

HelpGen Help

Purpose - Click on icon above

How to

- [Load or Create a Macro file](#)
- [Load or Create a Project File](#)
- [Build an RTF File](#)
- [Build a Help File](#)
- [Use the HelpGen Macro Language](#)

Menu Commands

- [File Commands](#)
- [Edit Commands](#)
- [Build Commands](#)
- [Test Command](#)
- [Option Commands](#)
- [Help Commands](#)



Glossary

If you are unfamiliar with how Windows Help works, choose **Help | How To Use Help** from the Help menu now.

HelpGen Information

HelpGen version 1.21. © 1994 by Rimrock Software, all rights reserved.

Under normal circumstances, if you want to create a help file with the Windows help compiler, you must first create a help topic file using a word processor that can save files in Rich Text Format (RTF). There are several basic problems with that approach:

1. Word processors cost a lot of money.
2. Once you've purchased the word processor, you have to learn how to use it. This may not be a very productive use of your programming time.
3. You also have to learn how to use the word processor to produce a file that is properly formatted for the help compiler. This involves using items like underlining, double-underlining, hidden text, footnotes, etc.

HelpGen eliminates these problems by allowing you to use your favorite ASCII text editor to produce a HelpGen macro file, which HelpGen uses to create the RTF formatted file required by the help compiler.

HelpGen automates the entire process of creating help files:

- Creates a skeleton macro file for you to edit.
- Invokes your editor to allow you to edit the macro file.
- Creates a help project file (.HPJ) to tell the help compiler how to compile the help file.
- Generates an RTF file from the macro file.
- Invokes the help compiler to compile the RTF file into a HLP file.
- Allows you to directly test the result.

The HelpGen macro engine is based on code written by David Spector, and published in Windows/DOS Developer's Journal, Vol. 5, No. 7, July 1994.

How to Load/Create Macro File

Select the **File | Open Macro File** menu option. Select an existing .MAC file or enter the name of a new file. If you enter the name of a file that does not exist, HelpGen will create the file for you, using a template that contains the minimum items required to build a help file.

If you have created a new macro file, you should use the **Edit | Macro File** option to make two changes to the file. The first change is a new title for the table of contents. The default title is xxxx. The second change is the selection of a .BMP graphic to place on the table of contents title line. The default is HlpGen.BMP.

How to Load/Create Project File

HelpGen looks for an associated project file when you open the current macro file. If it finds one, HelpGen will enable the **Edit | Project File** menu item, and you won't have to build a project file. If it doesn't find a project file, HelpGen will enable the **Build | Project File** menu item to allow you to create a project file. You must select **Build | Project File** to create the new project file.

NOTE: Before you build a project file, select the **Options | Project File** menu item and make sure that all the items in that dialog are to your liking. Those items will be used to create the project file.

How to Build an RTF File

Select the **Build | RTF File** menu item. HelpGen will ask you if you are sure you want to create an RTF file. If you answer 'yes', HelpGen will create a .RTF file by processing the .MAC file. Macros will be expanded and RTF commands and plain text will be passed through to the .RTF file. When processing is complete, HelpGen will tell you and will then enable the **Build | HLP File** menu item.

How to Build a Help File

Select the **Build | HLP File** menu item. HelpGen will ask you if you are sure you want to create a help file. If you answer 'yes', HelpGen will invoke the help compiler you have selected in the **Options | Directories** menu item. This will involve shelling out to DOS, since help compilers are DOS programs. When HelpGen gets control again, it will enable the **Test** menu item, to allow you to test the help file you have just created.

If there were errors generated during the help compilation, they will be listed in the error log file that is shown in **Options | Project File**.

<p>NOTE: When HelpGen invokes the help compiler, and for some reason the compiler can't execute, HelpGen will notify you by displaying an error message. To possibly correct your problem, make sure that the proper help compiler is correctly chosen in HelpGen's Options Directories menu item.</p>
--

Using the HelpGen Macro Language

- [Macro Language Basics](#)
- [Macro Language Commands](#)
- [Useful RTF Commands](#)

The File Commands

- Open Macro File
- Close Macro File
- Print Macro File
- Exit

The Edit Commands

- Edit Macro File
- Edit Project File

The Build Commands

- Project File
- RTF File
- HLP File

The Test Command

The **Test** menu item allows you to test the help file that you have just generated. It will only be enabled after you have generated a help file.

You may also select this menu item with the toolbar Test Help File button.

The Options Commands

- Directories
- Project File

The Help Commands

- [Contents](#)
- [About HelpGen](#)

Open Macro File

Open an existing macro file, or create a new macro file to be edited (**File | Open Macro File** menu item). A check will be made for a corresponding project file. If a project file exists, then the **Edit | Project File** menu item will be enabled. If a project file does not exist, then the **Build | Project File** menu item will be enabled. The HelpGen title bar is updated to reflect the current macro file name.

You may also select this menu item with the toolbar Open Macro File button.

Close Macro File

The Close command closes all files and disables all HelpGen menu items that are associated with open files (**File | Close Macro File** menu item). The HelpGen title bar is updated to show that no macro file is currently open.

You may also select this menu item with the toolbar Close Macro File button.

Print Macro File

The Print command will print the current macro file on any printer (**File | Print Macro File** menu item). You will be asked to select the proper printer before any printing takes place.

You may also select this menu item with the toolbar Print Macro File button.

Exit to Program Manager

The Exit command closes all open files and shuts down the HelpGen program (**File | Exit** menu item).

Edit Macro File

The **Edit | Macro File** command executes a text editor and allows you to make changes to the current macro file. The default text editor is NOTEPAD.EXE. The default editor may be changed using the **Options | Directories** menu item.

You may also select this menu item with the toolbar Edit Macro File button.

<p>NOTE: When HelpGen invokes the text editor, and for some reason the editor can't be executed, HelpGen will notify you by displaying an error message. To possibly correct your problem, make sure that the text editor is correctly chosen in HelpGen's Options Directories menu item.</p>
--

Edit Project File

The **Edit | Project File** menu item executes a text editor and allows you to make changes to the current project file. The default text editor is NOTEPAD.EXE. The default editor may be changed using the **Options | Directories** menu item.

You may also select this menu item with the toolbar Edit Project File button.

NOTE: When HelpGen invokes the text editor, and for some reason the editor can't execute, HelpGen will notify you by displaying an error message. To possibly correct your problem, make sure that the text editor is correctly chosen in HelpGen's Options | Directories menu item.

Build Project File

If you don't yet have a project file for your current macro file, this menu item will be enabled (**Build | Project File** menu item). It will create a project file for you, based on the entries contained in the **Options** menu. You may then make any changes to this file using the **Edit | Project File** menu item. The project file will have the same filename as the macro file, with an extension of .HPJ.

You may also select this menu item with the toolbar Build Project File button.

Build RTF File

The **Build | RTF File** menu item uses the current macro file to build a Rich Text Format (RTF) file of the same name. The RTF file is used as an input file for the help compiler.

You may also select this menu item with the toolbar Build RTF File button.

Build HLP File

The **Build | HLP File** menu item uses the current project file and the RTF file created by HelpGen to create a Microsoft Help File (HLP). It does this by invoking the Microsoft (or other) help compiler. The default help compiler is HC31.EXE. You may change the default help compiler by using the **Options | Directories** menu item.

You may also select this menu item with the toolbar Build HLP File button.

<p>NOTE: When HelpGen invokes the help compiler, and for some reason the compiler can't execute, HelpGen will notify you by displaying an error message. To possibly correct your problem, make sure that the proper help compiler is correctly chosen in HelpGen's Options Directories menu item.</p>

Set Directories

The **Options | Directories** menu item allows you to select the help compiler and text editor that HelpGen uses when building a help file or editing text. The entries that you make are also saved in HelpGen's .INI file and will be used each time you execute HelpGen.

Set Project File Options

The **Options | Project File** menu item allows you to specify the options that will be installed in a project file that is created by HelpGen. Some of these options are saved in HelpGen's .INI file, so they may be used in subsequent executions of HelpGen. Specifically, the Compression type, Warnings, Report status, Copyright notice and Error Log are all saved in the .INI file.

Title

The Title specifies the string that is placed on the Help File's title bar when the help file is executed.

Copyright

The Copyright specifies the string that is placed inside of the help file's About box. This box is activated by executing the help file and selecting **Help | About Help**.

Error Log

The Error Log is the file that the help compiler uses to write error messages to. If you have errors during a help compilation, look at this file to see what the errors were.

Icon File

The Icon File entry specifies the icon that will be used when the help file is minimized. This entry will override the default question mark icon.

Bitmap Root

The Bitmap Root specifies the directory that is used to store bitmap files, if they are not in the current directory.

Compression

The Compression option specifies what type of compression is used when building the help file. **0** specifies no compression, **Medium** specifies 40% compression and **High** specifies 50% compression.

Warnings

The Warnings option sets the level of error messages that the help compiler will display during compilation. **Some** means only the most severe messages will be displayed, **Many** means a medium amount of error messages will be displayed, and **.(All)** means display all messages.

Report

The Report option determines the type of error messages that are displayed during help file compilation.

Help Contents

The **Help | Contents** menu item executes the help file you are currently reading.

You may also select this menu item with the toolbar Help Contents button.

About HelpGen

The **Help | About HelpGen** menu item provides information about the current version of the HelpGen program.

Language Basics

HelpGen Macro Files consist of distinct blocks of text. These blocks are marked by HelpGen macros. The first block is the .start/.end/.end_file block:

```
.start(.....)
Table of Contents stuff here....
.end

Rest of macro file goes here....

.end_file
```

The rest of the macro file is broken up into .ent/.end and .pent/.end blocks:

```
.start(.....)
Table of Contents stuff here....
.end

.ent(.....)
Topic stuff here....
.end

.ent(.....)
Topic stuff here....
.end

.ent(.....)
Topic stuff here....
.end

.pent(.....)
Popup stuff here....
.end

.pent(.....)
Popup stuff here....
.end

.end_file
```

Note the use of white space to separate the various blocks. You may also separate the blocks with remarks:

```
.rem(===== Beginning of topic 1 =====)
.ent(.....)
Topic stuff here....
.end
.rem(===== Beginning of topic 2 =====)
.ent(.....)
Topic stuff here....
.end
```

In order to display each of these blocks, you need to have jumps (.j or .j1 for the .ent/.end blocks and .p or .p1 for the .pent/.end blocks) somewhere in your macro file. Since the help file begins at the Table of Contents, there should definitely be some jumps in that topic.

All the rest of the HelpGen macros are basically window dressing for the topic blocks. See the various macro commands for examples of how to use them.

HelpGen generates an RTF file that specifies four basic text fonts; font 0 is MS San Serif, font 1 is Roman, font 2 is Courier and font 3 is Symbol. Font 0 is used for topic headers, font 1 is used for normal text in a topic and font 2 is used inside of the .[and .] macros. Font 3 is not currently used.

Macro Language Commands

- .start
- .ent
- .pent
- .end
- .j
- .j1
- .p
- .p1
- .bj
- .top
- .bmp
- .bmpl
- .bmpr
- .box
- .bend
- .rem
- .in
- .un
- ##
- #b
- #n
- .b
- .i
- .l
- .l
- .n
- .s
- .end_file

Some Useful RTF Commands

- \brdrdb
- \brdrs
- \brdrsh
- \brdrth
- \li
- \ri
- \ql
- \qr
- \qj
- \qc
- \tab
- \'hh

NOTE: All RTF commands should be separated from regular text by some kind of 'white space'. This includes a space (' ') or a carriage return.

The .start Macro

SYNTAX: .start(label,topic,bitmap_file,application_name)

EXPLANATION: This macro marks the beginning of a macro file. It produces an attractive header for the Table of Contents topic. **label** and **topic** are as described for the .ent macro. **bitmap_file** is the filename of a .BMP file generated by a bitmap editor such as Windows Paintbrush. **application_name** is the title that will appear in the header of the Table of Contents topic. You must provide a topic entry labeled 'icon' which will be jumped to when the user clicks on the icon in the Table of Contents header. The .start statement also requires a corresponding .end to indicate the end of the table of contents.

EXAMPLE:

```
.start(main,Contents,DEMO.BMP,A Help Demonstration)
This is the table of contents topic. It is ended by the next line.
.end

.ent(icon,Version,Help File )
This is a mandatory topic. We jump to here when the DEMO.BMP icon is
clicked.
.end

.end_file
```

SEE ALSO: [.end_file](#)

The .ent Macro

SYNTAX: .ent(label,topic,prefix)

EXPLANATION: This macro marks the start of a topic entry. **label** is used to refer to the topic in the .j and .j1 macros, **topic** is the name that will appear in the topic header and in the Search dialog box; and **prefix** is a string that will appear as a prefix in the header (if the topic is "Frogs" and the prefix is "The", the header will be "The Frogs" but the entry in the Search dialog box will be "Frogs"). The prefix can be an empty string.

EXAMPLE:

```
.ent(file,File Commands, )  
.in  
##.j(open,Open File).n  
.un  
.end  
  
.ent(open,Open File, )  
This command allows you to open an existing file.  
.end
```

SEE ALSO: .end, .j, .j1

The .pent Macro

SYNTAX: .pent(label,topic)

EXPLANATION: This macro is identical to .ent, except that it creates popup topic boxes instead of topic pages. This type of popup box is usually used for definitions. **label** is used to refer to the topic in the .p and .p1 macros described below. **topic** should match the topic label in the corresponding .p or .p1 macro.

EXAMPLE:

```
.ent(size,Font Size,Changing )
This item allows you to change the size of the display font.  The new
size can be anywhere from 6 .p1(points) to 72 points.
.end

.pent(points,Points)
A sizing standard for text.  There are 72 points to the inch.
Therefore, 6 points is  $6/72" = 1/12"$ .
.end
```

SEE ALSO: .end, .p, .p1

The .end Macro

SYNTAX: .end

EXPLANATION: This macro marks the end of a topic entry that was started with .ent or .pent.

EXAMPLE: See the examples for .start, .ent and .pent.

SEE ALSO: .start, .ent, .pent

The .j Macro

SYNTAX: .j(label,string)

EXPLANATION: This macro creates an underlined **string** and uses the string as a jump area. If the user clicks on a jump area, the topic represented by **label** is shown on the screen.

EXAMPLE:

```
.ent(layout,Page Layout,The )
How to Lay Out
.s
.j (pageno,Page Numbers) .n
.j1(Headers) .n
.j1(Headers) .n
.j1(Footers) .n
.end

.ent(pageno,Page Numbers, Laying Out )
Text for page number layout goes here.
.end

.ent(Headers,Headers,Laying Out )
Text for page header layout goes here.
.end

.ent(Footers,Headers,Laying Out )
Text for page footer layout goes here.
.end
```

SEE ALSO: .ent, .j1, .bj

The .j1 Macro

SYNTAX: .j1(label)

EXPLANATION: This macro is identical to a .j with its **label** and **string** being identical. This macro is used for brevity.

EXAMPLE: See the example for the .j command.

SEE ALSO: .ent, .j

The .p Macro

SYNTAX: .p(label,string)

EXPLANATION: This macro underlines **string** with a dotted line and uses it as a jump area. If the user clicks on the jump area, the topic represented by **label** is shown in a popup window. This is used to show definitions.

EXAMPLE:

```
.ent(cmds,Keyboard Commands,New )
The keyboard that supports Chicago has three extra keys; the .p1(LWIN)
key, the .p1(RWIN) key and the .p(application,APP) key.
.end

.pent(application,APP)
When pressed, brings up the context menu at the current select
position.
.end

.pent(LWIN,LWIN)
Sets the focus to the Chicago User Interface. Same functionality as
the RWIN key, but uses a different scan code.
.end

.pent(RWIN,RWIN)
Sets the focus to the Chicago User Interface. Same functionality as
the LWIN key, but uses a different scan code.
.end
```

SEE ALSO: .pent, .p1

The .p1 Macro

SYNTAX: .p1(label)

EXPLANATION: This macro is identical to .p with its **label** and **string** being identical. This macro can be used for brevity.

EXAMPLE: See the example for the .p command.

SEE ALSO: .pent, .p

The .bj Macro

SYNTAX: .bj(label,bitmap_name)

EXPLANATION: This macro creates a graphical jump area using **bitmap_name** as the graphic. If the user clicks on the jump area, the topic represented by **label** is shown on the screen.

EXAMPLE:

```
.ent(filecmds,File Commands, )
.bj(new,new.bmp) New File
.bj(open,open.bmp) Open File
.bj(close,close.bmp) Close File
.end

.ent(new,New File, )
Text for new file command goes here.
.end

.ent(open,Open File, )
Text for open file command goes here.
.end

.ent(close,Close File, )
Text for close file command goes here.
.end
```

SEE ALSO: .ent, .j

The .top Macro

SYNTAX: .top(string)

EXPLANATION: This macro is used inside of .ent and .pent entries to add **string** as another associated topic name to the Search dialog box.

EXAMPLE:

```
.pent(mouse,Pointing Devices)
.top(Trackball)
.top(Graphics Tablet)
.top(Mouse)
.top(Pointing Stick)
.top(Joy Stick)
A device that moves the graphical cursor around the screen and that
allows you to select objects.
.end
```

SEE ALSO: .ent, .pent

The .bmp Macro

SYNTAX: .bmp(pathname)

EXPLANATION: This macro inserts a named .BMP bitmap file on the current line. The bitmap is positioned as though it were just another character.

EXAMPLE:

```
.ent(open,Open Macro File, )  
Use the .bmp(openrtf.bmp) Open Macro File button to open the file.  
.end
```

SEE ALSO: .bmpl, .bmpr

The .bmpl Macro

SYNTAX: .bmpl(pathname)

EXPLANATION: This macro inserts a named .BMP bitmap file at the far left side of the current line. Any text following the .bmpl entry is wrapped around the bitmap.

EXAMPLE:

```
.ent(close,Close Macro File, )  
.bmpl(closetf.bmp)Close the macro file and all associated files.  
Disable any menu items relating to open files.  
.end
```

SEE ALSO: .bmp, .bmpr

The .bmpr Macro

SYNTAX: .bmpr(pathname)

EXPLANATION: This macro inserts a named .BMP bitmap file at the far right side of the current line. Any text following the .bmpr entry is wrapped around the bitmap.

EXAMPLE:

```
.ent(keydisp,Key Information,Display )  
.bmpr(keydisp.bmp)The key display area shows the scan codes that will  
be sent for this particular key - unshifted, shifted, control and  
alternate.  
.end
```

SEE ALSO: .bmp, .bmpl

The .box Macro

SYNTAX: .box

EXPLANATION: This macro draws a box around the current paragraph. It applies to all subsequent paragraphs up to the .bend command. Box drawing should only be done in normal text areas in .ent or .pent areas.

EXAMPLE:

```
.ent(about,About HelpGen, )
Display a box containing version information and copyright
information.
.n
.box
.b(NOTE:) The integer portion of the version number indicates a major
revision and the decimal portion indicates a minor revision.
.bend
.s
All dialog boxes use icon-style pushbuttons and graphical elements.
.end
```

SEE ALSO: [.bend](#)

The .bend Macro

SYNTAX: .bend

EXPLANATION: This macro completes the box drawing that was started with a .box command.

EXAMPLE: See the .box example.

SEE ALSO: .box

The .rem Macro

SYNTAX: .rem(text)

EXPLANATION: This macro allows comments to be inserted into the macro file.

EXAMPLE:

```
.rem(-----)
.rem( Macro file for RENAULT.EXE help)
.rem(-----)
.start(main,Contents,renault.bmp,Renault Help)
.rem(Insert table of Contents here)
.end
.end_file
```

SEE ALSO:

The .in Macro

SYNTAX: .in

EXPLANATION: This macro starts an indented text section, which usually contains a list of items. The indented section is a hanging indent, which means that the first line is indented less than the following lines. Each beginning line will look good if it begins with either a number or a bullet.

EXAMPLE:

```
.ent(sellcars,Sell Cars,How to )
.in
#n(1)Find a customer..n
#n(2)Convince them this is the car of their dreams, no matter how bad
it is..n
#n(3)Close the sale. Get their John Hancock on the bottom line..n
.un
.end
```

SEE ALSO: .un, ##, #b, #n

The .un Macro

SYNTAX: .un

EXPLANATION: This macro ends ('unindents') an indented text section.

EXAMPLE: See the example for the .in command.

SEE ALSO: .in

The ## Macro

SYNTAX: ##

EXPLANATION: This macro represents a bullet for a list item. It should be followed immediately by text.

EXAMPLE:

```
.ent(advantages,Quilt Advantages, )  
.s  
##Uses no electricity..n  
##All natural construction..n  
##Reduces number of blankets needed..n  
.end
```

SEE ALSO: .un, #b, #n

The #b Macro

SYNTAX: #b(text)

EXPLANATION: This macro represents a list item that starts with a bullet and with the given **text** in boldface. It should be followed immediately by text.

EXAMPLE:

```
.ent(glossary,Glossary, Keyboard )  
.s  
#b(Autorepeat)If held down a key will begin to send its code  
repeatedly..n  
#b(Mode key)Modifier keys, used in conjunction with normal keys.  
Includes shift, control and alternate..n  
#b(Scan code)A byte value or values sent to the computer..n  
.end
```

SEE ALSO: .un, ##, #n

The #n Macro

SYNTAX: #n(1)

EXPLANATION: This macro represents a numbered list item. It should be followed immediately by text.

EXAMPLE: See the example for the .in command.

SEE ALSO: .un, ##, #b

The .b Macro

SYNTAX: .b(string)

EXPLANATION: This macro presents the given **string** in bold face type.

EXAMPLE:

```
.ent(startup,Engine Startup, )  
Always ensure there is no gas fume build-up in the bilge. .b(THIS IS  
VERY IMPORTANT.)  
Switch on the ignition and press the start button..  
.end
```

SEE ALSO: .i

The .i Macro

SYNTAX: .i(string)

EXPLANATION: This macro presents the given **string** in italic type.

EXAMPLE:

```
.ent(chars,Characters,Game )  
Your opponents in the game consist of .i(trolls), rabid .i(dogs) and  
defenseless .i(kittens).  
.end
```

SEE ALSO: .b

The .[Macro

SYNTAX: .[

EXPLANATION: This macro marks the beginning of a region of text that will appear in a bold, fixed-width font. This is suitable for showing program examples or tables.

NOTE: Since RTE files use the open and close curly braces as part of their syntax, if you want to display them in your help file, you must 'escape' them by putting a backslash in front of them. This is shown in the example below.

EXAMPLE:

```
.ent(progctl,Control,Program )
The program is controlled with a message router:
.s
.in
.[
switch(wParam) .n
\{.n
\tab case ID_OPEN: . . . .n
\tab \tab break;.n
\tab case ID_CLOSE: . . . .n
\tab \tab break;.n
\}.n
.]
.un
.end
```

SEE ALSO: .]

The .] Macro

SYNTAX: .]

EXPLANATION: This macro marks the end of a region that was started with .[. There should be a .] for every .[.

EXAMPLE: See the example for the .[command.

SEE ALSO: .[

The .n Macro

SYNTAX: .n

EXPLANATION: This macro is used at the end of a line, or on a line of its own, to indicate the actual end of the line. When .n is not used, separate lines are simply run together and used to fill whatever WinHelp window width the user has chosen.

EXAMPLE:

```
.ent(drjnk,Health,To Your )  
This sentence will adjust itself to the  
width of the help window, since there is no  
'2en at the end of each line..n.n
```

```
This sentence will not adjust itself,.n  
since there is a \'2en at the end of.n  
each line..n  
.end
```

SEE ALSO: .s

The .s Macro

SYNTAX: .s

EXPLANATION: This macro is used between lines to skip a space. If .s is used, then don't use .n.

EXAMPLE:

```
.ent(morjnk,Health,To My )
This paragraph will be printed with the appropriate
word wrapping and line wrapping. All parts of the
paragraph will remain connected.
.s
The .s above will separate this paragraph from
the preceeding paragraph. The separation is slightly
more than a single line spacing.
.end
```

SEE ALSO: .n

The .end_file Macro

SYNTAX: .end_file

EXPLANATION: This macro marks the end of the macro file.

EXAMPLE: See the example for the .start command.

SEE ALSO: .start

The \brdrdb Command

EXPLANATION: This command changes the border line style to a double line. It is used as a modifier to the .box command.

EXAMPLE:

```
.box  
\brdrdb  
This text will be displayed in a box with a double line border.  
.bend
```

The \brdrs Command

EXPLANATION: This command changes the border line style to a single line (this is the default). It is used as a modifier to the .box command.

EXAMPLE:

```
.box
\brdrth
This text will be displayed in a box with a thick border.
\brdrs
And this text will be displayed in a another box with a single line
border.
.bend
```

The \brdrsh Command

EXPLANATION: This command changes the border line style to shaded style. It is used as a modifier to the .box command.

EXAMPLE:

```
.box  
\brdrsh  
This text will be displayed in a box with a shaded border.  
.bend
```

The \brdrth Command

EXPLANATION: This command changes the border line style to a thick line. It is used as a modifier to the .box command.

EXAMPLE: See the example for the \brdrs command.

The \li Command

EXPLANATION: This command changes the left margin indentation to the value that immediately follows the command. The value is expressed in twips. To reset the margin to its default, use \li180.

EXAMPLE:

```
.[  
\li720  
\ri720  
This sentence will be indented on the left and right margins by  $720/1440 = 1/2$ ".  
.]  
\li180  
\ri180
```

The \ri Command

EXPLANATION: This command changes the right margin indentation to the value that immediately follows the command. The value is expressed in twips. To reset the margin to its default, use \ri180.

EXAMPLE: See the example for the \li command.

The \ql Command

EXPLANATION: This command changes the paragraph style to left justified. This is the default paragraph style.

EXAMPLE:

```
\qc  
This paragraph will have every line centered in the help window.  
.s  
\ql  
Now we are back to the default left justified paragraph.
```

The \qr Command

EXPLANATION: This command changes the paragraph style to right justified.

EXAMPLE:

```
This paragraph is normally left justified.  
.s  
\qr  
And this paragraph is right justified.  
.s
```

The \qj Command

EXPLANATION: This command changes the paragraph style to fully justified.

EXAMPLE:

```
\qj
The quick brown fox normally jumps over the lazy dog's back in a fully
justified fashion.
.s
```

The \qc Command

EXPLANATION: This command changes the paragraph style to centered lines.

EXAMPLE: See the example for the \ql command.

The \tab Command

EXPLANATION: This command places a tab character in the macro and in the generated RTF file.

EXAMPLE: See the example for the .[macro language command.

The \hh Command

EXPLANATION: This command allows you to put extended characters into a help file. Some commonly used extended characters are

\a9 => ©

\ae => ®

\bc => ¼

\bd => ½

\be => ¾

\99 => ™ (trademark symbol. This one may not work correctly, depending on the help compiler)

EXAMPLE:

```
\a9 1994 Rimrock Software. All rights reserved..n  
Distributed on 3\bd double density floppy disk.
```

Glossary

- help compiler
- macro
- popup
- project
- RTF
- RTF commands
- template
- title bar
- toolbar
- topic
- twips

The replacement of a long series of RTF commands with a single command word and (optional) series of arguments.

A series of text lines that constitute the minimal macro file that will convert to an RTF file and compile to a help file.

A file with the extension .HPJ, that contains options that specifies how a help file is to be built. Required by the help compiler.

Rich Text Format, a standard text file formatting standard. An RTF file consists of RTF commands and normal text.

Special commands in an RTE file that describe how text is to be displayed. These commands start with a backslash (\) character.

The program that creates a help (.HLP) file from the .RTF file, using the rules detailed in the .HPJ file.

The line of graphical push buttons that are located directly below HelpGen's menu bar.

The area at the top of the HelpGen program window that contains the program name, file name, system menu button and window size controls.

A block of text in a help file that will be displayed as a single display page.

A small window that appears when you select a topic with a dotted underline. The window will remain until you click the left mouse button again.

A 20th of a point. There are 72 points in an inch, so there are 1440 twips in an inch.

