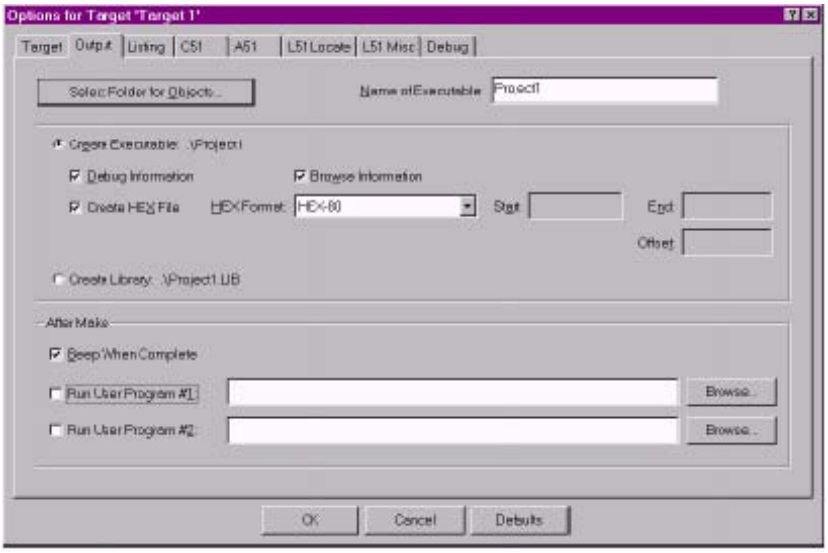
有些选项只有在你使用*LX51*连接器时才有用*LX51*连接器只在*PK51*中提供.

Build项目并生成HEX文件

通常情况下在Options – Target对话框中的设置已经足够使你开始一个新的应用通过单击工具条上的Build目标的图标你可以编译所有的源文件并生成应用当你的应用中有语法错误时uVision2将在Output Window – Build页显示这些错误和告警信息双击一个信息将打开此信息对应的文件并定位到语法错误处



一旦你成功的生成了你的应用你就可以开始调试了uVision2调试器功能的讲述请参照P93页Chapter 5的Testing Programs"在你调试完你的应用后需要创建一个HEX文件来烧写子或软件模拟当**Options for Target** – **Output**中的输出HEX文件使能时uVision2每进行一次Build都生成HEX文件如果定义了**Options for Target** – **Output**中的**Run User Program #1** 选项时在生成操作完成后将自动运行此处定义的操作如编程PROM器件



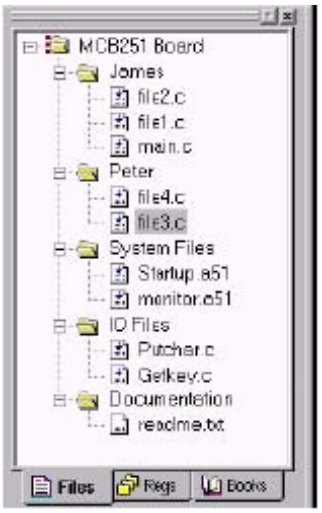
现在你能够修改已经存在的源文件或添加一个新的源文件到项目中Build目标工具条按钮只编译修改过的或新加进来的文件然后生成执行文件uVision2维护一个文件包含清单从而知道某个源文件用到的所有的包含文件而且工具配置的选项也被保存在此清单中所以uVision2能够只编译那些需要重新编译的文件利用Rebuild All Target(原文为Rebuild Target)命令所有的文件都被重新编译而不论是否被修改过

项目目标和文件组

通过使用不同的项目目标uVision2允许你为一个应用创建几个不同的程序你也许需要一个目标用来测试另一个目标是你应用系统的发行版本在同一个项目文件中允许每个目标进行独立的工具设置

文件组是用来让你把项目中相关的文件放在一组这在要求把文件按功能块组织和确定你的软件开发团体中的工程师们的职责时特别有用我们已经在本例子中使用了文件组去分离与CPU相联系的文件和其它文件利用这个技术可以轻松的管理拥有几百个文件的复杂项目

Project – Targets, Groups, Files 对话框允许你创建项目目标和文件组我们已经用此对话框增加了系统配置文件下图中显示了一个例子项目的结构



项目窗口向你展示所有的组和相联系的文件文件按照显示的顺序编译和连接你可以通过拖放来移动文件的位置你也可以单击目标或组的名字去修改它右击将打开弹出菜单弹出菜单可以让你

1)设置工具选项
2)删除条目
3)添加文件到组

4)打开文件

在Build工具条上你可以快速改变当前的目标

浏览项目窗口中的文件和文件组的属性

在项目窗口的文件页中用不同的图标来标识不同的文件和文件组属性下面是这些图标及其对应的属性的描述

此图标是在文件图标上加一箭头而成用来表示被编译连接到项目中去的文件

文件图标用来表示不被连接到项目中去的文件典型的如文档文件另外在文件的属性窗口中取消"**Include in Target Build**"复选项,将使该文件剔除出项目剔除出项目的文件也用此图标参照87页的文件和文件组的详细选项 – 属性对话框

图标上有一把钥匙用来表示只读文件典型的如被软件版本控制系统(SVCS)控制的文件因为SVCS把他所控制的文件的本地拷贝设置为只读属性参照76页的"使用SVCS菜单"

此图标左边有三个点用来表示有特殊选项的文件(图4)或文件组(图5)参照87页的文件和文件组的详细选项 – 属性对话框

注意

不同的图标让你能够快速浏览到一个项目不同的目标中的工具设置
你所看到的图标总是反映当前所选目标的属性例如你在一个目标中对一个文件或文件组设置了特殊选项那只有在你选择了此目标时那些你设置了特殊选项的文件的图标上才会有三个点

/**********译者注-开始**********/

第二个图标为文件图标所有属性文件的图标都以此为基础再加上箭头钥匙三点中的零到三个来表

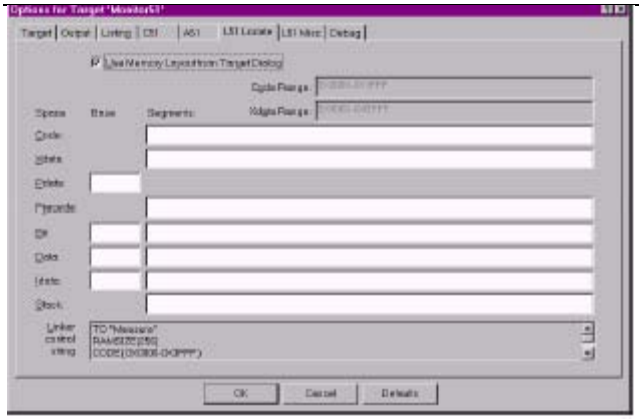
示文件对于所在目标的属性更进一步说只读是属于文件自己的即一个文件具有只读属性那它在任何目标中都具有但是否包含在项目中是否设置了特殊选项,是文件对于目标的属性即在一个目标中的一个文件的图标上有箭头和(或)三点,在另一个目标中并不一定如此
/**********译者注-结束**********/

浏览配置对话框

此选项对话框让你设置所有的工具选项通过**Project Window – Files**中的弹出式菜单你可以为文件组或者一个单独的文件设置不同的选项在此处的描述中你仅仅能够获得对话框的相应的标签页利用对话框中的帮助按钮你可以获得大多数对话框条目的帮助下表描述了目标对话框的选项

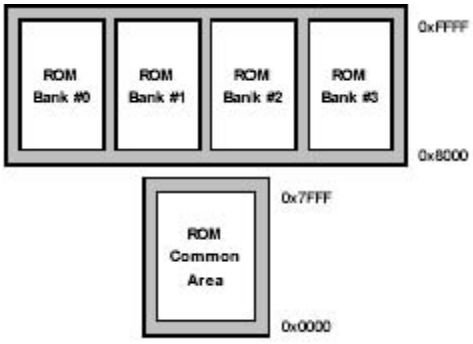
对话框的标签页	描述
Target	定义你应用的硬件详细信息参见第62页
Output	定义Keil工具的输出文件并让你定义生成处理后执行的用户程序更多信息参见P82
Listing	定义Keil工具输出的所有列表文件
C51	设置C51编译器的特别的工具选项如代码优化或变量分配更多信息参见P80
A51, AX51	设置汇编器的特别的工具选项如宏处理
L51 Locate LX51 Locate	定义不同类型的存储器和存储器的不同段的位置典型情况下你可以选择**"Memory Layout from Target Dialog"**来获得自动设置如下图所示更多信息参见P86
L51 Misc LX51 Misc	其它的与连接器相关的设置如告警或存储器指示
Debug	Vision2 Debugger的设置更多信息参见P101
Properties	文件和文件组的文件信息和特别选项参照P87的**"File and Group Specific Options – Properties Dialog"**

在L51标签页中一旦你使能**Use Memory Layout from Target Dialog**选项uVision2使用从你所选择器件和目标标签页中所得到的存储器信息你还可以添加附加的段到这些设置中



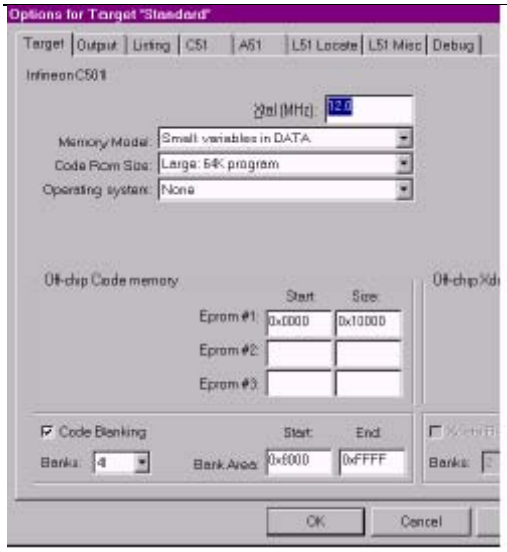
代码分块Code Banking

一个标准的8051器件能寻址64KB的代码空间为了突破64KB程序空间的限制Keil 8051工具支持代码分块这个技术让你管理一个公共的区域和32个每个最大达64KB的块,从而达到总共2MB的代码切换空间



例如你的硬件设计可以包括一个位于0x0000-0x7fff空间的32KB的ROM如你所知的公共区域或公共ROM和四块位于0x8000-0xffff空间的32KB的ROM如你所知的代码块ROM通常情况下代码块通过端口线选择右图显示了此应用的存储器结构

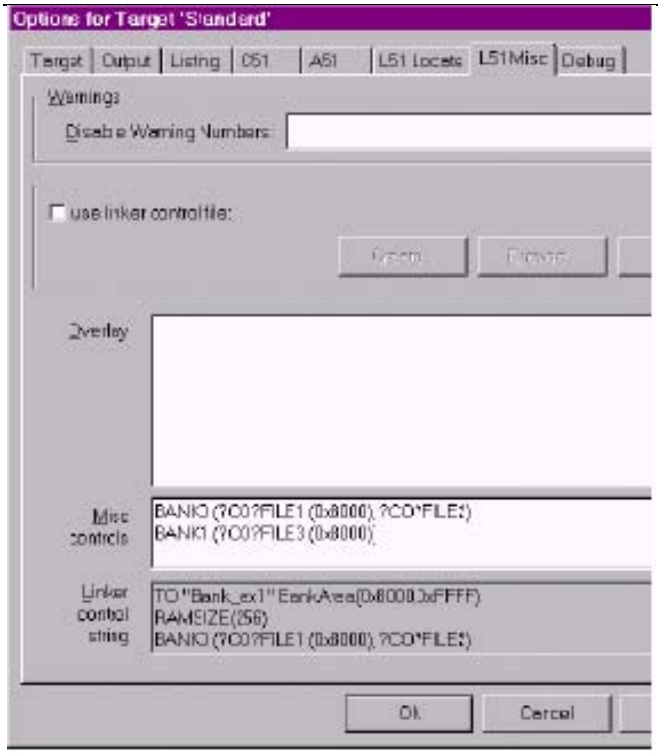
代码分段在**Options for Target – Target**标签页中使能和配置在此对话框中输入你硬件支持的代码分块的个数和分块的区域上例中的存储器分块的配置如右图所示

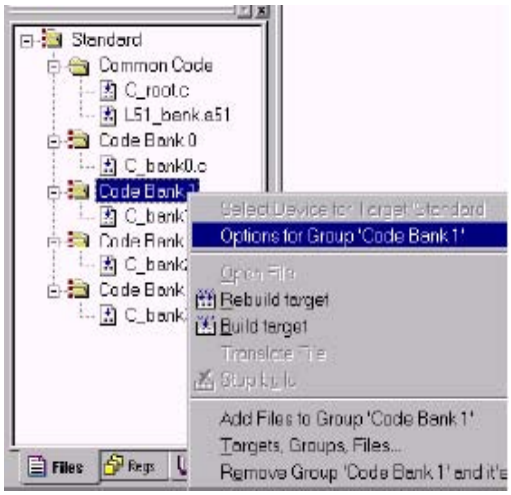


为了配置你的分块的硬件你需要将文件**C51\LIB\L51_BANK.A51**加入到你的项目中复制此文件到你的项目文件夹把它和你项目中的其它源文件放在一起并添加它到你项目的一个文件组文件**C51\LIB\L51_BANK.A51**必须修改以便迎合你的目标硬件

每个文件或文件组通过其**Options – Properties**对话框可以被定义到一个特定的代码块中

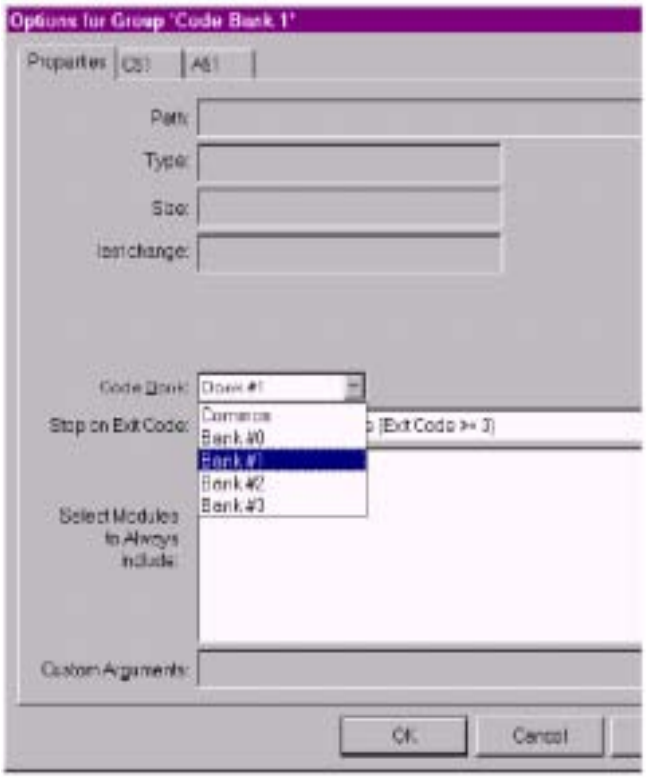
来打开此文件或文件组的**Options – Properties**对话框此对话框允许你选择哪个代码块或公共区域





公

进程和数据常量如中断进程中断和复位向量字符串常量和块切换进程所以连接器只把一个模块中程序段定位到相应的块区域中数据常量放在公共区域中如果你能够确定某些常量段的信息只被某个特定的代码块访问你可以在**Options forarget – L51 Misc**中用**BANKx**来指示连接器把这些常量段定位到此特定的代码块中　　以上的步骤完成了你的代



的配置uV

块从而让你能够调试程序　如果你在**Options for Target –Outpt**对话框中使能**"Create E**

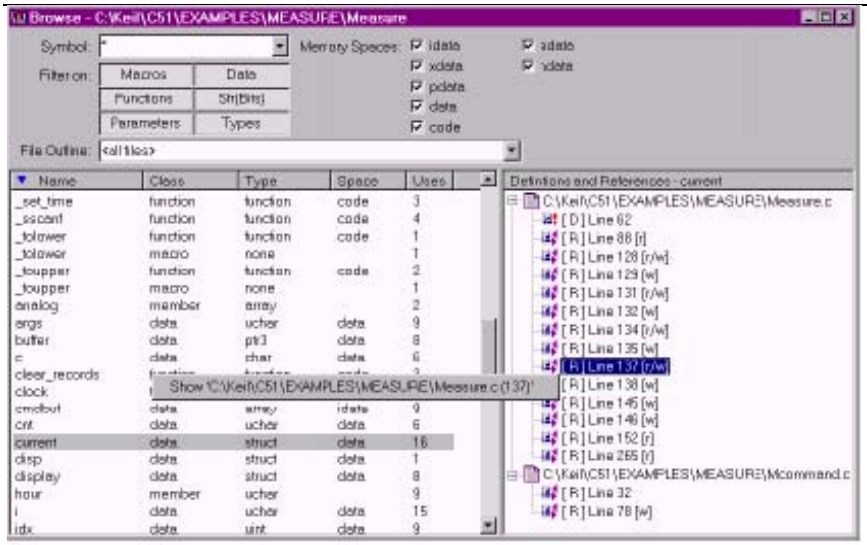
送uVision2将为每个代码块生成 一个从地址0开始的64KB的物理映像你需要EPROM中相应的存储空间中去

uVision2包含许多强大


菜单**Edit – Find**

编辑器将定位到匹配此字符串的文本行

单iew – Source Browser去打开资源浏览器窗口



此浏览

瓣

资

S

#匹配数字0-9

$ 匹配任意字符

* 匹配任意字符串

选择需要显示的符号的类型

选择一个文件显示此文件中的符号资源

Memory Sp 定义需要显示的符号的存储类

下

代码

#

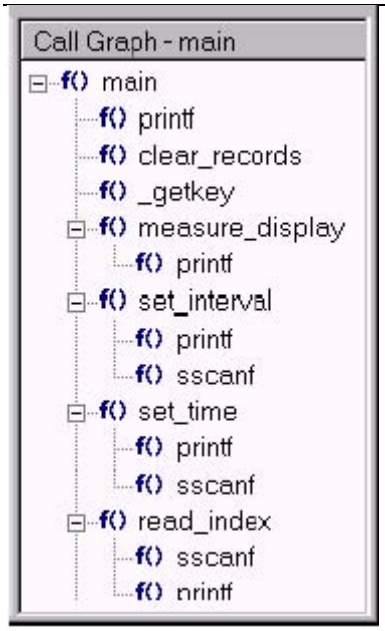
代表的符号为下划线开头接字母a,任一字母任一数字

可加任何字符

_*ABC 下划线加任意字符或不加然后接ABC

右击资 浏览器Definitions and references框中显

定



容对于函数你还可以看到调用和被调用的关系图在此Definitions ad references框中还为你提供了一些附加的信息　　符号

[D]

[R]

[r]

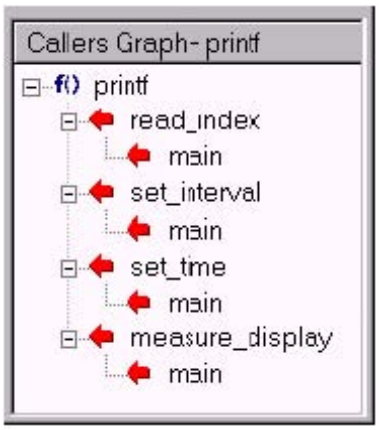
[w]

r/w]　/写操作处

[&]　　址引用处

目

　　快捷键



F

Shift+F1　　跳转到

Ctrl+　　跳转到下一处的引用或定义处

Ctrl+Num– 跳转到前一处的引用或定义处

开

一个键序列可以用来从uVision

参数传递中一个键序列是键码和文件码的组合下面列出有效的键码和文件码 键码 定义由文件码所确定的文件的路径

%

名如C:\MYPROJECT\PROJECT

% 包含扩展名的文件名,是不包含路径:PROJECT1.UV2

@ 不包含路径和扩展名的文件名如PROJECT1

$ 文件夹名如C:\MYPROJECT

~ 当前光标所在位置的行号仅仅在文件码为F时效

^ 当前光标所在位置的列号仅仅在文

码 文件码为F时有效要在用户程序的命令行使用$, #, , @,

#

文件码

F 回项目文件如果选择的文件组名则返回当

P 中的文件 当前项目名如PROJECT.UV2 连接器输出文

H HEX 应用文件PROJECT1.H86

uVision2 可执行文件如C:\KEIL\UV2\UV2.EXE

以下是应用在

件码 定义插入在用户程序命令行上的文件名或参数

Q

R 持修正码的字符串

C 为SVCS系统保持校验码的字符串

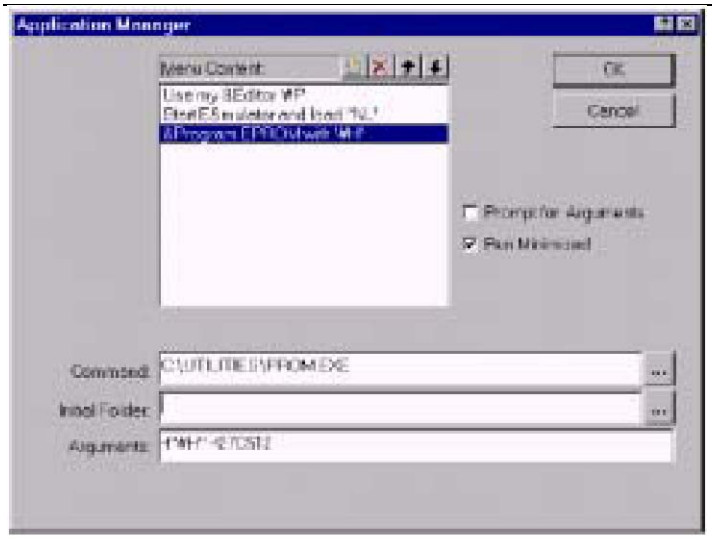
U 用户名在**SVCS-Configure Version Co**

V 文件名在**SVCS-Configure Version**

Q, U 和 V能和键码%组合使用

使用工具

通过工具菜单



菜单可以通过菜单**Tools – Customizeools Menu...**打开对话框在此对话框中配置外部用户应用的参数右边的对话框示例了一个工具设置　上图所示的话
打

此

对话框条目	描述
Menu Content	工具菜单上显示的文本。其中的每行都可以包括键码和文件码。用与号(&)来定义一个快捷键。你可以定义当前选择的菜单行的在下面列出的各个选项。

对话框条目
Menu Con

前选择

项 如果选中那么在你点击此菜单时一个对话框将弹出提示你输入用户程序的

Run Minimize Command	如果选中将使此应用程序运行时的窗口最小化 输入你点击此菜单时运
Initial Folder	应用程序的当前工作目录如果为空uVision2
Arguments	传递给应用程序的命令行参数
基于应用	的命令行输出被复制到一临时文件当此应用执行完成后此临时文
件的内容将	**tput Window -Build**页列出

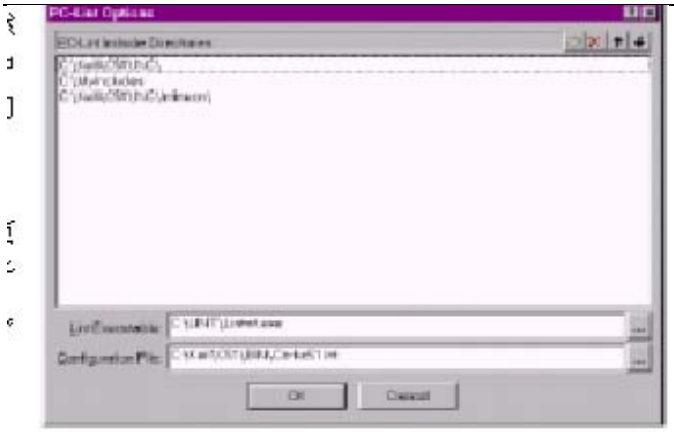
运

Gimpel Software

标

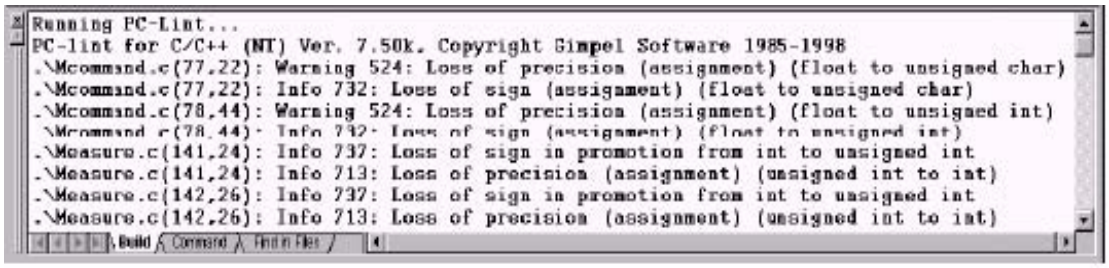
显著地减少你花费在目标应用上的调试精力　　在你的PC机上安装**PC-Lint**然后在**Tools**

输入参数本例显示了一个典型的PC-Lint的配置　　为了在**Ouu**



(lint)你的源代码了、菜单 **Tools － Lint** 运行 **PC-Lint**
得到正确
KEIL\C51\BIN 文件夹中的配置文件

在安装
来检查当前
中所有的C源文件**PC-Lint**的输出信息被重定向到**Output Window-Build**页双击**PC-Lint**的输出信息将使编辑器定位到相应的位置



的

置

Siemens Easy-Case

Vision2
文档编辑器你可以
Easy-Case
安装 **asy-Case:** 为了在 Easy-Case 中使用 Visio 调试器命令文件
C:\KEIL\UV2\UV2EASYP
EASY-CP

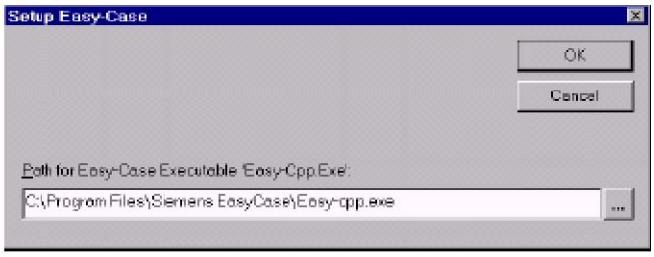
C:\>CD C:\WINNT C:\WINNT>COPY EASY-CPP.INI+C:\KEIL\UV2\UV2EASY-CPP.INI EASY-CPP.INI

在Vision2的**Tools**

Easy-Case对话框中输入

EASYCPP

了的配置

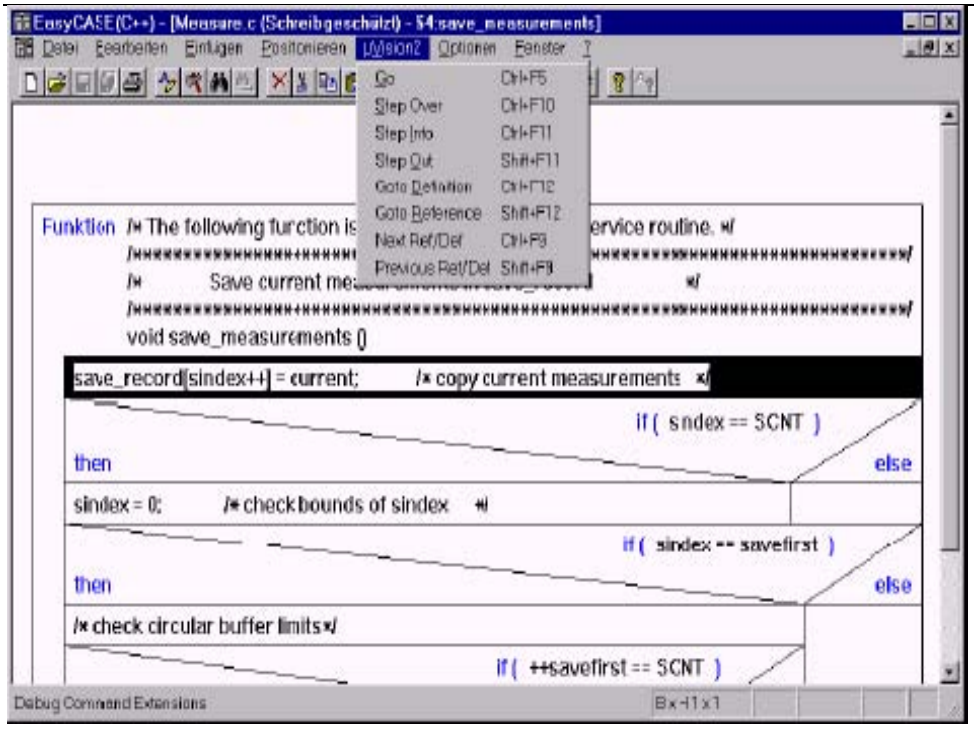


你可以用 **Tools** - **Start/Stop Easy-Case** 来启动… 在当前位置打开 µ Vison2 中活动编辑器中的文件。

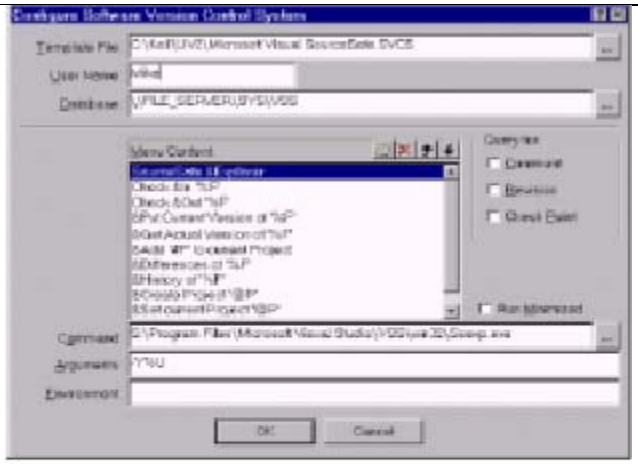
利用

E

E


EasyCASE(C++) - [Measure.c (Schreibgeschützt) - 54:save_measurements]
Datei Bearbeiten Einfügen Positionieren µVision2 Optionen Fenster ?
Go Ctrl+F5
Step Over Ctrl+F10
Step Into Ctrl+F11
Step Out Shift+F11
Goto Definition Ctrl+F12
Goto Reference Shift+F12
Next Ref/Def Ctrl+F9
Previous Ref/Def Shift+F9
Funktion
void save_measurements ()
save_record[sindex++] = current; /* copy current measurements */
if (sndex == SCNT)
then
else
sindex = 0; /* check bounds of sindex */
if (sindex == savefirst)
then
else
/* check circular buffer limits */
if (++savefirst == SCNT)
Debug Command Extensions

提供预先配置的模板



通过 SVCS 菜单，调用你的版本控制系统的命令行工具.SVCS 菜单的配置存储在一个模板文件中。

此菜单的配置通过 **SVCS – Customize SVCS Menu** 对话框进行. 此对话框的选项解释如下：

对话框条目	描述
Template File	SVCS 菜单配置文件的名字。推荐一个项目开发组的所有成员使用相同的模板文件。所以模板文件应该复制到文件服务器中。

控

配

此菜单的配置通**SVCS –Customize SVCS Menu**对话框行

此对话框的

对话框条目

Template

ser Name 在参数

Database SVCS系统使用的数据库的文件名或路径通过%V文件码传递 显示在SVC菜单中的文本行可以包括键码和文件码

快捷键对于当前被选中的菜单行你可以定义以下列出的选项

Query f 允许你在执行此SVCS命令前询问一些附加的信息注释将被复制到

Comment 个临时文件中此文件通过文件码%Q作为一个参数传递给SVCS命令版本和校验信息通过文件码%R和%C作为字符串传递

CheckPoint Run MinimCommaArgument 如果你想以最小化的窗口来执行应用使能此选项 当你点击此SVCS菜单项时将调用的程序文件

Environment 在执行此SVCS程序前需要设置的环境变量

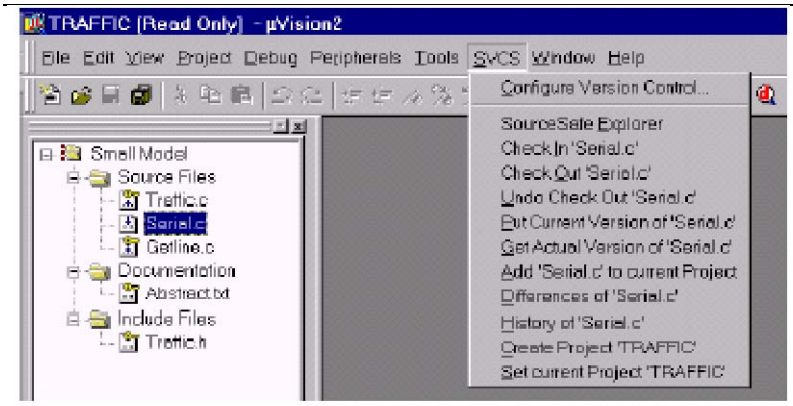
命令行SV 应用程序的输出被复制到一个临时文件中

件的内容被列 在**Output Window** – **Build** 页

如右图所示是一个

S

Window –



两个单独的文件。项目设置保存在***.UV2** 文件中：此文件将被指导编译生成应用代码。uVision2 的本地配置被保存在***.OPT** 文式设置。

SVCS参数目标名使用***.UV2**格式的项目名如你所见的文件的本地拷贝是一个只读文件所以它的图标上有一个钥匙样符号 uVisio项目被保存为SVCS查找并且可以用件

下表列出了典型的SVCS菜单项

其它增加的

SVCS菜单项	描述
Check Out	并把
ndo Check	取消Check Out操作
	Out
Put Current	把文件保存到SVCS的数据库中但对本地文件仍然可以修改
	Version
	Version
	Project ifferences
	History

你也许要

SSUSER环境变量使用工作站的登录名 *MKS*

这边是一个本地的发送邮箱形式的工作空间 *Intersolv PVCS* 在创建和维护项目上

编

工具设置都能产生良

存

照35页的存储模式

With Global Register Optimization	Without Global Register Optimization
main () { *unsigned char i;* *unsigned char a;* *while (1) {* *i = input ();*	*/* get number of values */*

?C0001:
LCALL input
;- 'i' assigned to 'R6' -
MOV R6,AR7
	?C0001:
	LCALL input
	MOV DPTR,#i
	MOV A,R7
	MOV @DPTR,A

do { *a = input ();*	*/* get input value */*

?C0005:
LCALL input
;- 'a' assigned to 'R7' -
MOV R5,AR7
	?C0005:
	LCALL input
	MOV DPTR,#a
	MOV A,R7
	MOVX @DPTR,A

output (a);	*/* output value */*
LCALL _output	LCALL _output
} while (--i);	*/* decrement values */*

DJNZ R6,?C0005	MOV DPTR,#i
	MOVX A,@DPTR
	DEC A
	MOVX @DPTR,A
	JNZ ?C0005

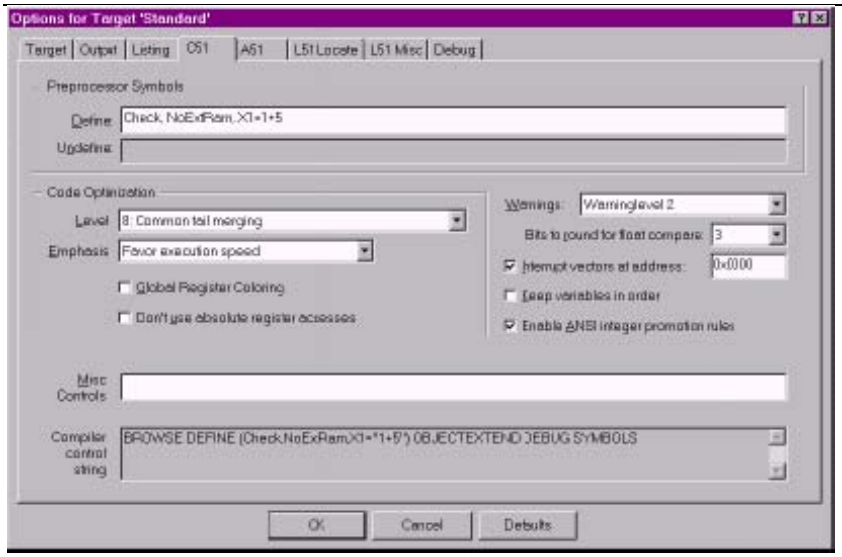
}	
SJMP ?C0001	SJMP ?C0001
}	
RET	RET
Code Size: 18 Bytes	**Code Size: 30 Bytes**

其它C51编译器指

还有好几个其它的C51编译参数能够改善代码质量这些参数的选择由**Options – C51**对话框提供在一个应用中你可以用不同的编译设置来编译C程序模块通过列表文件你可以比较由不同的编译设置编译生成的代码的质量

并

令



D't use absolute register accesses 禁止使用寄存器R0-R7的绝对地址访问由于C51不能使用ARx符号所以代码将有一点点增加比如用PUSH和POP指令时需要插入替代代码然而代码也将不依赖于寄存器
Interrupt vectors at
inter
String 的源文件的各个

8051 CPU数或长 是一个控制器用8位字节如char和unsigned char的操作比用型的操作要更有效

技

接下来的一节	你不会常用
入**uVision1**的	目到**uVision2**
你可以用下面的	来把在uVision1中创建的项目导入到uVision2中
步骤	isi
很重要的一点是	的uVision2项目必须创建在你已经存在的uVision1项目文件夹中
新	**t**
已经存在的uV	n1项目只有新项目的文件列表是空的时此菜单才是可用的
Options for Ta	**t-Target**对话框选项来定义你目标硬件的存储器结构一旦你这样做了**ns for Target – L51 Locate**对话框中的**UseM**
arget Dialog选项	并在此对话框中删除用户类型和用户区域的设置

注意 由于*uVision2*在许多方面和以的

换为

如
能复制到

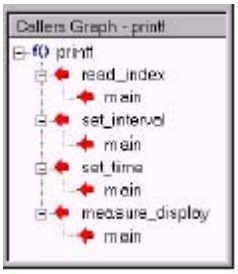
弹出一个提示对

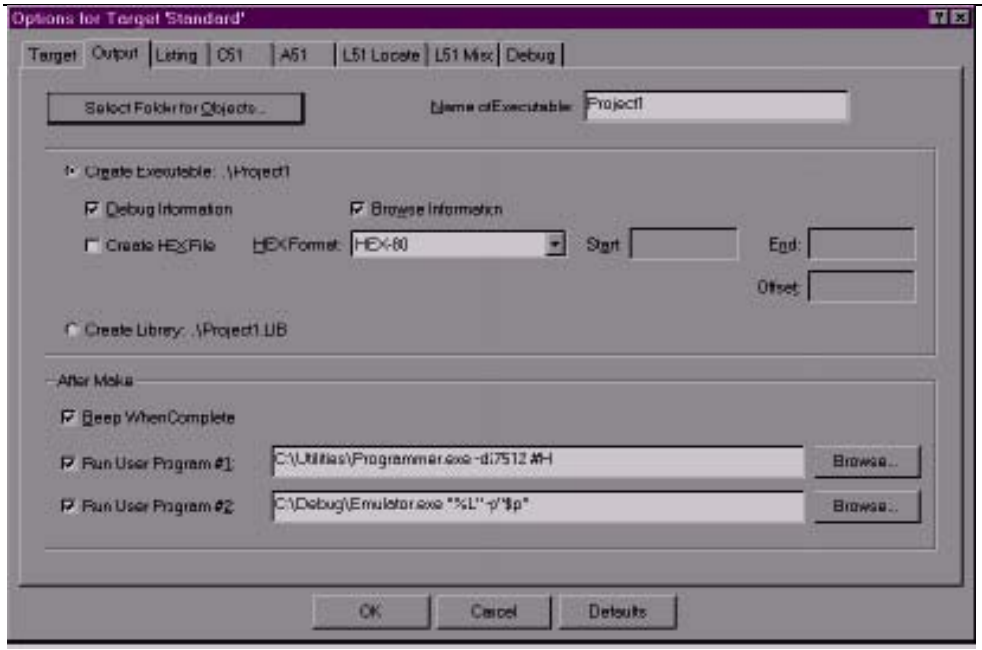
而从删除也许原文相关部分应翻译为并且此对话框中已经删除了用户类型和用户区域的设置

Build后运行外部程序
Options for Target – Output对话框允许你输入最多两个用户程序在Build成功后开始执行使用键序列码你可以从uVision2项目管理中传递参数到这些用户程序中

你可以在编辑器窗口中使用浏览器的信息选择你想要寻找的条目右击打开弹出式菜单或使用下面的快捷键

12 跳转到定义处跳转到引用处将光标置到符号引Ctrl+Num+





上例中**User Program #1**的调用参数是带绝对路径的Hex格式的输出文件如**C:\MYPROJECT\PROJECT1.HEXUser Program #**的用参数仅仅是连接器的输出文件**PROJECT1**的名不带后缀和参数**"-p"**表示的向项目的路径**:\MYPRJECT**如果你的文件夹名字中包含如~,#等特字

为列表文件和目标文件指定单独的文件夹

Options for Target –

目标文件视为有效即使你修改了你的项目目标Build Target命令也只是重新编译那些修改过的文件

提供同样的功能

用**uVisio** 器件库中没有列出的微控制

的

e

MODP2 Philips and Temic各种器件的双数据指针 MOD517DP Infineon C500列器件的多用途数据指针 MOD517AU Inineon C500系列器件的算术逻辑单元 MOD_CONT 使能支持Dallas 390 Contigious模式

创建一个库文件 在**Options for Target – Output**对话框中选择创建库文件uVision2将调用库管理器是连接/定位器

L51 Misc选项页的输入将被忽略同时在目标页中的CPU和储存器的设置也无关紧要

C

all Settings from Current Target从一个已经存在的目标复制工具设置到当前的目标的步骤如下

1

2选择你想要复制工具设置的目标为当前目标 能**Copy all Settings from Current Target**后再把刚刚删除的目标加进来

代码到绝对 器空间

需 代码段放在特定的存储器地

要放在0xC000地址处这个结构体在文件ALMCTRL.C中定义并且这个模块只包含此结构体的声明

unsigned ch

unsigned char status; }; struct alarm_st xdata alarm_control;

段

为的段放在xdata储存

*C51*提供关键词*_at_*和一些能够存取

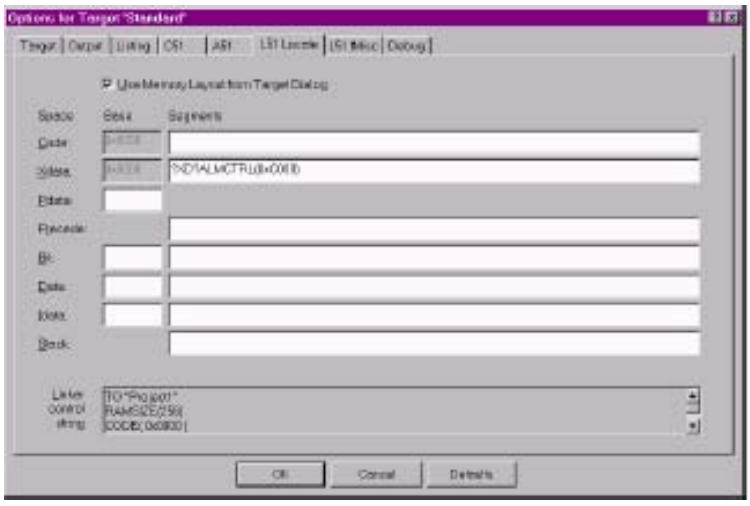
"

,

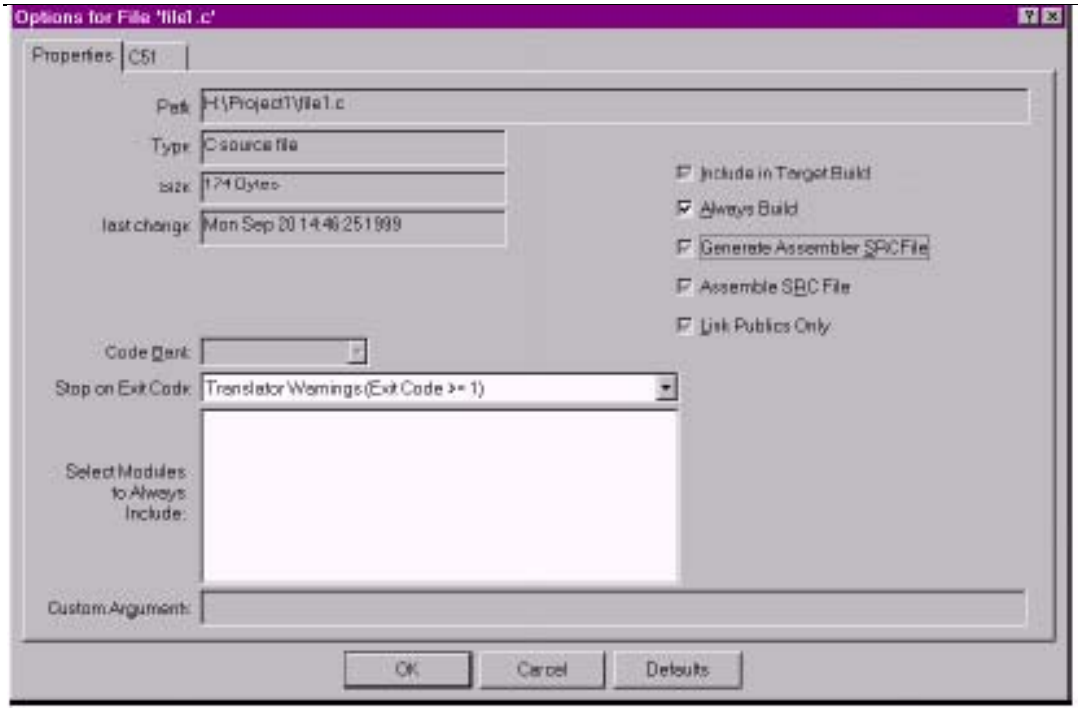
Type, Size

文件和文件组的特定选项 - 属性对话框 uVision2允许你通过**Project Window – Files**页的弹出式菜单设定以选项为灰色框

Last Change Include in 禁止此选项



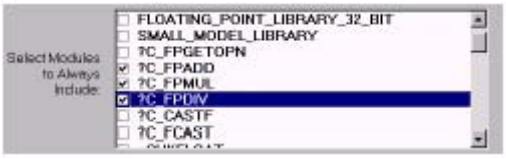
例中我们为文 FILE1.C定义的属性为编译时遇到告



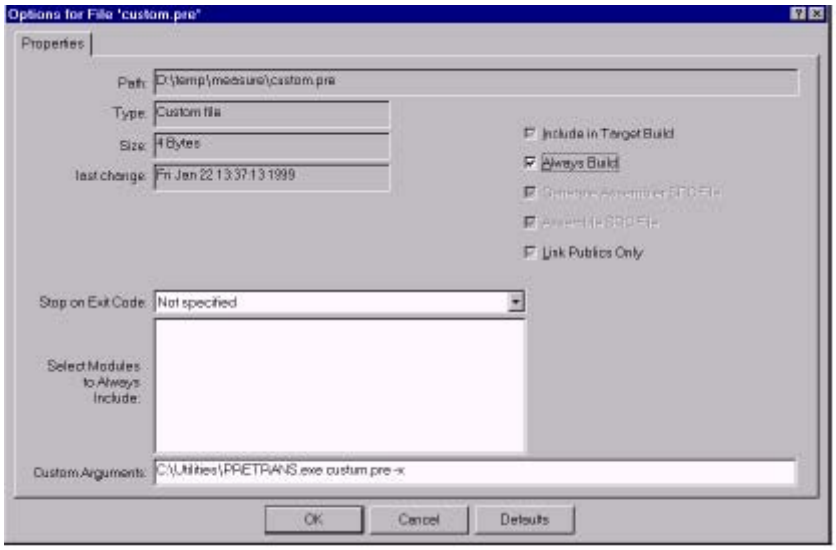
编

编

请核对你是否能　　你的C
提供了几个内联的函数不需要插入汇编指令　C

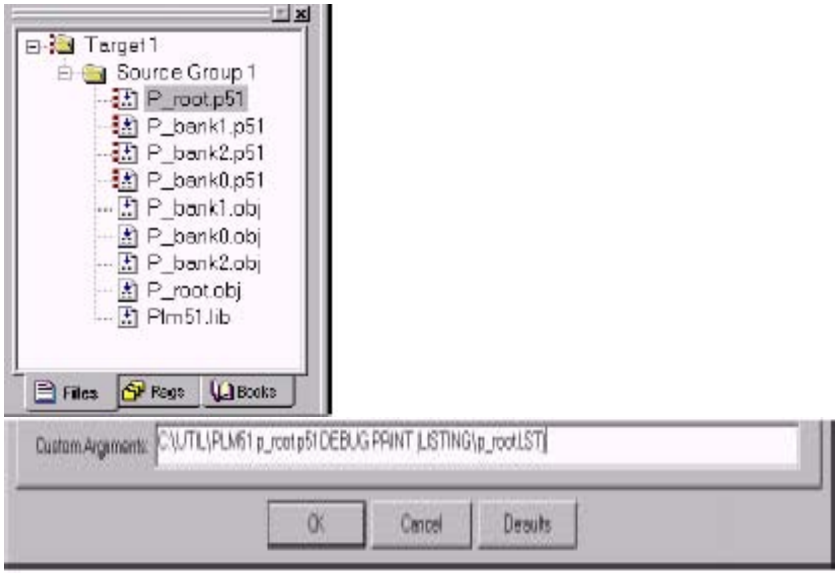


使用用户编译器　如果你增加一个带未知文件后缀
件
编译器将从用户文件产生一个源文件
将此源文件编译成能够连接到你的应用中的目标文件



使用Intel PL/M-5

如果你还有Intel PL/M-51源文件需要包含到你的uVision2项目中你可以以用户文件的形式增加这些源文件到uVision2项目树中另外你也必须插入由PL/M51生成的*.OBJ文件和Intel PLM51.LIB



Project – File Ex

不同的编译器
通过**Project Window – Files**页的弹出式菜单你可以为一不同选项在对话框页中有三态选择如果一个
你
该

打开菜单时将列出所有工具的详细的信息在你向我们报告