NAME

pscover − create cassette covers in postscript

SYNOPSIS

pscover [−{cd}] [−t] [−o *outfile*] data_files
pscover −i

DESCRIPTION

This file describes the pscover program, which defines a simple language to describe and generate covers for audio cassette tapes or DAT tapes. The **−c** option is the default, which will produce an audio cassette cover. The **−d** option may be specified to produce a DAT cover. The input file is parsed and postscript language is output which will produce the desired cover. If the **−t** option is supplied, *pscover* will perform a syntax check and output information on what was found. This option can be used to "debug" a cover input file.

The goal of this program was to provide a simple method for non-programmers to create cassette covers. The language is designed to be simple to remember, using reasonable defaults, as well as flexible enough to allow a more sophisticated user to take more control over the output.

Cassette covers consist of a shaded front tab which contains the date of the recording, a title and artist along the top edge, an optional "tab" containing additional explanatory information, and then songs along the back of the cover indicating what is on the two sides of the tape.

To run pscover, you must first prepare a file containing a description of the label you want to produce. Once this is completed, you run pscover and redirect the output to either a file to be printed later, or to the printer.

For example:

```
Unix:    % pscover cover.dat > cover.ps
         % pscover cover.dat | lpr

MSDOS:
         C:> pscover cover.dat > cover.ps
         C:> pscover cover.dat > prn:
```

A simple input file might be:

```
% this is either a short tape or long songs
FullArtist      The Best Band
FullTitle       Greatest Hits
LeftSide
                First Song
                Second Song
RightSide
                Third Song
                Fourth Song
```

PSCOVER BASIC LANGUAGE

Pscover supports a language designed specifically for cassette covers. Generally, a command consists of a a justification indicator (e.g., left, right, center, or full) and category keyword (e.g., title, artist). Keywords may be in upper or lower case (or even a mixture).

Although most commands have a left, right, and full justification attribute, you may mix these commands in whatever way you desire. Usually, you will either have a left (and possibly right command), or a full command. It is possible to use any combination. If all three are used, it is the users responsibility to make sure the fields don't overlap.

Some commands take no arguments, others take one or two. Commands must start in the first column. A line beginning with white space (tab or space) indicates a song to be printed. Arguments are separated by white space as well. The recommended format is:

```
command
```

command <tab> argument
command <tab> argument1 < tab > argument2
<tab> song_title

In the list below, all variations of a command are listed, followed by the argument(s) that are required.

% Comments
Any line starting with a percent sign (%) is considered a comment line, and is copied to the output file. (The percent sign indicates a comment in PostScript).

LeftCover, CenterCover, RightCover
You may place two covers on one page: a left cover and a right cover. For a DAT tape, you may also have a center cover. The first cover on each page is assumed to be a left cover, so LeftCover is seldom needed. This command is only required if you want to place a second cover on the right side (center for DAT) of the page.

LeftArtist, RightArtist, FullArtist *artist_name*
This command places the artist name(s) on the top half of the cover's edge. (If you want the artist to appear on the bottom, reverse the use of artist & title). If you use more than one justification command (e.g., Left & Right), you must be sure that the two names don't overlap.

LeftTitle, RightTitle, FullTitle *title_name*
This command places the title(s) on the bottom half of the cover's edge. (If you want the title to appear on the top, then reverse the use of artist & title). If you use more than one justification command (e.g., Left & Right), you must be sure that the two names don't overlap.

Date *date_string*
This command places the date of the recording on the front flap.

Noise *noise_suppression*
This command places the noise suppression on the front flap.

LeftTab, RightTab, FullTab *tab_information*
This optional command will produce additional information on the top of the back side of the cassette. If LeftTab or FullTab is used, a dotted line is used to create the tab area. If this option is not used, then the songs will start closer to the top of the back cover.

LeftSide, RightSide, FullSide
This command causes subsequent songs to be placed on the left or right side of the cover, or centered across the entire cover. Generally, Left and Right is used to indicate what is on each side of the cassette, and Full is used if both sides are the same. The *leftside* command will also draw a dotted vertical line down the middle of the cover where the songs would appear.

LeftSongs, RightSongs, CenterSongs
This command causes subsequent songs to be printed left or right justified, or centered on the line. If used with LeftSide or RightSide, the songs will be centered for the respective side. If a song is longer than one line, it will be left justified with continuation lines indented slightly.

NewLine
This command causes a blank line to appear. This command is generally used to insert spacing between songs.

OldLine
This command backs up to the previous line. This command is generally used to print more than one thing on a line.

LeftSong, RightSong, CenterSong *song_title*
This command prints a song title. This command can be used to print a song with a justification different than the default. (A line that begins with a tab will cause a song title to be printed using

the default justification.) After the song title is printed, a newline is inserted.

LeftLine, RightSide, CenterSide *string*

This command works similar to the *Song* command, except that a newline is not inserted. This command is useful for printing several strings on the same line.

DoubleSong *left_string* <tab> *right_string*

This command prints a left and right string on the same line, followed by a newline. This command is useful for printing song titles and times (or artists) on the same line. The tab character is required to separate the two strings. Each string can contains spaces, but not tabs.

DoubleLine *left_string* <tab> *right_string*

This command is similar to the *DoubleSong* command except that a newline is not inserted.

LeftMarg, RightMarg, CenterMarg

This command moves the current point to the left or right margin or center point of the current line.

BegCode ... EndCode

This pair of commands allows the user to insert actual PostScript code directly into the output file. No checking is done. All lines following the BegCode line up to, but not including, the EndCode line, is copied without modification. Extreme caution should be taken when using these commands.

WrapUp

This command is only required if more than one page of covers are being printed. The command causes the current page of covers to be printed and a new page set up. By default, the cover after a *wrapup* command will be the left cover.

ADVANCED COMMANDS

The following commands are intended for the more advanced users. They are not required, and the casual user is not expected to use them.

NewFont *font_name font_size*

This command changes the current font for subsequent songs. The *font_name* must be either a valid PostScript™ font preceeded by a slash (e.g., /Helvetica-Bold), or a font name variable that was previously defined (e.g., SongFont). The *font_size* must evaluate to points; it may be an expression or a previously defined variable. For a list of variables, see the *set* command).

Set *variable_name value*

This command may be used to define or modify PostScript variables. The *variable_name* may be a previously defined variable or a new one created by the user. The *value* must be a valid PostScript expression. For font names, this would either be a literal string (the font name preceeded by a slash), or the name of a previously defined variable. The command should be used with caution. When creating new variables, make sure the variable name is not already used in the prologue.

PRE-DEFINED VARIABLES

The following variables have been defined. Variable names are case sensitive; you must specify a name exactly as it appears below. A complete list can be obtained by examining prologue of the actual PostScript code which is generated.

Variable	**Default**
TitleFont"	/Helvetica
TitleSize	12
ArtistFont	/Helvetica-Bold
ArtistSize	12
TabFont	/Helvetica-Narrow
TabSize	10

SongFont	/Palatino-Bold
SongSize	8
DateFont	/Times-Bold
DateSize	10
NoiseFont	/Times-Bold
NoiseSize	10
TextUp	5
INDENT	10

Font variables are defaulted to literal strings (literal strings begin with a slash in PostScript). When specifying a font name, you must use either the name of a font variable or the actual font name prefixed with a slash. Font sizes are in points (1/72 of an inch). The above font names and sizes should be obvious.

In addition to font variables, some additional variables are available. The variable **TextUp** indicates how many points above a baseline text should appear. For example, the tab information will be located *TextUp* points above the bottom of the tab area, and the title will appear *TextUp* points above the bottom of the top edge. The variable **INDENT** indicates how many points to indent continuation song lines.

EXPRESSIONS

PostScript is a stack-based post-fix language. This means that items are placed on a stack until some operation removes them and performs some action on them. The *Set* command defines the user-supplied name with the user-specified value. The value expression must be a valid PostScript expression.

Literals are prefixed by a slash (e.g., **/Helvetica-Bold**), strings are enclosed in parenthesis (e.g., **(Sample String)**), otherwise, the expression will be treated as an arithmetic expression (e.g., **{ RM LM sub 2 div }**). When supplying arithmetic expressions for the set command, the add, sub, mul, and div operations may be used (*var1 var2 sub* subtracts *var2* from *var1*.)

For example, the following code will divide the line into three equal segments, and then print three strings centered in each of these segments. (We will assume that the strings will fit.)

```
% save current left & right margin
Set     LeftSaveLM
Set     RightSaveRM
% compute size of each segment
Set     Segment{ RM LM sub 3 div }
% get right margin for left segment
Set     RM{ LM Segment add }
% center the string in the left segment
CenterLineleft-string
% get left & right margins for middle segment
Set     LMRM
Set     RM{ LM Segment add }
% center string in the middle segment
CenterLinemiddle-string
% get left & right margins for right segment
Set     LMRM
Set     RMRightSave
% center string in the right segment
CenterLineright-string
% restore margins
Set     LMLeftSave
Set     RMRightSave
```

The following example changes the default font for Title and Artist:

```
Set     TitleFont/Palatino-Bold
Set     TitleSize14
```

Set ArtistFontTitleFont
Set ArtistSize10

CAVEATS

The *pscover* command does not check for valid font names. If you are unsure what fonts your printer supports, or if you want a sample of these fonts, consult the manual.