课题 1

认识单片机及 C 语言

计算机是应数值计算要求而诞生的，在相当长的时期内，计算机技术都是以满足越来越多的计算量为目标来发展的；但是当单片机出现后，计算机就从海量数值计算进入到智能化控制领域。从此，计算机就开始了沿着通用计算领域和嵌入式领域两条不同的道路发展。

1.1 单片机的发展

单片机自问世以来，以极高的性能价格比，越来越受到人们的重视和关注。目前，单片机被广泛地应用于智能仪表、机电设备、过程控制、数据处理、自动检测和家用电器等方面。

1.1.1 单片机名称的由来

无论规模大小、性能高低，计算机的硬件系统都是由运算器、存储器、输入设备、输出设备以及控制器等单元组成。在通用计算机中，这些单元被分成若干块独立的芯片，通过电路连接而构成一台完整的计算机。而单片机技术则将这些单元全部集成到一块集成电路中，即一块芯片就构成了一个完整的计算机系统。这成为当时这一类芯片的典型特征，因此，就用 Single Chip Microcomputer 来称呼这一类芯片，中文译为“单片机”，这在当时是一个准确的表达。但随着单片机技术的不断发展，“单片机”已无法确切地表达其内涵，国际上逐渐采用 MCU(Micro Controller Unit，微控制单元)来称呼这一类计算机，并成为单片机界公认的、最终统一的名词。但国内由于多年来一直使用“单片机”的称呼，已约定俗成，所以目前仍采用“单片机”这一名词。

1.1.2 单片机技术的发展历史

20 世纪 70 年代，美国仙童公司首先推出了第一款单片机 F－8，随后 Intel 公司推出了 MCS－48 单片机系列，其他一些公司如原 Motorola、Zilog 等也先后推出了自己的单片机，取得了一定的成果，这是单片机的起步与探索阶段。总体来说，这一阶段的单片机性能较弱，属于低、中档产品。

随着集成技术的提高以及 CMOS 技术的发展,单片机的性能也随之改善,高性能的 8 位单片机相继问世。1980 年 Intel 公司推出了 8 位高档 MCS－51 系列单片机,性能得到很大的提高,应用领域大为扩展。这是单片机的完善阶段。

1983 年 Intel 公司推出了 16 位 MCS－96 系列单片机,加入了更多的外围接口,如模/数转换器(ADC)、看门狗(WDT)、脉宽调制器(PWM)等,其他一些公司也相继推出了各自的高性能单片机系统。随后许多用在高端单片机上的技术被下移到 8 位单片机上,这些单片机内部一般都有非常丰富的外围接口,强化了智能控制器的特征,这是 8 位单片机与 16 位单片机的推出阶段。

近年来,Intel、NXP 等公司又先后推出了性能更为优异的 32 位单片机,单片机的应用达到了一个更新的层次。

随着技术的进步,早期的 8 位中、低档单片机逐渐被淘汰,但 8 位单片机并没有消失,尤其是以 80C51 为内核的单片机,不仅没有消失,还呈现快速发展的趋势。

目前单片机的发展有这样的一些特点:

CMOS 化 由于 CHMOS 技术的进步,大大地促进了单片机的 CMOS 化。CMOS 芯片除了低功耗特性之外,还具有功耗的可控性,使单片机可以工作在功耗精细管理状态。

低电压、低功耗化 单片机允许使用的电压范围越来越宽,一般在 3～6 V 范围内工作,低电压供电的单片机电源下限已可达 1～2 V,1 V 以下供电的单片机也已问世。单片机的工作电流已从 mA 级降到 μA 级,甚至 1 μA 以下;低功耗化的效应不仅是功耗低,而且带来了产品的高可靠性、高抗干扰能力以及产品的便携化。

大容量化 随着单片机控制范围的增加,控制功能的日渐复杂,高级语言的广泛应用,对于单片机的存储器容量提出了更高的要求。目前,单片机内 ROM 最大可达 256 KB 以上,RAM 可达 4 KB 以上。

高性能化 通过进一步改进 CPU 的性能,加快指令运算速度和提高系统控制的可靠性。采用精简指令集(RISC)结构和流水线技术,可以大幅度提高运行速度。现指令速度高者已达 100 MIPS(Million Instruction Per Seconds,即兆指令每秒)。

小容量、低价格化 以 4 位、8 位机为中心的小容量、低价格化是单片机的另一发展方向。这类单片机的用途是把以往用数字逻辑集成电路组成的控制电路单片化,可广泛用于家电产品。

串行扩展技术 在很长一段时间里,通用型单片机通过三总线结构扩展外围器件成为单片机应用的主流结构。随着低价位 OTP 及各种类型片内程序存储器技术的发展,加之外围接口不断进入片内,推动了单片机“单片”应用结构的发展。特别是 I^2C、SPI 等串行总线的引入,可以使单片机的引脚设计得更少,单片机系统结构更加简化及规范化。

ISP 技术 ISP(In－System Programming,在系统可编程)是指可以通过特定的编程工具对已安装在电路板上的器件编程写入最终用户代码,而不需要从电路板上

取下器件。利用 ISP 技术不需要编程器就可以进行单片机的实验和开发，单片机芯片可以直接焊接到电路板上，调试结束即成为成品，免去了调试时由于频繁地插入取出芯片对芯片和电路板带来的不便。

IAP 技术　IAP(In－Application Programming)是指在用户的应用程序中对单片机的程序存储器进行擦除和编程等操作，IAP 技术应用的一个典型例子是可以较为容易地实现硬件的远程升级。

在单片机家族中，80C51 系列是其中的佼佼者。Intel 公司将 80C51 单片机的内核以专利互换或出售的方式转让给其他许多公司，如原 Philips、Atmel、NEC 等，因此，目前有很多公司在生产以 80C51 为内核的单片机，这些单片机在保持与 80C51 单片机兼容的基础上，改善了 80C51 单片机的许多特性。这样，80C51 就成为有众多制造厂商支持的、在 CMOS 工艺基础上发展出上百品种的大家族，现统称为 80C51 系列。

这一系列单片机包括了很多种，其中 STC89C51 就是近年来流行的单片机，它由宏晶公司开发生产，其特点是内部有可以多次重复编程的 Flash ROM，可以通过单片机芯片自身的串口进行在系统编程，使用方便。

1.2　计算机数据表示

计算机用于处理各种信息，首先需要将信息表示成为具体的数据形式。选择什么样的数制来表示数，对机器的结构、性能和效率有很大的影响。二进制是计算机中数制的基础。

所谓二进制形式，是指每位数码只取二个值，要么是“0”，要么是“1”，数码最大值只能是 1，超过 1 就应向高位进位。为什么要采用二进制形式呢？这是因为二进制最简单，它仅有二个数字符号，这就特别适合于用电子元器件来实现。制造有两个稳定状态的元器件比制造具有多个稳定状态的元器件容易得多。

计算机内部采用二进制表示各种数据，对于单片机而言，其主要的数据类型分为数值数据和逻辑数据两种，下面分别介绍数制概念和各种数据的机内表示、运算等知识。

按进位的原则进行计数，称为进位计数制，简称“数制”。数制有多种，在计算机中常使用的有十进制、二进制和十六进制。

1.2.1　常用的进位计数制

1. 十进制数

按“逢十进一”的原则进行计数，称为十进制数。十进制的基为“10”，即它所使用的数码为 0～9，共 10 个数字。十进制各位的权是以 10 为底的幂，每个数所处的位置不同，它的值是不同的，每一位数是其右边相邻那位数的 10 倍。

对于任意一个十进制数,都可以写成如下的式子:

$$D_3D_2D_1D_0=D_3\times10^3+D_2\times10^2+D_1\times10^1+D_0\times10^0$$

上述式子各位的权分别是个、十、百、千,即以 10 为底的 0 次幂、1 次幂、2 次幂和 3 次幂,通常简称为 0 权位、1 权位、2 权位、3 权位等,上式称为按权展开式。

例:$3525=3\times10^3+5\times10^2+2\times10^1+5\times10^0$

2. 二进制数

按“逢二进一”的原则进行计数,称为二进制数。二进制的基为“2”,即其使用的数码为 0、1,共 2 个数字。二进制各位的权是以 2 为底的幂,任意一个 4 位二进制数按权展开式如下:

$$B_3B_2B_1B_0=B_3\times2^3+B_2\times2^2+B_1\times2^1+B_0\times2^0$$

由此可知,4 位二进制中各位的权是:

2^3	2^2	2^1	2^0
8	4	2	1

例:$(1011)_2=1\times2^3+0\times2^2+1\times2^1+1\times2^0=(11)_{10}$

3. 十六进制数

按“逢十六进一”的原则进行计数,称为十六进制数。十六进制的基为“16”,即其码共有 16 个:0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F。其中 A、B、C、D、E、F 所代表的数的大小相当于十进制的 10,11,12,13,14 和 15。十六进制的权是以 16 为底的幂,任意一个四位的十六进制数的按权展开式为:

$$H_3H_2H_1H_0=H_3\times16^3+H_2\times16^2+H_1\times16^1+H_0\times16^0$$

例:$(17F)_{16}=1\times16^2+7\times16^1+15\times16^0=(383)_{10}$

由于十六进制数易于书写和记忆,且与二进制之间的转换十分方便,因而人们在书写计算机语言时多用十六进制。

4. 二—十进制编码

计算机中使用的是二进制数,但人却习惯于使用十进制数,为此需要建立一个二进制数与十进制数联系的桥梁,这就是二—十进制。

在二—十进制中,十进制的十个基数符 0,1～9 用二进制码表示,而计数方法仍采用十进制,即“逢十进一”。为了要表示 10 种状态,必须要用 4 位二进制数(3 位只能表示 1—7,不够用)。4 位二进制一共有 16 种状态,可以取其中的任意 10 种状态来组成数符 0～9,显然,最自然的方法就是取前 10 种状态,这就是 BCD 码,也称之为 8421 码,因为这种码的 4 个位置的 1 分别代表了 8,4,2 和 1。

学习 BCD 码,一定要注意区分它与二进制的区别,表 1－1 列出几个数作为比较。

从表 1－1 中不难看出,对于小于 10 的数来说,BCD 码和二进制码没有什么区别,但对于大于 10 的数,BCD 码和二进制码就不一样了。

表 1-1　二进制、十进制、十六进制数、BCD 码的对应关系

十进制数	十六进制	二进制	BCD 码	十进制数	十六进制	二进制	BCD 码
0	0	00000000	00000000	10	A	00001010	00010000
1	1	00000001	00000001	11	B	00001011	00010001
2	2	00000010	00000010	12	C	00001100	00010010
3	3	00000011	00000011	15	F	00001111	00010101
4	4	00000100	00000100	100	64	110,0100	1,0000,0000

1.2.2　二进制的算术运算

二进制算术运算的规则非常简单，这里介绍常用的加法和乘法规则。

加法规则

0 + 0 = 0

0 + 1= 1

1 + 0 = 1

1 + 1 = 10

乘法规则

0× 0 = 0

0× 1 = 0

1× 0 = 0

1× 1 = 1

例 1： 求 11011+1101 的值

```
   1 1 0 1 1
+    1 1 0 1
-------------
 1 0 1 0 0 0
```

例 2： 求 11011×101 的值

```
     1 1 0 1 1
   ×     1 0 1
     1 1 0 1 1
   0 0 0 0 0
 1 1 0 1 1
-------------
1 0 0 0 0 1 1 1
```

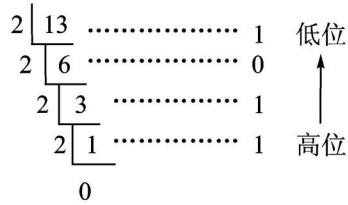
1.2.3　数制间的转换

将一个数由一种数制转换成另一种数制称之为数制间的转换。

1. 十进制数转换为二进制数

十进制转换为二进制采用“除二取余法”,即把待转换的十进制不断地用 2 除,一直到商是 0 为止,然后将所得的余数由下而上排列即可。

例 1:把十进制数 13 转换为二进制数



结果是十进制数$(13)_{10}$等于二进制数$(1101)_2$

2. 二进制数转换为十进制数

二进制数转换为十进制数采用“位权法”,即把各非十进制数按权展开,然后求和。

例 2:把$(1110110)_2$转换为十进制

$(1110110)_2=1\times2^6+1\times2^5+1\times2^4+0\times2^3+1\times2^2+1\times2^1+0\times2^0=(118)_{10}$

3. 二进制转换为十六进制

十六进制也是一种常用的数制,将二进制数转换为十六进制数的规则是“从右向左,每 4 位二进制化为一位十六进制,不足部分用零补齐”。

例 3:将$(1110000110110001111)_2$转化为十六进制

把$(1110000110110001111)_2$写成下面的形式:

0111 0000 1101 1000 1111

因此$(1110000110110001111)_2=(70D8F)_{16}$

4. 十六进制转换为二进制

十六进制化为二进制的方法正好和上面的方法相反,即一位十六进制数化为 4 位二进制数。

例 4:将$(145A)_{16}$转化为二进制数

将每位十六进制数写成 4 位二进制数,就是:0001 0100 0101 1010

即十六进制数$(145A)_{16}$等于二进制数$(1010001011010)_2$

1.2.4 数的表示方法及常用计数制的对应关系

1. 数的表达方法

为了便于书写,特别是方便编程时书写,规定在数字后面加一个字母以示区别,二进制后加 B,十六进制后加 H,十进制后面加 D,并规定 D 可以省略。这样 102 是

指十进制的 102，102H 是指十六进制的 102，也就是 258，同样 1101 是十进制 1101，而 1101B 则是指二进制的 1101，即 13。

2. 常用数制对应关系

表 1－2 列出了常用数值 1～15 的各种数制间的对应关系，这在以后的学习中经常要用到，要求熟练地掌握。

表 1－2　常用数制的对应关系

十进制	二进制	十六进制	十进制	二进制	十六进制
0	0000B	0H	8	1000B	8H
1	0001B	1H	9	1001B	9H
2	0010B	2H	10	1010B	0AH
3	0011B	3H	11	1011B	0BH
4	0100B	4H	12	1100B	0CH
5	0101B	5H	13	1101B	0DH
6	0110B	6H	14	1110B	0EH
7	0111B	7H	15	1111B	0FH

1.2.5　逻辑数据的表示

为了使计算机具有逻辑判断能力，就需要逻辑数据，并能对它们进行逻辑运算，得出一个逻辑式的判断结果。每个逻辑变量或逻辑运算的结果，产生逻辑值，该逻辑值仅取“真”或“假”两个值。判断成立为“真”，判断不成立为“假”。在计算机内常用 0 和 1 表示这两个逻辑值，0 表示假，1 表示真。

最基本的逻辑运算有“与”、“或”、“非”3 种。这 3 种运算分别描述如下：

1. 逻辑“与”

逻辑“与”也称之为逻辑乘，最基本的与运算有两个输入量和一个输出量。它的运算规则和等效的描述电路如图 1－1 所示。

逻辑“与”可以用两个串联的开关来等效。用语言描述就是：只有两个输入量都是“1”时，输出才为 1，或者说“有 0 为 0，全 1 出 1”。

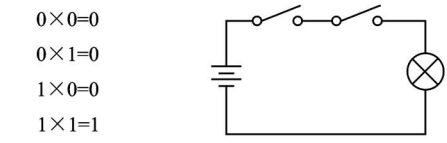


图 1－1　逻辑“与”的运算规则

2. 逻辑“或”

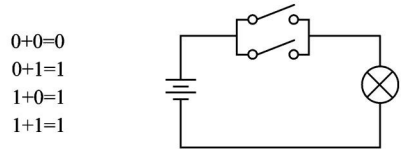
逻辑“或”也叫逻辑“加”，最基本的逻辑或有两个输入量和一个输出量。它的运

算规则等效的描述电路如图 1－2 所示。

逻辑“或”可用两个并联的开关来等效。用语言描述就是:只有两输入量都是“0”时,输出才为“0”,或者可以这样说“有 1 为 1,全 0 出 0”。

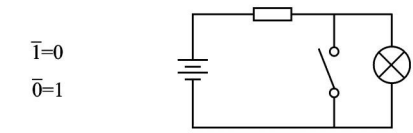
3. 逻辑“非”

逻辑“非”即取反,它的运算规则等效的描述电路如图 1－3 所示。



(a) 运算规则　　(b) 等效电路

图 1－2　逻辑“或”的运算规则



(a) 运算规则　　(b) 等效电路

图 1－3　逻辑“非”的运算规则

逻辑“非”可以用灯并联关来等效,用语言描述就是:1 的反是 0,0 的反是 1。

若在一个逻辑表达式中出现多种逻辑运算时,可用括号指定运算的次序,无括号时按逻辑“非”、逻辑“与”、逻辑“或”的顺序执行。

巩固与提高

1. 用十六进制数表示下列二进制:
   10100101B,11010111B,11000011B,10000111B
2. 将下列十进制转换为二进制:
   28D,34D,19D,33D
3. 将下列十六进制转换为二进制:
   35H,12H,8AH,F3H
4. 单片机内部采用什么数制工作?为什么?

1.3　计算机中常用的基本术语

在介绍概念之前,先看一个例子。

用于照明的灯有两种状态,即“亮”和“不亮”,如果规定灯亮为“1”,不亮为“0”,那么两盏灯的亮和灭的状态可列于表 1－3 中。

表 1－3　两盏灯的亮灭及数值表示

状　态	○	○	○	●	●	○	●	●
表　达	0	0	0	1	1	0	1	1

注:“○”表示灯不亮;“●”表示灯亮

从表 1－3 中可以看到，两盏灯一共能够呈现 4 种状态，即“00”、“01”、“10”和“11”，而二进制数 00、01、10、11 相当于十进制数的 0、1、2、3，因此，灯的状态可以用数学方法来描述，反之，数值也可以用电子元件的不同状态的组合来表示。

1. 位

一盏灯的亮和灭可以分别代表两种状态：0 和 1。实际上这就是一个二进制位，一盏灯就是一“位”。当然这只是一种帮助记忆的说法，位(bit)的定义是：位是计算机中所能表示的最小数据单位。

2. 字节

一盏灯可以表示 0 和 1 两种状态，两盏灯可以表达 00、01、10、11 共 4 种状态，也就是可以表示 0、1、2 和 3，计算机中通常把 8 位放在一起，同时计数，可以表达 1～255 一共 256 种状态。相邻 8 位二进制码称之为一个字节(byte)，用 B 表示。

字节(B)是一个比较小的单位，常用的还有 KB 和 MB 等，它们的关系是：

1 KB＝1 024 B

1 MB＝1 024 KB＝1 024×1 024 B

3. 字和字长

字是计算机内部进行数据处理的基本单位。它由若干位二进制码组成，通常与计算机内部的寄存器、运算器、数据总线的宽度一致，每个字所包含的位数称为字长。若干个字节定义为一个字，不同类型的微型计算机有不同的字长，如 80C51 系列单片机是 8 位机，就是指它的字长是 8 位，其内部的运算器等都是 8 位的，每次参加运算的二进制位只有 8 位。而以 8086 为主芯片的 PC 是 16 位的，即指每次参加运算的二进制位有 16 位。

字长是计算机的一个重要的性能指标，一般而言，字长越长，计算机的性能越好，下面通过例子作个说明。

8 位字长，其表达的数的范围是 0～255，这意味着参加运算的各个数据不能超过 255，并且运算的结果和中间结果也不能超过 255，否则就会出错，但是在解决实际问题时，往往有超过 255 的要求，比如单片机用于测量温度，假设测温范围是 0～1 000℃，这就超过了 255 所能表达的范围了，为了要表示这样的数，需要用两个字节组合起来表示温度。这样，在进行运算时就需要花更长的时间，比如做一次乘法，如果乘数和被乘数都用一个字节表示，只要一步(一行程序)就可以完成，而使用两个数组合起来，做一次乘法可能需要 5 步(五行程序)或更多才能完成。同样的问题，如果采用 16 位的计算机来解决，它的数的表达范围可以是 0～65 535，所以只要一次运算就可以解决问题，所需要的时间就少了。

1.4 存储器

存储器是任何计算机系统中都要用的,通过对存储器的工作原理的了解,可以学习计算机系统的一些最基本和最重要的概念。

在计算机中存储器用来存放数据。存储器中有大量的存储单元,每个存储单元都可以有“0”和“1”两种状态,即存储器是以“0”和“1”的组合来表示数据,而不是放入的如同十进制 1、2、3、4 这种形式的数据。

图 1-4 是一个有 4 个单元的存储器的示意图,该存储器一共有 4 个存储单元,每个存储单元内有 8 个小单元格(对应一个字节 8 个位)。有 D0～D7 共 8 根引线进入存储器的内部,经过一组开关,这组开关由一个称之为“控制器”的部件控制。而控制器则有一些引脚被送到存储器芯片的外部,可以由 CPU 对它进行控制。示意图的右侧还有一个称之为“译码器”的部分,它有两根输入线 A0、A1 由外部引入,译码器的另一侧有 4 根输出线,分别连接到每一个存储单元。

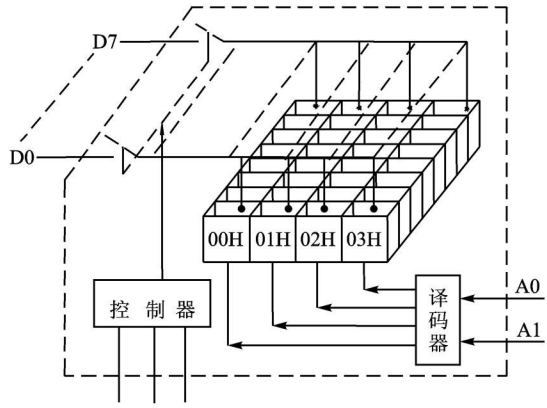


图 1-4 存储器单元示意图

为说明问题,把其中的一个单元画成一个独立的图,如图 1-5 所示,如果黑色单元代表“1”,白色单元代表“0”,则该存储单元的状态是 01001010,即 4AH。从图 1-4 可以看出,这个存储器一共有 4 个存储单元,每个存储单元的 8 根线是并联的,在对存储单元进行写操作时,会将待写入的“0”、“1”送入并联的所有 4 个存储单元中,换言之,一个存储器不管有多少个存储单元,都只能放同一个数,这显然不是所希望的,因此,要在结构上稍作变化。图 1-5 是带有控制线的存储单元示意图,在每个单元上有根控制线,CPU 准备把数据放进哪个单元,就送一个“导通”信号到这个单元

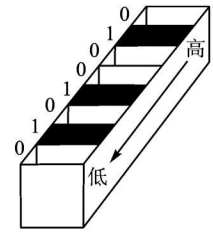


图 1-5 一个存储单元的示意图

的控制线，这个控制线把开关合上，这样该存储单元中的数据就可以与外界进行交换了。而其他单元控制线没有“导通”信号，开关打开着，不会受到影响，这样，只要控制不同单元的控制线，就可以向各单元写入不同的数据或从各单元中读出不同的数据。这个控制线应当由一个系统中的主机（CPU 或单片机）进行控制，因为 CPU（或单片机）是整个计算机系统的“大脑”，只有它才能确定什么时候该把数据放在某一个单元中，什么时候该从哪一个单元中获取数据。为了使得数据的存储不发生混淆，要给每个存储单元分配一个唯一的固定编号，这个编号就称为存储单元的地址。

为了控制各个单元而把每个单元的控制线都引到集成电路的外面是不可行的。上述存储器仅有 4 个存储单元，而实际的存储器，其存储单元数很多。比如，27C512 存储器芯片有 65 536 个单元，需要 65 536 根控制线，不可能将每控制根线都引到集成电路的外面来，因此，在存储器内部带有译码器，译码器的输出端即通向各存储单元的控制线，译码器的输入端通过集成电路外部引脚接入，被称之为地址线。由于 65 536 根控制线在任一时刻只有一根起作用，即 65 536 根线只有 65 536 种状态，而每一根地址线都可以有 0 和 1 两种状态，n 根线就有 2^n 种状态。因为 $2^{16}=65\ 536$，因此，只需要 16 根引线就能确定 27C512 的每一个地址单元。带有控制线的存储单元如图 1－6 所示。

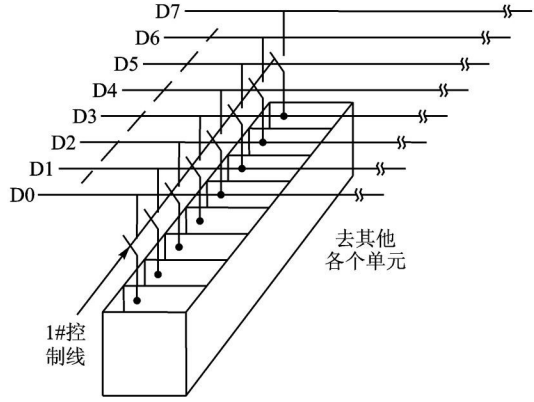


图 1－6　带有控制线的存储单元示意图

半导体存储器按功能可以分为只读、随机存取存储器和可现场修改的非易失存储器 3 大类。

1. 只读存储器

只读存储器又称为 ROM，其中的内容在操作运行过程中只能被 CPU 读出，而不能写入或更新。它类似于印好的书，只能读书里面的内容，不可以随意更改书里面的内容。只读存储器的特点是断电后存储器中的数据不会丢失，这类存储器适用于存放各种固定的系统程序、应用程序和表格等，所以人们又常称 ROM 为程序存储器。

只读存储器又可以分为以下这些品种：

(1) 掩膜 ROM：由器件生产厂家在设计集成电路时一次固化，此后便不能被改变，它相当于印好的书。这种 ROM 成本低廉，适用于大批量生产。

(2) PROM，称之为可编程存储器。购买来的 PROM 是空白的，由使用者通过特定的方法将自己所需的信息写入其中。但只能写一次，以后再也不能改变，要是写错了，这块芯片就报废了。

(3) 紫外线可擦除的 PROM(EPROM)，这类芯片的上面有一块透明的石英玻璃，透过玻璃可以看到芯片。在一定的紫外线的照射后能将其中的内容擦除后重写，紫外线就像“消字灵”，可以把写在纸上字消掉，然后再重写。

(4) 电可擦除的 PROM(EEPROM)，这类芯片的功能和 EPROM 类似，写进去的内容可以擦掉重写，而且不需要紫外光照，只要用电学方法就可以擦除，所以它的使用要比 EPROM 方便一些。EEPROM 芯片虽然能用电的方法擦除其内容，但它仍然是一种 ROM，具有 ROM 的典型特征，断电后芯片中的内容不会丢失。

不管是 EPROM 还是 EEPROM，其可擦除的次数都是有限的。

2. 随机存取存储器

随机存取存储器又称为 RAM，其中的内容可以在工作时随机读出和存入，即允许 CPU 对其进行读、写操作。由于随机存储器的内容可以随时改写，所以它适用于存放一些变量、运算的中间结果、现场采集的数据等。但是 RAM 中的内容在断电后消失。

RAM 可以分为静态和动态的两种，单片机中一般使用静态 RAM，其容量较小，但使用比较方便。

3. 可现场改写的非易失存储器

随着半导体存储技术的发展，各种新的可现场改写信息的非易失存储器逐渐被广泛应用，且发展速度很快。主要有快擦写 Flash 存储器、新型非易失静态存储器 NVSRAM 和铁电存储器 FRAM。这些存储器的共同特点是：从原理上看，它们属于 ROM 型存储器，但是从功能上看，它们又可以随时改写信息，因而作用相当于 RAM。所以，ROM、RAM 的定义和划分已逐渐开始融合。由于这一类存储器技术发展非常迅速，存储器的性能也在不断发生变化，难以全面、客观介绍各种存储器，这里仅对单片机领域中广泛使用的快擦写存储器 Flash 作一个简介。

Flash 存储器是在 EPROM 和 EEPROM 的制造基础上产生的一种非易失存储器。其集成度高、制造成本低，既具有 SRAM 读/写的灵活性和较快的访问速度，又具有 ROM 在断电后不丢失信息的特点，所以发展迅速。Flash 存储器的擦写次数是有限的，一般在万次以上，多者可达 100 万次以上。目前，有很多单片机内部带有 Flash 存储器。

巩固与提高

1. 为什么需要用 ROM 和 RAM 来组成微机系统的存储器？
2. 什么是单片微型计算机，它与一般的微型计算机有哪些不同？
3. 存储器有哪几种类型？存放程序一般用哪种存储器？

1.5　C 语言入门

随着单片机开发技术的不断发展，目前已有越来越多的人从普遍使用汇编语言到逐渐使用高级语言开发，其中主要是以 C 语言为主，市场上几种常见的单片机均有其 C 语言开发环境。本教材使用 C 语言来编写 80C51 单片机程序，下面首先来了解一下有关 C 语言的基本知识。

1.5.1　C 语言的产生与发展

C 由早期的编程语言 BCPL(Basic Combind Programming Language)发展演变而来，1970 年美国贝尔实验室的 Ken Thompson 根据 BCPL 语言设计出 B 语言，并用 B 语言写了 UNIX 操作系统。1972 年至 1973 年间，贝尔实验室的 D. M. Ritchie 在 B 语言的基础上设计出了 C 语言。

随着微型计算机的日益普及，出现了许多 C 语言版本，由于没有统一的标准，使得这些 C 语言之间出现了一些不一致的地方。为了改变这种情况，美国国家标准研究所(ANSI)为 C 语言制定了一套 ANSI 标准，成为现行的 C 语言标准。

1.5.2　C 语言的特点

C 语言发展非常迅速，成为最受欢迎的语言之一，主要因为它具有强大的功能，归纳起来 C 语言具有下列特点：

1. 与汇编语言相比

(1) C 语言是一种高级语言，具有结构化控制语句。

结构化语言的显著特点是代码及数据的分隔化，即程序的各个部分除了必要的信息交流外彼此独立。这种结构化方式可使程序层次清晰，便于使用、维护以及调试。

(2) C 语言适用范围大，可移植性好。

和其他高级语言一样，C 语言不依赖于特定的 CPU，其源程序具有很好的可移植性。只要某种 CPU 或 MCU 有相应的 C 编译器，就能使用 C 语言进行编程。目前，主流 MCU 都有 C 编译器。对于嵌入式系统的开发者，这一点尤为重要。目前可供选用的 MCU 型号极多，这些 MCU 各有特点，开发者在做项目时往往需要选用不同品种的 MCU 以各尽其能，但要熟悉每一种 MCU 的汇编语言并能写出高质量的

程序并非易事。如果使用 C 语言编程,借助于 C 语言的可移植性,只要熟悉所用 MCU 的特性即可编程,这可节省大量的时间,将精力专注于所要解决的问题上面。

作者经常进行项目开发工作,不同客户往往会提出一些具体的要求,其中就有使用特定 MCU 的要求。实际上项目开发中往往能够重复利用一些代码,特别是算法代码。因此,作者编程时一般都会注意,将算法部分和 I/O 驱动部分分离开来编写。这样,一旦需要移植,只要改写 I/O 驱动部分就可以了,十分快捷和方便。

2. 与其他高级语言相比

(1) 简洁紧凑、灵活方便。

C 语言一共只有 32 个关键字,9 种控制语句,程序书写自由,主要用小写字母表示。它把高级语言的基本结构和语句与低级语言的实用性结合起来。

(2) 运算符丰富。

C 的运算符包含的范围很广泛,共有种 34 个运算符。C 语言把括号、赋值、强制类型转换等都作为运算符处理,从而使 C 的运算类型极其丰富、表达式类型多样化,灵活使用各种运算符可以实现其他高级语言中难以实现的运算。

(3) 数据结构丰富。

C 的数据类型有:整型、实型、字符型、数组类型、指针类型、结构体类型、共用体类型等,能用来实现各种复杂数据类型的运算。

(4) C 语言程序设计自由度大。

对数组下标越界不做检查,由编程者自己保证程序的正确;对变量的类型使用比较灵活,整型、字符型等各种变量可通用。

(5) C 语言允许直接访问物理地址,可以直接对硬件进行操作。

因此 C 语言既具有高级语言的功能,又具有低级语言的许多功能,能够像汇编语言一样对位、字节和地址进行操作。

(6) C 语言程序生成代码质量高,程序执行效率高。

用 C 语言编写的程序,编译后一般只比有丰富经验的汇编编程人员所写的汇编程序效率低 10%~20%。

1.5.3 C 语言入门知识

下面将通过一些实例介绍 C 语言编程的方法,这里采用 80C51 系列单片机的 C 编译器 Keil 软件作为开发环境,关于 Keil 软件的具体用法将在第 2 章作详细介绍。

图 1-7 所示电路图使用 STC89C52 单片机作为实验用芯片,这种单片机性属于 80C51 系列,其内部有 8 KB 的 Flash ROM,可以反复擦写,并有 ISP 功能,支持在线下载,不需要反复拨、插芯片,非常适于做实验。如图 1-7 所示 89C52 的 P1 引脚上接 8 个发光二极管,下面一些例子的任务是让接在 P1 引脚上的发光二极管按要求发光。

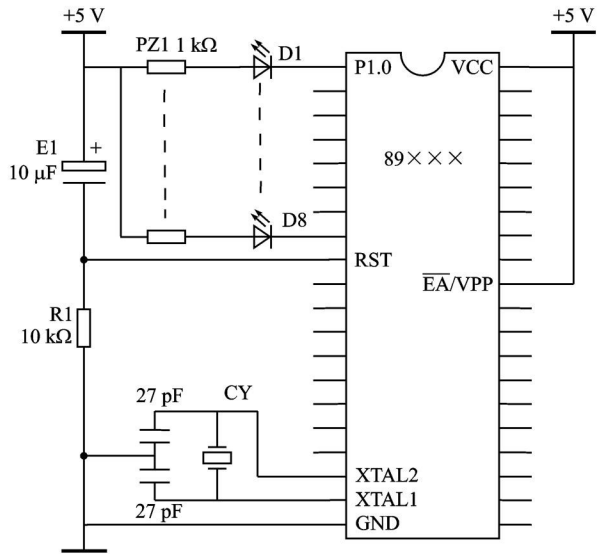


图 1-7　接有 LED 的单片机基本电路

例 1-1：让接在 P1.0 引脚上的 LED 发光。

```
#include  "reg51.h"
sbitP1_0 = P1^0;
void main()
{   P1_0 = 0;
    for(;;);
}
```

这个程序的作用是让接在 P1.0 引脚上的 LED 点亮。下面来分析一下这个 C 语言程序包含了哪些信息。

(1)“文件包含”处理

程序的第一行是一个“文件包含”处理。

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，所以看起来这个程序只有 4 行，但 C 编译器在处理这段程序时却要处理几十或几百行。这段程序中包含 REG51.h 文件的目的是为了要使用 P1 这个符号，即通知 C 编译器，程序中所写的 P1 是指 80C51 单片机的 P1 端口而不是其他。这是如何做到的呢？

打开 reg51.h 可以看到这样的一些内容：

```
/* ------------------------------------------------------------
REG51.H
Header file for generic 80C51 and 80C31 microcontroller.
Copyright(c)  1988 - 2001 Keil Elektronik GmbH and Keil Software, Inc.
```

```
All rights reserved.
-----------------------------------------------------------*/
/*   BYTE Register   */
sfr P0    = 0x80;
sfr P1    = 0x90;
sfr P2    = 0xA0;
sfr P3    = 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;
```

可以看到,sfr P1 =0x90;这样一行程序,其中 0x90 是 C 语言中十六进制数的表示方法,它表示的是地址,即 P1 这个符号与单片机内部 0x90 这个位置对应。而这个位置在设计 80C51 这块芯片时被确定为 P1 寄存器,其他如 P0、P2、P3 等符号也有类似的作用。这些符号是什么含义将在后面各章中逐一介绍,至于为什么它们与这些地址对应,这是由当初设计 80C51 的 INTEL 公司的工程师决定的,并一直沿用到现在。

定义符号使用了一个关键词:sfr。这不是标准 C 语言的关键字,而是 Keil C 编译器为了能够直接访问 80C51 中的 SFR 提供了一个新的关键词,其用法是:

```
sfr 变量名 = 地址值。
```

例如:

```
sfrP1 = 0x90;
```

即定义了 P1 这个名称与 0x90 这个地址的对应关系。通过这种方法,可以自行定义新的特殊功能寄存器(SFR)。随着技术的不断发展,新的 80C51 系列单片机层出不穷,这些新的 80C51 单片机通常会增加一些 SFR 以增强功能。通常每一种新的单片机出来之后,Keil 软件都会升级以支持这种单片机,但即便编程人员不能及时升级自己的 Keil 软件,也没有关系,可以通过 sfr 关键字自行定义 SFR 符号与其地址的对应关系,以便能使用这种新的 MCU。

例如,89S 系列单片机中增加了看门狗定时器,其数据手册上的名称为 WMCON,地址为 96H,因此,如果编程人员手边的 Keil 软件中找不到现成的头文件,那

么可以在 reg51.h 中或者程序开头增加下面的一行：

```
sfr WMCON    = 0x96;
```

这样就可以在程序中使用 WMCON 这个符号了。

(2) 符号 P1_0 用来表示 P1.0 引脚

为了使用单片机的引脚，设计 80C51 的工程师给每个引脚起了名字，如 P1.0、P1.1 等。但在 C 语言里如果直接写 P1.0，C 编译器并不能识别，而且 P1.0 也不是一个合法的 C 语言标识符，所以得给它另起一个名字，这里起的名为 P1_0，可是 P1_0 是不是就是 P1.0 呢？C 编译器可不这么认为，所以必须给它们之间建立联系，这里使用了 Keil C 新增的关键字 sbit 来定义，sbit 的用法有 3 种：

①：sbit 位变量名＝地址值；

②：sbit 位变量名＝SFR 名称^变量位地址值；

③：sbit 位变量名＝SFR 地址值^变量位地址值。

如定义 PSW 中的 OV 可以用以下 3 种方法：

```
① sbit OV = 0xd2      //0xd2 是 OV 的位地址值
② sbit OV = PSW^2     //PSW 必须先用 sfr 定义好
③ sbit OV = 0xD0^2    //0xD0 就是 PSW 的地址值
```

因此这里用：

```
sbit P1_0 = P1^0;
```

就是用符号 P1_0 来表示 P1.0 引脚，如果愿意也可以用 P10 之类的名字，只要程序中所有 P1_0 都要随之更改就行了。

Keil 软件在 AT89X52.H 中已定义了各引脚的变量，如果包含了这个文件，就不需要自己定义了，这个文件在 keil\c51\inc\atmel 文件夹下。

以下是 AT89X52.H 的有关内容：

```
/* ------------------------------------------------------------
AT89X52.H

Header file for the low voltage Flash Atmel AT89C52 and AT89LV52.
Copyright(c)  1988 - 2002 Keil Elektronik GmbH and Keil Software, Inc.
All rights reserved.
------------------------------------------------------------*/

#ifndef __AT89X52_H__
#define __AT89X52_H__

/* ------------------------------
Byte Registers
```

```
--------------------------------*/
    ……与上述头文件相同

/* --------------------------------
P0 Bit Registers
--------------------------------*/
sbit P0_0 = 0x80;
sbit P0_1 = 0x81;
sbit P0_2 = 0x82;
sbit P0_3 = 0x83;
sbit P0_4 = 0x84;
sbit P0_5 = 0x85;
sbit P0_6 = 0x86;
sbit P0_7 = 0x87;
这是有关 P0 引脚的定义,其他定义可自行打开该文件查看。
⋮
#endif
```

(3) 主函数 main

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。

通过上面的分析了解到部分 C 语言的特性,下面再看一个稍复杂一点的例子。

例 1-2: 让接在 P1.0 引脚上的 LED 闪烁发光

```
#include "reg51.h"
#define uchar unsigned char
#define uint  unsigned int
sbitP1_0 = P1^0;

/* 延时程序,由 Delay 参数确定延迟时间 */
void mDelay(uint Delay)
{  uint i;
   for(;Delay>0;Delay--)
   {  for(i=0;i<76;i++)
      {;}
   }
}
void main()
{  for(;;)
   {  P1_0 = !P1_0;          //取反 P1.0 引脚
      mDelay(1000);
   }
}
```

程序分析：main 函数的第一行暂且不看，第二行是“P1_0＝!P1_0;”，在 P1_0 前有一个符号“!”，符号“!”是 C 语言的一个运算符，就像数学中的“＋”、“－”一样，是做一种运算的符号，它所做的运算是“取反”，即将该符号后面的那个变量的值取反。如果原来 P1.0 是低电平(LED 亮)，那么取反后，P1.0 就是高电平(LED 灭)，反之，如果 P1.0 是高电平，取反后，P1.0 就是低电平，这条指令被反复地执行，接在 P1.0 上灯就会不断“亮”、“灭”。

main()函数中第三行程序是：mDelay(1000);，这行程序的用途是延时 1 s，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，亮、灭之间的间隔时间非常短，人眼根本无法分辨。

这里 mDelay(1000)并不是由 Keil C 提供的库函数，如果在编写其他程序时写上这么一行，会发现编译通不过。那么这里为什么又能正确编译呢？注意观察，可以发现这个程序中有 void mDelay(…)开始的一段程序行，可见，mDelay 是编程者自己起的名字，并且为此编写了一些程序行，如果程序中没有这么一段程序行，那就不能使用 mDelay(1000)了。那可不可以把这段程序复制到其他程序中，然后就可以在那个程序中用 mDelay(1000)了呢？回答是：那当然就可以了。还有一点需要说明，mDelay 这个名称是由编程者自己命名的，可自行更改，但一旦更改了名称，main()函数中的名字也要作相应的更改。

mDelay 后面有一个小括号，小括号里有数据(1000)，该 1000 被称之“参数”，用它可在一定范围内调整延时时间的长短，这里用 1 000 来要求延时时间为 1 000 ms。要做到这一点，必须由自己编写 mDelay 程序来决定，具体的编写方法将在以后介绍。

这里的两行程序：

```
P1_0 = ! P1_0;            //取反 P1.0 引脚
mDelay(1000);
```

会被反复执行的关键就在于 main 函数中的第一行程序：for(;;)，这里不对此作详细的介绍，读者暂时只要知道，这行程序连同其后的一对大括号“{}”构成了一个无限循环语句，一旦程序开始运行，该大括号内的语句会被反复执行，直到断电为止。

1.5.4　C 程序特性分析

通过上述的几个例子，可以得出一些结论：

(1) C 程序是由函数构成的。一个 C 源程序至少包括一个函数，一个 C 源程序有且只有一个名为 main()的函数，也可能包含其他函数，且一个实用程序中通常都有大量的函数，函数是 C 程序的基本单位。main 函数通过直接书写语句和调用其他函数来实现有关功能，这些其他函数可以是由 C 语言本身提供的(这样的函数称之为库函数)，也可以是用户自己编写的(这样的函数称之为用户自定义函数)。库函数

与用户自定义函数的区别在于,使用 Keil C 语言编写的任何程序,都可以直接调用 C 的库函数,调用时只需要包含具有该函数说明的相应的头文件即可,Keil C 提供了 100 多个库函数供用户直接使用;自定义函数则是完全个性化的,是用户根据自己需要而编写的有关代码。

(2) 一个 C 语言程序,总是从 main 函数开始执行的,而不管物理位置上这个 main()放在什么地方,例 1—2 中就是放在了最后。

(3) 程序中的 mDelay 如果写成 mdelay 就会编译出错,即 C 语言区分大小写。

(4) C 语言书写的格式自由,可以在一行写多个语句,也可以把一个语句写在多行。没有行号(但可以有标号),书写的缩进没有要求。但是建议读者按一定的规范来写,可以给自己带来方便。

(5) 每个语句和定义的最后必须有一个分号,分号是 C 语句的必要组成部分。

(6) 可以用/*.....*/的形式为 C 程序的任何一部份作注释,在"/*"开始后,一直到"*/"为止的中间的任何内容都被认为是注释,所以在书写特别是修改源程序时要注意,如果无意之中删掉一个"*/",结果,从这里开始一直要遇到下一个"*/"中的全部内容都被认为是注释了。

Keil C 也支持 C++风格的注释,就是用"//"引导的后面的语句是注释,例:

```
P1_0 = ! P1_0            //取反 P1.0
```

这种风格的注释,只对本行有效,不会出现上述问题,而且书写比较方便,所以在只需要一行注释的时候,往往采用这种格式。本书的源程序中,这两种注释的方式都会出现。

1.6 C 语言中的数据

数据是计算机处理的对象,计算机要处理的一切内容最终将以数据的形式出现,因此,程序设计中的数据有着很多种不同的含义,它们往往以不同的形式表现出来;而且这些数据在计算机内部进行处理、存储时有着很大的区别,所以本节介绍 C 语言数据类型的有关知识。

1.6.1 数据类型概述

C 语言常用的数据类型有整型、字符型、实型等。

C 语言中数据有常量与变量之分,它们分别属于以上这些类型。由以上这此数据类型还可以构成更复杂的数据结构,在程序中用到的所有的数据都必须为其指定类型。图 1-8 列出了 C 语言的数据类型。

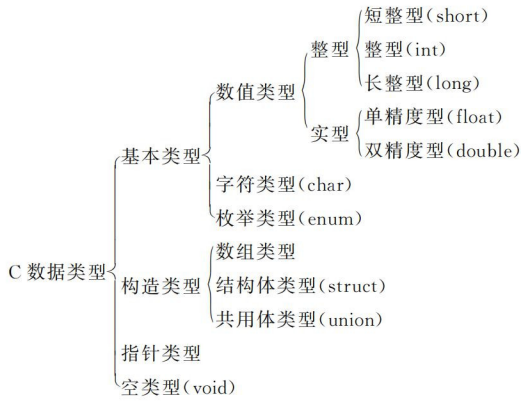


图 1-8　C 语言的数据类型

1.6.2　常量与变量

在程序运行过程中，其值不能被改变的量称为“常量”；在程序运行中，其值可以改变的量称之为“变量”。

使用常量时可以直接给出常量的值，如 3、5、0xfe 等，也可以用一些符号来替代常量的值，这称之为“符号常量”。

使用符号常量的好处是：

(1) 含义清楚。在书写程序时，有一些量是具有特定含义的，如某单片机系统扩展了一些外部芯片，每一块芯片的地址即可用符号常量定义，如：

```
#define PORTA 0x7fff
#define PORTB 0x7ffe
```

程序中可以用 PORTA、PORTB 来对端口进行操作，而不必写 0x7fff、0x7ffe。显然，这两个符号比两个数字更能令人明白其含义。在给符号常量起名字时，尽量要做到“见名知意”以充分发挥这一特点。

(2) 在需要改变一个常量时能做到“一改全改”。如果由于某种原因，端口的地址发生了变化(如修改了硬件)，由 0x7fff 改成了 0x3fff，那么只要将所定义的语句改动一下即可：

```
#define PORTA 0x3fff
```

不仅方便，而且能避免出错。设想一下，如果不用符号常量，要在成百上千行程序中把所有表示端口地址的 0x7fff 找出来并改掉可不是件容易的事，特别是如果程序中还有其他量正好也是 0x7fff 时，极易引起混淆而产生错误。

变量就是在运行中能够发生变化的量。

例 1-2 中用到了延时程序，其中 main 函数中调用延时程序时这么写：

```
mDelay(1000);
```

这其中括号中参数 1 000 决定了延时时间,也就决字了灯流动的速度。这个 1 000 是常量,在编写程序时确定了,在程序编译、链接产生目标代码并将目标代码写入芯片后,这个数据不能在应用现场被修改。如果使用中有人提出希望改变流水灯的速度,那么只能重新编程、编译、产生目标代码,再将目标代码写入芯片才能更改。

如果在现场有修改流水灯速度的要求,括号中就不能写入一个常数。为此可以定义一个变量(如 Speed),main 函数据中应该这么写:

```
mDelay(Speed);
```

然后再编写一段程序,使得 Speed 的值可以通过按键被修改,这样,流水灯的速度就可以在现场修改了,在这个应用中,就需要用到变量了。

每个变量都有变量名和变量值两个属性,编程时通过变量名来使用变量的值。如上述例子中 Speed 就是变量名,执行 mDelay(Speed);语句时使用的是其数值。变量必须保存在单片机内部的随机存取存储器(RAM)中,这样才能实现在运行中随时发生变化。

在 C 语言中,要求对所有用到的变量作强制定义,也就是“先定义,后使用”。

1.6.3 整型数据

C 语言中使用整型常量,也使用整型变量。

1. 整型常量

整型常量即整常数。C 整常数可用以下 3 种形式表示:

① 十进制整数。如 100,−200,9 等。

② 八进制整数。用数字“0”开头的数是八进制数。如 0224 表示八进制数 224,即$(224)_8$,其值为 $2\times8^2+2\times8^1+4\times8^0=128+16+4=148$。−023 表示八进制数−23,即$-(23)_8$,相当于十进制的−19。

③ 十六进制整数。以数字“0”和字母“x”开头的数是十六进制数,如 0x224,即$(224)_{16}$,其值为 $2\times16^2+2\times16^1+4\times16^0=512+32+4=548$。−0x23 表示十六进制数−23,即$-(23)_{16}$,相当于十进制的−35。

2. 整型变量

整型变量的基本类型是 int,可以加上有关数值范围的修饰符。这些修饰符分两类,一类是 short 和 long,另一类是 unsigned,这两类修饰符可以同时使用。

在 int 前加上 short 或 long 用来表示数的范围。对于 Keil C 来说,加 short 和不加 short 是一模一样的(在有一些 C 语言编译系统中是不一样的),所以,short 就不讨论了。如果在 int 前加上 long 的修饰符,那么这个数就被称之为“长整数”,在 Keil C 中,长整数用 4 个字节来存放,而基本 int 型用 2 个字节。显然,长整数所能表达的

范围比整数要大，一个长整数表达的范围为：$-2^{31} \leqslant x \leqslant 2^{31}-1$，即：$-2\ 147\ 483\ 648 \leqslant x \leqslant 2\ 147\ 483\ 647$。

而不加 long 修饰的 int 型数据的范围是－32 768～32 767。

第二类修饰符是 unsigned，即无符号的意思。加上这个修饰符，说明其后的数是一个无符号的数。无符号数、有符号数的区别在于数的范围不一样。对于 unsigned int 而言，仍是用 2 个字节（16 位）表示一个数，但其数的范围是 0～65 535，对于 unsigned long int 而言，仍是用 4 个字节（32 位）表示一个数，但其数的范围是 0～$2^{32}-1$。

下面以 Keil C 为例，将整型数据做个总结，见表 1－4。

表 1－4　整型变量的数据类型

数　系	符　号	说　明	字节数/byte	数据长度/bit	表示形式	数值范围
整数型	带符号	基本型	2	16	int	－32 768～＋32 767
		短整型	2	16	short	－32 768～＋32 767
		长整型	4	32	long	－2 147 483 648～＋2 147 483 647
	无符号	基本型	2	16	unsigned int	0～65 535
		短整型	2	16	unsigned short	0～65 535
		长整型	4	32	unsigned long	0～4 294 967 295

C 语言中的变量均需先定义，后使用，定义整型变量的方法是：

修饰符 变量名

定义整型变量用的修饰符是 int，可以在其前面加上表示长度和符号的修饰符。

例：

```
int  a,b;                    /*定义两个整型变量 a 和 b*/
long    a1,b1;               /*定义两个长整型变量 a1 和 b1*/
unsigned    int  x;          /*定义无符号的整型变量 x*/
unsigned    long  int x1;    /*定义无符号的长整型变量 x1*/
```

1.6.4　字符型数据

C 语言中有字符常量，也有字符型变量。

1. 字符常量

C 语言中的字符常量是用单引号括起来的一个字符。如‘a’、‘x’、‘1’等都是字符常量，注意，‘a’和‘A’不是同一个字符。

查看 ASCⅡ字符表，可以发现，有一些字符没有“形状”，如换行（ASCⅡ值为 10）、回车（ASCⅡ值为 13）等，有一些虽有“形状”，却无法从键盘上输入，如 ASCⅡ值

大于 127 的一些字符等,还有一些字符在 C 语言中有特殊用途,无法直接输入,如单引号“’”用于界定字符常量,但其本身就没法用这种方法来表示了。如果 C 语言的程序中要用到这一类字符,可以用 C 语言提供的一种特殊形式进行输入,就是用一个“\”开头的字符序列来表示字符。如用“\r”来表示回车,用“\n”来表示回车。

常用的以“\”开头的特殊字符见表 1-5。

表 1-5 转义字符及其含义

字符形式	含 义	ASCⅡ字符(十进制)
\n	换行,将当前位置移到下一行开头	10
\t	水平制表(跳到下一下 TAB 位置)	9
\b	退格,将当前位置移到前一列	8
\r	回车,将当前位置移到本行开头	13
\f	换页,将当前位置移到下页开头	12
\\	反斜杠字符“\”	92
\’	单引号字符	39
\”	双引号字符	34

除此之外,C 语言还规定,用反斜杠后面带上八进制或十六进制数字直接表示该数值的 ASCⅡ码,这样,不论什么字符,只要知道了其 ASCⅡ码,就可以在程序中用文本书写的方式表达出来了。

例如:可以用“\101”表示 ASCII 码八进制数为 101(即十进制数 65)的字符“A”。而“\012”表示八进制的字符 012(即十进制数 10)的字符换行(\n)。用\376 表示图形字符“■”。

2. 字符变量

字符型变量用来存放字符常量,一个变量只能存放一个字符。

字符变量的定义形式如下:

修饰符 变量名

字符型变量定义的修饰符是 char。例:

```
char c1,c2;
```

它表示 c1 和 c2 为字符型变量,各可以放一个字符。可以使用下面的语句对其进行赋值:

```
c1 = 'a';
c2 = 'b';
```

将一个字符型常量放到一个字符型变量中,实际上是将该字符的 ASCⅡ码放到存储单元中。如:

```
char c = 'a';
```

定义一个字符型的变量 c，然后将字符 a 赋给该变量。进行这一操作时，将字符“a”的 ASCⅡ码值赋给变量 c，因此，完成后 c 的值是 97。

既然字符最终也是以数值来存储的，那么和以下的语句：

```
int  i = 97;
```

究竟有多大的区别呢？对于使用这个值的对象来说，它们没有区别；只是它们在内存中的存储方式来说是有区别，c 在内存中用一个字节保存，而 i 在内存中需要 2 个字节保存。如果确切地知道某变量小于 255，那么不论用 i 还是用 c 来保存这个变量，使用效果都是一样的。C 语言字符型数据作这样的处理使用得程序设计时增大了自由度。

由于 51 系列单片机是 8 位机，做 16 位数的运算要比做 8 位数的运算慢很多，因此在使用单片机的 C 语言程序设计中，只要预知其值的范围不会超过 8 位所能表示的范围，就用 char 型数据来表示。

字符型变量只有一个修饰符 unsigned，即无符号的。对于一个字符型变量来说，其表达的范围是－128～＋127，而加上了 unsigned 后，其表达的范围变为 0～255。

使用 Keil C 编写程序时，不论是 char 型还是 int 型，要尽可能采用 unsigned 型数据，因为在处理有符号的数时，程序要对符号进行判断和处理，运算的速度会减慢；单片机工作于实时状态，任何提高效率的方法都要考虑。