Paper 229

Designing Accessible Help Systems

Cathy Brinsfield, Meridian Software, Inc.

ABSTRACT

SAS® GUI applications can benefit greatly from well-designed, integrated help systems. This poster uses WinHelp to show you how to design your help system using granular information and progressive disclosure. How many times have you opened the help for an application only to be overwhelmed by the amount of information you must sift through to get to the piece you need? Or you have spent so much time clicking on links that you are off somewhere in cyberspace?

This approach separates task information for a window from the overview information a user may need to understand the tasks. The help for each window presents the user with a selectable list of tasks and assistance with any navigational tools. The user can link to help for the task, or access overview information that provides a broader perspective on how the task in the window fits into the overall purpose of the application.

INTRODUCTION

Often the help system associated with an application is put together at the end of a project. Developers are reluctant to add help before the system is "stable", and stability sometimes means waiting until the product is going out the door. When help is developed this way, it may not be very helpful.

A better approach is to begin the help design as the system is prototyped and specifications are developed. Details about how to achieve specific tasks can wait until the system is stable, but you can define the tasks that will require help early in the development cycle.

The key to creating accessible help systems is to design the help before implementing it while working closely with other developers as the software takes shape. Of course, this assumes that the software developers are designing and prototyping the software application as it is developed as well.

This paper will guide you through some steps you can take to make your help systems more useful. You will see some design and development tools as well as examples from a help system for a GUI application developed by Meridian Software, Inc.

SELECTING A DEVELOPMENT TOOL

With SAS applications, you have several choices of tools for developing your help system. SAS provides CBT (computer based training) catalog entries that you can use to develop help. The advantage of SAS CBT entries is they are completely portable and, since they are part of SAS, don't require any additional steps to get SAS to recognize them as help entries. On the down side, they can be time consuming to develop, especially if you want to provide sophisticated links or graphics.

Another option for help for SAS applications is WinHelp. This is available currently for Version 6 SAS applications. (HTML-based WinHelp will be available with Version 7.) WinHelp is relatively easy to develop and the interface is familiar to most users. It is only available on Microsoft platforms, and isn't portable to other hosts. You have to provide SAS with a map file to associate WinHelp topics with each FRAME entry in the SAS interface. More on map files later!

There are many options for simplifying the development of WinHelp. One option is RoboHelp®, which provides a Microsoft Word interface and some extra tools for developing WinHelp.

There are other products designed to help with development, and Microsoft® provides a basic tool as well. These other products are not discussed in this paper. The tool selected by Meridian Software was RoboHelp, since we were already familiar with the Word interface.

The RoboHelp product provides

- an editing environment for creating help topics and links
- management tools for managing the components of the help system and setting up the help system properties
- support tools for creating map files, a table of contents, and indexes for the help system
- compilation tools for converting the Word documents containing the help topics to RTF (rich text format) files and compiled help files.

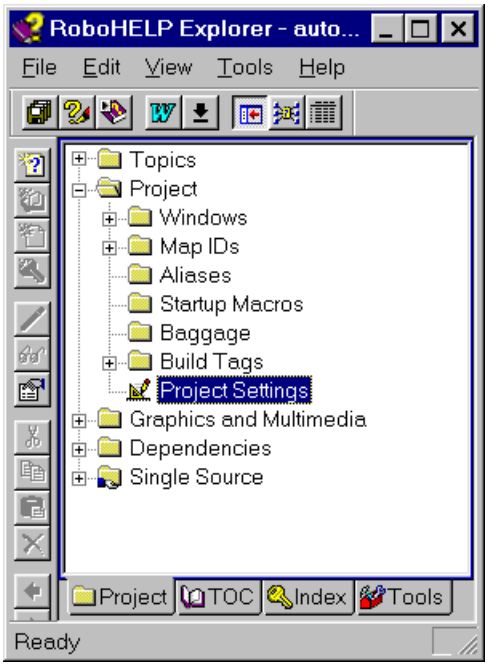

Figure 1 RoboHelp Explorer

Figure 1 shows the RoboHelp Explorer, which provides tools for managing and setting up the WinHelp project.

DESIGN STEPS

When designing the help system, a good starting point for the help developer is the functional specification for the project. Most software applications start with a list of requirements from the end user. These can be converted to written functional specifications that spell out in some detail what the functionality will be for each window in the GUI. You may be writing these functional specifications yourself or working with other software developers to develop them jointly. Or you may be working from a prototype of the system that has the proposed functionality displayed in the interface. In a prototype, each button may not actually do what it will ultimately do in the finished software, but you can still get an idea of the tasks that will be available to the end user from each window.

Now is the time to go through the prototype and ask the developers questions. Your feedback will provide valuable input as the interface is further defined. One way to perform early usability testing and to discover any weaknesses in the GUI is to see how complicated it is to provide help to users for a task. If you have to explain too much, the system may need some redesign or fine-tuning.

Each separate functional unit requires a primary help entry. The primary help entry should orient the user to the main tasks available in the window. An option to view overview or background information should also be offered in the primary help entry. Secondary help entries for each major task are generally links from the primary window.

When dealing with a large or complex system, it's helpful to create a table that lists each functional unit, or window in the system, and each secondary help that will need to be associated with it. When you develop this "design" table, you assign a logical name to each help entry and its associated sub-entries.

You can quickly see when the same sub-entry can be reused for different primary entries. These reusable subentries can be considered global, since they are reused multiple times for many different windows. An example of a reusable subentry is Help for Printing. If you allow the user to print each screen in an application, and you have a Print button in each window, you can reuse the same Help entry that describes how to print the screen as a subentry from multiple help windows.

The design table uses the following columns:

Component	FRAME entry name
Subcomponent	each action in the window that could require help support
Help Title	Title for Help Entry
Content	Brief description of the help entry content
Topic ID	WinHelp name for the entry. For each FRAME entry this is *framename*_1. For each element (widget) within a frame, the name is *framename_element-name.* SAS uses these names to associate the help with the correct frame or widget. The widget names are only important if you are providing field-level help.
Links to	Topic ID of each entry that you want to provide a link to from the component or subcomponent.

Once you complete your design table, you can develop an outline for the Table of Contents. Each grouping of activities can provide a book for your table of contents. Each subactivity that falls into that group (this usually maps to a window in the interface) will be a help topic in that book. You may want to include overview information for an activity as a topic within that book. Or you may choose to group all the overview information together in a separate book. Look at the Table of Contents for several different help systems to get an idea of which approach you like.

DEVELOPING THE HELP SYSTEM

RoboHelp enables you to use multiple Word documents to contain the topics for the help system you are building. It will be easier to manage your project if you group help topics together logically in separate Word documents. Putting the main help topic and all of its subtopic links into a single Word document makes sense, with the following exceptions:

- put global items used by multiple windows into a separate document called global
- put overview topics that may be linked to from multiple entries into a document called overview.

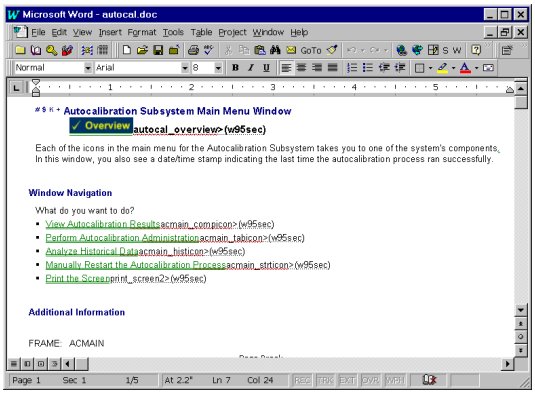

Figure 2 RoboHelp Development Environment

Figure 2 shows the RoboHelp development environment. This is the first topic in a Word document. You can see the links underlined in green. Each task the user can choose in the window is associated with a link. Each of these tasks is a topic in this Word document. When the help is compiled, the user links to the subtopics when they select one of the links. The goal is to minimize the amount of information confronting the user when he or she opens the help for the window. Here they are simply presented with a list of help for possible actions that can be initiated in the window the help supports. The user also has the option to select an overview if none of the actions make sense.

SAMPLE HELP SYSTEM

The help system we developed provided support for two related, but independent SAS GUI applications for a manufacturer. The systems were to be used by Quality Control experts at the company. There were some parallels between the two systems, but also some major differences. Both systems provided reporting, analysis, and administration functions. The Autocalibration System enabled quality analysts to calibrate production line meters statistically. The Data Visualization system automated the generation of graphs for production line data. The company had decided against hardcopy user documentation, so the help had to be comprehensive.

The challenge was to reuse as much information as possible while developing two independent help systems. In developing the design table, it was useful to group the help into several categories:

- Components that were the same in both systems
- Global components that were reused by multiple entries within both systems
- Components that were specific to one system or the other

The goal was to reduce the amount of redundancy and not to make the same changes to the help in multiple places. To accomplish this goal, the help for one system was developed first. Only the framework was developed for the second system. Once the reusable content was stable for the first system, it was ported to the second system.

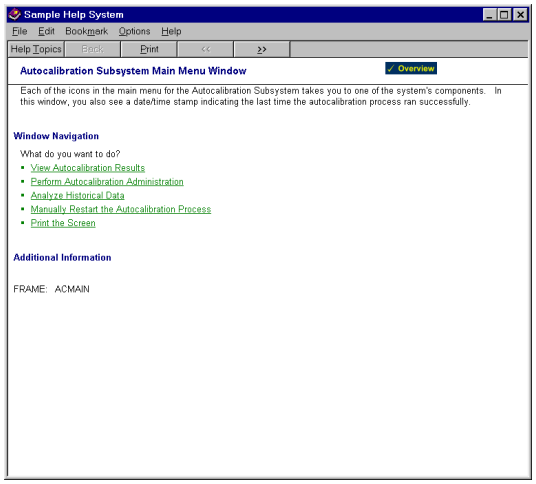

Figure 3 Main Menu Window

Figure 3 shows the compiled help entry for the Main Menu of the Autocalibration System. To simplify system support and debugging during testing, we listed the name of the FRAME entry the help supports. If the user or tester had a problem with the window, it could easily be identified by selecting the help for the window.

Once the user decides which task to get help for, more detailed information is made available in the secondary help link.

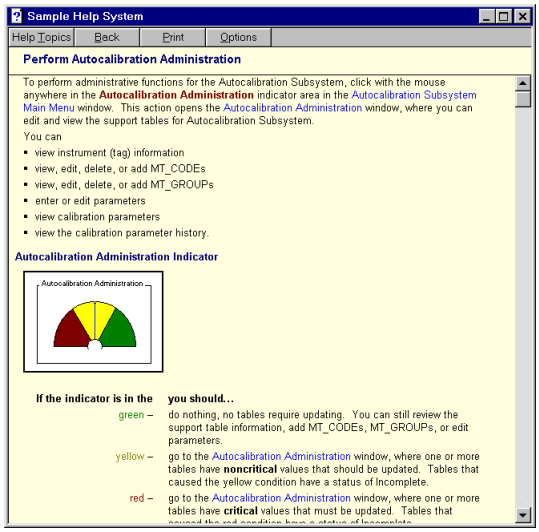

Figure 4 Link from Main Menu Help

Figures 4 and 5 show two of the links from the Main Menu help entry. These are secondary windows, so we used a different color to display them. Only the primary windows have a white background, all other secondary windows that are reached from another help entry are pale yellow. We also controlled the placement and size of the secondary windows, so the user would have some additional cues about where they were. The only links we allowed from secondary windows were to overview information.

The other choices were to go back to the primary help for the window, or go to the index or table of contents. We wanted to avoid too many confusing branches and the possibility of too many open overlapping help windows. We used color sparingly, but consistently throughout the help system to indicate window names and selectable links. We also included graphics where they would be helpful.

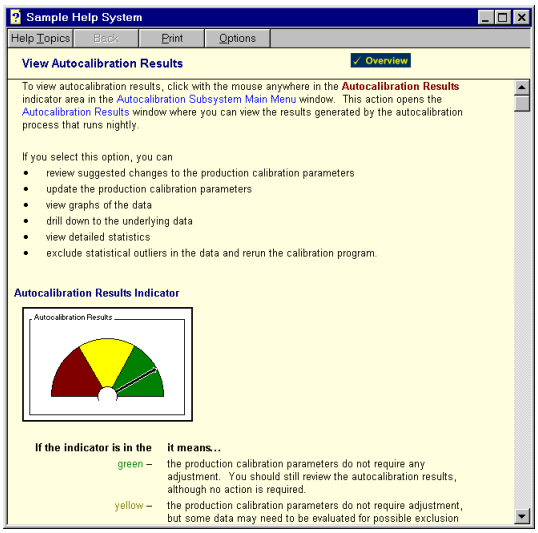

Figure 5 Link from Main Menu Help

The analysis facility built into each system provided the user with several different analysis types. Some of the analysis types were similar between the two main systems, so help was reused where appropriate. The analysis Run Options windows allowed the users to set various options for the analysis. These help windows described each of the options as well as providing overview information for the type of analysis.

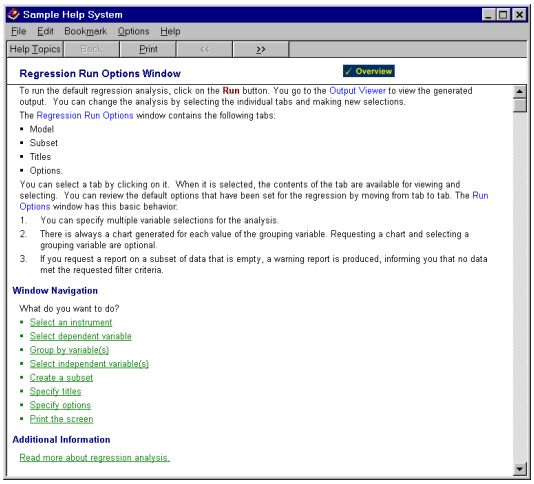

Figure 6 Regression Analysis Run Options Window

Both of the Analysis Windows shown in Figures 6 and 7 reuse many of the same secondary windows, including the Create Subset Help topic shown in Figure 8.

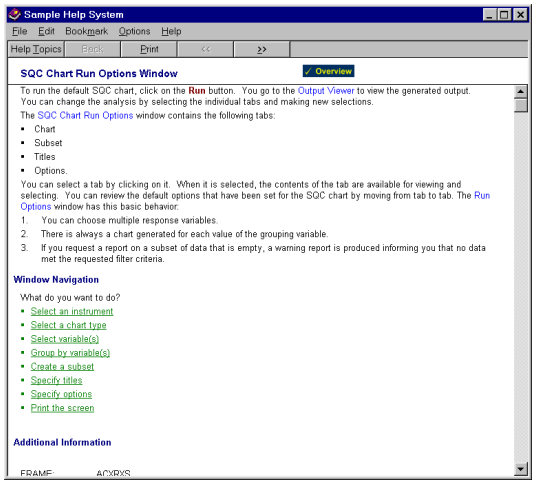

Figure 7 SQC Chart Run Options Window

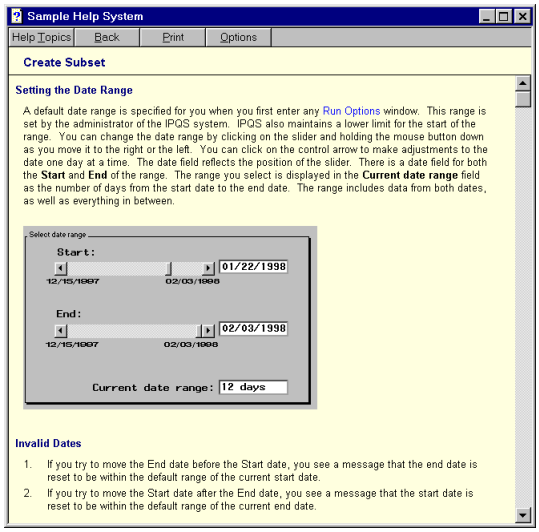

Figure 8 Help for Creating a Subset

ASSOCIATING THE HELP WITH THE APPLICATION

When you set up the help system in RoboHelp, you can specify that a map (.hh) file be created for each Word document to identify the help topics within it. With a few minor modifications, SAS can use the map file to associate the main help window for each FRAME entry. The map file created by RoboHelp has some extra information in it that SAS doesn't need. All SAS is expecting is

- the name of the FRAME followed by a space
- the number 1
- a unique number generated by RoboHelp.

To create the map (.hdx) file expected by SAS, we wrote a SAS program that grabbed all the map (.hh) files generated by RoboHelp, and combined them into one .hdx file with the extraneous information stripped out. We didn't create map files for overview or global topics, since they were not associated with a specific FRAME. If you don't want to create a map file for a specific Word document in the help system, RoboHelp lets you deselect it in the map file list. See Figure 9, Creating Map Files.

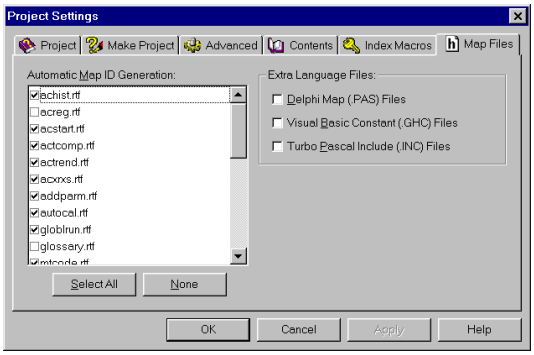

Figure 9 Creating Map Files

We did not provide field-level help in this application, but you can use the map file to provide those associations as well. For a detailed discussion on associating WinHelp with SAS applications, see John Kruth's paper on implementing native help in SAS applications in the SUGI 22 Proceedings.

You will also need to put your compiled help file and the associated table of contents and map files where your application can find them. When your icon initializes your application, you tell SAS where to look for the associated help with the -helploc option in the Target field of the icon properties.

Example: -helploc d:\clients\systemname\winhelp

If you need to provide access to other SAS help, you have three choices

- put your compiled help in the SAS help directory.
- copy the SAS help files you need into the directory where your application help resides
- create a shortcut to the SAS Help files and put it on your desktop to keep the regular SAS help available all the time.

To create the shortcut to SAS help, follow these steps:

1. Open the Microsoft Windows Explorer and find the location of SAS612 (or whichever release you are running) on your machine.
2. Expand the SAS612 directory by clicking on the + sign.
3. Find and expand the core directory the same way.
4. Double click the Winhelp directory to see of list of files.
5. Find the file called base.hlp. Highlight it and right click with the mouse. Select Create Shortcut.
6. The shortcut file is added to the bottom of the list in the window. Highlight it with the left mouse button and drag it to the desktop. This creates a Help icon on your desktop. You can now click on the icon to run SAS Help at any time.

CONCLUSION

To build accessible help systems, keep the following points in mind:

- Pick the best tool for delivering your help.
- Do an analysis of what functionality will be implemented in each window of the system
- Identify what tasks will be the same in multiple windows so you can reuse information
- Design the help system before you begin development.
- Provide one main help entry per window, then secondary help for each task.

- Keep overview information separate. Don't overwhelm users with too much text when they open the help. If the user is lost, give them an overview button that is easy to find.

REFERENCE INFORMATION

Cathy Brinsfield, Information Developer
Meridian Software, Inc.
12204 Old Creedmoor Road
Raleigh, NC 27613

Phone: 919.518.1070
FAX: 919.518.1170
email: merclb@meridian-software.com
WWW: www.meridian-software.com

BIBLIOGRAPHY

"Native Help Technology in SAS/AF Applications", John Kruth, SAS Users Group International (SUGI 22) Proceedings, San Diego, CA

ACKNOWLEDGMENTS

Thanks to Maria Spicer of Meridian Software for the instructions for putting a SAS Help icon on the desktop.

TRADEMARKS