高等职业技术教育机电类专业规划教材

单片机原理及其应用

高等职业技术教育机电类专业教材编委会 组编
陈立周 陈 宇 编

机械工业出版社
China Machine Press

高等职业技术教育机电类专业规划教材

单片机原理及其应用

高等职业技术教育机电类专业教材编委会　组编
陈立周　陈　宇　编

机械工业出版社

本书是参照高等专科学校机电类专业的教学计划和有关微型计算机原理及应用的教学大纲，以及机电类高等职业技术教育对主干课程的要求，为高等职业技术教育电气工程类专业的教学需要而编写的教材。内容包括微型电子计算机系统的基础知识、8051 系列单片机的结构、MCS-51 指令系统、单片机并行 I/O 接口、串行 I/O 接口、定时/计数器、存储器的扩展、单片机的开发与调试等。为适应高等职业技术教育电气工程类专业的需要，本书还增加了单片机 C 语言编程和汇编、编译流行软件 A51、C51 的使用。书中既注意讲授单片机系统的基础知识，又选用了大量的应用实例，以便于读者理解与应用。

本书可作为高等职业技术教育机电类专业微机课程的教材，也适合高等专科学校和成人高等教育使用，另外还可供机电工程技术人员的学习参考。

图书在版编目（CIP）数据

单片机原理及其应用/陈立周，陈宇编 .—北京：机械工业出版社，2000.12

高等职业技术教育机电类专业规划教材

ISBN 7-111-08197-8

Ⅰ.单… Ⅱ.①陈…②陈… Ⅲ.微处理机-高等学校：技术学校-教材 Ⅳ.TP368.1

中国版本图书馆 CIP 数据核字（2000）第 66450 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）

责任编辑：贡克勤 版式设计：霍永明 责任校对：张晓蓉

封面设计：李雨桥 责任印制：郭景龙

北京京丰印刷厂印刷·新华书店北京发行所发行

2001 年 1 月第 1 版·第 1 次印刷

1000mm×1400mm B5·6.75 印张·257 千字

0 001－4 000 册

定价：16.00 元

高等职业技术教育机电类专业教材编委会

序

职业教育指受教育者获得某种职业或生产劳动的职业道德、知识和技能的教育。机电行业的职业技术教育是培养在生产一线的技术、管理和运行人员，他们主要从事成熟的技术和管理规范的应用与运作。随着社会经济的发展和科学技术的进步，生产领域的技术含量在不断提高。用人单位要求生产一线的技术、管理和运行人员的知识与能力结构与之适应。行业发展的要求促使职业技术教育的高层次——高等职业教育蓬勃成长。

高职教育与高等工程专科、中专教育培养的人才属同一类型，都是技术型人才，毕业生将就业于技术含量不同的用人单位。高等职业教育的专业设置必须适应地区经济与行业的需求。高等职业教育是能力本位教育，应以职业分析入手，按岗位群职业能力来确定课程设置与各种活动。

机械工业出版社出版了大量的本科、高工专、中专教材，其中有相当一批教材符合高等职业教育的需求，具有很强的职业教育特色，在此基础上这次又推出了机械类、电气类、数控类三个高职专业的高职教材。

专门课程的开发应遵循适当综合化与适当实施化。综合化有利于破除原来各种课程的学科化倾向，删除与岗位群职业能力关系不大的内容，有利于删除一些陈旧的内容，增添与岗位群能力所需要的新技术、新知识，如微电子技术、计算机技术等。实施化是课程内容要按培养工艺实施与运行人员的职业能力来阐述，将必要的知识支撑点溶于能力培养的过程中，注重实践性教学，注重探索教学模式以达到满意的教学效果。

本教材倾注了众多编写人员的心血，他们为探索我国机电行业高职教育作出可贵的尝试。今后还要依靠广大教师在实践中不断改进，不断完善，为创建我国的职业技术教育体系而奋斗。

赵克松

前　言

本书是参照高等专科学校机电类专业的教学计划和有关微型计算机原理及应用的教学大纲，以及机电类高等职业技术教育对主干课程的要求，为高职电气类专业的教学需要而编写的教材。由于微型电子计算机技术的发展以及单片机的广泛应用，单片机的技术已经成为电气类专业技术人员的基础知识之一。为此把加强硬件与软件的基础知识，掌握单片机的实际应用，作为本书的出发点。

本书由福建高级工业专门学校陈立周副教授担任主编并编写 1～7 章，福建高级工业专门学校陈宇老师编写 8～9 章，全书由福建高级工业专门学校雷伍老师作了详细的审阅和校正，并根据他多年从事单片机开发的经验，为本书提出许多宝贵意见，在此表示深切的感谢。

对于书中存在的问题，敬请使用本书的老师与同学以及广大读者给予批评指正。

编　者

2000 年 6 月

目　录

第一章　单片机的基础知识

第一节　不同进位计数制及其互换

电子计算机包括单片机都是一种处理数据信息的机器，因此在学习计算机之前有必要先了解一下有关数的知识。

在人们日常生活中，都习惯于使用十进制，但在数字电路和电子计算机内部，由于只能通过电位高低表示 1 和 0 两个数码，所以计算机中不用十进制而用二进制。这样，在使用计算机的时候，就存在常用的十进制与计算机中使用的二进制进行互换的问题，又由于用二进制表示一个数，所用的数码长，不但书写和阅读不方便，而且容易出错，所以书写时又常把二进制转换为十六进制，在开始学习微型电子计算机的时候，首先要熟练掌握这三种进位计数制的互换。

任何一种进位计数制，其位数多少决定了所表示的数的大小，位数越多，所表示的数值也越大，考虑到本书所讲的单片机是一种字长为 8 位的计算机，所以下面讨论二进制时都取 8 位，当然并不是说凡二进制都是 8 位，而是本书常用的二进制数是 8 位。

一、二进制与十六进制数的互换

一个 8 位二进制数，可以写成 2 位的十六进制数。所以两种进位制的数进行互换时，可以把每 4 位的二进制数划为一组，然后对每一组进行相应的变换，例如二进制转换为十六进制可写成

0110　　1110

6　　E

把十六进制转换为二进制可写成

4　　B

0100　　1011

为了不使二进制、十六进制或十进制数相混淆，规定在二进制数的后面加上符号 B 如 0110 0110B；在十六进制数的后面加上符号 H，如 6EH、4BH 等；也可以不在后面加 H，而在前面加 $ 或 0X，如 $ 5E、$ 4B、0X5E、0X4B 等等。如果数的前后都没有符号，按习惯就认为是十进制数。

如果被转换的是一个小数，则分组时应以小数点为准，整数从小数点开始，从右向左每 4 位划为一组，小数部分则从小数点开始，从左向右也以 4 位为一

组，如果最后一组不足4位，可以用零补齐。以数1101100.11011B为例，数的前后都要补0，即

0110	1100	1101	1000
6	C	D	8

二、二进制与十进制数的互换

对于二进制整数，各位数的位权可以用底数为2的n－1次幂来确定，n表示该数的位数，即第1位的位权为$2^0=1$，第2位的位权为$2^1=1$，……。若已知一个二进制数为1010101B，对应的十进制值应为170，计算过程如下：

$$10101010B=1\times2^7+0\times2^6+1\times2^5+0\times2^4+1\times2^3+0\times2^2+1\times2^1+0\times2^0=170$$

对于二进制小数，其小数点以后各位的位权，可以用底数为2的负n次幂来确定，n同样表示位数，即从小数点向右算起，第1位的位权为$2^{-1}=0.5$，第2位的位权为$2^{-2}=0.25$，……。例如求11001100.00110011B的十进制值，则

$$\begin{aligned}11001100.00110011B&=1\times2^7+1\times2^6+1\times2^3+1\times2^2+1\times2^{-3}\\&\quad+1\times2^{-4}+1\times2^{-7}+1\times2^{-8}\\&=204.19921875\end{aligned}$$

反过来，要将十进制整数转换为二进制数，可以采用逐次除以2余数反序排列的方法，所谓反序排列，指第1次除以2的余数排在最低位。以十进制数25为例，逐次除以2列式如下：

$25\div2=12$ …… 余1

$12\div2=6$ …… 余0

$6\div2=3$ …… 余0

$3\div2=1$ …… 余1

$1\div2=0$ …… 余1

由于8位微型计算机习惯将二进制数写成8位，可得

$$25=00011001B$$

如果二进制数不超过8位，即十进制数不超过255，也可以不必列式，直接用口算转换。

要把十进制小数转换为二进制数，可以采用小数部分逐次乘2，每次乘积若产生整数则将整数个位（即所为溢出位）按正序排列，小数部分继续乘2。以33.6875为例，其整数部分

$$33=00100001B$$

其小数部分

$0.6875\times2=1.375$ …… 溢出数为1

$0.375\times2=0.75$ …… 溢出数为0

$0.75\times2=1.5$ …… 溢出数为1

$0.5 \times 2 = 1$ …… 溢出数为 1

可得出

$$33.6875 = 00100001.10110000B$$

三、十进制与十六进制数的互换

我们已经掌握了十进制与二进制的互换以及二进制与十六进制的互换。因此要把十进制数转换为十六进制数，可以先转换成二进制数，再改写成十六进制数。反之，十六进制数也可以先改成二进制数，再转换成十进制数。

十六进制数要直接转换为十进制数也可以按各位的位权求出该数对应的十进制数值。整数部分的位权等于底数为 16 的 n－1 次幂，（n 为位数），即第 1 位的位权为 $16^0 = 1$，第 2 位的位权为 $16^1 = 16$，依次类推。例如十六进制的数为 8A71H，可求出

$$8A71H = 8 \times 16^3 + 10 \times 16^2 + 7 \times 16^1 + 1 \times 16^0 = 35441$$

对于十六进制的小数，其小数点以后各位的位权，同样可以用底数为 16 的负 n 次幂来确定，n 表示位数，即从小数点向右算起，第 1 位的位权为 $16^{-1} = 0.0625$，第 2 位的位权为 $16^{-2} = 0.00390625$，……。例如某个小数为 0.4AC9H，转换为十进制值的计算过程为

$$0.4AC9H = 4 \times 16^{-1} + 10 \times 16^{-2} + 12 \times 16^{-3} + 9 \times 16^{-4} = 0.2921295$$

反过来，要把十进制转换为十六进制数，其方法与十进制数转换为二进制相似，即整数部分采用逐次除以 16 余数反序排列的方法。小数部分则采用逐次乘 16 溢出数正序排列的方法。例如将 13562 十进制数转换为十六进制：

$13562 \div 16 = 847$ …… 余 10（记作 0AH）

$847 \div 16 = 52$ …… 余 15（记作 0FH）

$52 \div 16 = 3$ …… 余 4

$3 \div 16 = 0$ …… 余 3

可得

$$13562 = 34FAH$$

在书写十六进制数时，若打头的数为 A～F，则应在 A～F 之前再加一个 0，以表示这是一个数而不是其他符号。

十进制小数转换为十六进制小数，同样采用小数部分逐次乘 16，每次乘积若产生整数，则将所得（即所谓溢出位）按正序排列，例如十进制小数 0.359375 转换为十六进制数

$0.359375 \times 16 = 5.75$ …… 溢出数为 5

$0.75 \times 16 = 12.0$ …… 溢出数为 C

可得

$$0.359375 = 0.5CH$$

第二节　带符号的二进制数

一、带符号二进制数与不带符号二进制数的区别

在数学运算中，表示一个数的正负，可以在数的前面冠以正号或负号。但计算机只能辨认0或1两个数码，不能辨认其他符号，因此在字长为8位的二进制数中，将最高位规定为符号位，最高位为0，表示该数为正，最高位为1，表示该数为负。例如数01101111B表示+1101111B，而数11101111B，则表示-1101111B。这种将最高位定为符号位的二进制数，称为带符号的二进制数。对于8位字长的带符号二进制数来说，表示数值的仅有7位，所能表示的范围为+127～-127。

应该注意，同样一个二进制数，它既可以是无符号数，也可以是带符号数。例如二进制数11001100B，既可以是无符号数204，也可以是带符号数-76，到底它是204还是-76，取决于事先约定，仅从数的本身无法判别它属于什么数。

例如下面要介绍的相对转移指令中的偏移量，约定必须使用带符号数，因此出现在偏移量中的二进制数，总是一个带符号数，在填写偏移量时，应使用带符号数，而程序中的地址值，约定为无符号数，对应使用无符号数。

总之，一个数是带符号的还是无符号的二进制数，从数的本身是无法区别的，只能根据它出现的场合，以及该场合约定使用什么数才能区分它们。

二、带符号数的表示方法

上面讲过，一个带符号数，它的最高位是符号位，其余表示数值。这种表示方式称为原码，实际上，带符号的二进制数，除了原码表示法之外，还有反码与补码，下面分别加以介绍。

（一）原码（True form）

用原码表示一个带符号二进制数，其最高位为符号位，其余表示数值，例如

$$x = +1010101B \quad [x]_{\text{true form}} = 01010101B$$

$$x = -1010101B \quad [x]_{\text{true form}} = 11010101B$$

式中，x为真值；$[x]_{\text{true form}}$表示真值x的原码，对于数0，其原码可以表示为

$$[+0]_{\text{true form}} = 00000000B$$

$$[-0]_{\text{true form}} = 10000000B$$

（二）反码（One’s complement）

反码也是带符号数的一种表示法，它同样规定最高位为符号位，其余则要看是正数还是负数。对于正数，其余各位表示数值，也就是正数的反码与原码相同。对于负数，其余各位应将1换成0，将0换成1，即所谓将逐位取反，例如

$$x = +1010101B \quad [x]_{o.c} = 01010101B$$

$$x = -1010101B \quad [x]_{o.c} = 10101010B$$

式中 $[x]_{o.c}$表示真值 x 的反码。对于数 0 的反码，也有两种形式

$$[+0]_{o.c} = 00000000$$

$$[-0]_{o.c} = 11111111$$

（三）补码（Two’s complement）

补码仍然把最高位定为符号位，对于正数，其余各位表示数值，可见正数的补码、反码与原码完全相同，对于负数，则其余各位逐位取反后再加 1，简称为取反加 1，例如：

$$x = +1010101B \quad [x]_{T.C} = 01010101B$$

$$x = -1010101B \quad [x]_{T.C} = 10101011B$$

式中，$[x]_{T.C}$为真值 x 的补码。

补码这个概念与某个具体记数器的最大容量有关，以常用的 8 位二进制数为例，扣除最高位作为符号位外，记数的最大容量为 7 位数。当计数器计至 128 时，最高位产生溢出，又由于计数器只有 7 位，溢出的数就被丢弃。这个被丢弃的值就是最大容量值，又称为模，用符号 Mod 表示。对于模等于 128 的 7 位计数器，0 与 128 的数码相同，或者说 0 与 128 等价。即

$$0 = 00000000B$$

$$128 = \boxed{1}0000000B$$

框中的 1 就是被丢弃的数。可见，若模为 M，则 a 与 $a + M$ 等价，例如模为 128 时，1 与 129 等价，2 与 130 等价。因为不计及溢出数，所以计数器内的示值是相同的，即

$$a = a + M \text{（Mod 为 } M\text{）}$$

再把这个概念推广到负数领域，例如 $a = (-2)$，代入上式有

$$(-2) = (-2) + 128 = 126 \quad (M \text{ 为 } 128)$$

即模为 128 时，（－2）与 126 等价，或者说这两个数互为补数，同样，可推出（－3）的补码为 125，（－4）的补码为 124。

为了进一步理解这个概念，可以用时钟做为例子，时钟的钟面最大示数为 12，时钟走到 12 点就等于 0 点，数 12 就被自动丢弃。可见时钟的钟面是一个模为 12 的计数器。时钟的时针向前拨 3 个字（$a = 3$），跟向前拨 15 个字（$a + M = 3 + 12 = 15$）都停在同一位置上，也就是说 +3 与 +15 等价。

如果将时钟向后拨定义为负数，那么时钟向后拨 3 个字（$a = (-3)$），跟时针向前拨 9 个字（$a = a + M = (-3) + 12 = 9$）都停在相同位置上，也就是 −3 与 9 等价。因为对于模为 12 的钟面来讲，−3 可以用其补码 9 来表示。

要求得一个数的补码，可以用公式 $a = a + M$ (Mod = M)。也可以采用正数补码等于原码，负数补码为取反加 1 的方法求得（注意：取反加 1 不包括符号位)。反过来，要从补码求原码，同样用取反加 1，表 1-1 是 8 位带符号二进制数的原码、反码、补码对照表，注意表中的

$$[+0]_{T.C} = 00000000B$$

$$[-0]_{T.C} = 00000000B$$

而 10000000B 不是（-0）的补码，而是（-128）的补码。

表 1-1　二进制数原码、反码、补码对照表

十进制数	二进制数	原码	反码	补码
+0	+0000000	0000000	00000000	00000000
+1	+0000001	00000001	00000001	00000001
+2	+0000010	00000010	00000010	00000010
⋮	⋮	⋮	⋮	⋮
+126	+1111110	01111110	01111110	01111110
+127	+1111111	01111111	01111111	01111111
-0	-0000000	10000000	11111111	00000000
-1	-0000001	10000001	11111110	11111111
-2	-0000010	10000010	11111101	11111110
⋮	⋮	⋮	⋮	⋮
-126	-1111110	11111110	10000001	10000010
-127	-1111111	11111111	10000000	10000001
-128	-10000000	无法表示	无法表示	10000000

三、带符号二进制数的运算

二进制数和十进制数的运算规则基本相同，所不同的仅仅是前者逢 2 进 1，后者逢 10 进 1，借位时，从高位借 1 到低位，前者当 2，后者当 10。但如果是带符号的二进制数，则情况比较复杂，因为带符号二进制数有三种表示方法，其中反码用得较少，下面主要介绍原码与补码运算中应注意的问题。

原码运算时，首先要把符号与数值分开。例如两数相加，先要判断两数的符号，如果同号，可以做加法，如果异号，实际要做减法，减后的差作为两数之和，和数的符号与绝对值较大的数的符号相同。两数相减也是一样，也要先判断符号，然后决定是相加还是相减，还要根据两数的大小与符号决定两数之差的符号。

补码运算不存在符号与数值分开的问题，而且加法运算就一定是相加，减法运算就一定是相减，因此在计算机中对带符号的数进行加减时，最好使用补码。

设有 x、y 两个数，用补码表示如下

$x = 10011111B$　（－97 的补码）

$y = 00001000B$　（＋8 的补码）

若求 $x + y$ 之和，可不用考虑两数的符号，直接相加，得出的和为 $x + y =$ 10100111B（－89 的补码），可见直接相加结果必定是正确的。

若求 $x - y$ 之差，也可以直接相减，即

$x = 10011111B$　（－97 的补码）

$-y = 00001000B$　（＋8 的补码）

$x - y = 10010111B$　（－105 的补码）

若求 $y - x$ 之差，同样也用减法即

$y = 00001000B$　（＋8 的补码）

$-x = 10011111B$　（－97 的补码）

$y - x = \boxed{1}01101001B$　（＋105 的补码）

也就是说做减法时，不论两数符号如何，其相减结果不论是数值还是符号都将是正确的。

在上述 $y - x$ 算式中，最高位发生进位只是因为在字长为 8 位的计算机中，若运算结果并未超出补码的记数容量（－128 ~ ＋127），这时的进位被视为自然丢弃。计算机在运算中，这种自然丢弃并不影响结果的正确。

但要注意，如果得数超过 8 位补码所允许的表示范围（即超出 ＋127 ~ －128），则其进位称之为溢出。溢出与自然丢弃是两种不同的概念。判别属于哪一种，则要看第 7 位与第 8 位的进位情况，如果第 7 位和第 8 位同时产生进位，即所谓双进位，则这种进位属于允许的自然丢弃。如果只有第 7 位或者只有第 8 位产生进位，即只有单进位，则这种进位属于溢出，溢出表示其数值超出计算机字长所能表示的范围，运算结果必然是错误的，因而也是不允许的。

例 1-1　求下列两组 x、y 之和。

$x = +1100100B$	$[x]_{T.C} = 01100101B$	$[+100]_{T.C}$
$y = +1000011B$	$+ [y]_{T.C} = 01000011B$	$+ [+67]_{T.C}$
	$[x+y]_{T.C} = 10100111B$	$[-89]_{T.C}$
$x = -1111000B$	$[x]_{T.C} = 10001000B$	$[-120]_{T.C}$
$y = -0011000B$	$+ [y]_{T.C} = 11101000B$	$+ [-24]_{T.C}$
	$[x+y]_{T.C} = 10111000B$	$[+112]_{T.C}$

例中第 1 组，只有第 7 位产生进位，第 8 位没有进位。在第 2 组中只有第 8 位产生进位，第 7 位没有进位。都是单进位，都属于溢出，运算答案 －89 和 ＋112 显然都是错误的。

还应注意，溢出主要是指带符号二进制数进行加减运算时可能产生的一种结果。对于无符号数，第8位不是符号，只有第8位的进位才称为进位，而不采用溢出这个概念。

对于无符号数还应注意一点，当两个无符号数相减时，不允许用小的数去减大的数，因为小减大它的差一定是负数，无符号数的前提是没有符号，显然也不允许有负数，如果这样做，减的结果也必然是错误的。

第三节　BCD码及文字符号代码

一个二进制数，可以表示一个无符号数，也可以表示一个带符号数，而且可以根据事先约定代表一个文字、一个符号或者代表一个特定的内容。当它代表文字或符号时称为代码。例如00100100B，作为数，它表示的是无符号数24H或是带符号数+24H，如果事先约定，它又能代表符号$。所谓约定，可以按标准约定，也可以自行约定，下面介绍两种按标准约定的常见文字符号代码。

一、BCD码（Binary coded decimal）

BCD码是一种以二进制形式表示十进制数的编码，又称二-十进制码，它貌似二进制，实际是十进制数。

BCD码以4位为一组，选用0000B~1001B 10种状态，代表0~9共10个数，舍弃其余的6种状态。当BCD码与十进制数进行互换时，可以按4位一组，逐组进行互换。

例1-2　将84.7转换为BCD码。

8　　4　.　7　　0

1000　0100　.　0111　0000

例1-3　将BCD码10010100.01110010转换为十进制数。

1001　0100　.　0111　0010

9　　4　.　7　　2

要将一个二进制数转换为BCD码，通常先将它转换为十进制数，然后再转换为BCD码。同样将BCD转换为二进制数，也是先转换成十进制数，再转换为二进制数。

例1-4　将二进制数11110011B转换为BCD码。

$11110111B = 1\times27 + 1\times26 + 1\times25 + 1\times24 + 0\times23 + 0\times22 + 1\times21 + 1\times20 = 247$

十进制数为　　2　　4　　7

BCD码为　　0010　0100　0111

即二进制数11110011B的BCD码为$(001001000111)_{BCD}$，或按习惯写成两个字节即等于$(00000010\ 01000111)_{BCD}$。

例 1-5 将 BCD 码 $(10001001)_{BCD}$转换为二进制数。

1000　1001

十进制数为　8　9

二进制数为 = 01011001B

BCD 码的低 4 位向高 4 位进位要遵循逢 10 进 1 的原则，这与二进制显然不同，二进制的低 4 位要向高 4 位进位，必须遵循逢 16 进 1。因此两个 BCD 码相加，可以按二进制数的相加方法，但要以 4 位为一组，逐组相加，凡相加后的和大于 9 者，还应进行加 6 修正。

例 1-6 将 BCD 码 $(01011000)_{BCD}$与 $(01101001)_{BCD}$相加

```
  0101  1000
+ 0110  1001
-------------
  1100  0001
进位 1←┘
```

由于第一组的和为 17，大于 9，除进位 1 外，余数还应加 6 修正。第二组和为 12，也大于 9，也应加 6 修正，即上式的和应予以修正。

```
  1100  0001                检验    58
+ 0110  0110 - - - 加 6 修正      + 69
-------------                -------
0001  0010  0111 - - - 正确值     127
```

两个 BCD 码相减，凡低 4 位有向高 4 位借位者，都要进行减 6 修正，无借位的当然无需修正。

例 1-7 将 BCD 码 $(10000101)_{BCD}$与 $(00101000)_{BCD}$相减

```
  1000  0101
- 0010  1000
-------------
0101  1101
借位 1←┘
  0101  1101          检验   85
+       0110               - 28
-------------          -------
  0101  0111             57
```

二、ASCII 码

ASCII 码是美国信息交换标准代码的简称，由 American Standard Code Information Interchang 的第一个字母组成。

计算机只能辨认或存储 0 和 1 两个数码，也就是计算机内部只能使用二进制数，但在编制计算机程序或信息时，还会碰到许多文字符号，这就需要把文字符号改成一串二进制代码，即把文字符号数码化，现在国际通用的文字符号代码是