S/09微型计算机系统

UNIFLEX PASCAL用户手册

中国计算机技术服务公司

一九八三年一月

目　　录

序

本手册详细说明了本Pascal 编译程序中的特殊实现。但并不是Pascal 语言程序设计的教材。若用户想先学Pascal，请看附录A所列的一些关于Pascal 语言的参考书。

在我们的实现中，首要的目标是产生快速而高效率的目的码。这是我们产生6809的机器码目标程序而不产生需解释的Pascal 代码（P-code）的主要原因之一。我们努力做了许多工作，希望尽可能多地实现在Jensen 和 Wirth 用户手册〔1〕中的性能。同时又兼顾快速，高效这个目标。在最初公布本Pascal 系统时，当时尚未公布标准Pascal，故在本Pascal 系统中，采用Jensen& Wirth 的"用户手册说明"作为标准。所以，本Pascal 系统包括了Jensen& wirth 的用户手册。有关标准Pascal 的句法、语法方面的问题请参考此手册。本系统盘中使用的程序实例是得到纽约Springer-verlag 出版社的允许的。

本手册向用户说明如何编译和运行Pascal 程序。还叙述了与标准Pascal 不同的特性，其中还有未实现的特性及增加的非标准情况。

〔1〕 Kathleen Jensen and Niklaus Wirth, Pascal User Manual and Report, Second Edition (New York: Springer-Verlag, 1974)

如何编译和运行 Pascal 程序

A、用户的Pascal 系统

UniFLEX ™ 操作系统所用的Pascal 系统盘，包括Pascal 编译程序，Pascal 运行程序包和几个示例程序。Pascal 编译程序应包括：

/bin /Pascal
/bin /Pascal.Passes /np6809
/bin /Pascal.Passes /npass1
/bin /Pascal.Passes /npass2
/bin /Pascal.Passes /npass3
/bin/ /Pascal.Passes /npass4
/bin /Pascal.Passes /npass5
/bin /Pascal.Passes /Pascal
/bin/ /Pascal.Passes /Prefix
/bin /Pascal.Passes /Sysprefx
/bin /nrun
/bin /sys-rt
/gen /errors /syrtem
/lib /pascdef

每个系统文件的用途如下：

Δ/bin /pascal 此程序是用来调用编译程序的。编译程序包括一个运行程序包和几个用于各次编译的Pascal 程序。开

始先执行编译程序的主程序——/bin/pascal。passes/pascal。

Δ/bin/pascal。passes/pascal，这是pascal编译程序的主程序，是二进制码的程序。它通过扫描命令行各参量，并检查选择项来控制编译过程，然后就依次调用各趟编译程序，各趟编译程序分别在/bin/pascal。passes/npass1，/bin/pascal。passes/npass2，…，/bin/pascal。passes/np6809 等文件中。

Δ/bin/pascal。passes/prefix 本文件包括Pascal的“标准环境”说明。我们要向你予先声明，这是一个运行过程，这过程包括管理动态存贮块程序，文本文件处理程序，浮点运算软件包等等。有关此文件的细节请看附录B中的完整清单。在编译时，文本文件自动加入到所有的Pascal 程序中。这就是“标准”程序的定义。

Δ/bin/pascal。passes/sysprefx 本文件包括Pascal 的“系统环境”说明。如同对上述标准Prefix 文件一样用予说明的方法说明这是运行过程。这些过程允许用户直接调用UniFlx 操作系统的子程序。包括改变文件的许可权，连接文件，建立文件等工作。这个Prefix 并不包括任何浮点运算过程，细则请见附录C，对每个特定程序来说，只能用一个Prefix

Δ /bin/nrun 和 Δ/bin/sys 这两个是运行软件包，用在标准运行环境和系统运行环境中。在执行Pascal 程序时，由shell 程序调用这 些文件。通常，这些文件不能被用户直接调用。

Δ/gen/errors/system 本文件为Pascal 运行时提供各种错误信息。在执行时，由相应的运行程序读取这些错误信息，并将其显示出来。请注意，它不是文本文件，不能列清单。

Δ/Lib/pascdef 本文件包括各种汇编程序，这些程序在由Pascal 编译程序所产生的程序时，是必不可少的。

8。如何编译和运行Pascal 程序

要执行Pascal 程序，首先必须对Pascal 源程序进行编译，成为用汇编语言写的源程序，然后再用标准UniFlex 汇编程序来汇编此源程序，才能产生二进制目标程序，此时的二进制程序方可执行。这个过程详述如下：

对Pascal 源程序进行编译，其命令的一般形式为：

++Pascal sourcefile [resultfile][+opt ions]

sourcefile 是指要编译的Pascal 源程序文件名，它必须是标准的Uniflex 文件名，一行命令只能编译一个源文件。想要在一行命令中编译多个源文件，将出错。rtsultfile 是选择项。也必须是标准的UniFlex 文件名，它所指的是编译后产生的目标程序。也就是说：它可能是用汇编语言助记符写的程序，也可能是最终可执行的二进制代码程序。如果源文件名是以“P”扩展名结尾，则Pascal 将编译该源程序，若无错误，将自动转向汇编程序进行汇编，产生可执行的二进制代码程序。如果指定了结果文件名，则此文件按此名存放。否则，按与源文件名相同无.P扩展名的名字存放。

例如：++Pascal test.P

将产生一个名为“test”的二进制文件。而

++Pascal tert.P/bin/tertprgm

将产生一个名为“/bin/testgrgm”的二进制文件。如果源文件名没有.P扩展名，则产生的是一个用汇编语言助记符写的

文件。此时如果指定了结果文件名，则按此名存放文件，否则此助记符文件则以源文件名加上“ a ”扩展名为其文件名进行存放。例如：

++Pascal test /usr/bin/tmp

将产生用汇编语言助记符写的文件。“ /usr/bin/din/tmp”

而 ++pascal test

将产生汇编语言助记符文件“ test. a”。应该注意：若盘中在此之前已存在“ test. a” 文件，则 Pascal 将自动删除已存在的文件，并建立一个新的“ test. a” 文件。

上述option 项可以是“ ablcnqu” 七个字母中的任意组合。每一字母都有独立含义。说明如下：

a—禁止对带. a扩展名的源文件进行汇编。

b—禁止生成编译程序的代码。

l—在标准输出设备上列出源程序清单。

c—禁止自动检查运行值。

n—对执行行号准确跟踪。

q—在第一遍编译时，一旦发现错误，将停止以后各遍编译。

u—不用标准 prefix 而用系统 prefix

上述选择项中，有些问题须进一步加以说明。“ c ”只是禁止检查某些运行值，即在运行时将不对标量或非NIL指针作检查。但对数组下标和case 语句是否越界仍要作检查。“ n ”选择项将让编译程序产生一个程序，此程序在运行时，准确地跟踪每一个行号是必不可少的。若运行时出错，则报告准确行号。否则，只报告运行时出错的那个过程的说明语句的行号。

编译程序产生的若是汇编语言助记符文件，则必须先调用这个

汇编文件进行汇编。

```
++asmb{+options} assemblyfile
```

有关句法及可用的选择项细节请参看UniFlex 6809 汇编程序用户手册。若在Pascal 命令中指定了结果文件名，它就是assemblyfile，若未指定，则Pascal 编译程序产生的结果文件，其名就是源文件名加。a扩展名。因此，在asmb 命令行中所用的assemblyfile 应反映这个扩展名。另外，若用户要汇编该程序，还必须用UniFlex 的 perm 命令，对此汇编设置许可字。

最后，用户准备运行无错的，编译过的，且汇编好的Pascal 程序，其命令为：

```
++binaryfile {paraneters}
```

binaryfile 应为汇编后的结果文件，但也可是在编译带有.p扩展名的源文件时，所产生的二进制文件。这时的文件名已去掉.p扩展名。命令行中可选择的参数应是字符串。各串之间可用分隔符空格或“，”来分开。这些参数将传送给用户的Pascal 程序，有关这些参数详见“本系统提供的标准特性”及“非标准特性一章中的“运行其他Pascal 程序”一节中的内容。

为了帮助大家理解，我们举几个例子：

```
++Pascal tests +lcyq
++asmb test.a +lso=testcase
++Perms testcase 0+x
++testcase
++Pascal gsort.p
++gsort <1 data > sorted
```

第一个例子用 3 + 1 cvq 选择项，将编译名为 test5 的 Pascal 源程序。编译完后，汇编 test5.a 文件，并产生名为 testcase 的二进制文件。并设置可执行特性，最后，只要直接打入这个文件名，就可执行之。第二个例子说明 .p 扩展名的应用。若在编译时未发现错误，则自动转入汇编。产生二进制文件，最后在执行该程序时，应输入名为 /data 文件入，运行后，将输出送到 sorted 文件。第二例使用方便，建议程序员采用 .P 扩展名。

本系统未提供的标准特性

为了指出本Pascal 系统与标准Pascal 之间的主要差别。我们先来看哪些在Pascal User Manual and Report中有而本系统没有的特性。我们直接指出“User Manual”中的页数，所以请你在“User Manual” 付本中记下这些差别。在本手册中，为使保留字明显起见，尽管我们的Pascal 编译程序都可接受大写或小写的保留字，我们仍全部采用大写字母。用户仍可用小写字母来写这些保留字。

在“User Manual” 的第9页中讨论了标识符。本Pascal 系统允许用户在使用标识符名时，最多可用160个字符拼写，允许使用大写字母或小写字母，除第一个字符以外，也允许使用底划线符(—) 作为任何一个字符。

在“user Manual” 的第14页中说：只要至少有一个操作数为实型数（其他可能是整型数），则下列运算产生实数值：

* 乘

/ 除（两个运算数均可为整型数，但结果恒为实型）

\+ 加

\- 减

在本Pascal 系统的版本中，在类型混合的表达式中，不能自动将整型数换成实型数。若X为整型数，要当实型数使用，就必须采用CONV(X) 函数将整型数X转换成实型数，CONV(X)是本Pascal 的标准函数。例如下列程序段是不正确的。

```
VARX,y:integer;
```

```
    z:real;
BEGIN
    z:= X/y
END;
```

但若将赋值语句改成：

```
    z:=CONV(X)/CONV(y);
```

则程序就是正确的。

在“User Manual”中的第十六页上指出：在各个程序的开头，应有起首语句（PROGRAM）。本Pascal已在Prefix文件中包括了这个语句，并自动将其置于各个Pascal用户程序之前，因此用户就不必写上此语句。细节请看附录B中的/bin/Pascal.Passes/Prefix和附录C中的/bin/Pascal.Passes/sysprfix完整清单。若用户程序中用了PROGRAM语句，则算出错。

通常，在PROGRAM路句中，要说明Pascal的内部文件名与外部文件名之间的连接关系。因为现在不用此语句了，用户只好另想别的办法。我们不用INPUT和OUTPUT语句来与外部文件名进行通讯。只好借助于命令行。我们将外部文件名作为参数写在命令行中。运行模块提供一个指针数组，以便指示在命令行中出现的各参数，这些参数中也包括程序名本身。运行模块还有一个称为“Param”的自变量数组，它包括一个带特征域且无固定内容形式的记录和一个称为“Params”的命令行参数记录。对于命令行参数来说，我们所感兴趣的只是Param的第一个元素，即Param(1)。有关Param数组的更多的信息，我们将在“非标准特性”一章中的运行其他Pascal程序一节中进行讨论。Params

记录中的内容包括命令行的参数个数。此命令行包括程序名和指向指针数组的一个指针。例如：

对于： ++ test +L

称为“ arg-count ”的参数个数是2，第一个参数是程序名“ “ test”，第二个是“ +1” 。对于

++teet +lsc data

其中 arg-count 是3，分别是“te t”，“+lsc” 和 “data”。应该注意：对所有程序的 Param〔1〕。 Parms、args 来说，arg-count 至少为一。即程序名本身。指针 Pa am〔1〕、Parms、args 指向指针数组。此数组的第一个下标为0 。包括程序名在内的命令行中，其参数个数最多为101 。此数组的第一个参数 Param〔1〕、Parms、args^〔0〕 指向组成程序名字符串的数组，若用户打入下列命令行：

++qgsort data/test data +le

则 Param〔1〕。Parms、args^〔0〕^等于字符“ gsort”，总数可达64个字符。此外，Param〔1〕、Parms。args^〔1〕^等于“ data/test data”，Param〔1〕。Parms。args^〔2〕^等于“ +le ”。命令行中的每个参数都可按这种方法存放，各参数长度可达64个字符。例如，若要将文件名“ /data” 送到 Pascal 程序中，可用：

++ binaryfile /data

用户程序需通过 Param 结构来存放该命令行中的参数。

```
VAR   file-name:   array[1:64]   of   char ;
          file-ok:   boolean ;
BEGIN
```

```
...
IF    Param(1).Parms.args^(1)^=file-name
    THEN  file-OK  :=true;
...
END; ...
```

有关Param 结构详见Prefix 文件清单。此结构是标准Uniflex 命令行参数传送结构，其简要说明见Uniflex 手册的“ Introduction to UniFlex™ System Call”一章中关于exec 系统调用的说明。

“User Manual” 的第16－31页讨论了语句标号和Goto 语句的使用问题，在本Pascal 系统中，无标号，标号说明或Goto 语句。我们认为这些语句对结构化程序设计无益。在大多数情况下，诸如WHILE 循环语句能代替它们。若程序员想把标号及Goto 语句等用来作错误退出，我们向你提供“ abort” 语句，可代替它们。此语句能停止程序执行。详见“ Non-standard Features”，

“User Manual” 的第50页讨论了集合和集合运算。本Pascal 系统的一个集合最多可容128个元素，且是整数集而不是实数集，各元素的值必须在0～127 范围之内，不能小于0或大于127。此集合可方便地容纳ASCII 字符表。

标准Pascal 允许用户进行过程和函数说明的嵌套。本Pascal 却不允许过程或函数体的嵌套，尽管嵌套特性很好，但实际上并无必要，没有这些特性，Pascal 系统效率将更高。

不能将过程或函数作为参数传送给过程或函数。

为提高效率，字符串组必须包含偶数个元素。

目前Pascal的限制是：不允许把字符组的一个元素当作字符型变量参数传送。例如：

```
VAR  ch:  chair;
       carry:  array  [1,10]of
                    chair;
PROCEDURE test (VAR c : chair);
  ...
BEGIN
       test (carray [2]);
  ...
```

是错误的。但如果先把carray[2]赋给ch，然后，通过传送给tert。此外，如果test的形式参数c为数值型参数，也不会有问题。

本系统不能实现标准过程“dis pose(p)”。这里用2个过程“mark”和“releave”来控制运行栈。它们各自带一个整型自变量。运行栈和堆栈相似。当调用mark(i)过程时由mark(i)过程通知系统记下运行栈指针的当前值。当用户向NEW作了几次调用后，移动了栈顶指针，运行栈中存放了新的动态分配内容的变量。但若调用release(i)，则栈顶指针恢复到上次调用mark时记下的栈顶地址值。实际上这是释放了一个运行栈。Mark和release在下一章“Non-standard procedures and functions”中详细讨论。

标准过程PACK和UNPACK不能用，但允许PACKED数组和记录。

本系统略去了变型(variant)记录的空域表(见User

Manual 第46页）。若用了配对的括号“( )来表示空域，就出错。编译程序对变形记录的所有情况并不一一作检查。因而从说明语句中略去空的情况并不出错。还应指出，变形记录必须包括一个特征域。

标准过程 eof 和 eoln 都要求用显式文件名，即使文件是 inpvt 时也不何外。

非 标 准 特 性

A、非标准特性

在本 Pascal 系统中有几个特性不同于Jensen 和 Wirth 的标准规定。其中包括字符串常数的内部表示、参数传送、十六进制常数等等，下面详细叙述每一个非标准特性。

在一个Pascal 源文件中，开头的第一列用“%”符，后面紧跟文件名，再紧跟回车符。就可将此源文件包括到其它的 Pascal 源文件中去，以供编译时使用它。例如，有一个名为“declare”的文件，它由几个 Pascal 程序公用变量说明组成。则开始从第一列起写上“%declare ”，在编译时，就将此文件中的变量说明包括到另一个源文件中去。

```
% declare
BEGIN
  ...
END
```

在文件名后只许用回车符。不能出现其他任何东西。指定的文

件名，首先在当前目录中寻找，然后到“Current-diretory/plib”中，最后到“/plib”中寻找。所包括的文件可嵌套，即在一个Pascal程序中所包含的某个文件内还可包括别的文件。

注释可用三种符号对括起来：标准的波形括号‘§+’对，‘(**)’对，及双引号‘“”’对。注释一般不能嵌套，即“Comment “next“ ”为非法。但“Commen (*next*) ”是合法的。

Pascal编译程序对Tab字符按有效字符读入。但输出清单时，用空格代替tab。

在Pascal程序中，对于显式书写的字符串，都在其末尾自动加一个空字符。例如，显示字符串“abc”，由于系统自动给其加上一个字符，所以，它是一个4字符的串。因此，当字符组和显式字符串进行比较时，必须将最后一个空字符考虑进去。例如：

```
TYPE   file-name =array  (1..14)of char;
VAR    file-array:array(1..5)of file-
       names;
BEGIN
  ...
  IF file-array(2)='datafile/data'THEN...
```

数组file-array为14字符的字符组。显式串“datafile/data”是13个字符外加一个空字符，所以，显式字符串总长为14。若显式串不用13个字符来写，则编译程序给出错误信息。例如：

```
“test/test1      ”
  1234567890123
```

第一个显式串最后必须加上空格，以保证总长为13。如果一个显式串与一个10字符长的字符组比较，显式串必须有9个字符，何如：

```
VAR  list: array[1..10] of char;
BEGIN
  ...
    IF List='123456789' THEN ...
```

非标量数据值用作实在参数时，一般通过引用来传送。对说明为"值"参数的形式参数，一般不能修改。编译时会进行检查。

RESET 和 REWRITE 语句已加以扩展。按标准UhiFlex记法，"file-name" 一般为磁盘文件，本系统允许"file-name"可以为内部Pascal文件。REWRITE 语句不设置EOF，例如：

```
VAR f:file of integer;
BEGIN
  REWRITE (f,param[1],pavms,args
                '[3]');
                    or
  RESET (f,'indata');
  ...
```

所有文件应在总程序块中说明，不支持过程或函数内的局部文件。

采用16位字长，整数范围为-32768到+32767实数范围为1.0E38 到 1.0E+38 和 -1.0E-38 到 -1.0E+38，本 Pascal 系统中的实数具有16.8 个数字位的精度。所有从0