# The Help Project

Close

Welcome to Help Assistant 3.0. This Word for Windows 6.0 macro greatly facilitates the development of Windows Help files, and provides an easy access to most of the functions supported by the Microsoft Help Compiler.

Help Assistant requires that a **Project** consisting of the following be created:

- the Topic files written in Rich Text Format (.RTF)
- any graphics files and video files (AVI) required for your Help file
- a Help Project Configuration file (.HPJ)
- a Setup file (.INI)

The Help Compiler links the graphics files with the topic files using the information contained in the Help Project Configuration file (.HPJ) and generates a Windows compatible Help file with the .HLP extension. (See Diagram). Video for Windows files (AVI) can be displayed within the Help file; however, they are not integrated within the resulting .HLP file, and must be distributed along with the help file.

Using the Help Assistant, you may Open/Edit existing projects or Create new ones. Help Assistant automatically tracks the project you are working on, as well as its settings. The basic setup parameters required by Help Assistant are automatically recorded in the Setup file (.INI).

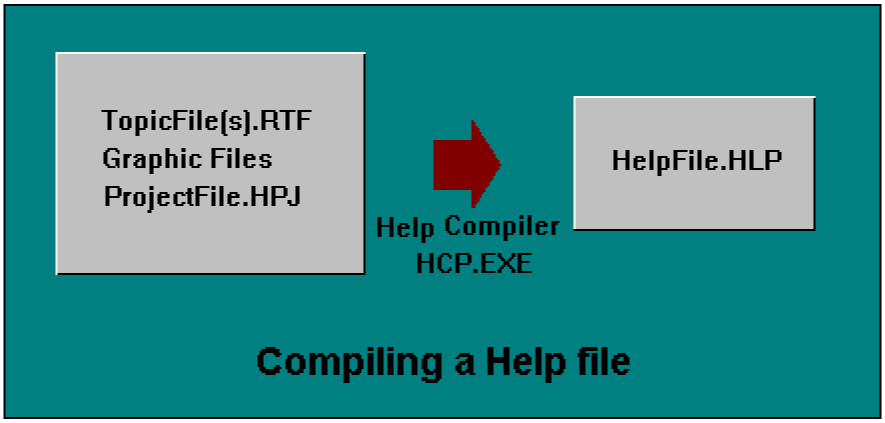
Although this is not required, it is recommended that each project be kept in a separate directory. This greatly facilitates the management of the project. When creating a new project, Help Assistant will automatically create a new directory if the specified Project Directory doesn't already exist.

To create a Windows Help file, you must first setup your project. Then you simply add one or more topic files to the project, and edit their contents, setting up Jumps and Pop-ups as required. Finally, you compile the project to generate the .HLP file.

**See Also:** Setting-up The Help project, Setting-up Topics, Creating Jumps, Creating Pop-ups

## Help Compiler

Help Assistant must use the Microsoft Help Compiler for Windows 3.1. It is recommended that the Microsoft (R) Help Compiler Extended Version; HCP.EXE be used. HCP.EXE runs as a DOS program in protected mode and generates help files for Windows 3.1.



Compiling a Help file

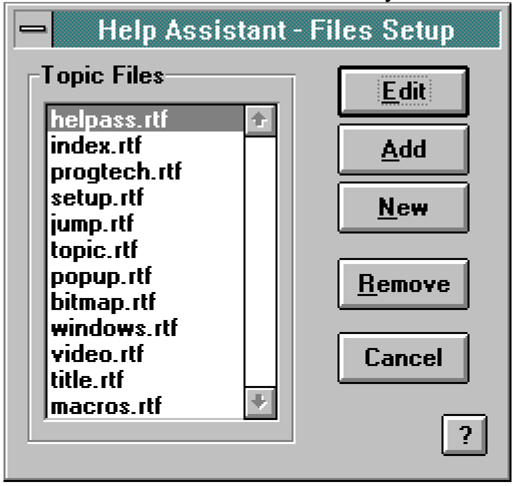
## Handling Topic Files

Close

To add or remove topic files from the project, or edit a topic file; select "HA Files Setup" from the "File" pull-down menu.   Help Assistant will display the Files Setup dialog box.

**Short Cut:** Click on the button on the Toolbar to call up the "File Setup" dialog box..

For more information click on any fields of the following Files Setup dialog box:



When Help Assistant displays the "Files Setup" dialog box, select one of the topic files displayed in the list box, and click on the desired option.

To add a new file to the project, click on the "NEW" button.   Help Assistant will create an empty **.rtf** file with the specified filename, and add it to the project. To add an existing file to the project, click on the "ADD" button.   Help Assistant displays the standard WinWord "Open" dialog box. Enter ***.rtf** to locate topic files in the selected directories, and click on the "OK" button once a file has been selected.

Topic files must be saved in the Rich Text Format (RTF).   You must therefore ensure that topic files are not overwritten in Word for Windows format during the editing process.

**Quick Save:** Click on the button to save and close an open RTF file.   Use this button carefully as Help Assistant will not prompt for a confirmation before saving the file.   This also ensure the you do not accidentally save the current file in a different format.

It is highly recommenced that the Help Assistant - Files Setup options be used at all time when working with a specific Help Project.   When creating a **New** file with Help Assistant, it is automatically added to the project's files list.   However, you may create a new RTF topic file using the Word for Windows "**File - New**" menu item using the helpass template.   If you do so, you must add these files to the project using the Help Assistant - Files Setup **Add** option.

Changes to the project file list will not be written to the HPJ project file until you save the project. Therefore, if you made any changes to the file content of the project, Help Assistant will prompt you to save the project before you can exit Help Assistant.

## Topic Files List Box

This box displays all the topic files found in the specified "Root" directory for which the extension matches the **.RTF** file format.

Click on this button to edit the selected topic file.

Click on this button to add a topic file to the project. The Open dialog box will appear. Enter *.RTF to locate Topic files in the selected directories.

Remove

Click on this button to remove the selected Topic file from the project. Help Assistant requests confirmation before deleting the file from the project directory . Answer "No" to keep the file. Answering "Yes" will permanently delete the file from the project directory.

Click on this button to create a new .rtf topic file and add it to the project.

Click on this button to abort operations.

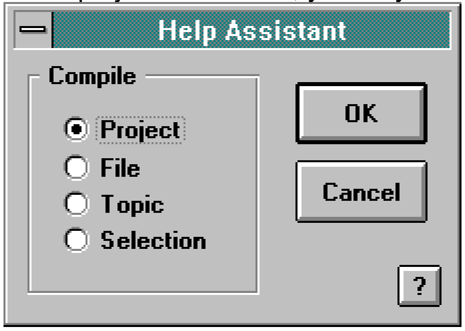
## Compiling a Help Project

Close

To compile the Help file, the Microsoft Help compiler must be installed in a directory that is in the file path.   Select "Compile" from the "Tools" pull-down menu to compile the current project.   Help Assistant can compile either the entire Project, or only the current File, Topic, or Selection.

**Short Cut:**   Click on the  button on the Toolbar.

When selecting this option the following dialog box will appear.   By default, Help Assistant compile the entire project.   However, you may want to quickly view only the current file, topic, or selection:



Help Assistant

Compile

● Project
○ File
○ Topic
○ Selection

Compiling only the current file, topic, or selection is much faster than compiling the entire project.   This is a useful feature to verify that the desired result is obtained.   However, this type of compilation does not change the help file defined by the project.   Therefore, you must compile the entire project if you want the changes to take effect.

The Compiler's error messages will be written in a file with the **.err** extension in the project directory.   When creating a new project, Help Assistant automatically set this file name to *projectname.err* if a different filename is not specified in the "Options - Setup" dialog box.   This file is a standard ASCII file that can be opened using WinWord or Notepad to view error messages and warnings.   It is by default automatically displayed upon compilation. You can disable the automatic display of the error message file in the "Options - Setup" dialog box.

When either the Medium or High Compress option is selected, Help Compiler creates a phrase-table file with the **.ph** extension in the Root directory if one does not already exist. If the compiler finds a file with the **.ph** extension, it uses that file for the current compilation. Because the **.ph** file speeds up the compression process when little text has changed since the last compilation, you might want to keep the phrase file if you compile the same Help file several times with compression. However, you will get maximum compression if you disabled the "Use Old Key Phrases" in the "Project Setup Dialog Box" before compiling.

Upon compilation, Help Assistant will automatically launch the resulting Help file.   You may disable this option for the Project compilation in the "Options - Setup" dialog box.   You cannot disable this option for the File, Topic or Selection compilation.

## Running a Help Project File

Close

The Help file can be run directly from within WinWord. Select "Run" from the "Tools" pull-down menu. Windows Help files can also be run from within the WinHelp utility or by double-clicking on the filename in the File Manager. For more information on how to run/set programs see the Windows User Manual.

**Short Cut:** Click on the 🔍 button on the Toolbar (or **Ctrl+R**).

Upon compilation, Help Assistant will automatically launch the resulting Help file. You may disable this option for the Project compilation in the "Options - Setup" dialog box. You cannot disable this option for the File, Topic or Selection compilation.

### Help Button

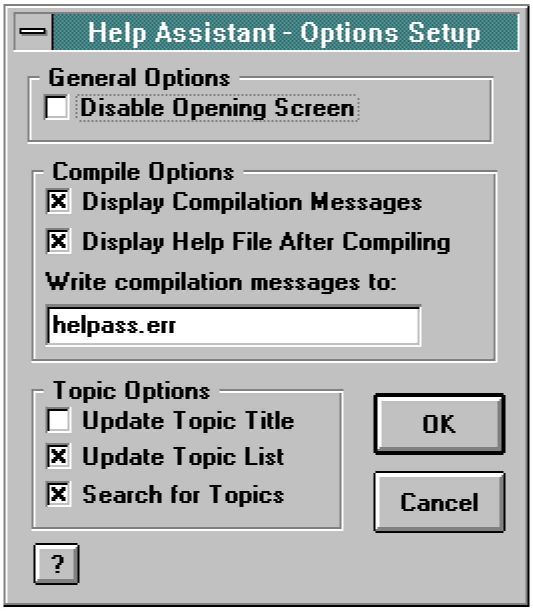
Click on this button to access context sensitive help.

## Setting-up Options

Close

It is possible to enable, or disable a number of options using the Help Assistant - Option Setup dialog box. To set-up options, select "Tools - Options" from the pull-down menu, and Help Assistant will display the following dialog box:

For more information click on any fields of the following Options Setup dialog box:



Help Assistant - Options Setup

General Options

☐ Disable Opening Screen

Compile Options

☒ Display Compilation Messages

☒ Display Help File After Compiling

Write compilation messages to:

helpass.err

Topic Options

☐ Update Topic Title

☒ Update Topic List

☒ Search for Topics

## Disable Opening Screen

Select this option to disable the display of the Opening Screen when launching Help Assistant, or creating a new topic.

## Update Topic Title

Select this option if you want Help Assistant to replace the topic title with the content of the Topic Title ($) field in the Topic Setup dialog box, and set its style to Topic Heading.

It may not be desirable to set this option when you want for instance, have a section heading appear in a non-scrolling region, and the actual topic in the scrolling region. In such case, Help Assistant would replace the section heading with the current topic heading. To avoid this problem, deselect this option.

## Search for Topic

When this option is set, Help Assistant search for existing topics in the Topic List when creating Jumps or Pop-ups. If the time required to search the topic list becomes excessive, you may want disable this option.

## Update Topic List

By default, Help Assistant writes new topics to the Topic List. You may want to disable this option if you are creating a serie of pop-up topic for instance and you don't want their titles to appear in the main topic list.   You may also disable this option each time in the Topic Setup dialog box.

## Display Compilation Messages

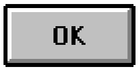
This option enable/disable the automatic display of compilation messages upon compilation.

### Display Help File After Compiling

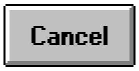
Help Assistant displays by default the newly compiled help file. You may disable this option only when Help Assistant is compiling the entire project.

## Write compilation messages to:

Specify the name of the file where the Help Compiler writes the compilation messages. By default this file is named after the Help Project filename with the **.err** extension.



Click on this button to accept the selected options.



Click on this button to abandon the changes.

## Help Project Functions

Setting-up The Help Project

Handling Topic Files

Setting-up Topic Windows

Using Help Project Macros

Setting-up Options

Compiling The Project

Running The Help Project File

## Topic Editing Functions

Creating And Setting-up Topics

Creating Jumps

Creating Pop-ups

Using Help Macros

Inserting Bitmaps

Inserting Video

## Other

The Help Project

Advanced Techniques

## Advanced Programming Techniques

Close

## Non-Scrolling Windows

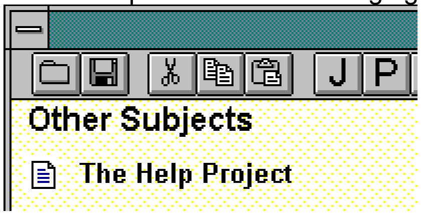
It is possible to specify a non-scrolling window region which will appear in the upper region of the Windows Help display window.   Non-scrollable windows may contain anything that is normally displayed in the Client Window (scrollable window); including hypergraphics, Jumps and Pop-ups. Its size is defined by its content, and its color specified in the Window Definition (See: Setting Up Topic Windows).

To incorporate elements (titles, bitmaps etc.) in a non-scrolling window you must set their paragraph style with the "Keep With Next" attribute.   All items to be included in this non-scrolling region must have this attribute.   These paragraphs must be at the very top of the topic.   The Help Compiler will generate an error message if a "Keep With Next" paragraph comes after a paragraph without the attribute, and will not create a non-scrolling window.

To set the "Keep With Next" paragraph attribute, position the cursor on the paragraph, select "Format"-"Paragraph" and check the "Keep With Next" check box, or change the paragraph style through the "Style..." menu option.

**Displaying a Button Bar in a Non-Scrolling Region:**

Using the non-scrolling window feature you can easily display a button bar at the top of a topic as depicted in the following figure.



This example was created this way:

*- In Window Setup, set the color for non-scrolling window to light-grey in .*
*- Using PaintBrush, created a bitmap image of the buttons.*
*- Generated a segmented graphic image using the MS Hot-Spot Editor, and created a "Hot-Spot" for each button that Jumps to a particular topic.*
*- Inserted this *.SHG image in the topic using the "Bitmap" feature of Help Assistant at the top of the topic.*
*- Set the paragraph "Keep With Next" Attribute for the reference to this bitmap (e.g. "bmc BTTNBAR.SHG" )*

## Setting-up The Help Project

Close

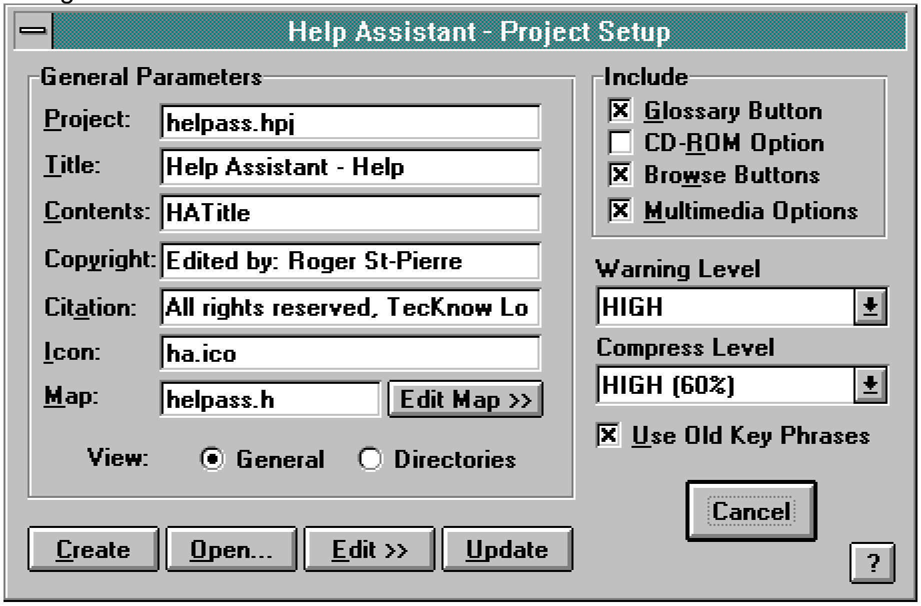
Each project requires that its characteristics be specified before it can be compiled (here the term project describes the making of a Help file). The file containing all the information required (topic files, title, options etc.) is generally referred to as the project file, and has the **.hpj** extension. Although this file can be created or edited manually, it is recommended not to since it is much easier to let Help Assistant do it for you.

When starting a new project, Help Assistant creates two files: the project file with the **.hpj** extension, and the Help Assistant setup file with the **.ini** file extension. The first file is used by the Microsoft Help Compiler while the second is used by Help Assistant to track changes to the project itself, options, settings and any other information that is required by Help Assistant itself. Throughout the Help file creation process, Help Assistant will maintain the project file for you; and unless you wish to use advance programming techniques, you will never have to edit the project file manually.

To setup a project, select "**Project...**" from the "HA" pulldown menu. Help Assistant displays the "Project Setup" dialog box from which most of the options offered by the Microsoft Help Compiler can be set.

**Short Cut:** Click the button on the Toolbar to access the "Help Assistant Project Setup" dialog box.

For more details about each feature, click on the desired elements of the "Help Assistant Project Setup" dialog box:



Help Assistant - Project Setup

General Parameters

Project: helpass.hpj

Title: Help Assistant - Help

Contents: HATitle

Copyright: Edited by: Roger St-Pierre

Citation: All rights reserved, TecKnow Lo

Icon: ha.ico

Map: helpass.h

View: ● General ○ Directories

Include

☒ Glossary Button
☐ CD-ROM Option
☒ Browse Buttons
☒ Multimedia Options

Warning Level: HIGH

Compress Level: HIGH (60%)

☒ Use Old Key Phrases

When creating a new project, all the parameters specified in the setup dialog box are automatically written to the Help Assistant setup file **.ini** and the project file **.hpj**; however, further changes will only be written to the project file **.hpj** when the project is compiled. When compiling a project that has been changed, Help Assistant requests if you wish to write the information to the project file before compiling. If you answer no, Help Assistant will not update the project file used by the Microsoft Help Compiler, and changes will not be reflected in the compiled Help file. However, the changes will be kept in the Help Assistant setup file and may be saved at a later compilation. If you do not wish to keep the changes made to a project, and that project has not been saved yet, re-open the project and answer **no** when requested if you wish to use the current **.ini** file. Help Assistant will then read again all the information contained in the **.hpj** project file and create a new clean **.ini** setup file. (This approach is used to increase performance and security)

## Project

In this box type the filename for the current Help project. Help Assistant will automatically set the default .HPJ file extension.

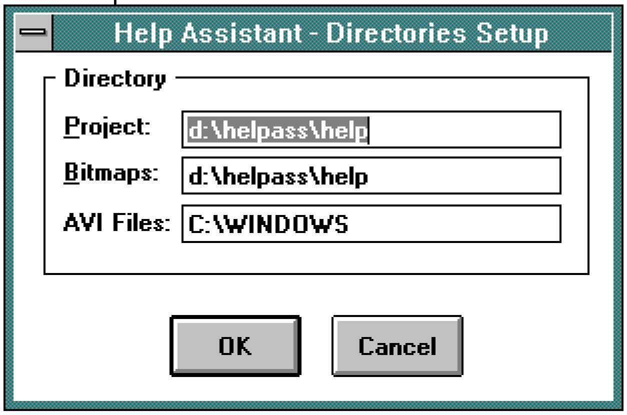
**Example:** *myhelp.hpj*

## Title

This is the text that will appear in the Title Bar of the Help window.

**Example:** *This is my Help File Title*

Press this button to access the Directory Setup text fields where the location of the Project, Bitmaps and Video files can be specified.



Project: d:\helpass\help

Bitmaps: d:\helpass\help

AVI Files: C:\WINDOWS

**See Also:** Setting Up Directories

## Citation

The citation differs from the Copyright notice in two ways: the text specified will not appear in the about dialog box, but is appended at the end of the topic when the "Copy" function is used, and will generally be much longer than the Copyright notice.

**Example:** *Copyright (c) 1993, Cie X , All rights reserved. Cannot be reproduced without prior authorisation.*

## Contents

This is a string (#) that identifies the Topic used as an index. The Index Topic is the topic that first appears when WinHelp is run.   If no Contents String is specified, the first topic of the first topic file specified in the project file will be used as the Index Topic.

**Example:** *TopicIndex*

**See Also:** Setting-up Topics.

## Copyright

This text appears in the About Dialog Box of WinHelp.

**Example:** *Copyright (c) 1993, MyName*

## Icon

This is the name of the icon to associate with the Help application.   The icon file must be in the Root directory.

**Example:**   *MyIcon.ico*

## Map

Enter the name of the header file (*.h) where context strings with context numbers are defined for context-sensitive Help. The context number corresponds to a value the parent application passes to Windows Help in order to display a particular topic.

This file must contain one or more #define statement(s) (Ex. #define ContextString 2), and can have additional #include statements as well. However, files may not be nested in this way more than five levels deep.

**Example:** *include.h*

## Include Multimedia Options

Select this option if you want to be able to play Windows wav files within a given topic.   The multimedia option is not required to play AVI files.   However, most future multimedia options will be accessible via this option.

**See Also:** Using Help Macros, Playing AVI files

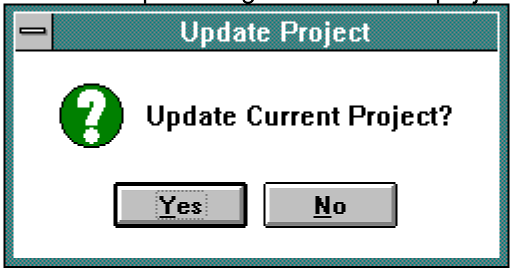
## Use Old Key Phrases option

This option is ON by default.   If you want the Help compiler to generate new Key Phrases, desable this option.   Desabling the Use Old Key Phrases option will increase the compilation time when the "Compress" option is used.   However, it is recommended to desable this option whenever significant changes have been made to the topic files.

Click on this button to access the Windows Setup dialog box.

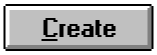
Click on this button to edit or create the header file (*.h) defined in the "Map" section.

Click on this button to exit the Help Assistant Project Setup dialog box and save changes you made to the setup parameters. Help Assistant ask if you wish to update the current project. Answer "Yes" to make the changes effective. Answer "No" to keep working on the current project.

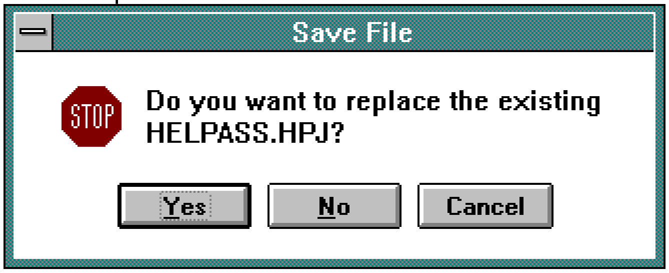


Update Current Project?

Click on this button to exit the Help Assistant Project Setup dialog box without saving changes made to the setup parameters.



Click on this button to generate a new Help Project (*.HPJ) file. Help Assistant will ask if you wish the new project to be the current project. Select "No" to generate a new project file but keep the project you are currently working on as the current project. If the project name already exists, Help Assistant will ask if you wish to replace the current project. Select "Yes" to erase all changes to the Help Project file. Select "No" to specify a different Help Project file name. Select "Cancel" to abort the operation.



Do you want to replace the existing HELPASS.HPJ?

Click on this button to select and make an existing project the current project. The standard WinWord Open dialog box appears. Enter *.HPJ to locate project files in the selected directories.

Once an existing project is selected, Help Assistant resets the current project parameters to those found in the selected project file.

Click on this button to edit the current Help Project file (.HPJ file). Help Assistant will not delete any additions to the project, unless they are overwritten during the "create new project" process.

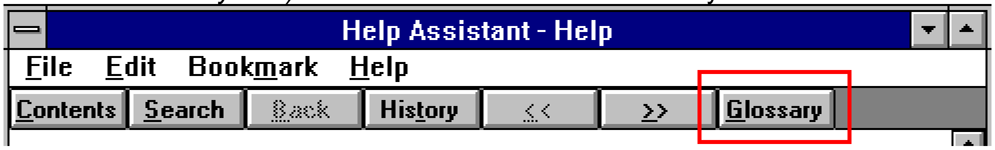
Click on this button to access the Project Macro Setup dialog box. Using this option, Project Macros may be modified, added, or removed from the project definition file (***project*.hpj** file).

Click on this button too insert / delete a file, or edit an existing file in the project.

**See Also:** Handling Topic Files.

## Include Glossary Button

Check this option to automatically include a "Glossary" button to the Help application window. This glossary is the standard Windows Glossary. To access you own glossary file, changes to the "*CreateButton ("btn_up" , "&Glossary" ...)*" statement must be made manually.

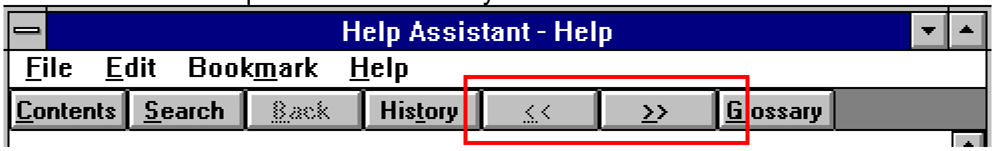


## Include CD-ROM Option

Check this option to automatically include the OPTCDROM option. This option optimizes a help file for display on CD-ROM by aligning topic files on pre-defined block boundaries.

## Include Browse Buttons

Check this option to automatically include the browse buttons.



## Warning LOW

The "WARNING" parameter specifies the amount of debugging information the Help compiler is to report.   Check this option to generate report only on the most severe errors.

## Warning MEDIUM

The "WARNING" parameter specifies the amount of debugging information the Help compiler is to report.   Check this option to generate report on an intermediate number of errors.

## Warning HIGH

The "WARNING" parameter specifies the amount of debugging information the Help compiler is to report.   Check this option to report all errors and warnings.

## Compress OFF

This option specifies that no compression is to be used when building the help file.

When developing a Help file, selecting "Compress OFF" provides the shortest compilation time.

## Compress MEDIUM

This option specifies that approximately 40% compression is to be used when building the help file.

Compression increases the compilation time.

## Compress HIGH

This option specifies that approximately 50% compression is to be used when building the help file.

This level produces the smallest help file, but also increases the compilation time significantly.

## Bitmaps Directory

This is the directory where Help Assistant locates the bitmaps to be included in the topic files.   All bitmaps must be in the same directory.

Although bitmaps can be directly inserted into the topic files, it is recommended that they be included during the compilation process.   This significantly reduces the amount of memory required to compile the project.

**Example:** *c:\bitmaps*

**See Also:** Handling Bitmaps.

## Project Directory

This is the directory where all your topic files for the current project are stored.   It must be created manually using the File Manager. See the Windows User Manual for more details on how to create a new directory.

Help Assistant automatically searches this directory for topic files.   To facilitate project management, it is recommended that each project be assigned its own directory.

**Example:** *c:\myproject*

## AVI Files Directory

This is the directory where all the Video for Windows AVI files for the current project are stored. This directory must be created manually using the File Manager. See the Windows User Manual for more details on how to create a new directory.

Help Assistant automatically searches this directory for AVI files. To facilitate project management, it is recommended that each project be assigned its own directory. Also, each video file used in the project must also be distributed with the project Help file.

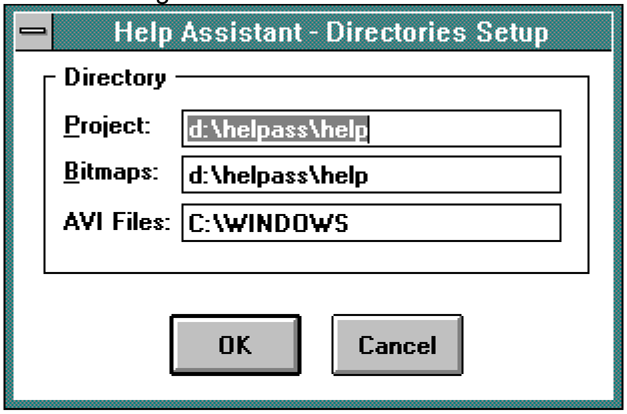
**Example:** *c:\avifiles*

## Setting Up Directories

Close

You must indicate where Help Assistant will look to find the Project files, Bitmaps and Video files in the following dialog box:

For more details, select any of the elements found in the "Help Assistant - Directories Setup" dialog box:



Project: d:\helpass\help

Bitmaps: d:\helpass\help

AVI Files: C:\WINDOWS

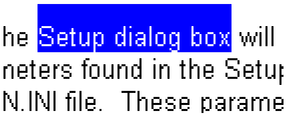
Click on this button to accept the current directory selection. Help Assistant will return to the Setup dialog box upon completion.

Click on this button to cancel any changes to the directory selection.   Help Assistant will return to the Setup dialog box.

## Creating Jumps Close

Clicking on underlined green text causes Help to jumps to the topic associated with the word or phrase selected. To create "Jumps" with Help Assistant, select the word or phrase to be used as the jump "hot-text" and select "Jump" from the "Insert" pull-down menu.

**Example:**

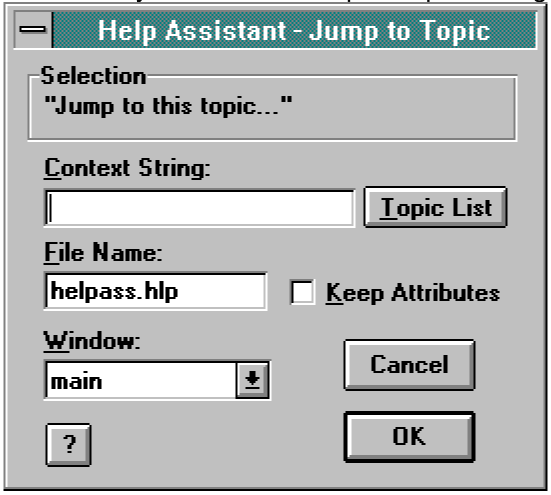


he Setup dialog box will
neters found in the Setup
N.INI file. These parame

Help Assistant displays the "Jump to Topic" dialog box. Specify the context string referencing the topic page to jump to when the underlined green text is selected.

**Short Cut:** Click on the Cancel button on the Toolbar.

Click on any field of the "Jump to Topic" dialog box to obtain more details:



Help Assistant - Jump to Topic

Selection: "Jump to this topic..."

Context String:

File Name: helpass.hlp

□ Keep Attributes

Window: main

## Context String

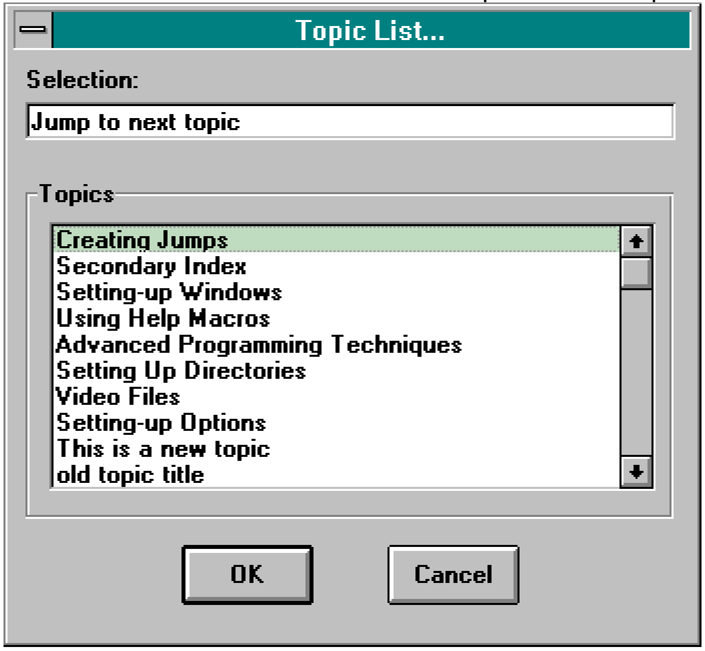
Identifies the topic page to jump to when the underlined green text is selected.

**Example:** *SetupIndex*

Click on this button to accept the context string.  Help Assistant will set the attributes of the selected text to green double underlined, and insert the context string as hidden text right next to the selected text.

Click on this button to cancel operations.

Click on this button to select a topic from the topic list.



Topic List...

Selection:

Jump to next topic

Topics

- Creating Jumps
- Secondary Index
- Setting-up Windows
- Using Help Macros
- Advanced Programming Techniques
- Setting Up Directories
- Video Files
- Setting-up Options
- This is a new topic
- old topic title

## Selection

This text box presents the current text selection which will be use as the "hot text" for the jump.   It is used as a reference only, and changes to the text will not be reflected in the topic file.

## File Name

The default file name for the jump is the current project. However, to jump to another help file, enter the name of the file where Windows Help will jump to.

**Example:** *helpfile.hlp*

## Keep Attributes Option

Windows Help use green underlined text as the default format for Jump Text. Select this option if the current features (font, style, color, underline) of the Jump text are to be preserved.

## Window

This ComboBox lists all the window styles defined in your project.  To Jump to a topic within the current topic window, do not specify any window style.  To Jump from a Secondary window to the Main window, select the "main" window style.

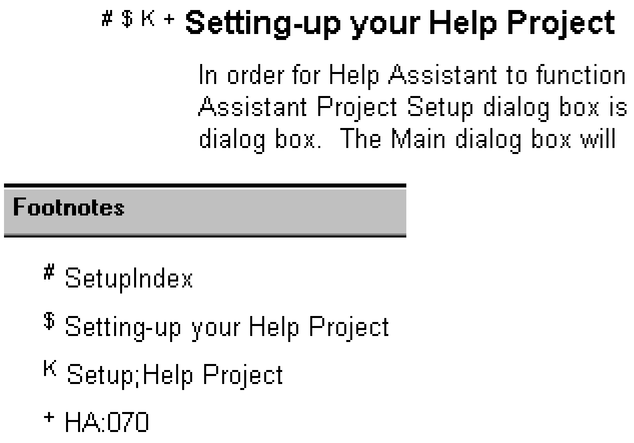
**See Also:** Setting-up Topic Windows

## Creating and Setting-up Topics Close

The topic pages are the individual sections of the help text that appear in a single scrollable window. Each topic page is identified by a context string (a unique character string like "Index") and is separated from other topic pages by a Hard Page Break (Ctrl-Enter). Each topic page may have the following features: a title, keywords, a browse sequence number, macros, jumps and pop-ups. Single files may contain as many topic as desired.

To set a topic, position the cursor anywhere within the page, and select "Topic" from the "Insert" pull-down menu. Help Assistant displays the Topic Setup dialog box from which the Topic features are specified. Those are inserted as footnotes, with one of the following signs as the footnote marker: **#, $, K, +, !, *, and @**.

**Example:**



# $ K + Setting-up your Help Project

In order for Help Assistant to function
Assistant Project Setup dialog box is
dialog box. The Main dialog box will

Footnotes

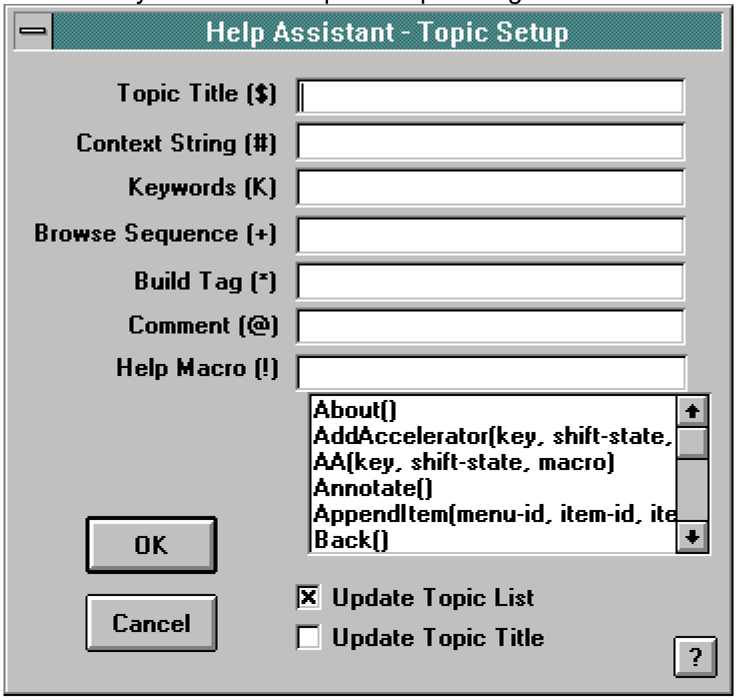
# SetupIndex

$ Setting-up your Help Project

K Setup;Help Project

+ HA:070

**Short Cut:** Click on the Cancel button on the Toolbar.
Click on any field of the "Topic Setup" dialog box to obtain more details.



Help Assistant - Topic Setup

Topic Title ($)

Context String (#)

Keywords (K)

Browse Sequence (+)

Build Tag (*)

Comment (@)

Help Macro (!)

About()
AddAccelerator(key, shift-state,
AA(key, shift-state, macro)
Annotate()
AppendItem(menu-id, item-id, ite
Back()

OK

Cancel

☒ Update Topic List

☐ Update Topic Title

?

Note that the last paragraph of each topic should be terminated with at least one <RETURN>, otherwise Word for Windows and Help Assistant will not recognize the following topic.

By default, Help Assistant does not insert the content of the Topic Title ($) field at the beginning of the topic.   You may change this behaviour by selecting the "Update Title" option in the Option Setup dialog box.   (See also: Setting-up Options)

Each time a topic is created, Help Assistant writes its title and context string to the topic list.   You may disable/enable this feature with the "Update Topic List" option in the Option Setup dialog box. (See also: Setting-up Options)

You may also add new features to a topic (e.g. Help Macros !) at any time.   However, the Topic Title ($) field of the "Topic Setup" dialog box should be cleared.   If you don't, Help Assistant will either assume you wish to replace the current topic features with the new settings and title, and delete all previous settings, or insert a second Topic Title ($) statement that includes the footnotes references.

## Context String (#)

This is the topic page ID.  To tell Windows Help where to Jump to, or what topic to Pop-up, reference the Topic Page's context string.

**Example:**  *SetupIndex*

## Topic Title ($)

It appears in the Help Bookmark menu, and in the Show topics list when a keyword search is performed in the Help file. Usually, the topic title is the same as the topic page's heading.

**Example:** *Setting-up your Help Project*

## Keywords (K)

Keywords can be assigned to each topic. They are used to search through a Help file for a specific topic. More than one keyword can be assigned to a topic. Each keyword are separated by a semicolon (;). They are not case-sensitive, but will appear in the Search dialog box exactly as they are entered.

**Example:** *Setup; Help Project*

## Browse Sequence (+)

The browse sequence numbers organize topic pages in relation to one another. Help files can be browsed through in the order of the assigned sequential numbers.  More than one browse sequence may be assigned.  Each sequence must be identified by a specific keyword.

The Help Compiler sorts browse sequence numbers alphanumerically; therefore to achieve a proper sort, the same number of digits for all the sequence numbers must be used. For example, 090 is used instead of 90 to ensure that the Help Compiler places topic 090 before 100.

**Example:**  *FirstSequence:010, FirstSequence:020..., SecondSequence:010, SecondSequence:020...*

## Build Tag (*)

Build tags may be added to a topic.

## Comment (@)

Comments may be added to each topic. Those comments will not appear anywhere in the compiled Help file.

**Example:** *This is my comment for this topic...*

## Help Macro (!)

WinHelp Macros can be associated with a particular topic.   Each time this topic is accessed, the specified macro is executed.

**Example:** *ExecProgram(`clock.exe', 1)*

Click on this button to accept all entries. It is not necessary to complete all fields. Help Assistant automatically creates footnotes for each field.

Click on this button to abort the operation.

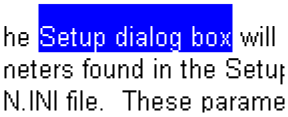
## Update Topic List

De-select this option if you do not wish to automatically create an entry in the topic list for this Topic. This function is particularly useful for pop-ups which are generally not main subject headings. Adding each pop-up to the topic list, may increase the size of the list substantially, and also the time required to read the list when setting up Jumps or Pop-ups. If the Topic Title field is empty, Help Assistant will not add any statement to the topic list.

## Creating Pop-ups Close

Clicking on dotted-underline green text causes Help to pop-up to the topic associated with the word or phrase selected, but the original topic page remains visible in the Help window. To create "Pop-ups" with Help Assistant, select the word or phrase to be used as the pop-up "hot-text" and then select "Popup" from the "Insert" pull-down menu.

**Example:**

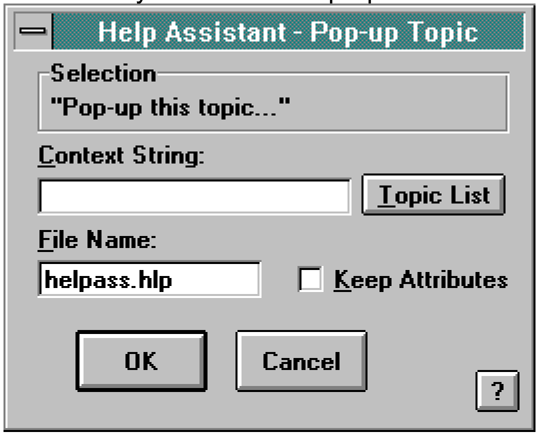


he Setup dialog box will
neters found in the Setuŗ
N.INI file. These parame

Help Assistant displays the "Pop-up Topic" dialog box. You must specify the context string referencing the topic page to pop-up. If Help Assistant finds the text selected in the list of topics, pop-ups can be created automatically. You may also select a topic from the topic list if you don't know the Context String.

**Short Cut:** Click on the Cancel button on the Toolbar.

Click on any field of the "Pop-up Definition" dialog box to obtain more details:



Help Assistant - Pop-up Topic

Selection
"Pop-up this topic..."

Context String:

File Name: helpass.hlp

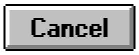
□ Keep Attributes

## Context String

Identifies the topic page you wish to pop-up when the dotted-underlined green text is selected.

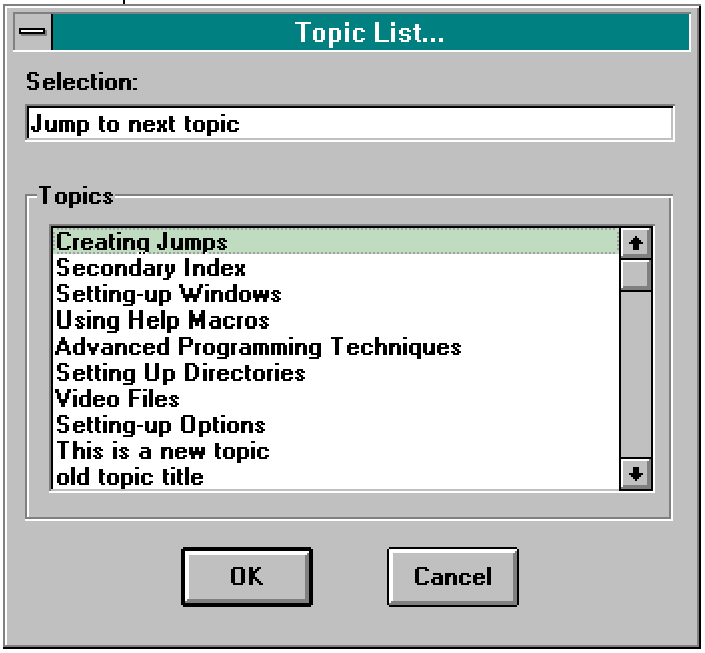
**Example:** *SetupIndex*

Click on this button to accept the current information.   Help Assistant will set the correct attributes for the selected text, and insert the pop-up string as hidden text next to the selected text.



Click on this button to cancel operations.

Click on this button to select a topic from the topic list:



Selection: Jump to next topic

Topics

- Creating Jumps
- Secondary Index
- Setting-up Windows
- Using Help Macros
- Advanced Programming Techniques
- Setting Up Directories
- Video Files
- Setting-up Options
- This is a new topic
- old topic title

**See Also:** Setting-up Topics

## Selection

This text box presents the current text selection which will be use as the "hot text" for the Pop-up.  It is used as a reference only, and changes to the text will not be reflected in the topic file.

## File Name

The default file name for the pop-up is the current project.   However, to pop-up a topic contained in another help file, enter the name of the file where WinHelp will find the topic.

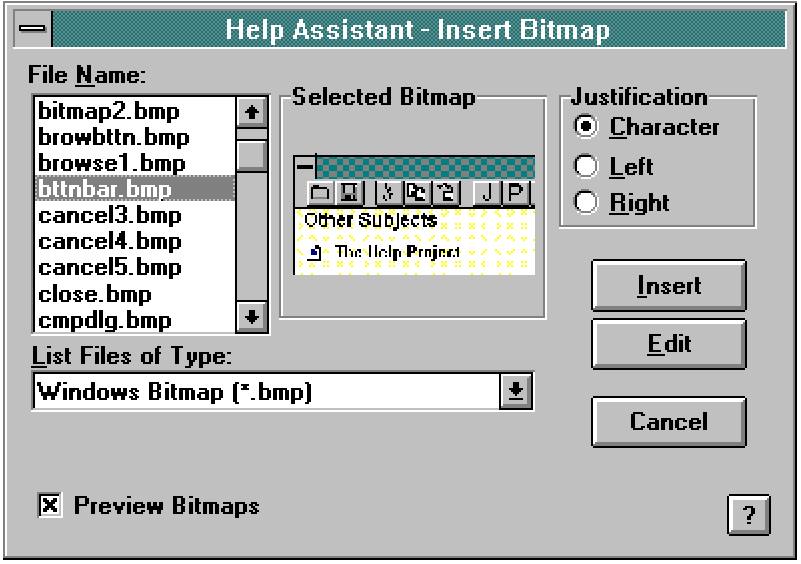
**Example:** *helpfile.hlp*

## Keep Attributes Option

Windows Help uses green dotted-underlined text as the default format for Pop-up Text.  Select this option if the current features (font, style, color, underline) of the pop-up text are to be preserved.

## Inserting Bitmaps Close

To insert a Bitmap or metafile anywhere in your topic, select "Bitmap" from the "Insert" pull-down menu, or click on the Cancel button on the HA Quick Edit Toolbar.


Help Assistant - Insert Bitmap

File Name:
bitmap2.bmp
browbttn.bmp
browse1.bmp
bttnbar.bmp
cancel3.bmp
cancel4.bmp
cancel5.bmp
close.bmp
cmpdlg.bmp

Justification
● Character
○ Left
○ Right

List Files of Type: Windows Bitmap (*.bmp)

☒ Preview Bitmaps

Select one of the bitmap files displayed in the list box and the desired justification, and click on the "INSERT" button to insert the bitmap reference at the cursor position.   To edit the selected bitmap file, click on the "EDIT" button.   To preview the selected bitmap before inserting it, select the Preview Bitmaps option.

Help Assistant will not insert the bitmap image selected, but will rather insert the **bml**, **bmc**, or **bmr** statements along with the name of the graphics file selected at the specified position. Although a bitmap image can be inserted directly in the text, it will consume a great amount of memory and limit the size of the individual topics.   Only 16-colors can be displayed by Windows Help; however, bitmaps may have more than 16 colors.

## Bitmap Image Formats

Select one of the following formats:

Windows Bitmaps (.BMP)
Segmented-graphic bitmaps (.SHG)
Placeable Windows Metafiles (.WMF)
Multiple-resolution Bitmaps (.MRB)
Device-independent Bitmaps (.DIB)

Click on this button to accept the file format selection.

## File List Box

This box displays all the files found in the specified "Bitmaps" directory for which the extension matches the graphics file format selected.   Use File Manager to move graphics files in the selected "Bitmaps" directory if they are to appear in this List Box.

## Character Justification

A bitmap can be inserted into a paragraph as if it were a character by using the default center justification. The bottom of the bitmap aligns with the base line of the current line of text and the left edge aligns with the next character position.

Paragraph properties also apply to the bitmap. Windows Help places text following the bitmap on the same base line at the next available character position.

In general, bitmaps inserted as characters, should be clipped to the smallest possible size. Extra white space at the top or bottom of the bitmap image affects the alignment of the bitmap with the text and may affect the spacing between lines.

## Left and Right Justification

A bitmap can be placed at the left or right margin of the Help window.  Use the **Left** or **Right** justification to wrap text around the bitmap. The left justification inserts a bitmap at the left margin; right justification inserts it at the right.

To wrap text around a bitmap, insert the left of right justified bitmap at the beginning of a paragraph. Windows Help aligns the start of the paragraph with the top of the bitmap and wraps around the left or right edge of the bitmap.

If a left of right justified bitmap is placed at the end of a paragraph, Windows Help places the bitmap under the paragraph instead of wrapping the text around the bitmap. To avoid wrapping text around a bitmap, insert a paragraph immediately before and after the bitmap insertion point.

Click on this button to insert the selected bitmap / metafile at the cursor's position with the specified justification.

Click on this button to edit the selected bitmap. At this time, only the **.bmp** and **.shg** formats are supported. You must have the Hot Spot Editor to be able to edit **segmented-graphics bitmaps**.

Click on this button to abort the current operation.

## Preview Bitmaps

Select this option to preview the selected bitmap image.
Only the **.bmp** format is supported.

## Selected Bitmap

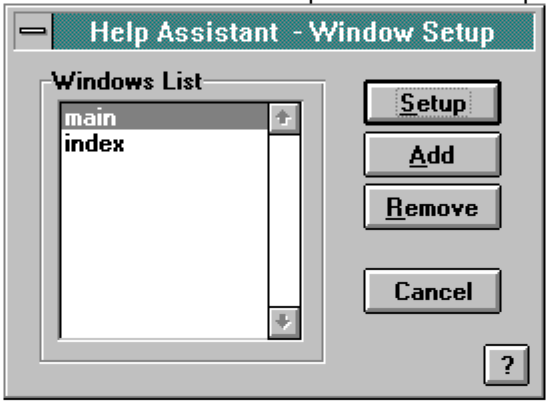
If the Preview Bitmap option is selected, the bitmap file currently selected will be displayed.

## Setting-up Topic Windows

Close

The size, position, colors, caption, and display mode for each topic window can be specified individually. Each new window name will appear in the window list of the "Jump to Topic" dialog box, where it is possible to specify the destination window used to display the topic identified by the context string. By default, WinHelp displays topics in the primary window. However, if you want to change its default characteristics, or define a number of secondary windows (secondary windows differ from the primary window by their lack of button and menu bar), you can specify their characteristics using the Windows Setup dialog box.

To access the Topic Windows Setup dialog box, select "Window Setup..." from the "Format" pull-down menu. Help Assistant will display the "Window Setup" dialog box shown below:

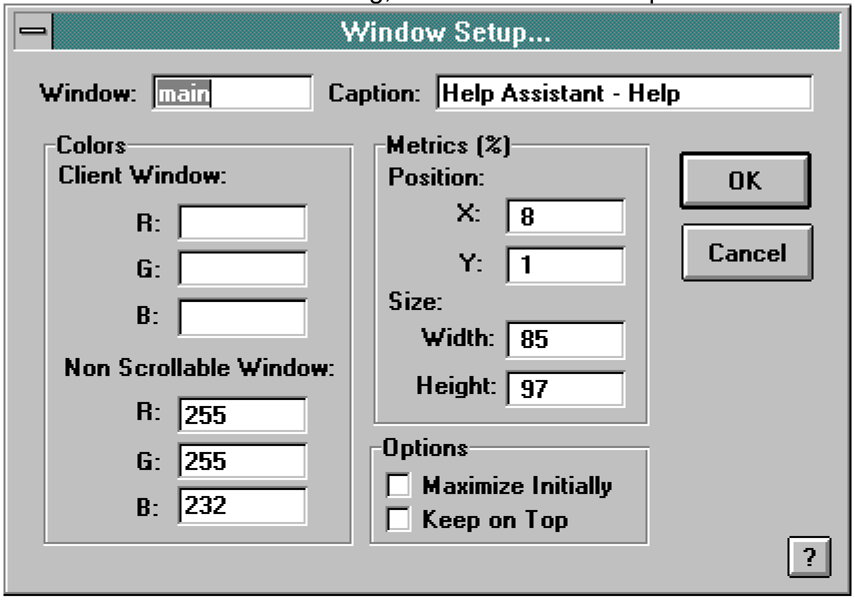


Windows List
main
index

**Short Cut:** Click on the button on the Toolbar.

Select one of the options. If you selected "Add" or "Setup", Help Assistant will display the Window Setup dialog box from which you can specify the selected topic window parameters.

For more details on each setting, click on the desired option:



Window: main
Caption: Help Assistant - Help

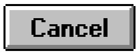
Colors
Client Window:
R:
G:
B:
Non Scrollable Window:
R: 255
G: 255
B: 232

Metrics (%)
Position:
X: 8
Y: 1
Size:
Width: 85
Height: 97

Options
□ Maximize Initially
□ Keep on Top

## Windows List Box

Select from this list the Topic Window to Setup or Remove.



Click on this button to "Add" a Topic Window definition to the project.

Click on this button to "Setup" the selected Topic Window.

Click on this button to "Remove" the selected Topic Window from the Project.

Click on this button to "Cancel" operations.

## Window Name

Indicates the name of the Topic Window. The name "main" is reserved for the primary display window. Any other name will be interpreted as a secondary window.

## Window Caption

Indicates the caption that will appear in the title bar of the selected window. If the "main" window was selected, this field contains the title specified in the "Title" field of the "Project Setup" dialog box.

### Color - Client Window

Specify in those fields the Red, Green, and Blue color components of the main topic window (also call scrollable window). These values must be between 0 and 255.

## Color - Non Scrollable Window

Specify in those fields the Red, Green, and Blue color components of the non-srollable topic window.   These values must be between 0 and 255.   The text appearing in a non-scrollable window must have its paragraph pagination set to "Keep With Next".

## Window Metrics - Position

Specify in those fields the X and Y position of the left corner of the window. These values are specified in percentage (0%-100%) of the screen area. If the sum of window size (Width and Height) and window position (X and Y) exceed 100, Help Assistant will calculate the X and Y values required to center the window on the screen.

## Window Metrics - Size

Specify in those fields the Width and Height of the window. These values are specified in percentage (0%-100%) of the screen area. They cannot exceed 100%.

## Option - Maximize Initially

If this option is selected, Windows Help will ignore the values specified for the size and position of the window.

## Option - Keep on Top

If this option is selected, Windows Help will keep the window on top of any other windows.

Click on this button to accepts entries.

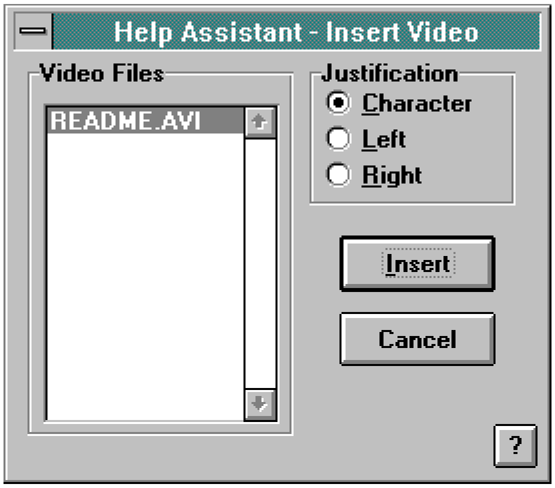
Click on this option to cancel operations.

## Inserting Video for Windows (AVI) Close

To insert a Video for Windows sequence in your topic, select "Video" from the "Insert" pull-down menu. Help Assistant inserts either the **ewl**, **ewc**, or **ewr** statements along with the name of the DLL, function call and the filename selected at the specified position.

**Short Cut:** Click on the Cancel button on the Toolbar to call up the "Play AVI" dialog box.. Help Assistant displays the "Play AVI..." dialog box. Select one of the video files displayed in the list box, click on the desired justification and click on the "INSERT" button to insert the required statement at the cursor position.

For more details on inserting Video for Windows, click on any of the "Insert Bitmap" dialog box feature:



To play Video for Windows sequences in Help files, the end user must have Microsoft Video for Windows driver installed on their system, and all avi files used with a given Help file must reside in the same directory as the Help file. Also, **helpass.dll** must also be in this directory. The **helpass.dll** can be freely distributed.

## Video Files List Box

This box displays all the AVI files found in the specified "Root" directory for which the extension matches the avi   file format. Use File Manager to move video files in the "Root" directory if they are to appear in this List Box.

## Character Justification

A Video Frame can be inserted into a paragraph as if it were a character by using the default center justification. The bottom of the frame aligns with the base line of the current line of text and the left edge aligns with the next character position.

Paragraph properties also apply to the Video Frame. Windows Help places text following the frame on the same base line at the next available character position.

## Left and Right Justification

A Video Frame can be placed at the left or right margin of the Help window. Use the **Left** or **Right** justification to wrap text around the frame. The left justification inserts a frame at the left margin; right justification inserts it at the right.

To wrap text around a Video Frame, insert the left of right justified frame at the beginning of a paragraph. Windows Help aligns the start of the paragraph with the top of the frame and wraps around its left or right edge.

If a left of right justified frame is placed at the end of a paragraph, Windows Help places the frame under the paragraph instead of wrapping the text around the bitmap. To avoid wrapping text around a frame, insert a paragraph immediately before and after the frame insertion point.

Click on this button to insert the selected Video for Windows frame (AVI) at the cursor's position with the specified justification.

Click on this button to abort the current operation.

{bmc TITLE.BMP}

## Using Help Macros

Close

Macros Reference

Help Macros can enhance significantly your help file. Among other thing, they can be used to add menus and push-buttons to the main help topic window, as well as adding multimedia capability to your project. Help Macros can be used in the same way as Jumps and Pop-ups, or added to the topic definition.

### Adding Macros to the Project definition file

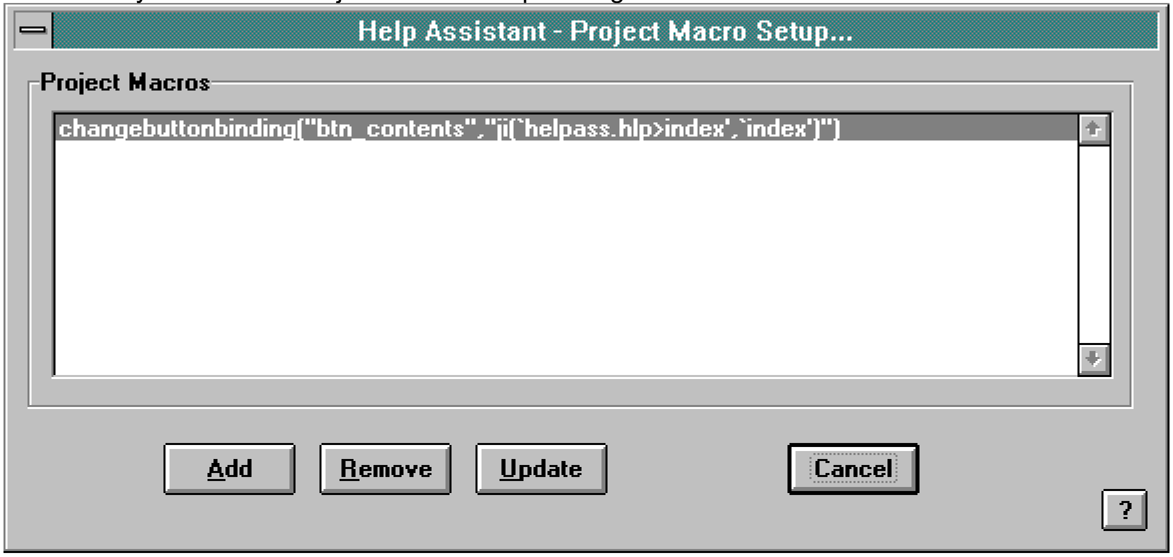
Macros may be specified in the **[CONFIG]** section of the Project (.hpj) definition file to add/modify menus and buttons in the WinHelp application, as well as registering dynamic link libraries (DLLs) to access additional functions. For instance the following macro registers the Message Box function from the **USER.EXE** dll provided with windows.

**Example:** RegisterRoutine("USER.EXE", "MessageBox", "uSSu")

To add macros to the [CONFIG] section of the Project definition file, select Project Macro from the Insert pull-down menu. Help Assistant displays the Project Macro Setup dialog box from which you can add, modify or remove macros from the project.

**Short Cut:** Click on the Cancel button on the Toolbar.

Click on any field of the "Project Macro Setup" dialog box to obtain more details:



changebuttonbinding("btn_contents","ji(`helpass.hlp>index',`index')")

### Adding Macros to a Topic

Macros which are specified with the topic definition will be executed each time the selected topic is displayed. On the other hand, macros could also be associated with a "hot-spot". This "hot-spot" can be either a bitmap image, or underlined green text. Clicking on a "hot-spot" causes Help to play the macro associated with the graphic image, word or phrase selected.

To insert "Macros" with Help Assistant, select the bitmap image, word or phrase to be used as the macro "hot-spot" and select "Macro" from the "Insert" pull-down menu.

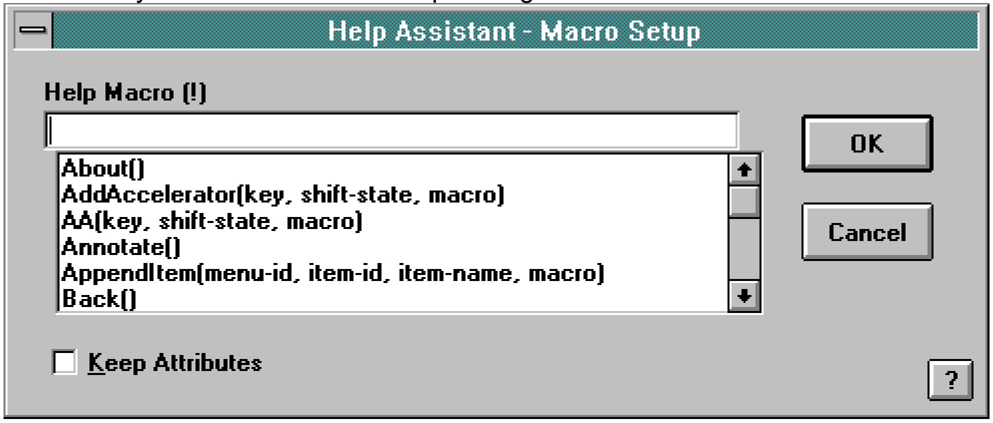
**Example:**

Cancel

Help Assistant displays the "Macro Setup" dialog box. Then select the macro to play when the underlined green text is selected. Macros may require that extra parameters be specified. In such case, parameters may be specified either before or after the macro has been inserted in the document.

**Short Cut:** Click on the M button on the Toolbar.

Click on any field of the "Macro Setup" dialog box to obtain more details:


Help Assistant - Macro Setup
Help Macro (!)
About()
AddAccelerator(key, shift-state, macro)
AA(key, shift-state, macro)
Annotate()
AppendItem(menu-id, item-id, item-name, macro)
Back()
OK
Cancel
□ Keep Attributes
?

### Playing Sound WAV files using macros

It is easy to play sound wav files using the macro features of WinHelp. Help Assistant provides access to the sndPlaySound macro function in the Macro List box found in the "Macro Setup" dialog box. Select this macro, and specify the name of the sound file to be played in replacement of the **`sound.wav'** filename found in the first macro statement.

The sndPlaySound macro statement specify the asynchronous playback mode as default. If synchronous playback is the desired mode, replace the **0x0001** parameter with **0x0000**.

**Example:** *!sndPlaySound(`tada.wav', 0x0001)*

It is necessary that the Multimedia Option be selected in the Help Assistant Setup dialog box to play sound wav files. (See Setting-up The Help Project)

## Help Macro List-Box

The Help Macro List-Box presents the custom macros found in the [CONFIG] section of the project file. Select from the list the macro you wish to remove or update.

Click on this button to ADD a custom macro to the [CONFIG] section of the project file.

Click on this button to REMOVE the selected custom macro from the [CONFIG] section of the project file.

Click on this button to UPDATE the selected custom macro.

Click on this button to CANCEL operations.

## Help Macro (!) Combo-Box

Select from the list or specify the macro you wish to insert in your application.   Some macros require that certain parameters be specified.   Parameters may be specified either in the edit field, or after inserting its template in the document.

Click on this button to accept the selection.

Click on this button to cancel operations.

## Keep Attributes Option

Select this option to conserve the current "hot-text" attributes (Style, Color, Font, Size).

## About()

This macro displays the "About" dialog box. This macro has the same effect as choosing the "About" command on the Help menu.

### Syntax

*About()*

| Parameter | Description |
|---|---|
| *none* | |

## AddAccelerator(key, shift-state, macro)

This macro Assigns an accelerator key (keyboard access) or key combination to a Help macro. The user can then execute the macro by pressing the defined key(s).

**Syntax**

*AddAccelerator(key, shift-state, "macro")*

*AA(key, shift-state, "macro")*

| Parameter | Description |
|---|---|
| *key* | Windows virtual-key value of the accelerator |
| *shift-state* | Number specifying the modifier key(s) to use with the accelerator key. Valid modifier keys are ALT, SHIFT, and CTRL. |

| Number | Modifier key(s) |
|---|---|
| 0 | (No modifier key) |
| 1 | SHIFT |
| 2 | CTRL |
| 3 | SHIFT+CTRL |
| 4 | ALT |
| 5 | ALT+SHIFT |
| 6 | ALT+CTRL |
| 7 | ALT+SHIFT+CTRL |

| Parameter | Description |
|---|---|
| *macro* | Help macro to be executed when pressing the accelerator key(s). The macro must be enclosed in quotation marks. Separate multiple macros in a string with semicolons (;). |

**Example**

The following macro assigns ALT+F10 key combination to the JumpContents macro:

*AddAccelerator(0x79, 4, "JumpContents(`Index.hlp')")*

**Comments**

This Help macro might not work in secondary windows.

**See Also:** RemoveAccelerator

## Annotate()

This macro displays the "Annotate" dialog box. This macro has the same effect as choosing the Annotate command on the Edit menu.

### Syntax

*Annotate()*

| Parameter | Description |
|---|---|
| *none* | |

### Comments

If this macro is executed from a pop-up window, the annotation is attached to the parent topic that contains the pop-up hot spot.

## AppendItem(menu-id, item-id, item-name, macro)

This macro appends a menu item at the end of a menu created with the InsertMenu macro.

### Syntax

*AppendItem("menu-id", "item-id", "item-name", "macro")*

| Parameter | Description |
|---|---|
| *menu-id* | Menu-ID used when creating the menu with the InsertMenu macro. When using *mnu_floating* as the *menu-id* Windows Help creates a popup Help menu that can be activated with the right mouse button. |
| *item-id* | Name which identify the menu item. This name is case sensitive. |
| *item-name* | Name displayed on the menu for the item. This name is case sensitive and must be enclosed in quotation marks. Within the quotation marks, place an ampersand (&) before the character you want to use for the macros accelerator key. |
| *macro* | Help macro to be executed when selecting the menu item.   The macro must be enclosed in quotation marks. Separate multiple macros in a string with semicolons (;). |

### Example

The following macro appends a menu item labeled "Video" to the menu "'View" identified by the mnu_view context string:

*AppendItem("mnu_view", "mnu_video", "&Video", "PI(`movie.hlp', `video')")*

Choosing the menu item "View - Video" pops-up the topic identified by the "video" context string in the MOVIE.HLP file. Note that the letter V serves as the accelerator key for this menu item.

### Comments

This macro cannot be executed in a secondary window.

**See Also:** ChangeItemBinding, CheckItem, DeleteItem, DisableItem, EnableItem, InsertItem, InsertMenu, UncheckItem, FloatingMenu, ResetMenu

## Back()

This macro has the same effect as selecting the "Back" button on the WinHelp toolbar, and displays the previous topic.

### Syntax

*Back()*

| Parameter | Description |
|---|---|
| *none* | |

### Comments

This macro is ignored if it is executed in a secondary window.

**See Also:** History

## BookmarkDefine()

This macro displays the "Bookmark Define" dialog box.   It has the same effect as selecting the "Define" command on the Bookmark menu.

### Syntax

*BookmarkDefine()*

| Parameter | Description |
|---|---|
| *none* | |

### Comments

If this macro is executed from a pop-up window, the bookmark is attached to the parent topic that contains the pop-up hot spot.

## BookmarkMore()

This macro displays the "Bookmark" dialog box. It has the same effect as choosing the "More" command on the Bookmark menu. (**Note:** This command appears on the Bookmark menu only if the user defines more than nine bookmarks.)

### Syntax

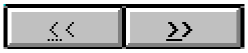
*BookmarkMore()*

| Parameter | Description |
|---|---|
| *none* | |

### Comments

If this macro is executed in a secondary window, Help displays the bookmarked topic in the secondary window, regardless of where the topic appeared when the user set the bookmark. For that reason, using this macro in secondary windows is not recommended.

## BrowseButtons()

This macro adds browse buttons  to the button bar in WinHelp.

**Syntax**

*BrowseButtons()*

| Parameter | Description |
|---|---|
| *none* | |

**Comments**

If the BrowseButtons macro is used with the CreateButton macro, the order of the browse buttons on the WinHelp button bar is determined by the order of the CreateButton and BrowseButtons macros.

Depending on how its used, the BrowseButtons macro may interfere with the DisableButton macro. This macro is ignored when it is executed in a secondary window.

**See Also:** CreateButton, DisableButton

## ChangeButtonBinding(button-id, macro)

This macro changes the assigned function of a standard Help button or any button created with the CreateButton macro.

**Syntax**

*ChangeButtonBinding("button-id", "button-macro")*

*CBB("button-id", "button-macro")*

| Parameter | Description |
|---|---|
| *button-id* | Button-ID assigned to the button in the CreateButton macro, or one of the following standard Help button IDs: |

| Button | Button ID |
|---|---|
| Contents | btn_contents |
| Search | btn_search |
| Back | btn_back |
| History | btn_history |
| << | btn_previous |
| >> | btn_next |

| | |
|---|---|
| *button-macro* | Help macro that executes when the user chooses the button. The macro must be enclosed in quotation marks. |

**Example**

The following macro changes the function of the Contents button so that choosing it causes a jump to the Table of Contents topic (identified by the tbl_of_contents context string) in the BOOK.HLP file:

*ChangeButtonBinding("btn_contents", "JI(`BOOK.HLP', `tbl_of_contents')")*

**Comments**

This macro is ignored when it is executed in a secondary window.

**See Also:** CreateButton, DestroyButton, DisableButton, EnableButton

## ChangeItemBinding(item-id, macro)

This macro changes the assigned function of a menu item added to a WinHelp menu with the AppendItem macro. This macro can also change the "How To Use Help" standard Help menu item: .

**Syntax**

*ChangeItemBinding("item-id", "item-macro")*

*CIB("item-id", "item-macro")*

| Parameter | Description |
|---|---|
| *item-id* | Item-ID assigned to the item in the AppendItem macro.   For the standard "How To Use Help" menu item, use **mnu_helpon** as the identifier. |
| *item-macro* | Help macro that executes when the user chooses the item. |

**Example**

The following macro changes the menu item identified by time_item so that it starts the Clock application:

*ChangeItemBinding("time_item", "ExecProgram(`clock', 0)")*

The following macro changes the "How To Use Help" menu item so that it opens a custom Help file:

*ChangeItemBinding("mnu_helpon", "JC(`myhelp.hlp')")*

**Comments**

Use the DeleteItem macro to remove the standard "How To Use Help" item from the Help menu. Use the SetHelpOnFile macro to specify the custom "How To Use Help" file you want to use. Then use the InsertItem macro to place the new menu item on the Help menu.   This macro is ignored if it is executed in a secondary window.

**See Also:** AppendItem, CheckItem, DeleteItem, DisableItem, EnableItem, InsertItem, InsertMenu, SetHelpOnFile, UncheckItem, FloatingMenu, ResetMenu

## CheckItem(item-id)

This macro displays a check mark next to a menu item added to a WinHelp menu with the AppendItem macro.

**Syntax**

*CheckItem("item-id")*

*CI("item-id")*

| Parameter | Description |
|---|---|
| *item-id* | Item-ID assigned to the item in the AppendItem macro. |

**Example**

The following macro checks the menu item identified by time_item:

*CheckItem("time_item")*

**Comments**

To clear the check mark from the item, use the UncheckItem macro.   This macro is ignored if it is executed in a secondary window.

**See Also:** AppendItem, ChangeItemBinding, DeleteItem, DisableItem, EnableItem, InsertItem, InsertMenu, UncheckItem, FloatingMenu, ResetMenu

## CloseWindow(window-name)

This macro closes the specified Help window.

**Syntax**

*CloseWindow("window-name")*

| Parameter | Description |
|---|---|
| *window-name* | Name of the window to close. The name *main* is reserved for the primary Help window. For secondary windows, the window name is defined in the Help Assistant "Window Setup" dialog box. |

**Example**

The following macro closes the main window:

*CloseWindow("main")*

**Comments**

This macro is ignored if the specified window does not exist.

**See Also:** Exit

## Contents()

This macro displays the Contents topic of the Help file that executes the macro. The Contents topic is specified in the Help Assistant "Project Setup" dialog box.

### Syntax

*Contents()*

| Parameter | Description |
|---|---|
| *none* | |

### Comments

If no Contents is specified in the Help project file, Help displays the first topic of the first RTF file specified in the Help project.   This macro is ignored if it is executed in a secondary window.

**See Also:** JumpContents, SetContents

## CopyDialog()

This macro has the same effect as choosing the "Copy" command on the Edit menu. It displays the "Copy" dialog box and places the text from the current topic in the copy box where the user can select text to copy to the Clipboard.

### Syntax

*CopyDialog()*

| Parameter | Description |
|---|---|
| *none* | |

### Comments

This macro and the CopyTopic() macro is the only way by which a user can copy the text displayed in a secondary window. It does not copy bitmaps or any other images in the Help topic. If this macro is executed from a pop-up window, only the text from the parent topic that contains the macro hot spot is copied to the "Copy" dialog box.

## CopyTopic()

This macro copies all the text in the currently displayed topic to the Clipboard. This macro as the same effect as pressing CTRL+INS in the main Help window.

**Syntax**

*CopyTopic()*

| Parameter | Description |
|---|---|
| *none* | |

**Comments**

This macro and the CopyDialog() macro is the only way by which a user can copy the text displayed in a secondary window. It does not copy bitmaps or any other images in the Help topic. If this macro is executed from a pop-up window, only the text from the parent topic that contains the macro hot spot is copied to the Clipboard.

## CreateButton(button-id, name, macro)

This macro creates a new button and adds it to the WinHelp button bar.

**Syntax**

*CreateButton("button-id", "name", "macro")*

*CB("button-id", "name", "macro")*

| Parameter | Description |
|---|---|
| *button-id* | Name that WinHelp uses to identify the button. |
| *name* | Label that is displayed on the button. Place an ampersand (&) before the character you want to use for the buttons accelerator key. The button name is case sensitive and can have as many as 29 characters. |
| *macro* | Help macro or macro string that is executed when the user selects the button. Separate multiple macros in a string with semicolons (;). |

**Example**

The following macro creates a new button labeled "Info". Selecting this button causes WinHelp to jump to the topic identified by the "Info_topic" context string in the BOOK.HLP file:

*CreateButton("btn_info", "&Info", "JI(`book.hlp', `Info_topic')")*

Notice that the letter I serves as the buttons accelerator key.

**Comments**

A maximum of 16 buttons may be defined on the button bar, making a total of 22 buttons, including the Browse buttons. If several buttons are created using project macros, the order of the buttons on the WinHelp button bar is determined by the order of the CreateButton and BrowseButtons macros.

This macro is ignored if it is executed in a secondary window.

**See Also:** BrowseButtons, ChangeButtonBinding, DestroyButton, DisableButton, EnableButton

## DeleteItem(item-id)

This macro removes a menu item added with the AppendItem macro.

### Syntax

*DeleteItem("item-id")*

| Parameter | Description |
|---|---|
| *item-id* | Item-ID used in the AppendItem macro. |

### Example

The following macro removes the "Video" menu item that was created in the example for the AppendItem macro:

*DeleteItem("mnu_video")*

### Comments

This macro is ignored if it is executed in a secondary window.

**See Also:** AppendItem, ChangeItemBinding, CheckItem, DisableItem, EnableItem, InsertItem, InsertMenu, UncheckItem, FloatingMenu, ResetMenu

## DeleteMark(marker-text)

This macro removes a text marker added with the SaveMark macro.

**Syntax**

*DeleteMark("marker-text")*

| Parameter | Description |
|---|---|
| *marker-text* | Text marker previously added by the SaveMark macro. |

**Example**

The following macro removes the "Help on Video" marker from a Help file:

*DeleteMark("Help on Video")*

**Comments**

WinHelp displays the "Topic not found" error message if the marker does not exist when the DeleteMark macro is executed, .

**See Also:** GotoMark, IfThen, IfThenElse, IsMark, Not, SaveMark

## DestroyButton(button-id)

This macro removes a button added with the CreateButton macro.

**Syntax**

*DestroyButton("button-id")*

| Parameter | Description |
|---|---|
| *button-id* | Button-ID assigned to the button in the CreateButton macro. |

**Example**

The following macro removes the "Info" button that was created in the example for the CreateButton macro:

*DestroyButton("btn_info")*

**Comments**

This macro cannot be used to remove a standard Help button. This macro is ignored if it is executed in a secondary window.

**See Also:** ChangeButtonBinding, CreateButton, DisableButton, EnableButton

## DisableButton(button-id)

This macro disables and greys out a button added with the CreateButton macro.

### Syntax

*DisableButton("button-id")*

*DB("button-id")*

| Parameter | Description |
|---|---|
| *button-id* | Button-ID assigned to the button in the CreateButton macro. |

### Example

The following macro disables the "Info" button that was created in the example for the CreateButton macro:

*DisableButton("btn_info")*

### Comments

This macro cannot be used to disable a button in a topic until the button has been enabled using the EnableButton macro.   If you use this macro to disable a standard Help button (Contents, Search, Back, or History), the users next action may reactivate the button.   When the BrowseButtons macro follows the DisableButton macros, it forces the standard buttons to refresh, creating the same effect as if the DisableButton macro had failed.   Consequently, to ensure that the DisableButton macro works as you intend, it must be placed after the BrowseButtons macro:

```
BrowseButtons()
DisableButton("btn_contents")
```

You can also disable the Search button in a Help file by not assigning any keywords to the topics.   This macro is ignored if it is executed in a secondary window.

**See Also:** BrowseButtons, ChangeButtonBinding, CreateButton, DestroyButton, EnableButton

## DisableItem(item-id)

This macro disables and greys out a menu item added with the AppendItem macro.

**Syntax**

*DisableItem("item-id")*

*DI("item-id")*

| Parameter | Description |
|---|---|
| *item-id* | Item-ID assigned to the menu item in the AppendItem macro. |

**Example**

The following macro disables the "Video" menu item that was created in the AppendItem macro example:

*DisableItem("mnu_video")*

**Comments**

You cannot use this macro to disable a menu item in a topic until the item has been enabled using the EnableItem macro. This macro is ignored if it is executed in a secondary window.

**See Also:** AppendItem, ChangeItemBinding, CheckItem, DeleteItem, EnableItem, InsertItem, InsertMenu, UncheckItem, FloatingMenu, ResetMenu

## EnableButton(button-id)

This macro re-enables a button disabled with the DisableButton macro.

**Syntax**

*EnableButton("button-id")*

*EB("button-id")*

| Parameter | Description |
|---|---|
| button-id | Button-ID assigned to the button in the CreateButton macro. |

**Example**

The following macro re-enables the "Info" button that was disabled in the DisableButton macro example:

*EnableButton("btn_info")*

**Comments**

When a standard Windows Help button (Contents, Search, Back, or History) is enabled using this macro, the next action may disable the button. For example, if the Contents button is enabled in one topic, it may be disabled again when jumping to a different topic. This macro is ignored if it is executed in a secondary window.

**See Also:** ChangeButtonBinding, CreateButton, DestroyButton, DisableButton

## EnableItem(item-id)

This macro re-enables a menu item disabled with the DisableItem macro.

**Syntax**

*EnableItem("item-id")*

*EI("item-id")*

| Parameter | Description |
|---|---|
| *item-id* | Item-ID assigned to the menu item in the AppendItem macro. |

**Example**

The following macro enables the "Video" menu item that was disabled in the DisableItem macro example:

*EnableItem("mnu_video")*

**Comments**

This macro is ignored if it is executed in a secondary window.

**See Also:** AppendItem, ChangeItemBinding, CheckItem, DeleteItem, DisableItem, InsertItem, InsertMenu, UncheckItem, FloatingMenu, ResetMenu

## ExecProgram(command-line, display-state)

This macro run an application.

### Syntax

*ExecProgram("command-line", display-state)*

*EP("command-line", display-state)*

| Parameter | Description |
|---|---|
| *command-line* | Command line for the application to be started. |
| *display-state* | This value indicates how the application is displayed when it is launched: |

| Value | Display |
|---|---|
| 0 | Normal |
| 1 | Minimized |
| 2 | Maximized |

### Example

The following macro runs the Notepad program in its normal window size:

*ExecProgram("notepad.exe", 0)*

### Comments

If you specify a path and command-line parameters, you must use double backslashes as showns in this example:

*ExecProgram("c:\\editors\\notepad.exe textfile.txt", 0)*

If you must use quotation marks as part of the command-line parameter, you can enclose the entire parameter in single quotation marks and omit the backslash escape character required for the double quotation marks delimiting the string, as shown in this example:

*ExecProgram(`command "string as parameter"', 0)*

## Exit()

This macro causes WinHelp to terminate. This macro has the same effect as choosing the "Exit" command on the File menu.

### Syntax

*Exit()*

| Parameter | Description |
|---|---|
| *none* | |

### Comments

This macro will close any secondary windows associated with the open Help file.

**See Also:** CloseWindow

## FileOpen()

This macro displays the "Open" dialog box. It has the same effect as choosing the "Open" command on the File menu.

### Syntax

*FileOpen()*

| Parameter | Description |
|---|---|
| *none* | |

### Comments

When using this macro in secondary windows, the user may be left without Help menus and navigation buttons.

## FocusWindow(window-name)

This macro changes the focus to the specified window.

**Syntax**

*FocusWindow("window-name")*

| Parameter | Description |
|---|---|
| window-name | Name of the window to receive the focus. The name *main* is reserved for the primary Help window. For secondary windows, the window name is defined in the Help Assistant "Window Setup" dialog box. |

**Example**

The following macro changes the focus to the "index" secondary window:

```
FocusWindow("index")
```

**Comments**

If the window does not exist, WinHelp ignores this macro.

**See Also:** CloseWindow, PositionWindow

## GotoMark(marker-text)

This macro causes WinHelp to Jump to a marker set with the SaveMark macro.

**Syntax**

*GotoMark("marker-text")*

| Parameter | Description |
|---|---|
| marker-text | Text marker previously defined by the SaveMark macro. |

**Example**

The following macro jumps to the "Help on Video" marker:

*GotoMark("Help on Video")*

**Comments**

WinHelp displays the "Topic not found" error message if the GotoMark macro specifies a marker that has not been previously defined by the SaveMark macro.

**See Also:** DeleteMark, IfThen, IfThenElse, IsMark, Not, SaveMark

## HelpOn()

This macro displays the How To Use Help file for the Windows Help application. This macro has the same effect as choosing the How To Use Help command on the Help menu.

### Syntax

*HelpOn()*

| Parameter | Description |
|---|---|
| *none* | |

### Comments

If the default How To Use Help file (WINHELP.HLP) is replaced with a custom version using the SetHelpOnFile macro, executing this macro will display the custom version of How To Use Help.

**See Also:** SetHelpOnFile

## HelpOnTop()

This macro changes the state of all Help windows to "on-top". An on-top window remains on top of other application windows, except certain windows that may also use the topmost window attribute. This macro has the same effect as choosing the Always On Top command on the Help menu.

### Syntax

*HelpOnTop()*

| Parameter | Description |
|---|---|
| *none* | |

### Comments

Microsoft does not recommend executing this macro in the main Help window. Instead use the on-top attribute when defining secondary windows.

## History()

This macro displays the history list, which shows the last 40 topics the user has viewed since opening a Help file. This macro has the same effect as choosing the History button.

**Syntax**

*History()*

| Parameter | Description |
|---|---|
| *none* | |

**Comments**

This macro is ignored if it is executed in a secondary window.

**See Also:** Back

## IfThen(test, macro)

This macro executes a Help macro if a given marker exists. It uses the IsMark macro to make the test. You can also use a DLL function as a condition for this macro.

### Syntax

```
IfThen(IsMark("marker-text"), "macro")
```

| Parameter | Description |
|---|---|
| *marker-text* | Text marker previously created by the SaveMark macro. The IsMark macro tests the marker you specify. If the marker value that the test returns is zero, the macro does not execute. If the value is something other than zero, the macro executes. The marker text must be enclosed in quotation marks. |
| *macro* | Help macro or macro string that executes if the marker exists. Separate multiple macros in a string with semicolons (;). |

### Example

The following macro jumps to the topic with the play_video context string if the SaveMark macro has set a marker named "Help on Video":

```
IfThen(IsMark("Help on Video"), "JI(book.hlp', `video_topic')")
```

### Comments

You can use the IfThen macro to create many custom effects in your Help file. For example, you can use it to add a button to some topics and not have it appear in other topics. In the topic(s) where you want the button to appear, you create a macro footnote with a macro string similar to this example:

```
IfThen(Not(IsMark(`Help on Video')), "SaveMark(`Help on Video) : CreateButton(``video_btn',
`&Video, `JumpContents(`book.hlp')')")
```

In the topics where you dont want the button to appear, you create a macro footnote with this macro string:

```
IfThen(IsMark(`Help on Video), "DeleteMark(`Help on Video) : DestroyButton(`video_btn')")
```

If a topic does not have this footnote, it will have the same button characteristics as the previously viewed topic.

**See Also:** DeleteMark, GotoMark, IfThenElse, IsMark, Not, SaveMark

## IfThenElse(test, macro1, macro2)

This macro executes one of two Help macros depending on whether a marker exists. It uses the IsMark macro to make the test. You can also use a DLL function as a condition for this macro.

### Syntax

*IfThenElse(IsMark("marker-text"), "macro1", "macro2")*

| Parameter | Description |
|---|---|
| *marker-text* | Text marker previously created by the SaveMark macro. The IsMark macro tests the marker you specify. |
| *macro1* | WinHelp executes macro1 if the test returns a nonzero marker value. Separate multiple macros in the string with semicolons (;). |
| *macro2* | WinHelp executes macro2 if the test returns a marker value of zero. Separate multiple macros in the string with semicolons (;). |

### Example

The following macro jumps to the topic with the man_mem context string if the SaveMark macro has set a marker named "Help on Video". If the marker does not exist, the macro jumps to the Contents topic in the BOOK.HLP file:

```
IfThenElse(IsMark("Help on Video"), "JI(`book.hlp', `video_topic')","JumpContents(`book.hlp')")
```

**See Also:** DeleteMark, GotoMark, IfThen, IsMark, Not, SaveMark

## InsertItem(menu-id, item-id, item-name, macro, position)

This macro inserts a menu item at a given position on an existing menu. The menu can either be one you create with the InsertMenu macro or a standard Windows Help menu.

### Syntax

*InsertItem("menu-id", "item-id", "item-name", "macro", position)*

| Parameter | Description |
|---|---|
| *menu-id* | Menu-ID used in the InsertMenu macro to create the menu or the name of a standard Windows Help menu. Standard menu names and identifiers are:<br>**Name**<br>IdentifierFile<br>mnu_fileEdit<br>mnu_editBookmark<br>mnu_bookmarkHelp<br>mnu_helpon<br>mnu_floating |
| *item-id* | Name used to identify the menu item. |
| *item-name* | Label that WinHelp displays on the menu for the item. This name is case sensitive. Place an ampersand (&) before the character you want to use for the items accelerator key. |
| *macro* | Help macro or macro string that is executed when the user chooses the menu item.  Separate multiple macros in a string with semicolons (;). |
| *position* | Number specifying the position in the menu where the new item will appear. The number must be an integer. Position 0 is the first or topmost position in the menu. |

### Example

The following macro inserts a menu item labeled "Video" as the fourth item on a View menu that has a mnu_view identifier:

*InsertItem("mnu_view", "mnu_video", "&Video", "JI(`book.hlp', `video_topic')", 3)*

Choosing the "Video" menu item causes WinHelp to jump to the topic with the "video_topic" context string in the BOOK.HLP file. In this case the letter V serves as the items accelerator key.

### Comments

The access key assigned to a menu item must be unique.

**See Also:** AppendItem, ChangeItemBinding, CheckItem, DeleteItem, DisableItem, EnableItem, InsertMenu, UncheckItem, FloatingMenu, ResetMenu

## InsertMenu(menu-id, menu-name, menu-position)

This macro adds a new menu to the WinHelp menu bar.

### Syntax

*InsertMenu("menu-id", "menu-name", menu-position)*

| Parameter | Description |
|---|---|
| *menu-id* | String which identifies the menu. Use this identifier in the AppendItem macro to add menu items (commands) to the menu. |
| *menu-name* | Label for the menu that WinHelp displays on the menu bar. This label is case sensitive. Place an ampersand (&) before the character used for the menus accelerator key. |
| *menu-position* | Number which specifies the position on the menu bar. This number must be an integer. Positions are numbered from left to right, with position 0 being the leftmost menu. |

### Example

The following macro adds a menu named "Volumes" to WinHelp:

*InsertMenu("menu_vols", "&Volumes", 3)*

"Volumes" appears as the fourth menu on the WinHelp menu bar, between the Bookmark and Help menus. The user presses ALT+V to open the menu.

### Comments

The accelerator key assigned to a menu must be unique.This macro is ignored if it is executed in a secondary window.

**See Also:** AppendItem, ChangeItemBinding, CheckItem, DeleteItem, DisableItem, EnableItem, InsertItem, UncheckItem, FloatingMenu, ResetMenu

## IsMark(marker-text)

This macro tests whether a marker set by the SaveMark macro exists. This macro is used as a parameter to the conditional macros IfThen and IfThenElse. The IsMark macro returns nonzero if the marker exists or zero if it does not.

### Syntax

*IsMark("marker-text")*

| Parameter | Description |
|---|---|
| *marker-text* | Text string tested by the IsMark macro. |

### Example

The following macro jumps to the topic with the "video_topic" context string if the SaveMark macro has set a marker named "Help on Video". The IsMark macro tests for the "Help on Video" marker:

*IfThen(IsMark("Help on Video"), "JI(`book.hlp', `video_topic')")*

### Comments

The "Not" macro can be used to reverse the results of the IsMark macro.

**See Also:** DeleteMark, GotoMark, IfThen, IfThenElse, Not, SaveMark

## JumpContents(filename)

This macro causes WinHelp to jump to the Contents topic of a specified Help file. The Contents topic is defined in the Help Assistant "Project Setup" dialog box.

**Syntax**

*JumpContents("filename")*

| Parameter | Description |
|---|---|
| *filename* | Name of the destination Help file for the jump. |

**Example**

The following macro jumps to the Contents topic of the BOOK.HLP file:

*JumpContents("book.hlp")*

**Comments**

If no Contents is specified in the Help project file, Help displays the first topic of the first RTF file specified in the Help project.

**See Also:** Contents, SetContents

## JumpContext(filename, context-number)

This macro causes WinHelp to jump to a specific context within a Help file. The context is identified by an entry in the [MAP] section of the Help project file.

**Syntax**

*JumpContext("filename", context-number)*

*JC("filename", context-number)*

| Parameter | Description |
|---|---|
| *filename* | Name of the destination Help file for the jump. |
| *context-number* | Context number of the topic in the destination Help file. This number must be defined in the [MAP] section of the destination Help files project file. |

**Example**

The following macro jumps to the topic mapped to the 22 context ID number in the BOOK.HLP file:

*JumpContext("BOOK.HLP", 22)*

**Comments**

If the context number does not exist or cannot be found in the [MAP] section, WinHelp jumps to the Contents topic or the first topic in the Help file and displays an error message.

**See Also:** JumpId, PopupContext

## JumpHelpOn()

This macro causes WinHelp to jump to the Contents topic of the "How To Use Help" file.

**Syntax**

*JumpHelpOn()*

| Parameter | Description |
|---|---|
| *none* | |

**Comments**

The "How To Use Help" file is either WINHELP.HLP or the Help file designated by the SetHelpOnFile project macro.

**See Also:** HelpOn, SetHelpOnFile

## JumpId(filename, context-string)

This macro causes WinHelp to jump to the topic with the specified context string in the specified Help file.

**Syntax**

*JumpId("filename", "context-string")*

*JI("filename", "context-string")*

| Parameter | Description |
|---|---|
| *filename* | Name of the Help file where the topic is located. |
| *context-string* | Context string of the topic. |

**Example**

The following macro jumps to a topic with the "video_topic" context string in the BOOK.HLP file:

*JumpId("book.hlp", "video_topic")*

**Comments**

You can use the JumpId macro to display topics in secondary windows by adding the window name to the filename parameter, as in this example:

*JumpId("book.hlp>wnd_two", "video_topic")*

The topic identified by the "video_topic" context string would appear in the "wnd_two" secondary window.

If the JumpId macro is used without specifying a filename, WinHelp performs the jump in the current Help file, as in this example:

*JumpId("", "video_topic")*

However, this method is not recommended.

**See Also:** JumpContext, PopupId

## JumpKeyword(filename, keyword)

This macro opens the specified Help file, searches through the keyword table, and displays the first topic which contains the keyword specified in the macro.

**Syntax**

*JumpKeyword("filename", "keyword")*

*JK("filename", "keyword")*

| Parameter | Description |
|---|---|
| *filename* | Name of the Help file. |
| *keyword* | Keyword to search for. |

**Example**

The following macro displays the first topic with "video" as a keyword in the BOOK.HLP file:

*JumpKeyword("book.hlp", "video")*

**Comments**

If WinHelp finds more than one keyword match, the first match found will be displayed.   If it does not find a match, the Contents topic of the destination Help file will be displayed.

**See Also:** Search

## Next()

This macro displays the next topic in the browse sequence for the Help file. This macro has the same effect as choosing the Browse >> next button .

**Syntax**

*Next()*

| Parameter | Description |
|---|---|
| *none* | |

**Comments**

This macro is ignored if it is executed in a secondary window.

**See Also:** BrowseButtons, Prev

## Not(test)

This macro reverses the result returned by the IsMark macro. It is used with the IsMark macro as a parameter to the conditional macros IfThen and IfThenElse.

**Syntax**

*Not(IsMark("marker-text"))*

| Parameter | Description |
|---|---|
| *marker-text* | Text marker previously created by the SaveMark macro. |

**Example**

The following macro executes a jump to the topic with the "video_topic" context string if the SaveMark macro has not set a marker named "Help on Video":

*IfThen(Not(IsMark("Help on Video")), "JI(`book.hlp', `video_topic')")*

**Comments**

The IsMark macro tests the specified marker.   The Not macro returns zero if the mark exists (IsMark returns nonzero) or nonzero if the mark does not exist (IsMark returns zero).

**See Also:** DeleteMark, GotoMark, IfThen, IfThenElse, IsMark, SaveMark

## PopupContext(filename, context-number)

This macro displays a pop-up window containing the topic identified by a specific context number. The context is identified by an entry in the [MAP] section of the Help project file.

### Syntax

*PopupContext("filename", context-number)*

*PC("filename", context-number)*

| Parameter | Description |
|---|---|
| *filename* | Name of the Help file that contains the topic to be displayed in the pop-up window. |
| *context-number* | Context number of the topic to be displayed in the pop-up window. This number must be defined in the [MAP] section of the specified Help files project file.. |

### Example

The following macro displays in a pop-up window the topic mapped to the 22 context ID number in the BOOK.HLP file:

*PopupContext("book.hlp", 22)*

### Comments

If the context number does not exist or cannot be found in the [MAP] section, WinHelp displays the Contents topic or the first topic in the Help file.

**See Also:** JumpContext

## PopupId(filename, context-string)

This macro displays a pop-up window containing the topic identified by a specific context string. Unlike the PopupContext macro,

**Syntax**

*PopupId("filename", "context-string")*

*PI("filename", "context-string")*

| Parameter | Description |
|---|---|
| *filename* | Name of the Help file which contains the topic. |
| *context-string* | Context string which identifies the topic |

**Example**

The following macro displays in a pop-up window a topic identified by the "video_topic" context string in the BOOK.HLP file:

*PopupId("book.hlp", "video_topic")*

**Comments**

If the context string does not exist or cannot be found, WinHelp displays the Contents topic or the first topic in the Help file.

**See Also:** JumpId

## PositionWindow(**x-coord, y-coord, width, height, window-state, window-name**)

This macro sets the size and position of a window.

**Syntax**

*PositionWindow(x-coord, y-coord, width, height, window-state,*

*"window-name")*

*PW(x-coord, y-coord, width, height, window-state,*

*"window-name")*

| Parameter | Description |
|---|---|
| *x-coord* | X-coordinate, in Help units, of the upper-left window corner. |
| *y-coord* | Y-coordinate, in Help units, of the upper-left window corner. |
| *width* | Default width, in Help units, of the window. |
| *height* | Default height, in Help units, of the window. |
| *window-state* | Specifies the windows state when it is displayed.   The values for the ShowWindow function are explained in the following table. |

| VALUE | CONSTANT | ACTION |
|---|---|---|
| 0 | **SW_HIDE** | Hides the window and passes activation to another window. |
| 1 | **SW_SHOWNORMAL** | Activates and displays a window. If the window is minimized or maximized, Windows restores it to its original size and position (same as SW_RESTORE). |
| 2 | **SW_SHOWMINIMIZED** | Activates a window and displays it as an icon. |
| 3 | **SW_SHOWMAXIMIZED** | Activates a window and displays it as a maximized window.   WinHelp ignores the x-coord, y-coord, width, and height parameters. |
| 4 | **SW_SHOWNOACTIVATE** | Displays a window in its most recent size and position. The window that is currently active remains active. |
| 5 | **SW_SHOW** | Activates a window and displays it in its current size and position. |
| 6 | **SW_MINIMIZE** | Minimizes the specified window and activates the top-level window in the systems list. |
| 7 | **SW_SHOWMINNOACTIVE** | Displays a window as an icon. The window that is currently active remains active. |
| 8 | **SW_SHOWNA** | Displays a window in its current state. The window that is currently active remains active. |
| 9 | **SW_RESTORE** | Activates and displays a window. If the window is minimized or maximized, Windows restores it to its original size |

and position (same as SW_SHOWNORMAL).

*window-name* Name of the window to position. The name main is reserved for the primary Help window. Secondary window names are defined in the Help Assistant "Window Setup" dialog box.

**Example**

The following macro displays and positions the "Index" secondary window in the upper-left corner (0,0) with half the width and height of the display:

*PositionWindow(0, 0, 512, 512, 5, "Index")*

**Comments**

Help Units are defined in a 1024-by-1024 coordinate system, regardless of screen resolution.

**See Also:** CloseWindow, FocusWindow

## Prev()

This macro displays the previous topic in the browse sequence for the Help file. This macro has the same effect as choosing the Browse << previous button.

**Syntax**

*Prev()*

| Parameter | Description |
|---|---|
| *none* | |

**Comments**

This macro is ignored if it is executed in a secondary window.

**See Also:** BrowseButtons, Next

## Print()

This macro prints the currently displayed topic.   It has the same effect as selecting the "Print Topic" command on the File menu.

### Syntax

*Print()*

| Parameter | Description |
|---|---|
| *none* | |

### Comments

This macro should be used only to print topics in windows other than the *main* Help window.   If the macro is executed from a pop-up window, WinHelp prints the topic that contains the pop-up hot spot.

## PrinterSetup()

This macro displays the "Print Setup" dialog box. It has the same effect as choosing the "Print Setup" command on the File menu.

### Syntax

*PrinterSetup()*

| Parameter | Description |
|---|---|
| *none* | |

## RegisterRoutine(DLL-name, function-name, format-spec)

This macro registers a function within a dynamic-link library (DLL) as a Help macro. These functions can be used in macro hot spots or footnotes within topic files or in the [CONFIG] section of the Help project file, the same as standard Help macros.

**Note**: This macro ignores all return values.

### Syntax

*RegisterRoutine("DLL-name", "function-name", "parameter-spec")*

*RR("DLL-name", "function-name", "parameter-spec")*

| Parameter | Description |
|---|---|
| *DLL-name* | String specifying the filename of the DLL being called. You can omit the .DLL filename extension. Specify the directory only if necessary. |
| *function-name* | String specifying the name of the function you want to use as a Help macro. |
| *parameter-spec* | String specifying the formats of parameters passed to the function. Characters in the string represent C parameter types. Valid parameter types include the following: |

| CHARACTER | DATA TYPE | Equivalent Windows data type |
|---|---|---|
| **u** | Unsigned short integer | UINT, WORD, WPARAM |
| **U** | Unsigned long integer | DWORD |
| **i** | Signed short integer | BOOL (also C int or short) |
| **I** | Signed long integer | LONG, LPARAM, LRESULT |
| **s** | Near pointer to a null-terminated text string | PSTR, NPSTR |
| **S** | Far pointer to a null-terminated text string | LPSTR, LPCSTR |
| **v** | Void (means no type; used only with return values) | None. Equivalent to C void data type. |

### Example

The following DLL call, registers a routine named sndPlaySound in the DLL named MMSYSTEM.DLL:

*RegisterRoutine("mmsystem.dll","sndPlaySound","Su")*

### Comments

Generally, DLLs are installed in the directory where WinHelp resides. If WinHelp cannot find the DLL, it displays an error message and does not perform the call.

## RemoveAccelerator(key, shift-state)

This macro removes an accelerator keyboard (access) key or key combination assigned to a Help macro.

### Syntax

*RemoveAccelerator(key, shift-state)*

*RA(key, shift-state)*

| Parameter | Description |
|---|---|
| *key* | Windows virtual-key value assigned to the macro using the AddAccelerator macro. |
| *shift-state* | Number specifying the key or key combination to use with the accelerator key. Valid modifier keys are ALT, SHIFT, and CTRL: |

| Number | Modifier key(s) |
|---|---|
| 0 | (No modifier key) |
| 1 | SHIFT |
| 2 | CTRL |
| 3 | SHIFT+CTRL |
| 4 | ALT |
| 5 | ALT+SHIFT |
| 6 | ALT+CTRL |
| 7 | ALT+SHIFT+CTRL |

### Example

The following macro removes the ALT+F10 key combination that was assigned in the AddAccelerator macro example:

*RemoveAccelerator(0x79, 4)*

### Comments

This macro is ignored if the author attempts to remove an unassigned accelerator key.

**See Also:** AddAccelerator

## SaveMark(marker-text)

This macro saves the location of the currently displayed topic and associates a text marker with that location. The GotoMark macro can then be used to jump to this location.

### Syntax

*SaveMark("marker-text")*

| Parameter | Description |
|---|---|
| *marker-text* | Text marker used to identify the topic location. |

### Example

The following macro saves the "Help on Video" marker in the current topic in the BOOK.HLP file:

*SaveMark("Help on Video")*

### Comments

Text markers are not saved if the user exits and then restarts WinHelp.   If the same text is used for more than one marker, WinHelp uses the most recently entered marker.

**See Also:** DeleteMark, GotoMark, IfThen, IfThenElse, IsMark, Not

## Search()

This macro displays the "Search" dialog box, which allows users to search for topics using keywords. It has the same effect as choosing the Search button.

### Syntax

*Search()*

| Parameter | Description |
|---|---|
| *none* | |

### Comments

This macro is ignored if it is executed in a secondary window.

**See Also:** JumpKeyword

## SetContents(filename, context-number)

This macro designates a specific topic as the Contents topic in the specified Help file.

### Syntax

*SetContents("filename", context-number)*

| Parameter | Description |
|---|---|
| *filename* | Name of the Help file containing the desired Contents topic. |
| *context-number* | Context number of the topic. This number must be defined in the [MAP] section of the specified Help files project file. |

### Example

The following macro sets the topic mapped to the 22 context ID number in the BOOK.HLP file as the Contents topic:

*SetContents("book.hlp", 22)*

After this macro executes, choosing the Contents button causes a jump to the topic mapped to 22.

### Comments

If the context number does not exist or cannot be found in the [MAP] section, WinHelp displays an error message.

**See Also:** Contents, JumpContents

## SetHelpOnFile(filename)

This macro designates the Help file that is to replace WINHELP.HLP, the "How To Use Help" Help file.

**Syntax**

*SetHelpOnFile("filename")*

| Parameter | Description |
|---|---|
| *filename* | Name of the new "How To Use Help" Help file. |

**Example**

The following macro sets the "How To Use Help" file as BOOKHLP.HLP:

*SetHelpOnFile("bookhlp.hlp")*

To ensure that the "How To Use Help" file is always displayed in the *main* Help window, add the window name "*main"* to the macro and add this macro to the list of project macros in the Help Assistant "Project Macro Setup" dialog box:

*SetHelpOnFile("bookhlp.hlp>main")*

**Comments**

If this macro is executed from a secondary window, the "How To Use Help" file will appear in the secondary window.

**See Also:** HelpOn, JumpHelpOn

## sndPlaySound(filename, flag)

This macro plays the specified waveform sound.

### Syntax

*sndPlaySound("filename", flag)*

| Parameter | Description |
|---|---|
| *filename* | Specifies the name of the sound to play. |
| *flag* | Specifies option for playing the sound using one of the following flags: |

| VALUE | CONSTANT | ACTION |
|---|---|---|
| 0 | **SND_SYNC** | The sound is played synchronously and the function does not return until the sound ends. |
| 1 | **SND_ASYNC** | The sound is played asynchronously and the function returns immediately after beginning the sound. To terminate an asynchronously-played sound, call sndPlaySound with *filename* set to NULL.. |

### Example

The following macro will play the file tada.wav asynchronously:

*sndPlaySound(`tada.wav', 1)*

### Comments

The sound must fit in available physical memory and be playable by an installed waveform audio device driver.

## UncheckItem(item-id)

This macro removes the check mark from a menu item added to a WinHelp menu with the CheckItem macro.

**Syntax**

*UncheckItem("item-id")*

*UI("item-id")*

| Parameter | Description |
|---|---|
| *item-id* | Item-ID assigned to the menu item in the AppendItem macro. |

**Example**

The following macro removes the check mark from the menu item identified by vide_video:

*UncheckItem("view_video")*

**Comments**

To check a menu item, use the CheckItem macro.   This macro is ignored if it is executed in a secondary window.

**See Also:** AppendItem, ChangeItemBinding, CheckItem, DeleteItem, DisableItem, EnableItem, InsertItem, InsertMenu, FloatingMenu, ResetMenu

## Help Macros Reference

The following list organize the Help macros according to function which provides a quick overview of the related macros.

- **Button Macros**
- **Menu Macros**
- **Linking Macros**
- **Window Macros**
- **Keyboard Macros**
- **Auxiliary/Multimedia Macros**
- **Text-Marker Macros**

## Button Macros

The following macros are used to access the standard Help buttons, create new buttons, or to modify a button's functionality.  Click on a macro to obtain more details:

| MACRO | FUNCTION |
|---|---|
| Back | Displays the previous topic in the Back list. |
| BrowseButtons | Adds the Browse buttons to the Help button bar. |
| ChangeButtonBinding | Changes the assigned function of a Help button. |
| Contents | Displays the Contents topic of the current Help file. |
| CreateButton | Creates a new button and adds it to the button bar. |
| DestroyButton | Removes a button from the button bar. |
| DisableButton | Disables a button on the button bar. |
| EnableButton | Enables a disabled button. |
| History | Displays the history list. |
| Next | Displays the next topic in a browse sequence. |
| Prev | Displays the previous topic in a browse sequence. |
| Search | Displays the Search dialog box. |
| SetContents | Designates a specific topic as the Contents topic. |

## Menu Macros

The following macros are used to access the standard Help menu items, create new menus and menu items, or to modify menus and menu items.   Click on a macro to obtain more details:

| MACRO | FUNCTION |
|---|---|
| **About** | Displays the About dialog box. |
| **Annotate** | Displays the Annotate dialog box. |
| **AppendItem** | Appends a menu item to the end of a custom menu. |
| **BookmarkDefine** | Displays the Bookmark Define dialog box. |
| **BookmarkMore** | Displays the Bookmark dialog box. |
| **ChangeItemBinding** | Changes the assigned function of a menu item. |
| **CheckItem** | Displays a check mark next to a menu item. |
| **CopyDialog** | Displays the Copy dialog box. |
| **CopyTopic** | Copies the current topic to the Clipboard. |
| **DeleteItem** | Removes a menu item from a menu. |
| **DisableItem** | Disables a menu item. |
| **EnableItem** | Enables a disabled menu item. |
| **Exit** | Exits the Windows Help application. |
| **FileOpen** | Displays the Open dialog box. |
| **FloatingMenu** | Displays a floating menu if defined. |
| **HelpOn** | Displays the How To Use Help file. |
| **InsertItem** | Inserts a menu item at a given position on a menu. |
| **InsertMenu** | Adds a new menu to the Help menu bar. |
| **Print** | Sends the current topic to the printer. |
| **PrinterSetup** | Displays the Print Setup dialog box. |
| **ResetMenu** | Resets the entire standard WinHelp menu to its default state. |
| **SetHelpOnFile** | Specifies a custom How To Use Help file. |
| **UncheckItem** | Removes a check mark from a menu item. |

## Linking Macros

The following macros can be used to create hypertext links to specific Help topics. Click on a macro to obtain more details:

| MACRO | FUNCTION |
|---|---|
| **JumpContents** | Jumps to the Contents topic of a specific Help file. |
| **JumpContext** | Jumps to the topic with a specific context number. |
| **JumpHelpOn** | Jumps to the Contents of the How To Use Help file. |
| **JumpId** | Jumps to the topic with a specific context string. |
| **JumpKeyword** | Jumps to the first topic containing a specified keyword. |
| **PopupContext** | Displays the topic with a specific context number in a pop-up window. |
| **PopupId** | Displays the topic with a specific context string in a pop-up window. |

## Window Macros

The following macros can be used to control or modify the behavior of the main Help window or secondary Help windows. Click on a macro to obtain more details:

| MACRO | FUNCTION |
|---|---|
| **CloseWindow** | Closes the main or secondary Help window. |
| **FocusWindow** | Changes the focus to a specific Help window. |
| **HelpOnTop** | Places all Help windows on top of other windows. |
| **PositionWindow** | Sets the size and position of a Help window. |

## Keyboard Macros

The following macros can be used to add keyboard access to a Help macro.   Click on a macro to obtain more details:

| MACRO | FUNCTION |
|---|---|
| **AddAccelerator** | Assigns an accelerator key to a Help macro. |
| **RemoveAccelerator** | Removes an accelerator key from a Help macro. |

## Auxiliary/Multimedia Macros

The following macros can be used to access applications and functionality not available in Windows Help. Click on a macro to obtain more details:

| MACRO | FUNCTION |
|---|---|
| **ExecProgram** | Starts an application. |
| **RegisterRoutine** | Registers a function within a DLL as a Help macro. |
| **sndPlaySound** | Play a wave audio file |

## Text-Marker Macros

The following macros can be used to create and manipulate text markers. Click on a macro to obtain more details:

| MACRO | FUNCTION |
|---|---|
| DeleteMark | Removes a marker added by SaveMark. |
| GotoMark | Executes a jump to a marker set by SaveMark. |
| IfThen | Executes a Help macro if a given marker exists. |
| IfThenElse | Executes one of two macros if a given marker exists. |
| IsMark | Tests whether a marker set by SaveMark exists. |
| Not | Reverses the result returned by IsMark. |
| SaveMark | Saves a marker for the current topic and Help file. |

## FloatingMenu()

This macro displays the floating menu. It has the same effect as clicking the right mouse button.   A floating menu must be defined using the AppendItem macro.

### Syntax

*FloatingMenu()*

| Parameter | Description |
|---|---|
| *none* | |

## ResetMenu()

This macro displays resets the entire standard WinHelp menu to its default state.

**Syntax**

*ResetMenu()*

| Parameter | Description |
|---|---|
| *none* | |