张志良　编著

80C51单片机实验实训100例

——基于Keil C和Proteus

- 实例项目内容丰富，便于选择
- 全部通过Keil调试和Proteus虚拟仿真
- 免费下载仿真DSN文件和驱动程序hex文件
- 电路与程序能直接或移植于实际工程项目
- 无需硬件板，全电脑软件实验实训操作
- 程序每条语句均给出注释，便于阅读理解

北京航空航天大学出版社
BEIHANG UNIVERSITY PRESS

80C51 单片机实验实训 100 例

——基于 Keil C 和 Proteus

张志良　编著

北京航空航天大学出版社

内 容 简 介

本书系单片机实验实训教材或单片机教学参考书。内容包括 C51 程序 Keil 调试、输出信号控制、片外扩展、显示、键盘、中断、定时/计数器、串行口、A-D/D-A、常用测控电路等 100 个应用实例，还编有 Keil C51 编译软件和 Proteus ISIS 虚拟仿真软件操作基础。读者可在 PC 机上，不涉及具体硬件实验设备，虚拟本书全部案例项目仿真运行。既能教学演示观赏，又可让学生课后边学边练、实验操作。

本书不配光盘，但可从网上(www. buaapress. com. cn)免费下载 100 实例仿真文件包，内含 Proteus 仿真电路 DSN 文件和驱动程序 hex 文件。100 实例全部通过 Keil 调试和 Proteus 虚拟仿真，电路与程序真实可靠，能直接用于或移植于实际工程项目。程序条例清晰，每条语句均有注释，便于阅读理解。本书适合本专科开设单片机课程的学校和学生使用。

图书在版编目(CIP)数据

80C51 单片机实验实训 100 例：基于 Keil C 和 Proteus/
张志良编著. -- 北京 ：北京航空航天大学出版社，
2015.1

ISBN 978-7-5124-1603-1

Ⅰ. ①8… Ⅱ. ①张… Ⅲ. ①单片微型计算机－高等学校－教学参考资料 Ⅳ. ①TP368.1

中国版本图书馆 CIP 数据核字(2014)第 235976 号

80C51 单片机实验实训 100 例——基于 Keil C 和 Proteus

张志良 编著

责任编辑 胡晓柏 张 楠

*

北京航空航天大学出版社出版发行

北京市海淀区学院路 37 号(邮编 100191) http://www.buaapress.com.cn

发行部电话：(010)82317024 传真：(010)82328026

读者信箱：emsbook@gmail.com 邮购电话：(010)82316936

北京楠海印刷厂印装 各地书店经销

*

开本：710×1 000 1/16 印张：22.75 字数：485 千字

2015 年 1 月第 1 版 2015 年 1 月第 1 次印刷 印数：3 000 册

ISBN 978-7-5124-1603-1 定价：49.00 元

前　言

单片机应用领域之广，几乎到了无孔不入的地步，自动化、数字化、智能化、信息化均离不开单片机的应用。因而高校工科类专业普遍开设了“单片机应用”课程。然而，单片机课程是一门实践性很强的课程，既需要学习理论知识，更需要实验实训应用。本书即为单片机实验实训应用教材，并有以下特点：

（1）基于 Keil C51 和 Proteus 软件。单片机实验实训需要配备价格不菲的开发设备，且各校硬件实验设备各不相同。本书编写基于 Keil C51 和 Proteus 软件，读者可在 PC 机上，不涉及具体硬件实验设备，虚拟单片机应用电路和目标程序调试运行。既能教学演示观赏，又可让学生课后边学边练、实验操作。使单片机教学变得相对方便和有效。

（2）网上免费下载仿真文件包。为降低书价不配光盘，将原光盘内容改为仿真文件包，内含 100 实例的 Proteus 仿真电路 DSN 文件和驱动程序 hex 文件，不设门槛，读者可以登录北京航空航天大学出版社网站 www. buaapress. com. cn 的“下载专区”免费下载。其中 hex 文件由书中相应程序在 Keil 编译时自动生成。可能有读者认为，自行输入冗长的 C51 程序，很不方便。但有利于感悟 C51 对程序输入的要求，这也是一个学习过程。况且，学习本书程序，不是简单地观看 Proteus 仿真运行效果，而是在理解的基础上，修改、验证、移植、拼接、创新，编写出自己的运行程序，并在 Proteus ISIS 虚拟电路上仿真运行。编者赞赏的是后一种学习方法，更能取得良好的学习效果。

（3）全部通过 Keil 调试和 Proteus 虚拟仿真。前 22 例因不涉及 80C51 单片机片外元件，无 Proteus 虚拟仿真，仅通过 Keil 调试；后 78 例全部通过 Keil 调试和 Proteus 虚拟仿真。因此，100 实例电路与程序真实可靠，能直接用于或移植于实际工程项目。

（4）实例项目内容丰富，便于选择。100 实例为常见/常用教学和工程案例，基本上能适用和满足绝大多数院校和专业的教学需求。但软件仿真不宜完全替代单片机实际硬件实验实训。编者建议：读者可根据本校硬件实验设备情况和专业需要，从

中选择部分案例，进一步硬件实验实训操作，以增强教学效果。

（5）程序条理清晰，每条语句均有注释，便于阅读理解。实例项目中，若遇有74系列TTL、CMOS4000系列、I^2C或其他接口电路芯片时，均给出电路芯片功能和应用介绍。

本书由上海电子信息职业技术学院张志良主编，邵瑛、邵菁、刘剑昀参编。其中第1、2章由邵瑛编写，第3、4章由邵菁编写，第5、6章由刘剑昀编写，其余部分由张志良编写并统稿。

限于编者水平，书中错误不妥之处，恳请读者批评指正（编者的Email：zzlls@126.com），有信必复。

张志良

2015年1月

目　录

第1章

C51 程序 Keil 调试

本章的实验实训例题不涉及外围电路元件，因此不用 Proteus，而用 Keil C51 纯软件调试。Keil C51 可以检测程序语法错误，观测程序运行的过程和结果。Keil C51 调试操作方法，可参阅本书第 8 章。

1.1 求 和

求和是单片机应用系统中的常见课题，也是学习 C51 编程的必备基础。

实例 1 sum＝1＋2＋…＋100

1. 程序设计

求和程序一般需用到循环语句，通常有 for 循环、while 循环和 do-while 循环 3 种形式。

(1) for 循环形式程序

```
void  main() {                    //主函数
  unsigned char  n;               //定义循环序号变量 n
  unsigned int  sum = 0;          //定义和变量 sum,并赋初值
  for(n = 1; n< = 100; n++)       //for 循环:初值 n = 1,条件 n< = 100,变量更新 n++
    sum = sum + n;                //循环体语句:累加求和(也可写成:sum += n;)
  while(1);}                      //原地等待,避免局部变量释放
```

(2) while 循环形式程序

```
void  main() {                    //主函数
  unsigned char  n = 1;           //定义循环序号变量 n,并赋初值
  unsigned int  sum = 0;          //定义和变量 sum,并赋初值
  while(n< = 100){                //while 循环:当 n≤100 时循环,否则跳出循环
    sum = sum + n;                //循环体语句:累加求和(也可写成:sum += n;)
    n++ ;}                        //修正循环变量,n = n + 1,并返回循环条件判断
  while(1);}                      //原地等待,避免局部变量释放
```

(3) do-while 循环形式程序

```
void  main() {                          //主函数
  unsigned char  n = 1;                 //定义循环序号变量 n,并赋初值
  unsigned int   sum = 0;               //定义和变量 sum,并赋初值
  do{sum = sum + n; n ++ ;}             //do - while 循环体语句:累加求和,并修正循环变量
  while(n< = 100);                      //循环条件判断:当 n≤100 时循环,否则跳出循环
while(1);}                              //原地等待,避免局部变量释放
```

2. Keil 调试

(1) 打开 μVision,建立工程项目,设置工程属性

双击桌面图标 μVsion( )后,进入工程编辑启动界面,按 8.1 节所述步骤方法操作:

① 选择 Project→New Project 菜单项,然后输入新项目名,选择路径,保存新项目,默认扩展名为".uV2";若打开已有项目,可选择"Open Project"。

② 保存新项目后,系统弹出选择单片机型号的对话框,可按需选择使用的单片机型号。例如,选择 Atmel 公司的 AT89c51 单片机。

③ Project→Options for Target 'Target 1',弹出工程属性设置对话框,该框中有 10 个标签页,大部分设置项都可以按默认值设置,有两处需要设置修改:一是 Target 标签页中的"Xtal(MHz)"框(默认值 24 MHz),可按需设置单片机的工作频率;二是 Output 标签页"Create Executable"框中的"Create Hex File"项默认为未选,若需要生成可执行 Hex 代码文件(用于写入单片机 ROM 或进一步 Proteus 虚拟电路仿真),则应选中打勾。

(2) 编写和输入源程序

编写源程序,以在 Word 中较为方便,而在 μVision 程序编写窗口,因幅面和字体较小,且不熟悉其功能图标和快捷键,编写相对不便。因此编者建议,先在 word 界面西文状态下编写源程序,然后再把该文本程序复制到 μVision 程序编写窗口。

需要特别提醒的是,程序语句中不能加入全角符号。例如全角的分号、逗号、圆括号、引号、大于小于号等。否则,编译器都将这些全角符号视作语法出错。

(3) 程序编译链接及语法纠错

① 直接单击图 8-19、图 8-20 工具栏中编译图标( ),在屏幕下方输出窗口的 Build 标签页中,将出现图 8-21 所示的编译信息。若显示"0 Error(s),0 Warning(s)",表示源程序语法无错;否则,会有错误报告示出,双击该行可以定位到出错的位置,修改后重新编译,直至全部修正完毕。

② 单击图 8-19、图 8-20 工具栏中链接制作图标( ),在屏幕下方输出窗口的 Build 标签页中将出现图 8-22 所示的链接信息。若显示"0 Error(s),0 Warning(s)",表示整个编译链接过程完成,可进入程序调试阶段。

(4) 进入调试状态，打开有关观测窗口

① 单击图 8-23 工具栏中进入/退出调试状态的图标按钮()，此时程序处于待运行状态。

② 单击图 8-23 工具栏中图标()，打开变量观测窗口，如图 8-29 所示，Locals 标签页显示程序中两个局部变量：n 和 sum，值均为 0。显示值形式可选择十进制数(Deciml)或十六进制数(Hex)，右击“Value”，弹出“Number Base”选项及其下拉式菜单，如图 8-31 所示，可选择显示值形式。

调试时可有多种选择：单步、全速等。本例只需观测程序运行的最终结果 sum 值，可先观测全速运行方式。

(5) 程序调试

① 单击图 8-25 调试工具条中全速运行图标()，此时调试工具条中暂停图标()变成红色(表示被激活，可操作)，同时变量观测窗口 Locals 标签页中局部变量 n 和 sum 消隐，单击红色暂停图标，该图标复原为灰色，Locals 标签页恢复显示：n=101，sum=5050。表示 n=101 时停止累加，之前累加值 sum=5050。

需要说明的是，Locals 页只能显示当前运行函数的局部变量，例如本函数的局部变量 n 和 sum。函数运行结束，这些局部变量就会被释放(释放后就观察不到了)。为此，本例函数在程序末尾加了一句 while(1)，表示程序运行尚未结束，可避免局部变量释放，这样就可以从 Locals 页读到 n 和 sum 的值。

但若在变量观测窗口 Watch#1 或 Watch#2 标签页按图 8-30 所示方法，设置观测变量 n 和 sum，则无论有否 while(1)语句，该标签页均有 n 和 sum 值显示。设置方法是：单击该标签页窗口中<type F2 to edit>，然后按 F2 健，再输入变量名 n，回车；再次单击<type F2 to edit>，按 F2 健，用同样方法输入变量名 sum，即能设置并显示该两个变量的动态值。

② 若选择单步运行，可观察到程序运行过程。不断单击单步运行图标()，程序逐行依次运行，变量观测窗口 Locals 标签页中 n 和 sum 值依次逐步增加：n=1，sum=0；n=2，sum=1；n=3，sum=3；n=4，sum=6；n=5，sum=10；…；n=101，sum=5050。注意，n 值的变化总是先于 sum 值的变化。

需要说明的是，本例 3 种形式循环语句程序运行的最终结果 sum 值相同，但运行过程和路径略有不同。读者可在 Keil 调试中，体会、比较上述 3 种程序运行过程的区别。

③ 修改变量 n 赋值数据，重新编译链接调试，可得到不同 n 值的程序运行结果。

3. 思考与练习

3 种形式的循环语句程序有什么区别？

实例 2　sum＝1＋3＋5＋…＋99

1. 程序设计

本例与实例 1 类同，也可编写 for 循环、while 循环和 do-while 循环 3 种形式的程序。

(1) for 循环形式程序

```
void  main() {                    //主函数
  unsigned char  n;               //定义循环序号变量 n
  unsigned int  sum = 0;          //定义和变量 sum,并赋初值
  for(n = 1; n< = 99; n = n + 2)  //for 循环:初值 n = 1,条件 n< = 99,变量更新 n = n + 2
    sum = sum + n;                //循环体语句:累加求和(也可写成:sum += n;)
  while(1);}                      //原地等待,避免局部变量释放
```

(2) while 循环形式程序

```
void  main() {                    //主函数
  unsigned char  n = 1;           //定义循环序号变量 n,并赋初值
  unsigned int  sum = 0;          //定义和变量 sum,并赋初值
  while(n< = 99) {                //while 循环:当 n≤99 时循环,否则跳出循环
    sum = sum + n;                //循环体语句:累加求和
    n = n + 2;}                   //修正循环变量,n = n + 2,并返回循环条件判断
  while(1);}                      //原地等待,避免局部变量释放
```

(3) do-while 循环形式程序

```
void  main() {                    //主函数
  unsigned char  n = 1;           //定义循环序号变量 n,并赋初值
  unsigned int  sum = 0;          //定义和变量 sum,并赋初值
  do{sum = sum + n; n = n + 2;}   //do - while 循环体语句:累加求和,并修正循环变量
  while(n< = 99);                 //循环条件判断:当 n≤99 时循环,否则跳出循环
  while(1);}                      //原地等待,避免局部变量释放
```

2. 同类题:sum＝2＋4＋6＋…＋100

偶数求和与奇数求和类同，仅循环初值、终值不同，也可编写 for 循环、while 循环和 do-while 循环 3 种形式的程序。

(1) for 循环形式程序

```
void  main() {                    //主函数
  unsigned char  n;               //定义循环序号变量 n
  unsigned int  sum = 0;          //定义和变量 sum,并赋初值
  for(n = 2; n< = 100; n = n + 2) //for 循环:初值 n = 1,条件 n< = 100,变量更新 n = n + 2
    sum = sum + n;                //循环体语句:累加求和(也可写成:sum += n;)
  while(1);}                      //原地等待,避免局部变量释放
```

(2) while 循环形式程序

```
void  main() {                    //主函数
  unsigned char  n = 2;           //定义循环序号变量 n,并赋初值
  unsigned int  sum = 0;          //定义和变量 sum,并赋初值
  while(n< = 100) {               //while 循环:当 n≤100 时循环,否则跳出循环
    sum = sum + n;                //循环体语句:累加求和
    n = n + 2;}                   //修正循环变量,n = n + 2,并返回循环条件判断
  while(1);}                      //原地等待,避免局部变量释放
```

(3) do-while 循环形式程序

```
void  main() {                    //主函数
  unsigned char  n = 2;           //定义循环序号变量 n,并赋初值
  unsigned int  sum = 0;          //定义和变量 sum,并赋初值
  do{sum = sum + n; n = n + 2;}   //do - while 循环体语句:累加求和,并修正循环变量
  while(n< = 100);                //循环条件判断:当 n≤100 时循环,否则跳出循环
  while(1);}                      //原地等待,避免局部变量释放
```

3. Keil 调试

本例 Keil 调试同实例 1,奇数求和程序运行结果:n=101,sum=2500。偶数求和程序运行结果:n=102,sum=2550。

4. 思考与练习

奇数求和程序与偶数求和程序有什么不同?

实例 3　sum=1!+2!+…+10!

1. 程序设计

```
void  main() {                    //主函数
  unsigned char  n;               //定义循环序号变量 n
  unsigned long  fac = 1;         //定义阶乘变量 fac,并赋初值
  unsigned long  sum = 0;         //定义和变量 sum,并赋初值
  for(n = 1; n< = 10; n++) {      //for 循环:初值 n = 1,条件 n< = 10,变量更新 n++
    fac * = n;                    //循环体语句:求阶乘
    sum += fac;}                  //循环体语句:累加求和
  while(1);}                      //原地等待,避免局部变量释放
```

2. Keil 调试

本例 Keil 调试同实例 1,程序运行结果:n = 11,fac = 10! = 362880,sum =4037913。

3. 思考与练习

变量 fac 和 sum 的数据类型为什么要定义为“long”?

1.2　排　序

实例 4　a、b、c 从小到大排序

已知 80C51 内 RAM 以 30H 为首地址的 3 个单元中分别存放无符号字符型数据 a、b、c，试按从小到大次序将它们重新存放在该 3 个单元中。

1. 程序设计

```
unsigned char data  a _at_ 0x30;     //定义变量 a 绝对地址内 RAM 30H
unsigned char data  b _at_ 0x31;     //定义变量 b 绝对地址内 RAM 31H
unsigned char data  c _at_ 0x32;     //定义变量 c 绝对地址内 RAM 32H
void  main() {                       //主函数
  unsigned char  m;                  //定义暂存器 m
  a = 3;b = 2;c = 1;                 //变量 a、b、c 赋值(<256,可随意设置)
  if(a>b)                            //第一个 if 语句,若 a>b
    if(b>c)  {m = a;a = c;c = m;}    //if - else 语句嵌套。若 a>b 且 b>c,则 ac 交换
    else  {m = a;a = b;b = m;}       //若 a>b 且 b<c,则 ab 交换
  else                               //对应于 if(a>b),即 a<b
    if(a>c)  {m = a;a = c;c = m;}    //属于 else 的内嵌语句。若 a<b 且 a>c,则 ac 交换
  if(b>c)  {m = b;b = c;c = m;}}     //第二个 if 语句,若 b>c,则 bc 交换
```

2. Keil 调试

① 按实例 1 所述步骤，编译链接并进入调试状态。

② 单击图 8 - 25 调试工具条中存储器窗口图标()，打开存储器窗口。在 Memory＃1 窗口的 Address 编辑框内键入“d:0x30”，以便观测 RAM 内 30H～32H 数据。

③ 先单步运行()一步，看到 Memory＃1 窗口内 30H、31H、32H 已分别存入 a、b、c 初始赋值数据。

④ 然后全速运行()，看到 RAM 30H～32H 3 个单元中的数据已按从小到大次序重新排列。

⑤ 改变程序中 abc 原始数据排列(按大小顺序共有 6 种)，重新编译链接进入调试状态，运行程序，执行结果均相同。

3. 思考与练习

试述第一个 if 语句执行后的结果是什么？

实例 5　数组 a[8] 从大到小(从小到大)排序

已知数组 a[8]={11,66,22,55,44,77,88,33}，试将其从大到小排列。

1. 程序设计

```
void  main() {                          //主函数
  unsigned char  i,j,k,m;               //定义循环序号变量 i、j,最大值序号 k,暂存器 m
  unsigned char  a[8] = {11,66,22,55,44,77,88,33};  //定义数组 a[8],并赋值
  for(i = 0; i<7; i++) {                //for 循环 1,循环变量 i,选择法排序
    k = i;                              //最大值序号 k 赋值,设最大值为首个元素
    for(j = i; j<8; j++)                //for 循环 2,循环变量 j,选出最大值
      if(a[k]<a[j])  k = j;             //与后续元素比较,若 a[k] <a[j],最大值序号变更
    m = a[k];a[k] = a[i];a[i] = m;}}    //交换位置
```

2. Keil 调试

① 按实例 1 所述步骤,编译链接并进入调试状态。

② 单击调试工具条中变量观测窗口图标(),打开变量观测窗口,观察 Locals 标签页中数组 a 的首地址为 0x09(注意:若编制程序不同,a 的首地址可能也不同)。

③ 单击调试工具条中存储器窗口图标(),打开存储器窗口,在 Memory＃1 窗口的 Address 编辑框内键入“d:0x09”(0x09 是根据变量观测窗口 Locals 页中数组 a 的首地址)。右击其中任一单元,在右键菜单中选择 Decimal(十进制)(参阅图 8-31)。

④ 先单步运行()一步,看到 Memory＃1 窗口首地址 0x09 的 8 个连续单元显示数组 a[]原始排列数据。

⑤ 全速运行()后,Memory＃1 窗口首地址 0x09 的 8 个连续单元显示从大到小排序数据:88、77、66、55、44、33、22、11。

⑥ 改变程序中数组 a 的原始数据排列,重新编译链接进入调试状态,观察运行结果。

3. 从小到大排序程序

a[8]从小到大排序,只需将上例程序第 7 行语句中“a[k]<a[j]”改为“a[k]>a[j]”即可。

```
void  main() {                          //主函数
  unsigned char  i,j,k,m;               //定义循环序号变量 i、j,最小值序号 k,暂存器 m
  unsigned char  a[8] = {11,66,22,55,44,77,88,33};  //定义数组 a[8],并赋值
  for(i = 0; i<7; i++) {                //for 循环 1,循环变量 i,选择法排序
    k = i;                              //最小值序号 k 赋值,设最小值为首个元素
    for(j = i; j<8; j++)                //for 循环 2,循环变量 j,选出最小值
      if(a[k]>a[j])  k = j;             //与后续元素比较,若 a[k] <a[j],最小值序号变更
    m = a[k];a[k] = a[i];a[i] = m;}}    //交换位置
```

数组 a[8]从小到大排序 Keil 调试与从大到小排列 Keil 调试相同。