Help Compiler Reference

Coding Help Macros

Authors must follow these rules when coding Help macros:

- Macros are not case-sensitive, so you can type the macro using the capitalization shown in the reference or any other capitalization convention you choose.
- A single macro string may include more than one Help macro, in which case you must use a semicolon to separate each macro in the string.
- Specify empty spaces in a macro string by placing the surrounding text in quotation marks.
- Insert special characters in a quotation-marked string prefaced by a backslash. Special characters include double quotation marks ("), opening and closing single quotation marks (` '), and backslashes (\).

Note The single open quotation mark is different from the single close quotation mark. The single open quotation mark (`) is paired with the tilde (~) above the TAB key on extended keyboards; the single close quotation mark (') is the same as the apostrophe.

- Quotation marks may be either matching double quotation marks, or a pair of single open and close quotation marks. You cannot use double quotation marks inside a string already enclosed in double quotation marks. Use single open and close quotation marks instead. For example,

```
CreateButton("time_btn", "&Time", "ExecProgram("clock", 0)")
```

  is illegal because the string clock uses double quotation marks within the double quotation marks used for the **ExecProgram** macro. The following example corrects the error by enclosing clock in single quotation marks:

```
CreateButton("time_btn", "&Time", "ExecProgram(`clock', 0)")
```

- Macros may be included within other macros. In other words, a macro can be used as a parameter value for another macro.
- A single macro must be 512 or fewer characters in length.

Macro Command Reference

About

This macro displays the About dialog box (same as the About command on the Help menu).

Syntax

About()

Parameters

None

Comments

Use of this macro in secondary windows is discouraged.

AddAccelerator (AA)

See Also

This macro assigns a Help macro to an accelerator key (or key combination) so that the macro is run when the user presses the accelerator key(s).

Syntax

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

Parameters

Argument	**Definition**
key	The Windows virtual-key value.
shift-state	A number specifying the combination of ALT, SHIFT, and CTRL keys used with the accelerator: 0 (none), 1 (SHIFT), 2 (CTRL), 3 (SHIFT+CTRL), 4 (ALT), 5 (ALT+SHIFT), 6 (ALT+CTRL), or 7 (SHIFT+ALT+CTRL)
macro	The Help macro or macro string that is run when the user presses the accelerator key(s). The macro must appear in quotation marks. Multiple macros in a string must be separated by semicolons (;).

Example

The following macro starts the Windows Clock program (provided in Windows version 3.1) when the user presses ALT+SHIFT+CTRL+F4:

```
AddAccelerator(0x73, 7, "ExecProgram(`clock.exe', 1)")
```

Comments

The Help macro that is run by **AddAccelerator** might not work in secondary windows, or its use may be discouraged if the macro it runs is prohibited or discouraged in secondary windows. Check the usage notes for the macro before using **AddAccelerator** to run it in a secondary window.

See Also

Virtual-Key Codes

Annotate

This macro displays the Annotation dialog box (same as the Annotate command on the Edit menu).

Syntax

Annotate()

Parameters

None

Comments

Use of this macro in secondary windows is discouraged.

If the **Annotate** macro is run from a pop-up window, the annotation is attached to the topic that contains the hot spot to the pop-up window.

AppendItem

This macro appends a menu item to the end of a menu you create with the **InsertMenu** macro.

Syntax

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

Parameters

Argument	Definition
menu-id	Name used in the **InsertMenu** macro to create the menu. This name must appear in quotation marks. The new item is appended to this menu.
item-id	Name that WinHelp uses internally to identify the menu item. This name is case-sensitive and must appear in quotation marks. Use this name in the **DisableItem** or **DeleteItem** macro if you want to disable or remove the item, or change the operations that the item performs in certain topics.
item-name	Name that WinHelp displays on the menu for the item. This name is case-sensitive and must appear in quotation marks. Within the quotation marks, place an ampersand (&) before the character used as the macro's keyboard access key.
macro	Help macro or macro string that is run when the user chooses the menu item. The name must appear in quotation marks. Multiple macros in a string must be separated by semicolons (;).

Example

The following macro appends a menu item labeled "Tools" to a menu that has an identifier "mnu_books":

```
AppendItem("mnu_books", "mnu_tools", "&Tools",
➥"JI(`tools.hlp',`first_topic')")
```

Choosing the menu item causes a jump to a topic with the context string "first_topic" in the TOOLS.HLP file.

Comments

WinHelp ignores this macro if it is run in a secondary window.

Make sure that the keyboard access keys you assign to menu items are unique. If you assign a key that conflicts with other menu access keys, WinHelp displays the error message "Unable to add item" and ignores the macro.

Back

This macro displays the previous topic in the Back list. The Back list includes the last 40 topics the user has displayed since starting WinHelp.

Syntax

Back()

Parameters

None

Comments

WinHelp ignores this macro if it is run in a secondary window.

If the **Back** macro is run when the Back list is empty, WinHelp takes no action.

BookmarkDefine

This macro displays the Define dialog box (same as the Define command on the Bookmark menu.

Syntax

BookmarkDefine()

Parameters

None

Comments

Use of this macro in secondary windows is discouraged.

If the **BookmarkDefine** macro is run from a pop-up window, the bookmark is attached to the topic that invoked the pop-up window.

BookmarkMore

This macro displays the More dialog box (same as the More command on the Bookmark menu). The More command appears on the Bookmark menu if the user has defined more than nine bookmarks.

Syntax

BookmarkMore()

Parameters

None

Comments

Use of this macro in secondary windows is discouraged.

BrowseButtons

This macro adds the **<<** Browse button (backward) and the **>>** Browse button (forward) to the button bar in WinHelp.

Syntax

BrowseButtons()

Parameters

None

Example

The following macros in the .HPJ file cause the Clock button to appear immediately before the two Browse buttons on the button bar:

```
CreateButton("&Clock", "ExecProgram(`clock', 0)")
BrowseButtons()
```

Comments

WinHelp ignores this macro if it is run in a secondary window.

If the **BrowseButtons** macro is used with one or more **CreateButton** macros in the [CONFIG] section of the .HPJ file, the order of the Browse buttons on the WinHelp button bar is determined by the order of the **BrowseButtons** macro in relation to the other macros listed in the [CONFIG] section.

Note WinHelp version 3.1 doesn't automatically provide >> Browse (forward) and << Browse (backward) buttons. If your Help file includes one or more browse sequences, you must use the **BrowseButtons()** macro so the user can browse forward or backward.

ChangeButtonBinding (CBB)

This macro assigns a Help macro to a Help button.

Syntax

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

Parameters

Argument	Definition
button-id	Identifier assigned to the button in the **CreateButton** macro or, for a standard Help button, one of the following predefined button identifiers: **btn_contents** (Contents), **btn_search** (Search), **btn_back** (Back), **btn_history** (History), **btn_previous** (<<), or **btn_next** (>>). The button identifier must appear in quotation marks.
button-macro	Help macro run when the user chooses the button. The macro must appear 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 context string "dict_contents") in the DICT.HLP file:

```
ChangeButtonBinding("btn_contents", "JumpId(`dict.hlp',
➥`dict_contents')")
```

Comments

WinHelp ignores this macro if it is run in a secondary window.

ChangeItemBinding (CIB)

This macro assigns a Help macro to an item that you add to a WinHelp menu using the **AppendItem** macro.

Syntax

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

Parameters

Argument	Definition
item-id	Identifier assigned to the item in the **AppendItem** macro. The item identifier must appear in quotation marks.
item-macro	Help macro that is run when the user selects the item. The macro must appear in quotation marks.

Example

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

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

Comments

WinHelp ignores this macro if it is run in a secondary window.

CloseWindow

This macro closes the specified window, which is either the main WinHelp window or a secondary window.

Syntax

CloseWindow*("window-name")*

Parameters

Argument	**Definition**
window-name	The 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 [WINDOWS] section of the .HPJ file. This name must appear in quotation marks.

Example

The following macro closes the secondary window "keys":

```
CloseWindow("keys")
```

Comments

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

Contents

This macro displays the contents topic in the current Help file. The contents topic is defined by the **CONTENTS** option in the [OPTIONS] section of the Help project file.

Syntax

Contents()

Parameters

None

Comments

If the project file does not have a **CONTENTS** option, the contents topic is the first topic in the first topic file specified in the Help project file.

CopyDialog

This macro displays the Copy dialog box (same as the Copy command on the Edit menu).

Syntax

CopyDialog()

Parameters

None

Comments

Use of this macro in secondary windows is discouraged.

CopyTopic

This macro copies all the text in the currently displayed topic onto the Windows Clipboard.

Syntax

CopyTopic()

Parameters

None

Comments

Use of this macro in secondary windows is discouraged.

This macro copies text only; it does not copy bitmaps or any other images in the Help topic.

CreateButton (CB)

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

Syntax

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

Parameters

Argument	Definition
button-id	Name that WinHelp uses internally to identify the button. This name must appear in quotation marks. Use this name in the **DisableButton** or **DestroyButton** macro if you want to remove or disable the button, or in the **ChangeButtonBinding** macro if you want to change the Help macro that the button runs in certain topics.
name	The text that appears on the button. This name must appear in quotation marks. To designate a letter a keyboard access key for this button, place an ampersand (&) before a letter in this text. The button name is case-sensitive and can contain up to 29 characters, beyond which the name is clipped.
macro	Help macro or macro string that is run when the user chooses the button. The macro must appear in quotation marks. Multiple macros in a macro string must be separated by semicolons (;).

Example

The following macro creates a new button labeled "Ideas" that jumps to a topic with the context string "directory" in the IDEAS.HLP file when the button is chosen:

```
CreateButton("btn_ideas", "&Ideas", "JumpId(`ideas.hlp', `directory')")
```

Comments

WinHelp ignores this macro if it is run in a secondary window.

WinHelp allows a maximum of 16 authored buttons. It allows a total of 22 buttons, including the standard Browse buttons, on the button bar.

If the **BrowseButtons** macro is used with one or more **CreateButton** macros in the [CONFIG] section of the .HPJ file, the order of the Browse buttons on the WinHelp button bar is determined by where the **BrowseButtons** macro is listed in relation to the other macros in the [CONFIG] section.

DeleteItem

This macro removes a menu item that was added using the **AppendItem** macro.

Syntax

DeleteItem*("item-id")*

Parameters

Argument	Definition
item-id	The item identifier string used in the **AppendItem** macro. The item identifier must appear in quotation marks.

Example

The following macro removes the menu item "Tools" appended in the example for the **AppendItem** macro:

```
DeleteItem("mnu_tools")
```

Comments

WinHelp ignores this macro if it is run in a secondary window.

DeleteMark

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

Syntax

DeleteMark("*marker-text*")

Parameters

Argument	Definition
marker-text	Marker text specified in the **SaveMark** macro. The marker text must appear in quotation marks.

Example

The following macro removes the marker "Managing Memory" from the Troubleshooting Help file:

```
DeleteMark("Managing Memory")
```

Comments

If the marker does not exist when the **DeleteMark** macro is run, WinHelp displays a "Topic not found" error message.

DestroyButton

This macro removes a button added with the **CreateButton** macro.

Syntax

DestroyButton*("button-id")*

Parameters

Argument	Definition
button-id	Identifier assigned to the button in the **CreateButton** macro. The button identifier must appear in quotation marks. The button idenitifier cannot duplicate an identifier used for one of the standard Help buttons. (See the **ChngeButtonBinding** macro for a list of these identifiers.)

Comments

WinHelp ignores this macro if it is run in a secondary window.

DisableButton (DB)

This macro disables and dims a button added with the **CreateButton** macro.

Syntax

DisableButton*("button-id")*
DB("*button-id*")

Parameters

Argument	Definition
button-id	Identifier assigned to the button in the **CreateButton** macro. The button identifier appears in quotation marks.

Comments

WinHelp ignores this macro if it is run in a secondary window.

A button disabled by the **DisableButton** macro cannot be used in the topic until an **EnableButton** macro is run.

DisableItem (DI)

This macro disables and dims a menu item added with the **AppendItem** macro.

Syntax

DisableItem("*item-id*")
DI("*item-id*")

Parameters

Argument	Definition
item-id	Identifier assigned to the menu item in the **AppendItem** macro. The item identifier must appear in quotation marks.

Comments

WinHelp ignores this macro if it is run in a secondary window.

A menu item disabled by the **DisableItem** macro cannot be used in the topic until an **EnableItem** macro is run.

EnableButton (EB)

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

Syntax

EnableButton*("button-id")*
EB("*button-id*")

Parameters

Argument	**Definition**
button-id	Identifier assigned to the button in the **CreateButton** macro. The button identifier must appear in quotation marks.

Comments

WinHelp ignores this macro if it is run in a secondary window.

EnableItem (EI)

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

Syntax

EnableItem("*item-id*")
EI("*item-id*")

Parameters

Argument	Definition
item-id	Identifier assigned to the menu item in the **AppendItem** macro. The item identifier must appear in quotation marks.

Comments

WinHelp ignores this macro if it is run in a secondary window.

ExecProgram (EP)

This macro runs a Windows - based application.

Syntax

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

Parameters

Argument	Definition
command-line	Command line for the application to be executed. The command line must appear in quotation marks. WinHelp searches for this application in the current directory, followed by the Windows directory, the user's path, and the directory of the currently displayed Help file.
display-state	A value indicating how the application is shown when executed. A value of **0** indicates normal, **1** indicates minimized, and **2** indicates maximized.

Example

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

```
ExecProgram("clock.exe", 0)
```

Exit

This macro exits the WinHelp application (same as the Exit command on the File menu).

Syntax

Exit()

Parameters

None

FileOpen

This macro displays the Open dialog box (same as the Open command on the File menu).

Syntax

FileOpen()

Parameters

None

Comments

Use of this macro in secondary windows is discouraged.

FocusWindow

This macro changes the focus to the specified window, which is either the main WinHelp window or a secondary window.

Syntax

FocusWindow("*window-name*")

Parameters

Argument	**Definition**
window-name	The name of the window to have the focus. The name "main" is reserved for the primary Help window. For secondary windows, the window name is defined in the [WINDOWS] section of the .HPJ file. This name must appear in quotation marks.

Example

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

```
FocusWindow("keys")
```

Comments

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

GotoMark

This macro jumps to a marker set with the **SaveMark** macro.

Syntax

GotoMark(*"marker-text"*)

Parameters

Argument	Definition
marker-text	Marker text specified in the **SaveMark** macro. The marker text must appear in quotation marks.

Example

The following macro jumps to the marker "Managing Memory" in the Troubleshooting Help file:

```
GoToMark("Managing Memory")
```

HelpOn

This macro displays the Using Help file for the WinHelp application (same as the Using Help command on the Help menu).

Syntax

HelpOn()

Parameters

None

History

This macro displays the Windows Help History window, which shows the last 40 topics the user has viewed since opening a Help file in WinHelp. It has the same effect as choosing the History button on the WinHelp button bar.

Syntax

History()

Parameters

None

Comments

WinHelp ignores this macro if it is run in a secondary window.

IfThen

This macro runs a Help macro if a given marker exists. It uses the **IsMark** macro to make the test.

Syntax

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

Parameters

Argument	Definition
marker-text	Marker text tested by the **IsMark** macro. The marker text must appear in quotation marks.
macro	The Help macro or macro string that is run if the marker exists. The macro must appear in quotation marks. Multiple macros in a macro string must be separated by semicolons (;).

Example

The following macro jumps to the topic with the context string "man_mem" if a marker named "Managing Memory" has been set by the **SaveMark** macro:

```
IfThen(IsMark("Managing Memory"), "JI(`trb.hlp', `man_mem')")
```

IfThenElse

This macro runs one of two Help macros, provided a marker exists. It uses the **IsMark** macro to make the test.

Syntax

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

Parameters

Argument	Definition
marker-text	Marker text tested by the **IsMark** macro. The marker text must appear in quotation marks.
macro1, etc.	WinHelp runs *macro1* if the marker exists and *macro2* if it does not. Both macros must appear in quotation marks. Multiple macros in either macro string must be separated by semicolons (;).

Example

The following macro jumps to the topic with the context string "man_mem" if a marker named "Managing Memory" has been set by the **SaveMark** macro. If the marker does not exist, it jumps to the contents screen for the TRB.HLP file:

```
IfThenElse(IsMark("Managing Memory"), "JI(`trb.hlp', `man_mem')",
"JumpContents(`TRB.HLP')")
```

InsertItem

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

Syntax

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

Parameters

Argument	Definition
menu-id	Either a standard WinHelp menu name or the name used in the **InsertMenu** macro to create the menu. Standard menu names are **mnu_file** (File menu), **mnu_edit** (Edit menu), **mnu_bookmark** (Bookmark menu), and **mnu_helpon** (Help menu). The menu identifier must appear in quotation marks. The new item is inserted into this menu.
item-id	Name that WinHelp uses internally to identify the menu item. The item identifier must appear in quotation marks. Use this name in the **DisableItem** or **DeleteItem** macro if you want to remove or disable the item, or change the operations that the item performs in certain topics.
item-name	Name WinHelp displays on the menu for the item. This name is case-sensitive and must appear in quotation marks. Within the quotation marks, place an ampersand (&) before the character used for the item's keyboard access key.
macro	Help macro or macro string that is run when the user chooses the menu item. The macro must appear in quotation marks. Multiple macros in a string must be separated by semicolons (;).
position	An integer specifying the position in the menu where the new item will appear. Position 0 is the first or topmost position in the menu.

Example

The following macro inserts a menu item labeled "Tools" as the third item on a menu that has an identifier "mnu_books":

```
InsertItem("mnu_books", "mnu_tools", "&Tools", "JI(`tools.hlp',
`first_topic')", 3)
```

Selecting the menu item causes a jump to a topic with the context string "first_topic" in the TOOLS.HLP file.

Comments

WinHelp ignores this macro if it is run in a secondary window.

Make sure that the keyboard access keys you assign to menu items are unique. If you assign a key that conflicts with other menu access keys, WinHelp displays the error message "Unable to add item" and ignores the macro.

InsertMenu

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

Syntax

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

Parameters

Argument	Definition
menu-id	Name that WinHelp uses internally to identify the menu. The menu identifier must appear in quotation marks. Use this identifier in the **AppendItem** macro to add commands to the menu.
menu-name	Name for the menu that WinHelp displays on the menu bar. This name is case-sensitive and must appear in quotation marks. Within the quotation marks, place an ampersand (&) before the character used for the menu's keyboard access key.
menu-position	Number telling WinHelp which position on the menu bar the new menu name will have. Positions are numbered from left to right, with position 0 being the leftmost menu.

Example

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

```
InsertMenu("menu_util", "&Utilities", 3)
```

The label "Utilities" appears as the fourth menu on the WinHelp menu bar. The user presses "ALT+U" to display the menu and its commands.

Comments

WinHelp ignores this macro if it is run in a secondary window.

Make sure that the keyboard access keys you assign to menus are unique. If you assign a key that conflicts with other menu access keys, WinHelp displays the error message "Unable to add menu" and ignores the macro.

IsMark

This macro determines whether a marker set by the **SaveMark** macro exists. It is used as a parameter to the conditional macros **IfThen** and **IfThenElse**.

Syntax

IsMark*("marker-text")*

Parameters

Argument	Definition
marker-text	Marker text tested by the **IsMark** macro. The **IsMark** macro returns a **True** value if the mark exists and a **False** value if it does not. The marker text must appear in quotation marks.

Example

The following macro jumps to the topic with the context string "man_mem" if a marker named "Managing Memory" has been set by the **SaveMark** macro:

```
IfThen(IsMark("Managing Memory"), "JI(`trb.hlp', `man_mem')")
```

Comments

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

JumpContents

This macro jumps to the contents topic of a specified Help file. The contents topic is indicated by the **CONTENTS** option entry in the [OPTIONS] section of the .HPJ file.

Syntax

JumpContents*("filename")*

Parameters

Argument	Definition
filename	The name of the destination file for the jump. The file name must appear in quotation marks. If WinHelp cannot find this file, it displays an error message and does not perform the jump.

Example

The following macro jumps to the contents topic of the PROGMAN.HLP file:

```
JumpContents("PROGMAN.HLP")
```

Comments

If the **CONTENTS** option is not specified, WinHelp jumps to the first topic in the Help file.

WinHelp ignores this macro if it is run in a secondary window.

JumpContext (JC)

This macro jumps to a topic identified by a context number. The context is identified by an entry in the [MAP] section of the .HPJ file.

Syntax

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

Parameters

Argument	Definition
filename	The name of the destination file for the jump. The file name must appear in quotation marks. If WinHelp cannot find this file, it displays an error message and does not perform the jump.
context number	Context number of the topic in the destination file. The context number must be mapped in the destination Help file's [MAP] section. 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 file instead, and displays an error message. (For more information about context numbers, see "Creating Context Numbers" in the *Help Compiler Guide,*Chapter 2, "Planning the Help System.")

Example

The following macro jumps to the topic mapped to the context number 801 in the PROGMAN.HLP file:

```
JumpContext("PROGMAN.HLP", 801)
```

JumpHelpOn

This macro jumps to the contents topic of the Using Help file. The Using Help file is either the default WINHELP.HLP file or the Help file designated by the **SetHelpOnFile** macro in the [CONFIG] section of the .HPJ file. (For more information, see the **SetHelpOnFile** macro later in this section.)

Syntax

JumpHelpOn()

Parameters

None

Example

The following macro jumps to the contents topic of the designated Using Help file:

```
JumpHelpOn()
```

Comments

If WinHelp cannot find the specified Help file, it displays an error message and does not perform the jump.

JumpId (JI)

This macro jumps to the topic with the specified context string in the Help file.

Syntax

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

Parameters

Argument	Definition
filename	Name of the Help file (.HLP) containing the context string. The file name must appear in quotation marks. If WinHelp does not find this file, it displays an error message and does not perform the jump.
context-string	Context string of the topic in the destination file. The context string must appear in quotation marks. If the context string does not exist, WinHelp jumps to the contents topic for that file instead.

Example

The following macro jumps to a topic with "second_topic" as its context string in the Help file SECOND.HLP:

```
JumpId("second.hlp", "second_topic")
```

JumpKeyword (JK)

This macro opens the indicated Help file (.HLP), searches through the K keyword table, and displays the first topic containing the keyword specified in the macro.

Syntax

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

Parameters

Argument	Definition
filename	The name of the .HLP file that contains the desired keyword table. The file name must appear in quotation marks. If this file does not exist, WinHelp displays an error message and does not perform the jump.
keyword	The keyword that the macro searches for. The keyword must appear in quotation marks. If WinHelp finds more than one match, it displays the first matched topic. If it does not find any matches, it displays a "Not a keyword" message and the contents topic of the destination file.

Example

The following macro opens the first topic that has "hands" as an index keyword in the Help file CLOCK.HLP:

```
JumpKeyword("clock.hlp", "hands")
```

Next

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

Syntax

Next()

Parameters

None

Comments

If the current topic is the last of a browse sequence, this macro does nothing.

WinHelp ignores this macro if it is run in a secondary window.

Not

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

Syntax

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

Parameters

Argument	Definition
marker-text	Marker text tested by the **IsMark** macro. The **Not** macro returns a **False** value if the mark exists or a **True** value if it does not. The marker text must appear in quotation marks.

Example

The following macro jumps to the topic with context string "Expanded Memory" if a marker named "Managing Memory" has not been set by the **SaveMark** macro:

```
IfThen(Not(IsMark("Managing Memory")), "JI(`trb.hlp', `Expanded Memory')")
```

PopupContext (PC)

This macro displays a topic identified by a context number. The context is identified by an entry in the [MAP] section of the .HPJ file.

Syntax

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

Parameters

Argument	**Definition**
filename	The name of the file that contains the topic to be displayed in the pop-up window. The file name must appear in quotation marks. If WinHelp cannot find this file, it displays an error message.
context number	Context number of the topic to be displayed in the pop-up window. The context number must be mapped in the [MAP] section of the specified Help file. 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 file instead. (For more information about context numbers, see "Mapping Context-Sensitive Topics" in the *Help Compiler Guide,*Chapter 7, "Building the Help System.")

Example

The following macro displays in a pop-up window the topic mapped to the context number 801 in the file PROGMAN.HLP:

```
PopupContext("progman.hlp", 801)
```

PopupId (PI)

This macro displays a topic from a specified file in a pop-up window.

Syntax

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

Parameters

Argument	Definition
filename	The name of the file that contains the pop-up window topic. The file name must appear in quotation marks. If this file does not exist, WinHelp displays an error message.
context-string	Context string of the topic in the destination file. The context string must appear in quotation marks. If the requested context string does not exist, WinHelp displays the contents topic or the first topic in the file in the pop-up window. (For more information, see "Coding Context Strings" in the *Help Compiler Guide,*Chapter 4, "Creating the Topic Files.")

Example

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

```
PopupId("second.hlp", "second_topic")
```

PositionWindow (PW)

This macro sets the size and position of the main Help window or an existing secondary window.

Syntax

PositionWindow*(x, y, width, height, window-state,"window-name")*
PW*(x, y, width, height, window-state, "window-name")*

Parameters

Argument	Definition
x, y	X and Y coordinates of the upper-left window corner. Positions are defined in terms of WinHelp's 1024-by-1024 coordinate system.
width, height	Gives the default width and height of the window. Window sizes, like positions, are defined in terms of WinHelp's coordinate system.
window-state	Specifies how the window is sized. This parameter is **0** for normal size and **1** for maximized. If the parameter is **1**, WinHelp ignores the *x*, *y, width*, and *height* parameters.
window-name	The name of the window to position. The name "main" is reserved for the primary Help window. For secondary windows, the window name is defined in the [WINDOWS] section of the .HPJ file. This name must appear in quotation marks.

Example

The following macro positions the secondary window "Samples" in the upper-left corner (100, 100) with a width and height of 500 (in WinHelp coordinates):

```
PositionWindow(100, 100, 500, 500, 0, "Samples")
```

Comments

If the window to be positioned does not exist, WinHelp ignores the macro.

Prev

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

Syntax

Prev()

Parameters

None

Comments

If the currently displayed topic is the first topic of a browse sequence, this macro does nothing.

WinHelp ignores this macro if it is run in a secondary window.

Print

This macro sends the currently displayed topic to the printer.

Syntax

Print()

Parameters

None

Comments

This macro should be used only to print topics in windows other than the main Help window. For example, it can be used to print topics displayed in secondary windows, provided the user doesn't have a dialog box open at the time of printing.

Use of this macro in secondary windows is discouraged.

PrinterSetup

This macro displays the Print Setup dialog box (same as the Print Setup command on the File menu).

Syntax

PrinterSetup()

Parameters

None

Comments

Use of this macro in secondary windows is discouraged.

RegisterRoutine (RR)

This macro registers a function within a DLL as a Help macro. Registered functions can be used in macro hot spots or footnotes within topic files, or in the [CONFIG] section of the .HPJ file, just as standard Help macros are used.

Syntax

RegisterRoutine*("DLL-name", "function-name", "format-spec")*
RR*("DLL-name", "function-name", "format-spec")*

Parameters

Argument	Definition
DLL-name	The file name of the DLL being called. The file name must appear in quotation marks. If WinHelp cannot find the DLL, it displays an error message and does not perform the call.
function-name	The name of the function to be executed in the designated DLL. The function name must appear in quotation marks.
format-spec	A string specifying the formats of parameters passed to the function. The format string must appear in quotation marks. Characters in the string represent C parameter types: "u" for **unsigned short**, "U" for **unsigned long**, "i" for **short int**, "I" for **long int**, "s" for string (**near char ***), "S" for string (**far char ***), or "v" for **void**. WinHelp automatically makes sure these formats match the parameter types specified in the function prototype.

Example

The following DLL call registers a routine "RetString" in the DLL named HELPLIB.DLL. RetString takes arguments of types **far char ***, **short int**, and **unsigned long**.

```
RegisterRoutine("HELPLIB", "RetString", "S=iU")
```

SaveMark

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

Syntax

SaveMark(*"marker-text"*)

Parameters

Argument	**Definition**
marker-text	Text used to identify the topic location. The marker text must appear in quotation marks, and it must be unique. If the same text is used for more than one marker, WinHelp recognizes only the most recently entered marker.

Example

The following macro saves the marker "Managing Memory" in the current topic in the Troubleshooting Help file:

```
SaveMark("Managing Memory")
```

Comments

In addition to **GotoMark**, WinHelp offers the following other macros for use with text markers:

- **DeleteMark** removes any defined marker.
- **IsMark** tests whether a given marker has been set in the Help file. **Not** negates the result of this test.
- **IfThen** and **IfThenElse** run one or more Help macros if a given marker has been set. These use the **IsMark** (and optional **Not**) macro to test whether the marker is set.

Text markers are not saved if the user exits and then restarts WinHelp.

Search

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

Syntax

Search()

Parameters

None

Comments

WinHelp ignores this macro if it is run in a secondary window.

SetContents

This macro designates a specific topic as the contents topic within the Help file.

Syntax

SetContents*("filename", context number)*

Parameters

Argument	Definition
filename	The name of the Help file that contains the desired contents topic. The file name must appear in quotation marks. If WinHelp cannot find the file, it displays an error message and does not perform the jump.
context number	Context number of the topic in the specified file. The context number must be mapped in the [MAP] section of the destination Help file. If the context number does not exist or cannot be found in the [MAP] section, WinHelp displays an error message.

Example

The following macro sets the topic mapped to the context number 801 in the PROGMAN.HLP file as the contents topic:

```
SetContents("PROGMAN.HLP", 801)
```

After running this macro, pressing the Contents button causes a jump to the specified topic.

SetHelpOnFile

This macro designates the specific Help file that replaces WINHELP.HLP, the default Using Help file in the Windows environment.

Syntax

SetHelpOnFile*("filename")*

Parameters

Argument	Definition
filename	The name of the replacement Using Help file. The file name must appear in quotation marks. If WinHelp cannot find this file, it displays an error message.

Example

The following macro sets the Using Help file as MYHELP.HLP:

```
SetHelpOnFile("myhelp.hlp")
```

Comments

If this macro appears within a topic in the Help file, the replacement file is set after execution of the macro. If this macro appears in the [CONFIG] section of the .HPJ file, the replacement file is set when the Help file is opened.

Virtual-Key Codes

The following list shows the symbolic constant names, hexadecimal values, and descriptive information for Microsoft Windows virtual-key codes. Virtual-key codes are used to assign an accelerator key or key combination to a macro (For more information on assigning accelerator keys or key combinations, see the **AddAccelerator** macro earlier in this chapter). The codes are listed in numeric order.

Name	Value	Description
VK_CANCEL	03H	Used for control-break processing
+	05H-07H	Undefined
VK_BACK	08H	BACKSPACE key
VK_TAB	09H	TAB key
+	0AH-0BH	Undefined
VK_CLEAR	0CH	CLEAR key
VK_RETURN	0DH	RETURN key
VK_SHIFT	10H	SHIFT key
VK_CONTROL	11H	CTRL key
VK_MENU	12H	ALT key
VK_PAUSE	13H	PAUSE key
VK_CAPITAL	14H	CAPITAL key
+	15H-19H	Reserved for Kanji systems
+	1AH	Undefined
VK_ESCAPE	1BH	ESC key
+	1CH-1FH	Reserved for Kanji systems
VK_SPACE	20H	SPACEBAR
VK_PRIOR	21H	PGUP key
VK_NEXT	22H	PGDN key
VK_END	23H	END key
VK_HOME	24H	HOME key
VK_LEFT	25H	LEFT ARROW key
VK_UP	26H	UP ARROW key
VK_RIGHT	27H	RIGHT ARROW key
VK_DOWN	28H	DOWN ARROW key
VK_SELECT	29H	SELECT key
+	2AH	OEM specific
VK_EXECUTE	2BH	EXECUTE key
VK_SNAPSHOT	2CH	PRINT SCREEN key for Windows version 3.0 and later
VK_INSERT	2DH	INS key
VK_DELETE	2EH	DEL key
VK_HELP	2FH	HELP key
VK_0	30H	0 key
VK_1	31H	1 key
VK_2	32H	2 key
VK_3	33H	3 key
VK_4	34H	4 key
VK_5	35H	5 key
VK_6	36H	6 key
VK_7	37H	7 key

VK_8	38H	8 key
VK_9	39H	9 key
+	3AH-40H	Undefined
VK_A	41H	A key
VK_B	42H	B key
VK_C	43H	C key
VK_D	44H	D key
VK_E	45H	E key
VK_F	46H	F key
VK_G	47H	G key
VK_H	48H	H key
VK_I	49H	I key
VK_J	4AH	J key
VK_K	4BH	K key
VK_L	4CH	L key
VK_M	4DH	M key
VK_N	4EH	N key
VK_O	4FH	O key
VK_P	50H	P key
VK_Q	51H	Q key
VK_R	52H	R key
VK_S	53H	S key
VK_T	54H	T key
VK_U	55H	U key
VK_V	56H	V key
VK_W	57H	W key
VK_X	58H	X key
VK_Y	59H	Y key
VK_Z	5AH	Z key
+	5BH-5FH	Undefined
VK_NUMPAD0	60H	Numeric key pad 0 key
VK_NUMPAD1	61H	Numeric key pad 1 key
VK_NUMPAD2	62H	Numeric key pad 2 key
VK_NUMPAD3	63H	Numeric key pad 3 key
VK_NUMPAD4	64H	Numeric key pad 4 key
VK_NUMPAD5	65H	Numeric key pad 5 key
VK_NUMPAD6	66H	Numeric key pad 6 key
VK_NUMPAD7	67H	Numeric key pad 7 key
VK_NUMPAD8	68H	Numeric key pad 8 key
VK_NUMPAD9	69H	Numeric key pad 9 key
VK_MULTIPLY	6AH	Multiply key
VK_ADD	6BH	Add key
VK_SEPARATER	6CH	Separater key
VK_SUBTRACT	6DH	Subtract key
VK_DECIMAL	6EH	Decimal key
VK_DIVIDE	6FH	Divide key
VK_F1	70H	F1 key
VK_F2	71H	F2 key

VK_F3	72H	F3 key
VK_F4	73H	F4 key
VK_F5	74H	F5 key
VK_F6	75H	F6 key
VK_F7	76H	F7 key
VK_F8	77H	F8 key
VK_F9	78H	F9 key
VK_F10	79H	F10 key
VK_F11	7AH	F11 key
VK_F12	7BH	F12 key
VK_F13	7CH	F13 key
VK_F14	7DH	F14 key
VK_F15	7EH	F15 key
VK_F16	7FH	F16 key
+	80H-87H	OEM specific
+	88H-8FH	Unassigned
VK_NUMLOCK	90H	NUM LOCK key
+	91H	OEM specific
+	92H-B9H	Unassigned
+	BAH-C0H	OEM specific
+	C1H-DAH	Unassigned
+	DBH-E4H	OEM specific
+	E5H	Unassigned
+	E6H	OEM specific
+	E7H-E8H	Unassigned
+	E9H-F5H	OEM specific
+	F6H-FEH	Unassigned

Interpreting Error Messages

The topic number given with an error message refers to the sequential position of that topic in the topic file (first topic, second topic, and so on). Remember that topics are separated by hard page breaks, even though there is no such thing as a "page" in the Help system.

Messages beginning with the word **Error** may indicate fatal errors. Fatal errors are always reported, since no usable Help file will result from the build. Messages beginning with the word **Warning** are less serious in nature. A build with warnings produces a valid Help file that WinHelp can open, but the file may contain operational errors. You specify the amount of warning information to be reported by the Help Compiler using the **WARNING** option.

During processing of the .HPJ file, the Help Compiler ignores lines that contain errors and attempts to continue with the build. This means that errors encountered early in a build may result in many more errors being reported as the build continues.

Similarly, errors encountered during the processing of the RTF topic files are reported, but if the errors are not serious, the Help Compiler continues with the build.

Note One easy way to avoid build errors is to make sure that the Help Compiler can access all the topic files and graphic files needed to build the Help file.

Warning Message Reporting

You can specify the level of warnings reported by the Help Compiler during the build process. Warning messages alert you to conditions that are not serious enough to stop the build but that might cause problems in your Help file.

The **WARNING** option in your .HPJ file sets the amount of warning information that the Help Compiler displays. You set the warning level under the [OPTIONS] section, using the following command:

WARNING = *level*

The following table describes each of the three possible reporting levels.

Warning level	Information reported
1	The build program reports only the most severe warnings.
2	The build program reports severe and less serious warnings.
3	The build program reports all warnings.

Help Compiler Error Messages

The Help Compiler displays a message when it encounters an error in building the Help file. Whenever possible, the Help Compiler displays the name of the topic file that contains the error, as well as the number used to identify the specific line of the Help Project (.HPJ) file or the topic that produced the error. Error message help is organized into the following groups:

Message numbers	**Type of errors**	**Message numbers**	**Type of errors**
1019–1536	Source file errors	2771–2932	Other options errors
2010	Project file errors	3011–3178	Build tag footnote and expression errors
2030–2214	Syntax errors	3511–3652	Macro errors
2273–2331	General section errors	4011–4196	Context string errors
2341–2372	ALIAS and MAP section errors	4211–4312	Footnote errors
2391–2501	WINDOWS section errors	4331–4393	Topic title errors
2511–2532	OPTIONS section errors	4412–4452	Keyword errors
2550–2570	Root option errors	4471–4492	Build tag errors
2591–2632	Font range option errors	4551	Entry macro errors
2651–2672	Forcefont errors	4616–4813	Topic file errors
2691–2752	Multikey errors	5035–5115	Miscellaneous errors

Note You might encounter errors other than the build errors described here. See the README.TXT file for more information.

File Errors

The following messages result from problems with the files used to build a Help file.

1019	**Project file extension cannot be .HLP or .PH.** You cannot specify a project file with an .HLP or .PH extension. Project files must use the .HPJ extension. Rename the Help project file and then recompile.
1030	**File name exceeds limit of 259 characters.** The combined length of the path and file name is more than the MS-DOS limit of 259 characters. Shorten the path and then recompile.
1079	**Out of file handles.** The Help Compiler does not have enough available file handles to continue the build.
1100	**Cannot open file *filename*: permission denied.** You do not have the required file privileges to open the requested file.
1150	**Cannot overwrite file *filename*.** The Help file cannot overwrite the specified file because the file has a read-only attribute. Rename the Help project file or change the read-only attribute.
1170	**File *filename* is a directory.** A subdirectory in the Help project root directory has the same name as the requested Help file. This is a MS-DOS file error. Move or rename the subdirectory and then recompile.
1190	**Cannot use reserved MS-DOS file *filename*.** A file has been referred to by a reserved MS-DOS name such as COM1, LPT2, or PRN. This is a MS-DOS file error. Rename the file and then recompile.
1230	**File *filename* not found.** The specified file could not be found or is unreadable. This is a MS-DOS file error or an out-of-memory condition. Check to see if the file exists and also check the amount of available memory.
1292	**File *filename* is not a valid bitmap.** The specified bitmap file could not be found or is not in a recognizable bitmap format. This is a MS-DOS file error or an out-of-memory condition. Check to see if the file exists, and if it does, check its format. If necessary, save the file again in your paint or draw program, and then recompile.
1319	**Disk full.** The Help file could not be written to disk. Create more space on the destination disk and then recompile.
1513	**Bitmap name *filename* duplicated.** The [BITMAPS] section contains duplicate bitmap names. The Help Compiler uses the first occurrence of the name. Rename the duplicate bitmap file names and then recompile.
1536	**Not enough memory to check and compress bitmap *filename*.** The specified bitmaps cannot be compressed because of insufficient memory. If any of the specified bitmaps are segmented hypergraphics, the context strings stored in them are not checked for validity during the build.

Project File Errors

The following messages result from errors in the .HPJ file used to build a Help file.

2010	**Include statements nested more than 5 deep.**
	The **#include** statement on the specified line has exceeded the maximum of five include levels.
	Do not nest **#include** statements more than 5 deep.

Syntax Errors

The following messages result from syntax errors in the Help project file.

2030 **Comment starting at line *linenumber* of file *filename* unclosed at end of file.**

The Help Compiler has unexpectedly come to the end of the Help project file. There may be an open comment in the .HPJ file or in an included file.

2050 **Invalid #include syntax.**

The correct **#include** syntax is as follows:

#include <*filename*>

Correct the syntax and then recompile.

2091 **Bracket missing from section heading [*sectionname*].**

The right bracket (]) is missing from the specified section heading.

Insert the bracket and then recompile.

2111 **Section heading missing.**

The section heading on the specified line is not complete. This error is also reported if the first entry in the Help project file is not a section heading.

2131 **Invalid OPTIONS syntax: option=value' expected.**

Check the syntax of the options in the [OPTIONS] section.

2141 **Invalid ALIAS syntax: context=context' expected.**

Check the syntax of the entries in the [ALIAS] section.

2151 **Incomplete line in [*sectionname*] section*.***

The entry on the specified line is incomplete.

The Help Compiler skips the line.

2171 **Unrecognized text.**

There is unrecognizable text following valid text.

The Help Compiler ignores the line.

2191 **Section heading [*sectionname*] unrecognized.**

A section heading that is not supported by the Help Compiler has been used.

The Help Compiler ignores the line.

2214 **Line in .HPJ file exceeds length limit of 2047 characters.**

There is a line in the .HPJ file that exceeds the maximum length of 2047 characters.

General Section Errors

The following messages result from general errors in the different sections of the Help project file.

2273 **[OPTIONS] should precede [FILES] and [BITMAPS] for all options to take effect.**

It is recommended that the [OPTIONS] section be the first section of the .HPJ file so that all the options will take effect. Also, if the **ERRORLOG** option is used, it should be the first line in the [OPTIONS] section.

2291 **Section *sectionname* previously defined.**

A duplicate section has been found in the Help project file.

The Help Compiler ignores the lines under the duplicated section and continues from the next valid section heading.

2305 **No valid files in [FILES] section.**

The file section is either empty or contains only invalid files.

2322 **Context string *context_name* cannot be used as alias string.**

A context string that has been assigned an alias cannot be used later as an alias for another context string. That is, you cannot map a = b and then c = a in the [ALIAS] section.

The Help Compiler ignores the attempted reassignment on this line.

2331 **Context number already used in [MAP] section.**

The context number on the specified line in the Help project file was previously mapped to a different context string.

The Help Compiler ignores the line.

Alias and Map Section Errors

The following messages result from errors in the [ALIAS] or [MAP] sections of the Help project file.

2341	**Invalid or missing context string.**
	The specified line is missing a context string before an equal sign.
2351	**Invalid context identification number.**
	The context number on the specified line is empty or contains invalid characters.
2362	**Context string *context_name* already assigned an alias.**
	A context string can only have one alias. That is, you cannot map a = b and then a = c in the [ALIAS] section. The specified context string has already been assigned an alias in the [ALIAS] section.
	The Help Compiler ignores the attempted reassignment on this line.
2372	**Alias string *aliasname* already assigned.**
	You can't alias an alias. An alias string cannot, in turn, be assigned another alias. That is, you cannot map a = b and then b = c in the [ALIAS] section.
	The Help Compiler ignores the attempted reassignment on this line.

Windows Section Errors

The following messages result from errors in the definitions of the types of secondary windows given in the [WINDOWS] section of the Help project file.

2391	**Limit of 6 window definitions exceeded.** The maximum number of window definitions is one main window definition and five secondary window definitions.
2401	**Window maximization state must be 0 or 1.** The value of the *fMax* variable in the window attribute specification in the .HPJ file is something other than **0** or **1**. Correct the entry and then recompile.
2411	**Invalid syntax in window color.** The correct syntax for the text and background color is: *.mono (rrr, ggg, bbb).endmono.* Correct the syntax and then recompile. For information on setting the window color, please see "Defining Window Attributes" in the *Help Compiler Guide,*Chapter 7, "Building the Help Files."
2421	**Invalid window position.** The correct syntax to indicate the predefined window position is: *.mono (x, y, DX, dy).endmono.* The Help Compiler ignores this line, and the window does not have a predefined position. For more information, please see "Defining Window Attributes" in the *Help Compiler Guide,*Chapter 7, "Building the Help Files."
2431	**Missing quote in window caption.** The value of the caption attribute for the window definition in the .HPJ file is not enclosed in quotation marks. Correct the syntax and then recompile.
2441	**Window name *windowname* is too long.** The window name exceeds the maximum length of 8 characters.
2451	**Window position value out of range 01023.** One or more of the window position coordinates (X, Y, X + dX, or [Y+dY]) exceed the maximum position value of 1023.
2461	**Window name missing.** The window specification in the .HPJ file is missing the window name.
2471	**Invalid syntax in [WINDOWS] section.** The entry for a main or secondary window is incorrect. The Help Compiler ignores the window entry. Check the window entry syntax and then recompile.
2481	**Secondary window position required.** The X, Y, dX, and dY entries for the secondary window definitions must be specified in the .HPJ file.
2491	**Duplicate window name *windowname*.** There are duplicate window names in the .HPJ file. Check the uniqueness of each member name and then recompile.
2501	**Window caption *windowcaption* exceeds limit of 50 characters.** The caption for the window exceeds the limit of 50 characters.

Options Section Errors

The following error messages are caused by problems in the [OPTIONS] section of the Help project file.

2511	**Unrecognized option *optionname* in [OPTIONS] section.** An option has been used that is not supported by the compiler. The Help Compiler skips this line.
2532	**Option *optionname* previously defined.** The specified option has been defined on a previous line. The Help Compiler ignores the attempted redefinition.

ROOT Option Errors

The following error messages are caused by problems with the **ROOT** option in the [OPTIONS] section of the Help project file.

2550	**Invalid path *pathname* in *optionname* option.** The Help Compiler cannot find the path specified by the **ROOT** option. The Help Compiler uses the current working directory.
2570	**Path in *optionname* option exceeds number of characters.** The path specified by the **ROOT** option exceeds the MS-DOS maximum limit. The Help Compiler ignores the path and uses the current working directory.

Font Range Option Errors

The following error messages are caused by problems with the **MAPFONTSIZE** option in the [OPTIONS] section of the Help project file.

2591	**Invalid MAPFONTSIZE option.** The font range syntax is invalid. The correct syntax is *m*[-*n*]:*p*.
2612	**Maximum of 5 font ranges exceeded.** The maximum number of font ranges that can be specified is five. The Help Compiler ignores any additional ranges.
2632	**Current font range overlaps previously defined range.** A font size range overlaps a previously defined mapping. The Help Compiler ignores the second mapping. Adjust one or both of the font ranges to remove any overlaps.

FORCEFONT Option Errors

The following error messages are caused by problems with the **FORCEFONT** option in the [OPTIONS] section of the Help project file.

2651	**Font name exceeds limit of 20 characters.**
	Font names cannot exceed 20 characters.
	The Help Compiler ignores this line.
2672	**Unrecognized font name *fontnam*e in FORCEFONT option.**
	The Help Compiler has encountered a font name that it does not support.
	The Help Compiler ignores the font name and uses the default Helvetica font.

MULTIKEY Option Errors

The following error messages are caused by problems with the **MULTIKEY** option in the [OPTIONS] section of the Help project file.

2691	**Invalid MULTIKEY syntax.**
	The Help Compiler does not recognize the syntax used in a **MULTIKEY** option. The valid syntax is **MULTIKEY** = *char*, where *char* is any capital letter other than "K."
2711	**Maximum of 5 keyword tables exceeded.**
	The limit of five keyword tables has been exceeded.
	The Help Compiler ignores the additional tables.
	Reduce the number of tables and then recompile.
2732	**Character already used.**
	A character used for indicating the keyword table (**MULTIKEY** = *char*) was previously used.
	The Help Compiler ignores the entry.
2752	**Characters K' and k' cannot be used.**
	These characters are reserved for Help's normal keyword table.
	Choose another character, and then recompile.

Other Options Errors

The following error messages are caused by problems with other options in the [OPTIONS] section of the Help project file.

2771	**REPORT option must be ON' or OFF'.**
	The **REPORT** option must be either **True**, **1, ON**, **YES**, **False**, **0**, **OFF**, or **NO**.
	Correct the entry and then recompile.
2811	**OLDKEYPHRASE option must be ON' or OFF'.**
	The **OLDKEYPHRASE** option must be either **True**, **1, ON**, **YES**, **False**, **0**, **OFF**, or **NO**.
	Correct the entry and then recompile.
2832	**COMPRESS option must be OFF', MEDIUM' or HIGH'.**
	The **COMPRESS** option must be either **1, YES, ON, True, HIGH, MEDIUM, 0, NO, OFF,** or **False.**
	Correct the entry and then recompile.
2842	**OPTCDROM option must be TRUE' or FALSE'.**
	The **OPTCDROM** option must be either **True** or **False**.
	The Help Compiler defaults to **False**.
	Correct the entry and then recompile.
2852	**Invalid TITLE option.**
	The **TITLE** option defines a string that is empty or contains more than 32 characters.
	The Help Compiler truncates the title.
2872	**Invalid LANGUAGE option.**
	You have specified an ordering that is not supported by the compiler.
	The Help Compiler defaults to U.S. sort ordering.
2893	**Warning option must be 1, 2, or 3.**
	The warning reporting level can be set only to **1**, **2**, or **3**.
	The Help Compiler defaults to full reporting (level 3).
2911	**Invalid icon file *filename*.**
	The Help Compiler cannot find the icon file specified in the **ICON** option, or the file is not a valid icon file.
2932	**Copyright string exceeds limit of 50 characters.**
	The maximum length of the copyright string in the About box is 50 characters.
	The Help Compiler truncates the string.

BUILDTAG Footnote and Expression Errors

The following messages are caused by errors in build-tag footnotes or build expressions in the [BUILDTAGS] section of the Help project file.

3011	**Maximum of 32 build tags exceeded.** The maximum number of build tags that can be defined is 32. The Help Compiler ignores the additional tags.
3031	**Build tag length exceeds 32 characters.** The build tag on the specified line exceeds the maximum of 32 characters. The Help Compiler skips this entry.
3051	**Build tag *tagname* contains invalid characters.** Build tags can contain only alphanumeric characters or the underscore (_) character. The Help Compiler skips this line.
3076	**[BUILDTAGS] section missing.** The **BUILD** option declared a conditional build, but there is no [BUILDTAGS] section in the Help project file. The Help Compiler includes all topics in the build.
3096	**Build expression too complex.** The build expression has too many expressions ("~", "|" or "&") or is too deeply nested.
3116	**Invalid build expression.** The syntax used on the specified line of the build expression contains one or more logical or syntax errors.
3133	**Duplicate build tag in [BUILDTAGS] section.** A build tag in the [BUILDTAGS] section has been repeated unnecessarily.
3152	**Build tag *tagname* not defined in [BUILDTAGS] section.** The specified build tag has been assigned to a topic, but not declared in the Help project file. The Help Compiler ignores the tag for the topic.
3178	**Build expression missing from project file.** The topics have build tags, but there is no **BUILD** = *expression* in the .HPJ file. The Help Compiler includes all topics in the build.

Macro Errors

The following messages result from errors in the use of Help macros in footnotes, hot spots, and the [CONFIG] section of the Help project file.

3511	**Macro *macrostring* exceeds limit of 254 characters.** The macro string exceeds the maximum limit of 254 characters.
3532	**Undefined function in macro *macroname*.** The specified macro is not on the list of macros supported by the compiler, nor is it specified in the **RegisterRoutine()**. The Help Compiler nonetheless passes the macro to the .HLP file.
3552	**Undefined variable in macro *macroname*.** The macro contains a variable that is not recognized by the compiler.
3571	**Wrong number of parameters to function in macro *macroname*.** There are too many or too few parameters in the macro.
3591	**Syntax error in macro *macroname*.** The syntax of the macro is invalid.
3611	**Function parameter type mismatch in macro *macroname*.** There is a type mismatch (string or numeric) in the function call.
3631	**Bad macro prototype.** The prototype string passed to **RegisterRoutine** is invalid.
3652	**Empty macro string.** The "!" footnote or a hidden text starting with "!" does not contain a macro.

Context String Errors

The following messages are caused by problems with context string footnotes or context strings specified in jumps or project file options.

4011	**Context string *contextname* already used.** The specified context string was previously assigned to another topic. The Help Compiler ignores the latter string, and the topic has no identifier.
4031	**Invalid context string *contextname*.** The context string footnote contains non-alphanumeric characters or is empty. The Help Compiler does not assign the topic an identifier.
4056	**Unresolved context string specified in CONTENTS option.** The Contents topic defined in the Help project file could not be found. The Help Compiler uses the first topic in the build as the Contents.
4072	**Context string exceeds limit of 255 characters.** The context string hidden text cannot exceed 255 characters. The Help Compiler ignores the string.
4098	**Context string(s) in [MAP] section not defined in any topic.** The Help Compiler cannot find a context string listed in the [MAP] section in any of the topics in the build. The Help Compiler ignores the entry.
4113	**Unresolved jump or pop-up *contextname*.** The specified topic contains a context string that identifies a nonexistent topic. Check the topic for spelling errors in the context string, and also check to see if the requested topic is included in the build.
4131	**Hash conflict between *contextname* and *contextname*.** The hash algorithm has generated the same hash value for both of the listed context strings. Change either of the context strings and then recompile.
4151	**Invalid secondary window name *windowname*.** The window name for the secondary window is "main" or another disallowed member name.
4171	**Cannot use secondary window with pop-up.** The hidden text defining the pop-up identifier contains a secondary window name.
4196	**Jumps and lookups not verified.** Due to the low memory conditions, the build is continued without the jump and keyword validity verification.

Footnote Errors

The following messages are caused by problems with footnotes in topic files.

4211	**Footnote text exceeds limit of 1023 characters.** The footnote text cannot exceed the limit of 1023 characters. The Help Compiler ignores the footnote.
4231	**Footnote text missing.** The specified topic contains a footnote that has no characters.
4251	**Browse sequence not in first paragraph.** The browse-sequence footnote is not in the first paragraph of the topic. The Help Compiler ignores the browse sequence.
4272	**Empty browse sequence string.** The browse-sequence footnote for the specified topic contains no sequence characters.
4292	**Missing sequence number.** A browse-sequence number ends in a colon (:) for the specified topic. Remove the colon, or enter a "minor" sequence number and then recompile.
4312	**Browse sequence already defined.** There is already a browse-sequence footnote for the specified topic. The Help Compiler ignores the latter sequence.

Topic Title Errors

4331	**Title not in first paragraph.** The title footnote ($) is not in the first paragraph of the topic. The topic will not have a topic title string.
4352	**Empty title string.** The title footnote for the specified topic contains no characters. The Help Compiler does not assign the topic a title.
4372	**Title defined more than once.** There is more than one title footnote in the specified topic. The Help Compiler uses the first title string.
4393	**Title exceeds limit of 128 characters.** The title for the specified topic exceeds the limit of 128 characters. The Help Compiler ignores the additional characters.

Keyword Errors

4412	**Keyword string exceeds limit of 255 characters.**
	The keyword string exceeds the maximum limit of 255 characters.
4433	**Empty keyword string.**
	There are no characters in the keyword footnote.
4452	**Keyword(s) defined without title.**
	The topic has a keyword assigned to it, but no title.
	The topic will appear as ">>Untitled Topic<<" in the history list and in the keyword search dialog.

Build Tag Errors

4471	**Build tag footnote not at beginning of topic.** The build tag footnote marker, if used, has to be the first character in the topic.
4492	**Build tag exceeds limit of 32 characters.** A build tag for the specified topic exceeds the maximum of 32 characters. The Help Compiler ignores the tag for the topic.

Entry Macro Errors

4551	**Entry macro not in first paragraph.** The "!" footnote (for running a macro) is not in the first paragraph of the topic. The Help Compiler ignores the macro.

Topic File Errors

The following messages result from formatting problems in one or more topic files.

4616	**File *filename* is not a valid RTF topic file.**
	The specified file is not an RTF file.
	Check to make sure that you have saved the topic file as RTF from your word processor.
4639	**Error in file *filename* at byte offset 0x%lX.**
	The specified file contains unrecognized RTF at that byte offset.
	This message should not appear if you are using Microsoft Word for Windows, Microsoft Word for MS-DOS, or Microsoft Word for the Macintosh..
	Check the RTF syntax and then recompile. If you are using Microsoft Word for the Macintosh, transfer the file to the PC again, and then recompile.
4649	**File *filename* contains more than 32767 topics.**
	The maximum number of topics allowed in one RTF file is 32767.
4652	**Table formatting too complex.**
	The Help Compiler encountered a table with borders, shading, or right justification.
	Remove the unsupported formatting and then recompile.
4662	**Side-by-side paragraph formatting not supported.**
	The side-by-side paragraph formatting is not supported in WinHelp in Windows version 3.1.
	The Help Compiler ignores the side-by-side text.
	If you are using WinHelp in Windows 3.1, use the table feature.
4671	**Table contains more than 32 columns.**
	The maximum number of columns in one table is 32. Some word processors may have different limits for the number of columns supported.
4680	**Font *fontname* in file *filename* not in RTF font table.**
	A font not defined in the RTF header has been entered into the topic.
	The Help Compiler uses the default system font.
4692	**Unrecognized graphic format.**
	The Help Compiler supports only Windows bitmaps and metafiles.
	The Help Compiler ignores the graphic.
	Make sure that you have not used Macintosh picture formats.
4733	**Hidden page break.**
	The page break was found as a part of the hidden text. A page break formatted as hidden text will not separate two topics.
4753	**Hidden paragraph.**
	A paragraph marker was found in the hidden text.
	The Help Compiler ignores the paragraph marker.
4763	**Hidden carriage return.**
	A carriage return was found in the hidden text.
	The Help Compiler ignores the carriage return.
4774	**Paragraph exceeds limit of 64K.**
	A single paragraph has more than 64K of text or 64K of graphics.
4792	**Nonscrolling region defined after scrolling region.**
	A paragraph that was authored as "keep with next" is not the first paragraph in the topic.
	The Help Compiler ignores the "keep with next" attribute, and the paragraph is treated as regular text and will be part of the regular topic text.

4813 **Nonscrolling region crosses page boundary.**
The "keep with next" paragraph formatting crosses a page break boundary.

Miscellaneous Errors

The following messages are caused by conditions such as MS-DOS file errors or out-of-memory conditions.

5035 **File *filename* not created.**

There are no topics to compile, or the build expression is **False** for all topics.

The Help Compiler does not create a Help file.

5059 **Not enough memory to build Help file.**

To free up memory, unload any unneeded applications, device drivers, and memory-resident programs.

5075 **Help Compiler corrupted. Please reinstall HC31.EXE.**

The virus checking code has detected a corruption in the Help Compiler.

Reinstall the Help Compiler from the original source disk.

5098 **Using old key-phrase table.**

Maximum compression can only result by deleting the .PH file before each recompilation of the Help topics or by setting the **OLDKEYPHRASE** option to **0**.

5115 **Write failed.**

Write to disk failed.

Contact Microsoft Product Support Services.