Getting Help With Help Systems

Although tools for developing help are archaic, the added value is worth the pain.

BY THEODORE KAHN

So, you've learned VB, finished your application, and are ready to start creating the help file: how difficult could this be? Hang on to your keyboard! While VB might be the most advanced programming environment commonly available, creating a Windows help file is more in the category of a blast from the past; it has very little to do with Windows itself and has no particular connection to VB. In fact, you can create a complete Windows help file without using Windows.

Most people think of online help as simply part of the documentation. While this is true, it can be much more. The help system in Crunch Software's statistical application has become a key selling point because of the added value and capabilities. Computers can present and relate information in ways that are impossible using printed media. Not only do help systems use hypertext, but multimedia as well, including graphics, sound and video. In the future, help will be so well integrated into the application that the two will be indistinguishable. Researchers in the field of computer/human interfaces recognize this trend. For example, computer scientists at Carnegie Mellon University in Pittsburgh, Pennsylvania have targeted help systems as a key area for advancing software ease of use, now that the fundamentals of graphical user interfaces have become widely accepted. However, help could be used more extensively than it is now, in part because help authoring tools are still primitive.

But before I get into the technical details of creating a help file, it is important to remember that a help file is a hypertext document. This means that a reader can jump from topic to topic within the help file and even to other help files. These jumps can be initiated in a variety of ways: by clicking on jump (green) text or a graphic; by clicking the Back, History, or Browse buttons (<< or >>), or by selecting a topic from the Search Dialog Box, to name a few. All these methods are under your control

Theodore Kahn is a software designer, programmer, and documentation writer for Crunch Software Corp. in Oakland, California. Crunch develops statistical software for DOS and Windows, and its current Windows project is being supported by the National Cancer Institute under a Small Business Innovation Research (SBIR) contract. Reach Ted via Internet at tedkahn@netcom.com or CompuServe at: 70353,2603.

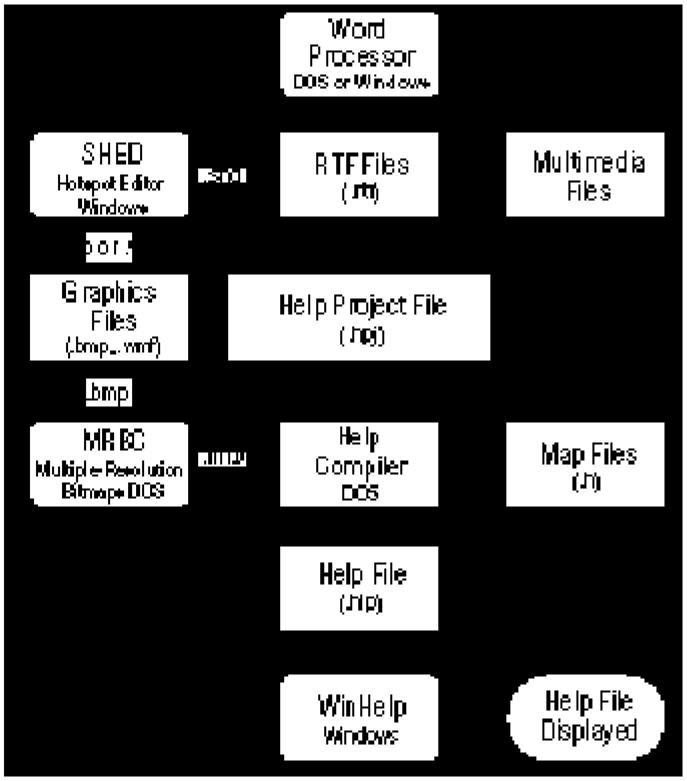


FIGURE 1. This diagram shows how the various programs and files required to create a help file are organized. The rounded rectangles represent programs and the square ones represent files. The Help Project File references the files pointing toward it. This information is then used by the Help Compiler to create the help file.

as the help file creator. (don't forget that you—or someone else if you're lucky—also has to write the text). The process of creating a help file requires specifying the hypertext jumps and visual layout for each topic. The topics, in turn, must relate logically and concisely to the program for which the help file is being written.

Many help file creators use Microsoft WinWord Version 2.0 as their word processor. We have therefore made several references to WinWord regarding the performance of specific tasks. If you are using a different word processor, look to its equivalent feature for executing the function. In addition, the features discussed here refer to Windows 3.1. If you design your help file to run under Windows 3.0, you must compile it using the 3.0 Help Compiler (HC30) or you will need to include WINHELP.EXE 3.1 with your help file.

If all this sounds difficult, it is: You are using yesterday's technologies to create state-of-the-art hypertext multimedia online documents. This doesn't make a lot of sense and the process is tedious at times, but it's reality. So put on your Simon & Garfunkel or Aretha Franklin records (no CD's) and join me in what can be best described as The Twilight Zone of Windows programming.

CREATING THE HELP SYSTEM

Creating a help file is really very similar to programming in more traditional languages under DOS (see Figure 1). You need one or more source files; in my example the source files are word processing files saved in rich text format (RTF), and a Project file that is analogous to other languages' MAK files. The Project file specifies help compiler options and the RTF file names (see Listing 1). If your help file requires graphics or other multimedia files, these too are referenced in the Project file. From DOS, run the Help Compiler and provide the Project file as its argument. The result is your help file, which requires WINHELP.EXE (a Windows program) to view.

If the Help Compiler detects an error, you'll need to edit the Project file and/or word processing file (in the latter case you also need to regenerate a new RTF file) and then rerun the Help Compiler. When no errors are found, go to Windows and run WINHELP.EXE to look at the help file. To be thorough, check all the hypertext links and macros. Again, if errors are encountered, make the appropriate changes and recompile. As in any programming project, the number of possible errors can be enormous, and their interactions may be unpredictable.

When you first get started building a help system, you create a help topic in your word processor. So let's dig in here with a real-world example of what happened to me the first day I tried to create a help file.

My help authoring tool (like many) consists of a set of macros for WinWord 2.0. The first thing I did was fire up WinWord, choose File New and select the template provided by the

```
[OPTIONS]
COMPRESS=False  ;Set to True for final release
WARNING=3  ;Show all errors
REPORT=On  ;Show compile progress
ERRORLOG=c:\sample.err  'Errors are written to this file
COPYRIGHT=Copyright ©1993 Your company  ;Included in the Help About
CITATION=Copyright © 1993 Your Company  ;Included in the Copy Dialog
[CONFIG]
BrowseButtons()  ;Adds Browse buttons to Help file (<< and >>)
CB("Glos_Btn","&Glossary","JI('sample.hlp>Glos_Win','Glossary')")
[FILES]
;This file is created by your word processor or Help Authoring Tool
SAMPLE.RTF
[MAP]
;Context-string <space> context number (aka HelpContextID)
Glossary  2
Read_Me_First  3
[BITMAPS]
c:\e 2\a_to_z.shg
c:\e 2\howdone.bmp
c:\e 2\computer.wmf
[WINDOWS]
main="Sample Help File",,0,,(255,255,128)
Glos_Win="Glossary"  ,(550,300,349,600),0,(0,192,192),(192,192,192)
```

A Little Help. This sample Help Project File is an ASCII file containing two types of information: Options and other parameters that define various help file characteristics and, references to the word processing and graphics files that will comprise your help file.

help authoring tool. A Dialog box popped up and asked me the name of the file and project title. I entered the information and pressed return. This is what I saw:

#$K+ Contents
The following Help Topics are available:
List of Help Topics goes here.
For Help on Help, Press F1

I was totally stumped. What did all this mean? What was I supposed to do next? Where was my help authoring tool? To be honest, it took me days to figure this out. I pored over the Help Compiler documentation and studied the ICONWORKS example, both of which came with VB Professional. Oh, and of course I read all 55 pages of the help authoring tool manual and then some (see sidebar "References And Resources," p. 35). It seems to me today that I spent the better part of two weeks trying to get a handle on what was going on. So I'll save you a couple of weeks and boil down what I learned:

1. A help file consists of one or more topics (there is no practical limit to the number of topics). In other words, the topic is the unit in which you work.
2. A topic can have any amount of text and/or graphics. Keep them small. If a topic gets too big, break up the text into multiple smaller topics and relate them with jumps, browse sequences, and keywords (more on these concepts later).
3. The order in which topics appear in your word processor (or help authoring editor) has no bearing whatsoever on their appearance in the help file. This is a critical concept to understand: it represents your introduction to hypertext.
4. The #$K+ characters are not really text, but custom footnote characters which are called Control codes. They are usually the first characters in the topic. The footnote contents for each Control code contains the information that controls the hypertext nature of your help file. The remaining text and graphics in the first paragraph (in this case the word "Contents") becomes the topic title. Additional paragraphs contain the topic text and graphics.

There are two other footnote characters that do not have a hypertext function. The * footnote specifies the build-tag. This allows you to conditionally include (or exclude) topics dependent on the build-tag string. This feature is useful if you need to create multiple versions of the help file. The ! footnote lets you specify a macro that is executed whenever the topic is displayed.
5. The end of a topic is indicated by a hard page code. In WinWord press Ctrl-Enter (it is shown as a thin horizontal line). The characters after the hard page code will then be the Control codes for the next topic, and so on.

So my help authoring tool automatically created the first topic. It called the title "Contents" and entered four lines of boilerplate text. I was supposed to replace the line "List of help Topics goes here" with a series of jumps to other topics that I would add later.

This all boils down to some pretty simple stuff: If you can figure out how to enter custom footnote characters into your word processor, you can create help files without a help authoring tool. In WinWord Version 2.0, choose Insert, Footnote and select Custom, then enter one of the four characters #$K+. Then enter the footnote text. This all seems pretty straightforward to me

Go to next page.

now, but it took some time before I really felt I understood this well.

UNDERSTANDING HYPERTEXT

You cannot write a good help file without truly understanding how the hypertext Control codes work. For me, this was one of those learning experiences where I always seemed to think I understood what was going on until I looked at the result. Trying to think and write in hypertext is fundamentally different from writing according to the way I've been taught. I can still remember my third-grade teacher Mrs. Murray saying "Tell them what you're going to say, say it, and tell them what you said." It doesn't work in this situation. Mrs. Murray's model used words to lead in and out of ideas and concepts: Hypertext uses Control codes *and* words.

On a physical level, this part of the article relates to only what you enter for the four Control code (# $ K +) footnote contents. These four Control codes define the hypertext methods by which a topic can be

The Microsoft Help Compiler documentation that comes with VB Professional is terse, and for the beginner, difficult to understand. Even worse, it's incomplete. Here are some places to look for additional information to supplement what Microsoft provides.

• If you have VB professional, the HC subdirectory contains all the programs necessary to create a help file (except a word processor), additional documentation relating to API calls, and a sample project (ICONWRKS).

• I highly recommend the book *Developing Online Help for Windows* by Scott Boggan, et al, Sams Publishing, 1993, (800-428-5331, $39). It covers everything from writing styles for help files to a spreadsheet to project development costs. It also includes a disk with a number of sample help files that illustrate various features. The disk also has a simple help authoring tool (macros for WinWord Version 2).

• The *Help Authoring Guide*. This is a help file created by Microsoft (possibly as a precursor to a more comprehensive help document). It contains far more information than the manual that comes with VB. The file is available on CompuServe in the WinSDK forum, library 16 as file HAG.ZIP. This is a highly technical document—most people would be better served getting Boggan's book.

• We have created a sample help file and VB program that illustrates many of the issues and features discussed in this article, and a lot more. It's called HELPINFO.ZIP. Probably the easiest and fastest way to learn how a help file is put together is to look at sample code. However, there are very few help files that also include the Help Project file as well as all the word processing and graphics files that go into creating help files. That's exactly what's included in HELPINFO.ZIP.

Of particular interest is a glossary with all the bells and whistles: graphics, hot spots, accelerator keys, secondary windows, and much more. All you need to do is add your specific text. In addition, we have also included a VB program (with source) that illustrates how to integrate the help into a VB program. All the API declarations, along with just about all the code you can use is in there. You are free to use any part of the help file or program in your own projects. The only thing we ask is that you do not sell the files as is. You'll find HELPINFO.ZIP on the Internet (FTP to ftp.cica.indiana.edu and go to /pub/pc/win3/programr/vbasic) or CompuServe (WinSDK forum library 16 and MSBASIC Programming Lib). n displayed. A topic can have any combination of these Control codes. If it has none, it cannot be displayed. Figure 2 illustrates how footnote information is translated into hypertext jumps. Let's take a close look at each of these important codes:

1 The # code refers to the context string. This is a string of characters that uniquely defines a topic. That is, each topic must have a unique string. For example, enter print_dialog_box as the text for the # footnote to define that string as the topic context string (spaces are not allowed). If a topic does not have a # footnote, there is no way to jump to it. By jump, I mean that green text you see in help files—you click on it and a different topic is displayed. Usually, you want users to be able to jump to topics, so most topics have context strings.

There is one very important fact to know about context strings: A topic can have any number of context strings and the # footnotes can be placed anywhere in the topic.

When you jump to that context string, the text just after its corresponding footnote (#) is displayed at the top of the help screen. This is sometimes referred to as a mid-topic jump. The VB Help Glossary is a good example of this feature.

The mid-topic jump is virtually undocumented. To the best of my knowledge no help authoring tool supports it. Therefore, if you want mid-topic jumps, it's likely you'll have to code them.

2. The + code refers to the browse sequence. You may have already noticed the two buttons labeled << and >> at the top of the help screen. This is the Control code that drives those buttons. The browse sequence provides the user with a quick and convenient way to move through related topics in an ordered fashion. Forward or backward movement is accomplished by clicking the browse buttons. The footnote text is composed of two parts—the group name and a sequence number, separated by a colon. Continuing with the Print example, suppose you had the following three topics: Printing Overview, Print Dialog Box, and Print Dialog Box Options. The information goes from general to specific, and so follows a natural progression. In this case, the browse sequence might look like this:

Printing Overview	PRINT:010
Print Dialog Box	PRINT:020
Print Dialog Box Options	PRINT:030

The group name is optional. Some help authoring tools do not use a group name or they use the same group name for all topics. This diminishes the browse sequence utility; jumps can be made to unrelated topics. If the topic has an ordered relationship to other topics, code the topics with a browse sequence accordingly. Otherwise, do not include this control code.

3. The $ code refers to the title in the Search Dialog Box topic list, and K refers to the topic keyword phrases. These two Control codes work together, and usually you will include or exclude them as a pair. This gets a bit tricky, so go slow. The first thing to know is that the text for the $ footnote can be different from the title that appears in the topic title paragraph, and both can be different from the context string. Generally, however, all three are the same (of course, underlines would be substituted for spaces in the context string).

The text of the K footnote contains the keyword phrases. This text appears in the top part of the Search Dialog Box. The text of the $ footnote contains the title as it will appear in the bottom part of the Search Dialog Box when one of its keyword phrases is selected.

To make this work correctly, related topics must have the same keywords. Pay particular attention to word endings. In Table 1 I used "printing" in all three topics. If the user selects "printing" from the keyword list, all three topics will appear in the topic list box, the user can select the most appropriate one. Note the "document" keyword for the Print Dialog Box topic. This help file might also have topics titled

CONTINUED ON PAGE 107.

CONTINUED FROM PAGE 36.

Opening Documents and Closing Documents, each with the keyword "document." Then when the user selects "document," the Open, Print, and Close topics (and maybe others) are displayed. Keyword footnotes (like context string footnotes) can be placed anywhere in the topic, not just at the beginning. When the user selects a topic having a keyword in the middle, the topic is displayed with the keyword at the top. This feature is not well documented, and few help authoring tools support it. This is difficult stuff to really understand and implement well, especially as the number of topics and keywords go up. Some help authoring tools provide various ways of organizing footnote text, as in Table 2.

Now let's take a look at pop-up topics. Some topics in a help file appear in a little pop-up window which goes away when you click it. These topics are called pop-up topics. You do not program pop-up topics, *per se*. Whether a topic appears as a pop-up or regular topic depends upon the type of hypertext jump that caused the topic to be displayed. It took me forever to figure this out.

As a general rule, topics that appear only in pop-ups do not have browse sequences, or keywords and titles (+K$ footnotes). This is why help authoring tools ask you whether a topic is to be a pop-up of not. If you select Yes, then they don't include these footnotes.

TACKLING TOPIC JUMPS

Jumping from topic to jump is the most fundamental hypertext attribute used. Previously, I discussed the # footnote, which contains the context string. This is the address that defines the place you'll be jumping to. Next, we need a method to define the start of a jump, both to the Help Compiler and to the user. Since there are two types of jumps—regular and pop-up, there are two methods to indicate the type of jump to be made.

The starting point, function, and destination are all conveyed to the Help Compiler through three different character formatting attributes shown in Table 2:

The hidden text follows immediately after the underlined text. No spaces are allowed.

That's all there is to it. If you can figure out how to underline characters and mark them as hidden in your word processor, you can program jumps. This is not very intuitive, and so at first it feels strange and somewhat complex. But consider how simple it really is. One more thing, you can also underline (double or single) a graphic, instead of text. Then the jump occurs when the user clicks on the graphic. Here's what a jump to our printing_overview topic looks like (the bold text is marked hidden):

```
#$K+ Contents
How to print a
document printing_overview
List of Help Topics goes here.
For Help on Help, Press F1
```

This syntax, underlined text or graphic, followed by hidden text, can also be used for jumps to secondary help windows, other help files, and for executing help compiler macros. Once you understand this concept, you'll really be able to make your help file jump to life.

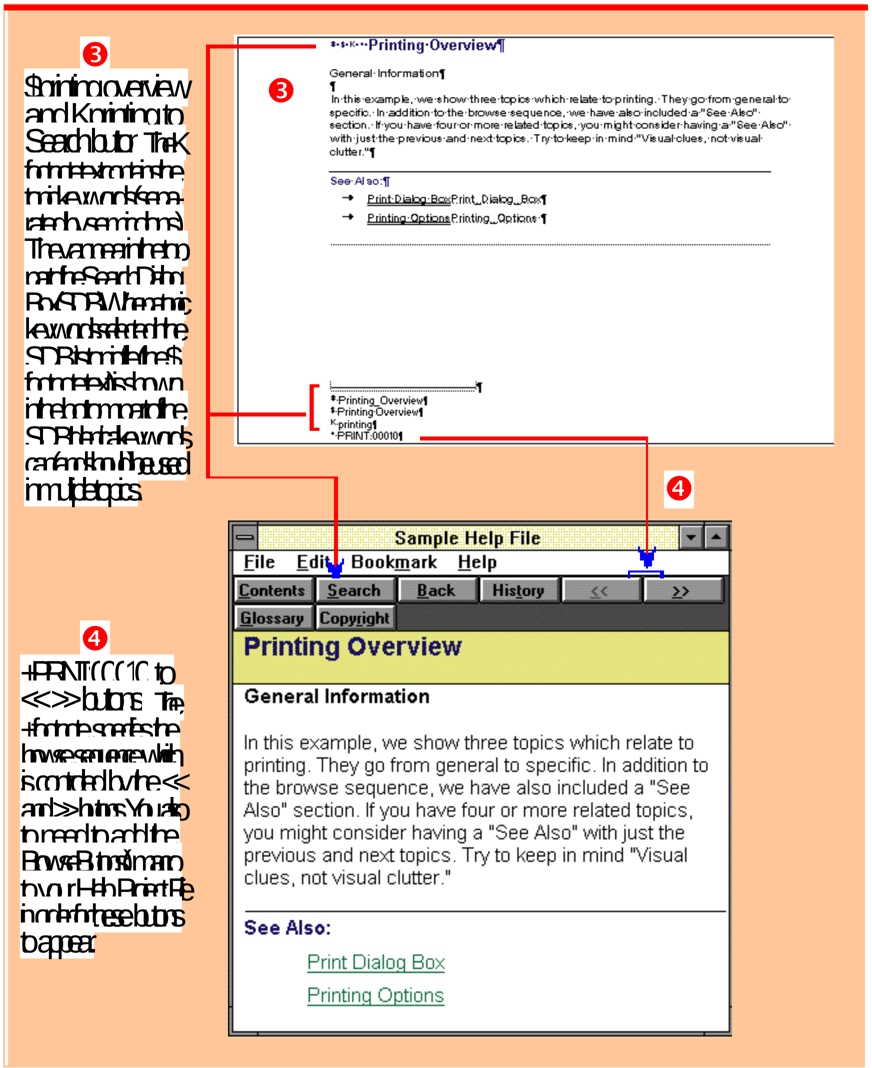
A final note. Most help authoring tools automate standard jumps, as discussed above. However, they don't always provide all possible options all the time. The mid-topic jump previously mentioned is one example. Another is allowing jumps in the topic title. Again, look at VB Help for an example of topic title jumps. Many of the titles have "See also" and "Example" as jumps in the title—a nice touch, I think.

GUIDE TO GRAPHICS

Graphics can really add a lot to a help file, and you should look for creative ways to use them. For example, all our program toolbar icons are in a topic, and next to each we've included their keystroke equivalents with a one-line explanation that can be clicked on for more detailed information. Another nice place for graphics is in the topic title. Again, look at the VB Glossary.

The Help Compiler supports four types of graphics files: bitmaps (BMP), windows metafiles (WMF), hypergraphic (SHG), and multiple-resolution bitmaps (MRB). The latter two formats are special formats that can only be used by the Help Compiler. The programs to create SHG and MRB files are included with VB professional.

All graphics are generally entered "By Reference," meaning that you enter a code with the graph file name at the place where the graph is to appear. Then, in the Help Project file [BITMAPS] section you enter the full file name. This allows you to display the graph in different places in your help file, but store only



❸ $printing overview and Kprinting to Search button. The K footnote text contains the topic keywords (separated by semicolons). The value in the top part of the Search Dialog Box (SDB). When a topic keyword is selected, the SDB's list in the bottom (the $ footnote text) is shown in the bottom part of the SDB. Note that keywords can (and should) be used in multiple topics.

❹ +PRINT:00010 to << >> buttons. The + footnote specifies the browse sequence which is controlled by the << and >> buttons. You also need to add the BrowseButtons() macro to your Help Project File in order for these buttons to appear.

#$K+ Printing Overview¶

General Information¶
¶
In this example, we show three topics which relate to printing. They go from general to specific. In addition to the browse sequence, we have also included a "See Also" section. If you have four or more related topics, you might consider having a "See Also" with just the previous and next topics. Try to keep in mind "Visual clues, not visual clutter."¶

See Also:¶
→ Print Dialog BoxPrint_Dialog_Box¶
→ Printing OptionsPrinting_Options¶

¶
Printing_Overview¶
$ Printing Overview¶
K printing¶
+ PRINT:00010¶

Printing Overview

General Information

In this example, we show three topics which relate to printing. They go from general to specific. In addition to the browse sequence, we have also included a "See Also" section. If you have four or more related topics, you might consider having a "See Also" with just the previous and next topics. Try to keep in mind "Visual clues, not visual clutter."

See Also:

Print Dialog Box

Printing Options

Each topic in your word processing file has encoded into it information that defines its hypertext nature to the Help Compiler. The top part shows how a topic looks in WinWord while the bottom shows how it is displayed in the help file. The title (yellow background) is nonscrolling (the title paragraph code "keep with next" is selected). The line above "See Also" is a top border.

Even if you are only going to write a small help file, you should look into getting a help authoring tool of some sort. Coding help topics is just too tedious to do manually. (*Developing Online Help for Windows* mentioned elsewhere includes a basic help authoring tool for WinWord Version 2.) help authoring tools generally break into two categories: those that include their own editor and those that work with a commercial word processor. I recommend those that use a commercial word processor. That way, if the help authoring tool does not support a feature, such as tables or mid-topic jumps, you can code them yourself. help authoring tools that provide their own editors allow you very little flexibility in this area. Here are some products to consider:

These products use WinWord

CreateHelp, Shareware, $40, CompuServe 100111,3452. In MSBASIC forum, search for the file CH**.ZIP. The asterisks indicate the release. Nice program, good value. Support and manual are limited. This would be a good first program to try, and it may be all you'll need.

Doc-To-Help, WexTech Systems, $295, (212) 949-9595. Some nice features for turning existing Word documents, such as procedures manuals, into help files.

HelpBreeze, Solutionsoft, $279, (408) 736-1431. Very nice program. Includes the right set of well-implemented features. Well-written manual and good support. Highly recommended.

RoboHelp, Blue Sky Software, $495, (800) 677-4946 or (619) 459-6365. Includes extra programs for working with graphics, such as screen and window captures.

These products include their own editors

Help Magician, Software Interphase, Inc. $199, (401) 397-2340. Nice support for multimedia.

VB/HelpMaker, Teletech Systems, Fred Bunn, CompuServe 72260,2217. Some nice features for integrating your VB program into the help file.

VisualHelp, ShareWare, $49, (800) 242-4775 or (713)-524-6394. In MSBASIC forum, search for the file VH.ZIP. Fun and simple to use for small help files, but has no editor. Instead, text is entered into text boxes. ■

one copy.

The code is {bmx *X filename.ext*} where *X* is L for a left-justified graphic, C for a character, and R for a right-justified graphic. (WinWord users note: this is not a field code, which the braces would otherwise indicate.)

The BMP and WMF formats need no explanation; you are, however, limited to 16 colors with this method. The MRB format is really only necessary if you intend your application to work with CGA and EGA monitors. The format of greatest interest is Segmented Hypergraphic, or SHG.

SHG files are BMP or WMF files that have been processed by the hot spot editor, SHED.EXE. SHED allows you to define hot spots (rectangular regions) to a BMP or WMF file. A jump (or macro) is then assigned to each hot spot. When the cursor is positioned over a SHG hot spot, it changes to a hand indicating to the user that clicking the mouse initiates the jump (or macro).

SECONDARY WINDOWS AND MORE

The [WINDOWS] section of the Help Project file lets you control the position, size, and background colors for the regular text and nonscrolling region (if defined) for each window. This is well documented in the manual that comes with the Professional Edition (*Professional Features, Help Compiler Section* p.128-130. Look at the bottom of page 132 for an example.)

The Help Compiler allows you to display two (or more) help windows at the same time. Putting a glossary topic in a secondary window is now commonly done. Also consider putting index type material in a secondary window with jumps to the primary window containing the detailed information. Another use for a secondary window is for a special graphic or other topic that is independent of the normal help flow.

Adding a secondary window is simple: add a line to the Help Project file [WINDOWS] section for each secondary window. To display a topic in a secondary window you need to also specify the help file and window name in the jump hidden text. (Remember, jumping to a topic requires double-underlining some text and following that immediately with the context string of the destination topic text.) The full syntax of the hidden text is:

```
context-string>WindowName@HelpFilename
```

If the window name is "g_win," the help file is "myfile.hlp," and context string is "glossary," the hidden text is:

```
glossary>g_win@myfile.hlp
```

Note that this syntax allows you to also jump to different help files. Secondary win-

#context string	$title	K keyword phrases	+browse sequence
printing_overview	Printing overview	printing	PRNT:010
print_dialog_box	Print Dialog Box	printing document	PRNT:020
printing_options	Printing Options	printing	PRNT:030

TABLE 1. Use the same keywords for related topics.

character formatting*	Function	Appearance in Help file
Double underline	Static regular jump	Green text with solid underline
Single underline	Static popup jump	Green text with dotted underline
Hidden text	Context string associated with the destination text. Usually the beginning of a topic.	

*applied to the text in the word processing file.

TABLE 2. Authoring tools provide a number of ways to organize text.

dows do not normally contain menu items or buttons. Therefore, you may want to add a jump back to the main window. Of course, the user can close the secondary window using the Control Box or pressing Alt-F4.

Topics appearing only in secondary windows should be treated the same as pop-up topics when it comes to the Control codes. That is, they should only be given a # context string footnote. Otherwise, the user will be able to select the topic from the Search Dialog Box and it will appear in the main window.

Another common feature is to make the title a nonscrolling region—the title stays at the top of the screen as the user scrolls through the text. It's a nice touch. This attribute is set on a topic-by-topic basis. To specify a nonscrolling region for a topic, set the title paragraph attribute "Keep with next" on. (This is the attribute that keeps two paragraphs together.) A note of caution: make sure this attribute is not set for topics appearing in pop-ups. If it is, the topic text is not displayed.

Although I've covered only the fundamentals, this information, with the Microsoft documentation, should be sufficient for you to create a help file without any special help authoring tools, other than a word processor capable of outputting RTF files (see sidebar, "Help Authoring Tools"). Create three- or four-topic help files and try putting in jumps, pop-ups, keywords and browse sequences so that you can get the hang of the process. Also, define and jump to a secondary window.

Because of length limitations, there are two areas I did not cover: macros and integrating the help file with your program. These subjects are covered in a sample help file and VB program available on the Internet and CompuServe's MSBASIC forum. If you download and examine these files, you'll be a long way toward developing your own custom help system. ■