MiniHelp Plus v3.2

The <u>Complete</u> Help Writers Authoring Utility

Users Manual

Table of Contents

Table of Contents (continued)

Alphabetical Table of Contents

Introduction

Welcome to MiniHelp Plus v3.2, the complete Help Writers Authoring Utility. With MiniHelp Plus v3.2, you can take an ordinary ASCII Text File and convert it to the RTF formatted file the Help Compiler needs to construct your finished Windows Help file. MiniHelp Plus v3.2 also produces a basic Help Project File (an ASCII text file that tells the Help Compiler what to do) as well as a C-type Header file (for defining **context-sensitive** Help) at the same time your ASCII text file is being converted to an RTF file. You simply construct your ASCII text file (containing any of the 33 MiniHelp Plus v3.2 commands), run MiniHelp Plus v3.2 on your ASCII text file, then run the Help Compiler on the resultant Help Project File. Your completed Help file can then be viewed.

MiniHelp Plus v3.2 makes writing your own Help files simple, inexpensive, and **very fast!** The commands are uncomplicated, easy to remember, and easier to use than with any other method I have ever used to create Help files. See, I really like writing Help files and figured there ought to be a quick way to create them without having to fork over $400 or better for Microsoft Word for Windows (or some other high-dollar word processor capable of RTF format output). MiniHelp Plus v3.2 was developed so that **anyone** can easily start constructing their own Help files quickly, with a minimum of commands to learn, at a very low cost, and produce professional-looking results. And what's even better, MiniHelp Plus v3.2 costs about 1/20th of Microsoft Word for Windows (W4W)...only **$25.00** (yes, MiniHelp Plus v3.2 is shareware) and it's so much easier to use than the more expensive W4W.

Complete online Help is available in the Help files, **mhplus32.hlp** and **mhpshell.hlp** (which, incidentally, were written entirely with MiniHelp Plus v3.2). These Help files will also give you a good idea of just what all the features of MiniHelp Plus v3.2. Refer to these Help files for detailed information on RTF formatting commands that may be used in MiniHelp Plus v3.2, as well as detailed information about the Help Project file. RTF formatting and the Help project file are not covered in depth in this manual. Here, we learn how to run MiniHelp Plus v3.2. Features new to MiniHelp Plus v3.2 are indicated with the word ***NEW!*** at the beginning of the paragraph explaining the new feature.

License Agreement

Copyright (c) 1994 Paul Arnote
All rights reserved

INSTALLATION OF **MINIHELP PLUS V3.2** ON YOUR COMPUTER SYSTEM IMPLIES AGREEMENT WITH THE TERMS AND CONDITIONS BELOW.

DISTRIBUTION OF **MINIHELP PLUS V3.2**, ITS ACCOMPANY PROGRAMS AND DOCUMENTATION IS CONSIDERED AS IS. THE AUTHOR/PROGRAMMER OFFERS NO WARRANTIES OF ANY KIND, EXPRESSED OR IMPLIED. THIS INCLUDES, BUT IS IN NO WAY LIMITED TO, WARRANTIES OF **MINIHELP PLUS V3.2**'S MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. UNDER NO CIRCUMSTANCES WILL THE AUTHOR/PROGRAMMER BE LIABLE FOR ANY DAMAGES WHICH RESULT FROM THE USE OF THIS PROGRAM OR THE INABILITY TO USE IT. EXCLUSION FROM LIABILITY INCLUDES, BUT IS NOT LIMITED TO, LOST PROFITS, LOST SAVINGS, OR ANY OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES.

MiniHelp Plus v3.2 is distributed as Shareware. It is not free, freeware, or in the public domain. You may use **MiniHelp Plus v3.2** for a trial period of thirty days, at no cost to you, to determine if it fits your needs. If you decide to use **MiniHelp Plus v3.2** regularly, you are expected to register it and pay the applicable registration fee. Individual copies of the unregistered version of **MiniHelp Plus v3.2** may be given to your friends and associates for the same thirty day free trial period. You may also upload **MiniHelp Plus v3.2** to the public section of a public BBS.

You may not modify or disassemble **MiniHelp Plus v3.2**, nor distribute any modified or disassembled versions of **MiniHelp Plus v3.2. MiniHelp Plus v3.2** may not be included with any other product without written permission from the author/programmer.

Registered copies of **MiniHelp Plus v3.2** can be used on more than one computer at a time as long as no more than one of these computers is running **MiniHelp Plus v3.2** at the same time. You may make backup copies of **MiniHelp Plus v3.2** as necessary for archival purposes only.

Getting Started

Installation

To have the MiniHelp Plus v3.2 files installed automatically in Program Manager, from the **FILE** menu of Program Manager, choose **RUN**. Type **a:\setup** (if installing from a floppy) or **c:\***directory containing MiniHelp Plus v3.2 files***\setup** (if installing from your hard drive). Everything from here on out is automatic. Just follow the screen prompts and MiniHelp Plus v3.2 will be installed in the directory **mhplus32** (or in whatever directory you designate, if you choose a different directory name).

File List:

MHPLUS32.EXE	MiniHelp Plus v3.2 Program
MHPLUS32.HLP	MiniHelp Plus v3.2 Online Help file
MHPSHELL.HLP	MiniHelp Plus v3.2 ToolPad Help file
MANUAL32.WRI	MiniHelp Plus v3.2 Users Manual
README.TXT	Notepad text file with installation instructions
REGISTER.TXT	Notepad text file with registration form.
HOWTORUN.TXT	Notepad text file with How-To-Run instructions
STEPS.TXT	Notepad text file with Step-By-Step Guide
HCP.EXE	Freely distributable MS Extended Help Compiler
HCP.ERR (must be present for HCP to run!)	Help Compiler error code file
MHPSAMPL.SRC	Sample MiniHelp Plus v3.2 source file
DISK.BMP	Bitmap for sample source file
BTN_UP.BMP	Bitmap for sample source file
KEY1.BMP	Bitmap for sample source file
FOOT.BMP	Bitmap for sample source file
MHP23D.ICO (Windows Directory)	Icon for MiniHelp Plus v3.2 ToolPad Help file
MHP25B.ICO (Windows Directory)	Icon for MiniHelp Plus v3.2 Online Help file
MHPICONS.DLL	Icon Collection for MiniHelp Plus v3.2
CTL3D.DLL (in the Windows System Directory)	Dynamic Linked Library for 3D look

How To Run MiniHelp Plus v3.2

MiniHelp Plus v3.2 EXPANDS THE new front end Windows® shell that was introduced with MiniHelp Plus v3.1. The same ease of use in MiniHelp Plus v3.1 is preserved, as well as additional features that make running MiniHelp Plus v3.2 less time consuming. Still, running MiniHelp Plus v3.2 is as easy as clicking the mouse on a button, and what's even better you can view **all** the files generated or used by MiniHelp Plus v3.2 without ever having to look at file manager or another command line. MiniHelp Plus v3.2 also sports the new, popular **"3-D"** look many users like by linking with Microsoft's CTL3D.DLL. Versions previous to MiniHelp Plus v3.1 had to be ran from Windows File Manager, from a configurable text editor that allows you to run programs from a menu or button bar, or from the command line. Now, though, all you do is configure MiniHelp Plus v3.2 and click on a button on the shell. In fact, MiniHelp Plus v3.2 ***will not*** run as versions previous to and including v2.3 did (thus, you cannot drag your source file onto the MiniHelp Plus v3.2 executable file and compile your files; nor can you run it from the command line as it accepts no command line arguements; nor can you run it from a configurable text editor).

You may, however, use the text editor of your choice to create your MiniHelp Plus v3.2 source file. I chose not to design yet another text editor dedicated to producing MiniHelp Plus v3.2 source files. I felt this was too restrictive and would significantly add to the learning curve. With this approach, everyone can use the text editor they are most accustomed to, while running MiniHelp Plus v3.2 either at the bottom of the screen or as an icon. Therefore, when you are done creating and saving your source file, you can pop up MiniHelp Plus v3.2, click on a couple of buttons, and be finished with the creation of your Help file. There was a "give and take" when creating the Windows® shell, but I feel the compromise took very little away while giving a lot in return, especially in making MiniHelp Plus v3.2 **extremely easy** to run.

The default (and mandatory) filename extension used by MiniHelp Plus v3.2 is **.SRC**. You **must** save your MiniHelp Plus v3.2 source files with this file extension. Plus, using the **.SRC** filename extension makes it easier to distinguish your MiniHelp Plus v3.2 source files from other text files on your system. So first, you need to associate that file extension with the text editor that you will use to write your MiniHelp Plus v3.2 source files (the ASCII text files). Windows® Notepad works really well, despite its limitations on file sizes. To overcome this limitation, MiniHelp Plus v3.2 has a command that allows you to create Help files that are limited only by the amount of memory on your computer.

In File Manager, click once on the test file MHPSAMPL.SRC. Under the **FILE** menu, select **Associate...**. In the dialog box that appears, scroll down the list to the next to the last selection (it should say Text File (notepad.exe)). Click on that selection and click on the **OK** button. Now, any time you double-click on a file that has the **.SRC** extension, it will be loaded into Notepad. To associate the .src file extension with a different text editor than Notepad, repeat the first two steps above, click on the **Browse...** button, go to the directory containing your favorite text editor, double-click the mouse on the executable file, and click the **OK** button.

NOTE: The ERRORLOG option is automatically inserted into your Help Project File in the [OPTIONS] sections. The name of the error log will be the same as your source file but with the file extension of **.err**. This will write the errors encountered during the Help Compiler build to a separate file so you can review them using any text editor (like the one you used to create your MiniHelp Plus v3.2 source file) at your leisure. This way, you have a record of them.

***All** your source files and related files must be in the same directory as MiniHelp Plus v3.2 in order for MiniHelp Plus v3.2 to run correctly. Failure to do so risks crashing your system with a General Protection Fault (GPF).*

How To Run MiniHelp Plus v3.2

Configuring MiniHelp Plus v3.2

Configuring MiniHelp Plus v3.2 is very, very simple. Click on the **Configure...** button, and in the edit control box following **Name of Source File:**, type in the name of your source file ***without a file extension.*** MiniHelp Plus v3.2 *automatically* appends the appropriate file extensions on the filename you enter, thus your source file, the RTF file, the header file, the help project file, the errorlog file, and the resulting Help file will all be named the same, each with a different file extension. For example, if you entered "test" in the edit control box, files named test.src, test.rtf, test.hhh, test.hpj, test.err, and test.hlp will be used/generated. Don't get too bogged down by all this. Just remember that all you need to do is enter the filename ***without a file extension*** in the top edit control box. And make sure you have saved your source file with the same name and with the **.src** file extension.

NEW! Now, with MiniHelp Plus v3.2, you can use the Microsoft Help Compiler of your choice. In previous version, you were limited to **HCP.EXE**. Of course, for the same reasons you were limited to HCP.EXE in MiniHelp Plus v3.1, it is still the recommended and **default** version of the Help Compiler. (The reason is that HCP.EXE makes use of the extended memory on your computer, while the other versions do not). Now, if you so choose, you may use **HC.EXE** or **HC31.EXE** with MiniHelp Plus v3.2. In the middle edit control box, type the name of your Help Compiler ***without any path information***. The Help Compiler and its error-code file (whichever one you choose to use, ***must*** be in either the MiniHelp Plus v3.2 or Windows directories in order to be located). You won't need to mess with this if you decide to use HCP.EXE, since MiniHelp Plus v3.2 defaults to it automatically and inserts this for you.

Next, if you want to configure MiniHelp Plus v3.2 to launch your favorite text editor when you click on the **Text Editor** button, enter ***the full path and filename*** (up to 150 characters) in the bottom edit control box. If you do not specify a text editor, Windows® Notepad will be launched by default when you click on the **Text Editor** button.

NEW! MiniHelp Plus v3.2 also gives you a trio of options from which to select. The first of these is the ability to select whether or not the currently configured MiniHelp Plus v3.2 source file is automatically loaded into your default text editor when you select the **Text Editor** button. Very few of us actually complete a Help file in one day or in one setting and this selection allows us to continue working on an ongoing project. The second option is to select whether or not the HPJ, Header, and ERR files are loaded into your default text editor. If not selected, Windows® Notepad is used to view the previously mentioned files. The third option allows you to bypass the welcome/reminder message box when MiniHelp Plus v3.2 starts (sorry, unregistered users; this does **not** prevent the Registration reminder dialog from appearing; the only way to do that is to register your copy).

Click on the **OK** button to accept what you have typed into the edit contol boxes, or on the **CANCEL** button to ignore any changes you have made or to get out of the Configure dialog box. Note that the **OK** button does not work if nothing is typed into the edit control boxes.

You will need to configure MiniHelp Plus v3.2 for each *different* project on which you are working. Say you just finished creating the Help file "**MyHelp**" and now want to start working on a new project called "**BestHelp**." MiniHelp Plus stores information about the last project in its own private **.ini** file in your Windows® directory. Thus, you will have to re-configure MiniHelp Plus v3.2 to rewrite the **.ini** file (called, of course, **mhplus32.ini**), to work with the BestHelp files. You **do not** have to, however, re-configure MiniHelp Plus v3.2 for repeat processing of the same project.

Now, to create the RTF, Header, and Help Project Files, simply click on the button "**Compile RTF**". A message box will appear informing you of success or failure. Clear it by clicking on its **OK** button. Now click on the button "**Help Compiler**," and the Help Compiler will compile your Help file. When control returns to MiniHelp Plus v3.2, you may click on the **View Help** button to view your newly created Help file. You may, also, wish to click on the **Error File** button to see any errors that may have been generated during the Help Compiler's build. Make sure that the Help Compiler is either in the same directory as MiniHelp Plus v3.2 or in your Windows® directory.

To view the MiniHelp Plus v3.2 Help File and the MiniHelp Plus v3.2 ToolPad Help File, simply click on the buttons "**MiniHelp Help**" and "**ToolPad Help**." Make certain that these files, as well as your new Help file, are either in the same directory as MiniHelp Plus v3.2 or in your Windows® directory, or MiniHelp Plus v3.2 will not be able to locate them. The program's Help Files are also available from the system menu.

NEW! Clicking on the **View HPJ File** or the **View Header** buttons allows you to view or edit the contents of the respective file. There ***will*** be times when you will want to view either file, such as when you are executing a Help Macro and need the ***context number*** of a topic. In that case, you will need to reference the Header file to find out what context number MiniHelp Plus v3.2 has assigned to that topic.

NEW! Clicking on the **View RTF File** button allows you to view the RTF file generated by MiniHelp Plus v3.2. The RTF file is loaded into Windows® Write for viewing or editting. And another nice feature is that MiniHelp Plus v3.2 takes care of the ***ANNOYING*** "Convert, No Conversion, Cancel" message box that appears when you attempt to load a "non-write" document into Write. This saves you mouse movements/keystrokes by you not having to respond.

NEW! Now you can view all the bitmapped images in your Help file project by clicking on the **View Bitmaps** button. A dialog box listing all the bitmaps properly defined in your project appears. Simply select the bitmap you want to view or edit by either double-clicking on the filename listed, or by clicking once on the listed filename and clicking on the **View** button. Windows® Paintbrush will be launched with the bitmapped image you selected ready to view or edit. Click on the **Cancel** button to close the dialog box and do nothing. Note that there are restrictions. Notice above where I said *properly* defined? This means that for each bitmapped image you include in you Help file, you **must** place a reference to it in the [BITMAPS] section of the HPJ file by issuing the **.hpjbitmap** command. Otherwise, this feature ***will not work!*** Also, only filenames with the **.bmp** file extension are listed, since Paintbrush is not real friendly with other graphic formats. A nice feature of this function of MiniHelp Plus v3.2 is if you have specified a separate directory for the Help file's bitmapped image with the **BMROOT** setting in the [OPTIONS] section of the HPJ file, that directory is used to obtain the filenames to list in the dialog box, ***so be sure all the bitmaps defined in your project are in that directory.*** If you choose not to use the **BMROOT** setting in the [OPTIONS] section of the HPJ file, the current directory containing MiniHelp Plus v3.2 is used and all your bitmaps will need to be in that directory.

The **About...** button displays a dialog box that tells you about MiniHelp Plus v3.2 (also available from the system menu). The **Exit** button allows you to quit and exit MiniHelp Plus v3.2. Just in case you clicked on the **Exit** button by mistake, a message box appears asking you if you are sure you want to quit. Click on the **YES** button if you do want to quit, **NO** if you don't.

NEW! MiniHelp Plus v3.2 also sports a "**status bar**" that informs you of your most recent action in MiniHelp Plus v3.2. Personally, there have been (and I'm sure will continue to be) times when I've been in the middle of running MiniHelp Plus and have been called away from the computer for one reason or another. Upon returning to the computer, I have found myself repeating some of the last performed functions, simply because I could not remember what I had done last. Now, you have a "visual" reminder of the last task completed with MiniHelp Plus v3.2.

NEW! Also new with MiniHelp Plus v3.2 is the addition of a "**Always On Top**" function. Selectable from the system menu, if a check mark is next to it, then "**Always On Top**" is active (this is the default when MiniHelp Plus v3.2 starts up). If no check mark is present, then MiniHelp Plus v3.2 is not selected to be the topmost window on your screen. To remind you of its state, the words "**On Top**" appear in the right-most status bar window if "**Always On Top**" is selected. If it is not selected, that area of the status bar remains empty, displaying nothing.

NEW! The **title bar** of MiniHelp Plus v3.2 changes with the configuration, displaying the name of the source file currently configured after the MiniHelp Plus v3.2 title. If no source file is configured (or if you accidentally delete your MiniHelp Plus v3.2 INI file, **mhplus32.ini**) the phrase "**No File Configured**" is displayed. Now you know at a glance what project MiniHelp Plus v3.2 is currently configured for.

NEW! Now, whenever MiniHelp Plus v3.2 is the active window on your screen, you need only press the letter on the keyboard that is underlined on the button you want to use. You **do not** need to use any ALT or CTRL key combinations; only the letter underlined on the button. So, if you want to compile your MiniHelp Plus v3.2 source file, simply press the "**C**" key on your keyboard (if MiniHelp Plus v3.2 is the active window on your screen) and the source file will be compiled. Or, if you want to run the Help Compiler, simply press "**H**", and so on.

NEW! MiniHelp Plus v3.2 now supports eight fonts, instead of only the three TrueType® Fonts shipped with Windows (Arial, Courier New, and Times New Roman). Previous versions of MiniHelp Plus only supported these three fonts for maximum compatibility between all Help user's systems. Arial is still the default font, but you may now use the following fonts with MiniHelp Plus v3.2: Symbol (TrueType®), System (Proportional), Fixed System (Fixed Spacing), MS Sans Serif (Proportional, to 24 points only), and MS Serif (Proportional, to 24 points only). Of course, you may still use Courier New and Times New Roman. And the addition of these fonts still preserves maximum compatibility since all these fonts come with Windows®.

What Are The Files Generated by MiniHelp Plus v3.2?

MiniHelp Plus v3.2 generates three different files and each Help file project will end up with no less than seven (7) different files. Of course, the **.src** file is your MiniHelp Plus v3.2 source file from which all the others are derived.

The Header file **(.hhh)** lists all the topics in the Help file you are building and assigns a unique number to each topic. These numbers are called **context id** numbers. This is used, primarily, with many of the Help macros, some of which cannot be executed without the context id numbers.

The Help Project File **(.hpj)** is a scripted, ASCII text file that tells the Help Compiler what files to include when building the Help file and what options you want to use. In a way, it is the **INI** file for the Help Compiler, telling it what functions to perform and how to do it.

The **.rtf** file is a specially formatted text file. In a way, RTF is kind of like a mini-language whose specialty is text formatting. This is the type of file required by the Help Compiler to build the contents of your topics. It tells the Help Compiler such information as which words to boldface or place in italics, margins, defines non-scrolling regions at the top of the screen, where to place bitmapped images, and so on.

The Error File **(.err)** is a file generated by the Help Compiler listing all the errors it encounters while compiling your Help file. If you have forgotten to assign a title to a topic or if you have not properly executed a Help macro, the information is written to this file so you can view it at your leisure. If the only thing listed is the name of the HPJ file used to build the Help file, then no errors were encountered.

Of course, the Help file **(.hlp)** is what you are ultimately attempting to produce and, please, do realize that Help files **can** be viewed as stand-alone programs/files. There are many uses for Windows® Help files...suffice it to say that anything worthy of putting down on paper or anything that is frequently referenced is an A-1 candidate for inclusion in a Windows® Help file. Help files may be installed in the program groups of Program Manager and accessed by any user in the same way they would access any other program (i.e., by double clicking on its icon).

The MiniHelp Plus v3.2 initialization file **(mhplus32.ini)** is the file that contains, among other things, the current configuration for MiniHelp Plus v3.2. It stores the names of the various files used by MiniHelp Plus v3.2, configuration information (options, the current project, the path to your default text editor, the name of the Help Compiler you are using, etc.). This file is stored in the same directory as the MiniHelp Plus v3.2 program file (mhplus32.exe).

If you have ***properly*** included bitmapped image files in your Help file project, you will have an eighth file, called **mhp32bmp.dat**. This file is nothing more than a private "**.ini**" file used by MiniHelp Plus v3.2 to list all the bitmaps in the **View Bitmaps** dialog box. This file is ***automatically*** deleted by MiniHelp Plus v3.2 whenever you change your configuration, and does not exist until the MiniHelp Plus v3.2 source file is compiled.

Topic Commands

The **topic commands** are the heart of MiniHelp Plus v3.2. There are 19 topic commands, and they are integral to the construction of a quality, well-written Help file. All of the commands are called **dot** commands. That is, each command is preceded by a "period" ("dot"). **Note that all topic and project commands are separated from what they are defining by a space (that means there must be a space after the command). Also, a space should not precede the command. Each topic and project command <u>must</u> appear on a line of all its own.** Also, where there is to be a space, I will type an underscore. You should **not** type the underscore when issuing any of the commands. Here is a list of the topic commands, listed in the order in which they would most likely appear in a MiniHelp Plus v3.2 source file. A brief description of each will follow, one by one. It will probably be best if you could print out the sample MiniHelp Plus v3.2 source file to see some actual examples, in use. The file is MHPSAMPL.SRC.

.topic	defines the topic name
.tmacro	defines a macro be executed when entering topic
.title	defines the title of a topic
.keyword *(note changes)*	defines a keyword the topic can be searched for
.browse	defines the browse sequence thru the Help file
.link	creates a link to another topic
.pop *(enhanced)*	creates a link that appears in a popup box
.bmlink	creates a link from a bitmap image
.macrolink *(enhanced)*	creates a link that executes Help macros
.fontsize	defines the size of the text on the screen, in points
.bold	defines boldface text
.italic	defines italic text
.boldital	defines boldface italic text
.ul, .ulbold, .ulitalic, .ulboldital	defines underlined text
.bitmap	defines a bitmap image as a character in text
.rbitmap	defines a bitmap to place at right margin
.lbitmap	defines a bitmap to place at left margin

.cfore	defines the color of the foreground (text color)
.include	defines other files to include when compiling

.link

The **.link** command establishes a "jump" to another topic & *page* when the user clicks on the text highlighted in green on the screen. (Green is the default color of the link in Windows Help, unless you or someone else has changed the setting in the win.ini file).

The format for the command is as follows:

.link_*topicname_"text to display on the screen"*

EXAMPLE:

.link TOC "Table of Contents"

Must appear after a .topic command.

HINTS & TIPS

- I recommend that you type in your topic names in all capital letters. It won't make any difference to MiniHelp Plus v3.2 or the Help Compiler, but it will make them easier to identify when you're reading the source file.

- Keep a list of topic names you have defined in the links written down somewhere so you can refer to them and their proper spelling without having to search back through the source file to remember what you called something. Believe it or not, this can really help you organize your Help file. It is difficult to remember all the topic names you are linking with, particularly when you are defining several links in a row. Also, this will help save you time. The list can also be used when defining your browse sequencing.

- Be sure the text you want to appear in green on the screen (the link text) appears in double quotes, just as in the example above.

- If you want to create a "hidden" link (one where the text doesn't change color and it is not underlined), precede the *topicname* with a percent sign (%). For example: .link %TOC "Table of Contents" would not appear in green or be underlined. Be aware, however, that the link will still be highlighted if the user selects the link with the TAB key.

· If you want to create a link that is underlined but **not** change color, precede the *topicname*
with an asterick (*). For example: .link *TOC "Table of Contents" would be underlined
but would not change color to the link text color (green).

.pop

.pop MYLINK "This is my link" **.pop**_*topicname*_*"text to display on screen"*

Must appear after a .topic command.

The **.pop** command works just like the link command, except that when the user clicks on the words highlighted in green, instead of jumping to another topic *page*, the associated topic is displayed in a popup window. The information remains visible until the user clicks on the left mouse button again. You can easily distinguish a popup link from a regular link: a regular link has a solid underline, while the popup link has a dashed-underline.

You may also define a bitmap as a popup link. Simply replace the text you want to appear on the screen with the name of a bitmapped image file. It **must** end with **.bmp** in order to be formatted as a bitmapped image popup link. Otherwise, only text will appear in its place.

Popup links are very useful for providing definitions of terminology the user may be unfamiliar with, or for providing further explanations of a particular point you want to make in your Help document. One good reason for using popup links for short subject matters instead of a regular link, is that it doesn't take the Help file reader away from the current topic.

There is, however, a limitation with popup links: the amount of data displayed should not take up more than 2/3 of a screen. Popup windows have no scroll bars and what does not fit within the confines of your monitor's screen is simply lost to the reader.

All the **Hints & Tips** that apply to links also apply to the popup links and the format is identical.

.bmlink

.bmlink TOC "disk.bmp" **.bmlink**_*topicname*_*"name of bitmap file"*

Must appear after a .topic command.

The **.bmlink** command works exactly like the .link command, except that instead of displaying text in green on the screen, a bitmap image is displayed as the link, or hot spot. When the user clicks the mouse on the bitmap image, the reader is linked with the associated topic &

page.

Also, the format, while almost the same, is a little different. Instead of placing text inside the quotes to display on the screen, you specify the bitmap image's filename and the **.bmp** file extension. The **Hints & Tips** for the .link command apply here also, except for changing the way the link text appears on the screen (since there is no link text).

.macrolink

.macrolink ExecProgram(`clock.exe', 0) "Run Clock"

.macrolink_*macroname*_*"text to display"*

Must appear after a .topic command.

The **.macrolink** command creates a link (hot spot) on the screen that, when the Help file reader clicks on it, executes the specified Help macro. And, just as with the **.pop** command, you may specify either text to display on the screen or a bitmapped image. To specify a bitmapped image, the text you type in quotes **must** end with **.bmp** or text will appear. For a more detailed explanation, please refer to the MiniHelp Plus v3.2 online Help file, where I have provided three examples. I have also listed 10 or so of the more common Help macros there, what they do, and their syntax. Help macros are some powerhouse tools that you should really learn how to use. There are 56 Help macros in all. The accompanying Help file for MiniHelp Plus v3.2 makes extensive use of several Help macros to increase the ease of use and functionality of the Help file. In fact, you can even access this User's Manual from the **Quick MiniHelp** menu. If you would like to learn more about how to effectively utilize the power of the Help macros, try to get your hands on the Microsoft Help Writers Authoring Guide (a Help file itself, available in free distribution on bulletin boards around the country) or Microsoft Windows 3.1: Programmer's Reference, Volume 4, Resources (Chapter 15 of this book covers all the macros in detail).

.topic

.topic TOC **.topic**_*topicname*

The **.topic** command is the first command issued when you are defining the topic you are jumping to from the associated hot spot. This means, the text that is displayed when a Help file reader selects a link (hot spot). ***Everything in a Help file is associated with the*** **.topic** ***command!*** If it does not appear in a topic, Help simply does not display it (because it does not know where to display it). The **.topic** command is the first *topic command* issued in any MiniHelp Plus v3.2 source file. Only the *project commands* may come before it.

NO spaces are allowed in the *topicname.* Make sure the topic name here is ***identical*** to the topic name you specified in the associated link. Here, spelling counts very much. Be off by as much as one letter or transpose a letter, and the Help Compiler will generate an error since it will not be able to find the associated link.

.title

.title My Table of Contents **.title**_*title for topic*

Must appear after the .topic command.

The **.title** command gives a title to a topic. Help expects every topic to have a title. Think of it as if the .topic command is a magazine article and the .title command is the title of that article. If you were to select the **History** button on the button bar in Help, the title is displayed there so you can reference what topics you have read and in what order. The title is also used in the Search window to direct Help file readers to a particular topic that is associated with a particular keyword (see the .keyword command below).

Spaces ARE allowed in the .title command. Your title can be as long or short as you wish (shorter is better), and even though Help expects every topic to have a title, they are optional. I strongly recommend them for a professional, well-thought out Help file. They orientate the Help file reader to what they have read (via the History dialog window) and direct the reader to particular topics in the Search window. Also, make it a habit of making the second command the .title command, right after declaring a new topic with the .topic command (except when using the .tmacro command, when you would make the .title command the third command you issue). Otherwise, the Help Compiler may generate a warning/error, declaring that there is a topic without a title.

.keyword *(note the format change)*

.keyword Bill Gates**;**Microsoft **.keyword_***search words or phrases*

Must appear after the .topic command.

The **.keyword** command is where you associate search words and phrases with a particular topic, and those *keywords* appear in the Search window when a Help file reader selects the **Search** button on the button bar at the top of the Help window. Be generous with keywords. Try to think ahead of how your potential audience might associate items or topics and try to associate those keywords with those topics. Try to think of how many ways you might want to look something up. A well written, well thought out Help file anticipates how the Help file reader will want to look something up and covers those bases with appropriate keywords or keyword phrases to cross reference related topics. For example, in the accompanying Help file, the commands **.link, .pop, .bmlink,** and **.macrolink** all have the keyword phrase *Creating a Hot Spot* associated with them. If the reader clicks on this phrase in the Search window, all four of these topics are displayed for him/her to choose from, thus creating a cross-reference or cross-index between these four topics.

Be cautious: you must not enter the command with an **'s'** on the end; it will be ignored if you do. Also, you **must separate** each additional keyword word or keyword phrase with a semicolon (**;**). You may use up to 255 characters after the keyword command (including the **;**'s). You may, also, issue as many .keyword commands in a row as you see fit.

Also, keywords referenced in the middle of a topic's text will take the reader to that portion of the text when selected from the list in the **Search** dialog box. They are declared the same way you would define a keyword at the beginning of a topic. Just make sure the **.keyword** command is issued on a line by itself. This is called a spot-referenced keyword. This allows the reader to jump only to that portion of the topic instead of having to start from the beginning of the topic and scroll down through all the other stuff to get to what he/she wants to read.

.browse

.browse 004	**.browse**_*unique browse number*
.browse TOC:010	**.browse**_*topicname:unique browse number*

Must appear after the .topic command.

The **.browse** command defines the sequence in which the topics appear within a Help file when the Help file reader clicks on the browse buttons (<< or >>). Although the browse buttons are an option you must define in the Help Project file, many Help writers include them because they allow the Help file reader "browse" through the topics in an organized and logical manner. I **always** include browse sequencing in the Help files I write for this reason. It helps direct the Help file reader along the path ***I*** have set up for him/her.

Notice that there are two acceptable formats for the .browse command. Either one is acceptable, but whichever one you choose, be consistent throughout the Help file. MiniHelp Plus v3.2 doesn't care which one you use, but if you start mixing styles, you may not end up with the results you had expected.

In the first example, you simply assign a unique browse sequence number to topics in the order in which you want them to appear when the user clicks on the browse buttons.

In the second example, you assign a topic and a unique browse sequence number to the topics in the order in which you want them to appear. Be sure to separate the topic name (must be a valid name of a topic) and the browse sequence number with a colon (:). This method has some definite advantages. For example, in the MiniHelp Plus v3.2 Help file, all the major sub-headings, or chapters, are assigned to a browse sequence that is associated with the TOC (Table

Of Contents) topic. That way, the Help file reader can use the browse buttons to rapidly access the different chapters of the Help file. Then, each sub-heading, or chapter, has its own browse sequence assigned to it. Thus, when a Help file reader clicks on one of the links in one of the sub-headings, he/she can use the browse buttons to access the rest of the links in the order I have assigned. Having done it this way, the Help file reader doesn't have to browse through all of the topic commands, all the project commands, and all the RTF formatting commands just to get to a section that describes the [WINDOWS] section of the Help Project file (which is, by the way, at the end of the Help Project File section of the Help file). If you have read many Help files, I'm sure you appreciated the thoughtfulness of the Help writer who chose to include browse buttons. They really can help "walk a reader through" the process of learning about something.

One other thing you should know about defining browse sequences: note the leading zeros in the browse number declaration. They are very important to preserve the order you want. First, decide how many digits you want to use in your declaration (three will work well for most Help files, but four may be needed if you intend to make a REALLY large Help file). **Be consistent!** Here is why the leading zeros are very important: when the Help Compiler builds the Help file, it is **not** using a numerical sort routine on the numbers in the browse sequencing; instead it is using an ASCII sort routine. Because of this, if you have a browse sequence declared as 99 and the next one is 100, the one declared as 100 will appear long before 99, since 1 appears in the ASCII table before 9 does. Thus, if you type 099, you insure that 100 does not appear before 99.

.tmacro

.tmacro CopyTopic()   **.tmacro_***macroname*

Must appear after the .topic command.

The **.tmacro** command, when used, should be the second command issued, right after the .topic command. In the cases where you use this command, you would then make the .title command the third command issued when defining a topic. This command executes one of the 52 Help macros when a topic is entered. For more information, see the MiniHelp Plus v3.2 Help file for a list of 10 of the most commonly used macros. (HINT: I have sprung another **new** macro on you here that I didn't mention to you anywhere before; the macro in the example above copies the entire topic to the clipboard when the topic is entered. It may also be used from the .macrolink command, described earlier.)

.fontsize

.fontsize 10   **.fontsize 24**   **.fontsize_***size of font in points*

Must appear after the .topic command.

The **.fontsize** command sets the size of the font displayed on the Help screen. Generally, you should limit your fontsizes to a maximum of 24 points in Windows Help. The command remains in effect until changed with the issuance of another **.fontsize** command. I think that pretty well explains it. Pretty simple, really. Just don't forget the space after the command.

.bold

This	
.bold text	**.bold**_*text you want in boldface*
will appear in boldface type	

Must appear after the .topic command.

The **.bold** command defines what text on your screen appears in bold face print. The MiniHelp Plus v3.2 online Help file gives a good example of the typeface commands. There is no limit, to my knowledge, of how much text can appear after the **.bold** command. The command must appear, as with all other dot commands, on a line of its own. Thus, if you are typing along and decide that you want some text to appear in boldface type, hit Enter, type the command (followed by a space), type the text you want to appear in boldface, hit Enter again, and continue typing the surrounding text you do not want bold face. The command is in effect until you hit the Enter key after the text you want to appear in boldface type. The example above illustrates this well, I think. In the example, the word *text* will appear in your Help file in boldface type.

.italic, .boldital, .ul, .ulbold, .ulitalic, .ulboldital

.italic_*text to appear in italics*
.boldital_*text to appear in bold italics*
.ul_*text to appear underlined*
.ulbold_*text to appear bold and underlined*
.ulitalic_*text to appear italic and underlined*
.ulboldital_*text to appear bold italic and underlined.*

Must appear after the .topic command.

The **.italic, .boldital, .ul, .ulbold, .ulitalic,** and **.ulboldital** commands are used exactly like the **.bold** command. Everything that applies to the **.bold** command applies equally to these commands. See the example above for the **.bold** command and substitute either the **.italic, .boldital, .ulbold, .ulitalic,** or **.ulboldital** commands where it says **.bold**.

.bitmap

.bitmap disk.bmp **.bitmap**_*filename of bitmap image*

This bitmap image appears
.bitmap disk.bmp
in the middle of the text.

Must appear after the .topic command.

The **.bitmap** command inserts a bitmap image in your line of text, *just as if it were another character in the line of text*. This is an important point to make, as this is what separates this comand from the **.rbitmap** or **.lbitmap** commands. There are many uses for this command. One use is to help break up the page by inserting bitmap images at strategic points in your text. Another is like when you are reading along in a children's book and instead of the word bird printed in a sentence, you see instead a *picture* of a bird. Another (and important) use is if you are using decorative style fonts in your Help file that not everyone has on their computer system.

Say, for example, you want to include an Olde English style typefaced "W" as the first letter of your first paragraph. If you go into Paintbrush, size your picture properly with the Image Attributes command, type the letter the way you want it to appear (bold, italic, color) and save it as a bitmap image, you can insert it into your Help file and ***anyone*** who reads your Help file will see the Olde English style "W" just as you had intended for them to, whether they have that font on their system or not. The Help Compiler will include the information about the bitmap in the resultant Help file, so it becomes a part of the Help file.

Notice that this command is issued much like the .bold, .italic, .boldital, and .ul commands, except that instead of text after the command, you have the name of the bitmap image file. Also notice that, unlike the .bmlink command, the filename is **not** set off in quotes. This is not necessary since there is no link to another topic. Instead, it's just a bitmap image that sits there and looks good. Also, since the bitmap is treated like a character, all the RTF Formatting Commands for formatting your text apply to the bitmap image. For more information about the RTF Formatting commands, refer to the MiniHelp Plus v3.2 online Help file.

In the example above, the bitmap image (disk.bmp) appears in the middle of the text. Simply place the bitmap image in the text where you want it to appear. If you want the bitmap to appear at the start of the text, issue the **.bitmap** command first, before the text is typed. If you want the bitmap to appear at the end of the text, place the command after the text is typed.

.rbitmap, .lbitmap

.rbitmap disk.bmp	**.rbitmap**_*filename of bitmap image*
.lbitmap disk.bmp	**.lbitmap**_*filename of bitmap image*

Must appear after the .topic command.

The **.rbitmap** and **.lbitmap** commands are different from the regular **.bitmap** command in that instead of treating the bitmap image as a character, the bitmap image is placed at the right margin (**.rbitmap**) or the left margin (**.lbitmap**) of the Help screen and the text that *follows* the command is wrapped around the bitmap images. In the case of **.rbitmap**, the text is wrapped around the left edge of the bitmap image. In the case of **.lbitmap**, the text is wrapped around the right edge of the bitmap image. Also, the RTF Formatting Commands for formatting your text have no effect on the bitmap image (outside of the left indent, right indent, and first indent commands).

.cfore

.cfore 07	**.cfore**_*number of color to display normal text*

Must appear after the .topic command.

The **.cfore** command changes the color of the normal text (not the hot spot, or link, text). Windows Help supports 16 colors (defined below). Simply issue the command and any text that appears after the command is displayed in the color specified and remains in effect until it is cancelled out or changed with another **.cfore** command. (**.cfore** stands for color, foreground).

The color codes (as defined by the color table in the MiniHelp Plus v3.2 RTF file) are:

Black = 01	Dk Blue = 09
Dk Gray = 02	Yellow = 10
Lt Gray = 03	Dk Yellow = 11
Lt Red = 04	Lt Cyan = 12
Dk Red = 05	Dk Cyan = 13
Lt Green = 06	Magenta = 14
Dk Green = 07	Purple = 15
Lt Blue = 08	White = 16

If Help encounters an invalid color number (one outside this range of 16 colors), it ignores the command and displays the default color (black).

.include

.include myfile.src **.include**_*file to include in MiniHelp output*

The **.include** command is a special command that helps you overcome the filesize limitations of Windows Notepad or many of the other text editors that are available and that you may be using. If you find that you are approaching the filesize limit of the text editor you are using (Notepad is about 27K with word wrap turned on, about 37K with word wrap turned off), issue the **.include** command followed by the filename of an additional source file for MiniHelp Plus v3.2 to include when it creates the RTF file. Save this source file and open up another one with the filename you specified in the **.include** command. Now you can continue your typing. What MiniHelp Plus v3.2 does when it encounters this command is close the first source file and open the source file you specified in the **.include** command. Therefore, the **.include** command ***must be the very last command in your source file!*** Any text that appears after this command, **will not** be included in the RTF file output by MiniHelp Plus v3.2.

There are no limitations to how many source files you can link together with the **.include** command other than the amount of memory you have on your computer. It is best, however, to conclude the topic you are in. If you are unable to, go to the top of the **.topic** you are working on and CUT the text from that point to the end of the file to the clipboard. Issue the **.include** command and save the source file. Open another file in the text editor, and paste the clipboard contents into the new file.

The best thing about how this command works is that instead of having MiniHelp Plus v3.2 compile four different source files into four separate RTF files, MiniHelp Plus v3.2 compiles the four different files into only *one* long RTF file.

Project Commands

The **project commands** assist you in tailoring the Help Project file (HPJ) that MiniHelp Plus v3.2 generates for you. Although these command are optional (because the Help Project file generated by MiniHelp Plus v3.2 contains all the basics needed by the Help Compiler to compile your Help file), adding to or tailoring the Help Project file generated by MiniHelp Plus v3.2 can greatly enhance the resultant Help file by adding buttons to the button bar, adding your copyright to the About Help... window, adding menus to the Help file, compressing the Help file so it takes up less room on your hard drive, writing out the errors encountered by the Help Compiler to a file for later viewing, and so on and so on. For a complete discussion of the the different settings you can set in the Help Project file, see the MiniHelp Plus v3.2 online Help file. Here, we will deal only with the **project commands** and how they are used. Note that the **project commands** are **dot** commands, just like the **topic commands.** Thus, all the rules that apply to the use of **topic commands** apply equally to the **project commands**, except that they can appear **anywhere** in the source file (before or after a **.topic** command). They must, however, just as with the **topic commands**, appear on a line by themselves. Here is a list of the project commands. A brief description of each will follow, one by one.

.hpjopt	defines items for the HPJ Options section
.hpjfiles	defines files for the HPJ Files section
.hpjmap	defines context numbers in the HPJ Map section
.hpjbtag	defines buildtags for the HPJ Buildtag section
.hpjconfig	defines items for the HPJ Config section
.hpjbitmap	defines bitmaps for the HPJ Bitmap section
.hpjalias	defines topic aliases for the HPJ Alias section
.hpjwin	sets window attributes for HPJ Windows section
.hpjbaggage	defines files for HCP to store internally

The default file extension used by the Help Compiler for the Help Project file (and generated by MiniHelp Plus v3.2) is **.HPJ**.

.hpjopt

.hpjopt compress = high **.hpjopt**_*option item*_=_*parameter*

The **.hpjopt** command is used to set any of the 18 option items in the [OPTIONS] section of the HPJ. The proper format is to issue the option item name, space, equal sign, space, and the setting or parameter that option item is to control (just like the example above). For a complete listing of the option items that can be set in the [OPTIONS] section of the HPJ file and what they do, see the MiniHelp Plus v3.2 online Help file.

Briefly, some of the settings can compress the Help file so it takes up less space (above), display the progress of the Help Compiler on the screen, determine what errors you are notified about by the Help Compiler (*automatically inserted by MiniHelp Plus v3.2)*, force the use particular fonts in the Help file, specify the root directories for the topic files and bitmap image files, display your copyright, specify an icon to display when the Help file is minimized, and much more.

.hpjfiles

.hpjfiles myhelp.rtf **.hpjfiles**_*filename of RTF files*

The **.hpjfiles** command is used to list the RTF files the Help Compiler is to use in building your Help file. MiniHelp Plus v3.2 *automatically* inserts the filename of the RTF file it generates into the [FILES] section of the HPJ file. You may use this command to list RTF files **not** generated by MiniHelp Plus v3.2 for the Help Compiler to include in building your Help file.

.hpjmap

.hpjmap #include<myhelp.hhh>	**.hpjmap**_*header file for context sensitive Help*
OR	***OR***
.hpjmap TOC 1	**.hpjmap**_*topicname_context number*

The **.hpjmap** command allows you to "map out" or specify context sensitive Help. Defining context sensitive Help is useful for a couple of different reasons. First of all, if you are a programmer and wish to program context sensitive Help into your application, you must call WinHelp and be able to pass the context number of the Help topic to WinHelp so it can find the proper topic. Secondly, and this applies to **anyone** who writes Help files, defining context sensitive Help allows better and more thorough use of the 52 Help macros, because some of the Help macros can only be used with context numbers.

With MiniHelp Plus v3.2, a C-type header file is automatically generated when it converts your source file into RTF format. The filename will be the same as your source file, but with the extension **.HHH.** MiniHelp Plus v3.2 *automatically* inserts a reference to this header file into the [MAP] section of the HPJ file, using the #**include** statement, just as in the first example above. The contents of the header file associates and defines each topic name with a unique context-id number. This is, by far, the easiest way to define context sensitive Help--simply let MiniHelp Plus v3.2 do it for you by default.

The **.hpjmap** command, therefore, is here to allow you to define context sensitive Help in RTF files that were not generated with MiniHelp Plus v3.2, either by creating your own header file or defining each topic name individually with its own, unique context-id number. For more complete information, see the MiniHelp Plus v3.2 online Help file.

.hpjbtag

.hpjbtag_*build-tag data*

The **.hpjbtag** command specifies data to include in the [BUILDTAGS] section of the HPJ file. This data is used to define which topics, defined with a particular "buildtag," are included in the construction of the Help file. For more information, refer to either the Help file, Microsoft Help Writers Authoring Guide, or the book, Windows 3.1: Programmers Reference, Volume 4, Resources.

.hpjconfig

.hpjconfig BrowseButtons() **.hpjconfig**_*Help macro to execute*

The **.hpjconfig** command specifies Help macros to execute when the Help file is started and places them in the [CONFIG] section of the HPJ file. Some of the common macros declared in this section are ones that create buttons on the button bar using the CreateButton macro, enabling the browse buttons (see the example above), or using the InsertMenu/InsertItem macro to insert menus/menu items in the Help menu bar. Refer to the MiniHelp Plus v3.2 online Help file for further discussion of some of the Help macros you can include, or for a more complete discussion, refer to either the Help file, Microsoft Help Writers Authoring Guide (available on many BBS's across the country), or the book, Windows 3.1: Programmers Reference, Volume 4, Resources (chapter 15 covers the Help macros thoroughly).

.hpjbitmap

.hpjbitmap disk.bmp **.hpjbitmap**_*filename of bitmap image*

The **.hpjbitmap** command lists the bitmap images the Help Compiler should include in the Help file. You may specify directory path information if you like, but I feel it is better to have all your bitmap image files in one directory and specify that path in the BMROOT option item in the [OPTIONS] section of the HPJ file. This way, all you have to specify is the filename of the bitmap image (as in the example above). I recommend that you issue this command whenever you *first* use a particular bitmap image (it is not necessary to list the same bitmap image every time you use it). For more complete information, see the MiniHelp Plus v3.2 online Help file.

.hpjalias

.hpjalias LEARN = DISCOVER **.hpjalias**_*topicname*_=_*alias*

The **.hpjalias** command places information about topic aliases in the [ALIAS] section of the HPJ file. In the example, this would make DISCOVER a second topic name to the topic LEARN, and the topic could be referenced by either name, DISCOVER or LEARN. For a more complete discussion of this, see the references quoted above in **.hpjconfig**.

.hpjwin

.hpjwin main = , , 1, , (192, 192, 192), 0
.hpjwin_*windowname_=_"caption",_(hpos,_vpos,_width,_height),_sizing,_(RGB client),_(RGB nonscroll),_ontop_state*

The **.hpjwin** command tells the Help Compiler the name, title, size, location, and colors of the Help window and specifies this information in the [WINDOWS] section of the HPJ file. This may be a secondary window you define, or if the parameter *windowname* is ***main***, then the attributes are applied to the main Help window. The *windowname* may be any name you choose, up to 8 characters, as long as that name is not 'main.' For a more complete discussion of how to define Help window attributes, see the MiniHelp Plus v3.2 online Help file or the references quoted previously in **.hpjconfig**.

.hpjbaggage

.hpjbaggage_*filename (path information also, if not specified in ROOT).*

The **.hpjbaggage** command tells Help to store the listed files in its internal file storage system, and not on the DOS system. This allows for faster access time when dealing with multimedia data.

Miscellaneous

To insert a comment in you source MiniHelp Plus v3.2 file, start the line with a semi-colon (;). When MiniHelp Plus v3.2 encounters such a line, it ignores it and goes on to the next line.

To insert a comment in your HPJ file, start the line with a semi-colon (;), just as you would in MiniHelp Plus v3.2. When the Help Compiler encounters such a line, it ignores it and goes on to the next line.

There are only two known bugs in this program. When running MiniHelp Plus v3.2 (or earlier versions), you **MUST** make sure that the files you are attempting to compile are in the same directory as the MiniHelp Plus v3.2 program (**mhplus32.exe**), especially if you are linking files together with the **.include** command. ***DO NOT*** attempt to force MiniHelp Plus v3.2 (or earlier versions) to process files from other directories. If you do, you will generate a General Protection Fault (a.k.a. GPF, UAE or whatever else you want to call it) and Windows may crash.

The other bug deals with the **.macrolink** command. Due to the way the name of the macro is parsed from the text you want to display on the screen, if the macro has spaces as part of its parameters, **.macrolink** will not correctly format the RTF file. Instead, all the text after the space, and up to the first quote, is displayed as the hot spot on the screen. The text you wanted to display just seems to go off to who knows where; it's just lost. I am working on this bug and will let you know when I get it worked out.

Hints & Tips

**** Sample your output periodically. That is, go ahead and compile your uncompleted file, from time to time, and run the Help Compiler on it to see what problems you may or may not have in your formatting or such. It only takes a few minutes, during which you can relax a bit. Also, it is much easier to make notes on a few things to change than trying to remember a whole google of errors.

**** Get into the habit of cancelling out your formatting commands when they are no longer needed by either issuing the \par \pard commands or \fi000 \li000 commands. This will help greatly in preserving the format you desire and cut down on the number of errors you will have to go back and fix. Also, the \pard command does not affect the fontsize command. If your resultant Help file is not what you had expected, you may have forgotten to change the fontsize at a place in the source file you had planned.

**** Also, if you are getting unexpected results in your formatting, go back and check your RTF commands. If you have limited them to only one or two to a line, this will be easy. Make sure they are declared with the \ and not the /. The former is the backslash. The latter is the slash, found on the "?" key. All RTF Commands must use the backslash character.

**** To print out a backslash in your Help file, you **must** enter it twice (i.e. \\ but not \).

**** ***NEW!*** To print out a curly brace, you **must** precede it with a backslash (i.e. \{ but not {).

**** Learn to use Recorder that comes with Windows and create Recorder Macros that can be executed with just one keystroke (i.e., any function key like F2; avoid using F1 since many programs use it for its Help file). I have found that these can be very useful for many different repetitive tasks. I strongly urge you to learn how to use the Recorder Macros. You will find they can save you time and keystrokes/mouse movements.

**** Also check out the **Hints & Tips** section of the MiniHelp Plus v3.2 Help file for even more tips you can employ while creating your Help files.

Registration Form

You are free to evaluate MiniHelp Plus v3.2 for thirty days. If you find it useful and decide to keep it, I ask you to please register it (it is, after all, shareware, not freeware). If you do not wish to register, or find it of little use to you, you are obligated to remove it from your computer system.

Registration is only **$25.00.** MiniHelp Plus v3.2 has not been disabled in any way, but does have a nag screen that appears only when you first start MiniHelp Plus v3.2. It will no longer appears once you register. Considering the current high prices of many word processors capable of producing Help files, as well as the prices of some of the other shareware Help Writer's Authoring Utilities, MiniHelp Plus v3.2 is a considerable bargain. Furthermore, all registered users will receive a free utility disk of Dynamic Link Libraries, bitmaps and hot spot graphics that will add to the functionality, appearance, and usefulness of your Help files.

Simply fill out the registration form below and mail it to the address below, with cash, check or money order in the amount of $25.00.

Name:____

Address:____

City/State/Zip:____

Program Name & Version:_________________________________

Disk size/density (for free upgrade):____________________________

Where did you obtain your copy?_______________________________

If from America Online, please include your screen name.

How would you rate this program as meeting your needs? 1 2 3 4 5 6 7 8 9 10

(1 = horrible, 5 = average, 10 = perfectly met your needs)

Suggestions or comments:_________________________________

____

____

Mail to:

Paul Arnote
1208 Randolph St.
Leavenworth, KS 66048

Acknowledgments

The original program, MiniHelp, appeared in the January, 1994 issue of Compute magazine. It was written by Tom Campbell, Copyright © 1994, All Rights Reserved.

MiniHelp Plus v1.0, first appeared in February, 1994. It featured 15 topic commands (vs only 5 in the original MiniHelp), adding a great deal of functionality to the program. It was posted on America Online as freeware. Copyright © 1994, Paul Arnote, All Rights Reserved.

MiniHelp Plus v2.3, March 1994. Brought the total number of topic commands to 19 and added 9 project commands. It also added automatic HPJ file generation, as well as automatic generation of a header file for context sensitive Help. Shareware: **$15.00** registration. Shipped with a sample source file, the freely distributed Microsoft Help Compiler (HCP.EXE), bitmaps to use in compiling the sample file, an extensive online Help file for MiniHelp Plus v3.2, a user's manual (which you are reading), and a registration form. Copyright © 1994, Paul Arnote, All Rights Reserved.

MiniHelp Plus v2.5, April/May 1994. **Unreleased**, except for two copies of a beta version (one of which is slated to be released in Italy, translated into Italian). Added the ability to use bitmapped images as a popup link. Also, smaller program size since I deleted several routines in the source code that were no longer being implemented or found to be no longer necessary.

MiniHelp Plus v3.1, June, 1994. Added the ability to use bitmapped images as popup links and macrolinks, reformatted the keyword command to avoid a minor bug, automatic inclusion of the errorlog option in the HPJ file, and, most importantly, a **Windows® front end shell/user interface!**

MiniHelp Plus v3.2, September, 1994. Added the following: buttons to allow you to view the HPJ, Header, RTF, and bitmapped image files, a status bar, an "Always On Top" feature, the ability to run the Help Compiler of your choice, options for determining which text editor to use when viewing the HPJ, Header, and ERR files, a "3-D" look by linking with Microsoft's® CTL3D.DLL, listing the currently configured project on the title bar, and increased the number of

fonts available for use in a Help file produced by MiniHelp Plus v3.2 from three to eight.

Enhancements are already underway and being considered for the next version of MiniHelp Plus v3.?! Your input could greatly influence the results. **All** users, registered or not, are invited to offer their comments and/or suggestions concerning MiniHelp Plus v3.2 (or previous versions). Email me on America Online (screen name **AerosolMan**), or send them, via U.S. Mail, to the address on the Registration Form.

Special thanks to my wife, Barbie, who so patiently puts up with me spending hours in front of the computer working on all the versions of MiniHelp Plus, as well as all my other programming endeavors. Also, special thanks to Brian Costner for his proofing and constructive suggestions on improving the readability and useability of this User's Manual. And many thanks to all the users of the previous versions who took to time to **a)** register their copies, and **b)** took the time to offer suggestions for improvement of previous versions. I think some of you will notice that I ***did*** listen and implemented some of your suggestions.