“十二五”高职高专精品规划教材
高等职业教育课程改革项目研究成果

51单片机基础实验与课程实训教程(C语言版)

夏西泉　王锡惠　主　编

北京理工大学出版社
BEIJING INSTITUTE OF TECHNOLOGY PRESS

“十二五”高职高专精品规划教材
高等职业教育课程改革项目研究成果

51单片机基础实验与课程实训教程(C语言版)

夏西泉　王锡惠　主编
任德齐　主审

北京理工大学出版社
BEIJING INSTITUTE OF TECHNOLOGY PRESS

内 容 简 介

本书是一本注重单片机理论应用实践、能充分调动读者的学习积极性和创新性、具有较好操作性的实验与实训教材。

本书以51单片机为载体，主要介绍单片机C语言基础知识、单片机应用开发软件Keil uVision3和Proteus的使用与操作步骤，重点讲述了基于单片机原理的课程基础实验和基于能力提高与创新设计的课程项目实训。全书共设9个课内基础实验，16个项目实训实例。通过对这些实验与项目实例的学习与操作，单片机技能开发水平会得到较大的提高。

本书语言通俗，实践项目内容丰富，实践项目程序分析详尽，有超强的实用性和较高的参考价值，既可作为高职院校自动化、计算机、电子、电气、控制及相关专业师生的教材，也可作为单片机开发人员和单片机系统设计人员的参考用书。

图书在版编目（CIP）数据

51单片机基础实验与课程实训教程：C语言版　/夏西泉，王锡惠主编. —北京：北京理工大学出版社，2012.8

ISBN 978-7-5640-6561-4

Ⅰ. ①5…　Ⅱ. ①夏…②王…　Ⅲ. ①单片微型计算机-高等职业教育-教材　Ⅳ. ①TP368.1

中国版本图书馆CIP数据核字（2012）第186495号

出版发行／北京理工大学出版社
社　　址／北京市海淀区中关村南大街5号
邮　　编／100081
电　　话／(010)68914775(办公室)　68944990(批销中心)　68911084(读者服务部)
网　　址／http：// www. bitpress. com. cn
经　　销／全国各地新华书店
印　　刷／三河市天利华印刷装订有限公司
开　　本／787毫米×1092毫米　1/16
印　　张／10.5
字　　数／235千字
版　　次／2012年8月第1版　2012年8月第1次印刷　责任编辑／王艳丽
印　　数／1～3000册　责任校对／周瑞红
定　　价／24.00元　责任印制／王美丽

前言

Foreword

Foreword

单片机经过多年的发展，性能不断提高，价格不断降低，技术日趋成熟，各个领域应用相当广泛，行业企业对掌握单片机技术的人才需求也与日俱增。如何在短期内提高学生单片机设计与开发实践技能，是教育工作者进行单片机实践教学改革追求的目标。因此，作者结合多年的单片机教学与工程经验，采用“实验平台化、实训项目化”的指导思想，编写一本既方便教师“教”，又方便学生“学与做”的实践教材。“实验平台化”，是指学生做的所有实验都在真实实验平台上进行，以增加实验的直观性、学生学习的主动性与积极性，达到更好地学习单片机原理的目的；“实训项目化”，是指在单片机课程实训教学时，选取不同的实际应用项目，以实现教学的开放性、学生学习的自学性与创新性。

本书以51单片机为载体，主要介绍单片机C语言基础知识、单片机应用开发软件Keil uVision3和Proteus的使用与操作步骤，重点讲述了基于单片机原理的课程基础实验和基于能力提高与创新设计的课程项目实训。实验内容与理论教学内容紧扣，实训内容用经典实践项目切入。不但介绍单片机产品硬件设计、软件设计、调试制作，同时还介绍编程思路与技巧，使读者对经典的实践项目进行实战演练，从而掌握单片机原理，实现单片机应用开发。主要特点有：

（1）实验内容与实训项目由浅入深地精心设置，以读者的兴趣为基础，以实际应用项目为依托。

（2）力求做到有一个真实、完整的单片机控制系统，从最简单的一个指示灯控制系统到较复杂的工程应用系统设计，均配以标准化的电路原理图，供读者进行单片机实战及应用开发。

（3）通过大量具有趣味性的实践项目的练习，注重软件与硬件的紧密结合，强调以单片机为核心的电子产品的综合开发能力，旨在使读者尽快掌握单片机系统开发的全过程。

（4）详细地介绍了单片机开发环境，在此过程中可使读者熟悉单片机的软硬件开发环境，综合演练单片机的编程能力，亲身体验单片机的开发成果。

（5）实践项目源于不同实际应用，有些可直接用于实际应用系统开发，这对于从事单片机系统开发的工程技术人员十分有用。

（6）对于不具备实验设备的读者，通过本书学习，能够更好地理解单片机知识。

本书第1章、第3章由王锡惠编写，第2章、第4章、第5章由重庆电子工程职业学

院夏西泉编写并负责全书统稿工作。在编写过程中，参考了大量工程案例和网络资源，重庆工商职业学院任德齐教授审阅了全书，并提出了许多宝贵的意见和建议。另外，在程序调试与仿真方面也得到了程洪斌工程师大力支持，在此一并致以诚挚的谢意。

由于作者水平有限和编写时间仓促，书中出现一些错误和不妥之处在所难免，恳请读者批评与指正。

编　者

目录

Contents

第 1 章

单片机 C 程序基础

C 语言是一种源于开发 UNIX 操作系统的语言，它是一种结构化的程序设计语言，可以生成非常紧凑的代码。

与汇编语言相比，C 语言的优势如下：

用 C 语言编写的程序可读性强；在不了解单片机指令系统而仅熟悉 8051 单片机存储结构时就可以开发单片机程序；寄存器分配和不同存储器寻址及数据类型等细节可由编译器管理；程序可分为多个不同函数，这使程序设计结构化；函数库丰富、数据处理能力很强；程序编写及调试时间大大缩短、开发效率远高于汇编语言；C 语言具有模块化编程技术，已编写好的通用程序模块很容易植入新程序，进一步提高了程序开发效率。

1.1 Keil C 程序基本结构

一般地，C 语言的程序可看作是由一些函数（function，或视为子程序）所构成，其中的主程序是以“main（）”开始的函数，而每个函数可视为独立的个体，就像是模块（module）一样，所以 C 语言是一种非常模块化的程序语言。C 语言程序的基本结构如图 1-1 所示，其中各项说明如下。

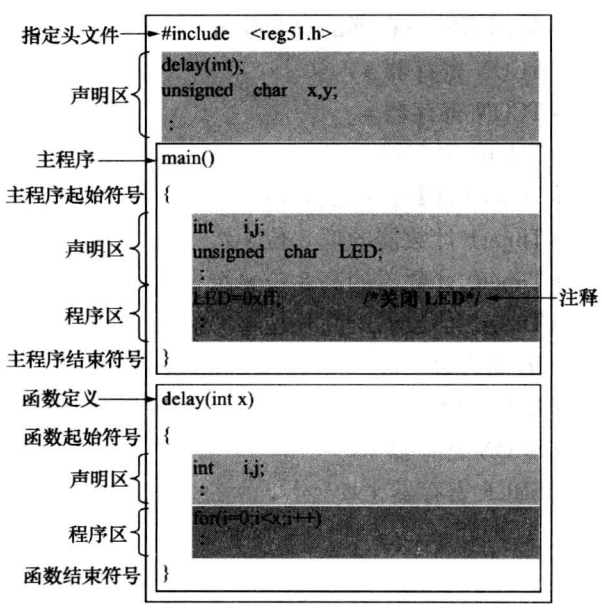

图 1-1 C 语言程序的基本结构

1.1.1　指定头文件

“头文件”或包含文件（*.h），是一种预先定义好的基本数据。在 8x51 程序里，必要的头文件是定义 8x51 内部寄存器地址的数据，如下所示：

```
/*-------------------------------------------------------------------
REG51.H
Header file for generic 80C51 and 80C31 microcontroller.
Copyright (c) 1988 - 2002 Keil Elektronik GmbH and Keil Software, Inc.
All rights reserved.
-------------------------------------------------------------------*/
#ifndef __REG51_H__
#define __REG51_H__
/*    BYTE Register    */
sfr P0    = 0x80;       /*P0*/
sfr P1    = 0x90;       /*P1*/
sfr P2    = 0xA0;       /*P2*/
sfr P3    = 0xB0;       /*P3*/
sfr PSW   = 0xD0;       /*程序状态字寄存器*/
sfr ACC   = 0xE0;       /*A 累加器*/
sfr B     = 0xF0;       /*B 寄存器*/
sfr SP    = 0x81;       /*堆栈指针寄存器*/
sfr DPL   = 0x82;       /*数据指针寄存器的低 8 位*/
sfr DPH   = 0x83;       /*数据指针寄存器的高 8 位*/
sfr PCON= 0x87;         /*PCON 寄存器*/
sfr TCON= 0x88;         /*TCON 寄存器*/
sfr TMOD= 0x89;         /*TMOD 寄存器*/
sfr TL0   = 0x8A;       /*Timer0 计数器的低 8 位*/
sfr TL1   = 0x8B;       /*Timer1 计数器的低 8 位*/
sfr TH0   = 0x8C;       /*Timer0 计数器的高 8 位*/
sfr TH1   = 0x8D;       /*Timer1 计数器的高 8 位*/
sfr IE    = 0xA8;       /*IE 寄存器*/
sfr IP    = 0xB8;       /*IP 寄存器*/
sfr SCON= 0x98;         /*SCON 寄存器*/
sfr SBUF= 0x99;         /*SBUF 寄存器*/

/*    BIT Register    */
/*    PSW    */
sbit CY    = 0xD7;        /*进位位*/
sbit AC    = 0xD6;        /*辅助进位位*/
```

```
sbit F0   = 0xD5;      /* 用户标志位 */
sbit RS1  = 0xD4;      /* 寄存器组选择位 1 */
sbit RS0  = 0xD3;      /* 寄存器组选择位 0 */
sbit OV   = 0xD2;      /* 溢出位 */
sbit P    = 0xD0;      /* 校验位 */

/*  TCON  */
sbit TF1  = 0x8F;      /* Timer1 的溢出位 */
sbit TR1  = 0x8E;      /* Timer1 的运行位 */
sbit TF0  = 0x8D;      /* Timer0 的溢出位 */
sbit TR0  = 0x8C;      /* Timer0 的运行位 */
sbit IE1  = 0x8B;      /* INT1 的中断标志 */
sbit IT1  = 0x8A;      /* INT1 的触发信号种类位 */
sbit IE0  = 0x89;      /* INT0 的中断标志 */
sbit IT0  = 0x88;      /* INT0 的触发信号种类位 */

/*  IE  */
sbit EA   = 0xAF;      /* 中断的总开关 */
sbit ES   = 0xAC;      /* 串行端口中断的启用位 */
sbit ET1  = 0xAB;      /* Timer1 中断的启用位 */
sbit EX1  = 0xAA;      /* INT1 中断的启用位 */
sbit ET0  = 0xA9;      /* Timer0 中断的启用位 */
sbit EX0  = 0xA8;      /* INT0 中断的启用位 */

/*  IP  */
sbit PS   = 0xBC;      /* 串行端口中断优先等级设置位 */
sbit PT1  = 0xBB;      /* Timer0 中断优先等级设置位 */
sbit PX1  = 0xBA;      /* INT1 中断优先等级设置位 */
sbit PT0  = 0xB9;      /* Timer1 中断优先等级设置位 */
sbit PX0  = 0xB8;      /* INT0 中断优先等级设置位 */

/*  P3  */
sbit RD   = 0xB7;      /* RD 引脚 */
sbit WR   = 0xB6;      /* WR 引脚 */
sbit T1   = 0xB5;      /* T1 引脚 */
sbit T0   = 0xB4;      /* T0 引脚 */
sbit INT1 = 0xB3;      /* INT1 引脚 */
sbit INT0 = 0xB2;      /* INT0 引脚 */
sbit TXD  = 0xB1;      /* TxD 引脚 */
sbit RXD  = 0xB0;      /* RxD 引脚 */
```

```
/*  SCON  */
sbit SM0  = 0x9F;      /* 串行端口方式设置位 0 */
sbit SM1  = 0x9E;      /* 串行端口方式设置位 1 */
sbit SM2  = 0x9D;      /* 串行端口方式设置位 2 */
sbit REN  = 0x9C;      /* 接收使能控制位 */
sbit TB8  = 0x9B;      /* bit 8 位 */
sbit RB8  = 0x9A;      /* 接收的 bit 8 位 */
sbit TI   = 0x99;      /* 发送的中断标志位 */
sbit RI   = 0x98;      /* 接收的中断标志位 */
#endif
```

头文件的指定方式有两种。第一种是在#include 之后，以 < > 包含头文件文件名，其格式为：

```
#include  <头文件文件名>
```

若采用这种方式，编译程序将从 Keil uVision3 的头文件夹查找所指定的头文件。如果 Keil uVision3 安装在 C 盘的根目录上，则编译程序将从“C：\ Keil \ C51 \ INC”路径中查找。

第二种是在#include 之后，以“ ”包含头文件文件名，其格式为：

```
#include  “头文件文件名”
```

若采用这种方式，编译程序将从源程序所在文件夹里查找所指定的头文件。

1.1.2　声明区

在指定头文件之后，就可声明程序之中所要使用的常数、变量、函数等，其作用域将扩展到整个程序。

若程序之中使用了许多函数，则可在声明区先声明所有使用到的函数，这样，函数放置的先后顺序将不会有所影响。换言之，函数放置在引用该函数的程序之前或之后都可以。

若没有在此声明函数，则在使用函数之前必须先定义该函数。

1.1.3　主程序

主程序（主函数）是以 main（）为开头，整个内容放置在一对大括号（即 {}）里的程序段，其中分为声明区与程序区，主程序的结构如图 1 - 2 所示。

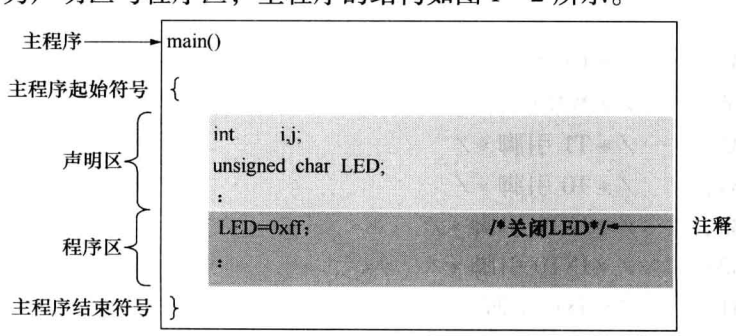

图 1 - 2　主程序的结构

在声明区里所声明的常数、变量等仅适用于主程序之中，而不影响其他函数。若在主程序之中使用了某变量，但在之前的声明区中没有声明，也可在主程序的声明区中声明。

程序区就是以语句所构成的程序内容。

1. 函数定义

函数是一种独立功能的程序，其结构与主程序类似。函数可将所要处理的数据传入该函数里，称为形式参数（arguments）；也可将函数处理完成后的结果返回调用它的程序，称为返回值。不管是形式参数还是返回值，在定义函数的第一行里应该交代清楚。

函数的定义格式如下：

返回值　数据类型　函数名称（数据类型　形式参数）

例如，要将一个无符号字符（unsigned char）实参传递给函数，函数执行完成时要返回一个整型（int），此函数的名称为 My_ func，则函数定义为：

```
int  My_ func (unsigned  char  x)
```

若不要传入参数，则可在小括号内指定为 void。同样地，若不要返回值，则可在函数名称左边指定为 void 或不指定。

另外，函数的起始符号、结束符号、声明区及程序区都与主程序一样。

在一个 C 语言的程序里可使用多个函数，并且函数中也可以调用函数。

2. 注释

所谓“注释”就是说明，属于编译器不处理的部分。

C 语言的注释以“/*”开始，以“*/”结束。放置注释的位置可接续于语句完成之后，也可独立于一行。其中的文字，可使用中文。不过，在 uVision 3 中对于中文的处理并不是很好，常会造成文字定位不准确等困扰。

另外，也可以输入“//”，其右边整行都是注释。

1.2 变量、常数与数据类型

在 C 语言里，常数（constant）与变量（variable）都是为某个数据指定存储器空间，其中常数是固定不变的，而变量是可变的。声明常数或变量的格式为：

数据类型　常数/变量名称［=默认值］;

其中的“［=默认值］”并非必要项目，而分号（;）是结束符号。

例如，若要声明一个整型类型的 x 变量，则语句为：

```
int  x;
```

若要声明一个整型类型的 x 变量，其默认值为 50，则语句为：

```
int  x=50;
```

若要同时声明 x、y、z 三个整型类型的变量，变量名之间以“,”分隔，则语句为：

```
int  x, y, z;
```

1.2.1 数据类型

在使用常数或变量时，首先必须要声明数据类型，其目的是让编译程序为该常数或变量保留存储器空间。要保留多大的存储空间？这与常数或变量的数据类型有关。

Keil C 所提供的数据类型可分为通用数据类型和 8x51 特有的数据类型。

1. 通用数据类型

通用数据类型可用于一般 C 语言之中，如 ANSI C 等，包括字符（char）、整型（int）、浮点数（float）与无值（void），其中字符与整型又分为有符号（signed）与无符号（unsigned）两类，如表 1 - 1 所示。

表 1 - 1　通用数据类型

数据类型	名称	长度（位）	值域
char	字符	8	-128 ~ +127
unsigned char	无符号字符	8	0 ~ 255
enum	枚举	8/16	-128 ~ +127/ -32 768 ~ +32 767
short	短整型	16	-32 768 ~ +32 767
unsigned short	无符号短整型	16	0 ~ 65 535
int	整型	16	-32 768 ~ +32 767
unsigned int	无符号整型	16	0 ~ 65 535
long	长整型	32	-2 147 483 648 ~ +2 147 483 647
unsigned long	无符号长整型	32	0 ~ 4 294 967 295
float	浮点数	32	-3.402 823E -38 ~ 3.402 823E +38
double	双倍精度浮点数	64	-1.7E -308 ~ +1.7E +308
void	空	0	无值

2. 8x51 特有的数据类型

专为 8x51 硬件装置所设置的数据类型有 bit、sbit、sfr 及 sfrl6 共四种，如表 1 - 2 所示。

表 1 - 2　8x51 特有的数据类型

名称	位数	范围
bit	1	0，1
sbit	1	0，1
sfr	8	0 ~ 255
sfr16	16	0 ~ 65 535

以下将特别介绍这四种 8x51 特有的数据类型。

（1）bit 数据类型

bit 数据类型是定义 1 位的变量，将会被指定到 0x20 ~ 0x2f 之间的地址。

（2）sbit 数据类型

sbit 数据类型是用于存取内部可位寻址的数据存储器，即 0x20 到 0x2f 之间的存储器，或存取可位寻址的特殊功能寄存器（SFR），即 0x80 到 0xff 之间的存储器。

若要使用 sbit 数据类型，则其声明方式有下列几种。

1）先声明一个 bdata 存储器形式的变量，再声明属于该变量的 sbit 变量，语句为：

```
char bdata scan;          /* 声明 scan 为 bdata 存储器类型的字符 */
sbit input_0 = scan^0;    /* 声明 input_0 为 scan 变量的 bit 0 */
```

当要指定（声明）某个变量的第 n 位，则可在该变量名称右边加上“^n”即可。例如 P0 的 bit 3 为 P0^3。

2）先声明一个 sfr 变量，再声明属于该变量的 sbit 变量，例如：

```
sfr P0  = 0x80;           /* 声明 P0 为 0x80 存储器位置，即 P0 */
sbit P0_0  = P0^0;        /* 声明 P0_0 为 P0 变量的 bit 0 */
```

这种用法最方便，因为 8x51 内部特殊功能寄存器的声明都在 reg51.h 里，而程序的开头都已将这个文件包含进来了。

3）直接指定存储器位置，例如要声明 P0 的 bit 0，语句为：

```
sbit P0_0 =0x80^0;        /* 声明 P0_0 为 0x80 地址的 bit 0 */
```

这种声明必须要熟记每个地址才行。

（3）sfr 数据类型

sfr 数据类型是用于 8x51 内部特殊功能寄存器（寄存器名称使用大写），即 0x80～0xff 地址，与内部存储器的地址相同。不过，特殊功能寄存器与内部存储器是两个独立的区域，必须以不同的存取方式来区分。特殊功能寄存器采用直接寻址方式存取，而内部存储器采用间接寻址方式存取。

在 Keil C 里，所谓直接寻址，就是直接指定其地址，以 P0 的声明为例，语句为：

```
sfr P0 =0x80;             /* 声明 P0 为 0x80 存储器位置，即 P0 */
```

所谓间接寻址，就是声明为 idata 存储器形式的变量，语句为：

```
char idata BCD;           /* 声明 BCD 变量为间接寻址的存储器位置 */
```

由于 reg51.h 里已声明了 8x51 内部特殊功能寄存器，不需要再声明，如果不手动配置存储器（交由编译程序处理），程序里就会较少出现 sfr 数据类型的声明。

（4）sfrl6 数据类型

sfrl6 数据类型是用于 8x51 内部 16 位的特殊功能寄存器（寄存器名称使用大写），如 Timer2 的捕捉寄存器（RCAP2L、RCAP2H）、Timer2 的计数器（TL2、TH2）、数据指针寄存器（DPL、DPH）等。以数据指针寄存器为例，语句为：

```
sfr16 DPTR =0x82;         /* 声明 DPTR 变量为数据指针寄存器 */
```

1.2.2 变量名称与保留字

在常数或变量的使用格式中，数据类型之后就是变量名称，而变量名称的指定除了容易判读外，还要遵守下列规则。

（1）可使用大/小写字母、数字或下划线（_）。

（2）第一个字符不可为数字。

（3）不可使用保留字。

所谓“保留字”，是指编译程序将该字符串保留为其他特殊用途，ANSI C 的保留字（小写）如表 1－3 所示，Keil C 也有其特有的保留字如表 1－4 所示。

表 1 - 3　ANSI C 和传统 C 的保留字

asm	auto	break	case	char	const
continue	default	do	double	else	entry
enum	extern	float	for	fortran	goto
int	long	register	return	short	signed
sizeof	static	struct	switch	typedef	union
unsigned	void	volatile	while		

表 1 - 4　Keil C 的保留字

_ at_	_ priority _	_ task _	alien	bdata	bit
code	compact	data	far	idata	interrupt
large	pdata	reentrant	sbit	sfr	sfrl6
small	using	xdata			

1.2.3　变量的作用范围

变量的适用范围或有效范围与该变量是在哪里声明的有关，大致可分为全局变量和局部变量两种。

1. 全局变量

若在程序开头的声明区或者是没有大括号限制的声明区所声明的变量，其适用范围为整个程序，称为全局变量，如图 1 - 3 所示，其中的 LED、SPEAKER 就是全局变量。

2. 局部变量

若在大括号内的声明区所声明的变量，其适用范围将受限于大括号，称为局部变量，图 1 - 3 中的 i、j 就是局部变量。

若在主程序与各函数之中都有声明相同名称的变量，则脱离主程序或函数时，该变量将自动无效，又称之为自动变量。

如图 1 - 4 所示，在主程序与 delay 子程序中各自声明了 i、j 变量，但主程序的 i、j 与 delay 子程序中的 i、j 为各自独立（无关）的 i、j。

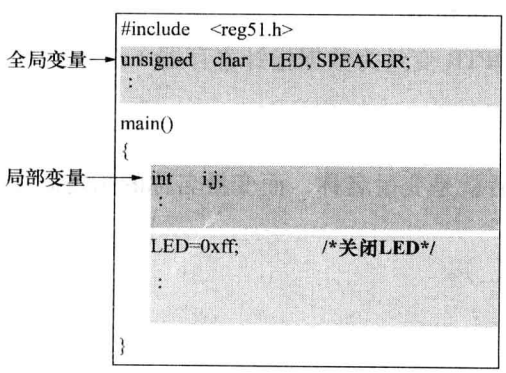

图 1 - 3　全局变量与局部变量

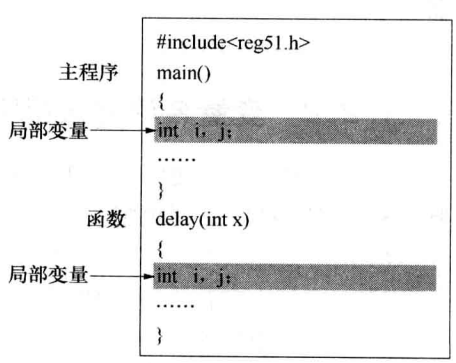

图 1 - 4　局部变量

1.3 存储器的形式与模式

8x51 的程序设计属于硬件的驱动程序，所以与 8x51 内部结构息息相关，特别是存储器，在本单元里将介绍 8x51 的存储器形式与其工作模式。

1.3.1 存储器的形式

Keil C 对于存储器的管理是将存储器分成 7 种形式，其中各种形式说明（使用小写）如表 1－5 所示。

表 1－5 存储器形式

存储器形式	说明	适用范围
code	程序存储器	0x0000～0xffff（64KB）
data	直接寻址的内部数据存储器	0x00～0x7f（128B）
idata	间接寻址的内部数据存储器	0x80～0xff（128B）
bdata	位寻址的内部数据存储器	0x20～0x2f（16B）
xdata	以 DPTR 寻址的外部数据存储器	64KB 之内
pdata	以 R0、R1 寻址的外部数据存储器	256B 之内
far	扩展的 ROM 或 RAM 外部存储器，仅适用于少数的芯片，如 Philips 80C51MX、Dallas 390 等。	最大可达 16MB

1. 程序存储器

程序存储器就是用来存放程序代码的存储器，除了用来存放程序代码外，也可存放固定的数据。例如，七段数码显示器的驱动信号、LED 点阵的显示信号、音乐的驱动信号、LCM 的显示字符串等，如下所示就是以数组方式存储的形式。

```
char code SEG［10］ = {0x03, 0x9f, 0x25, 0x0d, 0x99,
                       0x49, 0xcl, 0xlf, 0x01, 0x19};
```

标准的 8x51 内部的 4KB 程序存储器可扩展至 64KB；8x52 内部的 8KB 程序存储器可扩展至 64KB，而新版或兼容性的 51 芯片，其内部程序存储器容量达 16KB、32KB，甚至 64KB。

2. 内部数据存储器

由于 8x51 内部的特殊功能寄存器与数据存储器的地址相同，必须使用不同的寻址方式才能区分出是存取特殊功能寄存器还是存取数据存储器。当然，对于汇编语言而言，可以用不同的指令来区分直接寻址与间接寻址。

不过，Keil C 并没有直接寻址与间接寻址的语句，但可以用不同的存储器形式来区分操作的对象，因此就有 data、idata 及 bdata 三种存储器形式。其中 data 存储器形式可直接存取 Ox00～0x7f 数据存储器。例如，指定 x 为字符类型变量的语句为：

```
char  data  x;
```

idata 存储器形式可用间接寻址方式存取 0x80～0xff 数据存储器，其语句格式：