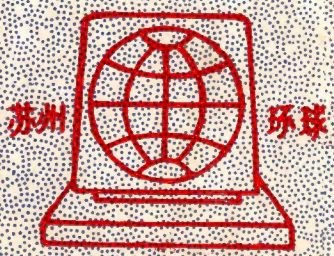
LASER-310 深入

徐敏



苏州市环球电脑软件公司

一九八六年八月

目　录

第一章　LASER—310 的常数、变量、函数功能的深入剖析……(1)

第二章　LASER—310使用的BASIC语言（基础部分深入）……(18)

第三章　录音机的使用 ……(27)

第四章　LASER PP40打印机的图案打印模式……(34)

第五章　LASER—310使用的BASIC语言（扩展部分）……(44)

第六章　LASER—310 屏幕功能、光笔、游戏棒 ……(62)

附录：

LASER—310 BASIC程序用字符及保留词机器码表……(73)

前　言

本书是继《LASER—310 入门》的第二册，是在第一册的基础上，对计算机的功能进行深入剖析。

通过对本书的学习和实践，能够比较充分地发挥 LASER—310 计算机所具有的丰富功能，大大提高编程的技巧。对软件开发中遇到的难题也比较容易解决，并为LASER—310微电脑控制家用电器打下基础。这本书中的应用举例及详细说明编在第三册《LASER—310 应用》书中。

书中不妥之处，敬请读者提出批评。

编者　一九八六年五月

第一章　LASER-310 的常数、变量、函数功能的深入剖析

第一节　常数功能

一、数值常数

下面按照整数、单精度实数、双精度实数进行分类说明，并在每类精度等级的常数中说明数值允许范围、数值在内存中的表示法、定点数和浮点数的输入、输出规定、尾标的使用等等。

（一）整数

1. 整数的数值范围　在LASER—310 中，整数的数值范围是—32768到＋32767闭区间内不带小数点的数，但－32768 这个数仅适合赋值给整数型变量。为了证明这一点，我们可以在整数后面加上整数标志的尾标%，看计算机能否输入输出这个整数。例如键入？ 32767％↓则屏幕显示32767，再键入？ 32768％↓则屏幕显示出错信息(语法错误)。这说明最大整数是32767。用同样方法可以证明最小整数(作为常数)是-32767，但赋值给整数型变量时，最小整数为-32768。例如键入 A％＝－32768:？ A％↓则屏幕显示－32768，若键入 A％＝－32769:？ A％↓则屏幕显示出错信息（数据溢出)。因而我们可以把整数的允许数值范围定义在－32768～＋32767。

从键盘输入整数时，不管是否带有尾标%，只要符合上述数值范围，计算机均作为整数存入内存。若超过上述数值范围或带有小数点(带有尾标％时将出错)，计算机将作为实数存入内存。

2. 整数在内存中的表示和占用字节数　每个整数在计算机内存中占用2个字节。由于每个字节由8位二进制数表示，因而每个整数用 16 位二进制表示，16位中的首位是数的正负符号位，0表示该数为正数，1表示该数是负数。例如整数1表示为0000，0000，0000，0001；整数 2 表示为 0000，0000，0000，0010。因此最大整数可以写为 0111，1111，1111，1111。这个数化为十进制即为＋32767（化为 16 进制是 7FFF)。这就是为什么在LASER-310计算机上最大整数是32767。

负整数在机器内存中的表示法是一个比较复杂的问题，这里只作简单介绍。在LASER－310 计算机中，负整数是按照补码方式表示的。16 位中的最高位用 1 表示负整数，补码是取原码（与负整数绝对值相同的正整数的二进制编码）的反码（把二进制表示的编码中凡为0的均变为1，凡为 1 的均变为0）再在末位加上1。例如32767的二进制编码的反码是1000,0000,0000,0000,而补码(即－32767)是1000,0000,0000,0001。把－32767再减去1将为－32768，因此最小整数是1000,0000,0000,0000即－32768(化为16进制是8000)。再如－1可以从原码(十进的＋1)0000,0000,0000,0001取反码1111,1111,1111,1110再在末位加1得到补码（即十进制的－1)：1111.1111,1111,1111。原码、反码，补码三者的关系

是：补码＝反码＋1；补码＋原码＝0。 例如(－1)＋1，在机器内是1111,1111,1111,1111＋0000,0000,0000,0001＝1,0000,0000,0000,0000。 通过上述说明， 可以知道为什么在LASER－310计算机上最小整数是－32768。

3．整数的定点数表示 在LASER－310中， 由于整数只占内存2个字节， 所以整数不能用浮点数方式表示（参见实数在内存中的表示)，只能用定点数方式表示。 例如，键入？3.45E2%↘ 即出错。因此无论输入整数还是输出（显示或打印）整数，均为定点数方式。

4．整数尾标%的使用 在通常情况下进行整数运算， 完全不必在整数后面带上尾标%，但在需要确定整数的数值范围(如前所述)时，尾标%起了很重要的作用。尾标%常用于数值变量，以节约内存。

（二）单精度实数

1．单精度实数(以下简称实数)的数值允许范围 为$-10^{38}\sim-10^{-38}$和$10^{-38}\sim10^{38}$。即最小数是-10^{38}，最大数是10^{38}； 最小正实数是10^{-38}， 最大负实数是-10^{-38}。 而在$-10^{-38}\sim10^{-38}$之间的实数均作为0。可以键入？1E＋38；－1E38；1E－38；－1E－38↘屏幕显示同样内容，若键入？1E－39；－1E－39↘则屏幕显示0与0。若键入？1E＋39↘或？－1E＋39↘则屏幕显示出错(溢出错误：OVERFLOW EROR)。

上述数值范围仅表示数值的数量级， 其界线不如整数数值范围那么严格， 例如键入？1E＋38＋1000↘则屏幕显示1E＋38而不出错。

2．单精度实数在内存中的表示和占用字节数 每个单精度实数在计算机内存中占用4个字节，即用32位二进制表示， 其中前8位表示小数点移动（向左或向右）的位数，后24位表示十进制正负实数的有效数字6位。 实数存放在内存中的形式与整数不同，其形式可用式子 $\pm2^t\times S$来表示($-128\leqslant t\leqslant127,0\leqslant S<1$)。例如，实数5化为二进制为111，机器用上述式子表示的形式$+2^3\times0.111$放在内存的4个字节中（此处t＝3表示把0.111小数点向右移3位，即得到二进制数111)如下：

0000,0011：0.111,0000,0000,0000,0000,0000

前8位中首位表示t的正负，0表示正(小数点向右移动)，1表示负(小数点向左移动)，后面7位000,0011表示t的值等于3，小数点移动的位数是3位。

后24位中首位表示实数的正负，同样0表示正实数，1表示负实数，后面23位都放在小数点后面(实际上内存中没有小数点，仅为了说明方便而加上的)，上例表示S的值为0.111。

根据以上说明，可以推算出式子$\pm2^t\times S$中t的数值范围和S所能表示的有效数字的位数。 由于前8位表示t的数值，故知t的最大值为二进制的0111,1111即为十进制的127(16进制为7F)；t的最小值为127的反码－128，其二进制表示为1000,0000(16进制为80)。由于2^{127}与10^{38}两个数相当接近： 若取对数$\log_{10}2^{127}\approx127\times0.3010=38.227$，而$\log_{10}10^{38}=38$，$\log_{10}2^{-128}\approx-128\times0.3010=-38.528$； 而$\log_{10}10^{-38}=-38$， 可知实数的取值范围在$-10^{38}\sim-10^{-38}$；$10^{-38}\sim10^{38}$。

由于后24位二进制编码表示正、负实数的有效位数，其中第一位符号位，因此最多能表示的有效数字位数用二进制表示就是 0111,1111,1111,1111,1111,1111。化为十进制是8388607；虽有7位有效数字，但比这个数再大一些的7位数如8388608或更大一些数字就不

能用24位二进制编码表示了，因而单精度实数只表示6位有效数字。这是正实数的情况。至于负实数，其所能表示的有效数位的数最多是－8388607，只能表示6位有效数字。

3．单精度实数的定点数方式与浮点数方式输入输出的规定　单精度实数具有定点数和浮点数两种方式表示。向计算机输入数值时，可以用定点数方式输入，如键入？123456↘则屏幕显示123456；也可用浮点数方式输入，如键入？123.456E3↘则屏幕显示123456。上面例子说明计算机对这个数只采用定点数方式输出，而不管输入方式如何。但并不是所有情况如此。如键入？1234567↘或键入123.4567E4↘，则屏幕的显示 1.234567E＋06。这个例子又说明计算机对1234567这个数只采用浮点数方式输出，也不管输入方式如何。是否超过6个有效数字计算机就一定以浮点数方式输出呢？若键入？12345678↘则屏幕显示12345678，又是定点数方式输出。再键入？123.45678E5↘则屏幕显示1.234567E＋07，再键入？123.45678 D5↘屏幕显示12345678，这些现象说明以定点数和双精度浮点数方式输入12345678这个数时，计算机的双精度定点数方式输出(详述见后)。而以单精度的浮点数方式输入的使整数部分超过6位有效数字的数值，计算机以浮点数方式显示。

我们可以发现，在用定点数方式向计算机输入实数时，对于正实数，若输入的数值在0.009999994～999999.4范围内且有效数字不超过7位，计算机就作定点数方式输出（显示或打印)。对于负实数，若输入数值的绝对值在上述范围内，也得到同样情况。

若用浮点数方式输入，则输入的数值在上述范围内（不管有效数字是否超过7位)，计算机均以定点数方式输出，读者可以用上述方法加以验证。

对LASER—310的定点数方式和浮点数方式输入、输出的规定作出结论如下：

(1) 定点数方式输入有效数字不超过7位；(2) 单精度浮点数方式输入且不管有效数字是否超过7位，两种输入方式输入实数的绝对值均在0.009999994～999999.4范围内时，计算机以单精度定点数方式输出(显示或打印)。

凡不符合以上条件之一者，计算机均以单精度浮点数方式或双精度方式输出。其中当定点数方式输入且有效字不足8位，或单精度浮点数方式输入（不管有效数字是否超过7位)，且两种方式输入实数的绝对值范围在0.9999994～999999.4之外时，计算机就以单精度浮点数方式输出。

4．单精度实数尾标！的使用　只适用于定点数方式输入实数。例如键入？123456781则屏幕显示1.23457E＋07。此时输入的实数若在单精度定点数规定的数值范围和有效数字以内，则以定点数方式输出；若超过数值范围和有效数字的规定，则以单精度浮点数方式输出（如本例)。在通常进行数值运算时，只要参加运算的各数字没有一个是双精度型的，则运算的结果(不管有效数字是否超过７位)超过整数范围时总是单精度实数，因此通常没有必要使用尾标!。

（三）双精度实数

1．双精度实数的数值允许范围与单精度实数相同　为 -10^{38}～-10^{-38} 和 10^{-38}～10^{38}，在 -10^{-38}～10^{-38} 之间的实数均作为0。试验方法与单精度实数完全相同。如键入？1D38；－1D38；1D－38；－1D－38↘则屏幕显示同样内容（输出速度比单精度方式慢得多)，其它操作可参照单精度实数的键入方法，允许数值范围的界线也是不严格的。

2．每个双精度实数占计算机内存8个字节　用64位二进制表示，其中前８位与单精

度实数一样，表示小数点前后移动的位数，后56位二进制表示有效数字为十进制16位的正负实数。双精度实数存放在内存中的形式同样可以用式子 $\pm 2^t \times S$ 表示（$-128 \leqslant t \leqslant 127$，$0 < S < 1$），从式子 $\pm 2^t \times S$ 中 t 的数值范围可推知双精度实数的取值范围与单精度实数相同；从S的56位二进制数码转换成十进制数（最大为36028797018963967）可推知双精度实数的有效数字达16位。

3．双精度实数的定点数方式、浮点数方式的输入、输出规定

双精度实数的定点数方式输入有三种方法：

(1) 在实数允许数值范围内输入8位以上有效数字，例如键入 ？12345678↘则屏幕显示12345678。

(2) 输入数字的有效数位不足8位时，可以加上尾标＃。例如键入？1284567＃↘则屏幕显示1234567（＃号不显示；若上例不加尾标＃。则屏幕显示单精度浮点数1.23457E＋06），再如键入？10＃/3↘（或？10/3＃↘），则屏幕显示 3.333333333333333 共16位有效数字。

(3) 通过STR$与VAL串函数转换(详见函数功能部分)。

双精度实数也可用浮点数方式输入，如下（不管有效数字是多少位），例如键入 ？1.234567D4↘则屏幕显示12345.67。又如键入？1D2/7↘则屏幕显示 14.28571428571429（注：$1D2 = 1 \times 10^2 = 100$）。

因此双精度实数的输入方法共有以上四种。

双精度实数的定点数方式输出的规定如下：

如果输入定点数方式（不带＃号）的实数其有效数字不少于 8 个，或带＃号的任一实数，或双精度浮点数方式的任一实数并且三种输入方式的绝对值符合以下条件。数值范围在0.00999999999999999279999990～9999999999999999.27999999之内，则计算机以双精度定点数方式输出。例如键入？9999999999999999.27999999↘则屏幕显示 9999999999999999。但若键入？99999999999999999.279999991↘则屏幕显示1D＋16。同法可对小数进行试验。

双精度实数的浮点数方式输出的规定如下：

如果输入的定点数方式（不带＃号）的实数其有效数字不少8位，或带＃号的任一实数，或双精度浮点数方式的任一实数，并且三种输入方式的绝对值在上述数值范围以外的，则计算机就以浮点数方式输出。例如键入12345678901234567↘则屏幕显示 1.23856789012345 7D＋16。又如键入？.009＃↘，则屏幕显示9D－03。

4．双精度实数尾标＃的使用　只适用于定点数方式输入双精度实数，在输入的实数后面加上尾标＃。在进行双精度实数的运算时，若定点数方式输入的实数有效数字不是8位，可以加上尾标＃使运算结果以双精度实数方式输出。顺便指出，在变量后面不可带尾标＃（这是LASER—310计算机的特点），关于变量取双精度的若干做法和规定详见后述。

（四）各精度等级的数值常数的混合运算

如果在一个运算式中出现各种精度等级的数值常数，应注意以下几点。

1．整数运算的结果仍在整数规定的范围内时，仍以整数方式输出。例如键入？32760

%/60↘则屏幕输出546；若整数运算的结果超过整数规定的范围时，则以单精度实数方式输出(输出定点数或浮点数则根据计算结果是否符合定点数方式输出的规定)。例如键入？32767%*32%↘则屏幕显示1.04854E＋06。

2．整数与单精度实数混合运算的结果与上述相同。

3．整数与单精度实数混合运算的结果，或整数、单精度、双精度实数混合运算的结果；或单精度实数与双精度实数混合运算的结果，若仍在整数规定的范围内时，仍以整数方式输出，否则以双精度(定点数或浮点数)方式输出(结果有时不正确)。

例如键入？(1%＋2!＋3#)/7↘则屏幕显示.8571428571428571（结果正确）

若键入？1#＋1/3↘则屏幕显示1.333333343267441，结果显然不正确。

又如键入？1.5/1.5#↘则屏幕显示1(结果正确)

而键入？1.2/1.2#↘则屏幕显示1.00000003973643(结果不正确)

为了在使用中避免差错，应注意以下几个问题：

(1) 单独运算的式子（如上面的例子中的1/3）若为非整数且不带#符号，与双精度实数进行四则运算时将导致计算结果不正确。

(2) 若参加运算的双精度实数带有小数，只有当小数部分为0.5或0.25，或0.125，或0.0625等等，才可与单精度进行混合运算，否则应在单精度数字后加上#号才能避免失误。如上面例子中？1.2/1.2#↘改为？1.2#/1.2#↘才能得到正确的结果。读者也可以再键入？1.2*1.2#↘与？1.2#*1.2#↘比较两者结果的差别。

(3) 双精度实数与整数的混合运算不会造成失误。

4．双精度实数经加、减、乘、除、括号等运算后结果为整数或双精度实数但经乘方运算后变为整数或单精度实数。例如键入？1.2345#↑2#↘屏幕显示1.52399，为单精度实数的定点数显示，而实际若键入？1.2345#*1.2345#↘(注意不能少写#号，否则结果不正确)，结果为1.52399025。

5．双精度实数经大多数数值函数运算后均变为整数或单精度实数，唯有绝对值函数及取整函数才能保留仍为整数或双精度实数。详见函数功能部分。

（五）数值常数浮点数方式的一般表示法

1．单精度浮点数的一般表示式为±n.nnnnnE±e。

其中第一个±号表示数的正负符号，第二个±符号称为阶符(阶码的符号)，＋号表示正阶，－号表示负阶。n为构成有效位数的数字0～9。输入浮点数时，n的个数没有限制，小数点位置任意，但计算机输出时，浮点数中n的个数不超过6个，小数点的位置(若有小数点)总是在第一个有效数字之后。

E为乘幂的底，等于*10↑。e为阶码，阶码必须是整数。E±e表示一个以10为底的整数指数幂。

2．双精度浮点数的一般表示式为±n.nnnnnnnnnnnnnnnD±d。

第一个±符号表示数值的正负，第二个±符号表示阶符(阶码的正负)。n为构成有效位数的数字0～9。输入浮点数时，n的个数没有限制，小数点位置任意。但计算机输出时，浮点数中n的个数不超过16个，小数点位置（若有小数点）总是在第一个有效数字之后。

D为乘幂的底，等于*10↑。d为阶码，阶码必须是整数。

第三节　变量功能

一、数值变量

在LASER—310计算机中，变量也具有三种精度等级：整数型变量(下称整型变量)、单精度实数型变量(下称实数变量)和双精度实数型变量(下称双精度实数变量)。

在第一册介绍格式符号键以及本章常数功能中已经说明整数型、单精度实数型、双精度实数型的常数可以分别加上尾标%，!，#。对于变量，只能在整型变量后面加上尾标%。对于不加尾标%的变量，通常为单精度实数型变量。例如键入 A=10/3:？A↘屏幕显示 3.33333；而键入 A%=10/3:？A%↘则屏幕显示3（若把尾标！或＃加在变量名后面，在LASER—310计算机中会导致错误)。采用加尾标的方法不能得到三种精度等级的变量，但用别的方法可以得到。请看下面的例子：

键入　POKE30977，8:A=10/3#:？A↘则屏幕显示 3.3333333333333333

键入　POKE30977，4:A=10/3#:？A↘则屏幕显示 3.33333

键入　POKE30977，2:A=10/3#:？A↘则屏幕显示 3

3个例子中都使用了 POKE 语句指令（把机器码写入计算机内存)。有关POKE语句的详细说明，参见本书第五章。这里仅作简单解释，以说明上面的例子。第一个例子先用 POKE 语句指令把机器码 8 写入内存中地址为 30977 的存贮单元，变量A就成为双精度实数变量；第二个例子先用 POKE 语句指令把机器码4写入内存 30977 的地址中去，变量A成为单精度实数变量；第三个例子再把4换成2，变量A成为整型变量。

我们知道整型数占内存2个字节，单精度实数占内存4个字节，双精度实数占内存8个字节。因此上述方法的实质就是分别用2,4,8,三个机器数码来规定变量的精度等级。当把数码8写入内存30977中去时，凡是以字母 A为开头的所有变量（如 A1,A2,AA,AB,AX,AY等等)和数组(如A(1)，A(2)，AB(10)等，详见数组变量的说明)，都成为双精度实数型；当把数码4或2写入该内存地址中去时，则以字母A为开头的所有变量和数组都成为单精度实数型或整数型。

对于以字母B为开头的变量，要改变其运算的精度，应在内存地址 30978 中写入相应的数码。对于以其它英文字母为开头的变量，其精度值所在的相应内存地址可按字母的字典排序次序类推，或用式子 30912＋变量第一个英文字母的ASCⅡ码，来求出其精度值所在的内存地址(从30977～31002)。

在刚开机时（或使用过RUN、NEW等操作指令，CLEAR 语句指令、修改、编辑过程序之后)，在上述变量精度值的内存地址中，精度值均为4。例如键入：

FOR I=0 TO 25:？PEEK(30977＋I)；:NEXT↘

则屏幕显示26个数字，每个数字都是4。上述操作使用了FOR…TO…NEXT 循环语句(详见第一册) 和 PEEK 读取内存机器数码的函数指令（详见第五章)。其意义是读取内存地址 30977～31002 中每个地址中机器码。若用前述 ①、②、③ 三个例子操作后，再分别用这个例子操作一下，可以验证在30977内存地址中数码的变化情况。从这个例子中可知在

通常情况下，变量是以单精度实数的形式进行运算的。

下面对每类精度等级的变量在赋值、运算中的情况进行说明。

1. 整型变量

获得整型变量的方法有二种。一种是在变量名后面加上尾标%，另一种是用 POKE 语句指令把数码2写入内存中该变量首字符所在的精度值地址，其形式为

POKE30912＋ASC(“变量首字符”)，2↘

(1) 对整形变量赋值时，不管是用整数、实数、双精度实数进行赋值(数值范围必须在 －32768～32767之间)， 整形变量都只取其整数部分，而略去其正小数部份，并以整数形式存入内存(占2个字节)。

例如： 键入 A%＝3.14159:？A%↘则屏幕显示3

键入 A%＝－3.14159:？A%↘则屏幕显示 －4 (因为 －3.14159＝－4＋0.85841，其中正小数0.85841被略去)。

(2) 整形变量进行四则运算的结果若仍在整数范围内时（数值范围在 －32768～＋32767，且不带小数点)，结果为整数；否则结果为单精度实数。

例如： 键入 A%＝10:B%＝7:？A%/B%↘则屏幕显示1.42857。

2. 单精度实数变量

单精度实数变量是LASER－310 计算机系统自行设定的， 除非在变量首字符所在的精度值内存地址中， 表示精度的数码已不等于4， 才需要重新设定。 重新设定的方法很多，如可用POKE 30912＋ASC(“变量首字符”)，4↘，或键入CLEAR↘（详见第一册，或RUN↘，NEW↘ 等)，甚至键入任一数字（最好不要等于程序的语句行号，否则该行语句被删去， 详见第一册)， 如键入0↘， 都会使变量的精度返回系统设定状态。 但用POKE语句指令只改变某一个字母开头的变量，而用其它方法将使所有变量都回到单精度状态(包括关机后重新开机)。

(1) 对单精度实数变量赋值时，不管是用整数、实数、双精度实数进行赋值，单精度实数变量均取其有效数字至多6位，超过6位的四舍五入，并以单精度实数的形式存入内存(占4个字节)。

用不超过6位有效数字的整数（如±999999) 赋值给实数变量时，不会带来误差，用不超过6位有效数字的数，且小数部分为0.5；0.25；0.125；0.03125;0.015625;0.0078125;0.00390625等赋值给实数变量时，也不会带来误差(包括用浮点数方式赋值)，用其它数值赋值给实数变量时，可能带来误差。

例如键入B＝0:FOR A＝0.1 TO 0.7 STEP 0.1:B＝B＋1:NEXT:？B↘

这个操作的意见是先使B＝0，然后用循环语句(详见第一册)使变量A从0.1增加到0.7。每次增加0.1。当A＝0.1 时，B＝0＋1＝1；当 A＝0.2 时，B＝1＋1＝2；…当 A＝0.7 时，B＝6＋1＝7。按理循环结束后 B＝7，但显示出来的B却为 6，这是因为 0.1 这个数存放在内存中时，其二进制表示有误差（象 0.5，0.25 等以及整数不会造成误差）在运算中误差逐渐积累，当 A＝0.6 时，再增加0.1，在内存中二进制表示实际数值已超过0.7，因而最后一次循环没有执行，即B＝6＋1这个语句没有执行，因此显示B的值时只显示 6。

(2) 单精度实数变量进行四则运算的结果或者为整数，或者为单精度实数。

3. 双精度实数变量

获得双精度实数变量的简便方法有一种。即键入POKE 30912 + ASC("变量首字符"), 8↘。

(1) 对双精度实数变量进行赋值时，不管是用整数、实数、双精度实数进行赋值，双精度实数变量均取其有效数字至多16位（即1～16位），超过16位的四舍五入，并以双精度实数的形式存入内存(占8个字节)。

用不超过16位有效数字的整数赋值给双精度实数变量时不会带来误差，用不超过16位有效数字的数，且小数部分为0.5；0.25；0.125，…$m/2^n$($n=1,2,\cdots$ $m=1,2,\cdots$) 等值赋值给双精度实数变量时，也不会带来误差，用其它数值赋值给双精度变量时，最好是双精度型的实数，以减小误差。

例如键入：POKE 30912 + ASC ("A"), 8:A1 = 10/3:A2 = 10/3# ? A1:? A2↘则屏幕显示3.333333253860474及3.33333333333333两个数。第一个数（变量A1 表示）是把10/3这个单精度实数(且小数点部分又不是 $m/2^n$ 的形式)赋值给双精度变量A1，结果带来了明显的误差。第二个数是把10/3#这个双精度实数（见常数功能中各种精度等级的数值常数的混合运算部分)赋值给双精度变量 A2，结果与实际情况一致。

(2) 双精度实数变量进行四则运算的结果，或者为整数，或者为双精度实数。但经乘方运算后，其结果或者为整数，或者为单精度实数，情况与常数功能一致。

4. 各精度等级的数值变量的混合运算

在同一运算式中各精度等级的数值变量进行混合运算，其出现的各种情况和使用中避免差错应注意的几个问题，均与常数的功能中介绍的情况一致。

双精度实数变量经数值函数后，除绝对值函数及取整函数运算的结果为双精度实数外，其它函数的运算结果为整数或单精度实数(详见第一册函数功能部分)。

第四节 函数功能

一、串函数的深入剖析

1. ASC(字符串)与CHR $(算术表达式)

这两个函数互为反函数。其功能如下：

(1) ASC(字符串)：字符串至少必须包含1个字符，若有1个以上字符时，仅第一个字符有效。字符串若为常量，必须用双引号括起来。例如键入？ASC("1")↘，则屏幕显示␣49，再键入？ASC("12")↘，屏幕也显示␣49，这就证明了此函数只取第一个字符的ASCⅡ码。由于空串是没有ASCⅡ码的，所以若键入 ？ASC(" ")↘，则屏幕显示FUNCTION CODE ERROR(意为函数码出错)。

利用上述操作方法，可以把键盘的每一种键入字符的ASCⅡ码显示出来，如空格的ASCⅡ码是32，可键入？ASC("␣")↘，则屏幕显示␣32。

从屏幕显示的特点可知ASC函数的输出结果为数值（数值在显示或打印时，前后均留格，数值前面的空格留作数值的正、负符号显示用，后面的空格留作与下一个数值的间隔空用，例如键入？1；2↘，则屏幕显示␣1␣␣2␣，若键入？－1；－2；↘则屏幕显示－1␣－2␣）。

ASC函数在前面取变量的精度时曾经使用过，这是利用了ASC函数的数值性质。因而30912＋ASC("A")等于30912＋65，即为30977。这仅是ASC函数应用的一个例子。在程序设计中，当需要实现字符与数码的交往时（例如做字母数字游戏、按字母进行检索、排序等），都可以利用ASC函数。

使用ASC函数时，可以把ASC函数作为数值与数值常量、变量、函数进行各种数学运算，以实现字符对数值运算的控制。

(2) CHR$(算术表达式)

与ASC函数功能相反，CHR$函数只能把算术表达式的值转换成1个字符，所以算术表达式的值必须在0～255数值范围内。由于ASCⅡ数码均为整数，因而算术表达式的值若为实数时，计算机将自动取整。由于算术表达式是表示数值的，因此不能用双引号括起来，初学者应特别注意。例如键入？CHR$(49)↘，则屏幕显示1（注意1的前面没有空格）。若键入？CHR$(49.8)↘。屏幕也显示1。

利用上述操作方法，可以把0～255每一个ASCⅡ数码转换成相应的字符（有一部分字符仅作控制码用，不能显示出来，称为非显示字符）。

为了说明算术表达式在CHR$函数中的应用，再举一些例子如下：

键入　？CHR$(4*16＋2)↘　　则屏幕显示B（B的ASCⅡ码为66）

键入　A＝48:？CHR$(A)↘　　则屏幕显示0（0前面没有空格）

键入　A＝10:？CHR$(SQR(2480)＋A*2)↘　　则屏幕显示E(E的ASCⅡ码为69)

键入　？CHR$(ASC("A"))↘则屏幕显示A，这个例子表明CHR$函数与ASC函数互为反函数，它们形成复合函数时，两者的作用互相抵消（也可以键入？ASC(CHR$(65))↘，屏幕显示␣65），而且表明算术表达式也可以包含函数值为数值的串函数。

从屏幕显示的特点可知，CHR$函数的输出结果为字符(不管是数码字符或其它字符，在显示或打印时，前后均不留空格)，因而CHR$函数只能作为字符使用，可以与字符串(常量、变量、函数)进行联结和比较，例如键入CHR$(65)＋"B"↘则屏幕显示AB；键入？CHR$(65)＝"A"↘，则屏幕显示－1，因此CHR$函数也能够实现数码与字符的交往，利用数值运算扩大字符串操作的功能。此外，CHR$函数还能把一些ASCⅡ控制码(非键入字符)存放在函数中或串变量中，增强字符串的功能。

2. VAL(数码字符串)与STR $(算术表达式)

这两个函数也是互为反函数，其功能如下：

(1) VAL(数码字符串)

数码字符串可以是数码组成的串常量（必须加双引号），例如键入？VAL("1.23")↘则屏幕显示␣1.23␣其显示方式具有数值的特征。数码字符串也可以用串变量来表示，例如键入A$＝"3.14159":？VAL(A$)↘，则屏幕显示␣3.14159␣。数码字符串也可以

用函数值为字符串的串函数表示，例如键入？VAL(CHR$(49))↘由于CHR$(49)表示数码字符串"1"，故屏幕显示␣1␣。

在数码字符串中，数码可以带正、负符号，而输出为的数值仅负号显示，例如键入？VAL("+1.23")；VAL("-1.23")↘，则屏幕显示␣1.23␣-1.23␣。

数码可以写成整数形式(但不可带%号，否则出错)，而输出为双精度数值。例如键入？10/VAL("3")↘则屏幕显示␣3.3333333333333333

数码若写为单精度实数形式，必须写成浮点数方式或定点数方式尾标！输出才为单精度数值；若用定点数方式且不带尾标！则VAL函数输出结果为双精度数值。例如键入？VAL("1.234567E6")↘或键入？VAL("1234567!")↘则屏幕均显示1.234567E+06而键入VAL("1234567")↘，则屏幕显示1234567。又若键入？10/VAL("1.2")↘，则屏幕显示8.3333333333333333

数码若写为双精度实数形式，不论是写成定点数方式(可以带尾标也可以省略)还是写成浮点数形式，输出结果均为双精度数值，例如键入？VAL("12345678")↘则屏幕显示12345678。

VAL函数中的数码字符串还可以用+号联结起来的方式，例如键入？VAL("1"+"2")↘则屏幕显示␣12␣，若键入？VAL("1"+"E"+"2")↘则屏幕显示␣101␣。

VAL函数与STR$函数配合使用，可以使数值运算的功能增强，利用字符串操作来完成某些数值运算较难完成的任务，如可以使单精度变量进行双精度运算等。

顺便指出，若数码字符串为空串或者是以非数码字符开头的，其输出结果均为数值0(除以%开头的将出语法错误外)，例如键入？VAL("")↘或？VAL("$123")↘或VAL("A14")↘等，屏幕均显示␣0，在使用中应加以注意。

(2) STR$(算术表达式)

与VAL函数的功能相反，其结果是把算术表达式的值转变成数码字符串，算术表达式可以为数值常数的形式，例如键入？STR$(5)；STR$(7)↘则屏幕显示␣5␣7，注意它们的显示方式与数值显示(可键入？5；7↘，屏幕显示␣5␣␣7␣)的微小差别。即虽然保留了数码前面的一个空格，但去除了数码后面的一个空格，其结果仍为字符串性质。若键入？STR$(5)+3↘将出现类型错配的提示。

上述例子告诉我们，STR$把数值转变成数码字符串的时候，将保留数码前面的符号位，这还可以用键入？LEN(STR$(5))↘屏幕显示␣2␣(表示数码字符串STR$(5)含有2个字符，其中第一个字符是空格，第2个字符才是数码5，再键入？ASC(STR$(15))↘则屏幕显示32(空格的ASCⅡ码)

这种现象的存在，在使用STR$函数时应该引起注意，例如在数制转换中，需要把一系列的十进制数化为16进制数，设某一个十进制数转化为16进制数结果为C5A。若分别用串变量A1$，A3$分别存放第一个与第三个字符，用变量B存放数码5，则打印的结果成为C␣5␣A(键入 A1$="C":B=5:A3$="A":LPRINT A1$；B；A3,$↘)，三个字符不能连在一起，为了使打印出来的16进制数码都整齐划一(如根据数值范围要求每个16进制数码都占3个字符位置)；若把数码5转变为数码字符串，如键入 A1$="C":B=5:A2$=STR$(B):A3$="A":LPRINT A1$；A2$；A3$↘，则打印的结果成为C␣5A，仍然没有达到要求。

但由于我们已经注意到这种现象，就可以设法去掉数码5前面的空格（可以有几种方法），例如键入A1$="C":B=5:A2$=RIGHT$(STR$(B),1):A3$="A":LPRINT A1$;A2$;A3$↘，则打印结果成为C5A，达到预定的目的（其中A2$部分也可以写为：A2$=MID$(STR$(B),2)或写为A2$=CHR$(8)+STR$(B)等，都可以达到同样的目的，RIGHT$，MID$的功能前已简单介绍。CHR$(8)是一个控制字符，详见第一册附录）。

上述例子是STR$函数应用中遇到的一个问题，也是其应用的一个举例。

STR$函数中的算术表达式写为数值常数形式时，若为整数(可带尾标%)或双精度实数(可带尾标#)的形式，则输出的数码字符串仍保留"整数"或"双精度实数"的形式，例如键入？STR$(1.2345678)↘，则屏幕显示␣1.2345678。键入？STR$(123D16)↘，则屏幕显示␣1.23D+18。

若数值常数为单精度实数（可带尾标!），则输出形式不管是定点数方式还是浮点数方式(按常数功能中所述的规定)，最多取6位有效数字，例如键入

？STR$(1.234567)↘，则屏幕显示␣1.234567(最末一位已四舍五入)。

这个现象在使用中应当特别引起注意，若遇到七位数字(如发票号码等)需转为数码字符串的时候，为了保留全部有效数字，应采取双精度方式转换，如键入

？STR$(1234567#)↘，则屏幕显示␣1234567

STR$函数中的算术表达式写为变量、函数，及其运算式时也要注意上面所述的同样的问题。

STR$函数与VAL函数配合使用，可以使整型或单精度实数变量具有双精度实数变量的功能。例如键入A=1.2:？8/A；8/VAL(STR$(A))↘则屏幕显示两个数6.66667与6.666666666666667，前一个数表示8/A的结果，为单精度实数，后一个数则通过串函数的转换，使变量A的值成为双精度。应用中仍需注意到单精度变量在赋值时其有效数字不要超过6位，否则会带来意外的误差。

STR$函数与VAL函数在数值范围上比ASC函数和CHR$函数更扩大了字符串与数值之间的交流，在程序设计中有许多重要的应用，例如在LASER—310计算机中，数值运算的结果最多为16位有效数字，要达到更多的有效数位则数值运算已无能为力（例如要计算两个15位数字乘积的准确值，保留每一位有效数字），但利用STR$函数、VAL函数、MID$函数及字符串的联结功能，把两个数进行分段运算，再把结果一段段联结起来，就可以达到预定的目的。

3. LEN(字符串)

字符串所含的字符数为0～255个，因此LEN函数的值为0～255，并称为字符串的长度，空串不含任何字符，因此若键入？LEN("")↘则任幕显示␣0␣，长度函数的结果具有数值性质。因而它可以与数值(常量、变量、函数)进行数值运算，达到利用字符串的长度来控制某些数值运算，或控制某些字符串操作。

字符串可以是串常量的形式，如键入？LEN("5")↘,则屏幕显示结果为1。也可以是串变量的形式，如键入A$="5":？LEN(A$)↘，则屏幕显示结果同上。也可以是串函数(函数值必须为字符串)，如键入？LEN(CHR$(65))↘，则屏幕显示结果为1。若键入

? LEN(STR$(5))↘，屏幕显示␣2␣(其原因已在STR$ 串数功能中说明)。这个例子说明，LEN函数还能计算出非显示（及非打印）字符的个数，对于诊断某些串函数的功能起了很大的作用。又如，键入 ? LEN("A"+CHR$(13)+"B")↘，屏幕显示␣3␣，其中一个字符就是控制字符，参见第一册附录。ASCⅡ数码13表示回车，例如打印(或显示)很长的一个字符串时，可以在需要换行的字母处加上此控制符，这个例子说明，LEN 函数中的字符串还可以采用联结的方式。

LEN 函数在程序设计中也是很有用的。例如用 LEN 函数测出某些数码字符串的长度，以决定分段的方式，或在设计人员名单时，对输入计算机的人名（若用汉语拼音）先测出其长度，由于各人姓名所占用的字符数长度不等，可以在长度较短的人名后面添上空格，使每个人的姓名都占用同样的字符数，在显示或打印时就能做到表格化，在人名检索或其它项目检索(如地址、电话、或课程、成绩等)时也不易搞错。

4. LEFT $(字符串,n))MID $(字符串,m,n))RIGHT （字符串，n)

这三个函数都是从字符串中取出限定的字符，得到新的字符串，是字符串联结的一种逆运算，通过这三个函数，可以把字符串拆成几段。

其中字符串可以包含0～255个字符，字符串为空串时，三个函数的结果均为空串，字符串可以是串常量、串变量、串函数(函数值必须是字符串)及其联结的形式。括号内的数n及m，可以是数值常数，也可以是数值变量或函数(函数值必须是数值)，n或m的取值范围为0≤n≤255，1≤m≤255

(1) LEFT$(字符串，n)

从字符串的第一个字符(包括非显示或非打印字符)起，取出包括第一个字符在内的 n 个字符，当n=0时，取出空串；当 n≥LEN（字符串)时，把整个字符串原封不动地取出来，n不可以省略(否则出错)，字符串与算术表达式n之间必须用逗号分隔。例如键入A$="LIU␣␣␣GUANG␣LIN␣␣␣":? LEFT$(A$, 5)↘ 屏幕即显示LIU␣␣ 3个字母。这个例子是用串变量 A$ 存放人姓名的汉语拼音。用左取串数函数LEFT$把A$（刘广林)中的姓取出来(刘)。这个例子在实用中可以实现人名的按姓分类或检索。

(2) MID$(字符串，m，n)

从字符串的第m个字符起（包括非显示或非打印字符），取出包括第m个字符在内的n个字符。当m=1时，MID$函数就能代替LEFT$函数取字符的功能；当m>LEN（字符串)时，取出空串；当n=0时，也取出空串；当n≥LEN（字符串）－m≥0时，则从第m个字符起一直取到最后一个字符；当n省略时，写成 MID$（字符串，m)，也是从字符串的第m 个字符起，一直取到最后一个字符。

仍采用上述例子，如键入? MID$(A$,1,5)↘则结果与 LEFT$(A$, 5)相同。如要取出人的名字(不取姓)，可键入 ? MID$(A$,7)↘则屏幕显示GUANG␣LIN␣␣␣8个字母。

串函数 MID$ 在程序设计中有着广泛的应用，前面在打印 16进制数码的例子中就曾经使用过这个函数。此外，利用MID$ 函数对数码字符串的操作，可以简化数码排列方面的数学问题，也可以根据数码的特征在数码检索(如查电话号码、发票号码、对奖号码等)中发挥其作用。

(3) RIGHT$ (字符串，n)

从字符串的最后一个字符起，倒取n个字符(包括最后一个字符)，但取出字符排列顺序仍按照原来字符串中的排列。当n = 0时取出空串，当n≥LEN(字符串)时，取出字符串的全部。

仍采用上面的例子，若键入？RIGHT$ (A$,12)↘ 则结果与 MID$ (A$,7) 完全相同，因此右取串函数RIGHT$也是完全可以用 MID$函数代替。

三个取字符串函数的相互关系可以表示如下：

LEFT$(字符串，n)等于MID$(字符串，1,n)

RIGHT(字符串，n)等于MID$ (字符串，LEN (字符串) - n + 1) (注n≤LEN (字符串))

5. INKEY$

这是串函数中的一个特殊函数。这个函数没有自变量(其它串函数都有自变量)调用这个函数时，它就扫描一次键盘(所需时间极短)，在扫描键盘期间若有键输入，就把键入的任何一个字符或单键指令(实际为ASCⅡ码，参见附录)立即作为 INKEY$ 的函数值 (函数值只具有字符串的性质)， 且函数值只含有一个字符或一条单键指令， 即 INKEY$ 函数中只能存放1个ASCⅡ码；若在扫描键盘期间没有键输入， INKEY$ 並不等待，扫描结果其函数值为空串。INKEY$接受键输入时，不需要按输入键(即↘)。

为了说明INKEY$上述特点，可键入下面的程序加以说明。

```
10  A$ = INKEY$  ↘
20  IFA$<>“”THEN PRINT“ASC(“; A$;“) = ”; ASC(A$)  ↘
30  GOTO 10  ↘
```

再键入RUN↘启动程序，即可立即看到屏幕显示

```
ASC(  ) = ␣1
ASC(
) = ␣13
```

再按任一键，例如按A键，则屏幕显示

```
ASC(A) = ␣65
ASC(A) = ␣65
```

下面结合上述现象说明程序的意义和使用INKEY$函数的注意事项。

10,20,30等数码都是程序的语句行号，10号语句用赋值方法调用 INKEY$ 函数，即 INKEY$ 函数把键入的字符或单键指令“扫描”输入并赋值给串变量A$，若没有键输入时就把空串作为函数值赋值给A$；20号语句用条件语句IF…THEN判断串变量A$中存放的是否为空串；若不是空串，就会屏幕显示该字符串的 ASCⅡ码，在PRINT语句中，有四个输出项，各项用分号分隔，带引号的部分是串常量，作输出说明用，输出项中 A$ 即为键入的字符 (包括单键指令，非显示字符)，ASC(A$)即为键入字符的 ASCⅡ码；若 A$ 中存放的是空串，则THEN后的语句不执行，而执行30号语句， 30 语句用GOTO转向语句指令令计算机再执行10号语句，若键盘没有键入，则计算机就往返执行10～30语句而不显示任何内容。 有关语句行号、 条件语句、 转向语句以及屏幕显示语句(即PRINT，也

即？）等详细内容和写程序、运行程序(RUN)等操作，详见第一册（每个语句行号后的回车符↵是写程序时输入的标志）

为了说明程序的执行过程，先键入POKE 31003，1↘（POKE 语句指令的意义和详细意义见第五章，此处是使计算机执行程序时同时显示语句的行号）。

再键入RUN↘启动程序，並迅速按A键和中断键，可看到屏幕显示

〈10〉〈20〉ASC(　) = ␣1

〈30〉〈10〉〈20〉ASC(

) = 13

〈30〉〈10〉〈20〉ASC(A) = ␣65

〈30〉〈10〉〈20〉ASC(A) = ␣65

〈30〉〈10〉〈20〉〈30〉〈10〉〈20〉〈30〉〈10〉〈20〉…

以上显示说明在启动程序的瞬间，键操作(↘)已作为键输入被INKEY$接受，在按键的短短瞬间内，程序2次执行了第10号和20号语句，第一次执行10号语句时 INKEY$扫描一次键盘，把回车键的输入（ASCⅡ码为1）作为函数值赋值给A$，ASCⅡ数码1是非显示字符，因而在 ASC 后的括弧内为空格，第二次执行 10 号语句时，INKEY$ 又扫描一次键盘，把回车键的输入（ASCⅡ 码为13）作为函数值赋值给 A$，ASCⅡ数码 13 也是非显示字符，并使其后的输出项换行返回行首显示（若按 ↘ 键时极为迅速轻巧，则ASCⅡ码13不出现；若按↘键时稍慢，则 ASCⅡ码13可能出现2次或2次以上），由此也可知执行程序的操作指令为 ASCⅡ码1，ASCⅡ码13是↘键兼有的功能。）此后键入的字符A，也是由于两次调用了 INKEY$ 函数，在按A键的瞬间两次扫描键盘而得到的结果，(若按键时速度稍慢，可能出现2次以上的 ASC(A) 显示）运行上面的程序还可以证明INKEY$还能接受单键指令（包括函数指令、操作指指、语句指令），並显示各指令的ASCⅡ码(还可以知道CTRL 键，SHIFT 键，FUNCTION 键，INVERSE 键等都不能键入 ASCⅡ 码)。

根据以上说明，可以了解到INKEY$函数有如下几个特点：

(1) 调用INKEY$函数时，计算机扫描键盘，若键盘没有键入，则INKEY$取值为空串，若键入具有ASCⅡ码的字符或单键指令，则INKEY$取值为相应 ASCⅡ 码的字符或单键指令，INKEY$取值不需要按↘键。

(2) 调用INKEY$函数时，计算机並不等待键输入

(3) INKEY$函数只能存放1个 ASCⅡ码的字符或单键指令，第2次存放字符或单键指令时，前一次存放的ASCⅡ码被取代。与变量存放数据相似 (1个变量一次只能存放1个数据)

(4) 当INKEY$放在程序中(特别是程序的第1句) 时，启动程序时的键操作（ASCⅡ码为1)将成为INKEY$的第1个值。

使用INKEY$函数的注意事项：

(1) 避免启动程序的键操作成为INKEY$的函数值(或设法消除此函数值)。

(2) 避免按键时(通常按键时速度不可能很快)让INKEY$取2次或2次以上的值。

要做到上述两个要求，调用 INKEY$ 函数才能避免失误，达到上述两个要求需要考虑程序的设计技巧，方法是很多的，详细内容可参见第三册中的一些程序举例。

二、逻辑函数：布尔函数的运算规律

布尔函数也是利用逻辑运算符来进行运算的，不过运算的对象不是关系表达式，而是算术表达式。三个逻辑运算符 NOT，AND，OR 读法不变，它们实际上都取自布尔函数。在 LASER—310 计算机上，布尔函数只有这三种，并且在对算术表达式的逻辑运算中，只取算术表达式的整数值进行运算，且算术表达式的值不能超过 LASER—310 整数范围(－32768～＋32767)否则会造成数据溢出。

布尔函数对整数的逻辑运算，实质是对整数的二进制码进行逻辑非、逻辑与、逻辑或的运算，分别介绍如下：

1．逻辑非函数：NOT算术表达式

在本章开始部分的常数功能中，曾经介绍了整数在 LASER—310 计算机中的二进制表示法。即每一个整数占内存2个字节，每个字节用8位二进制表示。因此每个整数用16位二进制表示。例如十进制数1是用二进制0000,0000,000,0001表示的，整数2是用二进制码000,0000,0000,0010表示的等等。在介绍整数的表示法时，还介绍了原码、补码和反码。在LASER—310计算机上，负整数是用补码方式表示，补码是取原码的反码再加上 1，例如－1，用补码方式表示，就是先取－1的原码(等于－1的绝对值，即＋1)，再取原码的反码，即把二进制数码中的0变为1，1变为0，得到二进制码 1111,1111,1111,1110，最后再加上1，因此－1在计算机内的二进制码就是1111,1111,1111,1111，显然－1这个数正好是0反码。事实上，逻辑非函数就是取反码的逻辑运算，因此若键入 ？NOT 0↘，屏幕就显示－1，当键入？NOT－1↘，屏幕就显示0，这种运算排成竖式可以写为

```
0:        0000,0000,0000,0000        -1:       1111,1111,1111,1111
NOT 0:    1111,1111,1111,1111        NOT -1:   0000,0000,0000,0000
```

逻辑非函数並不限于对整数0和－1的运算。例如键入 ？NOT3↘屏幕显示－4；若键入？NOT 32767↘则屏幕显示－32768，其算式如下：

```
3:        0000,0000,0000,0011        32767:    0111,1111,1111,1111
NOT3:     1111,1111,1111,1100        NOT32767: 1000,0000,0000,0000
```

如何看出NOT3的二进制码表示－4？由于－4＋3＝－1，所以当写为二进制形式后，得到111,1111,1111,1100＋0000,0000,0000,0011＝1111,1111,1111,1111我们就容易明白了。如果在－1的二进制编码上再加上1，如下：

```
          1111,1111,1111,1111
    +     0000,0000,0000,0001
    ------------------------
     [1],0000,0000,0000,0000
自然丢失↗
```

即得到 0，这是计算机用补码方式表示负整数的一种特有现象。同样的道理，容易明白

NOT32767的二进制编码表示－32768(把32767与－32768的两者的二进制编码按位相加，可得到－1的二进制表示)。

最后应指出，逻辑非函数对整数的二进制编码的运算规则(NOT 0结果为1，NOT1结果为0)不同于逻辑非函数对整数的运算，两者切勿混淆。

2．逻辑与函数：算术表达式AND算术表达式

逻辑与函数对整数的二进制编码进行逻辑运算的规则是：

1 AND 1 结果为 1；　　1 AND 0 结果为 0；

0 AND 1 结果为 0；　　0 AND 0 结果为 0。

根据上述规则，可以用笔算方法求出两个整数的“逻辑与”运算结果，再用机器的运算结果来证明。

例如：求 6 AND 3

解：先把整数化为二进制形式，再进行 AND 运算如下：

```
    6:  0000,0000,0000,0110
    3:  0000,0000,0000,0011
-------------------------------------
6 AND 3:  0000,0000,0000,0010  =2 (十进制)
```

再用机器证明上述结果：键入？6AND3↘屏幕显示2

根据整数的二进制编码表示法，负整数的二进制编码首位用 1 表示，正整数的二进制编码首位用0表示，因此正整数与正负数进行逻辑与运算的结果必为正整数或 0，0 与任一整数进行逻辑与运算的结果必为 0；负整数与负整数进行逻辑与运算的结果必为负整数。

例如，求 6 AND－5，其算式为：

```
    6:  0000,0000,0000,0110
   -5:  1111,1111,1111,1011
-------------------------------------
6 AND -5  :  0000,0000,0000,0010 = 2(十进制)
```

再键入？6 AND－5↘屏幕显示 2

利用逻辑与函数对整数的逻辑运算，可以很方便地使计算机把整数显示为二进制数码，其程序详见本书第三册的条件语句举例。

3．逻辑或函数：算术表达式OR算术表达式

逻辑或函数对整数的二进制编码进行逻辑运算的规则是：

1 OR 1 结果为 1；　　1 OR 0 结果为 1

0 OR 1 结果为 1；　　0 OR 0 结果为 0

根据上述规则，也可以用笔算的方法求出两个整数的“逻辑或”运算结果，并用机器的运算结果来证明。

例如，求 6 OR 3，其算式如下：

6：0000,0000,0000,0110

3：0000,0000,0000,0011

6 OR 3：0000,0000,0000,0111＝7（十进制）

再键入？6 OR 3↘则屏幕显示7。

根据逻辑或的运算规则可知，负整数与任何整数(包括0)进行逻辑或运算的结果必为负整数，其中－1与任何整数进行逻辑或运算的结果必为－1，只有正整数与正整数或0进行逻辑运算的结果才为正整数。

例如，求 6 OR－5，其算式如下：

6：0000,0000,0000,0110

－5：1111,1111,1111,1011

6 OR－5：1111,1111,1111,1111＝－1(十进制)

4．上述三种基本的布尔函数还可以用逻辑运算符连起来组成较为复杂的函数，其运算次序与逻辑表达式中介绍的规则相同，参见第一册和第二册函数功能部分

5．布尔函数的值在整数—32768～32767范围内，不管怎样运算也不会超出此范围，布尔函数的值可以赋值给数值变量。

第二章　LASER-310使用的BASIC语言（基础部分深入）

第一节　对循环语句的深入说明

一、循环变量

循环变量必须是数值变量，并且只能是单精度实数变量或整型变量．不可以使用双精度实数变量、串变量、数组变量(包括数值型或字符串型)，循环初值表达式，循环终值表达式、步长表达式的运算结果必须是数值，其数值范围为 -10^{38}～10^{38}，精度等级任意，但不允许使用尾标#，可使用定点数方式，也可使用浮点数方式（两种方式均可使用双精度实数，但由于循环变量不允许使用双精度变量，所以没有实际意义)。

二、循环语句执行过程

在 LASER—310 计算机中执行循环语句时，计算机首先算出循环初值、循环终值和步长(步长省略时计算机设定步长为 1)，把循环初值赋值给循环变量，根据循环变量的次序设定堆栈指针把循环终值和步长值存放在内存堆栈区中，循环变量的值存入内存变量区中，例如 FOR I=1 TO10:PRINT I；:NEXT 这样一个循环语句，计算机执行此语句时，先确定循环变量是 I，循环初值是1，终值是 10，步长是 1（由于没有表达式的一般形式，省去了计算表达式值的步骤，这种情况下初值、终值、步长都称为直接数)，把初值1赋值给循环变量 I，并在内存中设立FOR堆栈，把终值10，步长1存放在堆栈中，堆栈指针由循环变量I决定。然后计算机先执行一次循环体PRINT I；显示变量I的值1，接着执行 NEXT 语句。NEXT 语句根据其后的循环变量的名称设定堆栈指针，从堆栈区中依次取出步长和终值。若NEXT后面省略写循环变量时，则设立堆栈区时所设定的堆栈指针仍然保持不变，与NEXT后面写上正确的循环变量时情况相同（若NEXT后面写上错误的循环变量名时，堆栈指针发生错位，NEXT找到的堆栈区中没有内容就会出错停机)。然后NEXT语句根据步长的正负符号、循环变量的值与终值的大小来决定是继续环循（执行循环体)，还是结束循环（清除堆栈)，其过程是：当步长为正时，不管初值是否大于终值，先将循环变量的值加上步长，并赋值给循环变量，也即改变循环变量的当前值，再把循环变量的当前值与终值相比，如果当前值仍小于或等于终值，就往下执行循环体；如果当前值已超过终值，则结束循环，同时清除该 FOR 堆栈，往下执行NEXT之后的语句，NEXT之后没有语句时就结束程序的运行。当步长为负时，不管初值是否小于终值，先将循环变量的值减去步长的绝对值，并赋值给循环变量，使循环变量获得新的当前值，再比较循环变量的当前值与终值的大小，如果当前值仍大于或等于终值，计算机就往下执行循环体，如果当前值已超过终值(这里超过的意思是超过终值界限，即小于)，则循环结束，执行NEXT之后的语句。如上面例子中，执行一次NEXT语句，循环变量I的值就增加1，

循环体就接着显示这个循环变量的值，屏幕上显示的结果为

1 2 3 4 5 6 7 8 9 10

也就是当循环变量的值等于10时，计算机仍执行循环体，再往下执行NEXT时，循环变量的值成为11，计算机结束循环，这可以用下面的方法来证明：

键入 FOR I=1 TO 10:NEXT:PRINT I↘

则屏幕显示11

从计算机执行循环语句的过程来看， FOR 语句的功能主要是设立内存堆栈区域，兼有对语句中表达式的计算功能，赋值号仍起赋值作用。NEXT语句的主要功能是根据循环变量的堆栈指针取出堆栈的数据(步长和终值)改变循环变量的当前值，并判断循环体是否需执行，堆栈是否清除。

三、在执行循环体中，循环初值、环循终值和步长不能改变

例如：

```
10  A=1:B=10:C=2
20  FOR I=A TO B STEPC:A=10:B=1:C=-2
30  PRINT I; :NEXT
```

再键入RUN↘，程序执行后，屏幕显示 1 3 5 7 9 这几个数字，这就说明了上述结论。这是因为执行循环体中A=10:B=1:C=-2 这些语句时，只是在内存RAM的非堆栈区改变了这三个变量的赋值，而在堆栈区中循环终值10，步长 2 始终不变，循环变量 I 的值则根据初值1依次增加步长值变为3、5、7、9、11，当 I=11 时不执行循环体而结束循环。

四、循环变量的改变

在循环体中，循环变量的值不是不可改变的，但改变循环变量的值只能用在特殊设计的程序中，以满足某种特殊需要，程序设计中必须采取相应的措施，否则会造成循环次数的混乱以至出错。为了说明在循环体中改变循环变量的值及其对循环次数的影响，可举下面的例子来说明。

```
10  FOR I=1 TO 10:PRINT I; :IF I<5 GOSUB 50
20  NEXT:END
50  INPUT"I="; I:RETURN
```

键入 RUN 之后屏幕显示循环变量I的值为1，在循环体中的条件语句是当I<5 时调用50号语句写成的子程序，屏幕又显示"I="？若键入8，则屏幕接着又显示9、10两个数字，循环只执行了三次，这是因为键入8之后，循环变量的值成为8，子程序返回后执行20号语句 NEXT， 使循环变量增值 1，再执行循环体显示 9，此时I已大于 5 所以条件语句中的 THEN部分不执行 (此处省略了THEN这个指令)，往下执行到 NEXT，直至循环结束执

行 END 语句，使程序结束运行。

五、循环语句可用条件语句描述

设循环变量为I，循环初值用变量A表示，循环终值用变量B表示，步长用变量C表示，语句行号由LK(K=1,2,…)表示，则循环语句FOR I=A TO B STEP C:循环体：NEXT可用条件语句表示如下：

```
L1  A=循环初值：B=循环终值：C=步长值：I=A
L2      循环体
⋮           ⋮
LK      循环体
Lk+1  I=I+C (循环变量=循环变量当前值+步长)
Lk+2  IF C>0 AND I<=B THEN L2
Lk+3  IF C<0 AND I>=B THEN L2
Lk+4  程序其它语句
```

因此，循环语句完全可以用条件语句来代替，但循环语句具有一些独特的特点是条件语句无法取代的。第一，当需要采用循环来进行计算时，采用循环语句程序读起来明白清楚，循环体容易查找(尤其是多重循环，下面将谈到)，而采用条件语句就没有这个优点。第二，由于循环语句在内存中使用堆栈的方法存取数据，大大提高了运算速度，条件语句则望尘莫及。例如，前面我们提到运行下面一个循环约需1秒时间：

```
10  FOR I=1 TO 600
20  NEXT
```

现把它改为条件语句：

```
10  I=1:B=600
20  I=I+1 IF I<=B THEN 20
```

则运行时间约需5秒钟(如果把变量B直接写为600放在条件语句中也是一样)。

从以上条件语句的描述方法也可以看到，循环语句中循环初值与循环终值相等时，循环只执行1次，但不管是怎样的情况（步长为正时初值大于终值，或步长为负时初值小于终值)，循环体至少执行一次。这种情况在LASER-310中是如此，在其它计算机上不一定如此，有的计算机执行循环语句时，执行L1之后先进入Lk+2及Lk+3进行判断，再根据判断决定是执行循环体还是结束循环。

六、执行FOR…NEXT语句

执行 FOR…NEXT 语句时，FOR 语句执行1次，计算机即在内存中建立堆栈区，NEXT 语句可以执行的次数说明如下：NEXT 是根据循环变量的值超过终值(若步长为正）或者小于终值（若步长为负）时，即停止循环。一旦结束循环，堆栈区就被清除。因此，只要堆栈区尚未清除，NEXT 语句执行的次数就没有限制。但通常情况下，语句

NEXT语句的执行的次数即循环次数可按下式计算：

$$x = 1 + (\text{循环终值} - \text{循环初值})/\text{步长值}$$

当$x \leqslant 1$时，取循环次数为1；当$x > 1$时，取循环次数等于INT(x)，但要考虑步长的累计误差。所谓通常情况是指循环体中没有转向语句(包括GOTO，IF…THEN，或转子语句GOSUB)，使程序未执行到NEXT语句就转出循环，使循环不能完成到底(这种情况在程序设计中常常用到，下面有一个程序举例可以说明这一点)，但在这种情况下内存中的FOR堆栈仍然保留，因此转出后若要转入也是允许的(例如GOSUB转出之后从子程序返回，或用GOTO，IF…THEN转出之后再用GOTO或IF…THEN转入)。但是，一旦由NEXT语句转出循环，内存中的FOR堆栈被清除(或FOR语句尚未执行，内存中FOR堆栈尚未建立)，则不管从FOR语句之前或者NEXT语句之后，即从循环体外用GOTO，IF…THEN，或GOSUB语句不经过FOR语句直接转入循环体，都会发生NEXT WITHOUT FOR这种出错，下面举一个简单例子来证明这一点：

```
10  FOR I=1 TO 10
20  PRINT I; :IF I=10 THEN 50
30  NEXT
50  I=1:GO TO 20
```

程序运行后，先执行10号语句，把数值赋值给循环变量I之后，在内存堆栈区存入循环终值10和步长1，接着执行20号语句，即循环体语句。先显示循环变量的值1，再判断变量I的值是否等于10。若I=10则转向到50号语句，若I不等于10则按语句顺序执行30号语句。30号语句是NEXT语句，从内存堆栈中取出步长1和终值10，并把循环变量I的当前值1加上步长，则变量I成为2，再把变量I的值与终值10比较，由于步长，循环变量的值又小于终值，所以继续执行循环体，显示变量I的值2，…如此循环，直至当循环变量的值等于10时，转向到50号语句，50号语句重新给循环变量I赋值为1，并转入20号语句，显示循环变量I的新的值1，接着再执行NEXT语句，又重复上面的过程，这样无休止地进行下去，屏幕显示1,2,…10之后接着又显示同样的1,2…,10，这就说明了当内存堆栈未被清除时，NEXT语句执行的次数是没有限制的。如果把这个程序里面20号语句中的IFI=10 THEN 50这一句去掉，那么程序执行到I=10时不再转到50号语句，而继续执行NEXT语句，使变量I的值成为11(这时大于终值10)使循环结束。循环结束时，内存堆栈区清除。这时再往下执行50号语句，重新给变量I赋值后，转入循环体20号语句。20号语句仍然执行，显示变量I的值为1，再往下执行30号语句时即出错。

七、步长值不是整数，要考虑循环次数的累计误差

例如键入：

```
FOR I=0.1 TO 1 STEP 0.1:PRINT I; :NEXT↘
```

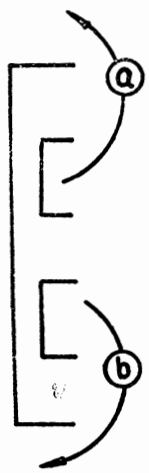
则屏幕显示0.1 0.2…0.9而最后一次循环没有执行，其原因已在常数功能中作了说明，不再重复。

八、循环体的堆栈指针

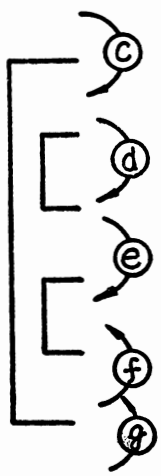
当从循环体中用 GOTO,IF…THEN 语句转出循环体或在循环体外用这些语句转入循环体(FOR堆栈区必须仍然存在)时，堆栈指针不会移动；但用 GOSUB 语句转出或转入时，堆栈指针将会移动，但用GOSUB语句转出循环体再用 RETURN 语句返回循环体时，堆栈指针将恢复原位，保证循环不会出错。反之，若用GOSUB语句转出循环体，而用其它语句如GOTO、IF…THEN或GOSUB再返回循环体时，堆栈指针移三位且没有复位，循环执行 NEXT 到语句时就立即出错。因此，当用控制语句转出或转入循环体时，必须保证两个条件才能使循环不出错，第一是内存中 FOR 堆栈区必须存在，第二是堆栈指针不能错位。

九、堆栈指针的指示

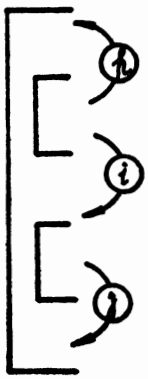
在多重循环中，循环体的转出与转入必须考虑内存中各循环语句的堆栈是否存在，堆栈指针是否错位，否则会造成计算机出错或导致错误的循环。下面给出10种情况，这些情况可分为三类。一类是正确的做法，一类是错误的做法，还有一类是在一定条件下正确做法，以便在使用中进行对照。



(i) 正确的做法



(ii) 错误的做法



(iii) 当NEXT后面的循环变量正确使用时允许

上面三类情况说明如下：

(i) 正确的做法：从多重循环（包括不嵌套的单层循环）的任何一个循环体中转到一级循环语句之前(即@的情形)或一级循环之后(即ⓑ的情形)都不会导致错误。但此时内存中的FOR堆栈仍然存在。

(ii) 错误的做法：当FOR堆栈尚未建立或FOR堆栈已经清除，从一级循环语句之外转入循环中任何一部分(如图示@，@的情形)，当执行到NEXT语句时出错。FOR堆栈建立后，从上一级循环语句的循环体直接进入下一级循环体时（如图示@，ⓕ的情形)，若NEXT后面的循环变量没有省略，执行到NEXT时计算机将发生出错；若NEXT后面的循环变量省略，执行到NEXT时将导致循环混乱，最后计算机出错。

(iii) 在一定条件下为正确的做法，是指NEXT后面的循环变量不能省略，而且正确使用时，从内层循环体转出外层循环体（如图ⓗ为向上转出，ⓘ，ⓙ为向下转出的情形）不会导致错误。如果省略了NEXT后面的循环变量，会导致循环混乱，但计算机并不出错。下面分别对ⓗ，ⓘ，ⓙ三种情形举例说明当NEXT之后省略循环变量导致循环混乱的例子。

ⓗ情形的例子如下：

```
10  B=1:FOR I=1 TO 2
15  LPRINT"I=";I;:IF B>=3 THEN 40
20  FOR J=4 TO 5:B=B+1
30  IF J=4 THEN 15
35  NEXT
40  LPRINT"J=";J;
50  FOR K=7 TO 8
60  NEXT
70  LPRINT"K=";K;
80  NEXT
```

```
I=1  I=1  I=1  J=4  K=9  J=6  K=9  I=2
J=6  K=9
```

执行10号语句后变量B的值为1，FOR I堆栈区建立，执行15号语句时打印第1个内容I=1往下执行20号语句FOR J堆栈迭加在FOR I堆栈的上面，变量B的值为2，然后转向到15语句，打印第2个内容I=1，再往下执行到20语句，新建立的FOR J堆栈与第一次的FOR J堆栈重合，变量B的值为3，再转向到15号语句，打印第3个内容I=1，此时由条件转向语句转入40号语句，打印第4个内容J=4，经50号语句，建立FOR K堆栈迭加在FOR J堆栈之上，经60号语句结束K循环(K的值成为9)，FOR K堆栈清除(但FOR J堆栈仍未清除，往下执行80号语句打印第5个内容K=9，再往下执行80号语句时，本来应该使循环变量I加上步长1，由于FOR J堆栈未清除。NEXT后面又没有写上循环变量I，所以NEXT语句使循环变量J加上步长1，并且不是去执行15号语句，而先错误地进入20号语句，从B=B+1开始执行到35号语句直至结束J循环(此时J=6，并且FOR J堆栈清除)再往下执行40号语句打印第6个内容J=6，此后运行转入正常。如果上述程序中在各个NEXT语句后面写上各自的循环变量堆栈指针不会错位，就不会出现上述循环混乱的情

况(由于这个程序比较简单，只需要在80语句的NEXT后面写上变量I即可)。

⑥情形的例子如下：

```
10  FOR I=1 TO 2
15  LPRINT"I=";I;
20  FOR J=4 TO 5
30  IF J=4 THEN 40
35  NEXT
40  LPRINT"J=";J;
50  FOR K=7 TO 8
60  NEXT
70  LPRINT"K=";K;
80  NEXT
```

I=1 J=4 K=9 J=6 K=9 I=2 J=4 K=9

J=6 K=9

10号语句建立FOR I堆栈，15号语句打印第1个内容I=1，20号语句把FOR J堆栈迭加在FOR I堆栈的上面，30号语句使程序转向到40号语句，打印第2个内容 J=4，50号语句把FOR K堆栈迭加在FOR J 堆栈上，60 号语句结束循环时循环变量的值为9，70号语句打印第3个内容K=9，80号语句本来是与FOR I 相对应的，由于 FOR J 堆栈未清除，NEXT之后又省略了循环变量I，所以第4个打印内容不是期望的I=2，而是误入 J 循环，使J循环结束后变量J的值成为6，进入40号语句，因而第4个打印内容成为 J=6，此后程序运行又发生了一次循环混乱， 情况与上述相同，留给读者分析。 这个程序也只需要在各个 NEXT 语句后写上各自相应的循环变量堆栈指针就不会错位， 也就不会导致循环的混乱。

⑦的情形与⑥的情形完全类似，只要把50号～70号语句去掉或放在15号～20号语句之间即可，不再重复。

十、多重循环的使用技巧

循环语句由于使用了堆栈方式而使运行速度大为提高，但在多重循环中，循环同样的次数，运行时间并不相同，现举两个简单的例子来比较。

运行下面一个空循环约需 2 秒钟

```
10  FOR I=1 TO 2
20  FOR J=1 TO 600
30  NEXT:NEXT
```

而运行同样次数的空循环却需要约6秒钟。

```
10  FOR I=1 TO 600
20  FOR J=1 TO 2
30  NEXT:NEXT
```

这是什么原因呢？在第一个空循环中，循环堆栈设立的次数和清除的次数都为 3 次。

第一次设立FOR I循环的堆栈。第二次(I = 1时)设立FOR J循环的堆栈，J循环结束后清除FOR J的堆栈，第三次(I = 2时)再设立FOR J循环的堆栈，J循环结束后又清除FOR J的堆栈，当I = 3时结束I循环，並最后清除循环堆栈，退出运行。

但在第二个程序中，I循环的堆栈也是设立一次，最后清除一次，而J循环的堆栈设立与清除达600次。计算机执行设立堆栈的指令是需要附加时间的，因此第二个程序运行速度明显变慢。如果含有多重循环的一个程序运行的时间很长（可能长达几十分钟以至几小时），则可以从这里的例子中看看循环安排是否合理，一般地说，应把循环次数最多的循环放在最内层循环中，把循环次数最少的放在最外层循环中。

十一、FOR语句与NEXT语句的对应关系

在通常设计程序时，FOR语句与NEXT语句必须一一对应，有一个FOR语句就必须要有一个NEXT语句与之相呼应，但了解了堆栈的意义之后，可以在特殊设计的程序中使多个FOR语句与一个NEXT语句对应，也可以使一个FOR语句与多个NEXT语句相对应，关键是保证FOR堆栈的存在与堆栈指针不发生错位，下面举两个例子来说明这一点。

几个FOR语句含用一个NEXT语句的例子：

```
10  FOR I=1 TO 2:FOR J=4 TO 5:FOR K=7 TO 8
20  LPRINT"I=";I;"J=";J;"K=";K,
30  NEXT
40  LPRINT
50  IF J=4 ORI=1 THEN 30
```

```
I=1  J=4  K=7  I=1  J=4  K=8
I=1  J=5  K=7  I=1  J=5  K=8

I=2  J=4  K=7  I=2  J=4  K=8
I=2  J=5  K=7  I=2  J=5  K=8
```

执行上述程序时，由10号语句先在内存中设立FOR I堆栈，再在FOR I堆栈上依次迭上FOR J和FOR K堆栈，执行20号语句时先打印出I = 1，J = 4，K = 7。再执行30号语句NEXT进行K循环，接着打印I = 1，J = 4，K = 8，再执行30号语句时完成K循环。FORK堆栈清除，进行40号语句使K = 8之后成为空行。执行50号语句时转向到30号语句，NEXT语句继续与J循环，使循环变量J加上步长成为5，又往下恢复K循环，结果打印出I = 1，J = 5，K = 7，I = 1，J = 5，K = 8这些内容。K循环结束后，使K = 8之后成为空行，又经40号语句和50语句转向到30号语句，NEXT语句继续参与J循环，使循环变量J成为6，从而结束J循环。此时FOR K堆栈、FOR J堆栈均清除，再次进入40号语句打印一个空行，再经50号语句转向到30号语句，此时NEXT语句参与I循环，使循环变量I成为2，然后再进入FOR J、FOR K等语句，重复上面的过程，这个程序的执行结果与具有3个NEXT语句执行的结果一致。

一个FOR语句使用几个NEXT语句的例子如下：

```
10  FOR I=1 TO 4
20  LPRINT I;
30  IF I=2 THEN 80
40  IF I=3 THEN 90
50  IF I=4 THEN 90
60  NEXT
70  END
80  LPRINT I*I*I;:NEXT
90  LPRINT I*I;:NEXT
1   2   8   3   9   4   16
```

上述程序运行时，执行10号语句在内存建立FOR I堆栈，执行20号语句先打印出变量I的值1，经30～50号语句到60号语句使循环变量I成为2，再打印出I的值2，再执行30号语句时转向到80号语句。打印出表达式I*I*I即2×2×2的结果为8，并执行80号语句中的NEXT语句，使循环变量I成为3，因此又进入循环体，打印变量I的值3，往下执行到40号语句时转入90号语句，打印出3×3的结果为9，再执行90号语句中的NEXT语句，使变量I值成为4，并自动返回20号语句执行循环体，打印变量I的值4，往下执行到50号语句时，再次转入90号语句，打印4×4的结果即16，接着执行90号语句中的NEXT，使循环变量I=5，从而不再进入循环体而结束循环，内存中的FOR堆栈同时被清除。

第二节　数组定义语句DIM中数组变量精度等级的使用

与数值变量一样，数组变量也具有3种精度等级。即整数型数组变量、单精度实数型数组变量，双精度实数型数组变量。按照通常方法定义的数组变量与简单变量一样，都是单精度实数型的，其中每个元素占用内存4个字节，如果在变量名之后加上%号，例如DIMA%(10)则该数组就成为整型数组变量，每个元素占用内存2个字节。如果数组的每个元素都是整数(例如年令、编号等)，数组又很大以至造成内存不够使用时，可采用整型数组以节省大量内存。

与数值变量一样，获得数组变量的3种精度等级可以在程序中使用 POKE 30912+ASC("变量首字符")，n这样的语句。当n=2时，该数组变量为整型；当n=4时该数组变量为单精度实数型；当n=8时，该数组变量为双精度实数型。例如键入

```
POKE 30912+ASC("A"),8:DIMA(3):A(1)=5#/3:PRINTA(1)↘
```

则屏幕显示数组变量A(1)的值为1.666666666666667

与数值变量一样，通过字符串函数STR$及VAL的转换，可以使整型或单精度实数型数组变量转变为双精度型数组变量，例如键入

```
A(8)=58:PRINTA(8)/3; VAL(STR$(A(8)))/3↘
```

则屏幕显示A(8)/3的结果为19.3333，而VAL(STR$(A(8)))/3的结果为

```
19.33333333333333
```

第三章　录音机的使用

录音机作为LASER—310计算机的外围设备之一，可以给LASER—310计算机的使用和开发带来很大的方便，而且显著地提高工作效率。

通过计算机的录音专门插口与录音机相连，可以把计算机内存中的BASIC程序或用机器码写的程序、数据，或是程序运行中的数据，通过相应的操作指令或语句指令贮存到磁带上保存下来。这样，计算机关闭电源或程序运行结束之后，需要保存的信息都在磁带上贮存下来了。以后需要使用磁带上贮存的程序或数据，只要通过相应的指令可以重新输入到计算机内存，从而节省了重复劳动的时间。如果没有磁带存贮方式，不但重新从键盘键入程序需要化费不少时间，而且容易失误，失误后又将在程序的查错和调试中化去许多时间。

把计算机内存中的程序通过录音机录到磁带上的过程，称为录程序，其相应的的操作指令是CSAVE。检验录到磁带上的程序是否与内存中的程序是否一致，其相应的操作指令是VERIFY。把录到磁带上的程序通过录音机输入到计算机内存的过程称为读入程序或装入程序，其相应的操作指令是CLOAD或CRUN，其中CRUN是读入程序后立即启动程序运行。把程序运行过程中的数据录到磁带上的过程，称为录数据文件。其相应的语句指令为PRINT#，把磁带上的数据文件输入到正在运行的程序中去，称为读数据文件，其相应的语句指令为INPUT#。本章将对以上操作过程详细说明。

第一节　录音机和磁带的准备

一、对盒式录音机的要求

LASER—310计算机配有专用的录音机LASER DR30供使用。但也可以选配其它盒式录音机来使用，其基本要求如下：

1．选配的录音机应具有ϕ3.5的输入(MIC或INPUT)插口与输出(EAR或OUTPUT)插口，使用时把随主机供应的录音机连接线中的黑色插头插入输入插口，红色插头插入输出插口。

2．选配的录音机应具有磁带记数器(TAPE COUNTER)，以记录各个程序所在的磁带圈数。如果没有磁带记数器，则磁带上什么位置录有什么程序就不清楚，这样读入程序比较麻烦，而录制程序不但麻烦，并且容易搞错位置而把原录在磁带上的程序抹掉。

3．选配的录音机应质量好，以方便使用。

4．通常录制程序(或数据文件)与读入程序(或数据文件)应该在同一型号的录音机上进行。如果录制的程序用不同型号的录音机来读入，由于不同录音机的走带速度、输出信号电平不完全相同，常常读入程序时会出错。例如把随主机供应的示范程序(在LASER DR30上录制)用普通欣赏音乐、语言的盒式录音机来读入，则通常的方法往往不能使程序

正确地读入内存，此时屏幕上将出现CLOAD ERROR（程序读入错误），此时可以采用如下方法：

(1) 把录音机连接线的红色插入输出插口(EAR)，另一个黑色插头不要插入录音机。

(2) 把录音机上的音调旋钮(TONE)调至接近最高的位置。

(3) 把录音机上的音量旋钮(VOLUME)调至中间位置的附近，并进行适当的试验，直至程序能正确地读入内存。

如果录制程序也采用普通盒式录音机，当用通常方法不能正确录入时，也可以采用上述方法，但第(1)点中应改为把录音机连接线的黑色插头插入输入插口（MIC)，另一个红色插头不要插入录音机。

二、对磁带的要求

录制程序用的磁带可采用普通盒式磁带，但贮存程序的全部磁带段不应有任何缺陷，如疵点、擦痕等，这方面的要求比音乐、语言磁带的要求高一点。因为微小的缺陷录制音乐或语言时只是使个别的音符或语言失真，而录制程序时却能使整个程序出错，因为一个程序中是容不得微小差错的。

为了保证录制与读入程序的准确性，应采用质量好的磁带。在使用中应防止轧带，轧带也会使录制好程序被破坏。

第二节　录制与读入程序前的准备

一、录音机的准备

录音机应完好无损，能够正常工作。用电池作电源时，正负极不要搞错。电池的电量要充足，不漏液。使用外接电源时，特别要注意插头的正负极性、电压的正确和稳定，然后把录音机的电源接通。把录音机上的各个可调旋钮调节到适应位置，音调旋钮通常调节至最高位置，音量旋钮通常可调节在中间位置附近。对于LASER DR30录音机，只需将录音机左侧的拨动开关拨至"310"处即可。

二、录音机的连接

把录音机与计算机正确连接好，通常应在计算机关闭电源的情况下连接，然后接通计算机的电源。当计算机中已有程序时，不需关闭计算机电源。LASER-310计算机没有遥控功能，因此遥控(REMOTE)插座不需使用。

三、录制程序操作

装上磁带。把盒式磁带装入录音机(先按带盒弹出键STOP/EJECT开启带盒门)，并关闭带盒门。按下倒带按钮(REWIND)把磁带全部倒回。再按下磁带计数器旁的复置按

钮，使磁带计数器回复零位。最后用快速卷带按钮找到开始录或读的磁带圈数。

第三节　录制程序和校对程序

在上述准备工作做好之后，可以把计算机内存中的程序录入磁带，其步骤如下：

一、录制程序

1．通过键盘键入CSAVE“文件名”，暂不键入回车。

其中文件名是给录入磁带的程序起名，一般可根据程序的工作内容给程序起名，如CSAVE “BASIC␣WORDS” 表示这个程序是用来学习BASIC语言的程序，也可以对程序进行编号的方式对程序起名，如CSAVE “IA1—10”，表示这个程序磁带A面的第1个程序，从第10圈开始。也可以根据程序的类型给程序起名，如CSAVE “CC.BAS”表示这个程序是用 BASIC 语言编写的汉字（CHINESE CHARACTER）程序，而用CSAVE “CC.EXE”表示这个程序是用二进制机器码表示的汉字，等等。

文件名必须是一个字符串常量，其中的字符可采用任何字符（包括字母、数字、标点符号、空格等），字符数(即字符串长度)必须小于或等于15。

不使用文件名也可以录制程序，这时录入磁带的程序就没有文件名，当磁带上录有许多不带文件名的程序时，以后读入程序时就不知道录入的程序是什么程序，如果忘记了磁带的圈数，就难以找到所需读入的程序。

2．同时按下录音机上的录音键和放音键（即RECORD键和PLAY键），使录音机进入录音状态，磁带开始转动。

3．当磁带记数器的数字符合需要时，在计算机键盘上键入回车，此时录音机(LASER DR30)上的存贮/电池状态(SAVE/BATT)显示器将发光表示计算机的信息正读入磁带，否则表示电源电压过低，将导致录制错误。

4．当屏幕出现 READY 提示符及光标时，表示录程序过程结束，但在录制过程中。磁带上录制的程序是否与计算机内存中的程序是否完全一致，在显示屏幕上并不给出提示，所以还不知道录制的程序是否正确，需要通过校对才能最后确定下来。

5．在LASER—310 计算机上，录制程序时程序在内存中的起始地址通常为十进制的31465，这个地址数据存放内存的十进制地址的30884与30885两个存贮单元中。两个单元中的数据分别为 233 与 122，其中地址为 30884 的存贮单元中存放地址数据的低位值，即233，地址为30885的存贮单元中存放的地址数据的高位值，即122，这两个数据是计算机接通电源时由ROM中的解释系统设定的。键入BASIC程序时，程序存放内存中的地址就是从31465开始的，所以录制时程序的起始地址不需加以改变。如果录制的程序不是从这个地开始（这种程序往往是二进制机器码程序），则在录制前必须改变录制的起始地址与终止地址，否则不能把程序录制在磁带上，这一点必要加以重视。这方面的例子可参见本书第三册GOSOB语句的程序举例“屏幕汉字库程序”。

二、校对程序

1．按下录音机上的倒带键（REWIND 键），使磁带倒转至录程序时的起始圈数。再按下录音机上的停止键(STOP/EJECT 键)。校对程序前必须使磁带记数器的数字稍小于或等于录程序的起始圈数。

2．由键盘键入操作指令 VERIFY “文件名” 并键入回车，再按下录音机上放音键，文件名可以省略，若不省略文件名必须与录程序时所起的文件名完全一致，否则不进行校对。为了避免这种麻烦，在LASER—310中可以省略 VERIFY后面的文件名， 而直接键入VERIFY↘，此时在屏幕显示区的最末一行显示反白字符（或称反视频字符）组成的提示符如下：

WAITING

其意思为计算机已准备好接受录音机输出的信息， 现在磁带上的信息尚未进入计算机，正在等待，当磁带上程序的文件名输入计算机时(不妨设文件名为1A1—10)，则在屏幕显示区最末一行显示反白字符组成的提示符及正常字符组成的文件名如下：

LOADING T:1A1—10

其意思为文件名为 1A1—10 的程序正在输入计算机（同计算机内存中的程序进行校对)。 其中T是英文TEXT的首字符，TEXT 表示输入的文件(即程序)是一个文本文件。文本文件由ASC码组成。

如果在 VERIFY 后面使用文件名，当文件名与需校对的程序具有相同的文件名时，也出现上述同样的情况；而如果文件名与磁带上程序的文件名不完全一致时，屏幕显示的内容成为：

FOUND T:1A1—10

其意思为计算机发现了(FOUND)磁带上有一个文件名为1A1—10 的文本文件，但这个文件的内容(即程序)不输入计算机同计算机内存中的程序进行校对。

3．当录到磁带上的程序与内存中的程序完全一致时，屏幕显示提示

VER1FY OK

表示录制的程序完全正确。此时应按下录音机上的停止键，记下磁带记数器上的数字。这个数字是表示程序结束时的磁带圈数。此时录制一个程序的工作才算最后完成。

4．当磁带上的程序与内存中的程序不完全一致时，屏幕显示

VERIFY ERROR

表示录制在磁带上的程序不完全正确，此时应按下录音机上的停止键，再倒回磁带到原来的记数后重新再录。直至校对无误后才可结束。

5．磁带经多次使用后，会伸长使磁带圈数稍有变动，因此录程序时， 两个程序间应间隔一定的圈数。这样既便于修改（即重录一个经修改后的程序)， 又便于保存前后录入磁带的程序不被冲掉。在原录程序的磁带上重复一个程序时，通常应从原来程序的磁带圈数之前一点开始录，以免以后读入程序时出错。

6．录有程序的磁带应妥善保管如下：

① 录有程序的磁带应配有相应的书面记录，包括磁带编号、程序编号、文件名，起讫圈数，程序名等，以便使用。

② 防止磁场、有害气体、液体的污染和外伤。磁带不用时应倒回磁带放入磁带盒。

第四节　读入程序

读入程序的准备工作按照第二节所述，读入程序的过程如下：

1．不管计算机内存中是否有程序，需要从录音机上读入程序时，不必键入NEW操作指令，只需键入CLOAD(或CRUN)操作指令。在CLOAD或CRUN后面可以使用文件名也可以省略文件名。键入CLOAD“文件名”时，文件名必须与录入磁带的文件名完全一致，否则程序不能读入。在LASER—310计算机中，为了方便，通常可以在找到磁带圈数后键入CLOAD或CRUN指令并键入回车，此时在屏幕底部出现如下提示：

WA1TING

表示计算机等待磁带输入程序。

2．按下录音机上的放音键(PLAY键)，此时录音机上(LASER DR30)的存贮/电池状态显示器将发光，表示录音机正在把磁带上的信息输出到计算机中去，否则表示电源电压过低，将导致计算机读入程序出错，不妨再设定读入程序的文件名为1A1—10，则当计算机读到这个程序时，屏幕显示提示

LOADING　T:1A1—10

表示计算机正在输入文件名为1A1—10的程序。

3．当磁带上的程序完整且正确无误地输入到计算机内存中时，屏幕显示READY提示符及光标，此时程序的读入工作已告完成，若使用CRUN操作指令，则当磁带上的程序完整且正确无误地输入到计算机内存中时，计算机就立即开始执行程序，因此CRUN操作指令相当于两个操作指令CLOAD与RUN的联合作用。

4．当磁带上的BASIC程序输入到计算机内存中去时，原在计算机内存中的BASIC程序已被新输入的程序取代；如果磁带上的程序没有输入到计算机内存，这时屏幕上不给出 LOADING　T:“文件名”这样的提示，则计算机内存中的原BASIC程序仍保留，

5．如果在计算机内存中含有机器码写的程序，并且机器码程的起始地址不在内存地址的31465，则从磁带上输入BAS1C程序时，机器码仍保留，例如第三章GOSOB语句的应用举例中，当屏幕汉字程序存放在内存地址的－30976～－30556，若需从磁带上输入BASIC程序，应把BASIC程序的起始地址在31465，然后输入程序（可以键盘键入或磁带输入），则机器码程序仍保留，但须注意BASIC程序不能过长以至冲掉机器码程序。如果机器码程序从内存地址31465开始存放，则输入BASIC程序（键盘键入或磁带输入)时，需把BASIC程序的起始地址改在机器码程序的末地址之后，这方面的例子可参见本书第六章光笔子程序部分。

第五节　数据文件的录制与读入

在运行BASIC程序中，可以把程序运算的数据、(数值数据或字符串数据)包括初始数据、中间结果或最终结果，在程序运行开始后或结束前，通过录音机把这些数据输送到磁带上保存起来，也可以在此期间把录制在磁带上的数据输入计算机。录入磁带的数据就称之为数据文件。文件数据以DATA来表示(简写形式为D)。

一、数据文件的录制：PRINT＃ 录数据语句

录数据语句 PRINT＃ 可以把程序中的数值常量、数值变量，数值型数组变量及其组成的算术表达式的值录到磁带，也可以把字符串常量录到磁带。其格式如下：

1. 录数据语句的一般格式

格式1：行号 PRINT＃“文件名”，算术表达式，算术表达式，……，算术表达式
格式2：行号 PRINT＃“文件名”，字符串常量

2. 录数据语句的使用规则与功能

(1) 录数据语句可以在程序中作为单独的一行语句，也可以用冒号并列在程序的其它语句之后，並可以放入循环语句(或其它控制语句中)。

(2) ＃号后面的文件名不可省略，否则出错。文件名必须是字符串常量，用双引号括起来，其中的字符数必须小于或等于15，文件名后面的逗号不可省略，也不可使用其它符号。

(3) 各算术表达式的运算结果必须是整数或单精度实数才有效，若为双精度实数则自动转变为零，各算术表达式之间只能使用逗号，若使用分号则数据不能读出。算术表达式中允许使用函数值为数值的字符串函数。

(4) 若存入磁带的数据为字符串，则必须为字符串常量，不允许使用字符串变量或字符串函数。字符串常量可以使用＋号联接的方式。字符串常量中不允许使用逗号，否则读入数据时出错，字符串常量只允许使用1个，且不允许与数值数据放在同一个数据文件中。

二、数据文件的读入：INPUT＃读数据文件语句

读数据文件语句 INPUT＃ 可以把录在磁带上的数据文件读入计算机内存，它必须与PRINT＃语句的格式完全相配。

1. 读数据文件语句的一般格式

格式1：行号 INPUT＃“文件名”，数值变量，数值变量，…，数值变量
格式2：行号 INPUT＃“文件名”，字符串变量

2. 读数据文件语句的使用规则与功能

(1) 读数据文件语句可以在程序中作为单独的一行语句，也可以用冒号并列在程序的其它语句之后，并可以放入循环语句中。

(2) #号后面的文件名不可省略，否则出错。文件名必须与磁带上的数据文件的文件名一致，否则不能读入数据，计算机执行INPUT#语句时，不管INPUT#后面的文件名是否与磁带上的文件名一致，屏幕均提示：

FOUND	D:文件名

表示已出现磁带上的数据文件，数据文件以D:表示，D是英文字母 DATA 第一个字母，屏幕提示的文件名即为磁带上的数据文件的文件名。如果 INPUT# 后面的文件名与屏幕提示的文件名一致，则磁带上的数据输入到计算机内存。(赋值给文件名之后的变量)。如果文件不一致，则磁带上的数据不输入。

(3) 各数值变量可以是简单变量， 也可以是数组变量， 变量的数目不能少于或多于该数据文件中的数据的数目。变量数目少于数据时，计算机出错，其出错提示为EXTRA IGNORED,意思为超过(EXTRA)计算机的接受能力(IGNORED)。当变量数目少于数据时，计算机出错，其出错提示为 OUT OF DATA ERROR 意思为数据已读完，不能再读。

(4) 使用格式时，字符串变量只能使用1个，与文件中的数据数目相一致。

(5) 在程序中使用INPUT－语句时，应根据磁带上数据文件的数目来使用INPUT#语句的次数，若数据文件有个15(如把PRINT#"文件名"放在循环语句中，该语句重复执行15次)，则应使用INPUT#语句15次，少于15次只能读入前几个文件中的数据，但不能使用INPUT#语句16次，即INPUT#语句使用的次数不能超过数据文件的数目，否则计算机将进入死锁状态，即使用中断键不能退出运行，只有关掉主机重来。因此在循环语句中使用INPUT#语句时， 特别要注意磁带上的数据文件的数目应与循环次数相配，循环次数不能多于数据文件的数目。

三、数据文件的录制与读入

数据文件的录制与读入举例：见第三册程序举例之十七；数据文件的录制与读入。

第四章　LASER PP40 打印机的图案打印模式

第一节　概　　述

在图案打印模式中，打印机是在 XOY 直角坐标系中进行工作的，可选择字体尺码大小，打印笔的颜色，笔架定位，任意移动笔架水平位置，任意进纸、退纸（最大可进、退纸6.55米），可打印各种字符101种（与文字模式相同），并可选择方向（上、下、左、右），可画直线、曲线、打印仪字，可选用实线或15种虚线，可绘制数轴，直角坐标轴或斜坐标轴，绘制立体图形，画线速度水平方向或垂直方向为每秒52毫米，45°（或135°）方向为每秒73毫米，打印机的这些功能都是用一套特殊的指令来完成的。可参见第一册的附录。

在图案模式中，打印机面板上的键功能与打印机底部输出功能开关的作用仍不变，可参见第一册的说明，本章主要介绍图案模式中 XOY 直角坐标系的使用，各种特殊指令的使用规则及其功能，各指令间的相互关系以及应用程序举例。

第二节　图案模式中XOY直角坐标的使用

一、关于XOY直角坐标系的说明

打印机进入图案模式后，笔架的运动是根据 XOY 坐标位置进行工作的，进入图案模式时，不管笔架处于什么位置，系统均设定该点的位置为XOY直角坐标系的原点，即X＝0，Y＝0，并以该点为原点建立XOY坐标如图：

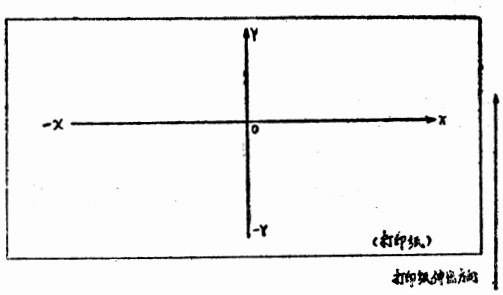


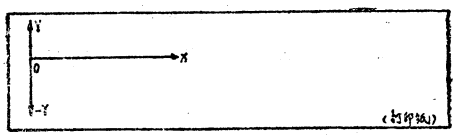
图4—1　图案模式中的XOY直角坐标系

图中0点为进入图案模式时打印笔尖的当前位置。例如，当从文字打印模式转入图案模式时，打印机字体尺码选择为1，并执行如下指令

$$\text{LPRINT}\overbrace{\text{“␣␣}\cdots\text{␣”}}^{\text{20个空格}};\ \text{CHR\$(18)}$$

则打印机以第21个空格符的起始位置作为坐标原点，这与图4—1的情形相同，如果开机后使

用这样的指令LPRINTCHR$(18)，则以笔尖所在的最左端位置为坐标原点，如图所示：



二、关于坐标值的使用

1．在数学中，平面坐标的坐标值采用(X，Y)来表示。在打印机中，则采用“指令X，Y”的形式或“指令”；X；“，”；Y 两种方式来表示，其中第一种方式X，Y 必须为数值常量，第二种方式X，Y可为算术表达式。

2．X表示笔架的水平方向动作，X为正值时笔架向X轴正方向动作，X 为负值时笔架向X轴负方向动作，这些动作都相对于当时的坐标原点或笔尖所在位置而言。

Y表示打印纸的进退方向动作，Y 为正值时打印纸后退，相当于笔架向 Y 轴正方向动作；Y为负值时打印纸前进，相当于笔架向 Y 轴负方向动作，这些动作也都相对于当时的坐标原点或笔尖所在位置而言。

3．X、Y的取值范围为三位整数有效(包括正、负数值)，整数超过3位时，只取末三位整数，其余舍去，但正、负符号不舍，X、Y 的数型可以是整数、实数，但有小数部分时最好事先用 INT 函数取整，因为当整数部分为零，小数部分超过3位时，打印机会误取末3位小数作为整数，这一点务必引起注意，以免发生错误， 数的表示法可用定点数，也可用浮点数，但使用浮点数时，也应注意先取整。

4．X、Y的取值与打印机动作的关系。

打印机动作包括笔架移动与打印纸的进退，如图所示，打印机笔架水平移动的范围在AA线与BB线之间，两线之间的距离为96毫米。

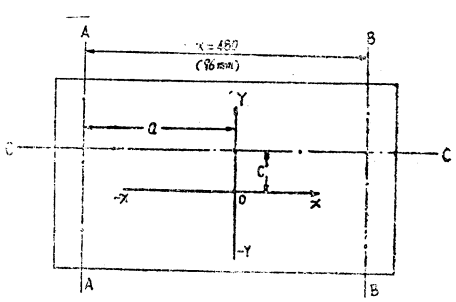


图4—2　X、Y的取值与打印机动作的关系

坐标X，Y的数值，与打印机动作之间的关系如下：

笔架水平动作的距离 = 0.2*X(毫米)

打印纸进退动作的距离 = 0.2*Y(毫米)

上述关系式在一定条件下成立，具体说明如下：

设坐标原点距AA线为a(或0.2×a毫米)，则当X的取值满足0≤a+X≤480时，(其中0≤a≤480)上述关系式才能成立，此时X、Y的值均有效(-999≤Y≤999)，但X、Y应均为整数。若a+X<0或a+X>480，则X的值只有一部分有效，同时Y的值也只有一部分有效；当a+X<0时，只执行X=-a,当a+X>480时，只执行X=480-a。例如设a=240，即笔架处于中间位置，则当-240≤X≤240时,关系式有效，若取X=-500(或X=500)，则笔架水平移位时只执行X=-240（或X=240)，同时，Y的值也会打折扣，例如设a=240，X=500，Y=300，则执行"指令X、Y"时，笔架水平动作的距离为0.2×240(毫米)，而不是0.2×500（毫米)，打印纸退回的距离为0.2×(300-m)（毫米)，(其中0<m<300，而不是0.2×300(毫米)，X的值超范围越多，对Y值的影响也越大(即m的值越大)，因此，X的值必须取得合理，否则打印机的动作就会失误。以上说明，适用于LPRINT "MX，Y"，LPRINT"DX，Y"，LPRINT"RX，Y"，LPRINT"JX，Y"等指令，详见第三节

第三节、图案模式中特定语句指令及其使用

一、单字符指令

单字符指令是表示指令后面不带数据，单字符指令均与XOY坐标系有关

1. 返回文字模式指令　LPRINT"A"（A—ALPHABET 字母)

格式：行号LPRINT"A"

功能：在图案模式中其功能完全等同于LPRINT CHR$(17)。打印机执行此语句指令之前，必须先得到一个回车信号，否则此语句不起作用，此语句的执行结果是清除图案模式终点的X、Y坐标，从图案模式返回到文字模式，笔架返回水平方向最左端，打印纸不动作。但图案模式中设定的字体尺码，颜色、线型仍保留。

用途：常用于两种模式的转换，特别是文字模式中需要选择字体尺码时，经常要使用此语句。

例如，在文字模式需使用2号字体打印"ABCDE"，可用下例语句

(行号) LPRINT CHR$(18)；"S2":LPRINT"A"；"ABCDE"

执行后即用2号字体打印ABCDE五个字母。

若改为LPRINT CHR$(18)；"S2"；CHR$(13)；"A"；"ABCDE"则效果也相同，更简单些，可改为LPRINT CHR$ (18)；"S2"；"，A"；"ABCDE" 或改为 LPRINT CHR$(18)；"S2，A"；"ABCDE" 都能达到同样目的，其原因是单数据指令S（指S后面跟一个数据）后面使用逗号就相当于输出一个回车信号，逗号必须放在双引号内，否则不起作用。

2. 复位语句指令　LPRINT"H"（H—HOME 回位)

格式：行号LPRINT"H"

功能：执行此命令后打印机笔架返回到坐标原点。

用途：在图案模式中打印机绘画或打印字符后，要使笔架返回坐标原点，可使用该指

令来实现。例如，在虚实线类型选择语句的举例（见第三册程序举例之20，16种虚实践的绘制)中，若在程序的最末添上70号语句 LPRINT“H”，则该程序执行后笔架将自动返回到第一条水平线的左端；如果在画线中途重新设定起点位置，例如在打印到第 5 行时使用 LPRINT“I”，则执行到70号语句将使笔架返的到第五条水平线左端。其程序如下：

```
10  LPRINTTAB(5); "*****  16  LINE  TYPES  *****"
20  LPRINT:LPRINTCHR$(18)
30  FORK = 0TO15
40  LPRINT"L"; K", PL"; K:IFK = 5LPRINT"I"
50  LPRINT"J480, 0"; CHR$(13); "R0, -10"
61  NEXT
70  LPRINT"H"
```

3. 定位语句指令　LPRINT“I”　(I—INDICATE指定)

格式：行号 LPRINT“I”

功能：取消原定的坐标原点，以笔架的当前位置为新的坐标原点。

用途：在图案模式中，打印机的绘图工作必须依靠 XOY 直角坐标系来定位，根据绘图的需要，常要变换坐标原点的位置，因此定位语句在这种情况下就成为必不可少的有力工具了。

定位语句应用的程序举例，见第三册程序举例之19：平面三色圆的绘制。

二、单数据指令

单数据指令是表示指令后面必须带有一个数据，有 3 个单数据指令与 XOY 坐标系无关。

1. 虚实线类型选择语句指令：LPRINT“L”　(L—LINE TYPE 线型)

格式1：行号LPRINT “L 数”

格式2：行号LPRINT“L”；算术表达式

功能：两种格式中数据取值范围为0≤数，算术表达式≤15。数 = 0时表示用实线绘图形，　打印机开机后系统自动设定 LPRINT “L0”。当数值在1～15范围内时，　均表示用虚线绘图形，数值越大，画虚线时线段越离散，该指令与XOY坐标系无关。

应用：在绘图中需要画虚线(如立体图形)时，可用到这一语句。绘制16种线型的程序见第三册程序举例之20：16种虚实线的绘制。

2. 颜色笔选择语句指令：LPRINT“C”　(C—COLOR颜色)

格式1:行号 LPRINT “C 数”

格式2：行号 LPRINT “C”；算术表达式

功能：两种格式中数据取值范围为0≤数，算术表达式≤3。如数超过此范围，打印机

自动与3进行布尔函数的运算，例如取数＝102，则102 AND 3＝2，相当于数＝2。数＝0时采取黑色笔，数＝1时采用兰色笔，数＝2时采用绿色笔，数＝3时采用红色笔， 该指令与XOY坐标系无关。

用途：根据打印字符或绘图的需要，可在程序中任意选择所需的打印笔的颜色。绘图中颜色转换的程序举例见第三册，程序举例之21。

3. 字符数码选择语句指令：LPRINT “S” （S—SCALE尺度）

格式1：行号LPRINT“S 数”

格式2：行号LPRINT “S”;算术表达式

功能：两种格式中数据的取值范围为0≤数，算术表达式≤63。 数的大小决定打印字符的尺码，数＝0时字体最小，数＝63时字体最大，每行打印的字符数＝$\frac{80(\text{需修整时}+1)}{\text{数}+1}$，返回文字模式(不关打印机电源)仍可保留所选的字体尺码。

用途：在图案模式与文字模式中可根据需要选择字体尺码。选择64种字体尺码进行打印的程序举例的见第三册，程序举例之22：打印字母A的64种字体。该指令与XOY坐标系无关。

4. 图案模式中打印字符的语句指令：LPRINT“P” （P—PRINT打印）

格式1：行号LPRINT “P字符串常量”。

格式2：行号LPRINT“P”；表达式；…；表达式

功能：格式1中字符串常量的长度仅受语句长度(64个字符以内)的限制。格式2中表达式可为算术表达式或字符串表达式，均打印其结果。各表达式之间必须用分号或逗号作分隔。在算术表达式中，可以进行整数，单精度实数，双精度实数的运算。使用CHR$字符转换函数，还可以打印101种单字符（参见第一册文字打印模式中的介绍及附录)。直至执行回车信号后该指令结束执行。例如下面一个简单程序可说明该指令的功能：

```
10  LPRINTCHR$(18); "S1, C2" :X=10:Y=5.5:Z=1 000:A$=CHR$(203)
20  LPRINT"PA"; 102+3*Y; X/3; VAL(STR$(X))/3; X; SQR(X); (Z/3)
    ^2.5;
30  LPRINT"CD"+"EF"; ASC("G"); A$; CHR$(13); "C0, A"
A   118.5  3.33333  3.333333333333333  10
    3.16228  2.0286E+06  CDEF  71β
```

上述程序说明如下：

10号语句用LPRINT CHR$(18)转入图案打印模式，并用S指令与C指令选择好 1 号字体和绿色打印笔，然后用赋值语句给变量X，Y，Z，A$赋值，其中CHR$(203)表示单字符β，(提前1格打印)

20号语句用P指令表示打印字符，其后的字符A代表1个字符的字符串常量，已失去A指令的意义，因此执行结果打印一个字母A，接着打印算术表达式102＋3*Y(＝118.5)、X/3(＝3.33333)，VAL(STR$(X))/3 (＝3.333333333333333)，X(＝10)，SQR (X) (＝3.16228)，(Z/3)↑2.5(＝2.0286＋6)的值。

30号语句继续打印字符串表达式“CD”+“EF”的值，G的ASCII码(=71)，A$的值：提前1格打印希腊字母β（数值71面留的一个空格被β占用），执行回车符CHR$(13)后，P指令不再执行，其后的“C0，A”不再作为字符串，而是作为C指令与A指令来执行，如果把30号语句最末的分号去掉，则当于输入一个回车信号，P指令失效，30号语句中的LPRINT“CD”+“EF”；ASC(“G”)；A$均不执行。

该指令执行后使笔架在坐标系中的当前位置发生变化。

用途：P指令使图案打印模式象征性地进入文字打印模式，以便在绘图工作中，随时打印所需的字符而不必返回文字模式，以免使原有的XOY坐标失效。

5. 字母编印方向选择语句指令：　　LPRINT“Q”　(Q—QUADRANT象限)

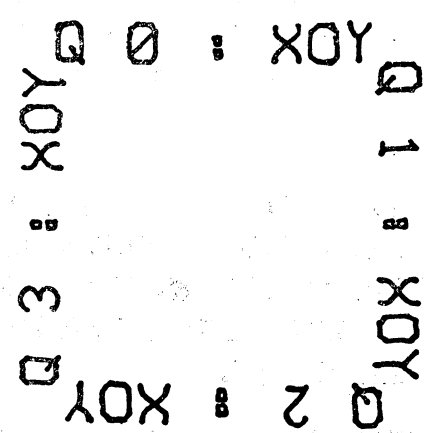
格式1：行号LPRINT“Q数”

格式2：行号LPRINT“Q”；算术表达式

功能：两种格式中数据的取值范围为0≤数，算术表达式≤3。该指令使打印字符（执行P指令）时XOY坐标系旋转，当数=0时，XOY坐标系保持从文字模式进入图案模式时的状态，打印字符时水平从左至右，当数=1时，XOY坐标系及字体顺时针旋转90°，打印字符时垂直从上至下，当数=2时，XOY坐标及字体顺时针旋转180°，打印字符时水平从左至右，字符成倒写形状，当数=3时，XOY坐标系及字体顺时针旋转270°，打印字符时垂直从下至上。该指令对绘图工作不起作用，即不能使绘图指令执行时发生坐标系旋转。当图案模式返回文字范式时，该指令失效，即恢复为数=0。

用途：在图案或打印表格中需打印不同方向的字符时可用到，应在P指令之前使用Q指令。例如下面一个简单程序可说明Q指令的作用：

```
10  LPRINTCHR$(18); "S3, 120, -30"
20  FORI=0TO3:LPRINT"Q"; I; ", C"; I; ", PQ"; I; ":XOY":NEXT
30  LPRINT"S1, C0, M0, -300";CHR$(13); "A"
```



上述程序说明如下：

执行10号语句时进入图案模式，并选择字体尺码为 3 号，M指令为移动笔架水平右移120步距(即0.2×120＝24毫米)，垂直下移30步距(即 0.2×30＝6 毫米)实际上笔架并不垂直移动，而是打印纸伸出6毫米(详见下述)。

20号语句是一个循环语句，循环体内先执行Q指令，C指令，接着执行P指令，执行 P指令时根据Q指令选择的打印字符方向打印字符。循环语句的执行结果正好说明了“Q 数”所表示的打印字符的方向。

30号语句重新设定字符尺码，打印笔颜色，并使笔架移位(打印纸伸出60毫米)，再返回文字打印方式。

三、两数据指令

两数据指令的特点是指令后面必须带有2个数据(或2个数据为一组的数据组)，第一个数据代表X方向的动作步距，第二个数据代表Y方向的动作步距。两数据指令均与XOY 坐标系有关，是绘图的主要工作语句。

1．移位语句指令：LPRINT“M” (M—MOVE移动)

格式1：行号LPRINT“M数1，数2”

格式2：行号LPRINT“M”；算术表达式1；“，”；算术表达式2

功能：两种格式中数据的取值范围为：

0≤a＋数1≤480；其中0≤a≤480，a表示当前 XOY 坐标系原点离开最左端打印位置的步距(参见图4—2)，对算术表达式1也同样为0≤a＋算术表达式1≤480

－999≤数2，算术表达式2≤999。

为了防止出错，当数据不为整数时，应当取整后再使用。

执行M指令后，打印机笔架水平移位至离坐标原点数 1所表示的步距处，打印纸进退一定的长度使笔架距坐标原点等于数2所表示的步距；当数1超范围时，数1数2的值均不能正确执行。数1与数2的每一步距均为0.2毫米。

移位语句写成LPRINT“M0，0” 时，其作用完全与复位语句LPRINT“H” 相同，复位语句可以看成是移位语句的特例，但复位语句格式简单，使用上 (与其它语句搭配使用时)也往往较移位语句方便。

用途：

(1) 在非一笔画的图案中必须使用移位语句，用M指令把笔架移至每一笔的起点处。

(2) 用于建立新的XOY坐标系， 通常用 LPRINT“M 数1，数2”；CHR$(13)；“I”这些指令即可把XOY坐标系重新建立在新的位置，新坐标系XOY的坐标原点在 X 轴方向上距原坐标系原点数1步距，在Y轴方向上距原坐标系原点数2步距， 移位语句的程序举例见第三册程序举例之23：三维坐标系的绘制。

2．相对移位指令：LPRINT“R” (R—ReLatively Move相对移动)

格式1：行号LPRINT“R数1，数2”

格式2：行号LPRINT“R”；算术表达式1；“，”；算术表达式2

功能：两种格式中数据的取值范围为 0≤aa＋数1≤480，（对算术表达式 1 也同样），－999≤数2，算术表达式2≤999。其中aa是笔架的当前位置距最左端的步距数，与坐标系XOY的原点无关。为防止出错，当数据不为整数时，应当取整后再使用。执行R指令后，打印机笔架水平移至离当前笔架位置数1的步距处，打印纸的进退使笔架距当前笔架位置等于数2所表示的步距；当数1超范围时，数1，数2的值均不能正确执行，相对移位语句实实质是把笔架的当前位置暂时作为坐标系的原点，然后再把笔架移位至数1，数2在暂时坐标系的数1、数2步距处，而原坐标系原点仍不变。

用途：与M指令相似，常与M指令根据实际需要配合使用，在某些场合下，使用R指令较为方便。相对移位语句的应用举例见第三册程序举例之24：利用相对移动语句绘制五角星。

3. 画线语句指令　LPRINT“D”（D—DRAWING 画图）

格式1：行号LPRINT“D数1，数2，…”

格式2：行号LPRINT“D”；算术表达式1；“，”；算术表达式2；…

功能：两种格式中数据可为一组或多组，当数据为多组时连续画线。数据的取值范围与M指令相同，数1超范围时，数1，数2的值均不能正确执行。执行该语句时打印笔从当前位置画到距坐标原点的数1，数2步距处，当数据为多组时，连续画线的各段均为直线。

用途：画线语句是打印画图的最主要的语句，打印机绘图，写汉字，都必须使用画线语句，对画线语句进行功能试验可用下列程序：

```
10  A=240:C=-50:LPRINTCHR$(18);"M";A;",";C;CHR$(13);"I"
20  INPUT"M OR D";A$:INPUT"X=? Y=";X,Y:IF A$<>"M"ANDA$<>"D"THEN20
30  IFA$="M"LPRINT"M";X;",";Y
40  IFA$="D"LPRINT"D";X;",";Y
50  GOTO20
```

执行上述程序，可以根据使用者的需要使打印机画一些几何图形，如三角形，平行四边形或其它图案，也可以对X、Y在M指令与D指令中的有效值进行试验。

上述程序说明如下：

10号语句给变量 A 与 C 赋值后，进入图案模式，然后使用移位指令把笔架移至变量A、C所表示的坐标处，X轴方向距最左端打印位置为240步距（即0.2×240＝48毫米），Y轴方向距进入图案模式时的坐标原点负方向(向下)50步距（即0.2×50＝10毫米），参见图4—2所示。接着输入一个回车信号 CHR$(13)，以结束M指令的作用，执行定坐标原点的指令 I，把坐标原点重新定在当前笔架位置。

20号语句用键输入语句要求提供数据M或D，并赋值给串变量A$，以决定打印机执行M指令或执行D指令，接着再用键输入语句要求提供坐标数据 X、Y 即数1，数2，并赋值给变量X与Y。接着再判断数据M或D是否输入，否则重新执行20号语句。

30号语句根据输入的数据为M时执行移位动作。

40号语句根据输入的数据为D时执行画线动作。

50号语句转向到20号语句，重新要求输入数据，以便继续进行绘图或移位。

画线语句的应用举例见第三册程序举例之25：打印机书写汉字“千里之行，始于足下”。

4．相对画线语句指令　　LPRINT“J”　（J—JOIN 连接）

格式1：行号LPRINT“J数1，数2”

格式2：行号LPRINT“J”；算术表达式1；“,”；算术表达式2

功能：两种格式中数据的取值范围与R指令相同，执行J指令时，打印笔从当前位置画线到以当前位置为坐标原点的数1，数2所表示的步距处，而原来的坐标原点仍不变。

用途：与D指令相似，常与D指令根据实际需要配合使用，在某些场合下，使用J指令较方便。若把J指令写为LPRINT“HJ数1，数2”，则与LPRINT“D数1，数2”完全相同。

相对画线语句的程序举例见第三册程序举例之26：正多边形的绘制。

四、三数据指令　　绘制XOY坐标轴语句指令LPRINT“X”

三数据指令是在指令后面必须带有三个数据：三数据指令只有1条。

格式1：行号LPRINT“X数1，数2，数3”

格式2：行号LPRINT“X”；算术表达式1；“，”；算术表达式2；“，”；算术表达式3。

功能：两种格式中数据的取值范围如下：（以笔架当前位置作为起点）

0≤数1，算术表达式1≤1。当数1＝0时绘制Y轴，当数1＝1时绘制X轴。实际上数1为不等于零时均画X轴（水平轴）

数2或算术表达式2的值必须根据画X轴还是画Y轴以及数3（或算术表达式3的值来定）

当画Y轴时（即数1＝0时），可取－999≤数2，算术表达式2≤999，此时可取1≤数3，算术表达式3≤255。

其中数2表示坐标轴的单位长度的步距，数3表示整个坐标轴有多少个单位长度，例如当数2＝50时，表示坐标轴的单位长度为10毫米，当数3＝5时，表示画出的坐标轴具有5个单位长度，总长为50毫米。

当画X轴时（即数1＝1时），应根据笔架的当前位置（距最左端打印位置的步距设为aa，应取0≤aa＋数2×数3≤480，或者0≤aa＋算术表达式2×算术表达式3≤480

执行此语句时根据数1，数2，数3的值自动绘制坐标轴并标出单位长度的分隔。

用途：利用此语可以方便地绘制各种尺寸单位长度的直角坐标轴。程序举例见第三册程序举例之27：直角坐标轴的绘制。

第四节　图案模式中指令的分类和综合使用

一、指令格式的分类

上述13种指令格式可以分为四种类型，各种类型的特点如下：

1．单字符指令，有三条：“A”，“H”，“I”，这些指令使用时后面不带数据，其中“H”、“I”指令后面跟有其它指令时可以连写，如写成“HP”，“HS2”，“IM100，－50”，“IC1”，“HX1，60，4”等等，即单字符指令后面不需要回车信号就能接下去执行下一个指令。

2．单数据指令，有5条：“C”，“L”，“S”，“P”，“Q”。这些指令使用时后面带一个数据，但P指令后面可以带有一组单数据指令。除P指令外，单数据指令后面若使用其它指令，只需用逗号作分隔，逗号必须放入双引号内，例如写成“L3，C2，IM200，0“，”S0，C0，Q1，HP”等，不需要输入回车信号，但输入回车信号也可以，而P指令使用时后面若使用其它指令时，应输入回车信号，否则以后的指令也当作字符串打印出来，例如进入图案模式后，用LPRINT“C1，PABC”；CHR$(13)；“C2，PABC”则可以按预定要求先打印兰色的字母ABC，转换颜色后再打印绿色的字母ABC，上式也可以写成LPRINT“C1，PABC”:LPRINT“C2，PABC”，同样也输入回车信号。

3．双数据指令有4条：“M”，“R”，“D”，“J”，这些指令使用时后面带2个数据，其中“D”，“J”指令后面还可以带多组数据，但每组要有2个数据，各数据之间必须用逗号分开，逗号应放在双引号内，双数据指令后面使用其它任何指令时必须插入回车信号，例如LPRINT CHR$(18)；“M240，－100”；CHR$(13)；“I”；“PBCD”或LPRINT CHR$(18)；“M240，－100”:LPRINT“IPBCD”，否则后面的指令不执行。

4．三数据指令即指令X，使用时后面必须带3个数据，并且只能带3个数据。即每次执行X指令只能画一条坐标轴，若后面跟其它指令必须输入回车信号，例如LPRINT CHR$(18)；“X1，100，4”:LPRINT“PX”

二、图案模式下指令的综合使用

根据上述各指令的语法特点，当各种指令联合使用时，一般先写单字符号指令或指令，随后写单数据指令，最后写需要回车信号作结尾的双数据指令，三数据指令或P指令，以节约指令本身占用的字节数。例如LPRINT“IC3，S2，L4，D100，0”

三、综合应用举例

见第三册程序举例之二十八：极坐标图形的绘制及程序举例之二十九：统计曲线的绘制。

第五章　LASER-310使用的BASIC语言（扩展部分）

本书第三章曾经介绍了LASER—310使用的基本 BASIC 语言共24条语句指令，此外还有7条语句指令用于扩展该机的功能，使计算机具有较高分辨率的屏幕图象处理功能，屏幕颜色变换功能，计算机的音乐功能，机器语句的编制和调用功能，对外围设备的控制功能等。其中屏幕图象处理与颜色变换的有关语句指令和函数以及机器语句的编制和调用详见第六章。

第一节　计算机音乐功能　　SOUND 语句指令

计算机的音乐功能是利用SOUND语句来实现的。SOUND语句格式如下：

SOUND 音调码，节拍码。

其中音调码的数值范围是0～31，节拍码的数值范码是1～9。音调码与节拍码都可以使用算术表达式，表达式的值不为整数时计算机将自动取整，表达式的值超范围时计算机出错。

一、音调码所对应的音名及其在钢琴键盘上的位置

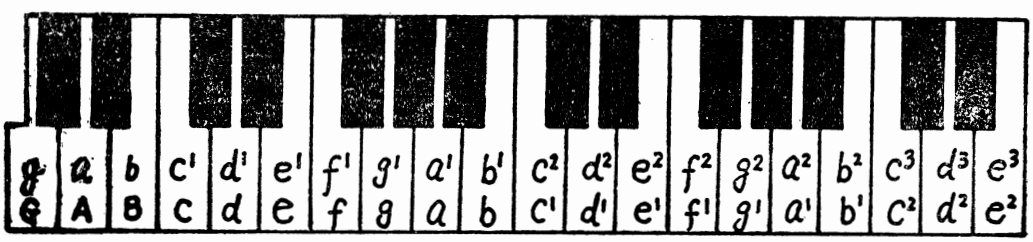
1. 音调码对应的音名：

音调码	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
音名	休止符	A2	A#2	B2	C3	C#3	D3	D#3	E3	F3	F#3	G3	G#3	A3	A#3	B3
音调码	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
音名	C4	C#4	D4	D#4	E4	F4	F#4	G4	G#4	A4	A#4	B4	C5	C#5	D5	D#5

2. 音调码与音名和钢琴键盘的位置对应关系（休止符的音调码为0）

音调码：2　5　7　10　12　14　17　19　22　24　26　29　30

音名　A#2　C#3　D#3　F#3　G#3　A#3　C#4　D#4　F#4　G#4　A#4　C#5　D÷5



音名：　A2　B2　C3　D3　E3　F3　G3　A3　B3　C4　D4　E4　F4　G4　A4　B4　C5　D5

音调码：　1　3　4　6　8　9　11　13　15　16　18　20　21　23　25　27　28　30

↑
中央 C

二、音调码、音名与大调音阶（或称大音价）简谱的对应关系：（休止符的音调码均为0）

1. C 调：

简　　谱	6 7 1 2 3 4 5 6 7 1 2 3 4 5 6 7 1 2
音 调 码	1 3 4 6 8 9 11 13 15 16 18 20 21 23 25 27 28 30
音　　名	A2 B2 C3 D3 E3 F3 G3 A3 B3 C4 D4 E4 F4 G4 A4 B4 C5 D5

2. G 调：（表中音阶比实际音阶低8度）

简　　谱	2 3 4 5 6 7 1 2 3 4 5 6 7 1 2 3 4 5
音 调 码	1 3 4 6 8 10 11 13 15 16 18 20 22 23 25 27 28 30
音　　名	A2 B2 C3 D3 E3 F*3 G3 A3 B3 C4 D4 E4 F*4 G4 A4 B4 C5 D5

3. F 调（表中音阶比实际音阶低8度）

简　　谱	3 4 5 6 7 1 2 3 4 5 6 7 1 2 3 4 5 6
音 调 码	1 2 4 6 8 9 11 13 14 16 18 20 21 23 25 26 28 30
音　　名	A2 A*2 C3 D3 E3 F3 G3 A3 A*3 C4 D4 E4 F4 G4 A4 A*4 C5 D5

4. D 调(表中音阶比实际音阶低8度)

简　　谱	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1
	•	•	•								•	•	•	•	•	•	•	• •
音　调　码	1	3	5	6	8	10	11	13	15	17	18	20	22	23	25	27	29	30
音　　名	A2	B2	C#3	D3	E3	F#3	G3	A3	B3	C#4	D4	E4	F#4	G4	A4	B4	C#5	D5

5. ♭B调(表中音阶比实际音阶低8度)

简　　谱	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4
	• •	•	•	•	•	•	•	•								•	•	•	•
音　调　码	1	2	4	6	7	9	11	13	14	16	18	19	21	23	25	26	28	30	31
音　　名	A2	A#2	C3	D3	D#3	F3	G3	A3	A#3	C4	D4	D#4	F4	G4	A4	A#4	C5	D5	D#5

6. A 调(表中音阶比实际音价低8度)

简　　谱	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4
	•	•	•	•	•	•	•								•	•	•	•
音　调　码	1	3	5	6	8	10	12	13	15	17	18	20	22	24	25	27	29	30
音　　名	A2	B2	C#3	D3	E3	F#3	G#3	A3	B3	C#4	D4	E4	F#4	G#4	A4	B4	C#5	D5

7. E 调(表中音阶比实际音阶低8度)

简　　谱	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7
	•	•	•	•								•	•	•	•	•	•	•
音　调　码	1	3	5	7	8	10	12	13	15	17	19	20	22	24	25	27	29	31
音　　名	A2	B2	C#3	D#3	E3	F#3	G#3	A3	B3	C#4	D#4	E4	F#4	G#4	A4	B4	C#5	D#5

8. B 调(或 ♭C 调)(表中音阶比实际音阶低8度)

简　　谱	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3
音 调 码	2	3	5	7	8	10	12	14	15	17	19	20	22	24	26	27	29	31
音　　名	A♯2	B2	C♯3	D♯3	E3	F♯3	G♯3	A♯3	B3	C♯4	D♯4	E4	F♯4	G♯4	A♯4	B4	C♯5	D♯5

9. ♯F 调(或 ♭G 调)(表中音阶比实际音阶低8度)

简　　谱	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6
音 调 码	2	3	5	7	9	10	12	14	15	17	19	21	22	24	26	27	29	31
音　　名	A♯2	B2	C♯3	D♯3	F3	F♯3	G♯3	A♯3	B3	C♯4	D♯4	F4	F♯4	G♯4	A♯4	B4	C♯5	D♯5

10. ♯C 调(或 ♭D 调)

简　　谱	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2
音 调 码	2	4	5	7	9	10	12	14	16	17	19	21	22	24	26	28	29	31
音　　名	A♯2	C3	C♯3	D♯3	F3	F♯3	G♯3	A♯3	C4	C♯4	D♯4	F4	F♯4	G♯4	A♯4	C5	C♯5	D♯5

11. ♭E 调(表中音阶比实际音阶低8度)

简　　谱	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1
音 调 码	2	4	6	7	9	11	12	14	16	18	19	21	23	24	26	28	30	31
音　　名	A♯2	C3	D3	D♯3	F3	G3	G♯3	A♯3	C4	D4	D♯4	F4	G4	G♯4	A♯4	C5	D5	D♯5

12. ^{b}A 调(表中音阶比实际音价低8度)

简　　谱	2•	3•	4•	5•	6•	7•	1	2	3	4	5	6	7	•1	•2	•3	•4	•5
音 调 码	2	4	5	7	9	11	12	14	16	17	19	21	23	24	26	28	29	31
音　　名	A*2	C3	C*3	D*3	F3	G3	G*3	A*3	C4	C*4	D*4	F4	G4	G*4	A*4	C5	C*5	D*5

三、节拍码与时值的对应关系

节 拍 码	1	2	3	4	5	6	7	8	9
时　值(秒)	1/8	1/4	3/8	1/2	3/4	1	3/2	2	3

四、节拍码与音符及简谱记法的对应关系。

节 拍 码	1	2	3	4	5	6	7	8	9
音符名称	十六分音符	八分音符		四分音符		二分音符		全音符	
简谱记法	6	6	6•	6	6•	6-	6--	6---	6-----

注：简谱记法中以"6"音为例

五、SOUND语句应用举例

使用SOUND 语句可以使LASER-310 计算机按照歌曲、乐曲的旋律发出音乐，每一个音符都要用音调码和节拍码来表示。

下面以歌曲"卖报歌"中的一段为例说明音调码与节拍码的用法：

$1=F\frac{2}{4}$

55 5 | 55 5 | 35 653 | 23 5 |

歌曲是用F调旋律，因此可利用F调的音调码与简谱对照表，其中“5”音应使用音调码16，第一个“5”是半拍，利用节拍码与音符与简谱记法的对照表可知节拍码是2，因此第一个“5”音用SOUND 16，2语句，同理第二个“5”音仍用SOUND 16，2语句，第3个“5”音用SOUND16，4语句，这样就把第一小节的旋律写下来了。第二小节相同，第三小节可以写为 SOUND 13，2，SOUND 16，2，SOUND 18，2，SOUND 16，1，SOUND 13，1，第四节用 SOUND 11，2，SOUND 13，2，SOUND 16，4 来表示，写入程序时，可把音调码，节拍码写入 DATA 语句，用 READA，B 的方法赋值给变量A，B，再用SOUND A，B的方法使计算机奏乐。

第三册程序举例之三十与程序举例之三十一为歌曲“我的祖国”的全曲旋律和奏乐程序，以及歌曲旋律“在那桃花盛开的地方”和奏乐程序。

第二节　计算机与外围设备的通讯联系

在LASER-310 的主机背面有一个计算机与外围设备进行通讯联系的输入/输出插口(移去主机背面的PERIPHERAL金属保护片即可看到，PERIPHERAL是外围设备接口的意思)，可与打印机，光笔、游戏棒联接，也可以同其它电气设备联接。这个插口又称为I/O插口，(I/O即英文INPUT/OUTPUT意为输入/输出) 具有30个引脚，其中数据传送线占用8个引脚(D_0～D_7)，端口地址线占用8个引脚(A_0～A_7)，控制线占用3个引脚 (即(I/O 请求，读，写三个控制指令)电源占用4个引脚 (高电平占用2个，低电平占用2个)，还有7个引脚为空接。

计算机通过这个插口与外部设备实现信息交换，如控制打印机打印字符或绘制图形，接受光笔或游戏棒的操作信号等等。

端口地址是用一个字节的二进制码表示，每个字节用八位二进制码表示，即从0000，0000～1111，1111。化为十进制数就是0～255，因此端口地址共有256个、在LASER-310计算机中，端口地址从0～127是保留给计算机系统用的，例如端口地址0～15是供打印机使用的，端口地址16～31是供磁盘驱动器使用的，端口地址32～47是供游戏棒使用的等等。还有128个端口地址(从128～255)则提供给用户使用。使用端口地址与外围设备进行联系时，需要使用2个特殊的指令：函数指令INP与语句指令OUT，简要介绍如下。

一、读取I/O接口数据的函数INP

函数使用格式：INP(算术表达式)，函数值的范围为0～255。使用INP函数时，可以把INP函数的值赋值给变量，例如A＝INP(算术表达式)，其中算术表达式的取值范围必须在0～255范围内，即算术表达式的值必须是端口地址中的某一个，否则出错，计算机执行INP函数指令时端口中3条控制线中的“I/O请求”与“读”2个控制信号有效，外围设备的数

据通过8位数据传送线传送给函数INP。因此函数值的范围为 0～255。INP 函数主要用于计算机向外围设备采集数据，也可以根据INP函数值了解外围设备的工作情况。例如用游戏棒作游戏时，游戏棒的操作动作将通过I/O接口把信息传送给函数 INP。详见第六章游戏棒部分。

二、把数据通过 I/O 接口输出到外围设备的语句指令OUT

语句格式：行号OUT地址表达式，数据表达式。

地址表达式的取值范围为0～255，用于选择 I/O接口地址，例如当地址表达式取值为0～15 中的任一个时，选中LASER-310的打印机，当地址表达式的取值为 32～47 任一个时，选中游戏棒。数据表达式的取值范围为 0～255，OUT 语句把数据表达式的值输送给地址表达式所选中的外围设备中去，实现计算机对外围设备的数据输出，例如当地址表达式取值范围在0～15时， 数据表达式的值就成为打印机输出的内容，包括打印字符，文字模式与图案模式的转换指令，绘图指令，各种控制指令等等，下面举例来说明这一点：

```
10  INPUT"CONTROL CODE OR ASCII (8 TO 212)"; J
20  FORI=0TO1:OUTI, J:NEXT
30  IFINKEY$<>""THEN30
40  IFINKEY$=""THEN40
50  RUN
```

执行这个程序时，根据 10 号语句的输入提示 CONTROL (控制)CODE(码)OR(或)ASCII 码 (8 TO 212)？ 若入若键入10↘则打印纸张推进一行，键入11↘则纸张倒回一行，键入18↘则进入图案打印模式，键入17↘则返回文字打印模式， 键入29↘则打印笔换颜色，在文字模式时，键入48～57 则打印0，1，2…，9数码，键入65～90则打印A～Z 共26个大字英文字母，键入97～122则打印 a～z 共26个小字英文字母，键入189～212则打印 α，β，γ，…，ω共24个希腊字母，等等。

程序中是使用OUTI，J语句来完成上述打印动作的。其中I表示端口地址， 这个程序中采用2个端口地址：0和1，也可以采用2和3，或4和5，…，或14和15。

程序的30～40号语句是控制程序运行方向的。50号语句相当于CLEAR:GOTO 10两条语句指令

利用这个程序的设计方法 (指 20 号语句)，可以在文字模式中实现纸张倒回一行的功能和24个希腊字母单独打印出来的功能，以及其它一些形状字符 (ASCII 码为 161～188)单独打印的功能。这些功能是LPRINT CHR$(表达式)不能实现的。

当LASER-310计算机通过 I/O 接口与用户的其它电气设备联接时(需附加硬件)，则可以通过OUT语句指令由计算机来指挥这些电气设备的工作了。

第三节　计算机内存机器码的输入与输出

LASER-310计算机的BASIC 解释系统，监控程序，用户输入的程序，变量、数据、堆栈等等，都是用机器码的方式存放在计算机的内存中，其中 BASIC 解释系统，监控程序都是放在内存ROM 中，其它内容则放在内存 RAM 中，放在 ROM 中的机器码只能读

出，不能写入，放在RAM中的机器码能够读出，也能够写入。当从键盘键入BASIC程序（或从磁带输入）程序时，计算机的BASIC解释系统自动地把BASIC程序译成代码存入RAM中。执行BASIC程序时，又自动地把变量，数据，堆栈存入RAM中。若使用特殊的语句指令POKE，则用户可以把机器码或指令代码，ASCII码直接写入RAM中，以写入BASIC程序或汇编程序。而使用特殊的函数指令PEEK，则可以把内存ROM或RAM中的机器码或指令代码读出。使用这两条指令，可以弥补LASER-310计算机在BASIC基本语言方面的许多不足，如检查RAM区顶端地址，检查BASIC程序的起始地址，终止地址，检查内存空余字节数，使计算机进入或退出示踪运行状态，写入光笔子程序（光笔子程序的写入与调用详见本书第六章），写入汉字显示子程序（详见第三册）及机器语言子程序的编写，使用，录制，读出等等。

一、LASER-310计算机内存地址分布

（一）未接RAM扩展模块时内存区的分布及地址

内存地址		内存区的使用情况	字节数 (1K=2^{10}=1024字节)
十六进制	十进制		
0000H 3FFFH	0000 16383	扩展BASIC解释程序区(ROM)	16K
4000H 67FFH	16384 26623	磁盘操作系统区(RAM)	10K
6800H 6FFFH	26624 28671	键盘矩阵区(RAM)	2K
7000H 77FFH	28672 30719	屏幕显示区（RAM）	2K
7800H 7AE8H	30720 31464	BASIC系统区(RAM) (745字节)	2K
7AE9H 7FFFH	31465 32767	用户RAM区(BASIC程序，变量区或机器码程序区)　(1303字节)	
8000H	−32768	用户RAM区（BASIC程序，变量区或机器码程序区）	14K
		FOR、GOSUB堆栈区(执行程序时使用)	
B7FFH	−18433	字符串堆栈区(根据CLEARn决定)	
B800H FFFFH	−18432 −1	空区(供扩展RAM及使用)	18K

（二）接上64K RAM扩展模块时内存区的分布及地址

内存地址		内存区的使用情况			字节数
十六进制	十进制				
0000H 3FFFH	0000 16383	扩展BASIC解释程序区(ROM)			16K
4000H 67FFH	16384 26623	磁盘操作系统区(RAM)			10K
6800H 6FFFH	26624 28671	键盘矩阵区(RAM)			2K
7000H 77FFH	28672 30719	屏幕显示区(RAM)			2K
7800H 7AE8H	30720 31464	BASIC系统区(RAM) (745字节)			2K
7AE9H 7FFFH	31465 32767	用户RAM区(BASIC程序，变量区或机器码程序区)(1303字节)			
8000H FFFFH	−32768 −1	用户RAM区(BASIC程序，变量区或机器码程序区) FOR、GOSUB 堆栈区(执行程序时使用) 字符串堆栈区(根据 CLEARn 决定)			32K
8000H BFFFH	−32768 −16385	第0区(机器码主程序区) 由OUT127，0选定			16K
C000H FFFFH	−16384 −1	第 1 区(16K) 机器码子程序区 由OUT127，1 选定	第 2 区(16K) 机器码子程序区 由OUT127，2 选定	第 3 区(16K) 机器码子程序区 由 OUT127，3 选定	48K

二、内存机器码读出的函数指令 PEEK 与机器码写入内存的语句指令 POKE

1. PEEK 函数

PEEK函数的使用格式为：PEEK(内存地址算术表达式)，其中内存地址算术表达式

的取值范围为－32768～32767，即LASER-310的64K寻址范围，可用PRINT PEEK(算术表达式)或LPRINT PEEK (算术表达式)直接读出内存的机器码(读出的机器码已自动转换成十进制数)，数值范围是0～255。 也可以在程序中使用，把PEEK函数的值赋值给变量，如A＝PEEK(算术表达式)

各内存区用PEEK函数读出的机器码代表不同的意义， 如在内存地址 0～16383 扩展BASIC 解释程序区读出的机器码表示用 Z-80 汇编语言编写的解释程序；在内存 16384～26623(当接有磁盘机时)磁盘操作系统读出的是DOS操作系统(用 Z-80 汇编语言写的系统程序)，在内存26624～28672键盘矩阵区读出的是键盘操作信息机器码，在28672～30719屏幕显示区读出的是屏幕显示的字符或象点的机器码 (与ASCII 码不同)，在 30720～31464 BASIC系统区读出的是由解释系统设定的一些具有监控功能的机器码，在31465～32767及－32768～－18433(未接有扩展模块)或－32768～－1(接有64KRAM扩展模块)用户RAM区读出的是用ASCII码及BASIC指令保留词所组成的BASIC程序， 用 ASCII码及其它特定数据码所组成的变量、数据、地址值，以及FOR、GOSOB堆栈，字符串堆栈等等，当用户RAM区存放机器码程序时，读出的是机器码(Z-80汇编)或机器码数据。

2. POKE 语句

POKE语句使用格式为： 行号POKE内存地址算术表达式，机器码算术表达式。 其中内存地址算术表达式，取值的范围除ROM区(0～16383) 不能用POKE语句写入机器码以外，其它与PEEK函数相同，机器码算术表达式的取值范围为0～255，这些机器码在各内存区同样也代表不同的意义，其各种不同的含义与PEEK函数所述相同。

POKE语句用于屏幕显示区时可以在屏幕上显示机器码所表示的字符或象点 (组成图案或汉字)， 用于BASIC系统区时可以改变监控状态， 用于用户RAM区时可以写入机器码程序或机器码数据， 或者使 BASIC 程序获得一些特殊的功能。 如检查内存空余字节数，程序保密功能等等。

三、POKE与PEEK在各内存区的使用

1. 扩展BASIC解释程序区(ROM)(0～16383)

只适合 PEEK 函数读出内存机器码， 这些机器码组成了 Z-80 汇编语言编写的扩展BASIC解释程序。

2. 磁盘操作系统区(RAM) (16384～26623)

当 LASER-310 主机接上磁盘驱动器时，该区存有 DOS 操作系统机器码程序，可用PEEK函数读出机器码程序。

3. 键盘矩阵区(RAM) (26624～28671)

键盘矩阵区的机器码在未按下任何键时，各地址中的机器码均为 255，当按下键盘的键时，在键盘矩阵区的各内存地址中即改变机器码的值，而内存地址 26624 中的机器码对

任一键入都会起变化，在矩阵区的其它地址中的机器码只对某些键入起变化，这可以用如下程序来测知：

```
10  INPUTI
20  PRINT"PEEK(26624+"; I; ")="; PEEK(26624+I)
30  FORJ=0TO200:NEXT:GOTO20
```

运行此程序后，先键入 0，则屏幕上显示 PEEK(26624+0)=255，此时再按键盘上任一键，可看到机器码255变为其它数值，放开键后，机器码又恢复为255；若键入1～2047之间的数，则不按键时机器码仍为 255，而按下键时，有的键按下时机器码改变为其它数，有的键按下机器码不改变。

根据上述原理，可以设计出一个子程序来测知是否按键，以此可作为程序中控制程序运行方向之用：

```
10  GOSUB100:PRINT"KEY HAS BEEN PUSHED!": END
100 IFPEEK(26624)<>255THEN100
110 IFPEEK(26624)=255THEN110
120 RETURN
```

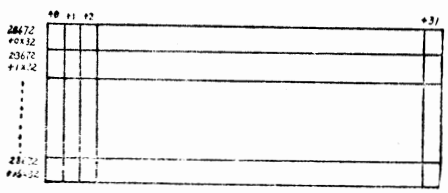
运行这个程序后，只要在键盘上按下任一键，屏幕即显示键 (KEY)已(HAS BEEN)按下！(PUSHED!)

4. 屏幕显示区(28672～30719)

屏幕显示区具有两种显表模式：文字模式与图象模式，屏幕显示区中 POKE 语句与 PEEK函数都可用。

(1) 文字显示模式，内存地址为28672～29183，其余地址无用。

上述地址值与屏幕上的位置对应关系如下：



使用 POKE 语句时，POKE 内存地址算术表达式，机器码算术表达式中内存地址为28672时，机器码所代表的字符就显示在第0行位第0列位。当内存地址为 28672+n×32+m时 (0≤n≤16，0≤m≤31)，机器码所代表的字符就显示在第 n 行位第 m 列位。使用 PEEK函数时，就把PEEK函数所指的内存地址中所表示的屏幕字符转换成机器码。

文字显示模式中屏幕显示字符与机器码的对应关系：

0	@	16	P	32		48	0	64	@	80	P	96		112	0
1	A	17	Q	33	!	49	1	65	A	81	Q	97	!	113	1
2	B	18	R	34	"	50	2	66	B	82	R	98	"	114	2
3	C	19	S	35	#	51	3	67	C	83	S	99	#	115	3
4	D	20	T	36	$	52	4	68	D	84	T	100	$	116	4
5	E	21	U	37	%	53	5	69	E	85	U	101	%	117	5
6	F	22	V	38	&	54	6	70	F	86	V	102	&	118	6
7	G	23	W	39	'	55	7	71	G	87	W	103	'	119	7
8	H	24	X	40	(	56	8	72	H	88	X	104	(	120	8
9	I	25	Y	41	)	57	9	73	I	89	Y	105	)	121	9
10	J	26	Z	42	*	58	:	74	J	90	Z	106	*	122	:
11	K	27	[	43	+	59	;	75	K	91	[	107	+	123	;
12	L	28	\	44	,	60	<	76	L	92	\	108	,	124	<
13	M	29	]	45	-	61	=	77	M	93	]	109	.	125	=
14	N	30	↑	46	.	62	>	78	N	94	↑	110	-	126	>
15	O	31	←	47	/	63	?	79	O	95	←	111	/	127	?

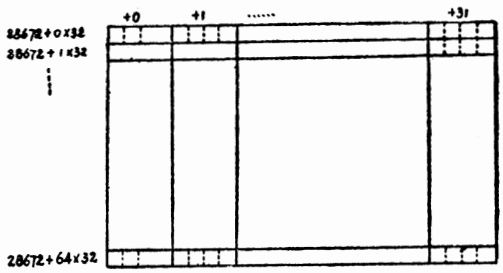
上表中左边一列为机器码(十进制形式)，右边一列为屏幕显示的字符，加上方框的字符呈反白显示状态(黑底白字)，从128～255共128个机器码，都是图案字符。其形状如下

X+0		X+1		X+2		X+3		X+4		X+5		X+6		X+7	
X+8		X+9		X+10		X+11		X+12		X+13		X+14		X+15	

当X＝128时，空白部分为绿色， X＝144空白部分为黄色， X＝160时，空白部分为兰色，当X＝176时，空白部分为红色，X＝192时，空白部分为浅黄色，X＝208时，空白部分为浅绿，当X＝244时，空白部分为紫色，当X＝240时，空白部分为橙色。

(2) 图象显示模式，当函数PEEK(算术表达式)括号内的值为28672～30719时，将屏幕显示的象点(每一个象点由4个象元组成，每个象元为1点)转换为机器码(十进制数表示)

上述地址在屏幕上的对应位置如下：(有关图象模式的使用详见第六章)



图象显示模式中屏幕显示的象点色彩与机器码的对应关系如下。

其中MODE(1)表示屏幕图象显示模式。MACHINE CODE 是机器码的意思，MCODE表示下面所对应的数码为机器码。COLOR—CODE表示以下所对应的数码为颜色码，颜色码从1～4。在绿色背景下，1表示绿色，2表示黄色，3表示兰色、4表示红色，在淡黄色背景下，1表示淡黄色，2表示淡绿色，3表示紫色、4表示橙色，每个机器码自左至右具有4个象元(即4点颜色)

MODE(1) MACHINE CODE

MCODE	COLOR-CODE				MCODE	COLOR-CODE			
0	1	1	1	1	1	1	1	1	2
2	1	1	1	3	2	1	1	1	4
4	1	1	2	1	5	1	1	2	2
6	1	1	2	3	7	1	1	2	4
8	1	1	3	1	9	1	1	3	2
10	1	1	3	3	11	1	1	3	4
12	1	1	4	1	13	1	1	4	2
14	1	1	4	3	15	1	1	4	4
16	1	2	1	1	17	1	2	1	2
18	1	2	1	3	19	1	2	1	4
20	1	2	2	1	31	1	2	2	2
22	1	2	2	3	23	1	2	2	4
24	1	2	3	1	25	1	2	3	2
26	1	2	3	3	27	1	2	3	4
28	1	2	4	1	29	1	2	4	2
30	1	2	4	3	31	1	2	4	4
32	1	3	1	1	33	1	3	1	2
34	1	3	1	3	35	1	3	1	4
36	1	3	2	1	37	1	3	2	2
38	1	3	2	3	39	1	3	2	4
40	1	3	3	1	41	1	3	3	2
42	1	3	3	3	43	1	3	3	4

44	1	3	4	1	45	1	3	4	2
46	1	3	4	3	47	1	3	4	4
48	1	4	1	1	49	1	4	1	2
50	1	4	1	3	51	1	4	1	4
52	1	4	2	1	53	1	4	2	2
54	1	4	2	3	55	1	4	2	4
56	1	4	3	1	57	1	4	3	2
58	1	4	3	3	59	1	4	3	4
60	1	4	4	1	61	1	4	4	2
62	1	4	4	3	63	1	4	4	4
64	2	1	1	1	65	2	1	1	2
66	2	1	1	3	67	2	1	1	4
68	2	1	2	1	69	2	1	2	2
70	2	1	2	3	71	2	1	2	4
72	2	1	3	1	73	2	1	3	2
74	2	1	3	3	75	2	1	3	4
76	2	1	4	1	77	2	1	4	2
78	2	1	4	3	79	2	1	4	4
80	2	2	1	1	81	2	2	1	2
82	2	2	1	3	83	2	2	1	4
84	2	2	2	1	85	2	2	2	2
86	2	2	2	3	87	2	2	2	4
88	2	2	3	1	89	2	2	3	2
90	2	2	3	3	91	2	2	3	4
92	2	2	4	1	93	2	2	4	2
94	2	2	4	3	95	2	2	4	4
96	2	3	1	1	97	2	3	1	2
98	2	3	1	3	99	2	3	1	4
100	2	3	2	1	101	2	3	2	2
102	2	3	2	3	103	2	3	2	4
104	2	3	3	1	105	2	3	3	2
106	2	3	3	3	107	2	3	3	4
108	2	3	4	1	109	2	3	4	2
110	2	3	4	3	111	2	3	4	4
112	2	4	1	1	113	2	4	1	2
114	2	4	1	3	115	2	4	1	4
116	2	4	2	1	117	2	4	2	2
118	2	4	2	3	119	2	4	2	4
120	2	4	3	1	121	2	4	3	2

122	2	4	3	3	123	2	4	3	4
124	2	4	4	1	125	2	4	4	2
126	2	4	4	3	127	2	4	4	4
128	3	1	1	1	129	3	1	1	2
130	3	1	1	3	131	3	1	1	4
132	3	1	2	1	133	3	1	2	2
134	3	1	2	3	135	3	1	2	4
136	3	1	3	1	137	3	1	3	2
138	3	1	3	3	139	2	1	3	4
140	3	1	4	1	141	3	1	4	2
142	3	1	4	3	143	3	1	4	4
144	3	2	1	1	145	3	2	1	2
146	3	2	1	3	147	3	2	1	4
148	3	2	2	1	149	3	2	2	2
150	3	2	2	3	151	3	2	2	4
152	3	2	3	1	153	3	2	3	2
154	3	2	3	3	155	3	2	3	4
156	3	2	4	1	157	3	2	4	2
158	3	2	4	3	159	3	2	4	4
160	3	3	1	1	161	3	3	1	2
162	3	3	1	3	163	3	3	1	4
164	3	3	2	1	165	3	3	2	2
166	3	3	2	3	167	3	3	2	4
168	3	3	3	1	169	3	3	3	2
170	3	3	3	3	171	3	3	3	4
172	3	3	4	1	173	3	3	4	2
174	3	3	4	3	175	3	3	4	4
176	3	4	1	1	177	3	4	1	2
178	3	4	1	3	179	3	4	1	4
180	3	4	2	1	181	3	4	2	2
182	3	4	2	3	183	3	4	2	4
184	3	4	3	1	185	3	4	3	2
186	3	4	3	3	187	3	4	3	4
188	3	4	4	1	189	3	4	4	2
190	3	4	4	3	191	3	4	4	4
192	4	1	1	1	193	4	1	1	2
194	4	1	1	3	195	4	1	1	4
196	4	1	2	1	197	4	1	2	2
198	4	1	2	3	199	4	1	2	4

200	4	1	3	1	201	4	1	3	2
202	4	1	3	3	203	4	1	3	4
204	4	1	4	1	205	4	1	4	2
206	4	1	4	3	207	4	1	4	4
208	4	2	1	1	209	4	2	1	2
210	4	2	1	3	211	4	2	1	4
212	4	2	2	1	213	4	2	2	2
214	4	2	2	3	215	4	2	2	4
216	4	2	3	1	217	4	2	3	2
218	4	2	3	3	219	4	2	3	4
220	4	2	4	1	221	4	2	4	2
222	4	2	4	3	223	4	2	4	4
224	4	3	1	1	225	4	3	1	2
226	4	3	1	3	227	4	3	1	4
228	4	3	2	1	229	4	3	2	2
230	4	3	2	3	231	4	3	2	4
232	4	3	3	1	233	4	3	3	2
234	4	3	3	3	235	4	3	3	4
236	4	3	4	1	237	4	3	4	2
238	4	3	4	3	239	4	3	4	4
240	4	4	1	1	241	4	4	1	2
242	4	4	1	3	243	4	4	1	4
244	4	4	2	1	245	4	4	2	2
246	4	4	2	3	247	4	3	2	4
248	4	4	3	1	249	4	4	3	2
250	4	4	3	3	251	4	4	3	4
252	4	4	4	1	253	4	4	4	2
254	4	4	4	3	255	4	4	4	4

本书第三册在屏幕上显示汉字的程序，就是利用POKE语句来完成的，而把汉字显示的机器码存入内存以及取出使用，则是分别使用 POKE 语句和PEEK函数， 详见第三册中GOSUB语句的程序举例。

5. BASIC系统区(RAM)(30720～31464)

BASIC系统区存放着BASIC监控程序的系统参数，例如内存地址30744中存放屏幕视频方式的控制码，可键入？PEEK(30744)↙读出，也可键入POKE30744,1↙改变屏幕的视频方式(详见本书第六章屏幕功能)。

在内存地址30862与30863中存放机器码程序(即汇编语言编写的程序)的起始地址，可键入？PEEK(30862)；PEEK(30863)↙读出，屏幕显示74与30两个数，即机器码程序的

起始地址为30×256+74=7754，也可用POKE30862，X1：POKE 30863，X2加以改变，X1与X2的数值根据用户机器码程序的起始地址而定(如使用光笔子程序时 X1=18，X2=123。即光笔的机器码子程序起始地址为123×256+18=31506。详见第六章光笔部分)

在内存地址30884与30885中存放BASIC程序（即用户输入的BASIC程序）的起始地址，可键入？PEEK(30884)；PEEK(30885)↵则屏幕显示233与122两个数值，即BASIC程序的起始地址为122×256+233=31465，也可以用POKE语句改变其起始地址（详见第三册GOSUB语句程序举例及第六章光笔程序）

在内存地址30897与30898与中存放着LASER-310计算机内存的终止地址，若键入？PEEK(30897)；PEEK(30898)↵时，屏幕显示255与183两个数，（未接上RAM扩展模块时）或显示255与255两个数（接上64K扩展模块时），分别表示内存的终止地址是183×256+255=47103与255×256+255=65535。由于这两个数均超过32767，应减去65536分别成为-18433与-1。这就是本节介绍POKE与PEEK语句开始时内存区的分布中的最终地址。

在内存地址30969与30970中存放着BASIC程序的终止地址，这一地址也是变量区的起始地址，可键入？PEEK(30969)；PEEK(30970)↵，当未输入BASIC程序时，屏幕显示235与122两个数，表示终止地址为122×256+235=31467，当输入BASIC程序时，该地址将增大如键入10 PRINT 3+5↵之后，再键入？PEEK(30969)；PEEK(30970)↵屏幕显示244与122两个数，表示上述程序的终止地址为122×256+244=31476。这个地址与BASIC程序起始地址31465的差值31476-31465=11表示上述程序占用内存的字节数为11个字节。

BASIC程序的终止地址与内存终止地址的差值即表示内存BASIC程序空余字节数，以未输入BASIC程序时为例，当接有64K扩展模块时，空余字节数为65535-31467=34067，当未接扩展模块时，空余字节数为47103-31467=15636。

6．用户RAM区（-32768～—18433与31465～32767或-32768～-1与31465～32767）

在用户RAM区中用PEEK函数可以读出BASIC程序在内存中的存放情况，变量区的存放情况，FOR、GOSUB堆栈及字符串堆栈的存放情况，也可以用POKE语句把机器码程序或数据写入该内存区中（参见第三册GOSUB语句程序举例及第六章光笔子程序的写入）。

四、转机器码子程序函数指令USR(X)

该函数使用的格式为：变量=USR(算术表达式)，例如A=USR(0)使用此函数指令必须具备如下条件(具体例子见第六章光笔部分)。

1．在用户RAM区中必须存在机器码子程序，在LASER-310计算机中，机器码子程序必须是用Z-80汇编语言译成的机器码。如果不存在机器码子程序，则使用此函数将出错，出错提示为FUNCTION CODE ERROR(函数功能出错)

2．机器码子程序可用POKE语句写入，也可以从磁带读入，但磁带上的程序必须是

机器码程序

3．机器码子程序中第一个被执行的指令所在RAM区的地址必须事先存放在内存地址的30862(低位字节)与30863(高位字节)中，存放的方法是：POKE 30862，低位地址值：POKE30863，高位地址值。否则出错，出错提示同 1。

4．算术表述式的值将直接赋值给变量，对执行机器码程序没有影响。

第六章　LASER-310屏幕功能、光笔、游戏棒

LASER-310 计算机屏幕功能相当丰富，本书的第一册作了一些简单介绍，本章将对屏幕功能作全面的介绍。并提供编写光笔，和游戏棒程序的方法。

第一节　屏幕功能

LASER-310 计算机具有两种模式的屏幕显示方式：文字模式和图象模式，计算机接通电源后，自动进入文字显示模式，在直接方式下不宜使用图象模式，通常通过运行程序进入图象模式，程序运行结束后计算机自动返回文字模式。

一、文字显示模式MODE(0)

文字模式是LASER-310 计算机屏幕功能的主要部分，程序的编制、录制、读入、打印机的使用等都必须在文字模式中进行。

文字模式通常是在浅绿色背景下显示黑色字体，开机后自动进入这种状态（视觉上感觉较舒适），也可以根据操作者的需要选用深绿色背景显示白色字体，这种状态只须在开机前按住CTRL键再接通主机电源开关，待屏幕显示白色字体，即可放开CTRL键。

下面详细介绍在文字模式下黑白字体转换的方法，背景色和前景色的变换，以及适用于文字模式的操作指令和语句指令。

1．黑白字体的转换

黑白字体的转换共有三种方法。上面已介绍了一种方法，其余两种方法是用POKE指令和反白转换键 INVERSE 具体说明如下：

(1) POKE 30744，0和POKE30744，1

① 当开机后屏幕显示白色字体时，若键入命令POKE30744,0↵就能使屏幕从深色背景转为浅色，在屏幕上原来显示的所有白色字体，全部变成黑色字体，此后从键盘键入(或从磁带输入)的所有字符及程序均以黑色字体显示。这一命令的作用相当于解除了开机时按住CTRL键所产生的作用，其本质是全屏幕的黑白反转(图案字符例外)

② 当开机后屏幕显示黑色字体时，若键入命令POKE 30744,1↵就能使屏幕从浅色背景转为深色，在屏幕上所有的黑色字体全部变成白色字体；如果屏幕上原先有一些白色字体(按通常方法开机后，再用 INVERSE 键使键盘写入一些白色字体)；则这部分白色字体都转变成黑色字体，这些字体的背景色也同时反转，此后键盘输入(或从磁带输入)的所有字符及程序均以白色字体显示。这一命令的作用是全屏幕黑白反转(图案字符例外)。

③ POKE30744,0和POKE30744,1的进一步认识。

在LASER-310计算机内存地址 30744 中，可存放0～255共256种机器码。当存放机

器码为0时，所有按通常方法(指不使用 INVERSE 键， 下同)键入的字符(不包括图案字符)均为黑色字体，(而当机器码不等于0即为1～255中任一数码时， 所有按通常方法键入的字符(不包括图案字符)均为白色字体，所以，POKE 30744,0命令执行后可以保证按通常方法键入的字符为黑色字体，而POKE 30744,n (n为1～255中任何一个数码)命令执行后可以保证按通常方法键入的字符为白色字体。

除了上述作用以外，改变内存地址30744中机器码还起着全屏幕黑白的反转作用(图案字符例外)。所以当内存地址30744中机器码为0时(按通常方法开机时，扩展BASICA解释程序就住该地址送入机器码0)， 若键入 POKE 30744,0 则不起任何作用，但键入POKE 30744,n (1≤n≤255) 即可使全屏幕黑白反转，并且使此后键入的均字符为白色字体。当内存地址30744中机器码不等于0时(例如按住CTRL键再开机， 使屏幕显示白色字体，深色背境，此时扩展BASIC能释程序已住该地址送入机器码96)， 若键入POKE 30744, n (如上述开机后取 n = 96,则不起任何作用)，当n等于原来存放在地址30744中的机器码时，则不起任何作用；当n不等于原存放在地址30744中的机器码时，则不管n是否为0，均能使全屏幕黑白反转：但键入POKE 30744,0才能使此后键入的字符显示黑色字体，否则只能使全屏幕黑白反转，不能使此后键入的字符转为黑色。例如按住CTRL键后开机，使屏幕显示深色背景， 白色字体、此时再键入POKE 30744,1可使当前屏幕黑白反转， 但此后键入的字符仍为白色字体。 如果在这种情况下键入操作命令或程序， 计算机是无法执行的，并给予出错提示，其原因是键入白色字体组成的指令，必须在深色背景 (仅需键入的一行为深色背景)中才能被执行(对黑色字体组成的指令，必须在键入的一行为浅色背景时才能被执行)。当然这种情况是在未使用 INVERSE 键的条件下才能成立(详见 INVERSE 键的说明)。

(2) 使用 INVERSE 键转换黑白字体，(图案字符例外)

按一次 INVERSE 键后，键盘输入的字符相对于当前屏幕显示的字体进行黑白反转，若当前屏幕显示黑色字体，按反转键后，则使键入的字体转为白色，若当前屏幕显示的白色字体， 按反转键后， 则使键入的字体转为黑色。 这种转换仅仅是对键盘输入起作用。(对磁带输入不起作用)，没有使全屏幕黑白反转的功能。

一旦使用反转键后，所有从键盘输入的操作指令，语句指令以及变量，数字和符号等均失去原来的意义，不管其是否与背境色调统一，只能作为说明性的提示存放在程序中。除非退出反转状态，才能恢复键盘输入的原有意义。

退出反转状态有三种情况：

① 再按一次 INVERSE 键，即再次反转，就能退回到原来状态。

② 按一次 BREAK 键，即中断命令(实际上对反转命令也起作用)使反转功能停止，从而退出反转，

③ 关机后重新开机，反转状态自行消失。

2. 背景色和前景色的转换 COLOR 语句

在文字模式中，背景色和前景色的转换是针对图案字符组成低解析图象的需要而设置的，特别是前景颜色的转换仅仅适用于图案字符。

颜色转换命令的一般格式是COLOR[I], J或COLOR I (I = 1～8; J = 0～1) 其中方

括号表示该项可以省略，即颜色转换命令有三种写法：COLOR，J；COLOR I；和COLOR I，J分别说明如下：

① COLOR，J 背景色的转换。键入COLOR，1↙全屏幕显示橙色背景(不管屏幕显示黑色字体还是显示白色字体，即与屏幕背景颜色的深浅无关)，键入 COLOR，0↙全屏幕显示绿色背景（亦与屏幕背境颜色的深浅无关)，开机时，屏幕显示自动处于这种状态。

② COLOR I 前景颜色的转换：即图案字符显示色彩部分的颜色转换

COLOR1	绿色 (GREEN)
COLOR2	黄色 (YELLOW)
COLOR3	兰色 (BLUE)
COLOR4	红色 (RED)
COLOR5	浅黄色 (PALE YELLOW)
COLOR6	浅绿 (PALE GREEN)
COLOR7	紫色 (PURPLE)
COLOR8	橙色(ORANGE)

键盘上共有16个图案字符，这些图案字符的显色部分(除黑色外)均按受COLOR I 的命令。COLOR I对其它键入的字符不起作用；COLOR I 命令只对命令执行后键入的图案字符起作用，对先前键入的图案字符不起作用，因此在同一屏幕上可以保留九种色彩(包括黑色)，COLOR，J 命令对图案字符的色彩不起作用，即背境颜色改变时，图案字符显示的色彩保持不变。

在LASER-310计算机上。图案字符对应的 ASCII码是128～191共64个，这些ASCII码可分为四组，每组16个ASCII码对应于在键盘标识 2 位置的16个图案字符，也即每个图案字符有4个对应的ASCII码，可列表如下：

ASCII码	图案字符	ASCII码	图案字符
128; 144; 160; 176		136; 152; 168; 184	
129; 145; 161; 177		137; 153; 169; 185	
130; 146; 162; 178		138; 154; 170; 186	
131; 147; 163; 179		139; 155; 171; 187	
132; 148; 164; 180		140; 156; 172; 188	
133; 149; 165; 181		141; 157; 173; 189	
134; 150; 166; 182		142; 158; 174; 190	
135; 151; 167; 183		143; 159; 175; 191	

表中字符涂黑部分只能显示黑色，不受COLOR I的影响，字符空白部分即为显示彩色部分。在文字模式中，COLOR I命令只对这些部分起转换颜色的作用。

除了可以从键盘直接输入上述图案字符外，还可以利用 ASCII码来输入图案字符，采用取ASCII字符函数 CHR$ 就可以实现这个目的。例如键入 PRINT CHR$(137)，就可以显示表格中ASCII码为137所对应的图案字符。表格中同一栏中的 ASCII 码显示相同形状和相同色彩（在相同的COLOR I命令下）的图案字符，并不因ASCII码的不同而显示不同色彩。因此，对于产生图案字符而言，表格中同一栏内的四组ASCⅡ码没有任何区别。

③ COLORI，J背境颜色与前境颜色的同时转换。

此命令实际上是把 ①、② 两条命令结合起来并成一条命令，其效果等于COLOR I: COLOR，J。

3．适用于文字模式的操作指令和语句指令。

① 操作指令，凡LASER-310计算机提供的操作命令(共15条，见第一册附录) 都可以在文字模式中使用。

② 语句指令：LASER-310计算机共提供了30条语句指令(见第一册附录)，除SET，RESET两条指令只能用图象模式之外，其余28条语句均指令可在文字模式中使用。

③ 在文字模式中可用直接方式使用的语句指令共17条

FOR，TO，NEXT，IF，THEN，ELSE，LET，PRINT，STEP，GOTO，POKE，CLS，LPRINT，COLOR，CLEAR，OUT，SOUND，

此外，所有数值函数，字符串函数，逻辑函数及特殊函数均可在直接方式中使用。

二、图象显示模式 MODE(1)

图象模式是LASER-310 计算机在屏幕上显示高解析图象所采用的一种模式。进入图系模式，一般通过程序实现，在文字模式中一般不采用直接方式进入图象模式，这是因为在图象模式中计算机不接受键盘输入的操作指令(除BREAK 此外)。

计算机进入图象模式后，遇到下列情形之一便退出，回到文字模式；

① 图象模式运行完毕，或在程序执行中遇到 END语句，即自动退回到文字模式。

② 图象模式运行中，遇到STOP语句或用BREAK 命令，则退回到文字模式。

③ 图象运行中，遇到 PRINT 语句，CLS语句，INPUT语句、MODE(0) 语句，执行后即退回到文字模式。

下面具体介绍在图系模系中背景色和前景色的转换，SET，RESET 语句指令和POINT函数指令的作用以及适用于图象模式的操作指令和语句指令。

1．图象模式中的背景色和前景色的变换

在图象模式中，背景色和前景色的转换是针对SET(画点)和RESET(擦点)以及POINT(检测颜色)这三条指令进行高解析图象的绘制而设置。颜色指令的一般格式与文字模式相

同，也有三种写法：① COEOR，J ② COLOR I ③ COLORI，J (I=1～4，J=0～1)但转换情况有所不同，分别说明如下：

① COLOR，J 背景颜色的转换。在图象模式中，执行到COLOR，1 语句时，全屏幕显示浅黄色背景，且同时改变所有的前景颜色（在文字模式中背景色的改变对前景颜色不起作用)。当执行到COLOR，0 语句时，全屏幕显示绿色背景，并同时改变所有的前景颜色。

从文字模式转入图象模式时，若转入前文字模式处于绿色背景，则转入图象模式时背景色不变，若转入前文字模式处于橙色背景，则转入图象模式时背景色转为浅黄。

从图象模式退回文字模式时，若退出前图象模式处于绿色背景，退回文字模式时背景色不变；若退出前图象模式处于浅黄色背景，退回文字模式时背景转为橙色。

② COLOR I 前景颜色的转换，即 SET(画点)颜色的转换。

显色命令	前景颜色(背景色为绿色时)	前景颜色(背景色为浅黄时)
COLOR 1	绿	浅黄
COLOR 2	黄	浅绿
COLOR 3	兰	紫
COLOR 4	红	橙

另外4个颜色码的作用是 COLOR5=COLOR 1；COLOR6=COLOR2；COLOR7=COLOR 3；COLOR8=COLOR 4；因此在图象模式中，颜色码5，6，7，8实际上已经失去了在文字模式中所具有的作用，若使用检测颜色函数指令 POINT，也只能检测到1，2，3，4这四个颜色码，(即使用颜色码5，6，7，8后画点也是如此)。此外，还可以从上述表格中知道，前景色的转换还将受背景景色的影响。例如用命令 COLOR 4。在文字模式中前景色必定是红色，而在图象模式中，前景色就不一定是红色，只有在绿色背景中才能使前景色转为红色。

最后，与文字模式不同的是在图象模式中屏幕上已经显示的前景颜色也会由于背景颜色的转变而跟着转变。（在文字模式中，屏幕上已经显示的前景颜色是不会改变的，除非在该位置用图案字符换上其它颜色)。

因此，在图象模式中，颜色显示具有两种状态，一种状态是可显示绿、黄、兰、红四种颜色，其中绿色为背景色。另一种状态也可显示四种颜色：浅黄、浅绿、紫、橙，其中浅黄是背景色。

3 COLORI，J 背景颜色和前景色的同时转换。

此命令相当于COLORI：COLOR，J 即同时实现背景色和前景色的转换。

2. SET，RESET语句指令及POINT函数指令在图象模式中的使用

相对于文字模式而言，图象模式中可在屏幕上进行高解象度的绘图，其解象度是64行每行128个点。

① 在屏幕上绘图是利用SET(X、Y)语句来进行的(其中X代表屏幕水平方向的位置，其值为0≤X≤127；Y 代表屏幕垂直方向的位置，其值为 0≤Y≤63)，X=0时绘制在屏幕的最左边；X=127时绘制在屏幕的最右边；Y=0 时绘制在屏幕的最上边；Y=63 时绘制在屏幕的最下边。

若把屏幕的底线作为横坐标，左边线作为纵坐标，则 SET(X、Y)语句实质是在屏幕上横坐标为X，纵坐标为63-Y的坐标位置画出一个点。当变量X，Y变化时，就可以相应地画出直线，曲线及各种图画。

X、Y可以为数值型表达式，对表达式的值计算机一律取整。

② 擦除已画在屏幕上的点是利用语句RESET(X、Y)来进行的，其中 X、Y 可为表达式，取值范围同SET函数中的情况一致。使用RESET语句可以把画在屏幕上的一些不需要的点擦去，(擦去后显示原有的背景色)，对图形进行某些修改。用在程序中可以使图形发生动态变化。

③ 检测已画在屏幕上的点是利用检测函数 POINT (X、Y)来进行的，其中X、Y可为表达式，取值范围同 SET 线 RESET 语句一致。函数POINT (X、Y)不能象SET，RESET一样为单独的语句指令使用，只能作为函数得出检测到的颜色码的值 (1～4)。如果检测(X、Y)位置上的点没有用SET画上不同于背景的颜色，则 POINT(X、Y)的函数值＝1 (表示该点的颜色与背景色一致)，如果检测 (X、Y) 位置上的点显示不同于背景色的点，则函数值POINT(X、Y)≠1；例如在(X、Y)位置上的点显示黄色(或浅绿色)时，POINT(X、Y)＝2；显示兰色(或紫色)时，POINT(X、Y)＝3；显示红色(或橙色)时，POINT(X、Y)＝4。

3. 适用于图象模式的操作指令和语句指令

(1) 操作指令：LASER-310计算机提供的15条操作命令除BREAK外，均不能在直接方式下从键盘上输入，但如果把某些操作命令写进程序，计算机还是执行的，其执行的情况分为两种，一种是执行该命令后退回到文字模式，另一种是执行命令后不退回到文字模式。

① 在程序运行中执行操作命令(非键盘输入)不退回到文字模式：

RUN, COPY

其中RUN具有清变量，清数组，清除FOR、GOSOB堆栈，把DATA指针恢复到初始位置的作用。

COPY命令虽执行，打印机动作，但PP-40打印机不能如实打印屏幕上的图象。

② 在程序运行中执行操作命令(非键盘输入)并立即退回到文字模式：

LIST, LLIST, CLOAD, CRUN,

这些命令的作用与它的在文字模式中完全相同。

(2) 语句指令：LASER-310 计算机提供的 30 条语句指令全部可以在图系模式中被执但真正在图系模式中使用的语句指令只有25条，其余5条语句虽能被执行，但一旦执行即退回到文字模式，分述如下：

① 执行后退回到文字模式的语句指令。

INPUT, PRINT, CLS, STOP, END, MODE(0)。

② 执行后不退回文字模式的语句指令：

READ, DATA, FOR, TO, NEXT, THEN, ELSE, IF, LET, STEP,
DIM, GOSUB, RETURN, GOTO, REM, POKE, COLOR, RESTORE,
SET, RESET, OUT, SOUND, LPRINT, CLEAR, MODE(1)

其中当程序进入图象模式后，可以再使用 MODE(1)语句指令，此时该指令的作用相当于

文字模式中的CLS，即起清屏幕的作用。

(3) 图象模式以各种方式退回到文字模式时，(除了遇到清变量语句或指令外)其中的运算变量和数据不清除。仍保留在内存中。

程序举例：屏幕上创作彩色图画，见第三册程序举例之三十。

第二节 光 笔

本节介绍光笔与电脑的连接方法，以及光笔程序输入电脑主机后如何调节电视屏幕的亮度使光笔正常发挥其作用并讨论光笔程序的编写，录制等问题。

一、光笔的拆装、调试与注意事项

1. 参见第一册图1—4。先关闭主机电源，再将光笔介面 LI-20 插进外围设备接口(PERIPH ERAL)(如果这个接口已插有打印机介面，应先拔除此介面；但应注意检查主机电源是否已关妥)，光笔介面的接口应完全插入主机接口后，再开启主机，此时屏幕应显示图1—6(第一册)所示的正常情况，若显示不正常，应即关闭主机电源，拔出光笔介面10秒钟后，再重复上述步骤。

2. 调试光笔之前应先编写好光笔程序，使主机运行光笔程序后再进行。或者用录音机把光笔程序示范带的程序输入主机，先键入 CRUN 命令，再按下录音机上的放音键(PLAY)，最后按一次输入键(RETURN键)

3. 主机进入光笔程序后，即可调试光笔，方法如下：

(1) 手持光笔如执笔状态，以光笔末端黑色触点接触电视屏幕，(光笔必须垂直于电视屏幕)将黑色触点轻压屏幕，此时屏幕应有水平兰线闪烁反应，电脑发声，光笔即可离开屏幕，此时，屏幕上发现光笔点过的标记(如点、图形等)这说明光笔已能正常工作，但尚须进行一些调节工作如下述2条。

(2) 如果屏幕只有闪烁反应，没有水平蓝线出现，电脑不发声，此时应把屏幕亮度加大。

如果屏幕只有水平蓝线闪烁，电脑不发声，此时应把屏幕亮减小。

(3) 如果在光笔垂直于屏幕的情况下，光笔感知的信号偏左，此时应把屏幕亮度稍微减小一点。

如果光笔感知的信号偏右，此时把屏幕亮度稍微加大一点。

4. 注意事项：拆装光笔介面时必须先关闭主机电话；主机外用设备接口以及光笔介面都不允许接触任何液体。否则损坏主机和介面。

二、光笔程序的编写

1. 光笔程序的编写与普通程序编写情况有些不同，在编写光笔程序之前，首先要做好下列准备工作：

① 先写入光笔子程序。在计算机内存地址的31465～32197（此为十进制的内存地址，

计算机内存区域的划分见第五章说明）写入机器语言光笔子程序的方法一般有两种：一种是用已录好光笔子程序的磁带送入，另一种是用POKE语句在上述内存区域写入光笔子程序的机器码。在光笔示范程序的磁带（随机供应）第3段至第7段程序中已录制了存放于上述内存区域的光笔子程序，这个子程序用LIST命令是不显示的，但可以用 PRINT PEEK (X)（X = 31465～32197）显示程序的各个机器码。读者如有兴趣，也可以自己录制（录制的详细方法包括用POKE语句写入光笔子程序下面介绍）以便编写光笔程序。

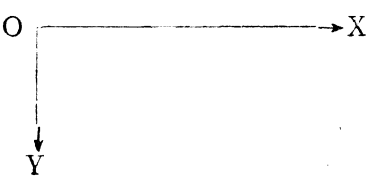
② 将光笔程序的内存起始地址改为32198。方法是键入 POKE 30884，198:POKE 30885，125↙在LASER-310计算机中，内存地址30884与30885是存放 BASIC 程序的起始地址编码的。（内存地址30969与30970则存放BASIC程序的终止地址编码），在30884中存放的是低位地址编码，在30885中存放高位地址编码，把高位地址编码乘以256再加上低位编码，就得到内存地址，因此，键入上面两个语句之后，BASIC 程序的起始地址就变为125 × 256 + 198 = 32198。（编写普通程序时，BASIC程序的起始地址是31465，开机后可用 PRINT PEEK(30884)；PEEK(30885)↙显示低位地址编码为233，高位地址编码为122，程序的起始地址为122 × 256 + 233 = 31465）

③ 用NEW命令请除内存中光笔示范程序，由于在上一步骤中把 BASIC 程序的起始地址改为32198。因此NEW命令只清除从内存地址32198开始的BASIC程序部分，而内存地址从31465到32198之间的光笔子程序则保留下来，此后写入的光笔程序就能为使用光笔服务了。

2. 光笔程序的编写

① 光笔程序必须调用在内存地址31465～32197中的光笔子程序。调用的方法是使用USR(X)转子程序函数，并且先设定函数USR(X)的内存地址。即机器语言子程序的首指令地址，该地址存放在两个专用的内存地址30862和30863，在内存地址30862中存放函数USR(X)的低位地址，在30863中则存放高位地址。在光笔程序中须用POKE 30862，18:POKE30863，123设定函数USR(X)的地址。按照上面介绍的方法容易算出这个地址为123 × 256 + 18 = 31506。该地址即为光笔子程序第一条被执行指令的地址，

② 调用光笔子程序后，可采用文字模式或图系模式进行工作，若用 MODE(1) 语句进入图象显示模式，建立X—Y座标系如下：



X的取值范围为0～127，Y的取值范围为0～63。

当光笔接触屏幕时，在屏幕亮度正常的情况下，光笔接触处的位置坐标(X，Y)将被计算机检测出来，并把X、Y的坐标数值分别送入内存地址31504和31505中，被光笔子程序采用。它们的坐标数值也可以用 PEEK 函数取出：X = PEEK(31504)；Y = PEEK (31505)，在编写光笔程序时，都是利用X、Y的数值确定屏幕显示的内容的。

③ 光笔试用程序说明：

```
10  POKE30862, 18:POKE30863, 123
20  MODE(1):COLOR2
30  A=USR(X)
40  X=PEEK(31504):Y=PEEK(31505)
50  IFX<2ORX>125THEN30
60  IFY<2ORY>61THEN30ELSESOUND31, 2
70  FORI=-2TO2:SET(X+2, Y+I):SET(X+I, Y-2):SET(X-2, Y+I)
80  SET(X+I, Y+2):NEXT:GOTO30
```

10号语句设定函数USR(X)调用光笔子程序内存地址，20号语句进入图象显示模式，并指定显示颜色为黄色，30号语句为调用内存中的机器语言子程序，40号语句取出光笔子程序在内存地址31504和31505中的坐标值，这个坐标值是光笔接触屏幕时送入内存中的，50号语句是判别光笔在屏幕水平方向的位置超范围时不显示图形，60号语句判别光笔在屏幕垂直方向的位置超范围时不显示图形，若光笔在屏幕合适的范围内时，就用SOUND语句发出声音，70号语句和80号语句则按照光笔在屏幕接触的位置画一个方框，这个试用程序是为调试光笔子程序是否能正常工作，调试屏幕亮度，调试光笔的作用情况用的。

在做好光笔程序编写的三个准备工作之后，可用键盘输入或磁带输入上述试用程序。

④ 光笔应用程序介绍，见第三册程序举例之33：光笔作图程序。

3. 光笔程序的录制

光笔程序运行、调试完成后，可录制在磁带上以便下次使用。在录制工作之前，应该先键入POKE30884, 233:POKE30885, 122:POKE31507, 205↵

其中前面2个POKE语句是恢复BASIC程序的地址到31465，以便录制时把光笔子 序一起录在磁带上，以后再用这段磁带输入计算机时就把机器语言子程序部分和光笔程序都存入计算机中，并可直接用RUN命令运行这个程序，因为在机器语句子程序中已写入POKE30884, 198:POKE30885, 125两个语句，因而一旦运行光笔程序（光笔正常工作之后），BASIC程序的起始地址就转到32198。这一点务必请读者注意。

最后一个POKE语句是在光笔子程序中存入机器码205。其作用是把光笔子程序和光笔应用程序连接起来。

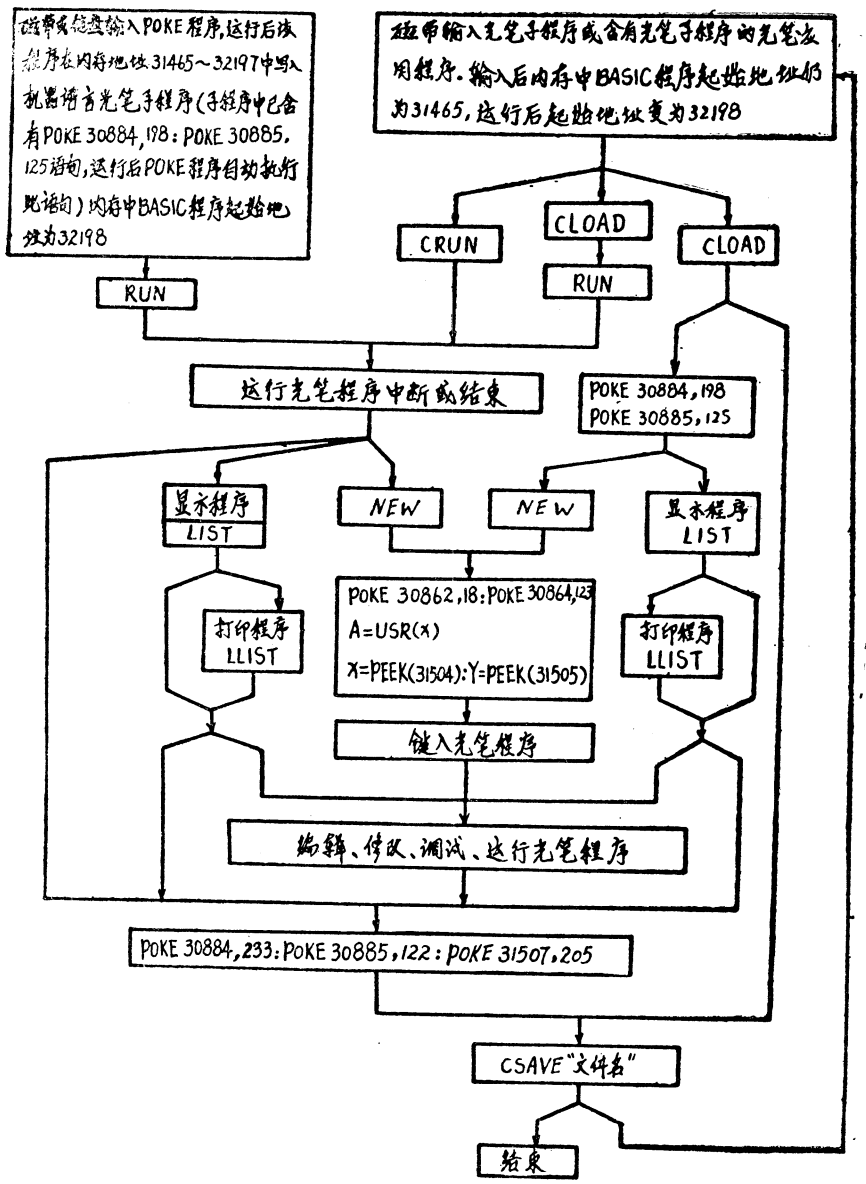
录制光笔程序可以CSAVE后面写上文件名（文件名是为编制的程序起名，例如CSAVE"LP TEST"说明这个程序是调试光笔程序，又如CSAVE"LP GRAPH"，说明这个程序是光笔画图程序，又如CSAVE "LP CHARACTER" 说明这个程序是光笔写字程序等等）

4. 光笔子程序的录制和光笔子程序的编写。

光笔子程序的录制可按光笔程序的录制方法进行，光笔子程序的编写（在没有示范带的情况下）可以用POKE语句写入内存，见第三册程序举例之34：光笔子程序的编写。

5. 光笔使用小结

光笔使用的步骤可用下列流程图表示：



第三节　游　戏　棒

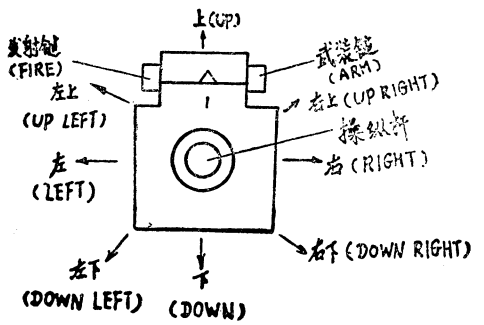
游戏棒是LASER-310机的外设设备之一，对于丰富计算机的功能有重要作用，目前游戏棒通常用于利用计算机做电子游戏，若能进一步应用到工业控制，则它将发挥更大的作用，游戏棒操作比键盘操作的优越性在于它具有直观性，操作容易，反应快速，可两人配合进行。

一、拆装游戏棒及注意事项

(1) 在关闭主机电源的情况下，将游戏棒介面 JI-20 插入主机背面的外围设备接口(PERIPHERAL)，注意介面的接口部分是否已完全插入，开启主机应显示如图1—6(第一册)所示的正常情况。

(2) 使用注意事项：在插入或拔除游戏棒介面时必须先关闭主机的电源，并且主机外围设备接口及游戏棒介面都不允许接触任何液体，否则会损坏主机和介面。

(3) 游戏棒的操作方向图示



游戏棒一套有 2 件，每件如图示各有 1 根操纵杆，1 个发射键和 1个武装键，操纵杆可以向图示 8 个方向板动，发射键和武装键可以按下，板动操纵杆或按键时，在电脑的输入/输出接口(接口地址为十进制的32～47)中输入特定的数码，可用PRINTINP(X) (X为32～47)显示。JS20 游戏棒与 LASER 310 机连接时，左游戏棒在接口地址43及38输入数据，右游戏棒在接口地址46及45输入数据。

二、游戏棒操作时输入/输出接口地址中数值的变化

方向杆 DIRECT	左杆 INP(43) 右杆 INP(46)	方向杆+发射键 DIRECT+FIRE	左杆 INP(43) 右杆 INP(46)	方向杆+发射键+武装键 DIRECT+FIRE+ARM	左杆 INP(39) 右杆 INP(45)
↑UP	62	DIRECT UP+FIRE	46	DIRECT UP+FIRE+ARM	46
↓DOWN	61	DIRECT DOWN+FIRE	45	DIRECT DOWN+FIRE+ARM	45
←LEFT	59	DIRECT LEFT+FIRE	43	DIRECT LEFT+FIRE+ARM	43
→RIGHT	55	DIRECT RIGHT+FIRE	39	DIRECT RIGHT+FIRE+ARM	39
↖UP LEFT	58	DIRECT UP LEFT+FIRE	42	DIRECT UP LEFT+FIRE+ARM	42
↙DOWN LEFT	57	DIRECT DOWN LEFT+FIRE	41	DIRECT DOWN LEFT+FIRE+ARM	41
↗UP RIGHT	54	DIRECT UP RIGHT+FIRE	38	DIRECT UP RIGHT+FIRE+ARM	38
↘DOWN RIGHT	53	DIRECT DOWN RIGHT+FIRE	37	DIRECT DOWN RIGHT+FIRE+ARM	37

发射键 FIRE	左杆INP(43) 右杆INP(46)	武装键 ARM	左资INP(39) 右杆INP(45)
FIRE	47	ARM	47

根据上表，可以编制游戏棒功能调试程序见第三册程序举例之三十五。

三、游戏棒的功能调试和游戏程序的编制。

游戏棒的功能调试可设计程序来进行，程序设计思路如下：在A、B、C、D四个变量中存入输入/输出接口地址中数值，当操纵游戏棒时，左游戏棒的动作使A，B变量发生变化，右游戏棒使C、D两个变量发生变化。每个游戏棒根据操作情况的不同共有26种变化，两个游戏棒共同使用时，就有52种信息输入电脑，然后把上述52种信息用字母显示出来，在调试游戏棒功能时，质量合格的游戏棒都应能显示上述52种信息。

利用功能调试程序可以把显示信息的语句改为PRINT@X，Y 语句，并使X的数值，Y 的内容作相应的变化，就可在文字模式中进行各种数字戏游、字母游戏，作图（低解折度）游戏或比赛，以开发人们的智力。或进入图系模式使用 SET (X, Y)、RESET (X、Y)，POINT(X、Y) 语句，并使 X、Y 的数值作相应的改变，就可以在屏幕上做各种电子游戏了。

附录

LAER-310 BASIC程序用字符及保留词机器码表

机器码	字符或保留词	机器码	字符或保留词	机器码	字符或保留词	机器码	字符或保留词
0	语句结束标志	45	-(字符)	63	? (字符)	81	Q
2	整数标志	46	.	64	@	82	R
3	字符串标志	47	/ (字符)	65	A	83	S
4	单精度实数标志	48	ϕ	66	B	84	T
8	双精度实数标志	49	1	67	C	85	U
32	(空格)	50	2	68	D	86	V
33	!	51	3	69	E	87	W
34	〃	52	4	70	F	88	X
35	#	53	5	71	G	89	Y
36	$	54	6	72	H	90	Z
37	%	55	7	73	I	91	]
38	&	56	8	74	J	92	/
39	, (字符)	57	9	75	K	93	]
40	(	58	:	76	L	94	↑(字符)
41	)	59	;	77	M	128	END
42	* (字符)	60	<(字符)	78	N	129	FOR
43	+(字符)	61	=(字符)	79	O	130	RESET
44	,	62	>(字符)	80	P	131	SET

132	CLS	151	COLOR	191	USING	216	INT
135	NEXT	152	VERIFY	193	USR	217	ABS
136	DATA	156	CRUN	198	POINT	219	INP
137	INPUT	157	MODE	201	INKEY$	221	SQR
138	DIM	158	SOUND	202	THEN	222	RND
139	READ	160	OUT	203	NOT	223	LOG
140	LET	175	LPRINT	204	STEP	224	EXP
141	GOTO	177	POKE	205	+(加)	225	COS
142	RUN	178	PRINT	206	−(减)	226	SIN
143	IF	179	CONT	207	*(乘)	227	TAN
144	RESTORE	180	LIST	208	/(除)	228	ATN
145	GOSUB	181	LLIST	209	↑(幂)	229	PEEK
146	RETURN	184	CLEAR	210	AND	243	LEN
147	REM	185	CLOAD	211	OR	244	STR$
58 147 251)	'	186	CSAVE	212	>(大于)	245	VAL
148	STOP	187	NEW	213	=(等于)	246	ASC
149	ELSE	188	TAB	214	<(小于)	247	CHR$
150	COPY	189	TO	215	SGN	248	LEFT$
						249	RIGHT$
						250	MID$

注1：表中带括号注明字符只表示该字符的形状，没有该字符所代表的意义，例如机器码43，只表示字符"+"的形状，没有+号的意义，而机器码205，才具有+号的意义。

注2：表中的机器码，字符及保留词仅适用于用户 RAM 区中 BASIC 程序区，变量区，堆栈区，并能用PEEK和POKE 进行操作，详见第五章。