WEIXINGJI RUANYINGJIAN SHEJI

罗伯特·M·布雷斯威兹
富兰克·斯特恩
著

微型机软硬件设计

编　者　的　话

在信息爆炸的当代社会中，微型计算机扮演着越来越重要的角色。各类微型计算机资料教材成为市场上的畅销书籍，无疑对于加快我国现代化建设的步伐起着相当重要的促进作用。

然而，也必须看到，我们所接触到的微型计算机教材，绝大多数是介绍一台微型计算机或微处理机的。在微型计算机发展日新月异的今天，势必存在这样一个问题：当我们学习掌握了一台微型计算机，准备实际应用时，这台机器很有可能已经过时了，甚至淘汰了。那么怎样才能使学到的计算机知识立于持久不败之地呢？经过认真努力和多方面的支持和帮助，我们翻译了《微型计算机系统——硬件/软件设计》，现将此书推荐给广大读者。

本书的最大特点是着眼于微型计算机发展，机器部件与体系结构并举，基础知识与应用实例共存。在介绍 Intel8080 微型计算机时，从零点起步，逐步深入。不仅介绍了微型计算机的功能，而且阐述了其优缺点，就其开发潜力，扩展方向都作了详细的介绍。这样，无论是对于计算机业余爱好者还是对于实际应用计算机的工程技术人员来说，都将是一本不可多得的参考书。

其次，本书立足于微型计算机系统的硬件、软件设计，从基础的软件程序到高级的数据检测，给出了大量的实例，进行了详细地讨论。尤其为其它微型计算机教材所罕见的是本书全面系统地介绍了各种各样的外围设备以及接口设计，这对于实际工程的微型计算机系统设计无疑是十分有益的。

由于本书是集体翻译的，加之专业水平有限，在翻译过程中难免存在一些错误或不妥之处，恳请广大读者提出宝贵意见。

参加本书翻译的有：

孙曙光：前言，第八章

关　平：第一章，第九章，答案

单维平：第二章

李　华：第三章

展广连：第四章

付永涛：第五章，第七章

富玉斌：第六章

雷云山：全书校对

《电站设备自动化》编辑部

1985年7月

前　言

本书全面地论述了以8080微处理机为基础构成的系统设计技巧。它由浅入深地介绍计算机的系统设计，并侧重于介绍计算机的构成元件。本书内容这样编排，使学习过开关电路和逻辑理论但缺乏编程基础知识的大学生能较容易地理解发展了的微型计算机系统。本书为这些学生选择编排了软件编程章节，以说明计算机结构和系统设计方面的特殊概念。硬料方面的详细内容提供给具备编程基本知识的大学生作为入门知识。因此本书满足了两方面的要求，即硬件和软件设计方面的要求以及把硬件和软件组成为系统的要求。

本书引导大学生从了解微机基础知识出发，通过对本课题基础知识的详细讨论，进而研究这方面的专门高级课题。本书内容的编排高度模块化，从而使教员能根据需要任意选择一些章节编写教学课程。前四章结合小规模和大规模集成设计技术论述组合式和序列式逻辑设计基础知识。其余章节论述计算机系统的元件，包括存贮器、微处理器、输入一输出和软件。本书给出的示例使大学生能够弄懂软件和计算机的组织结构。书中的高级课题包括数据采集系统、微型计算机板、16位微处理机和高级技术。

计算机技术发展如此迅速，以至于此领域内的任何一本教科书中的材料都容易过时。因此本书利用侧重介绍新技术发展和一般性论述的方法，解决内容过时问题。尽管可以选择许多其他微处理机，但我们还是选择了8080为主介绍，因为8080设计简明，应用广泛，并且计算功能和操作性能具有代表性。采用8080为基本微处理机能使大学生们经过不困难的过渡阶段，把他们的知识扩展到16位微处理机方面。

最后，感谢那些为本书尽过力的人们，只有通过他们的努力，这样一部洋洋万言之作才得以脱稿。

特别感谢托马斯·福克斯和诺尔曼·萨姆，感谢他们从开始就对本书写作给予鼓励和支持。衷心感谢罗伯特·敦纳尔博士和C·J·劳伦斯，他们在讨论修改初稿时花费了许多时间。

我们同样感谢芙格利德·狄特克，林达·卡斯娃，玛丽·布罗，他们花费大量时间，不厌其烦地在原稿改动后多次打字。我们也要感谢珍妮·塞尔波，是她编写了课后练习和答案。

最后但并非最不重要的是我们必须感谢我们的夫人们，弗吉娅和桑卓，感谢她们给予的鼓励和为此书而牺牲的那些周末。

目　录

第一章 绪 言

对计算机的基本要求是具有表示和存贮数，并完成以数表示操作的能力。第一章介绍各种计数制和数字代码。因为已证明二进制计数制是机器用的最普遍的和最有效计数制，故对其进行了详细介绍。还对表示信息的其他实用代码作了介绍和比较。

1.1 十进制与二进制

我们现用的十进制有十个独立符号，即0、1、2、3、4、5、6、7、8、和9。这十个符号称为阿拉伯数字。这个十进制以十为单位计算，大概是由于人有十个指头而来的。大于9的数通过规定数字占有的位置或位的意义的习惯方法表示。一般说来，任何数字N_b在基数b为正整数条件下可由位置记数法来表示。如：

$$N_b = A_{n-1}A_{n-2}\cdots A_2A_1A_0A_{-1}A_{-2}\cdots A_{-(m-1)}A_{-m}$$

式中的每个A代表给定顺序的数字。

b＝基数，n＝整数位数，m＝小数位数（对于十进制数374.26，b＝10，n＝3，m＝2，$A_2=3$，$A_1=7$，$A_0=4$，$A_{-1}=2$和$A_{-2}=6$）。

这个有序数字的数值计算是：

$$N_b = A_ib^{n-1} + A_ib^{n-2}\cdots A_ib^1 + A_ib^0 + A_ib^{-1} + A_ib^{-2}\cdots A_ib^{-(m-1)} + A_ib^{-m}$$

式中，$A_ib^{n-1} = A_{n-1}$，$A_ib^{n-2} = A_{n-2}$，$A_ib^1 = A_1$等等。位置系数A_i为：

$$0 \leqslant A_i \leqslant (b-1)$$

因而，可以明显看出，数表示为其基数b的幂乘以适当系数A_i的和〔在十进制中，很明显$0 \leqslant A_i \leqslant (b-1) \leqslant (10-1) \leqslant 9$〕。

一般说来，我们可以用如下和式表示任何数N_b。

$$N_b = \sum_{x=-m}^{n-1} A_ib^x$$

式中X等于位置系数的幂指数。

例如，十进制数7043〔$A_i=7$，0，4，3；m＝0；n＝4〕的数值如下计算：

$$\begin{aligned} 7043_{10} &= (7\times10^3) + (0\times10^2) + (4\times10^1) + (3\times10^0) \\ &= 7000 + 0 + 40 + 3 \\ &= 7043 \end{aligned}$$

现在我们可以看出十进制数系是非常完美和简单的。为了表示任何数，只需学会十个基本数和定位值制。

当然，完全可以利用基数不是10的数系，而且往往十分实用。例如，当涉及到时间、英寸、英尺以及一打或罗（＝12打）时，十二进制是非常方便的。然而，在数字式系统中，基数为2的数系极为实用。这种称为二进制的系统只利用两个简单的数字0和1。它针对数字电路应用的优点在于如下事实：我们可使两个数字（0和1）与用符号（0和1）表示的逻辑变量的两个可能

真值（真或假）一对一地对应。这些二进制变量只涉及到信号电平的有或无，因而，特别易于可靠地产生、传送和存储。我们往往能因此感到非常方便。最主要问题就是变量应能提供远远多于2的各种可能状态，因此必须采用称为代码的二进制变量的有序组合来表示。

在二进制中，具体数字代表2的系数，而不是10的系数，就如同在十进制中代表10的系数一样。还有，任何二进制数都可以表示为如下的形式：

$$N_2 = \sum_{x=-m}^{n-1} A_i 2^x$$

式中的x，m，和n的定义如前所述。因为(b－1)＝(2－1)＝1，所以

$$0 \leqslant A_i \leqslant 1$$

例如，二进制数11001表示如下：

$$11001_2 = (1\times2^4)+(1\times2^3)+(0\times2^2)+(0\times2^1)+(1\times2^0)$$
$$= 16+8+0+0+1$$
$$= 25_{10}$$

换句话说，二进制的11001等于十进制的25。

可利用计算过程指出十进制和二进制之间的相似性。在十进制中，计算过程是在特定位置上按0，1，2……8，9的次序增加数字。当在这个位置上的数达到10时，我们向相邻左边一位进1。因为，二进制只能通过两级。所以，直接向左位进1要比十进制进位频繁。这样，20个十进制数的二进制数表示如表1－1所示。

表格1—1 等效的十进制和二进制数

十进制表示法	二进制表示法
A_1A_0	$A_4A_3A_2A_1A_0$
00	00000
01	00001
02	00010
03	00011
04	00100
05	00101
06	00110
07	00111
08	01000
09	01001
10	0101_0
11	0101_1
12	0110_0
13	0110_1
14	0111_0
15	0111_1
16	1000_0
17	1000_1
18	1001_0
19	1001_1
20	1010_0
$N_{10}=(A_1\times10^1)+(A_0\times10^0)$	$N_2=(A_4\times2^4)+(A_3\times2^3)+(A_2\times2^2)+(A_1\times2^1)+(A_0\times2^0)$

注意，二进制数的小数部分也可用和十进制一样的方法进行转换。也就是说，正像在十进制中12.236等效于

$(1\times10^1)+(2\times10^0)+(2\times10^{-1})+(3\times10^{-2})+(6\times10^{-3})$一样，在二进制中1101.11101等效于：

$(1\times2^3)+(1\times2^2)+(0\times2^1)+(1\times2^0)+(1\times2^{-1})+(1\times2^{-2})+(1\times2^{-3})+(0\times2^{-4})+(1\times2^{-5})$

于是，不难看出任何计数中的每个数字在表示一完整数系中，必须采用一定的权或权系数。例如，把二进制数101.11转换为十进制数，如表1—2所示。

表格1—2，二进制至十进制的转换

二进制101.11　　**表格1—2 二进制至十进制的转换**

A_2	A_1	A_0	A_{-1}	A_{-2}	权
1	0	1	1	1	
2^2 4	2^1 2	2^0 1	2^{-1} 1/2	2^{-2} 1/4	二进制 十进制
4+0+1			1/2+1/4		
5			3/4		

十进制　5.75

这种换算法引入了把二进制数转换成十进制数的各种数学方法或技术。通常，十进制数可能具有整数部分和小数部分。每一部分应分别转换成等效的二进制数。完整的表示法由这两部分以及二进制小数点的组合构成。为把十进制数转换成等效的二进制数，通常采用如下两种方法：减法方法和除——乘法方法：

减法方法：

在利用这种方法时，首先必须从十进制数中减去2的最高次幂，然后在部分实现二进制数适当权位置置1。而且，必须把这个过程继续到十进制数减小到0为止，这是冗长和麻烦的数字转换过程。而且，这种方法也很麻烦。虽然因为能够心算，这种方法用于转换小数比较方便，但很少用于转换大数。下面的例子将有助于说明减法方法的转换过程。考虑基数为10的数53：

1.从十进制数中减去其包括的2的最高次幂。由于$2^5=32$和$2^6=64$，前者是最高次幂，可以把它减去：

$$\begin{array}{r} 53 \\ -32(\text{或}2^5) \\ \hline R=\ 21 \end{array}$$

权　阵	
$A_5\ A_4\ A_3\ A_2\ A_1\ A_0$	
$2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$	
1	

2.从余数R中减去下一个2的最高次幂。由于$2^3=8$，$2^4=16$和$2^5=32$，我们减去$2^4=16$：

$$\begin{array}{r} 21 \\ -16(\text{或}2^4) \\ \hline R=\ 5 \end{array}$$

权　阵	
$A_5\ A_4\ A_3\ A_2\ A_1\ A_0$	
$2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$	
1　1	

3.从余数R中减去下一个2的最高次幂。如果前一次减完后剩的余数减下一个最高次幂不够减，则在权阵该位置置0。例如，因为R=5减去$2^3=8$不够减，所以$A_3=0$。5减去下一个2的最高次幂$2^2=4$够减，则如下所示：

$$R=\begin{array}{r}5\\-\ 4(\text{或}2^2)\\\hline 1\end{array}$$

权　阵					
A_5	A_4	A_3	A_2	A_1	A_0
2^5	2^4	2^3	2^2	2^1	2^0
1	1	0	1		

4. 继续上述第三步，在权阵中如上所述填。由于$2^0=1$和$2^1=2$，我们减去前者：

$$R=\begin{array}{r}1\\-1(\text{或}2^0)\\\hline 0\end{array}$$

权　阵					
A_5	A_4	A_3	A_2	A_1	A_0
2^5	2^4	2^3	2^2	2^1	2^0
1	1	0	1	0	1

我们现在已经完成了整个转换。所得到的权阵如下：

A_5	A_4	A_3	A_2	A_1	A_0
1	1	0	1	0	1

$$\begin{aligned}(53)_{10}&=A_5\times2^5+A_4\times2^4+A_3\times2^3+A_2\times2^2\\&\quad+A_1\times2^1+A_0\times2^0\\&=(A_5A_4A_3A_2A_1A_0)_2\\&=(110101)_2\end{aligned}$$

小数可以用同样的方式转换。例如，考虑把基数为10的小数0.5625转换为二进制小数。过程如下：

1. 从十进制数中减去基数为2的最大的小数数值。由于$2^{-1}=.50$和$2^{-2}=.25$，我们减去前者：

$$R=\begin{array}{r}0.5625\\-.5000(\text{或}2^{-1})\\\hline .0625\end{array}$$

权　阵			
A_{-1}	A_{-2}	A_{-3}	A_{-4}
2^{-1}	2^{-2}	2^{-3}	2^{-4}
1			

2. 从余数中减去下一个基数为2的最大的小数数值。因为$2^{-4}=0.0625$，所以，减去它：

$$R=\begin{array}{r}0.0625\\-0.0625(\text{或}2^{-4})\\\hline 0.0000\end{array}$$

权　阵			
A_{-1}	A_{-2}	A_{-3}	A_{-4}
2^{-1}	2^{-2}	2^{-3}	2^{-4}
1	0	0	1

这样，利用这种减法方法得到如下结果：

$(0.5625)_{10}=(0.10010)_2$

应该注意，余数往往不能经过有限个有效十进制数字很快地收敛到0。实际上，在把十进制数表示为二进制形式时，很可能存在舍入误差，而不管可用于表示的二进制数字的多少。

除/乘法方法

这种方法是用把给定十进制数除2的方法，把它转换为二进制数。如除完有余数，在二进制权阵中的最低二进制位下必须置1。如果没有余数，在权阵中必须放一个0。然后，把第

一次除法所得结果除以2，并且重复进行这个过程直到所得结果减小到0为止。例如，考虑十进制数53：

	余数	权阵 A_5	A_4	A_3	A_2	A_1	A_0
		2^5	2^4	2^3	2^2	2^1	2^0
53/2							
26	1						1
26/2							
13	0					0	
13/2							
6	1				1		
6/2							
3	0			0			
3/2							
1	1		1				
1/2							
0	1	1					

$(110101)_2 = (53)_{10}$

		权阵 A_{-1}	A_{-2}	A_{-3}	A_{-4}	A_{-5}	A_{-6}	A_{-7}
		2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
0.5623 × 2 = 1.1246	>1	1						
0.1246 × 2 = 0.2492	<1		0					
0.2492 × 2 = 0.4984	<1			0				
0.4984 × 2 = 0.9968	<1				0			
0.9968 × 2 = 1.9936	>1					1		
0.9936 × 2 = 1.9872	>1						1	
0.9872 × 2 = 1.9742	>1							1

如果需要把一个十进制小数转换为二进制，把这个数字乘以2。若乘积小于1，则最高有效二进制小数位是0；如果乘积大于1，则最高有效二进制小数位是1。小数部分第二位数用同样法则对第一步得出结果的小数部分乘2，并判断乘积大于还是小于1得出。这个过程一直进行到达到所要求精度为止。例如考虑把$(0.5623)_{10}$转换为二进制数，其转换过程示于8页的下表。现在把我们得出的结果舍入到7位，则得出：

$$(0.5623)_{10}=(01000111)_2$$

因为我们没有把这种转换过程继续作下去，所以现在产生了0.0076的舍入误差。正如上面所述，二进制的等效数可能具有无限多位，因此产生了舍入误差。由于计算机的字长是有限的，很明显，十进制小数表示法通常都含有误差。这种误差的大小取决于计算机字长。

1.2 其他计数制

八进制计数制：

用二进制数表示一个数要比用十进制数表示所需要的数字多。自从制造了计算机以来，就必须具有易于被机器和操作人员理解的计数制。在读和写二进制数时，操作员很容易出错。八进制计数制能够消除这些问题中的许多问题。它是以8为基数的计数制，其数字是0，1，2，…7，这些数字可以用通用符号来重新表示如下：

$$N_8=\sum_{x=-m}^{n-1}A_i8^x$$

式中x，m和n如前面的定义。因为$(b-1)=(8-1)=7$，

$$0\leqslant A_i\leqslant 7$$

因为八进制计数制的基数是8，并且我们可以看出，

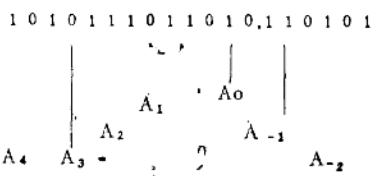
$$8=2^3$$

为表示八进制数只需将二进制数字分成三位一组。表1－3表明十进制、二进制和八进制数之间的相互关系：

表1—3 十进制、二进制和八进制的相互关系

十进制	二进制	八进制
00	0000	00
01	0001	01
02	0010	02
03	0011	03
04	0100	04
05	0101	05
06	0110	06
07	0111	07
08	1000	10
09	1001	11
10	1010	12

为说明如何把二进制数转换为八进制，分析二进制数1010111011010.110101。首先必须把二进制数分成三位一组。如下面所示：



利用表1－3，我们可以把每个八进制数表示如下：

A_4	A_3	A_2	A_1	A_0	A_{-1}	A_{-2}
1	2	7	3	2	6	5

因此得出：

$(1010111011010.110101)_2 = (12732.65)_8$

八进制数以同样法则转换成十进制数。于是：

$N_8 = A_{n-1}8^{n-1} + A_{n-2}8^{n-2} + \cdots + A_0 8^0 + A_{-1}8^{-1} + \cdots$代换，我们得出：

$$(12732.65)_8 = (1\times8^4 + 2\times8^3 + 7\times8^2 + 3\times8^1 + 2\times8^0 + 6\times8^{-1} + 5\times8^{-2})_{10}$$
$$= (4096 + 1024 + 448 + 24 + 2 + 0.75 + 0.078125)_{10}$$
$$= (5594.8281)_{10}$$

重要的是要记住，计算机不利用八进制数运算。这种计数制只用于简化二进制计数制的解释和利用。

为把一个整数从十进制转换成八进制，可以用8反复除十进制数。并且利用每个余数作为要得出八进制数的数字。例如，为把$(205)_{10}$转换为八进制表示，我们作如下除法：

$8\overline{)205}$ 商25 $R=5$

$8\overline{)25}$ 商3 $R=1$

$8\overline{)3}$ 商0 $R=3$

权 阵			
A_3	A_2	A_1	A_0
8^3	8^2	8^1	8^0
3	1	5	

把一个小数从十进制转换成八进制的过程类似于从十进制小数转换为二进制的过程，只是作乘法因数是8而不是2。例如：考虑十进制小数0.7792968：

$(0.7792968)_{10}$

$\times\quad 8$

6.2343680 整数 6

0.2343680

$\times\quad 8$

1.874944 整数 1

0.874944

$\times\quad 8$

6.999552 整数 7

权 阵				
A_{-1}	A_{-2}	A_{-3}	A_{-4}	A_{-5}
6	1	7	0	0

$(0.7792968)_{10} = (.617)_8$

二进制编码的十进制(BCD)

因为用于数字计算机的大部分电子电路元件本身固有的运算为二进制，所以对计算机来说二进制计数制是最适用的。另方面，对人来说最方便的计数制显然是十进制，所以，许多计算机的计算机侧输入—输出设备发送/接收二进制数。而操作人员侧输入—输出设备则接收/发送十进制数。这种代码的转换需要附加电路或计算时间，利用称为二进制代码的十进制(BCD)表示法，可把这种附加计算时间减少到最低限度。在这种系统中，利用位的代码组表示十个十进制数字中的每个数字。BCD码是用4位二进制数来代替十进制数中的每个十进制数字。所以，数字0到9的表示法如表1—4所示。

这种编码方案常常被称为8421代码。注意到每位十进制数字需要四位（二进制数的）表示。并且规定了每位的权，例如，最右边一位的权为1，而最左边一位的权为8。用所出现各位的权相加，可得出用代码组表示的十进制数字。如表示十进制数字312，需要如下12位：

$$(312)_{10} = (0011\ \ 0001\ \ 0010)_{(BCD)}$$

这是非常有用的代码，但仍然有它的缺点。首先，因为利用四位来表示十进制数0到9（而实际上四位可以表示十进制数0一直到15）所以没有充分利用二进制组中的位。第二个缺点是难于形成数的补数。这个课题将在下面提出。

表1—4 十进制数用二进制编码的十进制数表示法

十进制	二进制	二进制编码的十进制数（BCD）
00	0000	0000 0000
01	0001	0000 0001
02	0010	0000 0010
03	0011	0000 0011
04	0100	0000 0100
05	0101	0000 0101
06	0110	0000 0110
07	0111	0000 0111
08	1000	0000 1000
09	1001	0000 1001
10	1010	0001 0000
11	1011	0001 0001
12	1100	0001 0010

这种代码的最重要用途是用于表示字母、符号和数字。例如，因为两位十进制数00，01，02……09足以表示从0到9的所有数字，所以用余下的两位数10到99表示A，B，C……Z和其他符号，如“+”、“-”、“*”以及“?”。表1—5表明用BCD码表示各种符号以及数的表示方法。

表1—5 BCD表示法

符号	BCD
1	0000 0001
2	0000 0010
3	0000 0011
	0101 0010
☆	0101 0100
+	0110 0000
—	0100 0000
.	0111 0011
/	0010 0001
?	0111 0010

某些计算机采用BCD进行其所有运算。这些计算机称为字符机（character，machines），并且和普通计算机相比在某种程度上难于构成。在普通计算机中，采用BCD码表示数据的外围设备包括卡片读出器、穿孔和复制机；纸带读出器和穿孔；磁带记录器及某些字母数字显示设备。因而，怎样强调BCD码表示法的重要性也不会过分。

余3BCD数码

余3二进制代码的十进制数表示法，常常用于计算机中，并且是更常用于制表机中以简求出十进制反码的方法。通常的做法是在计算机中用加上减数的补码的方式进行减法计算。但是，当采用BCD码时，被存储数的自然补码不总是可用的。计算机求出二进制数补码的最直接方法是简单地把每个0换成1或把每个1换成0。然而，BCD代码数0010的自然补码是1101，或十进制数B。这个数不是合格的一位数BCD代码，因为这个数大于10。为了克服这个问题，已采用了一些代码方式，所谓的余3代码就是其中最重要的一种。因为它把十进制数加3，然后再用规定权的二进制代码形成BCD数，所以称为余3代码。表1—6示出了10个十进制数字及其对应的余3代码。

1—6 余3代码

十进制数	余3代码
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

求出用余3代码表示十进制数的二进制代码组的每位数字的补码，即形成了十进制反码。例如，0100（十进制数1）的补码是1011（十进制数8）。表1—7示出了10个十进制数字的余3代码表示法，及其对应的十进制反码。

应该注意，利用十进制数在余3代码中每个0变为1和每个1变为0的方法，就形成了十

进制数的十进制反码。

表1—7 BCD数的十进制反码表示法

十进制数字	余3代码	十进制反码
0	0011	1100
1	0100	1011
2	0101	1010
3	0110	1001
4	0111	1000
5	1000	0111
6	1001	0110
7	1010	0101
8	1011	0100
9	1100	0011

格雷代码

正像我们已经看到的一样，二进制用0和1序列来表示数，这种序列是自然序列，并且容易理解。因为它遵照我们已接受的十进制表述方法中对数字位规定有效数值的模式。然而，如果我们希望能用一个完全任意的序列来表示一个数，为此只要求每个数具有其各自不同的特定序列。除自然数系外的其他数系称为数字代码，因为我们必须给定代码，以确定序列所表示的数值。

虽然二进制代码非常适于进行数字计算。但它在采样两个相邻二进制代码位置的应用中遇到了严重问题，例如，当一台计算机控制一个轴的角位置而且用许多1和0区分位置值时，如果轴的角位置7和8用0111和1000表示，则系统能够产生输出值0111和1000，这个值是正

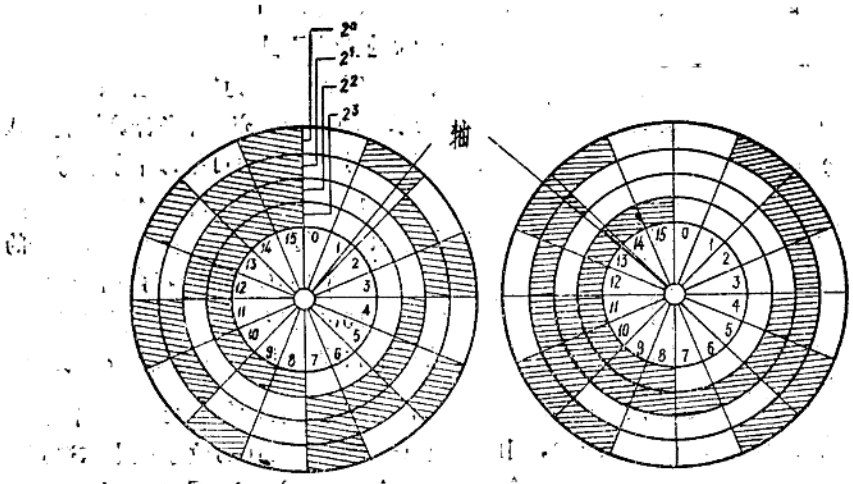


图1—1 四位二进制和格雷代码盘

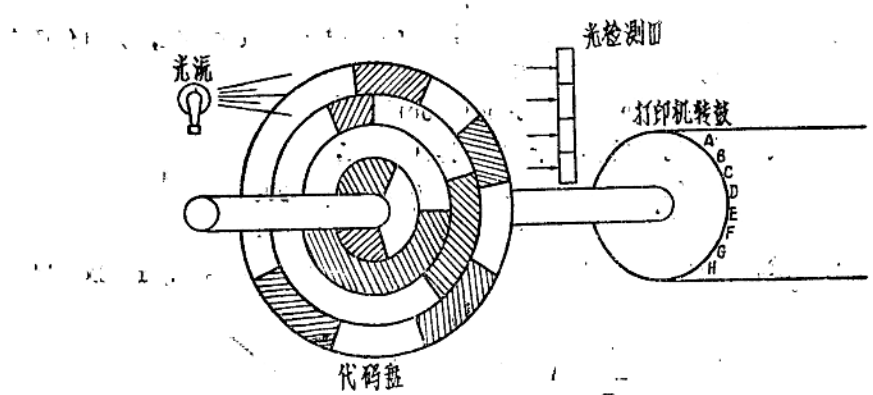


图1—2 操作中格雷码盘

确的。然而，在计算序列时，为了减小产生误差的机会，要求每一次改变的数字不超过一位。避免这种困难的代码称为循环码；在二进制中，它称为格雷代码。尽管该误差通常只是暂态的，但对系统运算仍然是非常有害的。注意到，在从0111变到1000过程中，四位位置的数值变化。如果这些位其中的一位不正确，将产生相当大的误差。

4位二进制代码轮和4位格雷代码轮示于图1—1。两个代码轮均由16个代码组成。每个代码的角度分辨率为360°/16或22.5°。

格雷代码的最普通应用，是用于确定转动轴位置。通常是把代码轮固定到旋转轴上。这个代码轮包含透明和不透明部分。用光电管或任何其他光敏传感器来测定通过代码轮的光（见图1—2），旋转轴的位置由光敏传感器检测出的1和0导出。格雷代码非常适用于位置转换器。

格雷代码在诸方面类似于二进制代码，它总具有相同的位长和相同的最高有效位。它与二进制代码的主要区别是其两个相邻数只有一位不同。表1—8表明，十进制数0到12的二种代码形式。

当检测元件处在其工作位置时，它针对每一段将产生0或者1输出。因此，可作为输出产生由1到15的任何数。因为一次仅有一段变化值，因而格雷代码消除了多义性错误。

表1—8 二进制和格雷代码的比较

十进制	二进制	格雷代码
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010

十六进制计数制

十六进制或基数为16的数系是普通BCD表示法的一种扩充。许多计算机利用八位字节组织运算。因为可把每个字节考虑为一个单元整体或者考虑为两个4位的实体，所以程序员有把每个8位字符表示为单一的二进制数或两个BCD数的选择自由。例如，字节0001，1000可以表示一个二进制数，在这种情况下它表示十进制数24；或者表示两个BCD数，在这种情况下它表示十进制的18。因此，如果有表示4位的每一可能变化的代码，就方便了。因为必须表示16个不同的数，因此必须有一个基数为16的数系或十六进制数系。因为数字0到9不能满足需要，所以采用六个附加符号A、B、C、D、E和F，如表1—9所示。

为把二进制数转换为十六进制数，简单地把二进制数划分成四位数字一组，并且依照表1—9转换由四位组成的每一位。例如，二进制数：

1101111110000110001

可如下划分成四位数字构成的组转换成十六进制数DF831：

1101	1111	1000	0011	0001
D	F	8	3	1

为了把该十六进制数转换为十进制数，我们利用基数变换法则。这样，十进制数可以表示为：

$$N_{10}=\sum_{x=-m}^{n-1}A_i16^x$$

表1—9 十六进制代码

十进制	二进制	十六进制
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

式中x、m和n如前面的定义。由于

$$(b-1)=(16-1)=15=F,$$

$$0\leqslant A_i\leqslant F$$

为把十六进制的DF831转换为十进制数，作如下运算：

$$\begin{aligned}(DF831)_{16}&=(D\times16^4)+(F\times16^3)+(8\times16^2)\\&\quad+(3\times16^1)+(1\times16^0)\\&=(13\times16^4)+(15\times16^3)+(8\times16^2)\\&\quad+(3\times16^1)+(1\times16^0)\\&=915505_{10}\end{aligned}$$

对很长一串的二进制数字来说，十六进制形式是非常有用的。因为减少了读无分段标志的0和1字符串时出现人错误的可能性。要求存储器转贮或读存贮时，大型计算机常常利用这种方式打印输出数据。

1.3 各种数系的算术运算

加法

因为我们很熟悉十进制加法，让我们从二进制加法讲起。二进制加法用十进制加法同样的方式进行。实际上，二进制加法是非常容易，因为整个二进制加法表如下

$0+0=0$　　$0+1=1$

$1+0=1$　　$1+1=0$　同时进1

进位用和十进制计算相同的方式处理。为说明二进制加法，下面举三个例子如左侧所示。

十进制	二进制
5	0101
7	0111
12	1100

十进制	二进制
12	01100
15	01111
27	11011

十进制	二进制
2.75	010.110
5.25	101.010
8.00	1000.000

八进制加法和十六进制加法类似（对于前者，8或大于8时进位1；而对于后者，16或大于16时进位1），正如下面的例子所示：

十进制	八进制
27	33
13	15
40	50

十进制	十六进制
27	1B
13	0D
40	28

减法

二进制数减法可用类似于十进制减法的方法直接相减。应该注意，如果我们要减去一个负数，可改变减数的符号，并作加法运算。因此，如何表示二进制负数就变为一个极为重要问题了。至少有三种不同的如下表示：

1.符号和数值

2.一的补码

3.二的补码