# CLARION TECHNICAL BULLETIN

Bulletin #112

Clarion Help Files



Clarion Software Corporation

**Overview**

This article answers frequently asked questions about Clarion help files and discusses some of the powerful help capabilities you can use.

## Clarion Help Files

When developing applications, many programmers save making help windows until the end, and it's not because they're saving the best for last.

Most programmers spend the bulk of their development time making sure their program performs properly. Other programmers don't enjoy "writing," so naturally they squirm at help window time. Often, the result is slapdash windows that, in the end, aren't much help at all. But with Clarion's Helper utility, you can easily and efficiently make help windows, or you can delegate the task to someone else.

Occasionally, Clarion users have questions about the way "help" is done. Two frequently asked questions are, "Why is each help window maintained in a file separate from the program file?" and "How does help work in my programs?" This article answers these questions and discusses some of the powerful help capabilities you can use to make the help windows task easier for you.

**"Why is each help window maintained in a file separate from the program file?"**

Clarion help windows are maintained in separate files for a number of reasons, including:

1. Independent help file development
2. Optimal use of machine resources
3. "Intelligent" help file maintenance
4. Powerful window relationships

First, a separate help file allows you to create and maintain help screens without having to edit or recompile the Clarion program. You only need to include program "hooks" into the help file. This way, help window development can be delegated as a task separate from program development.

And if the development staff doesn't create the help windows, who will? Perhaps technical writers or the tech support staff. Technical writers could create help windows that expand upon the topics covered in the documentation, thus tying the product and documentation closer together. The tech support staff could anticipate where users may have questions. They can create help windows that deal with complex topics, and therefore reduce the number of technical support calls they receive.

Since help can be developed independently with the use of other personnel, product development time can be shortened. Also, the "outside input" on the product can result in improvements.

Second, help files are maintained separately to reduce program resource requirements. Providing help is important, but it should not overtax the resources that could be better used by the main program. Including a help window for each field and/or screen that occurs in a program could require a large amount of memory. As we all know, memory is a precious commodity and most programs perform better when more memory is available. Inside Clarion programs, a minimum amount of space is reserved for displaying help windows, leaving the remaining memory available for program use.

Help windows are compressed on disk, and when required, are expanded for display, thereby providing the optimal use of space both in memory and on disk. Clarion help files can contain hundreds of windows and not degrade the program's performance by using additional memory. This is significant, especially if the users of these programs become "experts" and rarely access the help functions.

Third, since help files are compressed, more information can be stored in the file than just the windows. When a window is edited, Helper (CHLP.EXE) stores information about how the characters and attributes were placed on the screen, i.e. paints, enhanced, reversed, etc. By compressing this information into the help file, the file size remains small, yet it allows Helper to be more "intelligent" by "remembering" things about the windows. Before a help window is finalized, it may be edited a number of times. The more information that Helper can remember about the window, the easier it is to change the window later.

The fourth reason for having separate help files is that it allows the development of complex relationships between windows. Help windows can be "chained" to a succeeding window or to a variety of other windows. By allowing the development of menus, you can make a help file that not only provides context-sensitive help, but also allows the user to traverse the entire help file (achieving a hypertext capability).

Without Helper, maintaining these window relationships in the source code of a program would be an almost impossible task. Adding new or removing old windows would require a tremendous amount of sorting through source code to repair the relationships. Using Helper to maintain the help files, however, simplifies the requirements for altering relationships. Windows can be added, deleted, moved, or edited without having to change any source code.

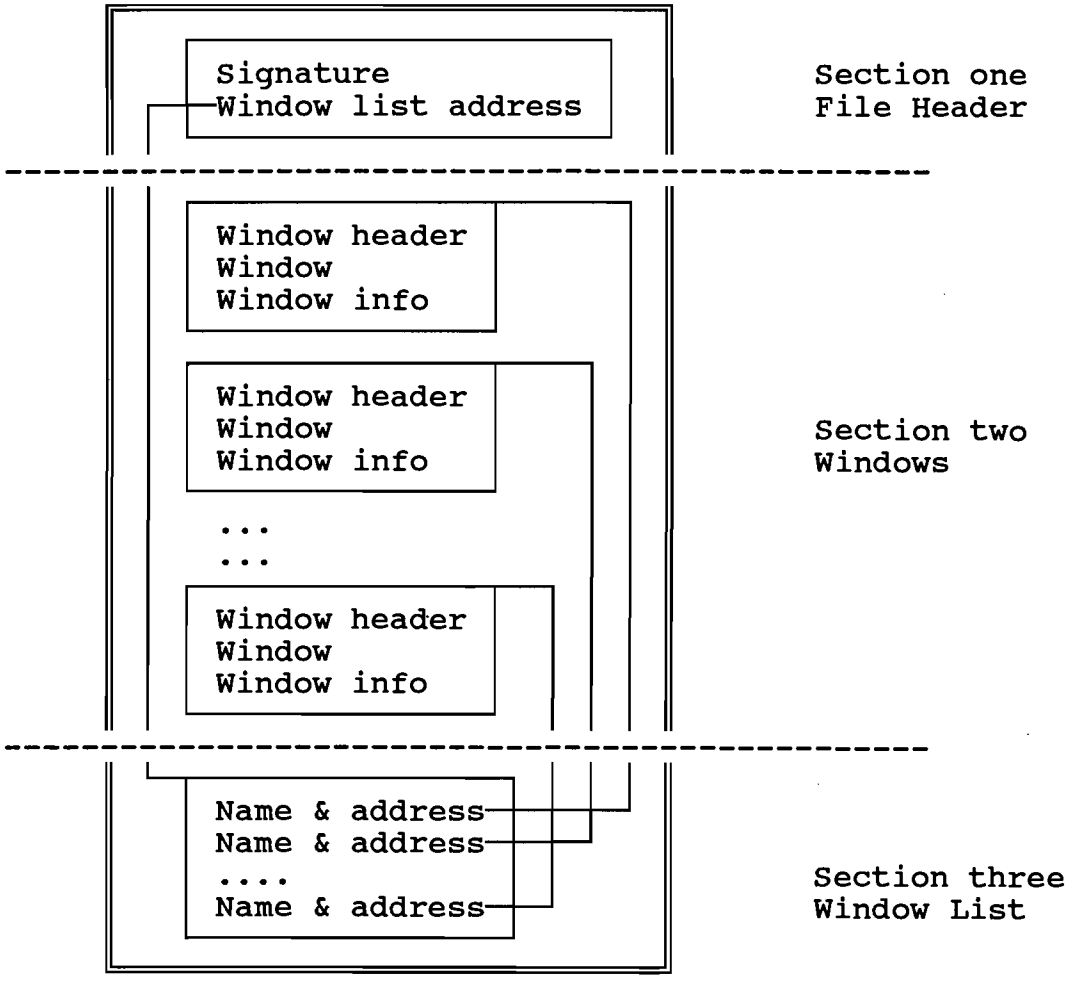
**How does help work in my programs?**

Understanding how help works in your programs may be easier once

you know about the structure of a help file, which looks something like the diagram shown next.

General Structure of a Help File




The structure of a help file can be divided into three general sections. The first section is the help file header. This header contains a "signature" that identifies the file as a Clarion help file, as well as the file offset where the window list can be found. When the help file is opened, Clarion validates the signature to ensure that an invalid file is not used as a help file.

The window list offset is used to find the third section of the help file. This section contains the names and addresses of all the windows contained in the file. This list occurs near the end of the help file after the windows. When a user accesses help for the first time, the help file is opened, validated, the window list is read into memory, and then the window is

displayed. Afterward, the list remains in memory so that future windows can be retrieved without having to re-validate the file or re-read the window list.

Windows are kept compressed in the file until a request to display a window is made. Windows are stored in section two of the help file. A window entry in the help file consists of some header information, the window itself, and the information that Helper remembers about the window. When a request is received to display a help window, the name is located in the window list (already in memory), the offset is used to read the window's information, the window is decompressed, and then displayed on the screen.

Help windows can be stand-alone, chained to a single window, or can contain a menu that links to several windows. Once displayed, a help window waits for the user to press a key that will remove the window or "call" another window. When a valid key is found, the window is removed and the original program screen is restored or another help window is displayed.

If a succeeding window is referenced, the previous window name is "remembered." Previous windows can be recalled, while inside the help display function, by pressing the PgUp key. Additionally, the original help window can be recalled at any time by simply pressing the F1 key again. The ability to move backwards through the windows, as well as offering menus to other windows, can allow a user to peruse as much of the help file as you want.

Note: Since batch release 2004, the Processor (CPRO.EXE) can overlay a portion of the Helper utility. This allows you to edit your help windows as your program runs. To activate the help window editing, press F1 to display the help window that you want to edit, press Control-F1, and then you will be able to edit the help window. This convenient feature allows you to edit the window "exactly" as it appears in your running application.