APPLE II微型计算机 LOGO语言技术手册

Harold Abelson
Leigh Klotz, Jr
田良木 译

中国儿童少年活动中心印

美国麻省理工学院LOGO课题组

一九八二版

内 容 目 录

7．1．用Logo系统作文本编辑

7．1．1．文件打印

7．2．自起动问题

7．3．打印磁盘文件

7．4．各种系统参数

7．5．储存组织表

TERRAPIN LOGO 第二版本

主要变更内容说明

命令在编辑过程中取消一行，理改用＜CTRL＞X 代替＜CTRL＞K。被删除最近的一行，定 拉回到编辑程序中，用＜CTRL＞Y。

Apple ‖ e 计算机＜DELETE＞键，可用为反向清除一个字符（＜ESC＞ 键一样）。四个箭头键用作编辑程序时，移动光标.

增加六个新的命令：MEMBER?，EMPTY?，ITEM，COUNT，LOCAL，SETDISK。

MEMBER? 要求两个输入，只有在第一输入是第二个输入的一项为“真”时，才有输出。

EMPTY? 只有在输入是一个空字或是一个空白表格为“真”时，才有输出。

ITEM 要求两个输入，一个是数字与一个表或字，输出第二项输入的第 *n* 项。

COUNT 执行输入一个字或是一个表的项数。

LOCAL 指在标题行内只作局部改变，并不消取，见计算与字，表一节内容。

SETDISK 要求两个输入，一个是驱动器号，一个是主机内插板槽号，因为所有文件命令都与驱动器有关。只有 SETDISK 1 6 例外。

把确实没用的东西加到“程序垃圾箱”（GCTMA）中，其结果必须全出现错误信息No STORAGE LEFT! 白浪费时间没有必要。

在用过REPEAT，RUN 和 DEFING后，以立即高式命令再调出前一行的内容，要用＜CTRL＞P。

用 STARTUP 实现文件的自起动，使用较为方便。

PO, SAVE, EDIT和ERASE要求输入一个过程列表，PSAVE已经废弃不用，现在要打＜SAVE " FILE NAME〔PROC1 PROC2 PROC3 … 〕＞。

使用操作系统的指令MON, NOMON和 VERIFY，可选择句子长短，不长的字符串可用空格分隔。例如，　打DOS〔BLOADPICTURE〕就是一个错误，在BLOAD 之后加一个空格便可以改正过来了，DOS指令地址必须用＋进位或＋六进位形式表示。

通过DOS可以阅读Apple DOS文本文件，使用标准命令READ．既可阅读特殊的Apple(LCS1)Logo文件，也可以阅读Terrapin Logo（当然，程序过程语法有问题，就不能执行这些文件）。

前 言

这是一本Apple ‖ 计算机Logo系统的参考书，它要求Apple ‖计算机在执行Logo语言时，机器配置，必需一台软盘驱动器和48K的储存能力，而且还要有一张16K的储存扩充板。系统包括Logo语言之外，同时还有综合性交互式的“乌龟”绘图，屏幕编辑和软盘文件系统1。这本手册帮助读者进一步熟习Logo语言与编写Logo程序，并且提供有关Apple ‖ Logo系统的技术资料，同时附有该系统的基本命令表。

Apple Logo以及Terrapin Logo都是由美国麻省理工学院Logo语言组开发的，这项工作曾得到美国科学基金会的支持和许可才公开发行的。作者Stephen Hain和Leigh Klotz，由Patrick Sobalvarro负责文字编辑，绘图与文件系统。这项业务上基本上是在美国国家基金会支持下，由麻省理工学院Logo课题的成员共同合作完成的。并且在Logo语言原型机试验的基础上，在德州仪器公司赞助下开发应用在微型计算机上，此项任务曾得到Harold Abelson 教授的监督与指导。

译者注：因为这本手册是应Terrapin 公司之邀而编写的，所以重叙述的是Terrapin Logo 的内容。通过阅读这本技术手册一方面可以解决Apple Logo的许多操作问题，又可以了解两种版本的不同之处，对普及推广这两种版本有一定帮助，现根据一些同志的要求译出，谨供参考。

注1：目前Apple Logo和Terrapin Logo都是利用6502机器语言写的；Leigh Klotz 正在利用6502汇编语言来写Logo

程序，以便进一步改进及扩充Apple 系统，并考虑设计在新的外围设备情况的接口等。

第 一 章

使用Logo语言的准备工作

这一章主要讨论执行Logo语言计算机的硬件配套和Logo系统要求计算机键盘操作特殊之处，同时还说明为何在Apple计算机与调Logo和起动文件。

1．1．机器与软盘

第一版本的Logo语言要求Apple ‖或Apple ‖ Plus的机器配置，包括一台软盘驱动器，48K的贮存能力及一张16K的储存扩充板，Apple计算机公司叫“语言板”，Microsoft Consumer Products公司叫“RAM板”。这个系统还应包括两张由DOS 3.3操作系统支持下的软盘，开始用Logo基本语言盘，另一张是Logo实用盘，后者包括各种表 的Logo程序在内，其内容见第四章说明。

1．2．键盘

Logo语言与其它语言如BASIC语言和Pascal语言，使用Apple计算机键盘的方法略有不同。现说明如下：

SHIFT 转换键

Logo语言和大多数Apple ‖ 系统一样，只用大写字母，打大写字母不用SHIFT-键。用SHIFT转换键，主要是一个键可以显示出两种符号，例如圆括号“（”要用SHIFT—9键。

括号键

Logo语言用开，是方括号“〔”和“〕”。但它未标志在Apple机任何键上，当用Logo语言时，则必须使用复合键SHIFT—N和SHIFT—M。建议用一个明显的括号标志或符号贴在N和M键上，以便经常提醒不会用错。

CTRL控制键

CTRL键在键盘的左下方，用以输入“控制 control 字符”。CTRL像SHIFT 键用法一样复合使用，如“ Control G ” 要按住CTRL键，再按G字符键。在这本手册内，始终是把CTRL放在前面，即“ CTRL—G ”。

箭头键

在Apple 计算机键盘的右侧有两个标志着向左，向右的箭头键，箭头键主要用在编辑程序时，向右或向左移动光标。

ESC 清除键

在键盘的左上方有一个ESC 键，也是为编辑程序用的。当按ESC 时，就把打出来的字符清除了。也可以在ESC 键上写上“ D E L E T E ”清除键。

REPEAT 重复键

按住一字符键同时按下REPEAT 键时，也就是重复一大串的一种字符。

在编辑程序中，偶尔也把REPEAT 键和箭头键复合使用。

RESET 复位键

在使用Logo 语言时，无论在那种情况都不要按RESET 键。因为Apple RESET 的复位功能不逊于像Logo 等各种对话式的程序语言。在Logo 系统中使用RESET 键操作会使计算机中途失败。过去曾有Apple 机的用户，以巧妙的方法处理一下，使RESET 键不起作用，这当然要看打字员的经验了。还有一种解决的方法，将RESET 键头暂时撬出来，用一块塑料挡板封粘。新型Apple 计算机专门规定，必须使用CTRL—RESET 复合键时才能达到复位的作用。

致于使用那种方法，都要避免用复位键，一旦误用，则必须重新

调用系统文件，见1．3节所述。

1．3．调用和起动Logo程序

起动Logo的方法有两种，一种是用Apple 计算机只读存储器自动起动的方式。另一种不用自动起动。

1．3．1．用Apple 计算机只读存储器自动起动：

·关上计算机，将Logo软盘放入驱动器内，并接通电源。

·屏幕上出现“] ”，并有一句话LOading, PleasE Wait …

大约等待30秒钟，Logo语言显示起始信息：

```
   THE TERRAPIN LOGO LANGUAGE
WRITTEN BY L·KLOTZ, P·SOPALUARRO AND
S·HAIN UNDER THE SUPERVISION OF H·ABELSON。
   COPYRIGHT  (C)  1981 MIT
   COPYRIGHT  (C)  1982, 1983, 1984
      TERRAPIN, INC。
   VERSION 2, 0
   WELCOME to LOGO
   ?■
```

或是

```
   PRESS THE ReTURN KEY TO BEGIN
   IF YOU HAVR YOUR OWN FILE DISKETTE,
   INSERT IT NOW, THEN PRESS RETURN
   ■

   (C) LOGO COMPUTER SYSTEMS 1982—83—84
```

按RETURN 后出现：

```
WELCOME TO LOGO                    A2. 0

?■
```

注意在屏幕上，在起始行前边应有一个？号，从光标位置开始就可以写Logo命令。

·从驱动器中取出的软盘，要放在一个安全的地方。

1．3．2．用Apple 计算机不用ROM 自动起动：

·将Logo语言盘放入驱动器内，并接通电源。

·打6及按CTRL—P，按RETURN 键。

1．4．Logo语言比较完善，而且是一种Apple ‖ 计算机上可以扩充的语言，V1．1版本之后，都可以在执行Logo语言中计算机停止下来的时候，在监视器上清楚地显示出故障的征兆。

```
CONGRATULATIONS! YOU FOUND A BUG!
TYPE 300. 311 <RETURN> AND WRITE DOWN
THE RESULT.THGN TYPE CTRL—Y <RETURN>。
```

当发现故障时，按照Logo的指引去作，在打300. 311 <RETURN> 之后，写下结果，再按CTRL—Y <RETURN>即可。

在大多数情况下在按CTRL—Y 和RETURN 之后，按照Apple计算机监视器显示内容志作，Logo系统都可以达到继续工作的目的。

最保险的办法是利用文件暂存，从软盘上再调Logo语言时，回头再阅读程序过程。对于有些故障，使用CTRL—Y 方法也不能工作，要按照1．3的顺序，重新起动。

第 二 章

使用Logo系统

Logo系统包括Logo语言全部解释程序，一套为编辑过程定义的文本编辑系统，和综合“乌龟绘图”或称“搭积木拼图”系统。本章主要着重说明不同功能的相互影响。

2．1．利用屏幕的方式

Logo系统使用显示屏幕有三种不同的方法或“方式。”

2．1．1．非绘图方式（NODRAW MODE）

这是Logo系统开始工作的一种方式。Logo语言在用户未输入命令前，在屏幕上显示一个问号，跟着出现一个闪烁的方块，称“光标”。打完命令一行之后，按RETURN 键，Logo语言便执行这一过程，并在显示屏幕幕反映出来。

无论什么时候光标总是可以看见的一个闪烁的方块，Logo语言等待在光标处打字，它并不涉及要作什么操作。

Logo系统还包括一个灵活的行编辑的功能，当用DRAW绘图方式或NODRAW 非绘图方式时，用于改正命令行内打字的错误。改错的编辑命令键有：ESC，箭头键，CTRL—A，CTRL—D，CTRL—E，CTRL—G，CTRL—K，CTRL—P。

2．1．2．编辑方式（EDIT MODE）

在执行Logo编辑方式的命令之前，要有TO或EDIT 引导，例如：用Logo语言打

```
TO POLY : SIDE : ANGLE
```

按RETURN 键之后，系统将按照屏幕打出文本的一行进入屏幕编辑状态。

在屏幕的底部以反色字母显示“ EDIT :Ctrl—C to define,

Ctrl—G to *abort*”，表示Logo已进入编辑状态。

在编写过程或编辑过程文本的时候，可以用2．2．2节中全部编辑操作，用CTRL—C 复合键表示由定义过程的文本状态，脱离编辑方式，而进入非绘图状态。用CTRL—G复合键处理编辑故障，由没有任何过程被定义的情况，脱离编辑状态而进入非绘图方式。开始编辑一个过程，起动之后什么也不改变，用CTRL—G键，过程的编辑内容也不变。

在编辑方式中，RETURN键还可以把光标移到下一行上去。在EDIT方式中，使用CTRL—C 以鉴别Logo的编辑内容起隔离的作用，正像RETURN 在DRAW和NODRAW 方式中，用以鉴别刚刚打出的内容一样。见2．2节。

编辑方式用EDIT定义程序过程是最方便的（或简化为ED）。在DRAW方式中，使用EDIT编辑定义过程或NODRAW 方式使用PO 一样，用EDIT而无输入，执行编辑方式其编辑内容互相隔离。例如在READ或SAVE之后，所阅读或存放的每一项内容在编辑过程中都是互相隔离的。在定义程序过程中使用CTRL—G 处理故障时，所编辑的内容也是隔离的。可以重新再打EDIT但不能再用同一个过程名。因为在EDIT之后要跟着有一个过程名，正像已定义的过程一样。这些虽然很繁，但过程是一目了然的。

2．1．3．绘图方式（DRAW MODE）

绘图方式指的是在屏幕上乌龟绘图，或搭积木拼图的方法，在非绘图方式一段时间内，要执行任何一条乌龟的命令，在执行这条命令之前系统将进入绘图状态，而非绘图命令（ND）必然脱离绘图状态而进入非绘图方式。实际上这是两种不同类型的绘图方式。

切分屏幕显示方式是绘图状态经常采用的方法。屏幕底部四行显

示文本，上部屏幕显示乌龟运动的图形。乌龟运动实际上延伸到底部只是被四行文本遮盖起来了。整个屏幕显示方式文本显示区消失了，整个屏幕可见的是全部乌龟运动范围。虽然打了命令，但是看不见。当系统需要显示错误信息时，首先要进入切分屏幕状态，错误信息才可以看得见。

用控制字符复合键 CTRL—F 和 CTRL—S 实现切分屏幕和整个屏幕显示方式来回的转换。在切分屏幕状态时按 CTRL—F，系统进入整个屏幕显示方式。反过来按 CTRL—S 系统进入切分屏幕显示状态。当然有时也可用切分屏幕（SPLITSCREEN）和整个屏幕（FULLSCREEN）的命令，通过计算机程序控制两者之间的转换。

在绘图方式工作时，Logo 语言在屏幕底部四行显示文本，但在错误信息有时比四行还要长的时候，就没有办法把内容全部显示出来。所以在绘图状态可用 CTRL—T 使乌龟运动的图形消失，而整个屏幕进入显示文本的状态，就像非绘图工作方式一样。计算机却仍然处于绘图状态。尽管看不见绘图的图形，但乌龟运动的命令已经在执行了。所以在绘图状态时，由于错误信息比四行还要长的情况之下，CTRL—T 命令特别有用。实际上和 TEXTSCREEN 文本显示状态一样。在使用 CTRL—T 之后，还想要看见图形，要按 CTRL—F 则返回整个屏幕显示图形方式，或按 CTRL—S 返回切分屏幕的状态。

屏幕文本显示 TEXTSCREEN 与非绘图方式（NODRAW）不同。NODRAW 清除了文本屏幕，又清除了绘图屏幕，又使全部绘图参数失效（包括笔头颜色，乌龟的可视性，下笔抬笔，底色，卷绕方式等）。

下边列出一个与编辑功能无关的控制键使用表，它们无论对绘图方式或是非绘图方式都可以用。它们脱离开编辑方式，有其使用的特殊用途。

非编辑方式控制字符复合键表

CTRL—F 进入绘图方式，使整个屏幕显示图形。

CTRL—G 进入编辑方式，但脱离不作编辑文本处理的编辑状态。

CTRL—S 进入绘图方式，但屏幕为文本与图形混合显示状态。

CTRL—T 进入绘图方式，但可以整个屏幕显示文本。

CTRL—W 停止执行程序。重复按CTRL—W 可以使Logo在显示第二行之后再停止（若显示一个列表，就在下一个列表单元后停止）。除去CTRL—W 或CTRL—G 之外，按任何一个控制字符键，计算机都将恢复正常的工作状态。同时按CTRL—W 和REPEAT 键有“慢动”的效果，见有关TRACE 的使用说明。

CTRL—Z 使Logo暂停。但只要打入的内容是已进入计算机中过程的一行，Logo便执行它。打CO 或CONTINUE 则Logo继续工作。

CTRL—SHIFT—M 恢复屏幕输入状态见OUTDEV 3.8节。

CTRL—SHIFT—P 在字符下面划线。作为一种正规的字符打印显示都可用于以上三种显示方式。

2.2.编辑功能

Logo系统包括一个完整的综合屏幕编辑系统，和一个与之共存的行编辑系统。在EDIT 方式中，用定义Logo 过程实现屏幕的编辑功能，在DRAW和NODRAW 方式中通过执行Logo 命令而实现行编辑功能。

2.2.1.行编辑

当打出一行字符的时候，可能会忽略Logo语言的行编辑功能，如果打错一个字符，可用ESCAPE 键把它清除掉。如果在一行的行

头忘记一个字，可用CTRL—A 移动光标到要加字的地方再打进去。其它字符就要向右推；也可能在字里行间丢掉什么或是多写了什么都可用这种方法加以修改。如果要在行内任意部位插入字符，最简单的方法是用箭头键移动光标到修正处再打上去。用CTRL—E 键把光标移到行尾。用CTRL—D 键，清除光标后边的一个字符。

光标移向行尾，按RETURN 键，Logo将作出相应反映。但不必光标非移到行尾末端不可，只要这一行语句的字符齐全，按RETURN键后Logo就可以读。用CTRL—K 从光标到行尾清除所有字符。

打完一行还可用Logo语言卷挠的功能，显示出下一行。如果一行超过四十个字符，行又不能延长，就要用编辑命令移到下一行。

无论是绘图方式或非绘图方式，用CTRL—P 键可以把打字的一行插到另一行当中去，但注意，当执行RUN, REPEAT, DEFINE 和所有文件命令后，像SAVE一样，Logo就会忘记最后一行所打的东西。

2．2．2．屏幕编辑系统

在EDIT, ED或TO之后跟着一个过程名，用以定义程序过程，Logo语言即处于屏幕编辑状态。

只有在Logo处于编辑状态EDIT MODE 方式所打的字符才能在屏幕上出现，按RETURN 键后光标移到下一行（若光标不在行尾，按RETURN 键后，将由一行切分为两行），但不是使Logo执行这一行。

在屏幕编辑过程中，当然也可用其它命令把光标从一行移到另一行上去。按CTRL—N 键，光标移到下一行；打CTRL—P 光标移到前一行。

利用编辑方法可将一行的长度隔离为适合的长度。当一行超过四十

个字的长度时，屏幕将进行卷卷运动。在一行最后一列位置出现（“！”）时，表现一行的字符继续在下一行写，这个警叹号虽不是要打入过程的一部分，它仅仅表示在这个符号处并不是行的终点。

尽管Logo程序过程很少超过几行的语句长，在一页屏幕上一般不必限制编辑文本的长短。如果所打的文本超过一页，系统就会自动将打字的一行从屏幕底部转到屏幕中间一行；按CTRL—P 键或CTRL—N 键，屏幕将由顶边或底边出现下一页，按CTRL—F 键，包括编辑方式在内也转到下一页的文本当中去。按CTRL—B 将回到前一页去。如果是在第一页上，按CTRL—B 将转到顶头位置。同样，如果是最后一页，按CTRL—F，将转到尾端的位置。当所编辑的文本超过一页时，用CTRL—L 命令这一行转到屏幕下一页的中间位置。

按CTRL—C 键，将脱离编辑状态，但打定义过程。按CTRL—G，脱离编辑状态而且不定义过程。按CTRL—G 之后，将返回ED 或EDiT 编辑状态及所有文本，而不是绘图状态。

编辑过程中所打的文本并不是一个过程，而是任意的Logo命令，但也不是绘图和文件命令。见7。1节如何使用Logo编辑文本，在软盘上储存，打印显示等。

总结可供使用的编辑命令有：

键盘编辑命令

ESC　　清除光标左边的一个字符，并且光标向左移一个空格。

箭头键　　每按一次，光标向左或向右移动一个字符的位置，但并不清除任何字符。

CTRL—A　　移动光标到行头处。这个命令所以用CTRL—A 是因为这两个键都在键盘的左边，同一列，A是第一个字母。

CTRL—B 当编辑的文本超过屏幕范围的时候，按CTRL—B 移动光标，整个屏幕显示文本向后退。或到隔断处始端，但光标之前文本较短。

CTRL—C 脱离编辑状态。被编辑的文本是经过定义的过程程序。

CTRL—D 清除光标位置的字符，光标闪烁。

CTRL—E 移动光标至行尾。

CTRL—F 当编辑的文本超过屏幕范围的时候，按CTRL—F 移动光标，整个屏幕显示文本向前进。或到隔断处尾端，但光标之后的文本较短。

CTRL—G 进入编辑方式，但脱离不作文本编辑处理的编辑状态，在所有方式中，停止执行和返回顶头的位置。

CTRL—K 清除光标右边所有的字符，就是除去一行文本。

CTRL—L 进入编辑方式，包括光标在内文本行内滚动至屏幕中心位置。

CTRL—N 移动光标到下一行。

CTRL—O 在光标位置开辟新的一行，等于按CTRL—P 之后再按RETURN。

CTRL—P 在编辑方式中，按CTRL—P 键，光标移到以前的一行去。在绘图或非绘图状态，恢复以前输入的一行，并可实现编辑和反复执行的作用。在绘图或非绘图状态时，REPEAT, RUN, DEFINE 和所有文件系统的命令都会引起Logo 忘记在先的一行。另外对行长也有限制，Logo程序也记不住以前那些很长的行。

2．3．使用Apple 计算机的外围设备

Logo语言一般的输入输出操作，是用Apple 计算机键盘，屏

幕和磁盘驱动器实现的。当然还有Apple 计算机的四支游戏棒也可作为阅读命令的输入设备（详见PADDLE 和PADDLEBUTTON 一节）。另外Logo程序还配有OUTDEV 即处理屏幕输出设备的功能，这个OUTDEV 输出命令只要求输入一个输出设备接口板位置的槽号。同时还可能用OUTDEV 表明一个用户提供的汇编语言程序，此处可称为标准字符输出程序。

OUTDEV 命令可以保证由Apple 监视器与插在指定接口板位置槽内的任何其它输出设备正常工作。它不像OUTDEV 不直接打给所选择的设备，Logo屏幕编辑和行编的功能都与之无关。当OUTDEV 输入为0时，则输出设备对屏幕复位，即置0。

按CTRL—SHIFT—M 键，改变屏幕输出，它与执行OUTDEV 0 等同，即便是Logo要打印机在打印过程中，也立即生效。但Apple 计算机一般情况并非如此，它坚持等待接收到一个规定的字符之后才能执行。

2．3．1．用打印机打印程序过程

当插入打印机接口板槽1号位时，依如下举例顺序工作，

将程序过程打印在一张纸上，称CIRCLE，则要打：

```
OUTDEV 1
PRINTOUT CIRCLE
OUTDEV 0
```

或是打

```
HPO "CIRCLE
```

随后要定义HPO，如：

```
TO  HPO :PROCEDURENAME
 OUTDEV 1
```

```
RUN L1ST "PR1NTOUT :PROCEDURENAME
OUTDEV 0
END
```

HPO是硬拷贝打印的意思（HARDCOPYPR1NTOUT），HPO "ALL要求列出整个过程和过程名的清单。下边的过程中过程表有一个输入，按列表内容规定在打印机上打印出来，但所用的是不包括列表中第一项的其余列表内容。

```
TO HPL :L1ST
 1F :L1ST=[] STOP
 HPO F1RST :L1ST
 HPL BF :L1ST
END
```

HPO和HPL已经定义，命令一行如打印出B1RD, HEAD, W1NGS, TA1L和LEGS过程，则令：

```
HPL [B1RD HEAD W1NGS TA1L LEGS]
```

如打印所有过程列表，而不是过程名列表，则打

```
HPO "PROCEDURES
```

2。3。2。图形打印

许多Apple计算机配套的打印机，都可以打印图形，在利用打印机打印一份屏幕乌龟图形之前，必须了解如下几个问题。

由于监视器或电视机不同，打印机的长宽比也不同，屏幕上的图形是一个方形，打印在纸上成了矩形，所以要想打印一张图纸或画面，首先必须了解所使用的打印机的长宽比。可通过调节在监视器与打印机之间长宽比例关系，得到满意的结果。如果忽视这一点则效果必然较差。详见。ASPECT 一节详细说明。

在磁盘上储存图形，之后再打印出来，包括屏幕图形硬拷贝，是最常用的方法。SAVEPICT 命令把图形储存在磁盘上，用SAVEPICT储存图形像Apple DOS 二进制文件那样，“。PICT” 不同于其它文件，在储存器中储存的屏幕乌龟图形是一种高清晰度的画面。

对各种不同的打印机，调用图形文件并打印出来有各种可买到的程序，其中Data Transforms 公司的GRAPHTRIX 软件包是一种被公认为使用方便的计算机储存程序。

Orange MicroSystems公司的Grappler 接口板

这种接口板可以将Apple II 计算机与许多种广用的打印机相连结，不同的打印机其接口也不同。当接口板插在一号槽位时，如下程序过程可以适用各种打印机，把图形打印出来：

```
TO  HC          ;hardcopy ot graphico screeu
 OUTDEV1
 CPRINT1
 OUTDEVO
END
```

Grappler 接口板打印图形有各种功能可供选择，包括反转颜色，旋转和放大。这些工作方式的字符都要排列在“G 的后边，关于Grappler 接口板的各种工作方式及有关选择性参数请详见说明书。

图形打印机 （SILENTYPE PRINTER）

Apple 计算机的图形打印机接口板插在1号槽内，下面的Logo过程将显示在屏幕上：

```
TO HC          ;silew ypc hardcopy ot grapluco
                                  screeu。
 OUTDEV 1
```

```
PRINT 1 CHAR 17
OUTDEV 0
END
```

图形打印机打印显示屏幕中的内容要求有ASCII 标准码17的字符，其中最引人注目的是“颠倒打印”方式，还可用其它方法可任意选择。从下边的过程中可以看出有关图形打印文件中有关控制信息，可供具体工作中选用：

```
TO HC          : inproued treasion tor sileu ype
 OUTDEV 7
 •DEPOSIT 53008 7      : Set printer to
                              darkest print
 •DEPOSIT 53007 128    : Set to uuidirectioncl
                              pnifiug
 •DEPOSIT 53012 0      : Set to iuverse prinfing
 PRINTI CHAR 17        : print fhe screen
 •DEPOSIT 53007 0      : turn off uridirecfiowcl
                              printiug
 OUTDEV 0
END
```

1DS 彩色打印机

采用PRISM 打印软件，综合数据系统9100—002—644 可以打印储存的彩色图形，但仍然需要用于Logo语言打印彩色图形的接口板。

2．4．颜色控制

用彩色电视监视器，必须用笔头颜色PENCOLOR 命令（缩写PC）

改变乌龟画图线条的颜色。还可用BACKGROUND底色命令（缩写BG）改变乌龟绘图的各种底色。PENCOLOR 和BACKGROUND输入数字0至6，数字与颜色的对应关系如下：

数字	颜色
0	黑色
1	白色
2	绿色
3	紫色
4	黄色
5	兰色

用PENCOLOS 6 画出的乌龟轨迹组成小点颜色反转。实际上颜色的变化，在各种情况下均与底色和乌龟轨迹组成小点的颜色有关。组成小点的反转色再反转则恢复原色。PENCOLOR 6最常用于黑白色绘图。

在未明确选用那一种BACKGROUND或PENCOLOR 命令之前，注意Logo语言不能执行BACKGROUND 0和PENCOLOR 1 。

2。4。1。彩色底色绘图

当在彩色底色上绘图的时候（2至5），在四种颜色——绿，紫，兰，黄——当中，只能用两种。当底色是绿色或紫色的时候，不能再用黄色；如用PENCOLOR 4 画的是绿色，PENCOLOR 5 画的是紫色。当底色为兰色或黄色时，PENCOLOR 2 画出的是黄色，PENCOLOR 3 画的是兰色。如果是在屏幕上画图，然后再改变底色，图面线条的颜色还会改变，或是这些线会意想不到的恢复原色；然而要恢复底色，又必须恢复画线的原色。这种奇怪的效果是Apple计算机颜色系统混合的结果。但这种情况是不允许发生的。因为绿点

和黄点是非常近似的。

2．4．2．不用颜色控制画图

用Apple计算机要想得到鲜明的颜色线条，Logo系统要求必须画一条很粗的线，如果画一条很细的线条不容易看清它到底是什么色。如果不着色，当然可以用细线。如选BACKGROUND 6，在BACKGRUOND 6，PENCOLOR 0 的情况下画的是黑色，用1至5画的是所谓“白”色，6为反转色。这个带括号的所谓“白色”，对于使用彩色监视器来说，“白”色的线条又不一定是真正的白色。实际上，一条垂直白线，可能是红色或绿色，这是和线条位置有关系。

2．5．Logo文件系统

Logo文件系统就是指存放在软盘上的过程内容。用户可以把许多文件同时存放在一张软盘上。这些文件是以不同文件名加以辨别的，这些文件名就是列在软盘上的目录表。

2．5．1．软盘文件

当使用Logo语言的时候，一般应有一张Logo文件软盘装在磁盘驱动器里。见2，5．3节中关于文件盘的叙述。要存放过程内容，必须用存放SAVE命令，例如：

SAVE “MYSTUFF

这里存放在暂存器内的过程内容及文件名是一个以MYSTUFF为命名的文件。虽然一个过程和一个文件用同一命名，然而通过SAVE存放在暂存器内的内容，不仅是存放这个命名的过程。当存放一个新命名的文件时，旧的文件即被清除，阅读READ命令要求输入一个文件名，这些过程和命名，将作为暂存器内增添的内容的一部分。因为Logo文件和画图，编辑过程一样都要占类似的储存空间。由绘图状态开始转变为非绘图方式时，也必须有各种的文件命令。

SAVE和READ要求在输入文件名之前必须有一个双点引号，不能在它的后边。

CATALOG目录命令将列出软盘上所有的文件目录。Logo暂存文件将列出文件名并附有“.LOGO” 字样。如：

SAVE “MYSTUFF

在目录中列出MYSTUFF•LOGO。当用READ或SAVE命令时，只列出名件名，而不再附.LOGO字样。

把磁盘上的文件清除掉，使用ERASEFILE 文件，此时输入的文件名也被清除了。

2．5．2．图形存储

在储存的过程当中，Logo语言也允许在软盘上储存屏幕画面图形，而且还可以再从软盘上读出来并重新显示在屏幕上，此时用SAVEPICT 和READPICT 命令。

SAVEPICT 类似SAVE要求输入图名，把屏幕上乌龟画图的图形储存在软盘上。当然SAVEPICT 只有在绘图状态时方才有效。READPICT 阅读的是用SAVEPICT 命令存在软盘上的图形，并且把图形重现在屏幕上。在目录CATALOG中，注意“.PICT” 图形文件与带“.LOGO”‘ 图形文件与带“.LOGO” 的正规Logo 文件一样。当用READPICT 命令时，并不包括“.PICT”图名。当然从Logo之外存取图形文件时，还必须有“.PICT ”.

从软盘上清除图形，用ERASEPICT 命令，此时输入的图形名也被清除。

可以用任何一种商品化的软件色，把储存的图形文件在打印机打印出来，Logo图形文件是以标准二进制文件格式书写的，在储存器内图形文件的格式化置于初始绘图文件页面内。

2．5．3．文件软盘的初始化

Logo文件存放在一般软盘上，软盘必须经过初始化过程，Logo文件软盘初始化也必须按照Apple soft BASIC规定进行：

1．首先把Logo实用盘放入Apple机中，并接通电源。实用盘并不包括Logo基本语言部分，它只录有表演程序和实用程序。

2．打

LOAD HELLO

按RETURN 键，等待计算机工作停时为止。

3．把实用盘从驱动器中取出，插入一片新的空盘。这张新盘要经过如下初始化过程，并把在软盘上的信息抹去。

4．打

INIT HELLO

按RETURN 键，并等待计算机工作，停时为止。（大约一分钟）软盘即被初始化，同时可以储存Logo文件了。

当使用Apple 计算机整数BASIC 语言进行软盘初始化时，按下列方法进行：

1．把实用盘的保护片取下来，插入驱动器内，开机后计算机将显示出“LANGUAGE NOT AVAILABLE”及（“>”）提出符。

打入如下命令：

```
10 PRINT "CTRL-D CATALOG"
UNLOCK HELLO
DELETE HELLO
SAVE HELLO
LOCK HELLO
```

2．特别注意；把实用量取出以后，务必把保护片贴好。

3．插入一张空盘，并照Apple Soft的第二条进行。

第三章

Logo系统的基本内容

Logo是一种强有力的计算机语言，它包括绘图，算术运算及表格处理的命令。同时还可以把实时屏幕编辑功能与行编辑命令相结合编辑程序过程的内容。

3．1．绘图命令

BACK	乌龟沿指定的方向后倒退运动。缩写为BK。
BACKGROUND	绘图屏幕底色，要求0至6数字代表对应的颜色。缩写为BG。
CLEARSCREEN	清除绘图屏幕，但不改变乌龟的位置和笔头状态，或乌龟隐身，或显示出来，缩写为CS。
DRAW	清除绘图屏幕，乌龟以屏幕中心为原点，即为下笔状态，DRAW与底色及笔头颜色无关。
FORWARD	乌龟按指定方向向前运动。缩写为FD。
FULLSCREEN	以绘图方式工作，在整个屏幕上绘图，与之对应的是SPLITSCREEN 切分屏幕。与中断复合字符键CTRL—F 等效。
HEADING	乌龟头部的输出值以十进位数字表示，由0至360。乌龟头向上的时候为0。
HIDETURTLE	乌龟箭头消失，缩写为HT 。
HOME	乌龟回屏幕中心位置，乌龟头向上。
LEFT	乌龟向左转，要求输入一个转角度数，缩写为LT。
NODRAW	脱离绘图工作方式，光标在清除一页文本后，回到屏幕的左上角，缩写为ND。

NOWRAP	脱离卷绕工作方式，乌龟从屏幕一边出，再回到对边上，任何命令执行过程都属正常现象，并非错误的结果。
PENCOLOR	乌龟画线的颜色，用数字0至6表示。缩写为PC。
PENDOWN	乌龟运动时会留下轨迹，与之对应的是PENUP，缩写为PD。
PENUP	乌龟运动时不会留下轨迹，缩写为PU。
RIGHT	乌龟向左转，要求输入一个转角度数，缩写为RT。
SETHEADING	表示乌龟转角的方向，输入度数。0为向上，顺时针方针旋转，角度数增加。缩写为SETH。
SETX	乌龟水平运动座标值。
SETXY	乌龟运动到一点的位置，要求两个输入值，0，0为屏幕中心，当Y座标（即第二个输入值）为负时要加圆括号。
SETY	乌龟垂直运动座标值。
SHOWTURTLE	在屏幕上出现乌龟标志与之对应的是HT；缩写为ST。
SPLITSCREEN	以绘图方式工作，屏幕混合文本与图形显示，对应的是FULLSCREEN，与中断复合字符键CTRL-S 等效。
TEXTSCREEN	以绘图方式工作，但整个屏幕显示为文本，与切分屏幕与全屏幕显示相对应，与中断复合字符键CTRL-T 等效。

TOWARDS　　要求两个输入数字，即屏幕上一点的X、Y座标值。TOWARDS输出乌龟头部指向的一点；SETHEADING TOWARDS :X :Y 指乌龟面向的一点之座标，与之对应的是ATAN命令。

TURTLESTATE　不要求有输入值。输出乌龟状态信息的一个四项内容的列表。表格的格式如下排列：第一项为下笔或抬笔的真伪，第二项为乌龟隐身或显示的真伪，第三项为底色，第四项为笔头颜色，缩写为TS。

WRAP　　绘图系统中的卷卷工作方式，任何时候都显示乌龟从屏幕一边出去，又在对边再现，WRAP对应为NOWRAP。

XCOR　　以十进位数字表示乌龟X座标值。

YCOR　　以十进位数字表示乌龟Y座标值。

3。2。数值运算命令

+　　加。

−　　减（两个输入）和负（一个输入）。

*　　乘。

/　　除（总可能由小数输出）。

>　　第一个输入值比第二项大为真实输出，否则为伪。

<　　第一个输入值比第二项小为真实输出，否则为伪。

ATAN　　要求两个输入值，输出反正切的商对应角度值，0至360范围内，一个象限对应二个输入值，

第二个输入为负时，要加圆括号。

COS　　输出输入的余弦值（输入角度值）。

1NTEGER　　要求输入一个数，输出整数部分，而略去小数部分。

NUMBER?　　输入一个数，输出为“真”。类似还有WORD?及L1ST?

QUOT1ENT　　输入两个数，只输出整数商，若输入不是整数，首先要改其接近的整数值。若第二个输入为负数，必须加在圆括号内。

RANDOM　　要求输入一正整数n，则输出为0至$n-1$之间的任意整数，调出任意数RANDOM 的顺序与每次起动Logo重复任意数的顺序是一样的，只有用RANDOM1ZE 产生任意数的情况除外。

RANDOM1ZE　　是任意数RANDOM 随机化的起源。当明确一个输入数时，其随机数将由它发生。例如每次执行(RANDOM1ZE 259)其随机数生成的顺序都是一样的，只有数不同。注意这里要求有输入值的RANDOM1ZE 用圆括号括起来。

REMA1NDER　　当第一个输入值被第二个输入值除时，只输出整数余数。若第一个输入不是一个整数，就要换成一个近似的整数值。第二个输入为负时，要加在圆括号内。

ROUND　　输出近似整数值。

S1N　　输出输入的正弦值。

SQRT　　输入一个正数，输出正数的平方根值。

3．3，字和表的操作命令

= 输入两数，比较其数值是否相等，如果输入两个字则要比较它们的字符串是否一致。注意在=之前要留一个空格。如果输入是两个表，则比较表内各项是否相同，最后输出有“真”与“伪”的区别。

BUTFIRST 输入为一列表时，输出除去表内第一项之外的其它所有项内容。若输入一个字词，则输出除去字的第一个字符以外所有字符。要调一个空字或一个空表则显示错误信息。

BUTLAST 输入为一列表，则输出除去表内最后一项之外的其它所有项内容。若输入一个字词，则输出除去字的最后一个字符以外所有字符，缩写为BL 。要调一个空字或一个空表则显示错误信息。

FIRST 输入一列表，只输出表的第一项。若输入一个字词，则输出字词的第一个字符。要调用空字或空表则显示错误信息。

FPUT 要求两个输入，第二个输入必须是一个列表，输出包括第一项，接下去第二项的一个列表。若第一项输入是一个表，如FPUT〔AB〕〔CD〕，结果将是〔〔AB〕CD〕。参见LPUT，LIST 和SENTENCE。

LAST 输入一列表，只输出表的最后一项，若输入一个字词，则输出字词的最后一个字符，要调空

字或空表则显示错误信息。

LIST? 当输入一列表为“真”时才有输出，参见WORD? 和NUMBER?

LPUT 要求两个输入，第二个输入必须是一个列表，输出包括第二项，接着是第一项的一个列表，如第一项输入是一个表，如LPUT〔AB〕〔CD〕，结果将是〔CD〔AB〕〕。参见FPUT，和SENTENCE，

SENTENCE 输入不一（两个以上），如果输入皆为列表，则可综合所有项为一个表。如输入是字词或数字，可当作一字之表的形式去操作，若第一和第二个输入是列表，如SENTENCE〔AB〕〔CD〕，其结果将为〔ABCD〕。如多于两个输入，则必须在SENTENCE 之前加括号，留一空格，在最后一个输入之后再加一个括号。参见FPUT，LPUT和LIST。缩写为SE。

WORD 输入数不一（两个以上），输出的是输入字符连在一起的一个字。若有两个以上的输入，则必须在WORD之前加括号，留空格后，在最后一个输入之后再加括号。

WORD? 输入一字为“真”时才有输出。可把数字也看成是一些字，输入一数为“真”时，才有结果。参见LIST? 和NUMBER？。

3．4．过程定义及编辑命令

DEFINE 要求两个输入，第一个输入是过程名。第二个

输入是一个表。表中每一项必须是关于表格本身的内容。表的第一项是输入过程的一个列表。（若没有过程输入，则第一项也是一个空表），每一个顺序项都应是与过程内容相对应一行的一个列表。如DEFINE "TRIANGLE [[:SIZE][REPEAT 3[FD:SIZE RT 120]]]，参见TEXT。注意一般在定义过程DEFINE 之前，加TO引导。

EDIT　　进入编辑状态，若过程名作为一个输入，则过程将作为编辑内容处理。若输入未加以说明，将进入以前的屏幕编辑内容的编辑状态，或进入刚刚被定义的过程以前未恢复的内容。还可附以字调：ALL, NAMES, PROCEDURES 。见键盘编辑命令的说明。EDIT缩写为ED 。

END　　打入编辑的过程内容终止符。最后内容已终止的当然不必再要END，一次要定义几个过程，则必须用END 语句选择过程内容。

ERASE　　清除暂存器内指定的过程，也可加修饰词ALL, NAMES, PROCEDURES。当未指出过程名时，将显示出错误信息。清除过程要用LOOKUP，打ERASE LOOKUP。ERASE 缩写为ER 。清除过程的一个列表，用以下过程实现：

```
TO ERPROCS :PROCLIST
 IF :PROCLIST =[ ] STOP
 RUN LIST "ERASE FIRST :PROCLIST
 ERPROCS BUTFIRST :PROCLIST
END
```

ERNAME　　要求输入过程名，并取消程序库存名，不用过程名即显示错误信息，ERNAME 不像ERASE那样，要求有输入，如清除TEMP名，打ERNAME "TEMP。若打ERNAME TEMP，TEMP是假设的一个过程，而且Logo将执行这个过程。

TEXT　　要求输入一个程序过程名，输出过程文本列表。过程名开始要有"符号，Logo将执行这个过程。若过程未被定义，TEXT输出〔〕，若输入是Logo指令名，输出也是输入的指令名。参见DEFINE。

TO　　引出过程定义，要求输入数不一样，进入以第一个输入为名的过程编辑状态。TO后边跟着输入第一个输入命名的过程，如果没有输入，TO进入空编辑缓冲器的编辑状态。

3．5．命名命令

MAKE　　要求两个输入，第一个必须是一个字，和对待一个变量一样。要求第二个输入是一个变量值（或事物）。

THING　　输出输入的有关事物，但它必须是一个字，注意它可作出非同一般的"超级"的判断
。THING"×××等同于：×××。

THING?　　当输入符合函义为"真"时才能输出。

3．6．条件命令

ALLOF　　要求几个输入（两个以上），只有在全都为

	“真”时，才输出“真”。多于两个输入，必须在ALLOF 之前加括号，留一空格，在最后一个输入之后再加括号。
ANYOF	要求几个输入（两个以上），至少有一个为“真”时，才输出为“真”。多于两个输入，必须在ANYOF 之前加括号，留一个空格，在最后一个输入之后加括号。
ELSE	用在IF…THEN…ELSE 中。
IF	用在基本条件语言IF＜condition＞ THEN＜action 1＞ ELSE＜action2＞中，对＜condition＞ 条件的检验；＜action 1＞第一项为真时才能成立，若为伪则＜action2＞第二项条件成立。字THEN是可选择的，ELSE ＜action2＞部分也不一定是非有不可的，＜condition＞条件必须用Logo表达，输出有“真”与“伪”，真与伪满足条件的函意也不同、各种检验功能为＜，＞，＝，和NOT。＜action＞ 和＜action2＞也可以用Logo表示其它任何一个数。
IFFALSE	当先前的结果经TEST 检验为伪时，执行剩下的那一行，缩写为IFF。
IFTRUE	当先前的结果经TEST 检验为真时，执行剩下的那一行，缩写为IFT。
NOT	输入为伪确真时才有输出，或输入为真确为伪

时才有输出。

TEST　　用以连结IFTRUE 和IFFALSE的检验条件，TEST要求一个输入，但它必须是真，或是为伪。用IFTRUE 和IFFALSE表示每一个过程检验的结果，并且还要写在执行的过程中。

THEN　　用在IF…THEN…ELSE…中。

3．7．控制命令

GO　　要求输入一个字，通过标志转到另一行上。在规定的一行的开头打上标志，跟着要有一个双点冒号。GO在Logo程序中极少应用。

GOODBYE　　清除暂存的内容，重新起动Logo。

OUTPUT　　要求一个输入，工作过程一停止，就会输出要调的过程。如果输入要求判断，输出则为判断的结果，缩写为OP。

REPEAT　　要求输入一个数字和一个列表。按指定的次数执行列表。

RUN　　要求输入一个列表，按打字一行命令执行列表•RUN执行的列表，不得超过地址255顶头一行最多的字符数，否则将显示错误信息。

STOP　　使执行程序停止，但STOP与END 意义不同，STOP是指停止现行过程，返回以前的过程。而END 是在编辑过程中程序过程的停止。

TOPLEVEL　　整个程序在执行中紧急中止，返回顶头一行，调出所有过程，在Logo程序中很少用。它与STOP不同，STOP虽然现行过程停止了，但

继续执行调用的过程。

3．8．输入与输出命令

OUTDEV　　要求输入打印机接口板插入的槽号。因为执行每一个命令都与这个槽号有关。OUTDEV 0 是表明屏幕输出。“槽号”的数字转成用户支持的汇编语言程序中，以字符输出的地址码，要比255大。详见6、3、1节中详细说明。按CTRL—SHIFT—M 复合键恢复屏幕输出。

ASCII　　当输入一字符后，输出ASCII码的数字。

CHAR　　输入一整数时，输出字符也为ASCII码中的整数。

CLEARTEXT　　清除文本屏幕，光标回原位。

CLEARINPUT　　清除输入缓冲器字符。

CURSOR　　要求两个输入，行与列即光标位置，屏幕上有0至39行，0至23列，光标位置以CH与CV表示。

PADDLE　　输入有0至3数字，表示游戏棒的位置。根据游戏棒的位置输出0至255的一个数字。例如用两个游戏棒为SETXY PADDLE 0 PADDLE 1。

PADDLEBUTTON　　输入有0至2数字，按纽位置输出相当于其和伪。例如1F PADDLEBUTTON 0＝“TRUE THEN CLEARSCREEN。在Apple ‖ 计算机上，没有游戏棒3的按纽。

PRINT　　输入数不一（最少一个）。指打印显示在屏幕上的意思。显示列表就像一个句子一样，不必加括号，下一个PRINT 指在屏幕上打印显示下一行。几个输入如（PRINT 1 2 3），这些输入打在同一行上，当中留空。但整个语句必须用括号。若输入PRINT 是一个过程，计算机不是打印显示这个过程，而是执行这个过程，并输出结果。参见PRINTOUT 说明PRINT 缩写为P R。

PRINT 1　　类似PRINT，但并不使输出的一行终止，而是继续打印，在几个输入之间不留空。

R C?　　在未按键盘上字符键时，计算机立刻认为"真"而输出。READCHARACTER还没等得按键，就输出了。相反为"伪"。

READCHARACTER　　输出字符缓冲器里原来的字符，如果缓冲器是空的，要等待输入字符。参见CLEARINPUT及7．4节。详见关于INSTANT实用盘上的瞬时程序有应用举例。READCHARACTER缩写为R C。

REQUEST　　等待操作者输入一行的内容，只有按RETURN才结束。并把这一行输出一个列表。缩写为R Q。

3．9．暂存器的文件及管理命令

CATALOG　　打印显示出软盘上的文件名目录。

DOS　　要求输入一个字或一个表。并把输入转译为操

作系统的操作命令。DOS〔RENAMEGMAE·LOGO, GAME LOGO〕储存重新命名SAVE"GMAE的内容。开启封闭文件，如打DOS〔UNLOCK ADDRESSES。LOGO〕, DOS有如下命令：DELE, VERIFY, CATALOG, LOCK, UNLOCK, MON, NOMON, RENAME, BLOAD, BRUN, BSAVE。详见Apple操作系统手册中有关语法信息的DOS 命令。

WARNING：是要撤消和替换一些独立内容的命令。

ERASEFILE　从软盘上清除用SAVE存放文件。要输入一个名件名，开始必须加一个“符号。

ERASEPICT　从软盘上清除用SAVEPICT 储存的图形。要输入图名，并加“符号。

PRINTOUT　当输入一个过程名时，计算机立即打印出程序过程的文本。如果未加输入，则打打印出最后一个过程。用打印机打印过程内容称CIRCLE，可以打PO CIRCLE，也可不打PO CIRCLE，还可用附加字：ALL, NAMES, PROCEDURES；打印出程序列表，可用如下过程：

```
TO POPROCS :PROCLIST
 IF :PROCLIST=[ ] STOP
 RUN LIST "PO FIRST :PROCLIST
 POPROCS BUTFIRST :PROCLIST
END
```

PRINTOUT 缩写为 PO。

POTS 为 PRINTOUT TITLES 的缩写。

READ　　阅读磁盘文件，显示图形消失，READ 要求输入一个文件名，但开头必须加“符号。

READPICT　　阅读存在软盘上的图形，并在屏幕上显示出来。

SAVE　　把暂存的内容储存在软盘上，详见有关文件注释。在储存时，屏幕上图形随之消失。SAVE 要求输入一个文件名，并在开头加“符号。

SAVEPICT　　把屏幕上的图形存放在软盘上，SAVEPICT 要求输入一个图名，在开头必须加“符号。

3．10 调试命令

CONTINUE　　在 PAUSE 暂停或 CTRL—Z 之后，恢复执行命令。缩写为 CO。

PAUSE　　暂时停止执行，使命令暂时放在一个局部环境之中，PAUSE 与中断复合键 CTRL—Z 等同，只有用 CONTINUE 恢复执行命令才不会显示错误信息。

NOTRACE　　脱离跟踪状态。

TRACE　　并不要求输入，在执行每一个程序之前 Logo 处于暂停状态。打印过程名并输入计算机，按复合键 CTRL—G 和 CTRL—Z，Logo 将在下一行继续显示字符。按 CTRL—G 键 Logo 中途停止，按 CTRL—Z 键暂停，之后再继续执行，详见 CTRL—W 有关说明。

3．3．1．杂项命令

•ASPECT　垂直方向改变Logo绘图的大小，要求一个数字输入，即可改变屏幕显示图形的长宽比。在一般比情况长宽比为0.8，因为不是所有电视监视器都有垂直变形的调整装置，利用乌龟程序画方形和圆，可能画出来的是矩形和椭圆。利用。ASPECT命令还可以调整显示图形畸变。但要注意长宽比的变化受Y座标值的限制。

◦BPT　双Logo语言中取出来进入Apple 计算机监视器中去，◦BPT只使用在Logo系统调试时，进入1BFg地址，Logo在起动之后使用之前为“冷起动”，系统再重新使用称1BFC为“热起动”。在冷起动后再起起动Logo，所有过程消失，就像打GOODBYE一样。热起动全部过程完整无损，继续执行Logo程序最好还是按CTRL—Y 及RETURN 键。

◦CALL　调用储存器内的机器语言子程序。第一个输入是子程序地址。第二个输入是存放在储存器内的程序位置。允许用户提供自己特殊用途的内容，并与Logo接口。详见6．2节。

◦CONTENTS　使Logo能理解的全部为字词组成的表格内容。包括在过程中所用的变量名，过程及字词等。编辑程序每个过程都要加以说明，是否还要删减。TEXT和THING？是经常用于列表的基本指令。但要注意不要把一些无用数据也填到

程序内容中去，在储存器中不常用的字，及经常在打字的时候发生错字也混在其中。应当使·CONTENTS 内容精炼得多。

·DEPOSiT　要求两个数字输入，一个地址，一个数值。在指定的储存位置存放数据字节。

·EXAMlNE　要求一个输入，输出指定地址的字节。详见6。1节。

·GCOLL　强迫转移到“程序垃圾箱”。

·NODES　输出自由节点数，包括自存储的节关在内时，在打·GCOLL 之前要打·NODES。

;　经常出现于过程内容和过程标题之中，保持一行，但把过程名，简短过程，子过程隔开。

第 四 章

实 用 盘

这里所说的实用盘是指包括美国麻省理工学院和 Terrap 的 Logo 的表演程序及应用程序在内的软盘。实用盘是一个由 Logo 程序组成的软件包。

在这一章中，将列出实用盘的文件，并叙述如何利用这些表演程序。

实用盘的使用方法

按照1．3节起动 Logo 之后，屏幕上应当出现？及一个闪烁的光标。

将实用盘插入驱动器内，打 CATALOG，再按 RETURN。屏幕显示目录的一部分，按空白条键就会显示其它部分。当打 CATALOG 之后，在屏幕上乌龟作图只能是切分屏幕的形式，即下边四行显示文本。按 CTRL—T 就变成全部屏幕显示文本，按 CTRL—S 返回绘图屏幕。

从软盘上读程序，要打 READ" 再写程序名，不用打 "．LOGO" 尾名，最后也不用再打引号。

在打印显示过程名之后，Logo 便会阅读这个文件，读完之后还会出现一个？号。

有些程序是自起动的，所以开始就会逐行阅读文件，但大多数程序不行。要求打原始过程名，实用盘上的表演程序原始过程和文件名是一样的，经过初始化的软盘，再用应用程序的过程叫"SETUP"。

当 Logo 显示？号时，就表示阅读文件完毕，可以开始编写程序，下边解释为何使用这些程序。举例说明如何使用表演程序，操作者打大写字母，Logo 语言以斜体字母回答：

```
? READ "ROCKET
SH DEFINED
GLOW DEFINED
TRAVEL DIFINED
CIRC DIFINED
STARS DIFINED
SHOW DIFINED
RDCKET DIFINED
? ROCKET
```

4．1．程序说明

这些Logo文件都是由美国麻省理工学院及Terrapin公司写的各种应用程序。在实用量上还包括其它一些文件：

ASSEMBLER	Logo汇编程序，见第六章。
AMODES	有关6502寻址方式的文件。
ADDRESSES	由汇编翻译成Logo的有关编址文件。
OPCODES	6502汇编助记文件。
SHAPE, EDIT	Logo乌龟形状编辑文件，详见第五章。
FID	应用程序文件。它对文件的删减及改名十分方便。看到这些文件便可以去用，重新开始要打FID。
PSAVE	如果把指定的所有过程名及内容用Logo语言SAVE命令放到另一个文件中，当然十分不方便。用PSAVE 程序可以把这些过程列表放到一个文件中。首先输入PSAVE 文件名，再输入要存放的过程列表。这里可用一个过程名

NAMES— 个字代表所有现存的Logo过程名。还可以用HPL 程序把过程列表在打印机上打印出来。

要用PSAVE，就打

READ "PSAVE

如果要把C1RCLE, SQUARE和TR1ANGLE 为名的文件放到DES1GN 文件里，就打

PSAVE "DES1GN [C1RCLE SQUASE TR1ANGLE]

TEACH

Logo的编辑系统对于过程的定义和编辑来说是十分灵活的，然而对新的用户可能感觉它十分复杂，这种教使用者的TEACH 程序，是专门为只能定义过程而又不会用编辑系统的人设置的。它的优点是在使用过程中可以不时发出提醒的信息而又不清除乌龟绘图屏幕。

要用TEACH 程序，要打READ "TEACH

打TEACH 教Logo一个新字，直到打END 才能停止，如下举例，Logo 提醒的语句用斜体字表示：

```
? TEACH
NAME OF PROCEDURE > COUNT
1NPUTS (1F ANY)? :N
<1F:N=0 STOP
<PR1NT:N
<COUNT:N-1
<END
COUNT D1F1NED
?
```

CURSOR

这是一个控制字符输出的程序。CURSOR 的任务只有一个，控制屏幕上文本中光标的位置。

这些程序在Logo语言中并不是直接参与工作的形式出现的。

CORSOR .HV　输出一个列表，第一项是光标水平的位置，第二项是垂直位置。

CURSOR .H　只输出水平位置。

CURSOR .V　只输出垂直位置。

CURSORPOS　要求一个输入（列表）并置光标在屏幕上相应的位置处。

FLASHING　在执行此命令之后，所有显示的字符是闪烁状态。

INVERSE　字符和屏幕黑白显示颠倒。恢复正常的电视黑白显示方式。

所有这些程序并不是立刻出现在屏幕上，而是在已经显示出来字符的基础上奏效的。

开始打TO之后，按RETURN 屏幕上显示的是一行黑体文本，按CTRL—G 进入编辑状态，用SAVETEXT 可以存放文件；用READTEXT ，打ED再按RETURN 阅读文件；若打TO则清除缓冲编辑内容，只好再用READTEXT 重新阅读文件。

TEXTEDIT

通过这个程序将Logo编辑系统作为文本编辑用。这个程序可以用Logo编辑系统去阅读和储存英文文本的文件。

SAVETEXT :FILE　将指名的Logo文件编辑内容存放在软盘上，举例：

SAVETEXT “LETTER

READTEXT :FILE　　类似SAVETEXT，阅读Logo文件进入编辑状态。

SHOWFILE :FILE　　在屏幕上打印显示出文件的内容。

PRINTFILE :FILE　　在打印机上打印文件的内容。当然那一台打印机都应该插在槽1号位上，如打MAKE "PRINTER7。

PRINTTEXT　　在打印机上打印所编辑的内容。

DPRINT

这个文件包括让打印显示的文本进入磁盘文件的程序在内，详见7.3节。

OPEN :FILE　　要求输入一个文件名。这个DPRINT程序的输入，将使打印显示的文本进入被打开的Logo文件中。

CLOSE　　合上被打开的文件。所有输出都将写在文件上。

DPRINT :ITEM　　使打印显示的有关条文进入文件中去。

OPEN. FOR. APPEND :FILE　　用以代替OPEN。利用DPRINT程序将显示文本加到被打开的文件中。而不必再重写。

用这个程序写的文件，也可通过TEXTEDIT文件有关程序进入文本编辑状态，实现打印显示及阅读。用READ阅读的文件，Logo语言可通过Logo命令执行文件的文本。

ARCS

这个文件是按照指定半径画圆弧和圆的程序。为了编写这一类程

序提供方便，在这个程序中加以再处理。

RARC:RADIUS:DEGREES, LARC:RADIUS :DEGREES

这个程序包括画右，左圆弧；每个圆弧要求两个输入，一个是半径，一个是包角的度数。

RCIRCLE :RADIUS , LCIRCLE :RADIUS

右画圆，左画圆，每项要求一个画圆的半径输入。

TCL

乌龟控制语言。

这个文件是专门为Terrapin 公司生产的电子乌龟配套的，如下程序过程为乌龟控制所用：

SETUP　　打SETUP 之后就调出有关乌龟接口原始文件及各种系统参数。

HELP　　这个程序提示说明可供电子乌龟所用有关过程。

TFD :DIST , TBK :DIST , TLT :ANGLE , TRT :ANGLE

这些过程是当电子乌龟在地板上画图时，与在屏幕上画图的FORWARD, BACK, LEFT和RIGHT 对应的命令。

EYESON, EYESOFF

开关乌龟眼睛灯的过程。

HORNLO　　乌龟的低音喇叭发声。

HORNHI　　乌龟的高音喇叭发声。

HORNOFF　　关闭喇叭。

TPU　　提起电子乌龟的画笔，不再画图

TPD　　放下画笔。

FTOUCH?　　电子乌龟半圆形外壳碰到前方阻障物时，则输TRUE，否则输出FALSE。

LTOUCH?　　碰到左方阻障物时，则输出。

还有RTOUCH?右边阻障物。

还有BTOUCH?后边阻障物。

FTOUCHONLY?　　因为乌龟碰到前方，左前方，右前方阻障物时FTOUCH?都会输出TRUE。而FTOUCHONLY?仅是在前方碰到东西的时候才有TRUE输出。同样也适用于其它方向。

ANYTOUCH?　　只要乌龟碰到东西就输出TRUE。

NOTOUCH?　　乌龟什么也没碰到，输出TRUE。

TOULH　　输出编号触意件感器的位置。这是一个乌龟体关于触意检测的程序。乌龟的触意传感器编号是由Logo语言:FBlT,:LBlT,和:RBlT命名而组的。详见乌龟接口手册的有关内容。

.TCMD :COMMAND :ARGUMENT

.TCMD是一个控制乌龟的最低的程序。它包括有关乌龟的命令和变元。可用这个程序实现程序办不到的事。详见电子乌龟接口手册中关于乌龟控制命令的说明。

4.1.1.表演程序

这些文件都是各种不同的Logo 语言表演程序。

ROCKET

Logo语言有关火箭ROCKET 表演程序和火箭辅助文件

ROCKET. AUX, 以及火箭外形的文件ROCKET. SHAPES是一个典型的用户自行指定乌龟形状的文件。

打READ "ROCKET 计算机执行ROCKET 过程，打ROCKET执行ROCKET·AUX 及ROCKET·SHAPES 过程。有关火箭形状编辑方式问题详见第五章中有关说明。

Animal

这是一个有关动物知识的应用程序。通过玩游戏的方式猜答有关动物的一些问题。例如可提出各种问题，"它有翅膀吗？"，可回答"有"或是"无"。

通过打READ " ANIMAL 和ANIMAL 来执行这个程序。当玩一个游戏结束后，打SAVE " ANIMAL 可以把游戏内容存在软盘上。可再玩另一个游戏。在实用盘上的动物游戏有许多种动物的内容。对于开始玩动物游戏的人，首先还是从有关基本组织的程序开始，即INITIALIZE KNOWLEDGE。

Animal. Inspector

在Animal 动物知识程序的基础上，检查该文件的一个程序。Apple Logo的动物游戏内容有许多关于动物的各种知识，这些文件中也包括INSPECT. KNOWLEDGE检查动物知识的过程。它显示的动物程序知识内容更容易理解，比较形象。对于学习研究动物程序有一定的帮助。

动物知识检查程序筑取类似画知识林的递归方法，把动物程序的知识内容显示出来，在编写这种文件程序时也可以参阅有关递归的程序。

Instant

Logo系统为幼儿专门设置操作简便的INSTANT瞬时程序。事

先把要简化的Logo程序通过打：READ “1NSTANT和1NSTANT，只要再按单个字符的命令就可以操纵乌龟运动及定义过程。每个字符都可以产生动作效果。例如打F键，乌龟立刻向前一段距离，打R键立刻向右转一个角度。顺序重复打F和R就可以画一个方形。

1NSTANT系统有将打印显示的程序储存的命令，打N定义屏幕上画图的过程，如用R和F画一个方形时，用N命令就可定义此过程名SQUARE；也就是在1NSTANT 系统中通过调FORWARD和RIGHT 定义SQUARE 过程。当按P键时，1NTANT 系统便通过访问过程名，执行图形显示并执行程序过程。下边列出1NSTAN 瞬时系统命令表：

?	提示符号。
D	清除屏幕。
F	向前走。
L	向左转。
N	新图名。
P	显示被访问图名的图形。
R	向右转。
U	取消最后的命令。

总之，1NSTANT程序只是用简单Logo程序改编的适合幼儿学习的一种“语言”，当然还可以编写其它形式的Logo程序，例如用RUN 和DEF1NE 去代替1NSTANT ，比1NSTANT更简单方便，或是更为复杂的命令等。

DynatracK

Dynatrack 是一种动态乌龟作图的程序，即在一个时间范围内乌龟总是运动的。这一种特殊的程序，主要用以模制沿一定轨道的

航行过程．打DYNATRACK 开始，打K乌龟沿指定方向运动加速，打L和R表示乌龟向左转或向右转。

F1D

F1D 是一种用Logo语言写的文件实用程序，用一次击键命令，即可见软盘的文件目录，用以更改文件名以及清除文件等。每一个文件命令通过访问文件名，而实现“文件的扩展”，在F1D 程序中“ .”命令表示文件扩展的变化。经常用的Logo文件扩展内容有“LoGo”，“P1CT”，“SHAPES”和“B1N”。当发生磁盘文件错误信息的时候，就要重新开始执行F1D 程序。

Music

这里有三个音乐文件：Music．LoGo 即为演奏音乐的Logo过程；MUS1C．SRC、LoGo即为演奏音乐的汇编语言程序；MUS1C、BiN音乐表演程序BLOAD 文件；执行音乐程序用READ“MUS1C 并打SETUP，一些比较经的表演节目，执行FRERE 过程。详见6．4节有关内容。

lnspi

这是一种盘旋线1NSP1 的图形文件，按照指定的乌龟转角，不同的颜色画笔，连续执行四次如下过程：

```
TO 1NSP1 :D1STANCE :1NCREMNT
 FD :D1STANCE
 RT :ANGLE
 1NSP1 :D1STANCE :ANGLE + :1NCREMENT
                          :1NCREMENT
END
```

可通过打READP1CT “1NSP1 显示图形。

TET

TET 程序是一种简单的Logo程序，通过递归方法画复杂的更为有趣的图形的文件。TET 程序要求两个输入，即SIZE和LEVELS；如TET 100 1 即为一次递归画图程序，当然也有2，3甚致更高的层次。

第 五 章

改变乌龟的形状

Logo动画问题，在Logo语言的基础上加上. EXAMINE 和 . DEPOSIT 命令，要作各种动物游戏的项目是十分方便的。

本来Apple--II—Logo 语言设计是没有动画效果的，Logo语言只有在TI—99/4计算机上有动画的特殊硬件系统，所以在屏幕的最佳运动也只是表现乌龟的运动状态。如果要穿过屏幕作一个圆运动，无论画与涂这个圆，都是一点一点的作圆弧运动。当然可以这样改变一下乌龟的形状，看来象一个圆，还可以通过乌龟的简单运动越过屏幕作圆运动。

因此Logo语言当中的乌龟也是通过Apple 改变乌龟形状功能作一种乌龟外形。关于乌龟形状的二位与三位矢量表说明，详见Apple sott程序使用手册。当设计所希望的乌龟形状的时候，也可列所需的矢量表，这个表的第一项存放地址USHAPE 可见6．5节关于Logo地址的说明。关于乌龟的大小或改变形状一个字节尺寸地址码为SSIZE 数值1代表最正规的乌龟形状，2代表比一般用户规定要大的形状。由用户自行确定的外形，还包括乌龟头部角度，有0，90，180，270度几种。头部在外形的那个部位，象限；即乌龟面对的方向，还可用乌龟转向的命令，LEFT和RIGHT 改变其方位。

关于形状表的格式问题，详见Apple sott操作手册，只有形状头部的信息可以省略以外，形状表格开始就以矢量表示。

在设置USHAPE 和SSIZE 之后，由用户码把手工编的形状表用. DEPOSIT 存放在Logo语言当中，如果要改变用户规定的形状就要用USHAPE。

Logo 乌龟形状编辑功能

手工编形状表格是一个使人厌烦的过程，在Logo软盘上有一种“形状编辑 ShApE EDITOR”的程序，它是专用为在屏幕直接画乌龟外形的方法，自动在形状表格内收集形状图形。形状编辑系统是由Henry Minsky 教授编写的。通过. EXAMINE 和. DEPOSIT 收集处理有关外形轮廓的图案，把这些Logo程序用形状编集系统加以整理，阅读Logo外形编辑程序，并根据规定写出外形对应的函数。

使用形状编辑程序，首先打 SETUP 并阅读 SHAPE, EDIT 文件，文件包括实时形状编辑系统及改变显示乌龟形状及大小的函数，开始用命令MAKERHAPE 设计所予想的形状，要求输入一个形状名，例如：

MAKESHAPE “BOX

再打另一个MAKESHAPE 命令，将再定义一个新的形状。不能编辑以前定义过的形状。如果要清除所有的形状过程，重新开始，就再一次打SETUP 。有关构成乌龟形状的编辑命令列出如下：

U	抬笔，
D	下笔，
CTRL—P	上移（下笔状态时，画出垂直线）。
CTRL—N	下移（下笔状态时，画出垂直线）。
箭头键	按箭头方向运动（下笔状态时，画水平线）。
CTRL—C	脱离形状编辑及形状定义状态。
CTRL—G	脱离非永久形状定义状态，主要用于中断形状定义而命令重新开始。
ESC	清除以前的少数命令。
1…9	改变形状大小打一个数字·打了就是执行

SIZE 3. 这时选择形状大小是操作比较方便的。

例如：要定义一个方形的乌龟形状时，可打 SETSHAPE：BOX。用SIZE过程改变形状大小，SIZE 1 就是其中一个。

SETSHAPE 要求一个输入，以改变乌龟形状，可以显示出来，也可以是看不见的隐藏状态，要恢复原始三角形乌龟形状打 SETSHAPE 0。

在过程内部SETSHAPE 为. SHAPE，它要求输入形状，但开始乌龟只能有显示状态。在Logo程序中经常用“颠倒”REVERSE方式，在屏幕上把乌龟形状显示出来或是清除掉。隐藏状态就是清除屏幕，看不见任何线条。. SHAPE 1 乌龟形状为不存在的状态。下边一个程序利用这种方法，标出用户指定的一种形状。第二个程序过程用 USHAPE 阅读实用量文件地址 ADDRESSES，最后一个程序过程在屏幕上的位置是随机变化的。

```
TO STAMP :SHAPE
 . SHAPE 1
 HIDETURTLE
 . SHAPE :SHAPE
 SHOWTURTLE
END
TO SS
 STAMP (. EXAMINE :USHAPE)+256*. EXAMIME !
  :USHAPE+1
END
```

```
TO STAMPRANDOM
 SS
 SETXY (120-RANDOM240)(110-RANDOM220)
 STAMPRANDOM
END
```

形状存储

把所定义的形状存放在软盘上用SAVESHAPES过程，它要求输入一个文件名，在SAVE一行要求有引号。

如下信息是十分重要的：开始在暂存器内并无被定义的过程的情况下，阅读形状编辑程序并定义形状，SAVESHAPES “PARTS 有两个文件，一个是PARTS．SHAPES包括形状表在内，还有一个PARTS．AUX．LOGO包括形状名以及下列过程：SETSHAPE，．SHAPE，SIZE，INITSHAPS 和要定义的程序过程，除非用低水平的程序写操纵形状的有关文件，而不会超出这些程序的水平。

“．AUX” 文件包括一个INITSHAPES过程，它可自动调用并定义形状，存放在文件内的外形程序称BLOCKS，所以过程应包括有：

```
READ “BLOCKS．AUX
INITSHAPES
```

当然可以从许多形状文件之中，选择一种来充实所需的程序，通过改变不同的文件而选取不同的形状。不必担心使用形状系统偶尔也会取消了那一个过程或是形状名。但只要还在使用形状编辑系统，可不去定义过程，即便偶尔不得已，也应从“．AUX” 文件结果的两种不同文件中选择一下。

抽样对话

```
? GOODBYE
? READ "SHAPE . EDIT
? MAKESHAPE "BLOCK
Define a shape here
? MAKESHAPE "TIRE
Define another shape here
? SAVESHAPES "BLOCKS
```

Logo所访问的文件软盘，必须是用来存放有形状及其程序的磁盘。只有在Logo存放了乌龟形状以后，再去访问包括形状编辑程序在内的软盘。之后将实用盘或是转用的实用盘，放入驱动器内，并按RETURN 键。Logo在阅读形状编辑程序后将暂停一会儿，再把要放在储存器内的两个新文件存放在文件软盘上：即BLOCKS, AUX, LOGO 和BLOCKS, SHAPES, 开始打的GOODBYE用以清除掉所有东西并且重新开始。要在调形状程序，则打

```
READ "BLOCKS . AUX
INITSHAPES
SETSHAPE :BLOCK
```

此时有两种工作方式可供选择：一种是用这个形状过程还可以写程序并可以把每一部分存放在软盘上。这种方法所阅读的形状过程实际上执行INITSHAPES；然而当改变文件BLOCKS 的形状时，必须清除所有的NAMES，并且还要阅读新的BLOCKS . AUX 文件。另一种是用GOODBYE 重新开始，在清除暂存器内容后，利用形状程序写新的程序，包括阅读形状过程中的READ和INITSHAPES命令，再重新开始执行程序，要检查这些程序时，可从过程暂存器中把形状调出来。在满足工作要求之后，再用ERASE 将NAMES,

SETSHAPE, SHAPE和INITSHAPES 清除。下边将简短的分别说明选用这两种文件的一个程序。

实用盘应用程序举例：ROCKET

在实用盘上应用程序当中，有一种关于火箭的程序ROCKET。打READ "ROCKET 则执行ROCKET 过程。ROCKET 过程包括READ和ROCKET.AUX 说明文件，及调.INITSHAPES 过程，INITSHAPES过程便自动准备乌龟形状。

用常规方法用形状程序也很方便，打PO ROCKET 即可见其工作情况，不再调形状文件要执行这个程序打SHOW。

第 六 章

Logo 汇编语言接口

Apple计算机Logo系统设计的使用很方便而且功能很强。在前一章中已介绍用Logo语言编写和执行程序的主要内容，当然有许多问题，要深入的研究必然涉及到直接需要机器语言才能处理的问题。

不过这里首先应当声明；这一章内容是为已具备计算机相当知识水平的人，了解Apple 计算机汇编语言而写的。

Logo系统中由许多问题涉及到用户直接访问Apple 计算机储存的位置，以及汇编语言程序与Logo程序的接口。在Logo实用盘上包括6502机器语言汇编程序。同时还涉及到Logo乌龟显示到形状的动画效果问题。也关系到使用Logo编辑程序问题，宁可用一般的文本编辑系统也不用过程编辑程序及非标准方式访问磁盘文件。

6．1．．EXAM1NE 和．DEPOS1T

这两个命令实质上是用 Apple 计算机PEEK和POKE程序。所不同的只是这两个命令的地址总是正整数，而Apple 计算机EEK和POKE要求的32K地址总是负数。．EXAM1NE 要求输入一个地址，把一个数的字节存在地址当中。．DEPOS1T 要求两个输入，一个地址和一个数值，把这个数值以字节放在地址中。这两个命令经常与特殊用途的输入输出装置相连接，但与OUTDEV 无关。不过这两个命令的地址均应把十进位Logo语言中数字转换为十六进位Apple计算机地址码。

要用

．EXAM1NE HEX “9E

不用

．EXAM1NE 158

Logo 执行这个命令，并不是指在 ROM 上的监护程序，这些部分 Logo 系统工作时要存放在 16K 存储板上，实际上在 ROM 上的内容被存储板所遮敝。调. EXAMINE 或调. DEPOSIT 实际上是指“语言卡”的对应地址。16K 的存储板占用 $C080—$C08F (49280—49295) 位置。

6．2．写机器语言程序

Logo 语言迈过. CALL 实现与机器语言程序接口，. CALL 要求两个输入，第一个是程序地址，第二个输入一个整数以检查程序，而这个程序总是输出一个整数或什么也不输出，所以. CALL 总是要有两个输入，而不管用户选择那一个去检查第二个输入程序。

. CALL 用第一个输入控制变换地址，在此之前将所占用的程序通过汇编并储存其地址，用户使用的储存器机器码由 $99A0 至 $9AA0，用 . DEPOSIT 通过汇编并储存程序，关于 Logo 汇编更多的内容详见第6．3节。

开始执行程序，首先从0页 NARG1 位置起，由. CALL 调包括第一个输入的 NARG1+1 —— 程序地址，NARG1+2 和 NARG1+3 被调的也包括0页的程序。在 NARG1 至 ANSN4+3 范围 Logo 可以变化储存的位置；Logo 不用 USERPZ 至 $FF 位置，在这个范围内是不能用 Logo 汇编程序去变化位置的，只能在0页储存。

. CALL第二个输入包括在 NARG2 至 NARG2+3 的范围内，有两种四位字节的形式表示，如 . CALL (S“99A0) 3 其储存的数值为：

NARG 2	NARG2+1	NARG2+2	NARG2+3
3	0	0	0
NARG1	NARG1+1	NARG1+2	NARG1+3
$A0	$99	0	0

由NARG2至NARG2+3包括$FF以1代替了。

在NARG2至NARG2+3四个位置当中，输出都是一个整数，储存的也是整数，但是转移到OTPFX2 位. 如果储存的数字是四其顺序不同（如NARG1），调Y地址，就要转到OTPF1X 位。

当Logo程序输出为“TRUE ，就转到OTPTRU 位，输出为“FALSE 就跳到OTPFLS。如果输出没有一个数值，则程序终止要用一个RTS 指令。

举例阅读磁带输入位置地址为$C060，结果“TRUE 或“FALSE 其声音也就不同。这里用的是6502汇编语言的标准格式代码写的。因此储存指令也必须用汇编语言来写。（详见6．3节内容说明）。

```
          ORG  $99AO
C1N       EQU  $C060
OTPTRU    EQU  <see Addresses . Logo>
OTRFLS    EQU  <see Addresses . Logo>
L1STEN:   LDA  C1N
          BM1  ON              ; See Apple ||
                                 Setrenco Marmal. C7.
          JMP  OTPFLS
ON:       JMP  OTPTRU
          END
```

例子要调L1STEN 标志的地址，执行这样一个新的任务，要打

.CALL : L1STEN 0

注意这里必须有一个输入，但往往被人们忽视。接着可像一般的程序那样：

PR1NT .CALL : L1STEN 0

必须要显示TRUE或是FALSE。

当执行Logo程序发生错误情况时，机器语言程序将跳到PPTTP，而正常有效执行Logo程序的时候为TOPLEVEL 位置。

6．3．Logo汇编语言

Logo汇编语言就是写Logo语言的6502汇编语言。这个程序是由Leigh. KLotZ 设计的。这个汇编语言写在实用盘上的ASSEMBLER 文件中、当使用汇编程序时，像阅读一段Logo文件那样，打SETUP 就可以执行这个过程：

```
READ "ASSEMBLER
SETUP
```

ASSEMBLER 程序阅读储存在Logo软盘上的数据，还有AMODES和OPCODES，辅助文件。

也可以用Logo语言的编辑系统，以Logo程序的格式来写汇编程序，例如上述阅读磁带的程序可改写成：

```
TO CASSETTE . CODE
 [MAKE "C1N $ "C060]
 [MAKE "OTPFLS <See Addresses .Logo>]
 [MAKE "OTPTRU <See Addresses .Logo>]
 L1STEN: LDA C1N
         BM1 ON
         JMP OTPFLS
CN:      JMP OTPTRU
END
```

注意Logo汇编程序和标准的6502汇编程序之间，其语法是不同的。关于汇编语言的语法解释见6．3．2中的说明。

要定义这个汇编程序的时候，打

ASSEMBLE "CASSETTE .CODE

这个ASSEMBLE 程序中的ASSEMBLE 指令位置为$99AO，只要打出其标志码就可调出这个汇编程序：

.CALL :L1STEN O

6．3．1．使用汇编程序写输入输出程序

用.EXAM1NE 和.DEPOS1T 就可控制各种外国设备，致于有关的机器语言程序要求则是另一种样子。为控制驱动器就要用OUTDEV命令。但OUTDEV 要求1至7其中一个插板槽号的输入，虽然直接以Logo语言的字符形输入，但驱动器槽号要以汇编形式输出。

有些驱动器可能要求输入和输出的是一种特殊的汇编程序。可能输入OUTDEV 的数字大于8，那么还必须用Logo一般字符输入，转释为储存器中的程序地址，以汇编程序形式输出。

许多外国设备要求接口技术，计算机向外国设备送的数据不能太快、下边介绍Logo语言对这种驱动器的接口问题，用STATUS 表示存储器输入输出的驱动器卡的地址，驱动器内计算机阅读程序到驱动器接受字符交于7个比特的速度，DATA表示所传送字节的储存地址。

执行OUTDEV:TYOWA1T 的汇编程序：

```
TO CODE
 [MAKE "STATUS <address>]
 [MAKE "DATA <address>]
 TYOWA1T:LDX STATUS
         BPL TYOWA1T
         STA DATA
         RTS
END
```

A字符输出的汇编程序，通过OUTDEV 表示A寄存器包括其字节的输出。

另一个输出汇编程序的例子，在字符当中有一个！符号，即打印显示一个空格的程序如下：

```
TO IOCODE
XCLOUT:   CMP#"!
          BNE OUTCHAR
          LDA #32
OUTCHAR:  JMP COUT
END
```

6．3．2．关于汇编语言的语法

为了保持Logo语言的特点，Logo汇编程序所用的格式也就和大多数汇编程序有明显的区别。每一种汇编语言程序必须像Logo程序过程那样被储存起来。尽管这些过程还不能直接执行，下边简要以Logo汇编程序的格式，举例说明汇编语言程序与Logo语言的接口。

程序的标志以后缀冒号表示，引用零页储存位置，无论间接一变址寻址为（LDA CFOO，X）或变址一间接寻址（LDA （FOO），Y）都必须在标志和解释之前加警叹号。如果忘记这个警叹号指令即以绝对码方式寻址，将占有比一个更多的字节。在！（指示零页）或#（立即寻址方式）之后必须留一个空，在每个标志的前后也必须留空。每个指令操作数可能是一个字（引用标志），一个数字，一个列表或是一个单字母，但开始必须有一个引号。如果是后一种情况，指令操作数就是该字母的ASCll 码值。

以一般Logo叙述方式评价一个列表的内部问题，如果列表一行当中只有第一项内容，不考虑输出值，仅仅说明在列表内部有指定标

志而已。如果是一个操作数跟着指令名，就是有输出内容。算术表示方法为：FOO+3，这里FOO 是标志或是一般Logo符号，要用方括号括起来。引用的标志值在方括号内，在标志前必须有(:)，并像一般表示记号一样要留空。所有标志都是Logo变量。DOT 也是一个Logo变量，它是一个汇编位置值。

HI8 和LO8表示比8个比特大或小的数，它经常用在列表当中，就像：

```
LDA # [LO8 :SOURCE]
STA ! DEST
LDA # [HI8 :SOURCE]
STA ! [:DEST+1]
```

$程序要求一个字的输入，必须是十六进位的数字，输入也要以数字表示，十六进位的数也包括列表内部的$在内。用MAKE命令确定标志值。

如果用八进位或二进位数字，必须用Logo字$BASE并改变这个数值，$过程的基数变成二进位的2等等，汇编语言程序为[MAKE "$BASE2]。

除去标志之外，以前的这些程序各行的数字，就没有什么汇编字节编码形式出现了。

这里列出一个一般汇编语法写的简单程序

```
ORG    EQU $99A0
NARG2  EQU $9E
BELL   EQU $1C40
PASS:  LDA NARG2
       CMP #$04
       BEQ YES
       RTS
YES:   JMP BELL
END
```

用Logo编辑程序语法写的程序：

```
TO CODE
PASS: LDA ! NARG2          ; means page zero ,
                             Note space after
      CMP #[$ "04]
      BEQ YES
      RTS
YES:  JMP BELL
END
```

汇编程序和调用汇编程序打SETUP 和READ "ADDRESSES。汇编程序的原始地址为$ 9 9 A O，相当于Logo变量值ORG。

即ASSEMBLE "CODE

这样在屏幕上将产生一个列表性的文件，并且它的编码放在储存器中。其标志可利用Logo符号表示.CALL,.DEPOSIT，和.EXAMINE.调用汇编程序打：

.CALL :PASS 4

这时计算机发嘟嘟响，.CALL : PASS < anythingbut 4> 什么事情也没有反响，因为这是一种密码程序。

如果汇编一个很长的程序，超出储存器的容量，解决更大储存容量的方法，就是只调用所用到的那些指令，即从实用盘上只阅读OPCODES文件，清除那些不用的指令（例如用ERNAME "BIT），然后再重新写新的OPCODES文件。当然不应用实用盘母盘，而是利用复制的实用盘，再去执行汇编的任务。

6．3．3．在磁盘上存放汇编程序

利用DOS 指令，存放的汇编程序实际是机器代码，为把一个汇

编程序放在一个叫ROUTINES 的文件当中：

DOS〔BSAVE ROUTINES .BIN, AS99AO, LS100〕

要调用Logo 汇编程序，打

DOS〔BLOAD ROUTINES .BIN〕

这里的BIN 是BINARY 二进位的缩简写法。这样所存放的实际是机器语言文件。要改变为Logo 变量，则用.CALL 去指定其地址。一种方法是打EDIT NAMES, 用CTRL—G 可脱离编辑状态，执行ERASE ALL 全部清除的任务。用CTRL—C 就可以重新进入编辑状态。用SAVE "ROUTINES 存放文件，打READ "ROUTINES 和DOS〔BLOADROUTINES〕重新再调汇编程序。

6。4。举例：音乐程序的形成

这一节介绍超出一般Logo 语言的范围，用汇编语言的实例。在Logo 语言的版本当中本来没有演奏音乐的内容，但可以用.CALL 通过机器语言程序便可在Apple 计算机的喇叭上产生出音调来。

喇叭所接受的很窄的脉冲其位为$C030(49200)，Logo 用.EXAMINE 重复地阅读这个位置。在一秒钟之内在程序中多次出现，即每秒出现的次数称为频率，一种音调其频比是固定的，音调变化，即音频的变化系几何级数关系增长。在西洋音乐中，有十二种音调，其余均成倍频方式关系。音调比即2的12次根值，近似为1。05946。

和频率密切相关的一个概念即周期PERIOD，音调的周期就是频率的倒数。给出周期，就是说在演奏相对应音调的时候，要等待满一个周期之后，再重复发生一个窄脉冲。这样的程序只有用机器语言写，才能达到执行很快的要求。

用Logo 语言演奏音乐，要写一种Logo 音乐程序。这个程序可

称“ PLAY”，它要求两个输入，一是音调的列表，与之相对应持续时间的列表。音调应当以数字表示，持续时间的长短，1 最短，2 为二倍长，以此类推。

```
TO PLAY :PITCHES :DURS
  IF :PITCHES =[ ] STOP
  PLAY.NOTE (FIRST :PITCHES)(FIRST :DURS)
  PLAY (BF :PITCHES)(BF :DURS)
END
```

其PLAY.NOTE 表示用一个音调及一定的节拍演奏一段音乐。

用机器语言程序演奏这一段音乐，要求音调的周期，我们必须用同样方法，用PLAY程序表现不同音调的周期相协调，可用一个表格方法，在表内每段音乐有一个周期。由Logo字词组成这张表，每一个周期也给一个周期名，我们可选择不同的表格的周期名，用不同字表示各个段落，有开始直到最后有表格名，也有音调数字。第三段周期“ # ”表，即Logo语言变为“ # 3 ”。Logo语言变为# R存储一个数值的时候，就是休止符的意思，所以只要有“ R ”便使PCAY程序休止。

在八度音中间上下以＋，－符号表示不同的八度。PLAY 程序4的意思是中间往前第四种音调，4 ＋表示往前一个八度的同一音调，4 －表示后一个八度的同一音调。实际上这涉及三个八度的问题。

下边列出每一种音调连同其周期相协合的 MAKE.PITCHES程序，它要求输入一个高八度数字和这个八度最高音调的周期：

```
TO MAKE·PITCHES :PERIOR
  MAKE·OCTAVE || :+ :PERIOD
  MAKE·OCTAVE || "  :PERIOD * 2
  MAKE·OCTAVE || "— :PERIOD * 4
  MAKE "#R 16384
END
TO MAKE·OCTAVE :PITCH :OCTIND :PERIOD
  IF :PITCH=0 STOP
  MAKE(WORD "# :PITCH :OCTIND) :PERIOD
  MAKE·OCTAVG :PITCH-1, :OCTIND :PERIOD *!
  1.0596
END
```

我们已经由Logo语言着手讲到音乐段落的发生程序，这些机器语言程序便会在规定的时间长度里，按指定周期而发出声音。

但是上边所说明的方法，计算机要等待满周期之后，并且直到一个段落之后Apple 计算机的喇叭才会发出声响。

要紧的是涉及机器语言程序的一个子程序CLICK，这个程序将重复PER.H 和PER.L 位置上的16 bit 周期。

一种方法是对持续时间进行注释，调用CLICK 程序一个固定的次数。然而CLICK 程序很自然地要求不同周期的各总合次数。所以这个程序必须把实际持续时间变换为发出声音的次数。

如果从所有其它音调当中可选一个基调，问题就解决了。CLICK 程序调用次数，给定持续时间和周期：DURATION *(:BASE .PERIOD/:PERIOD)。这样的持续时间范围是一种正常化的过程。在Logo语言中要乘和除比机器语言都非常容易，所以Logo音乐程

序将以Logo的规则处理有关持续时间范围的因素。机器语言程序要求输入一个数，即可多次调用CLICK 程序。

下边引用一段演奏音乐的机器语言程序：

```
10 MCODE
[MAKE "SPKR $ "C030]
[MAKE "DUR.L :NARG2]
[MAKE "DUR.H :NARG2+1]
[MAKE "PER.L :NARG2+2]
[MAKE "PER.H :NARG2+3]
[MAKE "COUNT.L :USERPZ]
[MAKE "COUNT.H :USERPZ+1]
[(PRINT [STARTING ADDRESS:]:ORG)]
TONE:   LDA ! DUR.L
        ORA ! DUR.H
        BEQ EXIT                    :A duration
                                of 0 means no note.
        LDA ! DUR.L
        SEC
        SBC # 1
        STA ! DUR.L
        LDA ! DUR.H
        SPC # 0
        STA ! DUR.H
        JSR CLICK
        JMP TONE
```

```
EXIT:    RTS
CLICK:   LDA ! PER.L
         STA ! COUNT.L
         LDA ! PER.H
         STA ! COUNT.H
         BIT ! COUNT.H
         BVS PDLOOP
         LDA SPKR              :Click
PDLOOP:  LDA ! COUNT.L
         ORA ! COUNT.H
         BEQ EXIT
         LDA ! COUNT.L
         SEC
         SBC # 1
         STA ! COUNT.L
         LDA ! COUNT.H
         SBC # 0               :propagagte carry
         STA ! COUNT.H
         JMP PDLOOP
[(PRINT "LENGTH: :DOT-:ORG "BYTES )]
END
```

过程体CLICK 和TONE程序具有一个有趣的属性：每次重复要求总合次数，还有一些写这种程序的方法，在DUR.H 或PER.H 为0时，执行的更快或更慢。这个方法能持续400次发音，可不是持续200发声的两倍长。

注意在CLICK 程序当中，还有一些局外的指令，仅仅执行音调的一个周期，这时产生的音调方为有效，每一个周期可在总合上增加一小部分，相反也可减少一小部分，FUDGE 条件是不变的。

机器语言程序与Logo程序的接口是完全必要的，.CALL 要

求仅可通过一个输入，但我们要通过一个周期（PER.H 和PER.L）和持续时间（DUR.H 和DUR.L）。二者合计要三十二位，也就是说.CALL 要机器语言程序必须能通过三十二位的能力。我们可安排.CALL 输入DUR.L/DUR.H 的储存位置低两个字节，PER.L/PER.H 高两个字节，下边列出两个输入的机器语言程序的过程如下：

```
TO .CALL.2 :ADDR :INPUT1 :INPUT2
  .CALL :ADDR (ROUND :INPUT2) + (ROUND!
    :INPUT1) * 65536
END
```

这里用到ROUND 指令，虽然我们没有称它为非整数周期，但相乘结果确实不是65536 的整数倍关系，因为和持续时间有矛盾。PLAY.NOTE 程序是一个包括正规化一步的综合程序。（.CALL.2 的三个输入，放在供选择的下一行上，这样更容易阅读）：

```
TO PLAY.NOTE :PERIOD :DURATION
  MAKE "PERICD THING (WORD "# :PERIOD)
  .CALL,2 :TONE
          :PERIOD — :FUDGE
          (:DURATION * :BASE .PERIOD/:PERIOD)
END
```

音乐表演程序

在实用盘上有两个与音乐有关的文件；一个Logo文件叫MUSIC，另一个是存放机器语言程序的文件叫MUSIC.BIN。当打READ "MuSiC 和SETUP 时，就可调用音乐表演程序，所有程序都包括在内了。

6．5．常用存储地址

本章主要叙述Logo程序中有关地址，主要是与.EXAM1NE 和.DEPOS1T 直接卦句的改良Logo语言部分，致于Logo与汇编语言程序接口问题已经在6．2节讲过了。在Logo实用盘上包括有ADDRESSFS 地址文件在内的地址的真正函意和具体数值。地址的具体数值由Logo语言释放而产生的变化。在Logo语言中执行READ"ADDRESSES 像一般Logo变化规律那样用整数值去定义地址。（在文件中说明地址数值应以十六进制标识）在使用这些地址的时候，应在前边加这样的字符：.EXAM1NE :EPO1NT。

零页位置（*Page Zero locations*）

EPO1NT　在编辑缓冲器内字符位，用于编辑程序及EDOUT 程序中。

ENDBUF　在编辑缓冲器内最后一个字符的地址。软盘储存程序（见SAVMOD）存放位置由$2000 到指定地址。见7．3中举例。

SAVMOD　如藏存的内容在零位，则READ和SAVE工作正常。如果非零位，SAVE所存放的无论是什么，也可能另有过程内容及过程各文本，都进入编辑缓冲器内，同时READ将重新从软盘上放在编辑缓冲器内，只是不再加以处理而已。详见第7．1节。

BKTFLG　如它存在包括位置1的话，Logo将要打印出显示反回去阅读的目标一部分。经常用于打印显示EDOUT 程序，详见7．3节。所有被打印的列表都必须包括在方括号内。没有括号就

被打印成“句子”形式。在打印显示过程名时，必须加引号，PO NAMES 打印出的Logo是个变量，如在MAKE “VARIABLE 3当中，代替“ VARIABLE 数值的是3。

NOINTP　用以控制特殊“中断”字符键CTRL—F ，CTRL—G ，CTRL—S ，CTRL—T ，和CTRL—W 的动作。如果存放在包括有零位的话，计算机在DRAW和NO DRAW 工作状态时，这些特殊字符键工作正常。如包括1的位置，则这些特殊字符键无专门意义，只是READCHARACTER操作中，CTRL-Z 和CTRL—SHIFT—M 尚能工作。如放在NOINTP当中255位，它们便失效了。

CH，CV　光标的行与列位置，.EXAMINE ：CH输出光标水平位置，详见CURSOR 指令说明。

OTPDEV　用以输出字符的程序地址。

INPDEV　类似OTPDEV 只是输入字符的程序地址。

USHAPE　用以指出用户定义乌龟的形状。见第五章。

SSIZE　乌龟形状大小尺寸，或是用户定义的乌龟形状的大小。一般不用的情况＝1。

INVFLG　表示字符黑白颜色。（一般不用，其内容＝255），白底黑字（内容＝0），或闪烁状态（内容＝64）。

NARG2　.CALL 第二个输入，四个字节。见6。2节。

NARG1　由用户利用的第一个临时位置。至ANSN4+3

的全部储存位置都是由用户编程用的。见6.2节。

ANSN4　由用户利用的最后一个临时位置。它和下边三个字节都是空的。

USERPZ　由用户利用的第一个永久性零页位置。由此到$FF 位Logo 是不用的。

H1MEM　(.EXAM1NE :H1MEM)+256 * (.EXAM1NE :H1MEM+1) 输出由用户机器语言程序所利用的最高地址。它用MAXF1LES1 DOS 命令放在$9AA5 处。见VZZZ .

其它常用地址

OTPFX2　由.CALL 输出的转移地址，在NARG2 至NARG2+3的整数位。见6. 2节。

OTPF1X　类似OTPFX2，但Logo所执行的是整数值，它是用Y寄存器指出零页开始位置，占用四个顺序字节。

CTPTRU　转移到程序输出“ TRUE。

OTPFLS　输出“ FALSE。

GETRM1　从这个位置上调出，或又存在这个位置上，分两次写；例如LDA GETRM1, LDA GETRM1, 使得Logo位置扩大储存量，ROM 监视程序失效。

K1LRAM　引用ROM 的Apple 监视程序位置，除非由。CALL 明确使其失效，否则始终它是处于工作状态的。

PPTTP	可选择的脱离用户机器语言程序，用 Logo 指令 TOPLEVEL 执行地址转移。它常用于发生错误条件时，执行 Logo 程序用。用它去继续执行。CALL程序是不适当的。
COUT	Logo 正常屏幕字符输出程序。在屏幕上打印显示字符为 A 的状态。
EDOUT	在编辑缓冲器内 A 寄存器的字符程序。用 EPO1NT 和增量 EPO1NT 指出存放 A 的位置。当 EPO1NT 比 $3FFF 大的时候，不再作什么事情。可用 OUTDEV 使 Logo 文本直接放在编辑缓冲器中。参见 7．3 节举例。
PNTBEG	在编辑缓冲器开头，清除 EPO1NT 的程序。用在缓冲器第一次的输出之前 (用 OUTDEV :EDOUT)。见 7．3 节举例。
ENDPNT	将 ENDBUF 放在 EPO1NT 程序中。用以终止缓冲器显示。见 PNTBEG, EPO1NT, ENDBUF。
B E L L	使喇叭发声的程序，用 pR1NT1CHAR 7 使 Logo 产生喇叭发声的命令。
HOME	光标原位，并清除屏幕。
CLREOP	从光标位开始清除至屏幕终点。
SCROLL	卷挠。
CLREOL	清除至行尾。
F1LLEN	包括被调出最后文件的长度。
F1LBEG	被调出最后文件的起始地址。

VZZZZZ　　(.EXAMINE :VZZZZZ)+256*.EXAMINE :VZZZZZ+1 输出用户机器语言程序所利用的最低地址。尽管这个地址比$99A0 还低。在中间范围不置码，直到让Logo用储存空间为止。

第 七 章

其 它 资 料

7．1．Logo文源编辑系统

Logo系统能够阅读和存放包括Logo过程在内的各种计算机文件。Logo系统经过进一步改进，还可以阅读及存放各种文本文件。当然指的是Logo编辑系统可以用到的文本编辑文件。还可以暂时地储存在软盘上，并没有“自起动”Logo程序。

一般用TO引导进入编辑状态，在TO的后边跟着是过程名。在Logo文本工作状态，储存器的编辑缓冲器必须是空的，只有在打TO之后，按RETURN 键才能达到进入空的编辑缓冲器的目的。从此，才可用编辑和编辑过程同样方法，去编写和编辑文本。存放文本时必须脱离编辑状态。不能按CTRL—C 去定义过程，而是要按CTRL—G才行。脱离编辑状态，文本的存放和显示，以及程序内容都不会有改变。然而过程编辑之后，要进入绘图方式的时候，文本就消失了。因此，SAVE必须像软盘上的文件一样将文本存放在缓冲器内。其它文件都要有一个文件名如SAVE“MYFILE。但SAVE必须用不一样的名，将在下边介绍。在实用盘上的TEXTED1T 文件是综合这些概念而设计的，使用更方便一些。有关TEXTED1T 的解释，详见实用盘的有关章节。

重要的是SAVE命令一般是把过程及过程名，通过编辑缓冲器存放到暂存器内，之后再存放到软盘缓冲器里。如果要在软盘上写文本，要先调回到编辑缓冲器里去，可对这种“存放方式”加以标志，通过它控制READ和SAVE，而文本在整个过程中也不会发生任何变化。一般情况下，SAVMOD 的储存位置应为零。当放入任何非零的数字时，将进入暂存器以前的内容，存放在编辑缓冲器里。READ从软盘

上阅读文本后，也不会对编辑缓冲器有影响。但是在GOODBYE之后，再有“存放方式”的标志，Logo语言便停止工作。实际上这个标志的地址是在实用盘的ADDRESSES 文件里建立的，只要打READ “ADDRESSES就可以访问这个标志的地址。在编写文本之前，它就失效了。在打CTRL—G 之后，存放文件之前，应打

.DEPOSIT :SAVMOD 1

只要用Logo过程编辑系统去编辑文本，就要从打这段字符处开始。如果要阅读或存放什么程序过程，或是过程名，应打

.DEPOSIT :SAVMOD 0

便回到正常工作状态。

打过.DEPOSIT :SAVMOD 1，实用盘从READ “ADDRESSES之后，地址文件将回到后边的数据上工作，或是从READ“MYFILE之后的工作状态。当然在打ED之后，必须按RETURN 键，如果在阅读文件之后，不打TO，缓冲器仍然空着，如要再读文件重新开始。

7．1．1．文件打印

Logo屏幕编辑程序如果不用文本编辑，打印文件也就没有意义了。当打印机插入SLOT位，要打印编辑缓冲器的内容，用如下程序。在使用这个程序之前，必须在Logo的ADDRESSES 文件中，列出ENDBUT 的数值。

```
TO HARDCOPY
  OUTDEV :SLOT
  PRINTMEM 8912 250*(.EXAMINE :ENDBUF+1)!
  +.EXAMINE :ENDBUF
  OUTDEV 0
END
```

```
TO PRINTMEM :FROM :TO
  IF :FROM > :TO STOP
  PRINT1 CHAR .EXAMINE :FROM
  PRINTMEM :FROM+1 :TO
END
```

当READ读完这个文件之后，打HARDCOPY 就会立刻把文件内容打印出来。如果用Logo屏幕编辑系统作文本编辑工作，在打HARDCOPY 之前，打CTRL—G 编辑程序命令，将把编辑缓冲器的内容打印出来。

当然，用这种方法打印程序内容，不是效率很高的。

7．2．自起动问题

当用SAVMOD 方式，存放在软盘上过程和过程名的尾部，还可附以文本。每次使用这个程序调用文件的时候，通常要自动起动。例如用SETUP 过程时，每次都自动执行所阅读叫GAME的文件。通过SEDUP 命令，每次都自动执行所阅读的GAME文件完成所安排的任务。

当然也需要定义GAME所有的文件。打EDITALL全部暂存器进入编辑缓冲器工作状态。可用CTRL—F 终止缓冲器工作。在工作当中插入命令就可以直接执行（如SETUP）。打CTRL—G 则脱离编辑状态。如打

```
.DEPOSIT :SAVMOD 1
SAVE "GAME
.DEPOSIT :SAVMOD 0
```

无论何时调用GAME文件，程序过程都将被定义，SETUP 指令将执行包括附带在编辑缓冲器最后的全部过程。

7．3．打印软盘文件

另外用SAVMOD 也可写Apple DOS 二进制文件的Logo程序。Logo用二进位文件去储存过程，即可写文本文件，Logo还可以阅读以后所有的程序。用DEFINE 命令定义程序过程当然比较方便，可是它必须把要打印的过程文本调进文件中去。例如，在一个指定的文件中要存放一个过程列表，可用如下协议写软盘文件。（实用盘上的PSAVE 程序就用这种方法。）

如果仅仅是为特殊用途而设置的程序过程，Logo语言就不能阅读软盘文件，而只能打印一般的有关信息。

下边的Logo程序DPRINT 是为打印软件文件而提供给用户的。它执行有文件名的OPEN输入条件，之后再用DPRINT 命令就能打印缓冲器文件。打CLOSE 则封闭这个文件，因为存放在软盘的文件是“打开”的状态的。在文件量“打开”状态的时候，不能使用编辑系统及绘图方式，打印字符不能超过8192个，超出这个范围的字符就打印不出来了。

汇编语言程序ENDBUF 位是Logo编辑缓冲器终点的指标。编辑缓冲器从$2000 位开始，扩大到$3FFF 。PNTBEG只不过把ENDBUF 置于$2000 位。EDOUT 程序要求有一个字符A，同时把它放到EPOINT 所指定的位置上，并且增补EPOINT 程序。软盘上存放汇编程序从$2000 到ENDBUF 。通过ENDPNT 程序的CLOSE 指令修订EPOINT 的ENDBUF 。

当写出一个文件，Logo又必须返回去阅读，并把数据转释之后，才能执行打印的命令。把要打印的列表必须用括号括起来，一般Logo打印：

```
PRINT [1 2 [A B ] 3]
1 2 [A B] 3
```

可能会认为打印为：

PR1NT [1 2 [A B] 3]

[1 2 [A B] 3]

把1放在BKTFLG 位置，就打印反回去阅读的内容，重新恢复到0的位置则实现另一种功能。从.DEPOS1T 开始在DPR1NT程序终止。它对其它Logo打印操作没有任何影响。

另外，BKTFLG 还有控制PO NAMES 的作用。当BKTFLG包括1的时候，才执行PO NAMES，按下边的形式打印其命名：

MAKE "NUM 259

关于软盘文件打印的程序，全部列出如下：

```
TO OPEN :FILE
 MAKE "OPEN.FILE :FILE     ; SAVMOD, PNTBEG,
                             EDOUT, EPOINT,
  CALL :PNTBEG 0           ; ENDBUF, and ENDPNT
                             ar8 found
END                        ; in the ADDRESSES
                             fil8.

TO DPRINT :THING
 OUTDEV :EDOUT
 PRINT :THING
 OUTDEV 0
END

TO CLOSE
 .CALL :ENDPNT 0                    ; uPdates
                            end—of—buffer
                            pcfnter.
```

```
. DEPOSIT : SAVMOD 1
 SAVE : OPEN.FILE
 ERNAME "OPEN.FILE
 . DEPOSIT : SAVMOD 0
END

TO OPEN.FOR.APPEND : FILE
 MAKE "OPEN.FILE : FILE
 . DEPOSIT : SAVMOD 1
 READ : FILE
 . DEPOSIT : SAVMOD 0
 . DEPOSIT : EPOINT . EXAMINE : ENDBUF
 . DEPOSIT : EPOINT+1 . EXAMINE : ENDBUF+1
END
```

7．4．各种系统参数

本章主要介绍有关Logo语言比较秘密的和有关特殊资料。对一般Logo的用户也没有必要知道这些，因为这些题目可能更深奥了一些。

关于绘图屏幕

乌龟作图直接向上引出的轨迹线，在以屏幕一边为限环绕动作之前，能走121步，向下能走120步，向左740步，向右140步。当然可以改变屏幕显示的长宽比，用.ASPEST 命令，可在水平范围保持不变的情况下，改变垂直范围的大小。

数字范围

Logo语言中最小的数字在1N38位完成运算，最大数字在9.9999E38位。非浮点运算时最大正数为2147483647，最大负数为—2147836 47。

ASCll 码值

它反应Logo字符与0－255之间的数字时应关系。ASCll指令是一个字母即一个字，输出是字母和数字结合方式。CHAR指令可使单个字母为倒数。0在Logo语言中有特殊用途，它表示一个空字。一个字不可能有空字，所以说空字就不是个字。

READCHARACTER指令可缩写为RC，从键盘上读一个键字，则输出一个单字母的字。按“中断”键之后，RC当然没有输出，不管Logo语言工作中处于绘图或非绘图状态；CTRL—F，CTRL—S，CTRL—T，CTRL—SHlFT—M，CTRL—W，CTRL—Z 和CTRL—G 都具有这种功能。

下边列表说明有关按键与ASCll 码值的对应关系，打PRlNT ASCll RC，再按要寻找的那个键，就可知道它的ASCll 码。

键	ASC ‖码值
ESC	27
向左箭头键	8
向右箭头键	21
CTRL—SHlFT—9	95
CTRL—SHlFT—N	30

有时是不准用CTRL—G 的，即使为了使用其中断字符，但它还有其它功能的。鉴于这种情况，除CTRL—Z 和CTRL—SHlFT—M之外，考虑到要中断字符的作用，又要抛弃那些特殊问题的影响，在Logo语言在.DEPOSlT 中，把1放在NOlNTP 位，中断字符的作用失效；把0放在这个位置时，则恢复了它的功能。

在中断字符功能失效的时候，READCHARACTER指令在按任何一个键的时候，都会有输出（只有CTRL—Z 和CTRL—SHlFT—M

除外）。最典型的是在实用盘的瞬时INSTANT程序时，系统有这个特徵。这个程序能在中断字符失效的情况下，同时还保留字符本身的意义，即保留一般Logo语言中字符的特殊功能。

另外要注意，在中断失效的时候，在未返回标题行之前，程序什么动作也不能完成了。在执行一个过程的时候，如果用户按 CTRL—C，此时 OUTDEV 将切换到另一个输出设备上去，所有要输出的任务都直接换到这一个输出设备上去了。

下边的程序通过NOINTP 的作用，在执行当中，不必担心由于出现“STOPPED!”和“PAUSE”停止的信息，而影响传送给输出设备。

```
TO TCMD :CMD :ARG
.DEPOSIT :NOINTP 255
OUTDEV 7                    devite 的 slot 7。
(PRINT :CMD :AGG)
OUTDEV 0
.DEPOSIT :NOINTP 0
END
```

行的长度

在行编辑程序当中，Logo语言每打印显示一行的字符不得超出256位。另外，在列表输入要执行RUN 和重复REPEAT 的时候，以及第二个输入的每一个子表，都必须遵守定义 DEFINE 的限制。

在屏幕编辑状态时，由TO引出过程名，所打的各行内容可以任意长，只要在编辑缓冲器里能容纳的长就可以。同样由软盘上读的文件也可以是任意长。

Logo语言的储存问题

Logo语言的储存功能比其它任何一种语言都更为有效。储存一个过程就像一行列表一样方便。这一行可以是几个列表和字。每一个字都要求一个储存空间，它和字的字符多少没有关系。解释过程和不同过程名的长短也不能形成什么障阻。

当Logo超出计算机储存空间的时候，即进入一个所谓garbage collectien垃圾箱里去，简而言之，Logo超出储存空间的内容再不能使用了。因此可把所有要储存的作成一个大的列表，这样计算机便可以当成一个列表而储存起来。

当计算机数据进入垃圾箱之后，Logo语言再也不能工作了，因为它影响到程序的实时响应是十分重要的，为此十分头疼，只好用.GCOLL指令使程序自然暂停。

7．5．储存组织表

这新表说明Logo系统使用Apple Ⅱ 计算机地址空间的情况。

Nil:	$00C0 — $0003:	$ 3 bytes	The empty list
Misc.:	$0004 -- $07FF:	$ 7FC bytes	Buffers and impure.
Stacks:	$0800 — $1BF5:	$13F6 bytes	Stacks (PDL. VDPL)
Vectors:	$1BFC — $1BFF:	$ A bytes	Re—entry addresses
Othercode:	$1C00 — $1FFF:	$ 400 bytes	I/O subroutines
Buffer:	$2000 — $3FFF:	$2000 bytes	Editor/graphics buffer
Logo:	$400C — $999F:	$599F bytes	(23K bytes) Logo code
User:	$99A0 — $DAA5:	$ 105 bytes	User Machine Code
DOS:	$9AA6 ← $BFFF:	$255F bytes	DOS code. buffers
I/O:	$C00C -- $CFFF:	$1000 bytes	Mapped I/O addresses
Nodearray:	$D000 — $FG5F:	$2650 bytes	(2456. nodes) Nodespace
Typearray:	$F660 — $FFF7:	$ 998 bytes	(2456. bytes) Type—codes
GnosLmem:	$D000 — $DFFF:	$1000 bytes	Static storage
Unused:	$FFF3 — $FFF9:	$ 2 bytes	
Intrpts:	$FFFA — $FFFF:	$ 6 bytes	Interrupt vectors