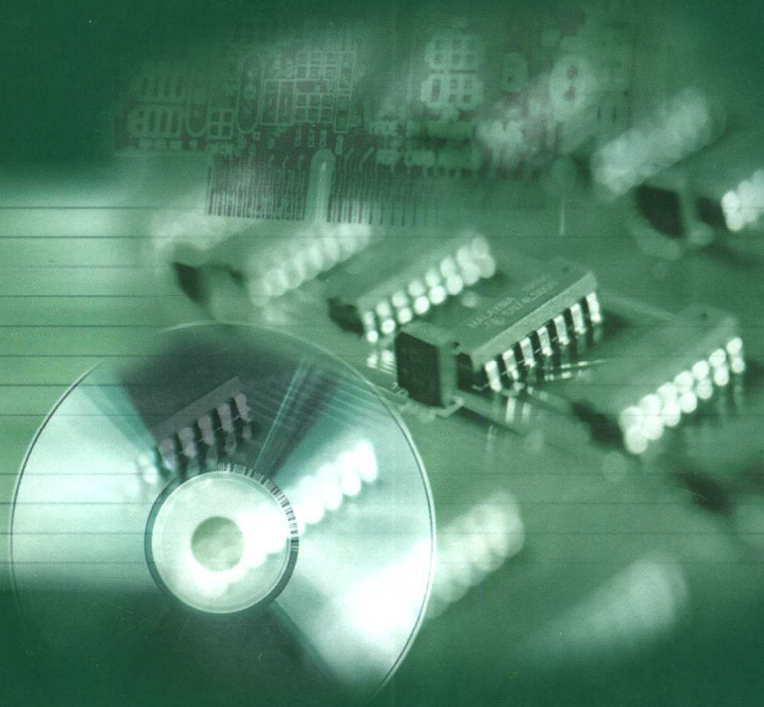
单片机语言C51应用实战集锦

范风强　兰婵丽　编著



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http:// www.phei.com.cn

单片机语言 C51 应用实战集锦

范风强　兰婵丽　编著

電子工業出版社
Publishing House of Electronics Industry
北京 • BEIJING

内 容 简 介

使用 C 语言开发速度快，代码可重复使用，程序结构清晰、易懂、易维护，易开发一些比较大型的项目。目前，许多编译器都已经支持了 C51，而且是 Windows 视窗界面。keilc51 是目前单片机开发最为流行的软件。

本书收集并整理了许多实用的采用 C51 单片机开发的程序，这些程序既可以给读者以开拓思路，参考的用途，又是实际的开发程序，可以直接作为程序应用在相同的开发系统上。通过本书的学习，读者可以进一步了解和掌握 C51 编程的思路和方法。

本书适用于从事单片机项目开发与应用的工程技术人员阅读。

图书在版编目（CIP）数据

单片机语言 C51 应用实战集锦/范风强等编著. —北京：电子工业出版社，2003.3
ISBN 7-5053-8590-9

Ⅰ. 单… Ⅱ. 范… Ⅲ. 单片微型计算机—程序设计 Ⅳ. TP368.1

中国版本图书馆 CIP 数据核字（2003）第 017889 号

责任编辑：竺南直
印　　刷：北京李史山胶印厂
出版发行：电子工业出版社　http://www.phei.com.cn
　　北京市海淀区万寿路 173 信箱　邮编 100036
经　　销：各地新华书店
开　　本：787×1092　1/16　印张：24.5　字数：548 千字
版　　次：2003 年 3 月第 1 版　2003 年 3 月第 1 次印刷
印　　数：6000 册　　定价：36.00 元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系。
联系电话：（010）68279077

前　言

采用单片机 C51 语言编程具有很多优越性。如果你不懂得单片机的指令集，也能够编写出完美的单片机程序；无须懂得单片机的具体硬件，也能够编出符合硬件实际的专业水平的程序；不同函数的数据实行覆盖，能有效利用片上有限的 RAM 空间。程序具有坚固性，数据被破坏是导致程序运行异常的重要因素。C 语言对数据进行了许多专业性的处理，避免了运行中间非异步的破坏；C 语言提供复杂的数据类型（数组、结构、联合、枚举、指针等），极大地增强了程序处理能力和灵活性；提供了 auto、static、const 等存储类型和专门针对 8051 单片机的 data、idata、pdata、xdata、code 等存储类型，自动为变量合理地分配地址；提供 small、compact、large 等编译模式，以适应片上存储器的大小。中断服务程序的现场保护和恢复，中断向量表的填写，是直接与单片机相关的，都由 C 编译器代办。它还提供常用的标准函数库，以供用户直接使用；在头文件中定义宏、说明复杂数据类型和函数原型，这有利于程序的移植和支持单片机的系列化产品的开发；有严格的句法检查，错误很少，可容易地在高级语言的水平上迅速地被排掉。可方便地接受多种实用程序的服务：如片上资源的初始化有专门的实用程序自动生成；有实时多任务操作系统可调度多道任务，简化用户编程，可提高运行的安全性等。

正是因为它有如此之多的优越性，使之成为了 51 系列单片机流行的开发语言。本书收集并整理了 40 个实用开发子程序，这些程序对学习开发 51 系列单片机的初学者有相当重要的入门指导作用，对于有经验的开发人员来说，也极具参考价值。

在单片机开发的过程中，最常见的是键盘、显示等接口部分用于人机交互，A/D、D/A 以及 I^2C 总线用于数据采集和传输，抗干扰、通信等用于把现场实时信号完整地发送到控制中心等。对于这些项目的开发，不同的开发者有不同的思路以及不同的技巧。这些都反映在开发的程序上的不同。对于我们提供的这些程序，不是说它们写得很好，而是它们经过了许多项目的检验,稳定而且实用。利用它们可以减少开发时间，同时也能给从事开发的技术人员启发思路，这也是本书的目的所在。

在本书的创作过程中，聂家财、钟开智、兰德昌、刘文涛等给予了许多帮助，在此表示感谢。

由于作者水平有限，书中难免有不足和缺陷之处，希望广大读者给予指正，并提出宝贵意见。作者的电子信箱是 fangfq2000@etang.com。

编著者

2002 年秋

目　录

程序一　实时时钟芯片 DS1302 的 C51 程序例子

```
/*********************************************************************/
/* 实时时钟模块 时钟芯片型号：DS1302 */
/*/
/*********************************************************************/
sbit T_CLK = P2^7; /*实时时钟时钟线引脚 */
sbit T_IO = P1^4; /*实时时钟数据线引脚 */
sbit T_RST = P1^5; /*实时时钟复位线引脚 */
/*********************************************************************
*
* 名称: v_RTInputByte
* 说明:
* 功能: 往 DS1302 写入 1Byte 数据
* 调用:
* 输入: ucDa 写入的数据
* 返回值: 无
*********************************************************************/
void v_RTInputByte(uchar ucDa)
{
uchar i;
ACC = ucDa;
for(i=8; i>0; i--)
{
T_IO = ACC0; /*相当于汇编中的 RRC */
T_CLK = 1;
T_CLK = 0;
ACC = ACC >> 1;
}
}
/*********************************************************************
*
* 名称: uchar uc_RTOutputByte
* 说明:
* 功能: 从 DS1302 读取 1Byte 数据
* 调用:
* 输入:
* 返回值: ACC
*********************************************************************/
uchar uc_RTOutputByte(void)
{
```

```
uchar i;
for(i=8; i>0; i--)
{
ACC = ACC >>1; /*相当于汇编中的 RRC */
ACC7 = T_IO;
T_CLK = 1;
T_CLK = 0;
}
return(ACC);
}
/*************************************************************************
*
* 名称: v_W1302
* 说明: 先写地址，后写命令/数据
* 功能: 往 DS1302 写入数据
* 调用: v_RTInputByte()
* 输入: ucAddr: DS1302 地址, ucDa: 要写的数据
* 返回值: 无
*************************************************************************/
void v_W1302(uchar ucAddr, uchar ucDa)
{
T_RST = 0;
T_CLK = 0;
T_RST = 1;
v_RTInputByte(ucAddr); /* 地址，命令 */
v_RTInputByte(ucDa); /* 写 1Byte 数据*/
T_CLK = 1;
T_RST =0;
}
/*************************************************************************
*
* 名称: uc_R1302
* 说明: 先写地址，后读命令/数据
* 功能: 读取 DS1302 某地址的数据
* 调用: v_RTInputByte() , uc_RTOutputByte()
* 输入: ucAddr: DS1302 地址
* 返回值: ucDa :读取的数据
*************************************************************************/
uchar uc_R1302(uchar ucAddr)
{
uchar ucDa;
T_RST = 0;
T_CLK = 0;
T_RST = 1;
v_RTInputByte(ucAddr); /* 地址，命令 */
```

```
ucDa = uc_RTOutputByte(); /* 读 1Byte 数据 */
T_CLK = 1;
T_RST =0;
return(ucDa);
}
/*********************************************************************
*
* 名称: v_BurstW1302T
* 说明: 先写地址，后写数据(时钟多字节方式)
* 功能: 往 DS1302 写入时钟数据(多字节方式)
* 调用: v_RTInputByte()
* 输入: pSecDa: 时钟数据地址 格式为: 秒 分 时 日 月 星期 年 控制
* 8Byte (BCD 码) 1B 1B 1B 1B 1B 1B 1B 1B
* 返回值: 无
*********************************************************************/
void v_BurstW1302T(uchar *pSecDa)
{
uchar i;
v_W1302(0x8e,0x00); /* 控制命令,WP=0,写操作?*/
T_RST = 0;
T_CLK = 0;
T_RST = 1;
v_RTInputByte(0xbe); /* 0xbe:时钟多字节写命令 */
for (i=8;i>0;i--) /*8Byte = 7Byte 时钟数据 + 1Byte 控制*/
{
v_RTInputByte(*pSecDa);/* 写 1Byte 数据*/
pSecDa++;
}
T_CLK = 1;
T_RST =0;
}
/*********************************************************************
*
* 名称: v_BurstR1302T
* 说明: 先写地址，后读命令/数据(时钟多字节方式)
* 功能: 读取 DS1302 时钟数据
* 调用: v_RTInputByte() , uc_RTOutputByte()
* 输入: pSecDa: 时钟数据地址 格式为: 秒 分 时 日 月 星期 年
* 7Byte (BCD 码) 1B 1B 1B 1B 1B 1B 1B
* 返回值: ucDa :读取的数据
*********************************************************************/
void v_BurstR1302T(uchar *pSecDa)
{
uchar i;
T_RST = 0;
```

```
T_CLK = 0;
T_RST = 1;
v_RTInputByte(0xbf); /* 0xbf:时钟多字节读命令 */
for (i=8; i>0; i--)
{
*pSecDa = uc_RTOutputByte(); /* 读 1Byte 数据 */
pSecDa++;
}
T_CLK = 1;
T_RST =0;
}
/*************************************************************************
*
* 名称: v_BurstW1302R
* 说明: 先写地址，后写数据(寄存器多字节方式)
* 功能: 往 DS1302 寄存器数写入数据(多字节方式)
* 调用: v_RTInputByte()
* 输入: pReDa: 寄存器数据地址
* 返回值: 无
*************************************************************************/
void v_BurstW1302R(uchar *pReDa)
{
uchar i;
v_W1302(0x8e,0x00); /* 控制命令,WP=0,写操作?*/
T_RST = 0;
T_CLK = 0;
T_RST = 1;
v_RTInputByte(0xfe); /* 0xbe:时钟多字节写命令 */
for (i=31;i>0;i--) /*31Byte 寄存器数据 */
{
v_RTInputByte(*pReDa); /* 写 1Byte 数据*/
pReDa++;
}
T_CLK = 1;
T_RST =0;
}
/*************************************************************************
*
* 名称: uc_BurstR1302R
* 说明: 先写地址，后读命令/数据(寄存器多字节方式)
* 功能: 读取 DS1302 寄存器数据
* 调用: v_RTInputByte() , uc_RTOutputByte()
* 输入: pReDa: 寄存器数据地址
* 返回值: 无
*************************************************************************/
```

```
void v_BurstR1302R(uchar *pReDa)
{
uchar i;
T_RST = 0;
T_CLK = 0;
T_RST = 1;
v_RTInputByte(0xff); /* 0xbf:时钟多字节读命令 */
for (i=31; i>0; i--) /*31Byte 寄存器数据 */
{
*pReDa = uc_RTOutputByte(); /* 读 1Byte 数据 */
pReDa++;
}
T_CLK = 1;
T_RST =0;
}
/*********************************************************************
*
* 名称: v_Set1302
* 说明:
* 功能: 设置初始时间
* 调用: v_W1302()
* 输入: pSecDa: 初始时间地址。初始时间格式为: 秒 分 时 日 月 星期 年
* 7Byte (BCD 码) 1B 1B 1B 1B 1B 1B 1B
* 返回值: 无
*********************************************************************/
void v_Set1302(uchar *pSecDa)
{
uchar i;
uchar ucAddr = 0x80;
v_W1302(0x8e,0x00); /* 控制命令,WP=0,写操作?*/
for(i =7;i>0;i--)
{
v_W1302(ucAddr,*pSecDa); /* 秒 分 时 日 月 星期 年 */

pSecDa++;
ucAddr +=2;
}
v_W1302(0x8e,0x80); /* 控制命令,WP=1,写保护?*/
}
/*********************************************************************
*
* 名称: v_Get1302
* 说明:
* 功能: 读取 DS1302 当前时间
* 调用: uc_R1302()
```

```
* 输入: ucCurtime: 保存当前时间地址。当前时间格式为: 秒 分 时 日 月 星期 年
* 7Byte (BCD 码) 1B 1B 1B 1B 1B 1B 1B
* 返回值: 无
*************************************************************************/
void v_Get1302(uchar ucCurtime[])
{
uchar i;
uchar ucAddr = 0x81;
for (i=0;i<7;i++)
{
ucCurtime[i] = uc_R1302(ucAddr);/*格式为: 秒 分 时 日 月
星期 年 */
ucAddr += 2;
}
}
```

程序二　C430 与 C51 的一点区别

C430 与 C51 语法上基本一样，但是编程时要注意以下几点。

（1）如果要判断 P2.0 是否为 1，C51 可以写为：if(P2&BIT0==BIT0)；但是在 C430 这样写会得不到结果，而要写为：if((P2&BIT0) == BIT0) 才对。

（2）在 C51 中如果要让程序等待可以直接用 while(1)，但是写 C430 程序时我曾经遇到 while(1)无效，后来发现是我没设置 WDT，加入 WDTCTL = WDTPW+WDTHOLD，一切正常。

（3）C51 有 bit flag 等指令来定义位，而 CP430 没有相关指令，但是可以这样实现：先定义一个变量 uchar flag，这样就有 8 个位变量可以使用。假设 C51 有这样的程序：

```
bit flag;
flag = 0;
while(flag==0); //等待
```

在 C430 里可以写成：

```
uchar flag;
flag &= ~BIT1;
while( (flag&BIT1) != BIT1 );
```

效果一样

程序三　一个菜单的例子

```
/*  Module  :menu.c

#include
#include
#define  SIZE_OF_KEYBD_MENU  20 //菜单长度

uchar  KeyFuncIndex=0;
//uchar  KeyFuncIndexNew=0;
void  (*KeyFuncPtr)();  //按键功能指针
typedef struct
{
  uchar  KeyStateIndex;  //当前状态索引号
  uchar  KeyDnState;  //按下"向下"键时转向的状态索引号
  uchar  KeyUpState;  //按下"向上"键时转向的状态索引号
  uchar  KeyCrState;  //按下"回车"键时转向的状态索引号
  void  (*CurrentOperate)();  //当前状态应该执行的功能操作
}  KbdTabStruct;

KbdTabStruct  code  KeyTab[SIZE_OF_KEYBD_MENU]=
{
  { 0, 0, 0, 1,(*DummyJob)},//顶层

  { 1, 2, 0, 3,(*DspUserInfo)},//第二层
  { 2, 1, 1, 9,(*DspServiceInfo)}, //第二层

  { 3, 0, 0, 1,(*DspVoltInfo)},//第三层>>DspUserInfo 的展开
  { 4, 0, 0, 1,(*DspCurrInfo)},//第三层>>DspUserInfo 的展开
  { 5, 0, 0, 1,(*DspFreqInfo)},//第三层>>DspUserInfo 的展开
  { 6, 0, 0, 1,(*DspCableInfo)},//第三层>>DspUserInfo 的展开
.....
{ 9, 0, 0, 1,(*DspSetVoltLevel)}//第三层>>DspServiceInfo 的展开
.....
};

void  GetKeyInput(void)
{
  uchar  KeyValue;
  KeyValue=P1&0x07; //去掉高 5bit
  delay(50000);
```

```
  switch(KeyValue)
  {
    case   1:  //回车键,找出新的菜单状态编号
    {
      KeyFuncIndex=KeyTab[KeyFuncIndex].KeyCrState;
      break;
    }
    case   2:  //向上键,找出新的菜单状态编号
    {
      KeyFuncIndex=KeyTab[KeyFuncIndex].KeyUpState;
      break;
    }
    case   4:  //向下键,找出新的菜单状态编号
    {
      KeyFuncIndex=KeyTab[KeyFuncIndex].KeyDnState;
      break;
    }
    default:  //按键错误的处理
.....
    break;
  }
  KeyFuncPtr=KeyTab[KeyFuncIndex].CurrentOperate;
  (*KeyFuncPtr)();//执行当前按键的操作
}
//其中 KeyTab 的设计值得读者仔细研究
```

程序四　DS1820 单芯片温度测量

```
//这里以 11.0592MHz 晶体为例
//sbit DQ =P2^1;//根据实际情况定义端口，DQ 代表端口

typedef unsigned char byte;
typedef unsigned int  word;

//延时
void delay(word useconds)
{
 for(;useconds>0;useconds--);
}

//复位
byte ow_reset(void)
```

```
{
 byte presence;
 DQ = 0; //设 DQ 低电平
 delay(29); //延迟 480μs
 DQ = 1; //高电平
 delay(3); //等待
 presence = DQ; //取允许信号
 delay(25); //延时
 return(presence); //返回允许信号
}    // 0=presence, 1 = no part

//从 1-wire 总线上读取一个字节
byte read_byte(void)
{
 byte i;
 byte value = 0;
 for (i=8;i>0;i--)
 {
  value>>=1;
  DQ = 0; //DQ 低
  DQ = 1; //返回高电平
  delay(1);  //for (i=0; i<3; i++);
  if(DQ)value|=0x80;
  delay(6); //延时
 }
 return(value);
}

//向 1-WIRE 总线上写一个字节
void write_byte(char val)
{
 byte i;
 for (i=8; i>0; i--) //一次写一字节
 {
  DQ = 0; //DQ 低
  DQ = val&0x01;
  delay(5); //端口悬挂
  DQ = 1;
  val=val/2;
 }
 delay(5);
}

//读取温度
char Read_Temperature(void)
```

```
{
 union{
  byte c[2];
  int x;
 }temp;

 ow_reset();
 write_byte(0xCC); //跳过 ROM
 write_byte(0xBE); //读可擦写芯片
 temp.c[1]=read_byte();
 temp.c[0]=read_byte();
 ow_reset();
 write_byte(0xCC); //跳过 ROM
 write_byte(0x44); //开始翻转
 return temp.x/2;
}
```

程序五　keilc 6.20c 版直接嵌入汇编的方法

```
//<asm.h>
#ifdef ASM
      unsigned long shiftR1(register unsigned long);
#else
      extern unsigned long shiftR1(register unsigned long);
#endif
//end of asm.h

//<asm.c>
#define ASM
#include <asm.h>
#include <reg52.h>
#pragma OT(4,speed)
unsigned long   shiftR1(register unsigned long   x)
{
   #pragma asm
   clr c
   mov a,r4
   rrc a
   mov r4,a

   mov a,r5
   rrc a
```

```
  mov r5,a

  mov a,r6
  rrc a
  mov r6,a

  mov a,r7
  rrc a
  mov r7,a

  #pragma endasm
  return(x);
}
//end of asm.c
```

将此源文件加入要编译的工程文件，
将光标指向此文件，选择右键菜单“option for file 'asm.c'”，
将属性单“properties”中的“Generate Assembler SRC File”“Assemble SRC File”
两项设置成黑体的“√”将“Link Public Only”的“√”去掉，再编译即可。
用此方法可以在 c 源代码的任意位置用#pragma asm 和#pragma endasm 嵌入汇编语句。
但要注意的是在直接使用形参时要小心，在不同的优化级别下产生的汇编代码有所不同，可以察看对应的.lst 文件，得到正确的优化级别后，#pragma OT(x,speed)锁定
优化级别（这里的值是 0～9）。

程序六　用计算机并口模拟 SPI 通信的 C 源程序

```
#include
#include
#include
#include
#include

#define LPT_PORT  0x378
#define CLR_WCK(X)  {X=X&(~(1<<0)); outportb(LPT_PORT,X); } // data.0
#define SET_WCK(X)  {X=X | (1<<0) ; outportb(LPT_PORT,X); }
#define CLR_BCK(X)  {X=X&(~(1<<2)); outportb(LPT_PORT,X); }  // data.2
#define SET_BCK(X)  {X=X | (1<<2) ; outportb(LPT_PORT,X); }
#define CLR_DATA(X) {X=X&(~(1<<3)); outportb(LPT_PORT,X); }  // data.3
#define SET_DATA(X) {X=X | (1<<3) ; outportb(LPT_PORT,X); }
#define FALSE 0
#define TRUE  1
void test_comm()
{
unsigned char data ;
```