Software Support for a Hand-held Bird Identification Aid

Victor Orooji-Heravi

A project report submitted to the Facuity of Engineering, University of the Witwatersrand, Johannesburg, in partial fulfilment of the requirements for the degree of Master of Science in Engineering

Johannesburg 1990

DECLARATIONS

I declare that this dissertation is my own, unaided work. It is being submitted for partial fulfillment in the Degree of Master of Science in Engineering in the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination in any other university.

Sintu Orowji 17/10/90

ABSTRACT

In view of the recent breakthroughs in memory chips, electronic fieldguides are becoming more feasible, specially when they offer better portability and faster information access when compared to conventional fieldguides. In this report we discuss the importance of inexpensive electronic fieldguides which at the same time provide suitable user interface for maximum information transfer. The implications of software architecture are discussed during the course of designing the software model of a bird identifier fieldguide. This model is used to evaluate the behavioral performance and effectiveness of the user interface; both of which play a vital role in establishing the cost of the device. These observations are brought together in conclusion, where the potential role of electronic fieldguides in the domain of education and science are highlighted.

ACKNOWLEDGMENTS

I wish to express my gratitudes to my supervisor, Prof. A.J. Walker for his continuous support and encouragement during the course of this study.

CONTENTS Page

LIST OF FIGURES

CHAPTER 1- APPLICATION OF ELECTRONIC FIELDGUIDES

1.1- BACKGROUND

A fieldguide as the name suggests is a means of providing guidance (knowledge or information) towards the process of making certain field decisions. Field environments are defined by those situations where the means of getting valid information are not favorable. The prevailing of such a situation is due to the lack of one, or several resources:-

* Convenient access to domain expertise.
* Efficient communication channel to provide rapid information access.

Depending on the complexity of the subject, fieldguide implementations can vary from a simple chart to a comprehensive manual. Naturally, the value of a fieldguide is directly related to the complexity of the subject, and therefore to the amount of stored information and catered functionality. Fieldguides of major significant are usually implemented in book form, using textual and graphical representations to compile domain expertise. But situations that warrant the use of these fieldguides, involve considerable amount of information processing. This results in two major inadequacies:-

* Fieldguides become voluminous and inconvenient to carry around.
* To speed up information retrieval, better communication channels must be catered for. In books this is done by

copying information in several search categories. This redundancy will further increase the size of the book and results in less portability.

Electronic fieldguides access domain expertise electronically. If designed properly, the communication channel can yield much faster information access, when compared to conventional fieldguides. Moreover, recent enhancements in the area of memory capacity[1], make electronic fieldguides more portable. The potential sound/image processing capability of these devices is a further advantage to certain applications, where sound and image information provide the most powerful, and sometimes the only, means of furnishing expert guidance.

Provided the mechanics of the software is completely decoupled from the database that it manages; the electronic fieldguide can readily be fed from a selection of databases to specify it's domain of expertise. Admittedly, due to the diversity of applications, some software adjustments will be necessary to cater for the new functions. But this modification can be minimized by carefully choosing applications that share these functionalities. For instance, all animal identification fieldguides share the size, color, and habitats specifications. But perhaps the most powerful means of employing this type of software-database independency is through database grouping. In other words, the database may be partitioned into several logical groups based on key application areas. For example in the domain of wild flowers one can partition the database into geographical areas such as Europe, Africa or Asia. This

technique can be used to overcome possible memory limitations associated with database storage.

The cost of any product is as important as the function it serves to provide; and fieldguides are no exception to this fact. It is therefore important that the cost of fieldguides be limited to certain feasible boundaries. These boundaries may differ from one application to the next, but it is vital to be aware of this context. The final cost of a fieldguide must be justified in the context of its application.

1.2- OBJECTIVES

The object of the following work is to investigate the feasibility of producing cost effective fieldguides, capable of storing significant size of information. Cost considerations demand that the trailing edge of the leading edge of technology be exploited. Further cost reductions may be achieved through coding techniques which serve to extract maximum technological benefit from available memory devices. However, whilst the issue of memory has great relevance here, this issue is not the theme of this work. The emphasis here is on the functional and architectural impact of the supporting software. The evaluation of this software model will provide the fundamental guidelines and valuable insight into the computational needs; and assessing the effectiveness of the man-machine interface (communication channel). Both the hardware engine and the Man-Machine Interface (MMI) have a profound impact on the cost of the device.

Due to relative availability of in-house expertise in the area of bird identification, this subject forms the environment for developing the envisaged software model. The bird identifier is an application of electronic fieldguides, in the domain of bird recognition and identification.

1.3- THE CHALLENGE OF BIRD IDENTIFICATION

Roberts[2] and Newman[3] are well known fieldguides in the domain of bird identification. They provide the user with comprehensive textual and graphical descriptions of different bird species. The identification process begins with concentrating on the known facts and clues about the cited bird. These initial search conditions help narrow down on the number of candidate species, and therefore speed up the identification process. This exercise, however, is not always trivial, when the database is large. For instance in South Africa alone, as many as 911 species have been registered. In North America this figure is in the order of 435.

Furthermore, amongst some bird species (warblers, pipits, larks) the degree of similarity in appearance is such that the only means of identification is the call sequences that they generate. The absence of visual contact is another situation where the bird call recognition becomes the only route to identification. Examples of such situations are wetlands and forests. Obviously, one can not hope to find any assistance from paper-based fieldguides such as Roberts, when faced with such situations. This type of shortcoming further justifies the need

for electronic fieldguides like the bird identifier. But in spite of the sound reproduction capability, the bird identifier should be viewed as a complementary tool to guides like Roberts, for they still carry a vast amount of detail information.

1.4- SUMMARY

Advantages of electronic fieldguides:-

* Fast information access
* Better portablility
* Sound processing capability
* Application context may be readily changed by replacing the database.

Guidelines for producing electronic fieldguides:-

* Inexpensive final device
* User-friendly communication channel which provides rapid information access.
* Use of effective coding techniques.
* Convenient choice of software architecture.

CHAPTER 2- THE BIRD IDENTIFIER FIELDGUIDE

2.1- THE TASK AT HAND

The bird identifier project is in fact a collection of three separate efforts.

1) The investigation in efficient methods of coding and storing graphical data.
2) The investigation of efficient methods of coding and regeneration of bird calls.
3) The producing of a personal computer model to develop and asses the implications of software architecture.

The fact that the first two endeavors are still under investigation, has forced us to restrict the software model to text type information only. As far as the objectives of this work is concerned, this choice will not have a great bearing on our investigations.

In this chapter we will start by discussing the architectural choice of the bird identifier. We will then continue with the system specifications, by considering the general and detailed behavior of the identifier and it's MMI. Although bird image and bird call are not implemented in the final model, for the sake of completeness these facilities will be included during behavioral descriptions.

2.2- ARCHITECTURAL DESIGN STRATEGY

From the point of view of software engineering discipline, software should be designed for the four fundamental goals of modifiability, efficiency, reliability and understandability.

The specifications will describe features and functional properties of the bird identifier which should be adequate in addressing the bird identification exercise. However, this design is regarded as a discovery design, and therefor it is more than likely that other functions and facilities will have to be incorporated to accommodate the features which have been overlooked from the present description. In such a situation where a clear functionality of final system is not realized, it is imperative that modification and growth should readily be absorbed by the software.

While functional decomposition methods provide the powerful and efficient "divide and conquer" notion, they are inherently fragile to change[6,7]. This is because the method gives rise to a hierarchical structure in which changes such as inclusion of new modules or deletion of existing ones, specially in the embedded levels of hierarchy, result in major architectural break-ups.

On the other hand an expert system approach does seem to be a natural solution to the present design. A closer examination of expert system properties will in fact reveal the contrary of this misconception. Expert systems are inherently highly interactive in the sense that almost every initiated activity is quarried in order to bring about user awareness and provide

clarification on the system status. This state of affairs may be convenient to the novice user, but this situation will soon become reversed as a reasonable level of competency is reached. At this stage the user has gained enough experienced to know how to perform a task, and realize it's implications. But the relentless interrogation scheme of the system will constantly get in the way of the user, providing a barrier between the task at hand and system capability. Recalling that fast information access is a prerequisite criteria for defining a fieldguide, it becomes evident that the expert system approach is inconvenient and even detrimental to this application.

In contrast, Object-Oriented Design (OOD) approach to systems design is not bound by such anomalies and it is also highly impervious to modifications[6]. This is because the system is formed from a group of objects that are highly independent and encapsulated, resulting in a flat hierarchy. The issue of modifiability is vital to the survival of the bigger systems and those of discovery designs, where the full behavioral specifications of the system is not clear, initially. In this regard the bird identifier is very much a discovery design, in as much as the full implications of the envisaged system behavior will only become manifest after device has been put into use.

The issue of understandability is also strongly addressed by the OOD approach, since objects directly emulate the real world model, and thus make a great contribution to the undestandability of the system.

Based on these observations it was decided that the OOD strategy was the most suitable architecture for the bird identifier.

2.3- BEHAVIORAL SPECIFICATIONS

Physical characteristics and bird call, form the two major means of bird identification. The bird identifier is a hand-held device that identifies the bird species that satisfy a set of user defined descriptions. The function of the identifier is to provide a set of user selectable features for defining descriptions of the cited bird. These selected features are then compared with the database to provide the user with a list of bird species that match the given descriptions. An example of features (or attributes), is bird size, bird coloring, etc.. A typical session would start by initializing all selected attributes; meaning that the entire database will be available for user inspection. However such a vast list can not possibly be of any help to the identification endeavor. So the user will continue to truncate this list by selecting the noted features of the cited bird. With each subsequent enabling of a feature the list of matched birds will be decreased from the maximum value corresponding to the full database.

Following the completion of the feature selection activity, the user will then initiate a scan and match activity to view the list of bird species that have matched the attributes constraints. In this manner the identification task will be simplified to a case of choosing from a handful of possible species rather than being faced with the entire database of birds.

In order to aid the process of final identification the user can be provided with a graphical description or a playback of the bird call. Once the user has identified the species of interest from the list of matched birds, additional information on other characteristics, such as migration habits, etc. can be furnished.

Other useful functions, such as bird counting, will be incorporated to complement the two main activities of attribute specification, attribute mode, and bird matching, find mode. Later in Section 3.4 we will discuss the role of these complementary functions. In the following Sections the terms features and attributes will be used interchangeably to denote the same bird characteristics.

It is vital that the impact of each attribute selection/de-selection; on the number of qualifying species, be available to the user at the earliest possible opportunity. It seems natural to assume that such information should be furnished in the find mode following the scan and match activity. But this would necessitate the invocation of scan and match activity, each time an attribute parameter is changed. To avoid this unwarranted operation it was decided to display and update the number of matched birds, whilst still in the attribute mode. In this way the user is continuously advised of the effectiveness of his attribute selections, without having to view the actual list of matched birds.

2.4- TECHNICAL SPECIFICATIONS

The technical specifications are aimed at providing an acceptable performance (ie. speed of execution), low cost of the device and a simple but effective user interface. As far as the software model is concerned the issue of performance will have to be addressed in the implementation stage. We will therefore proceed to discuss the user interface and it's implications.

In the recent years it has become vividly clear that the user interface (or MMI), constitutes a significant component of a system[4]. In comparing the MMI of operating systems and adventure games, Bernice[5] identifies the following characteristics:-

* Tunnel vision Vs. World vision:
  - The vast majority of operating systems are based on textual command-driven interfaces and therefore provide very little information on the system's structure and it's capabilities.
  - Through graphics interfaces, adventure games convey a high-level picture of the system's structure and the tasks it supports.

* Rigid syntax Vs. Flexible syntax:
  - In the operating systems environments there is no room for flexibility in syntax. There is only one way to issue a command, and it must be phrased just right.
  - Adventure games often provide several alternatives for issuing commands. The user is provided with quick command format as well as a relatively longer but more descriptive format.

* Insensitivity Vs. Context-awareness:

  - In operating systems, insufficient feedback, obscure messages and lack of contextual awareness projects a rather hostile environment to the user.
  - Adventure games can play a collaborating role in getting clarification or confirmation from the user. Also in providing information or help to the user the context of the system's activity is considered.

* Inconsistency Vs. Consistency:

  - Existence of multiple interface styles within the same product, results in confusion and discourages the user. A sever case of inconsistency can ultimately guarantee the failure of a product.
  - The creators of adventure games being fully aware of these implications, have taken measures to ensure consistency throughout the package, and in most cases, this consistency has even been carried-out across the vendor products.

* Copious Documentation Vs. Intrinsic documentation:

  - To compensate for the lack of communication between the system and the user, manuals of substantial volume are written. Generally there is a universal attitude towards the using of these manuals. Users like to start on the basics without having to resort to manuals to perform even the simplest of tasks. They rather spend their valuable time, getting results, and are only prepared to refer to the system manuals when using the more complicated and advanced features of the system.

- Attention to the elements of user interface can make extensive documentation unnecessary. Examples of these interfaces are abundantly available amongst adventure games.

In view of the above observations, conscious effort is made to adopt the MMI concepts of computer games. A key factor in achieving this is the need for graphical capability. Meanwhile to address the issue of cost, severe limitations on the graphical capabilities must be imposed. It was decided to adopt simpler graphics in favor of reducing costs, but at the same context sensitive help would be utilized to alleviate the reduced global vision. This was considered a necessary measure to ensure that the cost of the identifier would remain comparable with other conventional guides such as Robert's book of birds.

2.4.1- DISPLAY UNIT

In the light of our primary object of cost reduction, a modest two line liquid crystal display of twenty characters length is chosen as the display unit.

In order to exploit the full benefit of the display in creating a good MMI, it was partitioned into five windows.

* W1 is eighteen characters long and serves to indicate the menu headings.
* W2 window is one character long and is reserved for indicating that the text on window 4 continues to the right.(»)

* W3 window is one character long and is reserved for indicating that the text on window 4 continues to the left.(«)

* W4 is twenty characters long. It is the area where the current entry of the menu is written.

* W5 is of four characters in length. This window normally supports the number of matched birds. It also serves to display the number of found birds when in the 'List Found Birds' Menu; number of selected attributes when in 'List Attributes' menu.

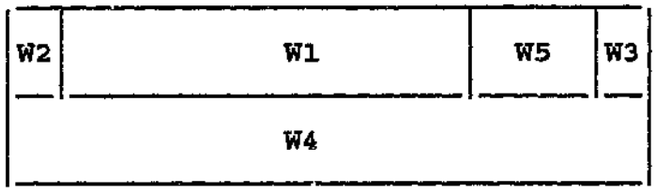


Figure 2.1- Display unit of the bird identifier and it's allocated window segments.

2.4.2- KEYBOARD

An early model of the bird identifier, developed in the department, provided the experience to crystallize other desired functions and a more improved system behavior. In this model a single key was dedicated to each category of features. Our experience from the use of this model showed that the mere abundance of keys, presented a barrier to the user as it was introducing an element of confusion. Furthermore, the growth of the number of attribute categories demanded that more keys be utilized, and therefore served to increase the cost of the device. Based on this experience it was decided to abort such an arrangement in the present model. Instead, a single Select key would serve to access these attribute clusters, through a

hierarchy of menus. A top level menu would support a list of the different available categories of attributes, where the Select key acts to activate the chosen category. In the same way that the Select key will transfer the system's activity down the hierarchy of menus/activities, an Escape key will transfer activity by ascending the hierarchy chain.

In Section 2.3 we identified the find and attribute modes as the two major areas of activity. All other functions may be classified as belonging to one of these two modes. In this regard the identifier may be viewed as performing two parallel but dependent tasks (modes). In the effort of creating an efficient and user-friendly means of switching between these modes, a Mode key was used. The function of the key is to toggle system activity from one mode to the other whilst conserving the context of the suspended activity. This approach will avoid the inherent impediment associated with unwinding nested menus. Thus, the member of the find or attribute mode. Functions associated with each mode are related through a hierarchy structure where the Select and Escape keys are the means of access. The Help key is the last member of the function keys, and provides the relevant explanation as to the behavior of the active function. Finally four keys are dedicated to navigational purposes.

2.4.2.1- NAVIGATION KEYS

NORTH(^): a) Scrolls up through the menu items. At the first entry, the next depression will wrap around the list to display the last entry.
b) Increments the bird count, when in Bird Count menu.

SOUTH(v): a) Scrolls down through the menu items. At the last entry, the next depression will wrap around the list to display the first entry.
b) Decrements the bird count, when in the bird Count menu.

EAST(>): Scrolls a word to the right when displaying texts longer than the display width.

WEST(<): Scrolls a word to the left when displaying texts longer than the display width.

2.4.2.2- FUNCTION KEYS

SELECT: a) Provides access to subordinate menus.
b) Toggles an attribute.
c) Produces bird call, when in Bird Call menu
d) Produces bird information, when in Bird Information menu.
e) Zoom in, when in Bird Image menu.

ESCAPE: a) Provides access to a higher level menu.
b) Aborts an action.
c) Zoom out, when in Bird Image menu.

HELP: Provides context sensitive help.

MODE: (A/F) Provides a means of context switching between the attribute selection and listing of candidate bird species.

Notice that a power key is of no significance to the purposes of the software model, and has therefore been avoided. In fact it was envisaged that the actual device could automatically switch to a low-power standby mode following a predefined lapse of keyboard inactivity. it was decided to include a power-off key so that the context saving of the last active process could be tested upon re-entering the program.

2.5- SUMMARY

The major resolutions of this chapter are listed below:-

* An Objetc-Centered approach will used for the design of the bird identifier.
* Cost implications and MMI considerations determine the MMI choice.

Figure 2.2 shows the physical appearance of the identifier. Naturally, as far as the personal computer model is concerned the LCD and the keyboard will be emulated.

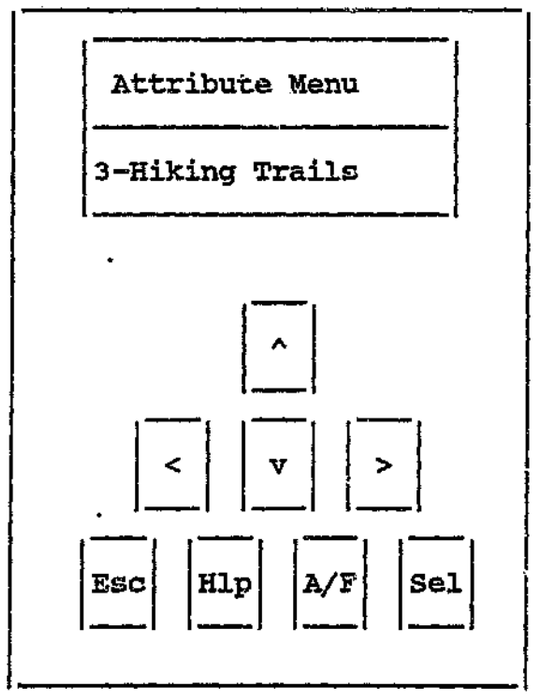


Figure 2.2- Depicted figure of the hand-held bird identifier

CHAPTER 3- DETAILED BEHAVIORAL DESCRIPTIONS

3.1- INTRODUCTION

To complete the behavioral specifications of Section 2.3, in this cahpter we will describe the detailed behavior of all system activities. Key behaviors in the context of each activity and in relation to neighbouring activities are fully described. finally, these descriptions are brought together to exhibit an overview of the system behavior.

3.2- ATTRIBUTES MAIN MENU

Attribute Main Menu
1 - Countries
2 - National Parks
3 - Hiking Trails
4 - Towns and Cities
5 - Bird Sanctuaries
6 - Habitats
7 - Status
8 - Bird Families
9 - Size
10 - Feeding Habits
11 - Features
12 - Coloring
13 - Markings
14 - Utilities

Figure 3.1- Attribute Main Menu

The attribute main menu supports categories of selectable features, in addition to the utilities menu. Features 1 to 5 form the long term attributes, since they would normally remain unchanged during a bird finding exercise, whilst features 6 to

13 are short termed attributes, in that rapid changes in these features is necessary to bring about final identification.

The utilities menu is a means of defining device features that may be specified by the user, such as response times of the key depressions.

The menu title is listed in window 2 of the display, and the current entry occupies window 4. Figure 3.2 shows a typical example of the display in the attribute menu, where the current menu item is item number 3 "Hiking Trails".

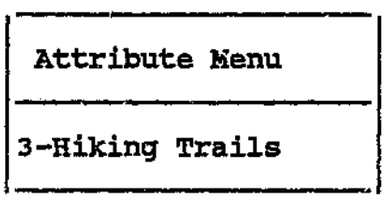


Figure 3.2- A typical display of the Attribute Main Menu

3.2.1- KEY BEHAVIOR IN ATTRIBUTE MAIN MENU

The Escape key is the only key that is not valid in this menu. The Select key will invoke the attribute sub menus.

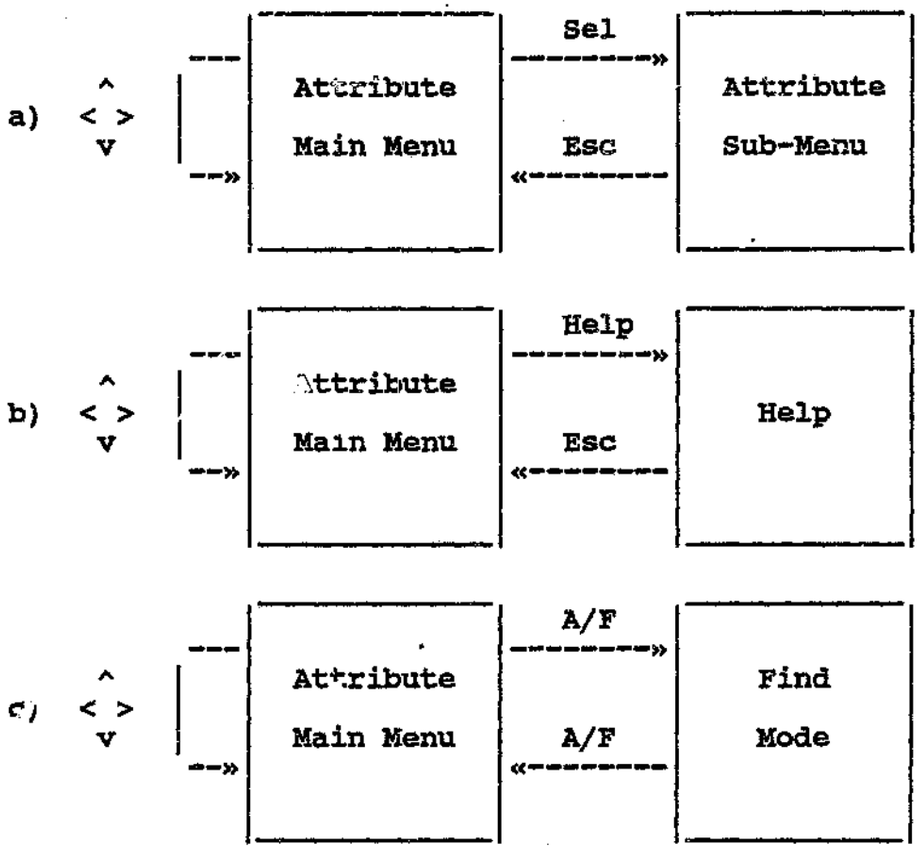


Figure 3.3- Full graphical description of the Attribute Main Menu.

a) Relationship of the Attribute Main Menu and it's sub-menus.
b) Relationship of the Attribute Main menu and it's help menu.
c) Mode switching with Attribute Main Menu as the currently active menu.

3.3- ATTRIBUTE SUB-MENUS

The countries sub-menu is shown below as an example. Other sub-menus are listed in Appendix A.

Attribute Sub-Menu (Countries)
1 - South Africa and National States 2 - Lesotho 3 - Swaziland 4 - Namibia 5 - Zimbabwe

Figure 3.4- Attribute Sub-menu of Countries

The display format of this menu is consistent with the main menu, in the sense that the title of the current menu will always occupy the top line on window 1, and the menu item will occupy the bottom line on window 4. This consistency is vital to the M.M.I as discussed in chapter 2; and thus it is upheld throughout the entire design. Figure 3.4 shows a typical display situation.

Countries
5-Zimbabwe

Figure 3.5- A typical display of the Attribute Sub-menu

3.3.1- KEY BEHAVIOR IN ATTRIBUTE SUB-MENUS

All keys are valid in this menu. The Select key is used to activate/deactivate features.

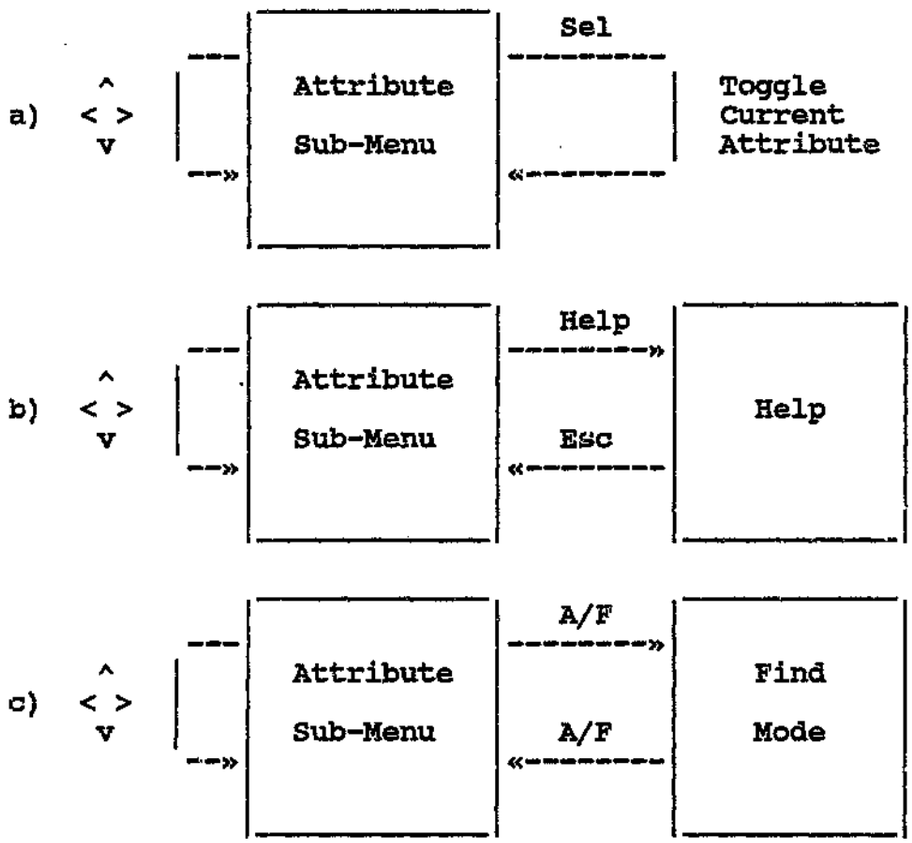


Figure 3.6- Full graphical description of the Attribute Sub-menus
a) Attribute selection/deselection
b) Relationship of the Attribute Sub-menu and it's help menu.
c) Mode switching with the Attribute Sub-menu as the currently active menu.

3.4- UTILITIES MENU

Utilities Menu
1 - Set initial key delay 2 - Set scroll rate 3 - Enable counting in List Found Birds 4 - Enable extraction in List Found Birds

Figure 3.7- Utilities Menu

Initially, this menu was designed to provide the user with a facility to adjust the keyboard response time and scroll rate of the navigation keys (items 1 and 2). The experience of an early version of the identifier proved that it was convenient to provide the two functions of bird counting and extraction within the List Found Birds. The latter function acts to drop an entry from the list of found birds.

Due to the limited keyboard, however, it was decided to employ the utilities menu to define which of the two functions would be available to the List Found Birds menu (items 3 and 4). Functions represented by items 1 and 2 do not have bearing on the purpose of this work and thus they were not implemented in the software model.

3.4.1- KEY BEHAVIOR IN UTILITIES MENU

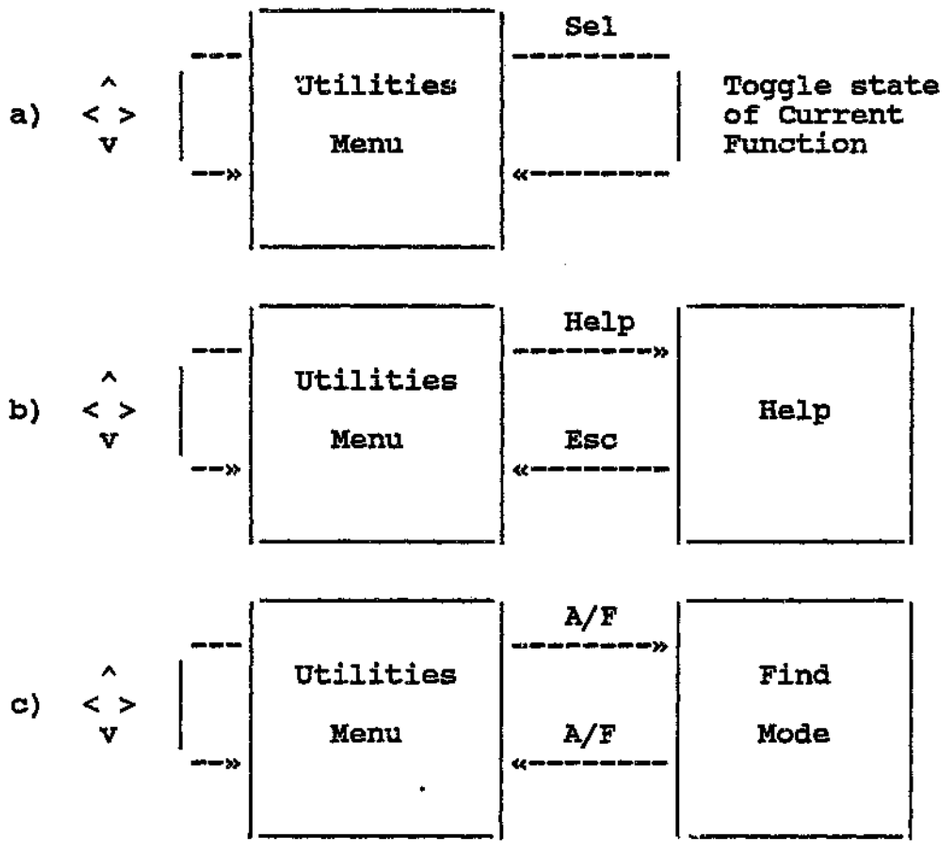


Figure 3.8- Full graphical description of the utilities menu.

a) Enable/disable function
b) Relationship of the utilities menu and it's help menu.
c) Mode switching with utilites as the currently active menu.

3.5- FIND MAIN MENU

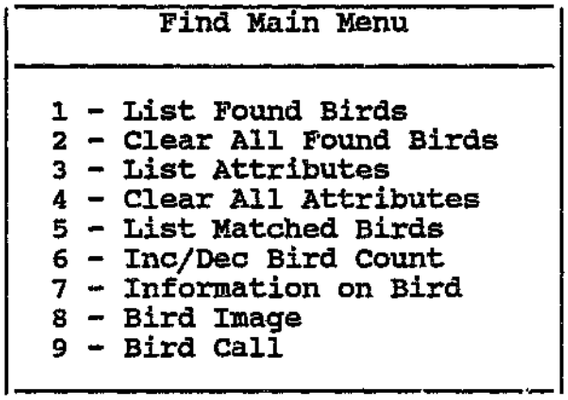


Figure 3.9- Find Main Menu

Items 1 to 5 represent sub-menus, while items 6 to 9 are selectable items to define the type of function that can be accessed when in List Matched Birds menu. In other words, When viewing the list of candidates in List Matched Birds menu, one of four functions can be performed on the current listed bird;

1- Edit the bird count.
2- Obtain more information on bird.
3- View the image of the current listed bird.
4- Generate the call of the current listed bird.

Only one of these functions can be selected at any one time. Here is an example of the display format of this menu. Notice the text indicator of window 3, denoting that more text lies to the right and may be accessed through the right arrow key (East key) depressions.

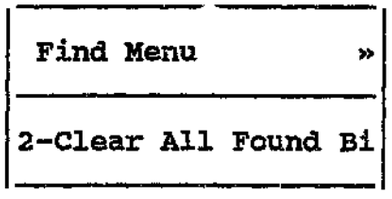


Figure 3.10- A typical display of the Find Menu

3.5.1- KEY BEHAVIOR IN FIND MAIN MENU

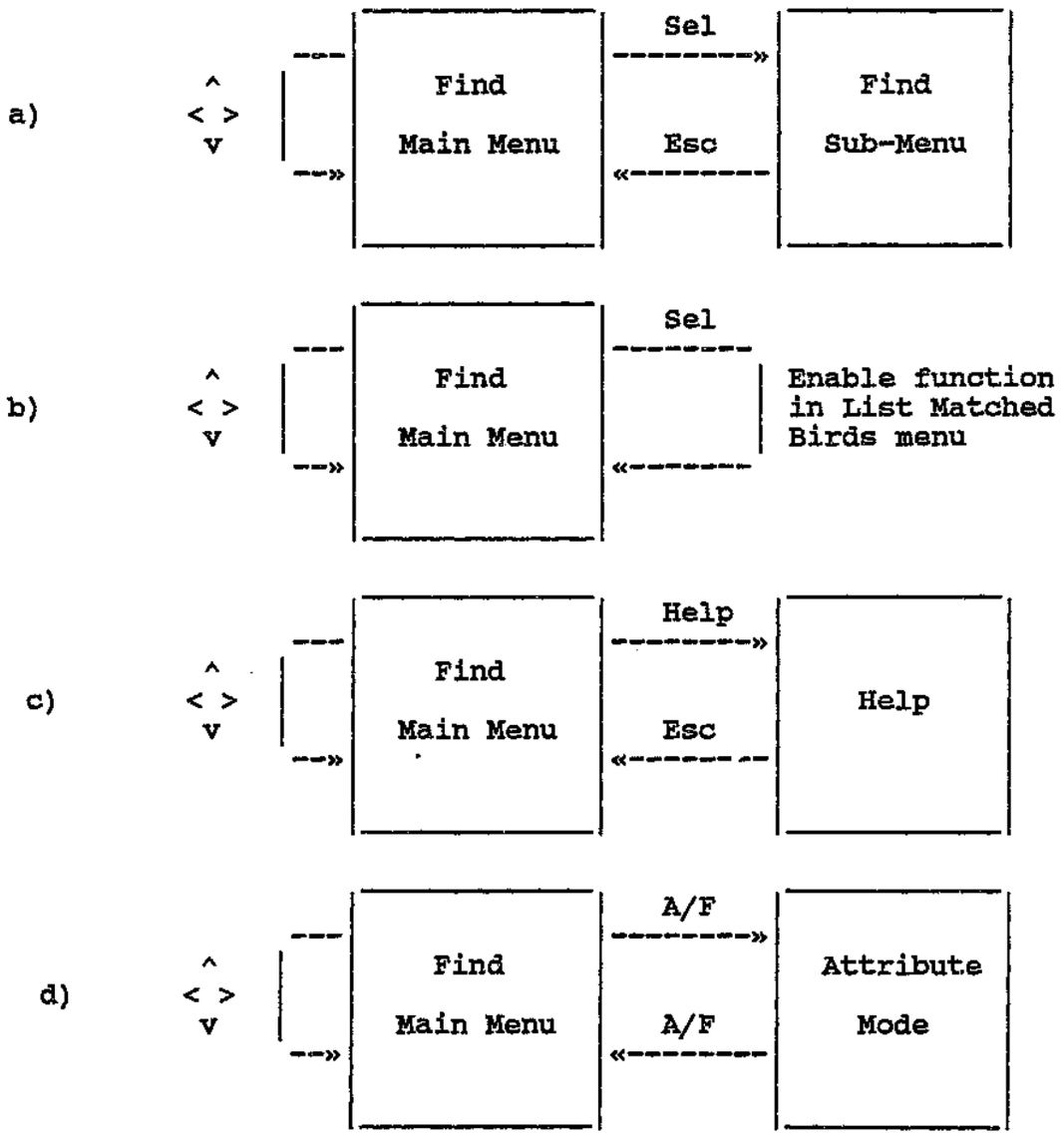


Figure 3.11- Full graphical description of the Find Main Menu

a) Relationship of the Find Main Menu and it's sub-menus; relevant to items 1 to 5.
b) Selection feature of items 6 to 9.
c) Relationship of the Find Main Menu and it's help menu.
d) Mode switching with the Find Main Menu as the currently active menu.

3.6- FIND SUB-MENUS

3.6.1- LIST FOUND BIRDS

This function is a very important facility of the identifier, as it is the mechanism that keeps track of the list of cited birds. This is done by attaching a non-zero count value to a bird species in the List Matched Birds menu. The List Matched Birds produces a list of birds the match the sellected attributes, and is therefore a dynamic list, since the sellected attributes keep changing from one session to the next. In order not to lose the bird of interest through modifying the sellected attributes, a bird count serves to tag the bird. All tagged birds will then be avaialable in List Found Birds.

Each entry in the list is identified by a family name, and a species name. Only one entry is displayed at any one time. The bird family is displayed at the top line in window 2 and the bird name at the bottom line in window 4. Here is an example of the display format of this menu.

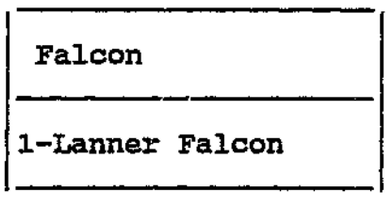


Figure 3.12- A typical display of List Found Birds

3.6.1.1- KEY BEHAVIOR IN LIST FOUND BIRDS

The utilities menu specifies which of the two fuction are accessable in this menu.

A) Count Bird: Count value is changed using north/south keys.

B) Extract Entry: The Select key extracts the current entry from the list following final confirmation.

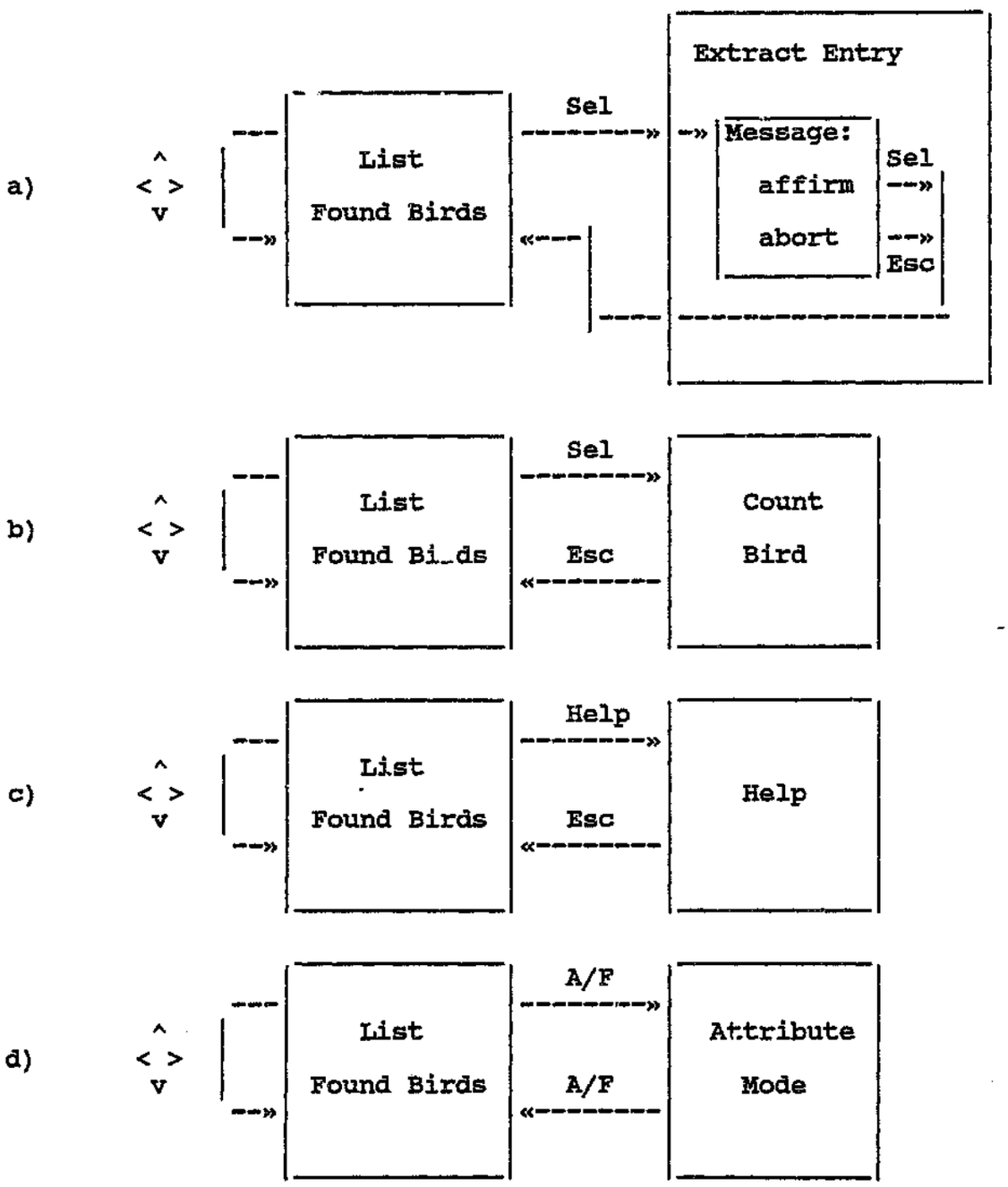


Figure 3.13- Full graphical description of List Found Birds

a) Relationship of List Found Birds and Extract Entry
b) Relationship of List Found Birds and Count Birds.
c) Relationship of List Found Birds and it's help menu.
d) Mode switching with List Found Birds as the currently active menu.

3.6.2- CLEAR ALL FOUND BIRDS

This function clears the list of found birds following final confirmation. Once clearing has been initiated by the Select key, the next Select will confirm this operation while the Escape key will abort the clearing operation. Throughout this design where a clearing operation of any kind is involved, both the key behavior and the display format of the clearing operation are consistently copied. In this regard the key behavior and the display form a generic module, accessible to other clearing operations.

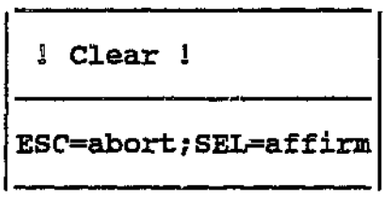


Figure 3.14- A typical display of Clear All Found Birds

3.6.2.1- KEY BEHAVIOR IN CLEAR ALL FOUND BIRDS

Select confirms the clearing operation and Escape aborts the operation.

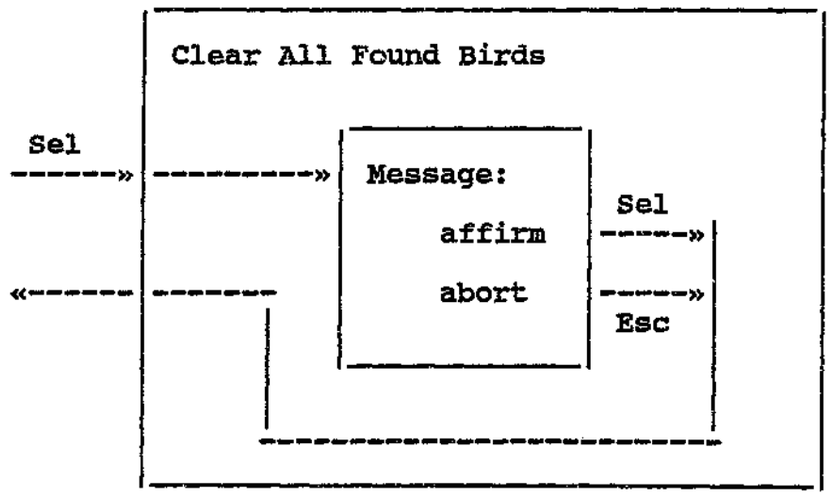


Figure 3.15- Graphical description of Clear All Found Birds

3.6.3- LIST ATTRIBUTES

A list of the selected attributes are provided by this function. This alleviates the need for a manual inspection of each of the attribute sub-menus in order to confirm selected attributes. Here is an example of this menu, where it is assumed that item 2 of the features menu is the current item.

Features
2-Long beak

Figure 3.16- A typical display of the List Attributes

3.6.3.1- KEY BEHAVIOR IN LIST ATTRIBUTES

The Select key clears the current attribute pending a final confirmation. Once clearing has been initiated by the Select key, the next Select will confirm this operation while the Escape key will abort the clearing operation. The clearing operation is consistent with previous clearing operations.

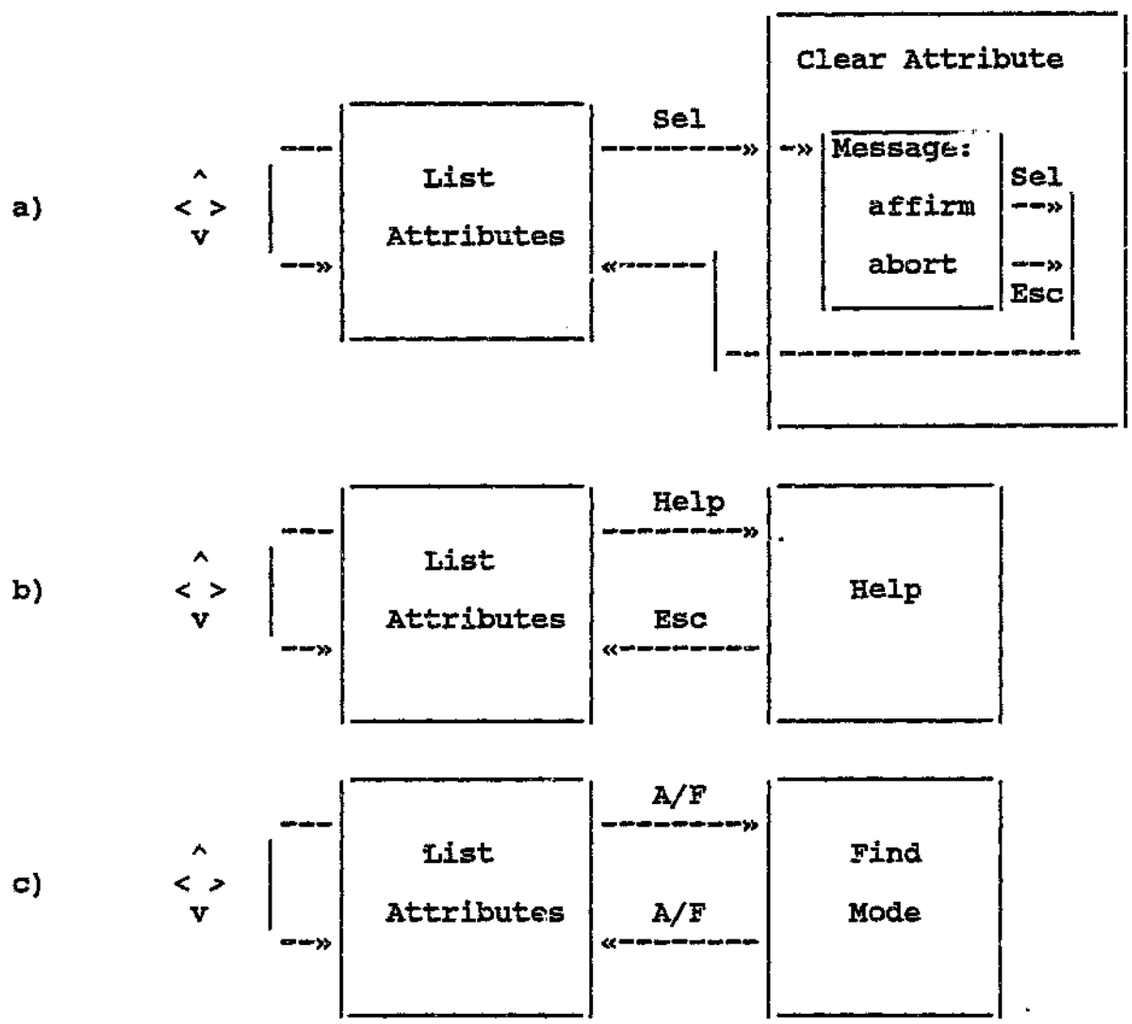


Figure 3.17- Full graphical description of List Attributes

a) Relationship of List Attributes and Clear Attribute.
b) Relationship of List Attributes and it's help menu.
c) Mode switching with List Attributes currently active.

3.6.4- CLEAR ALL ATTRIBUTES

In order to set out on a fresh identification exercise it is necessary that all attribute descriptions of the last session be removed so that new selections can be made as quickly as possible. This operation clears all the selected attributes after final confirmation. The display is that of the generic form shown in Figure 3.12.

3.6.4.1- KEY BEHAVIOR IN CLEAR ALL ATTRIBUTES

The Select key confirms the clearing operation, while the Escape aborts the operation.

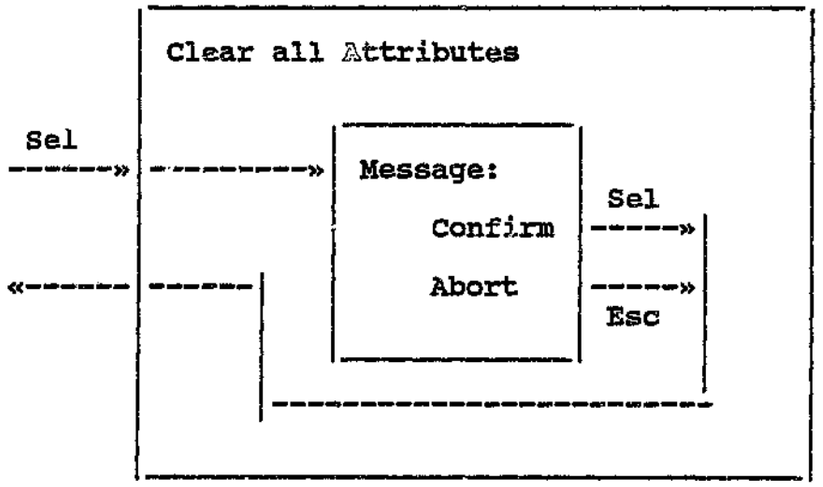


Figure 3.18- Graphical description of Clear All Attributes

3.6.5- LIST MATCHED BIRDS

A list of bird species matching the selected attributes is furnished by this menu. The format of this list is consistent with that of found birds. That is the bird family will appear at the top line of display and the species name at the bottom line. Here is an example of the display format.

Vulture
1-Cape Vulture

Figure 3.19- A typical display of List Matched Birds

3.6.5.1- KEY BEHAVIOR IN LIST MATCHED BIRDS

The second level of this menu can be one of four possibilities,

a) Increment/decrement bird count.

b) Information on bird.

c) Bird image.

d) Bird call.

Items 6 to 9 of the Find Main Menu specify the accessible function in this menu.

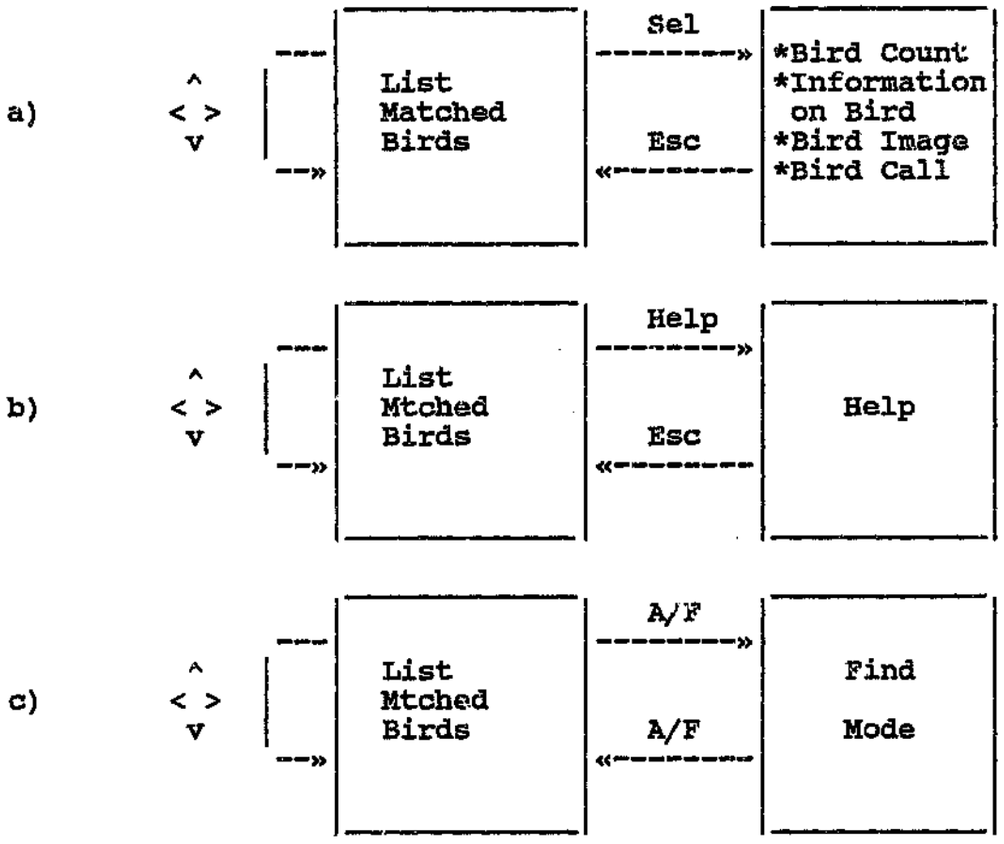


Figure 3.20- Full graphical description of List Matched Birds
a) Relationship of List Matched Birds and it's sub-ordinate menu.
b) Relationship of List Matched Birds and it's help Menu.
c) Mode switching with List Matched Birds currently active.

3.6.6- BIRD COUNT

This menu provides a counting facility, to increment/decrement the count of the selected bird. All bird species having a count value will then become available in List Found Birds.

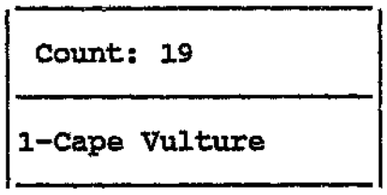


Figure 3.21- A typical display of Bird Count Menu

3.6.6.1- KEY BEHAVIOR IN BIRD COUNT

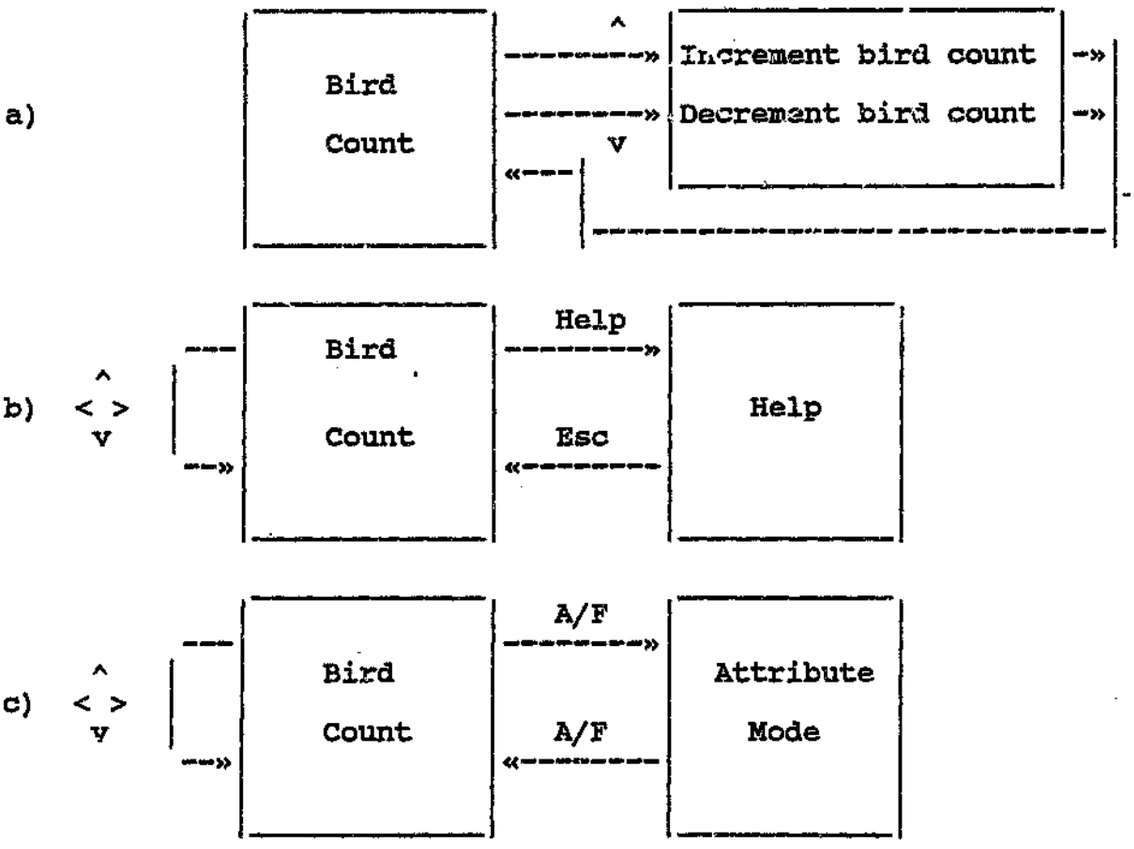


Figure 3.22- Full description of the Bird Count Menu
a) Increment and decrement facility through the North and South keys.
b) Rlationship of Bird Count and it's help menu
c) Mode switching with Bird Count currently active.

3.6.7- INFORMATION ON BIRD

Information on the current bird is displayed on window 2 of the display. This information may be several lines of text, but access to this text is made through the use of navigation keys. Here is an example of the display format of this menu. The content of text information is "Highly versatile bird". Notice the colon seperator between window 1 and 5, and the text on the right indicator in window 3.

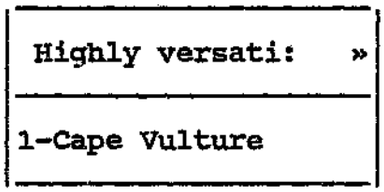


Figure 3.23- A typical display of the Information on Bird Menu

3.6.7.1- KEY BEHAVIOR IN BIRD INFORMATION

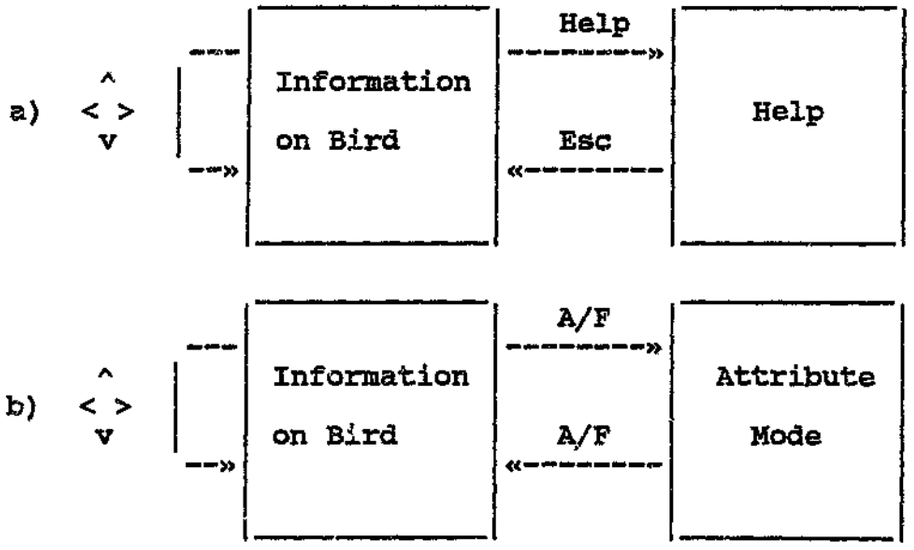


Figure 3.24- Full graphical description of Information on Bird
a) Relationship of Information on Bird and it's help menu.
b) Mode swithcing with Information on Bird currently active.

3.6.8- BIRD IMAGE

This menu is different from all other menus in two respects. Firstly, the display will serve to support the image of the of the bird instead textual descriptions. Secondly, the navigation keys take the role of panning across the full image in the four directions of North, South, East and West.

3.6.8.1- KEY BEHAVIOR IN [illegible]

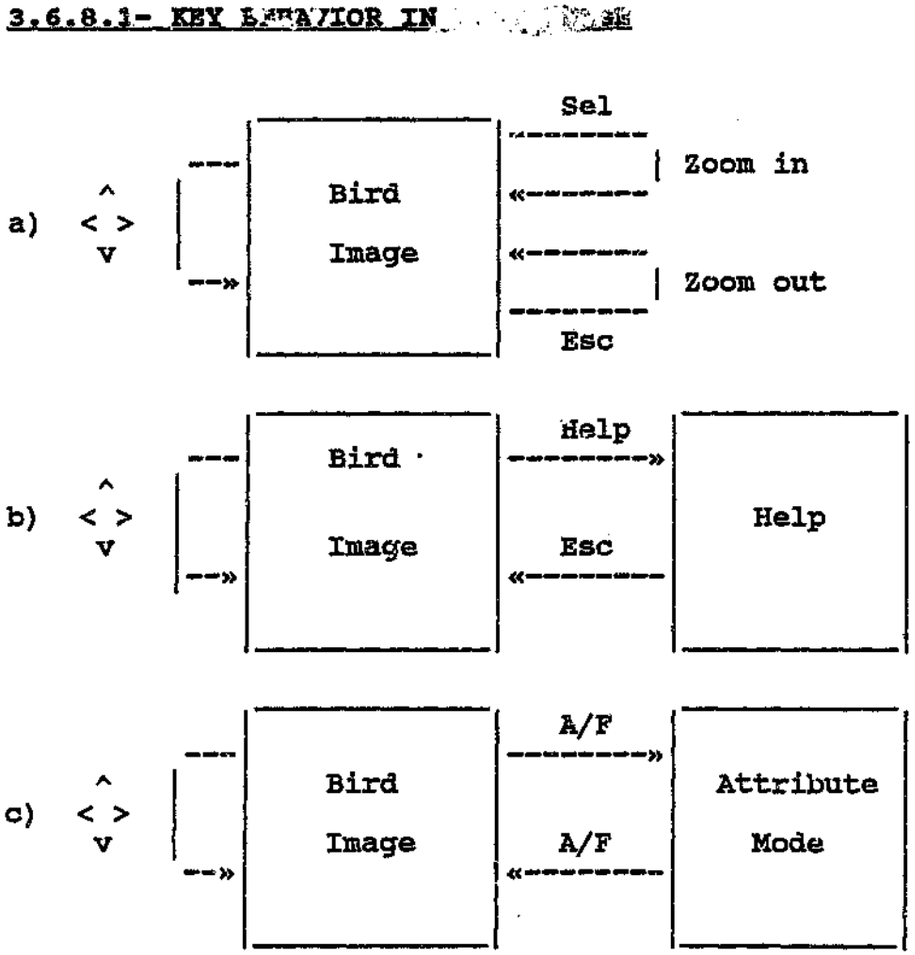


Figure 3.25- Full graphical description of Bird Count Menu
a) The zooming of the bird image.
b) Relationship of Bird Image and It's help menu.
c) Mode swithcing with Bird Image currently active.

3.6.9- BIRD CALL

In fact there is no definition of key behavior in this menu. Bird Call is really a function rather than a menu and therefore operates in a purely passive manner. This function is invoked from the List Matched Birds menu.

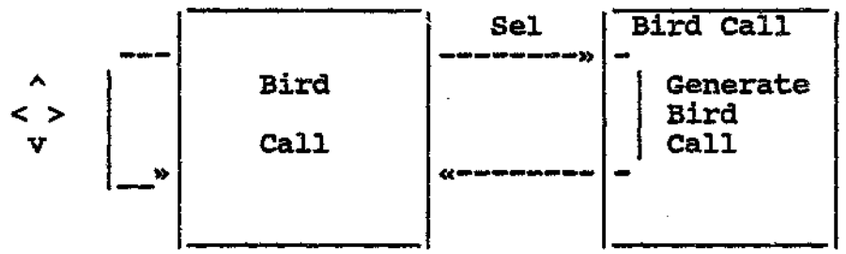


Figure 3.26- The graphical description of Bird Call

3.7- HELP MENU

This menu may be accessed at all levels of the system. The content of help depends on the context in which help has been requested.

3.7.1- KEY BEHAVIOR IN HELP MENU

The system consistency should render minimum help to be adequate. the help text is therefor limited to a single line, rendering the North/South keys redundant.

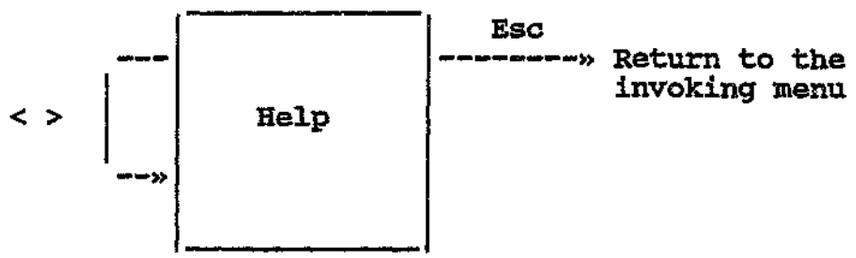


Figure 3.27- Graphical description of the Help Menu

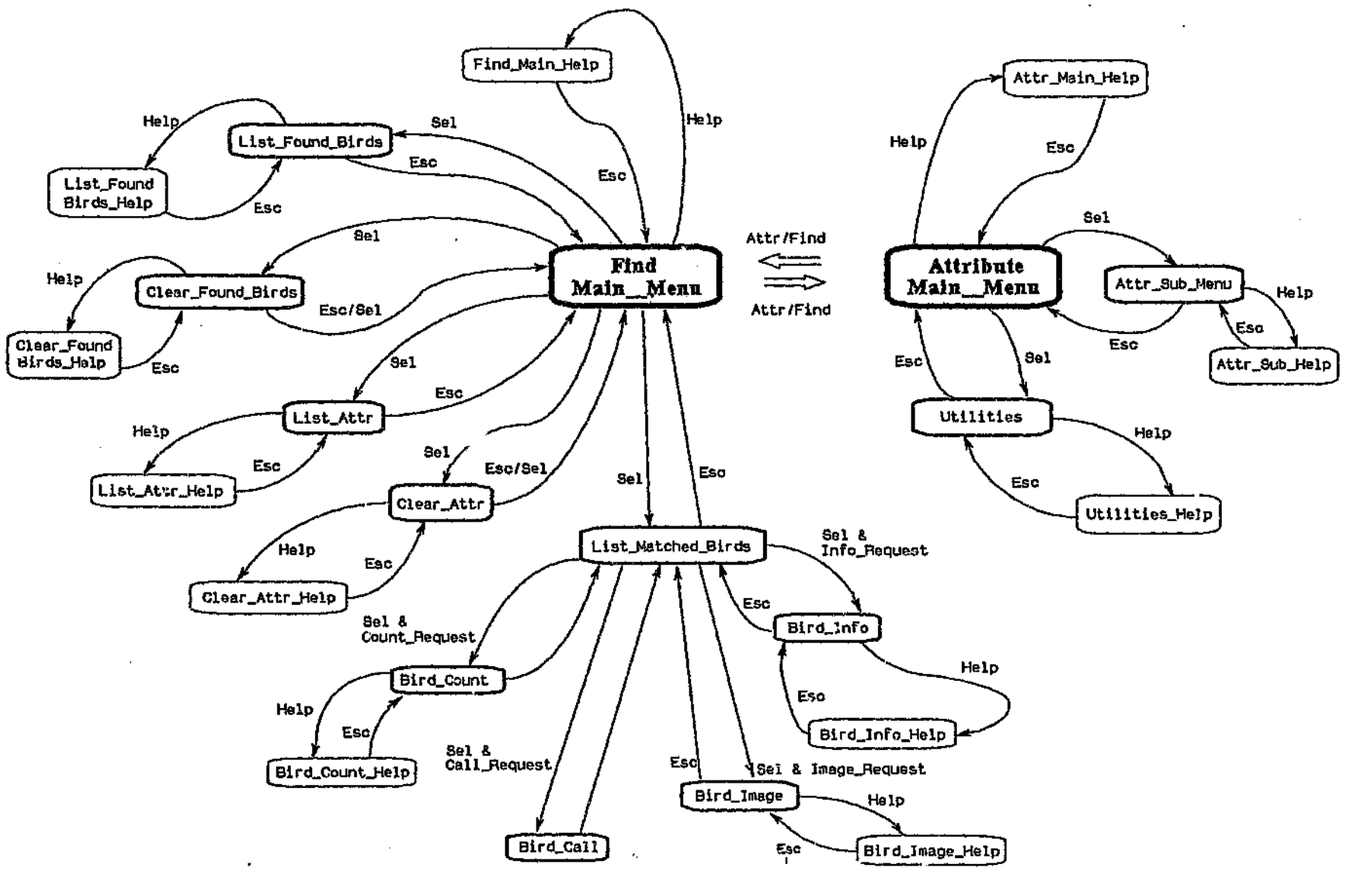
3.9- OVERVIEW OF THE BIRD IDENTIFIER FUNCTIONS

The integration of all system activities is shown by the state transition diagram of Figure 3.26. This diagram separates the activities associated with the two operating modes of the identifier. It is important to realize that transferring of system activity from the Attribute mode to the Find mode and vice versa, is possible from any state with the exception of help states.

In this chapter we have completely defined the functional behavior of all system activities. The computer game concepts have strongly been adhered to, in order to accommodate the important issues of Section 2. These issues are listed below in the same order.

* The keyboard has been reduced to a minimum size, thus making for a simple to use and cost effective device.
* Considering the limitations of the display, a reasonable global vision has been has been provided by defining multiple display windows
* Each activity is represented by a suitable title, which is unique to its environment. This in addition to the presence of clear messages serve to convey a clear status of the system.
* The consistency of the clearing functions and it's associated keyboard responses ensure system consistency, which is helpful to the user.
* The simplicity of the MMI and presence of context sensitive help makes lengthy documentation unnecessary.

State Transitions



CHAPTER 4- ARCHITECTURAL DESIGN

4.1- INTRODUCTION

The goal of this chapter is to apply our OOD architectural choice to the bird identifier model. A powerful way of implementing the OOD concept is through the process/resource classification of objects. After discussing the mechanics of process/resource method we will then apply the method to define the bird identifier system. In a systematic approach to this task, process/resource decomposition is carried-out at several levels of abstraction.

4.2- PROCESS/RESOURCE IMPLEMENTATION OF OBJECT-ORIENTED DESIGN

A process is formally defined as an actor object, and is responsible for management of the flow of data to and from resources. The access of data is furnished through well-defined operations supported by resources as an interface to the outside world. A resource is formally defined as a server object, responsible for the management of data in data aggregates[8].

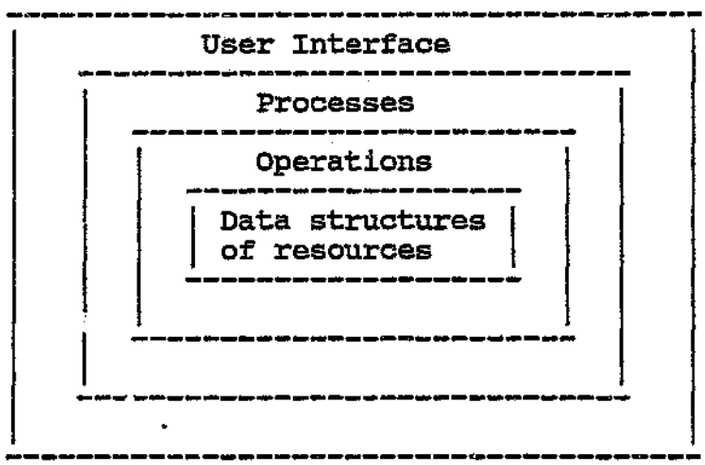


Figure 4.1- The levels of scope from the outside world to the heart of a process/resource system

The process/resource method is basically an object-oriented system design approach in that processes and resources may be considered as objects in their own right. The process/resource strategy classifies objects on the bases of processes and resources, rather than more routine OOD method of identifying actor objects and server objects [7]. Nevertheless, these two methods are essentially equivalent. The former method, however, is more descriptive in that identification of process and

resource objects, serves to define the scope and environment of each object. This will result in a rapid identification of the major objects of the system. The shifting of attention from the outside view of the system to the inside view of major objects, and subsequent identification of subclasses of objects will follow naturally. We will now proceed to identify the objects (processes and resources) of the the system at the highest level of abstraction. But before embarking upon this task we must discuss the importantance of isolating the conceptual stage of a design from it's implementation method.

4.3- AWARENESS OF DESIGN LEVEL AND DOCUMENTATION

It is not difficult to fall into the trap of combining 'what' is to be done with 'how' it is done. That is to say the conceptual stage is influenced by the implementation stage. An example in the context of programming is where the designer's thinking during the conceptual stage is conditioned by the employed implementation language. This confusion of interests can have undesirable consequences, the most serious of which is the impact on design documentation. It is well understood that design documentation goes a long way in providing for software management and growth[9]. In fact, Schneidewind [10] states that software should be regarded as a dynamic entity inasmuch as it undergoes change throughout its life, and there is no reason to distinguish between maintenance and initial development. But without good documentation it becomes impossible to manage growth and complexity.

One way of achieving good documentation is through the use of Program Description Language (PDL) [11]. PDL is a conceptual design language that uses the vocabulary of the english language in conjunction with syntax of a structured programming language to form a highly understandable and structured design tool. In de-coupling the concept (the "what") from the implementation (the "how"), the PDL tool can be used to focus efforts on producing sound conceptual designs prior to the implementation stage.

In this design we have employed a PDL[12] tool to concentrate on the architectural and conceptual issues without being affected by implementation issues. This PDL description will also form the backbone of our design documentation. Appendix B contains a full PDL description of the bird identifier model. Due it's powerful descriptive power, we will use a PDL syntax to describe the functionality of the system processes, at different levels of design.

4.4- FIRST-LEVEL DESIGN

At this level of abstraction the entire system is viewed as comprising of several classes of objects (high-level objects), where each class denotes a set of similar but unique objects. A class serves to factor the common properties of a set of objects. Here the interface, menu database and the bird database are identified as the major resources, or classes of objects, of the system. The bird identifier is defined as process object which employs these resources to manage the system data.

In the following demographics we will represent resource objects with oval shapes and the process objects with the rectangular shapes.

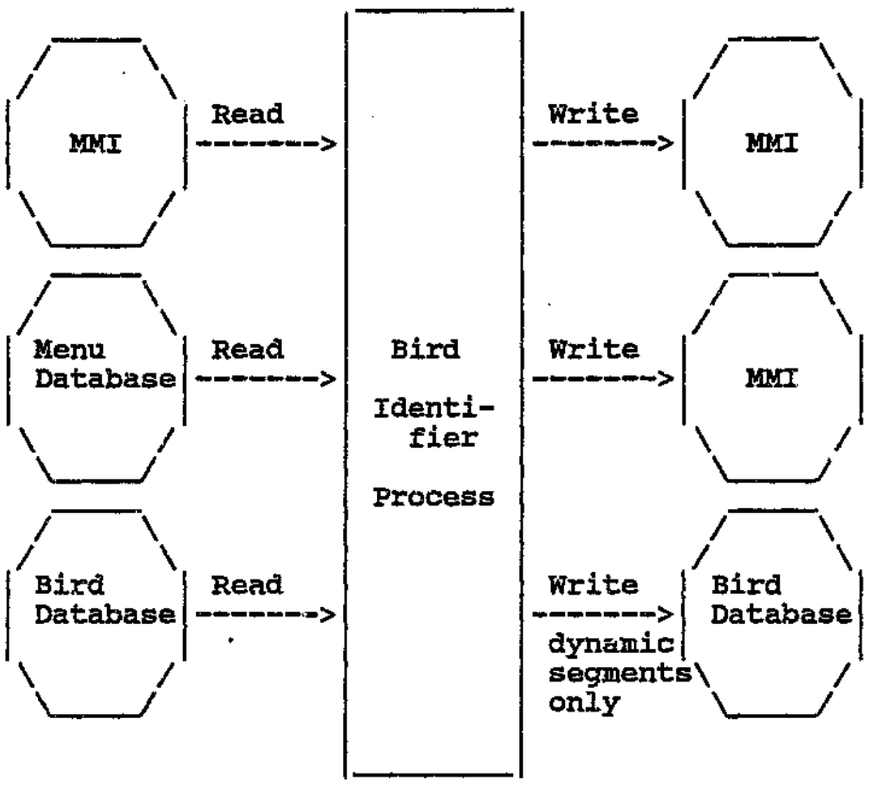


Figure 4.2- First-level description; bird identifier process

PDL description of the bird identifier process:-

```
Repeat:
  Begin:
    * Sense the MMI for user commands *
    * Read menu database or bird database to process command *
    * Use the MMI to feedback results to the user *
    * Save dynamic data to bird database *
  End:
```

4.5- SECOND-LEVEL DESIGN

Second-level process and resource decompositions are shown below

Resources:-

Man-Machine Interface (MMI):
  Keypad
  Screen
  Bird call output

Menu database:
  Attribute mode menus
  Find mode menus

Processes:-

Bird identifier process:
  Attribute process
  Find process

4.5.1- THE ATTRIBUTE PROCESS

This level of abstraction serves to discriminate between the two major activities of the system, namely that of the Attribute Mode and the Find Mode. In Figure 4.3 the Attribute Mode is represented by the attribute process, and resources of keypad, screen, attribute menus and selected attributes which provide the necessary data to this process. Notice that the the keypad and screen objects are sub-classes of the MMI class, while the attribute menu resource is a sub-class of the menu database.

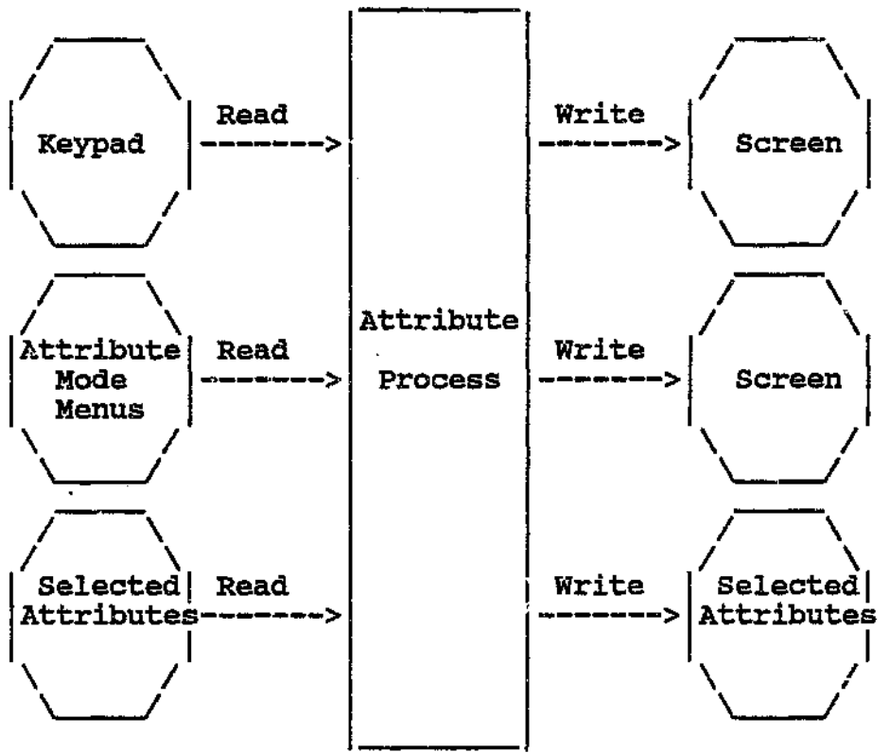


Figure 4.3- Second-level description: the attribute process

PDL description of the attribute process:-

```
Begin:
  Repeat:
    * Access the menu and write the current item to screen *
    * Sense the keyboard for user commands *
    Case * Request of *

      Scroll: * Scroll up and down within the current menu or
                right and left within the current item *

      Menu level: * Move up or down the menu hierarcy *

      Attribute toggle: * Toggle the attribute of the current
                          item *

      Mode switching: * Terminate process *

      Help: * Provide relevant help information *
    End: Case:
  Until: * Process is terminated *
End:
```

4.5.2- THE FIND PROCESS

This description is the counter-part of the previous Section, but occupies the same level of abstraction. Figure 4.4 shows the find process surrounded by a series of resources relevant to the Find Mode. Here the find menus resource is a sub-class of the menu databse, and the bird call output is a sub-class of the MMI resource. Furthermore, the bird database resorce has been decomposed into five sub-classes of objetcs.

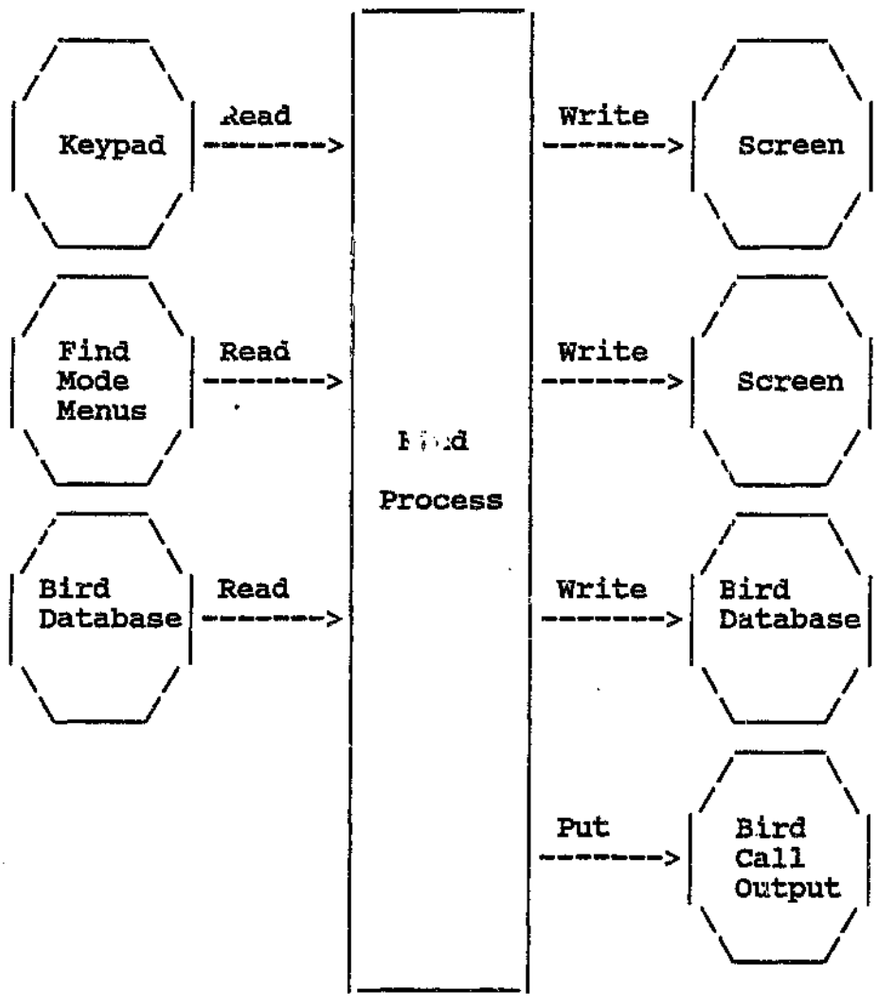


Figure 4.4- Second-level description; the find process

PDL description of the find process:-

```
Begin:
  Repeat:
    * Access the menu and write the current item to screen *
    * Sense the keyboard for user commands *
    Case * Request of *

      Scroll: * Scroll up and down within the current menu or
                right and left within the current item *

      Menu level: * Move up or down the menu hierarcy *

      Mode switching: * Terminate process *

      Help: * Provide relevant help information *

  End: Case:
  Until: * Process is terminated *
End:
```

4.6- THIRD-LEVEL DESIGN

Having identified the major objects of the system, we will now proceed to shift our attention to the inside view of these objects.

Third-level process and resource decompositions are shown below

Resources:-

Man-Machine Interface(MMI):
  Keypad
  Screen
  Bird call output

Menu database:
  Attribute mode menus:
    Selected attributes
  Find mode menus

Bird database:
  Possesed attributes
  Bird names
  Bird information
  Bird [illegible]unt
  Bird [illegible]ll

Process queue

Processes:-

Bird identifier process:
  Scheduler process:
    Help process
    Attribute process
    Find proces:
      Find main menu process
      List Found Birds process
      Clear Found Birds process
      List Attributes process
      Clear Attributes process
      Count Birds process

Since the activities of systems are centered around the data that they support, it seems logical to start by defining the system resources and their operations. However, for the sake of continuity, resources will be dealt with after describing the system processes

4.6.1- SYSTEM PROCESSES

4.6.1.1- THE SCHEDULER PROCESS

The purpose of this object is to schedule the system activities. Since our implementation is based on a flat hierarchy, the nested menus of the system (see Figure 3.26) will be achieved by means of a queuing operation. In fact all actvities are scheduled by dequeing from the queue, where the ready processes await execution. The queue structure will be supported by a queue resource as shown in Figure 4.6. The exact relationship of the nested menus are defined within the disc file containing the menu texts.

The dual mode operation of the identifier was stated earlier as the Find and Attribute mode. The swithcing between the modes is the function of the Mode key (A/F key), and is achieved through the EXCHANGE-Q-BLOCKS operation of the process queue resource, where activities related to the find and attributes are kept in different queues.

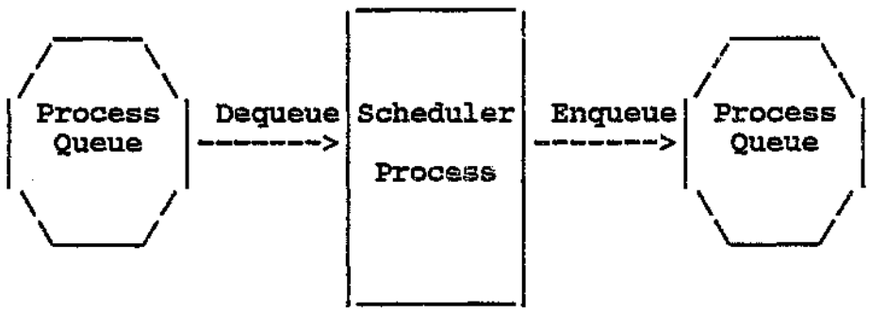


Figure 4.5 Third-level description; the scheduler process

PDL description of the scheduler process:-

```
Begin:
  Repeat:
    * Dequeue a process from the queue *
    * Identify the waiting process *
    * Execute the process *
  Until: * System is turned off *
End:
```

4.6.1.2- THE HELP PROCESS

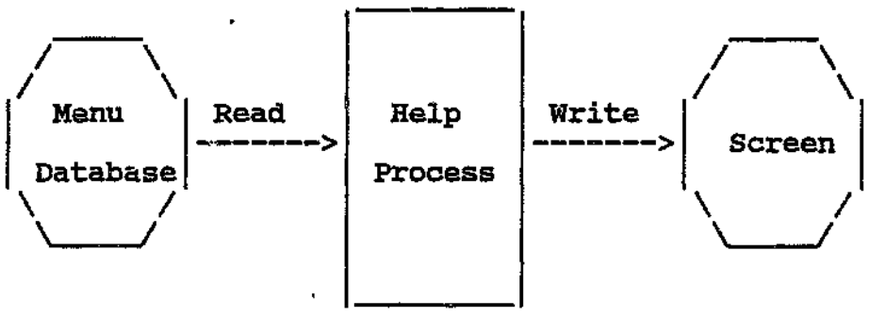


Figure 4.6 Third-level description; the help process

PDL description of the help process:-

```
Begin:
  Repeat:
    Case: * Request of *
      Scroll: * Scroll right and left within the help line *
      Exit help: * Terminate process *
    End: Case:
  Until * Process is terminated *
End:
```

4.6.1.3- FIND MAIN MENU PROCESS

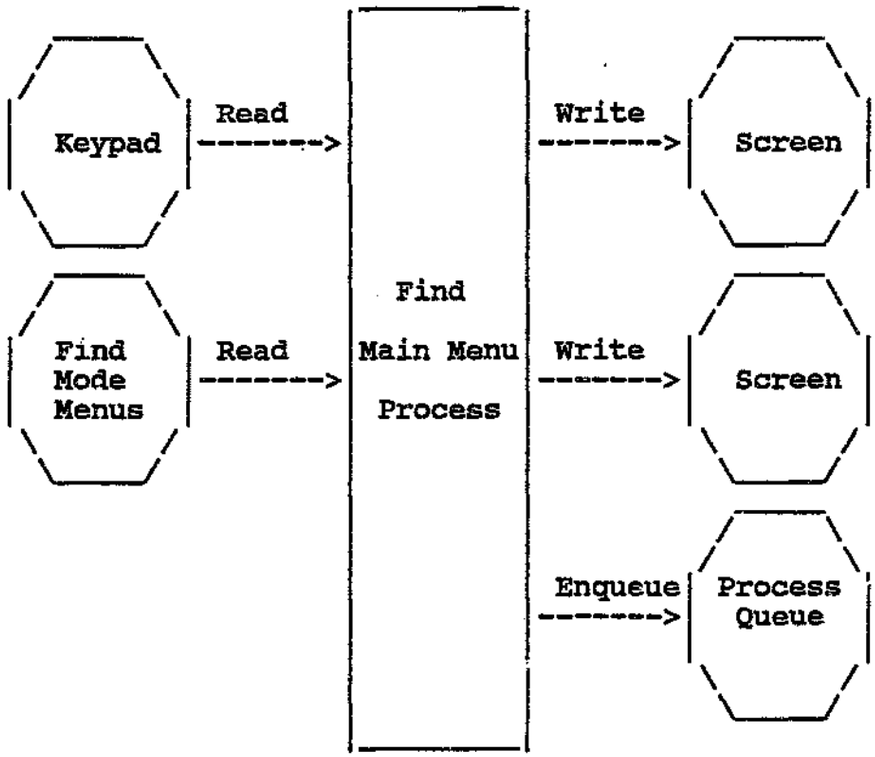


Figure 4.7 Third-level description: find main menu process

PDL description of find main menu process:-

```
Begin:
  Repeat:
    * Access the menu and write the current item to screen *
    * Sense the keyboard for user commands *
    Case * Request of *

      Scroll: * Scroll up and down within the current menu or
                right and left within the current item *

      Menu level: * Terminate process *

      Mode switching: * Put the current process in the queue *
                      * Terminate process *

      Help: * Provide relevant help information *

    End: Case:
  Until: * Process is terminated *
End:
```

4.6.1.4- LIST FOUND BIRDS PROCESS

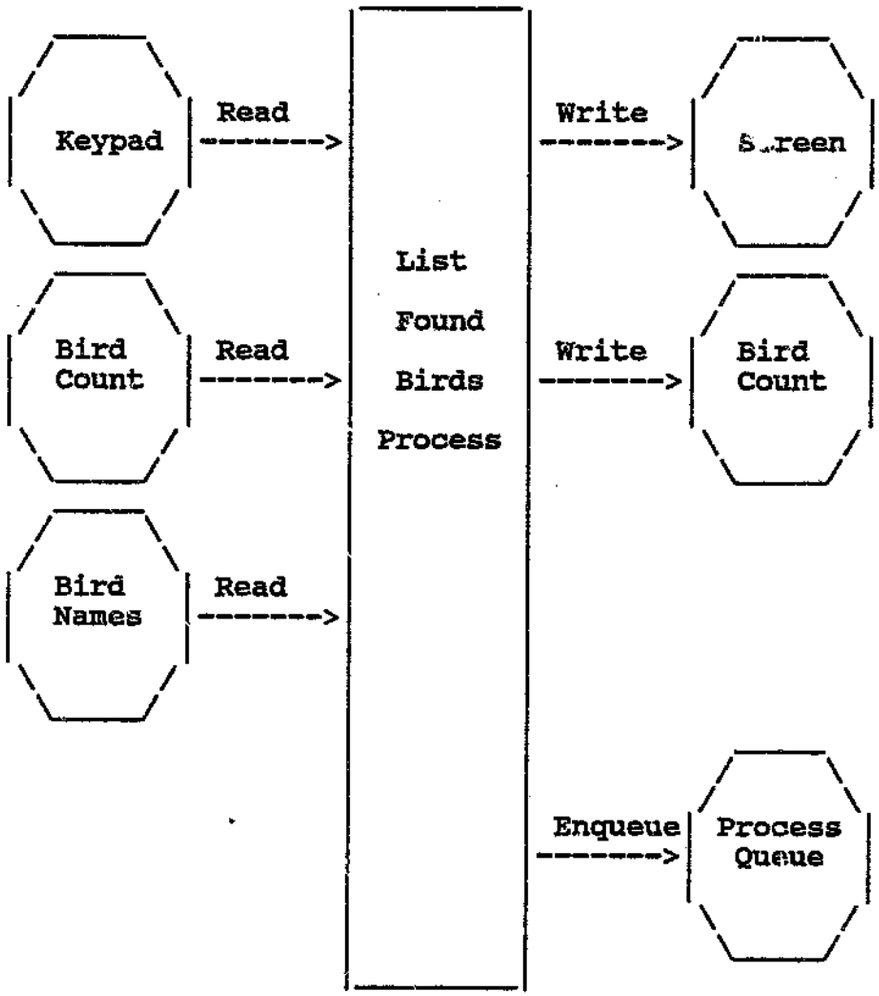


Figure 4.8 Third-level description: list found birds process

PDL description of list found birds process:-

```
Begin:
  Repeat:
    * Write number of found birds to window 5 *
    * Write bird name to window 4 *
    * Write count number of current bird to window 1 *
    * Sense the keyboard for user commands *
    Case * Request of *

      Scroll: * Scroll up and down within the list
                right and left within the current item *

      Change count: * Increment or decrement the count of the
                      current listed bird *

      Extract bird: * Extract the currently listed bird form
                      the list of found birds *

      Mode switching: * Put the current process in the queue *
                       * Terminate process *

      Terminate process: * Terminate process *

      Help: * Provide relevant help information *

    End: Case:
  Until: * Process is terminated *
End:
```

4.6.1.5- CLEAR FOUND BIRDS PROCESS

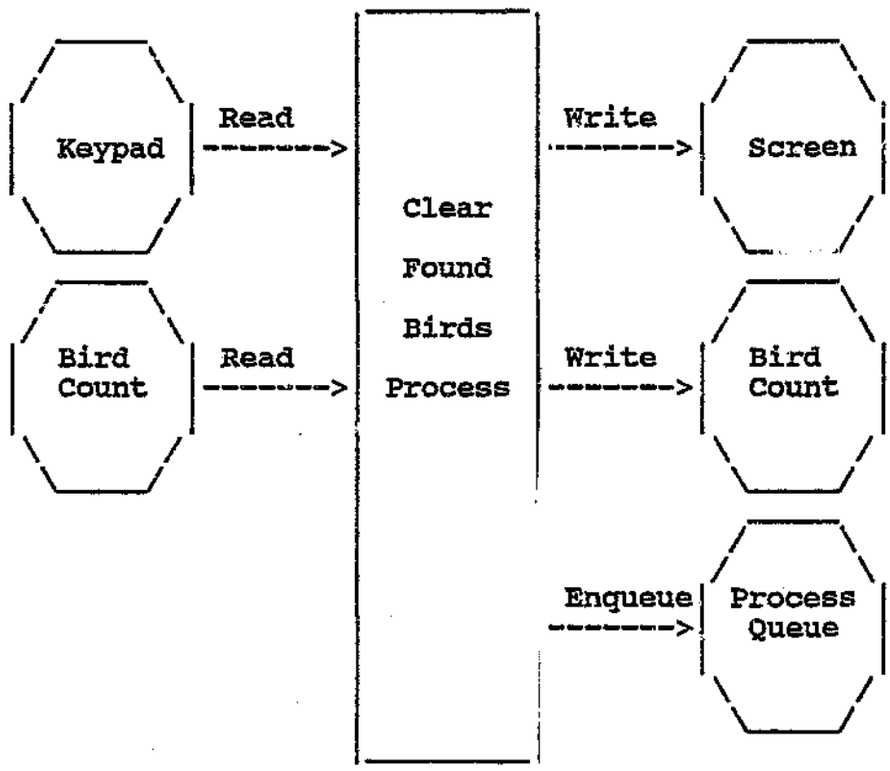


Figure 4.9 Third-level description: clear found birds process

PDL description of clear found birds process:-

```
Begin:
  Repeat
    * Sense the keyboard for user commands *
    Case * Request of *

      Confirm: * Clear the list of found birds *

      Abort: * Abort the clearing operation *

      Mode switching: * Put the current process in the queue *
                      *  Terminate process *

      Help: * Provide relevant help information *

    End: Case:
  Until: * Process is terminated *
End:
```

4.6.1.6- LIST ATTRIBUTES PROCESS

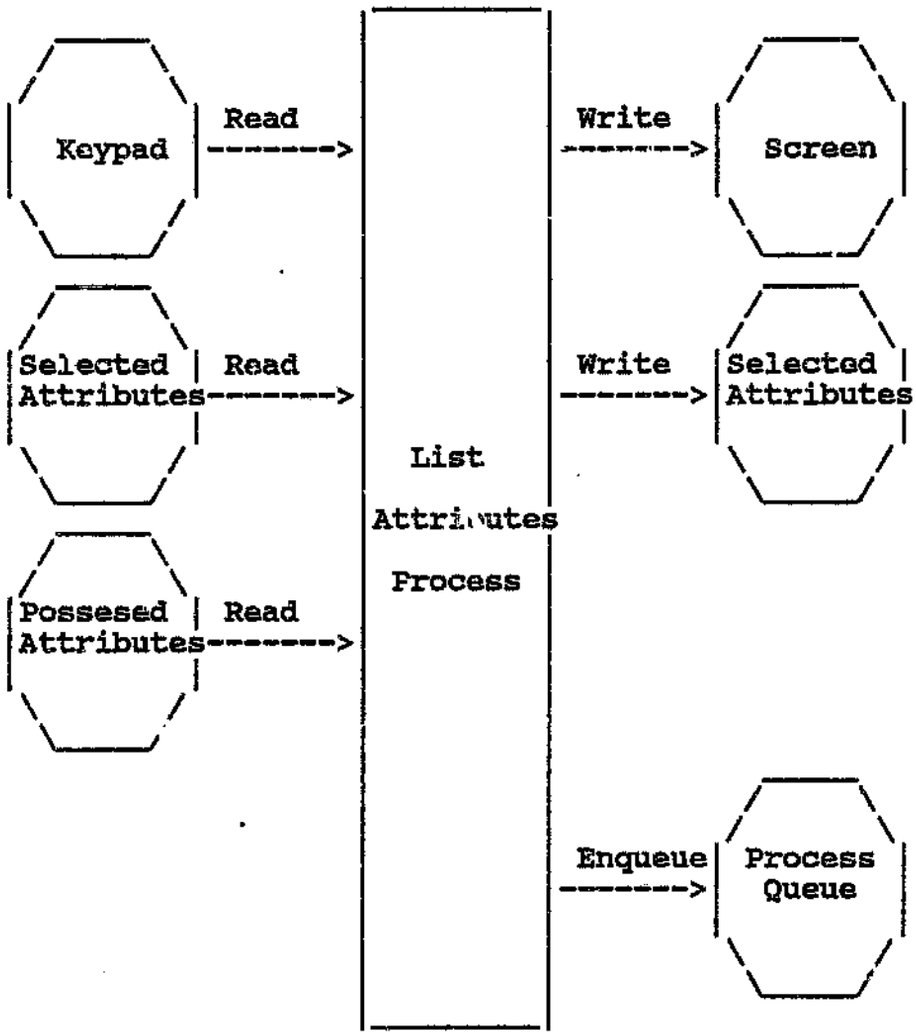


Figure 4.10 Third-level description: list attributes process

PDL description of list attributes process:-

```
Begin:
  Repeat:
    * Write number of selected attributes to window 5 *
    * Write title of selected attribute to window 1 *
    * Write name of selected attribute to window 4 *
    * Sense the keyboard for user commands *
    Case * Request of *

      Scroll: * Scroll up and down within the list
                right and left within the current item *
      Extract: * Extract the attributes from the selection list *
               * Recompile the list of matched birds *

      Mode switching: * Put the current process in the queue *
                      * Terminate process *

      Terminate process: * Terminate process *

      Help: * Provide relevant help information *

    End: Case:
  Until: * Process is terminated *
End:
```

4.6.1.7- CLEAR ATTRIBUTES PROCESS

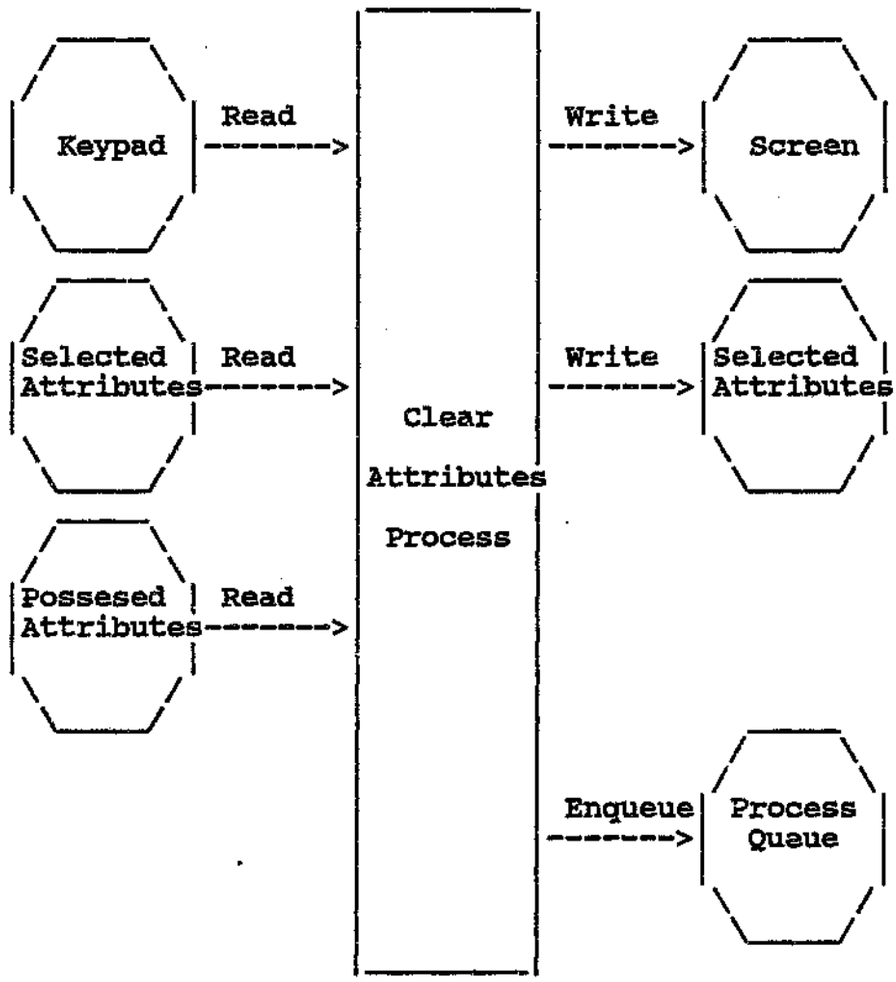


Figure 4.11 Third-level description: clear attributes process

PDL description of clear attributes process:-

```
Begin:
  Repeat:
    * Sense the keyboard for user commands *
    Case * Request of *

      Confirm: * Clear all selected attributes *
               * Recompile the list of matched birds *

      Abort: * Abort the clearing operation *

      Mode switching: * Put the current process in the queue *
                      * Terminate process *

      Help: * Provide relevant help information *

    End: Case:
  Until * Process is terminated *
End:
```

4.6.1.8- COUNT BIRDS PROCESS

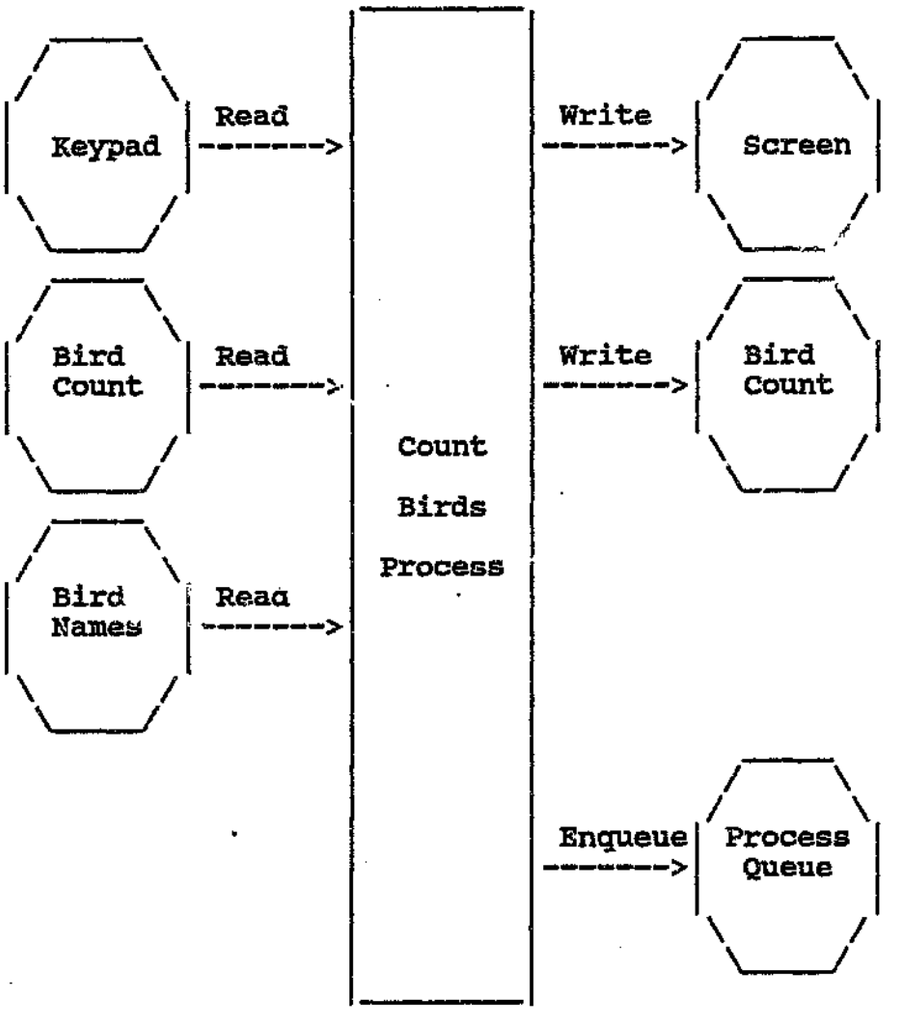


Figure 4.12 Third-level description: count birds process

PDL description of count birds process:-

```
Begin
  * Write bird name to window 4 *
  Repeat:
    * Write count number of current bird to window 1 *
    * Sense the keyboard for user commands *
    Case * Request of *

      Change count: * Increment or decrement the count of the
                      current listed bird *

      Mode switching: * Put the current process in the queue *
                      * Terminate process *

      Terminate process: * Terminate process *

      Help: * Provide relevant help information *

    End: Case:
  Until: * Process is terminated *
End:
```

4.6.2- SYSTEM RESOURCES

In the first-level design, system resources were broadly identified. In the following Sections we will discuss the role of each resource and the operations that they support. These operations are well-defined interfaces to the outside environment. They are the tools with which the system processes manage the flow of system data. This formulation of resources will furnish the OOD concepts of encapsulation and abstraction.

The function of each operation is described together with the passed parameters. Each operation is represented by an operation name followed by a list of passed-in parameter which are surrounded by a set of parenthesis, followed by a list of passed-out parameters.

4.6.2.1- THE SCREEN RESOURCE

This resource provides the display interface of the bird identifier in addition to the writing and clearing operations associated with it's management.

Data usage:-

Public:
- Variable:
  - **REV_W4** is the reverse video indicator of window 4
  - **W1_SIZE** is the physical size of window 1
  - **W4_SIZE** is the physical size of window 4
  - **GUIDE_ARROW_L** is the text indicator of window 2
  - **GUIDE_ARROW_R** is the text indicator of window 3

Private:
- Variables:
  - **W1_X** horizontal position of left most corner of window 1
  - **W1_Y** vertical position of left most corner of window 1
  - **W4_X** horizontal position of left most corner of window 4
  - **W4_Y** vertical position of left most corner of window 4
- Constant:
  - **W2_X** horizontal position of left most corner of window 2
  - **W2_Y** vertical position of left most corner of window 2

W3_X horizontal position of left most corner of window 3
W3_Y vertical position of left most corner of window 3
W5_X horizontal position of left most corner of window 5
W5_Y vertical position of left most corner of window 5
W2_SIZE length of window 2
W3_SIZE length of window 3
W5_SIZE length of window 5

List of Operations:-

1) O-SCREEN-FORMAT ()
   Behavior:
      This operation serves to emulate the screen display and the keypads of the identifier.
   Interface:
      Input: none
      Output: none

2) O-EXCH-W1-W4 ()
   Behavior:
      This operation will switch the physical position of window 1 with that of window 4. Effectively the text on the top line of display is exchanged with the text on the bottom line of display.
   Interface:
      Input: none
      Output: none

3) O-WRT-W1 (TEXT_DISPLAY)
   Behavior:
      This operation writes the passed in text parameter to window 1
   Interface:
      Input: TEXT_DISPLAY is the text that is written to window 1
      Output: none

4) O-WRT-W2 (W2_CHAR)
   Behavior:
      Indication of text lying to the left of the current window, (normally W4), is written into this single character window.
   Interface:
      Input: W2_CHAR is the character assigned to window 2
      Output: none

5) O-WRT-W3 (W3_CHAR)
   Behavior:
      Indication of text lying to the right of the current window, (normally W4), is written into this single character window.
   Interface:
      Input: W3_CHAR is the character assigned to window 3
      Output: none

6) O-WRT-W4 (TEXT_DISPLAY)
Behavior:
The passed in parameter, TEXT_DISPLAY, is written to window 4
Interface:
Input: TEXT_DISPLAY
Output: none

7) O-WRT-W5 (TEXT_DISPLAY)
Behavior:
The passed in parameter, TEXT_DISPLAY, is written to window 5. This window displays the number of matched birds, or the number of entries in list, when in sub-menus that support a list of birds.
Interface:
Input: TEXT_DISPLAY is the text that is written to window 5
Output: none

8) O-CLR-W1 ()
Behavior:
Clears the content of window 1.
Interface:
Input: none
Output: none

9) O-CLR-W2 ()
Behavior:
clears the content of window 2.
Interface:
Input: none
Output: none

10) O-CLR-W3 ()
Behavior:
clears the content of window 3.
Interface:
Input: none
Output: none

11) O-CLR-W4 ()
Behavior:
clears the content of window 4.
Interface:
Input: none
Output: none

12) O-CLR-W5 ()
Behavior:
clears the content of window 5.
Interface:
Input: none
Output: none

4.6.2.2- THE TEXT MENU RESOURCE

This resource supports the data structure of the text within the various menus, including help information relevant to each menu. This data structure is stored in a disc file. The operations of this resource provide access to the various menus.

As far as system processes are concerned all menus consist of a menu title, menu item, and relevant help information. Some operations return a status data item to indicate the outcome of the operation.

Data Usage:-

Public:
- Variable:
  - **MENU_ID** is the menu-id of the current/invoked process

Private:
- Variable:
  - **TXT_ARY** is the array containing the text of the menus
  - **FLAG_ARY** is the array containing select flag /menu id of next process
  - **FLAG_NO_ARY** is the array containing flag numbers. This information is used to indicate if the feature belongs to a group which does not allow multiple selections.
  - **NEXT_ID** is id of the next menu to which the current item is pointing
  - **KEY_ARY** array of type record; it serves to conserve the context of each process.
    - **MENU_ID** is the id of current menu
    - **MENU_TITLE** is the title of the current menu
    - **HELP** is a string of text containing help information
    - **C_POINTER** is the pointer to the current menu item supported by TXT_ARY
    - **F_POINTER** is the pointer to the first menu item supported by TXT_ARY
    - **L_POINTER** is the pointer to the last menu item supported by TXT_ARY
  - **NO_OF_SELECTED_ATTRIBUTES** stores the number of selected attributes
  - **LIST_FOUND_POINTER** is pointer to manage the list of found birds

List of Operations:-

1) O-READ-MENU-FILE () STATUS_
   Behavior:
      This operation retrieves and loads the menu text structure from a disc file.
   Interface:
      Input: none
      Output: STATUS_ has the outcome of operation successful, or operation failed.

2) O-WRITE-MENU-FILE ()
   Behavior:
      This operation will write the selected attributes to the menu disc file before terminating program.
   Interface:
      Input : none
      Output : none

3) O-READ-MENU-TITLE () MENU_TITLE, STATUS_
   Behavior:
      This operation reads the title of the current menu and returns the text via MENU_TITLE.
   Interface:
      Input: none
      Output: MENU_TITLE captures the title of the current menu and returns it to the invoking process.
              STATUS_ has the outcome of successful or unsuccessful

4) O-READ-TXT-ITEM (DISPLACEMENT_) TEXT_ITEM, STATUS_
   Behavior:
      This operation reads the line of text at the current/above/below text position. The relative displacement is defined by the value of DISPLACEMENT_. This text is then returned to the invoking process via TEXT_ITEM.
   Interface:
      Input: DISPLACEMENT_ specifies the requested text is at current position/above current position/below current position
      Output: TEXT_ITEM returns the line of text from the data structure.
              STATUS_ has the outcome of successful or fail

5) O_UPDATE_ATTR_FLAG () STATUS_
   Behavior:
      This operation has a dual function
      A) It toggles the state of selectable features (attributes).
      B) It serves to activate sub-menus from the higher level menus.
   Interface:
      Input: none
      Output: STATUS_ has the outcome of
              - selected menu item was of the selectable type
              - selected menu item was of a menu type

6) O_READ_NO_OF_SELECTED_ATTRIBUTES () NO_OF_S_ATTR
Behavior:
The fuction of this operation is to retrieve the number of attributes that have been selected, from the resource.
Interface:
Input: none
Output: NO_OF_S_ATTR is the number of selected attributes

7) O-CLEAR-ALL-ATTRIBUTES ()
Behavior:
All selected attributes are cleared through this operation.
Interface:
Input: none
Output: none

8) O_CLEAR_ATTRIBUTE;
Behavior:
Clears the current attribute from the list of attributes.
Interface:
Input: none
Output: none

9) O-LIST-SELECTED-ATTRIBUTES (DISPLACEMENT_) MENU_TITLE, MENU_ITEM
Behavior:
This operation will provide the list of selected attributes. The passed-in parameter, DISPLACEMENT_, specifies whether current attribute/last attribute/next attribute will be returned for display.
Interface:
Input: DISPLACEMENT_ defines the direction of search
Output: MENU_TITLE returns the menu title of the selected attribute.
MENU_ITEM returns the the selected item.

10) O-READ-LIST-FOUND-METHOD () METHOD_
Behavior:
This operation identifies whether extraction or bird count has been enabled in List Found Birds.
Interface:
Input: none
Output: METHOD_ indicates whether bird count or item extraction has been enabled in List Found Birds

11) O_READ_LIST_MATCHED_METHOD () LIST_MATCHED_METHOD
Behavior:
This operation which of the four sub-menus have been enabled in List Matched Birds.
Interface:
Input: none
Output: LIST_MATCHED_METHOD indicates which on of four functions has been enabled during List Mtched Birds

12) O-FILL-FILTER ()
Behavior:
The role of this operation is to compile a searching filter based on the selected attributes. This filter which is basically a short list of the selected attributes will be employed in the searching of the birds data base. Instead of returning the filter back through the interface it is placed in the global environmental module. In this way redundant data flow is avoided.
Interface:
Input: none
Output: none

4.6.2.3- THE QUEUE RESOURCE

This resource provides the facility of queuing the main activities of ATTRIBUTE and FIND.

Data Usage:-

Public:
Variable:
MENU_ID is the id of the current menu.
Private:
Variable:
STATUS_ is the status report of procedures and functions.
Q_ARY is the array emulating the queue
Q_TMP_ARY is the temporary queue array, used for context switching
Q_MIDL is the marker separating the find and attribute activities
Q_TAIL is the pointer marking the position of the last item in the queue
Constant:
Q_SIZE is the size of the queue set at 10

List of Operations:-

1) O-READ-Q-FILE ()
Behavior:
this operation retrieves the queue information from a disc file, Q_DAT.TXT, when the main program is first run
Interface:
Input: none
Output: none

2) O-WRITE-Q-FILE ()
Behavior:
This operation will write the queue data into file Q_DAT.TXT when quitting the program
Interface:
Input: none
Output: none

3) EXCHANGE-Q-BLOCKS ()
Behavior:
This operation will swap the segments of queue corresponding to the Attribute mode with segment corresponding to the Find mode, thus putting the relevant menu-id at the head of the queue
Interface:
Input: none
Output: none

4) O-FRONT-ENQ () STATUS_
Behavior:
This operation will enqueue the MENU_ID at the head of the queue
Interface:
Input: none
Output: STATUS_ has the outcome of
0 - successful
1 - unsuccessful / queue is full

5) O-DEQ () STATUS_
Behavior:
This operation will consume the item at the head of the queue, and writes it to the MENU_ID
Interface:
Input: none
Output: STATUS_ has the outcome of
0 - successful
1 - unsuccessful / queue empty

6) INITIALIZE_Q_FILE () Q_FILE
Behavior:
This operation checks the default drive for the queue file If it is not found then a second predefined path will be checked, before aborting.
Interface:
Input: none
Output: Q_FILE is the directory path where the queue file was found. This is the path where queue is saved before exiting from the program

4.6.2.4- THE SCROLL RESOURCE

This resource facilitates the function of word scrolling within the current line of text, which is stored in the variable TEXT_ITEM. This is achieved by displaying a segment of the text at any one time.

Data Usage:-

Public:

Variable:
TEXT_ITEM is the string text of the current menu item
REV_W4 is the reverse video indicator of window 4
W1_SIZE is the size of window 1
W4_SIZE is the size of window 4
GUIDE_ARROW_R
GUIDE_ARROW_L
W_NO is the window number of the active window, either 1 or 4

Private:
Variables:
W_L_POINTER is the window left pointer.
W_R_POINTER is the window right pointer.
TEXT_DISPLAY is the portion of text from the TEXT_ITEM to be displayed on the screen.
TEXT_LENGTH is the length of text in TEXT_ITEM.

List of operations:-

1) O-INITIAL-TEXT (TEXT_ITEM) TEXT_DISPLAY
Behavior:
This operation extracts the first x characters in the text contained in TEXT_ITEM and writes it to TEXT_DISPLAY
Interface:
Input: TEXT_ITEM contains the current line of text in the data structure
TEXT_DISPLAY contains a sub-string of text within TEXT_ITEM, for dislpaying on the screen
Output: TEXT_DISPLAY is the truncated text from the current line to be displayed on the current window

2) O-MOVE-TEXT-LEFT (TEXT_ITEM) TEXT_DISPLAY
Behavior:
This operation moves the displayed window of the text in MENU_ITEM one word to the left.
Interface:
Input: MENU_ITEM contains the current line of text in the data structure
Output: TEXT_DISPLAY contains a sub-string of text within MENU_ITEM, for dislpaying on the screen

3) O-MOVE-TEXT-RIGHT (MENU_ITEM) TEXT_DISPLAY
Behavior:
This operation moves the displayed window of the text in MENU_ITEM one word to the right
Interface:
Input: MENU_ITEM contains the current line of text in the data structure.
Output: TEXT_DISPLAY; contains a sub-string of text within MENU_ITEM for dislpaying on the screen

4.6.2.5- BIRD DATABASE RESOURCE

Bird attributes, bird information, bird name, bird family, bird S.A number, bird code and bird count data structures are managed in this resource. Operations of the resource allow access to bird data including the scan and compare mechanism required for short listing and generation of the list of matched birds. The static part of the database includes bird attributes, bird information, bird name, bird S.A number , beird code and bird family; and are stored on a disc file. The dynamic part of this database comprising of the bied count is stored in a seperate disc file.

Data Usage:-

Public:

Variable:
MENU_ID is the id of the current menu.
BIRD_FAMILY is the family to which the current bird belongs
BIRD_NAME is the name of the current bird
BIRD_COUNT is the count number of the current bird

Private:
Variable:
STRATEGY_ is the search and compare strategy of selected attributes against the bird attributes; four possible values. a)no attribute is selected; compile all birds for display b) only bird family is selected c)a flag type attribute is selected; bird family is not selected d) both family and flag type attributes have been selected
NO_OF_MATCHED_BIRDS is the number of birds matching the selected attributes
MATCH_LIST_NO identifies which of the four possible

functions of a)bird count, b)bird information, c)bird image or d)bird call have been enabled
NO_FOUND_BIRDS is the number of birds with a bird count greater than zero
FOUND_LIST_NO identifies whether extraction or bird count are enabled during the list found birds
DATA_ARY is of the type record; this is the bird database
BIR_NAME is the name bird
FAMILY_ is the family of the bird
SAOS_ is the South African bird number
FR_CODE is the bird habitat code
BIRD_ATTR contains the flag type attributes of the bird
BIRD_COUNT is the count number of the bird
MATCH_ indicates if the current record has matched attribute specifications
BIRD_REC is the DATA_ARY record pointer

List of operations:-

1) O-READ-BIRD-DATA ()
Behavior:
This operation retrieves the static (read-only) segment of the bird database, from a disc file.
Interface:
Input: none
Output: none

2) O-SCAN ()
Behavior:
The bird data base will be scanned and compared against the selected attributes, to generate a short list of the matched birds. This operation
Interface:
Input: none
Output: none

3) O-LIST-MATCHED-BIRDS (DISPLACEMENT_)
Behavior:
The retrieval of bird species from the short list of matched birds is achieved through this operation. The passed-in parameter, DISPLACEMENT, specifies whether current/next/previous entry will be retrieved. The name and family of the retrieved bird species will then be written to the display from the calling process. To prevent the redundant flow of data this information is placed into the global environmental module, rather than being returned to the calling process.
Interface:
Input: DISPLACEMENT_ defines the direction of retrieval
Output: none

4) O-READ-COUNT-FILE ()
Behavior:
The bird count information is read from a disc file.
Interface:
Input: none
Output: none

5) O-WRITE-COUNT-FILE ()
Behavior:
The bird count information is written to a disc file
Interface:
Input: none
Output: none

6) O-CHANGE-BIRD-COUNT (DISPLACEMENT_)
Behavior:
Bird counts are adjusted through this operation. The passed-in parameter, DISPLACEMENT_, specifies whether the count should be incremented or decremented.
Interface:
Input: DISPLACEMENT_ defines if count should be incremented or decremented.
Output: none

7) O-[illegible]ND-BIRDS () NO_OF_FOUND_BIRDS
Behavior:
This operation retrieves the number of birds which have a count value of greater than zero. This is necessary to display the total number of found birds, when in List Found Birds.
Interface:
Input: none
Output: NO_OF_FOUND_BIRDS is the number of found birds

8) O-LIST-FOUND-BIRDS (DISPLACEMENT_)
Behavior:
The retrieval of bird species that have a count value of greater than zero is achieved through this operation. The passed-in parameter, DISPLACEMENT, specifies whether current/next/previous entry will be retrieved. The name and family of the retrieved bird species will then be written to the display from the calling process. To prevent the redundant flow of data this information is placed into the global environmental module, rather than being returned to the calling process.
Interface:
Input: DISPLACEMENT_ defines the direction of retrieval.
Output: none

9) O-CLEAR-FOUND-BIRDS ()
Behavior:
clears the list of found birds.
Interface:
Input: none
Output: none

10) O-CLEAR-BIRD ()
   Behavior:
      Clears the current bird from the list of found birds.
   Interface:
      Input: none
      Output: none

11) O-READ-NO-OF-MATCHED-BIRDS () NO_OF_MATCHED_BIRDS
   Behavior:
      This operation retrieves the number of birds which match the selected attributes. This is necessary to display the total number of matched birds, when in List Matched Birds.
   Interface:
      Input: none
      Output: NO_OF_MATCHED_BIRDS is the number of matched birds

4.7- SUMMARY

During the course of this chapter we have covered the powerful method of process/resource decomposition of OOD objects. The importance of divorcing the conceptual design from the implementation method was next discussed. To complement this discipline a PDL documentation method was adopted. By partitioning the object identification and classification exercise into three progressive high to low level stages, the system processes and resources were described.

CHAPTER 5- DISCUSSION AND CONCLUSION

5.1- ACHIEVEMENTS

A personal computer software model has been developed. This model provides all the prescribed facilities of the identifier except the bird image and bird call facilities. In spite of this, bird identification is still very much possible. In fact except for those situations where bird image and call remain as the final means of identification, the simple model has proved to be very practical and efficient. We have provided the bird identifier model with a bird database of 742 bird species containing 194 attributes (bird family through to markings; see Appendix A) . In the interest of simplicity and modifiability this database is stored in disc file without any compression or optimizing schemes. Despite this lack of efficiency the scan and compare mechanism of 'List Matched Birds' activity is hardly noticeable to the user. Recall that the result of each attribute selection is immediately fed back to the user by recompiling the list through the List Matched Birds operation and displaying the number of birds in the list, on window 5.

5.2- IMPLEMENTATION

Turbo Pascal V5.5 was used to implement the PDL descriptions. The reason for this choice was due to the structured syntax of Pascal. It furnished a High-level programming platform, on which one could concentrate on producing a sound system, without being hampered with language barriers. Conscious effort was made to avoid the use of complex data structures such as pointers, in anticipation of making the code more portable. This was considered necessary as the personal computer model would eventually be imported into a hand-held environment such as a microcontroller. With this in mind we have further restricted our OOD realization to the use of Turbo Pascal 'Units', rather than Turbo 'Objects'.

The entire size of the bird identifier code is 36 Kb and the uncoded birds database is 67 Kb.(A list modules and their corresponding code size is listed in Appendix.C.). The programe code includes extensive status checking of most operations, indicating the precise outcome of the operation. This was done to build some degree of self testing into the model; thus making for simpler debugging procedure. Once the employed databases are successfully employed by the software model, the checking operations may be removed from the code to ensure that the final code volume can be comfortably accomodated by a hand-held device. However, even in the present form, the database together with the program can be supported by a 8051 microcontroller environment, without difficulty.

5.3- TESTING

Testing is a vital and necessary part of any software development. To ensure efficient debugging, the testing effort was systematically divided into two separate stages; the testing of individual processes and the final testing of the integrated system. All modules were tested individually, prior to system integration. The validation of the 'List Matched Birds' module was by far the most important exercise at this level of testing. This was done by truncating the bird database to eleven birds. Several combinations of features were selected and the list of generated matched birds were cross-checked with those of the database. Final testing stage involved the testing of the functional behavior of the entire system. Due to the simplicity of the model and it's user-interface, extensive generation of test vectors was not necessary. In fact the final functional testing was carried-out in less than thirty minutes.

5.4- IMPACT OF SOFTWARE ENGINEERING METHODOLOGY

In conforming with the objectives of this work we will discuss the software engineering implications of design architecture and development tools of the bird identifier fieldguide.

5.4.1- OBJECT-CENTERED APPROACH TO DESIGN

Although the architectural design does not exploit all OOD concepts such as inheritance, but it does conform to a system decomposition that is based on the idea of real world objects. We experienced that this discipline goes a long way in promoting

the understandability of the system, it's testing and maintenance. Firstly the object decomposition exercise followed the behavioral descriptions naturally. In this regard the Attribute and Find Mode descriptions played a two-fold role of providing fast and efficient access to the system functions as well as furnishing a reasonable model on which object decomposition could be based. Secondly, the effort of testing and modifications was much simplified, since the scope of activity was very much localized. Thirdly, because of the encapsulation property of objects, integration was achieved with very little difficulty.

5.4.2- TOOL SUPPORT

In spite of abundance of Computer Aided Software Engineering tools (CASE), users of these tools feel that it will be sometimes before a comprehensive and yet simple to use CASE tool package, is made available. In the case of the bird identifier design, the degree of complexity did not warrant a powerful development tool. A PDL formatter was used to develop the architectural concepts of the design and was later expanded to a more detailed level in order to furnish the documentation needs of the design. As far as implementation is concerned Turbo Pascal was used to perform the editing, running and debugging of code, all within the same environment.

5.4.3- REPORT OF DESIGN CYCLE TIME SCALE

Table below shows the different design stages and associated hours of activity. It is important to state that the figures in this table represent the lump amount of time spent in each stage; as this design cycle was reiterated several times to completion.

Activity	Used Hours	% of Total Hours
Architectural Design	100	11.8
Detailed Design (PDL)	200	23.5
Coding	180	21.2
Testing	70	8.2
Documentation	300	35.3
Total	850	100.0

5.5- CONCLUSIONS

Even with best coding techniques available today, storing of sound and picture information places considerable demands on the required memory space. With rapid advances in the field of memory chips and display units, however, electronic fieldguides offering sound and picture information as well as text are becoming more feasible. Notwithstanding these powerful capabilities, the electronic fieldguide remains the fastest means of information access; and therefore it is far superior to the conventional implementations. As we have previously indicated response time is a fundamental measure of capability and usefulness. A good example of such a situation is in the field of medical science, where the process of a swift diagnosis is of major importance. Electronic fieldguides can also play a major role in the educational field, where encouragement by the way of providing quick solutions, will serve to sustain the level of interest and will ultimately result in uplifting the knowledge of the student. This is particularly true in the case of the younger user, were it is well accepted that youngsters are quite fearless of the electronic technology. In fact as the electronic game market indicates, the sheer element of curiosity which is shared by most teen-agers, is the first and foremost advocate. By tapping into this tremendous facet of the younger generation, electronic fieldguides can provide a significant contribution towards the promotion of science.

BIBLIOGRAPHY

1. Cole B.C, "A 16-Mb DRAM Grabs The Spotlight", Electronics Week, Vol.60, No.5, pp59-60 1987

2. Maclean K.N "Roberts' Birds of Southern Africa", John Voelcker, Cape Town, 1985

3. Newman k, "Birds of South Africa", Mac,illan, South Africa, 1983

4. Stephen W.D and Donald A.N, "Software Engineering for User Interface", IEEE Transactions on Software Engineering, Vol.SE-11, No.3, pp252-258, March 1985

5. Bernice E.C., "A Games Approach to System Interface Design", IEEE Transactions on Professional Communications VOL.PC 29, No.4, pp65-71, December 1986

6. Stonebraker M, Allman E, "Observations on The Evolution of a Software System", IEEE Computer, pp27-32, June 1982

7. Booch G, "Object-Oriented Development", IEEE Transactions, Software Eng, Vol.SE-12, No.2, pp211-221, Feb 1986

8. Walker A.J, "Structured Information Processing System Design", Software engineering notes, University of the Witswtersrand, pp3.2, 1986

9. Allman E. and Stonebraker M, "Observation on the Evelution of a Software System", IEEE Computer, pp27-32, June 1982

10. Schneidewind N.F, "The State of Software Maintenance", IEEE Transactions on Software Engineering, Vol.SE-13, No.3, pp303-309, March 1987

11. Caine H.S and Gordon E.K, "PDL - A tool for Software Design" reprinted in Tutorial on Software Design, edited by P.Freeman and A Wasserman, IEEE Press, pp380-385, 1975

12. Walker A.J, Dinkelacker A.M, "PDL Formatter" Software Engineering Applications Lab. University of the Witwatersrand, 1988

APPENDIX. A- LIST OF ATTRIBUTE SUB-MENUS

A - National Parks (provides a single flag for selection)

A2.1 - Kruger National Park
A2.2 - Mountain Zebra National Park
A2.3 - Karoo National Park
A2.4 - Addo Elephant National Park
A2.5 - Langebaan National Park
A2.6 - Swartvlei
A2.7 - Tsitsikamma Forest National Park
A2.8 - Golden Gate Highlands National Park
A2.9 - Augrabies Falls National Park
A2.10 - Bontebok National Park
A2.11 - Kalahari Gemsbok National Park

A3 - Hiking Trails (provides a single flag for selection)

A3.1 - Fanie Botha
A3.2 - Soutspansberg
A3.3 - Otter
A3.4 - Outeniqua

A4 - Towns and Cities (provides a single flag for selection)

A4.1 - Johannesburg
A4.2 - Pretoria
A4.3 - Cape Town
A4.4 - Port Elizabeth
A4.5 - East London
A4.6 - Durban

A5 -Bird Sanctuaries (provides a single flag for selection)

A5.1 - Barberspan
A5.2 - Rondebult
A5.3 - Marievale
etc

A6 - Habitats (provides a single flag for selection)

A6.1 - Towns, Parks and Gardens
A6.2 - Roadside Birds
A6.3 - Cultivation and Grasslands
A6.4 - Fynbos
A6.5 - Karoo
A6.6 - Semi-desert
A6.7 - Thorn Trees in Grassveld
A6.8 - Broadleaved Bushveld and Woodland

A6.9 - Riverine and Coastal Bush: River Edge
A6.10 - Evergreen Forest
A6.11 - Wetlands
A6.12 - Estuaries and Seashores
A6.13 - Oceans
A6.14 - Mountains, Rocky and Grassy Slopes

A7 - Status (max. of 3 logically ANDed options)

A7.1 - Abundant	--flag #1
A7.2 - Common	--flag #1
A7.3 - Uncommon	--flag #1
A7.4 - Rare	--flag #1
A7.5 - Vagrant	--flag #2
A7.6 - Summer visitor	--flag #2
A7.7 - Winter visitor	--flag #2
A7.8 - Resident	--flag #2
A7.9 - Diurnal	--flag #3
A7.10 -Nocturnal	--flag #3

A8 - Bird Families (provides a single flag for selection)

A8.1 - Albatrosses
A8.2 - Alethe
A8.3 - Apalis
A8.4 - Avocet
A8.5 - Babblers
A8.6 - Barbets
A8.7 - Batis
A8.8 - Bee-eaters
A8.9 - Bishops
A8.10 - Booby
A8.11 - Broadbill
A8.12 - Bulbuls
A8.13 - Buntings
A8.14 - Bush Shrikes
A8.15 - Bustards
A8.16 - Buttonquails
A8.17 - Buzzards
A8.18 - Canaries
A8.19 - Chaffinches
A8.20 - Chats
A8.21 - Cisticolas
A8.22 - Coot
A8.23 - Cormorants
A8.24 - Coucals
A8.25 - Coursers
A8.26 - Crakes
A8.27 - Cranes
A8.28 - Crimsonwing
A8.29 - Crombecs
A8.30 - Crows
A8.31 - Cuckoo Finches
A8.32 - Cuckoohawk

A8.33 - Cuckoos
A8.34 - Cuckooshrikes
A8.35 - Curlew
A8.36 - Darter
A8.37 - Dikkops
A8.38 - Doves
A8.39 - Drongos
A8.40 - Ducks
A8.41 - Eagles
A8.42 - Egrets
A8.43 - Falcons
A8.44 - Finches
A8.45 - Finfoot
A8.46 - Fire Finches
A8.47 - Flamingos
A8.48 - Flufftails
A8.49 - Flycatchers
A8.50 - Francolins
A8.51 - Frigate Birds
A8.52 - Gallinules
A8.53 - Gannets
A8.54 - Geese
A8.55 - Godwits
A8.56 - Goshawks
A8.57 - Grassbird
A8.58 - Guineafowl
A8.59 - Gulls
A8.60 - Gymnogene
A8.61 - Hamerkop
A0.62 - Harriers
A8.63 - Helmet Shrikes
A8.64 - Herons
A8.65 - Honeyguides
A8.66 - Hoopoes
A8.67 - Hornbills
A8.68 - Hyliotas
A8.69 - Ibises
A8.70 - Jacanas
A8.71 - Kestrels
A8.72 - Kingfishers
A8.73 - Kites
A8.74 - Korhaans
A8.75 - Larks
A8.76 - Longclaws
A8.77 - Louries
A8.78 - Lovebirds
A8.79 - Manikins
A8.80 - Martins
A8.81 - Moorhens
A8.82 - Mousebirds
A8.83 - Nicator
A8.84 - Nightjars
A8.85 - Orioles
A8.86 - Ostrich
A8.87 - Owls
A8.88 - Oxpeckers

A8.89 - Oystercatchers
A8.90 - Palm Thrushes
A8.91 - Parrots
A8.92 - Penduline Tits
A8.93 - Penguins
A8.94 - Petrels
A8.95 - Phalaropes
A8.96 - Pigeons
A8.97 - Pipits
A8.98 - Pitta
A8.99 - Plovers
A8.100 - Pratincoles
A8.101 - Prinias
A8.102 - Pytilias
A8.103 - Quails
A8.104 - Queleas
A8.105 - Rails
A8.106 - Raven
A8.107 - Robins
A8.108 - Rockjumpers
A8.109 - Rockrunner
A8.110 - Rollers
A8.111 - Sandgrouse
A8.112 - Sandpipers
A8.113 - Secretary Bird
A8.114 - Seedcrackers
A8.115 - Shearwaters
A8.116 - Shrikes
A8.117 - Siskins
A8.118 - Skimmer
A8.119 - Skuas
A8.120 - Snipes
A8.121 - Sparrowhawks
A8.122 - Sparrows
A8.123 - Starlings
A8.124 - Stilt
A8.125 - Stints
A8.126 - Storks
A8.127 - Sugarbirds
A8.128 - Sunbirds
A8.129 - Swallows
A8.130 - Swan
A8.131 - Swifts
A8.132 - Tchagras
A8.133 - Terns
A8.134 - Thrushes
A8.135 - Titbabblers
A8.136 - Tits
A8.137 - Trogon
A8.138 - Tropic Birds
A8.139 - Turnstone
A8.140 - Twinspots
A8.141 - Vultures
A8.142 - Wagtails
A8.143 - Warblers
A8.144 - Waxbills

A8.145 - Weavers
A8.146 - Wheatears
A8.147 - Whimbrel
A8.148 - White-eyes
A8.149 - Whydahs
A8.150 - Widow Finches
A8.151 - Widows
A8.152 - Woodpeckers
A8.153 - Wryneck

A9 - Size (provides a single flag for selection)

A9.1 - Larger than a Turtle Dove
A9.2 - About the size of a Turtle Dove
A9.3 - Between turtle dove and Cape Sparrow
A9.4 - About the size of a Cape Sparrow
A9.5 - Smaller than a Cape Sparrow

A10 - Feeding Habits (provides a single flag for selection)

A10.1 - Flying
A10.2 - Perching
A10.3 - On Ground
A10.4 - Wet Areas
A10.5 - Water
A10.6 - Climbing and Creeping

A11 - Features (max. of 6 logically ANDed options)

A11.1 - Long legs --flag #1
A11.2 - Long beak --flag #2
A11.3 - Long neck --flag #3
A11.4 - Long tail --flag #4
A11.5 - Fork tail --flag #4
A11.6 - Square tail --flag #4
A11.7 - Bill heavy or hooked --flag #5
A11.8 - Bill short and conical --flag #5
A11.9 - Head crested --flag #6

A12 - Coloring (max. of 9 logically ANDed options)

A12.1 - Mostly white --flag #1
A12.2 - Mostly black --flag #2
A12.3 - Black and dark or white and gray --flag #3
A12.4 - Mostly brown --flag #4
A12.5 - Yellow or orange present --flag #5
A12.6 - Green or blue present --flag #6
A12.7 - Red present --flag #7
A12.8 - Rust or buff present --flag #8
A12.9 - Beak, eye or feet distinctively marked #9

A13 - Markings (max. of 6 logically ANDed options)

A13.1 - Underparts plain	--flag #1
A13.2 - Underparts spotted	--flag #1
A13.3 - Underparts stripped	--flag #1
A13.4 - Underparts barred	--flag #1
A13.5 - Upperparts marked	--flag #1
A13.6 - Stripe through or over eye	--flag #2
A13.7 - Top of head distinct	--flag #3
A13.8 - Eye ring distinct	--flag #4
A13.9 - Boldly marked sitting or standing	--flag #5
A13.10 -Boldly marked in flight	--flag #5
A13.11 -Tail longitudinally marked	--flag #6
A13.12 -Tail cross-marked	--flag #6

A14 - Utilities (no flags)

A14.1 - Set initial key delay (only applies to arrow keys)
A14.2 - Set scroll rate (only applies to arrow keys)
A14.3 - Enable counting in List Found Birds
A14.4 - Enable extraction in List Found Birds

APPENDIX. B- PDL DOCUMENTATION OF THE BIRD IDENTIFIER

****

Birds

Revision.2 (12/10/90)

Time 12 : 27

Date 15 10 1990

****

TABLE OF CONTENTS

TABLE OF CONTENTS

Public Constants

----------D- 1A -D----------

```
D  1    ID_ATTR_M ..is the process-id of the 'Main Attribute' menu
D  2    ID_FIND_M ..is the process-id of the 'Main Find' menu
D  3    ID_HELP ....is the process-id of the 'Help' menu
D  4    ID_COUNTRY .is the process-id of the 'Countries' menu
D  5    ID_N_PARKS .is the process-id of the 'National Parks' menu
D  6    ID_HIKING ..is the process-id of the 'Hiking Trails' menu
D  7    ID_TOWNS ...is the process-id of the 'Towns and Cities' menu
D  8    ID_BIRD_SAN is the process-id of the 'Bird Sanctuaries' menu
D  9    ID_HABITAT .is the process-id of the 'Habitats' menu
D 10    ID_STATUS ..is the process-id of the 'Status' menu
D 11    ID_BIRD_F ..is the process-id of the 'Bird Families' menu
D 12    ID_SIZE ....is the process-id of the 'Bird Size' menu
D 13    ID_FEEDING .is the process-id of the 'Feeding' menu
D 14    ID_FEATURE .is the process-id of the 'Features' menu
D 15    ID_COLORING is the process-id of the 'colorings' menu
D 16    ID_MARKING .is the process-id of the 'Markings' menu
D 17    ID_UTILITY .is the process-id of the 'Utilities' menu
```

Public Variables

----------D- 1B -D----------

```
D  1    PROCESS_ID .....is the id of the current menu.
D  2    LAST_PROCESS_ID is the id of the of the last process
```

Private Constants

----------D- 1C -D----------

```
D  1    MENU_TITLE is the title of the current menu.
```

Private Variables

```
------------D-  2A  -D-------------------------------------------------------------
```

```
     D  1     TEXT_ITEM ...is the string text of the current menu item.
     D  2     TEXT_DISPLAY portion of text from the TEXT_ITEM for display on screen.
     D  3     CONTINUE_ ...is the boolean variable for the while loop.
     D  4     STATUS_ .....is the status report of procedures and functions.
     D  5     KEY_ .......is the character containing the pressed key.
     D  6     C_POINTER ...points to the current entry in the menu
     D  7     F_POINTER ...points to the first entry in the menu
     D  8     L_POINTER ...points to the last entry in the menu
     D  9     W_L_POINTER .is the window left pointer.
     D 10     W_R_POINTER .is the window right pointer.
     D 11     CHAR_ .......is a character of the text TEXT_ITEM for testing purposes.
     D 12     TEXT_LENGTH .is the length of text in TEXT_ITEM.
     D 13     DISPLACEMENT_ defines vertical scroll direction relative to current position
     D 14     ATTRIBUTE_FLAG is an indicator for attribute selection
     D 15     W1_X is the screen column position of left corner of window 1
     D 16     W1_Y is the screen row position of left corner of window 1
     D 17     W3_X is the screen column position of left corner of window 3
     D 18     W3_Y is the screen row position of left corner of window 3
     D 19     W4_X is the screen column position of left corner of window 4
     D 20     W4_Y is the screen row position of left corner of window 4
     D 21     W1_SIZE is th size of window 1
     D 22     W2_SIZE is th size of window 2
     D 23     W3_SIZE is th size of window 3
     D 24     W4_SIZE is th size of window 4
```

****

Scheduler Process

****

Scheduler Data Usage

```
-------------C-   4A  -C-------------------------------------------------------

     C  1     Public Constants:
     C  2     Public Variables:
     C  3     ...PROCESS_ID is the id of the current menu.
     C  4     Private Constants:
     C  5     ...ID_ATTR_M ..is the process-id of the 'Main Attribute' menu
     C  6     ...ID_FIND_M ..is the process-id of the 'Main Find' menu
     C  7     ...ID_HELP ....is the process-id of the 'Help' menu
     C  8     ...ID_COUNTRY .is the process-id of the 'Countries' menu
     C  9     ...ID_N_PARKS .is the process-id of the 'National Parks' menu
     C 10     ...ID_HIKING ..is the process-id of the 'Hiking Trails' menu
     C 11     ...ID_TOWNS ...is the process-id of the 'Towns and Cities' menu
     C 12     ...ID_BIRD_SAN is the process-id of the 'Bird Sanctuaries' menu
     C 13     ...ID_HABITAT .is the process-id of the 'Habitats' menu
     C 14     ...ID_STATUS ..is the process-id of the 'Status' menu
     C 15     ...ID_BIRD_F ..is the process-id of the 'Bird Families' menu
     C 16     ...ID_SIZE ....is the process-id of the 'Bird Size' menu
     C 17     ...ID_FEEDING .is the process-id of the 'Feeding' menu
     C 18     ...ID_FEATURE .is the process-id of the 'Features' menu
     C 19     ...ID_COLORING is the process-id of the 'colorings' menu
     C 20     ...ID_MARKING .is the process-id of the 'Markings' menu
     C 21     ...ID_UTILITY .is the process-id of the 'Utilities' menu
     C 22     ...KEY_ is the character containing the pressed key
     C 23     PDL Description:
     C 24     ...
```

Scheduler Process

```
-------------P-   5A  -P-------------------------------------------------------

     P  1     if SYSTEM_ON is false
 61A P  2        format the screen [O-SCREEN-FORMAT]
 41A P  3        read bird database from file [O-READ-BIRD-DATA]
     P  4        read text data from file [O-READ-TXT-DATA]
     P  5        if STATUS is not succesfull [<>0]
     P  6           case STATUS_
     P  7           of [1], Text file not found;
     P  8              TEXT_ := 'Text file not found'
     P  9           of [2], % Charater expected
     P 10              TEXT_ := '% Character expected'
     P 11           of [3], %N Expected
     P 12              TEXT_ := '%N Expected'
     P 13           of [4], %N Expected
     P 14              TEXT_ := '%N Expected'
     P 15           endcase
 63A P 16           write TEXT_ to window 4 [O-WRT-W4]
 67B P 17           quit program [P-QUIT-PROG]
     P 18        else
     P 19           read queue data from file [O-READ-Q-DATA]
     P 20           SYSTEM_ON := true
 38B P 21           compile attribute filter [O-FILL-FILTER]
 42A P 22           scan database [O-SCAN]
 47A P 23           read number of matched birds [O-READ-NO-OF-MATCHED-BIRDS]
 63B P 24           write number to window 5 [O-WRT-W5]
     P 25        endif
     P 26     endif
     P 27     while STOP_ is false
 53A P 28        dequeue process-id [O-DEQ]
     P 29        if STATUS_ is not succeful [=1]
 63A P 30           write "Queue Empty" to window 4 [O-WRT-W4]
 67A P 31           wait for a keypress [O-WAIT]
 65A P 32           clear window 4 [O-CLR-W4]
     P 33        else
     P 34           case PROCESS_ID
     P 35           of help
  8A P 36              execute Help Menu Process [P-HELP]
     P 37           of attribute menu
 11A P 38              execute Attribute Menu Process [P-ATTR-MENU]
     P 39           of utilities
 13A P 40              execute Utility Menu Process [P-UTIL]
     P 41           of find main menu
 16A P 42              execute Find Main Menu Process [P-FIND-MENU]
     P 43           of list found birds
 1BA P 44              execute List Found Birds Process [P-LIST-FOUND]
     P 45           of clear found birds
 19B P 46              execute Clear Found Birds Process [P-CLEAR-FOUND]
     P 47           of list attributes
 21A P 48              execute List Attributes Process [P-LIST-ATTR]
     P 49           of clear attribute
 22B P 50              execute Clear Attributes Process [P-CLEAR-ATTR]
     P 51           of list matched birds
```

```
24A P 52                execute List Matched Birds Process [P-LIST-MATCHED]
    P 53            of count bird
26A P 54                execute Count Bird Process [P-COUNT-BIRD]
    P 55            of clear bird
27B P 56                execute Clear a Found Bird Process [P-CLEAR-BIRD]
    P 57            endcase
    P 58        endif
    P 59    endwhile
```

****

Help Process

****

Help Data Usage

```
-------------C-   7A  -C-------------------------------------------------------------

     C  1     Public Variables:
     C  2     ...PROCESS_ID is the id of the current menu.
     C  3     Private Variables:
     C  4     ...DISPLACEMENT_ defines vertical scroll direction relative to current position
     C  5     ...MENU_TITLE is the title of the current menu.
     C  6     ...TEXT_ITEM is the string text of the current menu item.
     C  7     ...TEXT_DISPLAY portion of text from the TEXT_ITEM for display on screen.
     C  8     ...CONTINUE_ is the boolean variable for the while loop.
     C  9     ...STATUS_ is the status report of procedures and functions.
     C 10     ...KEY_ is the character containing the pressed key.
     C 11     PDL Description:
     C 12     ...
```

Help Menu Process [P-HELP]

```
-------------P-   BA  -P-------------------------------------------------------------

 61B P  1     write "Help" to window 1 [O-WRT-W1]
     P  2     DISPLACEMENT_ := 0
 33A P  3     read an item from menu [O-READ-TXT-ITEM]
 56A P  4     initialize text [O-INITIAL-TEXT]
 62A P  5     write to window 2 [O-WRT-W2]
 62B P  6     write to window 3 [O-WRT-W3]
 63A P  7     write TEXT_DISPLAY to window 4 [O-WRT-W4]
     P  8     CONTINUE := true
     P  9     while CONTINUE_
     P 10         repeat
     P 11             read keyboard
     P 12             KEY_ := pressed key
     P 13         until KEY_ is (right arrow) (left arrow) or (escape)
     P 14         case KEY_
     P 15         of KEY_LEFT_ARROW
     P 16             move text left [O-MOVE-TXT-LEFT]
 62A P 17             write to window 2 [O-WRT-W2]
 62B P 18             write to window 3 [O-WRT-W3]
 63A P 19             write TEXT_DISPLAY to window 4 [O-WRT-W4]
     P 20         of KEY_RIGHT_ARROW
 56A P 21             move text right [O-MOVE-TEXT-RIGHT]
 62A P 22             write to window 2 [O-WRT-W2]
 62B P 23             write to window 3 [O-WRT-W3]
 63A P 24             write TEXT_DISPLAY to window 4 [O-WRT-W4]
     P 25         of KEY_ESCAPE
     P 26             CONTINUE_ := false
     P 27         endcase
     P 28     endwhile
```

****

Attribute Processes

****

Attribute Menu Data Usage

```
-------------C-  10A  -C-------------------------------------------------------------

     C  1     Public Variables:
     C  2     ...PROCESS_ID := is the id of the current process.
     C  3     Private Variables:
     C  4     ...DISPLACEMENT_ defines vertical scroll direction relative to current position
     C  5     ...MENU_TITLE is the title of the current menu.
     C  6     ...TEXT_ITEM is the string text of the current menu item.
     C  7     ...TEXT_DISPLAY portion of text from the TEXT_ITEM for display on screen.
     C  8     ...CONTINUE_ is the boolean variable for the while loop.
     C  9     ...STATUS_ is the status report of procedures and functions.
     C 10     ...KEY_ is the character containing the pressed key.
     C 11     PDL Description:
     C 12     ...
     C 13
```

Attribute Menu Process [P-ATTR-MENU]

```
------------P-  11A  -P-------------------------------------------------------

 32A P  1     read title of menu,[O-READ-MENU-TITLE]
 61B P  2     write MENU_TITLE to window 1 [O-WRT-W1]
     P  3     DISPLACEMENT_ := 0
 33A P  4     read an item from menu [O-READ-TXT-ITEM]
 56A P  5     initialize text [O-INITIAL-TEXT]
 62A P  6     write to window 2 [O-WRT-W2]
 62B P  7     write to window 3 [O-WRT-W3]
 63A P  8     write TEXT_DISPLAY to window 4 [O-WRT-W4]
     P  9     CONTINUE_ := TRUE
     P 10     while CONTINUE_
     P 11         repeat
     P 12             read keyboard
     P 13             KEY_ := pressed key
     P 14         until KEY_ is (help) or (mode) or (select) or (left arrow)
     P 15         or (right arrow) or (up arrow) or (down arrow)
     P 16         or (escape) or (quit)
     P 17         case KEY_
     P 18         of KEY_HELP
 52A P 19             enqueue process-id [O-FRONT-ENQ]
     P 20             PROCESS_ID := HELP_ID
 52A P 21             enqueue process-id [O-FRONT-ENQ]
     P 22             CONTINUE_ := false
     P 23         of KEY_MODE
 52A P 24             enqueue process-id [O-FRONT-ENQ]
 51A P 25             exchange queue blocks [O-EXCH-Q-BLOCKS]
     P 26             CONTINUE_ := false
     P 27         of KEY_SELECT
     P 28             ...enqueue the id of current process in case selected item is another menu
 52A P 29             enqueue process-id [O-FRONT-ENQ]
 34A P 30             update attribute flag [O-UPDATE-ATTR-FLAG]
     P 31             case STATUS_
     P 32             of 1
     P 33                 ...selected item points to another menu
 52A P 34                 enqueue process-id [O-FRONT-ENQ]
     P 35                 CONTINUE_ := false
     P 36             of 0
     P 37                 ...selected item was a flag type
 53A P 38                 dequeue process-id [O-DEQ]
 38B P 39                 compile attribute filter [O-FILL-FILTER]
 42A P 40                 scan database [O-SCAN]
 47A P 41                 read number of matched birds [O-READ-NO-OF-MATCHED-BIRDS]
 53B P 42                 write number to window 5 [O-WRT-W5]
     P 43                 DISPLACEMENT_ := 0
 33A P 44                 read an item from menu [O-READ-TXT-ITEM]
 56A P 45                 initialize text [O-INITIAL-TEXT]
 62A P 46                 write to window 2 [O-WRT-W2]
 62B P 47                 write to window 3 [O-WRT-W3]
 63A P 48                 write TEXT_DISPLAY to window 4 [O-WRT-W4]
     P 49             endcase
     P 50         of KEY_UP_ARROW
     P 51             DISPLACEMENT_ := -1
```

```
33A P 52            read item an from menu [O-READ-TXT-ITEM]
   P 53             if STATUS_ = 0
56A P 54                 initialise text [O-INITIAL-TEXT]
62A P 55                 write to window 2 [O-WRT-W2]
62B P 56                 write to window 3 [O-WRT-W3]
63A P 57                 write TEXT_DISPLAY to window 4 [O-WRT-W4]
   P 58             else
63A P 59                 write "Pointers Not Found" to window 4 [O-WRT-W4]
67A P 60                 wait for a keypress [O-WAIT]
   P 61             endif
   P 62         of KEY_DN_ARROW
   P 63             DISPLACEMENT_ := +1
33A P 64             read an item from menu [O-READ-TXT-ITEM]
   P 65             if STATUS_ is 0
56A P 66                 initialiise text [O-INITIAL-TEXT]
62A P 67                 write to window 2 [O-WRT-W2]
62B P 68                 write to window 3 [O-WRT-W3]
63A P 69                 write TEXT_DISPLAY to window 4 [O-WRT-W4]
   P 70             else
63A P 71                 write "Pointers Not Nound" to window 4 [O-WRT-W4]
67A P 72                 wait for a keypress [O-WAIT]
   P 73             endif
   P 74         of KEY_LEFT_ARROW
57A P 75             move text left [O-MOVE-TEXT-LEFT]
62A P 76             write to window 2 [O-WRT-W2]
62B P 77             write to window 3 [O-WRT-W3]
63A P 78             write TEXT_DISPLAY to window 4 [O-WRT-W4]
   P 79         of KEY_RIGHT_ARROW
58A P 80             move text right [O-MOVE-TEXT-RIGHT]
62A P 81             write to window 2 [O-WRT-W2]
62B P 82             write to window 3 [O-WRT-W3]
63A P 83             write TEXT_DISPLAY to window 4 [O-WRT-W4]
   P 84         of KEY_ESCAPE
   P 85             if PROCESS_ID = attribute main menu
   P 86                 ...do nothing
   P 87             else
   P 88                 ...restore control the main menu
   P 89                 CONTINUE_ = false
   P 90             endif
   P 91         of KEY_QUIT
67B P 92             quit program [P-QUIT-PROG]
   P 93             CONTINUE_ := false
   P 94             STOP_ := tr
   P 95         endcase
   P 96     endwhile
```

Utility Menu Data Usage

```
-------------C-  12A  -C-------------------------------------------------------------

     C  1     Public Variables:
     C  2     ...PROCESS_ID is the id of the current process.
     C  3     Private Variables:
     C  4     ...DISPLACEMENT_ defines vertical scroll direction relative to current position
     C  5     ...MENU_TITLE is the title of the current menu.
     C  6     ...TEXT_ITEM is the string text of the current menu item.
     C  7     ...TEXT_DISPLAY portion of text from the TEXT_ITEM for display on screen.
     C  8     ...CONTINUE_ is the boolean variable for the while loop.
     C  9     ...STATUS_ is the status report of procedures and functions.
     C 10     ...KEY_ is the character containing the pressed key.
     C 11     PDL Description:
     C 12     ...
```

Utility Menu Process [P-UTIL]

```
------------P-  13A  -P-------------------------------------------------------

 32A P  1     read title of menu,[O-READ-MENU-TITLE]
 61B P  2     write MENU_TITLE to window 1 [O-WRT-W1]
     P  3     DISPLACEMENT_ := 0
 33A P  4     read an item from menu [O-READ-TXT-ITEM]
 56A P  5     initialize text [O-INITIAL-TEXT]
 62A P  6     write to window 2 [O-WRT-W2]
 62B P  7     write to window 3 [O-WRT-W3]
 63A P  8     write TEXT_DISPLAY to window 4 [O-WRT-W4]
     P  9     CONTINUE_ := TRUE
     P 10     while CONTINUE_
     P 11         repeat
     P 12             read keyboard
     P 13             KEY_ := pressed key
     P 14         until KEY_ is (help) or (mode) or (select) or (left arrow)
     P 15         or (right arrow) or (up arrow) or (down arrow)
     P 16         or (escape) or (quit)
     P 17         case KEY_
     P 18         of KEY_HELP
 52A P 19             enqueue process-id [O-FRONT-ENQ]
     P 20             PROCESS_ID := HELP_ID
 52A P 21             enqueue process-id [O-FRONT-ENQ]
     P 22             CONTINUE_ := false
     P 23         of KEY_MODE
 52A P 24             enqueue process-id [O-FRONT-ENQ]
 51A P 25             exchange queue blocks [O-EXCH-Q-BLOCKS]
     P 26             CONTINUE_ := false
     P 27         of KEY_SELECT
 52A P 28             enqueue process-id [O-FRONT-ENQ]
 34A P 29             update attribute flag [O-UPDATE-ATTR-FLAG]
     P 30             case STATUS_
     P 31             of 1
     P 32                 ...selected item points to another menu
 52A P 33                 enqueue process-id [O-FRONT-ENQ]
     P 34                 CONTINUE_ := false
     P 35             of 0
     P 36                 ...selected item was a flag type
 53A P 37                 dequeue process-id [O-DEQ]
     P 38                 DISPLACEMENT_ := 0
 33A P 39                 read an item from menu [O-READ-TXT-ITEM]
 56A P 40                 initialize text [O-INITIAL-TEXT]
 62A P 41                 write to window 2 [O-WRT-W2]
 62B P 42                 write to window 3 [O-WRT-W3]
 63A P 43                 write TEXT_DISPLAY to window 4 [O-WRT-W4]
     P 44             endcase
     P 45         of KEY_UP_ARROW
     P 46             DISPLACEMENT_ := -1
 33A P 47             read item an from menu [O-READ-TXT-ITEM]
 56A P 48             initialise text [O-INITIAL-TEXT]
 62A P 49             write to window 2 [O-WRT-W2]
 62B P 50             write to window 3 [O-WRT-W3]
 63A P 51             write TEXT_DISPLAY to window 4 [O-WRT-W4]
```

```
    P 52          of KEY_DN_ARROW
    P 53              DISPLACEMENT_ := +1
33A P 54              read an item from menu [O-READ-TXT-ITEM]
56A P 55              initialise text [O-INITIAL-TEXT]
62A P 56              write to window 2 [O-WRT-W2]
62B P 57              write to window 3 [O-WRT-W3]
63A P 58              write TEXT_DISPLAY to window 4 [O-WRT-W4]
    P 59          of KEY_LEFT_ARROW
57A P 60              move text left [O-MOVE-TEXT-LEFT]
62A P 61              write to window 2 [O-WRT-W2]
62B P 62              write to window 3 [O-WRT-W3]
63A P 63              write TEXT_DISPLAY to window 4 [O-WRT-W4]
    P 64          of KEY_RIGHT_ARROW
58A P 65              move text right [O-MOVE-TEXT-RIGHT]
62A P 66              write to window 2 [O-WRT-W2]
62B P 67              write to window 3 [O-WRT-W3]
63A P 68              write TEXT_DISPLAY to window 4 [O-WRT-W4]
    P 69          of KEY_ESCAPE
    P 70              CONTINUE_ = false
    P 71          of KEY_QUIT
67B P 72              quit program [P-QUIT-PROG]
    P 73              CONTINUE_ := false
    P 74          endcase
    P 75      endwhile
```

****

Find Processes

****

Find Main Menu Data Usage

```
------------C-  15A  -C-------------------------------------------------------------

     C  1     Public Variables:
     C  2     ...PROCESS_ID is the id of the current process.
     C  3     Private Variables:
     C  4     ...DISPLACEMENT_ defines vertical scroll direction relative to current position
     C  5     ...MENU_TITLE is the title of the current menu.
     C  6     ...TEXT_ITEM is the string text of the current menu item.
     C  7     ...TEXT_DISPLAY portion of text from the TEXT_ITEM for display on screen.
     C  8     ...CONTINUE_ is the boolean variable for the while loop.
     C  9     ...STATUS_ is the status report of procedures and functions.
     C 10     ...KEY_ is the character containing the pressed key.
     C 11     PDL Description:
     C 12     ...
     C 13
```

```
Find Main Menu Process [P-FIND-MENU]

-------------P-  16A  -P-------------------------------------------------------

 47A P  1     read number of matched birds [O-READ-NO-OF-MATCHED-BIRDS]
 63B P  2     write number to window 5 [O-WRT-W5]
 32A P  3     read title of menu,[O-READ-MENU-TITLE]
 61B P  4     write MENU_TITLE to window 1 [O-WRT-W1]
     P  5     DISPLACEMENT_ := 0
 33A P  6     read an item from menu [O-READ-TXT-ITEM]
 56A P  7     initialize text [O-INITIAL-TEXT]
 62A P  8     write to window 2 [O-WRT-W2]
 62B P  9     write to window 3 [O-WRT-W3]
 63A P 10     write TEXT_DISPLAY to window 4 [O-WRT-W4]
     P 11     CONTINUE_ = true
     P 12     while CONTINUE_
     P 13         repeat
     P 14             read keyboard
     P 15             KEY_ := pressed key
     P 16         until KEY_ is (help) or (mode) or (select) or (left arrow)
     P 17         or (right arrow) or (up arrow) or (down arrow) or (quit)
     P 18         case KEY_
     P 19         of KEY_HELP
 52A P 20             enqueue process-id [O-FRONT-ENQ]
     P 21             PROCESS_ID := HELP_ID
 52A P 22             enqueue process-id [O-FRONT-ENQ]
     P 23             CONTINUE_ := false
     P 24         of KEY_MODE
 52A P 25             enqueue process-id [O-FRONT-ENQ]
 51A P 26             exchange queue blocks [O-EXCH-Q-BLOCKS]
     P 27             CONTINUE_ := false
     P 28         of KEY_SELECT
     P 29             ...enqueue the id of current process in case selected item is another menu
 52A P 30             enqueue process-id [O-FRONT-ENQ]
 34A P 31             update attribute flag [O-UPDATE-ATTR-FLAG]
     P 32             case STATUS_
     P 33             of 1
     P 34                 ...selected item points to another menu
 52A P 35                 enqueue process-id [O-FRONT-ENQ]
     P 36                 CONTINUE_ := false
     P 37             of 0
     P 38                 ...selected item was a flag type
 53A P 39                 dequeue process-id [O-DEQ]
 38B P 40                 compile attribute filter [O-FILL-FILTER]
 42A P 41                 scan database [O-SCAN]
 47A P 42                 read number of matched birds [O-READ-NO-OF-MATCHED-BIRDS]
 63B P 43                 write number to window 5 [O-WRT-W5]
     P 44                 DISPLACEMENT_ := 0
 33A P 45                 read an item from menu [O-READ-TXT-ITEM]
 56A P 46                 initialize text [O-INITIAL-TEXT]
 62A P 47                 write to window 2 [O-WRT-W2]
 62B P 48                 write to window 3 [O-WRT-W3]
 63A P 49                 write TEXT_DISPLAY to window 4 [O-WRT-W4]
     P 50             endcase
     P 51         of KEY_UP_ARROW
```

```
    P 52            DISPLACEMENT_ := -1
33A P 53            read item an from menu [O-READ-TXT-ITEM]
56A P 54            initialise text [O-INITIAL-TEXT]
62A P 55            write to window 2 [O-WRT-W2]
62B P 56            write to window 3 [O-WRT-W3]
63A P 57            write TEXT_DISPLAY to window 4 [O-WRT-W4]
    P 58        of KEY_DN_ARROW
    P 59            DISPLACEMENT_ := +1
33A P 60            read an item from menu [O-READ-TXT-ITEM]
56A P 61            initiallise text [O-INITIAL-TEXT]
62A P 62            write to window 2 [O-WRT-W2]
62B P 63            write to window 3 [O-WRT-W3]
63A P 64            write TEXT_DISPLAY to window 4 [O-WRT-W4]
    P 65        of KEY_LEFT_ARROW
57A P 66            move text left [O-MOVE-TEXT-LEFT]
62A P 67            write to window 2 [O-WRT-W2]
62B P 68            write to window 3 [O-WRT-W3]
63A P 69            write TEXT_DISPLAY to window 4 [O-WRT-W4]
    P 70        of KEY_RIGHT_ARROW
58A P 71            move text right [O-MOVE-TEXT-RIGHT]
62A P 72            write to window 2 [O-WRT-W2]
62B P 73            write to window 3 [O-WRT-W3]
63A P 74            write TEXT_DISPLAY to window 4 [O-WRT-W4]
    P 75        of KEY_ESCAPE
    P 76            if PROCESS_ID = attribute main menu
    P 77                ...do nothing
    P 78            else
    P 79                ...restore control the main menu
    P 80                CONTINUE_ = false
    P 81            endif
    P 82        of KEY_QUIT
67B P 83            quit program [P-QUIT-PROG]
    P 84            CONTINUE_ := false
    P 85        endcase
    P 86    endwhile
```

List Found Birds Data Usage

```
------------C-  17A  -C-------------------------------------------------------

     C  1     Public Variables:
     C  2     ...PROCESS_ID is the id of the current process.
     C  3     Private Variables:
     C  4     ...DISPLACEMENT_ defines vertical scroll direction relative to current position
     C  5     ...MENU_TITLE is the title of the current menu.
     C  6     ...TEXT_ITEM is the string text of the current menu item.
     C  7     ...TEXT_DISPLAY portion of text from the TEXT_ITEM for display on screen.
     C  8     ...CONTINUE_ is the boolean variable for the while loop.
     C  9     ...STATUS_ is the status report of procedures and functions.
     C 10     ...KEY_ is the character containing the pressed key.
     C 11     ...NO_OF_FOUND_BIRDS is number of found birds
     C 12     PDL Description:
     C 13     ...
     C 14
```

List Found Birds Process [P-LIST-FOUND]

```
------------P-  18A  -P-------------------------------------------------------

 45A P  1     read number of found birds [O-READ-NO-OF-FOUND-BIRDS]
 63B P  2     write number to window 5 [O-WRT-W5]
     P  3     if NO_OF_FOUND_BIRDS = 0
 61B P  4         write 'List is Clear' to window 1 [O-WRT-W1]
 63A P  5         write 'Esc to continue' to window 4 [O-WRT-W4]
     P  6         repeat
     P  7             read keyboard
     P  8         until Escape is pressed
     P  9     else
     P 10         if LAST_PROCESS_ID is (find) or (clear bird) or
     P 11             (bird count and bird count is zero)
     P 12             DISPLACEMENT_ := 0
 45B P 13             list found birds [O-LIST-FOUND-BIRDS]
     P 14         endif
 61B P 15         write BIRD_COUNT to window 1 [O-WRT-W1]
 63A P 16         write BIRD_NAME to window 4 [O-WRT-W4]
 37A P 17         read list found method [O-READ-LIST-FOUND-METHOD]
     P 18         CONTINUE_ = true
     P 19         while CONTINUE_
     P 20             repeat
     P 21                 read keyboard
     P 22                 KEY_ := pressed key
     P 23             until KEY_ is (help) or (mode) or (select) or (left arrow)
     P 24             or (right arrow) or (up arrow) or (down arrow) or (exit)
     P 25             case KEY_
     P 26             of KEY_HELP
 52A P 27                 enqueue process-id [O-FRONT-ENQ]
     P 28                 PROCESS_ID := HELP_ID
 52A P 29                 enqueue process-id [O-FRONT-ENQ]
     P 30                 CONTINUE_ := false
     P 31             of KEY_MODE
 52A P 32                 enqueue process-id [O-FRONT-ENQ]
 51A P 33                 exchange queue blocks [O-EXCH-Q-BLOCKS]
     P 34                 CONTINUE_ := false
     P 35             of KEY_SELECT
 52A P 36                 enqueue process-id [O-FRONT-ENQ]
     P 37                 case listing method in list found birds
     P 38                 of count bird
     P 39                     PROCESS_ID := count bird
     P 40                 of CLEAR BIRD
     P 41                     PROCESS_ID := clear bird
     P 42                 endcase
 52A P 43                 enqueue process-id [O-FRONT-ENQ]
     P 44                 CONTINUE_ := false
     P 45             of KEY_UP_ARROW
     P 46                 DISPLACEMENT_ := -1
 45B P 47                 read list of found birds [O-LIST-FOUND-BIRDS]
 61B P 48                 write BIRD_COUNT to window 1 [O-WRT-W1]
 63A P 49                 write BIRD_NAME to window 4 [O-WRT-W4]
     P 50             of KEY_DN_ARROW
     P 51                 DISPLACEMENT_ := +1
```

```
45B P 52                read list of found birds [O-LIST-FOUND-BIRDS]
61B P 53                write BIRD_COUNT to window 1 [O-WRT-W1]
63A P 54                write BIRD_NAME to window 4 [O-WRT-W4]
    P 55            of KEY_ESCAPE
    P 56                CONTINUE_ = false
    P 57            of KEY_QUIT
67B P 58                quit program [P-QUIT-PROG]
    P 59                CONTINUE_ := false
    P 60                STOP_ := true
    P 61            endcase
    P 62        endwhile
    P 63    endif
```

Clear Found Birds Data Usage

```
------------C-  19A  -C-------------------------------------------------------

     C  1     Public Variables:
     C  2     ...PROCESS_ID is the id of the current process.
     C  3     Private Variables:
     C  4     ...MENU_TITLE is the title of the current menu.
     C  5     ...TEXT_ITEM is the string text of the current menu item.
     C  6     ...TEXT_DISPLAY portion of text from the TEXT_ITEM for display on screen.
     C  7     ...CONTINUE_ is the boolean variable for the while loop.
     C  8     ...STATUS_ is the status report of procedures and functions.
     C  9     ...KEY_ is the character containing the pressed key.
     C 10     PDL Description:
     C 11     ...
     C 12
```

Clear Found Birds Process [P-CLEAR-FOUND]

```
------------P-  19B  -P-------------------------------------------------------

 61B P  1     write 'Clear' to w[illegible] [O-WRT-W1]
 63A P  2     write 'Esc=abort se[illegible]im' to window 4 [O-WRT-W4]
     P  3     repeat
     P  4         read keyboard
     P  5         KEY_ := pressed key
     P  6     until KEY_ is (help) or (mode) or (select)
     P  7     or (left arrow) or (escape)
     P  8     case KEY_
     P  9     of KEY_HELP
 52A P 10         enqueue process-id [O-FRONT-ENQ]
     P 11         PROCESS_ID := HELP_ID
 52A P 12         enqueue process-id [O-FRONT-ENQ]
     P 13         CONTINUE_ := false
     P 14     of KEY_MODE
 52A P 15         enqueue process-id [O-FRONT-ENQ]
 51A P 16         exchange queue blocks [O-EXCH-Q-BLOCKS]
     P 17         CONTINUE_ := false
     P 18     of KEY_SELECT
 46A P 19         clear found birds [O-CLEAR-FOUND-BIRDS]
     P 20     of KEY_ESCAPE
     P 21         ...do notihng
     P 22     endcase
```

List Attributes Data Usage

```
------------C-  20A  -C-------------------------------------------------------
     C  1     Public Variables:
     C  2     ...PROCESS_ID is the id of the current process.
     C  3     Private Variables:
     C  4     ...DISPLACEMENT_ defines vertical scroll direction relative to current position
     C  5     ...MENU_TITLE is the title of the current menu.
     C  6     ...TEXT_ITEM is the string text of the current menu item.
     C  7     ...TEXT_DISPLAY portion of text from the TEXT_ITEM for display on screen.
     C  8     ...CONTINUE_ is the boolean variable for the while loop.
     C  9     ...STATUS_ is the status report of procedures and functions.
     C 10     ...KEY_ is the character containing the pressed key.
     C 11     PDL Description:
     C 12     ...
     C 13
```

List Attributes Process [P-LIST-ATTR]

```
------------P-  21A  -P-------------------------------------------------------

 35A P  1     [O-READ-NO-OF-SELECTED-ATTRIBUTES]
 63B P  2     write number to window 5 [O-WRT-W5]
     P  3     if NO_OF_SELECTED_ATTRIBUTES is zero
 61B P  4         write 'none selected' to window 1 [O-WRT-W1]
 63A P  5         write 'Esc to cintinue' to window 4 [O-WRT-W4]
     P  6         repeat
     P  7             read keyboard
     P  8             KEY_ := pressed key
     P  9         until KEY_ is (escape)
     P 10     else
     P 11         DISPLACEMENT_ := 0
 36B P 12         list selected attributes [O-LIST-SELECTED-ATTRIBUTES]
 61B P 13         write title to window 1 [O-WRT-W1]
 63A P 14         write item to window 4 [O-WRT-W4]
     P 15         CONTINUE_ := true
     P 16         while CONTINUE_
     P 17             repeat
     P 18                 read keyboard
     P 19                 KEY_ := pressed key
     P 20             until KEY_ is (help) or (mode) or (select) or (left arrow)
     P 21             or (right arrow) or (up arrow) or (down arrow) or
     P 22             or (escape) or (quit)
     P 23             case KEY_
     P 24             of KEY_HELP
 52A P 25                 enqueue process-id [O-FRONT-ENQ]
     P 26                 PROCESS_ID := HELP_ID
 52A P 27                 enqueue process-id [O-FRONT-ENQ]
     P 28                 CONTINUE_ := false
     P 29             of KEY_MODE
 52A P 30                 enqueue process-id [O-FRONT-ENQ]
 51A P 31                 exchange queue blocks [O-EXCH-Q-BLOCKS]
     P 32                 CONTINUE_ := false
     P 33             of KEY_SELECT
 61B P 34                 write 'Clear' to window 1 [O-WRT-W1]
 63A P 35                 write 'Esc=abort Sel=affrim' to window 4 [O-WRT-W4]
     P 36                 repeat
     P 37                     read keyboard
     P 38                     KEY_ := pressed key
     P 39                 until KEY_ is (select) or (escape)
     P 40                 case KEY_
     P 41                 of KEY_SELECT
 36B P 42                     clear attribute [O-CLEAR-ATTRIBUTE]
 38B P 43                     compile attribute filter [O-FILL-FILTER]
 42A P 44                     scan database [O-SCAN]
 35A P 45                     [O-READ-NO-OF-SELECTED-ATTRIBUTES]
 63B P 46                     write number to window 5 [O-WRT-W5]
     P 47                     if NO_OF_SELECTED_ATTRIBUTES is zero
 61B P 48                         write 'none selected' to window 1 [O-WRT-W1]
 63A P 49                         write 'Esc to cintinue' to window 4 [O-WRT-W4]
     P 50                         repeat
     P 51                             read keyboard
```

```
    P 52                        KEY_ := pressed key
    P 53                    until KEY_ is (escape)
    P 54                    CONTINUE_ := false
    P 55                else
    P 56                    DISPLACEMENT_ := 0
36B P 57                    list selected attributes [O-LIST-SELECTED-ATTRIBUTES]
    P 58                endif
    P 59                if CONTINUE_ <> FALSE
61B P 60                    write title to window 1 [O-WRT-W1]
63A P 61                    write item to window 4 [O-WRT-W4]
    P 62                endif
    P 63            endcase
    P 64        of KEY_UP_ARROW
    P 65            DISPLACEMENT_ := -1
36B P 66            list selected attributes [O-LIST-SELECTED-ATTRIBUTES]
61B P 67            write title to window 1 [O-WRT-W1]
63A P 68            write item to window 4 [O-WRT-W4]
    P 69        of KEY_DN_ARROW
    P 70            DISPLACEMENT_ := +1
36B P 71            list selected attributes [O-LIST-SELECTED-ATTRIBUTES]
61B P 72            write title to window 1 [O-WRT-W1]
63A P 73            write item to window 4 [O-WRT-W4]
    P 74        of KEY_LEFT_ARROW
57A P 75            move text left [O-MOVE-TEXT-LEFT]
62A P 76            write to window 2 [O-WRT-W2]
62B P 77            write to window 3 [O-WRT-W3]
63A P 78            write TEXT_DISPLAY to window 4 [O-WRT-W4]
    P 79        of KEY_RIGHT_ARROW
58A P 80            move text right [O-MOVE-TEXT-RIGHT]
62A P 81            write to window 2 [O-WRT-W2]
62B P 82            write to window 3 [O-WRT-W3]
63A P 83            write TEXT_DISPLAY to window 4 [O-WRT-W4]
    P 84        of KEY_ESCAPE
    P 85            CONTINUE_ = false
    P 86        of KEY_QUIT
67B P 87            quit program [P-QUIT-PROG]
    P 88            CONTINUE_ := false
    P 89            STOP_ := true
    P 90        endcase
    P 91    endwhile
    P 92 endif
```

Clear Attribute Data Usage

```
------------C-  22A  -C-------------------------------------------------------

     C  1     Public Variables:
     C  2     ...PROCESS_ID is the id of the current process.
     C  3     Private Variables:
     C  4     ...MENU_TITLE is the title of the current menu.
     C  5     ...TEXT_ITEM is the string text of the current menu item.
     C  6     ...TEXT_DISPLAY portion of text from the TEXT_ITEM for display on screen.
     C  7     ...CONTINUE_ is the boolean variable for the while loop.
     C  8     ...STATUS_ is the status report of procedures and functions.
     C  9     ...KEY_ is the character containing the pressed key.
     C 10     PDL Description:
     C 11     ...
     C 12
```

Clear Attributes Process [P-CLEAR-ATTR]

```
------------P-  22B  -P-------------------------------------------------------

 61B P  1     write clear message to window 1 [O-WRT-W1]
 63A P  2     write 'Esc=abort;$el=affirm' to window 4 [O-WRT-W4]
     P  3     repeat
     P  4         read keyboard
     P  5         KEY_ := pressed key
     P  6     until KEY_ is (help) or (mode) or (select) or (escape)
     P  7     case KEY_
     P  8     of KEY_HELP
 52A P  9         enqueue process-id [O-FRONT-ENQ]
     P 10         PROCESS_ID .= HELP_ID
 52A P 11         enqueue process-id [O-FRONT-ENQ]
     P 12         CONTINUE_ := false
     P 13     of KEY_MODE
 52A P 14         enqueue process-id [O-FRONT-ENQ]
 51A P 15         exchange queue blocks [O-EXCH-Q-BLOCKS]
     P 16         CONTINUE_ := false
     P 17     of KEY_SELECT
 35B P 18         clear all attributes [O-CLEAR-ALL-ATTRIBUTES]
 38B P 19         compile attribute filter [O-FILL-FILTER]
 42A P 20         scan database [O-SCAN]
 47A P 21         [O-READ-NO-OF-HATCHED-BIRDS]   .
 63B P 22         write number to window 5 [O-WRT-W5]
     P 23     of KEY_ESCAPE
     P 24         CONTINUE_ = false
     P 25     endcase
```

List Matched Birds Data Usage

```
------------C-  23A  -C-------------------------------------------------------

     C  1     Public Variables:
     C  2     ...PROCESS_ID is the id of the current process.
     C  3     Private Variables:
     C  4     ...DISPLACEMENT_ defines vertical scroll direction relative to current position
     C  5     ...MENU_TITLE is the title of the current menu.
     C  6     ...TEXT_ITEM is the string text of the current menu item.
     C  7     ...TEXT_DISPLAY portion of text from the TEXT_ITEM for display on screen.
     C  8     ...CONTINUE_ is the boolean variable for the while loop.
     C  9     ...STATUS_ is the status report of procedures and functions.
     C 10     ...KEY_ is the character containing the pressed key.
     C 11     ...NO_OF_MATCHED_BIRDS is number of matched birds
     C 12     PDL Description:
     C 13     ...
     C 14
```

List Matched Birds Process

```
------------P-  24A  -P-------------------------------------------------------

 47A P  1     read number of matched birds [O-READ-NO-OF-MATCHED-BIRDS]
 63B P  2     write number to window 5 [O-WRT-W5]
    P  3     if NO_OF_MATCHED_BIRDS is zero
 61B P  4         write 'Not found' to window 1 [O-WRT-W1]
 63A P  5         write 'ESC to continue' to window 4 [O-WRT-W4]
    P  6         repeat
    P  7             read keyboard
    P  8             KEY_ := pressed key
    P  9         until KEY_ is (escape)
    P 10         if (LAST_PROCESS_ID <> help) or LAST_PROCESS_ID <> bird count)
    P 11             DISPLACEMENT_ := 0
 43A P 12            list matched birds [O-LIST-MATCHED-BIRDS]
    P 13         endif
 63A P 14         write BIRD_NAME to window 4 [O-WRT-W4]
 61B P 15         write BIRD_FAMILY to window 1 [O-WRT-W1]
    P 16         CONTINUE_ := false
    P 17         while CONTINUE_
 38A P 18             read list matched method [O-READ-LIST-MATCHED-METHOD]
    P 19             repeat
    P 20                 read keyboard
    P 21                 KEY_ := pressed key
    P 22             until KEY_ is (help) or (mode) or (select) or (left arrow)
    P 23             or (right arrow) or (up arrow) or (down arrow) or
    P 24             or (escape) or (quit)
    P 25             case KEY_
    P 26             of KEY_HELP
 52A P 27                enqueue process-id [O-FRONT-ENQ]
    P 28                 PROCESS_ID := HELP_ID
 52A P 29                enqueue process-id [O-FRONT-ENQ]
    P 30                 CONTINUE_ := false
    P 31             of KEY_MODE
 52A P 32                enqueue process-id [O-FRONT-ENQ]
 51A P 33                exchange queue blocks [O-EXCH-Q-BLOCKS]
    P 34                 CONTINUE_ := false
    P 35             of KEY_SELECT
 52A P 36                enqueue process-id [O-FRONT-ENQ]
    P 37                 case LIST_MATCHED_METHOD
    P 38                 of bird count
    P 39                     PROCESS_ID := bird count
    P 40                 of bird call
    P 41                     PROCESS_ID := bird call
    P 42                 of bird image
    P 43                     PROCESS_ID := bird image
    P 44                 of bird info
    P 45                     PROCESS_ID := bird info
    P 46                 endcase
 52A P 47                enqueue process-id [O-FRONT-ENQ]
    P 48                 CONTINUE_ := false
    P 49             of KEY_UP_ARROW
    P 50                 DISPLACEMENT_ := -1
 43A P 51                read list of matched birds [O-LIST-MATCHED-BIRDS]
```

```
61B P 52                write BIRD_FAMILY to window 1 [O-WRT-W1]
63A P 53                write BIRD_NAME to window 4 [O-WRT-W4]
   P 54              of KEY_DN_ARROW
   P 55                DISPLACEMENT_ := +1
43A P 56                read list of matched birds [O-LIST-MATCHED-BIRDS]
61B P 57                write BIRD_FAMILY to window 1 [O-WRT-W1]
63A P 58                write BIRD_NAME to window 4 [O-WRT-W4]
   P 59              of KEY_LEFT_ARROW
57A P 60                move text left [O-MOVE-TEXT-LEFT]
62A P 61                write to window 2 [O-WRT-W2]
62B P 62                write to window 3 [O-WRT-W3]
63A P 63                write TEXT_DISPLAY to window 4 [O-WRT-W4]
   P 64              of KEY_RI  _ARROW
58A P 65                move t xt right [O-MOVE-TEXT-RIGHT]
62A P 66                write to window 2 [O-WRT-W2]
62B P 67                write to window 3 [O-WRT-W3]
63A P 68                write TEXT_DISPLAY to window 4 [O-WRT-W4]
   P 69              of KEY_ESCAPE
   P 70                 CONTINUE_ = false
   P 71              of KEY_QUIT
67B P 72                quit program [P-QUIT-PROG]
   P 73                 CONTINUE_ := false
   P 74                 STOP_ := true
   P 75              endcase
   P 76           endwhile
   P 77        endif
```

Bird Count Data Usage

```
-------------C- 25A -C-------------------------------------------------------

     C  1     Public Variables:
     C  2     ...PROCESS_ID is the id of the current process.
     C  3     Private Variables:
     C  4
     C  5
     C  6     ...DISPLACEMENT_ defines vertical scroll direction relative to current position
     C  7     ...MENU_TITLE is the title of the current menu.
     C  8     ...TEXT_ITEM is the string text of the current menu item.
     C  9     ...TEXT_DISPLAY portion of text from the TEXT_ITEM for display on screen.
     C 10     ...CONTINUE_ is the boolean variable for the while loop.
     C 11     ...STATUS_ is the status report of procedures and functions.
     C 12     ...KEY_ is the character containing the pressed key.
     C 13     PDL Description:
     C 14     ...
     C 15
```

Count Bird Process [P-COUNT-BIRD]

```
-------------P-  26A  -P-------------------------------------------------------------

     P  1     if LAST_PROCESS_ID =(bird count) or (help)
     P  2         DISPLACEMENT_ := 0
 44B P  3         list bird count [O-CHANGE-BIRD-COUNT]
     P  4     endif
     P  5     CONTINUE_ := true
     P  6     while CONTINUE_
 38A P  7         read list matched method [O-READ-LIST-MATCHED-METHOD]
     P  8         repeat
     P  9             read keyboard
     P 10             KEY_ := pressed key
     P 11         until KEY_ is (help) or (mode) or (escape) or (quit)
     P 12         or (up arrow) or (down arrow)
     P 13         case KEY_
     P 14         of KEY_HELP
 52A P 15             enqueue process-id [O-FRONT-ENQ]
     P 16             PROCESS_ID := HELP_ID
 52A P 17             enqueue process-id [O-FRONT-ENQ]
     P 18             CONTINUE_ := false
     P 19         of KEY_MODE
 52A P 20             enqueue process-id [O-FRONT-ENQ]
 51A P 21             exchange queue blocks [O-EXCH-Q-BLOCKS]
     P 22             CONTINUE_ := false
     P 23         of KEY_UP_ARROW
     P 24             DISPLACEMENT_ := 1
 44B P 25             change bird count [O-CHANGE-BIRD-COUNT]
 61B P 26             write bird count to window 1 [O-WRT-W1]
     P 27         of KEY_DN_ARROW
     P 28             DISPLACEMENT_ := -1
 44B P 29             change bird count [O-CHANGE-BIRD-COUNT]
 61B P 30             write bird count to window 1 [O-WRT-W1]
     P 31         of KEY_ESCAPE
     P 32             CONTINUE_ := false
     P 33         of KEY_QUIT
 67B P 34             quit program [P-QUIT-PROG]
     P 35         endcase
     P 36     endWhile
```

Clear a Found Bird Data Usage

```
------------C-  27A  -C-------------------------------------------------------------

     C  1     Public Variables:
     C  2     ...PROCESS_ID is the id of the current process.
     C  3     Private Variables:
     C  4     ...MENU_TITLE is the title of the current menu.
     C  5     ...TEXT_ITEM is the string text of the current menu item.
     C  6     ...TEXT_DISPLAY portion of text from the TEXT_ITEM for display on screen.
     C  7     ...CONTINUE_ is the boolean variable for the while loop.
     C  8     ...STATUS_ is the status report of procedures and functions.
     C  9     ...KEY_ is the character containing the pressed key.
     C 10     PDL Description:
     C 11     ...
     C 12
```

Clear a Found Bird Process [P-CLEAR-BIRD]

```
------------P-  27B  -P-------------------------------------------------------------

 61B P  1     write clear message to window 1 [O-WRT-W1]
 63A P  2     write 'Esc=abort;Sel=affirm' to window 4 [O-WRT-W4]
     P  3     repeat
     P  4         read keyboard
     P  5         KEY_ := pressed key
     P  6     until KEY_ is (help) or (mode) or (select) or (escape)
     P  7     case KEY_
     P  8     of KEY_HELP
 52A P  9         enqueue process-id [O-FRONT-ENQ]
     P 10         PROCESS_ID := HELP_ID
 52A P 11         enqueue process-id [O-FRONT-ENQ]
     P 12         CONTINUE_ := false
     P 13     of KEY_MODE
 52A P 14         enqueue process-id [O-FRONT-ENQ]
 51A P 15         exchange queue blocks [O-EXCH-Q-BLOCKS]
     P 16         CONTINUE_ := false
     P 17     of KEY_SELECT
 46B P 18         clear bird [O-CLEAR-BIRD]
     P 19         LAST_PROCESS_ID := PROCESS_ID
     P 20     of KEY_ESCAPE
     P 21         ...do nothing
     P 22     endcase
```

****

Text Menu Resource

****

Data Usage

----------C- 29A -C----------

```
C  1     Public Variable:
C  2     ...PROCESS_ID is the process-id of the current/invoked peocess
C  3     Private Variable:
C  4     ...TXT_ARY is the array containing the text of the menus
C  5     ...FLAG_ARY is the array containing select flag /process id of next process
C  6     ...FLAG_NO_ARY is the array containing flag numbers. This information
C  7     ......is used to indicate if the feature belongs to a group which does
C  8     ......not allow multiple selections.
C  9     ...NEXT_ID is id of the next menu to which the current item is pointing
C 10     ...KEY_ARY array of type record; it serves to conserve the context of each process.
C 11     ...PROCESS_ID is the id of current menu
C 12     ...MENU_TITLE is the title of the current menu
C 13     ...HELP is a string of text containig help information
C 14     ...C_POINTER is the pointer to the current menu item supported by TXT_ARY
C 15     ...F_POINTER is the pointer to the first menu item supportd by TXT_ARY
C 16     ...L_POINTER is the pointer to the last menu item supported by TXT_ARY
C 17     ...NO_OF_SELECTED_ATTRIBUTES stores the number of sellected attributes
C 18
```

Behavior

----------C- 29B -C----------

```
C  1     This resource supports the data structure of the text within the various menus as +++
C  2     The resource supports the operations needs to access the data structure.
C  3     List of Operations:
C  4
```

O-READ-MENU-FILE

----------P- 30A -P----------

```
P  1    ...1)O-READ-MENU-FILE () STATUS_
P  2    ...Data Usage:
P  3    ......External: none
P  4    ......Private:
P  5    .........Variable:
P  6    ............LINE_ is a line counter of data file TEXT_DAT.TXT
P  7    ............NO_OF_LINES is the number of lines in the data file
P  8    ............TMP_ARY is a temp array for data
P  9    ............C_POINTER ...points to the current entry in the menu
P 10    ............F_POINTER ...points to the first entry in the menu
P 11    ............L_POINTER ...points to the last entry in the menu
P 12    ...Behavior:
P 13    ......This operation retrieves the text structure from file, TEXT_DAT.TXT when pro+++
P 14    ...Interface:
P 15    ......Input: None
P 16    .....Output:
P 17    ........STATUS_ has the outcome of
P 18    ............0: operation succesfull
P 19    ............2: error in file format: must start with %
P 20    ............3: error if nile format: M expected after this %
P 21    ............4: error in file format: H expected after this %
P 22    ...PDL Description:
P 23    ...
P 24    line pointer := 1 [LINE_ :=1]
P 25    while end of file is not encountered
P 26        read first line from file TEXT_DAT.TXT and write to TEXT_
P 27        write TEXT_ to temproray array at position line pointer [LINE_]
P 28        increment line poiner [LINE_]
P 29    endwhile
P 30    record pointer := 0 [REC_:=0]
P 31    number of lines := (line pointer - 1) [NO_OF_LINES := LINE_ -1]
P 32    line pointer := 1 [LINE_:=1]
P 33    CONTINUE_ := true
P 34    while (LINE_ is less than NO_OF_LINES) and (CONTINUE is true)
P 35        increment record pointer [REC_]
P 36        number of records := record pointer [NO_O_REC:=REC_]
P 37        CHAR_ := first character in temp array [TMP_ARY], position:[LINE_]
P 38        if CHAR_ is not "%"
P 39            STATUS_ := 2 & CONTINUE_ := false
P 40        else
P 41            CHAR_ := second character in temp array [TMP_ARY], position:[LINE_]
P 42            if CHAR_ is not "M"
P 43                STATUS_ := 3 & CONTINUE_ := false
P 44            else
P 45                STATUS_ := 0 & increment line pointer [LINE_]
P 46                PROCESS_ID := text in temp array [TMP_ARY]: position:[LINE_]
P 47                increment line pointer [LINE_]
P 48                MENU_TITLE := text in temp array [TMP_ARY]: position:[LINE_]
P 49                increment line pointer [LINE_]
P 50                C_POINTER := value of text in temp array [TMP_ARY]: position:[LINE_]
P 51                increment line number [LINE_]
```

```
P 52                F_POINTER := line number [LINE_]
P 53                CHAR_ := first character in temp array [TMP_ARY]: position:[LINE_]
P 54                I_ := 1
P 55                while CHAR_ is not "%"
P 56                    text array [TXT_ARY]: position I_ := temprary array [TMP_ARY]: pos+++
P 57                    increment line number [LINE_]
P 58                    flag array [FLAG_ARY]: position I_ := temprary array [TMP_ARY]: po+++
P 59                    increment I_ & increment line number [LINE_]
P 60                    CHAR_ := first character of text in temprary array [TMP_ARY]: posi+++
P 61                endwhile
P 62                L_POINTER := (line number [LINE_] - F_POINTER) /2
P 63                CHAR_ := second character of text in temprary array [TMP_ARY]: position+++
P 64                if CHAR_ is not "H"
P 65                    STATUS_ := 4 & CONTINUE_ := false
P 66                else
P 67                    increment line number [LINE_]
P 68                    H_F_POINTER := line number [LINE_]
P 69                    if line number <= number of lines [LINE_<NO_OF_LINES]
P 70                        I_ := 1
P 71                        CHAR_ := first character of text in temprary array [TMP_ARY]: p+++
P 72                        repeat
P 73                            help array [HELP_ARY]: position I_ := temprary array [TMP_+++
P 74                            increment line number [LINE] & increment I_
P 75                            CHAR_ := first character of text in temprary array [TMP_ARY+++
P 76                        until (line number [LINE_] > number of lines [NO_OF_LINES]) or+++
P 77                        H_L_POINTER := line number [LINE_] - H_F_POINTR
P 78                    else
P 79                        H_F_POINTER := 0
P 80                        H_L_POINTER := 0
P 81                    endif
P 82                endif
P 83            endif
P 84        endif
P 85    endwhile
```

O-WRITE-MENU-FILE

----------p- 31A -p----------

```
P  1    ...2) O-WRITE-MENU-FILE ()
P  2    ...Data Usage:
P  3    ......External: none
P  4    ......Private:
P  5    .........Variable:
P  6    ............TXT_ARY text array of menus
P  7    ............FLAG_ARY flag array of attributes
P  8    ...Behavior:
P  9    ......This operation will write the selected attributes to the file
P 10    ......TEXT_DAT.TXT, before terminating program.
P 11    ...Interface:
P 12    ......Input: None
P 13    .....Output: None
P 14    ...PDL Description:
P 15    ...
P 16    record pointer [REC_] := 1
P 17    while record pointer [REC_] <= number of records [NO_OF_REC]
P 18        write "%H" to file TEXT_DAT.TXT
P 19        write PROCESS_ID to file
P 20        write MENU_TITLE to file
P 21        write C_POINTER to file
P 22        I_ := 1
P 23        while I_ <= C_POINTER
P 24            write text array [TEXT_ARY]: position I_ to file
P 25            write flag array [FLAG_ARY]: position I_ to file
P 26            increment I_
P 27        endwhile
P 28        write "%H" to file
P 29        I_ := 1
P 30        while I_ <= H_L_POINTER
P 31            write help array [HELP_ARY]: position I_ to file
P 32            increment I_
P 33        endwhile
P 34        increment record pointer [REC_]
P 35    endwhile
```

O-READ-MENU-TITLE

----------P- 32A -P----------

```
P  1     ...3) O-READ-MENU-TITLE () MENU_TITLE, STATUS_
P  2     ...Data Usage:
P  3     ......Extrenal: none
P  4     ......Private: none
P  5     ...Behavior:
P  6     ......This operation reads the title of the current menu
P  7     ...Interface:
P  8     ......Input: none
P  9     .....Output:
P 10     ........MENU_TITLE title of the current menu passed to the invoking process, throu+++
P 11     ........STATUS_; has the outcome of
P 12     ..........0 - successful
P 13     ..........1 - unsuccessful / requested process-id was not found in data file
P 14     ...PDL Dsecription:
P 15     ...
P 16     record pointer [REC_] := 1
P 17     CONTINUE_ := true
P 18     while CONTINUE_
P 19         if PROCESS_ID is the same as MENUS_ARY: position record pointer [REC_]
P 20             current record := record pointer [C_REC, REC_]
P 21             CONTINUE_ is false
P 22         else
P 23             increment record pointer [REC_]
P 24         endif
P 25     endwhile
P 26     if record pointer [REC_] > number of records [NO_OF_REC]
P 27         STATUS_ := 1
P 28     else
P 29         STATUS_ := 0
P 30         MENU_TITLE := title specified by MENUS_ARY: position current record [C_REC]
P 31     endif
```

O-READ-TXT-ITEM

----------P- 33A -P----------

```
P  1    ...4) O-READ-TXT-ITEM (DISPLACEMENT_) TEXT_ITEM, STATUS_
P  2    ...Data Usage:
P  3    ......External:
P  4    .........Variable:
P  5    ............REV_W4 is reverse video indicator of window 4 write operation performe+++
P  6    ............on this data item read operation performed in Screen Resource
P  7    ......Private:
P  8    .........Variable:
P  9    ............H_C_POINTER is the pointer to the current line of help
P 10    ...Behavior:
P 11    ......This operation reads the line of text at the current/above/below
P 12    ......text position. The relative displacement is defined by the value
P 13    ......of DISPLACEMENT_. This text is then written into the data item
P 14    ......TEXT_ITEM for access by scroll manager resource. The current
P 15    ......PROCESS_ID activitates the relevent pointers.
P 16    ...Interface:
P 17    ......Input:
P 18    .........DISPLACEMENT_ specifies the requested text is at current/above/below curr+++
P 19    ............0 - requested text is at current position
P 20    ............+1 - requested text is above current position
P 21    ............-1 - requested text is below current position
P 22    .....Output:
P 23    .........TEXT_ITEM returns the line of text from the data structure
P 24    .........STATUS_ has the outcome of
P 25    ............0 - successful
P 26    ...PDL Dsecription:
P 27    ...
P 28    if PROCESS_ID is ID_HELP
P 29        case DISPLACEMENT_
P 30        of 0
P 31            H_C_POINTER := 1
P 32        of -1
P 33            if H_C_POINTER <> H_L_POINTER
P 34                decrement H_C_POINTER
P 35            endif
P 36        of +1
P 37            if H_C_POINTER <> H_L_POINTER
P 38                increment H_C_POINTER
P 39            endif
P 40        endcase
P 41        TEXT_ITEM := line of text in help array [HELP_ARY]: position H_C_POINTER
P 42    else
P 43        case DISPLACEMENT_
P 44        of 0
P 45            ...do not change position
P 46        of -1
P 47            decrement C_POINTER
P 48            if C_POINTER < 1
P 49                C_POINTER := L_POINTER
P 50            endif
P 51        of +1
```

```
P 52            increment C_POINTER
P 53            if C_POINTER > L_POINTER
P 54                C_POINTER :=
P 55            endif
P 56        endcase
P 57        CHAR_ := character in flag array [FLAG_ARY]: position C_POINTER
P 58        if CHAR_ is '*'
P 59            reverse video of window 4 := true [REV_W4]
P 60        else
P 61            reverse video of window 4 := false [REV_W4]
P 62        endif
P 63        TEXT_ITEM := text in text array [TXT_ARY]: position C_POINTER
P 64    endif
P 65    STATUS_ := 0
```

O-UPDATE-ATTR-FLAG

----------p- 34A -p----------

```
P  1    ...5) O_UPDATE_ATTR_FLAG () STATUS_
P  2    ...Data Usage:
P  3    ......External: none
P  4    ......Private:
P  5    .........Variable:
P  6    ............ATTRIBUTE_FLAG is an indicator to show if attribute is selected or not
P  7    ............FLAG_NUMBER is a number that distinguishes a group of attributes
P  8    ...Behavior:
P  9    ......This operation has a binary function
P 10    ......A) for the selectable items this operation toggles the enable/disable flag
P 11    ......B) for drop down menus this operation enqueues the process-id of the invoked+++
P 12    ...Interface:
P 13    ......Input: none
P 14    ......Output:
P 15    .........STATUS_ has the outcome of
P 16    ............0 - selected menu item was of selectable type
P 17    ............1 - selected menu item was of a menu type
P 18    ...PDL Description:
P 19    ...
P 20    if present item is not pointing to another menu
P 21        read ATTRIBUTE_FLAG
P 22        case ATTRIBUTE_FLAG
P 23        of 0
P 24            read FLAG_NUMBER
P 25            repeat
P 26                read the menu for matching FLAG_NUMBER
P 27            until end of menu
P 28            if FLAG_NUMBER was found elsewhere and flag was set
P 29                ...only on flag out of a group with same flag number can be selected
P 30                release flag selection at found position
P 31            else
P 32                ...no restriction on flag selection
P 33            endif
P 34        of 1
P 35            if FLAG_NUMBER is negative
P 36                ...this is a flag that can not be deselected
P 37            else
P 38                ...this is a normal flag that can be deselected
P 39                deselect the flag
P 40            endif
P 41        endcase
P 42    else
P 43        ...this is not a flag type but the id of the next menu
P 44        PROCESS_ID := poiting process id
P 45    endif
```

O-READ-NO-OF-SELECTED-ATTRIBUTES

```
------------P-  35A  -P-----------------------------------------------------------

     P  1     ...6) O-READ-NO-OF-SELECTED-ATTRIBUTES () NO_OF_S_ATTR
     P  2     ...Data Usage:
     P  3     ......External: none
     P  4     ......Private:
     P  5     .........NO_OF_SELECTED_ATTRIBUTES
     P  6     ...Behavior: Retrieves the numner of selected attributes for display
     P  7     ...Inteface:
     P  8     ......Input: none
     P  9     ......Output:
     P 10     .........NO_OF_S_ATTR is the number of selected attributes passed out
     P 11     ...PDL Description:
     P 12     ...
     P 13     NO_OF_SELECTED_ATTRIBUTES := 0
     P 14     repeat
     P 15         de-select attribute
     P 16     until end of attribute list
     P 17     NO_OF_S_ATTR := NO_OF_SELECTED_ATTRIBUTES
```

O-CLEAR-ALL-ATTRIBUTES

```
------------P-  35B  -P-----------------------------------------------------------

     P  1     ...7) O-CLEAR-ALL-ATTRIBUTES ()
     P  2     ...Data Usage:
     P  3     ......Extrenal:
     P  4     ......Private:
     P  5     .........Variable:
     P  6     ............ATTRIBUTE_FLAG
     P  7     ...Behavior:
     P  8     ......This operation clears all the selected attributes.
     P  9     ...Interface:
     P 10     ......Input: none
     P 11     ......Output: none
     P 12     ...PDL Dsecription:
     P 13     ...
     P 14     repeat
     P 15         set ATTRIBUTE_FLAG to zero
     P 16     until end of attribute list
```

O-CLEAR-ATTRIBUTE

```
------------P-  36A  -P-----------------------------------------------------------

     P  1     ...8) O-CLEAR-ATTRIBUTE ()
     P  2     ...Data Usage:
     P  3     ......External: none
     P  4     ......Private:
     P  5     .........ATTRIBUTE_FLAG
     P  6     ...Behavior: clears the current attribute from the list selected sttributes
     P  7     ...Inteface:
     P  8     ......Input:
     P  9     ......Output:
     P 10     ...PDL Decription:
     P 11     ...
     P 12     clear ATTRIBUTE_FLAG at the current position
```

O-LIST-SELECTED-ATTRIBUTES

```
------------P-  36B  -P-----------------------------------------------------------

     P  1     ...9) O-LIST-SELECTED-ATTRIBUTES (DISPLACEMENT_) MENU_TITLE, MENU_ITEM
     P  2     ...Data Usage:
     P  3     ......Extrenal:
     P  4     ......Private:
     P  5     .........DISPLACEMENT_ specifies the requested text is at current/above/below curr+++
     P  6     ...Behavior: retrieves the list of selected attributes
     P  7     ...Interface:
     P  8     ......Input: DISPLACEMENT_ defines the direction of search
     P  9     ......Output: MENU_TITLE, MENU_ITEM
     P 10     ...PDL Dsecription:
     P 11     ...
     P 12     case DISPLACEMENT_
     P 13     of 0 or 1 or -1
     P 14         case DISPLACEMENT_
     P 15         of 0
     P 16             ...first time listing
     P 17             set DISPLACEMENT_ to 1
     P 18         of 1 OR -1
     P 19             continue up or down the list of selected attributes
     P 20         endcase
     P 21         move in the direction of DISPLACEMENT_
     P 22     endcase
```

O-READ-LIST-FOUND-METHOD

----------p- 37A -p----------

```
P  1    ...10) O-READ-LIST-FOUND-METHOD () LIST_FOUND_METHOD
P  2    ...Data Usage:
P  3    ......External: none
P  4    ......Private:
P  5    .........LIST_FOUND_METHOD is the accessable function in List Found Birds
P  6    ...Behavior: retrieves the accessable function in List Found Birds
P  7    ...Inteface:
P  8    ......Input: DISPLACEMENT_
P  9    ......Output:
P 10    ...PDL Decription:
P 11    ...
P 12    if Count Birds is selected
P 13        ...enable counting in List Found Birds
P 14        LIST_FOUND_METHOD := 1
P 15    else
P 16        ...enable bird extraction from the list
P 17        LIST_FOUND_METHOD := 2
P 18    endif
```

O-READ-LIST-MATCHED-METHOD

----------P- 38A -P----------

```
P  1    ...11) O-READ-LIST-MATCHED-METHOD () LIST_MATCHED_METHOD
P  2    ...Data Usage:
P  3    ......External: none
P  4    ......Private:
P  5    .........LIST_MATCHED_METHOD is the accessable function in List Matched Birds
P  6    ...Behavior: retrieves the accessable function in List Matched Birds
P  7    ...Inteface:
P  8    ......Input: DISPLACEMENT_
P  9    ......Output:
P 10    ...PDL Decription:
P 11    ...
P 12    if Bird Count is selected
P 13        ...enable bird count
P 14        LIST_MATCHED_METHOD := 1
P 15    else
P 16        if Bird Information is selected
P 17            ...enable bird information
P 18            LIST_MATCHED_METHOD := 2
P 19        else
P 20            if Bird Call is selected
P 21                ...enable bird call
P 22                LIST_MATCHED_METHOD := 3
P 23            else
P 24                ...enable bird picture
P 25                LIST_MATCHED_METHOD := 4
P 26            endif
P 27        endif
P 28    endif
```

O-FILL-FILTER

----------P- 38B -P----------

```
P  1    ...12) O-FILL-FILTER ()
P  2    ...Data Usage:
P  3    ......Extrenal: none
P  4    ......Private:
P  5    .........SELECTED_ATTR is the selected attribute
P  6    .........SELECTED_ATTR_SET is the set of selected attributes
P  7    ...Behavior: compiles a search filter based on the selected attributes other than +++
P  8    ......This operation intializes the search filter with selected items
P  9    ...Interface:
P 10    ......Input: none
P 11    ......Output: none
P 12    ...PDL Description:
P 13    ...
P 14    SELECTED_ATTR_SET is null
P 15    repeat
P 16        add SELECTED_ATTR to SELECTED_ATTR_SET
P 17    until all attributes are scanned
```

****

Bird Resource

****

Data Usage

```
------------C-  40A  -C-------------------------------------------------------------

     C  1     Private Constant:
     C  2     External
     C  3     ...Variable:
     C  4     ......DATA_ARY
     C  5     ......FILTER_ARY
     C  6     ......FAMILY_
     C  7     Public Variable:
     C  8     ...PROCESS_ID is the id of the current menu.
     C  9     ...BIRD_FAMILY is the family to which the current bird belongs
     C 10     ...BIRD_NAME is the name of the current bird
     C 11     ...BIRD_COUNT is the count number of the current bird
     C 12     Private Variable:
     C 13     ...STRATEGY_ is the search and compare strategy of selected attributes
     C 14     ......against the bird attributes; four possible values
     C 15     ......A- no attribute is selected; compile all birds for display
     C 16     ......B- only bird family is selected
     C 17     ......C- a f     ype attribute is selected; bird family is not selected
     C 18     ......D- both  .mily and flag type attributes have been selected
     C 19     ......NO_OF_MATCHED_BIRDS is the number of birds matching the selected attributes
     C 20     ......MATCH_LIST_NO indentifies which of the four possible functions of
     C 21     .........a)bird count, b)bird information, c)bird image or d)bird call have been e+++
     C 22     ......NO_FOUND_BIRDS is the number of birds with a bird count greater than zero
     C 23     ......FOUND_LIST_NO identifies wheter extraction or bird count are enabled during +++
     C 24     ......DATA_ARY is of the type record; this is the bird database
     C 25     .........BIR_NAME is the name bird
     C 26     .........FAMILY_ is the family of the bird
     C 27     .........SAOS_ is the South African bird number
     C 28     .........PR_CODE is the bird habitat code
     C 29     .........BIRD_ATTR contains the flag type attributes of the bird
     C 30     .........BIRD_COUNT is the count number of the bird
     C 31     .........MATCH_ indicates if the current record has matched attribute specificatio+++
     C 32     ......BIRD_REC is the DATA_ARY record pointer
     C 33     ......
```

Behavior

```
------------C-  40B  -C-------------------------------------------------------------

     C  1     This resource furnishes the bird attributes and information data structure.
     C  2     Operations of the resource allows reading of bird data and scanning
     C  3     through it.
     C  4
```

O-READ-BIRD-DATA

----------P- 41A -P----------

```
P  1    ...1) O-READ-BIRD-DATA ()
P  2    ...Data Usage:
P  3    ......Private:
P  4    .........Constant:
P  5    ............DATA_FILE
P  6    .........Variable:
P  7    ............BIRD_NAME
P  8    ............BIRD_FAMILY
P  9    ............BIRD_SAOS
P 10    ............BIRD_FLAG
P 11    ...Behavior: reads the bird database from disc file
P 12    ......This operation reads the bird attribute and info data from file
P 13    ...Inteface:
P 14    ......Input: none
P 15    ......Output: none
P 16    ...PDL Decription:
P 17    ...
P 18    while not end of file
P 19        read BIRD_FAMILY
P 20        read BIRD_NAME
P 21        read BIRD_INFO: BIRD_SAOS
P 22    endwhile
```

O-SCAN

----------P- 42A -P----------

```
P  1    ...2) O-SCAN (DISPLACEMENT_) BIRD_NAME, BIRD_INFO, BIRD_REC
P  2    ...Data Usage:
P  3    ......External: none
P  4    ......Private:
P  5    .........Variable:
P  6    ............STRATEGY_ is the scanning strategy
P  7    ............MATCH_ indicates if a bird has matched the set of selected attributes
P  8    ............CONTINUE_ loop condition
P  9    ............NO_OF_MATCHED_BIRDS
P 10    ...Behavior: scans the database for birds tha macth the selected att butes
P 11    ......Birds matching the selected attributes will be found and displayed
P 12    ...Inteface:
P 13    ......Input: DISPLACEMENT_
P 14    ......Output:
P 15    .........BIRD_NAME
P 16    .........BIRD_INFO
P 17    .........BIRD_REC
P 18    ...PDL Decription:
P 19    ...
P 20    NO_OF_MATCHED_BIRDS :=0
P 21    set default listing strateg to display all, STRATEGY+ :=4
P 22    if selected family is not null and SELECTED_ATTR_SET is null
P 23        ...family search only
P 24        STRATEGY_ := 1
P 25    endif
P 26    if selected family is null and SELECTED_ATTR_SET is not null
P 27        ...filter search on other attributes
P 28        STRATEGY_ := 2
P 29    endif
P 30    if selected family is not null and SELECTED_ATTR_SET is not null
P 31        ...family anf filter search
P 32        STRATEGY_ := 3
P 33    endif
P 34    case STRATEGY_
P 35    of 1
P 36        ...family search only
P 37        BIRD_REC := 1
P 38        while BIRD_REC < NO_OF_BIRDS
P 39            do a family search
P 40            if there was a match
P 41                increment NO_OF_MATCHED_BIRDS
P 42            endif
P 43        endwhile
P 44    of 2
P 45        ...filter search only
P 46        do a search on filter spec
P 47        if there was a match
P 48            increment NO_OF_MATCHED_BIRDS
P 49        endif
P 50    of 3
P 51        ...family and filter search
```

```
P 52        BIRD_REC := 1
P 53        while BIRD_REC < NO_OF_BIRDS
P 54            do a family search
P 55            if there was a match
P 56                do a search on filter
P 57                if there was a match
P 58                    increment NO_OF_MATCHED_BIRDS
P 59                endif
P 60            endif
P 61        endwhile
P 62    of 4
P 63        ...display all birds
P 64        NO_OF_MATCHED_BIRDS := NO_OF_BIRDS
P 65    endcase
```

O-LIST-MATCHED-BIRDS

----------P- 43A -P----------

```
P  1    ...3) O-LIST-MATCHED-BIRDS (DISPLACEMENT_)
P  2    ...Data Usage:
P  3    ......External: none
P  4    ......Private:
P  5    .........BIRD_NAME
P  6    .........BIRD_FAMILY
P  7    ...Behavior: retrieves the list of birds that match the selected attributes
P  8    ...Inteface:
P  9    ......Input: DISPLACEMENT_
P 10    ......Output: none
P 11    ...PDL Decription:
P 12    ...
P 13    case DISPLACEMENT_
P 14    of 0
P 15        get the first item in the list
P 16    of -1
P 17        get the previous item in the list
P 18    of +1
P 19        get the next item in the list
P 20    endcase
P 21    update BIRD_NAME
P 22    update BIRD_FAMILY
```

O-READ-COUNT-FILE

----------P- 43B -P----------

```
P  1    ...4) O-READ-COUNT-FILE ()
P  2    ...Data Usage:
P  3    ......External: none
P  4    ......Private:
P  5    .........NO_OF_BIRDS
P  6    ...Behavior: reads the count file from disc
P  7    ...Inteface:
P  8    ......Input: DISPLACEMENT_
P  9    ......Output:
P 10    ...PDL Decription:
P 11    ...
P 12    while NO_OF_BIRDS is not reached
P 13        read bird counts from file
P 14    endwhile
```

O-WRITE-COUNT-FILE

----------P- 44A -P----------

```
P  1    ...5) O-WRITE-COUNT-FILE ()
P  2    ...Data Usage:
P  3    ......External: none
P  4    ......Private:
P  5    .........NO_OF_BIRDS
P  6    ...Behavior: writes the bird count information to disc file
P  7    ...Inteface:
P  8    ......Input: DISPLACEMENT_
P  9    ......Output:
P 10    ...PDL Decription:
P 11    ...
P 12    while NO_OF_BIRDS is not reached
P 13        write bird counts to file
P 14    endwhile
```

O-CHANGE-BIRD-COUNT

----------P- 44B -P----------

```
P  1    ...6) O-CHANGE-BIRD-COUNT (DISPLACEMENT_)
P  2    ...Data Usage:
P  3    ......External: none
P  4    ......Private:
P  5    .........BIRD_COUNT
P  6    ...Behavior: modifies the count of the current bird
P  7    ...Inteface:
P  8    ......Input:
P  9    .........DISPLACEMENT_ indicates bird count is to be incremented of decremented
P 10    ............-1 count down
P 11    ............+1 count up
P 12    ......Output:
P 13    ...PDL Decription:
P 14    ...
P 15    case DISPLACEMENT_
P 16    of -1
P 17        increment BIRD_COUNT
P 18        if BIRD_COUNT is < 0
P 19            set BIRD_COUNT to 0
P 20        endif
P 21    of +1
P 22        decrement BIRD_COUNT
P 23        if BIRD_COUNT is at max allowable value
P 24            set BIRD_COUNT to max allowable value
P 25        endif
P 26    endcase
```

O-READ-NO-OF-FOUND-BIRDS

----------P- 45A -P----------

```
P  1    ...7) O-READ-NO-OF-FOUND-BIRDS () NO_OF_FOUND_BIRDS_PRIVATE
P  2    ...Data Usage:
P  3    ......External: none
P  4    ......Private:
P  5    .........NO_OF_FOUND_BIRDS
P  6    ...Behavior: retrieves the number of found birds for display
P  7    ...Inteface:
P  8    ......Input: none
P  9    ......Output: NO_OF_FOUND_BIRDS_PRIVATE is the number of found birds
P 10    ...PDL Decription:
P 11    ...
P 12    NO_OF_FOUND_BIRDS :=0
P 13    while NO_OF_BIRDS is not reached
P 14        read BIRD_COUNT
P 15        if BIRD_COUNT <> 0
P 16            increment NO_OF_FOUND_BIRDS
P 17        endif
P 18    endwhile
P 19    NO_OF_FOUND_BIRDS_PRIVATE := NO_OF_FOUND_BIRDS
```

O-LIST-FOUND-BIRDS

----------P- 45B -P----------

```
P  1    ...8) O-LIST-FOUND-BIRDS (DISPLACEMENT_)
P  2    ...Data Usage:
P  3    ......External: none
P  4    ......Private:
P  5    ........BIRD_NAME
P  6    ........BIRD_FAMILY
P  7    ...Behavior: retrieves the list of found birds
P  8    ...Inteface:
P  9    ......Input: DISPLACEMENT_
P 10    ......Output:
P 11    ...PDL Decription:
P 12    ...
P 13    case DISPLACEMENT_
P 14    of 0
P 15        get the first item in the list
P 16    of -1
P 17        get the previous item in the list
P 18    of +1
P 19        get the next item in the list .
P 20    endcase
P 21    update BIRD_NAME
P 22    update BIRD_FAMILY
```

O-CLEAR-FOUND-BIRDS

```
------------P-  46A  -P-------------------------------------------------------

     P  1     ...9) O-CLEAR-FOUND-BIRDS ()
     P  2     ...Data Usage:
     P  3     ......External: none
     P  4     ......Private:
     P  5     .........BIRD_COUNT
     P  6     .........NO_OF_BIRDS
     P  7     ...Behavior: clears all found birds from the list of found birds
     P  8     ...Inteface:
     P  9     ......Input: DISPLACEMENT_
     P 10     ......Output:
     P 11     ...PDL Decription:
     P 12     ...
     P 13     while NO_OF_BIRDS is not reached
     P 14         set BIRD_COUNT to zero
     P 15     endwhile
```

O-CLEAR-BIRD

```
------------P-  46B  -P-------------------------------------------------------

     P  1     ...10) O-CLEAR-BIRD ()
     P  2     ...Data Usage:
     P  3     ......External: none
     P  4     ......Private:
     P  5     .........BIRD_COUNT
     P  6     ...Behavior: clears the current bird from the list of found birds
     P  7     ...Inteface:
     P  8     ......Input: DISPLACEMENT_
     P  9     ......Output:
     P 10     ...PDL Decription:
     P 11     ...
     P 12     set the current BIRD_COUNT to zero
```

O-READ-NO-OF-MATCHED-BIRDS

----------P- 47A -P----------

```
P  1    ...11) O-READ-NO-OF-MATCHED-BIRDS () NO_OF_FOUND_BIRDS_PRIVATE
P  2    ...Data Usage:
P  3    ......External: none
P  4    ......Private:
P  5    .........NO_OF_MATCHED_BIRDS
P  6    ...Behavior: retrieves the number of matched birds for display
P  7    ...Inteface:
P  8    ......Input: DISPLACEMENT_
P  9    ......Output:
P 10    ...PDL Decription:
P 11    ...
P 12    NO_OF_MATCHED_BIRDS_PRIVATE := NO_OF_MATCHED_BIRDS
```

****

Queue Resource

****

Data Usage

```
------------C-  49A  -C-------------------------------------------------------

     C  1     Public:
     C  2     ...Variable:
     C  3     ......PROCESS_ID is the id of the current menu.
     C  4     Private:
     C  5     ...Variable:
     C  6     ......I_ is aloop counter
     C  7     ......STATUS_ is the status report of procedures and functions.
     C  8     ......Q_ARY is the array emulating the queue
     C  9     ......Q_TMP_ARY is the temp queue array, used for context switching
     C 10     ......Q_MIDL is the marker seperating the find and attribiue activites
     C 11     ......Q_TAIL is the pointer marking the position of the last item in the queue
     C 12     ...Constant:
     C 13     ......Q_SIZE is the size of the queue set at 10
     C 14
```

Behavior

```
------------C-  49B  -C-------------------------------------------------------

     C  1     This resource provides the facility of queuing the main activities
     C  2     namely; ATTRIBUTE and FIND mode
     C  3
```

O-READ-Q-FILE

```
------------P-  50A  -P-------------------------------------------------------

     P  1     ...1) O-READ-Q-FILE ()
     P  2     ...Data Usage:
     P  3     ......External: none
     P  4     ......Private: none
     P  5     ...Behavior:
     P  6     ......this operation restrieves the queue information from file Q_DAT.TXT
     P  7     ......when the main program is first run
     P  8     ...Inteface:
     P  9     ......Input: none
     P 10     ......Output: none
     P 11     ...PDL Decription:
     P 12     ...
     P 13     read text from file, Q_DAT.TXT into TEXT_
     P 14     queue middle pointer [Q_MIDL]:= value of TEXT_
     P 15     read text from file into TEXT_
     P 16     queue tail pointer [Q_TAIL] := TEXT_
     P 17     I_ := 1
     P 18     while I_ <= queue tail pointer [Q_TAIL]
     P 19         queue array [Q_ARY] : position I_ := text from file
     P 20         increment I_
     P 21     endwhile
```

O-WRITE-Q-FILE

```
------------P-  50B  -P-------------------------------------------------------

     P  1     ...2) O-WRITE-Q-FILE ()
     P  2     ...Data Usage:
     P  3     ......External: none
     P  4     ......Private: none
     P  5     ...Behavior:
     P  6     ......This operation will write the queue data into file Q_DAT.TXT
     P  7     ......when quiting the program
     P  8     ...Inteface:
     P  9     ......Input: none
     P 10     ......Output: none
     P 11     ...PDL Description
     P 12     ...
     P 13     TEXT_ := queue middle pointer [Q_MIDL]
     P 14     write TEXT_ to file
     P 15     TEXT_ := queue tail pointer [Q_TAIL]
     P 16     write TEXT_ to file
     P 17     I_ := 1
     P 18     while I_ <= queue tail [Q_TAIL]
     P 19         write queue array [Q_ARY]: position I_ to file
     P 20         increment I_
     P 21     endwhile
```

Q-EXCH-Q-BLOCKS

----------P- 51A -P----------

```
P  1    ...3) EXCHANGE-Q-BLOCKS
P  2    ...Data Usage:
P  3    ......External: none
P  4    ......Private: none
P  5    ...Behavior:
P  6    ......this operation will swap the segments of queue corresponding to the
P  7    ......Attribute mode with segment corresponding to the Find mode
P  8    ......Thus putting the relevent process-id at the head of the queue
P  9    ...Inteface:
P 10    ......Input: none
P 11    ......Output: none
P 12    ...PDL Dsecription:
P 13    ...
P 14    I_ := 1
P 15    while I_ <= queue middle [Q_MIDL]
P 16        queue temp array [Q_TMP_ARY]: position I_ := queue array [Q_ARY]: position I_
P 17        increment I_
P 18    endwhile
P 19    I_ := 1
P 20    while I_ <= (queue tail [Q_TAIL]) - (queue middle [Q_MIDL])
P 21        queue array [Q_ARY]: position I_ := queue array [Q_ARY]:position (queue middle+++
P 22        increment I_
P 23    endwhile
P 24    I_ := 1
P 25    while I_ <= queue middle [Q_MIDL]
P 26        queue array [Q_ARY]: position Q_TAIL - Q_MIDL + I_ := Q_TMP_ARY position I_
P 27        increment I_
P 28    endwhile
P 29    Q_MIDL := Q_TAIL - Q_MIDL
```

O-FRONT-ENQ

----------P- 52A -P----------

```
P  1    ...4) O-FRONT-ENQ (), STATUS_
P  2    ...Data Usage:
P  3    ......External:
P  4    ......Private:
P  5    .........Q_ARY is the queue of waiting processes
P  6    .........Q_TAIL is the end of queue pointer
P  7    .........Q_MIDL is the pointer seperating the attribute processes from the find pr+++
P  8    .........PROCESS_ID is the id of the process at the head of the queue
P  9    ...Behavior:
P 10    ......This operation will enqueue the PROCESS_ID at the head of the queue
P 11    ...Interface:
P 12    ......Input: none
P 13    ......Ouput: STATUS_ has the outcome of
P 14    .........0 - successful
P 15    .........1 - unsuccessful / queue is full
P 16    ...PDL Dsecription:
P 17    ...
P 18    if Q_TAIL is equal to size of queue [Q_SIZE]
P 19        STATUS_ := 1
P 20    else
P 21        if Q_TAIL is 0
P 22            queue array [Q_ARY]: position 1 := PROCESS_ID
P 23            Q_MIDL := 1
P 24            Q_TAIL := 1
P 25        else
P 26            I_ := Q_TAIL
P 27            while I_ >= 1
P 28                queue array [Q_ARY]: position I_ +1 := queue array [Q_ARY]: position I_
P 29                increment I_
P 30            endwhile
P 31            queue array [Q_ARY]: position 1 := PROCESS_ID
P 32            increment Q_TAIL
P 33            INCREMENT Q_MIDL
P 34            STATUS_ := 0
P 35        endif
P 36    endif
```

Q-DEQ

----------P- 53A -P----------

```
P  1    ...5) Q-DEQ () STATUS_
P  2    ...Data Usage:
P  3    ......External:
P  4    ......Private:
P  5    .........Q_ARY is the queue of waiting processes
P  6    .........Q_TAIL is the end of queue pointer
P  7    .........Q_MIDL is the pointer seperating the attribute processes from the find pr+++
P  8    ...Behavior:
P  9    ......This operation will consume the item at the head of the queue, and
P 10    ......writes it to the PROCESS_ID
P 11    ...Interface:
P 12    ......STATUS_: has the outcome of
P 13    .........0 - successful
P 14    .........1 - unsuccessful / queue empty
P 15    ...PDL Dsecription:
P 16    ...
P 17    decrement Q_TAIL
P 18    decrement Q_MIDL
P 19    if Q_MIDL is less than 0
P 20        Q_MIDL := 0
P 21    endif
P 22    if Q_TAIL is less than 0
P 23        ...queue is empty
P 24        Q_TAIL := 0
P 25        STATUS_ ;= 1
P 26    else
P 27        PROCESS_ID := queue array [Q_ARY]: position 1
P 28        STATUS_ := 0
P 29        I_ := 2
P 30        while I_ <= Q_TAIL+1
P 31            queue array [Q_ARY]: position I_-1 := queue array [Q_ARY]: position I_
P 32            increment I_
P 33        endwhile
P 34    endif
```

****

Scroll Manager Resource

****

Data Usage

```
------------C-  55A  -C-------------------------------------------------------

     C  1     Public:
     C  2     ...Variable:
     C  3     ......TEXT_ITEM ...is the string text of the current menu item
     C  4     ......REV_W4 is the reverse video indicator of window 4
     C  5     ......W1_SIZE is the size of window 1
     C  6     ......W4_SIZE is the size of window 4
     C  7     ......GUIDE_ARROW_R
     C  8     ......GUIDE_ARROW_L
     C  9     ......W_NO is the window number of the active window, either 1 or 4
     C 10     Private:
     C 11     ...Variables:
     C 12     ......W_L_POINTER .is the window left pointer.
     C 13     ......W_R_POINTER .is the window right pointer.
     C 14     ......TEXT_DISPLAY is the portion of text from the TEXT_ITEM to be displayed on th+++
     C 15     ......TEXT_LENGTH .is the length of text in TEXT_ITEM.
     C 16
```

Behavior

```
------------C-  55B  -C-------------------------------------------------------

     C  1     This resource facilitates the function of word scrolling within the
     C  2     current line of text, which is stored in the variable TEXT_ITEM.
     C  3     This is achieved by displaying a window of this text.
     C  4     List of operations:
     C  5
```

O-INITIAL-TEXT

----------P- 56A -P----------

```
P  1    ...1) O-INITIAL-TEXT (TEXT_ITEM), TEXT_DISPLAY
P  2    ...Data Usage:
P  3    ......External:
P  4    .........TEXT_ITEM
P  5    ......Private:
P  6    .........W_L_POINTER left pointer of viewing window
P  7    .........W_R_POINTER right pointer of viewing window
P  8    ...Behavior:
P  9    ......This operation extracts the first x characters in the text
P 10    ......contained in TEXT_ITEM and writes it to TEXT_DISPLAY
P 11    ...Interface:
P 12    ......Input:
P 13    .........TEXT_ITEM contains the current line of text in the data structure
P 14    .........TEXT_DISPLAY contains a substring of text within TEXT_ITEM, for
P 15    ........................displaying on the screen
P 16    ......Output:
P 17    ......TEXT_DISPLAY is the truncated text from the current line to be displayed
P 18    ...................on the current window
P 19    ...PDL Dsecription:
P 20    ...
P 21    W_L_POINTER := 0
P 22    W_R_POINTER := W_SIZE+1
P 23    TEXT_DISPLAY := TEXT_ITEM: from W_L_POINTER+1 to W_R_POINTER-1
P 24    if TEXT_LENGTH <= W_SIZE
P 25        GUIDE_ARROW_R := false
P 26    else
P 27        GUIDE_ARROW_R := true
P 28    endif
P 29    GUIDE_ARROW_L := false
```

O-MOVE-TEXT-LEFT

----------P- 57A -P----------

```
P  1    ...2) O-MOVE-TEXT-LEFT (TEXT_ITEM), TEXT_DISPLAY
P  2    ...Behavior:
P  3    ......This operation moves the displayed window of the text in MENU_ITEM one word +++
P  4    ...Interface:
P  5    ......MENU_ITEM; contains the current line of text in the data structure
P  6    ......W_R_POINTER; right pointer of the text window
P  7    ......TEXT_DISPLAY contains a substring of text within MENU_ITEM, for displaying o+++
P  8    ...PDL Description:
P  9    ...
P 10    if first character of TEXT_ITEM at W_L_POINTER is blank
P 11        while first character of TEXT_ITEM at W_L_POINTER is blank
P 12            decerement W_L_POINTER
P 13        endwhile
P 14    endif
P 15    CONTINUE_ := true
P 16    while first character of TEXT_ITEM at W_L_POINTER is not blank and CONTINUE_ is tr+++
P 17        decerement W_L_POINTER
P 18        if W_L_POINTER <= 0
P 19            W_L_POINTER :=0
P 20            CONTINUE_ := false
P 21        endif
P 22    endwhile
P 23    W_R_POINTER := W_L_POINTER + W_SIZE+1
P 24    case W_L_POINTER
P 25    of 0
P 26        GUIDE_ARROW_L := false
P 27        if length of TEXT_ITEM <= W_SIZE
P 28            GUIDE_ARROW_R := false
P 29        else
P 30            GUIDE_ARROW_R := true
P 31        endif
P 32    default
P 33        GUIDE_ARROW_L := true
P 34        GUIDE_ARROW_R := true
P 35    endcase
P 36    TEXT_DISPLAY := portion of TEXT_ITEM from W_L_POINTER +1; of length W_SIZE
```

VE-TEXT-RIGHT

----------P- 58A -P----------

```
P  1    ...3) O-MOVE-TEXT-RIGHT (MENU_ITEM), TEXT_DISPLAY
P  2    ...Behavior:
P  3    ......This operation moves the displayed window of the text in MENU_ITEM one word +++
P  4    ...Interface:
P  5    ......MENU_ITEM; contains the current line of text in the data structure
P  6    ......W_R_POINTER; right pointer of the text window
P  7    ......TEXT_DISPLAY; contains a substring of text within MENU_ITEM, for dislpaying +++
P  8    ...PDL Description:
P  9    ...
P 10    if first character of TEXT_ITEM at W_R_POINTER is blank
P 11        while first character of TEXT_ITEM at W_R_POINTER is blank
P 12            increment W_R_POINTER
P 13        endwhile
P 14    endif
P 15    CONTINUE_ := true
P 16    while first character of TEXT_ITEM at W_R_POINTER is not blank and CONTINUE_ is tr+++
P 17        increment W_R_POINTER
P 18        if length of TEXT_ITEM+1 <= W_SIZE
P 19            W_R_POINTER := W_SIZE +1
P 20            CONTINUE_ := false
P 21        else
P 22            if length of TEXT_ITEM+1 < W_R_POINTER
P 23                W_R_POINTER := length of TEXT_ITEM +1
P 24                CONTINUE_ := false
P 25            endif
P 26        endif
P 27    endwhile
P 28    W_L_POINTER := W_R_POINTER - W_SIZE -1
P 29    if W_L_POINTER <>0
P 30        if W_R_POINTER := lenght of TEXT_ITEM+1
P 31            GUIDE_ARROW_R := false
P 32            GUIDE_ARROW_L := true
P 33        else
P 34            GUIDE_ARROW_R := true
P 35            GUIDE_ARROW_L := true
P 36        endif
P 37    endif
P 38    TEXT_DISPLAY := portion of TEXT_ITEM from W_L_POINTER+1; of length W_SIZE
```

****

Screen Resource

****

Data Usage

----------C- 60A -C----------

```
C  1     Public:
C  2     ...Variable:
C  3     ......TEXT_DISPLAY is the display text passed in to be displayed on screen
C  4     ......REV_W4
C  5     ......W1_SIZE
C  6     ......W4_SIZE
C  7     ......GUIDE_ARROW_L
C  8     ......GUIDE_ARROW_R
C  9     ......
C 10     Private:
C 11     ...Variables:
C 12     ......W1_X horizontal position of left most corner of window 1
C 13     ......W1_Y vertical position of left most corner of window 1
C 14     ......W4_X horizontal position of left most corner of window 4
C 15     ......W4_Y vertical position of left most corner of window 4
C 16     ...Constant:
C 17     ......W2_X horizontal position of left most corner of window 2
C 18     ......W2_Y vertical position of left most corner of window 2
C 19     ......W3_X horizontal position of left most corner of window 3
C 20     ......W3_Y vertical position of left most corner of window 3
C 21     ......W5_X horizontal position of left most corner of window 5
C 22     ......W5_Y vertical position of left most corner of window 5
C 23     ......W2_SIZE length of window 2
C 24     ......W3_SIZE length of window 3
C 25     ......W5_SIZE length of window 5
C 26
```

Behavior

----------C- 60B -C----------

```
C  1     This resource provides the screen emulation of the bird identifier in
C  2     addition to the writng and clearing operations of the different windows
C  3     of the screen.
C  4     List of Operations:
C  5
```

O-SCREEN-FORMAT

```
-------------P-  61A  -P-----------------------------------------------------------

     P  1     ...1) O-SCREEN-FORMAT
     P  2     ...Data Usage:
     P  3     ......External:
     P  4     ......Private:
     P  5     ...Behavior:
     P  6     ......This operation provides the screen display and the keypads of the
     P  7     ......identifier.
     P  8     ...Interface:
     P  9     ......Input: none
     P 10     ......Output: none
     P 11     ...PDL Description:
     P 12     ...
     P 13     position keypads on the screen [O-PUT-KEYPAD]
```

O-WRT-W1

```
-------------P-  61B  -P-----------------------------------------------------------

     P  1     ...3) O-WRT-W1 (TEXT_DISPLAY)
     P  2     ...Data Usage:
     P  3     ......External:
     P  4     .........Variable:
     P  5     .........TEXT_DISPLAY
     P  6     ......Private: none
     P  7     ...Behavior:
     P  8     ......This operation writes to window 1
     P  9     ...Interface:
     P 10     ......Input:
     P 11     ......TEXT_DISPLAY
     P 12     ......Output: none
     P 13     ...PDL Description:
     P 14     ...
 64A P 15     clear window 1 [O-CLR-W1]
     P 16     position cursor at W1_X, W1_Y
     P 17     write TEXT_DISPLAY to window 1
```

O-WRT-W2

```
-------------P-  62A  -P-------------------------------------------------------

     P  1     ...4)O-WRT-W2 ()
     P  2     ...Data Usage:
     P  3     ......External:
     P  4     ......Private:
     P  5     ...Behavior:
     P  6     ...Interface:
     P  7     ......Input: none
     P  8     ......Output: none
     P  9     ...PDL Description:
     P 10     ...
     P 11     case GUIDE_ARROW_L
     P 12     of false
 64B P 13         clear window 2 [-O-CLR-W2]
     P 14     of true
     P 15         position cursor at W2_X, W2_Y
     P 16.        write TEXT_DISPLAY to window 2
     P 17     endcase
```

O-WRT-W3

```
-------------P-  62B  -P-------------------------------------------------------

     P  1     ...5)O-WRT-W3 ()
     P  2     ...Data Usage:
     P  3     ......External:
     P  4     ......Private:
     P  5     ...Behavior:
     P  6     ...Interface:
     P  7     ......Input: none
     P  8     ......Output: none
     P  9     ...PDL Description:
     P 10     ...
     P 11     case GUIDE_ARROW_R
     P 12     of false
 64C P 13         clear window 3 [-O-CLR-W3]
     P 14     of true
     P 15         position cursor at W3_X, W3_Y
     P 16         write TEXT_DISPLAY to window 3
     P 17     endcase
```

O-WRT-W4

```
------------P-  63A  -P-------------------------------------------------------------

    P  1     ...6)O-WRT-W4 (TEXT_DISPLAY)
    P  2     ...Data Usage:
    P  3     ......External:
    P  4     ......Private:
    P  5     ...Behavior:
    P  6     ...Interface:
    P  7     ......Input: TEXT_DISPLAY
    P  8     ......Output: none
    P  9     ...PDL Description:
    P 10     ...
    P 11     if REV_W4 is true
    P 12         set foreground and background color to normal
    P 13     else
    P 14         set foreground and background color to reverse
    P 15     endif
 65A P 16    clear window 4 [O-CLR-W4]
    P 17     position cursor at W4_X, W4_Y
    P 18     write TEXT_DISPLAY to window 4
    P 19     set foreground and background color to normal
```

O-WRT-W5

```
------------P-  638  -P-------------------------------------------------------------

    P  1     ...7)O-WRT-W5 (W5_TEXT)
    P  2     ...Data Usage:
    P  3     ......External:
    P  4     ......Private:
    P  5     ...Behavior:
    P  6     ...Interface:
    P  7     ......Input: none
    P  8     ......Output: none
    P  9     ...PDL Description:
    P 10     ...
 658 P 11    clear window 5 [O-CLR-W5]
    P 12     position cursor at W4_X, W4_Y
    P 13     write to W5_TEXT to window 5
```

O-CLR-W1

```
-------------P-  64A  -P-----------------------------------------------------------

     P  1     ...8)O-CLR-W1 ()
     P  2     ...Data Usage:
     P  3     ......External:
     P  4     ......Private:
     P  5     ...Behavior:
     P  6     ...Interface:
     P  7     ......Input: none
     P  8     ......Output: none
     P  9     ...PDL Description:
     P 10     ...
     P 11     position cursor at W1_X, W1_Y
     P 12     write empty line to window 1
```

O-CLR-W2

```
-------------P-  64B  -P-----------------------------------------------------------

     P  1     ...9)O-CLR-W2 ()
     P  2     ...Data Usage:
     P  3     ......External:
     P  4     ......Private:
     P  5     ...Behavior:
     P  6     ...Interface:
     P  7     ......Input: none
     P  8     ......Output: none
     P  9     ...PDL Description:
     P 10     ...
     P 11     position cursor at W2_X, W2_Y
     P 12     write empty line to window 2
```

O-CLR-W3

```
-------------P-  64C  -P-----------------------------------------------------------

     P  1     ...10)O-CLR-W3 ()
     P  2     ...Data Usage:
     P  3     ......External:
     P  4     ......Private:
     P  5     ...Behavior:
     P  6     ...Interface:
     P  7     ......Input: none
     P  8     ... ...Output: none
     P  9     ...PDL Description:
     P 10     ...
     P 11     position cursor at W3_X, W3_Y
     P 12     write empty line to window 3
```

O-CLR-W4

```
-------------P-  65A  -P-------------------------------------------------------

     P  1     ...11)O-CLR-W4 ()
     P  2     ...Data Usage:
     P  3     ......External:
     P  4     ......Private:
     P  5     ...Behavior:
     P  6     ...Interface:
     P  7     ......Input: none
     P  8     ......Output: none
     P  9     ...PDL Description:
     P 10     ...
     P 11     position cursor at W4_X, W4_Y
     P 12     write empty line to window 4
```

O-CLR-W5

```
-------------P-  65B  -P-------------------------------------------------------

     P  1     ...12)O-CLR-W5 ()
     P  2     ...Data Usage:
     P  3     ......External:
     P  4     ......Private:
     P  5     ...Behavior:
     P  6     ...Interface:
     P  7     ......Input: none
     P  8     ......Output: none
     P  9     ...PDL Description:
     P 10     ...
     P 11     position cursor at W5_X, W5_Y
     P 12     write empt line to window 5
```

****

Auxiliary Functions

****

O-WAIT

----------P- 67A -P----------

```
P  1    ...Data Usage:
P  2    ......External: KEY_
P  3    ......Private:
P  4    ...Behavior:
P  5    ...Interface:
P  6    ......Input: KEY_
P  7    ......Output: none
P  8    ...PDL Description:
P  9    ...
P 10    repeat
P 11        read keyboard
P 12        KEY_ := pressed key
P 13    until KEY_ is not set to null
```

Quit Program [P-QUIT-PROG]

----------P- 67B -P----------

```
P  1    ...Data Usage:
P  2    ......External:
P  3    .........O-WRITE-TXT-DATA
P  4    .........O-FRONT-ENQ
P  5    .........O-WRITE-Q-DATA
P  6    .........SYSTEM_ON
P  7    ......Private:
P  8    .........Variable:
P  9    .........STATUS_
P 10    ...Behavior:
P 11    ...Interface:
P 12    ......Input: none
P 13    ......Output: none
P 14    ...PDL Description:
P 15    ...
52A P 16    enqueue process-id [O-FRONT-ENQ]
    P 17    write text data to file [O-WRITE-TXT-DATA]
    P 18    write queue data to file [O-WRITE-Q-DATA]
    P 19    SYSTEM_ON := false
```

PROCEDURES

```
 5A      1  Scheduler Process
61A      2  .  O-SCREEN-FORMAT
41A      3  .  O-READ-BIRD-DATA
63A      4  .  O-WRT-W4
65A      5  .  .  O-CLR-W4
67B      6  .  Quit Program [P-QUIT-PROG]
52A      7  .  .  O-FRONT-ENQ
38B      8  .  O-FILL-FILTER
42A      9  .  O-SCAN
47A     10  .  O-READ-NO-OF-MATCHED-BIRDS
63B     11  .  O-WRT-W5
65B     12  .  .  O-CLR-W5
53A     13  .  O-DEQ
63A     14  .  O-WRT-W4 (     4 )
67A     15  .  O-WAIT
65A     16  .  O-CLR-W4 (     5 )
 8A     17  .  Help Menu Process [P-HELP]
61B     18  .  .  O-WRT-W1
64A     19  .  .  .  O-CLR-W1
33A     20  .  .  O-READ-TXT-ITEM
56A     21  .  .  O-INITIAL-TEXT
62A     22  .  .  O-WRT-W2
64B     23  .  .  .  O-CLR-W2
62B     24  .  .  O-WRT-W3
64C     25  .  .  .  O-CLR-W3
63A     26  .  .  O-WRT-W4 (     4 )
62A     27  .  .  O-WRT-W2 (    22 )
62B     28  .  .  O-WRT-W3 (    24 )
63A     29  .  .  O-WRT-W4 (     4 )
58A     30  .  .  O-MOVE-TEXT-RIGHT
62A     31  .  .  O-WRT-W2 (    22 )
62B     32  .  .  O-WRT-W3 (    24 )
63A     33  .  .  O-WRT-W4 (     4 )
11A     34  .  Attribute Menu Process [P-ATTR-MENU]
32A     35  .  .  O-READ-MENU-TITLE
61B     36  .  .  O-WRT-W1 (    18 )
33A     37  .  .  O-READ-TXT-ITEM (    20 )
56A     38  .  .  O-INITIAL-TEXT (    21 )
62A     39  .  .  O-WRT-W2 (    22 )
62B     40  .  .  O-WRT-W3 (    24 )
63A     41  .  .  O-WRT-W4 (     4 )
52A     42  .  .  O-FRONT-ENQ (     7 )
52A     43  .  .  O-FRONT-ENQ (     7 )
52A     44  .  .  O-FRONT-ENQ (     7 )
51A     45  .  .  O-EXCH-Q-BLOCKS
52A     46  .  .  O-FRONT-ENQ (     7 )
34A     47  .  .  O-UPDATE-ATTR-FLAG
52A     48  .  .  O-FRONT-ENQ (     7 )
53A     49  .  .  O-DEQ (    13 )
38B     50  .  .  O-FILL-FILTER (     8 )
42A     51  .  .  O-SCAN (     9 )
47A     52  .  .  O-READ-NO-OF-MATCHED-BIRDS (    10 )
```

PROCEDURES

```
63B      53   .  .  O-WRT-W5 (    11 )
33A      54   .  .  O-READ-TXT-ITEM (    20 )
56A      55   .  .  O-INITIAL-TEXT (    21 )
62A      56   .  .  O-WRT-W2 (    22 )
62B      57   .  .  O-WRT-W3 (    24 )
63A      58   .  .  O-WRT-W4 (     4 )
33A      59   .  .  O-READ-TXT-ITEM (    20 )
56A      60   .  .  O-INITIAL-TEXT (    21 )
62A      61   .  .  O-WRT-W2 (    22 )
62B      62      .  O-WRT-W3 (    24 )
63A      63   .  .  O-WRT-W4 (     4 )
63A      64   .  .  O-WRT-W4 (     4 )
67A      65   .  .  O-WAIT (    10 )
33A      66   .  .  O-READ-TXT-ITEM (    20 )
56A      67   .  .  O-INITIAL-TEXT (    21 )
62A      68   .  .  O-WRT-W2 (    22 )
62B      69   .  .  O-WRT-W3 (    24 )
63A      70   .  .  O-WRT-W4 (     4 )
63A      71   .  .  O-WRT-W4 (     4 )
67A      72   .  .  O-WAIT (    15 )
57A      73   .  .  O-MOVE-TEXT-LEFT
62A      74   .  .  O-WRT-W2 (    22 )
62B      75   .  .  O-WRT-W3 (    24 )
63A      76   .  .  O-WRT-W4 (     4 )
58A      77   .  .  O-MOVE-TEXT-RIGHT (    30 )
62A      78   .  .  O-WRT-W2 (    22 )
62B      79   .  .  O-WRT-W3 (    24 )
63A      80   .  .  O-WRT-W4 (     4 )
67B      81   .  .  Quit Program [P-QUIT-PROG] (     6 )
13A      82   .  Utility Menu Process [P-UTIL]
32A      83   .  .  O-READ-MENU-TITLE (    35 )
61B      84   .  .  O-WRT-W1 (    18 )
33A      85   .  .  O-READ-TXT-ITEM (    20 )
56A      86   .  .  O-INITIAL-TEXT (    21 )
62A      87   .  .  O-WRT-W2 (    22 )
62B      88   .  .  O-WRT-W3 (    24 )
63A      89   .  .  O-WRT-W4 (     4 )
52A      90   .  .  O-FRONT-ENQ (     7 )
52A      91   .  .  O-FRONT-ENQ (     7 )
52A      92   .  .  O-FRONT-ENQ (     7 )
51A      93   .  .  O-EXCH-Q-BLOCKS (    45 )
52A      94   .  .  O-FRONT-ENQ (     7 )
34A      95   .  .  O-UPDATE-ATTR-FLAG (    47 )
52A      96   .  .  O-FRONT-ENQ (     7 )
53A      97   .  .  O-DEQ (    13 )
33A      98   .  .  O-READ-TXT-ITEM (    20 )
56A      99   .  .  O-INITIAL-TEXT (    21 )
62A     100   .  .  O-WRT-W2 (    22 )
62B     101   .  .  O-WRT-W3 (    24 )
63A     102   .  .  O-WRT-W4 (     4 )
33A     103   .  .  O-READ-TXT-ITEM (    20 )
56A     104   .  .  O-INITIAL-TEXT (    21 )
```

PROCEDURES

```
62A     105   .  .  O-WRT-W2 (    22 )
62B     106   .  .  O-WRT-W3 (    24 )
63A     107   .  .  O-WRT-W4 (     4 )
33A     108   .  .  O-READ-TXT-ITEM (    20 )
56A     109   .  .  O-INITIAL-TEXT (    21 )
62A     110   .  .  O-WRT-W2 (    22 )
62B     111   .  .  O-WRT-W3 (    24 )
63A     112   .  .  O-WRT-W4 (     4 )
57A     113   .  .  O-MOVE-TEXT-LEFT (    73 )
62A     114   .  .  O-WRT-W2 (    22 )
62B     115   .  .  O-WRT-W3 (    24 )
63A     116   .  .  O-WRT-W4 (     4 )
58A     117   .  .  O-MOVE-TEXT-RIGHT (    30 )
62A     118   .  .  O-WRT-W2 (    22 )
62B     119   .  .  O-WRT-W3 (    24 )
63A     120   .  .  O-WRT-W4 (     4 )
67B     121   .  .  Quit Program [P-QUIT-PROG] (     6 )
16A     122   .  Find Main Menu Process [P-FIND-MENU]
47A     123   .  .  O-READ-NO-OF-MATCHED-BIRDS (    10 )
63B     124   .  .  O-WRT-W5 (    11 )
32A     125   .  .  O-READ-MENU-TITLE (    35 )
61B     126   .  .  O-WRT-W1 (    18 )
33A     127   .  .  O-READ-TXT-ITEM (    20 )
56A     128   .  .  O-INITIAL-TEXT (    21 )
62A     129   .  .  O-WRT-W2 (    22 )
62B     130   .  .  O-WRT-W3 (    24 )
63A     131   .  .  O-WRT-W4 (     4 )
52A     132   .  .  O-FRONT-ENQ (     7 )
52A     133   .  .  O-FRONT-ENQ (     7 )
52A     134   .  .  O-FRONT-ENQ (     7 )
51A     135   .  .  O-EXCH-Q-BLOCKS (    45 )
52A     136   .  .  O-FRONT-ENQ (     7 )
34A     137   .  .  O-UPDATE-ATTR-FLAG (    47 )
52A     138   .  .  O-FRONT-ENQ (     7 )
53A     139   .  .  O-DEQ (    13 )
38B     140   .  .  O-FILL-FILTER (     8 )
42A     141   .  .  O-SCAN (     9 )
47A     142   .  .  O-READ-NO-OF-MATCHED-BIRDS (    10 )
63B     143   .  .  O-WRT-W5 (    11 )
33A     144   .  .  O-READ-TXT-ITEM (    20 )
56A     145   .  .  O-INITIAL-TEXT (    21 )
62A     146   .  .  O-WRT-W2 (    22 )
62B     147   .  .  O-WRT-W3 (    24 )
63A     148   .  .  O-WRT-W4 (     4 )
33A     149   .  .  O-READ-TXT-ITEM (    20 )
56A     150   .  .  O-INITIAL-TEXT (    21 )
62A     151   .  .  O-WRT-W2 (    22 )
62B     152   .  .  O-WRT-W3 (    24 )
63A     153   .  .  O-WRT-W4 (     4 )
33A     154   .  .  O-READ-TXT-ITEM (    20 )
56A     155   .  .  O-INITIAL-TEXT (    21 )
62A     156   .  .  O-WRT-W2 (    22 )
```

PROCEDURES

```
62B     157   .  .  O-WRT-W3 (    24 )
63A     158   .  .  O-WRT-W4 (     4 )
57A     159   .  .  O-MOVE-TEXT-LEFT (    73 )
62A     160   .  .  O-WRT-W2 (    22 )
62B     161   .  .  O-WRT-W3 (    24 )
63A     162   .  .  O-WRT-W4 (     4 )
58A     163   .  .  O-MOVE-TEXT-RIGHT (    30 )
62A     164   .  .  O-WRT-W2 (    22 )
62B     165   .  .  O-WRT-W3 (    24 )
63A     166   .  .  O-WRT-W4 (     4 )
67B     167   .  .  Quit Program [P-QUIT-PROG] (     6 )
18A     168   .  List Found Birds Process [P-LIST-FOUND]
45A     169   .  .  O-READ-NO-OF-FOUND-BIRDS
63B     170   .  .  O-WRT-W5 (    11 )
61B     171   .  .  O-WRT-W1 (    18 )
63A     172   .  .  O-WRT-W4 (     4 )
45B     173   .  .  O-LIST-FOUND-BIRDS
61B     174   .  .  O-WRT-W1 (    18 )
63A     175   .  .  O-WRT-W4 (     4 )
37A     176   .  .  O-READ-LIST-FOUND-METHOD
52A     177   .  .  O-FRONT-ENQ (     7 )
52A     178   .  .  O-FRONT-ENQ (     7 )
52A     179   .  .  O-FRONT-ENQ (     7 )
51A     180   .  .  O-EXCH-Q-BLOCKS (    45 )
52A     181   .  .  O-FRONT-ENQ (     7 )
52A     182   .  .  O-FRONT-ENQ (     7 )
45B     183   .  .  O-LIST-FOUND-BIRDS (   173 )
61B     184   .  .  O-WRT-W1 (    18 )
63A     185   .  .  O-WRT-W4 (     4 )
45B     186   .  .  O-LIST-FOUND-BIRDS (   173 )
61B     187   .  .  O-WRT-W1 (    18 )
63A     188   .  .  O-WRT-W4 (     4 )
67B     189   .  .  Quit Program [P-QUIT-PROG] (     6 )
19B     190   .  Clear Found Birds Process [P-CLEAR-FOUND]
61B     191   .  .  O-WRT-W1 (    18 )
63A     192   .  .  O-WRT-W4 (     4 )
52A     193   .  .  O-FRONT-ENQ (     7 )
52A     194   .  .  O-FRONT-ENQ (     7 )
52A     195   .  .  O-FRONT-ENQ (     7 )
51A     196   .  .  O-EXCH-Q-BLOCKS (    45 )
46A     197   .  .  O-CLEAR-FOUND-BIRDS
21A     198   .  List Attributes Process [P-LIST-ATTR]
35A     199   .  .  O-READ-NO-OF-SELECTED-ATTRIBUTES
63B     200   .  .  O-WRT-W5 (    11 )
61B     201   .  .  O-WRT-W1 (    18 )
63A     202   .  .  O-WRT-W4 (     4 )
36B     203   .  .  O-LIST-SELECTED-ATTRIBUTES
61B     204   .  .  O-WRT-W1 (    18 )
63A     205   .  .  O-WRT-W4 (     4 )
52A     206   .  .  O-FRONT-ENQ (     7 )
52A     207   .  .  O-FRONT-ENQ (     7 )
52A     208   .  .  O-FRONT-ENQ (     7 )
```

PROCEDURES

```
51A   209   .  .  O-EXCH-Q-BLOCKS (    45 )
61B   210   .  .  O-WRT-W1 (    18 )
63A   211   .  .  O-WRT-W4 (     4 )
36A   212   .  .  O-CLEAR-ATTRIBUTE
38B   213   .  .  O-FILL-FILTER (     8 )
42A   214   .  .  O-SCAN (     9 )
35A   215   .  .  O-READ-NO-OF-SELECTED-ATTRIBUTES (   199 )
63B   216   .  .  O-WRT-W5 (    11 )
61B   217   .  .  O-WRT-W1 (    18 )
63A   218   .  .  O-WRT-W4 (     4 )
36B   219   .  .  O-LIST-SELECTED-ATTRIBUTES (   203 )
61B   220   .  .  O-WRT-W1 (    18 )
63A   221   .  .  O-WRT-W4 (     4 )
36B   222   .  .  O-LIST-SELECTED-ATTRIBUTES (   203 )
61B   223   .  .  O-WRT-W1 (    18 )
63A   224   .  .  O-WRT-W4 (     4 )
36B   225   .  .  O-LIST-SELECTED-ATTRIBUTES (   203 )
61B   226   .  .  O-WRT-W1 (    18 )
63A   227   .  .  O-WRT-W4 (     4 )
57A   228   .  .  O-MOVE-TEXT-LEFT (    73 )
62A   229   .  .  O-WRT-W2 (    22 )
62B   230   .  .  O-WRT-W3 (    24 )
63A   231   .  .  O-WRT-W4 (     4 )
58A   232   .  .  O-MOVE-TEXT-RIGHT (    30 )
62A   233   .  .  O-WRT-W2 (    22 )
62B   234   .  .  O-WRT-W3 (    24 )
63A   235   .  .  O-WRT-W4 (     4 )
67B   236   .  .  Quit Program [P-QUIT-PROG] (     6 )
22B   237   .  Clear Attributes Process [P-CLEAR-ATTR]
61B   238   .  .  O-WRT-W1 (    18 )
63A   239   .  .  O-WRT-W4 (     4 )
52A   240   .  .  O-FRONT-ENQ (     7 )
52A   241   .  .  O-FRONT-ENQ (     7 )
52A   242   .  .  O-FRONT-ENQ (     7 )
51A   243   .  .  O-EXCH-Q-BLOCKS (    45 )
35B   244   .  .  O-CLEAR-ALL-ATTRIBUTES
38B   245   .  .  O-FILL-FILTER (     8 )
42A   246   .  .  O-SCAN (     9 )
47A   247   .  .  O-READ-NO-OF-MATCHED-BIRDS (    10 )
63B   248   .  .  O-WRT-W5 (    11 )
24A   249   .  List Matched Birds Process
47A   250   .  .  O-READ-NO-OF-MATCHED-BIRDS (    10 )
63B   251   .  .  O-WRT-W5 (    11 )
61B   252   .  .  O-WRT-W1 (    18 )
63A   253   .  .  O-WRT-W4 (     4 )
43A   254   .  .  O-LIST-MATCHED-BIRDS
63A   255   .  .  O-WRT-W4 (     4 )
61B   256   .  .  O-WRT-W1 (    18 )
38A   257   .  .  O-READ-LIST-MATCHED-METHOD
52A   258   .  .  O-FRONT-ENQ (     7 )
52A   259   .  .  O-FRONT-ENQ (     7 )
52A   260   .  .  O-FRONT-ENQ (     7 )
```

Birds
CALLING TREES

PROCEDURES

```
51A   261   .  .  O-EXCH-Q-BLOCKS (    45 )
52A   262   .  .  O-FRONT-ENQ (     7 )
52A   263   .  .  O-FRONT-ENQ (     7 )
43A   264   .  .  O-LIST-MATCHED-BIRDS (   254 )
61B   265   .  .  O-WRT-W1 (    18 )
63A   266   .  .  O-WRT-W4 (     4 )
43A   267   .  .  O-LIST-MATCHED-BIRDS (   254 )
61B   268   .  .  O-WRT-W1 (    18 )
63A   269   .  .  O-WRT-W4 (     4 )
57A   270   .  .  O-MOVE-TEXT-LEFT (    73 )
62A   271   .  .  O-WRT-W2 (    22 )
62B   272   .  .  O-WRT-W3 (    24 )
63A   273   .  .  O-WRT-W4 (     4 )
58A   274   .  .  O-MOVE-TEXT-RIGHT (    30 )
62A   275   .  .  O-WRT-W2 (    22 )
62B   276   .  .  O-WRT-W3 (    24 )
63A   277   .  .  O-WRT-W4 (     4 )
67B   278   .  .  Quit Program [P-QUIT-PROG] (     6 )
26A   279   .  Count Bird Process [P-COUNT-BIRD]
44B   280   .  .  O-CHANGE-BIRD-COUNT
38A   281   .  .  O-READ-LIST-MATCHED-METHOD (   257 )
52A   282   .  .  O-FRONT-ENQ (     7 )
52A   283   .  .  O-FRONT-ENQ (     7 )
52A   284   .  .  O-FRONT-ENQ (     7 )
51A   285   .  .  O-EXCH-Q-BLOCKS (    45 )
44B   286   .  .  O-CHANGE-BIRD-COUNT (   280 )
61B   287   .  .  O-WRT-W1 (    18 )
44B   288   .  .  O-CHANGE-BIRD-COUNT (   280 )
61B   289   .  .  O-WRT-W1 (    18 )
67B   290   .  .  Quit Program [P-QUIT-PROG] (     6 )
27B   291   .  Clear a Found Bird Process [P-CLEAR-BIRD]
61B   292   .  .  O-WRT-W1 (    18 )
63A   293   .  .  O-WRT-W4 (     4 )
52A   294   .  .  O-FRONT-ENQ (     7 )
52A   295   .  .  O-FRONT-ENQ (     7 )
52A   296   .  .  O-FRONT-ENQ (     7 )
51A   297   .  .  O-EXCH-Q-BLOCKS (    45 )
46B   298   .  .  O-CLEAR-BIRD
30A   299   O-READ-MENU-FILE
31A   300   O-WRITE-MENU-FILE
43B   301   O-READ-COUNT-FILE
44A   302   O-WRITE-COUNT-FILE
50A   303   O-READ-Q-FILE
50B   304   O-WRITE-Q-FILE
```

Birds
PROCEDURE INDEX

PROCEDURE DEFINED
REFERENCED IN PROCEDURE
LINES

Birds
PROCEDURE INDEX

PROCEDURE DEFINED
REFERENCED IN PROCEDURE
LINES

Birds
PROCEDURE INDEX

PROCEDURE DEFINED
REFERENCED IN PROCEDURE
LINES

Birds
PROCEDURE INDEX

PROCEDURE DEFINED
REFERENCED IN PROCEDURE
LINES

9A Help Menu Process [P-HELP]
21
11A Attribute Menu Process [P-ATTR-MENU]
80
13A Utility Menu Process [P-UTIL]
65
16A Find Main Menu Process [P-FIND-MENU]
71
21A List Attributes Process [P-LIST-ATTR]
80
24A List Matched Birds Process
65

41A O-READ-BIRD-DATA
5A Scheduler Process
3

43B O-READ-COUNT-FILE

37A O-READ-LIST-FOUND-METHOD
18A List Found Birds Process [P-LIST-FOUND]
17

38A O-READ-LIST-MATCHED-METHOD
24A List Matched Birds Process
18
26A Count Bird Process [P-COUNT-BIRD]
7

30A O-READ-MENU-FILE

32A O-READ-MENU-TITLE
11A Attribute Menu Process [P-ATTR-MENU]
1
13A Utility Menu Process [P-UTIL]
1
16A Find Main Menu Process [P-FIND-MENU]
3

45A O-READ-NO-OF-FOUND-BIRDS
18A List Found Birds Process [P-LIST-FOUND]
1

47A O-READ-NO-OF-MATCHED-BIRDS
5A Scheduler Process
23
11A Attribute Menu Process [P-ATTR-MENU]
41
16A Find Main Menu Process [P-FIND-MENU]
1 42
28D Clear Attributes Process [P-CLEAR-ATTR]
21
24A List Matched Birds Process
1

35A O-READ-NO-OF-SELECTED-ATTRIBUTES
21A List Attributes Process [P-LIST-ATTR]
1 45

50A O-READ-Q-FILE

33A O-READ-TXT-ITEM

PROCEDURE INDEX

PROCEDURE DEFINED
REFERENCED IN PROCEDURE
LINES

8A Help Menu Process [P-HELP]
3

11A Attribute Menu Process [P-ATTR-MENU]
4 44 52 64

13A Utility Menu Process [P-UTIL]
4 39 47 54

16A Find Main Menu Process [P-FIND-MENU]
6 45 53 60

42A O-SCAN
5A Scheduler Process
22

11A Attribute Menu Process [P-ATTR-MENU]
40

16A Find Main Menu Process [P-FIND-MENU]
41

21A List Attributes Process [P-LIST-ATTR]
44

22B Clear Attributes Process [P-CLEAR-ATTR]
20

61A O-SCREEN-FORMAT
5A Scheduler Process
2

34A O-UPDATE-ATTR-FLAG
11A Attribute Menu Process [P-ATTR-MENU]
30

13A Utility Menu Process [P-UTIL]
29

16A Find Main Menu Process [P-FIND-MENU]
31

67A O-WAIT
5A Scheduler Process
31

11A Attribute Menu Process [P-ATTR-MENU]
60 72

44A O-WRITE-COUNT-FILE
31A O-WRITE-MENU-FILE
50B O-WRITE-Q-FILE
61B O-WRT-W1
8A Help Menu Process [P-HELP]
1

11A Attribute Menu Process [P-ATTR-MENU]
2

13A Utility Menu Process [P-UTIL]
2

16A Find Main Menu Process [P-FIND-MENU]
4

18A List Found Birds Process [P-LIST-FOUND]
4 15 48 53

19B Clear Found Birds Process [P-CLEAR-FOUND]
1

21A List Attributes Process [P-LIST-ATTR]
4

Birds
PROCEDURE INDEX

PROCEDURE DEFINED
REFERENCED IN PROCEDURE
LINES

PROCEDURE INDEX

PROCEDURE DEFINED
REFERENCED IN PROCEDURE
LINES

Birds
DATA INDEX

```
DATA ITEM
          REFERENCED IN SEGMENT
               LINES
     30A  O-READ-MENU-FILE
               37 38 41 42 53 55 60 63 64 71 75 76
     33A  O-READ-TXT-ITEM
               57 58
CONTINUE_
      2A  Private Variables
                3
      8A  Help Menu Process [P-HELP]
                9 26
     11A  Attribute Menu Process [P-ATTR-MENU]
                9 10 22 26 35 89 93
     13A  Utility Menu Process [P-UTIL]
                9 10 22 26 34 70 73
     16A  Find Main Menu Process [P-FIND-MENU]
               11 12 23 27 36 80 84
     18A  List Found Birds Process [P-LIST-FOUND]
               18 19 30 34 44 56 59
     19B  Clear Found Birds Process [P-CLEAR-FOUND]
               13 17
     21A  List Attributes Process [P-LIST-ATTR]
               15 16 28 32 54 59 85 88
     22B  Clear Attributes Process [P-CLEAR-ATTR]
               12 16 24
     24A  List Matched Birds Process
               16 17 30 34 48 70 73
     26A  Count Bird Process [P-COUNT-BIRD]
                5  6 18 22 32
     27B  Clear a Found Bird Process [P-CLEAR-BIRD]
               12 16
     30A  O-READ-MENU-FILE
               33 39 43 65
     32A  O-READ-MENU-TITLE
               17 18 21
     57A  O-MOVE-TEXT-LEFT
               15 16 20
     58A  O-MOVE-TEXT-RIGHT
               15 16 20 24
C_POINTER
      2A  Private Variables
                6
     30A  O-READ-MENU-FILE
                50
     31A  O-WRITE-MENU-FILE
                21
     33A  O-READ-TXT-ITEM
               47 48 49 52 53 54 57 63
C_REC
     32A  O-READ-MENU-TITLE
               20 30
DISPLACEMENT_
      2A  Private Variables
               13
```

Birds
DATA INDEX

Birds
DATA INDEX

```
DATA ITEM
          REFERENCED IN SEGMENT
               LINES
GUIDE_ARROW_R
     56A  O-INITIAL-TEXT
               25 27
     57A  O-MOVE-TEXT-LEFT
               28 30 34
     58A  O-MOVE-TEXT-RIGHT
               31 34
     62B  O-WRT-W3
               11
 HELP_ARY
     30A  O-READ-MENU-FILE
               73
     31A  O-WRITE-MENU-FILE
               31
     33A  O-READ-TXT-ITEM
               41
 HELP_ID
     11A  Attribute Menu Process [P-ATTR-MENU]
               20
     13A  Utility Menu Process [P-UTIL]
               20
     16A  Find Main Menu Process [P-FIND-MENU
               21
     18A  List Found Birds Process [P-LIST-FOUND]
               28
     19B  Clear Found Birds Process [P-CLEAR-FOUND]
               11
     21A  List Attributes Process [P-LIST-ATTR]
               26
     22B  Clear Attributes Process [P-CLEAR-ATTR]
               10
     24A  List Matched Birds Process
               28
     26A  Count Bird Process [P-COUNT-BIRD]
               16
     27B  Clear a Found Bird Process [P-CLEAR-BIRD]
               10
 H_C_POINTER
     33A  O-READ-TXT-ITEM
               31 33 34 37 38 41
 H_F_POINTER
     30A  O-READ-MENU-FILE
               68 79
 H_F_POINTR
     30A  O-READ-MENU-FILE
               77
 H_L_POINTER
     30A  O-READ-MENU-FILE
               77 80
     31A  O-WRITE-MENU-FILE
               30
     33A  O-READ-TXT-ITEM
               33
```

Birds
DATA INDEX

DATA ITEM
REFERENCED IN SEGMENT
LINES

33A O-READ-'XT-ITEM
3/

ID_ATTR_M
1A Public Constants
1

ID_BIRD_F
1A Public Constants
11

ID_BIRD_SAN
1A Public Constants
8

ID_COLORING
1A Pub' c Constants
15

ID_COUNTR'
1A Public Constants
4

ID_FEATURE
1A Public Constants
14

ID_FEEDING
1A Public Constants
13

ID_FIND_H
1A Public Constants
2

ID_HABITAT
1A Public Constants
9

ID_HELP
1A Public Constants
3

33A O-READ-TXT-ITEM
26

ID_H:KING
1A Public Constants
6

ID_MARKING
1A Public Constants
16

ID_N_PARKS
1A Public Constants
5

ID_SIZE
1A Public Constants
12

ID_STATUS
'A Public Constants
10

ID_TOWNS
1A Public Constants
7

```
DATA ITEM
          REFERENCED IN SEGMENT
               LINES
ID_UTILITY
      1A  Public Constants
               17
I_
     30A  O-READ-MENU-FILE
               54 56 58 59 70 73 74
     31A  O-WRITE-MENU-FILE
               22 23 24 25 26 29 30 31 32
     50A  O-READ-Q-FILE
               17 18 19 20
     50B  O-WRITE-Q-FILE
               17 18 19 20
     51A  O-EXCH-Q-BLOCKS
               14 15 16 16 17 19 20 21 21 22 24 25 26 26 27
     52A  O-FRONT-ENQ
               26 27 28 28 29
     53A  O-DEQ
               29 30 31 31 32
KEY_
      2A  Private Variables
                5
      8A  Help Menu Process [P-HELP]
               12 13 14
     11A  Attribute Menu Process [P-ATTR-MENU]
               13 14 17
     13A  Utility Menu Process [P-UTIL]
               13 14 17
     16A  Find Main Menu Process [P-FIND-MENU]
               15 16 18
     18A  List Found Birds Process [P-LIST-FOUND]
               22 23 25
     19B  Clear Found Birds Process [P-CLEAR-FOUND]
                5  6  8
     21A  List Attributes Process [P-LIST-ATTR]
                8  9 19 20 23 38 39 40 52 53
     22B  Clear Attributes Process [P-CLEAR-ATTR]
                5  6  7
     24A  List Matched Birds Process
                8  9 21 22 25
     26A  Count Bird Process [P-COUNT-BIRD]
               10 11 13
     27B  Clear a Found Bird Process [P-CLEAR-BIRD]
                5  6  7
     67A  O-WAIT
               12 13
KEY_DN_ARROW
     11A  Attribute Menu Process [P-ATTR-MENU]
               62
     13A  Utility Menu Process [P-UTIL]
               52
     16A  Find Main Menu Process [P-FIND-MENU]
               58
```

DATA ITEM
REFERENCED IN SEGMENT
LINES

18A List Found Birds Process [P-LIST-FOUND]
50
21A List Attributes Process [P-LIST-ATTR]
69
24A List Matched Birds Process
54
26A Count Bird Process [P-COUNT-BIRD]
27

KEY_ESCAPE
8A Help Menu Process [P-HELP]
25
11A Attribute Menu Process [P-ATTR-MENU]
84
13A Utility Menu Process [P-UTIL]
69
16A Find Main Menu Process [P-FIND-MENU]
75
18A List Found Birds Process [P-LIST-FOUND]
55
19B Clear Found Birds Process [P-CLEAR-FOUND]
20
21A List Attributes Process [P-LIST-ATTR]
84
22B Clear Attributes Process [P-CLEAR-ATTR]
23
24A List Matched Birds Process
69
26A Count Bird Process [P-COUNT-BIRD]
31
27B Clear a Found Bird Process [P-CLEAR-BIRD]
20

KEY_HELP
11A Attribute Menu Process [P-ATTR-MENU]
18
13A Utility Menu Process [P-UTIL]
18
16A Find Main Menu Process [P-FIND-MENU]
19
18A List Found Birds Process [P-LIST-FOUND]
26
19B Clear Found Birds Process [P-CLEAR-FOUND]
9
21A List Attributes Process [P-LIST-ATTR]
24
22B Clear Attributes Process [P-CLEAR-ATTR]
8
24A List Matched Birds Process
26
26A Count Bird Process [P-COUNT-BIRD]
14
27B Clear a Found Bird Process [P-CLEAR-BIRD]
8

DATA ITEM
REFERENCED IN SEGMENT
LINES

KEY_LEFT_ARROW
8A Help Menu Process [P-HELP]
15
11A Attribute Menu Process [P-ATTR-MENU]
74
13A Utility Menu Process [P-UTIL]
59
16A Find Main Menu Process [P-FIND-MENU]
65
21A List Attributes Process [P-LIST-ATTR]
74
24A List Matched Birds Process
59

KEY_MODE
11A Attribute Menu Process [P-ATTR-MENU]
23
13A Utility Menu Process [P-UTIL]
23
16A Find Main Menu Process [P-FIND-MENU]
24
18A List Found Birds Process [P-LIST-FOUND]
31
19B Clear Found Birds Process [P-CLEAR-FOUND]
14
21A List Attributes Process [P-LIST-ATTR]
29
22B Clear Attributes Process [P-CLEAR-ATTR]
13
24A List Matched Birds Process
31
26A Count Bird Process [P-COUNT-BIRD]
19
27B Clear a Found Bird Process [P-CLEAR-BIRD]
13

KEY_QUIT
11A Attribute Menu Process [P-ATTR-MENU]
91
13A Utility Menu Process [P-UTIL]
71
16A Find Main Menu Process [P-FIND-MENU]
82
18A List Found Birds Process [P-LIST-FOUND]
57
21A List Attributes Process [P-LIST-ATTR]
86
24A List Matched Birds Process
71
26A Count Bird Process [P-COUNT-BIRD]
33

KEY_RIGHT_ARROW
8A Help Menu Process [P-HELP]
20

DATA ITEM
REFERENCED IN SEGMENT
LINES
11A Attribute Menu Process [P-ATTR-MENU]
79
13A Utility Menu Process [P-UTIL]
64
16A Find Main Menu Process [P-FIND-MENU]
70
21A List Attributes Process [P-LIST-ATTR]
79
24A List Matched Birds Process
64

KEY_SELECT
11A Attribute Menu Process [P-ATTR-MENU]
27
13A Utility Menu Process [P-UTIL]
27
16A Find Main Menu Process [P-FIND-MENU]
28
18A List Found Birds Process [P-LIST-FOUND]
35
19B Clear Found Birds Process [P-CLEAR-FOUND]
18
21A List Attributes Process [P-LIST-ATTR]
33 41
22B Clear Attributes Process [P-CLEAR-ATTR]
17
24A List Matched Birds Process
35
27B Clear a Found Bird Process [P-CLEAR-BIRD]
17

KEY_UP_ARROW
11A Attribute Menu Process [P-ATTR-MENU]
50
13A Utility Menu Process [P-UTIL]
45
16A Find Main Menu Process [P-FIND-MENU]
51
18A List Found Birds Process [P-LIST-FOUND]
45
21A List Attributes Process [P-LIST-ATTR]
64
24A List Matched Birds Process
49
26A Count Bird Process [P-COUNT-BIRD]
23

LAST_PROCESS_ID
1B Public Variables
2
18A List Found Birds Process [P-LIST-FOUND]
10
24A List Matched Birds Process
10 10
26A Count Bird Process [P-COUNT-BIRD]
1

```
DATA ITEM
          REFERENCED IN SEGMENT
                LINES
NO_OF_FOUND_BIRDS
     18A  List Found Birds Process [P-LIST-FOUND]
                3
     45A  O-READ-NO-OF-FOUND-BIRDS
                12 16 19
NO_OF_FOUND_BIRDS_PRIVATE
     45A  O-READ-NO-OF-FOUND-BIRDS
                19
NO_OF_LINES
     30A  O-READ-MENU-FILE
                31 34 69 76
NO_OF_MATCHED_BIRDS
     24A  List Matched Birds Process
                3
     42A  O-SCAN
                20 41 48 58 64
     47A  O-READ-NO-OF-MATCHED-BIRDS
                12
NO_OF_MATCHED_BIRDS_PRIVATE
     47A  O-READ-NO-OF-MATCHED-BIRDS
                12
NO_OF_REC
     31A  O-WRITE-MENU-FILE
                17
     32A  O-READ-MENU-TITLE
                26
NO_OF_SELECTED_ATTRIBUTES
     21A  List Attributes Process [P-LIST-ATTR]
                3 47
     35A  O-READ-NO-OF-SELECTED-ATTRIBUTES
                13 17
NO_OF_S_ATTR
     35A  O-READ-NO-OF-SELECTED-ATTRIBUTES
                17
NO_O_REC
     30A  O-READ-MENU-FILE
                36
PROCESS_ID
      1B  Public Variables
                1
      5A  Scheduler Process
                34
     11A  Attribute Menu Process [P-ATTR-MENU]
                20 85
     13A  Utility Menu Process [P-UTIL]
                20
     16A  Find Main Menu Process [P-FIND-MENU]
                21 76
     18A  List Found Birds Process [P-LIST-FOUND]
                28 39 41
     19B  Clear Found Birds Process [P-CLEAR-FOUND]
                11
```

DATA ITEM
REFERENCED IN SEGMENT
LINES

21A List Attributes Process [P-LIST-ATTR]
26

22B Clear Attributes Process [P-CLEAR-ATTR]
10

24A List Matched Birds Process
28 39 41 43 45

26A Count Bird Process [P-COUNT-BIRD]
16

27B Clear a Found Bird Process [P-CLEAR-BIRD]
10 19

30A O-READ-MENU-FILE
46

31A O-WRITE-MENU-FILE
19

32A O-READ-MENU-TITLE
19

33A O-READ-TXT-ITEM
28

34A O-UPDATE-ATTR-FLAG
44

52A O-FRONT-ENQ
22 31

53A O-DEQ
27

Q_ARY

50A O-READ-Q-FILE
19

50B O-WRITE-Q-FILE
19

51A O-EXCH-Q-BLOCKS
16 21 21 26

52A O-FRONT-ENQ
22 28 28 31

53A O-DEQ
27 31 31

Q_DAT

50A O-READ-Q-FILE
13

Q_MIDL

50A O-READ-Q-FILE
14

50B O-WRITE-Q-FILE
13

51A O-EXCH-Q-BLOCKS
15 20 21 25 26 29 29

52A O-FRONT-ENQ
23 33

53A O-DEQ
18 19 20

Q_SIZE

52A O-FRONT-ENQ
18

DATA ITEM
REFERENCED IN SEGMENT
LINES

Q_TAIL
50A O-READ-Q-FILE
16 18
50B O-WRITE-Q-FILE
15 18
51A O-EXCH-Q-BLOCKS
20 26 29
52A O-FRONT-ENQ
18 21 24 26 32
53A O-DEQ
17 22 24 30

Q_TMP_ARY
51/ O-EXCH-Q-BLOCKS
16 26

REC_
30A O-READ-MENU-FILE
30 35 36
31A O-WRITE-MENU-FILE
16 17 34
32A O-READ-MENU-TITLE
16 19 20 23 26

REV_W4
33A O-READ-TXT-ITEM
59 61
63A O-TXT-W4
11

SELECTED_ATTR
38B O-FILL-FILTER
16

SELECTED_ATTR_SET
38B O-FILL-FILTER
14 16
42A O-SCAN
22 26 30

STATUS_
2A Private Variables
4
5A Scheduler Process
6 29
11A Attribute Menu Process [P-ATTR-MENU]
31 53 65
13A Utility Menu Process [P-UTIL]
30
16A Find Main Menu Process [P-FIND-MENU]
32
30A O-READ-MENU-FILE
39 43 45 65
32A O-READ-MENU-TITLE
27 29
33A O-READ-TXT-ITEM
65
52A O-FRONT-ENQ
19

```
DATA ITEM
          REFERENCED IN SEGMENT
               LINES
     21A  List Attributes Process [P-LIST-ATTR]
               83
     24A  List Matched Birds Process
               63 68
     56A  O-INITIAL-TEXT
               23
     57A  O-MOVE-TEXT-LEFT
               36
     58A  O-MOVE-TEXT-RIGHT
               38
     61B  O-WRT-W1
               17
     62A  O-WRT-W2
               16
     62B  O-WRT-W3
               16
     63A  O-WRT-W4
               18
TEXT_ITEM
      2A  Private Variables
                1
     33A  O-READ-TXT-ITEM
               41 63
     56A  O-INITIAL-TEXT
               23
     57A  O-MOVE-TEXT-LEFT
               10 11 16 27 36
     58A  O-MOVE-TEXT-RIGHT
               10 11 16 18 22 23 30 38
TEXT_LENGTH
      2A  Private Variables
               12
     56A  O-INITIAL-TEXT
               24
TMP_ARY
     30A  O-READ-MENU-FILE
               37 41 46 48 50 53 56 58 60 63 71 73 75
TXT_ARY
     30A  O-READ-MENU-FILE
               56
     33A  O-READ-TXT-ITEM
               63
W1_SIZE
      2A  Private Variables
               21
W1_X
      2A  Private Variables
               15
     61B  O-WRT-W1
               16
     64A  O-CLR-W1
               11
```

DATA ITEM
REFERENCED IN SEGMENT
LINES

W1_Y
2A Private Variables
16
61B O-WRT-W1
16
64A O-CLR-W1
11

W2_SIZE
2A Private Variables
22

W2_X
62A O-WRT-W2
15
64B O-CLR-W2
11

W2_Y
62A O-WRT-W2
15
64B O-CLR-W2
11

W3_SIZE
2A Private Variables
23

W3_X
2A Private Variables
17
62B O-WRT-W3
15
64C O-CLR-W3
11

W3_Y
2A Private Variables
18
62B O-WRT-W3
15
64C O-CLR-W3
11

W4_SIZE
2A Private Variables
24

W4_X
2A Private Variables
19
63A O-WRT-W4
17
63B O-WRT-W5
12
65A O-CLR-W4
11

W4_Y
2A Private Variables
20

DATA ITEM
REFERENCED IN SEGMENT
LINES

63A O-WRT-W4
17

63B O-WRT-W5
12

65A O-CLR-W4
11

W5_TEXT
63B O-WRT-W5
13

W5_X
65B O-CLR-W5
11

W5_Y
65B O-CLR-W5
11

W_L_POINTER
2A Private Variables
9

56A O-INITIAL-TEXT
21 23

57A O-MOVE-TEXT-LEFT
10 11 12 16 17 18 19 23 24 36

58A O-MOVE-TEXT-RIGHT
28 29 38

W_R_POINTER
2A Private Variables
10

56A O-INITIAL-TEXT
22 23

57A O-MOVE-TEXT-LEFT
23

58A O-MOVE-TEXT-RIGHT
10 11 12 16 17 19 22 23 28 3

W_SIZE
56A O-INITIAL-TEXT
22 24

57A O-MOVE-TEXT-LEFT
23 27 36

58A O-MOVE-TEXT-RIGHT
18 19 28 38

APPENDIX. C- LIST OF MODULES AND CODE SIZE

Module Name	Code Size in Lines	% of Total Code Size
1-Data Environment	84	2.2
2-Scheduler Process	140	4.1
3-Attribute Menus Process	223	5.9
4-Utility Menu Process	200	5.3
5-Find Main Menu Process	210	6 6
6-List Found Birds Process	206	5.4
7-Clear Found Birds Process	99	2.6
8-List Attributes Process	204	5.4
9-Clear Attributes Process	108	2.9
10-List Matched Birds Process	198	5.2
11-Count Birds Process	125	3.3
12-Clear Bird From Found Birds Process	90	2.4
13-Help Process	109	2.9
14-Screen Resource	226	6.0
15-Scroll Resource	155	4.1
16-Text Resource	647	17.1
17-Bird Resource	527	14.0
18-Process Queue Resource	227	6.0
Total Code	3778	100.0