#1$2 Making Help Files

After creating a bug free program the lone programmer faces a greater challenge: *writing the help file!* Programmers who know the ins and outs of their development langauge find .RTF files harder to read than hex dumps! My modest contribution to the lone rangers of the computing world is **HELP.DOT**. This WinWord document template contains all the WordBasic macros and runs a couple of VB apps.

#3 $4 Rich Text Format

A format which embeds formatting codes in the text.
.RTF is a pure ASCII file.

3# RTF

4$ Rich Text Format

#5$6K7Setting Up Shop

You need to do three things prior to using HELP.DOT the first time: (1) Copy some files to directories of your choosing, (2) Add one line to the WIN.INI file and (3) Edit HCP.PIF. The following topics will tell you what you need to do:

5# SETUP

6$ Setting Up Shop

7K HCP.EXE;SHED.EXE:Directories;Help Macros

#8$9K10 Set Up: Copying Files

To insure this document template works, copy the following files to the appropriate directory.

Where WinWord looks for .DOT files (e.g. C:\WINWORD\DOT)

HELP.DOT
This file is in the HELPER.EXE archive.

In the WinWord directory (e.g. C:\WINWORD)

8514.TB8
VGA.TBV
These bitmap files are in the HELPER.EXE archive.

Anywhere on a drive in your system:

HCP.EXE	The Microsoft Help Compiler
SHED.EXE	The Hotspot Editor

You can download HCP.EXE and SHED.EXE from Lib 16 of the WinSDK forum on CompuServe.

In the directory where you will make help files (e.g. C:\WINWORD\HELP).

HELP.HLP	This help file
IMPPICT.EXE	A graphics file browser

These files are in the HELPER.EXE archive.

in the Windows\System directory

VBRUN200.DLL	The Visual Basic runtime library

You can download VBRUN200.DLL from Lib 5 of the MSBASIC forum on CompuServe.

See Also

Setting Up to Make Help Files
The Help Compiler PIF file

8# SetupCopyFiles

9$ Copying Files

10K Setup

#11 $12 K13 Set Up: The Help Compiler PIF file

Funny, but the compiler that creates all those neat .HLP files for Windows is a DOS application! In order to compile help files from the Windows host, you need to run the compiler in its own DOS box. HCP.PIF makes that possible.

HCP.PIF executes COMPILE.BAT, which is created when you run the **HELP.DOT:HelpCompile** macro. That creates a bit of a problem.

You see, rather than forcing you to either use ***my*** directory structure or edit a bunch of macros (the way I did in the first version of HELP.DOT), I decided to make installation easy. You can put things anywhere you want and locations will be stored in HELP.INI. But *you* must edit HCP.PIF to point to the directory where COMPILE.BAT will one day reside; the directory where you will create your help files.

Four things to remember:

1. HCP.PIF must be in the directory with your help compiler.
2. You must edit HCP.PIF to run COMPILE.BAT. Be sure to tell HCP.PIF which drive and directory will house COMPILE.BAT; this will be the directory where you told the install procedure you will use to create and store your help files.
3. Instructions for editing PIF files begin on page 259 of the Windows 3.1 User Guide. What you want to know is on the top of page 262.
4. I am not your mother. You can do this on your own.

See Also

Setting Up to Make Help Files
Copying Files

11# ModifyMacros

12$ Modifying Help Macros

13K Setup;Macros

#14$15K16 Making Help Files: An Overview

Creating help files is a simple process that only seems difficult. All you do is create two files and run the compiler. That is all you do!

1. Create two files:

   a. The .RTF file

      This is the file that contains the text and graphics that will be displayed in your help file. It has some unusual footnotes and formatting that govern how the help topics are displayed, how topics are linked and how the Browse and Search commands work. More on that later.

   b. The .HPJ file

      This is a file that tells the Help Compiler which .HLP file to create and what will be in it.

2. Run the compiler

3. Continue the Compile - Debug cycle until the help file looks the way you want it.

Help On

Writing the File
Creating the Project File
Compile and Test
Inserting Bitmaps

14# Overview

15$ Making Help Files: An Overview

16K Overview;Making Help Files

#17 $18 Making Help Step #1: Write the File

Help files begin life as text files in .RTF format. If you use a word processor capable of producing .RTF files, all you need to do is use certain character attributes a special footnote symbols to create all of the links, search lists and browse tables.

REMEMBER THESE FEW RULES AND YOU WILL START OFF OKAY

WinHelp displays each help topic one screen at a time.

Each topic must be on its own page in the source file.

Each help file screen is one topic. Each topic must be on its own page in a WinWord document.

Use your word processor to examine the sample file - HELP.RTF - from which I produced this help file. Notice that this topic is on its own page.

The Help compiler only works on .RTF files.

Be sure that you don't save files in WinWord's native file format. A common error is to save in WinWord's file format with an .RTF extension.

Use the File SaveAs command to save in .RTF format or use the FileSaveRTF macro.

Character attributes create jumps, popups and context strings.

While you are in the HELP.RTF, display the hidden text and look at the bordered paragraph above. Note that the first .RTF is ***single underlined***. Single underlining is for popup links. Note the ***hidden text*** immediately after the popup link. Hidden text is for context strings. Finally, notice that the word attributes is ***double underlined***. Double underlined is for jump or topic links.

Examples:

PopupLinkText*ContextString*	JumpLinkText*ContextString*
----------[Hidden Text]	**==========**[Hidden Text]

It is absolutely essential that no spaces intervene between a prompt text and context string!

Footnotes link jumps and popups with display topics and create titles and keywords.

This may be the hardest part for most people to understand. Probably the best way to figure this out is to browse through HELP.RTF and see how it uses footnote characters and footnote entries. Here are the important ones:

17# WriteFile

18$ Writing the File

is the footnote character used to signify a context string.

K is the footnote character used to signify keywords. Keywords appear in a list box when the user clicks the **Search** button.

$ is the footnote character used to signify a title. Titles appear in a box when the user clicks on a Keyword in the search box.

The toolbar buttons will automate all of this for you!

The first screen that appears when your help file is the Index.

The Index is the "top" of your help file. It's the screen that appears when the user clicks the **Contents** button in WinHelp.

If you use a different context string - which you are free to do - be sure to declare that context string as the index when you build the project file.

Now for the step-by-step explanation.

To create a popup box, you would.....

1. Toggle on the single-underlining.
2. Type the prompt text that you want to appear in your help file.
3. Toggle the single-underlining off and toggle hidden attribute on.
4. Type the context string ***immediately after the link text. No spaces between the prompt text and the context string!***
5. Insert a manual page break.
6. At the top of the next page insert the **#** footnote character. Use the custom footnote mark to do this!
7. In the footnote pane type the context string that you typed in step 4.
8. Close the footnote pane.
9. After the **#** character, type what you want to appear in the popup box.

To create a jump link to a new topic, you would....

Use double-underlining instead of single-underlining in step 3.

To create a Title you would.....

1. At the top of a new page type insert the **#** footnote character.
2. In the footnote pane, type the context string that will link this topic with a popup or a jump link from elsewhere in the help file.
3. Right next to the **#** character insert the **$** footnote character.
4. In the footnote pane, type the Title that you want to appear when the user searches the help file.

5. Close the footnote pane.

6. Type the topic just the way you want it to appear on the screen.

To create a Keyword you would....

Repeat the steps for creating a title, but insert a **K** instead of **$**.

Titles and Keywords must be associated with context strings! Otherwise they will be orphaned and not appear in the help file!

Whew! That's a long and obtuse explanation. Fortunately, all of this is automated for you by the macros in HELP.DOT. Take a look at the macros bound to the toolbar buttons and you will find that they handle all of this for you!

See Also

The Toolbar
Creating the Project File

#19$20K21 Context Strings

The most obscure part about writing help files is this notion of a ***context string***. A context string establishes a *link* between a prompt text (*the underlined, words in the help file than invite a click*) and a topic (*what appears on its own screen or in a popup box in response to the mouse click*). Look at this crude diagram which emulates what you might see when you examine a help files .RTF source code:

{page break}_____

Some Blather Here

Here is some sample*ContextStringSample* text. This is how it would appear in the .RTF file, blah, blah, blah... My fellow Americans, blah, blah, blah.

{page break}_____

Sample

A small portion, a taste, an example. A free demonstration copy. Etc.

==========[FOOTNOTE PANE]==========

*ContextStringSample*

The ***context string*** appears in two places. At the front end of the hyperlink it is attached to the prompt text; at the back end it is attached to the topic via a footnote symbol and footnote entry. When the user clicks the word sample, the context string points to the Sample topic which is displayed either as a popup box or separate topic.

If sample were double-underlined and *ContextStringSample* were hidden, WinHelp would the Sample topic on its own screen when the user clicked sample. If sample were single-underlined the Sample topic would appear in a popup box.

19# ContextDefine

20$ Context Strings

21K Context Strings

#22$23 Making Help Step #2: the Project File

The Microsoft Help Compiler uses a ***project file*** which is basically a Make file. The project file, which must have a **.hpj** extension, tells HCP how to make the help file. Although there are many possible settings, this document template builds only the basic settings. Clicking the **HPJ** icon on the toolbar presents this dialog box, which accepts values and builds the project file:

The dialog box has *hot spots*! Click on a control to see what it does!

{bmc e:\winword\bud\help\build.shg}

See Also

Writing the File
Compiling and Testing
The Help Compiler PIF file

22# ProjectFile

23$ Creating the Project File

Index

This is the context string of the very first topic in your help file. When the user clicks the Contents button, WinHelp returns to the Index. The Index is also the very first topic displayed when a help file is displayed. You can use anything you want as the context string for this first topic, but it may be a good idea to use the word *Index* as the context string! When you create the project file the context string must be entered in the Index text box!

24# IndexDefine

25$ Index

26K Index

#27 Help Window Title

This text box is where you enter the title that you want to appear in the help window title bar when your help file is used.

#28 Error Log

If you want the Help Compiler to write its error messages to a disk file, enter the name of that file in this text box. Be sure to follow correct DOS naming conventions because the macro which builds your project files does no error checking!

#29 Index

Enter the context string associated with the very first topic in your help file. This context string is typically **' *Index* '**, but it can be anything.

For example, this help file's very first topic, **Making Help Files**, has the context string **' *Index* '** attached to it.

#30 Warning Level

The Help Compiler will warn you of potential problems in your help file's source code. Level 1 gives the fewest warnings, Level 3 the most. I think you should use Level 3 during development.

#31 Compression

The Help Compiler will compress your files if you check this box. Because this selection slows compiling significantly, you should avoid selecting this option until the development cycle is complete.

#32 Map

Enter the name of the ASCII file which maps your help topics to integers. If you haven't built this file yet, don't worry. The MAP macro will take care of it for you and even insert the file's name into the project file.

#33 Build Tags

Build tags allow conditional compiles of the help file, including or excluding topics based upon their build tags. For example, if your help file attached build tags “WIN30” or “WIN31” to various topics, and you entered “WIN31” in this text box, the project file would tell the help compiler, “Only include those topics that are tagged WIN31”

You tag topics with build tags by using the FootnoteBuildTag macro. It is bound to the Toolbar button that has the big asterisk on it!

#34 Files to Include

Enter the names of the .RTF files that contain the source code from which the help file will be built. You don't have to use the .RTF file extension; the macro will strip out other extensions - if there are any - and append the .RTF extension.

Entering the name of a file that is not in Rich Text Rich Text Format will cause the compiler to abort.

File names should be separated by a **single space**.

#35 Okay

Once all fields are filled in and selections made, this button begins the compile.

35# BuildOK

#36 Cancel

This button aborts the compile.

#37 $38 K39 # Making Help Step #3: Compile and Test

Once you've created your help file source code and saved them in .RTF format, simply click the appropriate toolbar buttons to compile and then examine your help file. You can keep the .RTF file open in WinWord while it is being compiled.

See Also

The Toolbar
Writing the File
Creating the Project File

37# CompileTest

38$ Compile and Test

39K Compile

#40 $41 K42 The Toolbar

The toolbar automates the obscure, mundane tasks of making .RTF files for the Help Compiler. Run the cursor over it and an explanation of each button appears in WinWord's Status Bar.

When you open a document based on the HELP.DOT template the Toolbar will be positioned at the top of the screen and WinWord will be sized to fit beneath it. But the Toolbar will always remain on top; even when WinWord is maximized you can still see and use the toolbar.

{bmc e:\winword\bud\help\hlptool.shg}

See Also

Displaying the Toolbar
Writing the File
Context Strings

40# Toolbar

41$ The Toolbar

42K Toolbar;Macros

#43 Displaying the Toolbar

The bitmaps that HELP.DOT uses to display its custom toolbar were created with TBEDIT, a slick little program created by the fine folks at Pinecliffe International. Custom toolbars use an undocumented feature in WinWord and can, on rare occasions, be fooled by certain monitor-video card combinations. Here's what you need to do to use the custom toolbar.

Displaying

To display the custom toolbar, you need to make one modification to your WIN.INI file in the [MS Word 2.0] section. Add the following line:

LoadToolbarBitMaps=1

If you made this addition while you were running Windows, you will need to quit and restart Windows for the modification to take effect.

Troubleshooting

TBEdit needs to know two pieces of information in order to use your custom buttons. TBEdit will try its best to figure these out automatically. However, if the changes you are making to your custom Toolbar do not show up in WinWord, you may need to ensure that these two items are correct:

The Directory where Word for Windows 2.0 is Installed

If TBEdit guesses the wrong location of Word, the buttons will not appear. Create or edit the file TBEDIT.INI in your Windows directory, and add the following line:

```
[TBEdit]
WinWord=C:\WINWORD
```

Where "C:\WINWORD" is the location of your Word for Windows Program.

The Screen Resolution

Word for Windows uses two different Toolbar resolutions: VGA and 8514. Word uses a small amount of magic to choose which Toolbar to use. TBEdit tries its best to figure out which resolution Word will use. However, if you have changed the system font, or if you are using a not so standard video driver, you may need to add one of the following lines to the TBEDIT.INI file under the [TBEdit] heading:

For a VGA resolution Toolbar:

Resolution=1

For an 8514 resolution Toolbar:

Resolution=2

The above resolution lines will let TBEdit know which Toolbar Word for Window is using, so that TBEdit can create the proper buttons.

#44 $45 Camera

Activates IMPPICT.EXE which lets you browse your hard drives and look at bitmaps until you find the one you want. See Topic: Inserting Bitmaps

44# ToolbarCamera

45$ Camera

#46 Notepad

Starts the Windows Notepad.

#47 Map

Creates an ASCII file which maps help topics
to integers and adds the name of the ASCII
file to the [Include] section of the project file.

If the MAP already exists, this button will fire up the
Notepad and show you the MAP.

47# ToolbarMap

#48 File New

Opens a new document with HELP.DOT attached.

#49 File Open

The typical WinWord File open macro

49# ToolbarFileOpen

#50 File Save

Saves the current file in .RTF format.

50# ToolbarFileSaveRTF

#51 Standard Buttons

You probably know what they do!

#52 Toggle Hidden Text

Toggles hidden text off and on.

#53 Display Help

Displays this help file.

#54 Build Project File

Presents a dialog box from which the .HPJ file is made.

If the project file already exists, this button will fire up the Notepad and show you the project file.

#55 Shed

This button runs SHED, the Hotspot Editor.

#56 Compile Help

Compiles the current project into a help file.

#57 Test Help

Opens your help file so you can test it.

57# ToolbarTestHelp

#58 Color Auto

Sets the current color to “Auto”

58# ToolbarColorAuto

#59 Color Blue

Sets the current color to blue.

#60 Color Green

Sets the current color to green.

This might look good on some displays.

60# ToolbarColorGreen

#61 Color Red

Sets the current color to red.

61# ToolbarColorRed

#62 Color Magenta

Sets the current color to magenta.

62# ToolbarColorMagenta

#63 $64 K65 Inserting Bitmaps

Your help file will be more attractive and more easily understood if it contains bitmaps the represent different windows, dialog boxes, buttons, toolbars, etc. from your application. This document template provides a dialog box that facilitates the insertion of bitmaps into your .RTF files.

By using a screen capture utility such as HiJaak or Paint Shop Pro, you can include every one of your program's visual elements in the help file. With SHED you can create hot spots on these bitmaps that can let your users jump immediatley to appropriate help topics by simply clicking on a screen element with which they are already familiar!

The dialog box has *hot spots*! Click on a control to see what it does!

{bmc e:\winword\bud\help\bitmap.shg }

63# InsertBitmaps

64$ Inserting Bitmaps

65K Bitmaps

#66 Bitmap File Name

This box contains the name of the current file whose picture is displayed in the picture box. Clicking the Okay button will cause this file to be inserted by reference or value into the .RTF file at the cursor location.

#67 Bitmap Image

This box displays the bitmap whose name appears in the file name text boox. This picture box does ***not*** display .SHG files created or edited with SHED.

#68 By Value

This option inserts the bitmap image into the .RTF file.

68# BitmapValue

#69 By Reference

This option inserts the file name into the .RTF file

69# BitmapReference

#70 Bitmap Left

This option will cause the bitmap to be left-aligned in the help file.

#71 Bitmap Right

This option will cause the bitmap to be right-aligned in the help file.

#72 Bitmap Center

This option will cause the bitmap to be centered in the help file.

#73 Okay

When you have given all the instructions about this bitmap, this button fulfills your every wish.

#74 Quit

This cancels the bitmap operations.

#75$76K77 The Footnote Dialog Box

The most time consuming part of creating a help file is inserting the correct footnote characters in the proper order and making sure that you've used the correct character attributes in your document. This dialog box simplifies the creation of popup boxes, jump links and footnote characters.

The dialog box has *hot spots*! Click on a control to see what it does!

{bmc e:\winword\bud\help\dialog.shg}

To Create Popup Links

Select
Prompt Text Checkbox
Context String Checkbox
Popup Radio button
Type
The Prompt text in the text box
The Context String in the text box
Click OK

To Create Jump Links

Select
Prompt Text Checkbox
Context String Checkbox
Topic Radio button
Type
The Prompt text in the text box
The Context String in the text box
Click OK

To Create Context Strings, Titles and Keywords in Footnotes

Select
Footnote Checkbox
Type
The Context String in the text box
The Title in the text box [optional].
The Keywords in the text box [optional].
Click OK

See Also

75# ToolbarDialog

76$ The Footnote Dialog Box

77K Footnotes;Attributes;Links;Context Strings

#78 Popup

Neat little boxes that pop up onto your screen. Like this one!

#79 Jump Links

A link that takes you to another topic in the help file.
The word ***attributes*** in the first paragraph of this help topic,
at the top of the screen, is a jump link.

79# jump

#80 Context String

Prompts you for and inserts a context string.

#81 Keyword

Prompts you for and inserts keywords.
Should be used in conjunction with titles!

81# ToolbarKeyword

#82 Title

Prompts you for and inserts a title.

82# ToolbarTitle

#83 Build Tag

Prompts you for and inserts a Build Tag.

#84 $85 K86 Attributes

Attributes are the characteristics that distinguish the appearance of letters in a font set. The help compiler uses character attributes to build a help file's jump table, links and index.

Attribute	Use
Single underline	Creates a ***popup link***. Single underlined text is used to create a link with a popup box. When you click on the link text, the popup box appears.
Double underline	Creates a ***jump link***. Double underlined text is used to create a link with another topic in the help file. When you click on the link text, you jump to another location in the help file.
Hidden	Creates a ***context string***. Hidden text is used to connect both ends of the link. When you click on a *popup link* or a *jump link* the hidden text - which appears immediately after the link text in your .RTF file - tells the help engine where to go!

84# attributes

85$ Attributes

86K Attributes;Footnotes

#87 Prompt Text

Check this box if you want to enter a *link text* in your help file.
This will insert the prompt text you enter in the prompt text box
into your help file. In the .HLP file the prompt text will be underlined
and in color. The type of underlining (single or double) depends
upon whether you are creating a topic link or a popup link.

The words ***attributes***, ***popup*** and ***jump*** in this window are prompt text.

You create a popup link or a topic link by selecting the appropriate
radio button at left.

#88 Context String

Check this box if you want to enter a context string. ***If you are entering a prompt text, you must also check this box or the help file will not know where to jump to or which box to pop.***

To restate, this check box must be used in conjunction with either the Prompt Text or Footnotes check boxes. Otherwise, you will simply be inserting some useless hidden text in your file!

#89 Footnotes

Check this box if you want to insert the footnote characters for a topic, a popup box, a title or keywords.

If you checked the Context String but not the Prompt Text, then you ***must*** check this box. You would do this when you had created a prompt text and its context string manually but haven't yet entered it as a footnote at the beginning of the new topic (page).

#90 Popup Link

Selecting this button causes a popup link to be created. The prompt text will appear as single-underlined text in your work file.

The words ***popup*** and ***jump*** in the first paragraph of this help topic, at the top of the screen, in this window are popup links.

#91 Topic Link (aka Jump Link)

Selecting this button causes a topic link to be created. The prompt text will appear as double-underlined text in your work file.

The word ***attribute*** in the first paragraph of this help topic, at the top of the screen, is a topic link.

#92 Prompt Text Box

Enter the text that will actually appear in your help file as the prompt text.

#93 Context String Box

Enter the Context string that will link this prompt text to its topic. The context string ***must not*** contain any spaces or underscores!

BASIC's ***labels*** are a helpful metaphor for understanding context strings. A context string connects a prompt text to a topic. When the user clicks on a prompt text WinHelp displays the topic to which the context string points. Definition.

#94 Title

Enter the title that you want attached to the current topic. This will appear in the Scroll list when the user clicks **Search** in the help file.

#95 Keywords

Enter the keywords that you want associated with this topic. When the user clicks on a topic in the **Search** window of the help file, the keywords are listed in the box below.

Keywords must be separated by a semi-colon and no spaces. E.g.:

Compiling;Linking;Debugging

95# KeywordsText

#[96] Okay

Click this button to build the link, context strings and footnotes..

96# OKButton

#97 Cancel

This button cancels the process!

97# CancelButton

#98 L. E. Brown

Full time pastor, doctoral student and occasional dabbler in Clipper, Visual Basic and C.

If this help file and the document template were of any use, drop me a line and let me know. If you have some ideas for improvements, let me know that, too!

CIS 73277,3615.

98# Me