NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Advisor : A System to Develop and Use Online Help

By

Patricia Murphy

Thesis Submitted to
the School of Graduate Studies and Research
in partial fulfilment of the requirements for the
degree of Master of Computer Science

University of Ottawa
Ottawa, Ontario, Canada
1990

ISBN 0-315-60018-7

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

Abstract

The purpose of this thesis is to create a tool (Advisor) for the development and use of online help systems. The study of human-computer interface issues is central to this task. In particular, attention has been given to the analysis of learning principles as related to interface strategies.

The Advisor software is used by help designers to develop help systems and by application users to get assistance during application use. It has two parts: the Designer (editor/outliner used by the help developer for help screen definition) and the Browser (help viewer used by the software end user for assistance from within an application).

Using hypertext technology, the Advisor creates a tutorial environment where help screens and keyword links provide context sensitive help to assist the application user. A simple interface exists for access to the Browser by an application program.

I hereby declare that I am the sole author of this thesis. I authorize the University of Ottawa to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Patricia Murphy

I further authorize the University of Ottawa to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Patricia Murphy

Table of Contents

Figures

Tables

Part I: Introduction

Chapter One: Overview

Two fields which I find fascinating are computer science and education. I have pursued studies in both areas. It is the quest for knowledge in each realm that has lead me to the selection of a thesis topic; a system to aid in the development and use of an online help facility.

An effective help facility should take into account basic concepts of how the user learns while keeping focused on '... the successful performance of application tasks (Palmer et al, 1988, p. 44). However, most software developers have not studied educational theory and do not apply sound pedagogical principles to their program design. Knowledge of both fields is important to software development of help interfaces. Whether embodied in one person or in a team of specialists, underlying principles in both fields should be brought to bear on the final product.

The Advisor help interface is general in nature. Explanations are given as to procedures to attach to existing software packages and those currently being created.

The interface has been based on pedagogical principles of learning which will be described in general terms in chapter three and more specifically throughout the description of the functional specifications. The prototype can be thought of as a teaching aid in itself; as all help facilities should try to enhance the learning of the user.

Part I of the thesis contains an introduction, description of the scope of the thesis and discussion of design issues. Parts II, III and IV deal with the three major parts of the help interface: the Designer (used to develop help screen); the Browser (used to view help screens); and the requirements for interfacing the Browser with a given software package. These three parts are divided into chapters detailing: functional specifications and software design. Part V gives a summary and conclusions with notes on future directions. The appendixes give details on users' manuals, program specifications and customization.

Chapter Two: Scope of thesis

The advisory system can facilitate learning and software use. It has been built using pedagogically sound principles of learning and has been implemented using hypertext technology.

In keeping with these aspirations the goals of this thesis are:

> to design and develop a software tool to be used in the development and use of online help. The package includes a Designer which is an editor/outline facility for help screen development (authoring) and a Browser to be used as a help facility in conjunction with application software.
>
> to explore various avenues of hypertext technology.
>
> to implement research findings on human-computer interface design.
>
> to apply pedagogically sound principles, as endorsed by current educational research, as the foundation for a general help interface.

The Advisor Designer is used by the help developer to map out the help screens in a hierarchical structure, using various levels to define help menus and submenus. The outline structure gives an inherent relationship to the help items which can be followed by the user of the help facility. In addition, the help developer can identify other links between help items by defining index words which can be picked from an index table by the help user and keywords which will be highlighted throughout the text on the help screen and can be selected at any time. Each help outline item has an associated help screen or screens where the detail seen by the user is defined.

The Advisor Browser is used by the help user to view the help screens created by the Advisor Designer. Help screens can be viewed in the outline order, by index selection (from an index list of words) , by keyword selection (from highlighted words throughout the general text of the help screen) or in a graphical display of the help screen network with a highlighted path to the current help location. See figure 1 for an overview of the Advisor Designer and Browser.

The Browser has a simple interface which can be used by the application developer and linked to their software. The developer chooses a key (eg. F1) and calls the Browser unit from their program when this key is pressed.

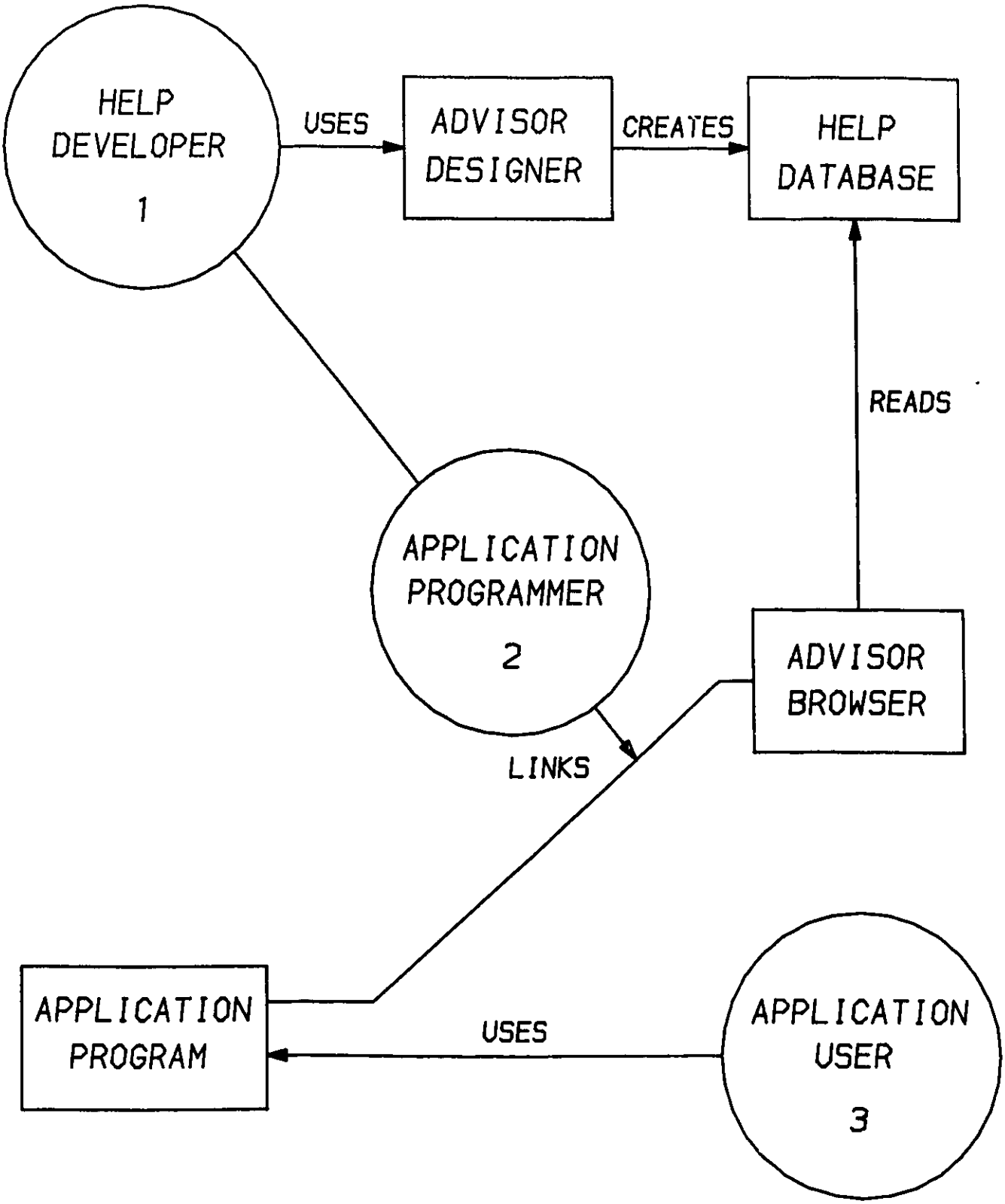


Figure 1. Overview of Advisor

Some of the design constraints on the Advisor are:

a. The tool must be easy to use. There should be a depth to the design which allows novice users to achieve success and provides experts with the power they require. To keep the Browser interface simple, it supports passive help (the user must ask for assistance). At the user's initiation, further more detailed help screens can be displayed if the upper level task reminders are not sufficient. Colour customization of help screens by the help developer and end user is provided.

b. The software should be designed to interface with existing systems and software under development. The Advisor must be developed in a modular fashion; with interface modules clearly named and accessible, to allow for easy attachment to other software.

Chapter Three:Development Issues for the Design of the Advisor

An important design issue of the Advisor is that of the human-computer interface. The interface is the place where '... the user and computer engage in a communicative dialogue whose purpose is the accomplishment of some task.' (Card, 1983, p. 4) It is important to view the interface as a point where there is communications with, rather operation of a machine (Card,1983).

'Many in the computer field agree that there is an obvious way to design better human-computer interfaces. Unfortunately, they disagree on what it is.' (Card, 1983, p.7) Much of the current research into efficient interface design is going hand in hand with efforts by cognitive psychologists to determine how we learn. Both groups suggest theories and try them out on each other. As a consequence, there are many divergent theories being tried in an effort to find the best interface. Rules for the design of the human-computer interface have not yet emerged.

However, some basic principles for user-interface design are:

1. **Early** in the design process, consider the psychology of the user and the design of the user interface.

2. Specify the **performance requirements**

3. Specify the **user population**

4. Specify the **tasks**

5. Specify the **method** to do the tasks

6. Match the **method analysis** to the level of commitment in the design process

7. To **reduce the performance time** of a task by an expert, eliminate operators from the method for doing the task.

8. Design a set of **alternative methods** for a task.

9. Design a set of **error recovery methods**

10. Analyze the **sensitivity** of performance predictions to assumptions. (Card, 1983, p.418)

These principles are also recognized as important by Nielsen and Molich (1989). Advice giving systems `... store information about a system and its commands, conditions, procedures, etc. and can access this information and provide it to users as on-line training and help'(Carroll and McKendree, 1987, p.15). The design process requires both knowledge issues (what information) and dialogue issues (how and when should it be provided. These issues will be considerations throughout the next section.

3.1 Early design decisions

Many researchers start with the premise, that an interface must have an early and continual focus on the user to be successful (Gould and Lewis, 1985; Card, 1983). The first priority in designing the Advisor was to use sound pedagogically proven principles on learning. Learning is a '... relatively permanent change in behaviour due to experience or practice.'(Knapper, 1980, p. 70) Some basic learning principles are: continuity (stimulus and response must be contained in a contiguous block of time); repetition (practice is needed to improve learning and lengthen retention period. An extension is that feedback further aids the process by reinforcement (coined by B. F. Skinner to describe events previously defined by Thorndike (1913) as 'the law of effect') (Knapper, 1980).

Gagne and Briggs (1974) consider additional internal factors: factual information (recalled from prior learning); intellectual skills (recalled from prior learning); and strategies (ways the individual uses to learn or remember from prior experience). The help developer is encouraged to implement these learning principles through the structure of the Advisor as detailed in later sections. Definition of the help screen objectives can support decisions on the sequence of screens, types of strategies (step by step or functional).

Early design decisions include: screen layout considerations (where the user would look for menus, error message, status message, etc); screen size (help screens overlapping existing applications but leaving them visible to allow the user to see where they came from) (Houghton, 1984); use of universal commands (common to outline and detail screens); simplicity of style; command or menu driven (command has faster task completion and menu selection aids to memory; status indicators; error feedback provisions (messages with brief and positive tone) (Nielsen and Molich, 1989); clearly marked exits and user customization (Smith et al, 1982; Nielsen and Molich, 1989).

Some of the types of learning that have been described by Gagne (1965) include: signal learning, stimulus-response, chaining, verbal association, multiple discrimination, concept learning, principle learning, and problem solving. This shows the complex-

ity of the learning process. Each level has as its prerequisite the previous level. The help system provides support in a problem solving environment.

It is clear that the individual needs masses of structurally organized knowledge to handle the problem solving type of learning. (Gagne, 1965). Hypertext technology offers solutions to the structuring of a database to contain this information. The idea of tapping into a knowledge repository was suggested in 1945 by Vannevar Bush. 'The human mind...operates by association. With one item in its grasp, it snaps instantly to the next that is suggested by the association of thoughts, in accordance with some intricate web of trails carried by the cells of the brain..... Man cannot hope fully to duplicate this mental process artificially, but he certainly ought to be able to learn from it.' The device proposed to aid in this process was the memex; '...a device in which an individual stores his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility. It is an enlarged intimate supplement to his memory.' (Vannevar Bush from 1945 Atlantic Monthly article 'As We May Think' repeated in Lambert et al 1986 p.15).

The Advisor provides the opportunity to use sound principles for the implementation of an advisory interface. It provides command assistance, index features, keyword selection and a global browser. It is up to the help developer to design help screens that are accurate, consistent, complete, visually appealing (avoid solid blocks of text), at an adequate reading level (grade five), non-anthropomorphic (user feels blamed when help is too personal), and consistent with written documentation (Houghton, 1984).

3.2 Performance requirements

The performance variables include functionality, time, errors, learning. Define requirements while keeping in mind the total picture; fast performance time may require the development of a system which is very difficult to learn. Typically response and display rates are: typing and cursor motion within 0.1 seconds; simple commands in less than 1 second; and similar commands with 20 percent deviation in time to process. (Shneiderman, 1982) In the case of the Advisor, priority was given to make the system easy to use and to provide the opportunity to test the learning theories listed above and the use of hypertext technology. Faster is not always better. It has been shown that users make more mistakes when trying to respond quickly to a fast system (Shneiderman, 1982).

3.3 User population

The Advisor has been designed for a wide variety of users. The expected audience for the Advisor Designer is experienced software developers. The Advisor Browser may be used both by novices or experts. General principles have been applied to the design of the system and options given to the help developer to select solutions appropriate to the target population.

One of the most important principles of learning, to be considered when assessing the user population is that there are individual differences in learning; both quantitative and qualitative. People differ in their speed of learning and in their learning styles (Knapper, 1980). 'Learning style is the unique way each individual gathers and processes information.' (Dixon, 1982, p.62)

If help is treated as an electronic page turner then it behaves as a textbook which 'symbolizes the assumption that learning is primarily concerned with abstract ideas and concepts' (Kolb, Rubin, McIntyre, 1974, p. 27). Kolb et al have designed a four dimensional model to show other areas to be investigated when analyzing learning styles: concrete experience vs formation of abstract concepts and generalization; and observation and reflection vs active experimentation. Users can show tendencies in any of these styles. Two approaches have been identified: the holistic or functional approach (learns as the functionality of whole picture takes shape); and a step by step or procedural approach (step by step guidance; learns during the process). Results on the optimum approach are inconclusive (Carroll and Aaronson, 1988). The Advisor provides the facility to present information to various types of learners. The help developer decides whether the content will be organized in a step by step or functional way. Previewing prototypes on sample groups is one method for determining the adequacy of the help screens.

Knowledge issues related to the user population include: general skills knowledge (knowledge of advisory strategies and natural language use); domain knowledge and user model knowledge. Although these issues focus on active assistance, some ideas can be used for the design and implementation of a passive system.

The advisory strategies include: socratic (the system provides questions and the user gives answers)(Carroll and McKendree, 1987); learning-by-doing (compares the user moves with those of an expert); learning-while-doing (makes use of a coaching technique) (Brown, Burton and Clancey, 1984; Burton and Brown, 1982) The coaching concept of transforming non-constructive bugs (user does not have sufficient means to change his behaviour as a result of a perceived error) into constructive ones (user has determined the cause of the error and can correct it) (Burton and Brown, 1982) is possible in a passive system. Through the in-

formation provided on the help screens, the sequencing of the help screens and the use of index and keyword features, the help developer can act in the capacity of the coach. Natural language interfaces are largely experimental (Carroll and McKendree, 1987) but might be considered as a part of a future artificially intelligent implementation of the Advisor.

Domain knowledge helps to determine the user's intentions from their current state and a history of their actions. Some of the most pressing questions in this field include: how to decide how much user activity to record; whether or not to use multiple representations of domains (procedural, factual, functional); and what grain of knowledge is ideal for an optimal learning environment. The Domain knowledge model can be represented factually and functionally. The help developer should provide screens which give step by step procedures to complete tasks, as well as examples showing functionality. The Advisor supplies the structure to achieve this goal.

User model knowledge determines if the user is a novice or an expert and on that basis, what behaviour to expect (Carroll and McKendree, 1987). Although not actively assessing the user, the Advisor has features which cater to the novice and expert as described in the functional specifications. The hierarchical structure of the Advisor help system allows for naive browsing of general items and extensive search of detailed items. Colour customization of the software allows the user to provide some input into the presentation of the screens. This is important to the learning modes of certain individuals (Dixon, 1982).

3.4 Tasks

The major tasks of the advisor are provision of:

a. a simple editor with outlining capabilities to design help screens.

b. the facility to identify index items and provide links between keywords and appropriate screens.

c. a browsing facility for the help user to view the help screens; including capability to navigate hierarchically and nonhierarchically (by index and keyword selection).

d. a facility to view the entire help structure with a path leading to the current location.

3.5 Method

The tasks are to be undertaken using hypertext technology. In general terms, hypertext has been described as 'a computer-based medium for thinking and communication'(Conklin, 1987, p. 32). Other definitions describe hypertext as a database method, a representation scheme or an interface modality (Conklin, 1987, p. 33).

The two main structures in a hypertext system are nodes and links. Each Help window is a node and is associated with an object in the database. Links between objects are provided by pointers in the database.

Some of the features of a hypertext system include:

1. The 'hyperdocument' which is a database of textual or graphical nodes.

2. Windows which are directly related to nodes in the database.

3. Windows contain 'link icons' used to reach other nodes in the database.

4. Standard window operations: repositioning, sizing, closing.

5. The facility for creating new nodes, new links to new nodes and new links to old nodes.

6. A database browser to
   a. follow links
   b. search for a string, keyword or attribute value
   c. navigate using a graphical display of the network

(Conklin, 1987, p. 19)

Some hypertext systems, and the special features attributed to them, are described in Table 1 (from Conklin, 1987, p. 21). As explained in Conklin's article, some of the systems are complete environments while others are no more than 'conceptual sketches'. Each has developed features which are appropriate to the specific system. The hypertext features implemented on the Advisor include:

1. Hierarchy

   The Advisor supports hierarchical structures through the natural organization of the outline.

2. Graph-based

The Advisor supports non-hierarchical structures (cross-reference) links through indexes and keyword links.

3. Paths

The Advisor strings links together while keeping a record of the current help path selected by the help end user.

4. Keyword or String Search

The Advisor searches for indexes and keywords.

5. Text Editor

The Advisor Designer is a text editor and outliner.

6. Graphical Browser

The Advisor Browser graphically represents nodes and links within the database in a tree browser.

The Help developer produces the help screens using the Advisor Designer. In doing so, the hypertext structure is used in the production of the outline itself. Links are automatically set up to reflect the hierarchical structure of the outline. The Advisor Designer provides the facility for the help developer to set up index words and highlighted keywords. These words are identified and the link to the help screen, to be located upon their selection, is defined by the help developer.

The help user may access the index list to make a selection or move the cursor to a highlighted word within the help screen text and make a selection. The chosen help screen will be displayed to the user. The help user may also traverse the help screens in the hierarchical order of the outline by taking the next and previous screen path. If lost in the middle of the help structure, the end user has the option to display the big help picture, showing the current position and highlighting a path from the main help screen. It may be useful for the help screen developer to view people using the target application, to make decisions about the most appropriate design of the nodes and links within the help system and to determine the best use of the advisory strategies provided by the Advisor Designer.

Hypertext Systems	Hierarchy	Graph-based	Link Types	Attri-butes	Paths	Ver-sions	Proced-ural Attach-ment	Keyword or String Search	Text Editor	Con-current Multi-users	Pictures or Graphics	Graphical Browser
Boxer	Yes	Yes	Fixed	No[1]	No	No	Yes	Yes	Emacs	No	Yes	Yes
CREF	Yes	Yes	Yes	No	No	By link	No	Yes	Zmacs	No	Yes	No
Emacs INFO	Yes	No	No	No	No	No	No	Yes	Emacs	No	No	No
IBIS	Yes	Yes	Yes	No	No	By link	No	No	A basic text editor	Yes	No	No
Intermedia	Yes	Yes	Yes	Yes	No[2]	No	No[2]	Yes	Custom	Yes	Yes	Yes
KMS	Multiple	Yes	Fixed	No	No[1]	Yes	Yes	Yes	Text/ graph. WYSIWYG	Yes	Yes	No
Neptune	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Smalltalk-80 editor	Yes	Yes	Yes
NLS/Augment	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Custom	Yes	Yes	No
NoteCards	Multiple	Yes	Yes	Nodes	No	No	Yes	Yes	Interlisp	Yes	Yes	Yes
Outline Processors	Yes	No	No	No	No	No	No	Yes	Various	No	No	No
PlaneText	Unix file sys.	Yes	No	No	No	No	No	Unix/ grep	SunView text ed.	Yes	Yes	Yes
Symbolics Document Examiner	Yes	Yes	No	No	Yes	No	No	Yes	None	No	No	No
SYNVIEW	Yes	No	No	No	No	No	No	No	line ed./ Unix	No	No	No
Textnet	Multiple	Yes	Yes	Yes	Yes	No	No	Keyword	Any	No	No	No
Hyperties	No	Yes	No	No	No	No	No	No[2]	A basic text editor	No	Yes	No
WE	Yes	Yes	No	Fixed	No[2]	No[2]	No[2]	No	Smalltalk-80 editor	No[2]	Yes	Yes
Xanadu	No	Yes	Yes	Yes	Yes	Yes	No	No	Any	No	Yes	No
ZOG	Yes	No	No	No	No	No	Yes	Full text	Spec. Pur.	Yes	No	No

Table 1. Hypertext Systems and Their Features

Hypertext technology has many advantages. Some of these include: ease of tracing references, ease of creating new references, information structuring, global views, customized documents, modularity of information, consistency of information, task stacking and collaboration. (Conklin, 1987, p. 38) These reasons make the technology an ideal choice for the development of an advisory system. The advisor prototype is designed to allow easy creation of help screen nodes and links. Modularity is critical for the simple interface required between the application program and the Browser software which will access requested help screens.

The main disadvantage of hypertext technology lies in orientation in complex systems. Getting 'lost in hyperspace' is the major pitfall software designers must ensure against. More work is needed on delivery of large-scale hypertext system. Graphical browsers and powerful query/search facilities are the main methods used to combat this problem. A second problem lies in the 'cognitive task scheduling problem' (Conklin, 1987, p. 40). Creating nodes and links, although vital to the strength of the hypertext concept, brings along with it the overhead of maintenance (creating, naming and keeping track of nodes and links). Again, graphical browsers help to solve this problem. Also fast system response time aids in keeping ideas present in the user's working memory. (Conklin, 1987, p. 40) The Advisor Browser uses a display which shows the user's current location and the path from the main menu to that point. The big picture should assist the user who is lost or has not found the answers they require along a particular route.

3.6 Method analysis and Alternative Methods

During the design process, the data structures remained flexible. Extra pointers were placed in link structures to accommodate later design decisions. Tools for navigation through the database were set up early, to allow new procedures to be written and changed easily. The use of prototyping is critical to the design of a good interface. The design process should be iterative; testing, modifications and more testing (Gould and Lewis, 1985; Burke, 1988). At each step of the way, changes should be made with a minimum amount of disruption. This philosophy also makes the end product more amendable to modifications.

The Advisor system has been designed to cater to both novices and experts. For example short cuts, using index list selection, have been defined for experts. Alternative methods for novices and experts allow novices to make use of a general purpose subset while on the way to becoming an expert. The novice may be content to stay at the upper help levels, with little need to traverse keyword links. Both the Advisor design and the help developers screen design will ensure that both novice and expert are accom-

modated. The provision of alternative methods allows 'incremental learning (Card, 1983, p. 423); as the novice gains more knowledge, they will pursue different problem solving strategies. Some diversity is provided with menu and command selection; keyword path search; and index access.

3.8 Error recovery methods

Some of the dialogue issues of the Advisor are addressed in defining the consequences of error conditions. Carroll and McKendree (1987) suggest some of the consequences could be: information (immediately feedback after an error); confirmation dialogue (protective measures to force the user to reaffirm their intention); control blocking (prevents the use of some parts of the software (eg. by novices)(Carroll and Carruthers, 1984); automatic correction (automatic spelling checker); protected mode (environment to investigate different solutions).

Error recover information is provided by both the Advisor Designer and Browser. The Designer also uses confirmation dialogue in critical situations. Blocking techniques have not been used in the Advisor. There did not seem to be a need to limit any users to a subset of the software's possibilities. Automatic correction techniques need more research before results can be effectively implemented. An UNDO facility has not been included in this version of help. Although there is merit in having an environment in which the user can explore without threat of destroying important data, the facility lends itself to a future thesis project due to its scope.

If the user is completely lost in hyperspace, the tree browser shows a hierarchy of help screen titles with a path to the current location highlighted.

3.9 Sensitivity

The help developer's assumption about the user must be tested. Pilot studies and acceptance tests can be used to decide whether the system is effective. One useful method is to establish a set of benchmarks to be completed in 30 minutes. If 80 percent of the test subjects can finish the benchmark, it could be considered a success.

Part II. The Advisor Designer

Chapter One: Functional Specifications

1.1 Summary

The Advisor Designer is used by the help developer to create the help system. The function of the Designer and the role of the help are shown in figure 1 - Advisor Overview. The Designer's purpose is to create help screens and links. Simple editing skills are sufficient to create or update a help database for a particular software package: text entry; deleting characters or lines of text; use of cursor control keys; use of insert and overwrite modes; find and replace functions; file saving; and printing reports. The help developer defines outline items which reference associated help screens. Standard outline functions include: creating and deleting help levels; collapsing and expanding headings and subheadings; and cursor movement throughout the outline.

Each outline level is automatically linked to one or more detail help screens. A function key gives access to the detail screen and the ESCape key is used to return to the outline screen. The structure of the outline provides the hierarchical links joining help screens in the database.

Nonhierarchical links are defined by the help developer as indexes and keywords. Indexes refer directly to the screen on which they are located. They will be displayed to the application end user on an index list for selection. Keywords are highlighted words within the text of the help screen and are used to link to alternate help screens.

The design goals of the Advisor Designer are:

1. to provide an outline processor for help screen development
2. to provide a help level hierarchy to allow the help developer to design help for novices and experts
3. to give options for examples of complex system functions
4. to be easy to use by software developers
5. to support hierarchical and non-hierarchical database structures
6. to support various linking mechanisms and node levels

1.2 Scenario - Help System Developer

The Advisor Designer is used by the help system developer to create the help outline, content and links (design goal one). It is a simple editor with some outline functionality including the ability to: create heading levels; collapse and expand headings, and subheadings; and move the cursor throughout the outline. Additional features include the ability to create links of various types: indexes and keywords.

A typical help session will show the ease of use of the Advisor Designer (design goal number 4). The help developer starts the Advisor Designer and provides filename which contains the help screens developed in a previous session. If the file exists, it will be read into a linked list using dynamic memory allocation, and displayed on the screen (see figure 2).

A menu is displayed on the bottom two lines (useful to jog the memory of the novice and infrequent user). The bottom menu line is cleared temporarily for error messages. Error messages provide the feedback required in a learning environment and have been designed with a positive tone, so as not to scold the user. The first item on each line is the outline level. These levels are indented, as lower outline levels of help are displayed, for quick viewing of the outline structure. This displacement provides a graphical view with colour coding enhancing the separation into sublevels. Following the level designation is the outline text, which will be used as the detail screen title is displayed beside each level.

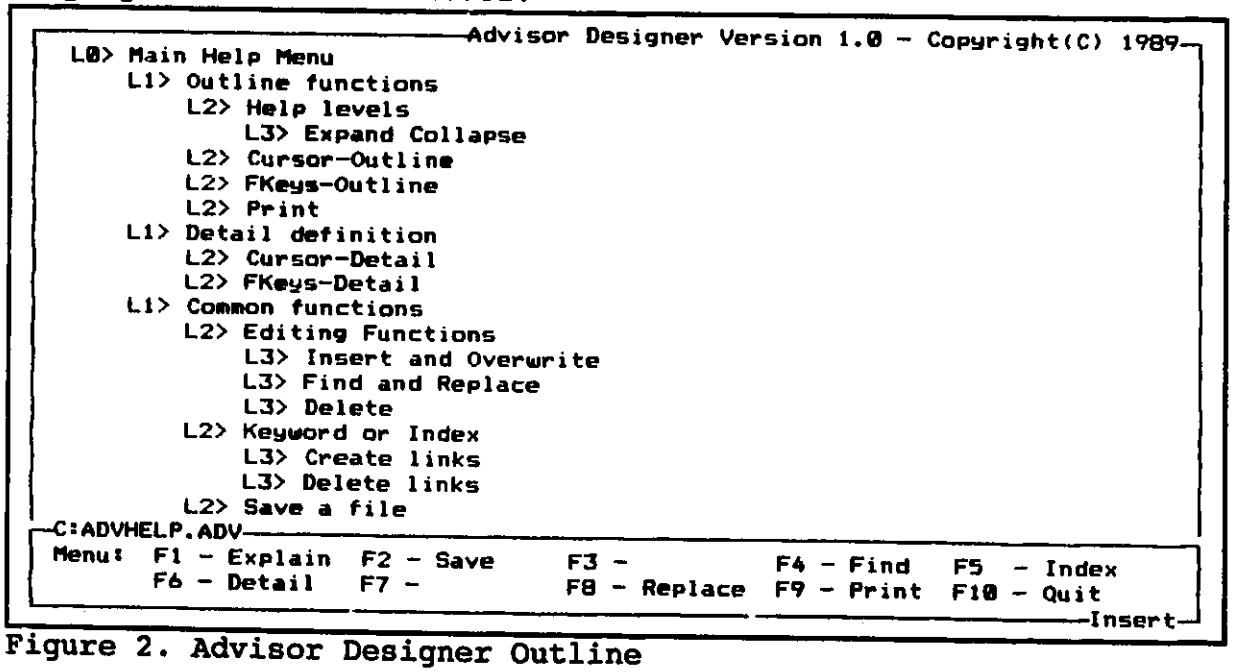


Figure 2. Advisor Designer Outline

New lines are created using the TAB key (new lower level) or the Carriage Return (new same level). Each outline item represents one node in the database and each outline level is a node type (design goal 6). The support structure is provided for the help developer to design upper level screens with routine information and lower levels for more detailed explanations of complex topics (design goal 3). Collapse and expand facilities allow as few or as many headings to be viewed as desired (see Table 2).

The Adviser Designer serves a dual purpose; outline editor and help screen detail editor. Cursor movement and function keys for the outline and detail screens are shown in Tables 2 and 3 respectively. Appendix F is a sample Advisor Designer function key keyboard template. A universality of commands and consistency of screen layout between the outline and detail screens was implemented to make the program easier to learn. The interface style chosen is both menu and command driven.

On the detail screen, help content is typed exactly as it will be seen by the application end user (see figure 3). The application screen is partially visible below the help screen as a reference point. Seeing both the application and help screen provides continuity; one of the most basic of the learning principles. After completing the detail screen definition, the user returns to the outline mode by pressing the ESCape key (resume) or the toggle key F6.

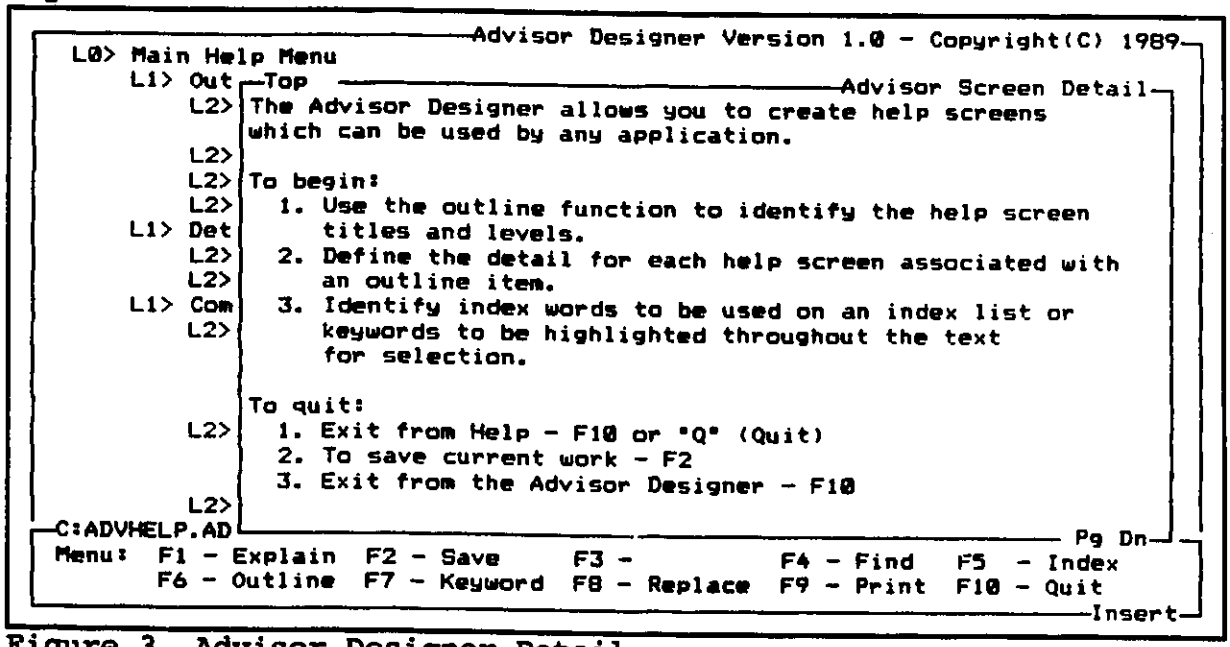


Figure 3. Advisor Designer Detail

Key	Outline	Detail
CR	Create new line same level	Create new line
Tab	Create new line next level	
Up arrow	Up one screen row	
Down arrow	Down one screen row	
Left arrow	Left one character	
Right arrow	Right one character	
Page up	Previous line same level	Previous help screen
Page down	Next line same level	Go to next help screen
CTRL Pg Up	Top of outline	First screen same outline item
CTRL Pg Dn	Go to bottom of file	Last screen same outline item
Home	First line same level	Go to beginning of line
End	Last line same level	Go to end of line
Insert	Toggle insert/overwrite	
Delete	Delete character under cursor	
Backspace	Delete character before cursor	
Shift Delete	Deletes line and detail screens	Deletes line
- (keypad)	Collapse subheadings under an outline item	
+ (keypad)	Expand subheadings under an outline item	
* (keypad)	Expand all headings	

Table 2. Advisor Designer keypad commands

Key	Outline	Detail
F1	Help	
F2	Save	
F3	not used	
F4	Find	
F5	Create index	
F6	Detail screen	Outline screen
F7	not used	Create keyword
F8	Replace	
F9	Print	
F10	Quit	
Ctrl F5	Delete Index	
Ctrl F7	not used	Delete keyword

Table 3. Advisor Designer function key descriptions

When the outline is completed and the detail screens are ready, the help developer has the option to create index names and keyword references. The Designer automatically creates links from the inherent hierarchical structure of the outline. Nonhierarchical links between index items and help screens; and highlighted keywords and help screens are set up by the help developer (design goal 5). Indexes are created by moving the cursor to the word to be indexed and pressing F5. The index is linked to the current outline item. These index words will form part of an index list available to the application end user.

Keyword links are identified throughout the text of the detail screens by moving the cursor to the intended word and pressing F7. When creating a keyword, the user is given an option to select an outline link from a list of outline items or enter the sequence ID associated with a given outline item. Figure 4 shows the menu for each of these options and the resulting screen from the outline list selection. The sequence ID's can be viewed by printing an outline listing (F9). Keywords link associated materials and attract attention to particular words. The help developer may use this facility to provide guidance on complex matters. The application end user selects a keyword from the detail screen and moves directly to another screen associated with this keyword. The use of the indexes and keywords allow the new user to investigate and explore the help system, while providing fast access to the screens requested by the expert user (design goal 2). After editing the file it is saved and a backup of the original(if one exists) is created.

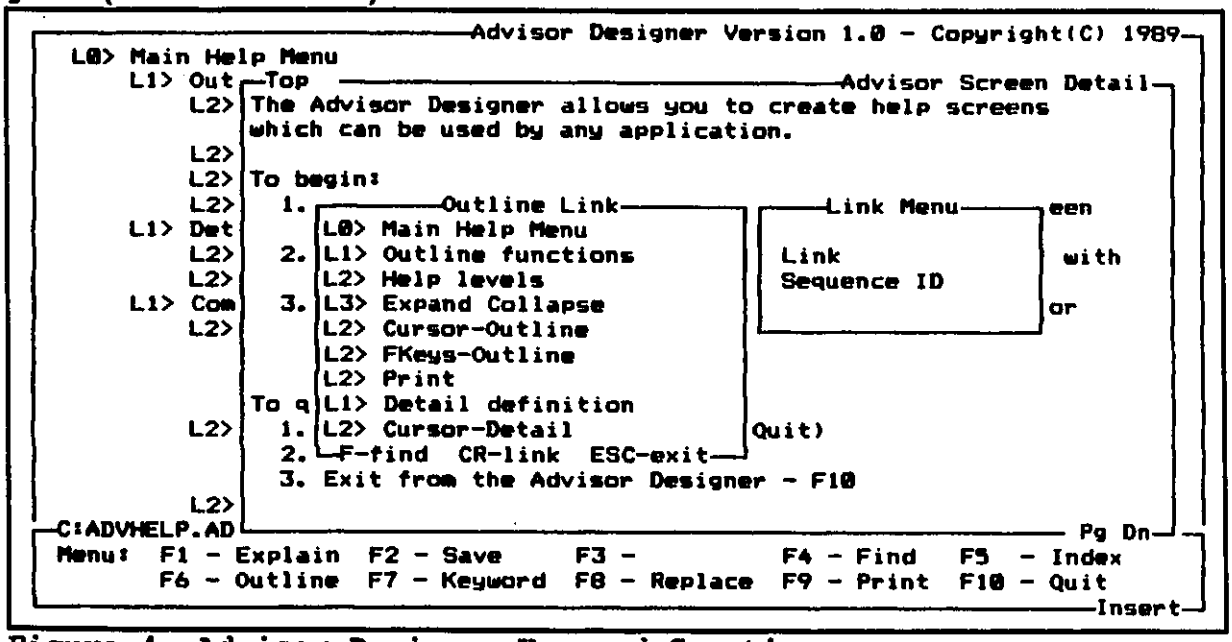


Figure 4. Advisor Designer Keyword Creation

Chapter Two: Software Design

2.1 Software Structure

The Advisor software has been broken down into modules according to criteria of functionality (see figure 5).

2.2 Designer Modules

Design goal 1 is the creation of the outline processor. The functionality of this facility was described in the functional specifications. Following is the modules used to create the Designer facility.

2.2.1 Unit A - Initializer

The Init unit contains routines required during the initialization process to:

1) initialize global variables
2) display initial window
3) open user specified help file

The following procedures are found in this unit:

```
procedure Initialize;
     Description    :    calls routines to initializes all global
                         variables, display initial screen with
                         copyright notice and opens the file.

procedure InitVar;
     Description    :    initializes all data structures.
```

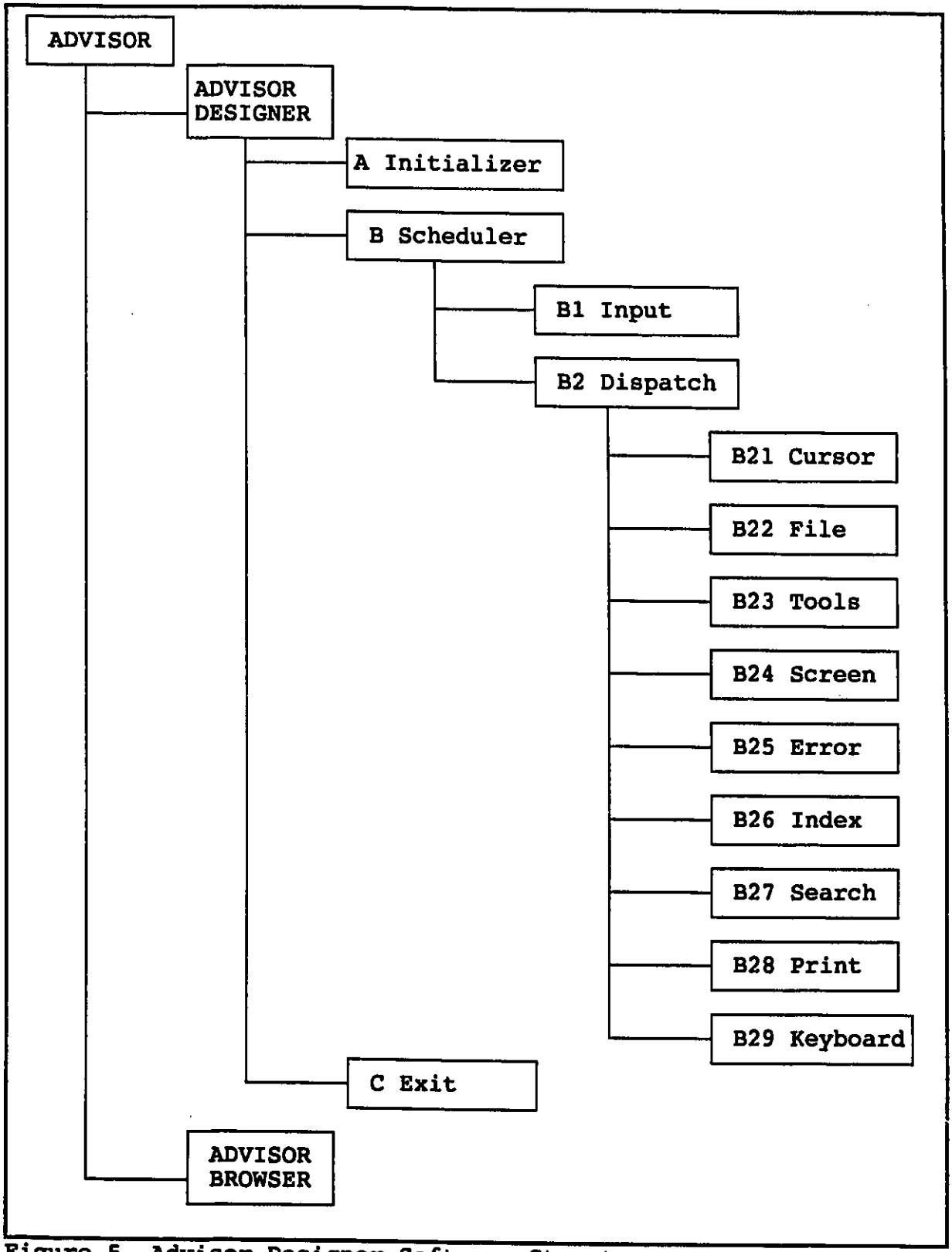


Figure 5. Advisor Designer Software Structure

2.2.2 Unit B - Schedule

The Schedule Unit has routines used to drive the main part of the Advisor Designer. The outliner is the main loop. It calls the Scheduler which initiates input classification. The terminating condition is set when the variable Finish is true.

The following procedures are found in this unit:

```
procedure InputClassifier;
     Description    :    Gets input, determines whether it is
                         text or command and takes the
                         appropriate action.

procedure Outliner;
     Description    :    Outliner driver calls scheduler until
                         variable Finish is true.


procedure Scheduler;
     Description    :    Calls classifier
                         (reserve possibility of background
                          processes)
```

2.2.2.1 Unit B1 - Input:

```
procedure TextProcessor;
     Description    :    processes text to linked list
```

2.2.2.2 Unit B2 - Dispatch:

```
procedure CommandProcessor;
     Description    :    dispatches commands received from the
                         Classifier.
```

Unit B21 - CursrCmd

```
procedure BottomDetail;
     Description    :    cursor to last page of detail for this
                         outline item.
procedure BottomFile;
     Description    :    cursor to last outline item;

procedure DetBeginningLine;
     Description    :    cursor to beginning of detail line
```

```
procedure DetEndLine;
     Description    :    cursor to end of detail line

procedure DetPageDown;
     Description    :    cursor to next detail page or create new
                         page

procedure DetPageUp;
     Description    :    cursor to previous detail page

procedure DownLine;
     Description    :    cursor control down one line

procedure LeftChar;
     Description    :    cursor control left one character

procedure OutEnd;
     Description    :    cursor to last item with same outline
                         level

procedure OutHome;
     Description    :    cursor to first outline item same level

procedure OutPageDown;
     Description    :    cursor to next outline item same level

procedure OutPageUp;
     Description    :    cursor to previous outline item same
                         level

procedure RightChar;
     Description    :    cursor control right one character

procedure TopDetail;
     Description    :    cursor to first line on detail screen
                         for current outline item

procedure TopFile;
     Description    :    cursor to first outline item

procedure UpLine;
     Description    :    cursor control up one line
```

Unit B22 - FileIO:

```
procedure CloseFile
     Description    :    closes the current file after asking the
                         user if the changes should be saved.

procedure DeleteFile(Dfilename:FnStr);
     Description    :    delete file (deletes old backup file
                         during save procedure)

procedure OpenFile;
     Description    :    opens a file named by the user.

procedure ReadColour;
     Description    :    Get colour scheme from customization
                         file

procedure ReadFile (Var Fn:FnStr);
     Description    :    reads files' contents  into linked lists


procedure Rename(OldFn,NewFn:FnStr);
     Description    :    renames old to new filename (Save)

procedure SaveFile;
     Description    :    writes to a file and closes it.
```

Unit B23 - OutTools:

```
procedure DeleteChar;
     Description    :    delete character under cursor

procedure DeleteDetline;
     Description    :    delete detail line

procedure DeleteDetlineStruc;
     Description    :    delete detail link structure and
                         reassign remaining links

procedure DeleteDetlink;
     Description    :    delete outline link to detail and all
                         attached detail lines

procedure DeleteOutline;
     Description    :    delete outline line and all associated
                         detail screens
```

```
procedure DeleteOutlineStruc;
     Description    :    delete outline link structure and
                         reassign remaining links

procedure ExpandContract;
     Description    :    expand/collapse outline screen display

procedure Genid;
     Description    :    generate unique sequence number for each
                         outline line

procedure GetFirstSameLevel;
     Description    :    get first outline item at same level

procedure GetLastChild;
     Description    :    get the last child of current outline
                         line

procedure GetLastSameLevel;
     Description    :    get last outline item at same level

procedure GetNextLink;
     Description    :    get next link in list

procedure GetNextScreen;
     Description    :    get next detail screen

procedure GetNextSameLevel;
     Description    :    get next outline item same level

procedure GetParent;
     Description    :    get parent of current outline line

procedure GetPrevLink;
     Description    :    get previous link in list

procedure GetPrevSameLevel;
     Description    :    get previous outline item at same level

procedure GetPrevScreen;
     Description    :    get previous detail screen

function HeapFunc(Size:word):integer;
     Description    :    causes New to return nil if unsuccessful

procedure IncrLevel;
     Description    :    increase outline level.
```

```
procedure InitDetail;
     Description    :    initializes a detail link

function InsertDetLine:boolean;
     Description    :    insert a new detail text line.

procedure InsertMode;
     Description    :    toggles between insert/overwrite.
                         Displays status on screen.

function InsertOutLine:boolean;
     Description    :    insert a outline text line.

procedure NewLine;
     Description    :    create a new outline or detail line
```

Unit B24 - Screen:

```
procedure DetailScreen;
     Description    :    display and initialize detail screen

function DetScreenFull:boolean;
     Description    :    returns true if detail screen full

procedure DetUpdateScreen;
     Description    :    updates detail screen

procedure DetWriteLine(DetRow:byte);
     Description    :    updates one detail row

procedure DisplayPgUpDown(UpDown:integer);
     Description    :    displays page up/down on detail screen

function OnCurrentScreen:integer;
     Description    :    returns screen row of current outline
                         link; otherwise zero

procedure OutUpdateScreen;
     Description    :    updates entire outline screen

procedure OutWriteLine;
     Description    :    updates one outline line

procedure ReturnOutline;
     Description    :    return to outline screen from detail
                         screen
```

Unit B25 - Error:

```
procedure DisplayError(ErrorNum:byte);
     Description     :     displays error message

function GetMessage(MsgNum:byte)Str80;
     Description     :     returns error message from error file
```

Unit B26 - Index:

```
procedure AssignLink;
     Description     :     assign outline item link to keyword

function BeginWordCol:byte;
     Description     :     get buffer column for beginning of
                           current word

procedure DeleteIndex;
     Description     :     delete index/keyword (Dispatch command)

procedure DeleteIndexStruc;
     Description     :     delete index/keyword link structures and
                           reassign links

procedure DelIndex;
     Description     :     delete index (DeleteIndex,Sort,Flush)

LinkMenu;
     Description     :     display link menu options

procedure DisplayLinkOutline;
     Description     :     display link outline item selection list

function FindIndex(VarKeywd:IndexStr;KeyType:char):boolean;
     Description     :     search index list for given word

procedure FlushIndex;
     Description     :     flush all index/keywords associated with
                           detail or outline line just deleted

procedure GetReferenceWord(Var Keywd:IndexStr);;
     Description     :     pick up word to be indexed or made into
                           a keyword

procedure GetSequenceID;
     Description     :     request and pick up sequence ID to be
                           used for keyword link
```

```
procedure IndexProcessor;
     Description    :    process index definition

procedure InitLinkMenu;
     Description    :    initialize link menu

function InsertIndex:boolean;
     Description    :    insert index/keyword into list; true if
                         successful

procedure KeyWordProcessor;
     Description    :    process keyword definition

procedure LinkItem;
     Description    :    select outline item to be linked

function LowerCase(ch:char):char;
     Description    :    returns lowercase character

procedure OGetWordLength;
     Description    :    returns word length

procedure OInitFind;
     Description    :    initializes find procedure

function OMatchLetter(VAR Keywd:FindStr;StartCol:byte):byte;
     Description    :    returns column of match, otherwise zero

function OMatchWord(VAR Keywd:FindStr;MatchCount:byte):byte;
     Description    :    returns end column of matched word or
                         zero

procedure OoutSearch(VAR FindEnd:integer);
     Description    :    returns last column of found outline
                         word, otherwise zero

procedure OutWriteLink(LinkRow,Highlight:byte);
     Description    :    display outline row for link selection

function ShiftRightChar:boolean;
     Description    :    returns true if shift operation
                         successful

function ValidateKey(VKey:byte):boolean;
     Description    :    returns true if link key exists in
                         outline
```

Unit B27 - Find/Replace:

```
procedure Find;
     Description    :    search outline or detail for requested
                         word

function GetWordLength(VAR Keywd:FindStr):byte;
     Description    :    returns length of keyword

procedure InitFind;
     Description    :    initializes Find procedure

procedure InitReplace;
     Description    :    initializes replace procedure


function MatchLetter(VAR Keywd:FindStr;StartCol:byte):byte;
     Description    :    returns column of match, otherwise zero

function MatchWord(VAR Keywd:FindStr;MatchCount:byte):byte;
     Description    :    returns end column of matched word or
                         zero

procedure OutSearch(VAR FindEnd:integer);
     Description    :    returns last column of found outline
                         word, otherwise zero

function ProcessFind:boolean;
     Description    :    returns true if requested word is found

procedure ProcessReplace;
     Description    :    carries out replace operation

procedure Replace;
     Description    :    calls replace if find succeeds
```

Unit B28 - Print:

```
procedure OutPrint;
     Description    :    prints outline with sequence IDs
```

Unit B29 - KeyMouse:

```
function KeyMousePressed:boolean;
     Description    :    true if keypressed

procedure Mouse(VAR M1,M2,M3,M4,M5:integer);
     Description    :    handles mouse IO

function ReadKeyMouse:integer;
     Description    :    returns integer for keyboard key

function SpecialKeys:integer;
     Description    :    deals with special key combinations
```

2.2.3 Unit C - Exit

The Exit unit holds a procedure for wrapping up all loose ends, clearing the screen, returning calling attributes and exiting the program.

The OExit procedure is as follows:

```
procedure OutExit;
     Description    :    wraps up any loose ends and exits
                         program
```

2.3 Data Structures

The Advisor Designer has two major data structures used for outline/screen storage and index/keyword storage.

2.3.1 Outline/Screen storage

A tree-shaped list stores all text from the outline and detail screens. The hierarchical structure of the outline naturally imposes this image on the data structure. Each node in the list represents an outline item. Figure 6 gives a high level picture of the data structure. Low level pictures of the outline and detail links are provided in figures 7 and 8. The link types for the outline structure include:

1. Forward/Backward links - to the next/previous nodes; regardless of help level.

2. Next/Previous same links - to the next/previous nodes; at the same level as the current node.

3. Detail links - to the first line of the detail screen associated with that outline item.

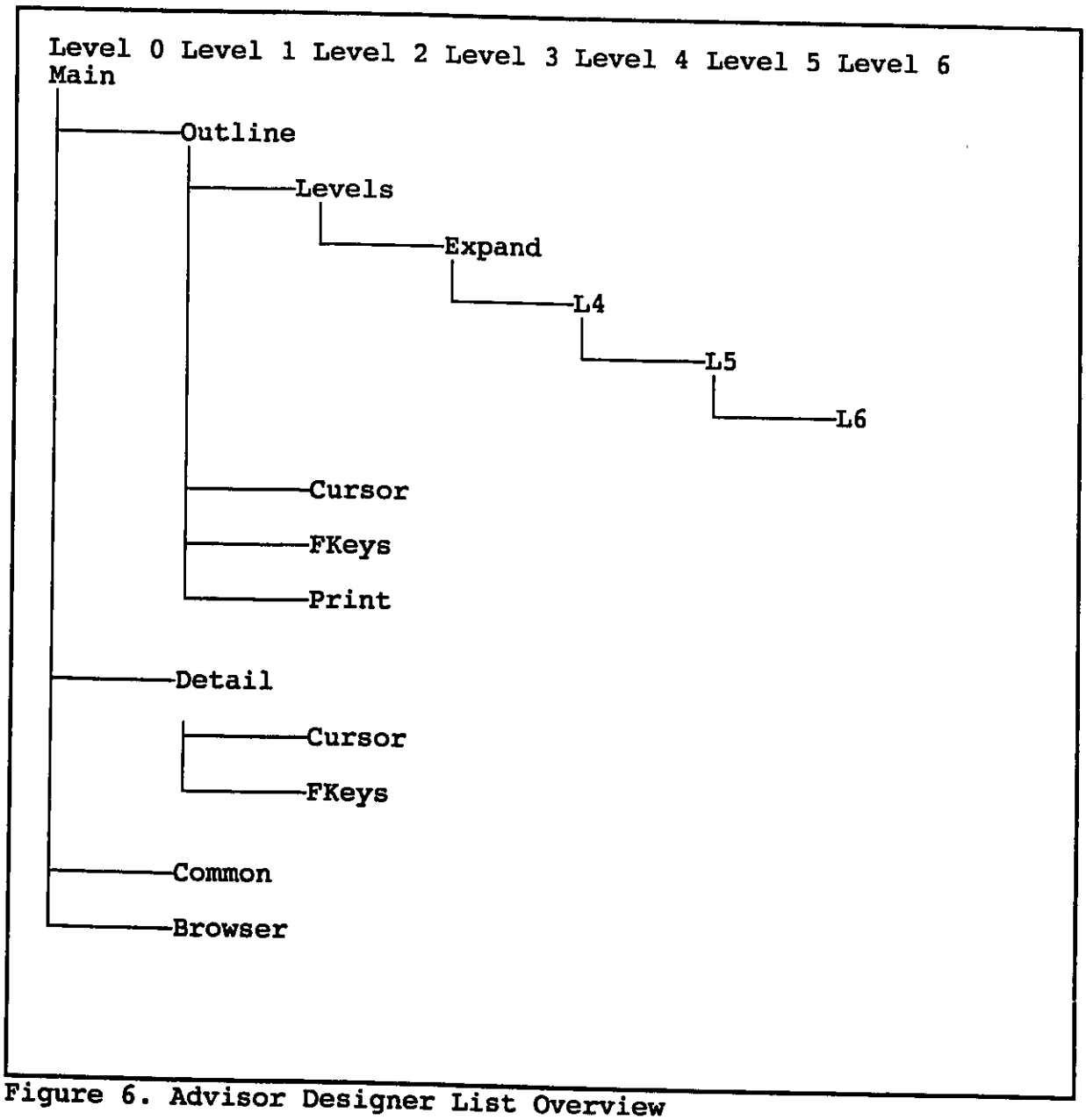


Figure 6. Advisor Designer List Overview

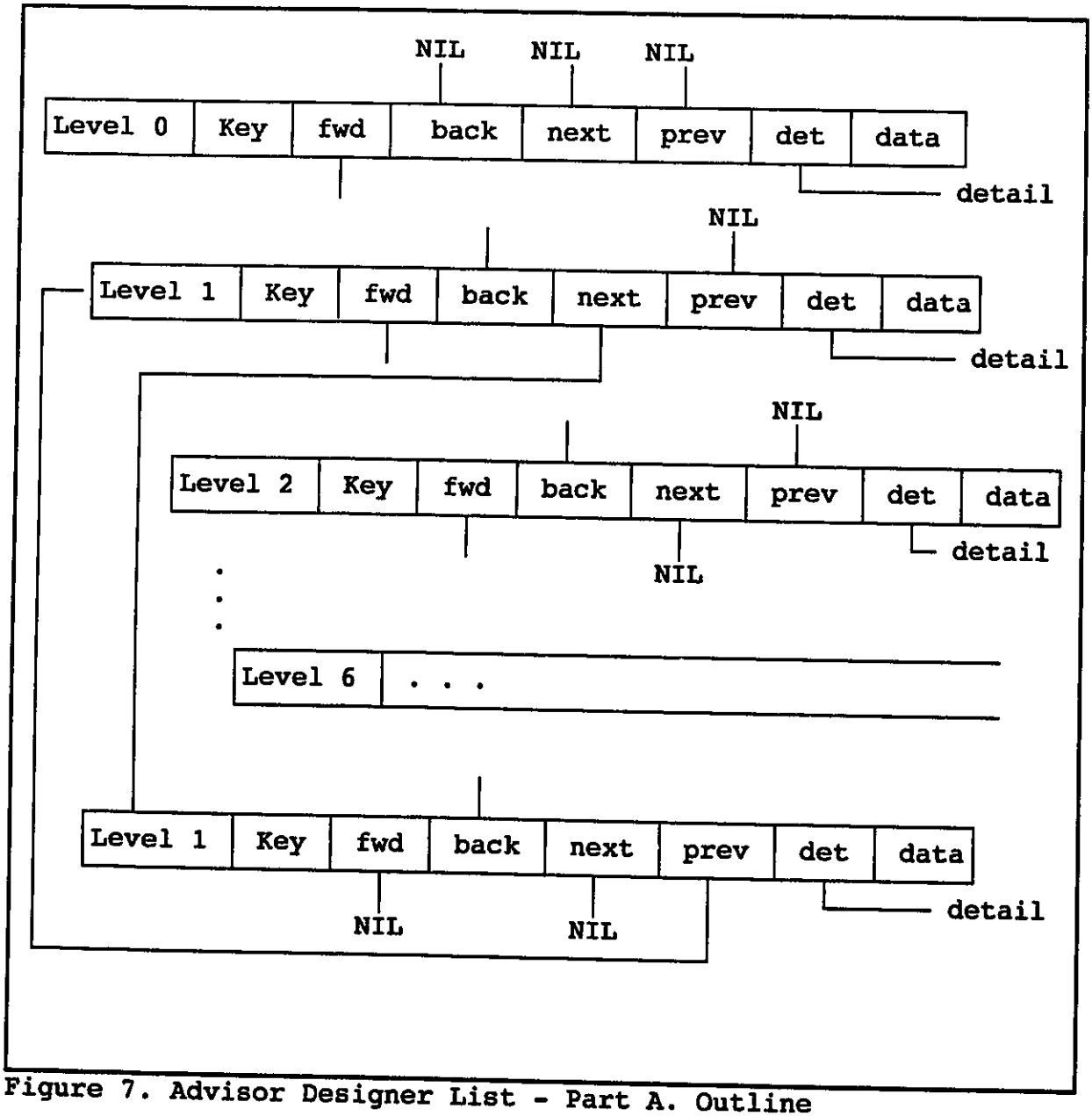


Figure 7. Advisor Designer List - Part A. Outline

Following the link (Detlink) from the outline node to the first detail line leads to an embedded list. Each detail link is a node with forward and backward links to each text line.

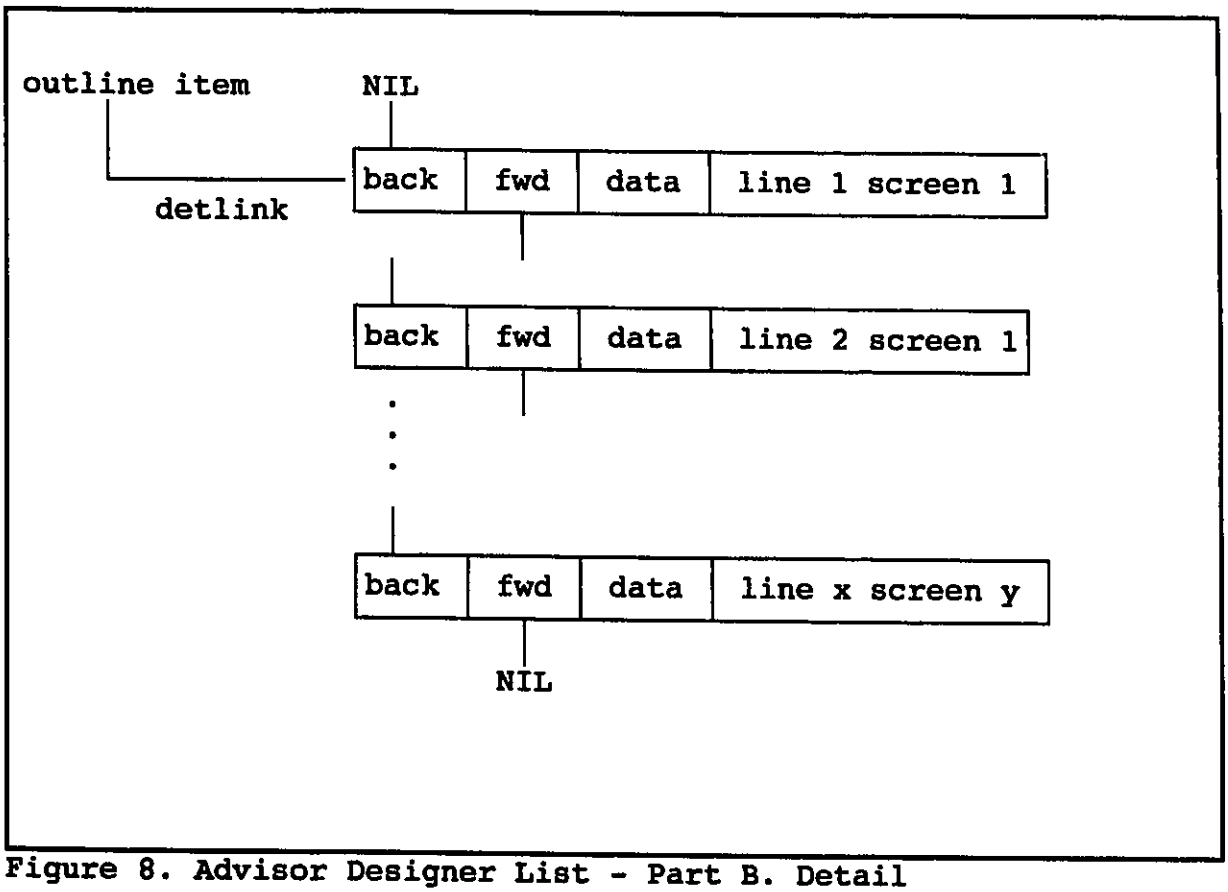


Figure 8. Advisor Designer List - Part B. Detail

2.3.2 Index/Keyword storage

A deque (double ended queue) is used to keep track of index and keyword assignments. This allows easy movement in either direction through the list. Each node in the list is one index word or keyword (variable keytype defines which link type). Figure 9 describes the list structure. SrcKey refers to the outline item associated with the keyword screen and the outline item which is linked to when a keyword is accessed is Ptrkey. These variables allow for the deletion of indexes/keywords when an outline line or detail line is deleted. When read by the Advisor Browser, these items form the basis for the creation of index and keyword links which give the non-hierarchical graphical links of the help database.

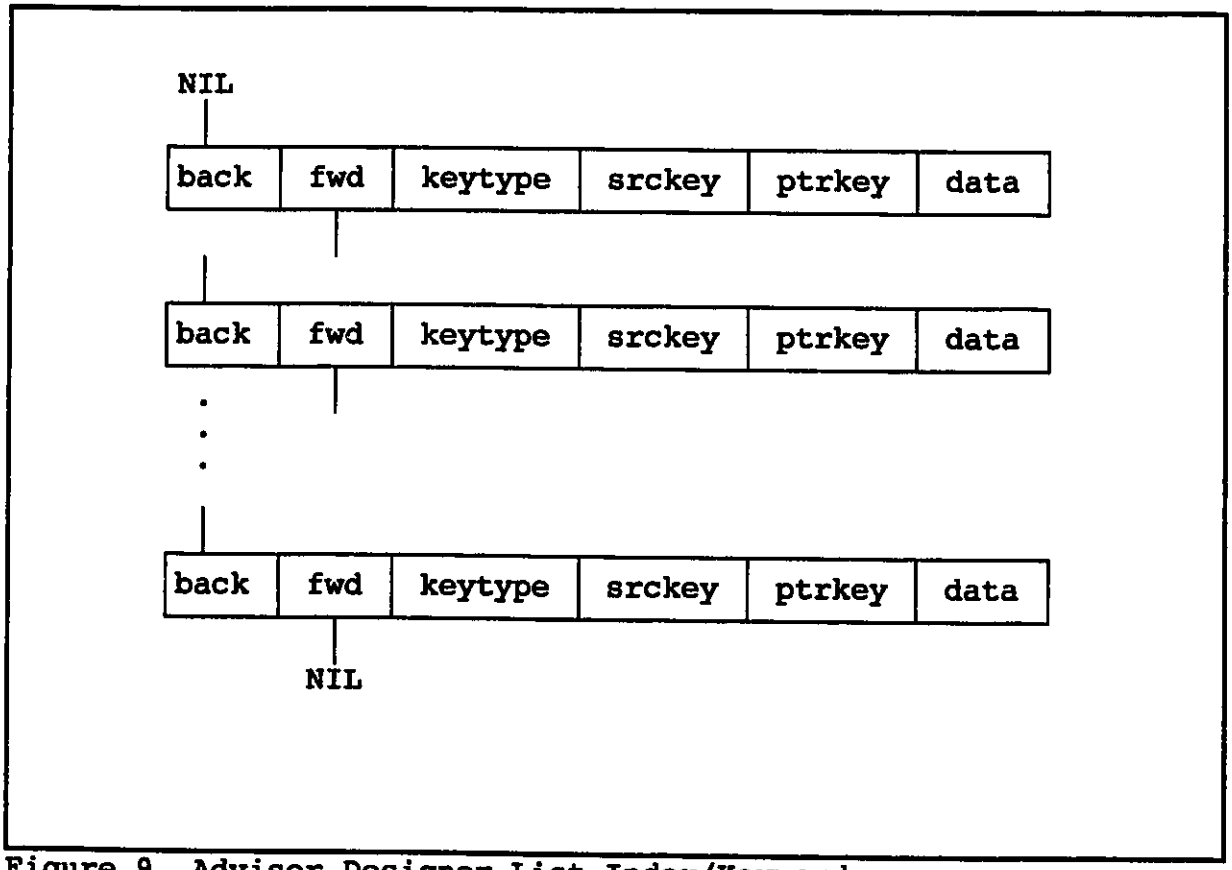


Figure 9. Advisor Designer List Index/Keyword

Part III. The Browser

Chapter One: Functional Specifications

1.1 Summary

The Browser is used for viewing help screens. Figure 1 - Advisor Overview, shows the relationship between the Advisor Designer and Browser; and where the application end user fits into the picture.

The Browser provides viewing of help to the application user in four ways: by following links; selection from an index list; selection of highlighted keywords from the help screen; and viewing a tree browser (tree structure with highlighted path showing location). Hierarchical links are followed by using the next and previous options from the main help menu. The index option displays an index list of help topics. Positioning the cursor on highlighted keywords on the detail screen and pressing the F1 links the help user with an associated topic. For a general view of the help location, the tree browser can be used. Window management of the tree browser gives the ability to change window size; access and overlay windows.

The help structure and links facility of the Advisor Designer gave the help developer the opportunity to provide depth and quick access for the expert; and upper level basic information to the novice. The software allows for the growth in knowledge of the user by layering the help to provided more detailed examples at lower levels; to be accessed as the user becomes more experienced.

The design goals of the Advisor Browser are:

1. design a help browsing facility to access the screens developed using the Advisor Designer.
2. give accessibility to help hierarchy for novice and experts
3. support hierarchical and non-hierarchical database links
4. support various link and node types
5. provide browser which shows nodes and links in a tree structure with the current path highlighted
6. provide keyword search capabilities

1.2 Scenario - Browser User

The Advisor Browser can be accessed by the application user by pressing a key (eg. F1); as implemented by the application programmer. Part of the application is visible; keeping the user in touch with the reason they needed help. The help user may browse in one of the following ways:

1. Following links by using the Next/Previous functions

2. Searching for a help topic - from an Index list

3. Select keyword to link to another help screen

4. Tree browser - displays entire help system and highlights the current path.

A typical example of a help session is as follows. The user presses F1 to access the main help menu. Menu selection is made by moving the cursor to a menu item; selection of first letter of the menu item; or a function key (see figure 10). Next and Prev cause the next and previous levels of help to be displayed. Tree is used to show the current location within the help system. Index provides an index list, from which to make a selection. Detail switches the cursor to the detail screen to make keyword selections. Quit allows exit to the application program. Resume takes the user back along the current help path. The menu is a useful reminder of the help facilities. A customization program allows colour selection.

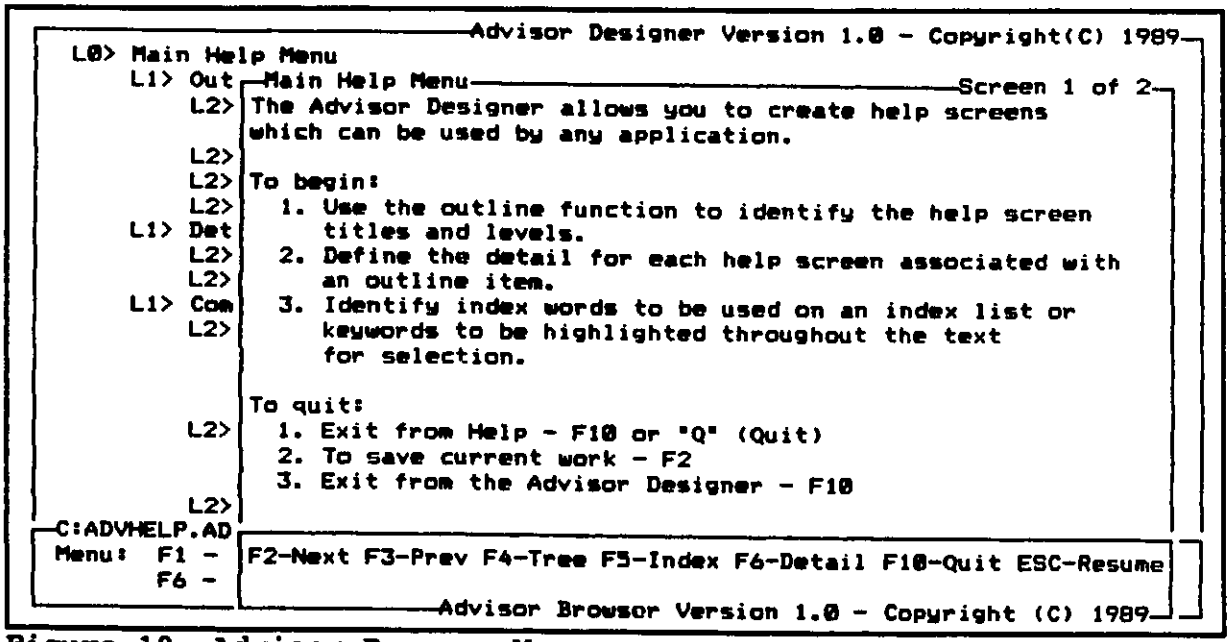


Figure 10. Advisor Browser Menu

The Browser path, showing the location of the current help screen within the help system, can be viewed by selecting the Tree option from the help menu. The display is divided up into seven sections: one across the top for the Level 0 (Main Help) title; and the rest for levels one through six help outline and detail. The Browser highlights the path (parent outline items) to the current help screen and displays the detail for that level in the appropriate segment of the screen. The location of the user in the overall help structure is shown as a highlighted path. Using Alt 1 through 6 gives direct access to any of the windows from level 1 to 6. The window can be 'zoomed' to full size and back again using the Zoom menu option. Design goal 5 is the development of a browser which shows nodes and links of the database with the current path highlighted. This facility gives the user a picture of the help hierarchy and location within that structure. It helps the individual who is 'lost in hyperspace'. Figure 11 shows an example where the user is coming from help level three. If Alt-3 is used to select the level 3 window and the Zoom option is chosen, the information from that screen can be blown up to full help screen size. To exit from the path display the ESCape key is used.

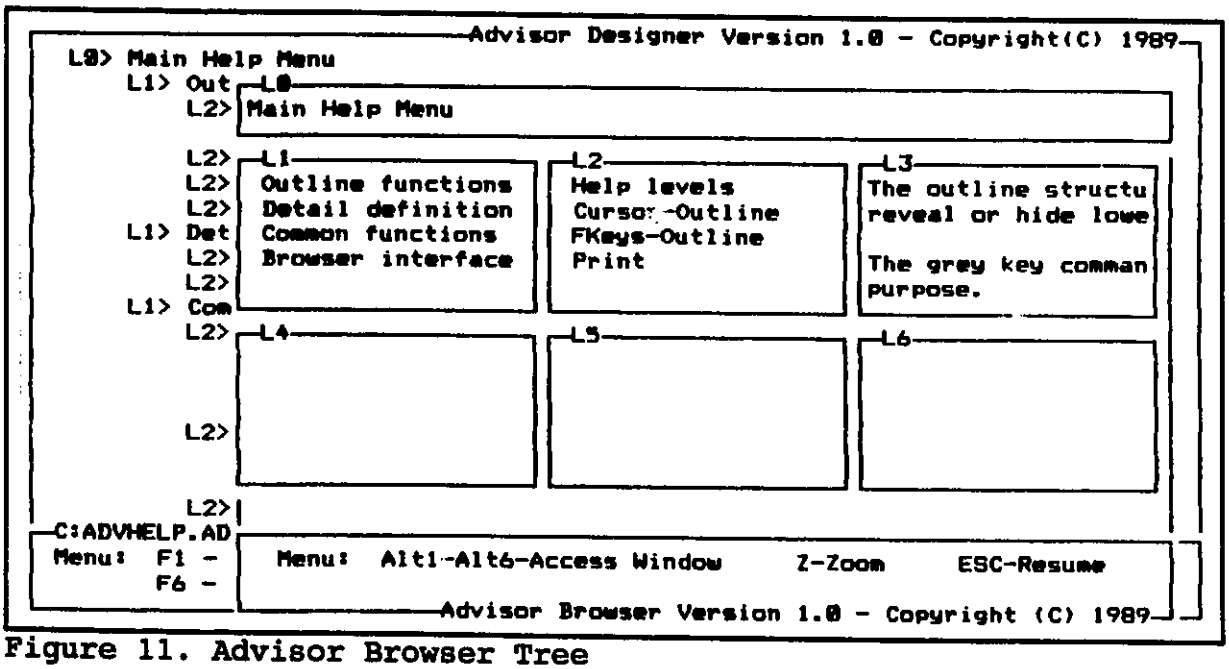


Figure 11. Advisor Browser Tree

The reference index list can be displayed in the lower half of any help screen by selecting the Index option on the menu. The resulting screen will display three columns of index items with the first item highlighted. The index search capability is useful to either the novice or expert (design goal 6). The novice may not understand the breadth of the help facility and need the index list to show all main topics. The expert wants to move quickly to a topic and finds that the index selection provides the quickest route. Figure 12 shows the appearance of the Index help screen. If the index screen is full, Page Down and Page Up keys can be used to view all of the indexes. To exit from the Index screen, the ESCape key is used.

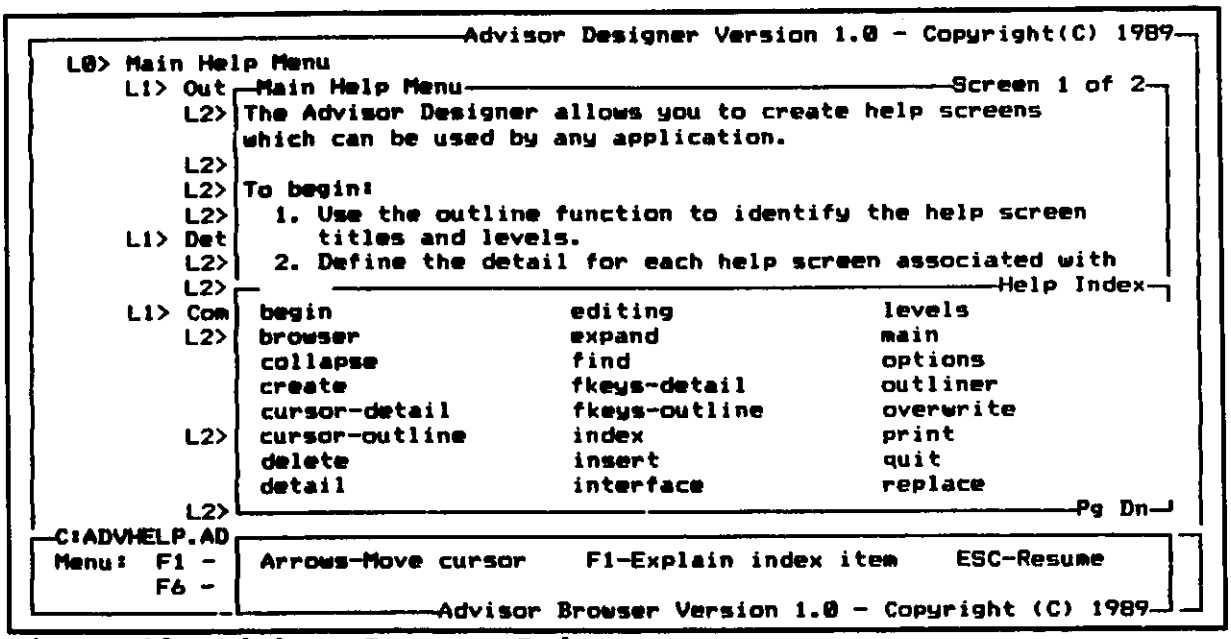


Figure 12. Advisor Browser Index

Chapter Two: Software Design

2.1 Software Structure

The Advisor software has been broken down into modules according to criteria of functionality (see figure 13). Modularity is essential as the Advisor is intended to interface a wide variety of applications. In keeping with design goal 9, modularity has been a top priority.

2.2 Browser Modules

2.2.1 Unit A - Initializer

The Browser Unit holds the initialization routines which are used to

1) initialize global variables

2) display initial window

3) read user requested help file and display requested level of help

4) call routine to set up menu for further access of help screens.

The following procedures are found in this unit:

```
Unit Browser;

function BrHeapFunc(Size:word):integer;
     Description    :    causes new to return nil if unsuccessful

procedure brInitVars;
     Description    : initializes all global
                      variables, display initial screen with
                      copyright notice and opens the file.

procedure Browse(BRFn:Str80;BRHLev:Integer);
     Description    :    calls procedures to initialize all
                         global variables, display initial screen
                         with copyright notice and opens the
                         file.
```

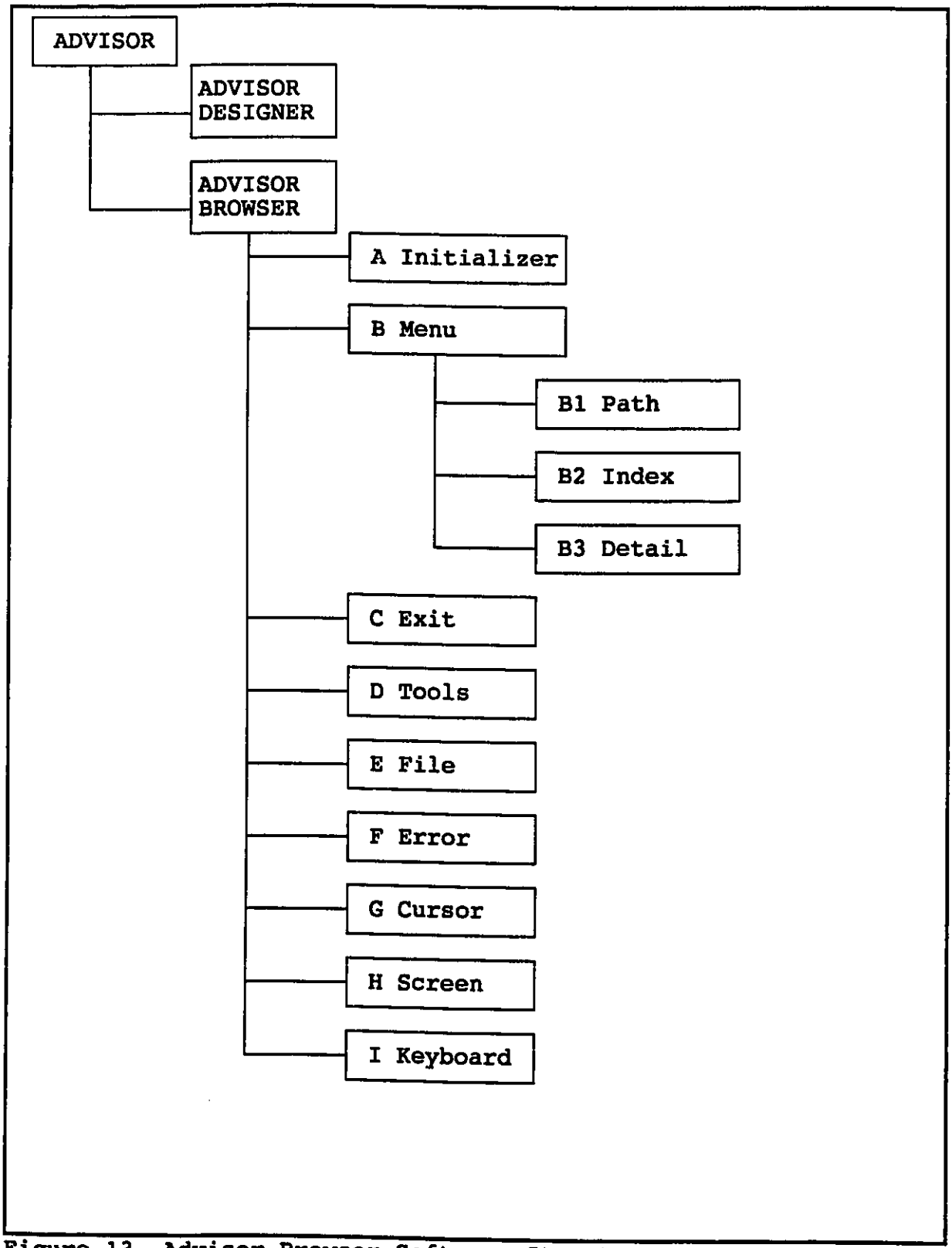


Figure 13. Advisor Browser Software Structure

2.2.2 Unit B - Menu

The Menu has the routines which drive the Browser to select and display help records.

Procedures and functions are as follows:

```
Unit BRuMenu;

procedure brDisplayMenu(Selection:byte);
     Description    :    display menu

procedure brInitMenu;
     Description    :    initialize menu

procedure brExitMenu;
     Description    :    exit from menu

procedure BrMenu;
     Description    :    process menu selection
```

2.2.2.1 Unit B1 - Path

The Path Unit displays the various help levels followed to obtain the current record. The routines it uses to do this are:

```
Unit BRPath;

procedure BrDisplayPath;
     Description    :    display outline windows

procedure brMainMakeWindow;
     Description    :    display main window

procedure brPathControl;
     Description    :    access and zoom windows

procedure brPathDetail;
     Description    :    display detail window

procedure brPathMakeWindow;
     Description    :    make path windows
```

2.2.2.2 Unit B2 - Index

The Index Unit is used to display the Index linked list and make selections according to the users request. Its routines are as follows:

```
Unit BRIndex;

procedure brDeleteIndex;
     Description    :    delete index; used during sort

procedure BrDisplayIndex;
     Description    :    display index list

procedure brInitIndex;
     Description    :    initialize index display

procedure brInsertSortPtr;
     Description    :    insert sort pointer

procedure brListIndex;
     Description    :    list index items on screen

procedure brMarkIndex(select:byte);
     Description    :    mark selected item

procedure BrPageDown;
     Description    :    go to next index page

procedure BrPageUp;
     Description    :    get previous index page

procedure BrSortIndex;
     Description    :    sort index list alphabetically
```

2.2.2.3 Unit B3 - Detail

Handles movement on detail help screen during selection of keyword.

```
function BrBeginWordCol:byte;
     Description    :    returns buffer column for beginning of
                         word

function BrFindKeyword(VAR BrKeywd:BrIndexStr):boolean;
     Description    :    search keyword list for given word.
                         Return true if found.
```

```
procedure BrGetReferenceWord(VAR BrKeywd:BrIndexStr);
     Description    :    picks up word from screen to use as
                         keyword

function BrPathBackTrack:boolean;
     Description    :    returns true if path stack not empty

procedure BrPop(VAR BrTopStack:PbrPathList);
     Description    :    pop element off top of path stack

procedure BrPush(VAR BrTopStack:PbrPathList);
     Description    :    push item onto path stack

procedure BrSelectKeyword;
     Description    :    select keyword link and display

procedure BrSelectWord;
     Description    :    pick up word from screen and process
                         keyword (or later glossary word)

procedure BrSwitchDetail;
     Description    :    switch to detail screen
```

2.2.3 Unit C - Exit

The exit unit is responsible for clearing Browser and returning to the application calling program.

```
procedure BrExit;
     Description    :    wraps up any loose ends and exits
                         program
```

2.2.4 Unit D - Tools

Routines in this unit are used to insert links in the list, get particular links on the list and display nodes once found. The routines are as follows:

```
UNIT BRTOOLS;

procedure BrDisplayNext;
     Description    :    display next detail screen

procedure BrDisplayPrevious;
     Description    :    display previous detail screen
```

```
procedure BrGetFirstSameLevel;
     Description    :    get first outline item same level

procedure BrGetLastSameLevel;
     Description    :    get last outline item same level

procedure BrGetNextLink;
     Description    :    get next outline link in list

procedure BrGetNextSameLevel;
     Description    :    get next outline item same level

procedure BrGetNextScreen;
     Description    :    get next detail screen

procedure BrGetParent;
     Description    :    get parent of current outline item

procedure BrGetPrevLink;
     Description    :    get previous list link

procedure BrGetPrevSameLevel;
     Description    :    get previous outline item same level

procedure BrGetPrevScreen;
     Description    :    get previous detail screen

procedure BrGetTopScreen;
     Description    :    get top of detail screen

function BrInsertDetLine:boolean;
     Description    :    returns true if detail insertion
                         successful

function BrInsertIndex:boolean;
     Description    :    returns true if index insertion
                         successful

function BrInsertKeyword:boolean;
     Description    :    returns true if keyword insertion
                         successful

function BrInsertOutLine:boolean;
     Description    :    returns true if outline insertion
                         successful

function BrLowerCase(Ch:char):char;
     Description    :    returns lower case character

procedure BBrPush(VAR BrTopStack:PbrPathList);
     Description    :    add item to top of path stack
```

2.2.5 Unit E - File IO

This unit has routines for reading the main help and index files; and creating linked lists for each. The routines it uses are:

```
Unit BRFileIO;

procedure BROpenFile(var BRFilename:Str80; BRHelpLevel:Integer);
     Description    :    opens help file

procedure BrReadColour;
     Description    :    reads customization file and picks up
                         colour scheme

procedure brReadFile(var BRFilename:Str80; BRHelpLevel:Integer);
     Description    :    reads files' contents  into linked lists
```

2.2.6 Unit F - Error

The error unit is used to read the error file and display the appropriate error message.

```
procedure BrDisplayError(BrErrorNum:byte);
     Description    :    displays error message

function brGetmessage(BrMsgNum:byte):BrStr80;
     Description    :    returns error message(associated with
                         error number) from error file
```

2.2.7 Unit G - Cursor Control

Handles cursor movements on the detail help screen when searching for keywords.

```
procedure BrDetBeginningLine;
     Description    :    go to beginning of detail line

procedure BrDetEndLine;
     Description    :    go to end of detail line

procedure BrDownLine;
     Description    :    cursor control down one line

procedure BrGetNextKeyword;
     Description    :    cursor control to next keyword

procedure BrLeftChar;
     Description    :    cursor control left one character
```

```
procedure BrRightChar;
     Description    :    cursor control right one character

procedure BrUpLine;
     Description    :    cursor control up one line
```

2.2.8 Unit H - Screen Control

Handles screen displays.

```
procedure BrDisplayDetail;
     Description    :    display detail screen

procedure BrDetWriteLine(BrDetRow:byte);
     Description    :    display the detail line

procedure BrDetWriteTitle;
     Description    :    display detail title
```

2.2.9 Unit I - Keymouse

Handles all mouse and keyboard activity.

```
function KeyMousePressed:boolean;
     Description    :    true if key pressed

procedure Mouse(VAR M1,M2,M3,M4,M5:integer);
     Description    :    mouse driver interface

function ReadKeyMouse:integer;
     Description    :    returns key pressed

function SpecialKeys:integer;
     Description    :    determines if special combination of
                         keys pressed
```

2.3 Data Structures

The Advisor Browser uses three major data structures for: outline/screen storage; index/keyword storage; and current path storage.

2.3.1 Outline/Screen storage

The structure is the same as that found in the Advisor Designer (see Part II Chapter 2).

2.3.2 Index/Keyword storage

Indexes and keywords are put into two separate lists when the index/keyword file is read. This facilitates searches by reducing the number of links to be checked. Also, since the index list is sorted alphabetically, it is more efficient to keep the list size small.

Both lists have the same structure. As with the Advisor Designer's index/keyword list, these lists are doubly linked. In addition, each node has a link to the outline item referenced in the index file by PtrKey; the outline item which is to be accessed upon selection of the index or mainlink. A diagram of the list structure is shown in figure 14.

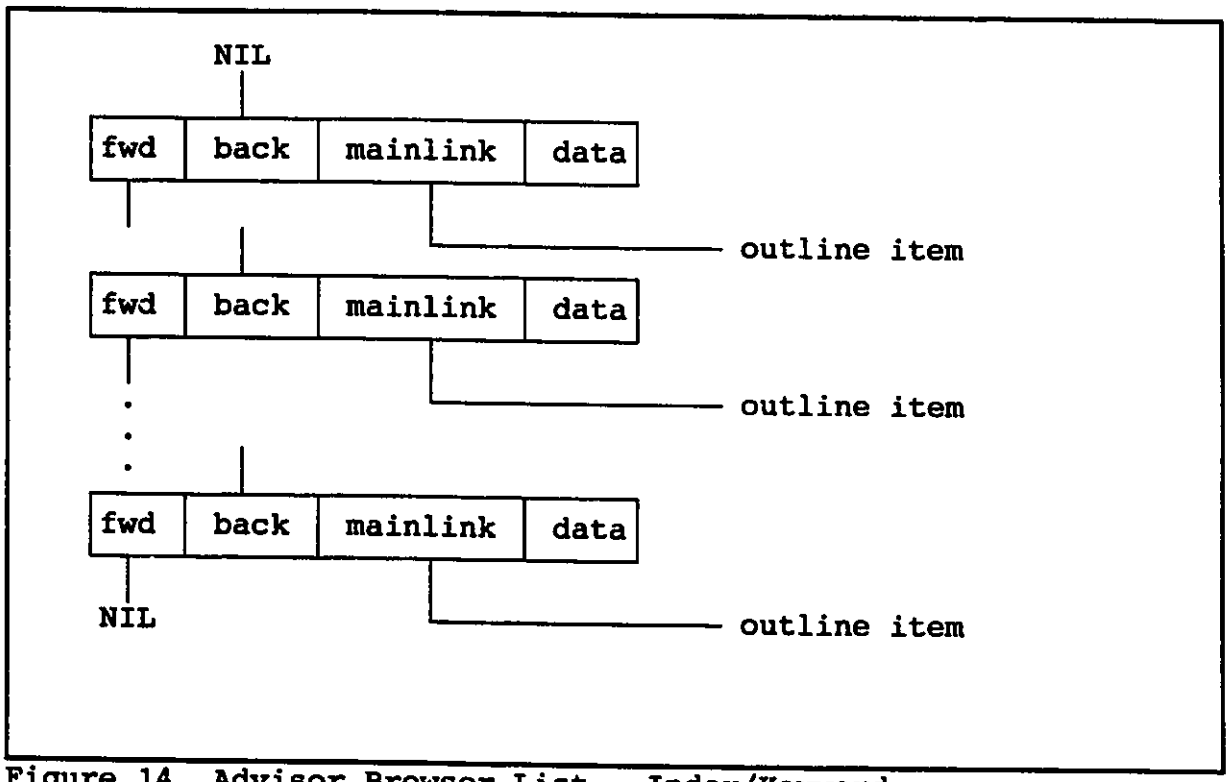


Figure 14. Advisor Browser List - Index/Keyword

2.3.3 Current path storage

The current path is stored in a stack structure. The first in/last out structure works well for pushing detail screen references onto the stack as more help screens are visited; and popping them off as the path backtracking operation is requested. Figure 15 shows the current path stack structure.

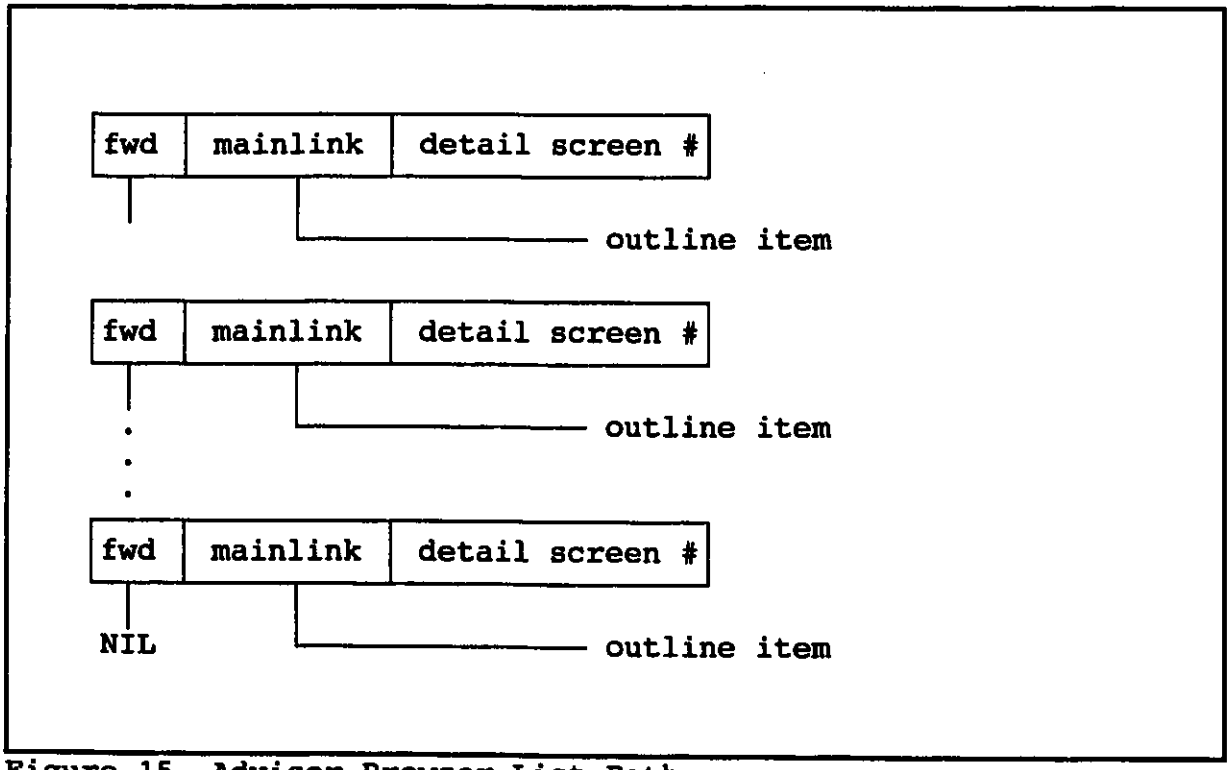


Figure 15. Advisor Browser List Path

Part IV. The Browser Interface

Chapter One: Functional Specifications

1.1 Summary

The Advisor Browser has been implemented in Turbo Pascal but could easily be rewritten using other programming languages. To see how the interface is situated with the Advisor software, see figure 1 - the Advisor Overview. The help developer will have created the help database using the Advisor Designer. The application programmer links their programme with the Browser and during programme execution, the Browser accesses the help database. Calls to Browser, from within the application programme, must be accompanied by the name of the help file created by the Designer and the help item requested. The application developer may choose to access the main help menu each time help is requested, or to specify which help item to access; providing context sensitive help. Browser's built in modularity simplifies the task of attaching it to the application program. The details on the interface hooks will be described in this chapter and in chapter 2.

There is one all important design goal of the interface; that it requires a minimum of effort to link the Advisor Browser to an application program. The Browser reduces application development time by providing a ready built help facility to plug into the application. The quicker the interface can be set up, the faster the programmer can prepare the application for use. If the programmer has a central dispatch to pick up keyboard entries and evaluate them, a key can be assigned to the help function and a call to the Browser system made as a consequence of the key being pressed. The programmer can choose which help screen to access depending on the user's location in the application programme or more simply access main help with any call to help and let the user choose their own destination via the Browser navigation facilities.

The sophistication of help systems (Streitz, 1988) today and the demands of the end users, warrants the inclusion of good help systems with each application. If a generic help system can be reused throughout several applications, it will be cost effective for the application developer. As new concepts in human-computer interface design and in help system technology are discovered, they can be implemented in the generic system and made available to a variety of programmes.

1.2 Scenario - Application Developer

The application programmer sets up the links between the program and the Browser help interface. The application software has calls to Browser using the help file name (file with ADV extension created by the Advisor Designer) and the help item requested. The programmer has the choice of having any help request access the help main menu or of providing context sensitive help by selecting a particular help screen. The specific help screen can be identified using the sequence ID of the help outline item. The help developer should provide the application programmer with a report showing the outline items and their respective sequence IDs.

The programmer identifies a key to define the help function. The action the programme takes when the help key has been pressed, depends on the choice of the programmer. If they are tracking the location of the user in the programme, context sensitive help can be provided by requesting that the Browser display the help screen tied to a particular sequence number. The application programme will selectively dispatch calls for different help screens depending on the environment from which the help is requested. See Grimm et al (1988) for a discussion on the type and scope of help requirements for context sensitive help. The Advisor interface must be easy to use by the software developer. The system whereby one call is made to the Browser unit could not be any easier.

An example of how one program connects to the Browser is the Advisor Designer application which has a dispatcher to handle all input. Therefore it is easy to trap F1 and call on the Browser module to display help for a particular help file called ADVHELP.ADV. The use of context sensitive help can be shown since the Advisor Designer has a flag to indicate whether the user is in outline or detail mode. This flag can be checked and a different help screen accessed depending on the contents of the flag.

The following steps would supply the interface requirements:

Step 1 : Initialize global variable to control the creation of linked lists upon multiple entries to help.

```
BrHelpFlag := 0;
```

Note: BrHelpFlag is defined in BrIntVar which is accessed by the program in the next step.

Step 2 : Add the following clause to pick up the Browser and utility units.

```
uses Browse, BrIntVar, Qwik, Goof, Strs, Wndw

Browse     : Main Browser unit
BrIntVar   : Global variables
Qwik       : Utility routines for window manipulation
Goof
Strs
Wndw
```

Step 3 : Add code to the application program to call the Browser when a certain key is pressed. eg.

```
F1 : Browse('HelpFilename.ADV', HelpLevel)

HelpFilename.ADV  : string
     : Name of the help file

HelpLevel : integer
     : Level of help (sequence ID) requested
      (context sensitive help). To get to the main menu
      use 0.
```

Step 4 : If application demands the saving of any specific nonglobal variables, do it.

Step 5 : Copy all of the following files from the distribution disk for compilation.

```
Browser.tpu     : Main Browser unit
Keymouse.tpu    : Browser/Designer keyboard/mouse unit
Br*.tpu         : Browser utilities
Goof.tpu        : Window management utility
Qwik.tpu
Strs.tpu
Wndw.tpu
Wutil.tpu
```

Step 6 : Prior to running the program, make sure that the following files are copied from the distribution disk.

BrSet.Col	: Colour setup files
BrSet.Bw	
Browser.err	: Error message file
Helpfilename.ADV	: Application main help file
Helpfilename.NDX	: Application index/keyword file

Step 7 : Run BrSetup to specify monitor type and colour customize (see Appendix E).

1.3 Browser Interface Notes

The application programmer has the option to colour customize the help program to match their application using the programme BrSetup. Also, the application end user can be given access to the same program to make their own choices. If the application programme already has a colour customization package in place, the programmer may wish to add code to update the Browser colour files (BrSet.Col (colour monitors) and BrSet.Bw (monochrome monitors)). Appendix E gives details on the file structure and contents.

It also possible to set up a hotkey option which would make the Browser available to existing software packages. Browser can be started up as a memory resident program, and accessed from the application program using a hotkey. However certain disadvantages to this system make it less than optimum in practical use. The loss of context sensitive help is a big drawback. Also problems such as conflicting interrupt selection and cleanup on application termination (accounting for other memory resident programs) are cause for concern.

Part V. Summary and Conclusions

Chapter One: Summary and Conclusions

The Advisor is a practical tool for help development and use. Thought has gone into human-computer interface design issues and help facilities which are designed to teach the system at the same time as providing assistance. It is hoped that the use of these principles will facilitate the use of a help facility and enhance learning of the attached application software. To prove that this is the case, extensive tests of the software in use, would have to be performed. Since researchers have not yet agreed upon the best way to design a human-computer interface, there was no magic formula for creating the interface. However, some basic principles do appear frequently throughout the research and were implemented in the Advisor software. Colour customization programmes have been provided to allow the users to give some of their own preferences and structure to the application. Future considerations deal with the need for more user customization of the interface.

The Advisor is made up of two parts: the Advisor Designer (used by help developers to create help screens) and the Browser (used by application programmers to interface with their applications and end users to view help). The Designer is an editor and outline facility which is used to organize an outline of help topics and create the detail screens associated with each topic. The Browser is used to view the help screens using various navigation techniques: hierarchical order; indexes, keyword selection and a tree browser. The interface between the Browser and the application program has been kept simple, to encourage its use with a variety of software packages.

Many design knowledge and dialogue issues were discussed at the design stages. Although, the best approach for the end user would have been to implement a variety of strategies, practical considerations and real life constraints (memory and disk size) limit the possible options. The main focus of the Advisor has been how to implement a better help interface. Through the use of nodes (help screens), hierarchical links (outline hierarchy) and nonhierarchical links (indexes and keywords), a structure has been created to facilitate the help developer in screen development and ultimately the end user in accessing help.

Chapter Two: Future Directions

There are many areas of interest to be considered in further exploration of the work begun in this thesis. Some of the topics include:

1. Use Advisor to test hypotheses on valid parameters for advice-giving systems.

2. The addition of a user modifiable graphics interface.

3. Updates to user interface
   - user customization of function keys
   - user modifiable screen layout

4. New editor features
   - allow end help user to create help screens. eg. to use as personal notepad for quick retrieval during application programme use.
   - undo facility

5. New hypertext features
   - user-designated attribute/value pairs can be associated with nodes or links
   - paths - links may be strung together in a single persistent object.

6. Stubs added for the inclusion of a glossary feature

7. Procedures in place to include mouse operations

8. Further development of more varied advisory techniques

9. Make pedagogical principles less a choice of the help developer and more a part of the programme structure.

10. Add use of artificially intelligent means for assessing the end user. Streitz includes help facilities as one of several classes of intelligent tutoring systems (1988). The capabilities of assessing the needs of the users and providing active or adaptive help are the next logical steps in the enhancement of the Advisor Browser. Since very few intelligent help systems exist, this would be breaking new ground. Many of the design issues for developing such systems have not been discussed. Researchers find it difficult to assess design issues with so few prototypes in existence (Carroll and Aaronson, 1988).

Appendices

Appendix 1: Advisor Designer User's Manual

1.1 Introduction

1.1.1 Overview

This section gives an overview of the contents of the manual with suggestions on how to get the most out of the manual. Topics covered in this section include:

- how to use the manual
- how to use the Explain facility
- how to handle error messages

1.1.2 How to use the manual

To understand the full capabilities of the Advisor, read this manual carefully. Hints as to possible presentation strategies, following pedagogically sound principles have been included throughout each section.

Getting started

Getting started describes the process of preparing help screens for a given application. It describes how to start the Advisor, what standard functions are available and necessary, how to obtain help through the explain facility, how to save a file throughout a working session, and how to exit from the system.

Outline creation

More detailed information on the use of the Designer feature is described in this section. Included is a description of how to create and access various help levels. General cursor movement is described and a definition of function key assignments is given. Procedures to use special outline features of collapsing and expanding the outline are also given. Finally directions on how to access the detail screen(s) associated with each outline item are provided.

Detail definition

The detail screens contain the help interface to be viewed and used by the end user. It is here that the help developer places the content of each help screen. Knowledge of cursor movement and function key assignment is given. Also, instructions on how to return to the outline screen are provided.

Find and Replace

Both find and replace facilities are provided. Each is described, along with options available during their use.

Index and Keyword

Instructions on how to create and delete indexes and keywords are given. Also, explanations on why certain words should be indexed or certain screens linked are discussed.

Report

The report facility issues a listing of the outline items. Certain users may prefer to use the sequence ids shown on this listing to make keyword link assignments.

Customization

Colour customization is provided for the help development screens. A separate program can be used by the help developer, application programmer or end user to customize the colour of the Browser screens.

1.1.3 How to use the Explain facility

On-line help is available by using the Explain facility, accessed by pressing F1 at any time during the running of the Advisor Designer. All available function keys are listed on the last two lines of the screen.

1.1.4 How handle error messages

All messages are displayed on the last line of the screen, overwriting the bottom line of the menu for a short time. Error messages explain what the error is and wait for the user to press ESCape before continuing. If the solution is not clear, check the appendix for details of error messages and follow-up action.

1.2 Getting started

This section describes how to get access to the Advisor Designer. and how to use the system. Most importantly, what are the basic functions of the Designer, how to obtain help through the explain facility, how to save files and how to get out of the system.

The Advisor Designer has been designed to encourage the use of pedagogically sound principles of learning. Just as a teacher defines objectives for a lesson, the help developer should sit down early in the design process and define the objectives of the help facility. Some design considerations include: screen sequencing, help strategies, users conceptual models and learning styles.

1.2.1 Accessing the Advisor

User Action	**System Response**
1. Type 'ADVISE [filespec]'	Advisor Designer is displayed.

Conditional responses:

1a. If no file specification (filespec) was entered, the following display will be seen.

User Action	**System Response**
	Open File:
Enter filename of file to be retrieved or carriage return to exit Advisor.	File validation will proceed.

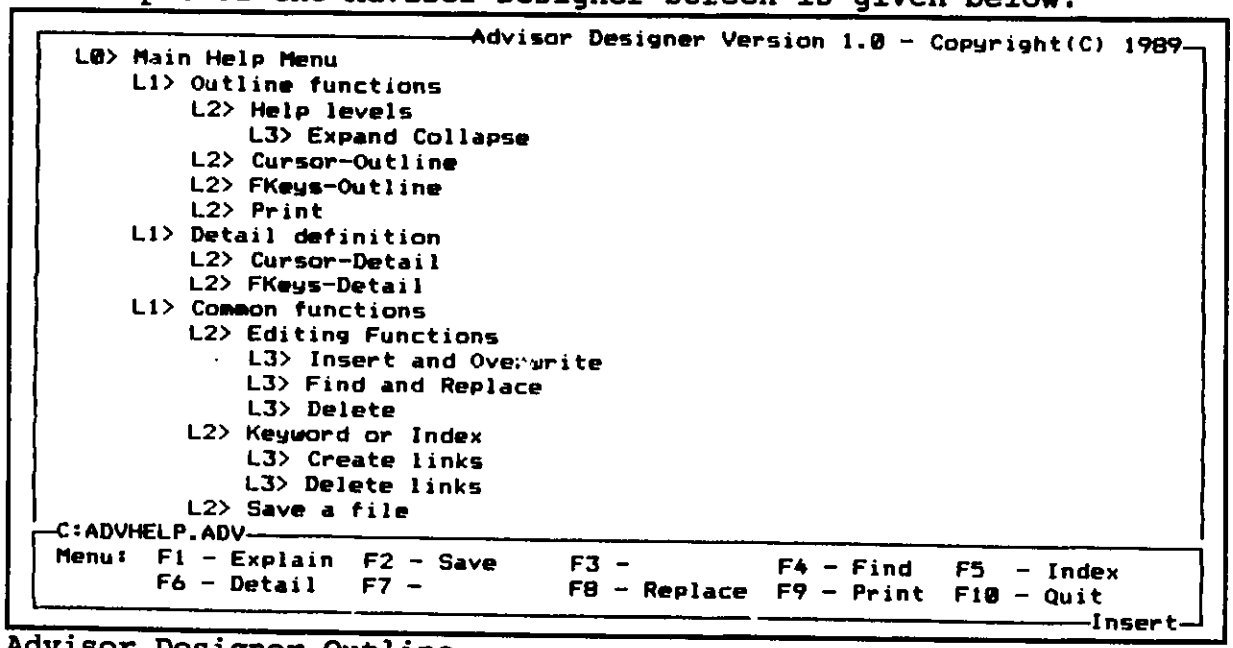
1b. If file specification was entered, validation of the name will commence.

2. After the filename has been entered in one of the above named methods, the following validation will occur.

User Action	System Response
If the file is new: If the answer is yes (Y), a file will be created and a blank screen displayed. If the answer is no (N), re-enter valid filename	File is new. Continue (Y/N)?
If the file exists:	Reading the file. Please Wait. File is displayed.

Errors	Corrections
Invalid filename	Retype file name. Check DOS manual for valid filename characters.
Path not found	Enter valid path and filename
Drive not ready	Enter valid drive ID
Customization file does not exist	Copy customization file (ADVSET.INI) to current directory
Error Message Not Found	Error file may not be accessible

An example of the Advisor Designer screen is given below.



Advisor Designer Outline

Note the following items:

1. Advisor logo in the upper right hand corner.

2. Menu - last two lines

3. Outline portion - remainder of screen

4. Message area - last line of screen

5. Filename - above menu

6. Insert status - bottom right hand corner (blank - overwrite mode) The insert key is used to toggle between insert and overwrite modes.

1.2.2 Overview of functions

The Advisor Designer is made up of a simple editor/outline facility. It is used to create or update a help database of help screens. It provides text entry; deleting characters or lines of text; cursor control keys to move to various outline levels and throughout detail screens; insert/overwrite modes; find and replace functions; file saving functions; and report printing. Typical outline features of collapsing and expanding heading levels are provided. Index words can be defined or deleted. These words will be added to an index list, to be used for quick reference by the end user. Keywords can also be defined or removed. These words are highlighted throughout the detail screen. When the end user accesses them, a link command to an associated screen is invoked. The help developer assigns indexes, keywords and their links.

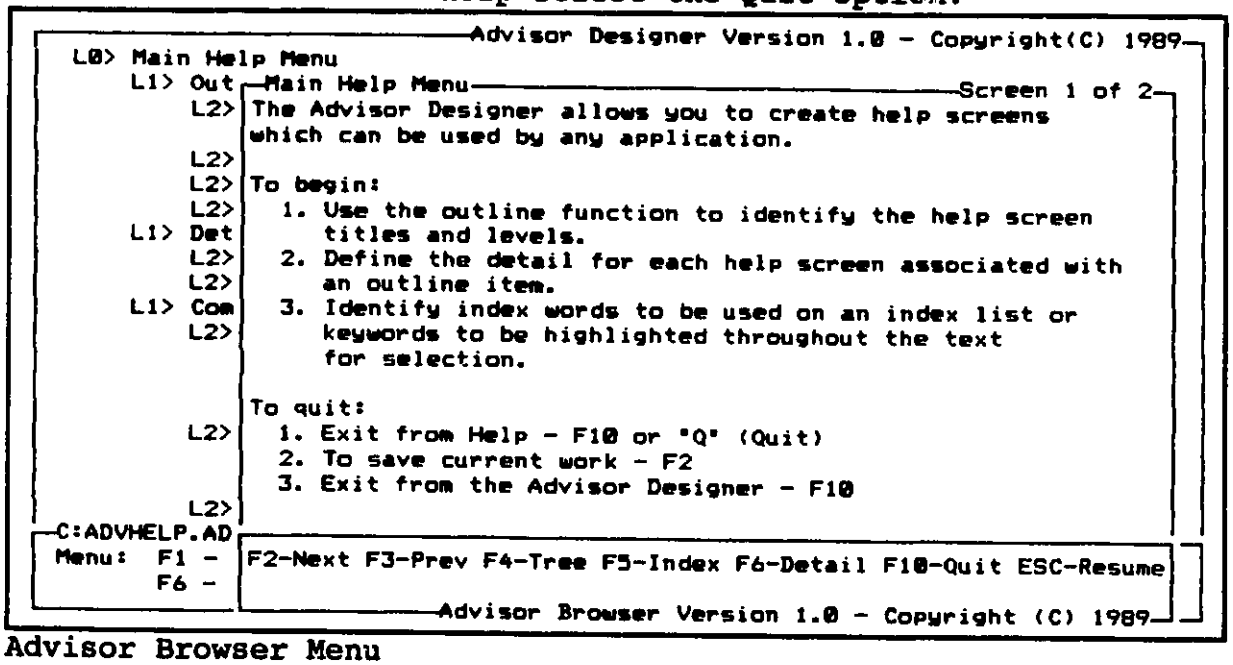
Since the Advisor Designer was used to create its own help screens, the functionality of the Advisor Browser facility will be seen if help (Explain) is accessed. The Browser allows the help user to view help by following links; selection from an index list; selection of highlighted keywords from the help screens; and viewing a tree browser (tree structure with highlighted path showing location). The user can follow hierarchical links by choosing the next and previous options. The index menu option displays an index list of help topics. Selection of highlighted keywords accesses links defined by the help developer. The browser tree gives the user the big picture of where they in relation to the overall help structure.

1.2.3 How to use the explain facility

The menu on the last two lines of each screen reminds the user of available functions and how to access them (eg. function key, first letter of menu item or cursor movement through the menu).

User Action	System Response
Press F1	Explain facility will describe the basics on how to get started, how to get more help, and how to quit.
Errors	Corrections
Customization file not found	Exit from help and make sure that the Customization file BRSET.INI has been copied from the distribution disk.

A help menu is provided to allow selection of various searches. Move the cursor right or left to choose a menu item and then press ENTER to select it OR type the first letter of a menu option. Function key commands are also provided. See below for details. To exit from help select the Quit option.



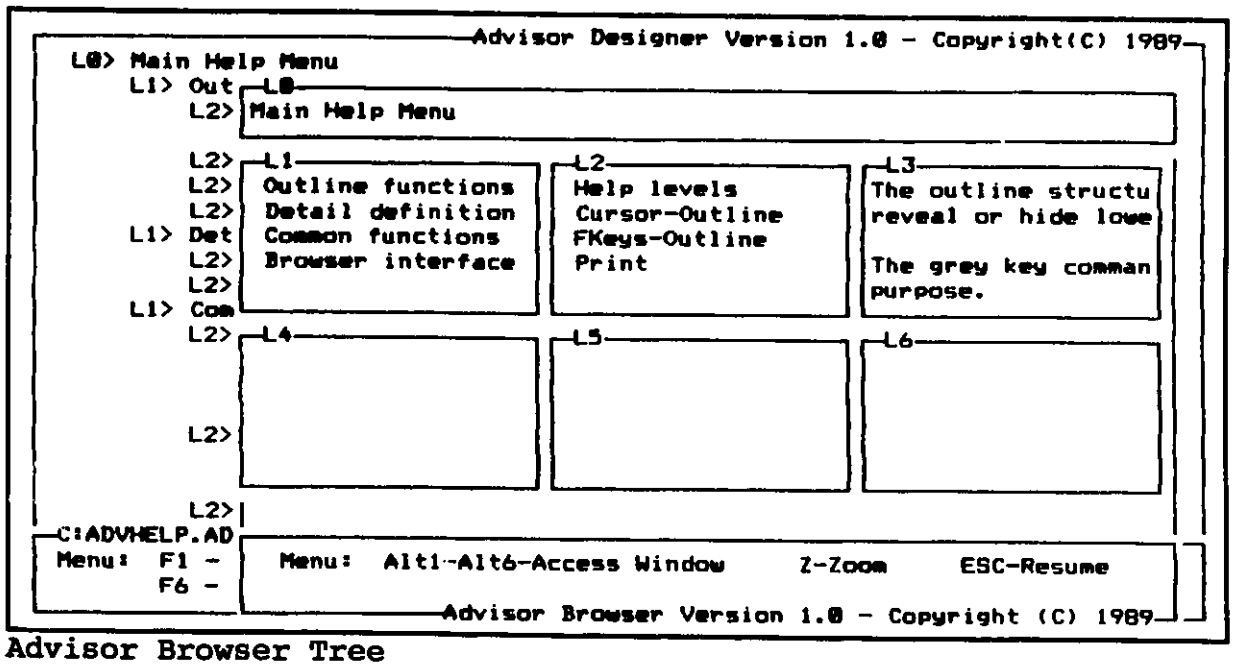
Advisor Browser Menu

The following actions are taken on menu selection.

1. **F2-Next** - Display next screen in help outline order.

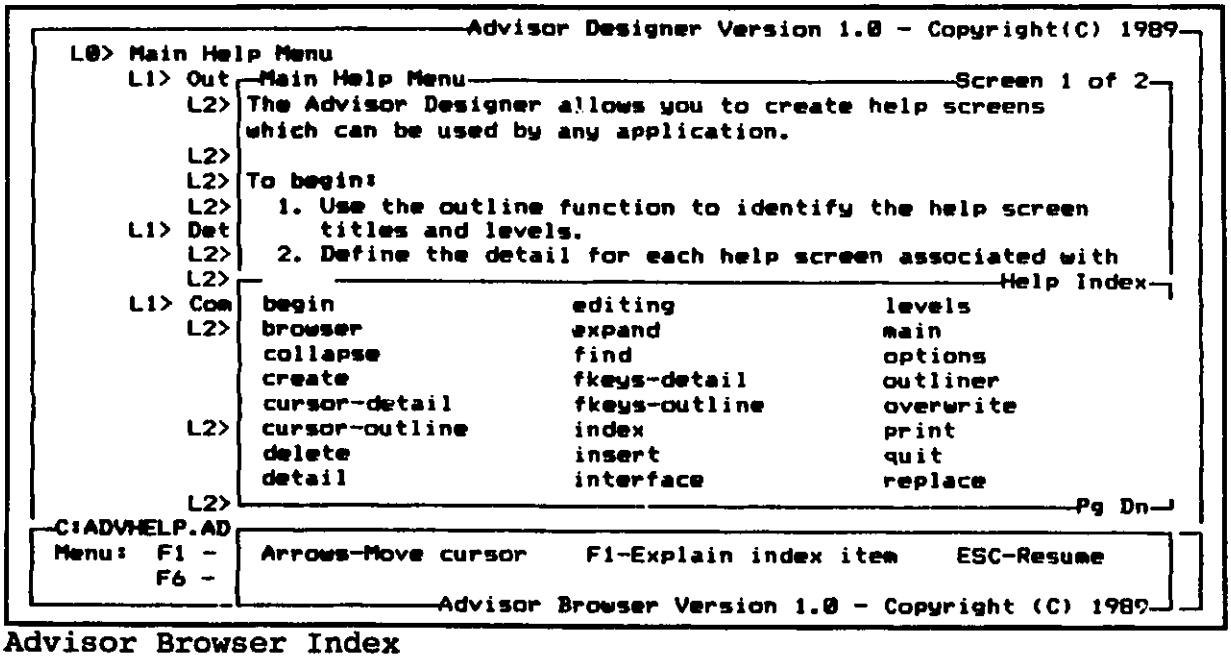
2. **F3-Previous** - Display previous screen in outline order.

3. **F4-Tree** - Shows current location; highlighting the path through various help levels. Upper levels show help titles with the current path highlighted. The current level's detail screen is displayed. Each window is labelled in the top left hand corner (eg. L0-Main help level, L1-L6 - levels 1 to 6). To get a larger picture of any of the levels: access the window associated with the window and use the zoom option to enlarge it. For example use ALT-4 (Alternate key and number 4) to access the window at level 4 (the window border will show double lines to indicate selection) and then press Z to zoom in on the information in that window. Press zoom again to return window to normal size. To overlay another window on top of the large window use Alt1 to Alt6. To exit from this display press the ESCape key.



Advisor Browser Tree

4. **F5-Index** - The Index option displays an index list, covering half of the detail screen. A selection bar highlights the current index option. Move the bar, using the arrow keys, to the desired item and press ENTER to select it; OR press ESCape to exit from the index selection screen. If the index list is longer than the allotted space on the screen, there will be a PG DN indicator in the bottom right hand corner of the index screen, showing that if the PAGE DOWN key is pressed the second page of indexes will be displayed. Once on the second page the PG UP indicator will appear in the upper left hand corner of the index screen to show that a previous page exists. Use the PAGE UP key to return to the first page.



Advisor Browser Index

Errors	Corrections
Index file does not exist	Index/keyword file (ADVHELP.NDX) may not have been copied from the distribution disk.

5. **F6-Detail** - Switch the cursor to the Detail window. From the detail window, use the TAB key to move to each keyword and press F1 to follow a keyword link OR press ESCape to switch back to the main menu. Use the Resume key to backtrack along the current help path.

Errors	Corrections
Not defined as a keyword	Cursor must be on a highlighted word before keyword selection. See if index/keyword file (ADVHELP.NDX) exists.

6. **F10-Quit** - Quit help.

7. **ESC-Resume** - Return along the path now being followed (includes route selected with keyword link, index selection and next/previous paths)(see F3-Previous).

Errors	Corrections
Error Message not found	No error message or no error file. Check for file BROWSER.ERR.

1.2.4 How to save a file

The original help file will be renamed as a backup (filename.BAK) and the current file will be saved (filename.ADV). If indexes and/or keywords have been defined, an index file will be created or updated (filename.NDX)

User Action	**System Response**
F2	File will be saved.

1.2.5 How to quit

User Action	**System Response**
F10	

Conditional response

1. If file has been saved and/or no changes have been made the program will end without any messages.

2. If the file has been changed the user will be prompted as to whether the file should be saved. Answer Y (Yes) to save or N (No) to exit without saving.

1.3 Outline creation

The outline design gives the basic hierarchical structure to the help screens. Each outline item is the title for its associated help detail screens. Seven levels of help are available for help development and movement between the levels is facilitated by special keys described in the following sections. Access to outline features (explain, save, find, index, detail, replace, print and quit) is available on the function keys (see main menu).

1.3.1 Help levels

There are up to seven help levels (L0-L6) which can be used by the help developer to set up a hierarchy of help screens. Use the TAB key to create a new level (next level). Various levels can be used to satisfy the naive browsing of the novice and give vent to the extensive search of the expert. Use the ENTER key to create a new outline line at the same level. Cursor movement between the levels is described in the next section and expansion and collapse of levels is in section 1.4.

1.3.2 Cursor movement

A brief description of the cursor movement keys is as follows:

CR	Create new line same level
TAB	Create new line next level
Up Arrow	Up one line
Down Arrow	Down one line
Left Arrow	Left one character
Right Arrow	Right one character
Page Up	Previous line same level
Page Down	Next line same level
Home	First line same level
End	Last line same level
Insert	Toggle insert/overwrite modes
Delete	Delete character under cursor
Backspace	Delete character before cursor
Shift Delete	Delete outline line and associated detail

Carriage Return (CR) and TAB are used to create new lines. Text can be entered at will and deleted using the DELETE or BACKSPACE keys. To remove the outline item and associated detail screens, use the SHIFT DELETE keys (An outline line cannot be deleted if lower levels of help exist) . An attempt to edit a word which is already highlighted as an index word will generate an error message (Index must be deleted before changing). A tone will sound if there is an attempt to exceed the maximum line length. The INSERT key is used to toggle between insert and overwrite modes. All of the other keys listed above are used to move throughout the outline. Familiarity with these keys will speed up the outline design process.

1.3.3 Function keys

F1	Explain		
F2	Save		
F3	not used		
F4	Find		
F5	Create Index	Ctrl F5	Remove Index
F6	Detail Screen		
F7	not used		
F8	Replace		
F9	Print		
F10	Quit		

1.3.4 Collapse/expand

The outline structure can be expanded and collapsed to reveal or hide lower levels. Press minus (on the keypad) to collapse subheadings under an outline item. Press plus (on the keypad) to expand subheadings under an outline item. Press * (on the keypad) to expand all headings.

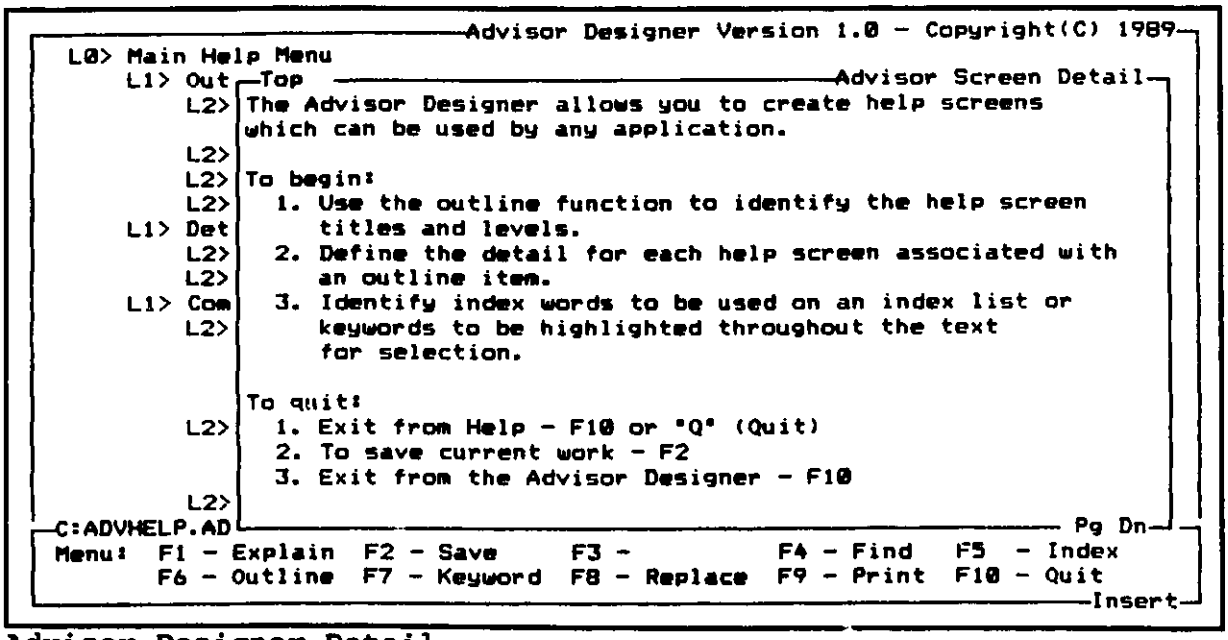
1.3.5 How to access detail screens

User Action	System Response
F6	The Detail screen for the current help outline item will be displayed.

Note: The screen size is the same as it will be during use within an application program. This allows the user to keep in touch with what they were doing before asking for help.

1.4 Detail definition

It is entirely up to the help developer to ensure that the following help screen concerns are met. The help screen detail must be accurate, consistent, complete, visually appealing (avoid solid blocks of text), adequate reading level (grade 5), non-anthropomorphic (otherwise the user feels blamed) and consistent with written documentation. A typical detail screen looks as follows.



Advisor Designer Detail

1.4.1 Cursor movement

A brief description of the cursor movement keys is as follows:

CR	Create new line
Up Arrow	Up one line
Down Arrow	Down one line
Left Arrow	Left one character
Right Arrow	Right one character
Page Up	Go to previous help screen
Page Down	Go to next help screen or create screen
Home	Go to beginning of line
End	Go to end of line
Insert	Toggle insert/overwrite modes
Delete	Delete character under cursor
Backspace	Delete character before cursor
Shift Delete	Delete detail line

Carriage Return (CR) is used to create new lines. Text can be

entered at will and deleted using the DELETE or BACKSPACE keys. To remove a detail line, use the SHIFT DELETE keys. An attempt to edit a word which is already highlighted as an index word or keyword will generate an error message (Index must be deleted before changing). The index/keyword status of a word must be removed before editing of that word is permitted. A tone will be heard if there is an attempt to write past the maximum line length. The INSERT key is used to toggle between insert and overwrite modes. All of the other keys listed above are used to move throughout the detail screens. The Page Down key has a dual purpose. If there are more detail screens associated with the outline item, the new screen will be presented when PG DN is pressed. If not, a new blank screen will be displayed; allowing the help developer to create a new help screen.

1.4.2 Function keys

The function key definition is the same as it was on the outline screen; with the exception of function key 7, which is used for keyword creation (this function is not available on the outline).

F1	Explain		
F2	Save		
F3	not used		
F4	Find		
F5	Create Index	Ctrl F5	Remove Index
F6	Outline screen		
F7	Create Keyword	Ctrl F7	Remove Keyword
F8	Replace		
F9	Print		
F10	Quit		

1.4.3 How to return to the outline

User Action	System Response
ESCape or F6	Removes detail window and returns to outline window.

1.5 Find and Replace

1.5.1 Find

User Action	System Response
F4	Find:
Type the word to be found.	Options:
Type the letter u and then ENTER if the search is to ignore upper and lower case. Otherwise press ENTER.	
	The cursor will line up after the searched word OR an error message will explain that it could not be found.

1.5.2 Replace

User Action	System Response
F8	Find:
Type the word to be found.	Options:
Type the letter u and then ENTER if the search is to ignore upper and lower case. Otherwise press ENTER.	
	Replace with:
Type word to replace found word and press ENTER.	
	If word is located and is not too long it will be replaced.

1.6 Index and Keyword

As mentioned in the introduction, heavily used help screen routes can be optimized through quick access by indexes and keywords. If pilot studies show that a particular topic is always a problem, it might be a candidate for an index item or it may require that some aspect of it be highlighted as a keyword and linked to further references. The index and keyword links provide the non-hierarchical links which should allow the user to freely explore the help system. Much attention must be given to users of the system to determine the most efficient access to be gained through links. These links can point the way to users in an effort to alleviate problems using a coaching strategy.

1.6.1 Creation

1.6.1.1 Index creation

User Action	System Response
Line up the cursor with the word to be indexed. Press F5.	The index word will be highlighted and added to the index list. It will appear on the help user's index selection screen.

1.6.1.2 Keyword creation

User Action	System Response
Line up the cursor with the word to be used as a keyword. Press F7.	A menu will be presented showing 2 options: Link Sequence ID
Move the cursor up or down and press ENTER on the chosen method of link creation.	

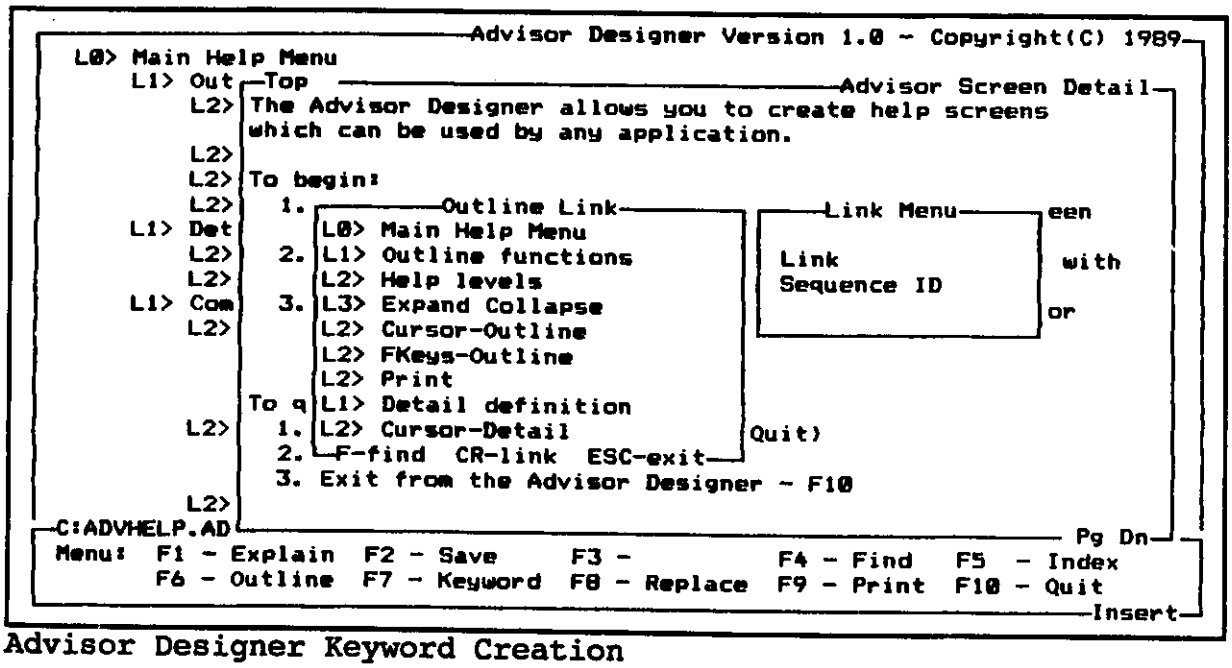
Note LINK will allow selection from a list of outline items. If the Sequence ID of the outline item to be linked to the chosen keyword is known, choose the Sequence ID option.

Conditional Response:

1. If the LINK option was chosen, a selection list of outline items will be displayed. The cursor can be moved up and down through the list until the desired item is highlighted OR the Find command can be used to search through the list (Press F). Once the outline item is highlighted, press ENTER to select it OR press ESC to cancel request for link assignment.

2. If the sequence number of the outline item to be linked is known, it can be entered after choosing the Sequence ID menu option. (Note: F9 can be used to print the outline; including all sequence numbers)

After the link has been established with either of these methods, the keyword will be highlighted. Following, is an example of the screen content if the link option was chosen.



Advisor Designer Keyword Creation

1.6.2 Deletion

1.6.2.1 Index deletion

User Action	System Response
Line up the cursor with the index word to be deleted. Press CTRL F5.	The index word will return to normal highlighting and the word will be removed from the list.

1.6.2.2 Keyword deletion

User Action	System Response
Line up the cursor with the keyword to be deleted. Press CTRL F7.	The keyword will return to normal highlighting and the link with the outline item broken.

1.7 Reports

User Action	System Response
F9	Outline listing will be printed.
Errors	Correction
Device Write Fault	Turn on printer and try again.

1.8 Customization

The colour scheme can be customized through the use of two programs. ADVSETUP is used to customize the screens used by the help developer for the Advisor Designer. BRSETUP is used to customize the help screens seen by the end user who is using the Advisor Browser.

Appendix 2: Advisor Browser User's Manual

The application end user's help manual must be written in a style which is compatible with the application user's manual. Help sample screens should relate to the application. Therefore, the writing of the manual is left up to the application programmer. The help procedures in the help developer's manual (Appendix 1) are available for reference by the application programmer.

Appendix A. Programmer's Specifications - Designer

A.1 Program Specifications

A.1.1 Files

Main files:

Advhelp.Adv - Advisor Designer help screens
Advhelp.Ndx - Advisor Designer indexes and keywords

Helpfilename.Adv - Help developers help screens
Helpfilename.Ndx - Help developers indexes and keywords

Miscellaneous files:

Outline.Err - Contains all error messages
Advset.Ini - Initialization file for screen colours
Advset.Col - Default colours for colour monitor
Advset.Bw - Defaults for monochrome monitor

A.1.2 Constants

```
ADVCOPYRIGHT : string[48] =
               'Advisor Designer Version 1.0 - Copyright(C) 1989';

CTRLI                 = #9;          Index
CTRLK                 = #11;         Keyword
DETLINELENGTH         = 90;          Length of detail text line
DETSCREENCOL : byte   = 64;          Detail screen-# of columns
DETSCREENLINELENGTH   = 60;          Length of detail text line
DETSCREENROW : byte   = 17;          Detail screen-number of rows
DETSCREENX   : byte   = 15;          Detail screen starting column
DETSCREENY   : byte   = 3;           Detail screen starting row
DETWINROW    : byte   = 19;          Detail window - number of rows
ERRORFILENAME = 'OUTLINE.ERR';       File containing error messages
                                     Valid file name characters
FILECHAR : Set of
               char=['A'..'Z','a'..'z','0'..'9','-','$','%','.',
               '@','{','}','~','!','#','(',')','&','.',':','\'];
                                     Valid alphanumeric keys
KEYALPHA : Set of char = ['A'..'Z','a'..'z','0'..'9','-'];
                                     Valid integers
KEYINT   : Set of char = ['0'..'9'];
                                     Index/Keyword characters
KEYCTRL  : Set of char = [CTRLI, CTRLK];
LINKMENUCOL  : byte = 20;            Link menu window-# of cols
LINKMENUROW  : byte = 6;             Link menu window-# of rows
LINKMENUX    : byte = 50;            Link menu window column
LINKMENUY    : byte = 8;             Link menu window row
LINKOUTCOL   : byte = 30;            Link outline window-columns
```

```
LINKOUTROW    : byte = 11;          Link outline window-rows
LINKOUTX      : byte = 20;          Link outline starting column
LINKOUTY      : byte = 8;           Link outline starting row
MAXLEVEL      : byte = 6;           Maximum number outline levels
                                    Menu display
MENU1ST1  : string[7]  = 'Menu:  ';
MENU1ST10 : string[2]  = 'F5';
MENU1ST11 : string[9]  = '  - Index';
MENU1ST2  : string[2]  = 'F1';
MENU1ST3  : string[12] = ' - Explain  ';
MENU1ST4  : string[2]  = 'F2';
MENU1ST5  : string[12] = ' - Save     ';
MENU1ST6  : string[2]  = 'F3';
MENU1ST7  : string[12] = ' -          ';
MENU1ST8  : string[2]  = 'F4';
MENU1ST9  : string[10] = ' - Find   ';
MENU2ST1  : string[7]  = '       ';
MENU2ST10 : string[3]  = 'F10';
MENU2ST11 : string[8]  = ' - Quit ';
MENU2ST2  : string[2]  = 'F6';
MENU2ST3A : string[12] = ' - Detail   ';
MENU2ST3B : string[12] = ' - Outline  ';
MENU2ST4  : string[2]  = 'F7';
MENU2ST5A : string[12] = ' -          ';
MENU2ST5B : string[12] = ' - Keyword  ';
MENU2ST6  : string[2]  = 'F8';
MENU2ST7  : string[12] = ' - Replace  ';
MENU2ST8  : string[2]  = 'F9';
MENU2ST9  : string[10] = ' - Print  ';
MENUCOL       : byte = 80;          Menu window-number of columns
MENUROW       : byte = 4;           Menu window-number of rows
MENUX         : byte = 1;           Menu window starting column
MENUY         : byte = 21;          Menu window starting row
MSGCOL        : byte = 50;          Message window-number of cols
MSGROW        : byte = 3;           Message window-number of rows
MSGX          : byte = 20;          Message window column
MSGY          : byte = 10;          Message window row
OUTLINELENGTH       = 50;         Length of outline text line
OUTSCREENCOL  : byte = 80;          Outline screen-# of columns
OUTSCREENLINELENGTH = 35;         Length of outline text line
OUTSCREENROW  : byte = 19;          Outline text-number of rows
OUTSCREENX    : byte = 1;           Outline screen column
OUTSCREENY    : byte = 1;           Outline screen starting row
OUTWINROW     : byte = 21;          Outline window - # of rows
```

A.1.3 Data Types

```
OutTextLine = string [OUTLINELENGTH]; Outline text line
DetTextLine = string [DETLINELENGTH]; Detail  text line
Str80       = string [80];  Multi purpose sting of 80 char
FnStr       = string [65];  File name string
```

```
IndexStr     = string [20];  Index item string
FindStr      = string [30];  Find word string
OptStr       = string [10];  Option word string
Poutlines    = ^Outlines;    Pointer to outline screen line
Pdetlines    = ^Detlines;    Pointer to detail screen line
PindexList   = ^IndexList;   Pointer to index list

Outlines = record            Text line - outline screen
           Fwdlink  : Poutlines; Forward link
           Backlink : Poutlines; Backward link
           NextSamelink : Poutlines; Forward to same level
           PrevSamelink : Poutlines; Backward to same level
           Detlink  : Pdetlines; Ptr to detail text lines
           Key      : byte;      Help screen key
           Level    : byte;      Outline level
           Bufflen  : byte;      Length text line + ctrl char
           Linelen  : byte;      Length of text line
           Txt      : OutTextline;Text line
           DscreenNum: byte;     Number of detail screens
           DlineLen : byte       Number of detail screen lines
           end;

Detlines = record                Each text line - detail screen
           Fwdlink  : Pdetlines; Forward link
           Backlink : Pdetlines; Backward link
           ScreenNum: byte;      Detail screen number
           Bufflen  : byte;      Length text line + ctrl char
           Linelen  : byte;      Length of text line
           Txt      : DetTextlineText line
           end;

IndexList = record               Index and keyword list
            Fwdlink  : Pindexlist; Forward link
            Backlink : Pindexlist; Backward link
            KeyType  : char;     Type:index,...
            SrcKey   : byte;     Keyword originates at this key
            PtrKey   : byte;     Keyword links to this key
            Txt      : IndexStr; Index text line
            DetFlag  : boolean   True if detail screen index
            end;

IndexRec = record                Index/Keyword file record
           KeyType : char;       Type:index,...
           Srckey  : byte;       Keyword originates at this key
           Ptrkey  : byte;       Keyword links to this key
           Txt     : IndexStr;   Index text line
           DetFlag : boolean     True if detail screen index
           end;
IndexFile = file of IndexRec;    Index/Keyword file
```

A.1.4 Variables

```
ChangeFlag       : boolean;    True if changes have been made
ColourBWFlag     : char;       Colour or monochrome flag
ColourIndexBG    : byte;       Index - Background
ColourIndexFG    : byte;       Index - Text foreground
ColourKeywordBG  : byte;       Keyword - Background
ColourKeywordFG  : byte;       Keyword - Text foreground
ColourW1BG       : byte;       Window 1 - Background
ColourW1Brdr     : byte;       Window 1 - Border
ColourW1ErrorBG  : byte;       Window 1 - Error background
ColourW1ErrorFG  : byte;       Window 1 - Error foreground
ColourW1FG       : byte;       Window 1 - Menu-Text foreground
ColourW1FileBG   : byte;       Window 1 - File background
ColourW1FileFG   : byte;       Window 1 - File foreground
ColourW1LetterBG: byte;        Window 1 - First letter background
ColourW1LetterFG: byte;        Window 1 - First letter foreground
ColourW1StatusBG: byte;        Window 1 - Status background
ColourW1StatusFG: byte;        Window 1 - Status foreground
ColourW2BG       : byte;       Window 2 - Background
ColourW2Brdr     : byte;       Window 2 - Border
ColourW2FG       : byte;       Window 2 - Outline-Text foreground
ColourW2Level0BG: byte;        Window 2 - Level 0 background
ColourW2Level0FG: byte;        Window 2 - Level 0 foreground
ColourW2Level1BG: byte;        Window 2 - Level 1 background
ColourW2Level1FG: byte;        Window 2 - Level 1 foreground
ColourW2Level2BG: byte;        Window 2 - Level 2 background
ColourW2Level2FG: byte;        Window 2 - Level 2 foreground
ColourW2Level3BG: byte;        Window 2 - Level 3 background
ColourW2Level3FG: byte;        Window 2 - Level 3 foreground
ColourW2Level4BG: byte;        Window 2 - Level 4 background
ColourW2Level4FG: byte;        Window 2 - Level 4 foreground
ColourW2Level5BG: byte;        Window 2 - Level 5 background
ColourW2Level5FG: byte;        Window 2 - Level 5 foreground
ColourW2Level6BG: byte;        Window 2 - Level 6 background
ColourW2Level6FG: byte;        Window 2 - Level 6 foreground
ColourW2SelectBG: byte;        Select bar
ColourW2SelectFG: byte;        Select bar
ColourW2TitleBG: byte;         Window 2 - Title background
ColourW2TitleFG: byte;         Window 2 - Title foreground
ColourW3BG       : byte;       Window 3 - Background
ColourW3Brdr     : byte;       Window 3 - Border
ColourW3FG       : byte;       Window 3 - Detail-Text foreground
ColourW3PgBG     : byte;       Window 3 - PgUp/Down background
ColourW3PgFG     : byte;       Window 3 - PgUp/Down Msg text
ColourW3TitleBG: byte;         Window 3 - Title background
ColourW3TitleFG: byte;         Window 3 - Title foreground
ColourW4BG       : byte;       Window 4 - Background
ColourW4Brdr     : byte;       Window 4 - Border
ColourW4FG       : byte;       Window 4 - Messages-foreground
ColourW4TitleBG: byte;         Window 4 - Title background
ColourW4TitleFG: byte;         Window 4 - Title foreground
```

```
CurrDir          : FnStr;          Current path
CurrLevel        : byte;           Current level
DetColumn        : byte;           Detail screen column indicator
DetCurline       : pdetlines;      Ptr. to current detail line
DetFindCol       : byte;           Last detail column found
DetFindPtr       : pdetlines;      Ptr. to last detail line found
DetPageDnFlag    : boolean;        True if detail page down used
ErrorFile        : FnStr;          Error file path and name
ExpandLevel      : byte;           Outline level for expand/collapse
Filename         : FnStr;          Outline file name
FindCase         : boolean;        Flag for find option
FindWord         : FindStr;        Keyword in find search
Finish           : boolean;        Main scheduler loop terminator
ForceUpdateFlag: boolean;          True if forcing screen update
IncrFlag         : boolean;        True if level has been raised
IndCurline       : pindexlist;     Ptr. to current index line
IndOutList       : pindexlist;     Ptr. to sorted index list
IndTopline       : pindexlist;     Ptr. to top index line
InsertFlag       : boolean;        True if insert mode is on
KeyWord          : IndexStr;       Index keyword
OutColumn        : byte;           Outline screen column indicator
OutCurline       : poutlines;      Ptr. to current outline line
OutFindCol       : byte;           Last outline column found
OutFindPtr       : poutlines;      Ptr. to last outline line found
Outline          : boolean;        True outline, false screen mode
Outlineid        : integer;        Key for each line of outline
OutRow           : byte;           Outline screen row indicator
OutTopline       : poutlines;      Ptr. to first outline line
OutTopWindow     : poutlines;Ptr. to outline top of screen
Parameters       : boolean;        True if program parameters
ReplaceWord      : FindStr;        Keyword for replacing text
SaveWindowName :WindowNames;       Window name for reaccess
```

A.1.5 Procedures & Functions

See Part II Chapter two for a description of each procedure and function.

Procedure	Unit
AssignLink	Index
BeginWordCol	Index
BottomDetail	CursrCmd
BottomFile	CursrCmd
CloseFile	FileIO
CommandProcessor	Dispatch
DeleteChar	OutTools
DeleteDetline	OutTools
DeleteDetlineStruc	OutTools

Procedure	Unit
DeleteDetlink	OutTools
DeleteFile	FileIO
DeleteIndex	Index
DeleteIndexStruc	Index
DeleteOutline	OutTools
DeleteOutlineStruc	OutTools
DelIndex	Index
DetailScreen	Screen
DetBeginningLine	CursrCmd
DetEndLine	CursrCmd
DetPageDown	CursrCmd
DetPageUp	CursrCmd
DetScreenFull	Screen
DetUpdateScreen	Screen
DetWriteLine	Screen
DisplayError	Error
DisplayLinkMenu	Index
DisplayLinkOutline	Index
DisplayPgUpDown	Screen
DownLine	CursrCmd
ExpandContract	OutTools
Find	Find
FindIndex	Index
FlushIndex	Index
Genid	OutTools
GetFirstSameLevel	OutTools
GetLastChild	OutTools
GetLastSameLevel	OutTools
GetMessage	Error
GetNextLink	OutTools
GetNextSameLevel	OutTools
GetNextScreen	OutTools
GetParent	OutTools
GetPrevLink	OutTools
GetPrevSameLevel	OutTools
GetPrevScreen	OutTools
GetReferenceWord	Index
GetSequenceID	Index
GetWordLength	Find
HeapFunc	OutTools
IncrLevel	OutTools
IndexProcessor	Index
InitDetail	OutTools
InitFind	Find
Initialize	Init
InitLinkMenu	Index
InitReplace	Find
InitVar	Init

Procedure	Unit
InputClassifier	Schedule
InsertDetLine;	OutTools
InsertIndex	Index
InsertMode	OutTools
InsertOutLine	OutTools
KeyMousePressed	KeyMouse
KeyWordProcessor	Index
LeftChar	CursrCmd
LinkItem	Index
LowerCase	Index
MatchLetter	Find
MatchWord	Find
Mouse	KeyMouse
NewLine	OutTools
OGetWordLength	Index
OInitFind	Index
OMatchLetter	Index
OMatchWord	Index
OnCurrentScreen	Screen
OOutSearch	Index
OpenFile	FileIO
OutEnd	CursrCmd
OutExit	OExit
OutHome	CursrCmd
Outliner	Schedule
OutPageDown	CursrCmd
OutPageUp	CursrCmd
OutPrint	Print
OutSearch	Find
OutUpdateScreen	Screen
OutWriteLine	Screen
OutWriteLink	Index
ProcessFind	Find
ProcessReplace	Find
ReadColour	FileIO
ReadFile	FileIO
ReadKeyMouse	KeyMouse
RenameFile	FileIO
Replace	Find
ReturnOutline	Screen
RightChar	CursrCmd
SaveFile	FileIO
Scheduler	Schedule
ShiftRightChar	Index
SpecialKeys	KeyMouse
TextProcessor	Input
TopDetail	CursrCmd
TopFile	CursrCmd
UpLine	CursrCmd
ValidateKey	Index

A.2 Error Handling

A.2.1 Error Messages

```
***  DOS errors - Pascal Reference Manual V 5

002  File not found
003  Path not found
004  Check number of open files
005  Check file access
006  Check file handle

008  Not enough memory

010  Check environment
011  Check format
012  Check file access code

015  Check drive number

017  Cannot rename across drives
018  No more files

020  Line too long
021  Check file name

***  I/O errors - IORESULT - Pascal Reference Manual V 5
100  Disk read problem
101  Disk write problem
102  File not assigned
103  File not open
104  File not open for input
105  File not open for output
106  Check numeric format

***  I/O errors - IORESULT - Advisor Designer
120  Unexpected end of file
121  Undefined file problem

130  Check page status display
131  Delete not possible - lower levels exist
132  Cannot create 2 versions of main help
133  Check menu selection
134  Customization file not found

140  Index not found
141  Index already exists
142  Keyword must be deleted before editing
143  Link not found
144  Keyword already exists
145  Index must be deleted before editing
146  Help Level does not exist
```

147 Search string not found
148 Replace string too long
149 Keyword not found

*** Critical errors - Pascal Reference Manual V 5
150 Disk is write-protected
151 Unknown unit
152 Drive not ready
153 Unknown command
154 CRC problem
155 Check drive request structure length
156 Disk seek problem
157 Unknown media type
158 Sector not found
159 Printer out of paper

160 Device write problem
161 Device read problem
162 Hardware problem

*** Critical errors - Advisor Designer
170 Not enough memory
171 Disk is full
172 Printer cannot be accessed
173 Directory is full

*** Fatal errors - Pascal Reference Manual V 5
200 Division by zero
201 Range check problem
202 Stack overflow
203 Heap overflow
204 Check pointer operation

Appendix B. Programmer's Specifications - Browser

B.1 Program Specifications

B.1.1 Files

Main files:

HelpFilename.ADV - Help developer's help screens
HelpFilename.NDX - Help developer's index and keywords

Miscellaneous

Browser.Err - contains all Browser error messages
Brset.Ini - Browser colour initialization file
Brset.Col - Browser colours for colour screens
Brset.Bw - Browser setup for monochrome screens

B.1.2 Constants

```
  BRCOPYRIGHT : string [48] =
           'Advisor Browser Version 1.0 - Copyright (C) 1989';
  BRCTRLG                    = #7;    Glossary word
  BRCTRLI                    = #9;    Index word
  BRCTRLK                    = #11;   Keyword
  BRDETLINELENGTH            = 90;    Length of detail text line
  BRDETSCREENLINELENGTH      = 60;    Length of detail text line
  BRERRORFILENAME = 'BROWSER.ERR'; Error file with error messages
  BRINDEXCOL         : byte = 64;     Browser index -number columns
  BRINDEXITEMS       : byte = 24;     Browser index -number of items
  BRINDEXROW         : byte = 10;     Browser index -number of rows
  BRINDEXX           : byte = 15;     Browser index starting column
  BRINDEXY           : byte = 11;     Browser index starting row
  BRINDLINELENGTH            = 20;    Length of index text line
  BRINDTEXTCOL       : byte = 3;      Browser index -# text cols
  BRINDTEXTROW       : byte = 8;      Browser index -# text rows
  BRKEYALPHA   : Set of char = ['A'..'Z','a'..'z','0'..'9','-'];
  BRKEYCTRL    : Set of char = [BRCTRLI,BRCTRLK];
  BRMENUCOL    : byte = 64;           Browser menu- number columns
  BRMENUROW    : byte = 4;            Browser menu-number of rows
  BRMENUX      : byte = 15;           Browser menu starting column
  BRMENUY      : byte = 21;           Browser menu starting row
  BROUTLINELENGTH            = 50;    Length of outline text line
  BROUTSCREENLINELENGTH      = 35;    Length of outline text line
  BRPATHCOL          : byte = 21;     Browser path -number columns
  BRPATHCOL0         : byte = 64;     Browser path 0-number columns
  BRPATHCOLNUM       : byte = 32;     Browser path border line
  BRPATHLINELENGTH   : byte = 30;     Browser path column width
  BRPATHLINENUM      : byte = 8;      Browser path number of lines
  BRPATHROW          : byte = 7;      Browser path -number of rows
  BRPATHROW0         : byte = 3;      Browser path 0-number of rows
  BRPATHX0           : byte = 15;     Browser path 0 starting column
  BRPATHX1           : byte = 15;     Browser path column L1,L4
  BRPATHX2           : byte = 36;     Browser path column L2,L5
```

```
BRPATHX3          : byte = 57;     Browser path column L3,L6
BRPATHY0          : byte = 3;      Browser path 0 starting row
BRPATHY1          : byte = 6;      Browser path row - L1-L3
BRPATHY2          : byte = 13;     Browser path row - L4-L6
BRSCREENCOL : byte       = 64;     Browser detail-number columns
BRSCREENROW : byte       = 19;     Browser detail-number of rows

BRST1  : string[4]  = 'F2-N';     Main Menu
BRST10 : string[6]  = 'etail ';
BRST11 : string[5]  = 'F10-Q';
BRST12 : string[4]  = 'uit ';
BRST13 : string[5]  = 'ESC-R';
BRST14 : string[5]  = 'esume';
BRST2  : string[4]  = 'ext ';
BRST20 : string[8]  = ' Menu:  '; Path Menu
BRST21 : string[9]  = 'Alt1-Alt6';
BRST22 : string[19] = '-Access Window     ';
BRST23 : string[1]  = 'Z';
BRST24 : string[10] = '-Zoom      ';
BRST25 : string[3]  = 'ESC';
BRST26 : string[7]  = '-Resume';
BRST3  : string[4]  = 'F3-P';
BRST30 : string[7]  = ' Arrows'; Index Menu
BRST31 : string[16] = '-Move cursor    ';
BRST32 : string[2]  = 'F1';
BRST33 : string[23] = '-Explain index item    ';
BRST34 : string[3]  = 'ESC';
BRST35 : string[7]  = '-Resume';
BRST4  : string[4]  = 'rev ';
BRST40 : string[4]  = ' TAB';     Detail Menu
BRST41 : string[30] = '-next keyword      ';
BRST42 : string[2]  = 'F1';
BRST43 : string[30] = '-Explain keyword      ';
BRST44 : string[3]  = 'ESC';
BRST45 : string[9]  = '-Resume';
BRST5  : string[4]  = 'F4-T';
BRST6  : string[4]  = 'ree ';
BRST7  : string[4]  = 'F5-I';
BRST8  : string[5]  = 'ndex ';
BRST9  : string[4]  = 'F6-D';
BRSTARTX    : byte  = 15;         Browser detail starting column
BRSTARTY    : byte  = 3;          Browser detail starting row
```

B.1.3 Data Types

```
BrIndexStr     = string [20];     Index keyword string
BrStr80        = string [80];     General purpose text string
BrFnStr        = string [65];     File name text string

BrOutTextLine = string[BROUTLINELENGTH]; Text line for outline
BrDetTextLine = string[BRDETLINELENGTH]; Text line for detail
```

```
PbrOutlines     = ^BrOutlines;     Ptr. to outline
PbrDetlines     = ^BrDetlines;     Ptr. to detail
PbrPathList     = ^BrPathList;     Ptr. to stack
PbrIndexList    = ^BrIndexList;    Ptr. to index
PbrKeywordList = ^BrKeywordList; Ptr. to keyword

BrOutlines = record                    Browser outline text line
             Fwdlink  : PbrOutlines;    Forward ptr. next line
             Backlink : PbrOutlines;    Ptr. to previous line
             NextSamelink: PbrOutlines; Ptr. to next same level
             PrevSamelink: PbrOutlines; Ptr. prev same level
             BrLink   : PbrDetlines;    Ptr. to detail screen
             Key      : byte;           Unique line key
             Level    : byte;           Help level
             Bufflen  : byte;           Length of text line
             Txt      : BrOutTextLine;  Text line
             DscreenNum: byte;          # of detail screens
             DlineLen : byte;           Number of detail lines
             end;

BrDetlines = record                 Browser detail screen line
             Fwdlink  : PbrDetlines; Forward ptr. to next line
             Backlink : PbrDetlines; Ptr. to previous line
             ScreenNum: byte;        Number of screen
             Bufflen  : byte;        Length of text line
             Txt      : BrDetTextLine Text line
             end;

BrPathList = record                   Browser current path stack
             Fwdlink      : PbrPathList; Forward ptr. next line
             Mainlink     : PbrOutlines; Ptr. to main help list
             DetScreenNum : byte         Detail screen number
             end;

BrIndexList = record                  Browser index text line
              Fwdlink  : PbrIndexList; Forward ptr. next line
              Backlink : PbrIndexList; Ptr. to previous line
              Mainlink : PbrOutlines; Ptr. to main help list
              SrcKey   : byte;        Outline keyword source
              Txt      : BrIndexStr;  Text line
              end;

BrKeywordList = record                Browser keyword text line
              Fwdlink  : PbrKeywordList; Forward ptr. next line
              Backlink : PbrKeywordList; Ptr. to previous line
              Mainlink : PbrOutlines;   Ptr. to main help list
              SrcKey   : byte;          Outline keyword source
              Txt      : BrIndexStr;    Text line
              end;
```

```
BrIndexElement = record              Index element for screen array
                 Row       : byte;       Screen row
                 Col       : byte;       Screen column
                 Highlight : boolean;    Highlight flag
                 Item      : BrIndexStr; Index text
                 Mainlink  : pBrOutlines; Link to outline item
                 end;
                                     Screen array of index elements
BrIndexArray  = array [1..24] of BrIndexElement;

BrIndexRec = record                  Index/Keyword file
             KeyType: char;          Index type: index, keyword...
             Srckey : byte;          Source of index keyword
             Ptrkey : byte;          Key to which index points
             Txt    : BrIndexStr;    Text of index keyword
             DetFlag: boolean        True if detail screen keyword
             end;

BrIndexFile = file of BrIndexRec; Index/Keyword file
```

B.1.4 Variables

```
BrArrayNum          : byte;      Index array element number
BrBottomIndexScreen: PbrIndexList; Bottom of index screen
BrColourBWFlag      : byte;      Colour flag
BrColourKeywordBG   : byte;      Background
BrColourKeywordFG   : byte;      Keyword:Foreground
BrColourW1BG        : byte;      Background
BrColourW1Brdr      : byte;      Border
BrColourW1ErrorBG   : byte;      Error background
BrColourW1ErrorFG   : byte;      Error foreground
BrColourW1FG        : byte;      Window 1 Menu:Foreground
BrColourW1LetterBG  : byte;      First letter background
BrColourW1LetterFG  : byte;      First letter foreground
BrColourW1SelectBG  : byte;      Select background
BrColourW1SelectFG  : byte;      Select foreground
BrColourW1TitleBG   : byte;      Title background
BrColourW1TitleFG   : byte;      Title foreground
BrColourW2BG        : byte;      Background
BrColourW2Brdr      : byte;      Border
BrColourW2FG        : byte;      Window 2 Detail:Foreground
BrColourW2TitleBG   : byte;      Title background
BrColourW2TitleFG   : byte;      Title foreground
BrColourW3BG        : byte;      Background
BrColourW3Brdr      : byte;      Border
BrColourW3FG        : byte;      Window 3 Index:Foreground
BrColourW3PgBG      : byte;      PgUp/Down status background
BrColourW3PgFG      : byte;      PgUp/Down status foreground
BrColourW3SelectBG  : byte;      Select background
BrColourW3SelectFG  : byte;      Select foreground
BrColourW3TitleBG   : byte;      Title background
```

```
BrColourW3TitleFG  : byte;       Title foreground
BrColourW4BG       : byte;       Background
BrColourW4Brdr     : byte;       Border
BrColourW4FG       : byte;       Windows 4-10 Path:Foreground
BrColourW4PathBG   : byte;       Highlight background
BrColourW4PathFG   : byte;       Highlight foreground
BrColourW4TitleBG  : byte;       Title background
BrColourW4TitleFG  : byte;       Title foreground
BrCurrDir          : BrFnStr;    Current directory and path
BrCurrSelection    : byte;       Current menu selection
BrDetColumn        : byte;       Current column in line buffer
BrDetCurline       : PbrDetlines;Current detail screen line
BrDetTopline       : PbrDetlines;Top detail text line
BrErrorFile        : BrFnStr;    Error file name and path
BrFile             : BrFnStr;    Browser file name
BrFinish           : boolean;    Loop help terminator
BrIndArray         : BrIndexArray;Index array for screen
BrIndCurline       : PbrIndexList;Current index text line
BrIndexFlag        : boolean;    True if index defined
BrIndSelect        : byte;       Current Index item selection
BrIndTopline       : PbrIndexList;   Top index text line
BrKeyCurline       : PbrKeywordList; Current keyword text line
BrKeyTopline       : PbrKeywordList; Top keyword text line
BrKeyWord          : BrIndexStr;     Index keyword
BrKeywordFlag      : boolean;        True if keywords defined
BrNumMenuItems     : byte;           Number of menu items
BrOutCurline       : PbrOutlines;    Current outline text line
BrOutTopline       : PbrOutlines;    Top outline text line
BrPathTop          : PbrPathList;    Top of current path stack
BrSaveWindowName   : WindowNames;    Window name for reaccess
BrSortList         : PbrIndexList;   Index list sorted
BrTopIndexScreen   : PbrIndexList;   Top of index screen
```

B.1.5 Procedures and Functions

See Part II Chapter two for a description of each procedure and function.

Procedures	Units
BBrPush	BrTools
BrBeginWordCol	BrDetail
brDeleteIndex	BrIndex
BrDetBeginningLine	BrCursor
BrDetEndLine	BrCursor
BrDetWriteLine	BrScreen
BrDetWriteTitle	BrScreen
BrDisplayDetail	BrScreen
BrDisplayError	BrError
BrDisplayIndex	BrIndex

Procedure	Unit
brDisplayMenu	BruMenu
BrDisplayNext	BrTools
BrDisplayPath	BrPath
BrDisplayPrevious	BrTools
BrDownLine	BrCursor
BrExit	BrUExit
brExitMenu	BruMenu
BrFindKeyword	BrDetail
BrGetFirstSameLevel	BrTools
BrGetLastSameLevel	BrTools
BrGetMessage	BrError
BrGetNextKeyword	BrCursor
BrGetNextLink	BrTools
BrGetNextSameLevel	BrTools
BrGetNextScreen	BrTools
BrGetParent	BrTools
BrGetPrevLink	BrTools
BrGetPrevSameLevel	BrTools
BrGetPrevScreen	BrTools
BrGetReferenceWord	BrDetail
BrGetTopScreen	BrTools
BrHeapFunc	Browser
brInitIndex	BrIndex
brInitMenu	BruMenu
brInitVars	Browser
BrInsertDetLine	BrTools
BrInsertIndex	BrTools
BrInsertKeyword	BrTools
BrInsertOutLine	BrTools
brInsertSortPtr	BrIndex
BrLeftChar	BrCursor
brListIndex	BrIndex
BrLowerCase	BrTools
brMainMakeWindow	BrPath
brMarkIndex	BrIndex
BRMenu	BruMenu
BROpenFile	BrFileIO
Browse	Browser
BrPageDown	BrIndex
BrPageUp	BrIndex
BrPathBackTrack	BrDetail
brPathControl	BrPath
brPathDetail	BrPath
brPathMakeWindow	BrPath
BrPop	BrDetail
BrPush	BrDetail
BrReadColour	BrFileIO
BrReadFile	BrFileIO

Procedure	Unit
BrRightChar	BrCursor
BrSelectKeyword	BrDetail
BrSelectWord	BrDetail
BrSortIndex	BrIndex
BrSwitchDetail	BrDetail
BrUpLine	BrCursor
KeyMousePressed	KeyMouse
Mouse	KeyMouse
ReadKeyMouse	KeyMouse
SpecialKeys	KeyMouse

B.2 Error Handling

B.2.1 Error Messages

```
***  DOS errors - Pascal Reference Manual V 5
002  File not found
003  Path not found
004  Check number of open files
005  Check file access
006  Check file handle

012  Check file access code
015  Check drive number
017  Cannot rename across drives

***  I/O errors - IORESULT - Pascal Reference Manual V 5
100  Disk read problem
101  Disk write Problem
102  File not assigned
103  File not open
104  File not open for input
105  File not open for output
106  Check numeric format

***  I/O errors - IORESULT - Advisor Browser
120  Unexpected end of file
121  Undefined file problem

130  Check page status display
133  Check menu selection
134  Customization file not found

140  Not defined as a keyword
141  Help screen not developed
142  Index link does not exist
143  Index file does not exist

***  Critical errors - Pascal Reference Manual V 5
```

```
150  Disk is write-protected
151  Unknown unit
152  Drive not ready
153  Unknown command
154  CRC problem
155  Check drive request structure length
156  Disk seek problem
157  Unknown media type
158  Sector not found
159  Printer out of paper
160  Device write problem
161  Device read problem
162  Hardware problem

***  Critical errors - Advisor Browser
170  Not enough memory

***  Fatal errors - Pascal Reference Manual V 5
200  Division by zero
201  Range check problem
202  Stack overflow
203  Heap overflow
204  Check pointer operations
```

Appendix C. Programmer's Specifications - Browser Interface

C.1 Program Specifications

C.1.1 Files

Main files for compilation:
- Browser.tpu - main browser unit
- Keymouse.tpu- handles keyboard and mouse input
- Br*.tpu -

Main files for running:
- BrSet.Col
- BrSet.Ini

Window utility:
- Goof.tpu
- Qwik.tpu
- Strs.tpu
- Wndw.tpu
- Wutil.tpu

C.1.2 Constants

```
BRALBS        = 1011;  Alt backspace
BRALKI        = 1008;  Alt
BRALLS        = 1010;  Alt left shift
BRALMM        = 386;   Alt minus
BRALPP        = 387;   Alt equals
BRALRS        = 1009;  Alt right shift
BRBACKSPACE   = 8;     Go back one space
BRBELL        = 7;     Bell
BRBSHI        = 1003;  Shift backspace
BRCABS        = 1015;  Control alt backspace
BRCALS        = 1014;  Control alt left shift
BRCARS        = 1013;  Control alt right shift
BRCEND        = 373;   Control end
BRCHOM        = 375;   Control home
BRCLFA        = 371;   Control left arrow
BRCPRI        = 370;   Control print screen
BRCR          = 13;    Carriage return
BRCRCH        = #13;   Carriage Return
BRCRTA        = 372;   Control right arrow
BRCRTL        = 1004;  Control
BRCTAL        = 1012;  Control alt
BRCTBS        = 1007;  Control backspace
BRCTLS        = 1006;  Control left shift
BRCTRLPGDOWN  = 374;   Control page down
BRCTRLPGUP    = 388;   Control page up
BRCTRS        = 1005;  Control right shift
BRCURSORDOWN  = 336;   Cursor down one line
BRCURSORLEFT  = 331;   Cursor left one character
```

```
BRCURSORRIGHT= 333;   Cursor right one character
BRCURSORUP    = 328;  Cursor up one line
BRDEL         = 339;  Delete character under cursor
BRENDKEY      = 335;  End
BRESC         = 27;   Exit
BRESCCH       = #27;  Exit
BRGREYAST     = 311;  Grey asterisk
BRGREYMINUS   = 330;  Grey minus
BRGREYPLUS    = 334;  Grey plus
BRHOME        = 327;  Home
BRINS         = 338;  Insert
BRLF          = 10;   Line feed
BRLSHI        = 1002; Left shift
BRNO          = 78;   Ascii N - No
BRPGDOWN      = 337;  Page down
BRPGUP        = 329;  Page up
BRRSHI        = 1001; Right shift
BRRUB         = 127;  Erase
BRSCRN        = 259;  Shift carriage return
BRSDNA        = 262;  Shift page down
BRSEND        = 268;  Shift end
BRSHFDEL      = 260;  Shift delete
BRSHOM        = 267;  Shift home
BRSINS        = 258;  Shift insert
BRSKRL        = 1012; Scroll Lock
BRSLFA        = 263;  Shift left arrow
BRSMIL        = 265;  Shift middle keypad - # 5
BRSPGD        = 270;  Shift page down
BRSPGU        = 269;  Shift page up
BRSRTA        = 264;  Shift right arrow
BRSTAB        = 271;  Shift tab
BRSUPA        = 261;  Shift up arrow
BRTAB         = 9;    Tab
BRYES         = 89;   Ascii Y - Yes

                      Function keys 1-10
BRF1=315; BRF2=316; BRF3=317; BRF4=318; BRF5=319;
BRF6=320; BRF7=321; BRF8=322; BRF9=323; BRF10=324;
                      Shift function keys 1-10
BRSF1=340; BRSF2=341; BRSF3=342; BRSF4=343; BRSF5=344;
BRSF6=345; BRSF7=346; BRSF8=347; BRSF9=348; BRSF10=349;
                      Control function keys 1-10
BRCF1=350; BRCF2=351;    BRCF3=352; BRCF4=353; BRCF5=354;
BRCF6=355; BRCF7=356; BRCF8=357; BRCF9=358; BRCF10=359;
                      Alt function keys 1-10
BRAF1=360; BRAF2=361; BRAF3=362; BRAF4=363; BRAF5=364;
BRAF6=365; BRAF7=366; BRAF8=367; BRAF9=368; BRAF10=369;
                      Alt letters
BRAltQ=272; BRAltW=273; BRAltE=274; BRAltR=275; BRAltT=2 76;
BRAltY=277; BRAltU=278; BRAltI=279; BRAltO=280; BRAltP=281;
BRAltA=286; BRAltS=287; BRAltD=288; BRAltF=289; BRAltG=290;
BRAltH=291; BRAltJ=292; BRAltK=293; BRAltL=294; BRAltZ=300;
```

```
BRAltX=301;              Exit
BRAltC=302; BRAltV=303; BRAltB=304; BRAltN=305; BRAltM=306;
                        Alt numbers
BRALT1=376; BRALT2=377; BRALT3=378; BRALT4=379; BRALT5=380;
BRALT6=381; BRALT7=382; BRALT8=383; BRALT9=384; BRALT0=385;
```

C.1.3 Data Types

none

C.1.4 Variables

```
BRHelpFlag     : byte;           Flags first entry to help
BRKeyboardHigh : byte absolute $40:$18; High bits key pressed
BRKeyboardLow  : byte absolute $40:$17; Low bits key pressed
BRKeyButton    : integer;        Key or mouse button
```

C.1.5 Procedures and Functions

Procedures	Unit
KeyMousePressed	Keymouse
Mouse	Keymouse
ReadKeyMouse	Keymouse
SpecialKeys	Keymouse

Appendix D. Customization - Designer

D.1 Program Specifications - Designer

D.1.1 Files

Main files:

- Advset.Ini : Colour setup file
- Advset.Col : Colour monitor file
- Advset.Bw : Monochrome monitor file

D.1.2 Constants

```
INITFILENAME    = 'ADVSET.INI';  Setup file
INITMFILENAME   = 'ADVSET.BW';   Monochrome setup file
INITCFILENAME   = 'ADVSET.COL';  Colour setup file
CR              = #13;           Carriage Return
ESC             = #27;           Escape key
CURSORUP        = #72;           Arrow up key
CURSORDOWN      = #80;           Arrow down key
CURSORRIGHT     = #77;           Arrow down key
CURSORLEFT      = #75;           Arrow down key

STW10A1 : string[2]  = ' F';     Main Installation menu
STW10A2 : string[22] = 'unction key assignment';
STW10B1 : string[2]  = ' S';
STW10B2 : string[22] = 'et Colours            ';
STW10C1 : string[2]  = ' Q';
STW10C2 : string[22] = 'uit/Save              ';
                                 Set Colours
STW12A1 : string[2]  = ' C';
STW12A2 : string[22] = 'olour Customization   ';
STW12B1 : string[2]  = ' D';
STW12B2 : string[22] = 'efault Colour Set     ';
                                 Colour Customization
STW20A1 : string[2]  = ' M';
STW20A2 : string[22] = 'ain Menu              ';
STW20B1 : string[2]  = ' O';
STW20B2 : string[22] = 'utline                ';
STW20C1 : string[2]  = ' D';
STW20C2 : string[22] = 'etail                 ';
STW20D1 : string[2]  = ' K';
STW20D2 : string[22] = 'eyword Reference      ';
                                 Main Menu
STW21A1 : string[2]  = ' A';
STW21A2 : string[15] = '. Text         ';
STW21B1 : string[2]  = ' B';
STW21B2 : string[15] = '. First Letter ';
STW21C1 : string[2]  = ' C';
STW21C2 : string[15] = '. Filename     ';
STW21D1 : string[2]  = ' D';
STW21D2 : string[15] = '. Status       ';
```

```
STW21E1 : string[2]  = ' E';
STW21E2 : string[15] = '. Error Message';
                                Outline
STW22A1 : string[2]  = ' A';
STW22A2 : string[15] = '. Text          ';
STW22B1 : string[2]  = ' B';
STW22B2 : string[15] = '. Title         ';
STW22C1 : string[2]  = ' C';
STW22C2 : string[15] = '. Index         ';
STW22D1 : string[2]  = ' D';
STW22D2 : string[15] = '. Level 0       ';
STW22E1 : string[2]  = ' E';
STW22E2 : string[15] = '. Level 1       ';
STW22F1 : string[2]  = ' F';
STW22F2 : string[15] = '. Level 2       ';
STW22G1 : string[2]  = ' G';
STW22G2 : string[15] = '. Level 3       ';
STW22H1 : string[2]  = ' H';
STW22H2 : string[15] = '. Level 4       ';
STW22I1 : string[2]  = ' I';
STW22I2 : string[15] = '. Level 5       ';
STW22J1 : string[2]  = ' J';
STW22J2 : string[15] = '. Level 6       ';
                                Detail
STW23A1 : string[2]  = ' A';
STW23A2 : string[15] = '. Text          ';
STW23B1 : string[2]  = ' B';
STW23B2 : string[15] = '. Title         ';
STW23C1 : string[2]  = ' C';
STW23C2 : string[15] = '. PG Up/Down    ';
STW23D1 : string[2]  = ' D';
STW23D2 : string[15] = '. Keyword       ';
                                Keyword
STW24A1 : string[2]  = ' A';
STW24A2 : string[15] = '. Sequence ID   ';
STW24B1 : string[2]  = ' B';
STW24B2 : string[15] = '. Link item bar';

                                Colour Defaults
DColourW1FG        : byte = $07;  Window 1 - Menu foreground
DColourW1BG        : byte = $10;  background
DColourW1Brdr      : byte = $10;  Brdr
DColourW1LetterFG  : byte = $04;  Letter    - foreground
DColourW1LetterBG  : byte = $70;              - background
DColourW1FileFG    : byte = $0E;  Filename   - foreground
DColourW1FileBG    : byte = $10;              - background
DColourW1StatusFG  : byte = $0E;  Status     - foreground
DColourW1StatusBG  : byte = $10;              - background
DColourW1ErrorFG   : byte = $0F;  Error      - foreground
DColourW1ErrorBG   : byte = $40;              - background

DColourW2FG       : byte = $07;   Window 2  - Outline Foreground
```

```
DColourW2BG       : byte = $10;                 - background
DColourW2Brdr     : byte = $30;    Brdr
DColourW2TitleFG  : byte = $0E;    Title       - foreground
DColourW2TitleBG  : byte = $10;                - background
DColourW2Level0FG: byte = $00;     Level -     0  foreground
DColourW2Level0BG: byte = $10;                 0  background
DColourW2Level1FG: byte = $02;                 1  foreground
DColourW2Level1BG: byte = $10;                 1  background
DColourW2Level2FG: byte = $03;                 2  foreground
DColourW2Level2BG: byte = $10;                 2  background
DColourW2Level3FG: byte = $04;                 3  foreground
DColourW2Level3BG: byte = $10;                 3  background
DColourW2Level4FG: byte = $05;                 4  foreground
DColourW2Level4BG: byte = $10;                 4  background
DColourW2Level5FG: byte = $06;                 5  foreground
DColourW2Level5BG: byte = $10;                 5  background
DColourW2Level6FG: byte = $07;                 6  foreground
DColourW2Level6BG: byte = $10;                 6  background
DColourW2SelectFG: byte = $0E;     Select bar foreground
DColourW2SelectBG: byte = $70;     Select bar background
DColourW3FG       : byte = $0E;    Window 3    - Detail Foreground
DColourW3BG       : byte = $30;                - background
DColourW3Brdr     : byte = $10;    Brdr
DColourW3TitleFG  : byte = $0E;    Title       - foreground
DColourW3TitleBG  : byte = $10;                - background
DColourW3PgFG     : byte = $0E;    PgUp        - foreground
DColourW3PgBG     : byte = $10;                - background
DColourW4FG       : byte = $07;    Window 4    - Keyword Foreground
DColourW4BG       : byte = $10;                - background
DColourW4Brdr     : byte = $10;    Brdr
DColourW4TitleFG  : byte = $0E;    Title       - foreground
DColourW4TitleBG  : byte = $10;                - background
DColourIndexFG    : byte = $0E;    Index       - Text Foreground
DColourIndexBG    : byte = $70;                - background

DColourKeywordFG  : byte = $04;    Window Keyword - Foreground
DColourKeywordBG  : byte = $70;                   - background
```

D.1.3 Data Types

none

D.1.4 Variables

```
MenuNum           : byte;          Menu number
CurrSelection     : byte;          Menu selection
NumMenuItems      : byte;          Number of items on a menu
SaveSelection10   : byte;          Number selected item -menu 1
```

```
   SaveSelection11   : byte;          Number selected item -menu 11
   SaveSelection12   : byte;          Number selected item -menu 12
   SaveSelection20   : byte;          Number selected item -menu 20
   ColourBWFlag      : char;          Indicates colour or monochrome
   Finish            : boolean;       True if program finished
   InitFile          : string[10];    Customization file

   Window colour customization - Permanent colours
                                      Window 1 - Menu
ColourW1FG         : byte;            Window 1 - Text foreground
ColourW1BG         : byte;            Window 1 - Background
ColourW1Brdr       : byte;            Window 1 - Border
ColourW1LetterFG   : byte;            Window 1 - 1st letter FG
ColourW1LetterBG   : byte;            Window 1 - 1st letter BG
ColourW1FileFG     : byte;            Window 1 - File foreground
ColourW1FileBG     : byte;            Window 1 - File background
ColourW1StatusFG   : byte;            Window 1 - Status foreground
ColourW1StatusBG   : byte;            Window 1 - Status background
ColourW1ErrorFG    : byte;            Window 1 - Error foreground
ColourW1ErrorBG    : byte;            Window 1 - Error background

                                      Window 2 - Outline
ColourW2FG         : byte;            Window 2 - Text foreground
ColourW2BG         : byte;            Window 2 - Background
ColourW2Brdr       : byte;            Window 2 - Border
ColourW2TitleFG    : byte;            Window 2 - Title foreground
ColourW2TitleBG    : byte;            Window 2 - Title background
ColourW2Level0FG   : byte;            Window 2 - Level 0 foreground
ColourW2Level0BG   : byte;            Window 2 - Level 0 background
ColourW2Level1FG   : byte;            Window 2 - Level 1 foreground
ColourW2Level1BG   : byte;            Window 2 - Level 1 background
ColourW2Level2FG   : byte;            Window 2 - Level 2 foreground
ColourW2Level2BG   : byte;            Window 2 - Level 2 background
ColourW2Level3FG   : byte;            Window 2 - Level 3 foreground
ColourW2Level3BG   : byte;            Window 2 - Level 3 background
ColourW2Level4FG   : byte;            Window 2 - Level 4 foreground
ColourW2Level4BG   : byte;            Window 2 - Level 4 background
ColourW2Level5FG   : byte;            Window 2 - Level 5 foreground
ColourW2Level5BG   : byte;            Window 2 - Level 5 background
ColourW2Level6FG   : byte;            Window 2 - Level 6 foreground
ColourW2Level6BG   : byte;            Window 2 - Level 6 background
ColourW2SelectFG   : byte;            Select bar foreground
ColourW2SelectBG   : byte;            Select bar background
                                      Window 3 - Detail
ColourW3FG         : byte;            Window 3 - Text foreground
ColourW3BG         : byte;            Window 3 - Background
ColourW3Brdr       : byte;            Window 3 - Border
ColourW3TitleFG    : byte;            Window 3 - Title foreground
ColourW3TitleBG    : byte;            Window 3 - Title background
ColourW3PgFG       : byte;            Window 3 - PgUp/Down Msg text
ColourW3PgBG       : byte;            Window 3 - PgUp/Down BG
                                      Window 4 - Sequence ID
```

```
ColourW4FG        : byte;          Window 4 - Text foreground
ColourW4BG        : byte;          Window 4 - Background
ColourW4Brdr      : byte;          Window 4 - Border
ColourW4TitleFG   : byte;          Window 4 - Title foreground
ColourW4TitleBG   : byte;          Window 4 - Title background
ColourIndexFG     : byte;          Window Index - foreground
ColourIndexBG     : byte;          Window Index - Background
ColourKeywordFG   : byte;          Window Keyword - foreground
ColourKeywordBG   : byte;          Window Keyword - Background
 Window colour customization - Temporary colours
                                   Window 1 - Menu
TColourW1FG         : byte;        Window 1 - Text foreground
TColourW1BG         : byte;        Window 1 - Background
TColourW1Brdr       : byte;        Window 1 - Border
TColourW1LetterFG   : byte;        Window 1 - 1st letter FG
TColourW1LetterBG   : byte;        Window 1 - 1st letter BG
TColourW1FileFG     : byte;        Window 1 - File foreground
TColourW1FileBG     : byte;        Window 1 - File background
TColourW1StatusFG   : byte;        Window 1 - Status foreground
TColourW1StatusBG   : byte;        Window 1 - Status background
TColourW1ErrorFG    : byte;        Window 1 - Error foreground
TColourW1ErrorBG    : byte;        Window 1 - Error background
                                   Window 2 - Outline
TColourW2FG         : byte;        Window 2 - Text foreground
TColourW2BG         : byte;        Window 2 - Background
TColourW2Brdr       : byte;        Window 2 - Border
TColourW2TitleFG    : byte;        Window 2 - Title foreground
TColourW2TitleBG    : byte;        Window 2 - Title background
TColourW2Level0FG   : byte;        Window 2 - Level 0 foreground
TColourW2Level0BG   : byte;        Window 2 - Level 0 background
TColourW2Level1FG   : byte;        Window 2 - Level 1 foreground
TColourW2Level1BG   : byte;        Window 2 - Level 1 background
TColourW2Level2FG   : byte;        Window 2 - Level 2 foreground
TColourW2Level2BG   : byte;        Window 2 - Level 2 background
TColourW2Level3FG   : byte;        Window 2 - Level 3 foreground
TColourW2Level3BG   : byte;        Window 2 - Level 3 background
TColourW2Level4FG   : byte;        Window 2 - Level 4 foreground
TColourW2Level4BG   : byte;        Window 2 - Level 4 background
TColourW2Level5FG   : byte;        Window 2 - Level 5 foreground
TColourW2Level5BG   : byte;        Window 2 - Level 5 background
TColourW2Level6FG   : byte;        Window 2 - Level 6 foreground
TColourW2Level6BG   : byte;        Window 2 - Level 6 background
TColourW2SelectFG   : byte;        Select bar foreground
TColourW2SelectBG   : byte;        Select bar background
                                   Window 3 - Detail
TColourW3FG         : byte;        Window 3 - Text foreground
TColourW3BG         : byte;        Window 3 - Background
TColourW3Brdr       : byte;        Window 3 - Border
TColourW3TitleFG    : byte;        Window 3 - Title foreground
TColourW3TitleBG    : byte;        Window 3 - Title background
TColourW3PgFG       : byte;        Window 3 - PgUp/Down Msg text
TColourW3PgBG       : byte;        Window 3 - PgUp/Down Msg BG
```

```
                                        Window 4 - Sequence ID
TColourW4FG         : byte;             Window 4 - Text foreground
TColourW4BG         : byte;             Window 4 - Background
TColourW4Brdr       : byte;             Window 4 - Border
TColourW4TitleFG    : byte;             Window 4 - Title foreground
TColourW4TitleBG    : byte;             Window 4 - Title background
TColourIndexFG      : byte;             Window Index - foreground

TColourIndexBG      : byte;             Window Index - Background
TColourKeywordFG    : byte;             Window Keyword - foreground
TColourKeywordBG    : byte;             Window Keyword - Background
```

D.1.5 Procedures and Functions

Procedures	Unit
ColourBarSelection	IScreen
DisplayColourBar	IScreen
DisplayMenu	IMenu
DisplaySampleWindows	IScreen
GetDefaults	IScreen
GetSelection	IMenu
GetTempColours	IScreen
initMenu	IMenu
Menu	IMenu
OpenFile	IFileIO
ReadFile	IFileIO
SaveDefaults	IScreen
SaveFile	IFileIO
SelectionItem	IMenu
UpdateColourBar	IScreen
UpdateSampleWindows	IScreen
WriteInstructions	IScreen

D.2 Running Advsetup

Advsetup is the customization program for the Advisor Designer. It allows the selection of colour or monochrome monitors. If colour is chosen, a complete set of menu items allows for the change of any screen display item. Sample screens are provided to view the overall effect of the colour change.

First entry to Advsetup generates a prompt to determine the monitor type. From this information the setup or initialization file ADVSET.INI is created. If a monochrome screen is selected, the program terminates. If a colour screen is selected a menu offering the options to change colours or quit is displayed.

If the Set Colours option is selected from the main menu, a submenu appears. Menu items include colour customization and de-

fault colours. Colour customization is used to select screen parts one by one and make colour choices. Default colours gives the option of returning to the default colours set.

From the colour customization selection, another menu of screen sections is displayed. Selection of any of these options brings up a final menu with specific screen items. Once one of these items is selected, a colour bar and sample screen are displayed.

Moving the arrow keys causes the cursor to move throughout the colour bar. The changes are reflected in the menu item of choice. The colour bar shows an X with foreground colour over the given background colour. To select a colour, press ENTER. Otherwise press ESCape to exit the colour bar without making changes.

The ESCape key is used to exit back to the main menu. When reached, F10 causes an exit from the program and save of any selected items.

Appendix E. Customization - Browser

E.2 Program Specifications - Browser

E.1.1 Files

Main files:

```
     Brset.Ini : Browser setup file (copy from .Col or .Bw)
     BrSet.Col : Browser setup file for colour monitor
     BrSet.Bw  : Browser setup file for monochrome monitor
```

E.1.2 Constants

```
    INITFILENAME    = 'BRSET.INI';    Setup file
    INITMFILENAME   = 'BRSET.BW';     Monochrome setup file
    INITCFILENAME   = 'BRSET.COL';    Colour setup file
    CR              = #13;            Carriage Return
    ESC             = #27;            Escape key
    CURSORUP        = #72;            Arrow up key
    CURSORDOWN      = #80;            Arrow down key
    CURSORRIGHT     = #77;            Arrow down key
    CURSORLEFT      = #75;            Arrow down key
                                      Main Installation menu
    STW10A1 : string[2]  = ' F';
    STW10A2 : string[22] = 'unction key assignment';
    STW10B1 : string[2]  = ' S';
    STW10B2 : string[22] = 'et Colours            ';
    STW10C1 : string[2]  = ' Q';
    STW10C2 : string[22] = 'uit/Save              ';
                                      Set Colours
    STW12A1 : string[2]  = ' C';
    STW12A2 : string[22] = 'olour Customization   ';
    STW12B1 : string[2]  = ' M';
    STW12B2 : string[22] = 'ain-Default Colour Set';
    STW12C1 : string[2]  = ' P';
    STW12C2 : string[22] = 'ath-Default Colour Set';
                                      Colour Customization
    STW20A1 : string[2]  = ' M';
    STW20A2 : string[22] = 'ain Menu              ';
    STW20B1 : string[2]  = ' D';
    STW20B2 : string[22] = 'etail                 ';
    STW20C1 : string[2]  = ' I';
    STW20C2 : string[22] = 'ndex                  ';
    STW20D1 : string[2]  = ' P';
    STW20D2 : string[22] = 'ath                   ';
                                      Main Menu
    STW21A1 : string[2]  = ' A';
    STW21A2 : string[15] = '. Text        ';
    STW21B1 : string[2]  = ' B';
    STW21B2 : string[15] = '. First Letter ';
    STW21C1 : string[2]  = ' C';
    STW21C2 : string[15] = '. Title       ';
```

```
STW21D1 : string[2]  = ' D';
STW21D2 : string[15] = '. Select Bar   ';
STW21E1 : string[2]  = ' E';
STW21E2 : string[15] = '. Error Message';
                              Detail

STW22A1 : string[2]  = ' A';
STW22A2 : string[15] = '. Text         ';
STW22B1 : string[2]  = ' B';
STW22B2 : string[15] = '. Title        ';
STW22C1 : string[2]  = ' C';
STW22C2 : string[15] = '. Keyword      ';
                              Index

STW23A1 : string[2]  = ' A';
STW23A2 : string[15] = '. Text         ';
STW23B1 : string[2]  = ' B';
STW23B2 : string[15] = '. Title        ';
STW23C1 : string[2]  = ' C';
STW23C2 : string[15] = '. Pg Up/Down   ';
STW23D1 : string[2]  = ' D';
STW23D2 : string[15] = '. Select Bar   ';
                              Path

STW24A1 : string[2]  = ' A';
STW24A2 : string[15] = '. Text         ';
STW24B1 : string[2]  = ' B';
STW24B2 : string[15] = '. Title        ';
STW24C1 : string[2]  = ' C';
STW24C2 : string[15] = '. Highlight Bar';
                              Colour Defaults
DColourW1FG        : byte = $07;  Window 1  - Menu foreground
DColourW1BG        : byte = $10;              - background
DColourW1Brdr      : byte = $30;  Brdr
DColourW1LetterFG  : byte = $04;  Letter    - foreground
DColourW1LetterBG  : byte = $70;              - background
DColourW1TitleFG   : byte = $0E;  Title     - foreground
DColourW1TitleBG   : byte = $30;              - background
DColourW1SelectFG  : byte = $0E;  Select    - foreground
DColourW1SelectBG  : byte = $30;              - background
DColourW1ErrorFG   : byte = $0F;  Error     - foreground
DColourW1ErrorBG   : byte = $40;              - background

DColourW2FG        : byte = $07;  Window 2  - Detail Foreground
DColourW2BG        : byte = $10;              - background
DColourW2Brdr      : byte = $30;  Brdr
DColourW2TitleFG   : byte = $0E;  Title     - foreground
DColourW2TitleBG   : byte = $30;              - background
DColourW3FG        : byte = $0E;  Window 3  - Index Foreground
DColourW3BG        : byte = $10;              - background
DColourW3Brdr      : byte = $30;  Brdr
DColourW3TitleFG   : byte = $0E;  Title     - foreground
DColourW3TitleBG   : byte = $30;              - background
DColourW3PgFG      : byte = $0E;  PgUp      - foreground
DColourW3PgBG      : byte = $10;              - background
```

```
DColourW3SelectFG: byte = $0E;   Select bar      - foreground
DColourW3SelectBG: byte = $30;                   - background

DColourW4FG      : byte = $0E;   Window 4  - Path Foreground
DColourW4BG      : byte = $10;             - background
DColourW4Brdr    : byte = $30;   Brdr
DColourW4TitleFG : byte = $0E;   Title     - foreground
DColourW4TitleBG : byte = $30;             - background
DColourW4PathFG  : byte = $0E;   Path - foreground
DColourW4PathBG  : byte = $30;        - background
DColourKeywordFG : byte = $04;   Text foreground
DColourKeywordBG : byte = $70;        background
```

E.1.3 Data Types

none

E.1.4 Variables

```
MenuNum           : byte;         Menu number
CurrSelection     : byte;         Menu selection
NumMenuItems      : byte;         Number of items on a menu
SaveSelection10   : byte;         Number selected item -menu 1
SaveSelection11   : byte;         Number selected item -menu 11
SaveSelection12   : byte;         Number selected item -menu 12
SaveSelection20   : byte;         Number selected item -menu 20
Finish            : boolean;      True if program finished
InitFile          : string[10];   Customization file
ColourBWFlag      : char;         Indicates colour or monochrome
 Window colour customization - Permanent colours
                                  Window 1  - Menu
ColourW1FG        : byte;         Foreground
ColourW1BG        : byte;         Background
ColourW1Brdr      : byte;         Border
ColourW1LetterFG  : byte;         First letter foreground
ColourW1LetterBG  : byte;         First letter background
ColourW1TitleFG   : byte;         Title foreground
ColourW1TitleBG   : byte;         Title background
ColourW1SelectFG  : byte;         Select foreground
ColourW1SelectBG  : byte;         Select background
ColourW1ErrorFG   : byte;         Error foreground
ColourW1ErrorBG   : byte;         Error background
                                  Window 2 - Detail
ColourW2FG        : byte;         Foreground
ColourW2BG        : byte;         Background
ColourW2Brdr      : byte;         Border
ColourW2TitleFG   : byte;         Title foreground
ColourW2TitleBG   : byte;         Title background
                                  Window 3 - Index
```

```
ColourW3FG        : byte;        Foreground
ColourW3BG        : byte;        Background
ColourW3Brdr      : byte;        Border
ColourW3TitleFG   : byte;        Title foreground
ColourW3TitleBG   : byte;        Title background
ColourW3SelectFG  : byte;        Select foreground
ColourW3SelectBG  : byte;        Select background
ColourW3PgFG      : byte;        PgUp/Down status foreground
ColourW3PgBG      : byte;        PgUp/Down status background
                                 Window 4 - 10   - Path
ColourW4FG        : byte;        Foreground
ColourW4BG        : byte;        Background
ColourW4Brdr      : byte;        Border
ColourW4TitleFG   : byte;        Title foreground
ColourW4TitleBG   : byte;        Title background
ColourW4PathFG    : byte;        Highlight foreground
ColourW4PathBG    : byte;        Highlight background
                                 Keyword
ColourKeywordFG   : byte;        Foreground
ColourKeywordBG   : byte;        Background
  Window colour customization - Temporary colours
                                 Window 1  - Menu
TColourW1FG         : byte;      Foreground
TColourW1BG         : byte;      Background
TColourW1Brdr       : byte;      Border
TColourW1LetterFG   : byte;      First letter foreground
TColourW1LetterBG   : byte;      First letter background
TColourW1TitleFG    : byte;      Title foreground
TColourW1TitleBG    : byte;      Title background
TColourW1SelectFG   : byte;      Select foreground
TColourW1SelectBG   : byte;      Select background
TColourW1ErrorFG    : byte;      Error foreground
TColourW1ErrorBG    : byte;      Error background
                                 Window 2 - Detail
TColourW2FG         : byte;      Foreground
TColourW2BG         : byte;      Background
TColourW2Brdr       : byte;      Border
TColourW2TitleFG    : byte;      Title foreground
TColourW2TitleBG    : byte;      Title background
                                 Window 3 - Index
TColourW3FG         : byte;      Foreground
TColourW3BG         : byte;      Background
TColourW3Brdr       : byte;      Border
TColourW3TitleFG    : byte;      Title foreground
TColourW3TitleBG    : byte;      Title background
TColourW3SelectFG   : byte;      Select foreground
TColourW3SelectBG   : byte;      Select background
TColourW3PgFG       : byte;      PgUp/Down status foreground
TColourW3PgBG       : byte;      PgUp/Down status background
                                 Window 4 - 10   - Path
TColourW4FG         : byte;      Foreground
TColourW4BG         : byte;      Background
```

```
TColourW4Brdr     : byte;       Border
TColourW4TitleFG  : byte;       Title foreground
TColourW4TitleBG  : byte;       Title background
TColourW4PathFG   : byte;       Highlight foreground
TColourW4PathBG   : byte;       Highlight background
                                Keyword
TColourKeywordFG  : byte;       Foreground
TColourKeywordBG  : byte;       Background
```

E.1.5 Procedures and Functions

Procedures	Unit
ColourBarSelection	IBrScr
DisplayColourBar	IBrScr
DisplayMenu	IBrMenu
DisplaySampleWindows	IBrScr
GetDefaults	IBrScr
GetSelection	IBrMenu
GetTempColours	IBrScr
initMenu	IBrMenu
Menu	IBrMenu
OpenFile	IBrFile
ReadFile	IBrFile
SaveDefaults	IBrScr
SaveFile	IBrFile
SelectionItem	IBrMenu
UpdateColourBar	IBrScr
UpdateSampleWindows	IBrScr
WriteInstructions	IBrScr

E.1.6 Browser Customization file

If the application programmer plans to use their own customization routine to update the Browser setup files, it will be necessary to write to the files in a way in which the Browser unit can read it back on startup. The Browser setup files include: BrSet.Col (colours for colour monitor); BrSet.BW (colours for monochrome monitor); and BrSet.Ini (copied from one of the previous two files during the setup of the Browser). The contents of these files is as follows:

```
ColourBWFlag     : byte; Colour flag
                         Window 1 - Menu
ColourW1FG       : byte; Foreground
ColourW1BG       : byte; Background
ColourW1Brdr     : byte; Border
ColourW1LetterFG : byte; First letter foreground
ColourW1LetterBG : byte; First letter background}
```

```
ColourW1TitleFG  : byte; Title FG
ColourW1TitleBG  : byte; Title BG
ColourW1SelectFG : byte; Select FG
ColourW1SelectBG : byte; Select BG
ColourW1ErrorFG  : byte; Error status foreground
ColourW1ErrorBG  : byte; Error status background
                         Window 2 - Detail
ColourW2FG       : byte; Text foreground
ColourW2BG       : byte; Background
ColourW2Brdr     : byte; Border
ColourW2TitleFG  : byte; Title foreground
ColourW2TitleBG  : byte; Title background
                         Window 3 - Index
ColourW3FG       : byte; Text foreground
ColourW3BG       : byte; Background
ColourW3Brdr     : byte; Border
ColourW3TitleFG  : byte; Title foreground
ColourW3TitleBG  : byte; Title background
ColourW3PgFG     : byte; PgUp/Down Msg text
ColourW3PgBG     : byte; PgUp/Down Msg background}
ColourW3SelectFG : byte; Select FG
ColourW3SelectBG : byte; Select BG
                         Window 4 - Path Levels 0 to 6
ColourW4FG       : byte; Text foreground
ColourW4BG       : byte; Background
ColourW4Brdr     : byte; Border
ColourW4TitleFG  : byte; Title foreground
ColourW4TitleBG  : byte; Title background
ColourW4PathFG   : byte; Highlight foreground
ColourW4PathBG   : byte; Highlight background
                         Window Keyword
ColourKeywordFG  : byte; Text foreground
ColourKeywordBG  : byte; Background
```

E.2 Running Brsetup

Brsetup is the customization program for the Advisor Browser. It allows the selection of colour or monochrome monitors. If colour is chosen, a complete set of menu items allows for the change of any screen display item. Sample screens are provided to view the overall effect of the colour change.

First entry to Brsetup generates a prompt to determine the monitor type. From this information the setup or initialization file BRSET.INI is created. If a monochrome screen is selected, the program terminates. If a colour screen is selected a menu offering the options to change colours or quit is displayed.

If the Set Colours option is selected from the main menu, a submenu appears. Menu items include colour customization and default colours for the main windows or path viewing screen.

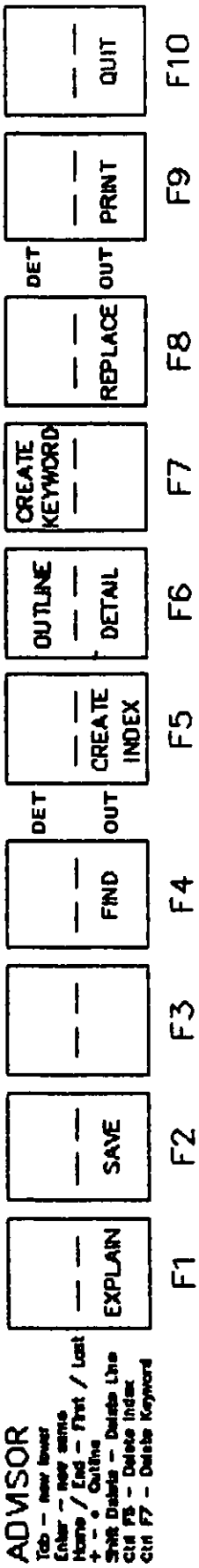
Colour customization is used to select screen parts one by one and make colour choices. Default colours gives the option of returning to the default colours set.

From the colour customization selection, another menu of screen sections is displayed. Selection of any of these options brings up a final menu with specific screen items. Once one of these items is selected, a colour bar and sample screen are displayed.

Moving the arrow keys causes the cursor to move throughout the colour bar. The changes are reflected in the menu item of choice. The colour bar shows an X with foreground colour over the given background colour. To select a colour, press ENTER. Otherwise press ESCape to exit the colour bar without making changes.

The ESCape key is used to exit back to the main menu. When reached, F10 causes an exit from the program and save of any selected items.

Appendix F. Advisor Designer Function Key Template



References

Brown J. S., Burton R. R., Clancey W. J. (1984).
Applications of Artificial Intelligence to Training and Education. AAAI 1984 Conference Tutorial Program, Menlo Park, California.

Burke M. (1988).
Incremental Implementation of an Online Help Facility. 35th ITCC, Philadelphia, 1988: Freedom to Communicate, Proceedings published by Society for Technical Communications, Washington, DC, ATA 122-124.

Burton R. R., Brown J. S. (1982).
An Investigation of Computer Coaching for Informal Learning Activities. In D. Sleeman and J. S. Brown (Eds.). Intelligent Tutoring Systems, Orlando, Fl, Academic Press, 79-98.

Bush V. (1945).
As We May Think. Atlantic Monthly, (July 1945). Reprinted in S. Lambert and S. Ropiequet (Eds.), The New Papyrus, CD Rom: The Current and Future State of the Art, Microsoft Press, Redmond W.A., 1986.

Card S. J., Moran T. P. and Newell A. (1983).
The Psychology of Human-Computer Interaction. Erlbaum, Hillsdale, N. J., 1983.

Carroll J. M., Aaronson A. P. (1988).
Learning By Doing With Simulated Intelligent Help. Communications of the ACM, 31:9 (September), 1064-1079.

Carroll J. M. and Carruthers C., (1984).
Training Wheels in a User Interface. Communications of the ACM, 27, 8 (August 1984), 800-806.

Carroll J. M., McKendree J. (1987).
Interface Design Issues For Advice-Giving Expert Systems. Communications of the ACM, 30:1 (January), 14-31.

Conklin J. (1987).
Hypertext: An Introduction and Survey.
Computer, 20:9 (September), 17-41.

Dixon N. (1982).
Incorporating Learning Strategies into Training Design. Training and Development Journal, 36, 62-64.

Gagne R. M. (1965).
The Conditions of Learning. Holt Rinehart and Winston, New York, 1965.

Gagne R., Briggs L. (1974).
Principles of Instructional Design. Holt Rinehart and Winston, New York, 1974.

Goodman D. (1987).
The Complete HyperCard Handbook. Bantam Books, Toronto.

Gould J. D. and Lewis C. (1985).
Designing for Usability : Key Principles and What Designers Think. Communications of the ACM, 28:3 (March, 1985), 300-311.

Grimm S., Malicki J., Obermeyer S. (1988).
A User Needs Approach to Context-Sensitive Help. SIGCHI (Special Interest Group on Computer and Human Interaction, 19:3 (January), 65-67.

Houghton R. C., Jr. (1984).
Online Help Systems: A Conspectus. Communications of the ACM, 27, 2 (February 1984), 126-133.

Knapper C. K. (1980).
Evaluating Instructional Technology. A Halsted Press Book, John Wiley and Sons, New York, 1980.

Kolb D. A., Rubin I. M., McIntyre J. M. (1974).
Organizational Psychology: An Experimental Approach. Prentice Hall, Englewood Cliffs, New Jersey, 1974.

Nielsen J., Molich R. (1989).
Teaching User Interface Design Based on Usability Engineering. SIGCHI (Special Interest Group on Computer & Human Interaction), ACM Press, 21:1 (July), 45-48.

Palmer J., Duffy T., Gomoll K., Gomoll T., Richards-Palmquist J., Trumble J. (1988).
The Design and Evaluation of Online Help for Unix EMACS: Capturing the User in Menu Design. IEEE Transactions on Professional Communications, 31:1 (March), 44-51.

Shneiderman B., (1982).
The Future of Interactive Systems and the Emergence of Direct Manipulation. In Y. Vassiliou (Ed.), Human Factors and Interactive Computer Systems. Proceedings of the NYU Symposium on User Interfaces, New York, (May 26-28), Ablex Publishing Corporation, Norwood New Jersey., 1982.

Smith C., Irby C., Kimball R., Verplank B. and Harslem E. (1982).
Designing the Star User Interface, Byte, 7:4, 242-282.

Streitz N. N. (1988).
Mental Models and Metaphors: Implications for the Design of Adaptive User-System Interfaces. In H. Mandl and A. Lesgold (Eds.), Learning Issues for Intelligent Tutoring Systems, Springer-Verlag, New York, 1988, p. 164-186.