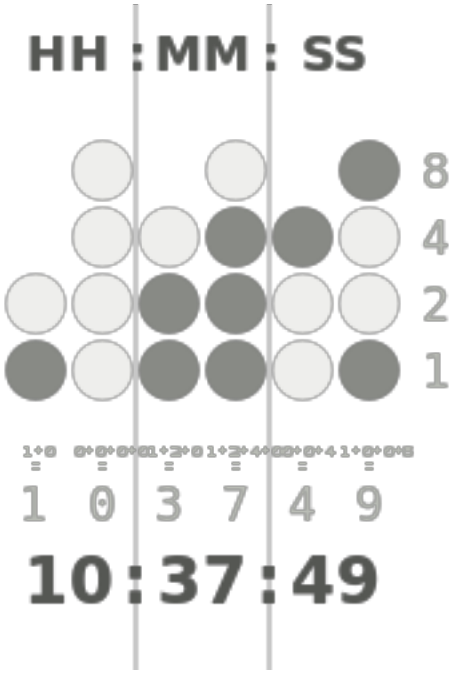
# 二进制编码的十进制

从维基百科，自由的百科全书




一个二进制时钟可能会使用LED灯来表示二进制值。在这个时钟中，每一列的LED的传统的六十进制的时间示出了二进制编码的十进制标号。

在计算和电子系统，每个十进制数字表示的固定数目的位，通常是四个或八个二进制编码的十进制数的**二进制编码的十进制**（**BCD**）是一类，但其他规格（如6位）有过去使用。特殊的位模式有时被用于一个符号或其他的指示（例如，错误或溢出）。

在面向字节的系统（即最现代化的计算机），该术语通常是*未压缩的* BCD表示一个完整的字节的每个数字（通常包括一个标志），而*填料*的BCD通常通过利用这样的事实，在一个单一的字节编码两个十进制数字四个位是足以代表范围为0—9。精确的4位编码可能会有所不同，由于技术上的原因，比如看到余3。

BCD的主要优点是更准确的代表性和数量小数四舍五入以及易于转换为人类可读的表示。二进制定位系统相比，BCD的主要缺点是所需的电路的复杂性，实现基本的算术和一个稍微不那么密集存储少量增加。

BCD码被用在许多早期的十进制计算机。虽然BCD还没有得到广泛的应用仍然是重要的，在过去，的十进制定点和浮点格式，并继续被使用在金融，商业，工业计算机，微妙的转换和浮点固有的舍入误差二进制表示不能被容忍的。[1]

**内容**

[ 隐藏 ]



## 基础 [ 编辑 ]

如引言中所描述，BCD利用的事实，即任一十进制数字表示的一个四比特码型：

| 十进制 | BCD码<br>8 4 2 1 |
|---|---|

| 数字 | |
|---|---|
| 0 | 0 0 0 0 |
| 1 | 0 0 0 1 |
| 2 | 0 0 1 0 |
| 3 | 0 0 1 1 |
| 4 | 0 1 0 0 |
| 5 | 0 1 0 1 |
| 6 | 0 1 1 0 |
| 7 | 0 1 1 1 |
| 8 | 1 0 0 0 |
| 9 | 1 0 0 1 |

由于大部分的计算机上的数据存储在8位字节，这是可以使用下列方法中的一个，以BCD数进行编码：

- **未压缩的**：每个数字被编码为一个字节，用4位表示数字和不具有意义的其它位。
- **盒装**：两个数字被编码为一个字节，一个数字的至少显著四位（位0到3）最重要的四位（4至7位）和其他数字。

作为一个例子，编码的十进制数**91，**使用未压缩的BCD码的查询结果在下面的两个字节的二进制模式：

十进制：9 1
二进制：0000 1001 0000 0001

压缩BCD，同样数量将融入一个字节：

十进制：9 1
二进制：1001 0001

因此，一个未压缩的BCD字节的数值范围为是通过九包容零，而一个压缩BCD是0到99（含）范围。

代表数大于一个字节的范围内，可以使用任意数量的连续字节。例如，为了表示压缩BCD 十进制数**12345**，使用大端格式，程序编码如下：

十进制：1 2 3 4 5
二进制：0000 0001 0010 0011 0100 0101

需要注意的是最重要的半字节的最重要的字节是零，这意味着该号码在现状**012345**。另外请注意压缩BCD是如何更有效地存储使用情况相比未压缩BCD编码相同数量未压缩格式存储会消耗100％以上。

转移和掩盖用于包装或解压缩一个压缩BCD数字。使用其他的逻辑运算（数字）转换为它的等效的位模式，或相反的过程。

## BCD电子 [ 编辑 ]

BCD是很常见的电子系统中的其中一个数字值将要被显示，特别是在纯粹的数字逻辑，并且不包含微处理器组成的系统。通过使用BCD码，可以极大地简化了用于显示的数值数据的操作，处理每个数字作为一个单独的单一子电路。这更加紧密地匹配硬件设计人员可能选择使用了一系列独立的相同七段显示器，建立计量电路，例如显示器的物理现实。如果数字量存储和操纵纯二进制中，这样的显示接口将需要复杂的电路。因此，在计算的情况下，整个BCD相对简单的工作可能会导致比一个简单的系统整体转换为二进制。

同样的道理也适用于这种类型的硬件时，使用一个嵌入式微控制器或其它小型处理器。通常情况下，较小的代码的结果时，表示数字的BCD格式的内部或转换为二进制表示，因为等有限的处理器可以是昂贵的。对于这些应用程序，一些小的处理器采用了BCD算术模式，协助写程序时，操纵BCD数量。

## 盒装BCD [ 编辑 ]

**压缩BCD**（或简单地**压缩十进制**），自20世纪60年代或更早的版本一直在使用和实施在所有的IBM大型机硬件从那时起，被称为双位数每字节编码的一个常见变化。在大多数形式，有一个十进制整数的一个或多个字节，其中每一个的两个半字节每个字节代表一个十进制数，更重要的位中的每一个字节的上半部分中，最左边的字节（驻留在最低的内存地址）包含压缩十进制值的最显着的数字之和。最右边的字节的低四位通常用作符号标志（虽然这四位在某些陈述可能被用作至少有效位数字压缩十进制值并没有在所有的一个标志，即，纯粹是无符号的）。作为一个例子，一个4字节的值由8个半字节，其特征在于，上7个半字节储存一个7位的十进位值的数字之和的最低四位表示十进制整数值的符号。

标准的符号值是1100（十六进制）阳性（+）和1101（D）为负（ - ）。这种约定是来自会计条款（信用卡和借记）的缩写，广泛应用于压缩十进制编码会计制度。[ *引证需要* ]其他允许的标志是1010（A）和1110（E）为正和1011（B）为负。有些实现也提供了无符号BCD值1111（F）的标志蚕食[ *引证需要* ]。ILE RPG使用1111（F）为负为正和1101（D）[ 2 ]在压缩BCD中，数字127表示的0001 0010 0111 1100（127C）和-127表示的0001 0010 0111 1101（127D）。巴勒斯系统使用1101（D）阴性，其他任何值被认为是一个积极的迹象值（处理器正常化一个积极的迹象，1100（C））。

| 注册数字 | BCD码<br>8 4 2 1 | 签署 | 笔记 |
|---|---|---|---|

| | | | |
|---|---|---|---|
| 一 | 1 0 1 0 | + | |
| 乙 | 1 0 1 1 | - | |
| Ç | 1 1 0 0 | + | 首选 |
| ð | 1 1 0 1 | - | 首选 |
| ê | 1 1 1 0 | + | |
| F | 1 1 1 1 | + | 无符号 |

无论有多少个字节宽的字，总是有偶数个半字节，因为每个字节有其中两个。因此，*n*个字节的一个字可以包含（2 *Ñ*）-1十进制数字，这始终是一个奇数个数字。的十进制数与*ð*数字需要½（*ð* +1）字节的存储空间。

例如，一个4字节（32位），字可容纳7个十进制数字，再加一个符号，可以代表的数值范围从±9,999,999。因此，数字-1,234,567，宽为7位编码为：

```
0001 0010 0011 0100 0101 0110 0111 1101
 1 2 3 4 5 6 7 -
```

（请注意，类似字符串的第一个字节的压缩十进制-最显着的两位数-通常是存储在存储器中的最低地址，独立于机器的字节序。）

相比之下，一个4字节的二进制补码整数值可以代表从-2,147,483,648到2147483647。

虽然压缩BCD没有最大限度地利用存储（约$^1/_6$使用的内存被浪费），转换为ASCII，EBCDIC，或各种Unicode的编码仍然是微不足道的，，没有算术运算。计算器或手计算，定点小数运算提供的准确度和相容性的需要额外的存储需求通常所抵销。密集填料的BCD存在避免存储死刑的，也需要共同的转换没有算术运算。

盒装BCD支持COBOL编程语言中的“计算3”（IBM扩展通过许多其他编译器厂商）或“压缩十进制”（1985 COBOL标准的一部分）的数据类型。除了IBM的System/360和更高版本兼容的大型机，包装BCD实施VAX处理器的原始数字设备公司在原生指令集和巴勒斯公司中等大型机系统的原生格式（后裔从20世纪50年代的200 Electrodata系列）。

## 定点压缩十进制 [ 编辑 ]

定点十进制数所支持的一些编程语言（如COBOL和PL / I）。在这些语言允许程序员指定的隐式小数点前面的数字。例如，压缩十进制编码值的字节12 34 56 7C代表定点值+1,234.567的隐含的小数点则位于第四和第五位之间：

```
12 34 56 7C
 12 34.56 7 +
```

小数点实际上并不存储在内存中，为压缩BCD存储格式不为它提供。它的位置是简单地称为编译器生成的代码的作用相应的各种算术运算。

## 高密度编码 [ 编辑 ]

如果一个十进制数需要4位，然后三位小数需要12位。然而，由于$2^{10}$，（1024）是大于$10^{3}$（1000），如果三个十进制数字一起进行编码，只有10位是必须的。两个这样的编码是*陈何编码*和*密密麻麻的小数*。后者具有的优点是，该编码的子集两位数字编码的最佳7 bits和一个在4位的数字，如定期BCD。

## 区位十进制 [ 编辑 ]

有些实现，例如IBM的大型机系统，支持**划十进制**数字表示。每位十进制数被存储在一个字节的低四位编码位数以BCD形式。高四位，被称为“区域”位，通常被设置为一个固定的值，使得字节保存一个数字相对应的字符的值。EBCDIC系统使用的区域为1111（十六进制F）的值，这样的产量字节范围F0到F9（十六进制），这是字符“0”到“9” 的EBCDIC代码。同样，ASCII系统使用给字符码为0011（十六进制）的区域值，30到39（十六进制）。

对于签署划十进制值，最右边的（至少）区四位持有的标志位，这是相同的一组用于签署压缩十进制数（见上文）的值。因此，区位十进制编码值的十六进制字节F1 F2 D3代表签署的十进制值-123：

```
F1 F2 D3
 1  2 -3
```

## EBCDIC划十进制转换表 [ 编辑 ]

| BCD数字 | 十六进制 | | | | EBCDIC字符 | | | |
|---|---|---|---|---|---|---|---|---|
| 0+ | C0 | A0 | E0 | F0 | {（*） | | \（*） | 0 |
| 1+ | C1 | A1 | E1 | F1 | ー | ~（*） | | 1 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2+ | C2 | A2 | E2 | F2 | 乙 | Ş | Ş | 2 |
| 3+ | C3 | A3 | E3 | F3 | Ç | 吨 | Ť | 3 |
| 4+ | C4 | A4 | E4 | F4 | ð | ü | ü | 4 |
| 5+ | C5 | A5 | E5 | F5 | ê | v | V | 5 |
| 6+ | C6 | A6 | E6 | F6 | F | 瓦特 | W | 6 |
| 7+ | C7 | A7 | E7 | F7 | ĝ | x | X | 7 |
| 8+ | C8 | A8 | E8 | F8 | ħ | Ÿ | Ÿ | 8 |
| 9+ | C9 | A9 | E9 | F9 | 我 | Ž | Ž | 9 |
| 0- | D0 | B0 | | | }(*) | ^(*) | | |
| 1- | D1 | B1 | | | ĵ | | | |
| 2- | D2 | B2 | | | Ķ | | | |
| 3- | D3 | B3 | | | ł | | | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 4 - | D 4 | B 4 | | | 中 号 | | | |
| 5 - | D 5 | B 5 | | | Ñ | | | |
| 6 - | D 6 | B 6 | | | Ø | | | |
| 7 - | D 7 | B 7 | | | P | | | |
| 8 - | D 8 | B 8 | | | Q | | | |
| 9 - | D 9 | B 9 | | | ɾ | | | |

（*）*请注意：这些字符取决于本地字符代码页设置。*

## 定点的区位十进制 [ 编辑 ]

有些语言（如COBOL和PL / I）直接支持定点的区位十进制值，分配一个隐含的小数点的十进制数字位数之间的某个位置。例如，给定签署划一个六字节的十进制值有一个隐含的小数点右侧的第四位，十六进制字节F1 F2 F7 F9 F5 C0代表的价值+1,279.50：

```
F1 F2键F7 F9 F5 C0
 1 2 7 9。5 +0
```

## IBM和BCD [ 编辑 ]

*主要文章：BCD（6位）*

IBM使用条款6位*字母数字*代码，代表数字，大写字母和特殊字符的**二进制编码的十进制**和**BCD**。在最早期的IBM计算机，包括IBM 1620，IBM 1400系列，和IBM的700/7000系列的非十进制的硬件架构的成员使用的BCD *alphamerics的*一些变化。

IBM 1400系列字符寻址的机器，每个位置6位标记为*乙，A，8，4，2*和*1*，加上奇校验位（*C*）和一个字标志位（*M*）。用于编码的位数是*1*至*9*，*乙甲*为零，位*8*到*1的*标准的4位BCD码表示的数字值。对于字符的位*乙甲*简单地来自从“12”，“11”，和“0”，“区域拳”中的穿孔卡字符代码，位*8*到*1*，从*1*到*9的*冲头。一个“12区”冲*乙*的“11区”设置*乙*，和一个“0区”（0冲床与任何其他组合），设置*à*。因此，信**Ä**，*（12,1）*在打孔卡格式，编码*（B，A，1）*，货币符号**$**，*（11,8,3）*在打孔卡，*（B，8,3）*。这使得电路转换之间的打孔卡格式和内部存储格式很简单，只有一些特殊情况。

一个重要的特殊情况下是数字*0*，代表一个孤独*0*冲卡，和 *( 8,2 )* 的核心内存。[3]

IBM 1620的记忆被组织成6位可寻址数字，通常*8*，*4*，*2*，*1*加*F*，作为一个标志位和*Ç*，奇校验位。BCD *alphamerics*编码位数对，在偶数地址的奇数地址的位数的数字和“位”与“区”，“区域”与*12*，*11*，和*0* “地带拳”在1400系列。输入/输出转换之间的内部数字对外部标准的6位BCD码转换的硬件。

IBM 7070，IBM 7072，和IBM 7074 *alphamerics的*十进制架构中使用数字对编码（数字，**而不是**使用两出五码BCD），10位的字，与“区”的数字和“数字”，在右边的数字。输入/输出转换之间的内部数字对外部标准的6位BCD码转换的硬件。

IBM System/360的引入，扩大6位BCD *alphamerics的* 8位EBCDIC，允许添加更多的字符（例如，小写字母）。盒装BCD *数字*数据类型的变量长度也被实施，提供机器指令直接压缩十进制数据进行算术。

在IBM 1130和1800，压缩BCD软件支持IBM的商业子程序包。

今天，BCD数据仍然大量使用在IBM的处理器和数据库，如IBM DB2，大型机和Power6处理器。在这些产品中，BCD通常划为BCD（EBCDIC或ASCII），盒装BCD（每字节的两位十进制数），或“纯”BCD编码（每个字节的低4位BCD存储为一个十进制数字）。所有这些都是内使用的硬件寄存器和处理单元，并在软件中。

## 其他计算机和BCD [ 编辑 ]

数字设备公司 的VAX-11系列包括指令，可以直接执行算术压缩BCD数据之间的转换压缩BCD数据和其他整数表示。VAX的压缩BCD格式兼容IBM的System/360 IBM更高版本兼容的处理器。从CPU的MicroVAX和后来的VAX实现下降这种能力，但保留了代码兼容较早的机器，通过实施操作系统提供的软件库中缺少的指示。这是通过自动调用不再执行指令时所遇到的异常处理，让使用它们的程序可以执行新的机器上不加修改。

在更近的电脑这样的能力几乎总是实现在软件而不是CPU的指令集，但BCD数值数据仍然是非常常见的商业和金融应用。

## 除了 使用BCD [ 编辑 ]

这是可能的：首先将二进制，然后转换为BCD码之后执行除了在BCD。的简单相加的两位数字的转换可以通过加入6（即16 - 10）时，将5位的结果，加入的一对数字有一个值，该值大于9。例如：

```
1001 + 1000 = 10001
  9 + 8 = 17
```

需要注意的是10001是二进制，而不是十进制数，表示期望的结果。BCD十进制，不可能存在每个数字的值大于9（1001）。要纠正这一点，6（0110）添加那笔钱，那么结果将被视为两个半字节：

```
10001 + 0110 = 00010111 => 0001 0111
  17 +6 = 23 1 7
```

，0001和0111的两个半字节的结果，对应于数字“1”和“7”。这将产生“17”在BCD码，这是正确的结果。

该技术可扩展通过添加组由右至左，传播作为进位的第二位，总是比较结果，对每个数字的总和的5位到第9，增加多个数字。

## 减法的BCD [ 编辑 ] [价格]

减法运算是通过添加的10的补码的减数。BCD码中的数字的符号来表示，数字0000是用于表示一个正数，和1001是用来表示一个负数。剩下的14个组合是无效的迹象。为了说明签署BCD减法，考虑下面的问题：357 - 432。

在符号的BCD，357 0000 0011 0101 0111。采取的9个的432 的补码，然后加一，可以通过以下方式获得的10的补码的432 。所以，999 - 432 = 567，和567 + 1 = 568。前568的BCD负号代码，可以表示数字-432。因此，符号的BCD -432 1001 0101 0110 1000。

现在这两个数字符号的BCD表示，可以将它们加在一起：

```
0000 0011 0101 0111 1001 0101 0110 1000 1001 1000 1011 1111
  0 3 5 7 9 5 6 8 9 8 11 15
```

由于BCD是十进制表示的一种形式，有几个数字的总和是无效的。在存在一个无效的条目（任何BCD数字大于1001）的情况下，添加了6产生进位位和导致的总和成为一个有效的条目。加6做的原因是，有16个可能的4位BCD值（自$2^4$ = 16），但只有10个值是有效的（0000到1001）。因此，添加6无效的条目在下面的结果：

```
1001 1000 1011 1111 0000 0000 0110 0110 = 1001 1001 0010 0101
  9 8 11 15 + 0 0 6 6 9 9 2 5
```

因此，相减的结果是1001 1001 0010 0101（-925）。要检查答案，注意，第一个位是符号位，哪些是消极的。这似乎是正确的，因为357 - 432应为负数。要检查剩余的数字，代表他们在小数。1001 0010 0101 925。十补 925 1000 - 925 = 999 - 925 + 1 = 074 + 1 = 75，所以计算出的答案是-75。要进行检查，执行标准扣除，以验证357 - 432 -75。

注意事件，有不同数量的啃加在一起（如1053 - 122），数量最少的数字必须先用零填充之前，十补或减去。因此，1053 - 122，122将首先被表示为0122，将不得不计算的10的补码为0122。

## 背景 [ 编辑 ]

在这篇文章中所描述的是二进制编码的十进制计划最常用的编码，但还有很多其他的。这里的方法可以被称为*简单的二进制编码的十进制*（*校本课程开发*）或*BCD 8421*。在表的标头中，“8 4 2 1'，*等等*，表示示出了每个位的权重;注意，第五列中的权重的两个是否定的。划BCD数字ASCII和EBCDIC字符代码的例子，也示于表中。

下表表示的十进制数字从0到9中的各种BCD系统：

| 数字 | BCD码 | 余3或Stibitz代码 | BCD 2 4 2 1或艾肯条码 | BCD 8 4 -2 -1 | IBM 702 IBM 705 IBM 7080 IBM 1401 8 4 2 1 | ASCII 0000 8421 | EBCDIC 0000 8421 |
|---|---|---|---|---|---|---|---|

| | 8421 | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0000 | 0011 | 0000 | 0000 | 1010 | 0011 0000 | 1111 0000 |
| 1 | 0001 | 0100 | 0001 | 0111 | 0001 | 0011 0001 | 1111 0001 |
| 2 | 0010 | 0101 | 0010 | 0110 | 0010 | 0011 0010 | 1111 0010 |
| 3 | 0011 | 0110 | 0011 | 0101 | 0011 | 0011 0011 | 1111 0011 |
| 4 | 0100 | 0111 | 0100 | 0100 | 0100 | 0011 0100 | 1111 0100 |
| 5 | 0101 | 1000 | 1011 | 1011 | 0101 | 0011 0101 | 1111 0101 |
| 6 | 01 | 1001 | 1100 | 1010 | 0110 | 0011 0110 | 1111 0110 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | 10 | | | | | | |
| 7 | 0111 | 1010 | 1101 | 1001 | 0111 | 0011 0111 | 1111 0111 |
| 8 | 1000 | 1011 | 1110 | 1000 | 1000 | 0011 1000 | 1111 1000 |
| 9 | 1001 | 1100 | 1111 | 1111 | 1001 | 0011 1001 | 1111 1001 |

## 法律史 [ 编辑 ]

戈特沙尔克在1972年的案件诉本森，美国最高法院推翻了下级法院的决定，允许专利BCD编码号转换为二进制的计算机上。这是一个重要的情况下，在确定软件和算法的专利。

## 与纯二进制比较 [ 编辑 ]

### 优点 [ 编辑 ]

- 许多非整数值，如0.2十进制，有无限的二进制位值表示（0.001100110011 ...），但有一个有限的二进制编码的十进制位值（0.0010）。因此，如果一个系统的小数基于二进制编码的十进制表示避免错误表示和计算这些值。
- 缩放因子为10（或10）电源是简单的，这是一个十进制的比例因子是有用的，当需要代表一个非整数的数量（例如，在财务计算）
- 四舍五入十进制数字边界是简单。小数的加法和减法，不需要四舍五入。
- 对齐两个十进制数（例如1.3 + 27.08），是一种简单，准确，移位。
- 转换为字符形式或用于显示（例如，一个基于文本的格式（如XML），或一个七段显示器的驱动信号）是一个简单的每个数字的映射，并且可以做线性（ö（Ñ））的时间。从纯二进制的转换涉及到比较复杂的逻辑，跨越数字，大量没有被称为线性时间的转换算法（参见二进制数字系统）。

### 缺点 [ 编辑 ]

- 有些操作更复杂，实现加法器需要额外的逻辑，导致他们包装和提前产生进位。15-20％以上的电路需要较纯二进制BCD [引证需要]乘法比移掩模加（二进制乘法，需要二进制移位和加法或相当于较为复杂的算法，它们需要使用每个数字或数字组是必需的）
- 标准BCD需要4个比特的数字，大约20％以上的空间比二进制编码（4位的比例，以记录$_2$ 10位是1.204）。包装使三位数字的编码在10位，存储开销大大降低，是未对齐的8位字节的边界上，对现有的硬件，从而导致较慢的实现在这些系统中的编码而牺牲。
- 实用现有实现的BCD通常比业务慢的二进制表示，尤其是在嵌入式系统中，[需要的引证]由于有限的处理器支持原生的BCD操作。

## 应用 [ 编辑 ]

许多个人电脑的BIOS中存储的日期和时间以BCD因为MC6818实时时钟芯片用在原来的IBM PC AT主板提供了BCD编码的时间。这种形式很容易转换成ASCII码显示。[4]

雅达利8位家族使用的计算机BCD实现浮点算法。MOS 6502处理器采用了一个的BCD模式，影响了加法和减法指令。

的PlayStation 3的早期型号BCD存储日期和时间。这导致了2010年3月1日的全球停运的控制台。作为BCD存储一年的最后两位数字被曲解为16造成一个悖论，使大多数功能无法使用本机的日期。

## 具象的变化 [ 编辑 ]

存在各种BCD实现，采用其他数字表示制造的可编程计算器，惠普，德州仪器和其他通常采用浮点 BCD格式，通常有两个或三个数字（十进制）指数。多余位的符号位可以用来表示特殊的数值，如无穷大，下溢 /上溢，错误（闪烁显示）。

### 签名的变化 [ 编辑 ]

符号十进制值可表示在几个方面。COBOL编程语言，例如，支持一共有五个区位十进制格式，每一个数字的符号进行编码，以不同的方式：

| 类型 | 描述 | 例子 |
|---|---|---|
| 无符号 | 没有迹象蚕食 | F1 F2 F 3中 |
| 签署后*（规范格式）* | 在最后一个（最显着）字节注册蚕食 | F1 F2 C 3 |
| 领导签名*（附加穿孔）* | 注册四位在第一（最重要的）字节 | C 1，F2，F3 |
| 尾随单独签署 | 独立的符号字符字节（'+'或' - '） | F1 F2 F3 2B |

| | | |
|---|---|---|
| | 的数字字节 | |
| 签署领先的独立 | 独立符号字符的字节（'+'或' - '）前位字节 | 2B F1 F2 F3 |

## 电话二进制编码的十进制（TBCD）[ 编辑 ]

GSM开发**TBCD**，剩余的（未使用）位组合被用于添加特定的电话字符扩展为BCD 。[5]它是向后兼容 BCD。

| 十进制数字 | BCD码<br>8 4 2 1 |
|---|---|
| * | 1 0 1 0 |
| # | 1 0 1 1 |
| 一 | 1 1 0 0 |
| b | 1 1 0 1 |
| Ç | 1 1 1 0 |
| 用作填充材料时，有一个奇数 | 1 1 1 1 |

| 个数字 | |
|---|---|

## 替代编码 [ 编辑 ]

如果错误表示和计算比的转换，并从显示的速度更重要，缩放的二进制表示可以被使用，存储的十进制数的二进制编码的整数和一个带符号的二进制编码的十进制数的指数。例如，可以表示为2× $10^{-1}$ 0.2 。

这表示允许快速乘法和除法，但可能需要在加法和减法小数点对齐移位10的幂。它是适当的应用具有固定的小数位数不那么需要调整，特别是金融应用，其中2个或4个小数点后的数字通常是不够的。事实上，这几乎是一个定点算术形式，因为是隐含的小数点的位置。

陈浩编码提供了一个布尔转换组，转化为三个BCD编码的位数和10位的值，可以有效地在硬件编码只有2或3门延迟。密密麻麻十进制是一个类似的计划，用于大部分的有效除铅的数字，IEEE 754-2008标准中指定的两种可供选择的十进制编码。

### [ 编辑 ]

- 双五元编码的十进制
- 陈浩编码
- 密密麻麻的十进制
- 双击涉足，将二进制数转换为BCD码的算法
- 格雷码
- 2000年问题
- 十进制电脑

## 参考文献 [ 编辑 ]


1. ^ “一般十进制算术”。
2. ^ “ILE RPG参考”。
3. ^ IBM BM 1401/1440/1460/1410/7010电码BCD顺序图
4. ^ http://www.se.ecu.edu.au/units/ens1242/lectures/ens_Notes_08.pdf信息
5. ^ “信令协议和交换（SPS）使用抽象语法记法一（ASN.1）在电信应用协议”的准则。


## 延伸阅读 [ 编辑 ]


- 麦肯齐，查尔斯·E.（1980）。*编码字符集：历史与发展*。Addison-Wesley出版，ISBN 0-201-14460-3。
- *算术运算数码计算机*，397pp，RK理查兹D.范NOSTRAND有限公司，NY，1955
- 赫尔曼·施密德，*十进制计算*，国际标准书号0-471-76180-X，266pp，Wiley出版社，1974

- *Superoptimizer：看看在最小的程序*，亨利Massalin，ACM SIGPLAN声明。22＃10（建筑支持的编程语言和操作系统的第二次国际会议上），pp122-126，ACM，IEEE计算机学会出版社＃87CH2440-6，1987年10月
- *VLSI设计冗余二进制编码的十进制此外*，科鲁兹Shirazi的，大卫YY韵，张全张，IEEE第七次年度国际凤凰计算机和通信会议，1988年，PP52-56，IEEE，1988年3月
- *数字逻辑基础*，2003年由布朗和弗拉内希奇的
- *修改进向前看BCD加法器，CMOS和可逆逻辑实现*，人士Himanshu Thapliyal和哈米德·Arabnia，电脑设计的2006年国际会议（CDES'06）提出诉讼，ISBN 1-60132-009-4，PP64-69，CSREA出版社，2006年11月
- *可逆实施密集的包装十进制转换二进制编码的十进制格式使用IEEE-754R*，A. Kaivani，A. ZAKER Alhosseini，S. Gorgin，和M. Fazlali的第九届中国国际信息技术会议（ICIT '06），pp273-276，IEEE，2006年12月。
- 十进制算术参考书目