PASPR v3.2

PASCAL AND MODULA2 LISTING FORMATTER

USER DOCUMENTATION

David A. Vavra
P. O. Box 206
Laurel, MD 20707-0206

1.0 INTRODUCTION

PASPR is a program that permits the generation of formatted listings for PASCAL and MODULA2 source files. PASPR accomplishes this through the use of user supplied headers, ejection and separation information.

If you are like me, you find it easiest to work on a program that's formatted in a way that illustrates the intended control flow within the program and its routines.

To me, this means:

- titles indicating the major code sections;
- procedures that aren't split across page boundaries; and
- graphic separation of procedures which are combined on the same page.

I developed PASPR to give me these capabilities.

You may ask: why use PASPR instead of a "pretty printer" or compiler generated listing?

Well, in my opinion, the so called "pretty printers" go too far. For one thing, they are too rigid for my tastes. I tend to work on a program in "chunks". The definition of "chunk" depends upon the task at hand. For example, the code "WRITE (sometext); WRITEOCT (n); WRITELN;" might be such a "chunk". If I could, I would do the sample output in one statement but the language often forces me to do it in three. My personal preference is to see the three statements on one line but few, if any, "pretty printers" give me this as an option. besides, how could it know what I intended?

Another thing about "pretty printers" are the indentation rules. My personal indentation rules usually center around my desire to keep "chunks" together. I have yet to see a program that automatically indents the way that I do.

A third shortcoming in my mind is that "pretty printers" tend to spread the listing too far. As I said before, placing more than one statement on a line often loses potentially useful information besides running into the danger of making a procedure larger than a page, which in turn, would force the procedure across page breaks, would take too long to print and would waste paper.

Summing it up: unless the "pretty printer" is a mind reader, it is unlikely that it could ever format a listing "properly".

On the other hand, compiler generated listings don't go far enough. For one thing, some don't generate a listing at all. Others only support

page ejects which can be a real problem to place so that the listing is compact without breaking a procedure across a boundary. Another consideration is that most compilers generate the listing as a side effect of compilation. It has always struck me as silly to have to compile the program just to get a listing!

1.1 PASPR CAPABILITIES

To overcome all of this, I developed PASPR which allows:

- generation of a major title for the listing
- generation and placement of subtitles
- insertion of separation lines lines between major routines
- placement of absolute page ejects
- placement of conditional page ejects.

Over the period of time that I have been using PASPR, some bells and whistles have crept into the program; primarily because I use different editors and printers in the course of my work. Some of these are:

- expansion of input tabs
- specification of page parameters (e.g. # lines/page, margin sizes)
- specification of printer parameters (e.g., print setup strings, ejection mechanisms)
- configuration files.

1.2 FUTURE RELEASES

If I get a reasonable response from other users, future extensions might include:

- support of additional languages (Ada, assembler, etc.)
- recognition of source files that are unchanged since the previous print (really useful if a large group of files has been specified)
- wildcards in file names
- automatic generation of procedure separators (despite what I said before, I ALWAYS insert a separator before a level 0 subroutine)

- generation of tabs for transmission to the printer (instead of spaces which is the current method)

I am sure that there are many things that I haven't even considered. I would be happy to receive your comments and suggestions for incorporation into the program.

1.3 TERMS AND AGREEMENT

This product (PASPR), including associated documentation, has NOT been released to the public domain. It remains the property of the author. Both the product and its associated documentation may be freely distributed and used provided:

1) Such use is noncommercial;

2) The product, including the documentation, is not sold or distributed as a commercial product; and

2) No money is charged for distribution except for a nominal charge to cover the cost of materials and shipping.

Commercial use of this product is strictly prohibited except by written permission of the author. IF you have a commercial use in mind, by all means, WRITE TO ME! I can be very reasonable.

While the noncommercial use and distribution of this product is free of charge, I will (for a $20 donation):

1) register you as a user of the program;

2) provide you with the latest version of the program and documentation;

3) keep you abreast of later changes;

4) fix any problems encountered by you (assuming that I can resolve them); and

5) attempt to incorporate suggested changes in future releases.

If you encounter a problem while using the program I would like to hear about it. If possible, I would appreciate receiving a copy of a source file that demonstrates the problem. If the problem is resolvable, I will fix it and return a corrected version to you. If you send me a floppy (IBM PC compatible, 360K DSDD) I will return it to you.

All of this assumes that you have registered as a user. To register, send $20.00 (U.S.) to:

David A. Vavra
P.O. Box 206
Laurel, MD 20707-0206

Please keep in mind that I spent quite a few hours developing and debugging this program. If it saves you even a couple of hours of time, it is well worth the registration cost.

2.0 INPUTS TO PASPR

The inputs to PASPR are: source files (for printing), queue files, initialization files, and command lines. The first input is the command line which contains the names of the source, queue and initialization files. The formats of these inputs are discussed below.

2.1 COMMAND LINES

The command line is used to inform PASPR where to get the source, queue and initialization files to be used during processing. The sources for command lines are: the DOS command tail, queue files, and initialization files. Once this initial command line has been processed, the program terminates. If no command tail is present when invoking PASPR, the program will prompt you for one. If started in this mode, PASPR will continue to prompt for more command lines after the previous has completed until you abort the program with ^C.

The command line specifies:

- Files to be printed (source files)
- Queue files (files containing command lines)
- Parameters
- Initialization files (for retrieving predetermined parameters).

The queue and initialization files are described in following sections.

Each command line has the following form:

```
     <desc>
or   <desc><sep><desc><sep>.....<sep><desc>

     where:    <desc> ::=     <global parameter list>
                          |   <file path> <local parameter> <list>

               <sep>  ::=     <any number of spaces or commas>
```

PASPR processes the files and parameters on the command line sequentially from left to right.

The major distinction between global and local parameter specifications is that global parameters do not have file names associated with them while local parameters do. As you may expect, a local parameter only affects the associated file while a global parameter affects all following files.

Parameters have the following general form:

```
          /<parm select>
     or   /<parm select> <value spec>

Where:    <parm select> ::=   <parm name>
                          |   <parm name> <+|->
                          |   <+|-> <parm name>

          <value spec>  ::=   = <value>
                          |   : <value>
```

<parm name> is the name of the parameter (described later)

<+|-> is an on/off switch setting the meaning of which is determined by the parameter

<value> is the value to be assigned as determined by the parameter.

2.2 QUEUES

A queue file allows you to specify a group of files to be printed together. Such a group may be all of the source files associated with a particular program. A queue file permits you to "can" a series of PASPR actions to circumvent the need for future entry of

long command lines.

The queue file consists of a series of command lines followed by <CR><LF> sequence terminated by an end of file character <^Z>. It can be created by any text editor which observes this format, such as: EDLIN, WORDSTAR in nondocument mode and the TURBO PASCAL editor.

Queue files may be nested. The nesting level is determined by the number of files which may be open at one time and the amount of available memory in your machine. I do not know what the actual limit is, but I have gone as deep as 4. Nesting beyond this level probably has no practical value.

Global parameters specified in a queue file affect only the remainder of the queue file and lower queue levels. The global parameters are saved on entry to a queue level and are restored on exit.

Queue files are specified on the command line with a /Q LOCAL parameter. If a file has an extension name of ".QUE" it is assumed to be a queue file. If you want to print a file with a ".QUE" extension name, specify the file with a /Q- parameter to disable its processing as a queue file. (NOTE: do not use /Q- with initialization files which are described later).

If the extension name is omitted on the command line and /Q is specified, PASPR will assume an extension name of ".QUE". If the extension name is omitted and /Q (or /F, described later) is also omitted, and PASPR finds "<file>.QUE" it will assume that the /Q was present and will process that file as a queue file.

> For example, if the DOS command line is: "PASPR <file>" and the local directory contains a file "<file>.QUE", PASPR will assume that you want to process it as a queue file. If you just wanted to print file "<file>.PAS" you should use the /Q- parameter.

To prevent infinite processing, PASPR attempts to detect and avoid mutual recursion of queue files. The duplicate queue file is bypassed and no message is generated. This permits a queue file to specify embedded source files that have the same filename as the queue file.

2.3 INITIALIZATION FILES

PASPR permits a special type of queue file called an

initialization file. The initialization file is a queue file in all respects except that global parameters specified in the initialization file permanently override previous global parameter specifications. (In other words, the global parameters are not saved and restored when processing initialization files).

Initialization files are specified with the /F parameter. If no extension name is supplied then ".INI" is assumed. To print an initialization file with PASPR, use the /F- parameter. If you want to use an initialization file, you must use the /F parameter or specify the ".INI" extension name on the command line. PASPR will not automatically search out initialization files as it does with queue files.

Initialization files are useful for establishing common printer parameters. For example, a file called FX80.INI might bed built to specify margins and tabstops for an EPSON FX80 printer. Or a similar file might be built for an HP Laserjet printer called HPLJ.INI. To use them you would enter:

```
     PASPR FX80/F   src1, src2, src3
or   PASPR FX80.INI src1, src2, src3
```

when printing on the FX80 printer. Or you would enter:

```
     PASPR HPLJ/F   src1, src2, src3
or   PASPR HPLJ.INI src1, src2, src3
```

when using a Laserjet.

If you do this a lot, you may want to create a batch file with the name "FX80.BAT" which contains the command line "PASPR FX80/F $1 $2 ...".

2.4 SEARCH STRATEGIES

PASPR uses the following strategies when processing files:

1. If the extension name is ".INI" and the /F- parameter is not specified, the file will be treated as an initialization file.

2. If the /F parameter is specified, PASPR assumes a ".INI" extension name, if necessary, and treats the file as an initialization file.

3. If the extension name is ".QUE" and /Q- is not specified, the file is treated as a queue file

4. If the /Q parameter is specified, the file is treated as a queue file with a default extension name of ".QUE" if needed.

5. if no extension name is provided and a matching file with a ".QUE" extension is found, it is used as a queue file provided that /Q- was not specified and provided that the queue file is not at a higher nesting level.

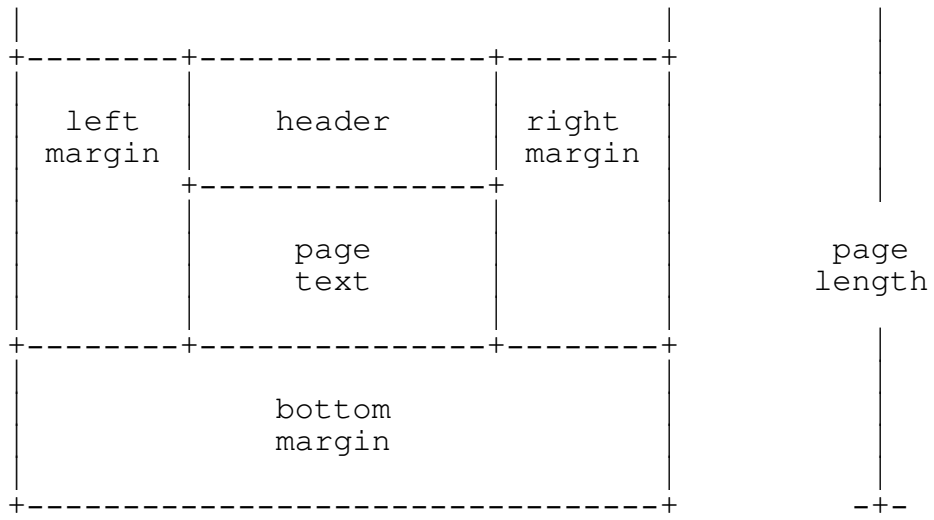
6. If an extension name is provided and the above strategies do not result in a queue or initialization file, the file is printed as a source file.

7. If the above strategies are unsuccessful, PASPR will print all source files that can be successfully found by appending the default source extension list (the /XN parameter) in order. The initial default for the /XN parameter is (.INC, .PAS, .SRC, .DEF, .MOD).

/XN is used to specify the default source extension names and their printing order. All successful matches will be printed. In other words, if X.INC and X.SRC are present, both will be printed (assuming .INC and .SRC are included in the /XN parameter). Up to 16 extension names may be specified in the /XN parameter.

3.0 PAGE SETUP PARAMETERS

The following parameters are used to establish the form of the printed page. The following illustrates the terms to be used:



The values given are the initial defaults.

/PL=66 sets the page length to the specified value. This value is the most common size for an 11 inch page. For some printers with automatic perforation skip, this value may have to be lessened by the amount of the automatic margin present on the page.

/TM=1 the number of blank lines + 1 to be printed before the first header line. These blank lines are in addition to any automatic margins inserted by the printer. Note that a value of one results in NO blank lines before the headers.

/BM=60 sets the beginning of the bottom margin relative to the top margin - 1. The value should be = (#top margin lines) + (3 header lines) + (#lines of text) - 1.

/LM=1 defines the size of the left margin by setting the values of the first text column on the page. This value is relative to the first printable column on the printer. (i.e., don't count margins inserted by the printer). LM-1 spaces are printed in the beginning of each line.

/RM=128 defines the last printable column on the page relative to the first printable column. The number of printable characters on a line = RM-LM+1.

The values of the LM and RM parameters determine the amount of text printed per line. Each text line from the source file is preceded by a 6 digit line number and ": ". The amount of printed

text space available per line is therefore RM-LM+1-8. Lines which exceed the right margin value are truncated.

4.0 PRINTER PARAMETERS

The following parameters are used to control the actions of the printer.

The action to be taken on the LAST LINE of the page varies from one printer model to the next. The end of page action parameter describes what to do with the last text line to get to the top of the next page. The LAST LINE is the one specified by the PL parameter. The EP parameter determines the end of line sequence that will result in positioning to the top of form if PASPR is printing on the last available print line.

/EP=NL The printer will automatically eject to the top of a page if the last line is followed by a <CR><LF> sequence. This is common with printers that automatically skip over page perforations. If used with a Laserjet printer, some lines may be lost.

/EP=NF a <CR><LF><FF> sequence is required to get to the top of the next page. This is the most probable setting to use if /PL doesn't really specify the last printable line on the page.

/EP=CF a <CR><FF> will get to the top of the next page without printing a blank page in between. This works for both EPSON and Laserjet printers. This is the default value for /EP.

/EP=FF only a form feed <FF> will get to the top of the next page.

The top of form action describes how to get to the top of the next page from a line which is not the last possible line on a page.

/TF=FF Get to the top of the next page by issuing a <FF>. This is the default action.

/TF=NL Get to the top of page by issuing newline (<CR><LF>) sequences. The number of newlines sent is determined by the PL parameter and the current page position.

Two things to keep in mind:

1. There is no way to detect the physical top of page. If PL, EP and TF are not set correctly, then blank pages or an insufficient number of title lines will be generated,

2. You must use a <FF> with the HP Laserjet printer. (At least you must with mine). Although the Laserjet has an automatic page eject, for some reason it doesn't always signal busy during the eject and some lines are lost. Using /TF=FF and /EP=CF solved this problem (for me).

PASPR also gives you the ability to transmit setup strings to the printer to establish automatic margins, tabstops, fonts, or whatever. To do this, use:

/IP="<text string>"

Everything enclosed in the quotes is sent to the printer verbatim except for:

""	which sends only a singe " character to the printer
^<char>	sends a control character to the printer. The character sent is the same one that is transmitted from the keyboard by simultaneously pressing the CONTROL (CTRL) key and the <char> key. Note that the character preceding <char> is the "up carat" found over the "6" on most keyboards.

5.0 TABS

The program always expands tab characters found in the input source file. The tab stops are defaulted to every eighth column starting in column 9 (the first column is column 1). This is the DOS default. The placement of these tabs can be changed with:

/FT=(<tabstops>)

where: <tabstops> ::= <tabstops>, <tabstop>
| <tabstop>

<tabstop> ::= <col>
| <start col>:<end col>
| <start col>:<end col>:<increment>

For example, the default setting is /FT=(9,17:255:8) or another way: /FT=(9:255)

The values given must be between 1 and 255. The default for <increment> is 8. <start col> must be less than <end col>.

If the above restrictions are not met, unpredictable results will occur.

Future releases will permit tabs to be generated for transmission to the printer.

6.0 PRESCAN MODE

During processing, PASPR can detect several errors. While a short error message is generated, it is sometimes difficult to determine where it has occurred without searching through the queue files. To resolve this problem, it is possible to execute in a "prescan" mode where PASPR will display the command lines prior to processing them along with the queue file names. The location of the error will become apparent. You may wish to use prescan to test queue files before using them.

During prescan mode, the source file names are displayed but the files are not printed.

Prescan mode is activated by /SCN. This parameter takes effect immediately but the command line containing /SCN will not be displayed (because /SCN was not in effect when the command line

was read). The /SCN parameter is absolutely global. That is, it is in effect until processing is finished or disabled by "/SCN-".

A typical usage of /SCN might be

```
PASPR /SCN MYQUE/Q
```

which would have the effect of displaying all of the command lines in MYQUE.QUE along with the names of the files which would have been printed and along with any detected errors.

7.0 SOURCE FILE SETUP

Source file setup is done with special embedded commands. The embedded commands are specially formatted PASCAL or MODULA2 comments. They have the form:

```
<comment beg> ! <cmd> <comment end>
```

where, <comment beg> ::= { | (*
<comment end> ::= } | *)

NOTE: there is no space between <comment beg>, the "!" and the command name.

The limitations of this method are:

1. The comment characters must be the proper ones for the language: MODULA2 does not permit "{" or "}" to be used for comments. Actually, PASPR could care less but you probably want to compile the source sometime!

2. PASPR does not properly identify nested comments. Depending upon the sequence and location of the comment marks, PASPR may become confused.

3. PASPR does not distinguish between: "{...}" , "(*...*)", "{...*)" and "(*...}". They are all valid comments.

As far as I know, limitations 2 and 3 occur only with TURBO PASCAL. Maybe I'll remove them if enough people demand it. As it stands they never affect me.

The valid commands which can be used are:

```
{!PT <title>}       Major title
{!PS <subtitle>} or
{!P  <subtitle>}    Minor title
{!P} or <FF> char   Eject
{!P?}               conditional eject
```

{!!} separator

These commands are discussed in more detail in the following sections.

NOTE: the conditional eject and separator commands actually permit the inclusion of a subtitle value, however, since I never really determined what such a title could mean, the inclusion of any subtitle on these commands may have strange results. I am only telling you this because it's too much trouble to check for them. You have been warned.

7.1 MAJOR TITLE

{!PT <title>}

Place this as the first line of your source program. All characters from the first nonblank character after PT to the "}" or first intervening <CR> become the top title of every following page.

The length of the title is restricted to 128 characters but since the title actually starts printing in column 40 use discretion.

This command may be included more than once in a source file. If it is, it is treated as if it were preceded by a page eject. If two or more consecutive titles are present in the source file, only the last one has an effect.

Any blank lines in the source file preceding or following the title are omitted from the listing. The line containing the title will also be omitted from the listing unless it contains nonblank characters outside of the comment containing the title command. If this is the case, and the line IS printed, the title command will not appear in the listing. Source preceding the title command will be printed on the preceding page and source following the title will be printed on the following page. I tried to keep the line numbers the same but since I've never actually tested this, who knows?

7.2 MINOR TITLES

{!PS <title>}
{!P <title>}
{!P} or <FF> *** Forced page eject

The above commands perform essentially the same function as the Major title command but the minor title is printed below the major title on the second line of the header. The minor title is subject to the same restrictions and capabilities as the major title.

I use the minor title to separate groups of procedures that have similar function or connected functions.

Note that it is not possible to clear a minor (or major) title once it has been set. Any minor title commands with blank titles are considered to be forced page ejects.

7.3 SEPARATOR

```
{!!}
```

I use the separator command to separate subroutines within a listing. What it does is place a logical mark in the source file. The meaning of this mark is as follows:

1. If the following source text up until the next command can fit on the current page, PASPR will print the separator as a blank line followed by a line of hyphens followed by another blank line.

2. If the following source text can not fit on the current page, the separator is treated as a page eject.

The "{!!}" itself will never appear in the source listing. If it is embedded in a nonblank source line the source line will be spilt by the command. I always place the command on a line by itself. What I've just described is what I have tried to get the program to do but I have never tested this. Use at your own risk.

7.4 CONDITIONAL EJECT

```
{!P?}
```

This has an effect that is similar to {!!} but a single blank line

is printed instead of the three separator lines. I use this to separate record statements to force them to the top of the next page if they would otherwise be broken across page boundaries. It's also handy for separating internal procedures.

8.0 QUICK REFERENCE

8.1 COMMAND Line Prarmeters

The following are the allowable parameters for a command line. The default settings are shown.

```
/SCN      Scan mode

/F        iniitalization file (default extent = ".INI")

/Q        queue file (default extent = ".QUE")

/XN = (.INC,.PAS,.SRC,.DEF,.MOD)
          default source file extension name list

/PL=66    Page length
/TM=1     Top margin (first header line number)
/BM=60    Bottom margin (= TM + 3 hdr lines + # text lines -1)
/LM=1     Left margin   (= # spaces to generate + 1)
/RM=128   Right margin  (= #chars/line + LM - 1 = last print column)
```

NOTE: PASPR assumes that the number of printable chars/line = RM-LM+1. Lines longer than this are truncated.

/FT=(9:255:8)
Source file tab stops. "beg:end:inc". First input col = 1 input to /FT is a list enclosed in parentheses:

(ts1,ts2,ts3,...,tsn)

where: ts = <beg> | <beg>:<end> | <beg>:<end>:<inc>

defaults: <end> = <beg>, <inc> = 8

/EP=<end of page action>
Eject action from last page line

/EP=NL <CR><LF>
/EP=NF <CR><LF><FF>
/EP=CF <CR><FF> (default)
/EP=FF <FF>

/TF=<top of form action>
Eject action from other than last page line

/TF=FF <FF> (default)
/TF=NL <CR><LF> until top

/IP="<string>"
Printer setup string: enter <"> as <""> and control characters as <^><char>. E.g., ^C= control-C

8.1 Commands Embedded in Source Files

General format: {!<cmd>}
| {!<cmd> <parm>}
| (*!<cmd>*)
| (*!<cmd> <parm>*)

{!PT <title>} Major title

{!PS <title>} Subtitle
{!P <title>}

{!P} Page eject
<FF>

{!P?}	Conditional page eject. Prints blank line or page eject.
{!!}	Procedure separator. Prints <blank line> <----> <blank line> or page eject