

玩转
单片机

51单片机

C语言 非常入门 与视频演练

刘建清 编著



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

<http://www.phei.com.cn>

玩转单片机

51 单片机C语言非常入门与 视频演练

刘建清 编著

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

内 容 简 介

本书简要介绍了 51 单片机 C 语言的基本知识、实验器材和实验方法,并演练了大量适宜初学者入门的典型实例;为方便读者实验,作者为本书的所有实例开发了 DD-900mini 实验板,并以视频的方式记录了书中主要实验的演示过程和现象。尤其珍贵的是,书中实例中的多数子程序均具有较强的通用性,读者只需将其简单修改甚至不用修改,即可移植到自己开发的产品中。

全书语言通俗、实例丰富、图文结合、简单明了,适合单片机爱好者和单片机初学者,也可作为中等专业技术学校、中等职业学校等教学用书。

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有,侵权必究。

图书在版编目(CIP)数据

51 单片机 C 语言非常入门与视频演练 / 刘建清编著. —北京: 电子工业出版社, 2010.5

(玩转单片机)

ISBN 978-7-121-10882-2

I. ①5… II. ①刘… III. ①单片微型计算机—C 语言—程序设计 IV. ①TP368.1 ②TP312

中国版本图书馆 CIP 数据核字 (2010) 第 087916 号

责任编辑: 徐 静 康 霞 特约编辑: 孙志明

印 刷: 北京市天竺颖华印刷厂

装 订: 三河市鑫金马印装有限公司

出版发行: 电子工业出版社

北京市海淀区万寿路 173 信箱 邮编 100036

开 本: 787×1092 1/16 印张: 12.5 字数: 312 千字

印 次: 2010 年 5 月第 1 次印刷

印 数: 4000 册 定价: 42.00 元 (含光盘 1 张)



凡所购买电子工业出版社图书有缺损问题, 请向购买书店调换。若书店售缺, 请与本社发行部联系, 联系及邮购电话: (010) 88254888。

质量投诉请发邮件至 zlts@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线: (010) 88258888。

前言

单片机就是把一个计算机系统集成到一个芯片上,概括地讲,一块芯片就成了一台计算机,目前,市场上流行的单片机,其价格出奇地便宜。对于广大爱好者来说,真是上帝的礼物。只要你玩起了单片机,你就会有一种成就感,我怎么这样聪明!单片机,再结合适当的硬件接口电路,有什么事情做不到呢?我对它的评价是八个字:软硬兼施,老少皆宜。

单片机虽然好玩,但是,很多人经过一番探索之后却深感学好单片机并非易事,甚至连入门都感到困难。作者本人也是从一位电子爱好者成长为一名工程师的,此过程自然少不了学习、探索、实践、再学习、再实践这样一条规律。因此深切地知道,学单片机难,主要是不得要领,难以入门。一旦找到学习的捷径,入了门,掌握简单程序的编写方法并观察到实际演示效果,必然信心大增。接下来,再向深度、广度进军时,心里就比较坦然了,最终能够一步一个脚印地去扩展自己的知识面,成为单片机的编程高手。

在与众多的单片机爱好者交流中得知,单纯讲单片机内部结构、指令太枯燥,且不易理解。他们感兴趣的是单片机编程的应用实例,而且主要喜欢简单、实用、有趣的初级实例。因此,编写本书的思路是:以实战演练为主线贯穿全书,且多数实例采用视频的方式进行演示。这样,初学者能够看得清、听得到、学得快,从而达到很好的立体学习效果。

在内容安排上,本书通过 51 单片机内部资源(中断系统、定时/计数器、串口通信)、键盘接口、LED 数码管显示、LCD 液晶显示、DS1302 时钟芯片、I²C 总线接口芯片 AT24C04、DS18B20 温度传感器、红外遥控、音乐发声等大量具体的实际例子,系统演练了 51 单片机中最为常用、最为典型的接口应用。另外,本书也包括了一些作者在学习和实际设计过程中总结的经验及方法,希望能够帮助读者更好地学习 51 单片机。

本书安排的实例大部分是由作者编写的,有一些是参考相关资料改写的,全部程序都由作者调试并通过。对于实例的使用说明也尽量详细,力争让读者“看则能用,用则能成”,保证读者在动手的过程中常常体会到成功的乐趣。另外,书中的所有实例,都是基于作者设计的 DD-900mini 实验板之上的。本书附带的光盘中含有所有实验的完整源程序、视频演示和工具软件。

本书主要面向的读者是具有一定 C 语言基础或刚接触 51 单片机的电子爱好者,对于已经熟悉 51 单片机 C 语言开发的工程师则意义不大。

本书编写过程中,参阅了《无线电》、《单片机与嵌入式系统应用》等杂志,并从互联网上搜索了一些有价值的资料,由于其中的很多资料经过多次转载,已经很难查到原始出处,在此谨向资料提供者表示感谢。

参与本书编写的人员有刘建清、贾绪岩、李凤伟、陈素侠、孙保书、刘为国等，最后由中国电子学会高级会员刘建清先生组织定稿。由于编著者水平有限，加之时间仓促，书中难免会有疏漏和不足之处，恳请专家和读者不吝赐教。

如果您在使用本书的过程中有任何问题、意见或建议，请登录顶顶电子网站：www.ddmcu.com，也可通过 E-mail: ddmcu@163.com 向我们提出，我们将为您提供超值延伸服务，另外书中所需实验板也可在该网站购买。

最后要说明的是，本书中所有实例均采用了 C 语言进行编程，如果您喜欢用汇编语言，建议选用本书的姊妹篇《51 单片机非常入门与视频演练》一书。

编著者

2010 年 3 月

目 录

第 1 章 单片机 C 语言非常入门	1
1.1 认识 C 语言	1
1.1.1 单片机采用 C 语言编程的好处	1
1.1.2 如何学习单片机 C 语言	2
1.2 简单的 C 语言程序	2
1.2.1 一个简单的流水灯程序	2
1.2.2 利用 C51 库函数实现流水灯	7
本章小结	7
第 2 章 51 单片机实验器材介绍与实验过程演示	9
2.1 DD-900mini 实验板介绍	9
2.1.1 DD-900mini 实验板硬件资源和接口	10
2.1.2 硬件电路介绍	10
2.1.3 仿真功能的使用	14
2.1.4 使用串口进行程序下载	15
2.1.5 笔记本电脑使用 DD-900mini 实验板	17
2.2 DD-51 编程器介绍与使用	18
2.2.1 DD-51 编程器介绍	18
2.2.2 DD-51 编程器使用	19
2.3 DD-F51 仿真器介绍与使用	20
2.3.1 DD-F51 仿真器介绍	20
2.3.2 DD-F51 仿真器的组成与使用	21
2.4 51 单片机实验过程演示	22
2.4.1 编写程序	22
2.4.2 编译程序	27
2.4.3 仿真调试	28
2.4.4 烧写程序	31
2.4.5 脱机运行检查	31
第 3 章 单片机 C51 学习与演练	32
3.1 标识符和关键字	32
3.1.1 标识符	32

3.1.2	关键字	32
3.2	数据类型介绍与演练	34
3.2.1	数据类型介绍	34
3.2.2	数据类型演练	36
3.3	常量、变量介绍与演练	38
3.3.1	常量	38
3.3.2	变量	40
3.3.3	常量与变量演练	41
3.4	运算符、表达式介绍与演练	44
3.4.1	运算符、表达式介绍	44
3.4.2	运算符、表达式演练	48
3.5	C51 基本语句介绍与演练	50
3.5.1	表达式语句和复合语句	50
3.5.2	条件选择语句	50
3.5.3	循环语句	52
3.5.4	C51 基本语句演练	55
3.6	C51 函数介绍与演练	63
3.6.1	函数概述	63
3.6.2	函数的参数和返回值	64
3.6.3	函数的调用	65
3.6.4	局部变量和全局变量	66
3.6.5	变量的存储种类	67
3.6.6	函数演练	68
3.7	C51 数组介绍与演练	71
3.7.1	一维数组	71
3.7.2	二维数组	73
3.7.3	字符数组	74
3.7.4	数组演练	74
3.8	C51 指针介绍与演练	76
3.8.1	指针概述	76
3.8.2	一般指针和基于存储器的指针	80
3.8.3	绝对地址的访问	81
3.8.4	指针演练	82
第 4 章	单片机内部资源视频演练	85
4.1	中断系统介绍与视频演练	85
4.1.1	51 单片机的中断源	85
4.1.2	中断的控制	86
4.1.3	中断的响应	88
4.1.4	中断的撤除	89

4.1.5	C51 中断函数的写法	89
4.1.6	视频演练 1——外中断练习	89
4.2	定时/计数器视频演练	91
4.2.1	定时/计数器的组成	91
4.2.2	定时/计数器的寄存器	91
4.2.3	定时/计数器的工作方式	92
4.2.4	视频演练 2——定时实验	96
4.2.5	视频演练 3——计数实验	98
4.3	RS-232 串行通信视频演练	99
4.3.1	什么是 RS-232 串行通信	99
4.3.2	51 单片机串行口的结构	100
4.3.3	串行通信控制寄存器	100
4.3.4	串行口工作方式	102
4.3.5	视频演练 4——PC 通过 RS232 和单片机通信	104
第 5 章	键盘接口电路视频演练	107
5.1	键盘接口电路基本知识	107
5.1.1	键盘的工作原理	107
5.1.2	键盘的扫描方式	108
5.1.3	DD-900mini 实验板中的独立按键	108
5.2	独立按键视频演练	108
5.2.1	视频演练 1——按键扫描方式练习	108
5.2.2	视频演练 2——可控流水灯	111
第 6 章	LED 数码管视频演练	114
6.1	LED 数码管介绍	114
6.1.1	LED 数码管的结构	114
6.1.2	LED 数码管的显示码	115
6.1.3	LED 数码管的显示方式	116
6.2	LED 数码管视频演练	118
6.2.1	视频演练 1——数码管动态扫描演示	118
6.2.2	视频演练 2——数码管电子钟	120
第 7 章	LCD 显示视频演练	127
7.1	字符型 LCD 介绍	127
7.1.1	字符型 LCD 引脚功能	127
7.1.2	字符型 LCD 内部结构	128
7.1.3	字符型 LCD 控制指令	130
7.1.4	字符型 LCD 与单片机的连接	133
7.1.5	字符型 LCD 驱动程序软件包的制作	133

7.2	字符型 LCD 视频演练	135
7.2.1	视频演练 1——1602 LCD 显示静止的字符串	135
7.2.2	视频演练 2——1602 LCD 显示从右向左移动的字符串	137
7.2.3	视频演练 3——1602 LCD 电子钟	138
第 8 章	时钟芯片 DS1302 视频演练	143
8.1	时钟芯片 DS1302 介绍	143
8.1.1	DS1302 概述	143
8.1.2	DS1302 的控制命令字	144
8.1.3	DS1302 的寄存器	144
8.1.4	DS1302 的数据传送方式	146
8.1.5	DS1302 驱动程序软件包的制作	146
8.2	DS1302 数码管电子钟视频演练	148
8.2.1	实现功能	148
8.2.2	源程序	148
8.2.3	源程序解读	151
8.2.4	视频演示	151
第 9 章	单片机读写 I ² C 总线视频演练	152
9.1	I ² C 总线介绍	152
9.1.1	I ² C 总线工作原理	152
9.1.2	I ² C 总线的电气结构	153
9.1.3	I ² C 总线器件的寻址方式	153
9.1.4	I ² C 总线数据的传输规则	153
9.1.5	I ² C 总线数据的读写格式	154
9.1.6	I ² C 总线接口芯片 24C04 介绍	155
9.1.7	I ² C 总线驱动程序软件包的制作	156
9.2	I ² C 总线接口芯片 24C04 视频演练	159
9.2.1	视频演练 1——具有记忆功能的计数器	159
9.2.2	视频演练 2——花样流水灯	161
第 10 章	温度传感器 DS18B20 视频演练	164
10.1	温度传感器 DS18B20 介绍	164
10.1.1	DS18B20 引脚功能	164
10.1.2	DS18B20 的内部结构	164
10.1.3	DS18B20 的指令	166
10.1.4	DS18B20 使用注意事项	167
10.1.5	温度传感器 DS18B20 驱动程序软件包的制作	167
10.2	DS18B20 LED 数字温度计视频演练	168
10.2.1	实现功能	168

10.2.2	源程序	168
10.2.3	源程序解读	170
10.2.4	视频演示	171
第 11 章	红外遥控视频演练	172
11.1	红外遥控基本知识	172
11.1.1	红外遥控系统	172
11.1.2	红外遥控的编码与解码	172
11.1.3	DD-900mini 实验板遥控电路介绍	174
11.2	红外遥控视频演练	174
11.2.1	视频演练 1——LED 数码管显示遥控器键值	174
11.2.2	视频演练 2——LCD 显示遥控器键值	177
第 12 章	单片机音乐发声视频演练	181
12.1	单片机音乐发声简介	181
12.2	单片机音乐发声视频演练	181
12.2.1	视频演练 1——单片机发出模拟枪声	181
12.2.2	视频演练 2——单片机发出模拟救护车声	182
12.2.3	视频演练 3——单片机发出模拟消防车声	185
12.2.4	视频演练 4——单片机唱歌	186
参考文献		190

第1章 单片机C语言非常入门

从前，汇编语言是单片机编程的唯一选择，但汇编语言的可读性和可移植性较差，调试和排错也比较困难，而且复杂一点的程序往往隔一段时间以后再看，还要花一定时间研究后才能看懂，可读性差。于是，人们开始寻求一种新的编程语言。C语言的出现，打破了这种现状，在本章，我们就来认识和熟悉一下C语言。

1.1 认识C语言

C语言是一种结构化语言。它层次清晰，便于按模块化方式组织程序，易于调试和维护。C语言的表现能力和处理能力极强，它不仅具有丰富的运算符和数据类型，便于实现各类复杂的数据结构。它还可以直接访问内存的物理地址，进行位（bit）一级的操作。由于C语言实现了对硬件的编程操作，因此，C语言集高级语言和低级语言的功能于一体，效率高，可移植性强，特别适合单片机系统的编程与开发。

1.1.1 单片机采用C语言编程的好处

与汇编语言相比，C语言在功能、结构性、可读性、可维护性上均有明显的优势，因而易学易用。用过汇编语言后再使用C语言来开发，体会更加深刻。下面简要说明单片机采用C语言编程的几点好处。

（1）语言简洁，使用方便灵活

C语言是现有程序设计语言中规模最小的语言之一，C语言的关键字很少，ANSI C标准一共只有32个关键字，9种控制语句，压缩了一切不必要的成分。C语言的书写形式比较自由，表达方法简洁，使用一些简单的方法就可以构造出相当复杂的数据类型和程序结构。同时，当前几乎所有单片机都有相应的C语言级别的仿真调试系统，调试十分方便。

（2）代码编译效率较高

当前，较好的C语言编译系统编译出来的代码效率只比直接使用汇编语言低20%左右，如果使用优化编译选项甚至可以更低。况且，随着单片机技术的发展，ROM空间不断提高，51系列单片机中，片上ROM空间做到32KB、64KB的比比皆是，代码效率所差的20%已经不是一个重要问题。

（3）无须深入理解单片机内部结构

采用汇编语言进行编程时，编程者必须对单片机的内部结构及寄存器的使用方法十分清楚；在编程时，一般还要进行RAM分配，稍不小心，就会发生变量地址重复或冲突。

采用C语言进行设计，则不必对单片机硬件结构有很深入的了解，编译器可以自动完成变量存储单元的分配，编程者可以专注于应用软件部分的设计，大大加快了软件的开发速度。

（4）可进行模块化开发

C语言以函数作为程序设计的基本单位，其中的函数相当于汇编语言中的子程序。各种

C 语言编译器都会提供一个函数库。此外，C 语言还具有自定义函数的功能，用户可以根据自己的需要编制满足某种特殊需要的自定义函数（程序模块），这些程序模块可不经修改，直接被其他项目所用；因此，采用 C 语言编程，可以最大程度地实现资源共享。

（5）可移植性好

用过汇编语言的读者都知道，即使是功能完全相同的一种程序，对于不同的单片机，必须采用不同的汇编语言来编写。这是因为汇编语言完全依赖于单片机硬件。C 语言是通过编译来得到可执行代码的，本身不依赖机器硬件系统，用 C 语言编写的程序基本上不用修改或者进行简单的修改，即可方便地移植到另一种结构类型的单片机上。

（6）可以直接操作硬件

C 语言具有直接访问单片机物理地址的能力，可以直接访问片内或片外存储器，还可以进行各种位操作。

在此，我想说一下我学习单片机编程的一个小插曲。在 20 世纪 90 年代中期，我最初接触单片机时，在心中觉得 51 就是单片机，单片机就是 51，根本不知道还有其他的单片机存在。那时候，我学习的是汇编语言，根本不知道用 C 语言也可以进行单片机开发。幸运的是，我有一个同事，比较精通 C 语言，我们一起搞一个项目时，我才真正发现 C 语言的威力。于是，在同事的影响下，我开始使用 C 语言进行单片机编程，并且一发而不可收！其实我也很庆幸学习和使用了两年多的汇编语言。由于有了这些锻炼，我对单片机底层结构和接口时序弄得很清楚。在使用 C 语言开发的时候，优化代码和处理中断也就不会太费劲。我觉得，虽然现在绝大部分单片机开发都使用 C 语言，这样对于项目的开展从时间上来说快了很多，在管理上也规范了不少，但是从学习和深入掌握单片机精髓的角度来说，还是需要掌握汇编语言的，等掌握到一定程度后，再学习单片机 C 语言编程，就会十分方便和容易。

总之，用 C 语言进行单片机程序设计是单片机开发与应用的必然趋势，我们一旦学会使用 C 语言之后，就会对它爱不释手，尤其是在进行大型单片机应用程序开发时，C 语言几乎是唯一的选择。

1.1.2 如何学习单片机 C 语言

C 语言常用语法不多，尤其是单片机的 C 语言常用语法更少，初学者没有必要将 C 语言的所有内容都学习一遍，只要跟着本书学下去，当遇到难点时，停下来适当地查阅 C 语言基础教材里的相关部分，便会很容易掌握。有关 C 语言的基础教材较多，在这里，笔者向大家推荐谭浩强的《C 程序设计》一书，该书语言通俗，实例丰富，十分适合初学者学习和查阅。

C 语言仅仅是一个编程语言，其本身并不难，难的是如何灵活运用 C 语言编写出结构完善的单片机程序。要达到这一点，就必须花费大量的时间进行实践、实验，仅看书不动手，等于是纸上谈兵，很难成功！因此，本书主要是通过不断地实践、实验，使读者在不知不觉中，进入单片机的大门。

1.2 简单的 C 语言程序

1.2.1 一个简单的流水灯程序

下面我们先来看一个实例，这个例子的功能十分简单，就是让单片机 P0 口的 LED 灯按

流水灯的形式进行闪烁，每个 LED 灯的闪烁时间为 0.5s，硬件电路如图 1-1 所示。

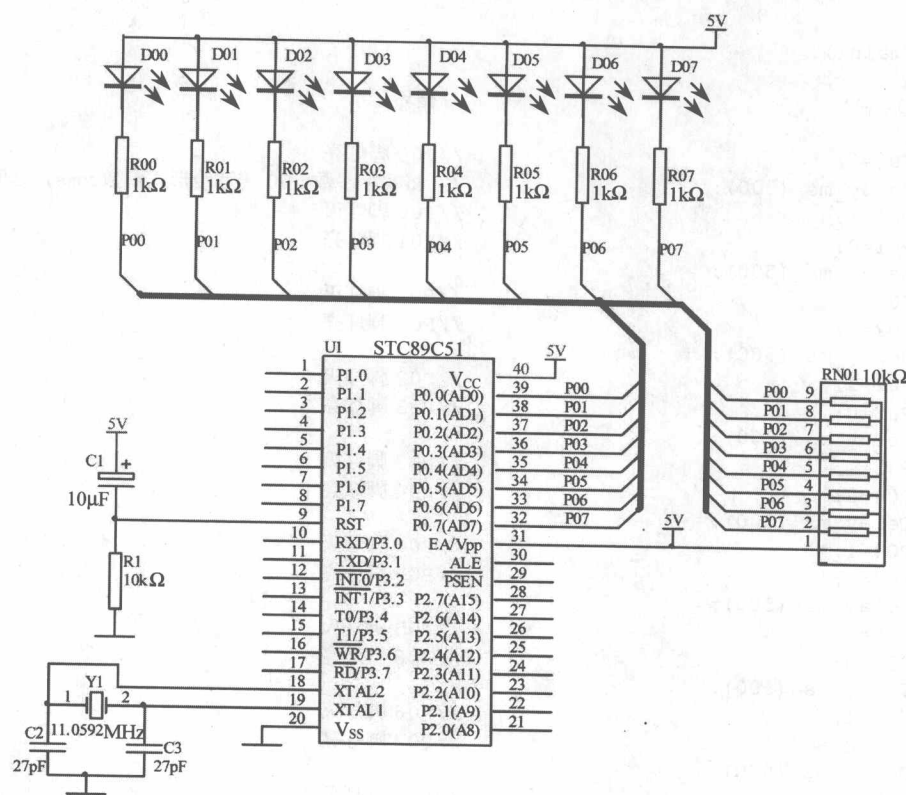


图 1-1 点亮 P0 口 LED 灯电路

图 1-1 中采用 STC89C51 单片机，这种单片机属于 80C51 系列，其内部有 4KB 的 Flash ROM 和 512B 的 RAM，并且可以通过串口进行 ISP 程序下载，不需要反复插拔芯片，非常适于做实验。STC89C51 的 P0 引脚上接 8 个发光二极管，R00~R07 为限流电阻，以免 LED 被烧坏。由于 P0 口是漏极开路的“双向 I/O 口”，需外接上拉电阻，图 1-1 中的 RN01 就是 P0 口的外接上拉电阻排。

根据要求，用 C 语言编写的程序如下：

```
#include<reg51.h>
#define uint unsigned int
sbit P00=P0^0; //定义位变量
sbit P01=P0^1;
sbit P02=P0^2;
sbit P03=P0^3;
sbit P04=P0^4;
sbit P05=P0^5;
sbit P06=P0^6;
sbit P07=P0^7;
void Delay_ms(uint xms) //延时程序，xms 是形式参数
{
    uint i, j;
    for(i=xms;i>0;i--) // i=xms,即延时 xms, xms 由实际参数传入一个值
    {
        for(j=115;j>0;j--)
```

```

    {;} //此处分号不可少，表示是一个空语句
}
}
void main()
{
    while(1) //循环显示
    {
        P00=0; //P00 脚灯亮
        Delay_ms (500); // 将实际参数 500 传递给形式参数 xms，延时 0.5s
        P00=1; //P00 脚灯灭
        P01=0; //P01 脚灯亮
        Delay_ms (500);
        P01=1; //P01 脚灯灭
        P02=0; //P02 脚灯亮
        Delay_ms (500);
        P02=1; //P02 脚灯灭
        P03=0; //P03 脚灯亮
        Delay_ms (500);
        P03=1; //P03 脚灯灭
        P04=0; //P04 脚灯亮
        Delay_ms (500);
        P04=1; //P04 脚灯灭
        P05=0; //P05 脚灯亮
        Delay_ms (500);
        P05=1; //P05 脚灯灭
        P06=0; //P06 脚灯亮
        Delay_ms (500);
        P06=1; //P06 脚灯灭
        P07=0; //P07 脚灯亮
        Delay_ms (500);
        P07=1; //P07 脚灯灭
    }
}

```

下面我们对这个程序进行简要分析。

程序的第 1 行是“文件包含”，所谓“文件包含”是指一个文件将另外一个文件的内容全部包含进来。所以，这里的程序虽然只有几行，但 C 语言编译器（Keil C51 软件）在处理的时候却要处理几十行或几百行。为加深理解，可以用任何一个文本编辑器打开 Keil/C51/inc 文件夹下面的 reg51.h 来看一看里面有什么内容，在 C 语言编译器处理这个程序时，这些内容也会被处理。这个程序包含 reg.h 的目的就是为了使用 P0 这个符号，即通知 C 语言编译器程序中所写的 P0 是指 80C51 单片机的 P0 口，而不是其他变量，这是如何做到的呢？用写字板程序打开 reg.h 显示如下：

```

#ifndef __REG51_H__
#define __REG51_H__
/* BYTE Register */
sfr P0    = 0x80;
sfr P1    = 0x90;
sfr P2    = 0xA0;
sfr P3    = 0xB0;
sfr PSW   = 0xD0;
sfr ACC   = 0xE0;
sfr B     = 0xF0;
sfr SP    = 0x81;
sfr DPL   = 0x82;
sfr DPH   = 0x83;
sfr PCON  = 0x87;
sfr TCON  = 0x88;

```



```

sfr TMOD = 0x89;
sfr TL0 = 0x8A;
sfr TL1 = 0x8B;
sfr TH0 = 0x8C;
sfr TH1 = 0x8D;
sfr IE = 0xA8;
sfr IP = 0xB8;
sfr SCON = 0x98;
sfr SBUF = 0x99;
/* BIT Register */
/* PSW */
sbit CY = 0xD7;
sbit AC = 0xD6;
sbit F0 = 0xD5;
sbit RS1 = 0xD4;
sbit RS0 = 0xD3;
sbit OV = 0xD2;
sbit P = 0xD0;
/* TCON */
sbit TF1 = 0x8F;
sbit TR1 = 0x8E;
sbit TF0 = 0x8D;
sbit TR0 = 0x8C;
sbit IE1 = 0x8B;
sbit IT1 = 0x8A;
sbit IE0 = 0x89;
sbit IT0 = 0x88;
/* IE */
sbit EA = 0xAF;
sbit ES = 0xAC;
sbit ET1 = 0xAB;
sbit EX1 = 0xAA;
sbit ET0 = 0xA9;
sbit EX0 = 0xA8;
/* IP */
sbit PS = 0xBC;
sbit PT1 = 0xBB;
sbit PX1 = 0xBA;
sbit PT0 = 0xB9;
sbit PX0 = 0xB8;
/* P3 */
sbit RD = 0xB7;
sbit WR = 0xB6;
sbit T1 = 0xB5;
sbit T0 = 0xB4;
sbit INT1 = 0xB3;
sbit INT0 = 0xB2;
sbit TXD = 0xB1;
sbit RXD = 0xB0;
/* SCON */
sbit SM0 = 0x9F;
sbit SM1 = 0x9E;
sbit SM2 = 0x9D;
sbit REN = 0x9C;
sbit TB8 = 0x9B;
sbit RB8 = 0x9A;
sbit TI = 0x99;
sbit RI = 0x98;
#endif

```

可以看到：“sfr P0=0x80;”，即定义符号 P0 与地址 0x80 对应，熟悉 80C51 内部结构的读者不难看出，P0 口的地址就是 0x80。

程序的第2行`#define uint unsigned int`是一个宏定义语句，注意后面没有分号，`#define`命令用它后面的第1个字母组合代替该字母组合后面的所有内容，也就是相当于我们给“原内容”重新起一个比较简单的“新名称”，方便以后在程序中直接写简短的新名称，而不必每次都写烦琐的原内容。该例中，我们使用宏定义的目的就是将 `unsigned int` 用 `uint` 代替，在上面的程序中可以看到，在我们需要定义 `unsigned int` 类型变量时，并没有写 `unsigned int`，取而代之的是 `uint`。

程序的第3~第10行用符号 `P00~P07` 来表示 `P0` 口的 `P0.0~P0.7` 引脚，在 C 语言里，如果直接写 `P0.0, P0.1, ..., P0.7`，C 语言编译器并不能识别，而且它们不是一个合法的 C 语言变量名，所以得给它另起一个名字，即 `P00~P07`，可是 `P00~P07` 是否就是 `P0.0~P0.7` 呢？你这么认为，C 语言编译器可不这么认为，所以必须让它们建立起来联系，这里使用了 Keil C51 的保留字 `sbit` 来定义。

`main` 称为“主函数”，每一个 C 语言程序有且只有一个主函数，函数后面一定有一对大括号“`{ }`”、在大括号里面书写其他程序。

`Delay_ms(500)` 的用途是延时，由于单片机执行指令的速度很快，如果不进行延时，灯亮之后马上就灭，灭了之后马上就亮，速度太快，人眼根本无法分辨，所以需要进行适当的延时。这里采用自定义函数 `Delay_ms(500)`，以延时 0.5s 的时间，函数前面的 `void` 表示该延时函数没有返回值。

`Delay_ms(500)` 函数是一个自定义函数，它不是由 Keil C51 编译器提供的，即你不能在任何情况下写这样一程序以实现延时，如果在编写其他程序时写上这么一行，会发现编译不过。注意观察本程序你会发现，在使用 `Delay_ms(500)` 之前，第11~第16行已对 `Delay_ms(uint xms)` 函数进行了事先定义，因此，在主程序中才能采用 `Delay_ms(500)` 进行使用。

注意：在延时函数 `Delay_ms(uint xms)` 定义中，参数 `xms` 被称做“形式参数”（简称形参）；而在调用延时函数 `Delay_ms(500)` 中，小括号里的数据“500”称做“实际参数”（简称实参），参数的传递是单向的，即只能把实参的值传给形参，而不能把形参的值传给实参。另外，实参可以在一定范围内调整，这里用“500”来要求延时时间为 0.5s，如果是“1000”，则延时时间是 1000ms，即 1s。

在延时函数 `Delay_ms(uint xms)` 内部，采用了两层嵌套 `for` 语句，如下所示：

```
void Delay_ms(uint xms)           //延时程序，xms 是形式参数
{
    uint i, j;
    for(i=xms;i>0;i--)           // i=xms,即延时 xms, xms 由实际参数传入一个值
    {
        for(j=115;j>0;j--)
        {;}                      //此处分号不可少，表示是一个空语句
    }
}
```

在这个延时函数中，采用的是一种比较正规的形式，C 语言规定，当循环语句后面的大括号只有一条语句或为空时，可省略大括号。因此，上面两个 `for` 循环语句中的大括号都可以省略，也就是说，可以采用以下简化的形式：

```
void Delay_ms(uint xms)           //延时程序，xms 是形式参数
{
    uint i, j;
    for(i=xms;i>0;i--)           // i=xms,即延时 xms, xms 由实际参数传入一个值
        for(j=115;j>0;j--);      //此处分号不可少
}
```


第 1 个 for 语句后面没有分号，那么编译器就会认为第 2 个 for 语句就是第一个 for 语句的内部语句，而第 2 个 for 语句后面有分号，编译器就会认为第 2 个 for 语句内部语句为空。程序在执行时，第 1 个 for 语句中的 i 每减一次，第 2 个 for 语句便执行 115 次，因此上面这个例子便相当于共执行了 $xms \times 115$ 次 for 语句。通过改变 xms 变量的值，可以改变延时时间。

1.2.2 利用 C51 库函数实现流水灯

上面介绍的程序虽然可以实现流水灯的功能，程序比较烦琐。下面采用 C51 自带的库函数 `_crol()` 来实现流水灯，具体源程序如下：

```
#include<reg51.h>
#include<intrins.h>
#define uint unsigned int
#define uchar unsigned char
void Delay_ms(uint xms)           //延时程序，xms 是形式参数
{
    uint i, j;
    for(i=xms;i>0;i--)             // i=xms,即延时 xms, xms 由实际参数传入一个值
        for(j=115;j>0;j--);       //此处分号不可少
}
void main()
{
    uchar led_data=0xfe;           //给 led_data 赋初值 0xfe, 点亮第一个 LED 灯
    while(1)                       //大循环
    {
        P0= led_data;
        Delay_ms(500);
        led_data=_crol_( led_data,1); //将 led_data 循环左移 1 位再赋值给 led_data
    }
}
```

显然，这个流水灯程序比上面的流水灯程序要简洁许多，下面简要进行说明。

程序中，`_crol` 是一个库函数，其函数原形为：

```
unsigned char _crol_(unsigned char c, unsigned char b);
```

这个函数是 C51 自带的库函数，包含在 `intrins.h` 头文件中，也就是说，如果在程序中要用到这个函数，那么必须在程序的开头处包含 `intrins.h` 头文件。函数实现的功能是将字符 c 循环左移 b 位。

函数中，`_crol` 是函数名，不用多讲，函数前面没有 `void`，取而代之的是 `unsigned char`，表示这个函数返回值是一个无符号字符型数据。有返回值的意思是说，程序执行完这个函数后，通过函数内部的某些运算而得出一个新值，该函数最终将这个新值返回给调用它的语句。小括号里有两个形参，`unsigned char c`，`unsigned char b`，它们都是无符号字符型数据。

现在我们应该清楚 `led_data=_crol_( led_data,1)` 这条语句的含义了，其作用就是将 `led_data` 中的数据向左循环移 1 位，再赋给变量 `led_data`。

有左移位库函数，当然也有右移位库函数，函数原形为：

```
unsigned char _crol_(unsigned char c, unsigned char b);
```

右移位函数与左移位函数使用方法相同，这里不再赘述。

本章小结

通过以上的几个简单 C 语言程序，我们可以总结出以下几点：