微型计算机软件资料汇编

第四册

机械工业部计算中心
合肥工业大学微型机应用研究所 编译

GRAPHICS

muLISP

rDBMS

CP/M

机械工业部仪表局情报室
《仪表工业》编辑部

微型计算机软件资料汇编

第　四　册

机械工业部计算中心
合肥工业大学微型机应用研究所　编译

机械工业部仪表局情报室

编 译 出 版 说 明

本资料汇编收集了近期从国外引进的微型计算机软件，包括CP/M操作系统及其支持程序、高级程序设计语言、数据库管理系统和应用软件包，可以在Zilog Z80系列、Intel 8080系列微型机上使用，并已在H/Z89微型机上验证。

收集在资料汇编中的有：

微型机操作系统 CP/M2.2；

小型关系数据库管理系统 CONDOR SERIES/20；

高级语言 COBOL-80、PASCAL/MT、FORTRAN-80、MBASIC、PL/I-80、C、muLISP；

编辑和字处理系统；

分类/合并程序；

库存管理程序；

图形软件包；

远程终端仿真程序等

这些资料大部分以使用手册形式提供，可作为微型机用户手册，也可供计算机系统软件和应用软件人员以及大专院校有关专业师生学习参考。

本资料汇编由机械工业部仪器仪表工业局组织，机械工业部计算中心和合肥工业大学微型机应用研究所编译，刘运基、康兴钨审校，并请旅美学者赵鉴芳教授指导审定。在编译出版过程中，得到许多同志的大力协助，谨在此表示谢意。

由于编译者水平所限，难免有错漏之处，敬请读者指正。

本资料汇编共分六册，由机械工业部仪表局情报室《仪表工业》编辑部陆续出版。

目　录

PL/I-80应用指南和语言手册

SuperSoft C 编译程序使用手册

PL/I-80 应用指南和语言手册

第一章 PL/I-80应用指南

第一节 绪 言

PL/I-80是为在Digital Research CP/M和多道程序设计MP/M操作系统下进行应用程序设计的一个完整的软件程序包。它是以ANS PL/I 标准化委员会X3J1定义的新的子集G语言为基准的。这个子集包含了整个PL/I的必要的应用程序设计结构，而去掉了不常用或冗余形式，使之更便于进行程序设计，同时简化了编译工作。

PL/I-80象所有程序设计语言（及大多数自然语言）一样，通过研究实例很容易学会。本章主要目的是介绍编译、连接和程序执行的技巧以及语言的一些有用功能，并且有详细的实例程序，说明输入/输出处理、科学计算、商业应用及表处理。

学习PL/I-80的最好方法是读有关教科书，研究实例程序，必要时查阅参考手册，在弄懂一个样本程序的操作后，可以修改这个程序以提高它的操作能力并增加你的语言经验。

PL/I-80系统软盘不包括CP/M操作系统，所以首先应制备PL/I-80程序的拷贝作日常使用，并在前两个系统磁道上生成CP/M系统。将新建立的软盘装到驱动器A，引导CP/M，打入DIR命令可以找到下列几类文件：

COM　CP/M命令文件或组合程序（PLI.COM是其中之一）
DAT　缺省数据文件类型
IRL　索引浮动码（PLILIB.IRL是库文件）
OVL　PL/I-80编译程序复盖（PLI0.PLI1和PLI2）
PLI　PL/I-80源程序（即OPTIMIST.PLI）
PRL　页浮动目标码（用于MP/M分区）
PRN　打印机磁盘文件（磁盘上的程序清单）
REL　浮动目标码（如用户开发的程序）

含有可打印字符的文件只有“PLI”源程序及“PRN”打印机列表文件。PL/I-80系统盘里包含了不同的程序，包括本手册中的实例以及其他更复杂的程序。开始，先试着运行一下已经编译和连接到PL/I-80运行库的程序。命令格式为：

```
OPTIMIST
```

这时，OPTIMIST程序被装入并响应：

```
What's up?
```

可以打下面这个句子来回答：

```
None of these programs make Sense.
```

（须用一句号结束输入，接着打回车）。从OPTIMIST得到响应后，还可以再打入一些句子，然后打control-c停止OPTIMIST。

OPTIMIST是一个PL/I程序，它以源码方式存于PL/I-80系统软盘中。显示这个程序可以打入命令：TYPE OPTIMIST.PLI

作为一个例子，按下面的步骤对一个OPTIMIST程序进行完整的编译和测试。注意，虽然OPTIMIST程序可在任意存储容量下运行，但PL/I-80编译程序运行至少需要48K的

CP/M系统。PLI.COM 和覆盖文件一定要在缺省磁盘上，否则当启动编译程序时将出现错误信息“NOFILE:PLIO.OVL”，可打入

PLI OPTIMIST

来编译OPTIMIST程序。

编译程序将用三个步骤处理程序，称为“扫描”。其信息标记是：

NO ERROR(S) IN PASS 1
NO ERROR(S) IN PASS 2
END COMPILATION

如果检查目录表，便会找到OPTIMIST.REL文件。它包含了由PL/I-80编译程序为OPTIMIST程序生成的浮动机器码。如果需要，可用列表任选重新编译，这样在编译时可以查看程序。要达到上述目的，只要打入：

PLI OPTIMIST SL

编译程序和以前一样进行，但这次在最后一遍扫描中要产生程序清单。

编译产生的浮动机器码不是直接可执行的，所以必须将REL文件与PL/I-80运行子程序库连接，这可通过打入“LINK OPTIMIST”来实现。LINK-80程序产生一个OPTIMIST.COM文件，它替换原来软盘上的同名文件，新的OPTIMIST程序可以用同样的方式操作。

第二节　PL/I-80系统操作

通常，PL/I-80编译程序读入在CP/M或MP/M下用标准的编辑程序（ED）建立的程序文件。程序先由PL/I-80编译程序处理，再用LINK-80连接，然后进行测试。例如一个简单的工资单程序的编译测试，如图2-1所示。编译程序通过进行前两遍扫描并列出含有错误的每一行，左边带行号，还有一个简单的错误信息，并在错误行下面位置上有一个“?”。无论在什么情况下，都可在控制台上打入一个回车来终止编译过程。这个功能对于错误诊断数过多或想在继续执行前作一改正是很有用的。程序的行号列在左边，后面跟着字符a～z，表示每一行嵌套层。主程序层为“a”，每个嵌套的BEGIN由前进一个字符表示，而每个嵌套的PRODURE层由前进两个字符表示，其后是每行相对的机器码地址，它是用四个十六进制数表示的。这个地址对于确定为每个语句生成的机器码的数量和为各程序行生成的相对机器码地址是很有用的。源语句打印在相对机器码后面。

在启动编译程序命令行时给出的SL参数称为“编译程序开关”，用于启动列表任选。编译程序开关表在下面列出。每一命令字符跟在命令行给出的符号“$”后，在美元符（$）后面最多可有七个命令字符。没有指定参数时，缺省值表示编译不产生清单，所有错误信息送到控制台。

B　内部子例程跟踪，给出被PL/I程序调用的库函数

D　磁盘文件打印。将列表文件送到磁盘，文件型为PRN

I　交叉列出源码和机器码。将编译产生的机器码以伪汇编语言形式表示

L　列出源程序。产生带有行号和机器码地址单元（由I开关自动设置）的源程序清单

```
PL/I-80 V1.0, COMPILATION OF:WAGE
L : List Source Program
   NO ERROR(S) IN PASS 1
   NO ERROR(S) IN PASS 2
PL/I-80 V1.0, COMPILATION OF:WAGE
   1 a 0000 payroll:
   2 a 0006      procedure options(main);
   3 a 0006
   4 c 0006      declare
   5 c 0006          name (100) character(30) varying,
   6 c 0006          hours (100) fixed decimal(5,1),
   7 c 0006          wage (100) fixed decimal(5,2),
   8 c 0006          done bit(1),
   9 c 0006          next fixed;
  10 c 0006
  11 c 0006      declare
  12 c 0006          (grosspay, withhold, netpay) fixed decimal(7,2);
  13 c 0006
  14 c 0006      /* read initial values */
  15 c 0006      done='0'b;
  16 c 000B          do next = 1 to 100 while(^done);
  17 c 0023          put list('Type ''employee'', hours, wage:');
  18 c 003A          get list(name(next),hours(next),wage(next));
  19 c 00A0          done = (name(next) = 'END');
  20 c 00C8          end;
  21 c 00C8
  22 c 00C8      /* all names have been read, write the report */
  23 c 00C8      put list('Adjust Paper to Top of Page, Type return');
  24 c 00DF      get skip(2);
  25 c 00F0
  26 c 00F0          do next = 1 to 100 while(name(next)^= 'END');
  27 c 011F          grosspay = hours(next) * wage(next);
  28 c 0157          withhold= grosspay * .15;
  29 c 0177          netpay = grosspay-withhold;
  30 c 0192          put skip(2) list('$',netpay,'for',name(next));
  31 c 01EA          end;
  32 c 01EA
  33 a 01EA      end payroll;
CODE  SIZE=01ED
DATA  AREA=0ED2
```

图2-1 工资程序清单

N　显示嵌套层。作第一遍扫描跟踪，表示出DO　PROC和BEGIN语句与和其相应的END语句的匹配关系

P　页方式打印。每60行插入换页符，并将列表送打印机

S　显示符号表。表示出程序变量名，以及它们的指派、缺省和扩展属性

PL/I-80允许每个过程分别编译，每次编译产生一个“REL”文件。只有一个过程可包含“options(main)”，这个过程就是模块的主程序，而所有其他子程序要有通常的PL/I过程标题。

PLILIB. IRL文件含有子程序，可以由PL/I80程序调出。如上一节所示，要使浮动机器码与PL/I运行库子程序相连接，可打入：

```
link wage
```

生成一个组合程序。如果是在MP/M系统下操作，命令Link wage〔op〕也生成一个组合程序，但在这种情况下，机器码是页浮动格式，在一个MP/M分区中运行。在第一种情况下，LINK-80产生一个可执行的“wage.com”文件，供在CP/M下执行或在MP/M下的一个绝对段中执行。在第二种情况下，LINK-80产生一个名为“wage. prl”的文件。除机器码文件外，LINK-80还产生一个符号表文件，称为“wage.sym”，它可在SID或ZSID下装入用于纠错。

图2-2列出了一个简单工资程序的LINK-80输出。按照惯例，从PL/I-80库中提出的子程序前面要加“？”符号，以避免与用户定义的符号名相冲突。用两个“/”括起来的是EXTERNAL变量，后面有‘*’号的是未定义的符号。注意，LINK-80采用Microsoft连接编辑格式，这是为了与其他语言处理程序相兼容。但这种格式将外部名的长度限制在6个字符，所以尽管内部变量名可以长达31个字符，但要保证所有外部定义名在前6个位置上是唯一的。

```
A>link wage
LINK V0.4
PAYROL  0100  /SYSIN/ 1F19 /SYSPRI/ 1F3E
ABSOLUTE   0000
CODE SIZE   1DCF (0100-1ECE)
DATA SIZE   10C5 (1F94-3058)
COMMON SIZE 00C5 (1ECF-1F93)
USE FACTOR  4E
```

图 2-2　工资程序的简单连接编辑

采用缺省时，LINK-80不从库中列出“？”符号，如果想列出这些符号，只须打入：

```
link wage〔q〕
```

要执行程序，可打入COM或PRL文件名，如图2-3所示。程序执行并提示控制台输入，如后面I/O节所述，从控制台输入是带有CP/M及MP/M的整行编辑的自由字段。在程序完成回到操作台命令级时，显示信息：

```
End of Execution
```

如在PL/I程序里不显式截获各种运行错误，这些错误会使程序执行终止，并显示下列信息：

error-condition (code), file-option, auxiliary-message

Traceback: aaaa bbbb cccc dddd # eeee ffff gggg hhhh

其中"error-condition"是下列标准PL/I条件之一：

ERROR FIXED OVERFLOW OVERFLOW UNDERFLOW ZERODIVIDE END OF FILE UNDEFINED FILE

"code"是一个错误子代码，它指明错误的起因。当错误产生于I/O操作时，打印出"file-option"并有如下形式：internal=external

其中"internal"是引用了含有出错文件的内部程序名，而"external"是与该文件有关的外部设备或文件名。当上述信息不足以指明错误时，就打印出"auxiliary-message"。最后"traceback"段列出多达8个内部堆栈元素，以便帮助标识产生错误的程序语句。如果堆栈中超出8个元素，则把最上面的4个元素置于"#"的左边，最下面的4个元素置于右边。在上面所示的形式中，元素aaaa对应于堆栈顶部，而hhhh对应于堆栈底部。主程序错误语句由hhhh值确定，除非错误语句已用一个字符或十进制临时值填充栈底。

图2-4表示一个工资程序的执行，它给出了一个诊断形式的例子。在这个例子中，第一行控制台输入已正确打入，但第二行用文件结束标志（control-z）结束控制台输入，对标

```
A>wage
Type 'employee',hours,wage:'Sidney Abercrombie', 35,6.70
Type 'employee',hours,wage:'Yolanda Carlsbad', 42,7.10
Type 'employee',hours,wage:'Ebenizer Eggbert', 30,5.50
Type 'employee',hours,wage:'Hortense Gravelpaugh',40,6.50
Type 'employee',hours,wage:'Franklin Fairweather',10,15.00
Type 'employee',hours,wage:'Tilly Krabnatz',32,4.10
Type 'employee',hours,wage:'Ricardo Millywatz', 45,7.20
Type 'employee',hours,wage:'Adolpho Quagmire', 60,4.30
Type 'employee',hours,wage:'Pratney Willowander',43.5.50
Type 'employee',hours,wage:'Manny Yuppgander', 40,3.25
Type 'employee',hours,wage:'END',0,0
Adjust Paper to Top of Page,Type return
$   199.33 for Sidney Abercrombie
$   253.47 for Yolanda Carlsbad
$   140.25 for Ebenizer Eggbert
$   221.00 for Hortense Gravelpaugh
$   127.50 for Franklin Fairweather
$   111.52 for Tilly Krabnatz
$   275.40 for Ricardo Millywatz
$   219.30 for Adolpho Quagmire
$   201.03 for Pratney Willowander
$   110.50 for Manny Yuppgander
End of Execution
```

图 2-3 工资程序的执行

准的控制台输入文件SYSIN，产生了END OF FILE条件。在本例中，连接到SYSIN的外部设备是操作员控制台，由CON表示。

```
A>wage
Type 'employee',hours,wage:'Sally Switzwigg',23,3.10
Type 'employee',hours,wage:^Z
END OF FILE(1),File:SYSIN=CON
Traceback: 0930 08DB 0146 3300 # 1F07 040A 8082 0146
End of Execution
```

图 2-4　工资程序的错误追踪

Traceback表示出最低的堆栈单元为0146（十六进制），对应于错误的主程序语句。回过来再看图2-2，PAYROL程序地址在左上角表示为0100，它是CP/M下通常用的起始单元；差值0146－0100＝0046是产生错误的相对位置；图 2-1 清单表明地址0046落在18行旁边列出的（003A～00A0）代码地址之间，因此错误就发生在这行里。

第三节　PL/I-80程序设计特点

在研究PL/I-80程序设计的细节之前，有必要讨论一下它的程序设计特点。PL/I是一个“自由格式”语言，也就是说编程序时可不考虑列的位置及特殊的行格式，每行长度可达120字符（由回车结束）且可顺次逻辑地连接到下一行。编译程序从第一行读源程序到最后一行，而不管行的界限。这种自由的表示法，要求程序员遵守习惯的编程规则，以使别的程序员可容易地阅读及了解这些程序。专业程序员都知道，仅有一个产生正确输出的子程序是不够的，程序还必须在格式上一致，并可分成容易理解的逻辑段。一个程序是从它的结构及其功能评价的。

下面给出的规则说明了一组编程约定，本手册中全部实例都将使用这些约定。

首先，要注意编写PL/I程序既可用大写也可用小写。在内部，PL/I编译程序将所有字符串引号以外的字符变成大写。通常我们更喜欢整个程序用小写，因为这样可减少程序密度及增加可读性。其次，整个PL/I用缩入空格来开始各种说明和语句。为了简化，PL/I编译程序将制表标记（Control-I字符）扩展到每四列一个，然而我们知道，CP/M实用程序如ED，将标记扩展到8的倍数列的位置上，所以在编辑和显示操作时，行变得较宽些。但要注意，当扩展的行长度超过120列时，将发出TRUNC（截断）错误信息。外部分程序级的程序语句起始于第一列位置。由一个DO，BEGIN或PROCEDURE组开始的每个后继分程序级开始于一个新的缩入空格层，可以是四个空格也可以是一个制表标记。同一组内的语句在相同的缩入层给出，过程名和标号单独在一行中。一个IF语句后面应直接跟着条件和关键字THEN，下一条语句缩进去放在下一行。当IF语句有相关的ELSE时，它起始于与IF相同的级，跟在ELSE后面的语句缩进去置于下一行。最后，说明语句格式是将关键字DECLARE单独放在一行，接着在下一行缩进去，写出要说明的元素。应该避免复杂属性的因子分解，因为它减少了程序的可读性。为使段落清楚，可加入空行（即，只含有回车的行），空行常常用于划分逻辑上独立的程序段。许多较长的PL/I关键字有缩写形式

(如DCL等于DECLARE)，因为缩写形式的用法不一致会造成程序不易懂，所以在一个程序设计方案里或用全称，或用缩写形式，而不能两种一起用。

一般说来，可将大程序分成一些逻辑组或“模块”，每个模块完成一个特定的基本功能。这些模块用PL/I的子例程表示，这些子例程，或者是局部的或者是外部定义的。局部子例程成为同一个主程序或子程序的一部分，而外部子例程是单独编译并用LINK-80连接的。局部定义的子例程放在程序末尾以便使程序开始部分只包含说明和调用局部子例程的顶层语

```
PL/I-80 V1.0,COMPILATION OF:TEST
L : List Source Program
   NO ERROR(S) IN PASS 1
   NO ERROR(S) IN PASS 2
PL/I-80 V1.0,COMPILATION OF:TEST
   1 a 0000          test:
   2 a 0006               proc options(main);
   3 c 0006               dcl
   4 c 0006                  (a,b,c) float binary;
   5 c 0006                  put list ('Type Three Numbers:');
   6 c 001D                  get list (a,b,c);
   7 c 0056                  put list ('The Largest Value is',
   8 c 007B                      max3(a,b,c));
   9 c 007B
  10 c 007B               max3:
  11 c 007B                  proc(x,y,z) returns (float binary);
  12 e 007B                  dcl
  13 e 008B                  (x,y,z,max) float binary;
  14 e 008B                  /* compute the largest of x,y,and z */
  15 e 008B                  if x > y then
  16 e 0099                  if x > z then
  17 e 00A7                  max=x;
  18 e 00B5                  else
  19 e 00B5                  max=z;
  20 e 00C3                  else
  21 e 00C3                  if y > z then
  22 e 00D1                  max=y;
  23 e 00DF                  else
  24 e 00DF                  max=z;
  25 e 00EA                  return(max);
  26 c 00F3                  end max3;
  27 a 00F3            end test;
CODE  SIZE=00F6
DATA AREA=0044
```

图 3-1　典型规则说明

句。通常规定，顶层语句或局部定义的子例程长度不能超过两页。在初学PL/I-80编程时，可能常常用到带有多个局部定义子例程的主程序，本书中这种例子很多。当应用程序较大时，最好的方法是将程序划分成独立的模块，以便对每个单独的程序段逐段进行编译和连接，这样可减少总的程序编制时间。

在程序中加注释可使程序一目了然，但不要在源文件中到处都加注释，因为这样有损于整体结构。另外，要注意一致性：最好是把注释放在子程序或逻辑语句组前面，在分析程序时，会发现这些注释及格式安排得很好的程序可提供了解程序操作所需要的信息。图3-1中的程序说明了本节中讲的一些规则。

第四节　PL/I-80输入/输出规则

本节我们对PL/I-80 I/O系统作详细讨论，为后面出现的例子提供必要的基础。如果你认为这部分讲得太细，可先看GET和PUT语句，其中讲到了最简单的I/O功能，再看一下后面的实例程序，然后再回来重读这些细节，可以加深理解。

PL/I-80提供一个独立于设备的I/O系统，此系统是PL/I-80程序与CP/M和MP/M文件系统的接口，接口的参数是由OPEN语句及通过GET，PUT，READ，WRITE语句的缺省方式提供的。

4.1 OPEN语句

OPEN语句是任选的，如果没有显式的OPEN，文件用GET、PUT、READ或WRITE存取时，自动执行打开功能。如果不想用缺省文件属性，就必须在文件存取之前显式地打开该文件。OPEN语句的格式为：

```
OPEN
    FILE (f)
    STREAM  RECORD
    PRINT
    INPUT  OUTPUT  UPDATE
    SEQUENTIAL  DIRECT
    KEYED
    ENV (B(i))  ENV (F(i))  ENV (F(i),B(j))
    LINESIZE(i)
    PAGESIZE(i)
    TITLE(c)
```

其属性可以用任意的次序列出。f表示文件常数或变量值，而且必须在OPEN语句里命名。其他所有属性是任选的，并且取下面给出的缺省值。值i及j表示定点二进制(FIXED BINARY)表达式，而c表示一个字符表达式。在同一行中示出的属性是矛盾的，如不包括这些属性，在有多个属性的行中的第一个属性就成为缺省值。最后四个属性取如下缺省值：

```
ENV (B(128))
LINESIZE(80)
PAGESIZE (60)
```

TITLE ('f.DAT')

STREAM文件含有变长ASCII数据，而RECORD文件一般包含纯二进制数据，一个ASCII数据文件的行由插入的回车换行序列来定义。注意，当文件用 ED 程序建立时，每个回车后包含换行。用PL/I-80建立的文件可以含有一系列换行而前面没有回车。在这种情况下，当遇到换行时就断定是行的末尾。PRINT属性只适用于STREAM文件，而通常假定最终数据显示在行式打印机上。

在OPEN语句出现时要求INPUT文件已经存在，而 OUTPUT 文件如果存在则先删除，并在OPEN语句中建立。UPDATA文件不能有STREAM 属性，并且可被读和写。如果UPDATA文件不存在，则建立该文件。

SEQUENTIAL文件是从头至尾读写的，而DIRECT文件则可随机存取。DIRECT文件自动接受RECORD属性。

KEYED文件可通过使用键值来存取，并自动接受 RECORD 属性。在PL/I-80中，KEYED 文件是简单的定长记录文件，这里的键是被存取记录的相对记录位置（根据定长记录大小计算）。

ENV（环境）属性定义定长或变长文件及内部缓冲容量。ENV(B(i))使得I/O系统设置缓冲存储器为 i 字节，其中 i 内部补足到128字节的倍数。在这种情况下，假设文件是变长记录的，因而不能有 KEYED 属性。

ENV(F(i))定义一个记录长度为 i 字节的定长记录文件，它在内部补足到 128 字节的倍数。为了遵守PL/I标准，也需要将定长记录文件定义为KEYED。这时，缺省的缓冲区容量是 i 字节补足到128字节的倍数。

ENV(F(i), B(j))定义一个含有 i 字节定长记录及 j 字节缓冲区容量的文件（i、j 按上述方法补足）。注意，可以指定定长记录的长度大于缓冲区容量，另外，要包含 KEYED 属性以便与标准PL/I保持兼容性。

如果指定KEYED属性，则记录长度须用ENV(f(i)) 或 ENV(F(i), B(j)) 形式给出。而且，PL/I-80要求所有的UPDATE文件用DIRECT属性说明，以便可以查找单个记录。在用缺省值后，要加入下面的属性：

```
          SEQUENTIAL  ---→  RECORD
              UPDATE*  ---→  RECORD
               KEYED** ---→  RECORD
DIRECT ---→ KEYED**    ---→  RECORD
               PRINT   ---→  STREAM
                       ---→  OUTPUT
```

* 在PL/I-80中，UPDATE也必须是DIRECT

** 在PL/I-80中，KEYED必须有ENV(F(i)) 或ENV(F(i), B(j))

这就是说，将RECORD属性加到SEQUENTIAL、UPDATE和KEYED 文件中，而将STREAM属性加到PRINT 文件中。PRINT文件也将被自动地给予OUTPUT属性。KEYED属性加到DIRECT文件中（依次加RECORD属性）。

OPEN语句本身不能含有不相容的属性，也不能通过缺省或蕴含的方法获得不相容的属性。

这即意味着，如果要读一个含有ASCII字符的文件，必须将它定义为STREAM文件，否则它必须是RECORD文件。一般来说，这都是将要遇到的。如果要进行随机存取，可将文件定义为DIRECT并用ENV定义记录长度。如果要读键值，则可将文件定义为KEYED，并去掉DIRECT属性。

LINESIZE任选项仅仅适用于STREAM文件，它定义了输入、输出行的最大长度。PAGESIZE任选项仅适用于STREAM OUTPUT文件，它定义了页的长度。

TITLE(c) 任选项允许一个内部文件名与一个外部设备或CP/M文件之间建立程序上的连接，若没有指明时，外部文件名取文件引用值，文件型为“DAT”。否则对字符串c求值以产生设备名：

```
$CON    系统控制台
$LST    系统列表设备
$RDR    系统阅读设备
$PUN    系统穿孔设备
```

或磁盘文件名：

```
d: x.y  Disk d, File x.y
```

其中“d:”是任选的设备名，而x和y分别表示文件名和文件类型。注意，x和y可以是$1或$2，如果指定了$1，则从命令行得到第一个缺省名并将它填入到$1位置上。同理，$2从第二个缺省名得到，并填入到它所在的位置上。文件名x不能是空白，而且x，y或d也不能含有“?”符号。物理I/O设备$CON、$RDR、$PUN和$LST只能作为STREAM文件打开，$RDR必须有INPUT属性，而$PUN和$LST必须有OUTPUT属性。

注意，当一个OPEN语句引用一个已经打开的文件时，该语句被忽略。

CLOSE语句的格式是：

```
CLOSE FILE(f);
```

其中f是文件变量或文件常数。所有打开的文件在程序结尾或执行STOP语句时自动关闭。

用STREAM属性打开的文件可通过GET或PUT语句存取，而由RECORD属性打开的文件要通过READ和WRITE存取，但后面要讲到一个例外情况。

4.2 PUT LIST语句

PUT LIST语句的格式是：

```
PUT
    FILE(f)
    SKIP  SKIP(i)
    PAGE
    LIST(d)
```

这里所有元素都是可任选的（但至少要指定一个）。PUT LIST任选项可按任何次序给出，但如指定了LIST任选项，则它必须出现在最后。在上面所示的格式中，f是一个文件变量或常数，i是一个整数表达式。LIST任选项包括一个数据表，由d指出并在后面描述。