日本 SHARP PC-1500

袖珍电子计算机
使用手册

陈奔勤 译校

广西计算中心
一九八三年三月印

日　　本

SHARP　PC—1500

袖珍电子计算机

使用手册

陈寿勤　译校

广西计算中心

一九八三年三月印

译 者 的 话

日本SHARP PC—1500袖珍计算机以其高级的BASIC程序语言、可观的存贮容量和良好的带动外部设备的能力而受到国内外各行业科技工作者和管理人员的欢迎，迅速在各地推广普及。为了帮助各方面的人员学习BASIC程序设计方法，充分发挥PC—1500机的效能，特将《PC—1500使用手册》全文译出，并译编入了印刷机、存贮模块使用说明及《PC—1500应用手册(程序库)》目录。翻译时对于原文中多处明显的错误，已在译文中作了改正，并且不作注明；为了节省篇幅，略去了程序输入键串。

尽管潜心校对推敲了译文，但限于译者水平，译文中错误或不确之处难免，诚望读者指正。

译 者

一九八三年三月 于南宁

目　录

绪　　言

请允许我们感谢你购买了 SHARP PC—1500袖珍计算机，我们深信，你将会乐于使用这种小型的但功能很强的机器，使成为你日常生活中的新伙伴。PC—1500是世界上最高级的手持式计算机之一。虽然它与其姐妹机 SHARP PC—1211袖珍计算机有许多共同特点，但PC—1500还具有如下高级的性能：

1．一个7×156个点的可编程序点阵式液晶显示屏；

2．一个能在程序控制下产生特殊作用的音频发生器；

3．具有大写和小写格式的ASCII字符集；

4．科学及数学函数；

5．用户定义函数键；

6．采用扩展BASC语言文本，它提供二维数组、可变长度字符串、绘图命令、程序链接，以及其它许多高级的性能；

7．相当于4K字节的随机存取存贮器（RAM）（CE—150）；

（译注：已有8K字节的备选随机存取存贮器（RAM）存贮模块CE—155）

8．备选用的印刷机／盒式磁带录音机接口（CE—150）。它能用四种颜色作X—Y绘图，存贮程序和数据，以及用九种不同大小的字符之一种打印程序和数据。

这种机器所具有的许多功能仅仅在几年前还要工程师们用许多电子管、导线才能实现。如此先进的技术不需要获得工程技术证书才能使用，相反，PC—1500的设计以及这本手册的编写，都使得你能迅速掌握这种新技术。

我们分五个主要部分编写这本手册，使初学习使用的人能迅速获得使用机器的能力。要求更高的使用者可以从“高级程序设计”、“高级计算部分及附录探索PC—1500的功能特点。

这本手册的文体是口语化的，并提供了许多例题。但不要单从我们的言词去理解，要看看如何容易起步，请翻到第0章。不过，首先要确保电池已经装入，假如电池未装入，请参考附录B提供的指示。

首要的是，要有兴趣，毫不踌躇地去尝试！

使 用 须 知

因为计算机的液晶显示屏是用玻璃制成的，须小心轻放。

为了保证SHARP袖珍计算机不出故障，我们建议：

（1）不要将计算机存放在温度变化极大、潮湿或粉尘的场所；在热天，运载工具受阳光直射会形成高温，长期暴露在高温中有损坏计算机的危险。

（2）只能用柔软的干布来清洁计算机，不可使用溶剂、水或湿布。

（3）计算机在长时间不用时要移出电池，以防漏出电解液。

（4）如果需要维修，只能将机器送到被委托的SHARP维修服务中心。

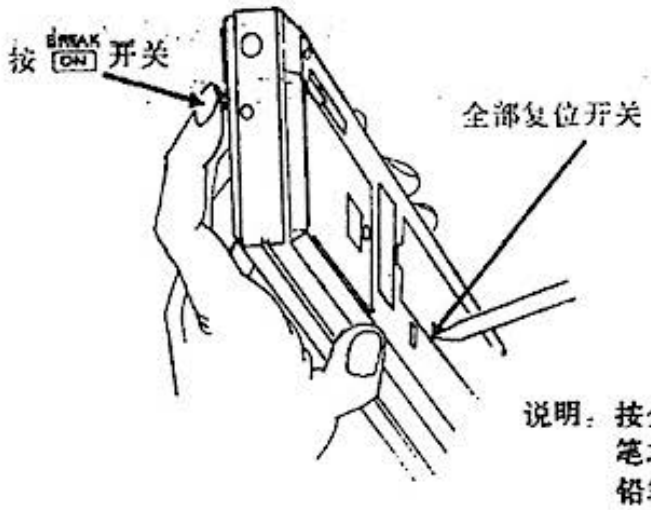
（5）保存本手册以备日后参考。

异常情况的处置

机器如果在使用中遭受强烈的外部噪声或撞击时，会引起其所有各键，包括 BREAK ON 键失灵。如果发生这种情况，可在按下 BREAK ON 键的同时，按压机器背后的 ALL RESET（全部复位）开关大约15秒钟，然后查对显示屏中是否指示"NEW 0 ? : CHECK"，接着按键 CL N E W 0 ENTER。如果显示不是 NEW 0 : CHECK，则再一次进行上述操作。

经过这些操作之后，机内的程序、数据等全部保留的内容便被清除，故除非发生了上述异常情况，不要按全部复位开关（ALL RESET）。

操作图示：（见下页）



说明：按全部复位开关时，使用诸如圆珠笔之类的点状物体，不要 用自动铅笔或针尖一类易断的点状物。

图0 全部复位开关的操作

PC—1500规格

型号： PC—1500袖珍计算机

计算数字位数：10位数字尾数＋2位数字指数

计算运算系统：按数学公式（具有优先级判别功能）

程序语言： BASIC

存贮容量： CPU： CMOS 8位

系统ROM：16K字节

存贮器容量RAM：3.5K字节

系统区： 0.9K字节

输入缓冲区： 80字节

存贮栈 196字节

其它：

用户区： 2.6K字节

固定存贮区：624字节

（A～Z，A$～Z$）

基本程序数据区：1850字节

保留区： 188字节

计　算： 四则算术运算，乘幂运算，三角和反三角函数，对数和指数函数，角度转换，开平方，符号函数，取绝对值，取整数和逻辑运算。

编辑功能： 光标位移（▶◀）

插入（INS）

删除（DEL）

行上移和下移（↑，↓）

存贮保护： CMOS后备电池

（保护程序、数据及保留存贮区的内容）

显　示： 液晶显示屏

26个字符宽度

7×156点阵式图象

键　钮： 65个键，包括：

字母键

数字键

用户定义函数键

预编程序键

电　源： 6.0伏直流：

4节干电池（UM－3，AA或R6型）

电源消耗： 6.0伏，直流：0.13瓦

电池使用寿命：约50小时（UM－3，AA或R6型干电池）

工作温度： 0℃～40℃（32°F～104°F）

尺寸： 195（长）×86（宽）×25.5（高）毫米

$7\frac{11}{16}''$（长）×$3\frac{3}{8}''$（宽）×1″（高）

重　量： 约375克（0.83磅）（包括电池）

附　件： 软盒、四节干电池，二块键盘模板，用户姓名卡及使用手册

备选件： 印刷机／盒式磁带录音机接口（CE－150），扩充存贮模块（插入式，4K字节RAM CE－151，8K字节RAM CE－155）

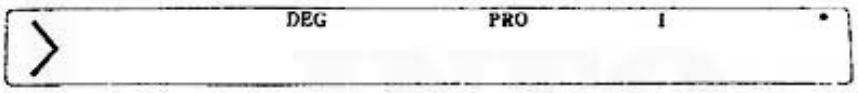
第0章　即时进行的程序设计

本章是专为那些其好奇心胜于忍耐力（也许还强于其常识）的人（包括作者自己）而准备的，对于你们这些决意要用这个现代电子学的奇迹做些事情的人，我们在此介绍一个简单的程序设计练习（注意，对于胆小怕事的人，还是从第一章开始介绍，以获得初步的入门知识，再对PC－1500作更深入的从容不迫的介绍）。

在开始练习之前，还有一点要注意的是次序，在练习时，遵循所有列出的给定步骤是很重要的。与通常的看法不同，计算机不是"超级大脑"，它不具备一般人的能"算出"你的要求的能力，PC－1500只能等待你的指令并执行指令。准备好了吗？好，让我们开始吧！

例1·

首先，找到键盘右上角标记"ON"的键，按下这个键就会使这个沉睡的电子怪物醒来（别指望它会喷出一团团烟雾），计算机的显示部分将出现和下面的插图类似的显示：

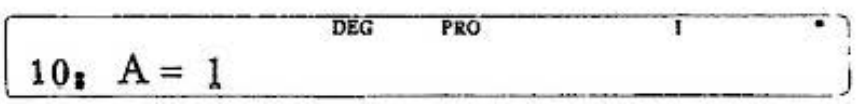


按下 [MODE] 键（在最右侧）直到显示屏的上部出现PRO（假如你按该键太多次，就再按一次这个键，直至得到所要求的结果），SHARP PC—1500就准备接收组成计算机程序的一系列指令。

输入下列键串：

[1] [0] [A] [=] [1] [ENTER]

注意，当按下 [ENTER] 键时，计算机会变更你敲入的内容，此时，显示将成为：



（译注：原手册中，用Ø表示数字零，以区别字母O，但为了便于排版，译本中不再这样做，字母数字键一般也不再加方框）。

继续按下列的键串，当按下后继符而每一行消灭时，不要惊慌。

```
20  PAUSE  A  [ENTER]
30  A = A + 2  [ENTER]
40  GOTO  20  [ENTER]
```

至此，你的第一个程序已经完成了，现在，你必须告诉计算机去“执行”它所存贮的指令。这一步称为“运行”程序，这是在RUN方式中进行的。再按一次 [MODE] 键，显示屏顶部的字母PRO就会被RUN所取代。

最后一步，敲入字母键 RUN 并按 [ENTER] 。

祝贺你！你的第一个 BASIC 程序现在正在运行。计算机正在遵循你的指令并在繁忙地依次列出全部正奇数。

“好，”你会对自己说，“我是个天才了。但是何时它才停下来呢？”

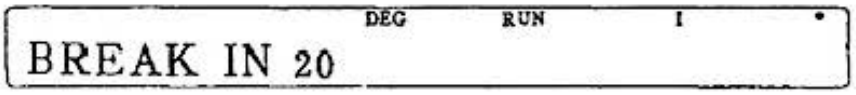
可是，不幸的是，没有你的干预或没待电池耗尽，这个独特的程序是决不会结束的。要看看到底是为什么，让我们重温一下程序：

```
10  A= 1
20  PAUSE  A
30  A=A+ 2
40  GOTO  20
```

编号为40的行的作用是使计算机重新执行编号为20开始的全部的行，当然也包括40行，它告诉计算机再一次执行20行，30行，40行……等等，没完没了。这种无休止的重复在计算机术语中称为“循环”。

我们的程序已经陷入“无限循环”之中了，只有你才有能力停止这种电池电源的无谓消耗。要做到这一点，可按 [ON] 键。由于 PC—1500 已经处于接通的状态，故实际上 你 是选择了BREAK（中断）功能。不过不用着急，不管名称怎么说，它不是一个自我损坏键。根据所获得的记录资料，仅仅通过按键，不管如何是不会损伤或损坏计算机的，那末，就大胆地去实验吧！

假如你已经按了BREAK键，就会在显示窗中出现类似于下图的显示：



它告诉你，当你中断计算机的运行时，刚执过或正在执行的是哪一条指令。再按一次BREAK键，计算机便等待你的下一条指令：

对于那些猛然记起家里的水龙头还在开着的人，这是一个好的停止点（离开之前，请按 OFF 键以节省电池）。你们中其他的人已经成了程序设计迷，希望用我们的第二个实例继续训练（假如这会误了你进中餐，我们就此暂停进行）。

例 2 •

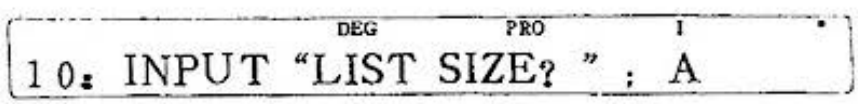
为了开始我们的第二个程序练习，需要按MODE键直到显示屏顶部的字母RUN被字母PRO（Program的缩写）所替代，使计算机进入程序方式。现在，PC—1500就可让我们输入一个新的程序或修改旧的程序。由于我们的新程序不是建立在旧程序的指令上，我们必须从计算机的存贮器中清除那些旧的指令。为此，敲入NEW并按ENTER键，稍停一下，就会出现 > 符号（称为提示符）。

敲下列的键串输入程序的第一行：

10 INPUT SHIFT "LIST SPACE SIZE SHIFT

/ SHIFT " SHIFT + A ENTER

注意，按 SHIFT 键后接着按某个上面标记有另一个字符的键，就会输入该键上方的字符。SHIFT 键使得两个字符能共用同一个键钮，有时就管它叫“第二功能键”。这样，在程序的第一行(上述的10行)，SHIFT 键后跟着一个 + 键就送入了 ; 符(分号)，存在计算机内的整行为：



这里要说明的是，在这本手册中，第二功能字符由 SHIFT 键和所需要的字符来选

用，例如，上述标记为10的行将写为：

10 INPUT [SHIFT] " LIST [SPACE] SIZE [SHIFT] ?
[SHIFT] " [SHIFT] ; A [ENTER]

现我们用下列的键串完成第二个程序的输入：

20 IF A [SHIFT] <= 0 GOTO 99 [ENTER]

30 FOR I= 1 TO A [ENTER]

40 PAUSE I [SHIFT] [,] I * I [ENTER]

50 NEXT I [ENTER]

99 BEEP 2 [SHIFT] [:] END [ENTER]

现在，我们的第二个程序便存贮在PC—1500的存贮器内了。你记得应该接着做什么吗？假如你说是"运行程序"，你就已经具备了编辑程序的能力了。

回到RUN方式（提示：用MODE键），敲入RUN，按 [ENTER] 键开始执行（或说"运行"）第二个程序。

计算机按下面的方式提问你了吗？(如果不是，回到程序方式重新校核你敲入的内容。)

```
                    DEG      RUN      I      *
LIST SIZE?—
```

好！我们的程序正在询问使用者有关指示计算机执行任务所需要的信息。读出我们的BASIC程序的第一行：

```
                    DEG      PRO      I      *
10: INPUT "LIST SIZE? " ; A
```

这一行的指令现在正由计算机执行，执行的结果是，计算机等待你"输入"（敲入）某些信息。

这个程序会输出一个若干个数及其平方（数自乘）的表，但是，它首先询问使用者要输出多少个数及其平方（表的体积？），使用者通过敲入一个数并按 [ENTER] 来回答。

例如，敲入8并按 ENTER .

密切注视屏幕上短暂地出现的数对．数对的第二个数（右边的数）是第一个数的平方。屏幕上将显示8对数，因为这个数目是你在回答程序的提问时所要求的．

当提示符恢复显示时，重新运行程序(敲入RUN，按 ENTER)，并对 "LIST SIZE?" 提问给出不同的回答。用不同的表体积运行这个程序进行几次实验，你会体验到可编程序计算机的一个重要优点是，它们能够重复执行繁琐冗长的任务，而每次只稍为改变一下对输入内容的回答。

重新运行一次程序，不过这一次是给表的体积输入一个零。这一回会发生什么情况？这一回，没有产生一个表就停止了程序的运行，虽然这可看作偶然的情况，但FC—1500是简单地遵循我们的指令的。

这就说明，为什么计算机是这样一个强有力的工具，能编制程序使它遵循各批指令去处理输送给它的各种各样的信息。因为在20行指令中，假如使用者输入的是一个零（或小于零），计算机就使程序跳过数表的计算而转到程序的末尾。实际上，计算机根据使用者的要求作了一次判断。作为一个程序员，你可以控制计算机在什么时候、进行什么可能的判断，这样，就可以利用计算机的全部能力按你认为最好的办法为你解决专门的问题。

第一章　机器的初步认识

当你打开SHARP袖珍计算机，赞尝这个漂亮的新伙伴之时，一定会感到不知从何入手。让我们考察一下SHARP计算机。

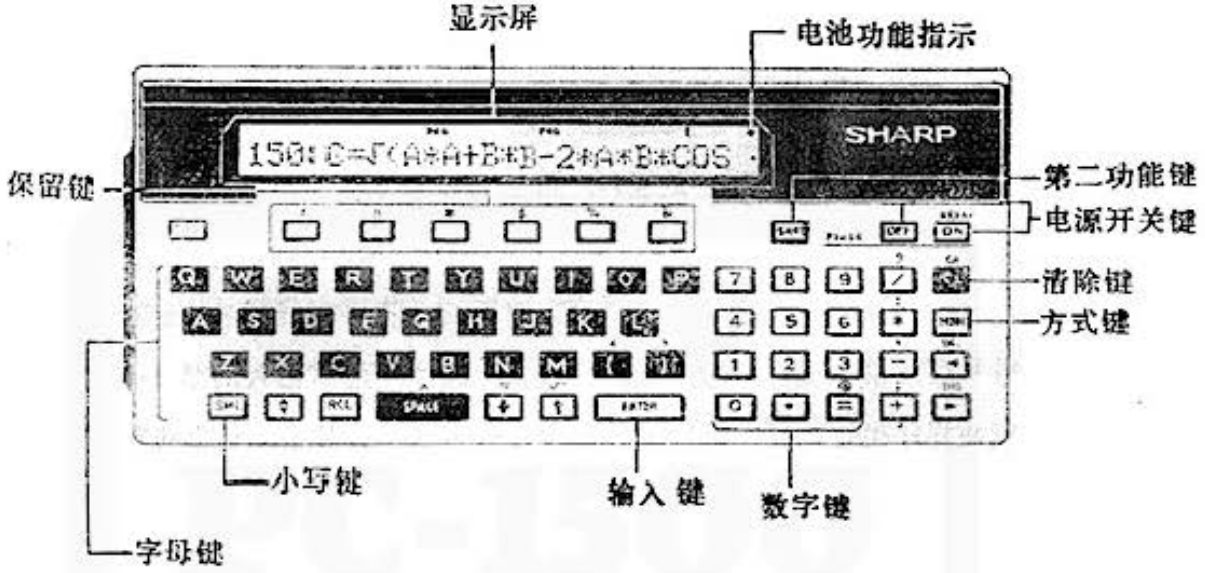


图1（见原版英文书图注）。

我们过片刻就要叙述显示部分。首先，即便是在打开机器的开关之前，也可注意到键盘的几个重要特点：

A• ON键和OFF键（开关键）

显而易见，这些键是接通和关断电源的。为了节省电池消耗，SHARP机能在大约7分钟内没有任何按键动作时自动关机，除非正在执行程序。注意到印在 ON 键上方的BREAK这个词，ON 键能够中断程序的执行，这个功能后面有更详细的描述。

B• 字母键

字母键使计算机使用者能给出指令和送入数据。另外，这些键可用于指定计算机的存贮器内你要存入数据或检索出数据的"存贮区"。这个用途将在变量一节中介绍。通过使用 SHIFT 键和 SML（SMALL）键，还可使用小写字母（见下述）。

C. 数字键及算术运算键

这些键可送入计算所需的数，[+]、[-]、[*]及[/]键告诉计算机分别进行加、减、乘、除运算。[E]键可用于送入“科学记数法”表示的数。关于这种记数法的使用和其它高级功能将在“高级计算”一章中叙述。

D．SHIFT 键（第二功能键）

这个键引出标记在许多非字母键上方的第二功能，例如，要敲入冒号，按[SHIFT]键，然后按[*]（星号）键。当[SHIFT]键后面跟字母键时，显示小写字母（注：在 SMALL（小写）方式，在字母键前按[SHIFT]键将产生大写字母）。

当 SHIFT 键生效时，显示屏的左上角将出现英文字“SHIFT”，第二功能方式一次仅对一个键起作用。

键盘顶上、显示窗正上方的 6 个键称为保留键，按我们在后面介绍的方法使用第二功能键，你可频繁地给这些键赋予敲入的命令或其它操作。

注：假如你错按了[SHIFT]键，可再按一次撤消它。

E．小写字母及小写键[SML]

[SML]键使你能给全部字母键规定小写方式。假如你没有规定小写方式，机器就对于你每一次按的字母键选择大写方式。（我们称这为“蕴含”方式，意为除非你另外指定，机器就以这种方式工作）。通过在字母键前面按[SHIFT]键，可敲入单一个小写字母。

[SML]键可用来使小写方式生效。在小写方式，通过按字母键可产生小写字母，而在字母键前面按[SHIFT]键又可产生单个的大写字母。当计算机处于小写方式时，在显示窗的顶部会出现英文字“SMALL”。一旦你使计算机置于小写方式，它就保持在这种方式，直至你再一次按[SML]键。

注：我们建议你暂时限制使用小写字母，这是由于机器只承认大写字母形式的指令，当你学习编制程序时，你会发现小写字母是较为方便的。

F．显 示 屏

计算机的“玻璃窗”部分称为“显示屏”，按下 ON 键，它看起来如下所示：

在显示屏中，你可看到一个“>”符号（称为“提示符”），几个英文字或缩写词以及一个圆点（表示电池有效）．如果出现在显示屏中的具体的缩写词不是我们说明的这些，不必顾虑，当你操作机器的过程中，显示的符号是会改变的。

G． 光标和提示符

在显示的最左端，可以看到一个提示符号（>），它提示你可与机器对话．当“提示符”出现的时候，意味着机器现时没有任务，等待着你的吩咐．选一个字母敲入，它就会取代显示左端的“>”号，而在字母的右边出现一个“_”（下横杠符号），它就是光标．每当你按下一个键，光标就随着显示缓慢地移动，指示你要敲入的下一个字符的位置。你不妨试试敲入自己的名字，注视光标的移动。

若敲入的字符多于２５个，即超过显示屏一次所能显示的限度，整一行便左移．（试一试看）．被“推出”显示屏的字符不是丢失了，它们仍保留在机内，作为输入的行的一部分．任一单行的字符最多可达到８０个．在下面的章节中，我们将会看到如何“读出”以及如何改变这些字符。

H．清 除 键

（右上角那个红色的 CL 键）

按下这个键，便清除了显示的内容，用 CL 键清除刚敲入的字符。注意到提示符重回到显示之中，表示计算机又在等待你的命令。

清除键也用于撤消不正确的命令（见下面“出错信息”一段）。

I．ENTER （输入）键

当你给计算机敲入信息时，字母或数字就会出现在显示中．然而，在你发完成输入信号之前，机器是什么也不做的（毕竟机器不能理解你的意思）．这个信号就是通过在其它键串

后面按 ENTER 键来实现。此时，计算机就检查你输入的字符的形式是否正确。某些错误，不是说所有的错误，会使计算机拒绝接受你的输入内容。

记住：每一次你希望将一条指令或一个数据项输入机器时，要按 ENTER 键。。

J．出错信息

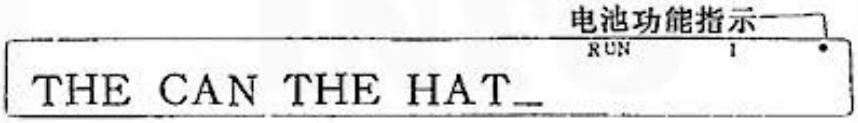
按下列的键：

1 + 1 =

现按ENTER键，就会显示出答案3，对吗？不对？你得到“ERROR 1”的答案吗？是计算机有毛病啦？绝对不是！是在命令的形式上有错误。“ERROR 1”是一个错误代码，他告诉你，已经不正确地进行了一个计算。（由于不易理解，附录中附有一个完全的出错信息表）。我们会对这第一个错误负责的，在后面的章节中，我们将告诉你如何用其它的键去改正一个错误的命令，不过现在，你可用清除键清除该出错信息。

K．电池功能指示

当指示电池功能的圆点消失时，表示要更换机器的电池了，见附录 B的说明，



L．如何使用RCL（读出）键

在保留方式，RCL键用于读出或检索前面存贮的语句或短语。

要完成对保留方式的输入内容的检索，按下面几步进行：

首先，通过按保留选择键（▲▼）选择保留内容存放的组号：Ⅰ组、Ⅱ组或Ⅲ组，（保留方式选择键就在RCL键的左边）。

然后按RCL（读出键）和先前指定的保留键。假如你忘记了前面赋给保留键的内容，只要遵照第七章 B节的指示去做就行了。

测验

从B栏中找出一个与A栏的每一条款配对的条款（答案在末尾找）。

提示：B栏中包含有错误的条款。

A	B
a）SHIFT键	1）由于不满引起经常的咒骂
b）显示屏	2）一种清除显示窗以及清除先前任何计算结果的键
c）光标	3）选择共用同一键钮的2个字符之一的一种键
d）ENTER键	4）没有这种键
e）BREAK键	5）出现在显示屏中，告诉使用者计算机等待他的命令的一种字符
f）清除键	6）一种通知计算机，使用者已完成敲入的键
g）提示符	7）计算过程中，中断程序执行的键
h）红色的HERRING键	8）一种稀有热带鱼的名称
	9）显示屏中，指出下一个敲入字符将出现的位置的一种符号
	10）显示信息的玻璃窗

（答案：A—3，B—10，C—9，D—6，E—7，F—2，G—5，H—4）

第二章　毅然尝试

实际上，本章很可能是起错名了，学习PC—1500并非冒险闯入水潭。然而，它要求你不要害怕接触计算机。正如我们前面说的，仅仅通过接键，是不会伤害或损坏计算机的。

在这一章里，我们将披露出PC—1500的基本性能特点，程序和更高级的计算就是建立在这些基本特点的基础之上的。拿出必要的时间将每一节中的例题做一遍，对基本知识的良好了解将使你能发掘这台机器的全部潜力。

虽然我们不提倡，但如果你感到自己已经充分明白了，可跳前到本章末尾的小结部分。

A．MODE（方式）键

让我们从一个到现在为止被忽略了的键开始叙述。在键盘右侧，可以找到一个标记MODE的很重要的键，重复按这个键，可以发现每次按它都要改变显示屏右上方的缩写词。用这个方法改变机器的工作方式，可将它比作汽车换档。每当新的方式或排档接上时，机器表面上看来没什么变化，却能执行各不相同的任务。象汽车一样，SHARP计算机也必须置于适当的工作方式以按计划行使它的功能。而且，也象汽车一样，当你企图在不适当的“排档”上工作时，计算机就会迅速地通知你有了错误。

重复按MODE键，就可进入机器的三种工作方式中最重要的两种方式：RUN方式和PRO方式（运行方式和程序方式），第三种方式，即RESERVE（保留）方式，通过在MODE键前按SHIFT键来安排。手册的后面各章中，将叙述这些方式的每一种方式是如何使机器顺利地进行工作的。但现在要记住的是，要利用机器作为一台计算器，必须使它处于RUN（运行）方式；以后，在PRO（程序）方式，你可写入或修改程序；使用RESERVE（保留）方式，你可给某一个键赋予频繁使用的指令，关于这方面的内容，将在第七章中详细解释。

注意，将电池装入计算机后第一次开机时，机器是置于PRO方式，其它时候，计算机将处于关机前最后使用的方式上。

B．简单计算

使机器置于RUN方式，让我们试进行几个基本数学运算。在进行每一次计算之前，需要按 CL 键，清除显示中可能干扰新的计算的先前的数据。求下面几个简单问题的答案：

输入	显示
5 + 2 ENTER	7
5 − 2 ENTER	3
5 / 2 ENTER	2.5
5 * 2 ENTER	10

注：不要按等号键。记得第一章中提到 ENTER 键，它通知机器，你已完成输入，希望机器执行命令或计算。

C．连续计算

你可在后来的计算中利用一个计算结果直接继续进行第二个计算（在两个计算之间不要按 CL 键），这样，假如你是在结帐，就可采用下面的方式操作机器：

输入	显示
161.16－47.50 ENTER	113.66
－12.33	113.66－12.33
ENTER	101.33

你会看得到，当你开始第二个计算时，第一个计算的结果会跳到显示屏的左边。

注：在计算中，送入数时不可使用美元号（$）或逗号（，），这些符号在BASIC语言中（因而在SHARP计算机中）有其特定的意义。

其它的运算可以类似地进行，例如：

输入	显示
5+3 ENTER	8
8+3－1 ENTER	10
10*3－1 ENTER	29
29/3－1 ENTER	8.666666667

D．带负数的计算

设想你送给了你的计算机科学教师翁奥夫先生两个苹果，还有5个苹果的存货．你也许纳闷："假如我不那么慷慨送给翁奥夫先生的话，现在该有多少个苹果？"为了求出答案，你可以设想从库存数反减过去，即减去减数．输入机器：5－－2 [ENTER]，你假想的库存是7．现用带有负号的数试作下面类似的计算：

5＊－2

5＋－2

5／－2

－5－2.3

－5＋－2

－5／－2

记住：各个计算之间要按 [CL] 键清除前面的结果。

E．混合计算

在要求计算机计算出答案之前，你可将一系列的计算串在一起．例如，你及两个朋友：霍布和诺布，希望一个星期中每天2次分享5个苹果，那末，你要买多少个苹果才能维持一个星期？计算式是：5个苹果 [／] 3个朋友 [＊] 每天2次 [＊] 7天：

键串	显示
5／3＊2＊7 [ENTER]	23.33333333

（买24个苹果和一只鹦鹉，你可将多余的2/3只苹果喂它）．作下面的计算（不过这一次要你自己根据算式编故事了）：

输　　入	显　　示
5＊2－3.675 [ENTER]	6．325
5/3＊6.2＋7－47 [ENTER]	－29．66666667

F．括号的使用

当我们研究混合运算时，随之而来的问题是关于优先权问题．例如，表达式 5－3/4

可以读成二种情形：（5减去3）除以4，此时答案为0.5；或者为5减去（3除以4），此时答案为4.25。

第一功能的键钮里面就有一对括号，你可用它来澄清那种模棱两可的含义。进行下面的计算：

输　　入	显　　示
5－3/4 ENTER	4.25
5－(3/4) ENTER	4.25
(5－3)/4 ENTER	0.5

SHARP计算机预先设置了使某些计算先于其它计算进行的功能（具有"隐含"优先权的能力）。关于机器计算中优先级别的完整清单，见附录O。除非用括号支配不同的顺序，除和乘运算一定是在加和减运算之前执行。这样，机器就将上面的第一个算式理解为与第二个算式是一样的，但不同于第三个算式。为了保证机器给你的答案是你所需要的，你可使用括号来指定机器要执行的适当计算顺序。

如在下面的问题一样，机器能够理解几层的括号：

输　　入	显　　示
((6－4)/2)•(((3－1)/4)•6)　ENTER	3

最内部一对括号里的算式总是首先计算。

记住：当对算术运算的顺序拿不准时，使用括号来澄清。

G．逻辑比较及不等式

SHARP 计算机具有比较两个值或表达式并指示比较结果的能力。这个能力是设计具有判断功能的程序的基础。作为不等式，对于学习"新数学"的学生，是可以认识这种比较的方法的。

不等式可看作是一个比较，其结果不是真就是假。例如，语句"6除以3等于2"是一个比较，它的结果为真，反过来，语句"6除以3大于5"是一个结果为假的比较式。

计算机和数学家们对于比较式的各种可能类型使用下列的符号：

符　　号	意　　义
<	小于
>	大于
=	等于
<=	小于或等于
>=	大于或等于
<>	不等于

这样，我们可用符号重新描述上面的不等式，分别为：6／3＝2以及6／3＞5。

给出一个不等式，机器就会判断该不等式是真还是假。SHARP 计算机的设计为了与流行的计算机的设计保持一致，也是用1表示"真"，用0表示"假"。例如，你敲入：

6／3＝2

机器就会回答一个1（表示真）。敲入键串：

6／3＞5

就会得出0（表示假）。

试试用下列的操作检验一下机器的判断力（务需在RUN方式）：

键　　串	结　果
4 SHIFT > 9 ENTER	0
5 SHIFT > 4 ENTER	1
2 * 2 SHIFT < 2 * 3 ENTER	1
1 8 SHIFT < SHIFT > 1 8 ENTER	0
2 = 2 ENTER	1
5 SHIFT < = 5 ENTER	1
2 5 * 2 SHIFT > = 1 0 0 / 3 + 4 ENTER	1

也许你已经看出，用以比较的式子可能不得不是非常复杂的（唯一的限制是一行80个字符的限度）。

这里有一个说明不等式的实际应用的简单例题：

诺布到奈尔斯恩原料商店去购买粘合剂，他发现商店袋装粘合剂有4磅、8磅、12磅装

的几种，他带来的钱可以买2袋12磅装的或买3袋4磅装和一袋8磅装的，他不明白到底哪一种买法可以得到数量较多的粘合剂，就问机器，是否：

2*12>(3*4)+8

机器回答1，于是诺布买了2袋12磅装的。

试用不等式的方法解答你自己的问题。

H．编辑键及其功能

多数人（除非天才）都有产生错误的可能性，考虑到人们在输入程序或数据中难免出错，PC—1500 的设计者们编造了几种便于修改的功能。

H．1 左箭头键／删除键 DEL ◀

至此，你们中的一些人可能已经发现了键盘右下方的左箭头键。这个键所起的作用类似于多数现代的打字机上的后移键，它使能从前面打入的字符后移。

在RUN方式，自清除显示开始（即，使显示出现提示符），敲入下列字符：

THE SPACE CAN SPACE THE SPACE HAT

将出现如下显示：

```
                    DEG      RUN      I
THR CAN THE HAT_
```

按下左箭头键并注意光标变化的方式。构成光标的这块“闪烁方格”使你能观察目前光标位置上的字符。重复按左箭头键（或压住），直至光标移到字符“N”上，（如你不小心使光标超过了“N”，再用右箭头键使光标前移）。

敲入一个T，则字符“T”取代“N”而光标前移。假如你记得光标指示的总是要置入的下一个字符的位置，就不会对此感到意外。

接着敲入这些字符：

SPACE MITHIN SPACE HAT

现在就会显示：

```
                    DEG      RUN      I
THE CAT WITHIN HAT_
```

如显示所示，前面打入的字符已经永远消失了。

除了后移功能之外，左箭头键还有第二个用途，这个键的上面写着缩写词 DEL，要删除一个字符，将光标置于要删除字符上并按 [SHIFT] [DEL] 。

让我们来试做一遍，将光标后移到W并按键串 [SHIFT] [DEL] 四次，显示就成为：

```
                DEG          RUN          I
THE CAT IN HAT
```

H. 2。右箭头键／插入键 [INS ▶]

正如我们已经看到的，右箭头键使光标移向前而不删除字符。和左箭头键一样，若该键压住，光标便接连移动。

将光标移到本行末尾。

右箭头键的第二功能使我们能在写入的行中插入字符，这个性能对于我们中那些往往忘记某些事情（如字母和单词）的人是方便的。

将光标移到 HAT 中的H处，连按键串 [SHIFT] [INS] 四次并注视目前移到右边的字符.这个新出现的匣形字符可以认为一个"空箱子"，这些"空箱子"可以填入新的信息。

打入：

T H E [SPACE]

转眼间，你已经插入了一个字。

H. 3。读出功能

到现在为止，我们已经讨论了改正尚未送入机器内存的语句的方法，（即是说，你按了键串之后，还未按 [ENTER] 键。）然而，一旦你按了 ENTER 键，机器就企图立即完成计算。如果机器完成了计算，显示中，计算结果就代替了算式。但算式并不从机器中消失，可以通过按左箭头键或右箭头键读出（重新显示）。

清除显示并敲入你所选择的算式，按 ENTER 键。计算出结果，读出原算式。注意，要重新得到计算结果，需重新计算算式（用 [ENTER] 键）。

幸而，如果机器在试图求解你输入的算式的值遇到错误时，也会有读出功能（懂得输入内容的意义）。它允许你用刚学过的任一种编辑功能来读出和改正错误的算式。

为了试验这个功能，输入下面书写不正确的表达式：

45＊63／＊2 [ENTER]

当出现错误信息（ERROR 1）时，按下任一个箭头键读出表达式。你可看到在第二个乘号（星号）上盖着一个闪烁的方格，这是机器指出有问题的点的方法，由此，就可以着手进行你认为适当的某种改正。

Ⅰ．变量

通过变量抽象地解题的能力是机器最强有力的功能之一。变量可想象为一组小盒子，每个盒子里可装入单个数据项，例如一个数或一个名字。

你可能记起中学的代数中变量的含义。你懂得（至少是学过），若5A＝30，则A必为6，但若5A＝35，则A等于7。在这种情形，A是一个变量，它容纳称为变量的值的单个数（不总是相同的，因而是一个变数）。在算式中用一个字母（例如A），代替某个特殊的数的能力是很有用的。

现在，让我们用下例看看为什么：

达弗上将有心要选购一套高级的高尔夫球及球棍。一种包装是含有每根5美元的球棍5根，一只袋21.99美元，以及每只1.56美元的球3只。当地的陆军消费合作社就出售这种货，售价九折，但另收8美元的装饰费和送货费。而迪斯康特伯尼斯的高尔夫商品提供同样的配套，倒扣5％，且自由交货。

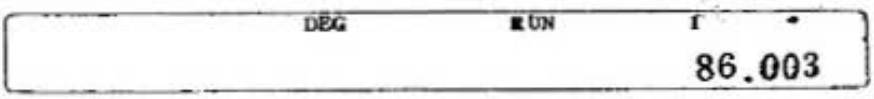
要计算哪一种交易较好，上将决定求助于SHARP PC—1500机，他计算出每套的基本价并将结果保存在变量G中：

G＝（5＊12）＋21.99＋（3＊1.56） ENTER

显示器显示出这个结果，它可以通过输入变量的名字从存贮器中读出。操作 G ENTER，显示：

现上将计算在陆军消费合作社的实际购价：

G－(G＊.10)＋8 ENTER



而在迪斯康特伯尼斯的购价是：

G－（G＊.05）[ENTER]

```
                    DEG          RUN          I      •
                                                82.3365
```

显然，迪斯康特伯尼斯的价格较便宜（聪明的读者会注意到上将已经能使用机器独特的变量读出功能（见第二章）来解决这个问题，尽管不同样容易）。

由上例，我们可以学到几点：

注意到上将的第一个计算具有如下的形式：

变量名＝表达式

具有这种形式的指令称为赋值语句。重要的是不可把赋值语句和等式相混淆。等式不像赋值语句那样能独立使用，但它可以组成其它程序设计指令的部分。

赋值语句命令计算机把表达式的计算结果连同给定的变量名存贮在存贮器单元中。此后，使用该变量的名字（在本例中是G）就如同使用该结果本身一样。同样要提及的是，在同一计算中，变量可按需要被多次使用。

变量也可用于其它简化技巧中。一个变量的值可以赋给另一个变量，如在赋值语句：

H＝G

中，会把上面的结果赋给变量H，而G的值不受改变，现在同一结果存于两个不同的变量。

存有数的变量可用一个语句增加或减少其值，如例子：

G＝G＋5

这个指令使机器读出变量G的值，加上5，将新的值存回到G。这个功能对于各种各样的计算都是有用的。

某种新机械产品的成本存于变量X中，假如税率为6.5%，问该产品的售价为多少？

X＝X＊1.065

当然，我们可以把这个例子的算式写成P＝X＊1.065，以保留X不变。假如我们已经将税率存于变量T，那末，我们还可以将算式写成：X＝X＊T或P＝X＊T。

直到现在，我们都用的单个字母作为变量名，这提供了２６个变量（A到Z）。实际上，对于单个的数，机器允许使用950个以上的变量名。另一组950个以上变量每个可用于存放

16个以内的字符。最后（好象还不够），可使用更高级的技术。使用者还能按需要建立更多的变量来存放数或字符，唯一的限制是计算机存贮器变量总数。

变量的命名法是简单易学的。数值变量（用于存放数值）的名字可选用下列规则。

——变量名可以是一个字母：A～Z；

——变量名可以是一个字母后跟一个数字（0～9）或一个字母。

那末，下列是数值变量的有效名字：

S、Q1，TX，MM，Z9，R0，E。

注：由于与在 BASIC 语言中有其它意义的缩写词相矛盾，这种机器不允许使用 LF，IF，LN，PI，TO这些符号作变量名。

除在名字的末尾附加一个$号（美元号）之外，字符变量（用于存放字符）的命名规则与上述相同。$号提醒机器，该量存放的是字符信息。下列是有效的字符变量名的例子。

T$，P2$，T7$，AA$，YR$，X$，ZH$，B5$

注：由于与BASIC语言的词相矛盾，机器不允许使用 LF$，IF$，LN$，PI$，TO$作为变量名。

要明白，变量A和变量A$是两个不同的变量，第一个变量只能存放数，而第二个变量只能存放字符。下面，我们会看到，SHARP 的BASIC 包括把字符转换为数和把数转换为字符的指令。

为了把字符存放在字符变量中，我们使用变种的赋值语句：

字符变量名＝"字符"

例如，打入：

D$＝"DAVY JONES"

现在，按键 D $ ENTER 读出D$的内容，显示为：

DAVY JONES

注意到，Y和J之间的空格被存了进去，而"号（双引号）字符不存入。双引号起"定义符"的作用，指示要存入哪些字符。除双引号本身外，任何字符（包括空格）都可以存入。括在双引号内的字符序列称为字符串，字符变量常称为字串变量。每个字符变量最多可容纳16个字符。为了不超过这个限制或丢失信息，须稍加小心。用下面的赋值语句来说明。

这个语句无意中把约翰的饮食习惯归一化了：

F$= "JOHN EATS BUTTER FLIES"

现读出信息，按 [F] [$] [ENTER]，可见发生了某些变化！

关于变量，最后一点要记住的是，它们具有象绘图纸一样的存贮器，除非发生了下列情形，信息存放在变量中保持不变，直到：

1）对于同一变量执行另一个赋值语句；

2）执行了一个NEW或 CLEAR 指令；

3）用 RUN 命令运行了一个程序；

4）更换了计算机的电池。

当电源关闭时，变量的值仍然保留在机内。试做一遍，把机器的开关关上，然后又打开，在 RUN 方式，查找 G 的值（按 [G] [ENTER]）。一点不假，是吧？下一章当你开始程序设计时，你会发现变量是必不可少的。

J。歌 语

祝贺你！假如你坚持到这一点，你就会懂得 SHARP PC-1500 的基本知识，足以进行各种各样的计算了。由于 PC-1500 是如此的万能，每个读者将会在各自感兴趣的领域中找到它的许多用途。然而，不管你的应用领域是什么，为了充分发挥这种神奇机器的功能，你终归是想学习程序设计的。

在这里你要作一个选择。对于那些在程序设计思想方面心明、反应快速的人，我们要求他回头阅读第0章，把新的知识和那里的内容结合起来，然后转到第三章。

其他需要立刻使用那些诸如科学记数法和三角函数功能的读者，可直接转到"高级计算"一章。

小 结

1）方式——SHARP PC-1500 工作在三种不同的方式之一：RUN（运行），PRO（程序）和 RESERVE（保留）方式。方式的每一次改变都会使机器的内部功能稍有不同的变化，与汽车换挡的情形类似。RUN 方式是作计算和运行（执行）程序用的，在 PRO 方式，进行程序的写入和编辑（添加和改正）。

2）计算——在RUN方式，机器执行通常的算术运算，每一次计算之前，要按 CL 键（清除键）以清除前面的计算结果，而在计算间不要按 CL 键，使一系列计算能直接利用前面的计算结果。

3）专用字符——"$"（美元号）和 "，"（逗号）在 BASIC 中具有专门的意义，不能作为计算中的数的一部分符号。

4）负数——在数的前面用一个"－"（负号）来表示，例如：－5＋2＞＝－3。

5）混合运算——遵循通常的代数规律。括号用于指出所给定表达式的正确意义。附录O给出了表达式求值次序的全部信息。

6）不等式——不等式使用符号：＜，＞，＜＝，＞＝ 和＜＞，分别表示小于，大于，小于或等于，大于或等于及不等于。不等式为假时，其值为0，不等式为真时，其值为1。

7）左箭头键——左箭头键起无破坏性光标后移键的作用。按住这个键会使光标自动重复后移。当光标置于前面敲入的字符时，下横杠光标变为闪烁方格。按了 SHIFT 键之后按左箭头键，则是行使 DEL（删除）键的功能，删除光标置于其上的字符。

8）右箭头键——右箭头键无破坏性地使光标前移，按住这个键会使光标自动重复前移。按 SHIFT 键后按右箭头键，则是行使 INS （插入）键的功能，在当前的光标位置插入一个"空箱子"字符，可在这个位置上写上要插入的字符。

9）读出功能——按下 ENTER 键之后，便显示计算结果，原来的算式可通过按左箭头键或右箭头键读出，此时，可对算式进行修改并重新送入，对于任何出错的非程序表达式，读出功能也能起作用，在这种情形，光标将置于检出错误的字符位置上。

10）变量的使用——变量的使用大大地增加了 PC－1500 的计算能力，并使复杂的表达式获得简化。

11）赋值语句——形式为：

变量名＝表达式

字符变量名＝"字符"

分别允许存贮单一个数或16个以内的一串字符，除双引号外，字符串内可使用任何可打印的字符。

12）变量名——对于数值变量，变量名可为：

1．一个字母（A～Z）；

2．一个字母后跟一个数字（0～9）或另一个字母。

字符变量名的拼写规则相同，但所有名字后加一个$（美元号）为其词尾。

13）例外——下列的词属于命名格式的例外情形，不能用作变量名，它们是：IF，LF，N，PI，TO，IF$，LF$，LN$，PI$，TO$。

14）信息存在时间——变量内部的信息保留到：

1．执行了 CLEAR 命令或 NEW 命令；

2．用 RUN 命令运行了一个程序；

3．对同一变量执行了另一赋值语句；

4．更换计算机的电池。

关闭计算机开关不会影响存在变量中的值。

第三章　初级程序设计

前　言

程序设计技术被毫无必要地笼罩上一层神秘的色彩已经好长时期，以至多数人把它和术士或数学天才联系在一起。事实是，程序设计不需要特殊技能，也不需要你擅长于解偏微分方程。最可宝贵的是你的耐心，你的逻辑推理能力，你的细心以及好学精神。应战的愿望也是有用的（我们不想哄骗你，程序设计时是很有趣的，这里只是开一个玩笑）。

程序设计是一门技术，需要有一点技巧和训练以及大量的实践。使你成为熟练的程序设计员不是本手册的目的，我们的目的是使你熟悉基本的操作和程序设计的概念，要成为一个胜任的程序设计人员有更多的要求，就如良好的驾驶员不应只是懂得如何操纵方向盘和换挡一样。

现在已经有了许多程序设计方面的好书，我们主张你向当地的计算机商和图书馆联系。附录F列举了几本关于程序设计方面的通用的和 BASIC 语言专用的好书。

A• 什么叫做程序？

你也许会感到惊奇地发现，一个程序只不过是一组指令，计算机一次执行一条。这些指令要用计算机能够理解的语言输入计算机。SHARP PC—1500 "讲" 的是 BASIC 语言的一种方言，BASIC 语言是一种使用广泛的很流行的程序设计语言。同其它语言一样，BASIC 语言有其专门的词汇和语法规则，用这些词汇和语法规则合成语句。如果你与计算机的交谈不合语法，或用计算机所不熟悉的词汇，计算机就会提醒你出现错误。但要正确地指令计算机工作并不困难，BASIC 语言本来就已经发展成为便于教授程序设计原理的语言。它的许多语句包含英语单词和其它熟知的符号。

B．怎样使用一个程序？

当使用机器运行一个程序时，要遵循一定的方法：构成程序的指令要在 PRO（程序）方式输入机器，这些指令在 BASIC 语言中称为 "语句"。要开始执行这些语句，需要将工作方式转到 RUN（运行）方式，然后，敲入 RUN 命令指示机器着手运行程序。这一

点，对于那些在第0章中使用过两个程序的内行人来说，已经是熟悉的了。但对于跳过该章的人，让我们试输入和运行一个程序：

把工作方式转到 PRO 方式，并发 BASIC "命令"（详见下面的"C．命令与语句"）：

[N] [E] [W] [ENTER]

就会清除前面可能留在存贮器内的任何语句．敲入下列的程序行：

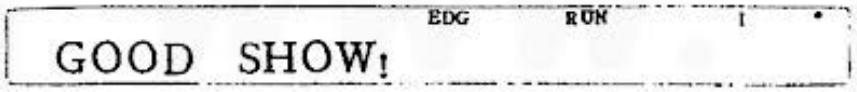
程序单：

```
10 PRINT "GOOD  SHOW! "
```

我们的一行的程序就完成了．转到 RUN 方式并敲入：

[R] [U] [N] [ENTER]

这个命令指示计算机开始执行程序中的语句．接着，机器在显示器中输出：



要对程序进行某种添加、变更或删除时，必须回到 PRO 方式．假如程序含有某种错误，不能成功地完成运行，就需要回到 PRO 方式去改正有错误的语句．这样，对程序进行加工是在 PRO方式中完成，而执行和检验程序是在 RUN 方式中进行的．

C．命令与语句

在上例中你也许注意到，我们是用两种不同的方法把我们的要求通知计算机的．NEW 和 RUN 指令在我们按了 ENTER 键之后就立即被执行，这类指令称为命令．另一方面，PRINT（输出）指令稍有不同，它在 PRO 方式送入机器，在它的前面有一个数（10），它不立即被执行，这类指令称为语句．

在某种意义上说，"命令"是告诉机器对语句做什么．例如，命令 NEW 清除目前所保存的全部语句．重要的是记住，命令不能在程序内部使用而语句几乎总是（但不完全是）组合而成为程序．

D．行　号

BASIC 程序由一系列的标记有号码的程序行构成，每一行含有一个或多个语句，本计

算机使用“行号”以保持正确的执行次序，但当程序运行时不会成为输出内容的一部份，语句可按任何次序输入，但计算机是按行号依次执行的（但受行号的修改所支配，以后我们将会看到）。

作为一个示范，让我们在前面的单行程序中加上一个语句，调到 PRO 方式，然后敲入：

键串：5 P R I N T [SHIFT] " J O L L Y [SPACE] [SHIFT] " [SHIFT] : [ENTER]

现在，运行这个修改了的程序，会发生什么现象？试在显示“JOLLY”后按 [ENTER] 键。

在本例中要注意两件事：计算机做了些什么以及要你做些什么。机器正确地按顺序排列和执行5行和10行，此外，从5行到10行，要看10行的执行结果，需要在输出之间按 [ENTER] 键。

虽然，行号可以是1至65279中的任一个整数，我们还是建议你按增量10编行号（即10，20，30，…，等等）。这样做使你能在当前的语句之间插入9个以内的语句（例如，11至19）。有些程序设计者推荐使用更大的间隔，按增量20来设置行号。不过，只要仔细进行程序设计和写入程序，将很少需要插入语句，因而，不要依赖于插入！按10增加设置行号远较后来重新标号要来得容易。

记住，两个程序行不能有相同的行号，如果发生这种情况，则旧的行（早先输入的行）就会失去。这个特性可用来删除不想要的行，只要按键输入欲删除行的行号并按键 [ENTER] 就行了。这样，当时标记行号的一个空行就有效地删除了一个具有相同行号的程序行。

然而，假如无意中使用了重复的行号，就会发生麻烦的事情。为了说明这一点，送入下列的行（当然要在 PRO 方式）：

键串：

10 PRINT [SHIFT] " A W F U L [SHIFT] " [ENTER]

现在，用与前面相同的方法运行这个程序，结果显示“JOLLY AWFUL”而失去了了原来的程序行。对吗？由于程序行的失去可能导致某些很隐秘的错误，因此，在写入程序时要谨慎小心。

E．程序行检视键

"可是"，也许你会问自己，"我怎样记得送入了什么样的行呢？"胆怯不是无畏的程序员！检视的需要已经预先考虑到了，机器提供了 [↑] 键（上箭头键）和 [↓] 键（下箭头键）．可以把这些键认为是程序行检视键．在 PRO 方式中按相应的键，一个程序行就会从当前的程序行"移上"或"移下"（这个过程形象地称为"翻卷"）．

调到 PRO 方式，使用下箭头键按升行的次序检视我们的程序，接着又用上箭头键移回到程序的顶部（10行）．注意到，假如将其中一个箭头键按住，将会自动地逐行显示．（遗憾的是，这个特点在只有二行的程序上不容易看出）．

一旦用程序行检视键到达了一个指定的行，就可用左箭头键或右箭头键编辑该程序行了．你一定会发现这些键在 PRO 方式中的操作与在 RUN 方式中（见第二章）是相同的．删除键 DEL 和插入键 INS 也可利用在语句编辑中，操作的方式也与前相同．

注：程序行进行了某种改变之后，为了使这些改变生效，必须按 [ENTER] 键．未按 [ENTER] 键，就不可用上箭头键或下箭头键移到下一个相邻的行上，否则，你所做的任何编辑会失去作用．

F．回顾已学过的一些指令

既然我们已经懂得了如何输入、运行和编辑程序，我们将要扩充有用的语句和命令的词汇．让我们通过观察某些"老朋友"：NEW 命令、LET 语句和 PRINT 语句入手．

F1．NEW 命令

如我们在前面的程序设计例子中所看到的，命令 NEW 删除现存在存贮器内的全部程序行．我们在每一个程序例子之前都使用 NEW 命令（在 PRO方式）以保证贮存在机器存贮器里的指令只有我们现程序的指令．虽然同时在存贮器内保存几个程序是可能的（而且是经常需要的），但为了避免出现错误，我们暂缓使用这个特点．

在 PRO 方式，发布命令 NEW，现在按上箭头键和下箭头键时有什么现象？

F2．LET 语句

如果你不认识 LET 语句这个"老朋友"，不要感到惊奇，我稍稍地把这也填入"老

朋友”之列。实际上，LET 语句不是别的，而正是变了形的赋值语句（假如你还不认识赋值语句，请立即重新阅读第二章“变量”一节）。

在早期的 BASIC 语言中，每一个语句都用一个“关键字”开头（象 PRINT，INPUT，等等），它指出指令做什么事情。LET 是一个关键字，它确定这样一个事实，要将一个值存贮到某个变量中。现在，普遍同意 LET 这个字不是实在地需要的。因此，在 PC—1500的 BASIC 中，关键字 LET 是可选择使用的。这样，把数 7 存贮到变量 S 的语句可以用下列两种方法之一来写：

```
S＝7　或
LET　S＝7
```

一个必须设置 LET 的例外情形是，赋值语句发生在它作为 IF 语句的一部份时。虽然我们还未讨论过 IF 语句，但这一点也是可以解释的。

使用 IF 语句，你可写一条象下面形式的指令：

```
IF　表达式　THEN　语句
```

假如 THEN 后面的语句是一个赋值语句，就必须使用关键字 LET，它要以这样的形式出现在语句中：

```
IF　表达式　THEN　LET　变量名＝表达式
```

注：在这种情形，省略 LET 会产生出错信息“ERROR 19”（或其它出错信息）。

F 3．PRINT 语句

你写的大部分程序，那些指令要遵循一个基本的模式：有读原始数据的指令，有处理数据的指令，有打印或显示结果的指令。这个模式称作输入、处理、输出的过程。

PRINT 语句是用来产生输出的主要语句。因此，PRINT 语句有几个不同的变种是不足为奇的。PRINT 语句的一般形式是关键字 PRINT 之后跟着要输出的一个项或一个项表，这些项包括要输出其值的字符串、表达式或变量名。在项表中每一项用逗号或分号分隔。

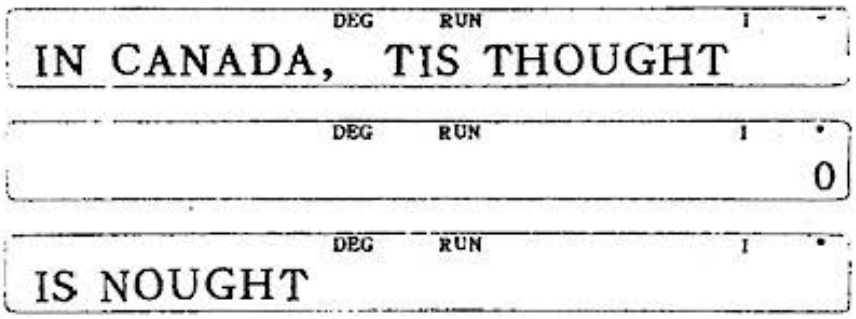
输入下列程序来说明单项的输出（记住开始之前发 NEW 命令）：

程序单：

```
10　Z$＝“IS　NOUGHT”
20　ZZ＝0
```

```
30 PRINT "IN CANADA, TIS THOUGHT"
40 PRINT ZZ
50 PRINT Z$
```

当运行这个程序时，将产生三行的输出如下（读每一行后按 ENTER 键）：



第一个 PRINT语句（30行）显示一个字符串或“文字”。注意到，双引号不作为输出的一部分输出来，它们在定界或标记你希望输出的字符串序列的始末时是必需的，字符串序列可以除双引号之外的任何字符。

第二、第三个输出语句（40行和50行）的项对于所有的读者来说应认为是变量。当输出表中使用了某个变量名时，该变量的值就被输出。在本例，我们知道变量ZZ和Z$的值是什么，因为它们的值已在10行和20行中规定了。输出一个“空值”变量的结果为零或一个空行，取决于变量是数值变量还是字符变量。

聪明的读者会观察到，字符串和字符变量的值是自显示器的左边开始输出的，这称为“左对齐”，反过来，数字或数值变量的值是“右对齐”的。

输出包含在 PRINT 语句内的表达式的值也是可以的，下面的单行程序说明了这个问题：

程序单：

```
10 PRINT (1982—1956) *365.25
```

如你所料，计算结果是一个数，是实行“右对齐”的。

为了提高效率起见，最好避免在 PRINT 语句内部计算表达式，除非该语句（及其相关的表达式）在程序中仅执行一次。

由于多数程序都是一次计算几个结果，因而普遍都是同时输出几个项。

也许最简单的多项输出语句是将显示窗分为两个部份的输出语句，每一部份包含输出表中规定的两个项中的一个。两个项在输出表中用一个逗号隔开。试用下列程序来考察输出格

式：

程序单：

```
10  A=2*PI
20  PRINT "2  TIMES  PI=  ", A
```

运行这个程序．如在单项 PRINT 语句一样，显示中数是右对齐而字符串是左对齐的．在这种情形，调整是在显示中两部分的每一部份进行的。

运用编辑技巧，将20行改为：

```
20  PRINT  A,  "=2  TIMES  PI"
```

可有几种方法完成这个编辑．

这个修改过的程序的输出内容有助于你识别显示的两个部份．

利用分号改变 PRINT 语句，可在同一行中显示两个以上的项．试建立和校核下面的程序：

程序单：

```
10  B$=  " BE"
20  T=2
30  PRINT  T; B$; "OR  NOT" ; 12/3-2; B$
```

运行这个程序将会产生如下输出内容：

```
                DEG          RUN                I
2 BE  OR  NOT  2 BE
```

这个例子说明了可输出的项用分号分隔时的 PRINT 语句的作用．在这种格式中，各输出项之间用最小的间隙邻接显示．这个输出能力便于产生习惯看的输出内容（即输出内容排在一起）．

注：假如显示信息的长度超过了显示器可利用的空间（25个字符），则输出表末尾的项就看不到。

分号的另一用途是设置在 PRINT 语句的最末端，在这种用法中，分号指示机器，目前显示器中的输出内容保留起来，而且任何新的输出内容（从下一个 PRINT 语句得到的）会同旧的输出内容在同一行中输出．这个过程做来比说容易明白，那末，让我们用下列程序来实验（别忘了用 NEW 命令）：

程序单：

```
100 PRINT "HUMPTY" ;
110 PRINT "DUMPTY"
```

现运行这个程序，与往常一样，在第一个 PRINT 语句已经显示 "HUMPTY" 之后，必须按 ENTER 键．由于分号的作用，"HUMPTY" 仍然保留在显示中，而 "DUMPTY" 就和它分占显示器．把它同下面的程序的执行作个对比，下面的程序不使用分号，其不同之处就很清楚了．

程序单：

```
10 PRINT "HUMPTY"
20 PRINT "DUMPTY"
```

下一个程序例子说明，分号可以加到要在同一显示行输出的许多行输出语句中．在显示器中一次输出太多的信息是错误的，因为机器不给出信号．作为一个程序员，就该确保不发生这种情况．试试下列的音阶程序：

程序单：

```
 20 PRINT "DO" ;
 40 PRINT "RE" ;
 60 PRINT "MI" ;
 80 PRINT "FA" ;
100 PRINT "SOL" ;
120 PRINT "LA" ;
140 PRINT "TI" ;
160 PRINT "DO"
```

现在运行这个程序，当你高兴地唱着每一个音符的时候，重复按 ENTER 键（对，对，…你也可不唱，但还是要按 ENTER 键）．

G．PAUSE 语句

PAUSE 语句是 PRINT 语句的半自动形式，它按固定和短暂的时间周期显示联合输

出表中的各个项．这样，使用者可以免除连续按 ENTER 键的麻烦．可以把 PAUSE 设想为在 PRINT 语句后跟着一个时间记数器，当计数时间结束时，继续执行程序．

PAUSE 语句的格式与 PRINT 语句的格式是相同的，我们在 PRINT 语句中所讨论的各种技巧同样适用于 PAUSE 语句．不过，结果的输出当然略有不同．为了说明 PAUSE 语句的一个用途，我们重新写入音阶程序：

程序单：

```
10 PAUSE "DO" ;
20 PAUSE "RE" ;
30 PAUSE "MI" ;
40 PAUSE "FA" ;
50 PAUSE "SOL" ;
60 PAUSE "LA" ;
70 PAUSE "TI" ;
80 PAUSE "DO"
```

观察在最后一个音阶输出之后发生什么现象：此时程序结束，显示回到提示符．这是由于在最后一个 PAUSE 语句的后面再没有其它的语句了．为了在输出最后一个音阶后"冻结"显示，我们可以将80行改为：

```
80 PRINT "DO"
```

毕竟我们没有理由不能把 PRINT 语句和 PAUSE 语句混用在程序内，你自己可试一试．

H．INPUT 语句

利用各种形式的 PRINT 语句，单独使用或组合使用，能把信息很好地提供给计算机的使用者．然而，要输出的大多数项是处理某些初始数据的结果，这个初始数据同样要由计算机的使用者提供给程序．控制这个过程的指令是 INPUT 指令．

在这个指令的最简形式：INPUT〈变量名〉中，INPUT 指令只是输出一个？（问号），然后等待使用者敲入所要求的信息．要求输入什么类型的信息取决于 INPUT 语句中变量的类型．例如：假如变量是数值型的，使用者就要送入一个数，这个数就会被贮存在该变量中．

你看出这种形式的 INPUT 语句有问题吗？有的，你们中的一些人会发现，除非使用程序的人同时又是程序设计者（也许尽管如此，也不一定），当看到问号时会不懂得该送入什么类型的数据。这就得由程序设计者始终给使用者某些信息。作为一个没有做到这一点的程序实例，送入下面的程序：

程序单：

```
10  A=0
20  INPUT  A
30  PRINT  A*PI
```

现在，我们想象使用者运行这个程序时会出现什么情况。首先是，在显示中出现一个问号，有知识的人会想到要求输入某个数据，但他不能知道该数据应是什么。假如他胡猜想，胡乱送入一个数，可能立即就会出现一个又长又复杂的数。这个数又是什么意思？按 ENTER 键继续执行下去，可是程序结束了。按他这个方法，获得全部经验需要花费许多时间。为什么？因为程序设计得不好。

解决这个问题的一个办法是使用 PRINT 语句或 PAUSE 语句，以帮助使用者使用程序。为此，我们重写上面的程序如下：

程序单：

```
10  A=0
20  PAUSE "ENTER  ANY  NUMBER"
30  INPUT  A
40  AP=A*PI
50  PRINT  A; "TIMES  PI=  "; AP
```

这个方法相当有用，因为它提示（或提醒）使用者输入信息，且又同输出的形式一致。

提示操作（输出一个请求输入的信息）很常用，后来创造了包含有提示的输入语句的各种更高级形式。首先一种是用一个分号：

```
INPUT "字符"；变量名
```

机灵的读者会认识老朋友字符串。这个字符串会被输出作为提示。光标代替了问号。在同一行中跟在提示字符串的后面，机器就等待使用者输入数据。这种形式的输入语句允许我

们把前例程序改写为：

程序单：

```
10  A=0
20  INPUT "ENTER  ANY  NUMBER" ; A
30  AP=A*PI
40  PRINT  A; "TIMES  PI=  " ; AP
```

当你运行这个程序时，注意观察使用 INPUT 语句与 PAUSE 的不同之处。

第二种提示输入的形式几乎是与第一种形式相同，它用一个逗号代替分号：

INPUT "字符"，变量名

当执行这个语句时，有关的字符串显示出来，计算机又等待输入，然而，这时，当使用者开始输入时，提示字符便被清除，使用者输入的数据就出现在显示中。这就允许数据在一个长的提示字符后面送入又不溢出显示器的末端。

试将上程序２０行的分号改为逗号并重新运行该程序。

当然，送入的数据不象我们的例子到现在为止还蕴含着的那样限于数值。要送入字符，只要在输入语句中指定字符变量，如在下面这个卓越的、演绎出来的程序：

程序单：

```
10   INPUT "ENTER  YOUR  LAST  NAME" ; L$
20   INPUT "ENTER  YOUR  FIRST  NAME" ; F$
30   I$=LEFT$ ( F$, 1 ) + "." +LEFT$ ( L$, 1 ) + "."
40   PAUSE "GEE, "
50   PRINT "YOUR  INITIALS  ARE" ; I$
```

注：不要急于弄懂３０行，它用上了下面要学到的高级技术。

INPUT 语句除了能够按受单项数据以外，还可用于将几个数据一起输入和贮存。为了达到这个目的，只要在 INPUT 语句中列出该语句要接收数据的变量，变量表中的每一个变量用逗号分隔。

作为一个多项输入的程序实例，建立一个统计程序：

程序单：

```
10   W=0 : X=0 : Y=0 : Z=0
```

```
20    INPUT "ENTER  AGES  OF  4  PEOPLE", W, X, Y, Z
30    S=W+X+Y+Z:A=S/4
40    PAUSE "TOTAL  YEARS=" ; S
50    PRINT "AVERAGE  AGE IS" ; A
```

运行时，这个有点紧凑的程序会提示你送入4个数。你送入的第一个数将取代提示字符串，因为我们在INPUT语句中用了逗号，紧跟在字符串之后。第一个数完全被入后，必须按ENTER键，在逐次送入数组之前有一个问号出现，而每一个数后都必须按ENTER键。

如我们在INPUT语句中提示字符串后用上一个分号，送入的数便不取代提示字符串，而是与提示字符串共占一行。而逐次送入数的情形与使用逗号时相同。试将20行略作改变并运行：

```
20    INPUT "ENTER  AGES  OF  4 PEOPLE" ; W, X, Y, Z
```

I·捷径以及有益的提示

因为你是那样地勤奋和耐心，我提供这一节作为一个小小的报答，我知道，这虽然没有金钱那样好，但这里包含的内容却可以使你节省一些工夫。

I·1缩写词

你们定会注意到，我们的范例程序在不断地变大。计算机的设计者们出于良好的心理，已经预先为你考虑到困难，他们使机器能够承认频繁使用的语句和命令的缩写词。

缩写词的一般形式是一个或多个指定的字母后跟一个园点。为了避免与变量名相混淆，这个园点是至关紧要的。例如，送入下面的程序，它的语句是缩写的：

程序单：

```
15    PA· "HELLO, HUMAN."
25    I· "WHAT  IS  YOUR  NAME?" ; N$
35    P· "GLAD  TO  MEET  YOU, " ; N$
```

注意到当你完成了每一行的输入（按ENTER键）时，机器就将该行的缩写词扩充为原词。它使程序变得清晰，这对于以后你查核程序错误时是极其有用的。另一潜在的好处是，存贮语句中缩写词的空间不大于（也不小于）存贮整个语句的空间。因此，缩写功能对于程

序设计者来说是十分方便的。

为了增强易读性，我们在范例程序单中将不使用这些缩写词（当然，你在送入程序时可以使用）。下面是已经采用过的语句和命令允许使用的缩写词。

PRINT	P. PR. PRI.	
PAUSE	PA. PAU.	
INPUT	I. IN. INP.	
RUN	R.	

本手册附录中包含有一个完全的缩写词表。

I·2 多语句与冒号

如在行号这一节所述，几个语句可以共占一行，机器将从左至右依次执行语句。为使机器能够把一个语句的末尾与下一个语句的开头相区分，必须使用"发信号"的字符，这个字符就是冒号（：）。用类推法，冒号的功能与本页上分隔句子的句点的功能类似，它告诉你什么时候停止阅读。

冒号的用法已在上面最后几个程序例中说明了，其一般形式是，

行号　语句1：语句2：语句3（等等）

何时使用冒号的问题是属于程序设计风格的问题，不加选择地使用冒号写出的程序很密集，会使程序难于阅读、修改或扩充。由于PC—1500本来的优越性就是个人使用以及会话性，这就高度要求你能够修改和扩充程序以适合你自己的需要。因此，我们建议你学会节制使用冒号。

假如使用冒号，它只在组合概念上相联系的那些语句上是有利的，即，只在组合那些构成程序完成单一任务的而不是完成许多种可能的任务的几个语句在同一行中是有利的。

例如，考察下面的程序，它读入三个数，求它们的平均值，计算每个数与平均值之差并输出这些差的和：

程序单：

```
10    N1=0：N2=0：N3=0
20    INPUT "ENTER 3 NUMBERS"，N1，N2，N3
30    A=(N1+N2+N3)/3
40    D1=N1-A：D2=N2-A：D3=N3-A
50    SD=D1+D2+D3
60    PRINT "SUM OF DIFFERENCES=" ；SD
```

注意程序的每一行是如何完成一项完全而又必需的步骤的。第10行清除前面可能留在变量N1、N2和N3中的值（这一步是预防使用者没能输入全部三个数）。20行收集使用者输入的数据。30行求这些数的平均值，40行计算三个差值，50行计算这些差值之和，60行输出结果。

每一行中的指令与描述程序的英语语句是对应的，这并非偶然，而正是你要遵照的良好的程序设计的一个方法。

与使用缩写词的情形不同，冒号的使用会对用于贮存程序的存贮器的占用量产生影响。这主要是用了冒号把几个语句设置在同一行中的缘故。每个行号占用了几个程序步（存贮单元的单位），因此，程序中行号越少，程序占用的存贮器的体积越小。

关于冒号的使用，最后一点要说明的是，每一个程序设计者必须解决每一个程序的易读性和易变性与应用这个程序时需要存贮器的量之间的矛盾。

J·在PRO（程序）方式改正错误

虽然缩写词和冒号能帮助我们易于送入程序，但它们却不能担保我们免于发生错误，甚至专业程序设计者在检视他的程序时也难免会发现有错。这个意思就是说，当你运行程序时，迟早会遇到某个错误。（假如你承认它们是要解决的难题并仔细地追踪下去，大多数是易于改正的。）PC—1500有几个特性能帮助你解决这方面的问题。

在显露不正确的语句方面，机器会停止程序并用一个简短的信息指出问题，例如：

```
                                    RUN    I
ERROR 1 IN 20
```

为了说明问题，让我们编写一个故意含有错误的程序，送入：

程序单：

```
25    PAUSE "HUMPTY  DUMPTY"
50    PRIMT "WAS  AN  EGGHEAD"
```

现运行这个程序．当出现错误信息时，按↑键（上箭头键）．只要你按住这个键，显示器中就会显示着机器检出错误的行．闪烁的方格能够提供一个考究问题的性质的线索．

要改正错误的语句，按CL键退出程序的执行并转换到PRO方式，按上箭头键，但不要按住它，显示器中将再一次显示出错的行：

```
50    PRIMT "WAS AN EGGHEAD"
```

现在你便可着手用熟悉的INS，DEL和箭头键来编辑程序．当你完成编辑后，必须按ENTER键以指示机器存贮已改正的行。

K·LIST（列表）命令

显示程序中个别的行的另一个方法是使用LIST命令，其形式为：

LIST

或 LIST〈行号〉

若不给出行号，则显示第一个程序行．若指定了一个行号，则显示带有此行号的程序行．假如程序中没有所给定行号的程序行，则显示行号数较大的下一程序行．例如，设有下列程序：

```
15  PRINT "MOTHER  GOOSE"
30  PRINT "WAS  A" ;
45  PRINT "QUACK"
```

那么，命令LIST 40就显示45行，命令LIST将显示15行，命令LIST 30将显示30行．

注：如果你指定的行号比任何现存的程序行号都大，会产生出错信息 ERROR 11.

L·多个程序的存贮和运行

正如上面所提及的，同时在计算机的存贮器内保存一个以上的程序是可能的．要做到这一点的技巧是给每个程序以一定的行号范围．例如，一个程序可以有行号10～200，而第二个程序就要从300标记到500．当然，你必须细心编程序，不可偶然混杂各个程序的语句（由于编号不正确），否则，会产生不可预测的结果．

L．1　END 语句

同时存贮几个程序所发生的另一个问题是，每一个程序正好是一组带标号的语句，语句的行执是按行号从小到大的次序进行的．因而，机器怎么知道何时完成了给定程序的语句的执行？答案是，除非你告诉机器，它是不能知道的．用于通知计算机已经到达程序末尾的语句是END语句．

至今，我们未用到，也不需要用到END语句．机器直通通地按从小到大的行号顺序执行全部程序行直到执行完语句。当它执行完语句时，计算机就会判定程序运行已经完成，并返回到等待下一个命令的状态。然而现在，我们必须告诉机器在继续执行到下一个程序去执行之前就停止执行指令．

为了说明多程序的使用，送入下面的行：

```
10    PAUSE "HUMPTY  DUMPTY"
20    PAUSE "HAD A  GREAT  FALL"
30    PRINT "BUT  NOT  A  GOOD  SPRING"
40    END
200   PAUSE "THE  OLD  MANS  GLASSES"
210   PRINT "WERE  FILLED  WITH  SHERRY"
220   END
```

L．2　RUN< 行号 >（从某行号开始运行）

好，现在存贮器中已经有了二个程序，你将如何分别运行每一个程序？如果你胡乱试用普通的RUN命令来运行，你就会发现，完全是从第一个程序开始运行．要从第二个程序开始运行，我们需要使用一个变形了的 RUN 命令，它告诉机器自哪一行开始运行，这就是 RUN < 行号 > 命令。那末要开始第二个程序，操作：

```
RUN    200
```

象大多数有帮助的命令一样，RUN 行号命令也会出现问题．因为它命令计算机从什么地方开始执行下面的指令，而且由于计算机是一个忠实的工具，这样，程序就可能在中间开始执行．可想而知，这是不能凭自己选择而行的，如果这样做，就会产生某些异常的结果．试发命令：

```
RUN    30
```

运行第一个程序．这个错误较之在较复杂的程序中出现的错误来，是属于轻微的．

M • 控制语句

到目前为止我们的程序指令都具有一个连贯的顺序，计算机每一次执行一条指令．那些编写程序有经验的人会发觉，它不总是完成一个任务的最好的方法．也许你常允许听众选择几个方案之一，有时，你想通过包含一个命令来"压缩"你的指令，象"现重复最后三步直至结束"之类．

这些功能已经编进了BASIC的语句中，称为控制语句，这些语句决定什么时候执行、是否执行以及如何层次执行其它语句．控制语句使你能编写功能很强的通用的程序．下面几节中我们将讨论BASIC中的主要控制语句：IF…THEN，GOTO，和GOSUB 语句．

N • IF…THEN语句

在 BASIC语言中，可能性的选择由IF语句提供．带有IF语句的程序能对输入的数求值并作出判断：

程序单：

```
10 PAUSE "ARE  YOU  ASLEEP?"
20 INPUT "TYPE  YES  OR NO" ; SX$
30 IF SX$= "YES" THEN PRINT "OH, SORRY TO DISTURB YOU"
40 END
```

如果你的回答是肯定的话，这个催眠的程序才产生输出．30行的程序说明了，IF语句的一般形式是：

IF 〈条件〉 THEN 〈语句〉

程序执行时，对嵌在 IF 子句中的条件进行检验，是否执行THEN后面的语句，取决于检验的结果．这种检验通常是一个不等式，称为"条件"．记住，不等式是比较式，它不是真就是假，若不等式为真，则执行THEN后面的语句，若不等式为假，则跳过这个语句．

在我们的程序例子中，要检验变量SX$是否等于字符串"YES"．假如相等并且仅当相等时，执行 PRINT 指令．假如SX$不等于"YES"，则忽略 PRINT 指令．无论在何种情形，不管 PRINT语句是被执行还是被忽略，机器将同一般情形一样，处理下一行（在我们的程序例子中，下一行是40行）．

注意到，我们可以通过改变为几行的方法将检验条件修改为下列形式：

```
30    IF  SX$ = "NO"   THEN  END
40    PRINT "OH, SORRY  TO DISTURB YOU"
```

然而，这个程序与原来的程序不尽相同，它允许计算机与人对话时，不论误敲入答案或是不回答 "NO" 都能处理．另外，这个程序设置了二个终点，即30行的END语句和40行中蕴含着的END语句．在一个程序中有几个终点不是良好的程序设计，尽管这些语句反过来说明，要使程序正确地运行，需要正确的语句顺序和适当的检验．在下一节中我们将会看到表示我们的程序的第三种方法，这个方法采用 GOTO 语句解决第二种方案中的问题．

虽然 IF 语句中，条件通常是一个不等式，但它不是非要不可的．在PC—1500的BASIC语言中，任何值为正的、非零数的表达式便被认为真，任何表达式，例如 5 — 9，它的值是负数或零，便被认为假．这就说明了为什么不等式能作为检查的条件：记住，值为真的不等式得到 1，而值为假的不等式会得到 0．这种表示真假值的方法不是各种计算机的标准表示法，一律套用会产生模糊不清的程序，故使用这种方法时要审慎．

还有一点要提醒的是次序，假如 THEN 后面的语句是赋值语句，必须使用关键字 LET，不这样做就会产生出错条件，这一点已在LET语句一节中讨论过了．

6．GOTO 语句

在上一节中，你也许注意到，我们的选择受到 IF 语句中对条件进行检验的限制，假如条件为真，我们只能执行一个语句．为了方便，我们可能会要求执行几个语句，GOTO 语句正是使我们能够这样做的语句．

GOTO 语句改变语句的执行流程，它告诉机器到某一行（而不是下一行）去并从这一行开始，顺序执行语句．这种 "跳跃" 的作用是完全地跳过某些语句．例如考察下列程序：

程序单：

```
10    PAUSE "ESCHEW" ;
20    GOTO  50
30    PRINT X * 3 / 4 + 2
40    PRINT "A  BOOK WHICH  EMPLOYS" ;
50    PRINT "OBFUSCATION! "
60    END
```

当然，这个程序一般是按行号自小到大的顺序执行的，但20行的 GOTO 语句的作 用是使机器立即跳到50行并自这一行开始往下执行．30行和40行就决然执行不到．

GOTO语句的一般形式是：

GOTO〈表达式〉

这里，表达式求值为一个数，它应是一个有效的程序行号（即 1 ～65279）．

注：指定一个不存在的行号会产生出错信息 ERROR 11．

如果我们希望根据某种选择的结果需要执行几个语句，可把 GOTO 语句与 IF 语句连用：

```
10    IF 条件式    THEN  GOTO  100
20    这里的语句仅当条件检验为假
      ⎧                    ⎫
      ⎨                    ⎬
      ⎩时才执行            ⎭
90    GOTO    200
100   这里的语句仅当条件检验为
      ⎧                    ⎫
      ⎨                    ⎬
      ⎩真时才执行          ⎭
200   这里的语句
      ⎧                    ⎫
      ⎨                    ⎬
      ⎩总是要执行的        ⎭
999    END
```

这个问题的逻辑适用于许多场合，任何数目的语句都可以填入由 GOTO 语句形 成的两个部分中的每一段．

作为这种结构的一个实例，下面从支票簿程序节选出的程序段判断所给的输入的量是存款（正数）还是取款（负数），最初的余额由使用者输入：

程序单：

```
10  INPUT "INITIAL  BALANCE?", B
20  INPUT "TRANSACTION  AMOUNT?", TA
25  IFTA=0  THEN  GOTO  80
30  B=B+A
40  IF  TA<0    THEN  GOTO  70
50  PRINT "EPOSIT  OF  $"; TA; "POSTED"
```

```
60    GOTO 80
70    PRINT TA; "DOLLARS WITHDRAWN"
80    PRINT "FINAL BALANCE=" ; B
90    END
```

注意到这个程序仅当交易额为零时才立即结束。第二个 IF 语句就是一个说明前面提及的逻辑结构的语句。注意，除了有二个与 IF 语句相联系、起不同作用的语句（50行和70行）之外，还有一些语句（80行和90行）的执行与 IF 语句的检验结果无关。

现在我们可以写出上一节程序的第三种方案了：

程序单：

```
10    PAUSE "ARE YOU ASLEEP?"
20    INPUT "TYPE YES OR NO" ; SX$
30    IF SX$ < > "YES" THEN GOTO 99
40    PRINT "OH, SORRY TO DISTURB YOU"
99    END
```

这个方案修改了检验条件，它是通过指定使用者的输入不等于 "YES" 时所要执行的某些动作而实现的，这样，就免除了第二种方案中的问题，并将 IF 语句扩展为如上所述较大型的形式。在程序设计中，用这种方法重新安排语句往往是一个有用的策略。花费一些时间多做练习，你就能编写出较好的程序。

GOTO 语句的另一个很普通的用途是使机器重复执行一系列的语句，这种处理方法称为 "无条件循环"。下面是一个说明无条件循环的简单例子：

程序单：

```
10    WAIT 30
20    PRINT "CHUG CHUG CHUG CHUG"
30    PRINT "     CHUG CHUG CHUG CHUG"
40    GOTO 20
```

不幸的是，这个程序始终不停地交替显示20行和30行的字符串（可用 BREAK 键停止程序的执行）。我们应提供一种终止这种程序的方法。可用一个 "计数器" 以及一个 IF 语句。计数器是一个变量，里面记录我们做了某种事情多少次（即用它计数）。使用这种技术，IF 语句就能检验 PRINT 语句的执行是否已经完成了预定的次数。当然，该重复多少

次完全是由我们决定的。让我们选择每一个 PRINT 语句执行10次（10次后停止）。

用计数器和 IF 语句，我们可以写出：

程序单：

```
10  WAIT  30
15  C=1
20  PRINT "  CHUG  CHUG  CHUG  CHUG"
30  PRINT "        CHUG  CHUG  CHUG  CHUG"
40  C=C+1
50  IF  C<=10  THEN  20
60  END
```

循环的每一次执行都跟随计数器进行。注意在15行我们必须给计数器赋一个初值，在40行给它增值以体现语句20行和30行又执行了一次。

在程序内部使用计数器和循环还有许多种其它方法，遗憾的是，我们这里无法一一介绍。建议你自学附录 F 所列的参考书目中的一本。

同语句一样，GOTO 指令也是一条命令。作为一条命令，它的用法自然与作为一条程序语句的用法不同。在 RUN 方式，GOTO 指令作为命令使用时与 RUN 命令类似，开始程序的执行。差别在于程序指令执行前的内部处理有所不同（关于比较用来启动程序的方法不同的问题，见第八章"开始程序的执行"）。与RUN命令不同，GOTO命令启动程序时将不清除任何变量的值。

要用GOTO命令开始程序的执行，送入：

GOTO 〈行号〉

这里，行号是要执行的程序的第一行。

注：指定一个不存在的行号将产生出错信息 ERROR 11。

P·FOR…NEXT 语句

正如我们在上节所看到的，重复一串指令的能力是非常有用的。事实上，这个特点经常用到，为此，BASIC 语言中编入了几个语句使这种处理工作自动化。这就是 FOR 语句及其伙伴 NEXT 语句。FOR 语句与 NEXT 语句一起，括入需要重复执行若干次的一串指令。FOR 语句有一个辅助计数变量及一个内装检验条件，还允许说明计数变量的初值和增

量的值。

FOR—NEXT 语句的一般形式为：

FOR〈计数变量〉=〈初值〉 TO 〈终值〉 STEP 〈增量值〉

这里：

计数变量为用以负责循环计数的变量名；

初值为第一次进入循环之前计数变量的值，这个值允许取的范围是－32768～32767；

终值为用于检验的数，若计数变量所含的值超过终值，循环便终止，这个值的合法取值范围是－32768～32767；

步长增量值是一个任选子句。增量值指示每一次通过循环时计数变量要增加或减少的值。它必须取－32768～32767中的整数，若省略了整个步长子句，则假定增量值为1。

这个语句有许多内容要讨论，故让我们用一些简单的程序例子来观察这个语句的工作情况。第一个程序类似于使用计数器的“CHUG CHUG”程序的方案，这里是用输出计数变量C的值来代替显示“CHUG”串：

程序单：

```
15   FOR  C=1  TO  10
30   PAUSE  C
50   NEXT  C
```

（要使这个程序象前面的程序那样连续显示“CHUG”，只要插入该程序的语句10、20和30），注意到，这个方案较之原来的方案要简短精炼，它用较少的语句完成相同的计算和循环功能。

为了防止对 FOR—NEXT 语句是做什么的存在某些模糊的认识，这里我们把 FOR…NEXT 语句与等价的 IF 语句作个比较：

```
10   FOR  I=1  TO  8
20
:    （要重复执行的某些语句）
90   NEXT  I
100  END
```

```
10   I=1
12   IF  I>8  THEN  100
20
:    （要重复执行的某些语句）
90   I=I+1
92   GOTO  12
100  END
```

第二个程序例子说明如何设计一个由变量的值控制循环的“通用循环”程序。这个程序开始时询问使用者计划重复多少次并送入次数，把这个数存放在变量N中，并开始循环，通过几个语句累进一个数进行处理，循环执行N次。在N小于或等于零的情形，程序终止而不做进一步的处理。

程序单：

```
10   N=0 : V=0 : T=0 : A=0
20   WAIT 0
30   INPUT "HOW MANY VALUES?" ; N
40   IF N=0 THEN GOTO 999
50   FOR I=1 TO N
60   CLS : CURSOR 0
70   INPUT V
80   T=T+V
90   NEXT I
100  WAIT : CLS : CURSOR 0
110  A=T/N
120  PAUSE "TOTAL=" ; T
130  PRINT "AVERAGE=" ; A
999  END
```

FOR…NEXT语句不要求计数变量总是增加1或总是给它置初值1．程序员可用STEP子句规定计数变量增量（或减量）的大小。下一个程序例子说明这一点：

程序单：

```
10 WAIT 30
20 FOR HS=2 TO 8 STEP 2
30 PRINT HS; "I " ;
40 NEXT HS
50 WAIT 60 : CLS : CURSOR 0
60 PRINT "WHO DO WE APPRECIATE?"
```

```
70    PRINT "SHARP  PC—1500! "
80    END
```

除了允许往上计数之外，STEP 子句还允许机器往下计数．这可通过修改初值、终值和规定一个负的增量来实现，下面的程序（献给各地的消费者）说明这一点：

地下地毯商场以每平方英尺0.99美元的惊人低价供应地毯给 LAQANN．地毯的半径范围从40英尺（旅馆会客室用）到1英尺（浴室用），每一种规格的地毯与下一种较小规格的地毯之间半径之差为3英尺．顾客克拉夫蒂先生决定在他的SHARP 计算机上用程序算出每一种地毯的价格．他写出下列的程序：

```
10    FOR  R=40  TO  1  STEP  -3
20    P=.99*(PI*R↑2)
30    PRINT R;" FOOT WODEL  IS $";P
40    NEXT R
50    END
```

他发现地毯的价格范围是从4976.2美元到3.11美元。

Q• WAIT 语句：

WAIT语句允许程序员改变输出语句的工作，由PRINT语句显示的信息可停留在显示器中一个由WAIT语句规定的时间周期。

WAIT 语句的格式为：

WAIT 〈参数〉

参数是任选的．如果不规定参数，那末，隐含着一个"不固定"的时间周期，即：显示器中的输出信息保留到使用者按ENTER键为止．这是到此为止我们的大部份程序所使用的工作方式。

假如给出了一个参数，则所有后继的PRINT语句就会使输出信息保持在显示器中一个时间周期，这个时间周期与参数规定的数成正比．这种形式的输出与PAUSE语句类似，所不同的是PAUSE语句的时间周期是固定的．注意，WAIT语句并不影响PAUSE语句的操作．

参数（或求值为一个数的表达式）必须取0～65535 之间的数．WAIT 0使显示的信

息相当快，使得实际上无法阅读。WAIT 65535 会使得每一个 PRINT 语句的显示输出信息约17分钟！较为实用的是，WAIT 64 给出的时间周期约为一秒，而WAIT 3840 约为1分。为了重新调整 PRINT 语句的操作，使它能等到使用者按 ENTER 键，可用不带参数的WAIT指令。

示范程序

这个程序说明 WAIT 语句对后继的 PRINT 语句的影响。这里我们让等待的时间周期从0变到102，按增量2在循环中改变。BEEP 语句只是为了帮助你领略输出的时间间隙，因为所发生的作用太快，难以看清。

程序单：

```
10   FOR W=0 TO 102 STEP 2
20   WAIT W
30   BEEP 1, 5
40   PRINT "·";
50   NEXT W
60   END
```

R．READ、DATA 和 RESTORE 语句

程序里的数据并非都要由程序使用者输入的。某些有用的数据往往是相对固定的，例如，财政上的税率表或工程上的应力常数。这些类型的数据可以通过使用 DATA、READ 和 RESTORE 语句放置在程序内，而在需要时利用它。这些语句一起联合使用，规定程序中使用的数据，将数据送进变量，以及需要时重复这个过程。

DATA 语句由关键字DATA后跟一个数据项表构成。数据项表中包括实数或科学记数法表示的以及字符串，表中的项用逗号分隔。数据语句可以出现在程序中的任何地方，但许多程序员喜欢将它们集中在程序的开头，这样使得在阅读程序时较易找到。

典型的DATA语句可以象如下的语句：

```
10   DATA "MOBY DICK", 20000, "WHITE", "M", 112
```

READ 语句由关键字READ后跟一个变量名表构成。这些变量名可以是数值变量名或字符变量名，表中的变量名用逗号分隔。READ 语句使数据项从 DATA 语句中读出并将

它存放到相应的变量之中。和前面的 DATA 语句对应的READ语句为：

```
120  READ  N$, WT, C$, SX$, L
```

机器要求，每一次执行 READ 语句就要有一个DATA语句的相应的项。那末，下面的程序会在３０行出现错误，因为全部数据项已被２０行的 READ 语句用完了。

```
10  DATA  1, 2, 3
20  READ  A, B, C
30  READ  D
```

要改正这种情形，可给10行加上一个数据项：

```
10  DATA  1, 2, 3, 65
```

或者我们可以用另一个 DATA 语句，放在程序的任何地方：

```
10  DATA  1, 2, 3
20  READ  A, B, C
30  READ  D
40  DATA  65
```

这就说明，机器把全部DATA语句看成一个单一的数据项表，当计算机遇到 READ 语句内的每一个变量名时，就将表中的下一个数据项赋给该变量。假如机器不能满足数据项的要求，就停止程序的执行并发出一个出错信息。当程序以正常的方式结束时，没用上的多余数据项被忽略。

假如下一个数据项的类型（数值或字符）与要读取数据的变量类型不匹配，就会产生错信息。良好的程序设计员将数据项分别编进不同的 DATA 语句中，每一个数据项与程序中的 READ 语句对应。下面是一个说明四次读取三个数据项的程序：

```
10  DATA  1, "A", 1
20  DATA  2, "B", 3
30  DATA  5, "C", 8
40  DATA  13, "D", 21
50  FOR  I=1  TO  3
60  READ  A, A$, Z
70  T=T+A*Z
80  NEXT  I
```

10行到40行可以写成：

```
10  DATA  1, "A", 1, 2, "B", 3, 5, "C", 8, 13, "D", 21
```

或甚至可以写成：

```
10  DATA  1
20  DATA  "A"
30  DATA  1
40  DATA  2
    :
```

（等等）

这两种变换形式使得难以看清每一次由READ语句读三个数据项，由于数据项的排列的方式是：数、字符、数，不易立即分辨，因而这些变换形式还使得难以识别数据项的类型。

有时要求重新利用 DATA 语句中的某些或全部数据项。RESTORE 语句就允许这样做。RESTORE 语句使机器重新利用程序里第一个 DATA 语句开始的数据项，这样，RESTORE 语句后执行的任何 READ 语句便读取前面用过的数据项。

RESTORE 语句还能有选择地指定重复利用的数据项。语句：

```
RESTORE <行号>
```

能使自指定行号的 DATA 语句开始重复利用数据项。例如：

```
10  DATA  8, 4
15  DATA  1, 3, 6, 8E2
100 READ  X, Y
110 READ  M, N, O, P
120 RESTORE 15
130 READ  A, B
```

程序执行完毕时，变量A和B的值将分别是1和3。

DATA 语句除了用行号标记之外，还可加上一个字符标记，然后，RESTORE语句就可以自加上标记的 DATA 语句开始重新利用数据项。下面的语句是加标号的 DATA 语句的例子：

```
20 "A" : DATA  1, -12, 2, 8
```

下列程序段使读取数据恢复回到标号为"X"的 DATA 语句，

```
10    DATA 4.2, 3, 1
20    "X" : DATA -2, 0, 3, 5
100   READ Q, Y, Z
110   READ MA, MB, MC, MD
120   RESTORE "X"
130   READ N, Z
```

程序执行完毕，变量N取－2，Z取0。

S．REM（注释）语句

REM 语句提供了一个在程序的语句间插入注释的能力，虽然这些注释会被机器忽略，但却是极重要的，因为能帮助其他人阅读你的程序。你的程序越容易阅读和理解，就会有更多其他的程序员希望使用它，并且，也许会改进它。

为了减少敲键输入的负担，在这本手册的例子中我们省略了注释。我们建议你不要照搬这些例子。注释对于程序的打印、保存或交流来说大概是最重要的了。不要靠你的记忆力去记程序中每一个变量的意义，要使用注释来记！如果不这样，六个月后你就会忘记的。

注释内容跟在关键字 REM 后面，可以插在程序中自成一行，或者插在另一个语句或一连串语句的末尾，注释不能出现在可执行语句的前头或中间，原因是：机器一见到 REM 这个关键字时就忽略了这一行中剩下的任何字符。假如 REM 用在一行的开头，则该行有效的程序语句就会被忽略。

T．GOSUB…RETURN 语句

开始设计程序时，你会发现在单一程序内部，重复地利用了某些功能，例如，那些功能可能包括计算圆面积或接收和检查使用者所给的数。这种重复使完成该项功能的语句成倍增加。为了避免这种不经济的语句重复，程序员可以使用 GOSUB 语句

GOSUB 语句允许将程序中多处出现的一组语句设置在另一边，这组语句称为"子程序"（因而就有术语 GOSUB）。在程序中这组语句要出现的每一个地方，就插入一个 GOSUB 指令。

GOSUB 语句命令机器开始执行设置在另一边的一组语句，这一过程称为"调用子程序"。由于 GOSUB 语句改变程序的正常执行顺序，因而它和 GOTO 语句类似。但差别在于计算机开始执行子程序的语句之前，"记住"现在在什么地方，当机器完成了子程序的执行后就返回到离开主程序的那个地方。这称为从子程序"返回"。

但是，机器如何判明构成子程序的语句终结了？回答是：必须用 RETURN 语句来通知机器。

GOSUB 语句的形式为：

GOSUB 〈行号〉

这里，行号是子程序的第一行的编号。RETURN 语句的形式很简单，它是：

RETURN

作为一个使用子程序的例子，我们考虑一个计算并比较两个给定长和宽的矩形面积的程序。

程序单：

```
10  REM READ IN LENGTH AND
20  REM WIDTH OF TWO RECTANGLES
30  FOR I=1 TO 2
40  PAUSE "RECTANGLE"; I; ":"
50  INPUT "ENTER LENGTH, WIDTH", L, W
60  GOSUB 200
70  IF I=1 LET A1=A
80  IF I=2 LET A2=A
90  NEXT I
100 REM PRINT AND COMPARE THE
110 REM AREAS OF THE RECTANGLES.
120 PRINT "AREA OF RECTANGLE 1"; A1
130 PRINT "AREA OF RECTANGLE 2"; A2
140 IF A1>A2 THEN 170
```

```
150  PRINT "AREA OF 2 IS > AREA 1"
160  GOTO 180
170  PRINT "AREA OF 1 IS > AREA 2"
180  END
200  REM SUBROUTINE TO COMPUTE AREA
210  REM OF RECTANGLE GIVEN LENTH
220  REM OF SIDES IN L AND W
230  A=L*W
240  RETURN
```

注意子程序开始的地方。所有的子程序应放在主程序的最后语句 END 语句的后面，防止被按正常过程顺序执行。每一个子程序必须用一个 RETURN 语句结束。

子程序可以包含任何合法的语句，可以完成任一种所要求的处理。良好的 BASIC 程序员将他们的程序设计为一组“程序片”或“程序块”，子程序往往用来给每一个程序块编上号码，然后用主程序来控制子程序的执行次序。关于这方面的更深入的内容可参考附录F所列的关于结构程序设计的书籍。

程序方式编辑功能归纳

1）程序行要至少按10的间隔编号。这样就能在任意两个现存程序行之间，仅仅通过选用该两行的行号之间的某一个行号来插入新的程序行。例如，要在40行和50行之间插入一行，可从41至49中给该新行一个行号。

2）要删除一个现存的行，只要敲入它的行号并按 ENTER 键就行了。

注：由于几个语句可以一起组合在一个单行里，因此，当删除程序行时要多加小心。

3）上箭头键和下箭头键可用于逐行显示机内程序行，每次一行（移上或移下）。按住其中一个键可使自动重复移动。

4）LIST 命令可用于直接地显示给定的行。命令 LIST 能显示存贮器内第一个程序的第一行。LIST N（这里N是一个行号）会显示行号为N的行，或者，假如不存在行号N，则显示第一个行号大于N的行。

5）一旦显示了一行，左箭头键和右箭头键可用于移动光标到行中任何地方。然后可用

INS 键和 DEL 键对程序进行插入或删除的修改。

注：假如对一行进行修改，必须按 ENTER 键，然后才能显示下一行，否则，这种修改无效（即没进行修改）。

6）假如程序在执行中遇到错误发生，上箭头键会"记住"错误发生的行。要读出这个行，可进入 PRO 方式并按上箭头键。

注：程序数据区为程序和数据所共用，当程序区和数据区的总和可能大于程序数据区的容量时，程序设计时按 6 5 2 7 9 END 。

第四章　高 级 计 算

A．科学记数法

要送入一个科学记数法表示的数（A×10∧B）时，可输入尾数，按字母键 E 并送入指数．

例 1：按键输入 6.7×10^{8}：

键串	显示
6.7	6.7_
[E]	6.7E_
8	6.7E8_

例 2：按键输入 -9.12×10^{-34}：

键串	显示
−9.12 [E] −34	−9.12E −34_

只有前面 10 个数字的尾数是有效的（见例子），对于小于 1 但大于－1 的数据，精确到最大 10 个数字．

例 3：按键输入 1 2 3 4 5 6 7 8 9 8 7 6 5：

键串	显示
1 2 3 4 5 6 7 8 9 8 7 6 5	1 2 3 4 5 6 7 8 9 8 7 6 5_
[ENTER]	1.2 3 4 5 6 7 8 9 8E 1 2

例 4：按键输入 9.87654321234：

键串	显示
9.87654321234	9.87654321234_
[ENTER]	9.876543212

例5：按键输入0.0000000002345678：

键串	显示
.0000000002345678	.0000000002345678_
ENTER	2.345678E−10

例6：按键输入 $0.00001234567\times10^{24}$：

键串	显示
.00001234567 E 24	.00001234567E24 _
ENTER	1.234567E 19

注意，关于指数，只有最后敲入的二位数字有效：

例7：按键输入 3×10^{123}：

键串	显示
3 E 1 2 3	3 E 123_
ENTER	3 E 23

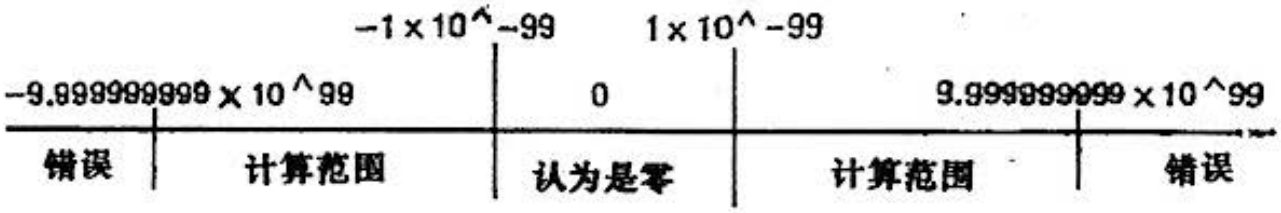
例8：按键输入 4×10^{-3210}：

键串	显示
4 E−3210	4 E−3210_
ENTER	4 E−10

B．计算范围，上溢出，下溢出

大多数的计算机都有一个能够进行计算的数值范围，在 PC−1500 中，这个范围是 9.999999999×10^{99} 到 $-9.999999999\times10^{-99}$ 之间的任何数。超过这个范围的数对于机器来说是大得无法处理了，会产生“上溢出”错误状况（发错误信息 ERROR 37）。也存在“下溢出”的情形，在这种情况下，数变得太小。下溢出情况不发出信号，也不出现错误信息

或出现停机。进入 -1×10^{-99} 至 1×10^{-99} 范围内的数被机器认为是零，可用下图说明如下：



例 1：假如试图解算式 $(5.67\times10^{55})*(8.90\times10^{65})$，会产生上溢出：

键串	显示
(5.67 E 55) * (8.90 E 65) ENTER	(5.67 E 55) * (8.90 E 65)_ ERROR 37

错误 3 7 表示计算上溢出。

C．开方，乘方和 PI（π 值）

开方：

例 1：求 7 3 的平方根：

键串	显示
SHIFT √ 7 3 ENTER	√ 73_ 8.544003745

例 2：求 $\sqrt[4]{256}$：

键串	显示
SHIFT √ SHIFT √ 2 5 6 ENTER	√ √ 256_ 4

例 3．求 $\sqrt{3^2+4^2}$

键串	显示
SHIFT √ (3 • 3 + 4 • 4 (ENTER	√ (3 * 3 + 4 * 4)_ 5

这个算式也可用下列方法计算：

例4：

键串	显示
SHIFT √ (3 SHIFT ^ 2 + 4 SHIFT ^ 2) ENTER	√ (3^2+4^2)_ 5

乘方

乘方或取幂功能，允许将一个数进行乘方。

例1：计算 4^3（$=4\times4\times4$）：

键串	显示
4 SHIFT ^ 3 ENTER	4^3_ 64

例2：计算 $3^{3.2}\times4^{-2.4}$：

键串	显示
3 SHIFT ^ 3.2 • 4 SHIFT ^ −2.4 ENTER	3^3.2•4^−2.4_ 1.207380162

例3：计算 $4^{(3^3)}$：

键串	显示
4 SHIFT ^ 3 SHIFT ^ 2 ENTER	4^3^2 262144

PI（π值）

π值（3.141592654）作为固定常数已经贮存在符号 PI 和 π 中。在需要用到π值的地方，就可以用其中之一个符号参与计算。

作为一个实例，我们求直径为5英尺的地毯的面积。在 RUN 方式中按键：

键串	显示
PI• （5／2） [SHIFT] [^] 2 [ENTER]	PI•(5/2)^2_ 19.63495408

D．角度计量方式

PC—1500 允许在下列三种角度计量方式的任一种方式中进行角度函数的计算。

要设置 PC—1500 进入角度制方式，按键：

DEG . [ENTER]

（DEG会出现在显示器顶部）

要设置PC—1500进入弧度制方式，按键：

RAD . [ENTER]

（RAD会出现在显示器顶部）

要设置PC—1500 进入百分度制方式，按键：

GRA . [ENTER]

（GRAD会出现在显示器的顶部）

E．三角函数

PC—1500 提供了六种三角函数，它们是 SIN，COS，TAN，ASN，ACS和ATN。每一种函数可在 GRAD、DEG 或 RAD任一种方式中进行计算。执行方法如下：

DEG . [ENTER] ——→ 设置角度制方式

SIN 30 [ENTER] | 0.5 | 角度制中SIN30°的值

[CL] ——→ 清除显示

GRA . [ENTER] ——→ 设置百分度制方式

SIN 30 [ENTER] [4.539904997 E−01] 百分度制中 SIN 30° 的值

[CL] ——→ 清除显示

RAD . [ENTER] ——→ 设置弧度制方式

SIN 30 [ENTER] [−9.880316241 E −01] 弧度制中 SIN30 的值

上面的例子中，在每一种角度计量方式中计算 30 的正弦，得出三个不同的（但却是等价的）结果，反三角函数也可如下进行计算：

RAD . [ENTER] ——→ 设置弧度方式

ASN-0.5 [ENTER] [−5.235987756 E−01] −0.5的反正弦值

DEG . [ENTER] ——→ 设置角度制方式

ASN-0.5 [ENTER] [−30] −0.5的反正弦值

ACS(-.5+.1) [ENTER] [113.5781785] 反余弦值

ATN(7/3) [ENTER] [66.80140949] 反正切值

在弧度或百分度方式，假如需要计算：SIN（x／y），cos（x／y），TAN（x／y），（这里 x／y 为非零数），把分数作成等式，这样 P=x／y，那末，变成计算 SINP，COSP，TANP。

F．对数函数LN，LOG及指数函数EXP

LN，LOG

函数 LN 计算自然对数（以 e 为底），而函数 LOG 计算常用对数（以 10 为底）。

这些计算在RUN方式进行如下：

LN 7.4 [ENTER]	2.00148
LOG 7.4 [ENTER]	8.692317197E−01
LN 25 [ENTER]	3.218875825
LOG 100 [ENTER]	2

EXP

LOG 的反函数是 10 的幂值，例如：

LOG 100 [ENTER]	2
10 [SHIFT] [^] 2 [ENTER]	100

由于自然对数（LN）不是以10为底而是以e为底，故需要一个反函数。这个函数就是EXP。

例：LN 7.4 [ENTER]	2.00148
EXP 2.00148 [ENTER]	7.399999998

G．角度转换

PC−1500 能完成角度的转换，从 DMS （度，分，秒）转换为 DEG（十进制度）的形式。当将十进制的度转换为等价的度、分、秒形式时，答案包括一个整数部分，表示度，以及一个小数部分，它的第1和第2位小数数字表示分，第3和第4位小数数字表示秒，第5位至末尾的小数数字表示秒的小数部分。要转换一个度、分、秒给出的角为十进制度的形式，必须按整数、小数的次序输入。

例1：转换十进制的16.1932°为度、分、秒形式：

DMS 16.1932 [ENTER]	16.113552

例2：转换 32°25′13″ 为十进制度的形式：

DEG 32.2513 [ENTER]

```
32.42027778
```

H．其他各种函数

ABS——绝对值函数

ABS 函数求取一个数值或变量的绝对值．

例1：ABS （25－86） [ENTER]

```
61
```

本来，25－86＝－61，ABS 函数是取两个数的差的绝对值，得到61．

INT——取整函数

INT 函数取不大于该数的最大整数．

例1：（25／3）＋7 [ENTER]

```
15.33333333
```

INT (25／3)＋7 [ENTER]

```
15
```

例2：(31.62+21.18) [ENTER]

```
52.8
```

INT (31.62+21.18) [ENTER]

```
52
```

在例2中，答案不是从52.8舍入为53，因为这样会使答案大于原来的数．

注：不可忘记求值的次序．假如你希望取结合式的整数，必须使用括号．例如，让我们改写前面的例题，不再使用括号：

例3：INT 31.62+21.18 [ENTER]

```
52.18
```

计算机对第一个数 31.62取整数（结果：31）并将它加上 21.18，得到 52.18，与第一个计算稍有不同．

SGN——符号函数

对于任何数X，符号函数 SGN 求得一个代表该数是负数、零或正数的值，这个值如

下：

$$\mathrm{SGN}\ x = \begin{cases} 1, & \text{若 } x > 0 \\ 0, & \text{若 } x = 0 \\ -1, & \text{若 } x < 0 \end{cases}$$

例：

5－10 ENTER	－5
SGN（5－10） ENTER	－1
12－4 ENTER	8
SGN（12－4） ENTER	1
15－15 ENTER	0
SGN（15－15） ENTER	0

第五章　高级程序设计

A、数组和下标变量，DIM 语句

到目前为止，我们的大部份程序例子都是用少量的变量，当你开始充分利用 PC－1500的处理能力的时候，就会发现，只能容纳一个单数的变量可能是有缺陷的。例如，也许要求一个程序读入50个数并进行排序，你便会立即断定，虽然PC－1500有很多的变量表示能力，也还是需要一个方便的方法的。这个方便的方法是有的，称为"数组变量"。

数组只不过是具有一个单名的一组连续存贮区，或称一组"单元"，每个存贮区可以存放一个单数，或者每个存贮区可以存放一个字符串。一个给定数组内的所有存贮区必须存放相同类型的数据。

单个数组内的存贮单元数目可以多到256个，个数由使用者规定，这样，假如定义一个数值数组具有50个存贮单元，这个数组就允许贮存与某一个单名相关联的50个数。假如定义一个存贮字符串的数组（称为"字符数组"），还可以规定字符串的体积，每个字符串最长可达到80个字符。有关不同体积的字符串的建立在本章B．1节叙述。

定义一个数组，要使用 DIM（维数 dimension 的缩写）语句，数组在使用之前，总必须先"说明"（定义），（不象我们用着的单值变量）。数值型数组的 DIM 语句的形式为：

DIM　数值变量名（体积）

这里，数值变量名是一个变量名，它的构成规则与前面讨论的数值变量名的构成规则一致。

体积为存贮单元数目，它必须是0至255间的某个数。注意，你为数组体积规定了一个数时，可用数组的存贮单元比规定的数多1。

合法的数组定维语句的例子：

```
DIM X(5)
DIM AA(24)
DIM Q5(0)
```

第一个语句建立一个具有6个存贮单元的数组X，第二个语句建立一个具有25个存贮单

元的数组AA，第三个语句建立一个具有一个存贮单元的数组，其实这是相当笨的，因为规定的数最小，和说明一个单值数值变量是一回事。

数组变量X与变量X对于机器来说是截然不同的，弄清这一点很重要。第一个X表示一串数值存贮单元，而第二个X表示单个不同的存贮单元。

至此，你已经懂得了如何建立数组，但你会感到疑惑不解的是如何查询每一个存贮单元？由于整组单元只有一个名字，我们查询单个存贮单元（称为元素）的方法是，在组名的后面跟着一个放在括号内的数，这个数称为"下标"，那末，比方说要把数8存贮在前面说明的数组X的第5个元素，我们写：

X（4）＝8

也许你不理解为什么下标要用4，记住元素的编号是从0开始连续数到在 DIM 语句中说明的体积数的。

数组的真正本领在于它可使用一个表达式或变量名作下标的能力。例如，要造一个包含从0到9的平方的表，我们可以写出下列语句：

```
10  DIM  SQ(9)
20  FOR  I=0  TO 9
30  SQ(I)=I*I
40  NEXT  I
```

在本例中，变量 I 被用来选择存放计算结果的存贮单元，且又被用于计算平方值。

要说明一个字符数组，使用一个稍为不同的 DIM 语句：

DIM 字符变量名（体积）*长度

这里，字符变量名遵循前面讨论的普通字符变量名的构成规则：

体积为存贮单元数目，它必须取0到255之间的数。注意，你所得到的存贮单元个数为指定数加1。

*长度是任选的项，假如使用，它规定构成数组的每一个字符串的长度。长度是1到80中的一个数，假如不使用这个项，字符串具有隐含的长度16个字符。

合法的字符数组说明的例子：

```
DIMX$(4)
DIM  NM$(10)*10
```

```
DIM  IN$(1)*80
DIM  R$(0)*26
```

第1个例子建立一个有5个字串、每个可存贮16个字符的字符数组；第二个 DIM 语句定义一个数组 NM，它有11个字串，每个字串有10个字串，这里，明确定义字串长度小于隐含长度，有助于节省存贮空间；第三个例子说明了一个有2个元素的数组，每个字符串可有80个字符，而最后一个例子说明一个有26个字符的单一字串（见本章B．1节）。

除了刚才研究的简单数组之外，PC—1500还允许二维的数组。由类推，一维数组是一个排成一个单行的数据表，而二维数组是一种具有行和列的数据表。二维数组用下列语句说明：

DIM 数值变量名（行数，列数）

或 DIM 字符变量名（行数，列数）*长度

这里：行数规定数组的行数，它必须是0到255范围内的数。注意，你可用的行数比规定的数加1。

列数规定数组的列数，它必须是0到255范围内的数。注意，你可用的列数比规定的数加1。

下图阐释了从数组说明 DIM T（2，3）得到的存贮单元和与每一存贮单元相应的下标（现由两个数组成）：

	第1列	第2列	第3列	第4列
第1行	T(0, 0)	T(0, 1)	T(0, 2)	T(0, 3)
第2行	T(1, 0)	T(1, 1)	T(1, 2)	T(1, 3)
第3行	T(2, 0)	T(2, 1)	T(2, 2)	T(2, 3)

注：二维数组能迅速地耗尽存贮空间，例如，一个有25行35列的数组就要占用875个存贮单元！

数组是功能很强的程序设计工具，为了更熟练地处理数组，我们建议进一步阅读辅助读物。

B、再谈字符串

B、1．字符串的定维

用隐含长度的方法，字符串被限制为16个字符，通过给字符串定维，可以生成长度在80个

字符以内的字符串。减少字符串的长度以节省存贮空间，也是可能的。

字符串的长度用 DIM 语句规定如下：

DIM 变量名（边界）＊长度

这里：变量名为字符串数组的名字；

边界为数组的最大下标；

长度为数组内每一字串的长度。

假如只需要一个字串，为了节省存贮空间，可以规定一个具有一个元素的数组（下标0）。这可用下面的一个定义26个字符的字串来说明：

DIM A$(0)＊26

B、2．字符串的连接

几个字符串（或几个字符变量中的字符）可以一起组成单个字串，这种字符串的"相加"称为字符串的"连接"。连接的形式为：

$$\text{变量}=\frac{\text{字符串}}{\text{字符变量}}+\frac{\text{字符串}}{\text{字符变量}}$$

例1：

```
10 S$= "SUPER"
20 T$=S$+ "MAN"
30 PRINT T$
```

将输出：

SUPERMAN

在20行中，变量 S$ 的内容（"SUPER"）"加上"字串 "MAN"。注意，连接时不插入空格。几个字符串在同一表达式中连接，如下例：

例2：

```
10 A$= "ER"
20 B$= "AND"
30 C$= "MOTH"
40 D$= "GR"
```

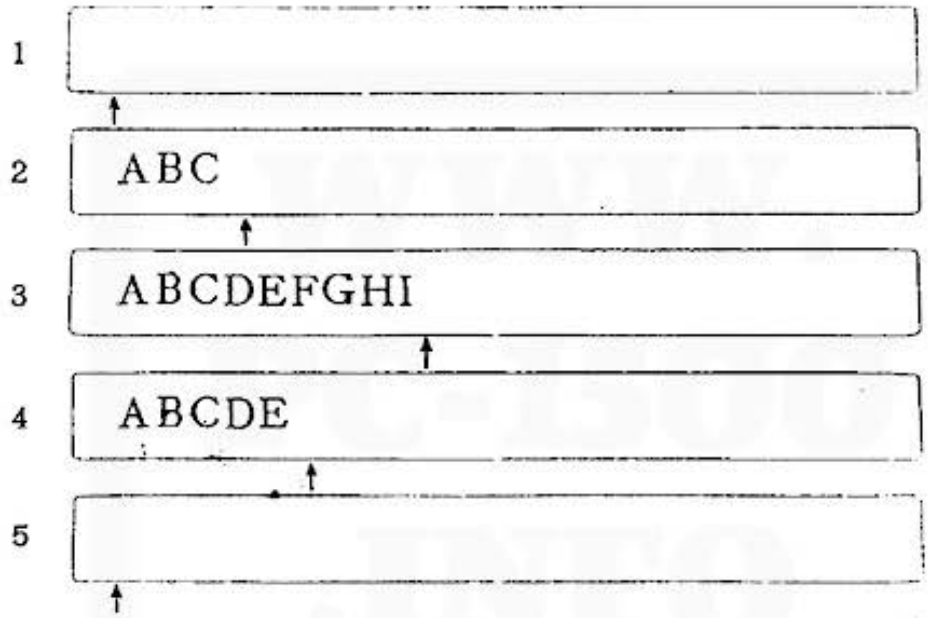
```
50 S$="WRITE YOUR " +D$+B$
60 PRINT S$+C$+A$
```

输出：

```
WRITE YOUR GRANDMOTHER
```

当执行连接操作时，用一个内部暂时字符存贮区来建立新字符串，这个存贮区的存贮能力为80个字符。假如新字符串超过这个长度，就会产生 ERROR 15的出错信息。下图说明连接操作期间这个存贮区的情况：

例3： X=LEN("ABC" +LEFT$("DEFGHI", 2)) ENTER



注：↑号代表内部字符指针，它总是记录着已用存贮空间总量。

1）开始执行时，存贮区被清除，字符指针置于开始位置；

2）"ABC" 送进存贮区，占去前面3个位置；

3）"DEFGHI" 跟着 "ABC" 加到字符存贮区；

4）LEFT$ 函数作用于字串 "DEFGHI" 抽取字符 "DE"，取代存贮区中的 "DEFGHI"；

5）执行赋值并再一次清除存贮区。

B、3．字符串的比较

字符串可以进行比较以判定哪个字串 "大于" 或 "小于" 其它字串。这些判定是基于计算机所承认的全部字符的排列顺序（见附录C）而进行的。

假如字符串包含的字符数量不相等，则较短的字符串被用空格字符（ASCII 0）填满。合法的字符串比较运算符是：

= 若两个字符串的长度相等且所含字符相同，顺序也相同，则比较结果为真；

< > 若两个字符串长度不同、字符不同或字符排列次序不同，则比较结果为真；

> 若第一个字符串的字符“大于”第二个字符串的字符（排列顺序居后），则比较结果为真；

< 若第一个字符串的字符“小于”第二个字符串的字符（排列顺序居前），则比较结果为真。

字符串比较的格式为：

字符串/字符变量 | 比较运算符 | 字符串/字符变量

这里比较运算符见上文。

例：

"MARY" > "MARI"	结果为真
"MARY" = "MARY "	结果为假
"abc" < > "ABC"	结果为真
"DATA1" < "DATA2"	结果为真
"?" < "#"	结果为假

注：象A$<=B$，A$>=B$的形式不能用于比较字符串。但象（A$<B$）OR（A$=B$）以及（A$>B$）OR（A$=B$）的形式的比较是可以的。

C、函数

C、1．ASC

有两个函数用于字符与 ASCII 码之间的互换。函数 ASC 将单一字符转换为 ASCII 十进制代码形式，其反函数 CHR$ 将 ASCII 的十进制代码转换为单一字符的字串。

ASC 函数的格式是：

ASC { "字符" / 字符变量名 }

函数的自变量是任意的字符串或字符变量名，这个函数得到的值是指定字符串的第一个字符对应的 ASCII 码。

例1：

```
10 LETX$="PATTI"
20 LET A=ASCX$
30 PRINT A
```

输出：

```
RUN
 80
```

在上例中，X$赋值为字符串"PATTI"，ASC 函数取其第一个字符 P 并将它转换为它的 ASCII 十进制代码80。

例2：

```
10 PRINT ASC "K"
```

输出：

```
RUN
 75
```

函数 ASC "K" 得到 "K" 的 ASCII 十进制代码为75。

C、2．CHR$

CHR$ 函数是 ASC 函数的反函数。CHR$ 函数将 ASCII 十进制代码（从0至127）转换为等价的字符。（注：某些代码代表特殊的字符，不能打印出来。）CHR$ 函数的一般格形式为：

$$\text{CHR\$}\begin{cases}\text{ASCII 十进制代码}\\ \text{数值变量}\end{cases}$$

例1：

```
10 PRINT(CHR$67)+"OP"
```

输出：

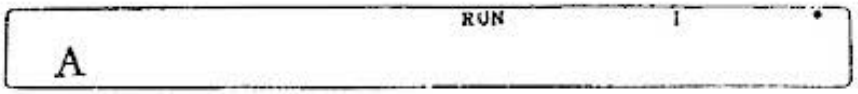
```
RUN
COP
```

例2：

```
10 Z=65
20 PRINT CHR$ Z
```

74

输出：



第一个例子说明 ASCII 十进制代码作为自变量的情形，取字符函数的结果与字符串"OP"连接成为"COP"。第二个例子中，将一个数值赋给变量 Z，然后该变量名用作 CHR$ 函数的自变量，结果得到字符"A"。

下面是一个把电文字串中的大写字母转换为小写字母的程序，ASC 函数和 CHR$ 函数两个都用上了。

例3：

```
5  WAIT  0
10 INPUT "ENTER MESSAGE", M$
20 FOR I=1 TO LEN(M$)
30 T$=MID$(M$, I, 1)
40 L=ASC(T$)
45 IF(L<65)OR(L>90)THEN 60
50 T$=CHR$(L+32)
60 PRINT T$;
70 NEXT I
80 WAIT: PRINT
```

C、3. INKEY$

这个函数从键盘接受任一字符并将它存放进指定的变量中，由于字符是自动接收的，因而不需按 ENTER 键。

变量=INKEY$

这个语句执行时，除非前面用了 PRINT 语句，否则不显示提示符号，输入的字符不传回显示中，且显示状况不受影响。

例：

```
10 WAIT  0
20 A$=INKEY$
```

```
30 IF A$=" " THEN PRINT "NO KEY" : GOTO 20
40 PRINT A$
50 GOTO 20
```

这个函数只接受一个字符，假如敲入一个以上字符，只读入第一个字符，其它字符被忽略。

C、4．LEN

在处理字符的时候，要求知道字符串的字符个数，这可用 LEN 函数来达到．这个函数可以测出指定字符串或字符变量的字符个数．

LEN { "字符串" / 字符变量名 }

例1：

```
10 A$="CATHY"
20 C=LEN A$
30 PRINT C
```

输出：

```
RUN
 5
```

例2：

```
10 C=LEN "CAT"
20 PRINT C
```

输出：

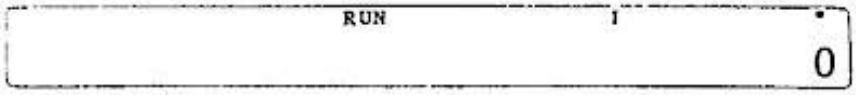
```
RUN
 3
```

假如 LEN 函数作用于空字串（即引号内没有任何字符），将得到0．

例3：

```
10 A=LEN ""
20 PRINT A
```

输出：



C、5．LEFT$

有三个函数可用于选择或抽取字符串的指定部份．LEFT$ 函数用于从字符串的左端开始抽取字符，RIGHT$ 函数用于从右端开始抽取字符，而 MID$ 函数则从中间某位置开始。

$$\text{LEFT\$}\begin{cases}(\text{“字符串”，数})\\(\text{字串变量名，数})\end{cases}$$

括号内的自变量“数”指定从字串左端开始抽取几个字符。

例1：

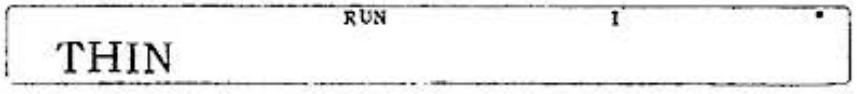
```
10 A$=LEFT$("DRESSER", 5)
20 PRINT A$
```

输出：

例2：

```
10 B$="THINK BIG"
20 A$=LEFT$(B$, 4)
30 PRINT A$
```

输出：



在上面两个例子中，都是从字符串的左端开始抽取字符并存放在 A$ 中，输出 A$ 的结果分别是“DRESS”和“THIN”．

C、6．MID$

函数 MID$ 用于抽取字符串的中间部分．

$$\text{MID\$}\begin{cases}(\text{“字符串”，表达式，表达式})\\(\text{字串变量，表达式，表达式})\end{cases}$$

例1：

```
10 A$="I NEED HELP"
20 B$=MID$(A$, 3, 4)
30 PRINT B$
```

输出：

```
RUN
NEED
```

例2：

```
10 T$=MID$("(415)743-1602", 6, 3)
20 PRINT T$
```

输出：

```
RUN
743
```

这个函数的第一个自变量是一个字符串或字符串变量；第二个自变量是一个代表所要抽取的第一个字符位置编号数；第三个自变量是所要抽取的字符总个数，包括上述第一个字符在内。在第一个例子中，"I NEED HELP" 存放在 A$ 中，MID$ 自第三个字符开始抽取4个字符，并将它们存放在 B$ 中，当输出 B$ 时，可以发现抽取的字符包括 "NEED"。在第二个例子中，第一个自变量是一个包含一个电话号码的字符串。找到第6个字符是 "7" 并同下面的2个字符一起存放在变量 T$ 中，输出结果为 "743"。

C. 7. RIGHT$

函数 RIGHT$ 的工作情况与 LEF T$ 函数极相似，唯一的差别在于它是自字符串的右端开始抽取字符，自变量与 LEF T$ 函数相同。

RIGHT$ { ("字符串", 数)
       { (字符变量, 数)

括号内的自变量"数"指定要从字符串的右边开始抽取多少个字符。

例1：

```
10 X$="READ ONLY MEMORY"
20 Y$=RIGHT$(X$, 6)
```

```
30 PRINT Y$
```

在这个程序中，函数 RIGHT$ 自字符串的右边开始抽取6个字符，并存放在变量Y$中，现 Y$的内容是"MEMORY"。

C、8. RND

有时，你也许希望你的程序能够提供随机数。RND 函数允许计算机产生从1到指定范围内的随机数（注：随机数的范围总是从1开始）。

例1：

```
10 A=RND 5
```

在这个例子中，A可能取1到5（包括5）的任何数。假如你希望随机数的范围自1除外的某个数开始（例如40到50），你就要通过产生1到10以内的随机数并加上一个常数（本例中是39）来模拟这个数。

例2：

```
10 FOR I=1 TO 5
20 B=40+RND 10
40 PRINT B;
50 NEXT I
```

输出：

```
RUN
 42  44  47  48  42
```

C、9. RANDOM

随机数由一个数学公式产生并通过使用 RND 函数来存取。无论什么时候打开计算机，机器就产生一系列的随机数。除非用了 RANDOM 函数，否则这个随机数表不受改变。这就是说，计算机每一次接通电源时，会使用相同系列的"随机数"。为了克服这一点，要用 RANDOM 函数重新设定产生随机数的数学公式所用的"种子"以产生随机数。

例1：

```
10 FOR I=1 TO 5
15 A=RND 3
20 PRINT A;
30 NEXT I
```

例2：

```
10 FOR I=1 TO 5
15 A=RND 3
20 PRINT A;
30 NEXT I
```

输出：1 2 2 3 1

输出：1 2 2 3 1

在这种情形，为了得到真正的随机数，应在 RND 语句之前出现 RANDOM 函数。这个函数播下一个新的产生随机数的种子，这样，使得产生的随机数不同。因而，运行在相同条件下的一个程序将产生不同的输出：

例1：

```
10 RANDOM
15 FOR I=1 TO 5
20 A=RND 3
30 PRINT A;
40 NEXT I
```

输出：3 1 2 2 3

例2：

```
10 RANDOM
15 FOR I=1 TO 5
20 A=RND 3
30 PRINT A;
40 NEXT I
```

输出：2 3 1 3 2

C、10. STR $

STR$ 的作用与 VAL 相反，它将一个内部数值变量转换为字符串表示形式。

例：

```
10 INPUT "ENTER A NUMBER" ; I
20 S$=STR$ (I)
30 PAUSE " THAT NUMBER IS"
40 PRINT "THE STRING" ; S$
```

上述程序接收一个数，形成这个数的字符表示式并将它存放在变量 S$ 中。由于内部的数值表示式不能显示出来，故自动地将它转换回为一个字符串并 PRINT 语句输出。

C、11. STATUS

要显示你留下多少个可用程序步或程序已用去多少程序步，使用 STATUS 函数。STATUS 0 将显示仍有多少个可用的程序步，STATUS 1 将显示已用程序步数。可用程序步的最大数目为1850 。（译注：用4K存贮模块时为 5496，用8K模块时为 10042 。）

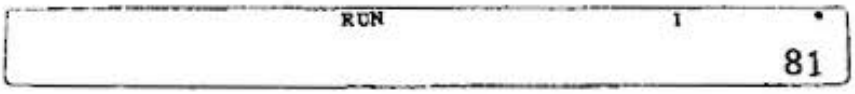
例：STATUS 0

输出：

```
RUN                    I
                    1700
```

STATUS 1

输出：



指令 STATUS 0等价于 PC-1211 机的 MEM 命令，MEM 命令在PC-1500 机上仍然有效。

C. 12. TIME

要显示或调定月份、日期和时刻，可按如下方式使用 TIME 函数实现：

调定时间：TIME=MMDDHH. mmss ENTER

显示时间：TIME ENTER

这里，MM 表示两位数字的月份，DD 表示两位数字的日期，HH 表示两位数字的小时；小数部份 mm 和 ss 确定分和秒。

当显示时间时，输出也是采用刚才给出的格式。函数的结果能按与数值相同的方式进行处理并能在表达式中自由使用。

下面是一个模拟时钟的程序，在程序运行前务需调定时间。

程序单：

```
10 WAIT 0
20 A$=STR$ TIME
30 IF TIME>99999 THEN 50
40 A$="0"+A$
50 M$=LEFT$(A$, 2)
60 D$=MID$(A$, 3, 2)
70 H$=MID$(A$, 5, 2)
80 DS$=M$+"/"+D$+"/82"
90 DS=VAL(M$+D$+"00")
100 PRINT DS$;
110 T=TIME-DS
120 IF T>=1 THEN 140
130 T=T+12
```

```
140  IF  T>23.5959  GOTO  20
150  CURSOR  18:PRINT  USING "####.####";T
160  GOTO  110
```

C. 13. VAL

VAL和STR$互为反函数，它们将字符串转换为数值变量或反之。函数VAL将包含代表一个数的字符串转换为一个数，并将它存放在一个变量中。

注：除数字0和9，．（小数点），＋（正号），－（负号）或E（科学记数法）可用于VAL函数的表达式外，其它的任何字符会使转换以非法的字符结束。

例1：

```
10  ZS=VAL "-37"
```

结果：ZS包含数-37。

例2：

```
10  A=VAL "237.6"
```

结果：A包含数237.6

D. PRINT USING

USING 语句允许程序员严格地控制输出信息的格式，它容许标准化输出并防止丢失输出信息。

当 USING 子句单独出现或在PRINT 语句或 PAUSE 语句中出现时，它规定后继的PRINT 语句或 PAUSE 语句的输出格式，直至遇到下一个 USING子句。

在单一个PRINT语句或PAUSE语句中，可以出现几个USING子句，在这种情形，每一个USING子句规定用来输出变量表中变量的格式，直至遇到下一个USING子句。

输出格式是通过称为“编辑字串”的专用字符来规定的。编辑字串中的字符定义 输出信息可利用的显示区域以及限制在这些显示区中可以输出的信息类型。这个方案与其它语言诸如COBOL和PL/I使用的方案一样是通用的。

编辑字串可以存贮在一个字串变量中，然后，该变量名就可代替USING 子句里面的编辑字串，它使得能在程序控制下选用多种输出格式。

编辑字串中可使用的字符归纳如下：

编辑字符一览表

字 符	用 途
#	规定一个数字域，在这个数字域中，数实行右对齐。假如数字域的宽度不足以容纳一个数，则产生出错信息ERROR36。先行零转换为空格输出。
*	规定在数字域中不含有数据的指定位置上用星号（*）填充。
.	使在一个数字域内显示小数点。
,	用在数字域的前部以规定从小数点算起每三位数字前面插入一个逗号（，）。
↑	用于数字域内使输出的数按科学记数形式表示。
+	用在数字域中输出数据的符号。
&	规定一个字符域，在这个字符域内，字符实行左对齐，若域宽不足以容纳字符串，则字符串被截尾。

注：1）数字域的宽度必须总是比数据宽度大1，使能表示数据的符号。它与是否使用+号编辑字符无关。

2）使用逗号（，）号时要求在编辑字符串中多插入一个#号。

例子：

```
X=PI  Y=1234  A$="ABCDEF"
PRINT USING "###" ; X
                                  3
PRINT USING "+####.###" ; X
                               +3.141
PRINT USING "###.##↑" ; X
                            3.14E 00
PRINT USING "###.↑" ; X
                             3.E 00
```

```
PRINT USING "*######" ; Y
```

```
                **1234
```

```
PRINT USING "*****" ; Y
```

```
                1234
```

```
PRINT USING "&&&&&&########" ; A$; Y
```

```
ABCDEF          1234
```

```
PRINT USING "&&&" ; A$
```

```
ABC
```

```
10 U$="*######.##"
20 USING U$
30 PRINT Y; "$"
```

```
**1234.00$
```

```
PRINT X; "$"
```

```
*****3.14$
```

```
PRINT USING; A$; X
```

```
ABCDEF          3.141592654
```

```
PRINT USING "###, ###, ###" ; 246813
```

```
                246,813
```

注：对于变量（整数）使用#号（包括*号）的个数规定在下列范围：

不用带3位数字为一组的标点（，）时，在11个以内（包括符号）；

使用3位数字为一组的标点（，）时，在14个以内（包括符号）。

这台计算机能处理10个有效数字的数，当用USING语句指定的格式超过10位整数数字时，由PRINT（或LPRINT）语句显示（或打印）出超过10个整数的数字，但显示（或打印）的数可能是不正确的。

例：在RUN方式：　　显　示

USING "＃＃＃＃＃＃＃＃＃＃＃＃＃＃" ENTER ——→ 〉

PRINT 888888888888 ENTER ——————→ 888888888800

LPRINT 888888888888 ENTER ——————→ 88888888880

E• 计算控制转移

除第三章所述的基本控制语句之外，PC—1500还提供了两种大有用处的控制语句，这就是ON GOSUB语句及ON GOTO语句。从它们的名称，你会猜想到，这两个语句起的作用象前面讨论的GOSUB和GOTO语句一样。但不同的是在于它们自动进行转移控制的能力（即执行不同地方的语句的能力）。这就是说，GOTO或GOSUB的功能是根据数值变量的值转到几个语句之一去执行，而这两个语句所依赖的控制变量是在ON语句中给出，因而管它们叫"计算控制语句"。

ON语句的形式如下：

$$\text{ON}\langle\text{表达式}\rangle\begin{cases}\text{GOTO}\langle\text{行号}\#1\rangle,\langle\text{行号}\#2\rangle,\langle\text{行号}\#3\rangle,\cdots(\text{等等})\\ \text{GOSUB}\quad\text{〃}\quad\text{〃}\quad\text{〃}\quad\text{〃}\quad\text{〃}\end{cases}$$

关键字ON之后的表达式必须求值为一个大于零且小于或等于关键字GOTO或GOSUB之后所列行号的个数的正整数。程序执行时，当计算机遇到一个ON语句，就将程序执行流程移到序号与表达式的值对应的那个行号上去。

典型的ON语句可以是：

ON TX GOSUB 100, 200, 250, 300

在这种情形，由于只列出4个行号，变量TX的值必须取1到4的范围内的数。TX的任何其他取值将导致出错，因为没有与其对应的行号。正是由于这个缘故，在程序中包含充分性检验（IF语句）以确保表达式的取值是一个有效的数，这一点是重要的。

ON语句对于自动进行一系列的选择方面是很有用的。例如，考察下面一个允许用户选

择几个税表之一的程序段，不用ON语句时，可以写成：

```
10 PAUSE "SPECIFY TAX TABLE TO USE: "
20 PAUSE "(1)SINGLE, "
30 PAUSE "(2)MARRIED, "
40 INPUT "(3)BUSINESS?" : TT
50 IF (TT<1)OR(TT>3)THEN 10
60 REM USE APPROPRIATE TABLE
70 IF TT=1 THEN 220
80 IF TT=2 THEN 300
90 IF TT=3 THEN 450
   (等等)
```

使用 ON 语句时，我们可以把70行、80行和90行合并成为单一语句：

```
70 ON TT GOTO 220, 300, 450
```

ON ERROR GOTO语句

使用另一种形式的控制转移语句，程序能检测出何时发生了错误，检出之后，程序就可以执行发现错误的语句，该语句会报告并指示使用者，或者，保存重要的数据。

ON ERROR GOTO 语句指示机器，检出错误之后到什么地方去执行程序，这个语句的形式是：

ON ERROR GOTO〈行号〉

这里，行号为包含有发生错误时要遵照执行的指令的程序行。

F• 显示程序设计

装入PC—1500的显示窗是一种使用相当灵活的显示装置。为了使程序员能发挥显示器的全部能力，PC—1500使用的BASIC语言中增加了几个新的语句，本节就叙述这些扩展语句。

显示器本身是利用液晶技术，能够一次显示26个以内的字符，计算机字符集中的每一个字符占据一个5×7点阵。利用GPRINT命令，程序员还可发展和显示他自己的字符。

为了能够绘图，整个显示区域可以用作一个7×156的点阵。156列内的各个点都可以用来构造图形、数字或特殊符号。 POINT 命令能够对任一列进行检测，揭示当前用了哪些点。

一个扬声器及乐音发生器使程序员能给人—机对话增加一个一定范围的声音。可以产生256个频率内的任一个乐音（频率范围230赫至大约7千赫）。在程序控制下，也可以自动重复一个乐音和控制乐音的持续时间。

F.1 BEEP

BEEP 语句使程序员能产生游戏乐音，出错信号，以及其他对话方面的应用。产生声音方面的BEEP语句的格式为：

BEEP 〈表达式1〉,〈表达式2〉,〈表达式3〉

这里，表达式1是唯一需要的参数，它规定嘟嘟声重复响的次数，重复次数的允许范围是0至65535；

表达式2是任选的，它规定乐音的频率，这个数的范围在0至255之间；

表达式3也是任选的，它规定每一次乐音的持续时间，这个持续时间用0至65279范围内的数来规定。

BEEP 语句也可用于接通或关闭 PC—1500 的内部扬声器，这样，便可免除由录音机SAVE（保存）或LOAD（装入）操作引起的噪声。

控制内部扬声器的BEEP语句的格式为：

BEEP OFF

或BEEP ON

注：当PC—1500的电源关闭而后接通时，扬声器的状态恢复到工作方式。

示范程序：

```
10  D=60
20  DATA 14
30  DATA 245, 1, 245, 1, 160, 1, 160, 1
40  DATA 143, 1, 143, 1, 160, 2
50  DATA 180, 1, 180, 1, 195, 1, 195, 1
60  DATA 220, 1, 220, 1, 245, 2
```

```
100 READ X
110 FOR I=1 TO X
120 READ N, S
130 BEEP 1, N, (D•S)
140 NEXTI
150 END
```

F.2 CURSOR

CURSOR 语句把光标移到显示器中26个可用字符位置的某个字符位置。这个语句的形式为：

CURSOR〈位置表达式〉

这里，位置表达式求值为0至25范围内的某个数，它规定光标要移到的字符位置。

CURSOR 命令的一般用途是确定在输出某些信息之前光标的位置，用这个方法，程序员就能自行规定输出数据项之间的间隙，如下例所示：

程序单

```
10  WAIT 20
20  X=8
30  PRINT "A"
40  CURSOR 4
50  PRINT "B"
60  CURSOR X
70  PRINT "C"
80  CURSOR (X^2/4)-4
90  PRINT "D"
100 END
```

本程序会使字母A，B，C和D分别出现在显示屏上的第0，4，8和12号字符位置：

```
                                    RUN
A       B       C       D
```

注：规定的光标位置号大于25或小于0会产生出错信息 ERROR 19。

示范程序

```
10 WAIT 0
20 DIM A$(0)*13
30 INPUT "ENTER YOUR NAME", A$(0)
40 C=0 : CLS
50 FOR I=1 TO (LEN A$(0)-1)
60 CURSOR C
70 PRINT MID$ (A$(0), I, 1)
80 C=C+2
90 NEXT I
100 WAIT
110 CURSOR C
120 PRINT RIGHT$ (A$(0),1)
130 END
```

F.3 CLS（清除显示屏）

CLS 命令通过关闭显示窗上所有的点而清除显示屏，它用在其他语句前面，从显示器中清除任何残留的数据．光标被 CLS 命令重新定位到显示屏的左端．这个命令的格式很简单，为CLS.

示范程序

```
10 GPRINT "7F7F7F7F7F"
20 CLS
30 PRINT "NEW INFO"
```

F.4 GCURSOR

GCURSOR 语句规定显示屏中可利用的156个点列中的一个点列作为后继显示的信息的开始列．GCURSOR 指令的格式为：

GCURSOR < 位置表达式 >

这里，位置表达式求值为一个 0 ～155范围内的数，这个数规定显示屏上156个 7 点列之

一。

注：若位置值是一个小于0或大于155的数，会产生出错信息ERROR 19

GCURSOR语句常同GPRINT语句一起被用于建立图形显示（这一点将在下节较详尽地说明）。其他类型的语句可以跟在GCURSOR语句之后，因为这个语句不写出任何信息，GCURSOR语句只是指示后继的输出信息将自哪一列开始写起。下面的程序说明了这一点。

程序单

```
10 GCURSOR 50
20 PRINT "A"
30 GCURSOR 80
40 PRINT 26/3
```

在位置50列和80列上产生的输出看来是正常的：

```
RUN
A                    8.666666667
```

假如将30行的开始输出位置数改为93，会发生什么情况？我们用下面的行代替原来的30行试试看：

```
30 GCURSOR 93
```

想不到！第二个输出内容被截断了，因为越出了显示屏的末端。

作为GCURSOR语句的最后一个例子，我们给出一个高级的程序，它通过重叠字符产生不寻常的效果。

程序单

```
10 WAIT 0
20 DIM A$(0)*10
30 A$(0)="" :C=0
40 INPUT "ENTER MESSAGE(<10 CHARS)", A$(0)
50 CLS
90
```

```
60 FOR I=1 TO (LEN A$(0))
70 GCURSOR C
80 FOR J=1 TO 3
90 GCURSOR C
100 PRINT MID$(A$(0), I, 1)
105 C=C+3
110 NEXT J
120 C=C+5
130 NEXT I
140 WAIT
150 GCURSOR 155
160 GPRINT "0 0"
170 END
```

F.5.GPRINT

GPRINT 语句对显示窗上的点提供直接的、可编程序的控制，由于GPRINT 语句起着设置和清除任一七点列内的点的作用，它通常是与 GCURSOR 指令连用。GCURSOR 语句是通过选择适当的列调整输出位置，而GPRINT 语句是控制列为内的点。GPRINT 语句也能用单一语句输出几个相邻列的信息。

为了弄清楚 GPRINT 语句的格式，需要弄清列内的点是如何控制的。在一列中被激活的点的图案可用十进制的数或十六进制的数字串来规定。假如用十进制数，则每一行可看作是从上到下用 2 的乘方进行编号，这可说明如下：

```
1  ---------
2  ---------
4  ---------
8  ---------
16 ---------
32 ---------
64 ---------
```

由于一列有 7 个点，每一个点可以激活也可以清除，故会有 128 种可能的点的图案。这样，要规定一个特殊的图案，人们可以用下面的格式：

GPRINT〈图案表达式1〉;〈图案表达式2〉;……(等)

这里，图案表达式求值应是0至127间的一个数，它规定激活点的图案。可以任选规定几个图案表达式，每个表达式之间可用分号或逗号分隔，假如使用逗号，则在每一个输出列之间留出一个空列。

让我们通过构造一个新的字符上箭号的例子说明 GPRINT 语句的利用：

```
1  ----*----
2  --*-*-*--
4  *---*---*
8  ----*----
16 ----*----
32 ----*----
64 ----*----
   1  2  3  4  5
```

因为我们的字符有5列宽，因而我们在GPRINT语句的表达式表中需要5个不同的数。代表第1列和第5列的数每一个必须规定标号为4的行上的一个点，类似地，代表第2列和第4列的数每一个必须规定标号为2的行上的一个点，最后，语句为：

```
GPRINT 4; 2; 127; 2; 4
```

第三列规定的是一个独特的数127，其由来也许不能立刻看出。数127是第一行(1)、第二行(2)、第三行(4)等等各点编号的总和，这样，1+2+4+8+16+32+64=127规定了一列中全部的7点。通过规定一行或几行的编号之和就可以构造任何图案。

注意。

观察下面的例子，当把程序中的PRINT命令改为GPRINT命令时的做法如下：

```
        ↓------完全把 "PRINT" 改写为 "GPRINT"
20  PRINT  A$
    ↑------仅在此处插入 "G" 不能使计算机识别为 "GPRINT"
           (这样会被认为是 "G" 和 "PRINT" )
```

当把CURSOR命令改为GCURSOR命令时或者对于打印机命令，同样适用这个说明。

假如使用十六进制定址结构的话，显示屏的7行在概念上被划分为下组的3行和上组的4行。每一组中从它顶上的行开始按2的幂编号，说明如下：

这样，用单一个十六进制数字就能表示一个组中的所有图案，因为下组只有3行，故本组允许的数字范围就从0至7。要规定一个整列需要2个十六进制数字，第一个数字代表下组，第二个数字代表上组。

十六进制的GPRINT语句的格式为：

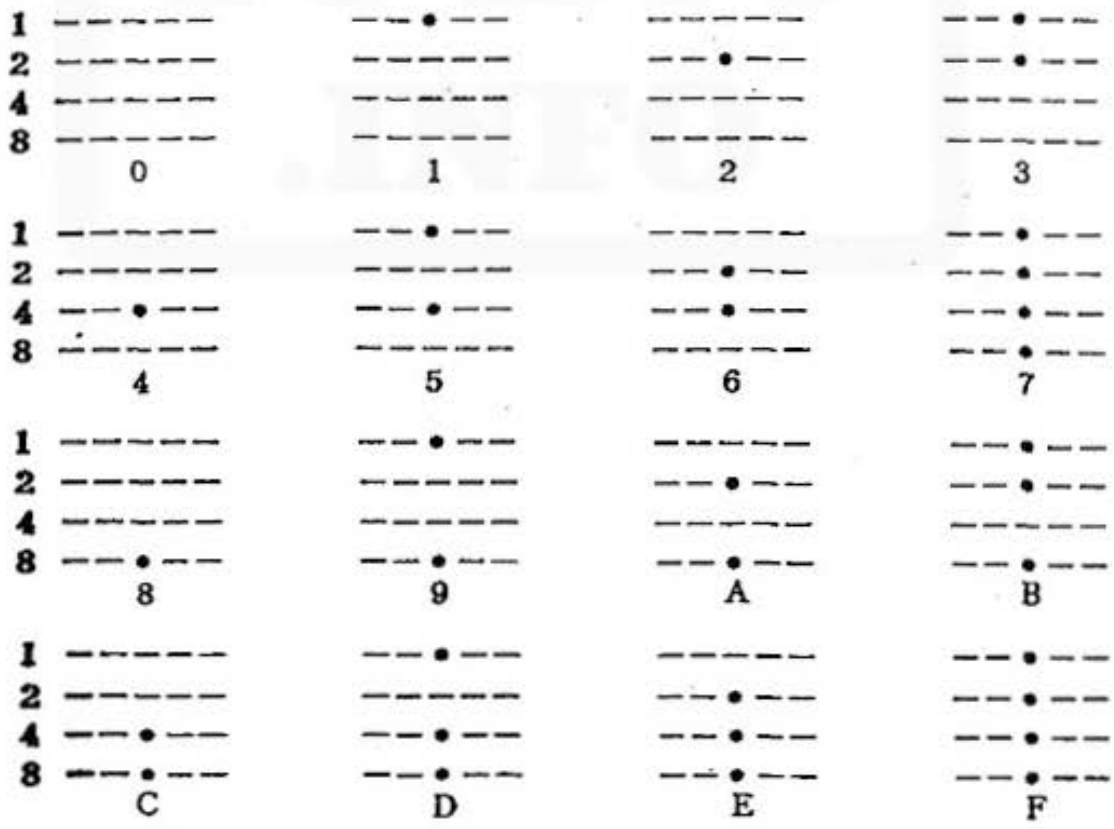
GPRINT "十六进制数字串"

这里，十六进制数字串是一个由十六进制数字组成的字串，每一对十六进制数字规定一个单列的点的图案。

使用这个格式来构造上例的上箭号图案，我们写出语句：

GPRINT "0 4 0 2 7 F 0 2 0 4"

十六进制字符表



示范程序：

这个程序依次输出从0至127所有可能的点的图案。

程序单

```
10  WAIT 0 : CLS
20  FOR I= 0  TO 127
30  GCURSOR I
40  GPRINT I
50  NEXT I
60  WAIT
70  GCURSOR 155 : GPRINT "00"
80  END
```

F.6 POINT

POINT函数求得的结果是一个数，它表示给定列中被激活的点的图案。这样，POINT函数使得能在程序的控制下"检测"显示屏中任一列的图案情形。

POINT函数的格式为：

POINT 〈位置表达式〉

这里，位置表达式的值为0至155范围内的一个数，它代表所要检测的列。

POINT函数求得的值是一个0至127范围内的数。它是2的各次幂值之和，其解释如GPRINT节所述。

作为一个说明，假设显示屏上有一个大写字母I位于40列到44列：

```
1     ----*--*--*----
2     --------*--------
4     --------*--------
8     --------*--------
16    --------*--------
32    --------*--------
64    ----*--*--*----
      4  4  4  4  4
      0  1  2  3  4
```

表达式：

POINT 40 的值为0。

POINT 41 的值为65。

POINT 42 的值为127。

示范程序

这里所列的程序能够用贮存在变量 A$ 中的字符来充填显示屏以及通过改变这个字符构造奇异的图案。这个程序利用了本章讨论的几个语句。

程序单

```
10   A$ = "X"
20   WAIT 0
30   Y=5 : X=155/Y : C=0
40   FOR I=1 TO X
50   GCURSOR C
60   PRINT A$
70   C=C+Y
80   NEXT I
90   FOR I=0 TO 155
100  GCURSOR I
110  A=127-POINT I
120  GPRINT A
130  NEXT I
140  GOTO 90
```

注：假如程序有最后这一行，执行时就会处于无限循环之中（这可用 BREAK 键来停止）。此外，显示的字符可以通过在第10行的赋值语句中插入新的字符来改变。

G·跟踪调试

无论你是如何地仔细，最后建立的程序往往还是不那么理想，为了剖析问题的所在，SHARP 计算机的设计者们为我们提供了一种执行程序的特殊方法，称为"跟踪"方式。在跟踪方式，PC—1500会显示每一程序行的行号，并在执行完该行之后就停止下来。这就使你能够跟随（或跟踪）指令的实际执行顺序。当程序执行完了一行暂停的时候，你便可检查

或改变变量的值。

进入跟踪方式的指令形式很简单，即，TRON。TRON指令既可以在运行方式作为命令发出，也可以作为一个语句嵌入程序中。作为命令使用时，TRON命令告诉计算机，在执行后继的全部程序时要求执行跟踪。然后，要跟踪的程序就可按正常的方法，用GOTO命令或RUN命令开始运行。

假如TRON作为一个语句使用，就只当含有这个语句的行执行时才进入跟踪方式。假如由于某种原因，终不能达到这一行，则跟踪方式仍是不起作用。

一旦进入跟踪方式，就一直保持这个运行方式，直到被TROFF指令所撤消。TROFF指令同样既可用作一个命令也可用作一个语句发出。此外，跟踪方式也可使用键串SHIFT CL来撤消。

作为一个使用跟踪方式的例子，送入下面一个给定直角三角形两边的长，计算其斜边的长的程序：

程序单：

```
10    INPUT A, B
20    A=A*A:B=B*B
30    H=√(A+B)
40    PRINT "HYPOTENUSE=  ";H
```

在运行方式，发出TRON命令，跟着发出RUN命令。注意到INPUT命令仍是按通常的方法执行，对每一个要求输入的值显示一个问号。一旦你输入了两个值，就出现INPUT语句的行号：

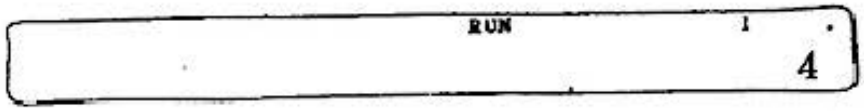
```
RUN
10:
```

按下上箭头键（↑），并按住它，你便可以观察到整一行：

```
RUN
10  INPUT A,B
```

要继续执行程序，可按下 ⊟（下箭头键）一次，它使机器执行下一行并显示该行行号，再而，你又可用上箭头键观察这整一行。还可通过键入变量为名字并按 ENTER 键来检查任一变量的内容，如：

```
A  ENTER      （这里A为程序的变量名）
```



每执行一行，都要按下箭头键 ↓ 一次，直至程序执行完毕。假如你不希望正规地一行一行执行程序，可按 ENTER 键中止程序的执行。假如你又改变主意，中止的程序可用 CONT 命令继续执行。

一个使用上例计算直角三角形斜边的程序的操作样本如下：

键　　串	显　　示
	>
[T][R][O][N]	TRON_
[ENTER]	>
[R][U][N]	RUN_
[ENTER]	?
[3]	3_
[ENTER]	?
[4]	4_
[ENTER]	10:
[↑]	10: INPUT A, B_
[↓]	20:
[↑]	20: A=A*A: B=B*B_
[A]	A_
[ENTER]	9
[B]	B_
[ENTER]	16
[↓]	30:
[H]	H_
[ENTER]	5
[↓]	HYPOTENUSE= 5
[↑]	40: PRINT "HYPOTENUSE= "; H_
[↓]	40:
[↓]	>

H·十六进制数与布尔函数

H．1．十六进制数

PC—1500 机具有在任何表达式中能够使用十进制数的地方使用十六进制数（以十六为底的数）的能力．在使用上与十进制数的区别在于，十六进制数的前面带有&号．下列为有效的十六进制数的例子：

&16　　&F　　&7E CA　　&08　　&99A　　&—5B

十六进制数可用于手动计算中，如：

10+&A　ENTER

也可用于程序中，如，

程序：

```
35    GPRINT  &F, 54, &3E
40    DATA    67, &7F, &2B  12, 305
```

H．2．AND函数

AND 函数能够对内部表示的两个数值进行布尔“与”运算，这两个值必须在—32768到32767的范围，超过这个范围的数将产生出错信息 ERROR 19．

例	运算结果
10 AND &F	10
1 AND 0	0
—1 AND 1	1
55 AND 64	0
16 AND 63	16

H．3．OR函数

OR函数对内部表示的两个数值执行布尔“或”运算，这两个值必须在—32768到32767的范围，超过这个范围的数将产生出错信息 ERROR 19．

例			运算结果
10	OR	&F	15
1	OR	0	1
−1	OR	1	−1
55	OR	64	119
16	OR	63	63

H. 4. NOT函数

NOT 函数对内部表示的单个数值进行布尔“非”或“补”运算 这个值必须在−32768到32767的范围，假如这个值超过这个范围，会产生出错信息 ERROR 19。

例			运算结果
NOT	0		−1
NOT	&F		−16
NOT	55		−56
NOT	1		−2
NOT	−2		1

I. 程序执行暂停

STOP, CONT

STOP语句能使计算机中止程序的执行，当程序的执行暂停时，全部变量的值能够保留下来，程序员可以检查和改变这些变量的值。而后，可以用 CONT 命令在暂停的点上继续执行程序。

计算机在执行过程中遇到 STOP 语句时，会显示出类似于下列内容的信息：

```
                    RUN                I
BREAK  IN  60
```

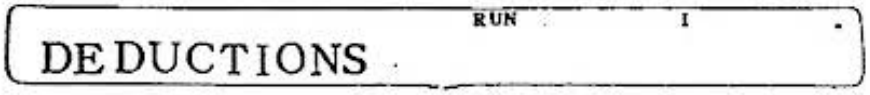
这里，60是一个包含 STOP 语句的行号。

假如你希望回过头来观察这一行，可按住上箭头键↑。

当出现BREAK信息时，还可以检查和改变变量的值。例如：

HQ [ENTER]

A$ [ENTER]



无论何时你准备重新开始执行程序的话，只要使显示回到显示提示箱（>）的状态并敲入CONT [ENTER] 即可。

J．方式控制

LOCK，UNLOCK

LOCK指令可用于控制计算机操作的方式（RUN，PRO或RESERVE方式），把它写进程序中，能防止使用者偶然改变工作方式和破坏程序．LOCK指令把计算机"锁定"在当前所处的方式，使MODE键不起作用。

为了起用MODE键，要用UNLOCK指令，这个指令使MODE键恢复正常的功能，使计算机能改变它的工作方式。

这两个指令既可以用作命令，也可以用作语句，其形式很简单，为：

LOCK

UNLOCK

第六章　PC—1500　功能的扩展

A．印刷机／盒式磁带录音机接口（CE—150）

印刷机／盒式磁带录音机接口是 SHARP　PC—1500　袖珍计算机的备用装置，这台装置可以和一台或两台盒式磁带录音机连接。磁带录音机可用于把程序和数据贮存在标准录音磁带上。日后要用时，又可把程序和数据"装回"到 PC—1500 机的内存贮器中，省去了再次敲键输入它们之烦。

A．1．计算机与接口的连接

计算机／录音机接口 CE—150 与计算机的连接按下述方法进行：

（1）关闭计算机的电源开关。

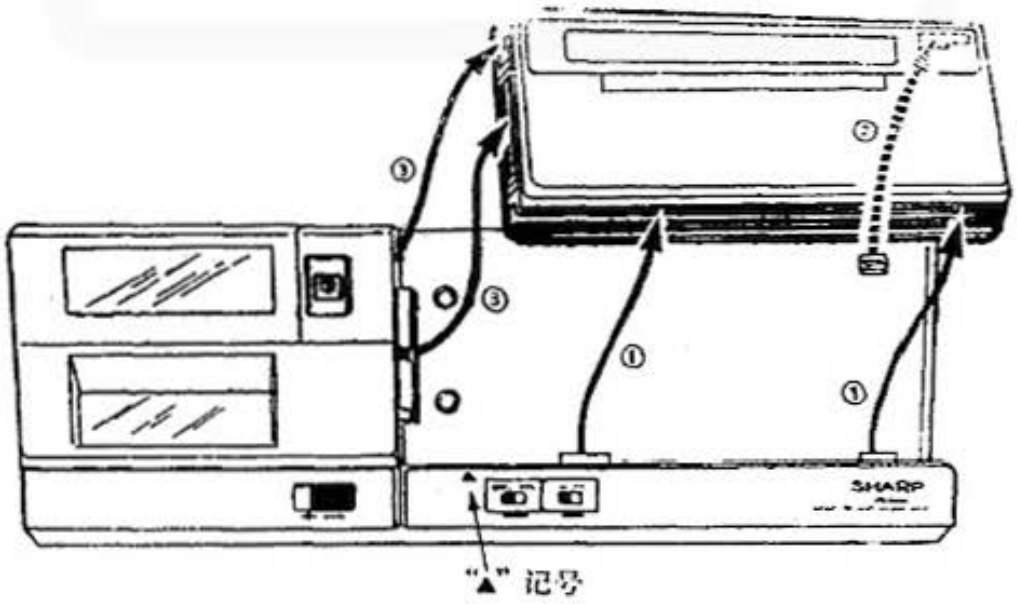
十分注意！关闭电源开关是必要的。假如电源接通，计算机会"挂起来"（全部键不能操作）.如果发生这种情况，可在按着 ON 键的同时，按下计算机背后的 ALL RESET 开关。

（2）从计算机的左边揭出插口保护罩并将它扣进印刷机底部的空位（图略）。

（3）把计算机的底面放进支架，使印刷机的导轨与计算机的导槽套装在一起。

（4）把计算机放平。

（5）让计算机慢慢地滑到支架的左边，使印刷机的插头插入计算机的插座内（见图2）。不要强行将计算机与印刷机接合在一起，假如不容易接合，可细心地左右移动计算机到正确的接合面上。



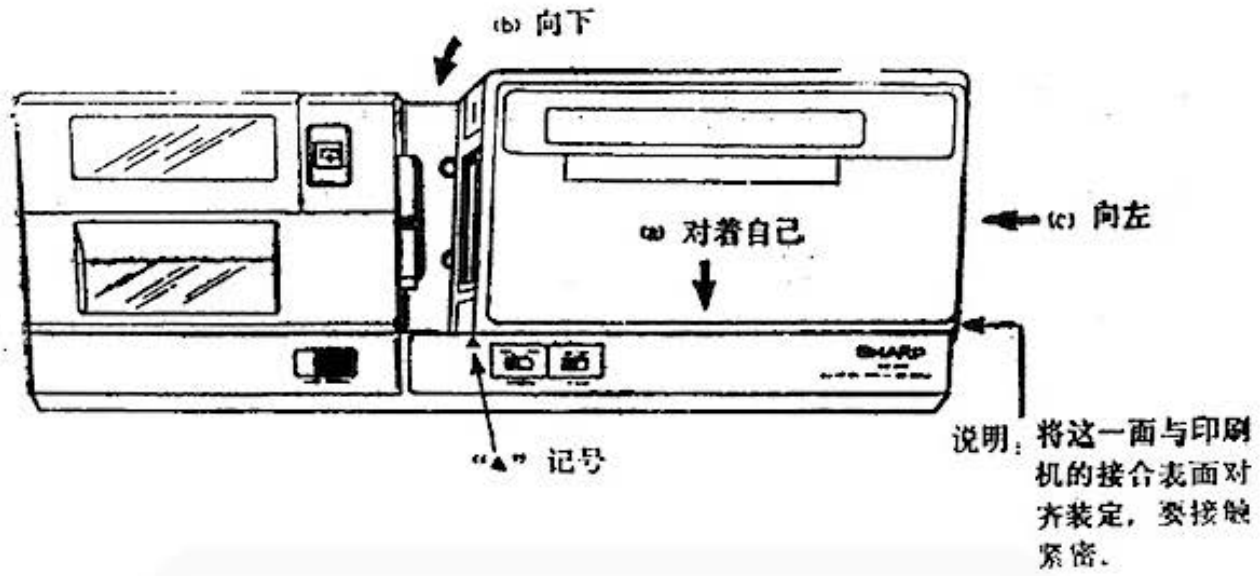


图2、计算机与CE—150接口的连接

A．2．电源

CE—150 连接器使用可重新充电的镍—镉蓄电池作为电源，因而，在电能行将耗尽，显示下面的信息之后就应该给蓄电池重新充电。（在充电时，印刷机被关闭起来，要开启印刷机，可在充电之后依次按计算机的OFF键和ON键。）

失电时显示的信息：

（1）ERROR 80　或　ERROR 78

注：当印刷机处于换笔状态时，会出现ERROR 78的信息。

（2）：CHECK 6　或　NEW 0？：CHECK 6

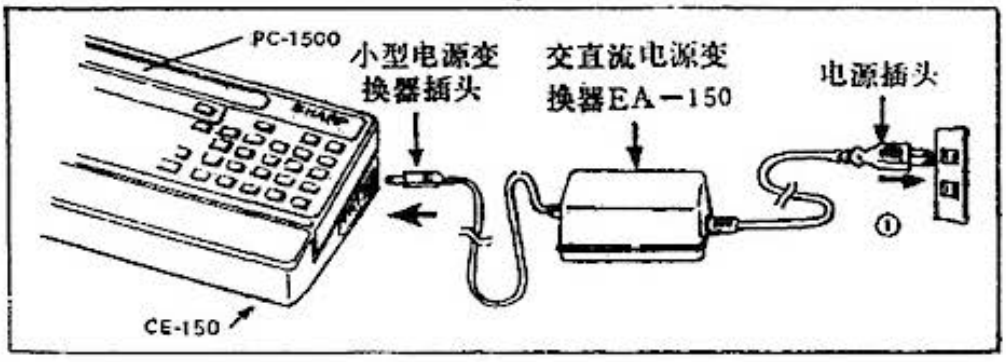


图3、如何连接交直流电源变换器

A。3。磁带录音机与接口的连接

先将 CE—150 连接器与计算机连接起来，再按下图把磁带录音机与 CE—150 连接器连接。

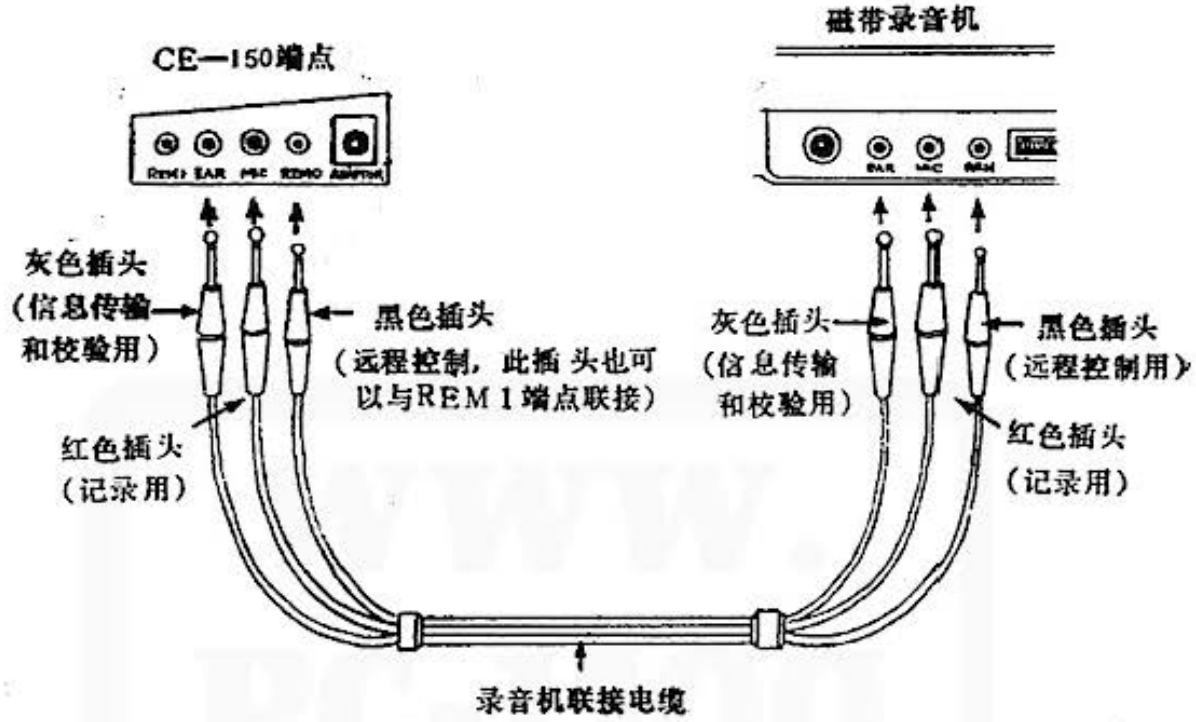


图 4，磁带录音机与CE—150接口的连接

下表叙述与 CE—150 接口连接的磁带录音机必需具备的最低限度技术要求：

项　　目	要　　求
1．录音机类型	符合下述要求的任何磁带录音机：盒式录音机、微型盒式录音机或开盘式录音机都可使用
2．输入插孔	录音机应有一个标记为“MIC”的小型插孔输入端，但决不能使用标记为“AUX”的插孔。
3．输入阻抗	输入插孔应为低阻抗输入（200～1000欧姆）
4．最低输入电平	3毫伏或－50分贝以下
5．输出插孔	应是一个小型插孔，标记为“EXT”（外部扬声器插孔），“MONITOR”，“EAR”（耳机）或其他类似插孔。
6．输出阻抗	应在10欧姆以下
7．输出电平	应为1伏以上（实际最大输出在100毫瓦以上）
8．失真	在2千赫至4千赫范围内不大于15%
9．速度不均匀性（晃抖度）	最大0.3%（W. R. M. S）
10．其他	录音机马达速度不能波动

注：（1）在 CE—150 提供的小型插头不适合你的磁带录音机的输入/输出插孔的情况，可利用市面上的专门线路转换插头。

（2）某些磁带录音机由于规格指标不同，可能不适合连接，或某些录音机在长年使用后失真、噪声增加或电源品质变坏，可能会由于电性能的变化导致不能得到满意的结果。

（3）磁带录音机使用注意事项：

1）进行信息的传输或校验时，要使用原先用来记录的那种录音机，假如作传输或校验时用的录音机与用来记录的不同，就可能不能进行传输或校验。

2）假如磁带录音机的磁头脏了，会增加失真或降低记录电平，故应保持磁头清洁。

3）要使用那种频率响应不是太低、无划伤和皱折的磁带。

A．4．纸带的装入

关于纸带的装入，详见 CE—150 使用手册。

装纸带的操作步骤是，

（1）打开印刷机的盒盖，方法是，沿箭头方向推移印刷机盒盖锁杆，图 5．

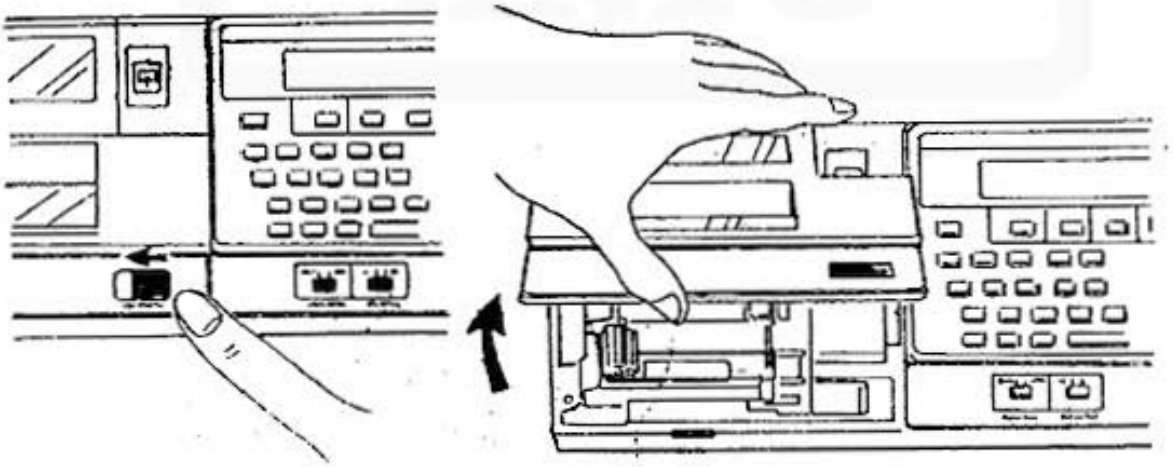


图 5

（2） 把纸头裁剪平整，将打印纸正确地插入进纸缝隙（纸头弯曲或皱折会防碍纸的插入），图6。

图6

（3） 按计算机的 ON 键，接通计算机电源，并按打印机的 [↑] 键以便把纸送过去，此时打印纸的传动机构便把纸卷过，使纸头高出印刷机3至5公分，图7。

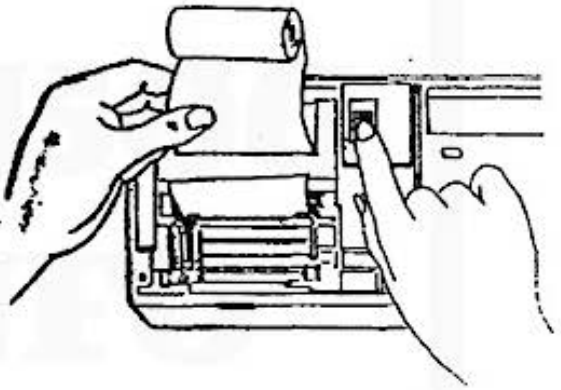


图7

（4） 将打印纸的传动轴插入并卷内皱把纸卷装入纸盒，图8。

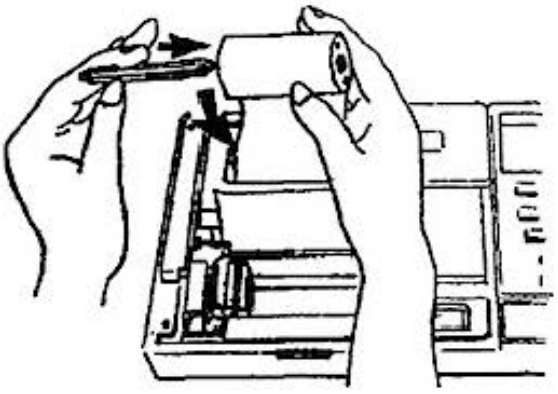


图8

（5） 盖好打印机盒盖，此时，要把纸卷头从切纸器穿出打印机，图9。

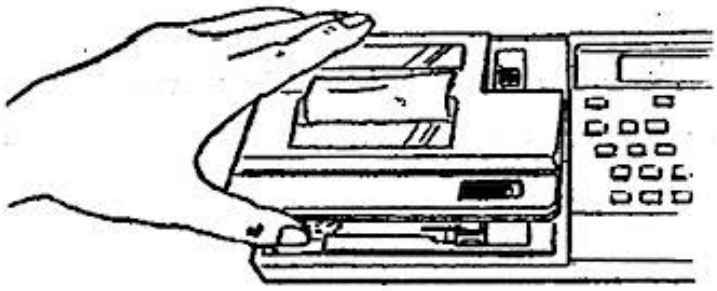


图9

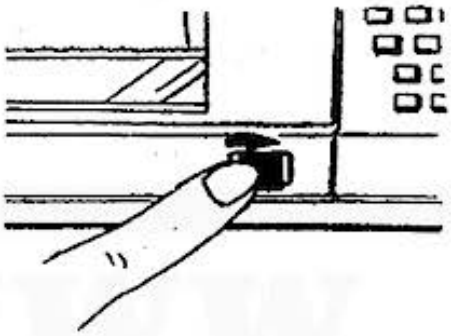


图10

（6） 锁上打印机盒盖，图10。

A．5．换笔

印刷机能装上四种颜色的笔。色笔的安装位置如下图所示，详见印刷机使用手册。

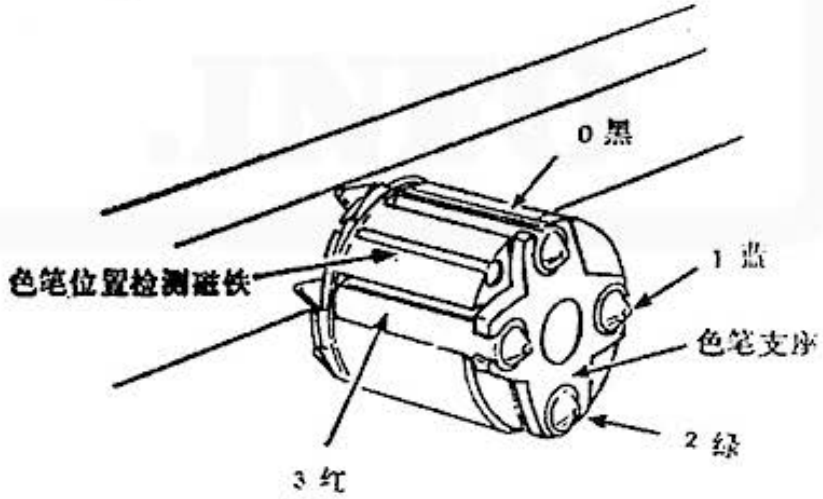
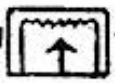


图11

笔槽自色笔位置检测磁铁开始顺时针方向编号为0，1，2和3，这些编号对应于COLOR命令的色笔位置的选择。

关于色笔的安装或更换，遵照如下的步骤：

（1）按计算机的数字键0之后，按下印刷机上的  键，当色笔支座移到左边并

旋转时，便使印刷机进入了换笔状态。随着在顶上的色笔的变更，色笔支座便向右边移动。（当色笔支座开始移动时，释放该键。）

（2）卸色笔时，按下卸笔杆，座顶的色笔便跳出来，见图12。

注：卸下色笔时，稍为按住它，以防跳进印刷机盒内。

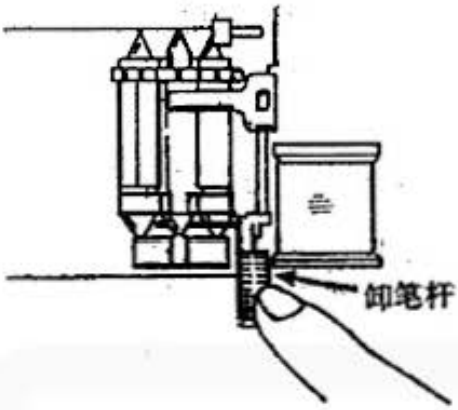


图12

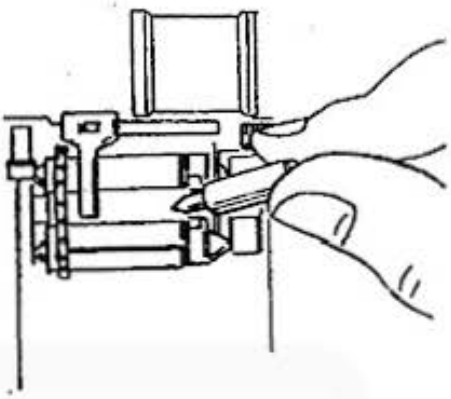
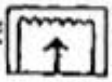
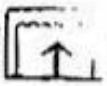


图13

（3）装上新的笔，图13。

（4）要装入或卸下下一支笔时，可按  键，色笔支座返回左侧，旋转支座使下一支笔转到座顶来，换笔后，又移到右边去。按步骤（2）和（3）卸下和更换新的一支。

（5）更换或安装完毕，按下印刷机的  键及计算机的 CL 键，使印刷机脱离换状态笔且使色笔返回左边。

注：使用印刷机时，要装入四支笔到支座上，缺笔工作甚至少一支也可能使笔色的更换出现错误。

色笔的保管：

色笔在使用时装上印刷机，用后卸下，盖上帽盖并装入笔筒存放。

注意，装上印刷机的色笔长时间留在机上，或者反过来，长时间暴露在外会使印字墨水干枯。

B• 磁带录音机的使用

说明：此后所述的有关打印机和磁带录音机的命令仅适用于在备用的打印机 CE—15（带插入录音机的接口）上使用。由于计算机没有配备这些命令，因而使用这些命令的程序设计只当连上了 CE—150 时才能进行。所以，使用这些命令进行程序设计时务需连上 CE—150 。

B. 1. 磁带录音机的操作

我们建议你用一个小型、简单的程序遵照下面的步骤来练习，这样，万一有问题，也易于重新执行操作。现在，送入程序。

你必须准备好磁带录音机以便进行程序和数据的传输。为此，要遵循下面的步骤：

1. 把 CE—150 接口上的 "remote" 远程控制开关置于 OFF 位置。

2. 把磁带装入录音机（重要的一步）。

3. 找到磁带的空白部分，假如磁带是全新的，就跨过前面不能录音的部分。若录音机有转数记数器，记下当时的转数，这对于你重新寻找保存的程序极有帮助。好了，进行下一步，……

4. 假如你的录音机有自动音量控制的话，就置于自动方式，若是手动音量控制，则把音量电平调在中间与最大之间的中间（即3/4位置）。假如你的录音机有音调控制，则把音调控制旋钮调到中间与最大之间的中间（即3/4位置）。

5. 把接口上的 "remote" 开关调回 ON 位置，如果你的录音机没有 "remote" 即远程控制功能，（这意味着没有位置与布线束的一条线相连接），可利用 "PAUSE"（暂停）键临时中止记录。如果你的录音机连 "PAUSE" 键也没有，事情就很难办。我们热切建议你取得一部有这种功能的录音机，它将使你新开始的程序设计生涯方便得多，最好是远程控制功能和暂停开关两项都具备。

6. 同时按下 RECORD（录音）键和 PLAY（放音）键。假如你使用的是不具备暂停开关的录音机，则必须在想保存程序之前直接这样做。

都准备好了吗？好继续阅读下面的好内容，……

B. 2. CSAVE 命令

好，现在用倒着计数的方式回答下面的核对清单。

1. 磁带录音机准备好了吗？　　是—否—

2. 磁带装入了吗？　　是—否—

3. 计算机开关打开了吗？　　是—否—

4. 程序输入计算机了吗？　　是—否—

5. 全部连线都正确接上了吗？　　是—否—

6. 记下计数器的转数了吗？　　是—否—

7．你穿好衣服了吗？　　没关系的，—

假如你对上面任一个提问回答是"否"，就重新阅读上节，解决那个问题。

好啦，再说一点你就可以开始了。

保存程序使用的命令是同一个，必须给程序一个"文件名"，这是出于引用的目的。你起的文件名不能长过16个字符。要用一个文件名存贮程序时，敲入：

CSAVE [SHIFT] "PROG. －1 [SHIFT]"

你的程序就以名字"PROG. －1"存贮起来。你可以按你的要求赋任何文件名，无论什么名都要是最易记的。另外，注意文件名有16个字符长的限制，假如文件名超过16个字符，则超过部分无效。较好的方法是文件名包含程序的记录，包括程序名、程序在磁带上开始和结束的位置（从转数计数器获得），以及有关该程序是做什么的主要描述。

按下 ENTER 键，此时应听到一声尖锐的蜂鸣声，磁带开始转动，而且，指示符"BUSY"应显示出来，它告诉你，计算机正忙于将你的程序从存贮器转送到磁带。假如不出现这种现象，再从本节开头重来。我试验了十次才获得成功，故不要泄气。我能成功，你也能成功的。

一旦计算机传送程序到达程序的末尾，"BUSY"指示符就消失，录音机停止转动，显示屏中就重新出现提示符。祝贺你第一个程序已经保存了起来，将来就可引用了。为了证实真正完成了程序的存贮，我们可按下节所述从磁带上把程序读回计算机存贮器。

B、3．CLOAD命令

既然你的第一个程序存到了磁带上，必然想看看是否真正地存了进去。当然，做到这一点相当简单，只要使用 CLOAD？命令就行了。

你也许会问，"这个命令是做什么的？"好的，在你存贮了程序之后，只要敲入 CLOAD？命令，计算机就会把被保存在磁带上的程序与存贮器里的那个程序进行比较，如果一切顺利，就会平静地显示你的文件名并结束校对。假如不是一切顺利，就会显示出错信息，通常是"ERROR 43"。这告诉你，磁带上的程序由于某种未弄清的原因与机内存贮器中的程序不同。抹去这部分磁带的程序，重新开始录写和检验。还要查对所有的接线，以及试将音量和音调控制调大一点。

使用 CLOAD？命令时，在记录内容的前面会插入一段引导乐音，这种引导乐音，由一种频率大约2.5千赫的稳定的、未调制的信号组成。假如磁带没有正确接上，不指示任何

出错信息，显示屏上也不出现任何信息。

下面说明将磁带上的程序写入计算机内存的过程：

1．把录音机上的远程控制开关调到 OFF 位置；

2．还是利用转数计数器，把磁带倒绕到你开始录写的位置。（可见计数器是何等的方便！）；

3．停止倒带；

4．把远程控制开关调回 ON 位置；

5．按下录音机上的 PLAY 键钮；

6．敲入命令：

CLOAD [SHIFT] "PROG.－1 [SHIFT] "

并按 ENTER 键。

（记住："PROG.－1" 是我们原先给程序起的文件名，假如是用另一名字来保存程序就要用那个名字代替 "PROG.－1"）；

7．显示器中出现指示符 "BUSY"，磁带上的程序就会被引导回到计算机的内存贮器中来；

8．敲入 RUN，前面保存的程序便重新出现在显示器中，磁带上仍然保留着程序的副本，因此，同一程序可以反复不断地写入机内。写入过程中，当显示出错信息 ERROR 43或 ERROR 44时，重新从上述（1）步开始写入。

B、4．PRINT # 和 INPUT # 命令

PRINT # 命令：

至此，我们已经介绍了 CSAVE 和 CLOAD 命令的用法，我们打算采用两个类似的的命令。命令 PRINT # 将一个或一组变量的值保存在磁带上。与 CSAVE 命令保存程序的作用不同，保存数据的目的是使你能在另外一个程序中使用相同的数据。例如，在下列程序中，用了变量 T $：

```
10: PRINT "WHAT IS YOUR NAME?"
20: INPUT T$
30: PRINT T$
```

如果你想把 T $ 的值保存起来以备另一个程序使用，必须发 PRINT # 命令以保存

在磁带上，可以利用两种方法来进行：

1．手动操作录写

2．通过程序录写

注：这种操作要求使用磁带录音机，故要准备好录音机接收数据。假如你对连接录音机的方法无把握，可回到前一节查阅。

1．手动操作

手动操作的方法有几种，可任选用：

方法 1：

程序运行之后，把方式开关调在 RUN 方式，敲入：

PRINT SHIFT #A，B，C（然后按 ENTER）

磁带录音机就开始工作，将各变量的值录在磁带上。

方法 2：

假如你想对某些要保存的变量作标记，敲入下列键串：

PRINT SHIFT #"文件名"：A，B，C

在这种情况下，你恰好是规定变量A、B和C作为给定文件名下要保存在磁带上的变量。

方法 3：

也可规定保存全部相关变量的值，敲入：

PRINT SHIFT #"文件名"：B(*)

*号表示保存全族的"B"，包括B(1)和B本身。

2．通过程序录写

要由程序来录写变量，只要在程序中写上带行号的 PRINT # 命令。你可使用上述的任一种格式。当机器遇到这一行时，就自动地启动磁带录音机，开始把变量数据传送到磁带上录写下来。请你再实验一次。假如不行，回头阅读上一节，说不定忽略了某些很简单的事

情。

INPUT＃命令：

这个命令允许使用的格式与 PRINT＃ 命令相同。所不同的在于 INPUT＃ 命令把数据从磁带传输到机器(而 PRINT＃ 命令是把数据从计算机传输到磁带)。可以手动地使用 INPUT＃ 命令，也可以写入程序中执行。记住在开始使用这个命令之前，要准备好磁带录音机。

注：假如你在 INPUT＃ 命令行里规定从磁带读入计算机的数据变量多于磁带上存有的变量，则多余的变量被赋零值。若在 INPUT＃ 命令行中规定读入计算机的数据变量少于磁带上存有的变量，则磁带上多余的变量被忽略。

B、5．归并命令 MERGE 的使用

归并命令使你在同一时候能在计算机存贮器中存贮多个程序。例如，假定计算机存贮器中存有下列程序：

```
10: PRINT "DEPRECIATION ALLOWANCE"
20: INPUT "Enter method,     "; A
```

到此，你记起磁带上在文件名"DEP1"下有一类似的程序段，当然你会想看看这个程序中是否有目前构造的程序可利用的部分。第一步是找到具有文件名"DEP1"的磁带，尾接磁带到"DEP1"开始的地方。

现在敲入：

MERGE "DEP1" 并按 ENTER

此时，计算机就将程序"DEP1"接着上面的程序写入到计算机的存贮器中。程序"DEP1"写入后，就可发现存贮器内存入了类似于下面的程序内容：

```
10: PRINT "DEPRECIATION ALLOWANCE"
20: INPUT "Enter methed,     "; A
10: "DEP1"; REM>>Second Module<<
20: PRINT "INTEREST CHARGES"
```

```
30: INPUT "Amount Borrowed: " ; B
    ⋮
```

（等等）

注意到，不象 CLOAD 命令，新写入的程序不会取代正存在机内的程序，而且，有些行号还是重复的。还注意到，归并的程序块中的第一行用了一个"标志"，它允许将程序块"链接"在一起（见下面"归并程序块的链接"）。

在着手进行任何进一步的编辑或程序设计之前，研究一下下列的问题是重要的：

重要注释

1）。一旦执行了一个 MERGE 命令，就不允许对前面存在机内的程序行进行插入、删除或改变。

例：

```
10: "A" REM This is existing program
20 FOR I=1 TO 100
30 LPRINT T
40 NEXT I
    ⋮
```

（等等）

面在它和下一程序进行合并之前，可对它进行必需的修改。然后，合并下一个程序，例如，MERGE "PROG2"：

例：

```
10 "B" REM This is MERGED program
20 INPUT "Enter depreciation: " ; D
30 INPUT "Number of Years: " ; Y
40 …（等等）
```

现在，你就可以对上面的程序作修改，因为它是最后合并的部分，假如你需要对第一个程序作进一步的修改，可遵循下列步骤进行：

1．使用 CSAVE 命令，把这时存贮器内的程序录写到磁带上。使用一个新的名字，例如，"PROG3"。

2．再回头把在第一步中保存到磁带上的程序写入计算机存贮器内。

3．现进行任何需要的修改。然而，记住：修改只能对第一个程序进行或者只对重复程序行中第一次出现的行进行。即，假如程序行10出现在第一个程序中，又出现在第二个程序，则修改只对第一次出现的10行起作用。那末，假如你的意图是对第二次出现的10行进行编辑修改，但实际上仅错误地影响到第一个程序部分的10行。

附加程序行的添加：

附加的程序行可以加到现存程序的末尾，但是，仅当附加程序行的行号大于前面所用的那些行号时才能添加。注意，附加的程序行并非一定要出现在程序单的"底部"，因为程序块的链接须经由标志进行（见下述），因而，是否出现在底部是无关紧要的。

归并程序块的链接

由于机器是按逻辑顺序执行程序行的，当它遇到标记的行号在顺序上出现中断时，即假如行号10，20，30后面又跟着第二程序块的重复行号，机器就会在第一个程序块的30行后停止程序的执行。

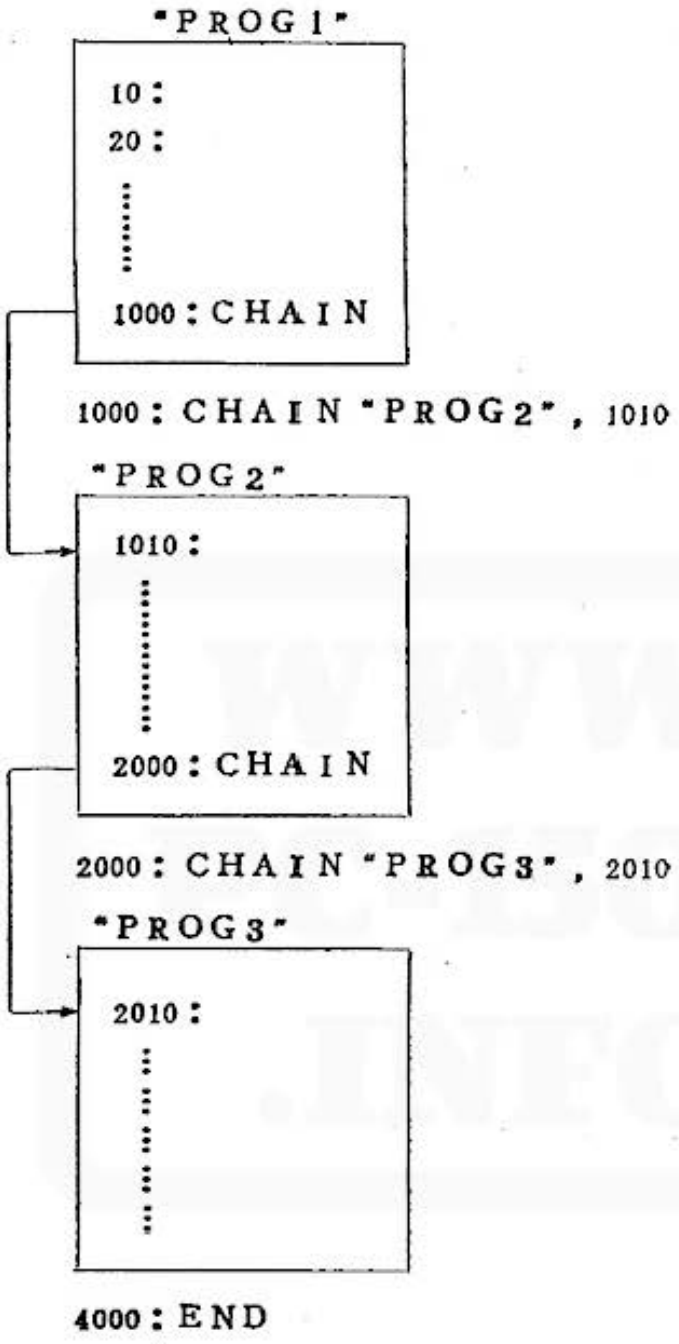
为了把各种程序块"链接"到一起，可以使用下列有效的链接技术：GOTO "B"，GOSUB "B"，IF…THEN "B"（这里使用 B 只是作为一个例子，除键盘下第二行出现的保留字或保留字母（即从Q到P）外，其它任何标志都可使用。

B、6．CHAIN 语句（链接语句）

CHAIN 是一个指令，它只能在程序内部使用，它不能象 CSAVE、CLOAD 和 MERGE 命令一样可以手动操作。

CHAIN 语句使得一个大得不能一次全部存入计算机存贮器的程序也能够运行。这样大的程序必须分割为几段，每一段的末尾使用一个 CHAIN 语句，这些程序段可用 CSAVE 命令录写在磁带上。

例如：让我们假定有三个程序段，分别名为 PROG1、PROG2、PROG3，每一程序段，用一个 CHAIN 语句结尾。



程序执行中，当计算机遇到 CHAIN 语句时，就把下一个程序段调入计算机的存贮器中并执行这个程序段．照这个方法，所有的程序段便可以最后运行完毕．

B．7．使用两个磁带录音机

当使用两个录音机时，其中一个用于录写，另一个用于读出，如图所示．印刷机／盒式磁带录音机接口与两台录音机之间使用连接电缆连接，并用辅助电缆进行远程控制．

CE—150 接口配备有两个远程控制端 REM0 和 REMI，每一个都可使用．然而，

在程序操作方式（不是手动操作），程序中要指定磁带录音机应连接到 REM 0 端还是 REM 1 端。因此，这些远程控制端也应根据程序中的指定连接。

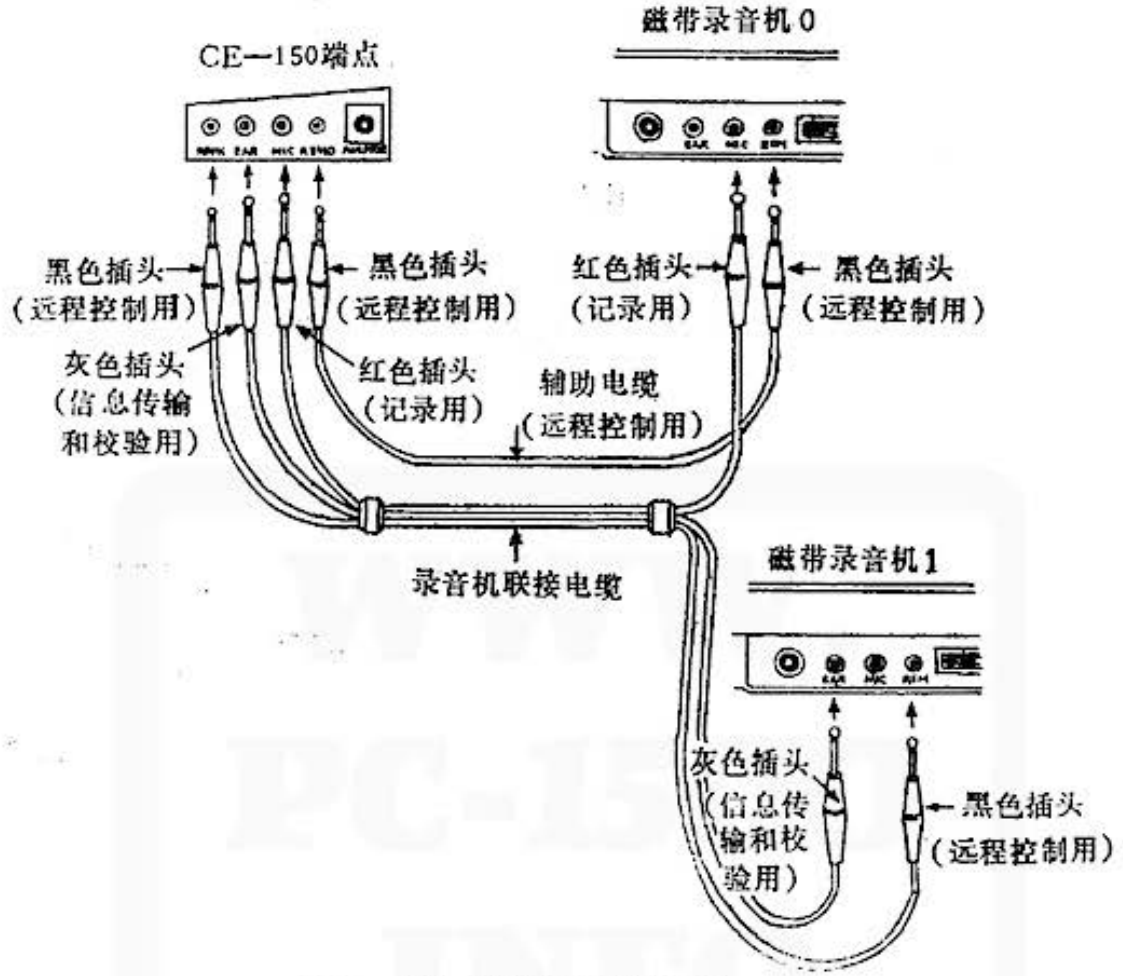


图14　两台磁带录音机的连接

这一节解释如何操作与接口的 REM 1 端连接的第二台磁带录音机。

1．录写

步骤：

（1）敲入 RMT OFF 并按下 ENTER 键，清除第二远程控制功能（磁带录音机 1 的控制按上述的说明）；

（2）把磁带装入录音机中；

（3）敲入 RMT ON 并按 ENTER，设置第二远程控制功能；

（4）按照在前面单台磁带录音机的使用中所解释的相同方法，调节音量和音调控制；

（5）同时按下录音机上的 RECORD（录音）键和 PLAY（放音）键；

（6）执行录写指令；

对于程序录写：CSAVE－1 "文件名" ENTER

对于数据录写：PRINT #－1，"文件名"；变量，变量，…

ENTER

例：指定 PRO 或 RUN 方式：

CSAVE－1 "PR－1" ENTER

录写完成之后，显示器中重新出现提示符，磁带停止转动，重绕磁带以备校验。

2．计算机和磁带存贮内容的校验

步骤：

（1）敲入 RMT OFF ENTER 键串以清除远程控制功能；

（2）还是使用转数计数器，重绕磁带到你开始录写的位置；

（3）敲入 RMT ON ENTER 键串，设置远程控制功能；

（4）按照在前面单台磁带录音机的使用中所解释的相同方法，调节音量和音调控制；

（5）按下放音键；

（6）执行校验指令：

CLOAD ？－1 "文件名" ENTER

例：指定 PRO 或 RUN 方式，操作：

CLOAD？－1 "PR－1" ENTER

当两个内容相同时，停止执行，显示回到提示符。

3．将磁带上保存的信息传送到计算机

步骤

（1）敲入 RMT OFF ENTER 键串，以清除远程控制功能；

（2）把录有信息的磁带装入录音机中；

（3）敲入 RMT ON 并按 ENTER 键，设置远程控制功能；

（4）按照在前面单台磁带录音机的使用中所解释的相同方法，调节音量和音调控制；

（5）按下放音键；

（6）执行传送指令：

对于程序的写入：CLOAD－1 "文件名" ENTER

对于数据的写入：INPUT#－1，"文件名"：变量，变量，……

ENTER

例：指定 PRO 或 RUN 方式，操作：

CLOAD－1 "PR－1" ENTER

传送完成后，显示屏中出现提示符。

C、印刷机的使用

说明：本节中所有的解释和例题假定你已经：

1）正确地连接 PC－1500 计算机到CE－150接口上；

2）给 CE－150 提供了 SHARP EA－150 电源变换器；（译注：当电池电力足够时，也可不用）

3）把色笔装入了印刷机中；

4）把纸带装入了印刷机中。

假如还没做好这些事情，请回到本章A节，参照那里的说明。

C．1．CE－150 印刷机的技术指标

每行字符数：每行4，5，6，7，9，12，18或36个字符，取决于字符大小的选择。

字符大小：1.2×0.8mm至10.8×7.2mm，取决于字符大小的选择。

打印速度：打印最小字符时最大速度为每秒11个字符。

旋转：两条座标轴上的任一方向都可打印字符。

颜色：红、蓝、绿、黑四种。

绘图系统：X－Y轴绘图系统。

走纸：手动或可编程序。

C．2．TEST 命令

使用印刷机时，你想做的第一件事情是检验 CE－150 印刷机的功能，接通计算机和接口的电源后，敲入：

TEST 并按 ENTER

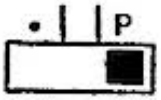
印刷机就绘出四个方框，每个各用一种颜色，方框的颜色从左到右，就是你装入色笔时所选择的0至3号颜色。

C、3．计算过程印出

利用 CE—150 接口，能够印出在 PC—1500 计算机上所进行的一系列手动计算的副本，如下所示：

```
12000*.065
                780
780+25.56
                805.56
SIN 30
                0.5
TX=.065
                0.065
P=20000*TX
                1300
P/12
                108.3333333
```

这只要把接口上的印刷机功能转换开关置于"P"的位置即可：



为了防止自动印出手动计算内容，印刷机的功能转换开关须置于园点"·"位置。开关置于此位置后，仍可通过在计算前面预置 LPRINT 命令的方法印出所选择的结果（见 LPRINT 命令段）。在此方式中，其它诸如 LLIST，LINE 等等产生印出字符或绘图的命令同样起作用。

打印时，所用的颜色将是前面指定的颜色，假如你刚打开机器的电源，此时所选用的色

笔便是对应于 COLOR 0（0号笔）命令，要改变颜色，必须发 COLOR 命令（见有关的章节内容）。

用来打印手动计算内容的字符的大小取决于前面的指定，假如前面规定的字符大小是1号或2号（见下面的C. 6小节），则这个字符保持有效，假如前面的字符号数大于2，则用2号。

自动打印时，印刷机的工作方式要设置为 TEXT。假如想要在 GRAPH 方式中，并希望回到这个方式来，必须发 GRAPH 命令（印刷机功能方式在下节解释）。

C. 4. 印刷机的工作方式

印刷机可在两种方式之一工作：TEXT（文本）方式或 GRAPH（绘图）方式，这两种方式大体上与人敲键盘和绘图对应。由于大多数印刷命令只能在一种方式或另一种方式中工作，因而，在发指令之前选择适当的方式是十分重要的。

TEXT 方式用于打印数和字符，印刷纸的宽度划为若干列，其列数与指定的字符大小有关。对于格式化的文本信息的打印，本机提供了垂直的和水平的打印命令。

在 GRAPH 方式，可以印出各种不同的数字、字符和表格，还可使用命令在直接座标或相对座标系中画实线和虚线，所有的绘图利用标准的X—Y座标系统。

指定 TEXT 方式用语句：

```
TEXT
```

就行了，某些命令（下面讨论）会使机器自动转换到 TEXT 方式。

规定 GRAPH 方式的方法也很简单，语句

```
GRAPH
```

就会使机器进入这个方式，同时把印刷笔置于打印纸的最左边。

C. 5. 程序的列表印出

LLIST 命令使当前存在计算机内的程序或程序的一部分打印出来，由于能有选择地打印程序部分，LLIST 命令在程序的调试和发展过程中极其有用。

LLIST 命令的形式与 LIST 命令的形式类似，不过，由于 LLIST 命令的用途较多，各种命令形式略有不同。

LLIST命令的形式如下：

LLIST

打印当前存在于程序存贮器中的全部程序行。

LLIST〈表达式〉

只打印表达式所给定行号的程序行。

LLIST，〈表达式〉

打印程序开头直到表达式给定行号（包括该行）的全部程序行。

LLIST〈表达式〉，

打印自表达式给定行号开始的程序行。

LLIST〈表达式1〉，〈表达式2〉

打印自表达式1给定行号开始直至表达式2给定行号止的程序行。那末，假如命令为：

LLIST 100，150

则 100 行和 150 行之间的全部程序行（如果有的话），会被列表印出。

LLIST "标志"

打印包含给定标志的程序行

LLIST "标志"，

打印自包含给定标志的程序行开始直至程序末尾的全部程序行。

注：规定一个不存在的标志会发出 ERROR 11 的出错信息。

打印程序时，所用的颜色是前面指定的颜色，假如刚刚打开机器开关，你所选择的色笔便对应于 COLOR 0，要改变颜色，必须发 COLOR 命令（见有关的章节）。

用于打印程序的字符大小取决于前面的规定，假如前面规定的字符号数为1号或2号，则这个号数保持有效，假如前面规定的号数大于2号，则使用2号。

LLIST 命令使打印机的工作方式设置为 TEXT 方式。假如你要的是在 GRAPH 方式而希望回到这个方式，须发 GRAPH 命令。

程序列表印出期间，PC—1500 计算机会对程序行进行调整以使容易阅读，这是通过在行号中留空位来实现的：1至3个数字长的行号会在3个字符的区域内进行右对齐，在4或

5个数字长的行号会在5个字符区域内印出.

例如:

```
10: REM  WIDTH 3
20: REM      〃
300: REM     〃
2001: REM  WIDTH 5
2010: REM    〃
```

C. 6. 可编程序印刷机控制命令

C. 6. 1. CSIZE

CSIZE 命令为所有后继的印出规定字符的大小，有九种字符号可以利用，范围是从每个打印行36个字符到每个打印行4个字符.

CSIZE命令的形式是:

CSIZE 〈表达式〉
(适用于任一方式)

命令中，表达式的值必须是1到9范围内的某个数，每一种字符号的字符宽度和高度由表一给出:

表一

CSIZE	1	2	3	4	5	6	7	8	9
每一打印行字符数	36	18	12	9	7	6	5	4	4
每个字符高度(mm)	1.2	2.4	3.6	4.8	6.0	7.2	8.4	9.6	10.8
每个字符宽度(mm)	0.8	1.6	2.4	3.2	4.0	4.8	5.6	6.4	7.2

C. 6. 2 ROTATE

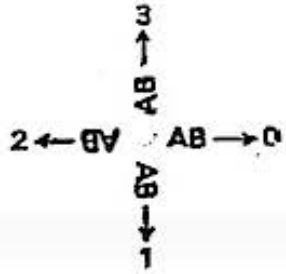
ROTATE 命令仅适用于 GRAPH 方式，用以规定打印的方向，印刷机能在四个方向上打印：向上，向下，左到右以及右到左(字符颠倒)，这些方向如下图所示. ROTATE

命令的形式为：

ROTATE 〈表达式〉
（仅适用于绘图方式）

表达式的值必须是0到3范围内的某个数．ROTATE 0是指定通常的自左至右打印字符的方式．

方向图：



C．6．3　COLOR

COLOR 命令允许为后继的所有打印字符和绘图规定要用的色笔．假如每一个装笔的位置上装有不同颜色的笔，则可用COLOR命令改变笔的颜色．COLOR 命令的形式为：

COLOR 〈表达式〉
（适用于任一方式）

表达式的值必须为0到3范围内的一个整数．每一个整数对应于不同的笔．整数所代表的颜色是变化的，这依赖于色笔安装在支架上的次序．它们的对应关系可由 TEST 命令确定（上已述）．

0到3范围内的非整数会被截为整数，其它的数将产生错误信息 ERROR 19。

当 PC—1500计算机关断然后接通时，又选择了对应于0号的色笔。

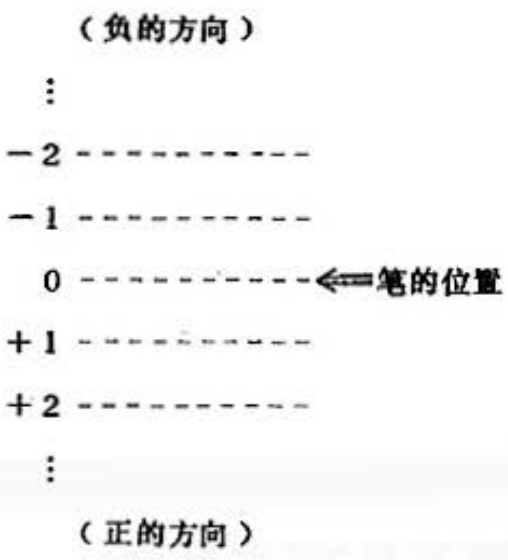
在 TEXT 方式，执行 COLOR 命令时会使笔的位置重新设置在打印纸的左边，在 GRAPH 方式，笔的位置会回到前面的打印位置。

C．6．4　LF（换行）

LF 命令可使印刷机内的打印纸向前或向后移动．LF 命令的形式是：

LF 〈表达式〉
（仅适用于 TEXT 方式）

如果表达式的值是一个正数，则纸带向前进，前进的行数由表达式的值确定，表达式的值为负数，则使纸带向后退回，后退的行数由表达式规定。用下图表示：



纸带移动的实际距离与LF指令规定时有效的字符号数有关。

当纸带沿反方向行进时（即被拉回去时），内部计数器会阻止它向后移动10.24厘米(约4英寸）以上。

注意：当纸带馈送机构工作时，不要企图装入纸卷，因为这样会损坏印刷机。

C．6．5．LPRINT

LPRINT 命令是在印刷机上显示文本信息的主要命令，它在本质上与 PC—1500 的 PRINT 命令类似，也具有 PRINT 命令的几个相同的形式．然而，由于可利用印刷机的附加功能，LPRINT 指令是较为复杂的，因此，我们将集中讨论 LPRINT 语句的使用中有关的细微区别．

下面对 LPRINT 命令的讨论假定只在 TEXT 方式，虽然，所给出的命令形式可在 GRAPH 方式中使用，但它们的操作不同．

单项的打印仍和普通的相同：

LPRINT〈项〉

这里，"项"是指其内容要打印出的表达式、字符串、数或变量名．像往常一样，字符是左对齐的而数是右对齐的．

象显示屏上的光标一样，假如打印笔不是定位在打印纸的左边的话，打印便是从笔的位

置开始，笔的位置可由 LCURSOR 语句或 TAB 子句改变（见下述）。

由于当前字符大小的限制，当试图打印一个大到一个打印行无法打完的项时，问题就来了：假如这种项是一个数，就出现 ERROR 76 出错信息；假如这种项是一个字符串，该字符串就会在下一行继续打印。

对于打印两项的 LPRINT 语句，注意打印项的大小也是重要的，这种 LPRINT 语句的形式为：

LPRINT 〈项1〉, 〈项2〉

用 CSIZE 1 能保证两个数项打印在同一行上，在这种情况下，数项将在打印行的两半中按通常的方法对齐。而对于涉及字符串的 LPRINT 语句，则打印的图象就不象打印数项那样整齐；假如两个项能在一行打完，就会象通常一样对齐并打印在同一行；如果一行装不下，结果就是分为两行打印。对于字符号较大的两项则在两个连续行上打印。

在 LPRINT 语句中，同样可以使用分号，它有两个作用，一是指示打印项之间留的空位最少，而语句末尾的分号则是把后继的打印项集合到同一行上来。不论在何种情形，假如各项的总长度超过打印行的容量，超过的项就打印在后继的行上（需要多少行就打印多少行）。

使用分号的 LPRINT 语句的形式为：

LPRINT 〈项1〉; 〈项2〉; …（等等）

或写为：

LPRINT 〈项表〉;

下面的示范程序使用了上述几种形式的 LPRINT 语句：

```
10  A$ = "ABCDEFG"
20  B= 1 2 3 4 5 6
30  FOR I = 1 TO 3
40  CSIZE I
50  LPRINT A$
60  LPRINT A$, B
70  LPRINT A$; B
```

```
80 LF 5
90 NEXT I
```

产生的输出为：

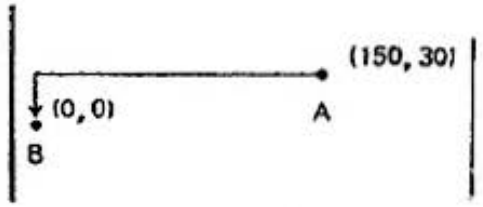
```
ABCDEFG
ABCDEFG
                123456
ABCDEFG 123456
```

```
ABCDEFG
ABCDEFG
        123456
ABCDEFG 1234
56
```

没有打印项的指定，也可使用 LPRINT 命令，适用于 TEXT 方式，也适用于 GRAPH 方式：

```
LPRINT
```

用这种方法，可以产生一个回车换一单行，然而，它不能使 GRAPH 方式的计数器复位，这样，在下面的例子中，尽管用了 LPRINT 语句把打印笔从点A移到了点B，但印刷机却自认为还是在座标（150，30）位置，且象先前一样执行后继的命令。



LPRINT 语句也可与 USING 子句联合使用，与在 PRINT 语句中的方法相同。但在 GRAPH 方式执行时，USING 子句只能出现在 LPRINT 语句中。

C．6．6 LCURSOR

LCURSOR 语句允许按类似于显示中的 CURSOR 语句的方法给打印笔定位。

LCURSOR 语句的形式为：

LCURSOR〈位置表达式〉

（仅适用于 TEXT 方式）

打印笔能移动的字符位置当然取决于有效字符号，一般，打印笔可以定位到小于有关字符号最大字符个数的一个空格的位置，关于每一种字符号的打印行宽表，可查阅本节 CSIZE 命令。

C．6．7． TAB

除了可以用在 LPRINT 语句内之外，TAB 语句与 LCURSOR 语句是相同的，这类 LPRINT 语句的形式为：

LPRINT TAB〈位置表达式〉：〈项表〉

对 LCURSOR 的位置表达式的有关注释同样适用于 TAB 命令的位置表达式。语句中，如果项表为空，上列指令的最终结果将是产生换一行。

C．6．8．SORGN（设立座标原点）

SORGN 命令为后继的绘图命令设立X—Y座标系的原点。SORGN 语句使当前的色笔位置作为原点。因之，这个指令通常是直接用于将打印笔移到纸上一个给定点的指令后面。SORGN 命令的形式很简单，为：

SORGN

（仅适用于 GRAPH 方式）

注：CE—150 打印机允许所要指定的打印笔位置位于可绘图位置的范围之外，在这种情形，打印笔移到尽可能远的位置，然后停下来，假如打印笔移进了这个虚假的区域然后执行 SORGN 命令的话，则后继的打印语句或绘图语句无效，就好象出现了程序错误或接口损坏的情形。

下列程序将座标原点设置在当前打印笔位置的上方 100 个单位长度，右边 100 个单位长度的位置，然后在新的原点画一个具有１０个单位长度的方框：

```
10 GRAPH
20 LINE (0, 0)-(100, 100), 9
30 SORGN
```

```
40 LINE (0, 0)-(10, 10), 0, 0, B
50 TEXT
60 END
```

C. 6. 9. GLCURSOR

GLCURSOR 语句能将笔移到任何X—Y座标点上而不画线。GLCURSOR 语句的形式为：

GLCURSOR （表达式1，表达式 2)

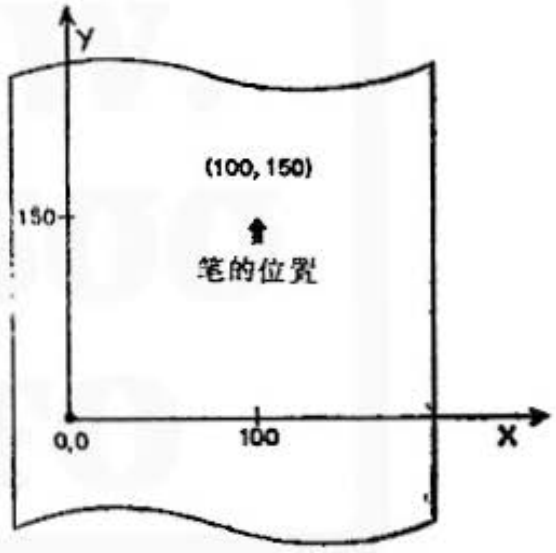
两个表达式的值须取－2047 至＋2047 范围内的数。表达式1 代表到指定点的 X 方向的距离，表达式2 代表到指定点的Y方向的距离。

注：假如指定点落在打印笔实际能够移到的点的范围之外时，笔就会在纸的一边上停下来，但在内部，控制笔的移动的计数器仍是算到终点。

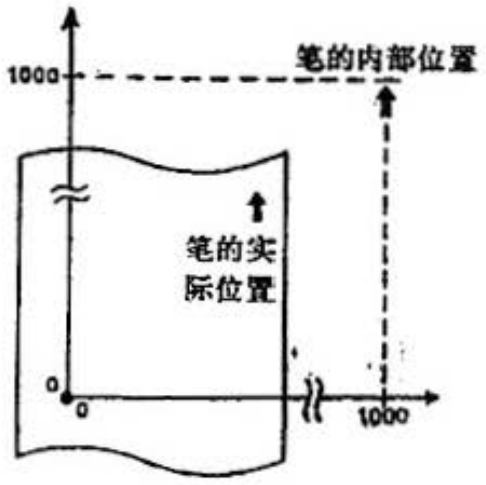
使用 GLCURSOR 语句的例子如下：

```
10 CLS
20 GLCURSOR (100, 150)
30 PRINT "笔的位置"
40 LINE (0, 0)-(10, 10), 0, 0, B
50 TEXT
60 END
```

右例中，笔移到了位置（100，150）上



在这个例子中，不正确地把笔置于虚拟区域的位置（1000，1000）上，而实际上，笔移向右边，并将打印纸卷回来。笔一到右边就继续向上移动直到它达到内设的例退极限１０Ｃｍ，在此停了下来。



C．6．10．LINE

LINE 命令是 GRAPH 方式的初始命令，它规定打印笔从一点移动到另一点，假如笔移动时放了下来，就画出一条线．LINE 命令还允许画出 8 种具有不同虚线长度的虚线．LINE 命令的第一种形式是：

LINE（X1，Y1）－（X2，Y2），线型号，色号

线的起点由表达式X1和Y1的值决定，线的终点由表达式X2和Y2的值决定，两个X值和Y值必须在 －2048到＋2047 的范围，规定的值超过这个范围就将出错．

线型和色号的参数是任选的，假如省略这两项，所用的值就是该命令之前有效的值．当然，颜色的代号是0到3范围内的一种，线型号这个表达式的值必须在0至9的范围内．表二标示每一种线型号的意义．

表二：

线型号	产生的线型	
0	实线	0 ————————
1	0.4mm虚线	1 ····
2	0.6mm虚线	2 - - - - -
3	0.8mm虚线	3 - - - - - -
4	1.0mm虚线	4 - - - - - -
5	1.2mm虚线	5 - - - - - -
6	1.4mm虚线	6 - - - - - -
7	1.6mm虚线	7 - - - - - -
8	1.8mm虚线	8 - - - - - -
9	提笔(不画线)	

LINE 命令的另一种用法是，把规定的点看作是一条对角线的端点，那末，命令就被处理为画一个该对角线所代表的方框．这种 LINE语句的形式是：

LINE（X1，X2）－（Y1，Y2），线型号，色号，B

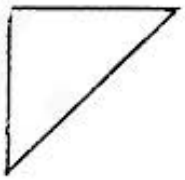
大写字母 B表示这是一个画方框的命令，其它参数的意义与该命令的上一种形式的参数相同。

最后一种 LINE 命令的形式允许规定多个点，第一个点之后的每一个点代表要画出的下一条线段的终点，当前的位置假定在线段的始点。这个形式的 LINE 命令是：

LINE (X1, Y1)－(X2, Y2)－…－(X6, Y6), 线型号, 色号

上面的形式中，使用了三个点，以表示在一个语句行中可以指定一系列的点（直至6个）。注意到，在这种命令形式中，不使用参数 B。下面给出一个使用四条线段绘制一个三角形的程序实例，程序中15行只起建立原点的作用，而三角形是由20行来绘制的。

```
10： GRAPH
15： LINE (0, 0)－(10, 0), 9 :SORGN
20： LINE (0, 0)－(50, 50)－(－50, 50)－(－50, －50)－(0, 0), 0, 0
30： TEXT
```

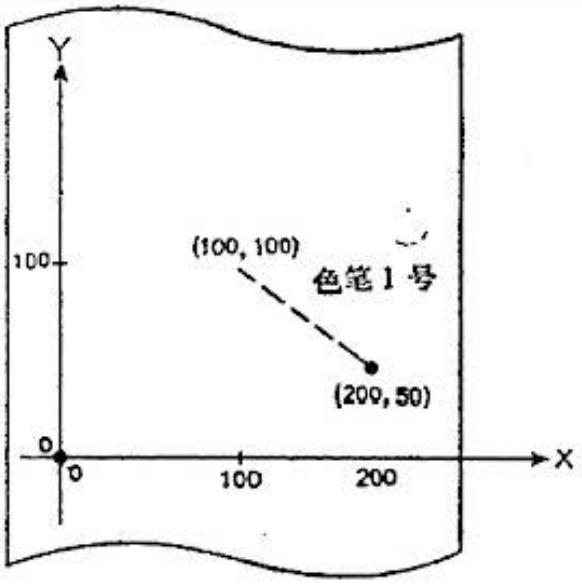


C. 6. 11. RLINE

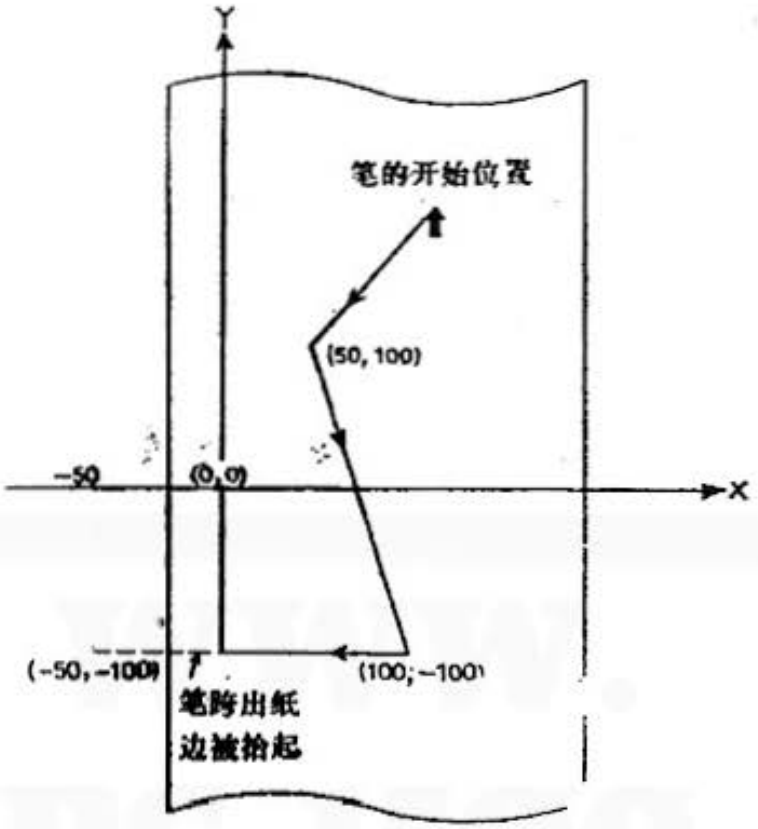
除了所有指定的点表示点的位置是相对于当前笔的位置，而不是相对于原点之外，RLINE 命令和 LINE 是相同的。RLINE 命令的形式与 LINE 命令的那些形式相同，只是关键字 RLINE 代替 LINE。

举例如下：

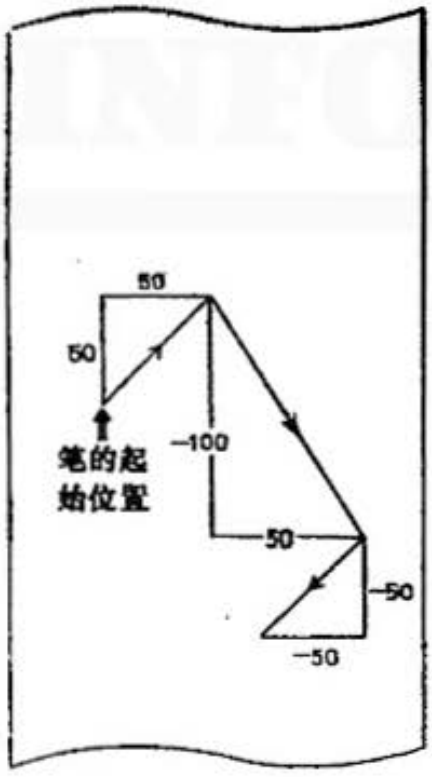
例1：LINE (100, 100)－(200, 50), 2, 1



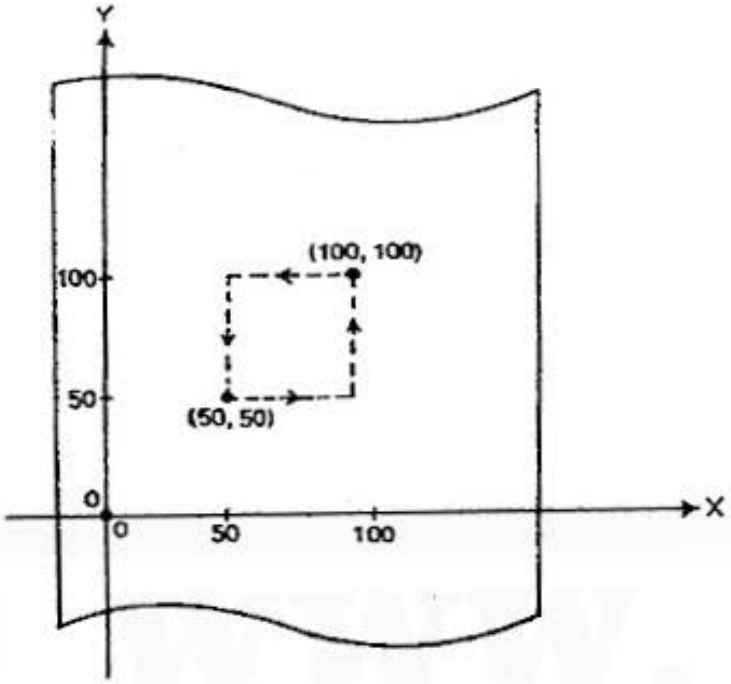
例2，LINE－(50，100)－(100，－100)－(－50，－100)



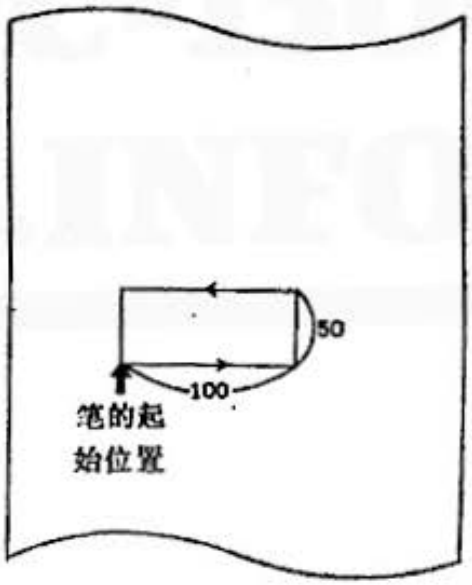
例3，RLINE－(50，50)－(50，－100)－(－50，－50)



例4：LINE (50, 50)－(100, 100), 2,,B



例5：RLINE－(100, 50),,,B



第七章 保留方式

A．保留键的定义和选择

SHARP PC－1500的一个重要的、可以减轻劳动的性能是具有6个其功能可重新定义的键。这些保留键允许计算机使用者规定某些频繁敲入的短语或键字，然后，按少量的键便可调用。这些保留键就是键盘顶行标记！，"，#，$，%和& 的六个键。这些键中的每一个可以赋给3个短语或键字，因而，总共可存放18个短语。

给保留键赋予短语在机器的第三种方式即保留方式中进行。为了进入RESERVE（保留）方式，按键：

SHIFT MODE

显示屏顶部的方式指示器现在就会出现 RESERVE 字样，要脱离保留方式，只要按一次MODE键就行了。

因为每个保留键可用来调用3个存放着的短语中任一个，因而就必须有一个按压保留键时选择调用哪一个短语的方法，这个方法就是用键盘的左下角上标记符号（▲▼）的键，称为保留选择键，这个键选择当前存放着的哪个短语与保留键对应。这里要强调说明的是保留选择键改变一次，所有保留键的对应关系就跟着改变，即是说，保留选择键选择的是一组短语，每一个短语与单一个保留键对应，究竟当前选择何组短语，由显示器顶部的小写罗马数字（Ⅰ、Ⅱ和Ⅲ）指示。

要给保留键赋短语时，先进入RESERVE（保留）方式（按 SHIFT MODE）并用保留选择键选择该短语要存放的组号（Ⅰ，Ⅱ或Ⅲ），然后按适当的保留键（！，"，#，$，%或&），根据你所按的保留键，显示器中将出现类似于下面的一些符号：

```
                                    RESERVE  I
F6:
```

（F后面的数字，在本例中为6，表示按了哪个保留键。）当出现这个提示时，就可按键输入要存放的短语。例如，敲入下列内容：

R U N I 0 0 ENTER

现在，这个短语就与你所选择的保留键联系在一起了。

让我们试验一下，按 MODE 键回到运行方式，再按你在本例中使用的保留键，这时，显示器中就会出现刚才贮存的命令：

RUN 100_

假如你按下 ENTER 键，计算机就会运行 100 行起的程序。保留键的好处在于每一次需要发命令时，不须按键敲入整个命令。

在保留方式，允许使用一个专用符号，它可以节省我们在上面例子中的某些麻烦，这个专用符号是使用@来代表ENTER命令。假如我们已经给保留键赋予短语“RUN 100@”，那末，一旦回到RUN方式并按那个保留键，就会开始执行程序。为了演示这一点，输入下列语句作为222行：

```
222 BEEP 5, 50 : END
```

现进入保留方式并用键串：

GOTO 222 @ ENTER

定义一个保留键（注意，为了给保留键本身进行定义，ENTER 键还是需要的）。

回到运行方式并按刚才定义的保留键，你就会观察到，这次保留键后面的 ENTER 键串是多余的了。

实际上，上面的产生蜂鸣声的例子可以通过把短语：

BEEP 5, 50 @

直接赋给一个保留键来实现，你不妨试试。

B• 保留键的识别

随着使用到的保留键的增加，你必将希望记住哪个键赋给了什么功能，PC—1500 允许你贮存三串字符（每一组保留键一串），用以识别保留键的功能，这些字符串与 PRO（程序）方式中的注释类似。

识别字串（称为“名单”）在RESERVE（保留）方式中建立。把方式开关转换到这个

方式并用保留选择键选择适当的保留键的组号，然后，不是象通常要做的那样按保留键，而是敲入一个“名单”并按ENTER，这个名单就与这一组建立了联系而被存贮了起来。

作为一个实例，假设我们已经给第Ⅰ组的1至6号保留键赋予了三角函数（正弦，余弦，正切，反余弦，反正弦和反正切）的名字，为了记住哪个键对应于哪一种函数，我们要规定一个名单。为此，转到RESERVE（保留）方式（按SHIFT MODE），并用保留选择键选择第1组（此时显示屏上出现罗马数字Ⅰ），现敲入：

"SIN COS TAN ACS ASN ATN"

就定义并存放了该名单。

按MODE键回到RUN（运行）方式。要读出保留键的意义，只要按RCL（读出）键，这个名单就出来了！再按RCL键一次就又出现提示符。

可以为每一组保留键建立一个名单，且每个名单的长度可以直至26个字符。注意，这种名单严格地说只是你的备忘录而已，你存放在名单上的单词或字母对于计算机是毫无意义的。

C• 保留程序的删除

正如你所知道的，N E W ENTER 能清除全部保留存贮器的内容。

请注意，上面这些键的操作必须在RESERVE（保留）方式中进行。

第八章　开始程序的执行

A• DEF键

DEF键（定义 define 的缩写）提供了几条节省时间的捷径，……

A.1　可定义程序的运行

DEF 键是第三种启动程序的方法，正如我们在关于RUN 命令这一节中所看到的，一个程序可用一个字母来“标记”。DEF 键可用来迅速地启动一个带标号的程序，这可通过在按了 DEF 键之后跟着按一个与程序的标号相应的字母键来实现。在这个方面，下列的键可以使用：

A, S, D, F, G, H, J, K, L, Z, X, C, V,

B, N, M, SPACE 和 =

作为一个实例，送入下列的语句，建立三个带标号的程序：

程序单：

```
10 "Z" : GOSUB 500
 20 PRINT "Z KEY"
 30 END
140  "A" : GOSUB 500
150 PRINT "A KEY"
160 END
270  "  " : GOSUB 500
280 PRINT "SPACE KEY"
290 END
500 CLS: PAUSE "YOU PRESSED THE " ;
510 RETURN
```

在 RUN 方式，试用 DEF（定义）键启动每一个程序。注意，指定一个字母如无相应的

带标号的程序存在，会产生出错信息 ERROR 11。

A.2 预先指定的关键字

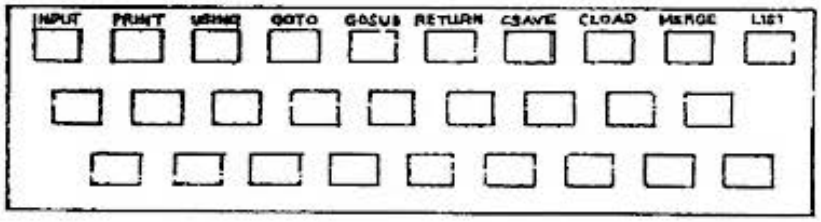
少数几个最频繁使用的关键字已经固定地指定给键盘上第二行的每一个字母键。这些字可在任一种方式通过按 DEF 键接着按该字母键来取用。例如，要取用关键字 USING，只要敲入：

```
DEF  E
```

可用的关键字及其对应的字母键是：

字母键	关键字	
Q	INPUT	
W	PRINT	
E	USING	
R	GOTO	
T	GOSUB	
Y	RETURN	
U	CSAVE	当印刷机／录音机接口联上计算机时可以使用
I	CLOAD	
O	MERGE	
P	LIST	

模板



计算机配有两块模板，用它们来识别规定给定义键的功能操作

A.3. AREAD 语句

用 DEF 键启动的带标号的程序，每一次运行时，不用 INPUT 语句也可以给它一个单

值，读这个值是由AREAD语句完成的，这个语句必须跟在程序标号后面，与标号在同一行上。AREAD语句的形式为：

AREAD 〈变量名〉

这里，变量名是合法的数值变量名或字符变量名。

为了把一个值传送给一个含有 AREAD 语句的程序，使用者要敲入这个值，按 DEF 键，然后按相应于程序标号的字母键。

作为一个实例，送入下列两个程序：

程序单

```
10 "X" : AREAD TM
20 TIME=TM
30 PRINT "TIME SET TO" ; TM
40 END
100 "Z" : AREAD D$
110 PRINT "TODAY IS" ; D$
120 END
```

回到RUN方式并敲入月、日、时刻，再按 DEF X，启动带标号X的程序：

[1] [2] [3] [1] [0] [2] [.] [4] [0] [0] [0] [DEF] [X]

这个程序会把系统钟调定在 DEF 键串之前指定的时刻（见TIME函数）。

为了启动带标号Z的程序，敲入一个星期几和 DEF Z：

[F] [R] [I] [D] [A] [Y] [DEF] [Z]

B• 自动程序启动 ARUN

程序不但可用 DEF 键迅速而容易地启动，而且它们可以在打开 PC—1500 电源开关时完全自动地启动。

为了实现这一点，要使用 ARUN 语句，这个语句必须是程序存贮器中的最前面的一个语句，否则无效。此外，为使 ARUN 语句起作用，必须具备其它几个条件，这就是：机器要在 RUN 方式中关闭电源开关，以及当机器接通电源开关时没有检出错误。

下面是一个向计算机操作员祝贺的程序，使用了 ARUN 语句：

程序单

```
10 ARUN
30 CLS
50 BEEP 5, 50
70 PRINT "WELL, HI THERE! "
90 END
```

C. 启动方法的比较

虽然，各种启动程序的方法表面上取得同样的结果，其内部操作是不同的。为了便于你利用这些差异，有必要讨论一下计算机内部的数据存贮，另外，包括一小节有关机器在运行程序之前所做的内部准备情况的比较。

C.1. 固定存贮区

虽然所有相同类型的变量是用相同方法利用的，但它们在机器内部的处理是不同的。PC—1500 包含一个具有足够存贮空间的"固定存贮区"，可以存放26个数值变量和26个字符串变量（字串长度可达16个字符）。因此，在这个区域内，固定地指定了变量 A到Z，以及变量A$到Z$。

所有其它的变量，包括那些由两个字符命名的变量，是在计算机的主存贮区内分配的。程序的指令也是占用了这个主存贮区，不过，变量在存贮器内是从与指令存贮单元相反的一端开始分配存贮单元的。由于程序指令和数据共享同一个存贮区，它们就有可能用尽全部可利用的存贮器，在这种情形，就会发生177至181类的出错信息。

重要的一点是，实现了程序启动时对两个存贮区所作的处理不同，这一点，在下一节的表中作了说明。实际上，除非用了一个显式CLEAR语句，固定存贮区里的变量是决不会被清除的，不论何时，当用RUN命令启动程序时，存放在主存贮器里的那些变量便被清除。

关于固定存贮区，有另外一个特性，就是这个区的数据可以重新定义为一个数组，对于数值变量，其名为@（At符），对于字串变量，名为@$，那末，记号@（1）与变量A是同一存贮单元，而@（26）与变量Z是同一存贮单元；记号@$（5）被认为与E$是同一单元，而记号@$（20）被认为与T$是同一单元。由于显而易见的原因，下标超过26是不允

许的。注意，数组@和@$在使用前不需定维。

C.2. 程序启动方法比较表

	RUN	GOTO	DEF
清除显示	是	是	否
光标回到第一列	是	否	否
WAIT命令的时间间隔调为不固定	否	否	否
跟踪方式改变	否	否	否
固定存贮区被清除	否	否	否
主存贮区被清除	是	否	否
FOR—NEXT, GOSUB内部存贮栈被清除	是	是	是
ON ERROR GOTO 指令被撤消	是	否	否
READ 操作的数据指针被恢复	是	否	否
USING格式被撤消	是	否	否

第九章 附 录

A• 缩写词简表

印刷机指令

COLOR	COL. COLO.	LPRINT	LP. LPR. LPRI. LPRIN.
CSIZE	CSI. CSIZ.	RLINE	RL. RLI. RLIN.
GLCURSOR	GL. GLC. GLCU. GLCUR. GLCURS. GLCURSO.	ROTATE	RO. ROT. ROTA. ROTAT.
GRAPH	GRAP.		
LCURSOR	LCU. LCUR. LCURS. LCURSO.	SORGN	SO. SOR. SORG.
LF	——	TAB	——
LINE	LIN.	TEST	TE. TES.
LLIST	LL. LLI. LLIS.	TEXT	TEX.

盒式磁带录音机指令

CHAIN	CHA. CHAI.	INPUT#	I.# IN.# INP.# INPU.#
CLOAD	CLO. CLOA.	MERGE	MER. MERG.
CLOAD?	CLO.? CLOA.?	PRINT#	P.# PR.# PRI.# PRIN.#
CSAVE	CS. CSA. CSAV.	RMT OFF RMT ON	RM.OF. RM.O.

语句

AREAD	A. AR. ARE. AREA.	DATA	DA. DAT.
ARUN	ARU.	DEGREE	DE. DEG. DEGR. DEGRE.
BEEP	B. BE. BEE.	DIM	D. DI.
CLEAR	CL. CLE. CLEA.	END	E. EN.
CLS	——	ERROR	ER. ERR. ERRO.
CURSOR	CU. CUR. CURS. CURSO.	FOR	F. FO.

GCURSOR	GCU.
	GCUR.
	GCURS.
	GCURSO.
GOSUB	GOS.
	GOSU.
GOTO	G.
	GO.
	GOT.
GPRINT	GP.
	GPR.
	GPRI.
	GPRIN.
GRAD	GR.
	GRA.
IF	——
INPUT	I.
	IN.
	INP.
	INPU.
LET	LE.
LOCK	LOC.
NEXT	N.
	NE.
	NEX.
ON	O.
PAUSE	PA.
	PAU.
	PAUS.
PRINT	P.
	PR.
	PRI.
	PRIN.
RADIAN	RAD.
	RADI.
	RADIA.
RANDOM	RA.
	RAN.
	RAND.
	RANDO.
READ	REA.
REM	——
RESTORE	RES.
	REST.
	RESTO.
	RESTOR.
RETURN	RE.
	RET.
	RETU.
	RETUR.
STEP	STE.
STOP	S.
	ST.
	STO.
THEN	T.
	TH.
	THE.
TRON	TR.
	TRO.

TROFF	TROF.	USING	U.
			US.
			USI.
UNLOCK	UN.		USIN.
	UNL.	WAIT	W.
	UNLO.		WA.
	UNLOC.		WAI.

命令

CONT	C.	NEW	—
	CO.		
	CON.	RUN	R.
			RU.
LIST	L.		
	LI.		
	LIS.		

函數

ABS	AB.	INKEY $	INK.
			INKE.
ACS	AC.		INKEY.
AND	AN.	INT	—
ASC	—	LEFT $	LEF.
			LEFT.
ASN	AS.		
ATN	AT.	LEN	—
CHR $	CH.	LOG	LO.
	CHR.	LN	—
COS	—	MEM	M.
			ME.
DEG	—		
		MID $	MI.
DMS	DM.		MID.
EXP	EX.	NOT	NO.

OR	——	SQR	SQ.
PI	——	STATUS	STA.
POINT	POI.		STAT.
	POIN.		STATU.
RIGHT$	RI.	STR$	STR.
	RIG.		
	RIGH.	TAN	TA.
	RIGHT.		
		TIME	TI.
RND	RN.		TIM.
SGN	SG.		
		VAL	V.
SIN	SI.		VA.

B• 电池的更换

更换电池时，注意以下几条可使你免除许多问题：

1）四只电池总要同时更换；

2）不要把新电池同用过的电池混在一起；

3）只能使用AA型、R6或SUM－3，1.5V干电池四只。

电池更换步骤：

计算机使用之前或不论何时电池指示符消失时，就需要更换电池，请遵照下面的步骤更换：

1．按 OFF 键关闭计算机电源开关；

2．用一枚硬币或一把小螺丝刀卸下电池盖板上的螺丝；

3．更换四只电池；

4．重新拧上螺丝钉时，轻轻地推移一下电池盖板；

5．着手计算前，按 ON 和 CL 键，并敲入 NEW 0 再按 ENTER 键。

6．检查下列的显示，

假如显示空白或显示“)”以外的其它任何符号的话，可取出电池并再装一次，并检查显示是否正常．

注意：

1 ）计算机长期不用时要把电池取出，以免电池的电解液漏出可能损坏机器．

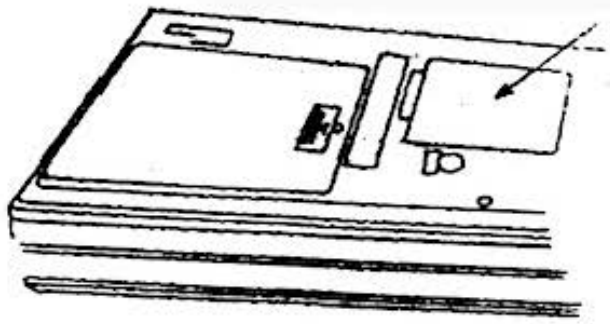
2 ）耗尽了的电池留在机内会由于电解漏液引起计算机损坏，故应及时将耗尽了的电池取出．

3 ）可重新充电的电池不能用作干电池在PC—1500机上使用．

4 ）CE—150 上用的交直流电源变换器 EA—150 当和 CE—150 分离时，也可用作PC—1500的交直流电源变换器．

（当PC—1500与印刷机／录音机接口 CE—150连接时，不要把 EA—150与 PC—1500计算机连接）．

模块存贮室使用注意



模块室：
这里容纳任选的存贮模块的地方。
不可触摸模块室内的任一插脚，因为这会由于静电效应对计算机起有害的影响。

图15　模块存贮室

C• PC—1500 ASCII字符编码表

高位二进制数字位置

低位二进制数字位置 b4,b3,b2,b1 ↓ / b7, b6, b5 →	000	001	010	011	100	101	110	111
十六进制	0	1	2	3	4	5	6	7
0000	0			空格 (SPACE)	0	@	P		p
0001	1			!	1	A	Q	a	q
0010	2			"	2	B	R	b	r
0011	3			#	3	C	S	c	s
0100	4			$	4	D	T	d	t
0101	5			%	5	E	U	e	u
0110	6			&	6	F	V	f	v
0111	7			'	7	G	W	g	w
1000	8			(	8	H	X	h	x
1001	9			)	9	I	Y	i	y
1010	A			•	:	J	Z	j	z
1011	B			+	;	K	√	k	{
1100	C			,	<	L	¥	l	:
1101	D			−	=	M	π	m	}
1110	E			.	>	N	∧	n	~
1111	F			/	?	O	—	o	▮

E. PC—1500 出错信息代码表

出错代码　　解　释

1．语法错误，这个错误发生在输入的内容错误．例如：

遗漏信息：

```
10：GOTO
```

或无效的命令：

```
10：5 A=1
```

或

```
10：NEW
```

（因为NEW不能作为语句）

〈显示〉 ERROR 1 IN 10

2．这个错误发生在出现一个NEXT命令而没有FOR命令或出现一个RETURN命令而没有对应的 GOSUB命令时．例如：

```
10：FOR A=1 TO 10
    ⋮
100 NEXT B
```

〈显示〉ERROR 2 IN 100

4．这种错误发生在没有与 READ 语句对应的数据时．例如：

```
10：READ X, Y
20：DATA 10
30：END
```

〈显示〉 ERROR 4 IN 10

5．这种错误发生在当数组变量用一个已存在的变量名来说明时．例如：

```
10：DIM A (10,10)
20：DIN A (5)
```

〈显示〉 ERROR 5 IN 20

6．这种错误发生在使用了一个未用 DIM 语句说明的数组变量时．例如：

```
10：CLEAR
20：A(3)＝1
```

〈显示〉 ERROR 6 IN 20

7．这类错误发生在当变量名与数据类型不对应时．例如：

```
10：A$＝10  或
10：FOR A$＝1 TO 10
```

〈显示〉 ERROR 7 IN 10

8．这种错误出现在当说明的数组变量超过 2 维时．例如：

```
10：DIM A(3,4,5,6)
```

〈显示〉 ERROR 8 IN 10

9．这种错误出现在数组变量的下标数超过在DIM语句说明的数组下标大小时．例如：

```
10：DIM A(3)
20：A(4)＝1
```

〈显示〉 ERROR 9 IN 20

10、这种错误发生在没有足够的存贮器可用来建立更多的变量时．例如：

键操作	显示
MEM ENTER	7
AB＝10 ENTER	ERROR 10

11、这种错误发生在程序中不存在指定的行时．例如：

```
10：PRINT "X＝"；X：GOTO 5
```

〈显示〉 ERROR 11 IN 10

12、这种错误发生在 USING 指令规定了一个不正确的格式时．例如：

```
100：PRINT USING "＃＃＃A＃"；10
```

〈显示〉 ERROR 12 IN 100

13、这种错误发生在当程序超过了程序存贮器容量或当保留键的规定超过了保留区容量时．例如：

150

键操作	显示
MEM ENTER	7
15 A=A+1 ENTER	ERROR 13

14、这种错误发生在：

（1） FOR 语句嵌套太深，超过了存储栈容量时；

（2）当分析表达式中，超过缓冲区空间时。

15、这种错误发生在：

（1）GOSUB 词句嵌套太深，超过堆栈区时；

（2）当分析表达式中，被处理的字符串超过字串缓冲区大小时。

16、（1）数值大于 1E 100或小于－1E100时。例如：123E99

（2）用十六进制设定的值超过65535时，例如：&IFFAB

17、数据类型（数值、字符串）与计算式的类型不对应时。例如：

```
1 + "A"  ENTER
```

18、自变量的数与表达式不对应时。例如：

```
LEFT$ ( "ABC" )  ENTER
SIN (30, 60)  ENTER
```

19、指定的数值超过容许范围时。例如：

```
10 : DIM  A(256)
```

〈显示〉 ERROR 19 IN 10

20、当指定一个固定存贮器数组变量但没有括号“(”跟在“@”或“@$”的后面时。

例如：100 : @$ = "A"

〈显示〉 ERROR 20 IN 100

21、在表达式需要变量的地方没有变量时。

例如： 10 : FOR I = 0 TO 10

〈显示〉 ERROR 21 IN 10

22、这种错误发生在当往计算机写入程序但没有可用于写入程序的存贮空间时。

23、这种错误发生在设定了一个不正确的时间时。例如：

```
TIME=131005.10  ENTER
```

26、这种错误发生在命令不能在当前的方式中执行时。例如：

〈RUN方式〉　　NEW　ENTER

27、这种错误发生在当备用印刷机没有连接上以及没有与给定标号对应的程序时。

例如：　键操作　　显示

DEF　I　ENTER　　ERROR　27

28、这个错误发生在当一个命令或函数代码被插入" "内部，或当你企图用INPUT命令或 AREAD 命令代替字符变量时。

例如：10：INPUT　A$

键操作　　显示

DEF　W　ENTER　　ERROR　28

30、这种错误发生在行号超过65539时（在65280～65539范围内，显示 ERROR　1），

例如：　102235　A＝10　ENTER

32、这种错误发生在执行键盘输入命令期间，图形指针位于152列～155列之间时（输入代码无法显示）。

例如：　100：GCURSOR　152

110：INP　X

〈显示〉　ERROR　32　IN　110

177～181　在生成程序期间，该程序修改了数据区，发生了这两种存贮区重叠的情况时。

O，224～241　在执行 INPUT 命令或 AREAD 命令期间，给了不正确的输入数据时。

例如：　10：INPUT　A

键操作　　显示

1 2 3　PRINT　ENTER　　ERROR　240

36、按照USING命令规定的格式，数据或字符不能显示。

例如：　10：USING "＃＃＃＃．＃＃"

20：PRINT　1 2 3 4 5

整数部分连同数的符号一起超过了4个数字位。

37、这种错误发生在数值计算中，当计算结果超过9.999999999E 99时。

38、这种错误发生在用0作分母进行除法运算时。

例如：　10：PRINT　5/0

39、这种错误发生在当进行不合理的计算时：

＊负数对数计算，例：LN（－10）

＊ASN，ACS在$|x|>1$的情形，例：ASN（1·5），ACS（100）

＊负数的平方根计算，例：SQR（－10）

磁带录音机使用有关的错误：

40、表达式的规定不合适。

41、给只读存贮器存贮区指定了SAVE和LOAD命令。

42、磁带文件数据规模太大无法写入机内。

43、当用CLOAD ？命令校验数据时，写入的数据格式与文件格式不相符。

44、发生了检查和（CHECK SUM）错误。

印刷机使用有关的错误：

70、（1）打印笔超过了座标范围：$-2048<=x, y<=2047$

（2）在执行后继的命令时，打印笔将超过座标范围。

71、（1）在TEXT方式，打印纸后退10.24cm以上。

（2）在TEXT方式，执行后继命令时，纸带将后退10.24cm以上。

73、TAB命令中所给的值不合适。

73、一个命令被用于错误的打印机方式（GRAPH／TEXT）。

74、在LINE或RLINE指令中，逗号（，）的个数太多。

注：送入的逗号7个以上就会出错，另外，若第一个逗号省略，则6个以上就会出错。

76、在TEXT方式，使用LPRINT指令打印计算结果不能在一行中打印完时。

78、（1）打印笔正在更换之中；

（2）电池电压低下的状态未被改变（见注1）。

这种错误发生在，由于上述两种原因之一，移动打印笔的命令（诸如LPRINT和LINE）不能执行时。

79、颜色信号没有执行（见注2）

80、电池电压低下（见注3）

注：

（1）假如ERROR 78是由于印刷机的电池电压低下引起的，可关闭CE—150的电源开关，对印刷机电池进行重新充电之后，便可继续工作。

（2）颜色信号是对COLOR命令而言的，且仅当色笔回到左边时才能执行换笔。当色笔在这个位置时，就能知道当前色笔的颜色位置号。

（3）充电后，立即按PC—1500的ON键，开始操作。

F．进一步阅读的建议

BASIC 程序设计

《BASIC题解和结构程序设计》（Problem Solving and Structured Programming in BASIC）by Elliot Koffman and Frank Friedman, (Addison—Wesley Publishing Co., Reading, Massachusetts), 1979.

《基本 BASIC 语言》（basic BASIC）by Donald M. Monro, (Winthrop Publishers, Inc. Cambridge, Massachusetts, 1978.

《BASIC 及其在商业中的应用》（BASIC With Business Applications）by Richard W. Lott, (John Wiley & Sons, New York, New York, 1977.

《实用 BASIC 程序》（Practical BASIC Programs）edited by Lon Poole, (OSBORNE／McGraw—Hill, Berkeley, Cali-fornia.), 1980.

一般参考书

《计算机与数据处理入门》（Introduction to Computers and Data Processing）by Gary B. Shelley and Thomas J. Cashman, (Anaheim Publishing, Co., Fullerton, California), 1980.

O・表达式求值的顺序

表达式的计算是按照下列层次进行的：括号内的表达式的优先权最高，而逻辑运算的优先权最低。假如在同一表达式或子表达式中遇到2或多个具有相同优先权的运算，则它们的求值从左到右进行。

1）括号内的表达式；

2）从变量TIME、PI、MEM、INKEY$取得值；

3）函数（SIN，COS，LOG，EXP等）；

4）取幂（例如：2A↑3＝2＊（A↑3））

5）算式符号（＋，－）；

6）乘，除（＊，／）；

7）加，减（＋，－）；

8）比较运算（<、<=，=，>=，>，<>）；

9）逻辑运算（AND，OR，NOT）。

说明：

当同一表达式中既有算术符号又有取幂运算时，取幂运算先于符号运算。

例如：－5↑4求值得－625而不是625。

括号内的运算首先求值，在带几层括号的表达式中，计算是自最内层的括号对开始进行到最外层的括号对。

表达式求值范例：

```
7↑2+3*  √144/√81 + SIN(120 + 150)*-3
7↑2+3*  √144/√81 +     SIN(270)   *-3
7↑2+3*     12 /    9  +          -1    *-3
 49  +3*     12 /    9  +          -1    *-3
 49  +       36    /    9  +           3
 49  +            4          +           3
                 53          +           3
                                  56
```

计算范围：

函　数	动　态　范　围
$y\wedge x$ (y^x)	$-1\times10^{100} < x\log y < 100$ 〔$y=0$，$x\leqq 0$：ERROR 39 / $y=0$，$x>0$：0 / $y<0$，$x\neq$整数：ERROR 39〕 注①
SINx COSx TANx	DEG：$\lvert x\rvert < 1\times10^{10}$ RAD：$-\lvert x\rvert < \frac{\pi}{180}\times10^{10}$ GRAD：$\lvert x\rvert < \frac{10}{9}\times10^{10}$ 然而在TANx，也包括下列情形： DEG：$\lvert x\rvert = 90(2_n-1)$ RAD：$\lvert x\rvert = \frac{\pi}{2}(2_n-1)$ GRAD：$\lvert x\rvert = 100(2_n-1)$，(n为整数)
SIN^{-1}x COS^{-1}x	$-1\leqq x\leqq 1$
TAN^{-1}x	$\lvert x\rvert < 1\times10^{100}$
LNx LOGx	$1\times10^{-99}\leqq x < 1\times10^{100}$
EXPx	$-1\times10^{100} < x\leqq 230.2585092$
$\sqrt{x}$	$0\leqq x < 1\times10^{100}$

上列函数之外的其它函数仅当x取下列数值范围时方能进行计算：

$1\times10^{-99}\leqslant \lvert x\rvert < 1\times10^{100}$及0

注① $y\wedge x$计算的几种情形：

```
    0 ^ 0      ENTER ──→ ERROR 39
    0 ^ 5      ENTER ──→       0
(-4)^0.5       ENTER ──→ ERROR 39
  -4 ^0.5      ENTER ──→      -2
```

作为一个规则，在上述计算范围内，函数计算的误差是在显示数值的最低位数字（在科学记数系统的情形是尾数的最低位数字）小于±1。

X． PC—1500 与 PC—1211 指令比较表

X，1 可在PC—1500机上使用的PC—1211机的指令

1、函数

ABS
ACS
ASN
ATN
COS
DEG
DMS
EXP
INT
LOG
LN
π（PI）
SGN
SIN
$\sqrt{\ }$（平方根）
TAN
^ （指数运算）

2、语句

AREAD
USING
CLEAR
DEGREE
END
FOR—TO—STEP
GOSUB
GOTO
GRAD
IF
INPUT
LET
MEM
NEXT
PAUSE
PRINT
RADIAN
REM
RETURN
STOP
THEN
USING

3、命令

CONT
LIST
NEW
RUN

4、盒式录音机命令

CHAIN
CLOAD
CLOAD?
CSAVE
INPUT#
PRINT#

※. 2. 只在 PC－1500 机上才能使用的 PC－1500 机的命令

1、函数

AND
ASC
CHR$
INKEY$
LEFT$
LEN
MID$
NOT
OR
POINT
RIGHT$
RND
STATUS
STR$
TIME
VAL

2、语句

ARUN
BEEP（与PC－1211的有所不同）
CLS
CURSOR
GCUKSOR
GPRINT
DATA
DIM
LOCK
ON ERROR
ON GOSUB
ON GOTO
POINT
RANDOM
READ
RESTORE
TRON
TROFF
UNTOCK
WAIT

3、命令

（与 PC－1211相同）

4、盒式录音机指令

CHAIN
CLOAD
CLOAD?
CSAVE
INPUT#
MERGE
PRINT#
RMT OFF
RMT ON

5、印刷机指令

COLOR
CSIZE
GCURSOR
GLCURSOR
GPRINT
GRAPH
LCURSOR
LF
LINE
LLIST
LPRINT
RLINE
ROTATE
SORGN
TAB
TEST
TEXT

Z. 指令对照表

1. 函数

函 数	缩 写	注 释
ABS	AB.	绝对值
ACS	AC.	COS^{-1}
AND	AN.	〈表达式〉AND〈表达式〉（逻辑"与"）
ASC	ASC	转换字符为ASCII代码 ASC〈"字符"〉 〈字符变量〉
ASN	AS.	Sin^{-1}
ATN	AT.	tan^{-1}
CHR$	CH. CHR.	转换ASCII代码为字符 CHR$ 〈ASCII十进制代码或数值变量〉
COS	COS	余弦
DEG		转换度、分、秒为十进制的度
DMS	DM.	转换十进制的度为度、分、秒值
EXP	EX.	e^x
INKEY$	INK. INKE. INKEY.	〈字符变量〉=INKEY$ 若在执行INKEY$命令时按下一个键，则该ASCII字符将会被读进这个字符变量
INT		取不大于一个数的最大整数 例：INT（10/3） ENTER〈显示〉3
LEFT$	LEF. LEFT.	LEFT$（字符变量，数值表达式） 从指定字符串的左边起取字符串的指定个数的字符.

函 数	缩 写	注 释
LEN		LEN〈"字符串"〉 〈字符变量〉 测出指定字符串的字符个数，即字符串长度
LOG	LO.	$\log_{10}X$
LN		$\log_e X$
MEM	M. ME.	显示存贮器中剩余的可利用程序步数，与 STATUS 0 同.
MID$	MI. MID.	MID$（字符变量，数值表达式1，数值表达式2）：从指定的字符串中间某处开始取字符.
NOT	NO.	NOT〈表达式〉 （逻辑"非"运算）
OR	OR	〈表达式〉OR〈表达式〉 （逻辑"或"运算）
π(PI)		规定圆周率（=3.141592654）
POINT	POI. POIN.	POINT〈数值表达式〉 该函数求得表示给定列中被激活的点的图案的一个数.
RIGHT$	RI. RIG. RIGH. RIGHT.	RIGHT$ （字符变量，数值表达式） 从指定字符串的右边起取规定个数的字符.
RND	RN.	RND〈表达式〉 产生随机数的命令
SGN	SG.	符号函数
SIN	SI.	正 弦
$\sqrt{\ }$(SQR)	SQ.	平 方 根

函 数	缩 写	注 释
STATUS	STA. STAT. STATU.	STATUS 0 或 1 （0）可用程序步数 （1）已占用程序步数
STR$	STR.	STR$〈数值表达式〉 转换数值为字符串
TAN	TA.	正 切
TIME	TI. TIM.	（1）TIME=月，日，时，分，秒 TIME函数 调定月，日，时，分，秒 （2）TIME 〔显示日期和时刻〕
VAL（值）	V. VA.	VAL {"字符" / 字符变量}，转换字符串为数值
^		乘 幂

2．语句

命 令	缩 写	注 释
AREAD （auto read）	A. AR. ARE. AREA.	AREAD〈变量〉 当用定义键执行程序时，AREAD将显示的内容送入指定的变量。
ARUN （auto run）	ARU.	ARUN 当 PC－1500 的电源接通时，自动开始程序的执行
BEEP	B. BE. BEE.	BEEP〈表达式1〉,〈表达式2〉,〈表达式3〉声音命令，控制发声机构的通断，规定发音的响度和长度

命 令	缩 写	注 释
CLEAR	C L. C L E. CLEA.	清除所有变量的数据的命令.
C L S (clears)		显示器清除命令，清除显示
CURSOR	C U. C U R. CURS. CURSO.	(1) CURSOR 〈表达式〉 (0<=表达式<=25) 规定显示的开始位置. (2) CURSOR 撤消前面的规定
DATA	D A. D A T.	DATA 〈表达式〉, 〈表达式〉, … 提供用 READ 命令要读入的数据
DEGREE	D E. D E G. DEGR. DEGRE.	规 定 角 度 制 方 式 指定度(°)
D I M (dimension)	D. D I.	(1) D I M 变量名(表达式) (2) D I M 变量名(表达式)*表达式3 (3) D I M 变量名(表达式1, 表达式2) 变量名: A, B, C $, D $ 等等 ( ): 规定数组的体积和维数 表达式3: 规定字串长度
E N D	E. E N.	程 序 结 束 命 令
F O R	F. F O.	(1) F O R 数值变量=〈表达式1〉T O〈表达式2〉 开始 FOR—NEXT 循环, 与 NEXT 命令联用.

TO STEP	 S T E .	(2) F O R 数值变量=〈表达式1〉 T O 〈表达式2〉 S T E P〈表达式 3〉 表达式1：初值表达式 表达式2：终值表达式 表达式3：每次循环中变量增加的值
GCURSOR (graphic cursor)	G C . GCU. GCUR. GCURS. GCURSO.	以 点 为 单 位 规 定 显 示 位 置 GCURSOR〈表达式〉 (0<=表达式<=155) 或 (&0<=表达式<=&9 B)
GOSUB	GOS. GOSU.	GOSUB { 表 达 式 / "字符" / 字符变量 } 转子程序命令，使程序执行指针移到指定的行或标号上去。在这一点上开始的语句被作为一个子程序执行。要与RETURN命令联用。
GOTO	G. GO. GOT.	转向指令 GOTO { 表 达 式 / "字符" / 字符变量 } 使程序执行指针转移到指定的行或标号。
GPRINT (graphic print)	G P . GPR. GPRI. GPRIN.	在 打 印 机 上 印 出 显 示 器 的 内 容 (1) GPRINT "OO OO OO …" (" " 号内的是十六进制数) (2) GPRINT O; O; … (3) GPRINT &O; &O; …

GRAD	G R. G R A.	指定角的计量方式 指定百分度制 〔g〕
I F		(1) I F 〈条件表达式〉〈执行的命令〉 (2) I F 〈算术表达式〉〈执行的命令〉 求出给定条件表达式的值，转去执行下一行或执行本行的命令。
INPUT	I. I N. I NP. I NPU.	(1) I NPUT〈变量〉，〈变量〉，… (2) I NPUT 〈"字符"〉,〈变量〉,〈"字符"〉,〈变量〉, (3) I NPUT 〈"字符"〉;〈变量〉,〈"字符"〉;〈变量〉
L E T	L E.	(1) L E T〈数值变量〉=〈表达式〉 (2) L E T〈字符变量〉=〈"字符"〉 (3) L E T〈字符变量〉=〈字符变量〉 L E T跟在I F命令之后，在其它情形可省略。
LOCK	LOC.	锁定计算机所处的方式。
NEXT	N. N E. N E X.	NEXT 〈数值变量〉 出现在 FOR—NEXT 语句末尾，表示 FOR—NEXT 循环终止。 数值变量必须与 FOR 命令后的数值变量相同。
O N ERROR	O. ER. ERR. ERRO.	O N ERROR GOTO 〈表达式〉 错误捕捉命令
O N GOSUB	O. GOS. GOSU.	O N〈表达式〉GOSUB 〈表达式1〉，〈表达式2〉，〈表达式3〉 转子程序命令，由表达式的值决定转移到的位置（表达式1，表达式2，或表达式3）

ON GOTO	O. G. GO. GOT.	ON〈表达式〉GOTO〈表达式1〉,〈表达式2〉, 〈表达式3〉 转移指令，由 ON 后面的表达式的值决定要转移到的位置（表达式1，表达式2，或表达式3）
PAUSE	PA. PAU. PAUS.	命令形式与 PRINT 命令相同，显示指定内容约 0.85秒，然后执行程序。
POINT	POI. POIN.	POINT〈表达式〉（0<=表达式<=155） （&0<=表达式<=&9B） 读出显示在指定点上的点的图案的信息。 例： A=POINT 56
PRINT	P. PR. PRI. PRIN.	(1) PRINT {表达式 / "字符" / 字符变量} ; (2) PRINT {表达式 / "字符" / 字符变量} , {表达式 / "字符" / 字符变量} (3) PRINT {表达式 / "字符" / 字符变量} ; {表达式 / "字符" / 字符变量} ;…; {表达式 / "字符" / 字符变量}
RADIAN	RAD. RADI. RADIA.	指定角的计量方式 指定弧度制〔rad〕。
RANDOM	RA. RAN. RAND. RANDO.	在使用 RND 命令之前，设置随机数的种子。

READ	REA.	READ〈变量〉,〈变量〉,… 数据读入指令，从 DATA 语句中把数据送入指定的变量之中。
REM (remark)		REM… 文件注释… 规定不被执行的注释。
RESTORE	RES. REST. RESTO. RESTOR.	(1) RESTORE〈表达式〉 改变 READ 命令读入数据的顺序。 (2) RESTORE 在第一个 DATA 语句的开头开始读数据。
RETURN	RE. RET. RETU. RETUR.	回到调用该子程序的 GOSUB 语句后面继续执行。
STOP	S. ST. STO.	命令程序执行中止。
THEN	T. TH. THE.	定义IF语句的执行内容的指令。 对于IF语句，转移指令只能被定义为执行命令。 THEN {表达式 / "字符" / 字符变量}
TRON (trace on)	TR. TRO.	规定执行跟踪调试的方式。
TROFF (trace off)	TROF	撤消执行跟踪调试的方式

UNLOCK	UN. UNL. UNLO. UNLOC.	撤消LOCK（锁定）方式.
USING	U. US. USI. USIN.	（1）USING "＃＃＃·＃＃＃＃＃＾" （2）USING "&&&……&&&" （3）PRINT USING "格式串"；…… （4）USING （5）PRINT USING; 根据表达式的值规定显示的格式.
WAIT	W. WA. WAI.	WAIT〈表达式〉 （0<＝表达式<＝65535） 规定用PRINT命令时显示持续时间. WAIT 后面不带参数时，撤消前面的规定， （持续时间＝无穷）.

3．命令

命　令	缩　写	注　　释
CONT	C. CO. CON.	重新开始执行被暂时中止的程序. 在 RUN 方式有效.
LIST	L. LI. LIS.	执行程序列表的命令. 在 PRO 方式有效.
NEW		（1）NEW　　（2）NEW　0 在PRO方式消除程序存贮器内全部程序和变量.
RUN	R. RU.	（1）RUN （2）RUN〈表达式〉 （3）RUN〈"字符"〉 〈字符变量〉 在 RUN 方式有效.

4．盒式磁带录音机命令

命　令	缩　写	注　释
CHAIN	CHA. CHAI.	传输命令．当用在程序内部时，从磁带读入程序(传输)并执行该程序． (1) CHAIN "文件名" (CHAIN—1 "文件名") (2) CHAIN "文件名"，〈表达式〉 (CHAIN—1 "文件名"，〈表达式〉) (3)、(1)和(2)的"文件名"省略形式．
CLOAD (cassete load) (load)	CLO. CLOA.	传输命令．这个命令从磁带中把程序或保留内容传送到计算机存贮器中． (1) CLOAD (CLOAD—1) (2) CLOAD "文件名" (CLOAD—1 "文件名")
CLOAD？ (cassette load？)	CLO.？ CLOA.？	比较命令．这个命令将存贮器中的程序或保留内容与记录在磁带上的内容进行比较． (1) CLOAD？ (CLOAD？—1) (2) CLOAD？"文件名" (CLOAD？—1 "文件名")
CSAVE (cassette save)	CS. CSA. CSAV.	该命令将程序存贮器和保留存贮器的内容记录在磁带上． (1) CSAVE (CSAVE—1) (2) CSAVE "文件名" (CSAVE—1 "文件名")

命　令	缩　写	注　　释
INPUT#	I.# IN.# INP.# INPU.#	为数据传输命令。该命令将记录在磁带上的数据传送到指定的变量中。 所取的格式与 PRINT# 命令的相同。
MERGE	MER. MERG.	为程序传输命令。该命令将磁带上的程序传输到计算机内。所取的命令形式与CLOAD 命令的相同。 在这个命令中，先前记录在磁带上的程序将照样保留，而新读入的程序会被加上给计算机内已有的程序。
PRINT#	P.# PR.# PRI.# PRIN.#	数据录制命令。该命令将存贮在 PC—1500 机上的数据记录到磁带上。 （1）PRINT#变量名，变量名…… (PRINT#—1，变量名，变量名，……) （2）PRINT#“文件名”；变量名，…… (PRINT#—1，“文件名”；变量名，……)
RMT OFF (remote off)	RM.OF. RMTOF.	该命令撤消 REM 1 端的远程控制功能（对于第二台录音机）。
RMT ON (remote on)	RM.O. RMTO.	该命令重新设定 REM 1 端的远程控制功能（对于第二台磁带录音机）。

5．印刷机命令

命　令	编　号	注　　释
COLOR	COL. COLO.	规定打印的字符的颜色代号。 COLO R〈表达式〉。（0<=表达式<=3）
CSIZE (character size)	CSI. CSIZ.	规定要打印的字符的大小。 CSIZE〈表达式〉　（1<=表达式<=9）
GLCURSOR (graphic line cursor)	GL. GLC. GLCU. GLCUR. GLCURS. GLCURSO.	该命令只在GRAPH方式有效，它命令把打印笔从起点位置移到座标为（X，Y）的点上。 GLCURSOR（表达式1，表达式2）
GRAPH (graphic)	GRAP.	这个方式用于绘制曲线图和图表。
LCURSOR (line cursor)	LCU. LCUR. LCURS. LCURSO.	移动打印笔到所要求的位置。
LF (line feed)		执行走纸，走纸行数由表达式规定。只对TEXT方式有效。 LF〈表达式〉
LINE	LIN.	划线命令。 仅对GRAPH方式有效。 (1) LINE(表达式1,表达式2)－(表达式3,表达式4) (2)LINE(表达式1,表达式2)－(表达式3,表达式4),表达式5，表达式6

		(3) LINE(表达式1，表达式2)－(表达式3,表达式4), 表达式5，表达式6，B 表达式5：规定线型 表达式6：规定颜色 B：规定划一方框 (4) LINE (表达式1,表达式2)－(表达式3,表达式4) ——……——(表达式11，表达式12)
LLIST	LL. LLI. LLIS.	程序列表印出.
LPRINT	LP. LPR. LPRI. LPRIN.	打印指定的内容. 仅对TEXT方式有效. 所取命令形式与 PRINT 命令相同.
RLINE (relative line)	RL. RLI. RLIN.	命令用当前笔的位置作为起点划线. 只在 GRAPH 方式中有效. 命令形式同 LINE 命令.
ROTATE (rotate)	RO. ROT. ROTA. ROTAT.	规定要打印的字符的方向(打印方向). 只在 GRAPH 方式中有效. ROTATE〈表达式〉 (0<=表达式<=3)
SORGN (set origin)	SO. SOR. SORG.	该命令规定当前打印笔的位置作为新的起点(座标原点). 只在 GRAPH 方式中有效.

TAB		规定打印笔的位置，仅在 TEXT 方式中有效。 (1) TAB〈表达式〉 (2) LPRINT TAB〈表达式〉；……
TEST	TE. TES.	颜色检验。当执行该命令时，会用每一种色笔画出一个5 mm×5 mm的方框。
TEXT	TEX.	规定 TEXT 方式。这个方式打印字符和数字。

印刷机错误和校验

假如印刷机工作不正常，计算机上便指示出错信息（错误和校核信息）。下面叙述各种可能的错误。

（1） 当打开计算机开关后就显示出 CHECK 6 信息，便是 CE—150电力不足或 印刷机有了故障。当发生这种情况时，不要试图开动印刷机（包括纸带馈送和换笔）。要在再用之前给印刷机重新充电。

（2） 假如显示 ERROR 78，79或80的话，务需用 OFF 键关闭计算机电源。若机器仍开着，就会发生不正确的操作，例如，执行 BEEP 指令时没有声音，或显示功能不正确。

错误信息及其发生原因和处理方法见下页对照表。

错误信息	原　　因	处理方法
：CHECK 6 NEW 0 ？：CHECK6 （6是CE—150的代号）	（1）内部可重新充电电池电压已降至工作电压之下。	给电池充电
	（2）在打印笔位置检出颜色信号不能起作用。若色笔或任何其它物体降落进印刷机内，打印笔支架就不能旋转或移动。	按下计算机的OFF键以关闭计算机电源，取出阻碍打印头的任何物品。
ERROR 80 （电池电压低下的检测）	内部可重新充电电池的电压太低。	给电池充电，充电后按OFF键和ON键解除对印刷机的闭锁。
ERROR 79	在色笔位置检出颜色信号不能起作用。	按下计算机的OFF键关闭计算机电源，取出阻碍打印头的任何物品。
ERROR 78	（1）印刷机处于换笔方式。	按计算机的CL键的同时按印刷机的↑键，清除换笔方式。
	（2）印刷机被低压电池检测（ERROR 80）闭锁后仍处于闭锁状态。	电池充电后，按OFF键和ON键解除印刷机的闭锁。
	（3）颜色信号不起作用	如ERROR 79的情形处理。

假如在校验了上述原因之后还是显示CHECK 6或ERROR 78的信息，说明CE—150可能出故障了，请与指定的SHARP服务中心联系。

SHARP CE—151、CE—155 存贮模块使用说明书

一、使用注意：

1）在将模块装入计算机或取出之前，需取出电池并按ALL RESET（全部复位）开关；

2）模块可因静电放电而损坏，故切不可触摸模块的插头；

3）当从计算机取出模块时，要立即将模块罩装到模块上并将模块放回其保护盒内；

4）不要将模块置于阳光下曝晒；

5）不要将模块靠近热源如暖气片等。

二、模块的安放

（1） 按OFF键关闭计算机电源，取出电池；

（2） 在按住ON键的同时，按计算机背面的ALL RESET（全部复位）开关约15秒钟；

（3） 揭开模块盖；

（4） 将模块从保护盒中取出，并从模块上拔下插头盖；

（5） 将模块放置在计算机的模块匣内，按箭头所示方向推模块，使模块插头与模块匣内的插座连接（图略）；

（6） 重新装上模块盖及电池；

（7） 打开计算机开关并按CL NEW 0 和ENTER键以清除存贮的内容。

注意：

（1） 为了避免由静电引起的模块的可能损害，在装卸模块之前，接触一下金属物体（如门把手、台灯等）。静电集聚多发生于空气自然湿度低的冷天（或气候非常干燥的地区）。

（2） 在从计算机上取出或装入模块之前，一定要取出电池。

三、取出模块

（1） 按 OFF 键关断计算机电源并取出电池；

（2） 在按住 ON 键的同时，按住计算机背面的 ALL RESET（全部复位）开关，大约15秒钟；

（3） 揭开计算机底部的模块盖板；

（4） 小心地从计算机中取出模块；

（5） 立即将插头盖装到模块上，并将模块放回保护盒内；

（6） 盖好模块盖板，装上电池；

（7） 打开计算机开关，并按 CL NEW 0 及 ENTER 键。

《PC—1500 应用手册》目录