

单片机C语言 编程与实例

■ ■ ■ 赵亮 侯国锐 编著

图书在版编目(CIP)数据

单片机C语言编程与实例 / 赵亮, 侯国锐编著. — 北京: 人民邮电出版社, 2003.9

ISBN 7-112-11247-8

单片机C语言 编程与实例

■ ■ ■ 赵亮 侯国锐 编著

单片机C语言编程与实例

编 著 赵 亮 侯 国 锐

编 审 侯 国 锐

人民邮电出版社

ISBN 7-112-11247-8

定价: 32.00元

ISBN 7-112-11247-8

ISBN 7-112-11247-8

ISBN 7-112-11247-8

ISBN 7-112-11247-8

ISBN 7-112-11247-8

ISBN 7-112-11247-8

ISBN 7-112-11247-8

ISBN 7-112-11247-8

ISBN 7-112-11247-8

ISBN 7-112-11247-8

ISBN 7-112-11247-8

人民邮电出版社

图书在版编目 (CIP) 数据

单片机 C 语言编程与实例/赵亮, 侯国锐编著. —北京: 人民邮电出版社, 2003.9
ISBN 7-115-11547-8

I. 单... II. ①赵...②侯... III.①单片微型计算机—程序设计②C 语言—程序设计

IV.①TP368.1②TP312

中国版本图书馆 CIP 数据核字 (2003) 第 074954 号

内容提要

本书结合目前应用非常广泛的 C 语言以及 Keil C51 编译器, 详细介绍单片机 C 语言编程的方法。本书通过大量应用实例对单片机资源及其外围芯片进行详细介绍。

全书分为上、下两篇。上篇侧重于介绍基础知识, 主要有 C51 语言和 C 语言以及汇编语言的对比, 单片机内部资源、扩展资源及其编程等内容; 下篇侧重于应用实例, 通过实例, 读者既可以在工作中进行类比编程, 又可以开阔思路, 提高实际工作能力。

本书特点是实例新颖、内容齐全、实用性强, 可作为单片机爱好者以及单片机开发人员的实用参考书。

单片机 C 语言编程与实例

◆ 编 著 赵 亮 侯国锐

责任编辑 刘 浩

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号

邮编 100061 电子函件 315@ptpress.com.cn

网址 <http://www.ptpress.com.cn>

读者热线 010-67132692

北京汉魂图文设计有限公司制作

北京密云春雷印刷厂印刷

新华书店总店北京发行所经销

◆ 开本: 787×1092 1/16

印张: 21

字数: 512 千字

2003 年 9 月第 1 版

印数: 1-5 000 册

2003 年 9 月北京第 1 次印刷

ISBN7-115-11547-8/TP·3573

定价: 32.00 元

本书如有印装质量问题, 请与本社联系 电话: (010) 67129223

前 言

8051 是 Intel 公司开发的一款相当成功的单片机, 现在已普遍应用于工业生产中。目前有很多半导体芯片公司制造出了与 8051 兼容的单片机, 它们构成了通常所说的 51 系列单片机。

目前介绍 51 单片机应用的书籍很多, 但基本上都是基于汇编语言的。与汇编相比, C 语言在功能上、结构性、可读性、可维护性上有明显的优势, 易学易用, 因此出现了专门用于 51 系列单片机编程的 C 语言——C51。目前最先进、功能最强大的 C51 编译器是 Keil C51。

本书没有把太多的篇幅放在介绍 51 系列单片机的结构原理以及汇编指令上, 因为介绍这方面知识的书籍和资料很多, 而是把主要篇幅放在基于 C51 语言的单片机应用上。本书分上篇和下篇, 上篇侧重于基础知识, 下篇侧重于实际应用。

上篇包括第 1~4 章。

第 1 章介绍 C 语言基础知识以及 C51 语言对标准 C 语言的扩展, 并与汇编语言作了简单对照。

第 2 章以实例的形式介绍 uVision2 集成开发环境 (IDE)。

第 3 章介绍单片机内部资源及其 C 语言编程。

第 4 章介绍单片机资源扩展及其 C 语言编程, 介绍的扩展芯片是目前应用比较广泛的。

下篇包括第 5~14 章。

第 5 章介绍液晶显示 LCD 应用, 分别介绍了两种液晶显示控制芯片。

第 6 章介绍虚拟 I²C 接口技术。

第 7 章介绍红外通信接口应用。

第 8 章介绍语音芯片 ISD4004 及其应用。

第 9 章介绍时钟芯片应用, 也分别介绍了两种时钟控制芯片。

第 10 章介绍数据采集, 详细介绍数据采集的关键芯片 AD 转换器以及 DA 转换器的应用。

第 11~13 章分别介绍单片机之间的通信、单片机与 PC 通信以及单片机在网络通信中的简单应用。

第 14 章详尽地介绍了单片机在模糊控制、医疗以及语音数据采集方面的综合应用。

需要指出的是, 本书特别适合具有一定单片机基础知识的读者阅读。但对初学者, 只要掌握一些单片机硬件方面的基础知识, 也可以读懂本书。

各章的程序代码可以到 <http://ucbook.com.cn> 处下载。

本书内容丰富、新颖、通俗、实用, 所举的实例实用性都很强, 稍作改动即可应用到实际当中。由于作者水平有限, 书中难免有错误和不妥之处, 恳请读者批评指正 (可发信至 ridge1@sina.com)。

编者
2003 年 7 月

目 录

上 篇

第 1 章 C51 程序设计基础	1
1.1 单片机 C 语言与汇编语言	1
1.1.1 单片机汇编语言与 C 语言程序设计对照	2
1.1.2 汇编语言和 C 语言混合编程	3
1.2 C51 对标准 C 语言的扩展	8
1.2.1 数据类型	8
1.2.2 存储类型及存储区	9
1.2.3 存储器模式	12
1.2.4 特殊功能寄存器(SFR)	13
1.2.5 C51 指针	14
1.2.6 函数	17
1.2.7 重入函数	20
1.2.8 中断函数	21
1.2.9 绝对地址访问	22
1.2.10 动态存储分配	23
1.2.11 使用 C51 的技巧	25
1.3 C51 使用规范	27
第 2 章 μ Vision2 集成开发环境	30
2.1 关于开发环境	30
2.2 菜单条、工具栏和快捷键	30
2.3 创建项目实例	34
2.4 μ Vision2 功能	44
2.5 编写优化代码	50
2.6 技巧和窍门	52
第 3 章 单片机内部资源及其 C 语言编程	56
3.1 中断系统	56
3.1.1 中断系统介绍	56
3.1.2 C51 编写中断服务程序	60
3.1.3 共用中断	62
3.1.4 外部中断的扩充	63
3.2 定时器/计数器	65
3.2.1 定时器/计数器结构	65
3.2.2 定时器/计数器控制寄存器	66

3.2.3	定时器/计数器工作模式	67
3.2.4	定时器/计数器的初始化	68
3.2.5	定时器/计数器综合应用	68
3.3	并行 I/O 口	70
3.3.1	并行 I/O 口简析	70
3.3.2	编程实例	74
3.3.3	LED 显示电路	79
3.3.4	键盘控制电路	89
3.4	串行口及其通信	95
3.4.1	8051 单片机的串行口结构	95
3.4.2	串行口应用	98
第 4 章	单片机资源扩展及其 C 语言编程	102
4.1	可编程外围并行接口 8255A	102
4.1.1	8255 简介	102
4.1.2	程序设计实例	108
4.2	三线制 Microware 串行总线 E ² PROM 的应用	113
4.2.1	三线制 Microware 串行总线简介	113
4.2.2	三线制 Microware 总线的 E ² PROM	113
4.2.3	在 51 单片机上的应用	114
4.2.4	程序设计	115
4.3	键盘与 LED 控制芯片 HD7279A	118
4.3.1	简介	118
4.3.2	控制指令	120
4.3.3	时序	123
4.3.4	HD7279A 与 AT89C51 的接口以及程序设计	124

下 篇

第 5 章	液晶显示 LCD	133
5.1	液晶显示简介	133
5.2	内置 HD61202 控制驱动器图形液晶显示模块	134
5.2.1	液晶显示模块的电路特性	134
5.2.2	液晶显示模块的软件特性	136
5.2.3	液晶显示模块 12864 和 19264 的应用	138
第 6 章	虚拟 I ² C 接口技术	163
6.1	I ² C 总线简介	163
6.1.1	I ² C 总线的基本结构	163
6.1.2	双向传输的接口特性	163
6.1.3	I ² C 总线上的时钟信号	164
6.1.4	数据的传送	164
6.1.5	总线竞争的仲裁	165

6.1.6	I ² C 总线接口器件	165
6.2	模拟 I ² C 总线的 C51 程序	167
6.3	I ² C 总线在 IC 卡设计中的应用	173
6.3.1	简介	173
6.3.2	硬件特性	173
6.3.3	AT24C01 与单片机接口	174
6.3.4	程序设计	174
第 7 章	红外通信接口	180
7.1	P87LPC762 单片机简介	180
7.2	NB9148 简介	180
7.3	接收处理电路	184
7.4	程序设计	186
第 8 章	语音芯片 ISD4004 及其应用	193
8.1	ISD4004 简介	193
8.2	引脚功能描述	193
8.3	工作原理与功能特性	195
8.4	典型应用	197
第 9 章	时钟芯片	203
9.1	时钟芯片 DS1302	203
9.1.1	DS1302 简介	203
9.1.2	结构与工作原理	204
9.1.3	DS1302 与 89C51 的连接电路	207
9.1.4	程序设计	207
9.2	时钟/日历芯片 PCF8563	216
9.2.1	PCF8563 简介	216
9.2.2	PCF8563 与 I ² C 总线	218
9.2.3	应用概述	218
9.2.4	程序设计	218
第 10 章	数据采集	224
10.1	A/D 转换器 ADS7804	224
10.1.1	ADS7804 简介	224
10.1.2	ADS7804 与 51 单片机的接口	226
10.1.3	C51 语言程序设计	227
10.2	MAX1247、MAX525 与单片机接口	228
10.2.1	MAX1247 和 MAX525 简介	228
10.2.2	工作原理	228
10.2.3	硬件接口及软件编程实例	232
10.2.4	其他同类产品的应用	237
第 11 章	单片机通信	242
11.1	单片机双机通信	242

11.1.1	双机通信原理	242
11.1.2	双机通信协议	243
11.1.3	双机通信程序设计	243
11.2	单片机多机通信	245
11.2.1	多机通信原理	245
11.2.2	程序设计	246
第 12 章	单片机与 PC 通信	254
12.1	RS-232C 介绍与 PC 硬件	254
12.2	通信程序设计	259
第 13 章	单片机与网络	271
13.1	51 单片机内置定时器作 TDMA 控制	271
13.1.1	TDMA (时分多址) 简介	271
13.1.2	TDMA 的单片机实现	272
13.1.3	保持节点器件同步	279
13.2	单片机实现载波监听多址接入 (CSMA)	279
13.2.1	载波监听多址访问 (CSMA) 简介	280
13.2.2	单片机实现载波监听多址接入 CSMA	281
第 14 章	51 单片机系统应用实例	285
14.1	语音数据采集、回放和串行数据传输系统	285
14.1.1	系统功能简介	285
14.1.2	DS1270 接口及 51 扩展方案	286
14.1.3	LCM1602 总线方式驱动接口	287
14.1.4	外围器件	289
14.1.5	语音处理模拟部分设计	291
14.1.6	系统原理图	292
14.1.7	程序设计	295
14.2	医疗激光器功率控制	304
14.2.1	系统功能简介	304
14.2.2	行列式扫描键盘及 C51 程序设计	305
14.2.3	数字电位器 DS1867 驱动	305
14.2.4	LCM1602 口线方式驱动接口	307
14.2.5	数字温度计 DS1820 及 1-wire 总线	307
14.2.6	系统原理图	311
14.2.7	程序设计	312

第 1 章 C51 程序设计基础

1.1 单片机 C 语言与汇编语言

在单片机的开发应用中，逐渐引入了高级语言，C语言就是其中的一种。对用惯了汇编语言的人来说，高级语言可控性不好，不如汇编语言那样能够随心所欲。但是使用汇编语言会遇到很多问题，首先它的可读性和可维护性不强，特别是当程序没有很好标注的时候，其次就是代码的可重用性也比较低。使用C语言就可以很好地解决这些问题。

C语言具有良好的模块化，容易阅读和维护等优点。由于模块化，用C语言编写的程序有很好的可移植性，功能化的代码能够很方便地从一个工程移植到另一个工程，从而减少了开发时间。

用C语言编写程序比用汇编语言更符合人们的思考习惯，开发者可以更专心地考虑算法而不是考虑一些细节问题，这样就减少了开发和调试的时间。使用像C这样的语言，程序员不必十分熟悉处理器的运算过程。很多处理器支持C编译器，这意味着对新的处理器也能很快上手，而不必知道处理器的具体内部结构，这使得用C语言编写的程序比汇编程序有更好的可移植性。

所有这些并不说明汇编语言就没有了立足之地，很多系统特别是实时时钟系统都是用C语言和汇编语言联合编写的。对时钟要求严格时，使用汇编语言是唯一的方法。除此之外，包括硬件接口的操作都应该用C语言来编写。C语言的特点就是可以使程序员尽量少地对硬件进行操作，它是一种功能性和结构性很强的语言。

对于大多数51系列单片机，使用C语言这样的高级语言与使用汇编语言相比具有如下优点：

- (1) 不需要了解处理器的指令集，也不必了解存储器结构。
- (2) 寄存器分配和寻址方式由编译器进行管理，编程时不需要考虑存储器的寻址和数据类型等细节。
- (3) 指定操作的变量选择组合提高了程序的可读性。
- (4) 可使用与人的思维更相近的关键字和操作函数。
- (5) 与使用汇编语言编程相比，程序的开发和调试时间大大缩短。
- (6) C语言中的库文件提供许多标准的例程，例如格式化输出、数据转换和浮点运算等。
- (7) 通过C语言可实现模块化编程技术，从而可将已编制好的程序加入到新程序中。
- (8) C语言可移植性好且非常普及，C语言编译器几乎适用于所有的目标系统，已完成的软件项目可以很容易地转换到其他的处理器或环境中。

1.1.1 单片机汇编语言与 C 语言程序设计对照

本书中的程序都是用Keil C51语言编写的，但为了让读者对C51语言与汇编语言之间的区别与联系有个了解，下面给出几个分别用汇编语言和C51语言编写的例子，以便于对照。

例如将外部数据存储器的000BH和000CH单元的内容相互交换。

首先用汇编语言编程，汇编程序如下：

```

ORG 0000H
MOV P2,#0H          ;送高八位地址至 P2 口
MOV R0,#0BH          ;R0=0BH
MOV R1,#0CH          ;R1=0CH
MOVX A,@R0           ;A=(000BH)
MOV 20H,A            ;(20H)=000BH
MOVX A,@R1           ;A=(000CH)
XCH A,20H            ;(20H)←→A
MOVX @R1,A           ;交换后的数据送各单元
MOV A,20H
MOVX @R0,A
SJMP $
END

```

C语言对地址的指示方法可以采用指针变量，也可以引用头文件absacc.h作为绝对地址访问。下面的程序采用绝对地址访问方法。

```
#include<absacc.h>
```

```
void main(void)
```

```
char c;
```

```
do
```

```
{
c=XBYTE[11];
```

```
XBYTE[11]=XBYTE[12];
```

```
XBYTE[12]=c;
```

```
}while(1);
```

```
}
```

程序中为了方便反复观察，使用了死循环语句while(1)，只要使用“Ctrl+C”即可退出死循环。上面程序通过编译，生成的反汇编程序如下：

```
0000 LJMP 0014H
```

```
0003 MOV DPTR,#000BH
```

```
0006 MOVX A,@DPTR
```

```
0007 MOV A,R7
```

```
0008 INC DPTR
```

```
0009 MOVX A,@DPTR
```

```
000A MOV DPTR, #000BH
000D MOVX @DPTR, A
000E INC DPTR
000F MOV A, R7
0010 MOVX @DPTR, A
0011 SJMP 0003H
0013 RET
0014 MOV R0, 7FH
0016 CLR A
0017 MOV @R0, A
0018 DJNZ R0, 0017H
001A MOV SP, #07H
001DLJMP 0003H
```

对照C语言编写的程序与反汇编程序，可以看出：

- (1) 进入C程序后，首先将RAM地址为00~7FH的128个单元清零，然后置SP为07H（SP根据变量多少而不同）。因此，如果要对内部RAM置初值，一定是在执行了一条C语句之后。
- (2) 对于C程序设定的变量，C51编译器自行安排寄存器或存储器作参数传递区，通常在R0~R7（一组或两组，根据参数多少而定）。因此，如果对具体地址置数，应该避开R0~R7这些地址。
- (3) 如果不特别制定变量的存储类型，变量通常被安排在内部RAM区。

1.1.2 汇编语言和 C 语言混合编程

由于单片机硬件的限制，在有些场合无法用C语言编写，而只能用汇编语言来编写程序。大多数情况下汇编程序能和用C语言编写的程序很好地结合在一起。本章将详细介绍如何进行汇编语言和C语言的混合编程，以及如何修改由C程序编译后产生的汇编代码，从而得到精确的控制时间。

1. 增加段和局部变量

在把汇编程序加入到C程序中之前，必须使汇编程序和C程序一样具有明确的边界、参数、返回值和局部变量。

一般用汇编语言编写的程序中，变量的传递参数所使用的寄存器是无规律的，这将导致汇编语言编写的函数之间参数传递混乱，难以维护。如果在编写汇编功能函数时仿照C函数，并按照C51的参数传递标准，则程序就会有很好的可读性，并有利于维护，而且这样编写出来的函数很容易和C语言编写的函数进行连接。

汇编程序中的每一个功能函数都有自己的程序存储区，如果有局部变量，就会有相应的存储空间DATA、XDATA等。当程序中需要快速寻址的变量时，就可以把它声明在DATA段中，如果需要查寻表格，可声明在CODE段中。需要特别注意的是，局部变量只对当前使用它们的程序段有效。

2. 函数声明

为了使汇编程序段和C程序能够兼容，必须为汇编语言编写的程序段指定段名并进行定义。如果要在它们之间传递函数，则必须保证汇编程序用来传递函数的存储区和C函数使用的存储区是一样的。被调用的汇编函数不仅要在汇编程序中使用伪指令以使CODE选项有效，并声明为可再定位的段类型，而且还要在调用它的C语言主程序中进行声明。函数名的转换规律如表1-1所示。

表 1-1 函数名的转换规律

主函数中的声明	汇编符号名	说明
void func(void)	FUNC	无参数传递或不含寄存器的函数名不作改变转入目标文件中，名字只是简单地转为大写形式
void func(char)	_FUNC	带寄存器参数的函数名，前面加“_”前缀，它表明这类函数包含寄存器内的参数传递
void func(void) reentrant	__FUNC	对于重入函数，前面加“_”前缀，它表明该函数包含栈内的参数传递

以下是一个典型的可被C程序调用的汇编函数，该函数不传递参数。

```
?PR?CLRMEM SEGMENT CODE      ;程序存储区声明
PUBLIC CLRMEM                  ;输出函数名
RSEG ?PR? CLRMEM               ;该函数可被连接器放在任何地方

/*****

函数：CLRMEM
功能描述：清除内部 RAM 区
参数：无
返回值：无

*****/

CLRMEM:
MOV R0,#7FH
CLR A
IDATALOOP:
MOV @R0,A
DJNZ R0,IDATALOOP
RET
END
```

由此可以看出汇编文件的格式化是很简单的，只需给存放功能函数的段指定一个段名即可。因为是在代码区内，所以段名的开头为PR，这两个字符是为了和C51的内部命名转换兼容的，如表1-2所示。

表 1-2 命名转换规律

存储区	命名转换
CODE	?PR?CO
XDATA	?XD
DATA	?DT
BIT	?BI
PDATA	?PD

RSEG为段名的属性，这意味着连接器可把该段放置在代码区的任意位置。当段名确定后，文件必须声明公共符号，如上例中的PUBLIC CLRMEM语句，然后编写代码。对于有传递参数的函数必须符合参数的传递规则，Keil C51在内部RAM中传递参数时一般都用当前的寄存器组。当函数接收3个以上参数时，存储区中的一个默认段将用来传递剩余的参数。用作接收参数的寄存器如表1-3所示。

表 1-3 接收参数寄存器

参数序号	char	int	long,float	通用指针
1	R7	R6 & R7	R4~R7	R1~R3
2	R5	R4 & R5	-	-
3	R3	R2 & R3	-	-

下面是几个参数传递的例子。

func1(int a): “a”是第一个参数，在R6、R7中传递。

func2(int a,int b,int *c): “a”在R6、R7中传递，“b”在R4、R5中传递，“c”在R1、R2、R3中传递。

func3(long a,long b): “a”在R4~R7中传递，“b”不能在寄存器中传递，而只能在参数传递段中传递。

3. Keil C51 与汇编的接口

(1) 模块内接口

有时候，需要使用汇编语言来编写程序，比如对硬件进行操作或在一些对时钟要求很严格的场合，但又不希望用汇编语言来编写全部程序或调用汇编语言编写的函数，那么可以通过预编译指令“asm”在C代码中插入汇编代码。

方法是用#pragma语句，具体结构是：

```
#pragma asm
```

```
汇编行
```

```
#pragma endasm
```

这种方法是通过asm与endasm告诉C51编译器，中间行不用编译为汇编行，例如：

```
#include <reg51.h>
```

```
extern unsigned char code newval[256];
```

```
void func1(unsigned char param)
```

```
{
```

```
unsigned char temp;
```

```
temp=newval[param];
```

```
temp*=2;
```

```
temp/=3;
```

```
#pragma asm
```

```
MOV P1, R7
```

```
NOP
```

```
NOP
```

```
;预编译指令“asm”
```

```
;输出 temp 中的数
```

```

NOP
MOV P1, #0
#pragma endasm

```

(2) 模块间接口

C 模块与汇编模块的接口较简单, 分别用 C51 与 A51 对源文件进行编译, 然后用 L51 连接 obj 文件即可。模块接口间的关键问题在于 C 函数与汇编函数之间的参数传递。C51 中有两种参数传递方法。

① 通过寄存器传递函数参数

汇编函数要得到参数值时就访问这些寄存器, 如果这些值正被使用并保存在其他地方或已经不再需要了, 那么这些寄存器可被用作其他用途。下面是一个 C 程序与汇编程序的接口例子, 应该注意到通过内部 RAM 传递参数的函数将使用规定的寄存器, 汇编函数将使用这些寄存器接收参数。对于要传递多于 3 个参数的函数, 剩余的参数将在默认的存储器段中传递。

//C 程序中汇编函数的声明

```

bit devwait(unsigned char ticks, unsigned char xdata *buf);
if (devwait(5, &outbuf))
{bytes_out++;}

```

//汇编代码

```

?PR?_DEWAIT SEGMENT CODE;           //在程序存储区中定义段
PUBLIC _DEWAIT;                       //输出函数名
RSEG ?PR?_DEWAIT;                     //该函数可被连接器放在任何地方

```

函数: _devwait

功能描述: 等待定时器 0 溢出, 向外部器件表明 P1 中的数据是有效的。如果定时器尚未溢出, 将被写入 XDATA 的指定地址中。

参数: R7——存放要等待的定时长度; R4R5——存放要写入的 XDATA 区地址。

返回值: 读数成功返回 1, 时间到返回 0。

_DEWAIT:

CLR TR0 ;设置定时器 0

CLR TF0

MOV TH0, #00

MOV TL0, #00

SETB TR0

JBC TF0, L1 ;检测定时标志位

JB T1, L2 ;检测数据是否准备就绪

L1:

DJNZ R7, _DEWAIT ;减 1

CLR C

CLR TR0 ;停止定时器 0

```

RET
L2:
MOV DPH, R4           ;取地址并放入 DPTR
MOV DPL, R5
PUSH ACC
MOV A, P1             ;得到输入数据
MOVX @DPTR, A
POP ACC
CLR TR0              ;停止定时器 0
SETB C                ;设置返回位
RET
END

```

在上面的代码中，并没有讨论返回值问题。在这里，函数返回一个位变量。如果时间到，将返回0；如果输入字节被写入指定的地址中，将返回1。当从函数中返回值时，C51通过转换使用内部存储区，编译器将使用当前寄存器组来传递返回参数。返回参数所使用的寄存器如表1-4所示。返回这些类型的函数可使用这些寄存器来存储局部变量，直到这些寄存器被用来返回参数。如果函数要返回一个长整型，就可以方便地使用R4~R7这4个寄存器，而不需要声明一个段来存放局部变量，存储区就更加优化了。返回值类型与寄存器对照如表1-4所示。需要注意的是，函数不应随意使用没有被用来传递参数的寄存器。

表 1-4

返回值类型与寄存器对照

返回值类型	寄存器	说明
Bit	C (标志位)	由具体标志位返回
char/unsigned char 1_byte 指针	R7	单字节由 R7 返回
int/unsigned int 2_byte 指针	R6 & R7	双字节由 R6 和 R7 返回，高位在 R6 中，低位在 R7 中
long/unsigned long	R4~R7	高位在 R4 中，低位在 R7 中
float	R4~R7	32bit IEEE 格式，指数和符号位在 R7 中
通用指针	R1~R3	存储类型在 R3 中，高位在 R2 中，低位在 R1 中

② 通过固定存储区传递(Fixed Memory)

这种方法将 bit 型参数传到一个存储段中：

```
? function_name?BIT
```

将其他类型参数均传给下面的段：

```
? function_name?BYTE
```

且按照预选顺序存放。至于这个固定存储区本身在何处，则由存储模式默认指定。

(3) SRC 控制

该控制指令将 C 文件编译生成汇编文件 (.SRC)，该汇编文件在改名后，生成汇编.ASM 文件，再用 A51 进行编译。

1.2 C51 对标准 C 语言的扩展

C 语言是一门应用非常普遍的高级程序设计语言，因此本书并不准备花太多的时间来介绍 C 语言的基本用法，而把主要精力集中到分析 51 系列单片机 C 语言（以下简称为 C51）和标准 C 语言之间的区别上来，或者说 C51 对标准 C 语言的扩展上。如果读者对标准 C 语言不是很了解，可以参考任何一本专门介绍 C 语言的书籍。

C51 语言的特色主要体现在以下几方面：

(1) C51 虽然继承了标准 C 语言的绝大部分的特性，而且基本语法相同，但其本身又在特定的硬件结构上有所扩展，如关键字 `sbit`、`data`、`idata`、`pdata`、`xdata`、`code` 等。

(2) 应用 C51 更要注重对系统资源的理解，因为单片机的系统资源相对 PC 机来说很贫乏，对于 RAM、ROM 中的每一字节都要充分利用。可以通过多看编译生成的 .m51 文件来了解自己程序中资源的利用情况。

(3) 程序上应用的各种算法要精简，不要对系统构成过重的负担。尽量少用浮点运算，可以用 `unsigned` 无符号型数据的就不要用有符号型数据，尽量避免多字节的乘除运算，多使用移位运算等。

C51 相对于标准 C 语言的扩展直接针对 51 系列 CPU 硬件，大致有以下几个方面。

1.2.1 数据类型

C51 具有标准 C 语言的所有标准数据类型，除此之外，为了更加有效地利用 8051 的结构，还加入了以下特殊的数据类型。

- `bit` 位变量值为 0 或 1。
- `sbit` 从字节中声明的位变量 0 或 1。
- `sfr` 特殊功能寄存器，`sfr` 字节地址为 0~255。
- `sfr16` 同上，只是 `sfr` 字地址为 0~65 535。

其余数据类型如 `char`、`enum`、`short`、`int`、`long`、`float` 等与标准语言 C 相同，完整的数据类型表如表 1-5 所示。

表 1-5 数据类型

数据类型	位数	字节数	数值范围
<code>bit</code>	1		0~1
<code>char</code>	8	1	-128~+127
<code>unsigned char</code>	8	1	0~255
<code>enum</code>	16	2	-32 768~+32 767
<code>short</code>	16	2	-32 768~+32 767
<code>unsigned short</code>	16	2	0~65 535
<code>int</code>	16	2	-32 768~+32 767
<code>unsigned int</code>	16	2	0~65 535
<code>long</code>	32	4	-2 147 483 648~+2 147 483 647
<code>unsigned long</code>	32	4	0~4 294 967 295
<code>float</code>	32	4	+1.175 494E-38~+3.402 823E+38
<code>sbit</code>	1		0~1
<code>sfr</code>	8	1	0~255
<code>Sfr16</code>	16	2	0~65 535

bit、sbit、sfr和sfr16数据类型专门用于8051硬件和C51编译器，并不是标准C语言的一部分，不能通过指针进行访问。bit、sbit、sfr和sfr16数据类型用于访问8051的特殊功能寄存器，例如sfr P0 = 0x80，表示声明变量P0，并为其分配特殊功能寄存器地址0x80。

当结果为不同的数据类型时，C51编译器自动转换数据类型，例如位变量在整数分配中就被转换成一个整数。除了数据类型的转换之外，带符号变量的符号扩展也是自动完成的。

1.2.2 存储类型及存储区

C51编译器支持8051及其扩展系列，并提供对8051所有存储区的访问。存储区可分为内部数据存储区、外部数据存储区以及程序存储区。8051CPU内部的数据存储区是可读写的，8051派生系列最多可有256字节的内部数据存储区，其中低128字节可直接寻址，高128字节（从0x80到0xFF）只能间接寻址，从20H开始的16字节可位寻址。内部数据区又可以分成3个不同的存储类型：data、idata和bdata。外部数据区也是可读写的，访问外部数据区比访问内部数据区慢，因为外部数据区是通过数据指针加载地址来间接访问的。C51编译器提供两种不同的存储类型xdata和pdata访问外部数据。程序CODE存储区是只读的，不能写。程序存储区可能在8051CPU内或者在外部或者内外都有，这由8051派生的硬件决定。

每个变量可以明确地分配到指定的存储空间，对内部数据存储器的访问比对外部数据存储器的访问快许多，因此应当将频繁使用的变量放在内部数据存储器中，而把较少使用的变量放在外部数据存储器中。各存储区的简单描述如表1-6所示。

变量的声明中还包括了对存储器类型的指定，即指定变量存放的位置。

表 1-6 存储区描述

存储区	描述
DATA	RAM 的低 128 字节，可在一个周期内直接寻址
BDATA	DATA 区可字节、位混合寻址的 16 字节区
IDATA	RAM 区的高 128 字节，必须采用间接寻址
XDATA	外部存储区，使用 DPTR 间接寻址
PDATA	外部存储区的 256 字节，通过 P0 口的地址对其寻址。使用指令 MOVX @Rn，需要两个指令周期
CODE	程序存储区使用 DPTR 寻址

下面分别详细介绍各存储区并给出应用实例。

● DATA区 DATA区的寻址是最快的，所以应该把经常使用的变量放在DATA区；但是DATA区的空间是有限的，DATA区除了包含程序变量外，还包含了堆栈和寄存器组。DATA区声明中的存储类型标识符为data，通常指低128字节的内部数据区存储的变量，可直接寻址。声明举例如下：

```
unsigned char data system_status=0;
unsigned int data unit_id[2];
char data inp_string[16];
float data outp_value;
mytype data new_var
```

标准变量和用户自声明变量都可存储在DATA区中，只要不超过DATA区的范围即可，