8051 单片机技术应用系列

光盘内含书中范例源程序

8051 单片机

彻底研究 经验篇

林伸茂 编著

中国电力出版社
www.infopower.com.cn

8051单片机技术应用系列

8051 单片机

彻底研究 经验篇

林伸茂 编著

中国电力出版社
www.infopower.com.cn

内 容 提 要

本书共分成三大部分：第一部分是应用研究篇，介绍8051单片机的诸多设计理念与硬件保护手段。第二部分是介绍以FLAG51为主体的烧录器，包括Atmel Flash Microcontroller的烧录、EPROM的烧录以及E^2PROM的烧录等。第三部分是历年来作者接触到设计项目的心得以及对一些问题的看法。

本书属于8051进阶级书籍，适合对单片机已经有一些经验的读者，或了解软硬件开发的工程师阅读。

图书在版编目（CIP）数据

8051单片机彻底研究——经验篇／林伸茂编著. —北京：中国电力出版社，2007
（8051单片机技术应用系列）
ISBN 978-7-5083-5153-7

Ⅰ.8… Ⅱ.林… Ⅲ.单片微型计算机—基本知识Ⅳ.TP368.1
中国版本图书馆CIP数据核字（2007）第004573号

北京市版权局著作权合同登记号 图字：01-2006-5849号

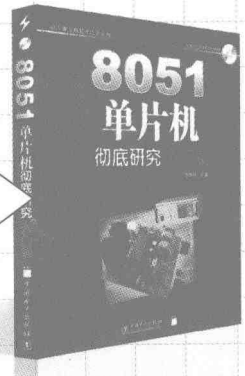
版权声明

责任编辑：刘 炽
责任校对：崔燕菊
责任印制：李文志

丛 书 名：8051单片机技术应用系列
书 名：8051单片机彻底研究——经验篇
编 著：林伸茂
出版发行：中国电力出版社
地址：北京市三里河路6号 邮政编码：100044
电话：（010）68362602 传真：（010）68316497
印 刷：北京丰源印刷厂
开本尺寸：185 × 260 印 张：17 字 数：408千字
书 号：ISBN 978-7-5083-5153-7
版 次：2007年5月北京第1版
印 次：2007年5月第1次印刷
印 数：0001—4000
定 价：32.00元（含1CD）

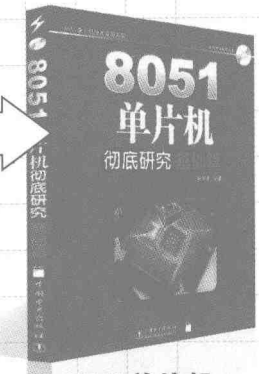
8051 单片机学习地图

8051 单片机的应用到处都是，可是我一点基础都没有，要怎样开始学习呢?



8051单片机
彻底研究——入门篇

我对 8051 已经有了基本的认识，我想更进一步彻底学好 8051 汇编语言!



8051单片机
彻底研究——基础篇

要想学好 8051，练习是少不了的，我希望找一些有趣的题目自己动手做做看，不然怎样写都是一些简单的范例。

8051单片机
彻底研究——实习篇

基本功都练扎实了，想进一步提高水平，吸取前辈的经验是最好的办法。

8051单片机
彻底研究——经验篇

序　言

知识分享的喜悦

20年来我一直坐在研究与开发的位置上，曾经在风和日丽的天气下进入台电核三厂的控制室内，为的是协助找出发电机组跳闸的原因；曾经在寒风冷雨的冬季到新竹工研院，查看仪器连线为何经常出问题的原因；曾经带着工具箱与手册，到鲜血尚未清洗干净的外科手术室里，检查自己设计的电动手术台是否有足够的安全性和稳定性；也曾经参与电动环保汽车的部分研发工作。

不管这些项目的结果如何，都有一个共同点：它们都是 8051 单片机的应用范例。每一个项目都是8051的深度应用。这也是10多年前我开始在《RUN!PC》杂志上写8051应用实例、陆陆续续大约写了三四十篇技术性文章的原因。与此同时，我个人的系统设计工作始终没有间断过。

20年来，我们碰到了许多技术上的瓶颈，很幸运我们解决了大部分，剩下的问题很可能不属于纯粹的技术问题了。那么这些宝贵的资料与经验永远放在旗威科技公司的文件柜中吗？我选择了出书这一方式，来把这些经验提供给广大读者朋友，虽然这样做很辛苦，但是套一句老话“喜欢做，甘愿受”。纵使外面的环境非常复杂，我个人还是认为值得这样做的。

千万不能只顾着吃鱼，也要学会打鱼。我个人觉得做任何研究或写程序应进一步掌握其实现过程。在本书中您会看到一件产品是被如何开发出来的，以及为何要这样处理，我们在书中都会有详细的说明。

我经常告诉别人：健康不能分享，金钱也不能分享，但是知识除了独享外，可以与别人分享。旗威科技公司很乐意邀请您一起来分享8051的专业知识，一起分享我们写的程序。

系列书籍介绍

《8051单片机彻底研究——经验篇》作为8051单片机系列丛书的一部分，与《8051单片机彻底研究——入门篇》《8051单片机彻底研究——基础篇》、《8051单片机彻底研究——实习篇》构成了一个系统的、有机的专业技术教程丛书。

如果您想彻底了解8051单片机与汇编语言的写法，请务必参考《8051单片机彻底研究——基础篇》，书中会对8051各个指令的用法与寄存器的操作，做了深入详尽的交待。而《8051单片机彻底研究——实习篇》主要强调在8051单片机系统扩展与集成应用两方面专业知识。

内容收录

本书的内容取材主要有以下三个方面：

历年来我们运用8051单片机的实例应用。旗威科技有限公司用8051单片机设计过100种以上的工业产品，产品范围涵盖医疗、精密测量、通信及自动化等专业领域。

取自《RUN!PC》杂志历年来所发表的技术与评论性文章，并删除部分不合时宜的内容，

最后又加上我们最近几年的设计心得。

网络上有关 8051 单片机应用与 C 语言的技术文献与报道，这方面我们深信外国人开放，许多资料在网站上唾手可得，不过只有您仔细消化后，才可以算是您自己的东西。

如何阅读本书

本书属于 8051 进阶级书籍，适合对单片机已经有一些经验的读者，或横跨软硬件的工程师阅读。我们希望您在阅读本书时，已经对 8051 的结构与程序有基本的认识，例如 SFR、bit addressable、SBUF 等等。

本书共分成三大部分：

第一部分是应用研究篇，介绍 8051 单片机的诸多设计理念与硬件保护手段。想要以子之矛攻子之盾，在这里有第一手的解答。我们也提到与硬件保护始终对立的逆向工程与 8051 之间的关系。最后还有一章专门提到 8051 的优势与缺憾。8051 当初被人称道的功能至今被贬为重大瑕疵，为何如此？请看我们的深入分析。

第二部分是介绍以 FLAG51 为主体的烧录器，包括 Atmel Flash Microcontroller 的烧录、EPROM 的烧录以及 E^2PROM 的烧录等，我们很乐意地公布这些烧录程序，而且是 100%的源代码公布。以前我们彻底研究别人的程序为的是理解当初设计者的理念。今天我们公布程序内容，欢迎您来研究我们写的程序，如果您是用心的读者绝对可以从中获得更多的启示。

第三部分是历年来我们接触到设计项目的心得以及我们对一些问题的看法。

致谢

编写 8051 单片机系列书绝对不是单一个人所能完成的，它绝对是一个团队的工作总整合，3 年前我就开始筹备新书的出版事宜，所有的文章与内容经过整理过滤与调整补充。我要特别感谢以下帮助我的人们：

王圣心小姐与姜莹贞小姐：初步整理已发表过的文章，光是校稿就校了多次，并拍摄许多额外的照片，让本系列的书籍得以完成初步的架构。

李浩蓁先生与曾琼惠小姐：进行本书版面调整与最后的校稿，整本书是在他们的手中完成的。

罗仕林、庄昱宏与黄芳川先生：提供最高级的示波器与逻辑分析仪，以及技术上的协助，让本书的图表资料与数据更有看头。

凌全伯先生：提供了对 8051 另类的观点，在本书第 6 章“8051 的是是非非”中，他也加入了独到的见解，面对 8 位 CPU 就要有 8 位的思维。

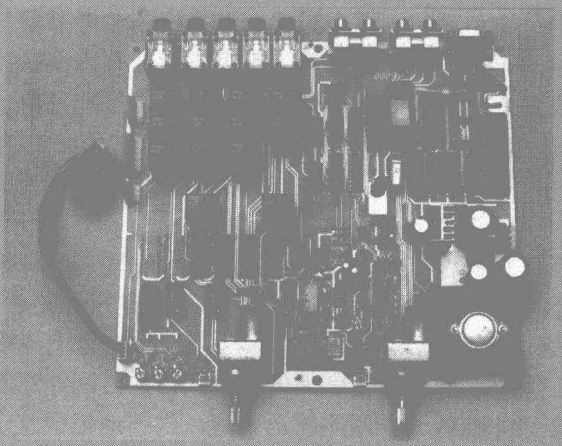
最后，我还是要谢谢家人所给予的鼓励，尤其是刚在牙牙学语的小女儿，没有他们几近狂热的激励与支持，就不会有本系列丛书的问世。

林伸茂

chipware@chipware.com.tw

目　录

1



KTV 使用的点歌与视频控制器，含红外线摇控及音量控制，右侧上方为 8052 单片机。

第1章　8051的汇编语言与C语言

有些程序设计人员对汇编语言说“恨之入骨”也不为过，习惯用C语言来设计系统。其实C语言和汇编语言各有其应用的空间，当你需要程序要精简且速度超快时，汇编语言绝对是首要之选。选择用C来开发就一帆风顺了吗?那也未必，欲知详情，请看本章彻底的分析。

最近，在BBS上的Electronic（即关于电子设备的）板上经常看到这样的讨论：8051的程序开发，是选用汇编语言较好，还是使用C语言较好?两种语言的支持者都引经据典并结合自己的实践经验为自己辩解。其实双方的说法都有道理。笔者在汇编语言与C语言的应用开发上都有超过10年以上的实践经验，愿意把自己从书本上看不到的体验与您分享。本章最终的目的并不是断定哪种语言是最好的，但是会引导读者根据自身情况以及开发的价格选用最适合自己的8051开发语言。

1-1　汇编语言短小精干

从学生时代开始我就使用汇编语言来写硬件的控制程序，最初是LED的灯号控制及七段显示器的数值显示，到多轴步进电动机的实时控制，都是借用强而有力的汇编语言直接控制CPU的运行。早期没有PC（个人计算机）及Apple II微电脑的时代，我是使用所谓的微电脑学习机来学习汇编语言的。首先我把汇编程序写在记事本的右方，然后由微电脑的DATABOOK上查找该指令的机器码（MACHINE CODE），将此机器码写在记事本的原始程序左侧，再将此机器码以十六进制码的方式一个接一个地输入到微电脑学习机上，最后一个操作是按下学习机上的“GO”键开始执行我们先前辛苦输入的程序。以上所有的操作都是手动的，除错的步骤当然也是纯手动的。那个时代有许多好的参考书籍与资料，反倒是现在资料变少了，想自己动手做的人也成了稀有品种。程序不幸若有错误时，所有的步骤都要重来一次，这也促使我们养成每个步骤都做双重核对的习惯，而且把每个子程序都模块化，以便其他场合也可派上用场，因为写汇编语言是相当艰苦的，有些程序上的臭虫（bug）要几星期的奋斗才可以确认出来，有些不应该出错的地方竟然出错了，这些苦头只有曾经亲自尝试过的人才有可能体会。

个人计算机流行之后，写程序就没这么累了，程序的编写与编译都可以在计算机上做，剩下的操作就是将程序下传给待测的控制板，然后确认所有的操作是否都如同原先所设置的。每次打印汇编程序的报表时，数据长度超过20页是很正常的，还有许多程序都是数百页以上的规模。我们的经验是汇编语言的程序长度加倍时，除错的时间绝对超过原来的两倍。开发这些程序的同时，我都会另外准备一本厚厚的记事簿，详细地记载整个开发的过程以及修改的地方，当然也包括了所有的测试步骤及方式。在原始程序SOURCE PROGRAM上我们也做了详尽的操作说明。

有人做过一个统计：开发汇编语言程序时，每20行大约需要1 man-day的成本；这也就是说：一个有1000行的汇编程序，若交由一个有经验的工程师来处理，至少要花上50天的时间才能完成。一个人一天只能完成20行的程序，好像有点不可思议!但是依我们多年的经验来看，这个说法在写汇编语言的场合中确实是对的，程序越大时这个说法越有其正确性。

由于汇编语言是直接控制系统的所有输入/输出点及存储器，所以刚开始写程序时，一定常会碰到系统突然死在某个循环中，无法跳出该循环的状态，最后只好采用最直接且最有效的方法——关机重来。写汇编语言对程序设计师来讲，是有相当的杀伤力及折磨性的，每次新的程序开始时，所有的程序小环节都要逐一测试与验证，所以压力是一定有的，完成一个程序后就好像历经一次母亲怀胎到生产的过程一样。写了多个大的汇编程序后，若无法适度调整心态及善用辅助的开发工具，很可能从此就永远不碰汇编程序了。真的，何苦来哉！

1-2 C语言可以缩短开发时间

在C语言成功地由UNIX移转到PC后，写汇编语言的梦魇终于解除了，接着我们也可以看到适合各种CPU的C语言开发系统，其价格从几百元到上万元都有，这些开发软件工具的广告可在国外知名的专业杂志上看到，当然国内的CPU在线仿真器(In Circuit Emulator)开发厂商也会依国内用户的要求进口部分产品，但是数量不多，而且这些厂商对这方面的专业知识也是严重缺乏，所以凡事只有靠自己了。

我们先学习的是PC上的C语言程序（Microsoft C及Turbo C），这段学习时间至少持续一年以上，再进而购入2500AD的8051C语言编译程序，将我们原先的汇编程序改用C程序来处理，这时我们才发觉一个简单的控制程序若用C语言来撰写，要限时在7天以内完成是非常有可能的!举一个简单的例子来说：若要让8051的P1端口产生类似指示灯的LED亮法，用汇编语言的写法大概是这样的：

程序1　用汇编语言写成的指示灯程序

```
                    ORG     0000H
START   NOV         R1,#00H             ;加入这段小DELAY
$1                  DJNZ    R1,$1       ;等待系统稳定后再开始
                    MOV     SP,#60H     ;STACK值的设置
LOOP                MOV     A,#01H
LOOP_NMOV           P1,A
                    CALL    DELAY
                    RL      A
                    CJNE    A,#80H,LOOP_N
                    SJMP    LOOP
;
;延迟一小段时间，以便可以观看出其变化
DELAY   MOV         R0,#00H
$1                  MOV     R1,#00H
$2                  DJNZ    R1,$2
                    DJNZ    R0,$1
                    RET
```

若改用2500AD的C语言来写时，可就轻松多了。看起来简单明了，何时看程序都是一清二楚的。唯一的缺憾是用C转换出来的程序代码过大，下面这个程序链接了链接库后，最

后的程序空间可能会超过 600 字节。

程序 2　改用 C 来处理的指示灯程序

```
#include"c8051io.h"
#include"c8051sr.h"
main()
{
car out;
     out=0x01;
     while(1)
     {
     P1=out;
     out=out<<1;/*shift left*/
     delay();
     if(out= =0x80)out=0x01;
     }
}

int delay()
{
int m;
   for(m=0;m<1000;m++){};/*just delay,do nothing*/
}
```

以下就是上述 C 语言程序经过转译成 8051 汇编语言的程序，由于篇幅的限制，我们仅列出程序重要的部分，其余的声明部分都暂时删除，这是由 2500AD 公司的 8051 C 编译程序转译的结果，改用其他编译器时结果可能不会相同。

```
32  0000                    _main:          .equal$
33                          ;
34                          ;
35  0000                    ?ASC0:          .equal0
36  0000                    ?TSC0:          .equal0
37  0001                    ?LSC0:          .equal1
38  0000        C2 AF                       clr     ea
39  0002        EE                          mov     a,r6
40  0003        24 FF                       add     a,#.low.-?TSC0-?LSC0
41  0005        FE                          mov     r6,a
42  0006        EF                          mov     a,r7
43  0007        34 FF                       addc    a,#.high.-?TSC0-?LSC0
44  0009        FF                          mov     r7,a
45  000A        D2 AF                       setb    ea
46                          ;out=0x01
47  000C        90 00 00                    mov     dptr,#-1+?LSC0
48  000F        74 01                       mov     a,#1
49  0011        12 00 00                    lcall   st_oper1_stack_char
50                          ;while
51  0014                    ?TOL0:          .equal $
52  ;(1)
53  ;
54  0014        79 00                       mov     r1,#0
```

```
55  0016        74 01                       mov     a,#1
56  0018        C3                          clr     c
57  0019        99                          subb    a,r1
58  001A        70 03       jnz             +3
59  001C        02 00 55                    ljmp    ?BOL0
60  001F                    ?PLC0:          .equal $
61                          ;P1=out
62  001F        90 00 00                    mov     dptr,#-1+?LSC0
63  0022        12 00 00                    lcall   __ld_oper1_stack_char
64  0025        F5 00                       mov     .low._P1,a
65                          ;out=out<<1
66  0027        79 01                       mov     r1,#1
67  0029        90 00 00                    mov     dptr,#-1+?LSC0
68  002C        12 00 00                    lcall   __ld_oper1_stack_char
69  002F        12 00 00                    lcall       __sl_char
70  0032        90 00 00                    mov     dptr,#-1+?LSC0
71  0035        12 00 00                    lcall   __st_oper1_stack_char
72                          ;delay()
73  0038        12 00 62                    lcall_delay
74                          ;(out==0x80)
75  003B        79 80               mov     r1,#128
76  003D        90 00 00                    mov     dptr,#-1+?LSC0
77  0040        12 00 00                    lcall   __ld_oper1_stack_char
78  0043        C3                          clrc
79  0044        99                          subba,r1
80  0045        60 03               jz      +3
81  0047        02 00 52                    ljmp    ?BOL1
82  004A                    ?PLC1:  .equal  $
83                          ;out=0x01
84  004A        90 00 00                    mov     dptr,#-1+?LSC0
85  004D        74 01               mov     a,#1
86  004F        12 00 00                    lcall   __st_oper1_stack_char
87  ;;
88                          ;
89                          ;
90  0052                    ?BOL1:  .equal $
91  0052        02 00 14            ljmp    ?TOL0
92  0055                    ?BOL0:  .equal $
93                          ;
94  0055                    ?BOF0:  .equal $
95  0055        C2 AF               clr ea
96  0057        EE                          mov     a,r6
97  0058        24 01               add     a,#.low.?LSC0+?TSC0+0
98  005A        FE                          mov     r6,a
99  005B        EF                          mov     a,r7
100 005C        34 00               add     ca,#.high.?LSC0+?TSC0+0
101 005E        FF                          mov     r7,a
102 005F        D2 AF                       setb    ea
103 0061        22                          ret
104                         ;
```

```
105              0062          _delay:.equal $
106                            ;
107
108              0000          ?ASC1:      .equal 0
109              0000          ?TSC1:      .equal 0
110              0002          ?LSC1:      .equal 2
111 0062         C2 AF                 clr ea
112 0064         EE                        mov     a,r6
113 0065         24 FE                 add a,#.low.-?TSC1-?LSC1
114 0067         FE                        mov     r6,a
115 0068         EF                        mov     a,r7
116 0069         34 FF                 addc    a,#.high.-?TSC1-?LSC1
117 006B         FF                        mov     r7,a
118 006C         D2 AF                 setb    ea
119                            ;m=0;
120 006E         90 00 00              mov     dptr,#-2+?LSC1
121 0071         78 00                 mov r0,#.low.0
122 0073         7A 00                 mov r2,#.high.0
123 0075         12 00 00                  lcall_st_oper1_stack_int
124 0078                       ?FLC2:  .equal $
125                            ;m<1000;
126 0078         79 E8                 mov r1,#.low.1000
127 007A         7B 03                 mov r3,#.high.1000
128 007C         90 00 00                  mov     dptr,#-2+?LSC1
129 007F         12 00 00                  lcall   _ld_oper1_stack_int
130 0082         12 00 00                  lcall       __cmp_int
131 0085         30 E7 06                  jnb     acc.7,+6
132 0088         30 D2 09                  jnb     ov,+9
133 008B         02 00 B1                      ljmp        ?BOL2
134 008E         20 D2 03                  jb      ov,+3
135 0091         02 00 B1                      ljmp        ?BOL2
136 0094         02 00 AE                      ljmp        ?FLB2
137 0097                       ?FLA2:      .equal $
138 ;m++)
139 0097         90 00 00                  mov     dptr,#-2+?LSC1
140 009A         12 00 00                  lcall   __ld_oper1_stack_int
141 009D         E8                        mov     a,r0
142 009E         24 01                 add     a,#1
143 00A0         F8                        mov     r0,a
144 00A1         EA                        mov     a,r2
145 00A2         34 00                 addc    a,#0
146 00A4         FA                        mov     r2,a
147 00A5         90 00 00                  mov     dptr,#-2+?LSC1
148 00A8         12 00 00                  lcall   __st_oper1_stack_int
149 00AB         02 00 78                      ljmp        ?FLC2
150 00AE                       ?FLB2:      .equal $
151                            ;
152 00AE         02 00 97                      ljmp        ?FLA2
153 00B1                       ?BOL2:      .equal $
```

```
154                             ;
155 00B1                        ?BOF1:      .equal $
156 00B1        C2 AF                       clr ea
157 00B3        EE                              mov     a,r6
158 00B4        24 02                       add a,#.low.?LSC1+?TSC1+0
159 00B6        FE                              mov     r6,a
160 00B7        EF                              mov     a,r7
161 00B8        34 00                       add ca,#.high.?LSC1+?TSC1+0
162 00BA        FF                              mov     r7,a
163 00BB        D2 AF                       setb    ea
164 00BD        22                                  ret
```

1-3 8051编译器使用经验谈

写8051的汇编程序时是完全无法取巧的。如果刚开始学习8051的程序语言，我们强烈建议由汇编语言开始学起，因为只有这样才能促使学习者完全理解整个8051 CPU动作的来龙去脉。不会设计线路也不要紧，因为市面上8051书籍多得很，只要参考几种现成的线路后，就可以依样照做了，可是汇编程序的撰写，除了自己以外，别人都无法帮得上!我们的经验是别人的程序看得越多，你的程序架构就会写得越好。如果有好的写法或子程序也可以在自己充分地吸收后归为己有。如果身旁收集的工具程序很多，程序开发的时间会相对地减少许多，不需要每次都从头开始除错。

市面上的8051参考书籍适合初学者的居多，可能无法找到相当理想的参考程序，我们建议可以到旗威科技公司的网站（http://www.chipware.com.tw）上获取以下几个相当有参考价值的参考程序：

```
readtemp.asm  温度的测量显示与联机串行通信程序。
flagdisp.asm  七段显示器的控制程序。
```

以上程序除了包含有许多常用的汇编语言子程序外，都有使用定时中断及串行通信等等的使用范例，必须逐一推敲所有的程序，才能对控制系统有更进一步的理解与认识。这些学习的经历必须自己走过才算是把8051给带进门来。也可以把许多8051应用产品的程序EPROM拔下来，用EPROM烧录器将程序数据转存在PC机中，然后用8051的反汇编程序DIS51.EXE（请自行到旗威科技公司的网站中下载此程序）将原先的机器码翻译成汇编语言的格式，再来分析其程序的写法与控制模式。我们所认识的汇编语言高手中，有不少人是用这种方式去磨练的，当然也包括我个人在内。这种学习方法最辛苦，但是收获却是最大的。我曾经用最原始的方法读过多个Z80、6502及8051满载64KB的系统程序，所以现在碰到类似的系统时，都可以在最短的时间内了解问题所在。至于旗威科技公司开发的其他8051应用程序，请不要全面予以反汇编，因为这些程序有大多数是用8051的C语言来处理的，程序规模相当大，全部通过堆栈传递参数，没有相当的程序基础是完全无法了解的。

我们另一方面也建议：如果有多余预算的话，不一定非得买ICE不可，但是买一台较像样的示波器，绝对可以在8051程序的开发过中帮上大忙。以前我们只用一台20MHz的Hitachi（日立）示波器，再通过汇编程序中的无穷循环，就可以对硬件线路及系统程序进行全面性的除错。近几年来我们发现若用数字存储示波器DSO（Digital Storiage Oscilloscope）来除错更是方便，尤其是串行通信及AD转换的程序除错，至少可以减少一半以上的时间。