单片机开发环境μVision2使用指南及USB固件编程与调试

尹 勇 王洪成 编著

- 单片机Keil C51集成开发环境入门精华
- μVision2使用详解
- USB固件C语言编程与调试
- 以经验为基础，实例丰富

北京航空航天大学出版社

单片机开发环境 μVision2 使用指南及 USB 固件编程与调试

尹　勇　王洪成　编著

北京航空航天大学出版社

内 容 简 介

Keil C51 是美国 Keil Software 公司出品的 51 系列兼容单片机 C 语言软件开发系统，μVision2 IDE 是 Keil C51 基于 Windows 的开发平台，是用户开发和调试单片机 C 语言源代码的最理想的工具之一。固件程序的编程是整个 USB 外设开发中非常重要的一环，它不是单纯的软件，而是软件和硬件的结合，开发者需要对单片机的端口、中断和 USB 协议处理芯片的硬件结构非常熟悉。本书的重点在于如何使用 Keil C51 的 Windows 集成开发环境 μVision2，如何进行 USB 设备固件代码的开发和仿真调试，以助读者达到熟练掌握使用 μVision2 开发和调试程序、进行 USB 固件开发和调试的目的。书中示例丰富，所有的例子都经过上机操作和认真审核。本书可作为从事单片机和 USB 设备开发的工程技术人员、工程师的参考书籍，也可供高等学校工科电子类专业师生参考。

图书在版编目(CIP)数据

单片机开发环境 μVision2 使用指南及 USB 固件编程与调试/尹勇等编著. —北京：北京航空航天大学出版社，2004.10

ISBN 7-81077-493-X

Ⅰ.单… Ⅱ.尹… Ⅲ.单片微型计算机 Ⅳ.TP368.1

中国版本图书馆 CIP 数据核字(2004)第 099661 号

单片机开发环境 μVision2 使用指南及 USB 固件编程与调试

尹 勇 王洪成 编著

责任编辑 王鑫光

责任校对 陈 坤

*

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(100083) 发行部电话：(010)82317024 传真：(010)82328026

http://www.buaapress.com.cn E-mail:bhpress@263.net

涿州市新华印刷有限公司印装 各地书店经销

*

开本：787×1092 1/16 印张：22.75 字数：582 千字

2004 年 10 月第 1 版 2004 年 10 月第 1 次印刷 印数：5 000 册

ISBN 7-81077-493-X 定价：32.00 元

前　言

单片机系统的开发和应用非常广泛，传统的单片机方面的书籍主要介绍内部结构、工作原理、指令系统和软硬件的开发，而C语言方面的书籍主要介绍语法和程序设计方面的知识。本书不是介绍Keil C51语言和单片机硬件方面的知识，而是指导读者如何使用μVision2开发和调试使用单片机C语言开发的USB固件程序。本书采用实例方式，循序渐进，简洁明了。读者只要按照实例步骤实践，就能在最短的时间内达到使用μVision2开发和调试程序的中、高级水平，并具备开发和调试USB固件的能力。

全书的章节安排如下：

第1章为Keil C51的简单回顾，让读者在最短的时间内具备单片机的C语言编程能力。

第2章为Keil C51开发工具简介，介绍了单片机C语言的开发工具的发展历程和各种开发工具的特点，并给出了一个完整的实例。

第3章深入μVision2的集成开发环境，详细介绍了C51的Windows集成开发环境μVision2的菜单命令、工具、窗口及其使用。

第4章用μVision2管理项目，介绍了项目程序的建立、设置和编译等，是本书的重点篇章。

第5章用μVision2调试项目，介绍了如何用C51的Windows集成开发环境μVision2调试项目程序以及各种调试技巧，也是本书的重点篇章。

第6章介绍了USB的设备规范，让读者能够阅读规范手册，理解USB固件开发机理。

第7章探讨了USB数据传输中的各种数据包和数据传输的4种方式，让读者在固件开发中选择适合自己的数据传输模型。

第8章介绍了USB接口芯片PDIUSBD12的资源和命令集，这对USB器件的了解是固件开发不可缺少的知识。

第9章介绍了USB固件程序实现的具体过程，并给出了固件程序的源代码，是本书的重点篇章。

第10章在μVision2的开发环境中如何对USB固件进行软件仿真和结合开发板进行硬仿真，是本书的重点篇章。

本书由华中科技大学尹勇博士、武汉理工大学王洪成老师编写，参与本书的编写工作的还有：华中科技大学欧光军博士、尚会超博士、李洪杰博士、范良志博士、张超勇博士和朱传军博士等，在此特表示感谢。同时得到北京航空航天大学出版社的大力支持和鼓励，在此深表敬意。

由于作者水平有限，书中图表较多，难免出现错误和不妥之处，恳请广大读者批评指正。

作　者

2004年2月23日

华中科技大学

目　录

第 1 章　Keil C51 的基础知识

1.1　C51 程序的基本结构 …… 1
1.2　C51 的标识符与关键字 …… 3
1.3　C51 的数据类型 …… 5
1.4　C51 的常量和变量 …… 9
1.4.1　C51 的常量 …… 9
1.4.2　C51 的变量 …… 11
1.5　C51 的函数 …… 14
1.5.1　函数的说明 …… 14
1.5.2　函数的定义 …… 15
1.5.3　函数的调用 …… 16
1.6　C51 的数组与指针 …… 16
1.6.1　C51 的数组 …… 16
1.6.2　C51 的指针 …… 18
1.7　C51 的结构与联合 …… 21
1.7.1　C51 的结构 …… 21
1.7.2　C51 的联合 …… 24
1.8　C51 类型定义 …… 26
1.9　C51 的编译预处理 …… 27
1.9.1　宏定义 …… 27
1.9.2　文件包含 …… 30
1.9.3　条件编译 …… 32

第 2 章　Keil C51 开发工具简介

2.1　μVision2 集成开发环境介绍 …… 35
2.2　DOS 下的 C51 开发工具 …… 37
2.2.1　C51 开发工具介绍 …… 37
2.2.2　Keil C51 的 C 编译器 …… 38
2.2.3　Keil C51 的 A51 宏汇编器 …… 39
2.2.4　Keil C51 的 BL51 代码连接器/定位器 …… 39
2.2.5　Keil C51 的 OC51 目标文件转换器 …… 41
2.2.6　Keil C51 的 OH51 目标十六进制转换器 …… 41
2.2.7　Keil C51 的 LIB51 库文件管理器 …… 42
2.3　Windows 下的 C51 开发工具 …… 42
2.3.1　μVision1 版 …… 43
2.3.2　μVision2 版 …… 45
2.4　μVision2 的安装 …… 50

2.4.1 系统需求 …… 50
2.4.2 安装注意事项 …… 50
2.4.3 μVision2 的安装过程 …… 50
2.5 μVision2 安装后的文件组织结构 …… 55
2.6 一个完整的应用实例 …… 55

第 3 章 μVision2 的集成开发环境

3.1 μVision2 集成开发环境 …… 63
3.2 μVision2 项目管理窗口 …… 64
3.2.1 目标、文件组和文件的管理 …… 64
3.2.2 项目窗口中的文件和文件组的属性 …… 67
3.3 μVision2 的菜单栏 …… 70
3.4 μVision2 工具栏的使用 …… 83
3.5 μVision2 快捷键的使用 …… 85
3.6 μVision2 的各种窗口 …… 88
3.6.1 设置窗口属性 …… 88
3.6.2 源代码编辑窗口 …… 92
3.6.3 反汇编窗口 …… 94
3.6.4 Watch &Call Stack 窗口 …… 95
3.6.5 Memory 窗口 …… 97
3.6.6 CPU 寄存器窗口 …… 99
3.6.7 串行窗口 …… 99
3.6.8 性能分析窗口 …… 100
3.6.9 代码覆盖窗口 …… 102
3.6.10 符号观察窗口 …… 103

第 4 章 用 μVision2 管理项目

4.1 启动 μVision2 并创建一个项目 …… 106
4.1.1 创建一个新的项目 …… 106
4.1.2 新建一个源文件 …… 108
4.2 增加和配置启动代码 …… 109
4.3 μVision2 的 CPU 和程序启动代码详解 …… 110
4.4 为目标设置工具选项 …… 113
4.4.1 配置对话框 …… 113
4.4.2 例子项目的设置 …… 115
4.5 编译项目并生成 hex 文件 …… 136
4.6 代码分块 …… 138
4.7 使用资源浏览器 …… 153
4.8 Keil C51 与汇编语言的接口 …… 156
4.8.1 模块内接口 …… 156
4.8.2 模块间接口 …… 158
4.9 列表文件的使用 …… 162

4.9.1 C语言列表文件 …… 162
4.9.2 汇编语言列表文件 …… 166
4.10 μVision2的使用技巧 …… 169
4.10.1 导入μVision1的项目到μVision2 …… 169
4.10.2 为列表文件和目标文件指定单独的文件夹 …… 169
4.10.3 复制工具设置到一个新的目标中 …… 171
4.10.4 使用μVision2器件库中没有的微控制器 …… 171

第5章 用μVision2调试项目

5.1 调试设置 …… 173
5.1.1 设置调试参数 …… 176
5.1.2 指定调试器初始化文件 …… 177
5.1.3 启动代码调试模式 …… 179
5.2 项目调试 …… 179
5.2.1 使用反汇编窗口 …… 179
5.2.2 使用断点 …… 182
5.2.3 使用变量和函数观察(Watch)窗口 …… 187
5.2.4 使用CPU寄存器观察窗口 …… 190
5.2.5 使用内存观察窗口 …… 191
5.2.6 使用串口观察窗口 …… 193
5.2.7 使用执行效果观察窗口 …… 194
5.2.8 使用内存标记窗口 …… 196
5.2.9 使用符号观察窗口 …… 197
5.2.10 程序的运行 …… 199

第6章 USB设备规范

6.1 USB概述 …… 200
6.1.1 USB的发展历程 …… 201
6.1.2 USB1.1的特点 …… 203
6.1.3 USB存在的问题 …… 204
6.1.4 USB的应用 …… 204
6.2 USB的通信模型 …… 205
6.3 USB设备状态 …… 208
6.3.1 外置的设备状态 …… 208
6.3.2 USB设备的枚举过程 …… 210
6.3.3 USB设备的数据传输过程 …… 211
6.4 通用USB设备操作 …… 211
6.4.1 动态插接与拔出 …… 212
6.4.2 地址分配 …… 212
6.4.3 配　置 …… 212
6.4.4 数据传送 …… 213
6.4.5 电源管理 …… 213

6.4.6 请求处理 …… 214
6.4.7 请求错误 …… 215
6.5 USB 设备的标准请求 …… 215
6.5.1 BmRequestType 域 …… 216
6.5.2 BRequest 域 …… 217
6.5.3 WValue 域 …… 217
6.5.4 WIndex 域 …… 217
6.5.5 WLength 域 …… 218
6.5.6 各种标准请求 …… 218
6.6 USB 设备中的固件描述表 …… 223
6.6.1 设备描述表 …… 223
6.6.2 配置描述表 …… 224
6.6.3 接口描述表 …… 225
6.6.4 端点描述表 …… 226
6.6.5 字串描述表 …… 227
6.6.6 固件描述表举例 …… 228

第 7 章 USB 的数据包及数据传输方式

7.1 USB 的数据传输 …… 231
7.2 各种包的格式 …… 232
7.2.1 标记包 …… 233
7.2.2 帧开始包 …… 233
7.2.3 数据包 …… 233
7.2.4 握手包 …… 234
7.2.5 握手回答包(Handshake Response) …… 235
7.3 标记包的字段格式 …… 236
7.3.1 包标识符字段 …… 236
7.3.2 地址字段 …… 237
7.3.3 帧号字段 …… 238
7.3.4 数据字段 …… 238
7.3.5 循环冗余校验 …… 238
7.4 USB 的数据传输方式 …… 240
7.4.1 批处理传送 …… 240
7.4.2 控制传送 …… 242
7.4.3 中断事务 …… 245
7.4.4 同步传送 …… 247

第 8 章 USB 接口芯片 PDIUSBD12

8.1 PDIUSBD12 的芯片特点 …… 248
8.2 PDIUSBD12 的引脚说明 …… 249
8.3 PDIUSBD12 的内部结构 …… 251
8.4 PDIUSBD12 与 80C51 的典型连接 …… 253

8.5 PDIUSBD12 的端点描述 …… 253
8.6 PDIUSBD12 的命令 …… 255
8.6.1 命令总汇 …… 255
8.6.2 初始化命令 …… 256
8.6.3 数据流命令 …… 259
8.6.4 普通命令 …… 263

第 9 章 PDIUSBD12 固件的编程实现

9.1 固件编程介绍 …… 264
9.2 固件的文件结构 …… 265
9.3 固件的编程实现 …… 267
9.3.1 主循环 MAINLOOP.c …… 267
9.3.2 命令接口 D12Cl.c …… 280
9.3.3 中断服务程序 ISR.c …… 286
9.3.4 协议层 CHAP_9.c 和 PROTODMA.c …… 297
9.4 固件编程注意事项 …… 314

第 10 章 PDIUSBD12 固件代码在 μVision2 中的调试

10.1 打开项目 …… 316
10.2 固件代码的软仿真调试 …… 317
10.2.1 项目的设置 …… 317
10.2.2 项目的编译 …… 318
10.3 固件代码的软件仿真调试 …… 319
10.4 固件代码的硬仿真调试 …… 325
10.4.1 关于硬件的调试 …… 325
10.4.2 μVision2 软件的配置 …… 326

附录 A μVision2 的高级编程技巧

附录 B μVision2 的错误信息

B.1 致命错误 …… 340
B.2 语法和语义错误 …… 342
B.3 警　告 …… 349

参考文献

第1章
Keil C51 的基础知识

1.1 C51 程序的基本结构

C51 源程序结构与一般 C 语言没有什么差别。C51 源程序文件的扩展名为“.c”，如 Add.c、Max.c 等。一个 C51 源程序大体上是一个函数定义的集合，在这个集合中有且仅有一个名为 main( )的函数，也称为该程序的主函数。主函数是程序的入口，它是一个特殊的函数，程序的执行都是从 main( )函数开始的。主函数中的所有语句执行完毕，则程序执行结束。

C51 源程序的编程要点如下：

① C 语言是由函数构成的。一个 C51 源程序至少包含一个 main( )函数，也可以包含一个 main( )函数和若干其他函数。因此，函数是 C51 源程序的基本单位。被调用的函数可以是编译器提供的库函数，也可以是用户根据需要自己编制设计的函数。

② 一个 C51 源程序总是从 main( )函数开始执行的，而不论 main( )函数在整个程序中的位置如何。

③ C51 源程序书写格式自由，一行内可以写几个语句，一个语句可以分写在多行上，C51 源程序无行号。

④ 每个语句和数据定义的最后必须有一个分号，分号是 C51 源程序语句的必要组成部分，它不可缺少，即使是程序中最后一个语句也应包含分号。

⑤ C 语言本身没有输入/输出语句。输入和输出的操作是由库函数 Scanf()和 Printf()等函数来完成的。C51 源程序对输入/输出实行“函数化”。

⑥ 可以用/*…*/对 C51 源程序中的任何部分作注释。一个好的、有使用价值的程序都应当加上必要的注释，以增加程序的可读性。

下面是一个简单的 C51 源程序例子(test.c)：

```
# include <reg52.h>                    /*C编译器内部自带的H文件,使用<>*/
void Test1(void);                      /*C函数声明*/

void main(void)
{
    Test1( );                          /*调用函数 Test1( )*/
    unsigned char i;                   /*C变量声明*/
    while(1){
        i++;
    }
}

void Test1(void)
{
unsigned char Work;
 Work++;
}
```

在C51源程序中,常用项目来管理各个文件。项目一般分为两大块:C文件块和头部文件块。常把不同功能写在不同的C文件中,依靠项目的管理,最后把所有文件连接起来,这样就得到可以烧录的hex文件或bin文件。这些C文件中,有且只有一个包括main( )函数。可以用头部文件把各个不同的C文件互相连接起来。一个C文件基本上要对应有一个H头部文件,这个H文件就包含本C文件中可以提供给外部使用的变量和函数,没有在H文件中列出的文件,可以算是该C文件的内部函数和变量,外部C文件不能使用。

从上面例子可以看出,C51源程序一般具有如下结构:

```
# include< >                           /*预处理命令*/
int function_1( );                     /*函数1定义*/
char function_2( );                    /*函数2定义*/
……
float function_n( );                   /*函数n定义*/
main( ) {                              /*主函数*/
……
}

function_1( ){                         /*功能函数1*/
……                                     /*功能函数1函数体*/
}

char function_2( ){                    /*功能函数2*/
……                                     /*功能函数2函数体*/
}
……
```

```
float function_n( ){                    /*功能函数 n*/
……                                      /*功能函数 n 函数体*/
}
```

一个 C 语言程序至少应包含一个 main( )函数，也可以包含一个 main( )函数和其他的许多功能函数。函数之间可以相互调用，但 main( )函数只能调用其他的功能函数，而不能被其他函数所调用。功能函数可以是 C 语言编译器提供的库函数，也可以由用户按实际需要自行编写。再次强调，不管 main( )函数处于程序中的什么位置，程序总是从 main( )函数开始执行。

1.2　C51 的标识符与关键字

标识符是用来标识源程序中某个对象的名字的，这些对象可以是语句、数据类型、函数、变量、数组等等。C 语言是大小写敏感的一种高级语言，如果要定义一个定时器 1，可以写做“Timer1”，如果程序中有“TIMER1”，那么这两个是完全不同定义的标识符。标识符由字符串、数字和下划线等组成，要注意的是，第一个字符必须是字母或下划线，如“1Timer”是错误的，编译时便会有错误提示。有些编译系统专用的标识符以下划线开头，所以一般不要以下划线开头命名标识符。标识符在命名时应当简单，含义清晰，这样有助于阅读理解程序。在 C51 编译器中，只支持标识符的前 32 位为有效标识，这在一般情况下也足够用了，基本上不会出现超过 32 位的标识符。

关键字则是编程语言保留的特殊标识符，它们具有固定名称和含义，在程序编写中不允许标识符与关键字相同。在 Keil μVision2 中的关键字除了有 ANSI C 标准的 32 个关键字外，还根据 51 单片机的特点扩展了相关的关键字，如表 1.1 和表 1.2 所列。在 Keil μVision2 的文本编辑器中编写 C51 源程序，系统可以把保留字以不同颜色显示，缺省颜色为天蓝色。

表 1.1　ANSI C 标准的关键字

关键字	用　途	说　明
auto	存储种类说明	用以说明局部变量
break	程序语句	退出最内层循环
case	程序语句	switch 语句中的选择项
char	数据类型说明	单字节整型数或字符型数据
const	存储类型说明	在程序执行过程中不可更改的常量值
continue	程序语句	转向下一次循环
default	程序语句	switch 语句中的失败选择项
do	程序语句	构成 do...while 循环结构
double	数据类型说明	双精度浮点数
else	程序语句	构成 if...else 选择结构
enum	数据类型说明	枚举
extern	存储种类说明	在其他程序模块中说明了的全局变量

续表 1.1

关键字	用　途	说　明
flost	数据类型说明	单精度浮点数
for	程序语句	构成 for 循环结构
goto	程序语句	构成 goto 转移结构
if	程序语句	构成 if... else 选择结构
int	数据类型说明	基本整型数
long	数据类型说明	长整型数
register	存储种类说明	使用 CPU 内部寄存的变量
return	程序语句	函数返回
short	数据类型说明	短整型数
signed	数据类型说明	有符号数，二进制数据的最高位为符号位
sizeof	运算符	计算表达式或数据类型的字节数
static	存储种类说明	静态变量
struct	数据类型说明	结构类型数据
swicth	程序语句	构成 switch 选择结构
typedef	数据类型说明	重新进行数据类型定义
union	数据类型说明	联合类型数据
unsigned	数据类型说明	无符号数数据
void	数据类型说明	无类型数据
volatile	数据类型说明	该变量在程序执行中可被隐含地改变
while	程序语句	构成 while 和 do... while 循环结构

表 1.2　C51 编译器的扩展关键字

关键字	用　途	说　明
bit	位标量声明	声明一个位标量或位类型的函数
sbit	位标量声明	声明一个可位寻址变量
sfr	特殊功能寄存器声明	声明一个特殊功能寄存器
sfr16	特殊功能寄存器声明	声明一个 16 位的特殊功能寄存器
data	存储器类型说明	直接寻址的内部数据存储器
BDATA	存储器类型说明	可位寻址的内部数据存储器
IDATA	存储器类型说明	间接寻址的内部数据存储器
PDATA	存储器类型说明	分页寻址的外部数据存储器
XDATA	存储器类型说明	外部数据存储器
code	存储器类型说明	程序存储器
interrupt	中断函数说明	定义一个中断函数
reentrant	再入函数说明	定义一个再入函数
using	寄存器组定义	定义芯片的工作寄存器

1.3　C51 的数据类型

数据在计算机内存中的存放情况由数据结构决定。C 语言的数据结构是以数据类型决定的，数据类型可分为基本数据类型和复杂数据类型，复杂数据类型是由基本数据类型构造而成的。在标准 C 语言中基本的数据类型为 char、int、short、long、float 和 double，而在 C51 编译器中 int 和 short 相同，float 和 double 相同。表 1.3 中列出了 Keil μVision2 C51 编译器所支持的数据类型，下面具体说明其含义。

表 1.3　Keil μVision2 C51 编译器所支持的数据类型

数据类型	长　度	值　域
unsigned char	单字节	0～255
signed char	单字节	−128～+127
unsigned int	双字节	0～65 535
signed int	双字节	−32 768～+32 767
unsigned long	四字节	0～4 294 967 295
signed long	四字节	−2 147 483 648～+2 147 483 647
float	四字节	±1.175 494e−38～±3.402 823e+38
*	一～三字节	对象的地址
bit	位	0 或 1
sfr	单字节	0～255
sfr16	双字节	0～655 35
sbit	位	0 或 1

1. char 字符类型

char 类型的长度是一个字节，通常用于定义处理字符数据的变量或常量，它分为无符号字符类型 unsigned char 和有符号字符类型 signed char 两种，默认值为 signed char 类型。unsigned char 类型用字节中所有的位来表示数值，可以表达的数值范围是 0～255。signed char 类型用字节中最高位字节表示数据的符号，“0”表示正数，“1”表示负数，负数用补码表示，能表示的数值范围是−128～+127。unsigned char 常用于处理 ASCII 字符或用于处理小于或等于 255 的整型数。

注意：正数的补码与原码相同，负二进制数的补码等于它的绝对值按位取反后加 1。

2. int 整型

整型长度为两个字节，用于存放一个双字节数据。它分为有符号整型数 signed int 和无

符号整型数 unsigned int，默认值为 signed int 类型。signed int 表示的数值范围是－32 768～＋32 767，字节中最高位表示数据的符号，“0”表示正数，“1”表示负数。unsigned int 表示的数值范围是 0～65 535。

3. long 长整型

长整型长度为四个字节，用于存放一个四字节数据。它分为有符号长整型 signed long 和无符号长整型 unsigned long，默认值为 signed long 类型。signed long 表示的数值范围是－2 147 483 648～＋2 147 483 647，字节中最高位表示数据的符号，“0”表示正数，“1”表示负数。unsigned long 表示的数值范围是 0～4 294 967 295。

4. float 浮点型

浮点型在十进制中具有 7 位有效数字，是符合 IEEE—754 标准的单精度浮点型数据，占用四个字节。浮点数的结构较复杂，在内存中的存放格式如下：

字节地址	＋0	＋1	＋2	＋3
浮点数内容	S EEEEEEE	E MMMMMMM	MMMMMMMM	MMMMMMMM

其中，S 为符号位，“0”表示为正，“1”表示为负。E 为阶码，占用 8 位二进制数，存放在两个字节中，阶码 E 的值是以 2 为底的指数再加上偏移量 127，这样处理的目的是为了避免出现负的阶码值，而指数是可正可负的。阶码 E 的正常取值范围是 1～254，从而实际指数的取值范围为－126～127。M 为尾数的小数部分，用 23 为二进制数表示，存放在 3 个字节中，尾数的整数部分永远为 1，因此不需要保存，但它是隐含存在的。小数点位于隐含的整数值“1”的后面。

一个浮点数的数值是：

$$(-1)^S \times 2^{e-127} \times (1.M)$$

比如浮点数－12.5＝0xC1480000 在内存中的存放格式如下：

字节地址	＋0	＋1	＋2	＋3
浮点数内容	11000001	01001000	00000000	00000000

值得注意的是，51 单片机不包括捕获浮点运算错误的中断向量，比如溢出错误等，因此，用户必须自己根据可能出现的错误条件，用软件来进行适当的处理。

5. bit 位标量

位标量是 C51 编译器的一种扩充数据类型，利用它可定义一个位标量，但不能定义位指针，也不能定义位数组。它的值是一个二进制位，是 0 或 1，类似一些高级语言中的 Boolean 类型中的 True 和 False。

6. sfr 特殊功能寄存器

sfr 也是一种扩充数据类型，占用一个内存单元，值域为 0～255。利用它可以访问 51 单

片机内部的所有特殊功能寄存器。比如用“sfr P1 = 0x90”这一句，可定义 P1 为 P1 端口在片内的寄存器，在后面的语句中用“P1 = 255”或“P1=0FFH”(对 P1 端口的所有引脚置高电平)之类的语句来操作特殊功能寄存器。AT89C51 的特殊功能寄存器如表 1.4 所列。

表 1.4　AT89C51 特殊功能寄存器

符　号	地　址	注　释
ACC *	E0H	累加器
B *	F0H	乘法寄存器
PSW *	D0H	程序状态字
SP	81H	堆栈指针
DPL	82H	数据存储器指针低 8 位
DPH	83H	数据存储器指针高 8 位
IE *	A8H	中断允许控制器
IP *	D8H	中断优先控制器
P0 *	80H	端口 0
P1 *	90H	端口 1
P2 *	A0H	端口 2
P3 *	B0H	端口 3
PCON	87H	电源控制及波特率选择
SCON *	98H	串行口控制器
SBUF	99H	串行数据缓冲器
TCON *	88H	定时器控制
TMOD	89H	定时器方式选择
TL0	8AH	定时器 0 低 8 位
TL1	8BH	定时器 1 低 8 位
TH0	8CH	定时器 0 低 8 位
TH1	8DH	定时器 1 高 8 位

* 可以进行位寻址的特殊功能寄存器。

7. sfr16 16 位特殊功能寄存器

sfr16 占用两个内存单元，值域为 0～65 535。sfr16 和 sfr 一样用于操作特殊功能寄存器，不同的是它用于操作占两个字节的寄存器，例如定时器 T0 和 T1。当然，sfr16 也可以像 sfr 一样用一个字的方式访问，比如定时器 T2，可以分别以 TL2 和 TH2 进行访问。

8. sbit 可寻址位

位是 C51 中的一种扩充数据类型，利用它可以访问芯片内部的 RAM 中的可寻址位或特殊功能寄存器中的可寻址位。比如先定义：

```
sfr P1 = 0x90;              //因 P1 端口的寄存器是可位寻址的,所以可以这样定义
sbit P1_1 = P1^1;           //P1_1 为 P1 中的 P1.1 引脚
                            //同样可以用 P1.1 的地址去写,如"sbit P1_1 = 0x91";
```

这样在以后的程序语句中就可以用 P1_1 来对 P1.1 引脚进行读/写操作了。通常这些可以直接使用 Keil 系统提供的预处理文件,里面已定义好各特殊功能寄存器的简单名字,直接引用可以省去一点时间,当然也可以写自己定义的文件,用自己认为好记的名字。

下面写个小程序,演示一下 unsigned char 和 unsigned int 用于延时的不同效果,来说明它们在存储器中的长度是不同的。虽然它并没有实际的意义,但这里只学习它们的用法。试验用的硬件电路如图 1.1 所示。实验中用 D1 的点亮表明正在用 unsigned int 数值延时,用 D2 点亮表明正在用 unsigned char 数值延时。

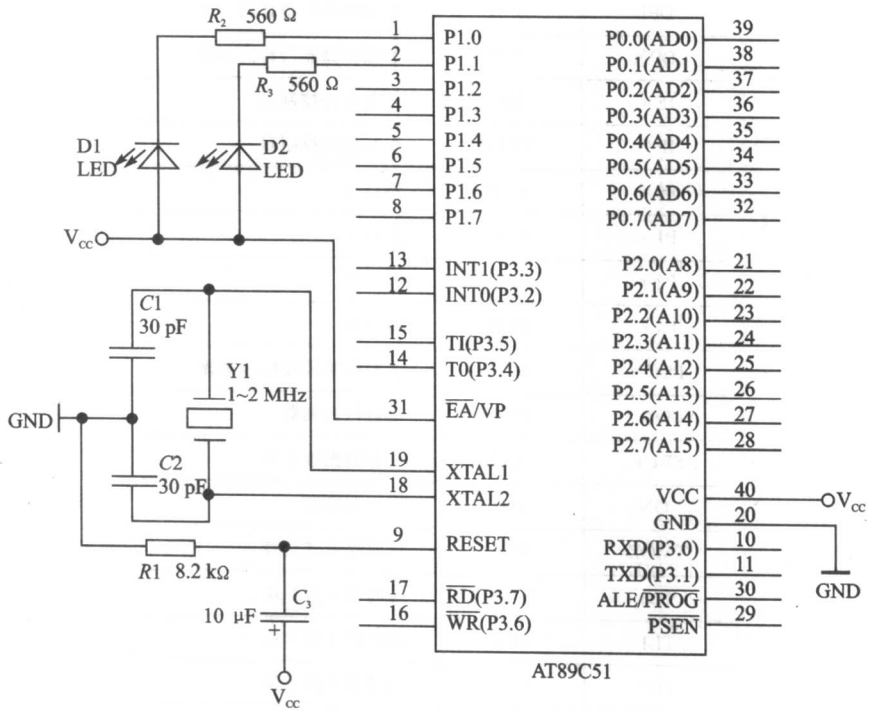

图 1.1　试验用的硬件电路

实验程序如下:

```
#include <AT89X51.h>                    //预处理命令
void main(void)                         //主函数名
{
unsigned int a;                         //定义变量 a 为 unsigned int 类型
unsigned char b;                        //定义变量 b 为 unsigned char 类型

do
{                                       //do while 组成循环
for (a=0; a<65535; a++)
```