单片机C语言编程与实例

赵亮 侯国锐 编著

人民邮电出版社
POSTS & TELECOM PRESS

单片机C语言编程与实例

赵亮 侯国锐 编著

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

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 I^2C 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

下 篇

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
        MOV   A,20H
        MOVX @R0,A                ;交换后的数据送各单元
        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
001D LJMP 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置在任何地方
/*************************************************************************************
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                    ;预编译指令“asm”
MOV P1, R7                     ;输出 temp 中的数
NOP
NOP
```

```
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?_DEVWAIT SEGMENT CODE;          //在程序存储区中定义段
PUBLIC _ DEVWAIT;                   //输出函数名
RSEG ?PR?_ DEVWAIT;                 //该函数可被连接器放置在任何地方
/*************************************************************************
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；R4lR5——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。
*************************************************************************/
_ DEVWAIT:
  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, _ DEVWAIT                ;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（以下简写为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
bit	1		0~1
char	8	1	-128~+127
unsigned char	8	1	0~255
enum	16	2	-32 768~+32 767
short	16	2	-32 768~+32 767
unsigned short	16	2	0~65 535
int	16	2	-32 768~+32 767
unsigned int	16	2	0~65 535
long	32	4	-2 147 483 648~+2 147 483 647
unsigned long	32	4	0~4 294 967 295
float	32	4	±1.175 494E-38~±3.402 823E+38
sbit	1		0~1
sfr	8	1	0~255
Sfr16	16	2	0~65 535

bit、sbit、sfrs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、sfrs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，