NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

ENHANCED VAX/VMS PROGRAMMING SOLUTIONS WITH APPLICATIONS FOR PRELIMINARY MARINE VEHICLE DESIGN

by

James R. Plosay

September 1990

Thesis Advisor. F.A. Papoulias

Approved for public release; distribution is unlimited.

classified

rity classification of this page

REPORT DOCUMENTATION PAGE

Report Security Classification Unclassified		1b Restrictive Markings			
Security Classification Authority		3 Distribution/Availability of Report Approved for public release; distribution is unlimited.			
Declassification Downgrading Schedule					
Performing Organization Report Number(s)		5 Monitoring Organization Report Number(s)			
Name of Performing Organization aval Postgraduate School	6b Office Symbol *(if applicable)* 34	7a Name of Monitoring Organization Naval Postgraduate School			
Address *(city, state, and ZIP code)* onterey, CA 93943-5000		7b Address *(city, state, and ZIP code)* Monterey, CA 93943-5000			
Name of Funding Sponsoring Organization	8b Office Symbol *(if applicable)*	9 Procurement Instrument Identification Number			
Address *(city, state, and ZIP code)*		10 Source of Funding Numbers			
		Program Element No	Project No	Task No	Work Unit Accession No

Title *(Include security classification)* ENHANCED VAX/VMS PROGRAMMING SOLUTIONS WITH APPLICATIONS
OR PRELIMINARY MARINE VEHICLE DESIGN

Personal Author(s) James R. Plosay

a Type of Report aster's Thesis	13b Time Covered From To	14 Date of Report *(year, month, day)* September 1990	15 Page Count 144

Supplementary Notation The views expressed in this thesis are those of the author and do not reflect the official policy or po-
ion of the Department of Defense or the U.S. Government.

Cosati Codes			18 Subject Terms *(continue on reverse if necessary and identify by block number)*
ld	Group	Subgroup	Preliminary ship design, VAX/VMS programming, VAX graphics

Abstract *(continue on reverse if necessary and identify by block number)*

A Mechanical Engineering Department project in which the VAX/VMS system was utilized to create an interactive menu
iven program to solve basic preliminary ship design problems.

Enhancement of an existing program was initiated to improve the user interface by adding user-friendly help information.
so, routines were written to calculate propulsive power requirements based upon the ship form coefficients selected and
mparisons made using the Method of Silverleaf and Dawson and the Admiralty Coefficient prediction method. Further
mputational routines were added to predict range and endurance figures for estimated voyage data and selected propulsion
ant types, using the U.S. Navy Design Data Sheet DDS9400-1 methodology. Finally, the detailed printed report generated
the system was updated to include reports of these calculations for the users design study.

Distribution Availability of Abstract unclassified/unlimited ☐ same as report ☐ DTIC users	21 Abstract Security Classification Unclassified	
a Name of Responsible Individual A. Papoulias	22b Telephone *(include Area code)* (408) 646-3381	22c Office Symbol ME/Pa

D FORM 1473,84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

security classification of this page

Unclassified

Enhanced VAX/VMS Programming Solutions with Applications for Preliminary Marine Vehicle Design

by

James R. Plosay
Lieutenant, United States Navy
B.S.,Nuclear Engineering, Pennsylvania State University,1983

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 1990

[illegible]

[illegible]

Anthony J. Healey, Chairman,
Department of Mechanical Engineering

ABSTRACT

A Mechanical Engineering Department project in which the VAX/VMS system was utilized to create an interactive menu driven program to solve basic preliminary ship design problems.

Enhancement of an existing program was initiated to improve the user interface by adding user-friendly help information. Also, routines were written to calculate propulsive power requirements based upon the ship form coefficients selected and comparisons made using the Method of Silverleaf and Dawson and the Admiralty Coefficient prediction method. Further computational routines were added to predict range and endurance figures for estimated voyage data and selected propulsion plant types, using the U.S. Navy Design Data Sheet DDS9400-1 methodology. Finally, the detailed printed report generated by the system was updated to include reports of these calculations for the users design study.

THESIS DISCLAIMER

A. SOFTWARE DISCLAIMER

The reader is cautioned that computer programs developed in this research may not have been exercised for all cases of interest. While every effort has been made, within the time available, to ensure that the programs are free of computational and logic errors, they cannot be considered validated. Any application of these programs without additional verification is at the risk of the user.

B. COPYRIGHTS AND TRADEMARKS USED

The following copyrights and trademarks are used throughout the text of this thesis and are considered the property of their registering corporations and owners:

Apple	A trademark of Apple Computer Corporation
Lisa	A trademark of Apple Computer Corporation
MacIntosh	A trademark of Apple Computer Corporation
ASSET	A trademark of Boeing Company
CA-DISSPLA	A trademark of Computer Associates, Incorporated
DEC	A trademark of Digital Equipment Corporation
VAX	A trademark of Digital Equipment Corporation
VAX/VMS	A trademark of Digital Equipment Corporation
MS Windows	A trademark of Microsoft Corporation
MS-DOS	A trademark of Microsoft Corporation
Star	A trademark of XEROX Corporation

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

A. BACKGROUND OF GRAPHICAL USER INTERFACES

Advances in the electronics industries since the 1970's have enabled the development of increasingly more capable and complex computer systems. To harness these systems, software engineers have produced programs that enable todays users to have a degree of computational power that was unfathomable just 10 years ago. As a result, software programs have been increasing in complexity every year. This has turned out to be a double edged sword; on the one hand the desktop computer power has enabled the development and solution of problems that would require the resources of a mainframe computer not too long ago, but it has also meant that the user is confronted with the requirement to comprehend all of this capability. This has led to the development of user interfaces that seek to simplify and manage the task confronting the user [Ref. 1: pp. 115-139]. Among these user interfaces have been a host of graphical systems that attempt to simplify the program presentation to the user by means of visual icons and multiple screen windows for specific tasks. These interfaces are termed *Graphical User Interface*, or GUI for short.

The very first development of the GUI occurred in the late 1970's at XEROX Corporation's Palo Alto Research Center (PARC) and was embodied in the XEROX *Star Office System* computer [Ref. 2]. The *Star System* was not agressively marketed and languished in the PARC laboratories until 1984 when its basic ideas were consolidated into the APPLE Corporation's *Lisa*, and subsequently the *MacIntosh* system, which was ultimately more commercially successful and exists even this day. Since that time several other GUI's have been developed and popularized for the express purpose of interfacing between man and machine [Ref. 1 : pp. 337- 371].

B. 'TOOL BOX' AND PRELIMINARY SHIP DESIGN

This brings us to the point of developing a program that will enable a user to solve basic preliminary ship design problems without necessarily being completely familiar with the inner details of the program code. Having at our disposal a Digital Equipment Corporation (DEC) VAX/VMS Network, we have taken a basic ship design program previously developed [Ref. 3] at the Naval Postgraduate School and enhanced its usefulness by the careful application of graphical presentations. This program as it existed previously computed some basic ship static and hydrodynamic stability parameters, predicted the ship's turning circle, and printed a detailed hard copy report of those calculations if so desired. It began the design process utilizing basic GUI principles but did not reach its potential with respect to its ability to communicate with the user. This does not at all reflect upon the abilities of its developer, but only indicates the scope of the software developement task. In the short time available to the preliminary author, he was able to implement many of the VAX Systems User Interface Services (UIS) routines in the initial version. However, time and other constraints kept him from a full implementation of the VAX systems capabilities for graphical presentation. This is the point where this author has engaged the project and made some important enhancements of the basic code and also some additions to its computational abilities.

Figure 1 on page 3 (from Ref. 4, p. 15), shows a generally accepted version of the Iterative Design Spiral, typically used in Preliminary Ship Design to refine a design as conditions and data become more concrete. *TOOL BOX* , as it existed, sought to define the early parameters required for the design, and attempted to show quickly that the design would be stable and maneuverable, essentially completing the boxes labeled (a) and (b) in Figure 1. From this point we will progress further along the iteration pattern and provide the calculations indicated in box (c) of Figure 1, and go back and try to enhance some of what was accomplished before. The modular concept of *TOOL BOX* supports this type of application where the design calculations will be performed time and again, as the design data is finalized. Thus each module will provide the specific calculation required by one of the Design Spirals major elements, and the modules can be accessed in any order required.

Additionally, we will add some commonly utilized GUI features to enhance the programs ease of use and information presentation, and attempt to make full use of the VAX/VMS Operating Systems capabilities for Graphical Programming in High Level Languages.

C. WHY WINDOWING SCHEMES?

One common feature shared by most, if not all GUI's, is the ability to display multiple windows on screen simultaneously, with separate windows for separate tasks. This scheme has the advantage of concentrating the user's attention on a certain window that is considered the *active* window. Windowing GUI's are present in all forms of computing environments today, from *Microsoft Windows* for MS-DOS based systems, to the Apple *MacIntosh*, to a variety of systems for UNIX and its derivatives, and many others.

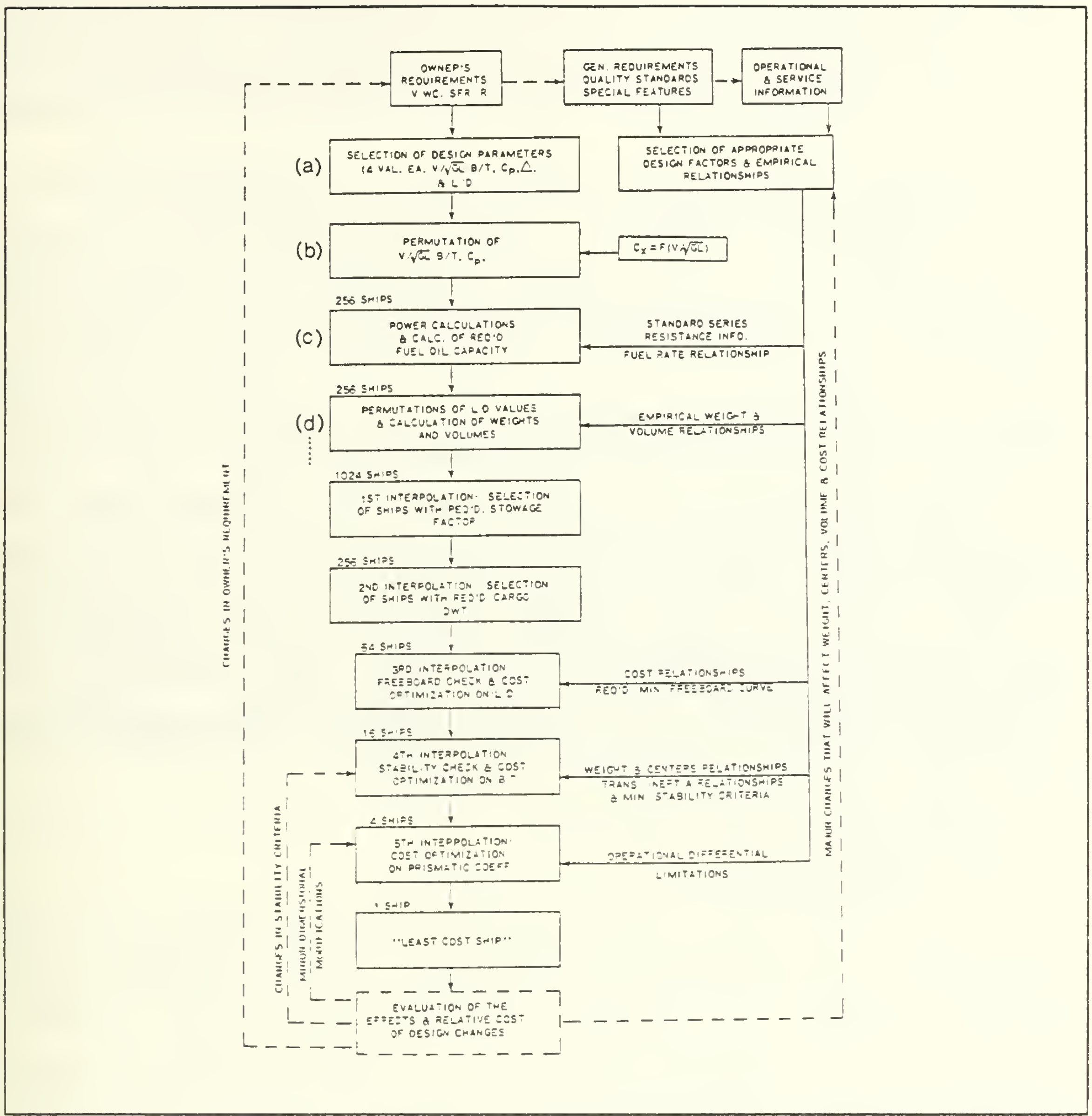

OWNER'S REQUIREMENTS V WC. SFR R
GEN. REQUIREMENTS QUALITY STANDARDS SPECIAL FEATURES
OPERATIONAL & SERVICE INFORMATION
(a)
SELECTION OF DESIGN PARAMETERS (4 VAL. EA. V/√GL B/T, Cp, Δ. & L/D
SELECTION OF APPROPRIATE DESIGN FACTORS & EMPIRICAL RELATIONSHIPS
(b)
PERMUTATION OF V/√GL B/T, Cp.
Cx = F(V/√GL)
256 SHIPS
(c)
POWER CALCULATIONS & CALC. OF REQ'D FUEL OIL CAPACITY
STANDARD SERIES RESISTANCE INFO.
FUEL RATE RELATIONSHIP
256 SHIPS
(d)
PERMUTATIONS OF L/D VALUES & CALCULATION OF WEIGHTS AND VOLUMES
EMPIRICAL WEIGHT & VOLUME RELATIONSHIPS
1024 SHIPS
1ST INTERPOLATION: SELECTION OF SHIPS WITH REQ'D. STOWAGE FACTOR
256 SHIPS
2ND INTERPOLATION SELECTION OF SHIPS WITH REQ'D CARGO DWT
64 SHIPS
3RD INTERPOLATION FREEBOARD CHECK & COST OPTIMIZATION ON L/D
COST RELATIONSHIPS
REQ'D MIN FREEBOARD CURVE
16 SHIPS
4TH INTERPOLATION STABILITY CHECK & COST OPTIMIZATION ON B/T
WEIGHT & CENTERS RELATIONSHIPS
TRANS INERT A RELATIONSHIPS & MIN STABILITY CRITERIA
4 SHIPS
5TH INTERPOLATION: COST OPTIMIZATION ON PRISMATIC COEFF
OPERATIONAL DIFFERENTIAL LIMITATIONS
1 SHIP
"LEAST COST SHIP"
EVALUATION OF THE EFFECTS & RELATIVE COST OF DESIGN CHANGES
CHANGES IN OWNER'S REQUIREMENT
CHANGES IN STABILITY CRITERIA
MINOR DIMENSIONAL MODIFICATIONS
MAJOR CHANGES THAT WILL AFFECT WEIGHT, CENTERS, VOLUME & COST RELATIONSHIPS

Figure 1. Iterative Design Spiral (typical)

However windowing systems do allow for the presentation of information to the user in a manner not obtainable by command-line systems.

For example, one of the most powerful ship design programs available today is *ASSET* from Boeing Company, based in Seattle, Washington. This program employs strictly a command-line interface where the user inputs various commands to the system and awaits a prompt to continue. Nevertheless, for all the power available in *ASSET*, it suffers from this command-line interface in the worst fashion. A user either has to know the commands available and all the options for them, or they must have the reference manual handy at all times. Since manuals are generally not available for all users at once, it is ***considerably harder*** to master all of its features without constant use and practice.

My primary goal in this project was to enhance the methods used by *TOOL BOX* in which information was presented to the user by utilizing some basic GUI presentation methods such as 'pop-up' dialog boxes that would convey some important or useful information to the user at the appropriate time, then disappear when *not* required. This form of windowing has the advantages of getting the user's attention quickly since the display screen changes in a most abrupt fashion. Since the user will now be concentrating on the small piece of information conveyed by the dialog box, and any options will be displayed for them at this time, no requirement for reference manuals or memorization is needed. This technique is employed by all the GUI's available today, so the method is consistent with current software design practice. GUI traits that we will incorporate will be timed display of choices or information, visual prompting for needed input, multiple display of windows, and so forth. Also, we shall improve the calculational power embodied in *TOOL BOX* by adding other routines based around the same central core code and utilizing the same interface.

D. STANDARDS AND CONVENTIONS

1. Attribute Blocks

Attributes are used to describe the basic default format representation of text and graphics. These attributes are stored in blocks that can be modified by the programmer. Whenever any modification (for example the screen font) is desired, it is necessary to copy the basic attribute block 0 to another numbered block and then modify the attribute concerning screen fonts. Otherwise a previously modified block can be used as the basic block and modified again. Attribute block 0 can never be changed, as it contains all the system defaults. All previous modifications remain unless changed by the current modification. The system programmers references [Ref. 5 Chapter 9, section 2.2] contain more on what modifications can be performed and how to accomplish the desired changes. The previous author has utilized blocks 1 thru 10 for his preliminary code; in order to ensure that we don't mistakenly modify a block that is in use in some other undocumented portion of the program, we will use blocks 20 thru 50 for our additions to the code. The documentation for the previous section of code is incomplete, thus we will skip blocks 11 thru 19 as a safety feature. The DEC VAX/VMS system allows up to 255 different attribute blocks, so we should have no problem utilizing separate blocks for this part and any subsequent additions. Optimization and minimization of code and resources is not an objective at this point in the development.

2. Code Documentation

This is the standard we will use to add documentation and comments into the code section to explain each step to a reader. All too often a source code that has not been documented internally by the author will be reviewed by another programmer and the lack of clear concise documentation will lead to lengthy frustration. Therefore, our method will be as follows;

- Comments in the source code shall begin with **C*****, and every line that is not intuitively obvious will have a comment preceeding it.
- Subroutines will be preceeded by a description of the purpose and utilization of the routine. If the routine is called by more than one section, it will be noted.
- New sections or changes added by this author to existing code will be annotated as changed, so credit may fall to the appropriate author.
- Complete copies of the TOOL BOX source code, including parts not covered in this document, may be obtained from Professor F.A.Papoulias, Code ME/Pa, Naval Postgraduate School, Monterey CA 93943-5000 or from Professor J.F. Hallock, Code ME/Hl, Naval Postgraduate School, Monterey CA 93943-5000.

II. DIALOG BOXES

A. FUNCTION OF THE DIALOG BOX

The use of 'pop-up' dialog boxes is fundamental in the operation of most GUI's in that, at some point during the execution of the program, some element of information will need to be conveyed to the user. *TOOL BOX* is designed in this manner to overcome the limitations of command-line based ship design analysis programs thus every effort will be made, mostly via context sensitive dialog boxes, to present information to the user as it is required **without requiring the reference manual to be handy**. Basically the dialog boxes function as follows:

1. The point of the program that the author deems important calls the dialog box routine.
2. The dialog box is presented, displaying the information or choices available to the user. This involves VAX UIS routines to create a display, to create a window, to modify the text as the author desires for presentation, and to execute its function.
3. If some action is required by the user, the program waits for that action.
4. If only information is to be displayed, then a suitable time is given for it to be viewed by the user.
5. Upon completion the dialog box is removed from view by the system. This step involves VAX UIS routines to time the display to completion, delete the display (and associated keyboards, windows, and viewports) and remove itself from the screen.

The best resource for the actual routines used and the parameters required is the specific system manual [Ref. 5 Chapter 18]. This is an indispensable source for the VAX/VMS programmer.

B. TYPES OF DIALOG BOXES

Two distinct types of dialog boxes were required to be used. The similarities are apparent, yet each functions differently enough to warrant separate discussion.

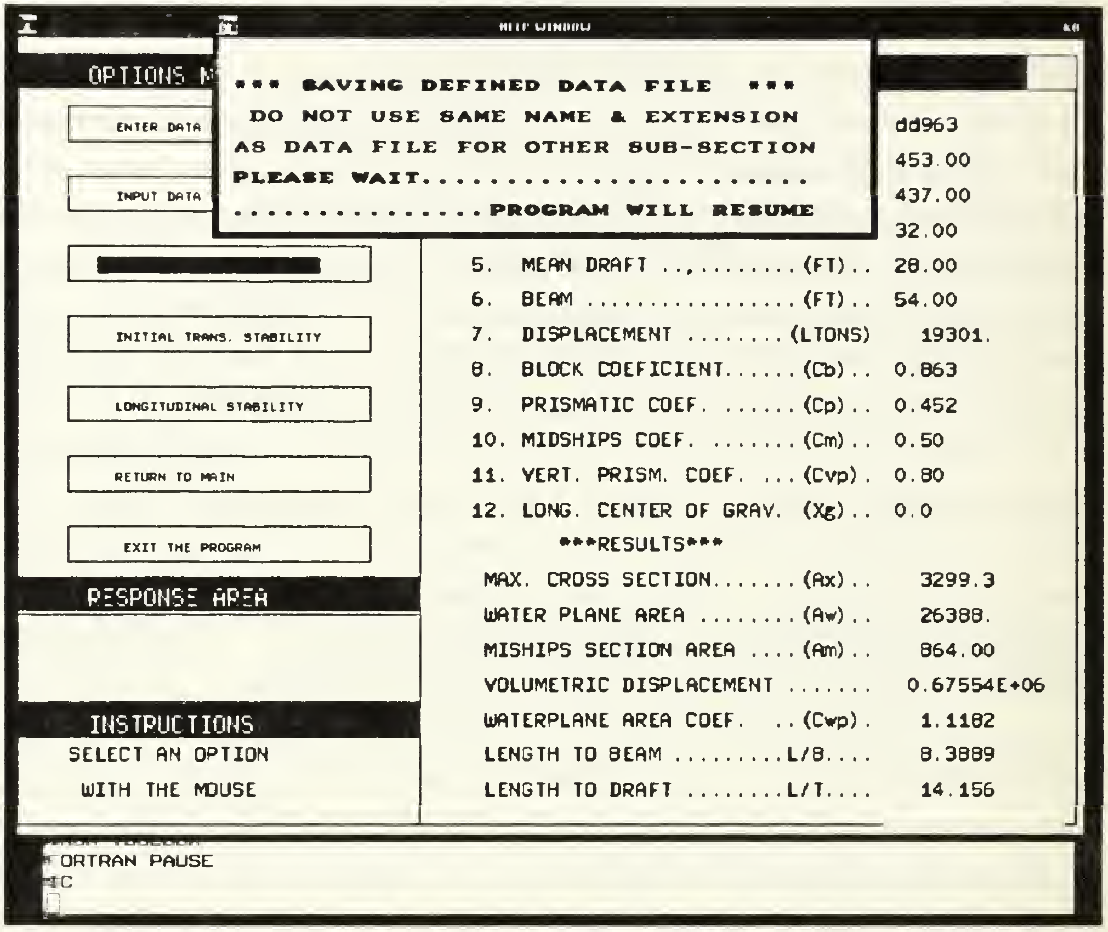
1. Visual Display Information Box

This type of dialog box is used to display information to the user at the appropriate time. No conversation between the user and the system is required, therefore the routine can be fairly simple. An example of this type of display is the subroutine **SHOW_SAVE** used during any attempt by the user to save a data file. Its purpose is simple: it reminds the user not to save the file under the same name and extension as was used previously in another part of the program, since this will cause the new data file to overwrite the old one. This subroutine code is simple and is reproduced in Appendix A. Essentially the subroutine is called during an attempt to save a data file by the system. It pops up and writes to a new display box and window the message about using a different filename and extension. It does not need any action by the user, as it is for informational purposes only. Since no action is required, it must remove itself from display after an appropriate time has passed, yet give the user enough time to read and comprehend its message. For this time delay we have used a system library call to the VAX function **LIB$SPAWN('WAIT 00:00:15')**. This function 'spawns' a new process that calls the intrinsic VAX function **WAIT** for a delay set as 15 seconds. The time format is HH:MM:SS in hours, minutes, and seconds. Upon completion of this delay, the display window is deleted and the system proceeds at the point the delay process was spawned. This method is rather elegant in that only system functions are used for the delay. The UIS routines required for this action are **UIS$CREATE_DISPLAY**, **UIS$CREATE_WINDOW**, **UIS$SET_FONT**, **UIS$SOUND_BELL**, **UIS$TEXT**, and **UIS$DELETE_DISPLAY**. The font and bell routines are optional; however, they add to the display by making it both audible and visually different than the existing display when the window is created. Also, the bell volume is fully variable by the programmer. A value of '0' for the second argument will set the volume to its most quiet position, and an '8' will set it to the loudest position. We have used a value of '4' which creates a medium loud bell, loud enough to draw attention but not to cause the user to be startled. If no volume is specified, then the system defaults the bell volume to the workstation volume setting. The window created by this process is shown as Figure 2 on page 8 below.

The presentation of this display lasts approximately 15 seconds depending upon the number of windows and processes currently open on the terminal, and so will be slightly longer if multiple concurrent process are active. Upon removal of itself, the display returns to the window that was active when the dialog box routine was called.

2. Conversational Dialog Box

This type of dialog box is used to display information to the user and also to give the user a choice for proceeding. An example of this type of procedure is the subroutine **NOWHERE** that is called by the mouse selection of a Reserve Module from the Main Menu. This routine uses essentially the same UIS routines as the previous method, but also incorporates a FORTRAN **PAUSE** statement that interrupts the progress of the program. Using the **PAUSE** statement here gives the user two choices:



```
OPTIONS M
ENTER DATA
INPUT DATA
INITIAL TRANS. STABILITY
LONGITUDINAL STABILITY
RETURN TO MAIN
EXIT THE PROGRAM
RESPONSE AREA
INSTRUCTIONS
SELECT AN OPTION
 WITH THE MOUSE

*** SAVING DEFINED DATA FILE   ***
 DO NOT USE SAME NAME & EXTENSION
AS DATA FILE FOR OTHER SUB-SECTION
PLEASE WAIT.........................
................PROGRAM WILL RESUME

                                  dd963
                                  453.00
                                  437.00
                                  32.00
5.  MEAN DRAFT ..,........(FT)..  28.00
6.  BEAM .................(FT)..  54.00
7.  DISPLACEMENT ........(LTONS)    19301.
8.  BLOCK COEFICIENT......(Cb)..  0.863
9.  PRISMATIC COEF. ......(Cp)..  0.452
10. MIDSHIPS COEF. .......(Cm)..  0.50
11. VERT. PRISM. COEF. ...(Cvp).  0.80
12. LONG. CENTER OF GRAV. (Xg)..  0.0
        ***RESULTS***
 MAX. CROSS SECTION.......(Ax)..    3299.3
 WATER PLANE AREA ........(Aw)..    26388.
 MISHIPS SECTION AREA ....(Am)..    864.00
 VOLUMETRIC DISPLACEMENT .......     0.67554E+06
 WATERPLANE AREA COEF.   ..(Cwp).   1.1182
 LENGTH TO BEAM .........L/B....    8.3889
 LENGTH TO DRAFT ........L/T....    14.156

FORTRAN PAUSE
C
```

Figure 2. Visual Display Information Dialog Box

1. Enter 'CONTINUE' or 'C' and the program will resume at the point of interruption.

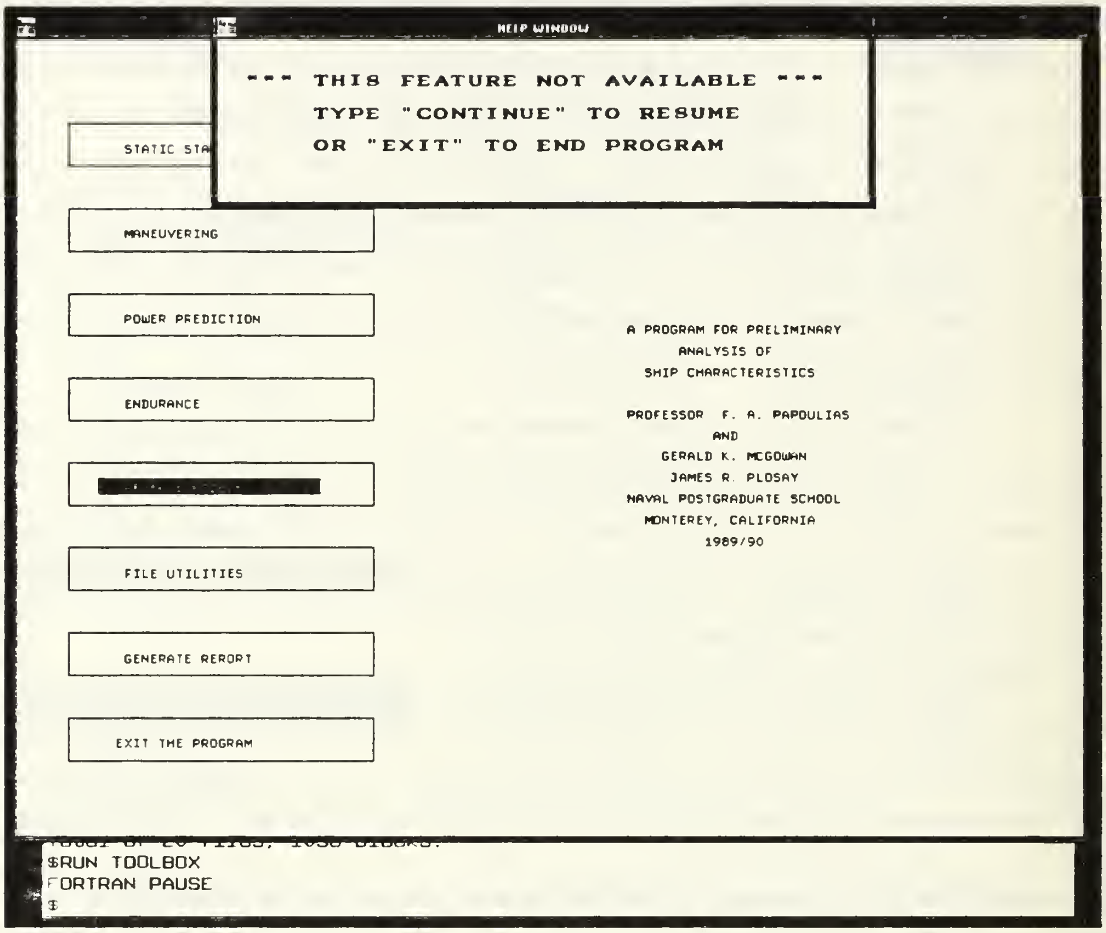
2. Enter 'EXIT' and the program will terminate at that point, deleting all displays and windows associated with it, and return the user to the active process that was used to call the program TOOL BOX.

This selection of choices was chosen since at this point the user is in the Main Menu Window and the direction to proceed will be either, (1) to continue if that selection is entered, or (2) to end the program and exit if the user has selected these future modules just out of curiosity. The user need not see what has been typed at the terminal when a choice is made; however, if the calling window is visible at the bottom of the screen as in Figure 3 on page 10, then the typed text reply will appear here. This is only necessary if the user happens to be a poor speller, since the program will not proceed without the entry of a properly spelled command. Thus the VAX shorthand command for ' **CONTINUE**' as 'C' is best utilized. Otherwise, the user enters a choice by typing on the terminal keyboard followed by a carriage return, and the program proceeds along that selection. The result of this process is shown in Figure 3 on page 10 after presentation of the dialog box.

No timing of the display is required since it will remain there until a proper command is entered. The routine **UIS$SOUND_BELL** is used again to signal audibly the user and gain his attention; the bell volume again is set to a medium volume at '4'. The code for this routine is reproduced in Appendix A.

3. Other Enhancements

Although we have not used the technique here, is is possible to have a conversational dialog box that shows the users reply in the window that requests it. For this routine, we would use essentially the same format as in the subroutine **NOWHERE**, except we would attach and enable a keyboard to the display by utilizing the UIS routines **UIS$CREATE_KB** and **UIS$ENABLE_KB**. Then a small routine would have to be added to accept and display keyboard input into the window. This is not difficult; and it might be helpful by allowing the user to see what has been typed as their reply, thereby alleviating some errors caused by typing mistakes. This technique has not been used here since there are only two allowable responses when using the FORTRAN **PAUSE** statement; **CONTINUE** and **EXIT**. Any other response will be ignored without exception and no action will be taken by the program until a proper response has been entered. Upon completion of the routine, the attached keyboard would have to be disabled and deleted by UIS routines and the text reply passed to the calling program as input for some action. This sequence is possible and some use may be found for it later in the development.


HELP WINDOW

*** THIS FEATURE NOT AVAILABLE ***
TYPE "CONTINUE" TO RESUME
OR "EXIT" TO END PROGRAM

A PROGRAM FOR PRELIMINARY
ANALYSIS OF
SHIP CHARACTERISTICS

PROFESSOR F. A. PAPOULIAS
AND
GERALD K. MCGOWAN
JAMES R. PLOSAY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA
1989/90

```
$RUN TOOLBOX
FORTRAN PAUSE
$
```

Figure 3. Conversational Dialog Box Display

III. PROPULSIVE POWER REQUIREMENTS

A. BACKGROUND

One of the more useful pieces of information to have during preliminary ship design is an estimate of the vessel's propulsion plant power requirements. This is especially handy since the *size* of the plant required to drive the vessel will also give an initial estimate of volume required to enclose it, manpower required to operate it, and the cost to purchase and maintain it. So a single 'ball-park' figure will go a long way towards helping to advance the preliminary design.

One of the foremost sources of information and research conducted in this calculation was Rear Admiral David W. Taylor, USN(ret,dec), for whom the David Taylor Naval Ship Research and Development Centers (DTRC) in Carderock and Annapolis, Maryland were named. His studies for the then US Navy Construction Corps at the beginning of this century still serve as required reading for students of ship design and naval architecture [Ref. 6]. Some of this research has been duplicated and enhanced by more recent scholars utilizing more powerful calculational abilities than Adm. Taylor ever envisioned. These studies [Refs. 7, 8 , 9, 10] have resulted in some variations of Adm. Taylor's basic equations and will be examined here for two methods of power prediction that we will utilize.

B. METHODOLOGY

Many methods of power prediction exist in the literature and a short amount of research will produce most of them. The list of methods begins with the 'Quick and Dirty' method of Admiralty Coefficients, which assumes that an already established hull form (parent ship) with its basic parameters is known and that it is simply being modified for the new design. This method has been slightly modified [Ref. 8 : pp. 308- 310] and is used here in that form. There are several graphical methods for power prediction that require charted information on the specific hull form or a family of hull forms close to the design in question. These methods will not be used since those charts would have to be available in digitized form for all ship classes of interest. There is also an entire class of prediction calculations based upon the total estimated hull form resistance to movement through the water. This method will not be used here because it entails much more detail than we desire, or have at this stage of design. Then there is a method originally proposed by Silverleaf and Dawson [Ref. 9 : pp. 167- 196] and subsequently modified by Stian Erichsen [Ref. 10 : pp. 83- 115] in a University of Michigan College of Engineering departmental paper in 1971. This method will also be presented here and the user can choose the desired result with care.

Some notes about the methods presented here must be discussed first, in order that no potential conflicts arise later on:

- These methods were developed for monohull surface craft only; no capability to predict power requirements for catamaran, trimaran, SWATH, etc., types of vessels is intended.
- The resulting power requirement figures do not include added resistance due to waves, fouling, or other environmental factors.

1. Method of Admiralty Coefficients

The first method to be used is based upon the Admiralty Coefficient since a great deal of information about that Coefficient is available in the literature. This method assumes that the ship's resistance is all frictional, and thus the power required for propulsion varies proportionately as the cube of the ships speed. The Admiralty coefficient A_c for a vessel when no parent vessel is used is given by :

$$A_c = 3.70(\sqrt{L} + \frac{75.0}{V}) \qquad (1)$$

where L is length of vessel in meters between forward and aft perpendiculars and V is speed of vessel in meters per second. Otherwise, if a parent vessel is available, the Admiralty coefficient is given as

$$A_c = \frac{\Delta^{2/3} V^3}{P_s} \qquad (2)$$

where P_s is parent ship power in kilowatts needed to make a speed of V meters per second and Δ is parent ship displacement in metric tonnes.

Then, using that coefficient to characterize the vessel, we take

$$P_n = \frac{\Delta^{2/3} V^3}{A_c} \qquad (3)$$

where P_n is power in kilowatts of the new ship needed to make the same speed V and Δ is the new ships displacement mass in tonnes.

An improved version of this formula that incorporates almost all of the essential ship parameters except that it neglects the influence of the block coefficient C_B is given by

$$P = \frac{5.0\Delta^{2/3}V^3(33.0 - 0.017L)}{15{,}000 - 110n\sqrt{L}} \tag{4}$$

where n is propulsion shaft revolutions per second needed to make a speed of V meters per second and all other parameters are as described previously above in metric units. This form is easily equated to our more common form of English Horsepower (hp) units by a simple conversion factor, and is the equation we will use for our first method of power calculation in the program. Also, some obvious limitations to this formula appear readily enough and should be noted :

- L cannot be larger than 1941.177 meters or the numerator becomes a negative value.
- The product of shaft speed n in revolutions per second and the square root of L cannot exceed 136.364 or the denominator becomes a negative value.
- A realistic maximum shaft speed of 4 revolutions per second (240 rpm) limits ship length even further to 1162.196 meters. This is not too much of an overall limitation, but it is present nonetheless.

These limitations will not affect us for the most part but should be noted in any event as they could end up being important. It should be noted further that this formula (4) tends to fit special cases and types of ships. When used outside of these allowable ranges, the results tend to produce erroneous expectations. According to Harvald, [Ref. 7 : p. 290], "It is not often mentioned within which area the formulas can be applied". Thus we have a method that can be quite accurate for power prediction, *if we happen to fall within the allowable range*, and which otherwise could generate a false prediction. For the first method however, it will be satisfactory for an initial estimate. We can then couple these results with the second method. If the results are close, we can assume that our prediction is reasonable. If they depart, then we will have to use them with caution.

2. Method of Silverleaf and Dawson

The second method that we will use for power prediction is given by Silverleaf and Dawson [Ref. 9] as modified by S. Erichsen [Ref. 10]. It uses almost all of the ship specific parameters of the design and is usually highly reliable for ships within the following ranges of speed - length ratio $V_k/\sqrt{L}$ and beam - draft ratio B/T :

$$0.4 \leq \left(\frac{V_k}{\sqrt{L}} \right) \leq 1.2 \tag{5}$$

and

$$2.0 \leq \left(\frac{B}{T} \right) \leq 4.5 \tag{6}$$

with C_B and length - beam ratio L/B for single screw ships

$$0.50 \leq C_B \leq 0.86 \tag{7}$$

$$3.33 \leq \left(\frac{L}{B} \right) \leq 9.50 \tag{8}$$

or for twin screw ships

$$0.54 \leq C_B \leq 0.80 \tag{9}$$

$$3.80 \leq \left(\frac{L}{B} \right) \leq 11.50 \tag{10}$$

Where the parameters are specified as follows for these and the following equations :

V_k is ship speed in knots

V_B is the ships Boundary Speed, which is that speed, for a given hull form, below which the resistance coefficient does not vary greatly, and above which it begins to increase rapidly, [Ref. 9 : p. 168].

L is ship length between perpendiculars in feet

B is ship beam in feet

T is ship draught in feet

Δ is displacement mass in tons

C_B is the Block Coefficient given by

$$C_B = \frac{35\Delta}{LBT} \tag{11}$$

for ships in standard seawater.

The Silverleaf and Dawson formula then is given by

$$P_d = \frac{(1+x)\Delta^{2/3} V_B^3 K_{BT} K_V K_B}{427.1\eta_0 (H_{400} K_{LBP})} \tag{12}$$

where P_d is delivered horsepower required at the propeller. Equation (12) will be used as the second method of power calculation in the program development. The terms in this formula are all specified for design parameters that may or may not be same as the model from which the formula was derived. Thus some corrections need to be applied to each parameter below. For a design that does not exactly match the model design, then :

Utilizing ship speed at the Boundary Speed V_B expressed in knots :

$$V_B = (1.7 - 1.4C_B)\sqrt{L} \tag{13}$$

Correcting for ship Length :

$$(1+x) = \begin{cases} 0.85 & \text{for } L > 1000.0 \\ 0.85 + 0.00185\left[\dfrac{1000.0 - L}{100.0}\right]^{2.5} & \text{for } L \leq 1000.0 \end{cases} \tag{14}$$

Correcting for the Beam - Draft ratio :

$$K_{BT} = \begin{cases} 0.982 & \text{for } B/T < 2.4 \\ 0.96 + 5.4{*}10^{-4}(10^{0.67}\frac{B}{T}) & \text{for } B/T \geq 2.4 \end{cases} \tag{15}$$

Correcting for ship speed V_k not equal to the Boundary speed V_B :

$$K_V = \begin{cases} 2.75 - 7.25(\dfrac{V}{V_B}) + 5.5(\dfrac{V}{V_B})^2 & \text{for } V/V_B \leq 1 \\ 21.2 - 43.2(\dfrac{V}{V_B}) + 23.0(\dfrac{V}{V_B})^2 & \text{for } V/V_B > 1 \end{cases} \tag{16}$$

Correcting for a ship hull that benefits from a design with a bulbous bow :

$$K_B = \begin{cases} 1.00 & \text{for ships without a bulbous bow} \\ 0.95 & \text{for ships with a bulbous bow} \end{cases} \tag{17}$$

Correcting for Hydrodynamic Efficiency of the hull form at $L = 400.0$ feet :

$$H_{400} = \left| \begin{array}{ll} 2.60 - 0.2917 \dfrac{V}{\Delta^{1/6}} & \text{for a single screw} \\ 2.38 - 0.2917 \dfrac{V}{\Delta^{1/6}} & \text{for twin screws} \end{array} \right. \tag{18}$$

with a correction for ship length L not equal to 400.0 feet :

$$K_{LBP} = \left| \begin{array}{ll} 0.9196 + 2.31*10^{-4}L - 7.5*10^{-8}L^2 & \text{for } L \neq 400.0 \\ 1.0 & \text{otherwise} \end{array} \right. \tag{19}$$

Correcting for open water efficiency at propeller speed not equal to 120 revolutions per minute, where :

$$\eta_0 = \eta_{0,120} + \delta\eta_0 \tag{20}$$

with

$$\eta_{0,120} = \left| \begin{array}{ll} 0.98 - 0.55C_B & \text{for a single screw} \\ 0.90 - 0.33C_B & \text{for twin screws} \end{array} \right. \tag{21}$$

and for propellor rotating speed N revolutions per minute

$$\delta\eta_0 = \left| \begin{array}{ll} 0.360 - 0.0029N & \text{for a single screw} \\ 0.135 - 0.0011N & \text{for twin screws} \end{array} \right. \tag{22}$$

Thus we have all the required parameters for the Silverleaf and Dawson formula.

This formula tends to be more accurate over the entire range of ship specific parameters. The parameter limitations are delineated in the beginning and cover a wide range of vessels.

C. COMPARISON OF THE METHODS

A quick comparison of the two methods utilizing equations (4) and (12) as generated by a test code of the formulas and data from established merchant designs [Ref. 4 : pp. 137- 171] is presented in tabular form in Table 1.

Table 1. COMPARISON OF POWER PREDICTION METHODS

Ship Design	Design Power (hp)	Admiralty Method (hp)	S & D Method (hp)
Large General Cargo	24,000.0	20,351.91	23,780.55
Container 'A'	17,500.0	14,127.90	11,503.22
Container 'B'	32,000.0	30,511.81	29,480.57
Roll On-Roll Off	37,000.0	28,573.58	27,435.27
LASH Barge Carrier	32,000.0	26,203.87	23,787.66
SEABEE Barge Carrier	36,000.0	26,772.95	21,918.27
Tanker 'A'	15,000.0	12,045.93	11,077.69
Tanker 'B'	45,000.0	53,068.73	36,030.91
LNG Tanker	43,000.0	39,727.77	37,910.40
Bulk Carrier	15,300.0	10,940.75	8,578.57
Ore/Bulk/Oil Carrier	24,000.0	21,700.58	18,981.25

Immediately we can draw some conclusions of the two methods used here to calculate propulsive power requirements;

- In all cases except one the Admiralty Method has underestimated the power requirements, although after an allowance for a service and fouling margin, that underestimation is not excessive. The one case where this estimation did exceed the actual power requirements (Tanker 'B') the result was not grossly excessive, and was in error of only 15 percent.
- The Silverleaf and Dawson Method also will always underestimate the power requirements by a slight margin. This margin of error for underestimation is usually not excessive and can be accounted for by service, fouling, and other margins.

A test program was used to generate the data for Table 1 since the *TOOL BOX* screen displays would consume excessive space in this paper.

Also, the following should be noted :

- These values are computed for essentially calm seaways only. No added resistance from waves in a rough sea is predicted by these methods.

- The figures shown are **not** the total plant requirement for propulsion, only that directly needed at the propeller to drive the ship. Thus margins for gear train losses, service losses, fouling losses, etc must be added for a total plant size estimation figure.
- And similarily, from Silverleaf & Dawson's formula there is also a correction for a ship design that includes a hydrodynamically well designed (WD) bulbous bow. For that case, the power requirement for propulsion is 95% of the power requirement shown, since the improved bow shape lessens the hull form resistance. Those figures are also listed in Appendix C on the next line directly below the S & D calculation.

Overall though, the two methods taken in combination do allow a fairly accurate prediction of a designs propulsive power requirements. The results generated by this module of *TOOL BOX* can thus be used to size and estimate the machinery box volume, estimate the manpower required to maintain and operate the machinery plant, estimate the cost of the propulsion plant, and so forth.

D. THE TOOL BOX POWER PREDICTION MODULE

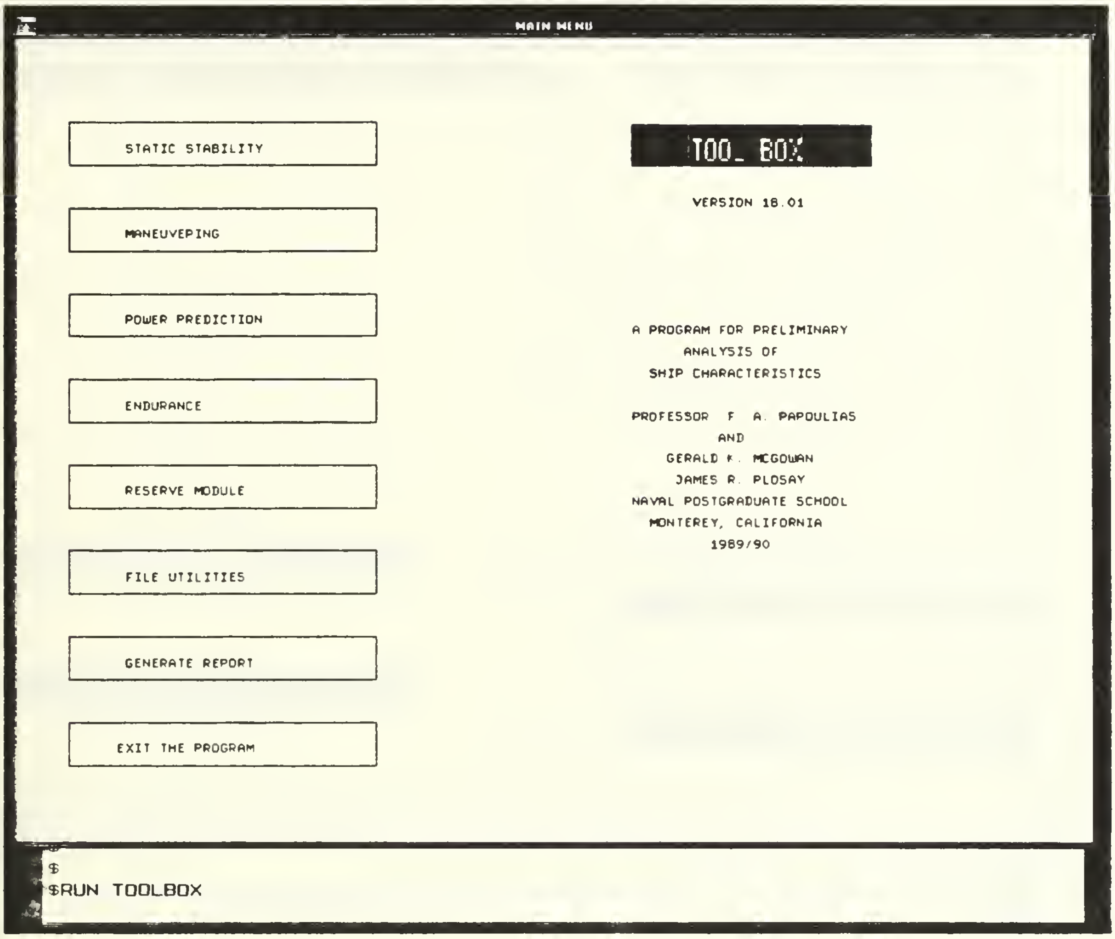
Implementation of this algorithm of code into the existing *TOOL BOX* framework constituted a fairly easy and straight forward process. An examination of the subroutine methodology of the previous modules was made and the same basic pattern was followed. The resulting code is shown in Appendix D and borrows many elements of the STATIC STABILITY module (whose flow diagram was utilized) in addition to the visual display dialog boxes for HELP WINDOWs that present pertinent information to the user and then disappear.

1. Operation of the Power Prediction Module.

The procedure used to operate the Power Prediction Module is essentially a logical progression through the available menu choices. We will progress through the screen displays in an example designed to show all the relevant points of the *TOOL BOX* Power Prediction Module. It is understood that when the user is instructed to enter a response to a program prompt, that they will follow-up that action by depressing the **Return** key on the VT220 LK201 keyboard. That action is needed to complete any data entry operation.

2. An Instructional and Illustrative Example

Shown in Figure 4 on page 19 is the TOOL BOX Opening Menu as the program is initiated by the user. This screen is common to the core code of the program, and all the program modules are initiated through this screen using the mouse device.



A PROGRAM FOR PRELIMINARY
ANALYSIS OF
SHIP CHARACTERISTICS

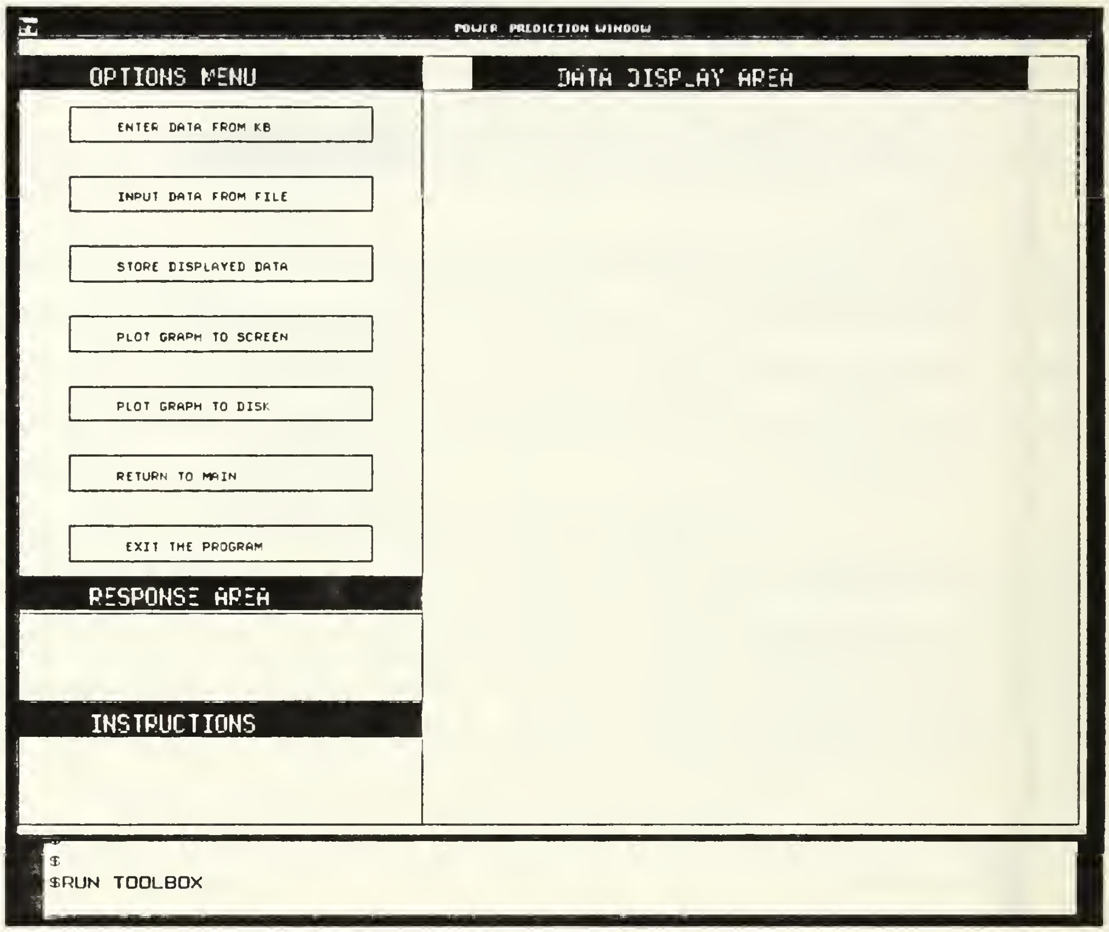
PROFESSOR F. A. PAPOULIAS
AND
GERALD K. MCGOWAN
JAMES R. PLOSAY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA
1989/90

```
$
$RUN TOOLBOX
```

Figure 4. TOOL BOX Opening Menu Screen Display

The third mouse selection available to the user is now POWER PREDICTION and a click on this selection (when the mouse pointer is in that box and the text has become reverse highlighted) will initiate this module and call up its associated display screen. The screen resulting from this selection is shown as Figure 5 on page 20.

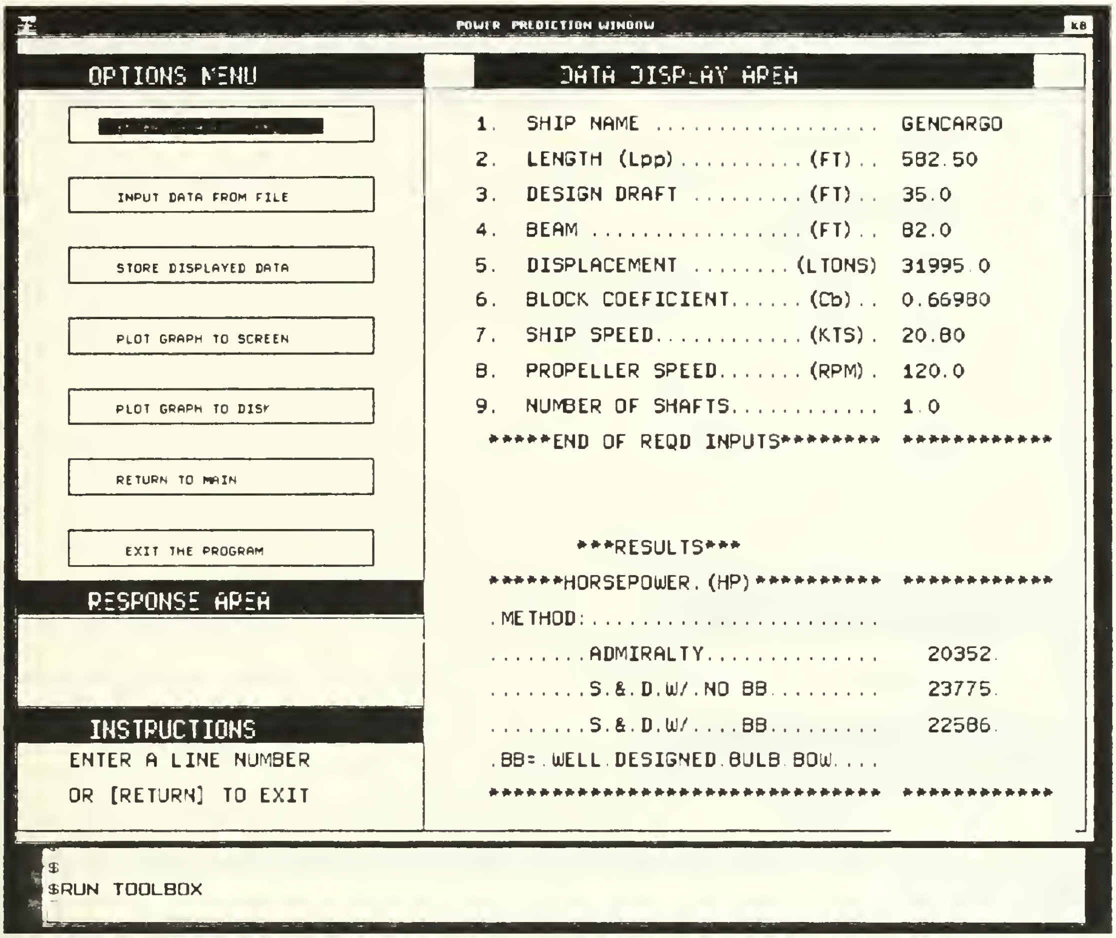
As Figure 5 is displayed to the user, it should be noted that nothing is shown in the DATA DISPLAY AREA on the right hand side. That is because no data is passed in at initialization time. A mouse click on the first menu choice ENTER DATA FROM KB will initialize the data entry subroutines and the program defaults will dis-



```
$
$RUN TOOLBOX
```

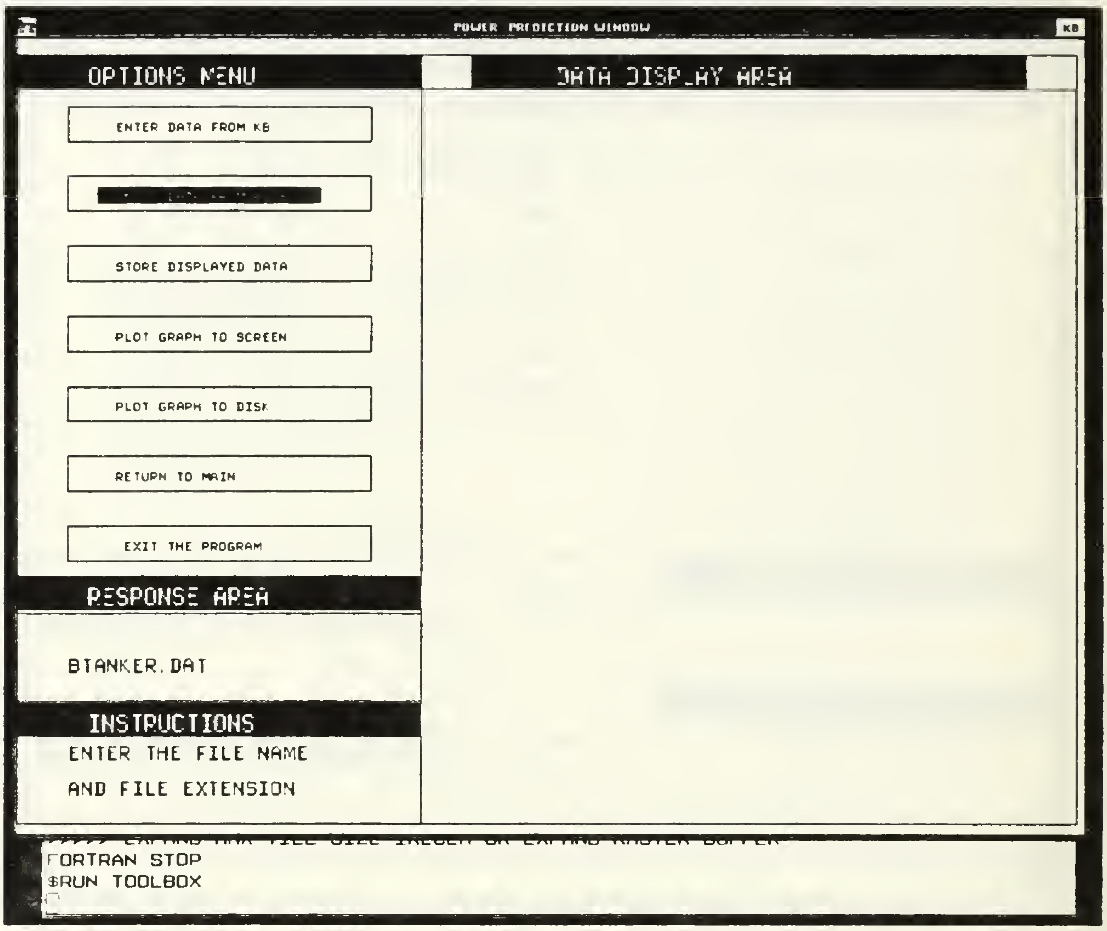
Figure 5. POWER PREDICTION MODULE Opening Menu Screen Display

play in the data region as shown in Figure 6 on page 21. Otherwise the user can click the mouse pointer on the second box INPUT DATA FROM FILE and load a predefined data file, perhaps one used before or created by the instructor. This sequence is shown as Figure 7 on page 22, with an existing data file entered in the RESPONSE AREA to be loaded for data entry.



```
POWER PREDICTION WINDOW                                             KB

OPTIONS MENU                    DATA DISPLAY AREA

[illegible]                     1.  SHIP NAME ..................   GENCARGO
                                2.  LENGTH (Lpp)..........(FT)..   582.50
INPUT DATA FROM FILE            3.  DESIGN DRAFT .........(FT)..   35.0
                                4.  BEAM .................(FT)..   82.0
STORE DISPLAYED DATA            5.  DISPLACEMENT ........(LTONS)   31995.0
                                6.  BLOCK COEFICIENT......(Cb)..   0.66980
PLOT GRAPH TO SCREEN            7.  SHIP SPEED...........(KTS).   20.80
                                8.  PROPELLER SPEED.......(RPM).   120.0
PLOT GRAPH TO DISK              9.  NUMBER OF SHAFTS............   1.0
                                *****END OF REQD INPUTS********   ************
RETURN TO MAIN

EXIT THE PROGRAM                        ***RESULTS***
                                ******HORSEPOWER.(HP)**********   ************
RESPONSE AREA                   .METHOD:.......................
                                ........ADMIRALTY.............       20352.
                                ........S.&.D.W/.NO BB........       23775.
INSTRUCTIONS                    ........S.&.D.W/....BB........       22586.
ENTER A LINE NUMBER             .BB=.WELL.DESIGNED.BULB.BOW....
OR [RETURN] TO EXIT             *******************************   ************

$
$RUN TOOLBOX
```

Figure 6.   DATA ENTRY Display Screen at Initialization of KB Entry



BTANKER.DAT

ENTER THE FILE NAME
AND FILE EXTENSION

```
FORTRAN STOP
$RUN TOOLBOX
```

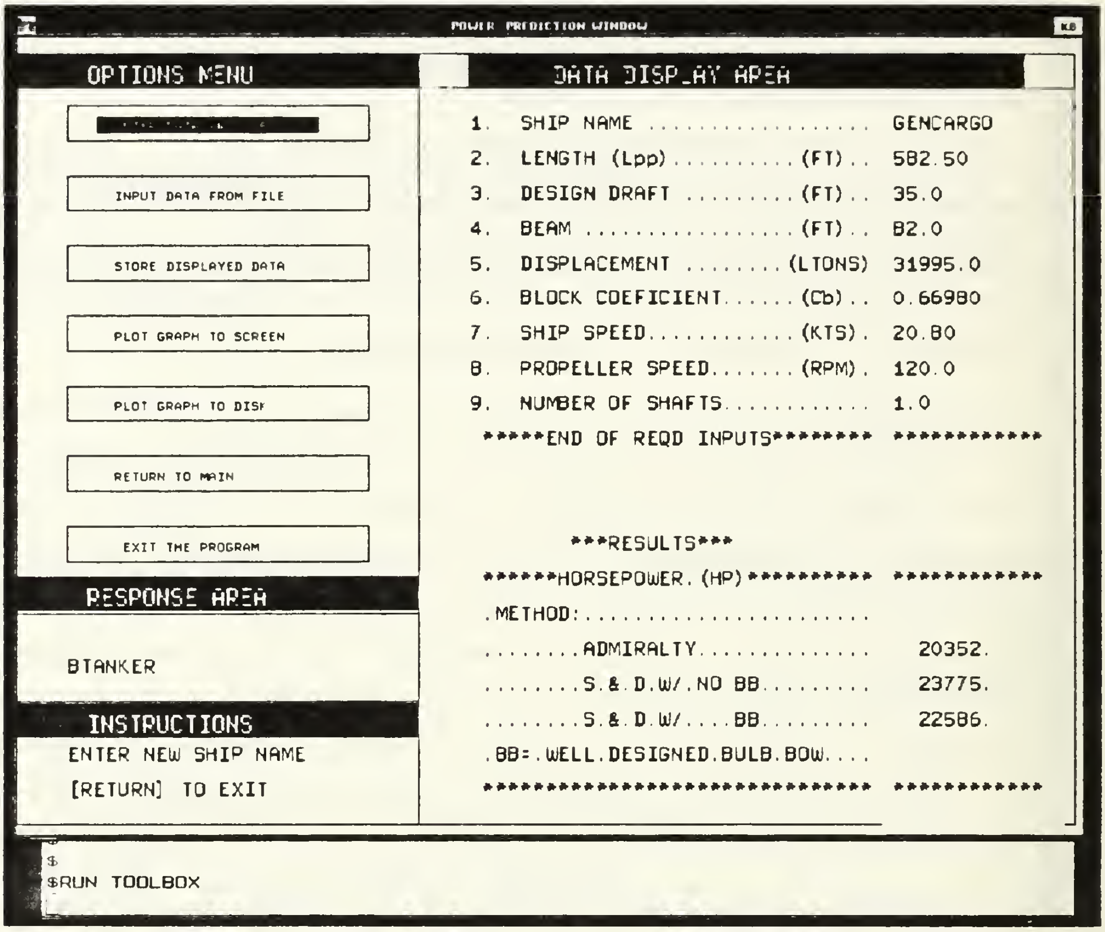
Figure 7. INPUT DATA FROM FILE Filename Entry

After the user clicks the mouse pointer on the ENTER DATA FROM KB selection, the program calls in the stored default values of ship parameters. This is done for several reasons:

- It makes little sense to see all zeroes for data by a new user. Default values will give the user an idea of what is required.
- There are so many restricting data checks on allowable entries in the algorithm (eqns. 5- 10) that the resulting HELP displays would consume inordinate amounts of time (if all initial entries were zeroes). This is because as we progress from zero length, beam, draft, displacement, block coefficient, ..., etc., to our final values the error checking codes would be called in with almost every parameter change due to the large changes in magnitude from zero to any realistic value.
- It speeds up the processing of the initial calculations.

Proceeding, note that, in the lower left hand INSTRUCTIONS box of Figure 6 on page 21, the user is prompted to enter a line number for whichever line it is desired to modify. This can be entered as an integer (1) or a real number (1.). The program code does not care specifically one way or another. Suppose we wish to modify the existing data from the set displayed, which comes from the LARGE GENERAL CARGO vessel of Table 1 on page 17, to another set from that same table for the TANKER B design. First we would need to change the data file name to a unique value for that design. We would enter a 1 for the line to change, then enter the name of our design. Notice that the user is prompted for the exact detail required for the corresponding Line number chosen, as in the INSTRUCTIONS box at the lower left corner of Figure 8 on page 24. So here we'll use **BTANKER** for our design name, as shown in Figure 8 above the INSTRUCTIONS box, where the user will also note that their data entry is mimicked in the RESPONSE AREA on the left middle screen box as shown.

This is one feature that will help the user : to have all input echoed back to the screen as it is being entered, so that changes can be made on the spot if required. After that entry is completed, this data will be written to the screen corresponding to the Line entry that was modified, in this case Line 1.



```
BTANKER
```

```
ENTER NEW SHIP NAME
[RETURN] TO EXIT
```

```
1.  SHIP NAME ..................  GENCARGO
2.  LENGTH (Lpp)..........(FT)..  582.50
3.  DESIGN DRAFT .........(FT)..  35.0
4.  BEAM .................(FT)..  82.0
5.  DISPLACEMENT ........(LTONS)  31995.0
6.  BLOCK COEFICIENT......(Cb)..  0.66980
7.  SHIP SPEED............(KTS).  20.80
8.  PROPELLER SPEED.......(RPM).  120.0
9.  NUMBER OF SHAFTS............  1.0
 *****END OF REQD INPUTS********  ************



          ***RESULTS***
 ******HORSEPOWER.(HP)**********  ************
 .METHOD:.......................
 .........ADMIRALTY.............    20352.
 .........S.&.D.W/.NO BB........    23775.
 .........S.&.D.W/....BB........    22586.
 .BB=.WELL.DESIGNED.BULB.BOW....
 *******************************  ************
```

```
$
$RUN TOOLBOX
```

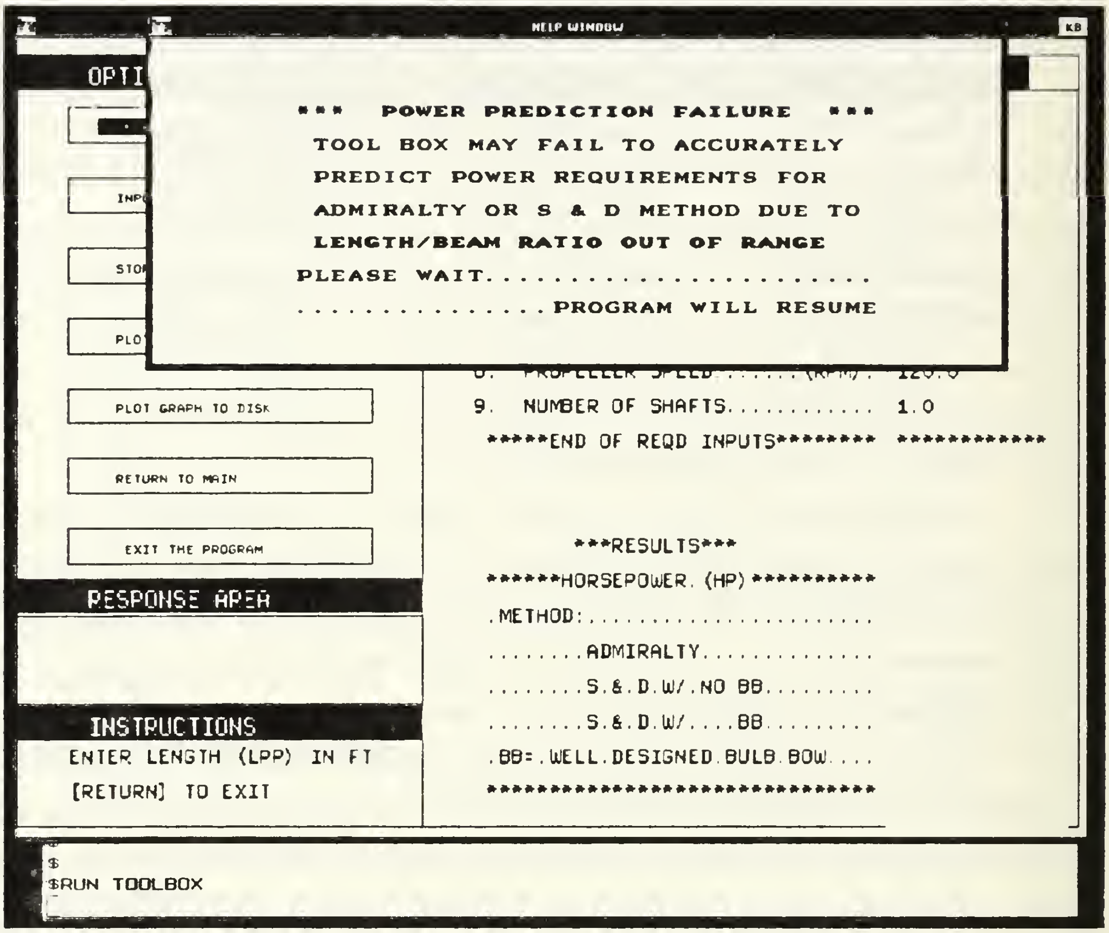
Figure 8. DATA ENTRY Screen showing Mirror of User Input Values

Continuing then, we need to modify some other design parameters for our new design, so we'll move on to Line 2, design LENGTH (LPP). For this entry we'll enter a **2** for Line 2, then the value **1143.00** that corresponds to our design length.

Almost as soon as this operation is complete, a HELP WINDOW appears because we have violated a range requirement of one of the two sets of calculations. Which specific equation we violated is unknown at this point, but the HELP WINDOW tells us that the LENGTH/BEAM RATIO IS OUT OF RANGE for one of the calculations, as shown in Figure 9 on page 26. This screen is a timed display screen of the type shown in Chapter 2 previously, so it will disappear after approximately 15 seconds. The user can keep this display on screen if desired by depressing the VT220 key F1 which corresponds to HOLD SCREEN for the VAX system. Then the screen message can be copied or noted as to what problems might have arisen.

Also notice that in Figure 9 on page 26 that the Results Block on the lower right corner of the display has been blanked to avoid confusion while this display is in view. After this display has been in view for 15 seconds, it will disappear from view and the algorithm will proceed. Upon completion of the calculations, the results will reappear and the screen will be updated to include our changes to Line 2. This is shown in Figure 10 on page 27. Notice in this case that the Silverleaf and Dawson result has been set to zero, since we exceeded the algorithms limits on Length-to-Beam ratio as delineated in Equation 8. The value we entered is perfectly reasonable; however, we made such a drastic change in one parameter that is linked to another parameter through these range limitations that we exceeded the range for $\frac{L}{B}$ given by Equation 8. This display (Figure 10) now shows the value for the Admiralty Prediction, so we can easily guess that our data was invalid for the Silverleaf and Dawson Prediction. The Admiralty value is valid, we just have no second value to correlate it against as desired. Notice also in Figure 10 on page 27 that the Displacement value of Line 5 has been updated using Equation 11, assuming a constant C_B value.

Proceeding further towards our goal of entering the TANKER B design data, we need to update the next item, Line 3, DESIGN DRAFT. Thus we'll enter a **3** for Line 3, then **74.00** for the TANKER B draft. Again, almost immediately, another HELP WINDOW appears, this time displaying a message that we have entered data that makes the BEAM/DRAFT RATIO OUT OF RANGE. Again, note that the Results Block has been cleared since our design data has changed, making the previous calculations invalid. We see this screen display in Figure 11 on page 28. This display shows that the same



```
***  POWER PREDICTION FAILURE  ***
 TOOL BOX MAY FAIL TO ACCURATELY
 PREDICT POWER REQUIREMENTS FOR
 ADMIRALTY OR S & D METHOD DUE TO
 LENGTH/BEAM RATIO OUT OF RANGE
PLEASE WAIT........................
................PROGRAM WILL RESUME
```

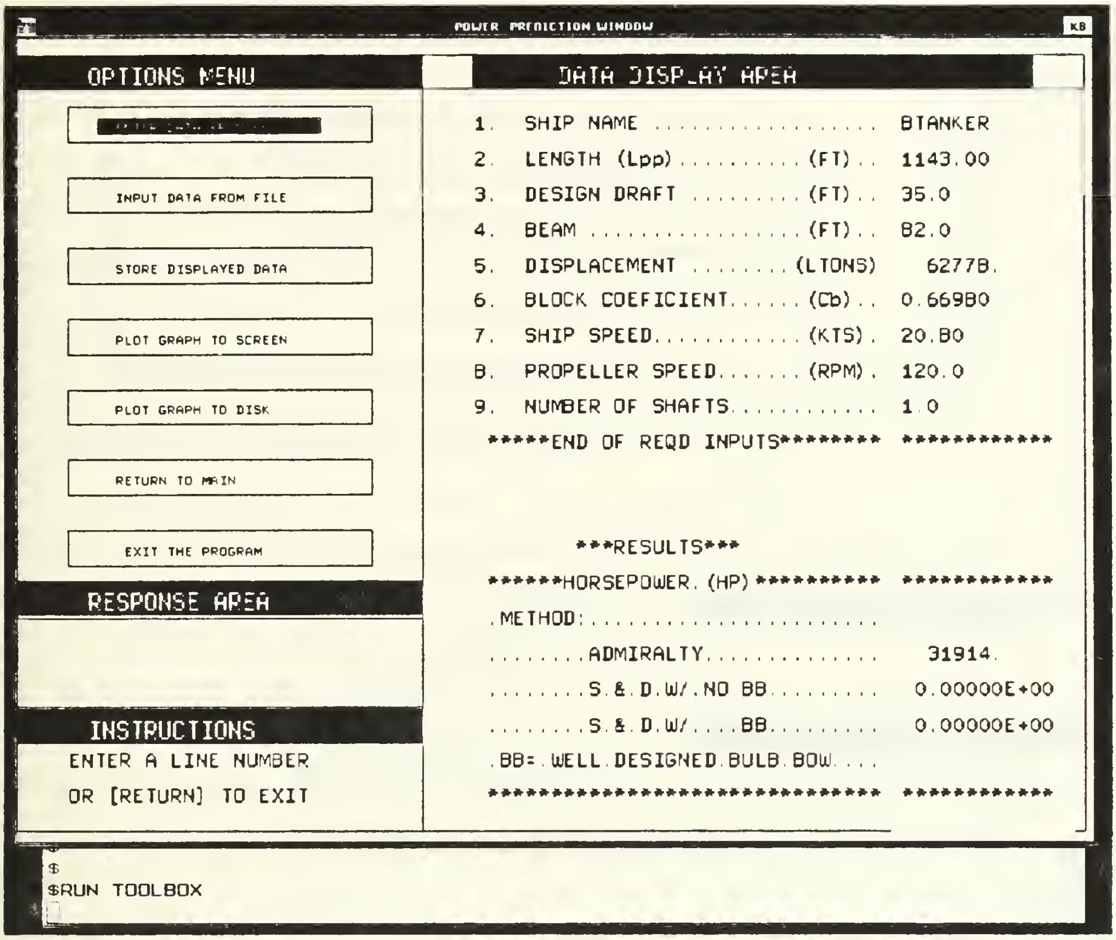
```
ENTER LENGTH (LPP) IN FT
[RETURN] TO EXIT
```

```
9.  NUMBER OF SHAFTS............  1.0
 *****END OF REQD INPUTS********  ************

          ***RESULTS***
 ******HORSEPOWER.(HP)**********
 .METHOD:.......................
 ........ADMIRALTY..............
 ........S.&.D.W/.NO BB.........
 ........S.&.D.W/....BB.........
 .BB=.WELL.DESIGNED.BULB.BOW....
 *******************************
```

```
$
$RUN TOOLBOX
```

Figure 9.   DATA ENTRY Help Screen Display (1)

routine is used for all the error messages generated in the calculations section, making it an efficient subroutine since it can display context sensitive help messages.

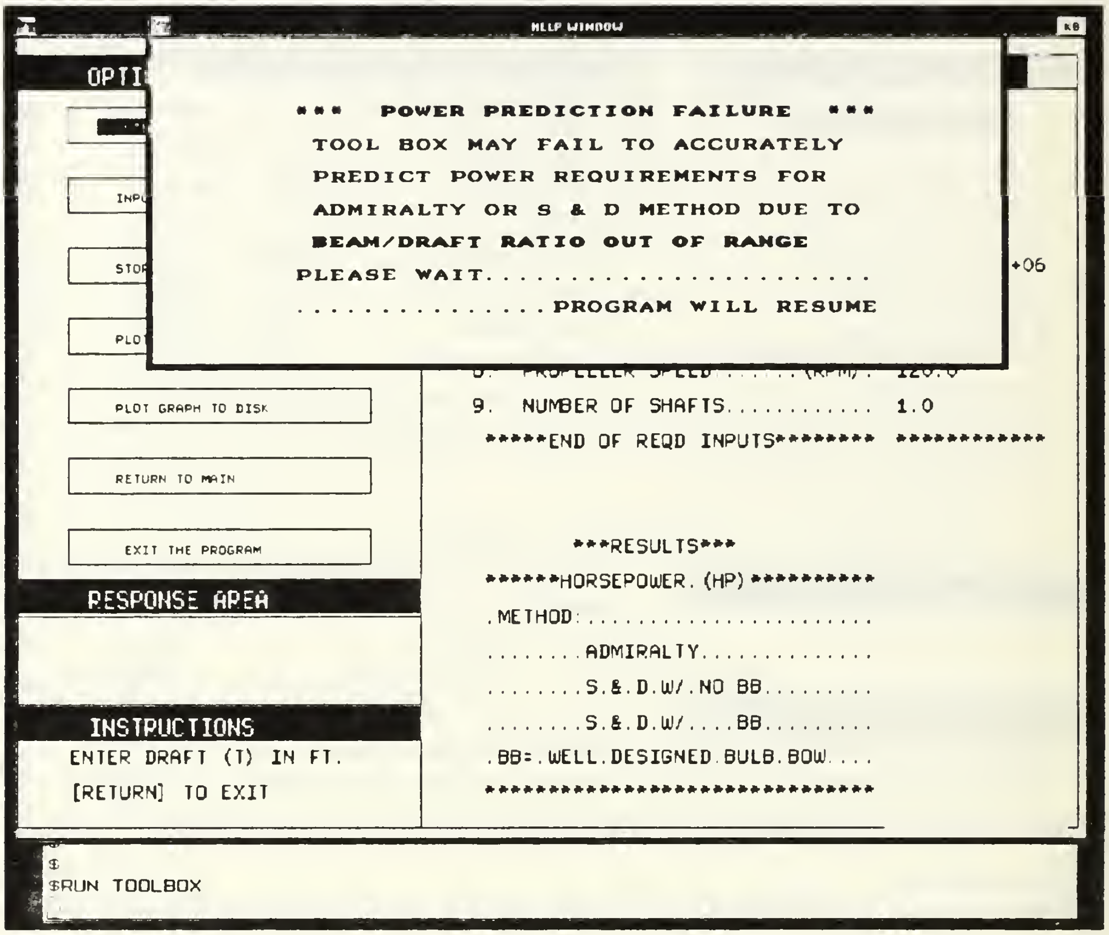


```
ENTER A LINE NUMBER
OR [RETURN] TO EXIT
```

```
1.  SHIP NAME ...................  BTANKER
2.  LENGTH (Lpp)..........(FT)..  1143.00
3.  DESIGN DRAFT .........(FT)..  35.0
4.  BEAM .................(FT)..  82.0
5.  DISPLACEMENT ........(LTONS)    62778.
6.  BLOCK COEFICIENT......(Cb)..  0.66980
7.  SHIP SPEED............(KTS).  20.80
8.  PROPELLER SPEED.......(RPM).  120.0
9.  NUMBER OF SHAFTS............  1.0
 *****END OF REQD INPUTS********  ************


          ***RESULTS***
 ******HORSEPOWER.(HP)**********  ************
 .METHOD:.......................
 ........ADMIRALTY.............     31914.
 ........S.&.D.W/.NO BB.........   0.00000E+00
 ........S.&.D.W/....BB.........   0.00000E+00
 .BB=.WELL.DESIGNED.BULB.BOW....
 *******************************  ************
```

```
$
$RUN TOOLBOX
```

Figure 10.   DATA ENTRY Display after Update



```
***  POWER PREDICTION FAILURE  ***
 TOOL BOX MAY FAIL TO ACCURATELY
 PREDICT POWER REQUIREMENTS FOR
 ADMIRALTY OR S & D METHOD DUE TO
 BEAM/DRAFT RATIO OUT OF RANGE
PLEASE WAIT........................
...............PROGRAM WILL RESUME
```

```
9.  NUMBER OF SHAFTS...........  1.0
 *****END OF REQD INPUTS********  ************

        ***RESULTS***
 ******HORSEPOWER.(HP)**********
 .METHOD:.......................
 ........ADMIRALTY.............
 ........S.&.D.W/.NO BB........
 ........S.&.D.W/....BB........
 .BB=.WELL.DESIGNED.BULB.BOW....
 *******************************
```

```
ENTER DRAFT (T) IN FT.
[RETURN] TO EXIT
```

```
$
$RUN TOOLBOX
```

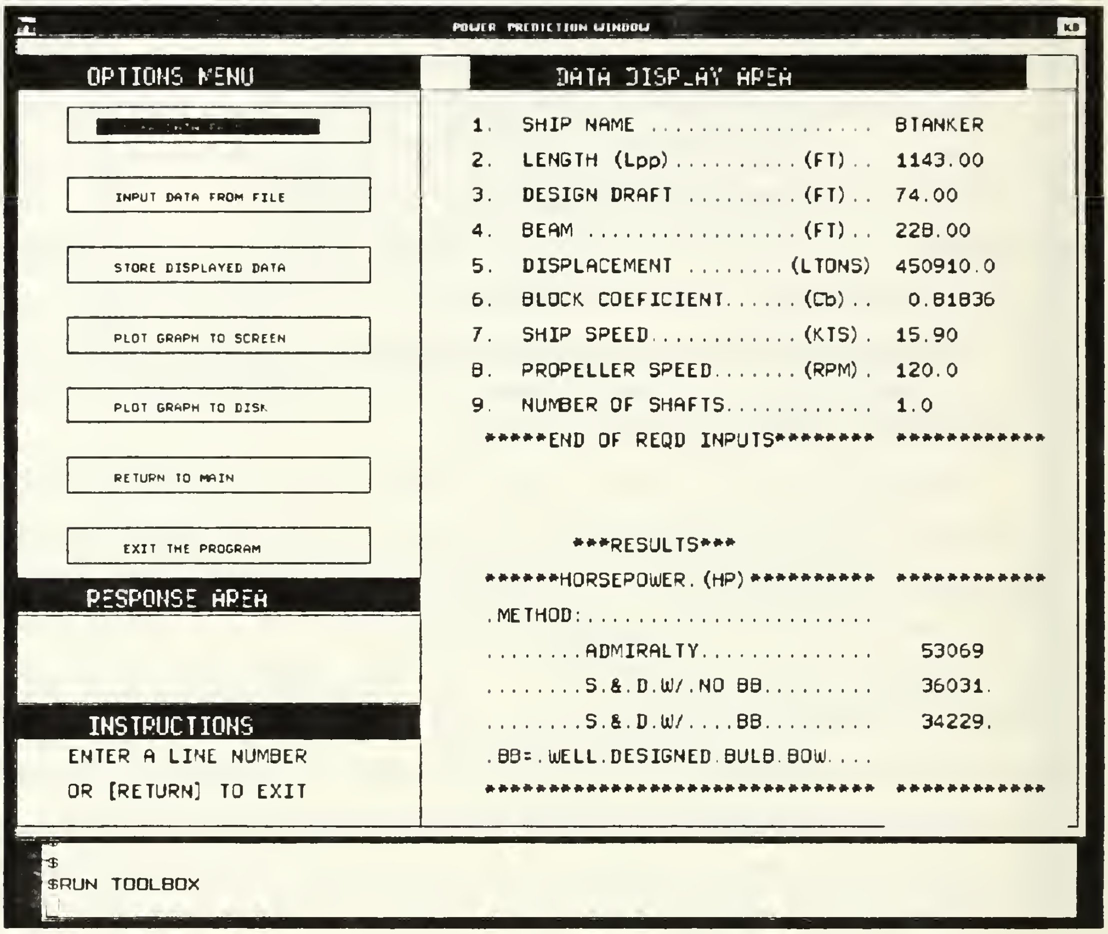
Figure 11. DATA ENTRY Help Screen Display (2)

After another 15 second delay, the HELP WINDOW in Figure 11 disappears, and we can continue. Proceeding along this route further, we must also change the data for Lines 4, 5, and 7 to make all the neccessary changes to get to the TANKER B design parameters. Notice here that we will not change Line 6 for the Block Coefficient C_B, since the algorithm knows that we have uniquely specified Draft, Beam, Length, and Displacement, thus from Equation 11 again, the Block Coefficient is directly specified. At any time if we change either the Beam, Draft, or Length, then the code recalculates Displacement assuming C_B does not change. But if we force a new Displacement, then the code calculates a new C_B based upon Equation 11 and the specified Length, Beam, and Draft. Finally, all the DATA is updated and our DATA DISPLAY AREA appears similar to Figure 12 on page 30, with all the required data in place and new calculations for Predicted Power appear in the Results Block.

At this point we can enter a simple **Return** key to exit from KEYBOARD ENTRY mode, and the screen appears as in Figure 12 on page 30. The mouse pointer is now enabled again, and we can proceed through the menu options. Here it would be prudent to save our new data set for later use or retrieval, so we can select Option 3 on the menu, STORE DISPLAYED DATA, to a file using the filename we gave on Line 1. Clicking on that Option will bring up another HELP WINDOW, reminding us to use different and unique filenames for our data since saving files under the same name will simply overwrite the previous data, rendering it useless in the future. This screen appears in Figure 13 on page 31 and uses another timed display dialog box developed in Chapter 2 Figure 2 on page 8.

After the file save operation is complete, the program will inform the user of the actual filename saved by writing it to the screen display in the INSTRUCTIONS box as shown in Figure 14 on page 32. It is important to note here that the program uses a default .DAT filename extension and truncates filenames to an 8.3 format. That is, eight (8) letters plus the three (3) letters for the .DAT extension. Names longer than this will be truncated to this format.

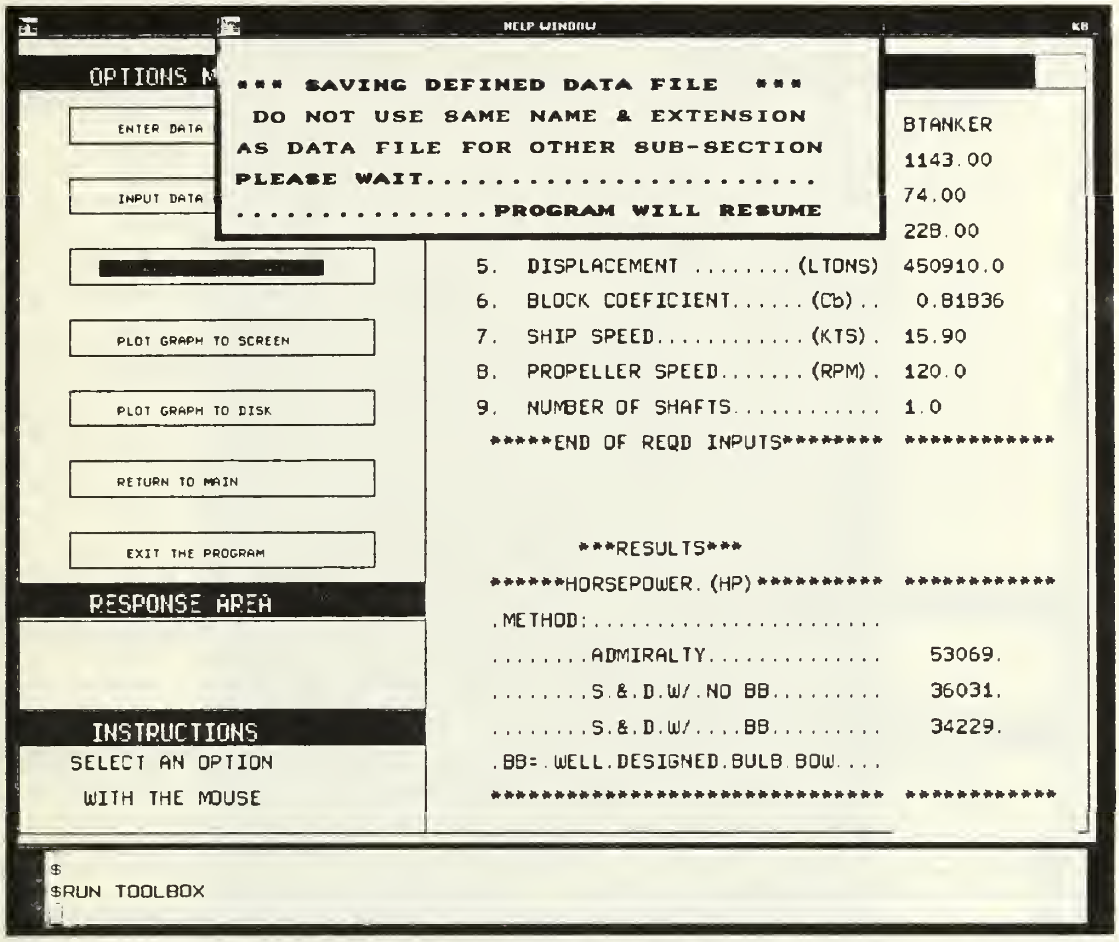
The user can now select the fourth option on the menu, PLOT GRAPH TO SCREEN to have a CA-DISSPLA plot of speed versus power required plotted to the VAX terminal screen. Selection of this option will show a screen display as seen in Figure 15 on page 33 after the graph is calculated and plotted by the system. This process takes a few seconds, so a message appears in the INSTRUCTIONS BOX for the user suggesting that they sit back and relax for a moment. The time delay is not all that lengthy, and the screen display will soon appear.



```
POWER PREDICTION WINDOW

OPTIONS MENU                     DATA DISPLAY AREA

[illegible]                      1.  SHIP NAME ..................    BTANKER
INPUT DATA FROM FILE             2.  LENGTH (Lpp) ..........(FT) ..  1143.00
STORE DISPLAYED DATA             3.  DESIGN DRAFT .........(FT) ..   74.00
PLOT GRAPH TO SCREEN             4.  BEAM ..................(FT) ..  228.00
PLOT GRAPH TO DISK               5.  DISPLACEMENT ........(LTONS)    450910.0
RETURN TO MAIN                   6.  BLOCK COEFICIENT......(Cb) ..    0.81836
EXIT THE PROGRAM                 7.  SHIP SPEED............(KTS) .   15.90
                                 8.  PROPELLER SPEED.......(RPM) .   120.0
                                 9.  NUMBER OF SHAFTS............    1.0
                                  *****END OF REQD INPUTS********  ************

                                          ***RESULTS***
RESPONSE AREA                     ******HORSEPOWER.(HP)*********  ************
                                  .METHOD:.......................
                                  ........ADMIRALTY.............      53069
                                  ........S.&.D.W/.NO BB........      36031.
INSTRUCTIONS                      ........S.&.D.W/....BB........      34229.
ENTER A LINE NUMBER               .BB=.WELL.DESIGNED.BULB.BOW....
OR [RETURN] TO EXIT               ******************************  ************

$
$RUN TOOLBOX
```

Figure 12. DATA ENTRY Screen Display with all Input Changes

Here we encounter the first difficult point of the program, deleting the CA-DISSPLA plot and continuing our work. Although a number of methods have been tried, the easiest method of deleting the plot from the screen is to click the mouse pointer on the calling window just visible at the lower edge of the TOOL BOX windows, in order to bring this window to the foreground. Now a simple carriage return will delete the plot window, and the user can again click the mouse pointer on the TOOL BOX display window (now in the background) to bring it back to the foreground, and continue. This method is archaic, but it is the only *consistent* method of deleting the display this



```
*** SAVING DEFINED DATA FILE ***
 DO NOT USE SAME NAME & EXTENSION
AS DATA FILE FOR OTHER SUB-SECTION
PLEASE WAIT........................
................PROGRAM WILL RESUME
```

```
                                         BTANKER
                                         1143.00
                                         74.00
                                         228.00
5.  DISPLACEMENT ........(LTONS)  450910.0
6.  BLOCK COEFICIENT......(Cb)..   0.81836
7.  SHIP SPEED............(KTS).  15.90
8.  PROPELLER SPEED.......(RPM).  120.0
9.  NUMBER OF SHAFTS............  1.0
 *****END OF REQD INPUTS********  ************

          ***RESULTS***
******HORSEPOWER.(HP)**********  ************
.METHOD:.......................
........ADMIRALTY.............      53069.
........S.&.D.W/.NO BB........      36031.
........S.&.D.W/....BB........      34229.
.BB=.WELL.DESIGNED.BULB BOW....
*******************************  ************
```

```
SELECT AN OPTION
 WITH THE MOUSE
```

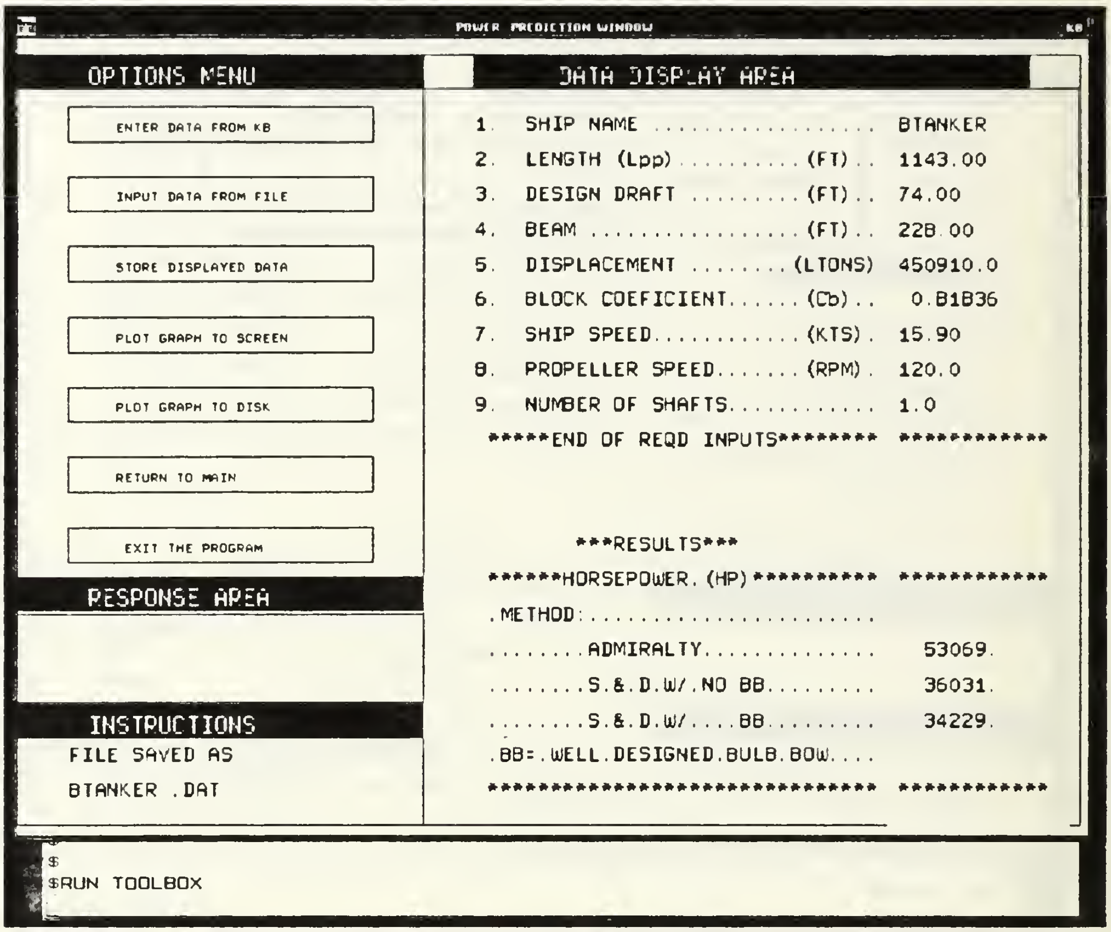
```
$
$RUN TOOLBOX
```

Figure 13. DATA STORE Display using Timed Dialog Boxes

author has found. It does not seem entirely intuitive, but it works. A better method must be found.

Alternatively, the user can select the next screen option, Option 5, PLOT GRAPH TO DISK to plot the same screen display graph to a disk file. If we select this option, again, some time is involved in plotting the CA-DISSPLA graph, so the program informs the user of this in the INSTRUCTIONS area as before.

This graph plotting operation takes approximately 3 minutes to complete with no progress updates for the user to follow, so some patience is required. When the plot



```
FILE SAVED AS
BTANKER .DAT
```

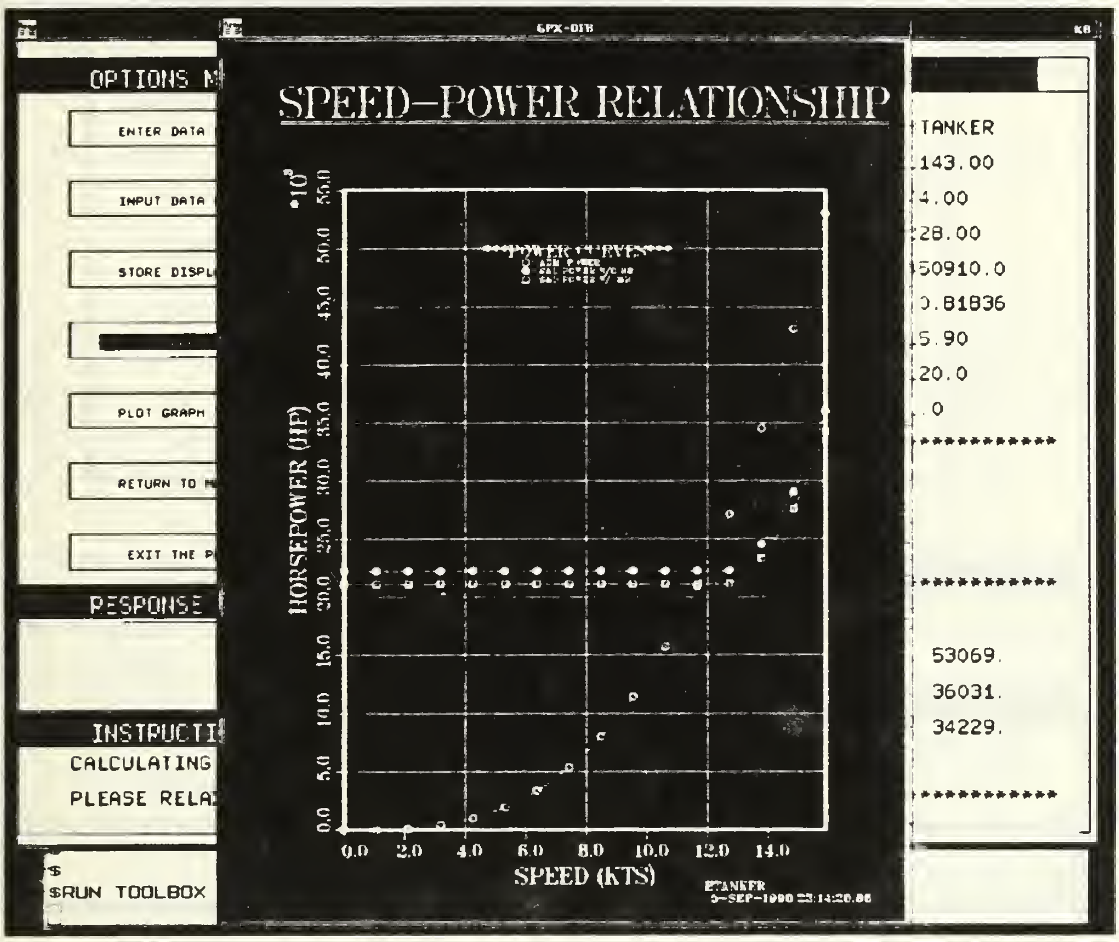
```
1.  SHIP NAME ..................          BTANKER
2.  LENGTH (Lpp) ..........(FT) ..        1143.00
3.  DESIGN DRAFT .........(FT) ..         74.00
4.  BEAM .................(FT) ..         228.00
5.  DISPLACEMENT ........(LTONS)          450910.0
6.  BLOCK COEFICIENT......(Cb) ..          0.81836
7.  SHIP SPEED............(KTS) .         15.90
8.  PROPELLER SPEED.......(RPM) .         120.0
9.  NUMBER OF SHAFTS............          1.0
 *****END OF REQD INPUTS********   ************



         ***RESULTS***
 ******HORSEPOWER.(HP)**********   ************
.METHOD:.......................
........ADMIRALTY.............       53069.
........S.&.D.W/.NO BB........       36031.
........S.&.D.W/....BB........       34229.
.BB=.WELL.DESIGNED.BULB.BOW....
********************************   ************
```

```
$
$RUN TOOLBOX
```

Figure 14. DATA STORE Filename Screen Display

is complete, the program informs the user via another HELP WINDOW dialog box shown in Figure 16 on page 34 and uses a timed dialog box routine similar to those developed previously. This window informs the user of the filename that the graph was saved under, STD00001.DAT by default. It also informs the user how to get a hardcopy printout of the plot using the high resolution of the VAX Laser printer.

Upon completion of this display, the user can either EXIT in one of the modes discussed before or can continue to develop other ship designs using different parameters. One further note is needed though; CA-DISSPLA by default appends new plots

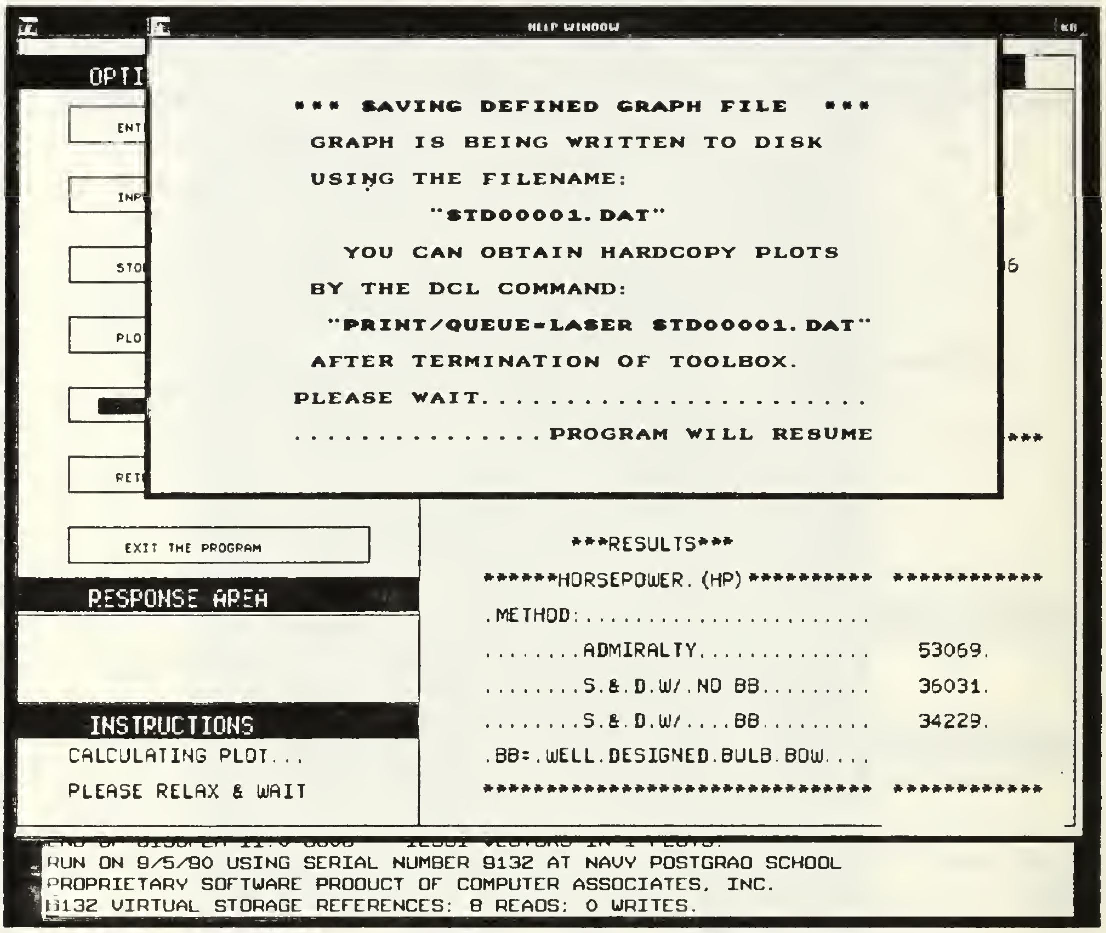


```
CALCULATING
PLEASE RELA
$
$RUN TOOLBOX
```

```
TANKER
143.00
4.00
28.00
50910.0
0.81836
5.90
20.0
.0
***********
***********
53069.
36031.
34229.
***********
```

Figure 15. PLOT GRAPH Screen Display for Screen Plot

onto old existing plot files under the same filename, so any new plot will not have unique VAX/VMS filenames or version numbers. This can be something of a sore spot if you only desire the last plot in a large plot file; however, the file can be edited with the EDT EDITOR to remove old plots. But a bonus is that new plots will not delete old plots since the VAX/VMS system defaults to storing only the two most recent version of a file, so no data will be lost due to that method, it may just be at the end of a very large file.



```
*** SAVING DEFINED GRAPH FILE ***
 GRAPH IS BEING WRITTEN TO DISK
 USING THE FILENAME:
        "STD00001.DAT"
   YOU CAN OBTAIN HARDCOPY PLOTS
 BY THE DCL COMMAND:
  "PRINT/QUEUE=LASER STD00001.DAT"
 AFTER TERMINATION OF TOOLBOX.
PLEASE WAIT........................
................PROGRAM WILL RESUME
```

```
CALCULATING PLOT...
PLEASE RELAX & WAIT
```

```
         ***RESULTS***
******HORSEPOWER.(HP)**********  ************
.METHOD:.......................
........ADMIRALTY.............    53069.
........S.&.D.W/.NO BB........    36031.
........S.&.D.W/....BB........    34229.
.BB=.WELL.DESIGNED.BULB.BOW....
******************************  ************
```

```
RUN ON 9/5/90 USING SERIAL NUMBER 9132 AT NAVY POSTGRAD SCHOOL
PROPRIETARY SOFTWARE PRODUCT OF COMPUTER ASSOCIATES, INC.
9132 VIRTUAL STORAGE REFERENCES; 8 READS; 0 WRITES.
```

Figure 16. PLOT GRAPH Screen Display upon Completion of Disk Plot

Hopefully, we have shown enough of a complete process here to allow a new user enough familiarity to accomplish preliminary design tasks using this Module.

3. Power Prediction Module Output

There are four (4) primary outputs of the Power Prediction Module;

- The screen displays, for quick calculations or information.
- The saved data file, used alone or in conjunction with the Report Module.
- The Speed-Power Relationship plots.
- The Report Module printout of the Power Prediction data, which is much more useful than the second item, but requires the use of another Module.

a. *SCREEN DISPLAYS*

These displays have been dealt with in some depth, and there is little else to say except to use the program, get familiar with it and its capabilities, and **use it as a design tool**. The speed at which new calculations can be performed by simply changing any variable make this type of program excellent for the *What if.....* type of calculations, without the boring repetitiveness of hand performed calculations.

b. *STORED DATA FILE*

Figure 17 is an example of a stored data file typically output from the program. In this case, it corresponds to the previous developmental example, the BTANKER.DAT file:

```
 BTANKER
 1143.0
 74.000
 228.00
0.45091E+06
0.81836
 15.900
 120.00
 1.0000
 53069.
 36031.
 34229.
```

Figure 17. STORED DATA File (Typical); Output from Worked Example

As it is easily seen that, unless the user is familiar with the order of the result variables, very little information is *easily* obtained from Figure 17, although all the information is there in the same order as the screen displays dealt with previously.

c. *SPEED-POWER RELATIONSHIP Plots*

Figure 18 on page 37 is also from the BTANKER development and is the disk plot that resulted from that data. The screen plot is identical to this plot.

As can be seen from this figure, much useful information is displayed on this plot. In the lower righthand corner we have both the filename that the plot was developed under and the time it was developed. Also, we have a full plot of speed in knots versus power in horsepower for the full range of ship speeds from zero (0.0) to the design operating speed (PW_VK). It is easily noted that all three plots trend upward in a cubic relationship to speed through the water. One interesting point however was the tendency of the Silverleaf and Dawson plots to start at **very high power at low speeds**, decrease to a minimum, then trend upwards again similar to the Admiralty plots. This error can be ignored since our equations show a definite relationship of speed at or near the Boundary Speed V_B from Equation 13. At low speeds, the ship speed is far away from the required proximity to the Boundary Speed, so the prediction may not be entirely valid. That plus the speed-to-length ratio at low speed may preclude the ships power from even being predicted at these extremely low speeds. The solution then was to find the minimum Predicted Power level and its corresponding speed in the power matrix and set all Predicted Power points at speeds slower than that speed equal to the minimum Predicted Power. This has the effect of saying to the user that some speed exists at which efficient operation will result in minimum power useage at speeds below that particular speed. At operations above that speed, the penalty is seen in the cubic growth of the power required curve. This is a side benefit of using this Predicted Power curve Relationship that would not have been obvious without the plots.

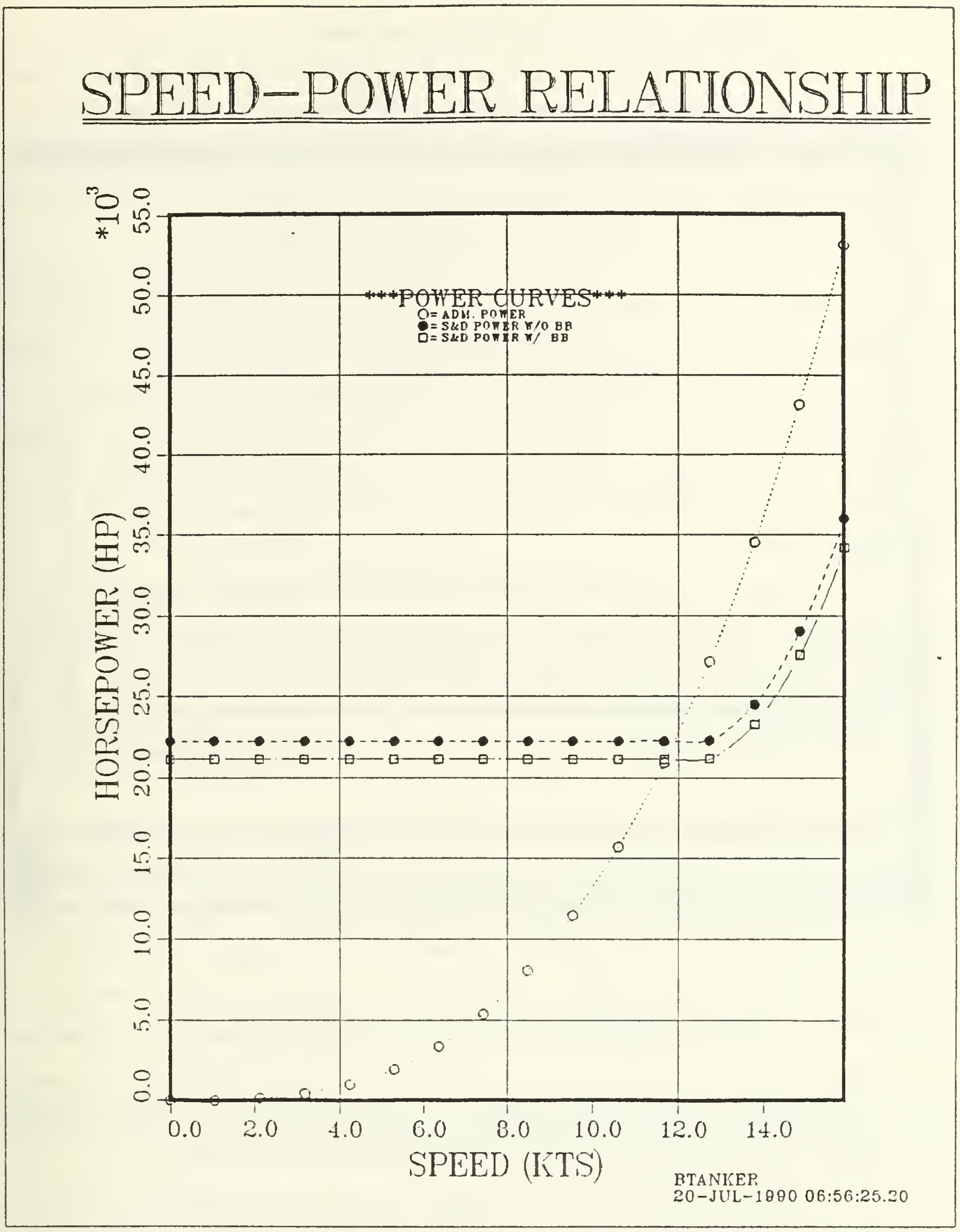


Figure 18. SPEED-POWER RELATIONSHIP Plot

d. REPORT MODULE Predicted Power Report

It was also desired to use the existing Report Module to add an additional report using the data generated by the Power Prediction module. This subroutine source code is shown in Appendix D and is consistent with the display and data entry format of the other Report sections. Again, we will use our BTANKER data file and develop an illustrative example for the report generated by this section.

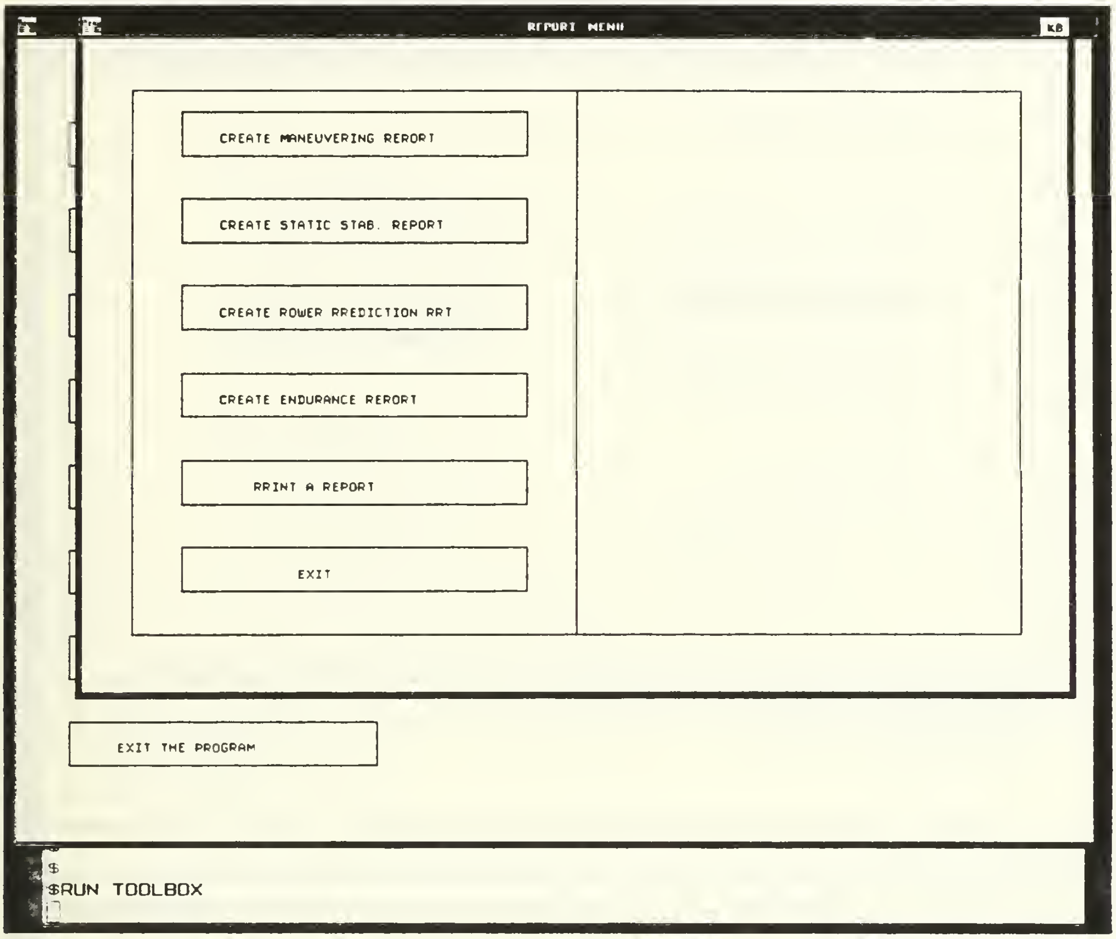
After the user has finished generating the design data using the Power Prediction Module, the data file must be saved to user storage and the filename noted for future use. Then, exiting the Power Prediction module using the RETURN TO MAIN option, the user can now select the MAIN MENUs GENERATE REPORT option using the mouse pointer. This selction will produce a new screen display for the Report Module as shown in Figure 19 on page 39. This report menu is common to all the sections of the Report Module, with the new addition of Option 3, CREATE POWER PREDICTION RPT.

The user can select this option using the mouse device, and the screen display will shift to appear as Figure 20 on page 40, with the users selection highlighted in reverse video and the program query filling the right hand side. Here we will enter BTANKER.DAT in response to the programs request for the data filename and extension to be used in the report. Notice that what we type is also echoed to the screen display as we type it.

Upon entering the data filename as requested, the right hand side of the screen display will clear, and again the user will be queried for another entry, this time for the filename to be given to the produced report, as shown in Figure 21 on page 41 For this we can give any name desired, however a logical choice would be the same filename as the data with a .RPT or .PRN extension to indicate what it contains. We have chosen to use BTANKER.PRN, although any other selection would be adequate.

- Again, the user is cautioned not to use the same name and extension as another pre-existing file (unless that data is to be discarded) since the VAX System will only save two copies of a file with the same name and extension. Thus it is possible for old data files to be deleted by the system if the user is not careful.

Upon entering the output filename and extension as requested in Figure 21, the right hand side will again clear, and the report will be written to disk storage using the filename specified by the user. At this point, the user can enter EXIT and be returned to the MAIN MENU screen, or the Report we have just generated can be sent to an output device. This selection is option 5, PRINT A REPORT, and is

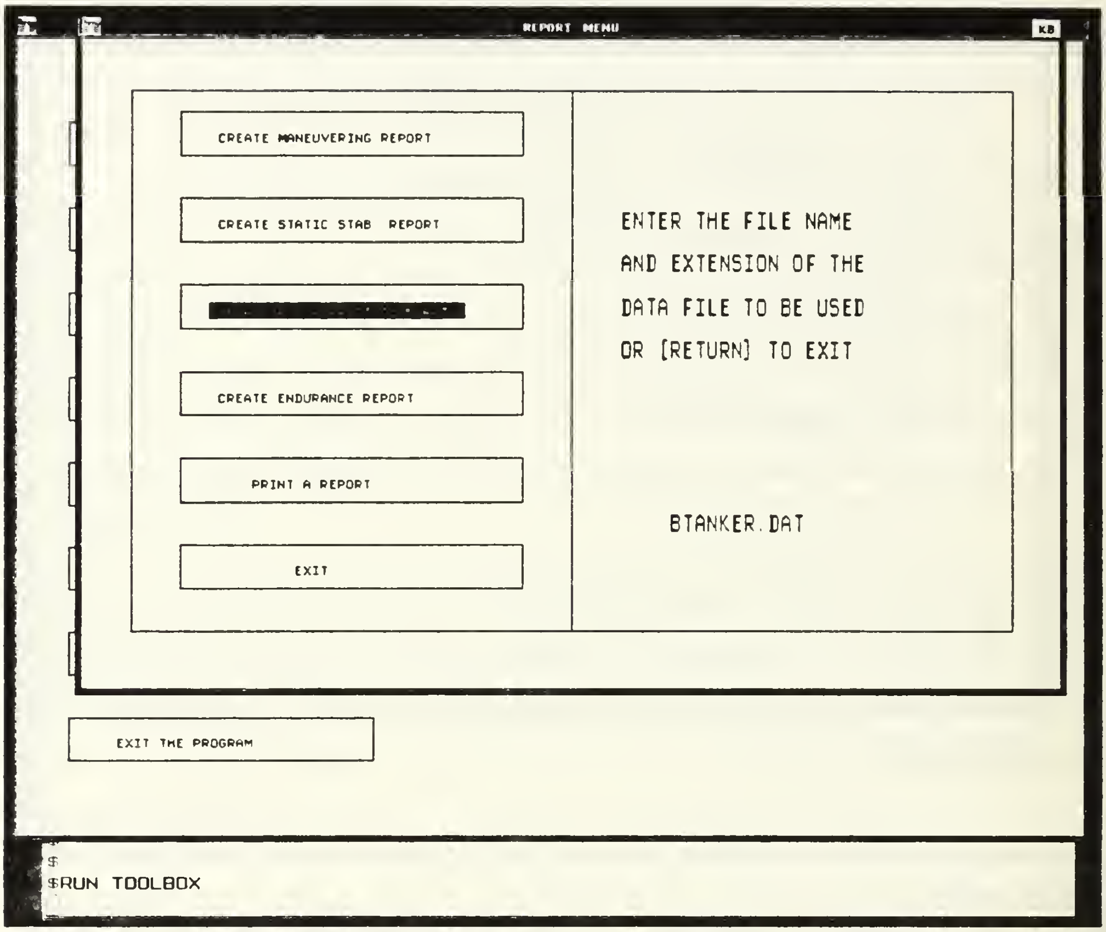

REPORT MENU
KB
CREATE MANEUVERING REPORT
CREATE STATIC STAB. REPORT
CREATE POWER PREDICTION RPT
CREATE ENDURANCE REPORT
PRINT A REPORT
EXIT
EXIT THE PROGRAM

```
$
$RUN TOOLBOX
```

Figure 19. REPORT MODULE Opening Screen Display

shown in Figure 22 on page 42 after the selction has been made. Here the program will again query the user for the filename and extension of the report to be printed. Any existing filename and extension can be used, but it would be prudent to print out the report that we have just finished generating. So here we'll enter BTANKER.PRN as requested, which is shown in Figure 22, to complete the action.

Finally, the program needs to know what output device to be used to print the report on, and it will query the user for this information as shown in Figure 23 on page 43. At this point we must know the VAX system names for output devices, either

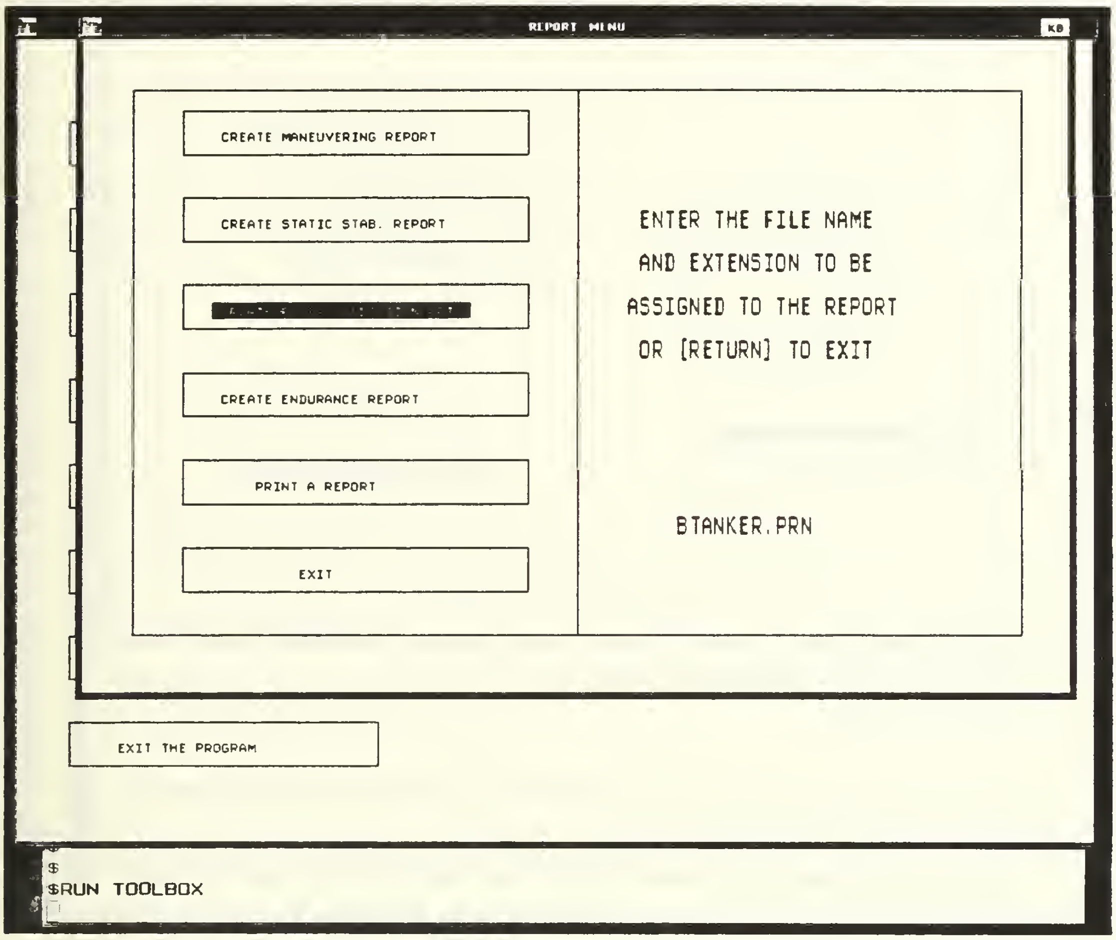


```
ENTER THE FILE NAME
AND EXTENSION OF THE
DATA FILE TO BE USED
OR [RETURN] TO EXIT

    BTANKER.DAT
```

```
$
$RUN TOOLBOX
```

Figure 20. REPORT MODULE Request for Data Entry (Input)

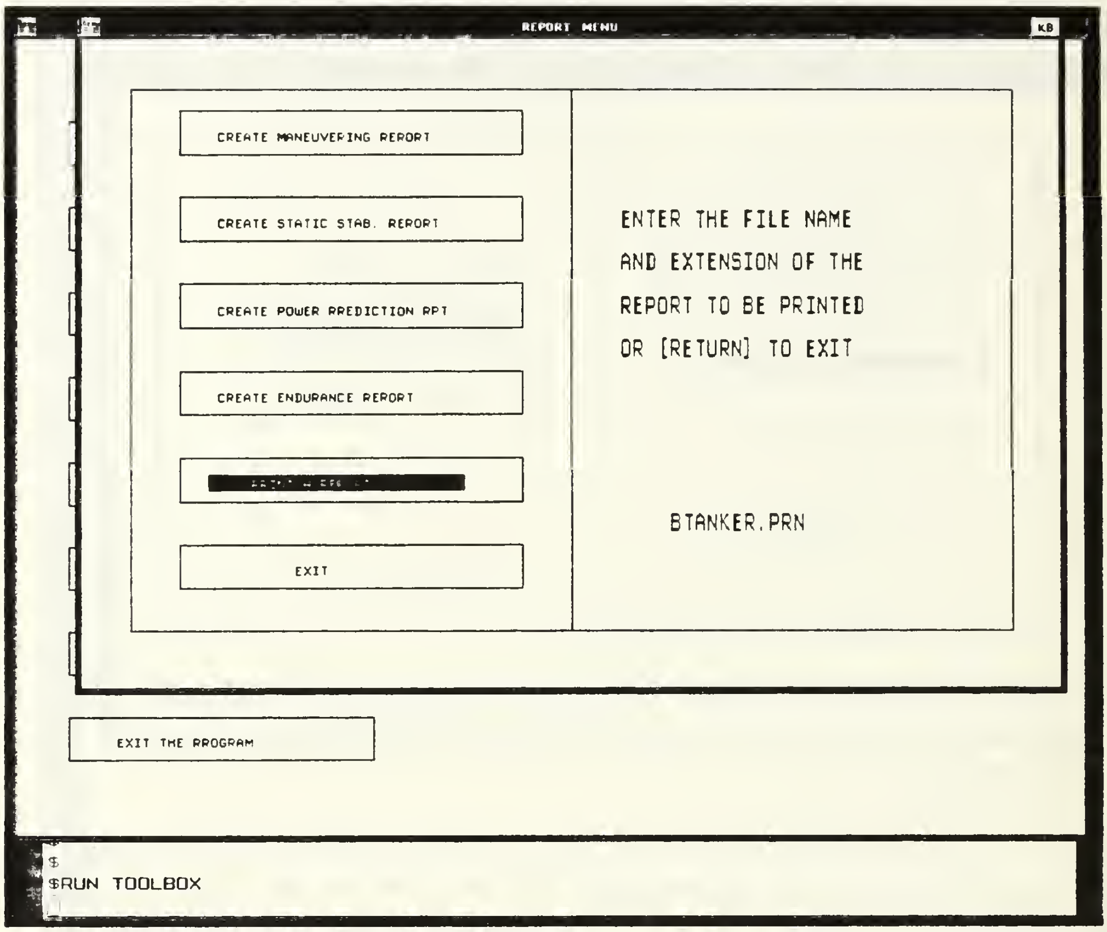
LASER, LA210_i (i= 1,2), or LA75_i (i= 1,2), as valid printing devices. For this example, we'll enter LASER as shown, and the report file will be sent to the named device as requested. At this point we can EXIT, or accomplish another print task, if we have generated multiple data files from the POWER PREDICTION Module. In this manner, multiple reports can be created and printed in one session, so long as unique files contain the input data required.



```
ENTER THE FILE NAME
AND EXTENSION TO BE
ASSIGNED TO THE REPORT
OR [RETURN] TO EXIT

BTANKER.PRN
```

```
$
$RUN TOOLBOX
```

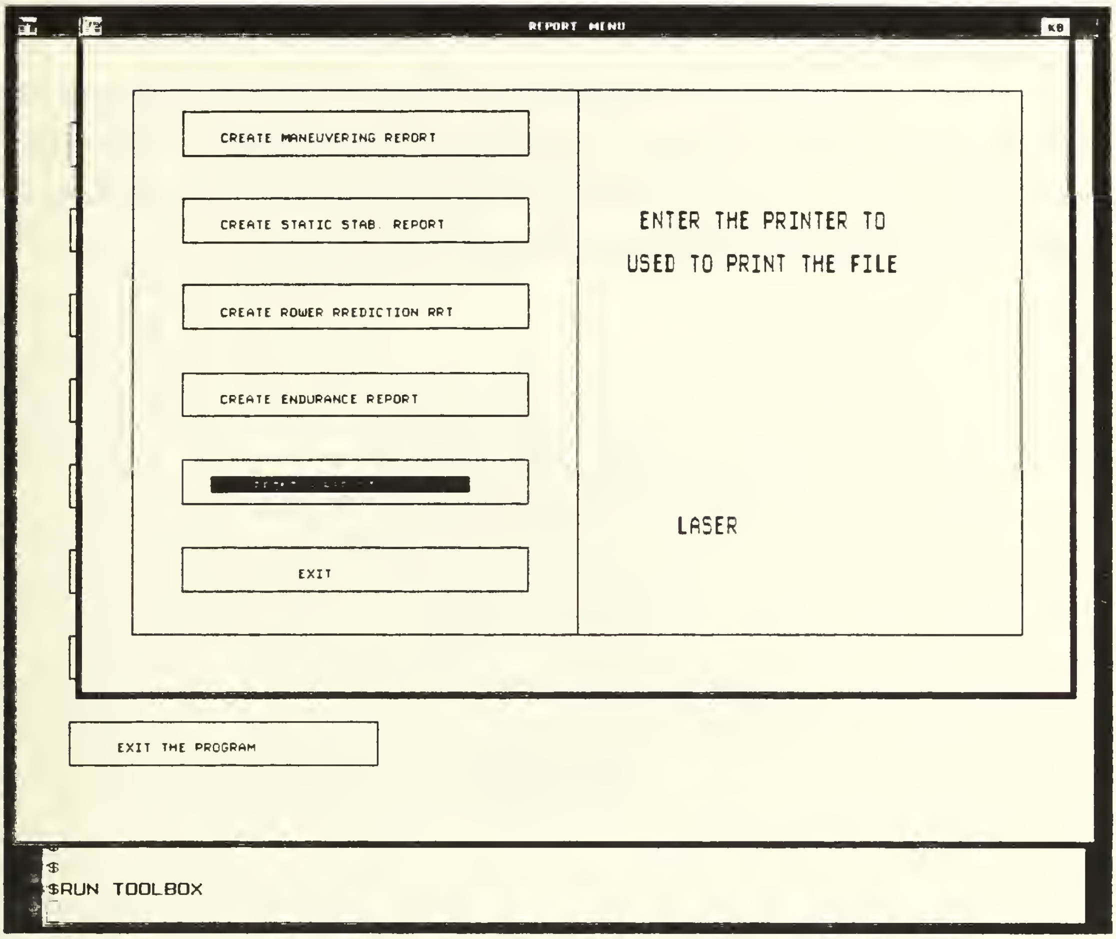
Figure 21. REPORT MODULE Request for Data Entry (Output)


REPORT MENU
KB
CREATE MANEUVERING REPORT
CREATE STATIC STAB. REPORT
CREATE POWER PREDICTION RPT
CREATE ENDURANCE REPORT
[illegible]
EXIT
EXIT THE PROGRAM

```
ENTER THE FILE NAME
AND EXTENSION OF THE
REPORT TO BE PRINTED
OR [RETURN] TO EXIT


    BTANKER.PRN
```

```
$
$RUN TOOLBOX
```

Figure 22.   REPORT MODULE Report Output File Request


REPORT MENU
CREATE MANEUVERING REPORT
CREATE STATIC STAB. REPORT
CREATE POWER PREDICTION RPT
CREATE ENDURANCE REPORT
EXIT
EXIT THE PROGRAM

ENTER THE PRINTER TO
USED TO PRINT THE FILE

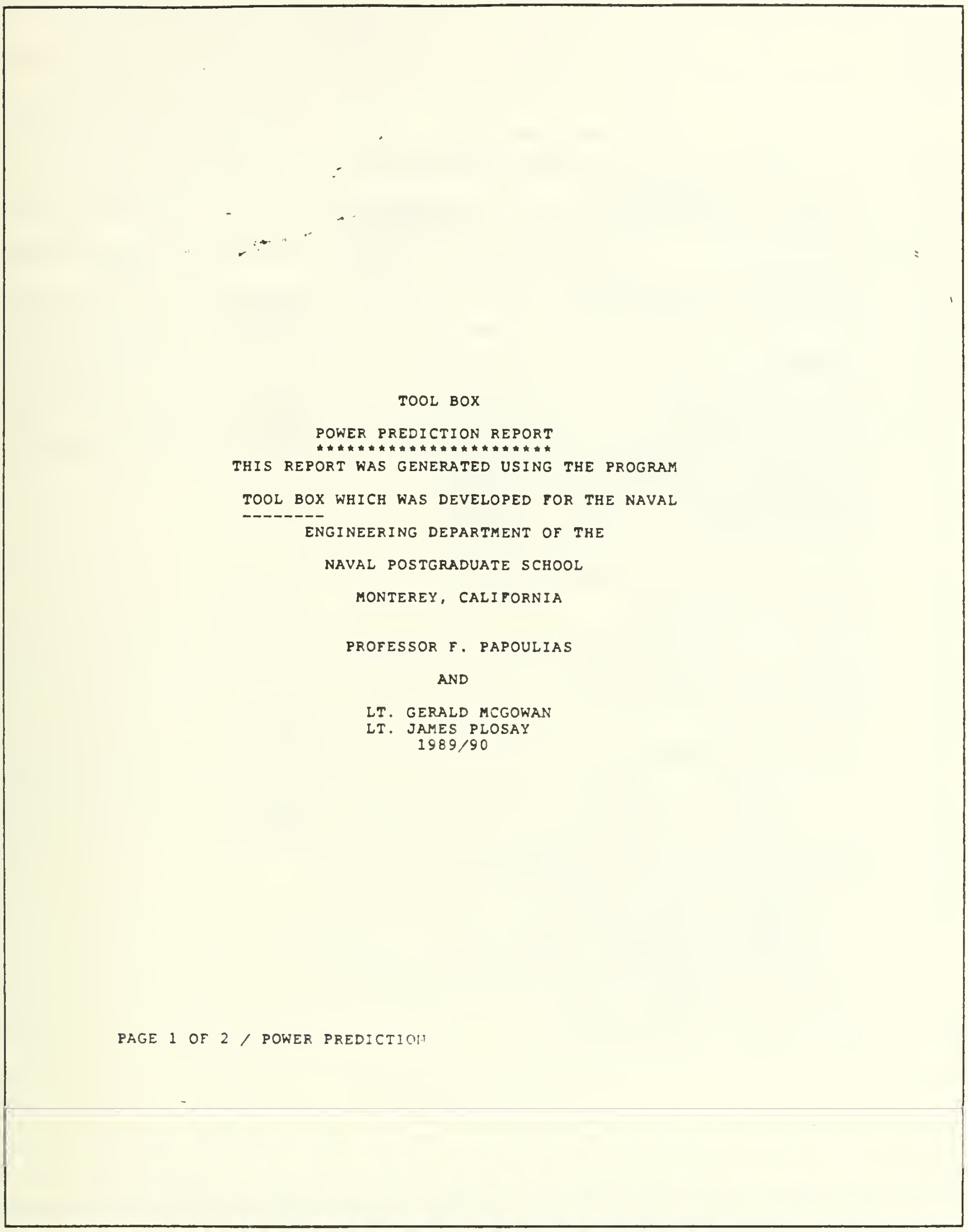
LASER

```
$
$RUN TOOLBOX
```

Figure 23. REPORT MODULE Output Device Request

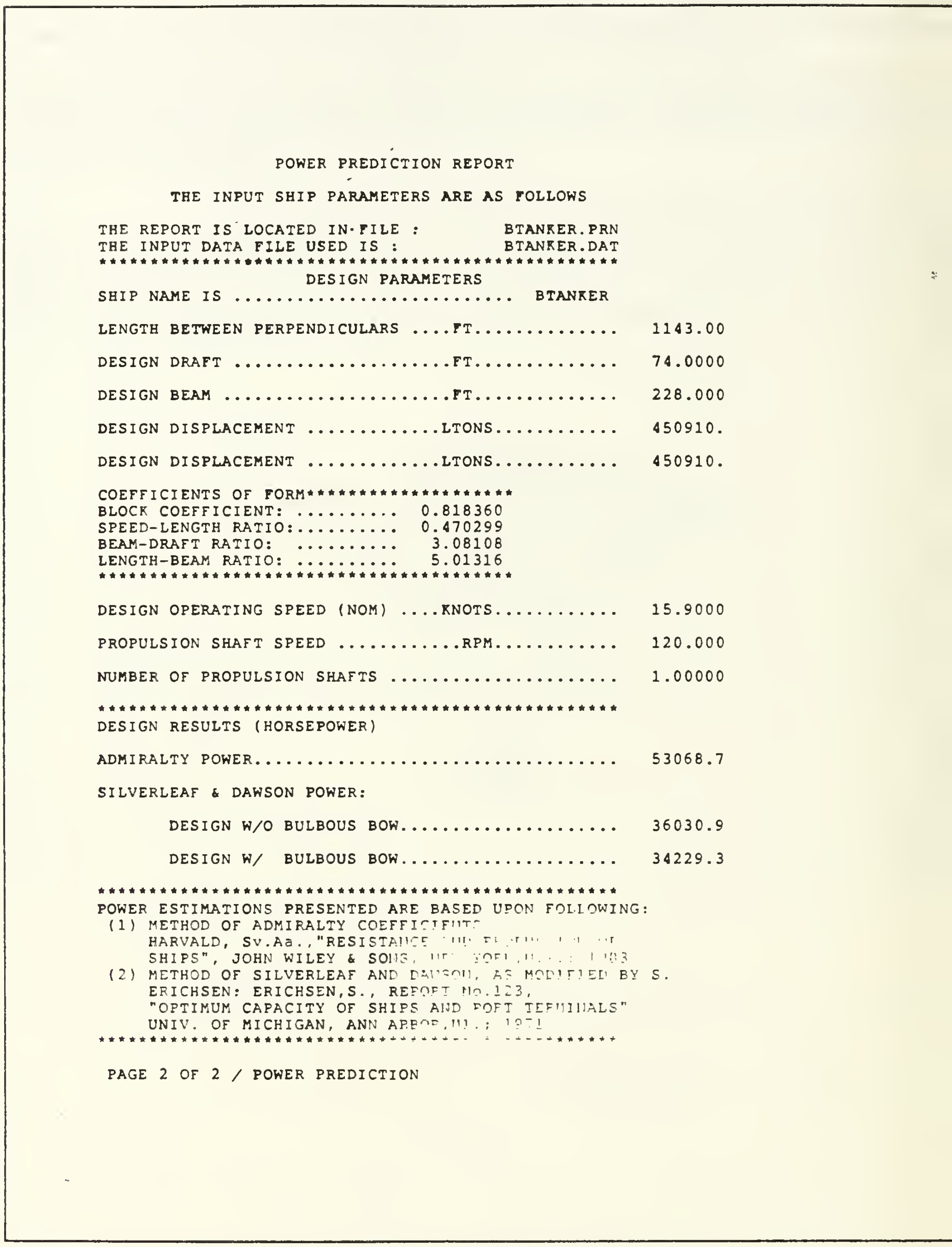
(1) PREDICTED POWER Output Report. Shown in Figure 24 on page 45 and Figure 25 on page 46 is the Report file created by the Report Module, using the data we created from the BTANKER development. As can be seen, some other useful information is given in the Report that was not available in either the stored data file or via the screen displays.

We have added a calculation for some of the Coefficients of Form that are so readily used in preliminary ship design at various points, and are always helpful to have handy. We have made them available here in the Report since they would not comfortably fit on the Screen Displays shown previously.



```
                    TOOL BOX

              POWER PREDICTION REPORT
              ***********************
        THIS REPORT WAS GENERATED USING THE PROGRAM

         TOOL BOX WHICH WAS DEVELOPED FOR THE NAVAL
         --------
              ENGINEERING DEPARTMENT OF THE

                NAVAL POSTGRADUATE SCHOOL

                  MONTEREY, CALIFORNIA

                 PROFESSOR F. PAPOULIAS

                          AND

                   LT. GERALD MCGOWAN
                   LT. JAMES PLOSAY
                        1989/90




PAGE 1 OF 2 / POWER PREDICTION
```

Figure 24.   REPORT MODULE Output Report, Page 1 of 2



```
                    POWER PREDICTION REPORT

           THE INPUT SHIP PARAMETERS ARE AS FOLLOWS

THE REPORT IS LOCATED IN FILE :        BTANKER.PRN
THE INPUT DATA FILE USED IS :          BTANKER.DAT
****************************************************
                     DESIGN PARAMETERS
SHIP NAME IS ........................... BTANKER

LENGTH BETWEEN PERPENDICULARS ....FT.............   1143.00

DESIGN DRAFT .....................FT.............   74.0000

DESIGN BEAM ......................FT.............   228.000

DESIGN DISPLACEMENT .............LTONS...........   450910.

DESIGN DISPLACEMENT .............LTONS...........   450910.

COEFFICIENTS OF FORM********************
BLOCK COEFFICIENT: ..........  0.818360
SPEED-LENGTH RATIO:..........  0.470299
BEAM-DRAFT RATIO:  ..........   3.08108
LENGTH-BEAM RATIO: ..........   5.01316
*****************************************

DESIGN OPERATING SPEED (NOM) ....KNOTS...........   15.9000

PROPULSION SHAFT SPEED ...........RPM............   120.000

NUMBER OF PROPULSION SHAFTS .....................   1.00000

****************************************************
DESIGN RESULTS (HORSEPOWER)

ADMIRALTY POWER..................................   53068.7

SILVERLEAF & DAWSON POWER:

       DESIGN W/O BULBOUS BOW....................   36030.9

       DESIGN W/  BULBOUS BOW....................   34229.3

****************************************************
POWER ESTIMATIONS PRESENTED ARE BASED UPON FOLLOWING:
 (1) METHOD OF ADMIRALTY COEFFICIENTS
     HARVALD, Sv.Aa.,"RESISTANCE AND PROPULSION OF
     SHIPS", JOHN WILEY & SONS, NEW YORK,N.Y.; 1983
 (2) METHOD OF SILVERLEAF AND DAWSON, AS MODIFIED BY S.
     ERICHSEN: ERICHSEN,S., REPORT No.123,
     "OPTIMUM CAPACITY OF SHIPS AND PORT TERMINALS"
     UNIV. OF MICHIGAN, ANN ARBOR,MI.; 1971
****************************************************

 PAGE 2 OF 2 / POWER PREDICTION
```

Figure 25.   REPORT MODULE Output report, Page 2 of 2

E. CONCLUSIONS

Thus we conlude the development and examples for the Predicted Power calculations section. Using the data displayed in Table 1 on page 17 we can conclude that our methods seem reasonably accurate enough for preliminary design work, and certainly when both values are available.

It would be possible to use this Module and the Report with only one of the three Power predictions available by the Module, either just the Admiralty Prediction, or the two Silverleaf and Dawson Predictions. But this gives us no real *feel* for the accuracy of the predicted value by cross checking it against another value from an independently developed method. But using these figures and some additional data from reference material [Ref. 4 : p. 127, Ref. 11 : p. 16], we can begin to get the preliminary data for Machinery plant size, manning requirements, costs, fuel storage requirements, and other vital pieces of the puzzle that we need to continue the iteration and design the vessel.

IV. ENDURANCE CALCULATIONS

A. BACKGROUND

Another useful calculation to perform during Preliminary Ship Design once the expected Propulsive Power requirements are known, is the Voyage Endurance calculation. From this calculation, a number of useful figures arise: expected range of the vessel at design operating speed, the amount of fuel required to be carried for a certain length voyage or speed, the added weight of that fuel to the vessels total displacement, and so on. These figures will give us the total amount of fuel that needs to be carried, and thus the tank volume required to store it, the additional dimensions and size required to contain that storage, the relative efficiency of using one type propulsion plant versus another, and so forth. Additionally, this type of programmed calculation is well suited to the *What if....* type of iterations where design variables are modified to see their overall effect upon the design. Specifically, items such as:

- *What if we speed up our transit from xx to yy knots? How will that effect the amount of fuel we have to carry?*
- *What if we change from geared diesels to steam turbine propulsion equipment? What effect will that change in fuel economy have on the amount of fuel we are required to carry?*

As we can see here, this application is well suited to the applications of modern GUI programming that can reduce or eliminate the need for repetitious hand calculations, thus making it relatively simple to modify a parameter of the preliminary design at any point and check its effect upon the overall design.

B. METHODOLOGY

Basically, we can proceed along one of several directions: (1) we can set the design speed and voyage length, then see how much fuel would be required, or (2) we can set the amount of fuel carried and how fast we want to go, then see how far we can go, or (3) we can set the voyage distance and amount of fuel available, then see how fast we can go to get there. The first option, (1), seems to be the most likely candidate for calculation since it provides the most important factor in this stage of design: the total requirement of fuel in weight and volume that we are required to carry. The other options can be achieved through an iterative process by adjusting the voyage length or speed to get a desired weight of required fuel stowage or fixed volumetric limitations.

From several sources [Ref. 6 : pp. 173- 174, Ref. 12 : pp. 253- 282], there are algorithms for calculating endurance based upon all the relevant ship data that should be available up to this point in the preliminary design. The essential core parts of the algorithms are the same however, so we will only examine the most thorough and involved method researched.

1. Method Of U.S. Navy, Design Data Sheets (DDS 9400-1)

This method, [Ref. 13], utilizes a tabular form provided by the Navy to its surface combatants to calculate all the essential parameters that we require here for endurance calculations. This method and its form , which is shown in Figure 26 on page 50, is dated from 01 November 1963, but is still valid in all its calculations.

In order to complete the calculation, the following sets of data must be known:

- Endurance Variables:

(1) Endurance Required (miles)

(2) Endurance Speed (knots)

(3) Endurance Fuel Load (tons)

- Design Parameters:

(4) Full Load Displacement (ltons)

(5) Rated Full Power (shp)

(6) Design Endurance Power (shp; at 1,2, and 3)

(7) Cruising Electric Load (kW)

(8) Tailpipe Allowance Factor (%)

- Engineering Parameters:

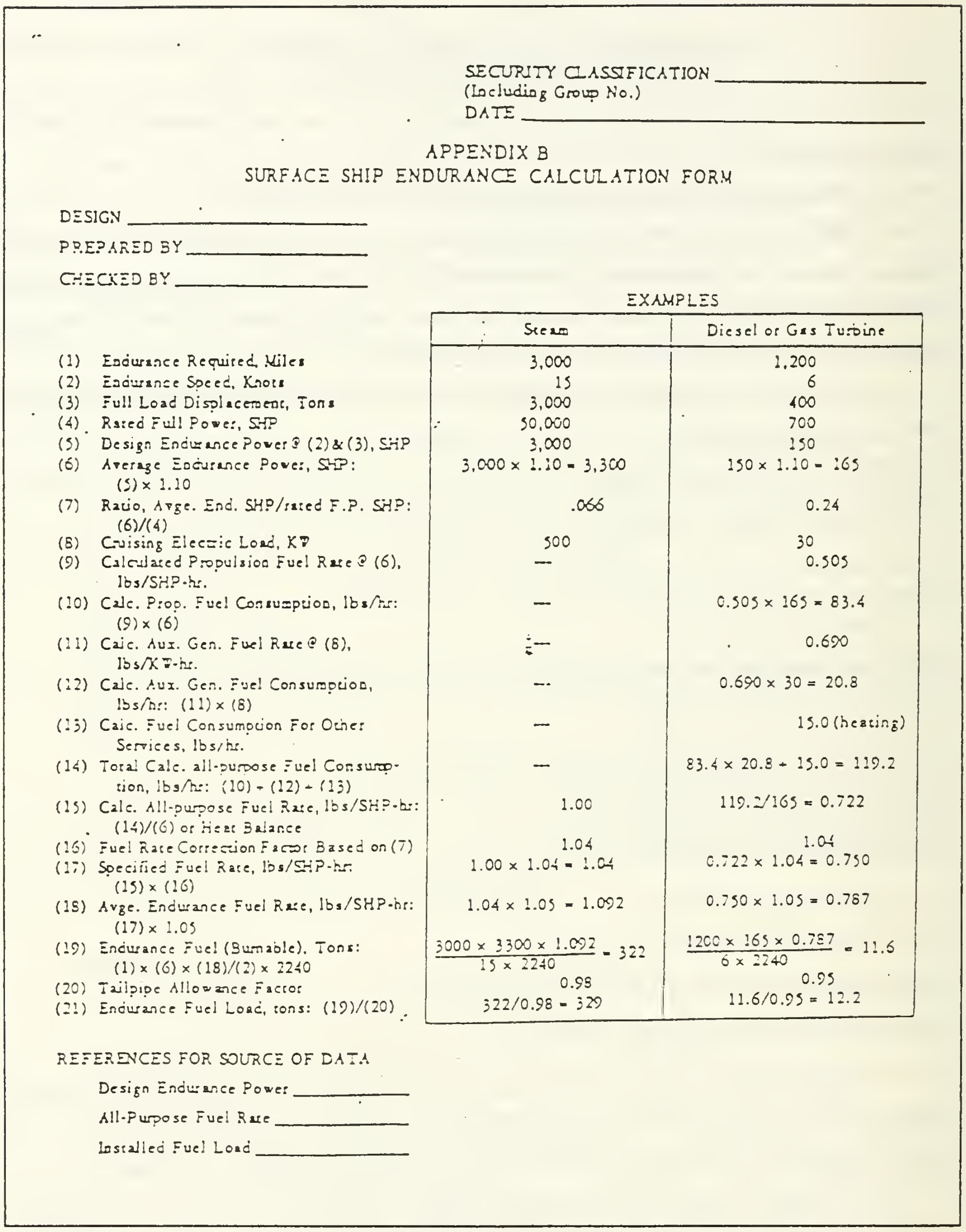
(9) Propulsion Fuel Rate (lbs/shp- hr)

(10) Auxiliary Generator Fuel Rate (lbs/kw- hr)

(11) Other Fuel Consumption (lbs/hr)

Then, from these sets of numbers, we can proceed to calculate the estimated endurance in terms of the amount of fuel we are required to carry, in the following manner (using (*n*) as referring to the data parameter required from the above list):

1. First we need the Average Endurance Power, estimated as (6) × 1.10. We will call this result (A). Then we can find the ratio of (A) to the rated Full Power, (*A*) / (5) and call this result (B).
2. At (A), we also need to know the Propulsion Fuel Rate in (lbs/shp- hr). We will call this item (C).



SECURITY CLASSIFICATION ______________
(Including Group No.)
DATE ______________

APPENDIX B
SURFACE SHIP ENDURANCE CALCULATION FORM

DESIGN ______________
PREPARED BY ______________
CHECKED BY ______________

EXAMPLES

	Steam	Diesel or Gas Turbine
(1) Endurance Required, Miles	3,000	1,200
(2) Endurance Speed, Knots	15	6
(3) Full Load Displacement, Tons	3,000	400
(4) Rated Full Power, SHP	50,000	700
(5) Design Endurance Power @ (2) & (3), SHP	3,000	150
(6) Average Endurance Power, SHP: (5) × 1.10	3,000 × 1.10 = 3,300	150 × 1.10 = 165
(7) Ratio, Avge. End. SHP/rated F.P. SHP: (6)/(4)	.066	0.24
(8) Cruising Electric Load, KW	500	30
(9) Calculated Propulsion Fuel Rate @ (6), lbs/SHP-hr.	—	0.505
(10) Calc. Prop. Fuel Consumption, lbs/hr: (9) × (6)	—	0.505 × 165 = 83.4
(11) Calc. Aux. Gen. Fuel Rate @ (8), lbs/KW-hr.	—	0.690
(12) Calc. Aux. Gen. Fuel Consumption, lbs/hr: (11) × (8)	—	0.690 × 30 = 20.8
(13) Calc. Fuel Consumption For Other Services, lbs/hr.	—	15.0 (heating)
(14) Total Calc. all-purpose Fuel Consumption, lbs/hr: (10) + (12) + (13)	—	83.4 × 20.8 + 15.0 = 119.2
(15) Calc. All-purpose Fuel Rate, lbs/SHP-hr: (14)/(6) or Heat Balance	1.00	119.2/165 = 0.722
(16) Fuel Rate Correction Factor Based on (7)	1.04	1.04
(17) Specified Fuel Rate, lbs/SHP-hr: (15) × (16)	1.00 × 1.04 = 1.04	0.722 × 1.04 = 0.750
(18) Avge. Endurance Fuel Rate, lbs/SHP-hr: (17) × 1.05	1.04 × 1.05 = 1.092	0.750 × 1.05 = 0.787
(19) Endurance Fuel (Burnable), Tons: (1) × (6) × (18)/(2) × 2240	$\frac{3000 \times 3300 \times 1.092}{15 \times 2240} = 322$	$\frac{1200 \times 165 \times 0.787}{6 \times 2240} = 11.6$
(20) Tailpipe Allowance Factor	0.98	0.95
(21) Endurance Fuel Load, tons: (19)/(20)	322/0.98 = 329	11.6/0.95 = 12.2

REFERENCES FOR SOURCE OF DATA
Design Endurance Power ______________
All-Purpose Fuel Rate ______________
Installed Fuel Load ______________

Figure 26. U.S.N. Design Data Sheet (typical), for Endurance Calculations

3. Using (C) we can estimate the Propulsion Fuel Consumption, $(C) \times (A)$, in (lbs/hr), and label this result (D).

4. Also, we will need to know the Auxiliary (Electric) Generator Fuel Rate in (lbs/kW-hr), and the 'Other' Fuel Rate in (lbs/hr). We will label these as (E) and (F). The information required to make these calculations should become available after the preliminary Propulsion requirements have been estimated, and a plant type is selected from the available literature.

5. Using (E), we can estimate the Generator Fuel Consumption as $(E) \times (7)$, which we will label as (G), and then $(D) + (G) + (F)$ which we will call the Total Fuel Consumption, (H).

6. Now we need a Fuel Consumption Correction, based upon the magnitude of (B). This parameter will be labeled (I).

7. Using (I), we can get the Specified Fuel Rate $(I) \times (A)$, and label this as (J).

8. Finally, we can obtain the Average Endurance Fuel Rate, $(J) \times 1.05$, labeled (K).

9. This figure will give rise to the Theoretical Endurance Fuel required, or

$$\frac{(1) \times (A) \times (K)}{(2) \times 2240.0} \tag{23}$$

   which we will label as (L).

10. Before we can finish the calculation however, we must account for (8), the Tailpipe Allowance that indicates how much fuel is carried that is below the suction point of the fuel oil transfer system and is thus unaccessible for use. This varies with tank geometry, being 0.95 for broad shallow tanks, and 0.98 for tall narrow tanks.

11. Using (8), we can calculate the Endurance Fuel Load as $(L)\ /\ (8)$ and save this as the desired result.

From the development above, we can easily have the user input the required values (1) through (11), and have a simple routine to calculate required fuel load in tons, as well as total trip time at the input values of speed and distance, volumetric storage requirements for the fuel using standard conversion factors, and total percent of full load devoted to fuel storage. These last parameters are useful for checking the validity of the design and whether or not it is *reasonable* in terms of standard design practices.

C. COMPARISON OF THE METHOD

Using the same Design Data as in Chapter 3 from Reference 4 that was used to generate Table 1 on page 17, we can compare our estimated Endurance calculations against the actual Endurance Fuel Oil load values of some modern commercial vessels at their design endurance load and speed.

Table 2. COMPARISON OF DDS9400-1 ENDURANCE ESTIMATION

Ship Design	Actual Endurance (Tons Fuel Oil)	DDS9400-1 Endurance (Tons Fuel Oil)
Large General Cargo	3596.00	3200.8
Container 'A'	3380.00	2435.1
Container 'B'	6943.00	4032.4
Roll On-Roll Off	3465.00	4617.7
LASH Barge Carrier	4928.00	4088.7
SEABEE Barge Carrier	5997.00	5169.9
Tanker 'A'	3624.00	2662.9
Tanker 'B'	17857.00	8427.1
LNG Tanker	4400.00	6040.7
Bulk Carrier	1868.00	2115.0
Ore/Bulk/Oil Carrier	4845.00	4024.2

The values used here in comparison are for a general voyage of 10,000.0 nautical miles at rated power, using fuel economy values from the literature [Ref. 11 : p.16], and an estimated cruising electric load given by $KW = 0.015shp + 1.6N + 9\sqrt{N} + 80.0$, where shp is cruising shaft horsepower and N is the number of crew and passengers onboard, with the resultant electic load in Kilowatts (kW).[1] All other data is obtained from Reference 4, pages 137- 171, and the calculations were performed using TOOL BOX screen displays.[2]

1 Rule-of-Thumb equation for Preliminary Electic Load calculations obtained from Naval Sea Systems Command, Washington D.C.

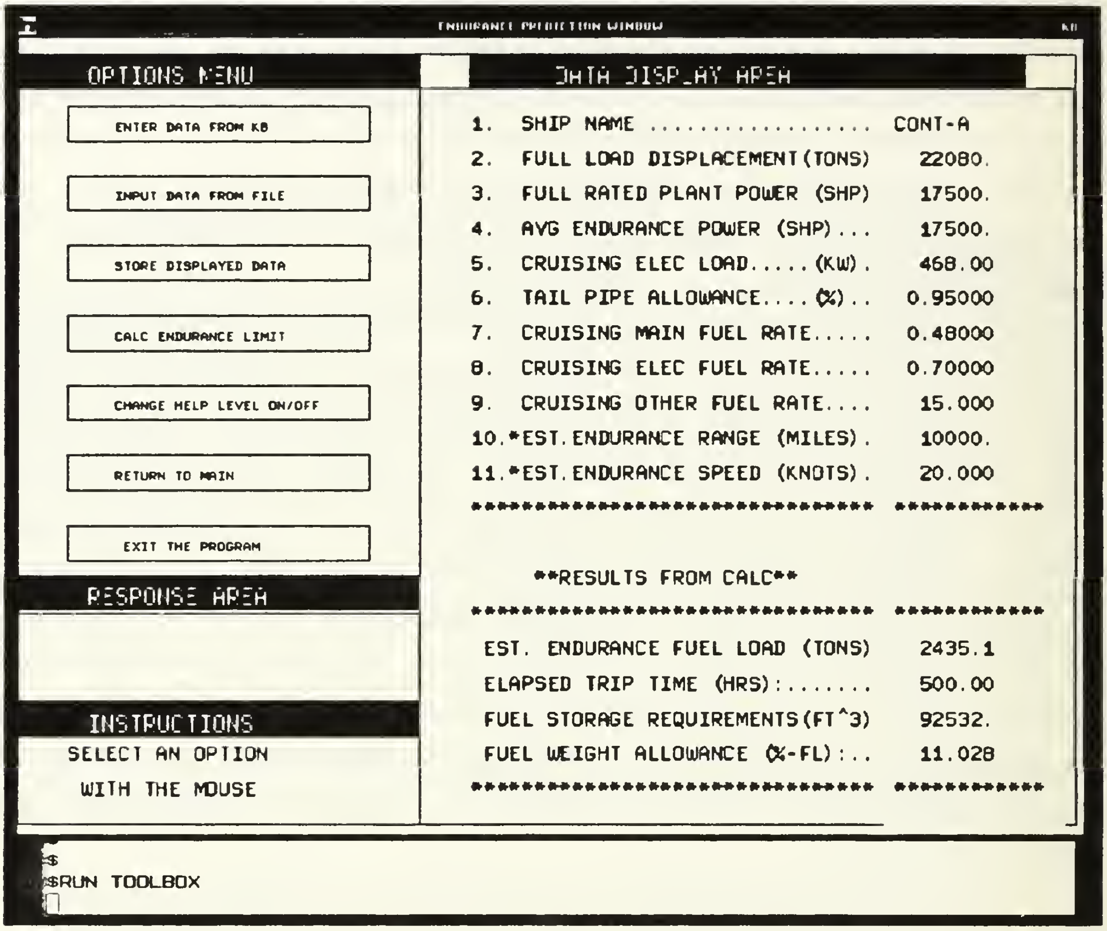
2 Other standards used for unknown values include: Tailpipe Allowance of 0.95, Auxiliary generator fuel oil rate of 0.700 lb/kW-hr, and 'other' fuel oil rate of 15.0 lb/hr.

D. THE TOOL BOX ENDURANCE ESTIMATION MODULE

1. Operation of the Endurance Estimation Module

The major operations of the ENDURANCE Module function the same as all the other modules. A quick examination of Figure 27 on page 54, will show that the first three mouse selection functions: **ENTER DATA FROM KB, READ DATA FROM FILE**, and **STORE DISPLAYED DATA** are all consistent functions from the other modules. The fourth selection, **CALCULATE ENDURANCE LIMIT**, performs the actual calculation of fuel load for the data as it is shown on the program screen. Notice from the screen display shown in Figure 27 that the last two required data points, lines 10. and 11. are highlighted by an asterisk. These two data points have the calculational routine tied in to them so that as this data is modified, the calculations are performed and the screen updated immediately. The reasoning for this is that these are the significant variables that will be modified most often. Also, these values will most likely not be entered until all the other variables are set to non- zero values, saving the time involved in tracking through the HELP WINDOW error displays during the program initialization. Otherwise, if a higher line number is modified, the user will need to enter the new value, hit RETURN to exit keyboard data entry mode, and click the mouse pointer on the fourth selection, CALCULATE ENDURANCE LIMIT. The functioning methodology, data entry process, and HELP windows function similarily to the POWER ESTIMATION Module that we have just covered in some depth, so they will not be repeated here.

The other new feature that has been added to this module is the fifth mouse selection shown in Figure 27, that is labeled **CHANGE HELP LEVEL ON/OFF**. This new feature has been added because of the limitations imposed on the programmer due to having only a small area available for information presentation to the user in the INSTRUCTIONS area at the lower left corner. This small area limits the available number of lines of text information that can be presented to the user to provide hints or examples of input data that is required. We can easily institute context sensitive HELP WINDOWS using our previously developed dialog box routines, but wanted the ability to provide the user the ablity to disable HELP information and prompting for each value if they desire once they become proficient. This has been accomplished using a simple subroutine named SET_HELP_LEVEL that is called from the ENDURANCE main menu, and resets a pointer that is an input value to the SET_HELP windowing routine used to create and display the help information. The HELP level setting is ON by default each time the ENDURANCE Module is called from the MAIN MENU, so



```
SELECT AN OPTION
 WITH THE MOUSE
```

```
1.  SHIP NAME ..................  CONT-A
2.  FULL LOAD DISPLACEMENT(TONS)    22080.
3.  FULL RATED PLANT POWER (SHP)    17500.
4.  AVG ENDURANCE POWER (SHP)...    17500.
5.  CRUISING ELEC LOAD.....(KW).    468.00
6.  TAIL PIPE ALLOWANCE....(%)..  0.95000
7.  CRUISING MAIN FUEL RATE.....  0.48000
8.  CRUISING ELEC FUEL RATE.....  0.70000
9.  CRUISING OTHER FUEL RATE....    15.000
10.*EST.ENDURANCE RANGE (MILES).    10000.
11.*EST.ENDURANCE SPEED (KNOTS).    20.000
******************************** ************

     **RESULTS FROM CALC**
******************************** ************
 EST. ENDURANCE FUEL LOAD (TONS)    2435.1
 ELAPSED TRIP TIME (HRS):.......    500.00
 FUEL STORAGE REQUIREMENTS(FT^3)    92532.
 FUEL WEIGHT ALLOWANCE (%-FL):..    11.028
******************************** ************
```

```
$
$RUN TOOLBOX
```

Figure 27.   ENDURANCE Module Main Data Display Screen

an advanced user can simply reset the help level indicator upon entering this module if they so desire. Figure 28 on page 56 shows this process and the display window that informs the user of this selection. Alternatively, the user can set the help level indicator back to ON at any time during the progress of the program, without any loss of continuity. For this case, the screen display shown would be exactly similar to Figure 28 except that the instructions would be reversed.

- NOTE: Setting the HELP_LEVEL indicator to OFF does not disable the presentation of HELP WINDOWS containing error messages or information messages.

This feature solely pertains to the parameter definition displays presented during the process of entering data from the keyboard in this module.

Thus the user has the ability to determine some of the functionality of their work environment by changing the way in which information is presented. Once a design begins to develop, there will be less need for the parameter definition windows, so this option will allow the user the flexibility of definitions only when they are desired. The user could also get a program parameter definition by entering that data line number while HELP_LEVEL is ON, without changing the actual value entered.

E. ENDURANCE ESTIMATION MODULE OUTPUT

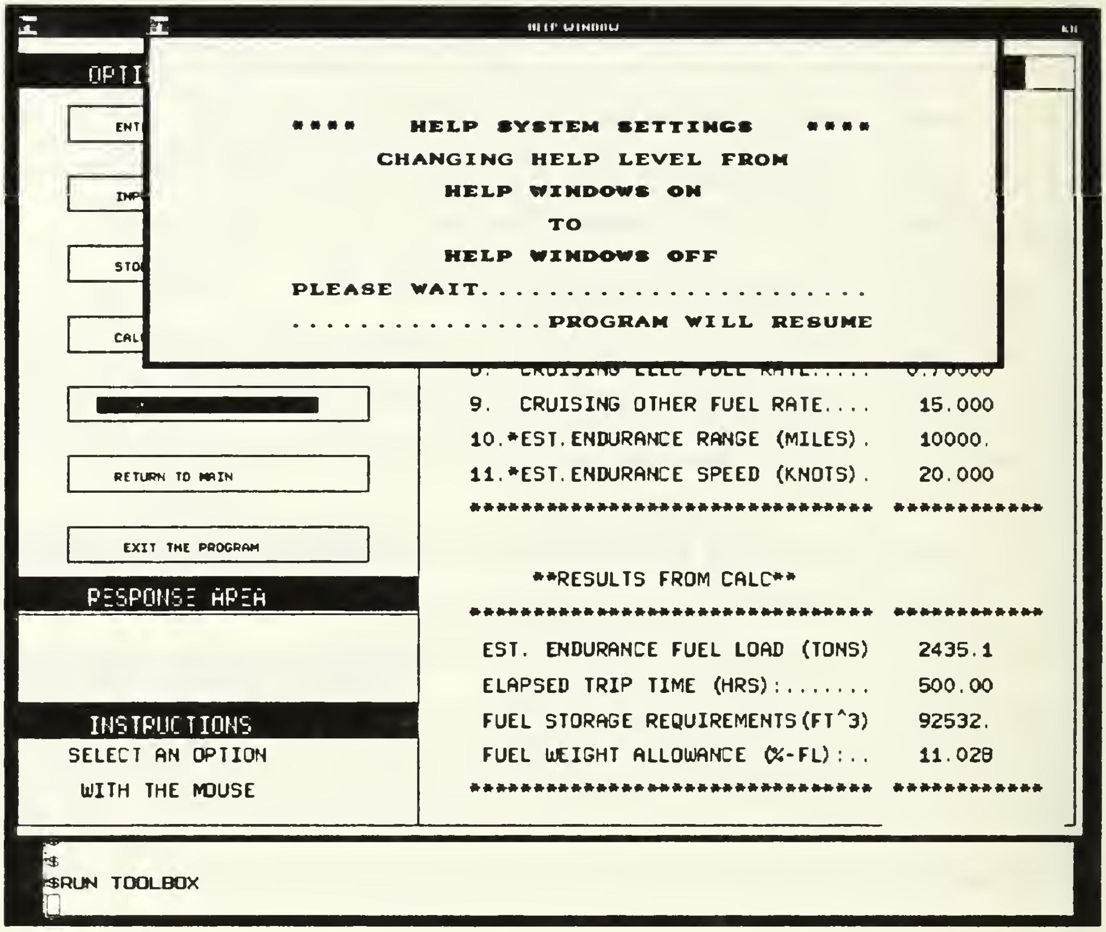
Again, the available output methods are essentially similar to those of the other modules, except that in this case there is no CA-DISSPLA graph output available. The screen displays, and stored data files are all closely related to those presented previously, so we won't devote space to reviewing them.

1. REPORT MODULE Estimated Endurance report

Examining the printed report available from the REPORT Module, we are presented with a two page report detailing all the Endurance variables and the results obtainined from the ENDURANCE Module as shown in Figure 29 on page 57, and Figure 30 on page 58. The method of obtaining a final printout of the report is identical to that used by any of the other modules incorporated into TOOL BOX, and any printout device available may be used.

F. CONCLUSIONS

Thus we conclude the development and discussion of the Endurance Estimation Module. The usefulness of the information calculated here can be very helpful in determining the size of the storage tanks used for Fuel Oil, the predicted or estimated amount of Fuel Oil stowage required in tons, and the percent of full load that this amount constitutes. The parameters that we have calculated all become essential elements of the iterative design process shown in Figure 1 on page 3, and thus the easier we can make the process of obtaining them, then the entire design process will become easier.

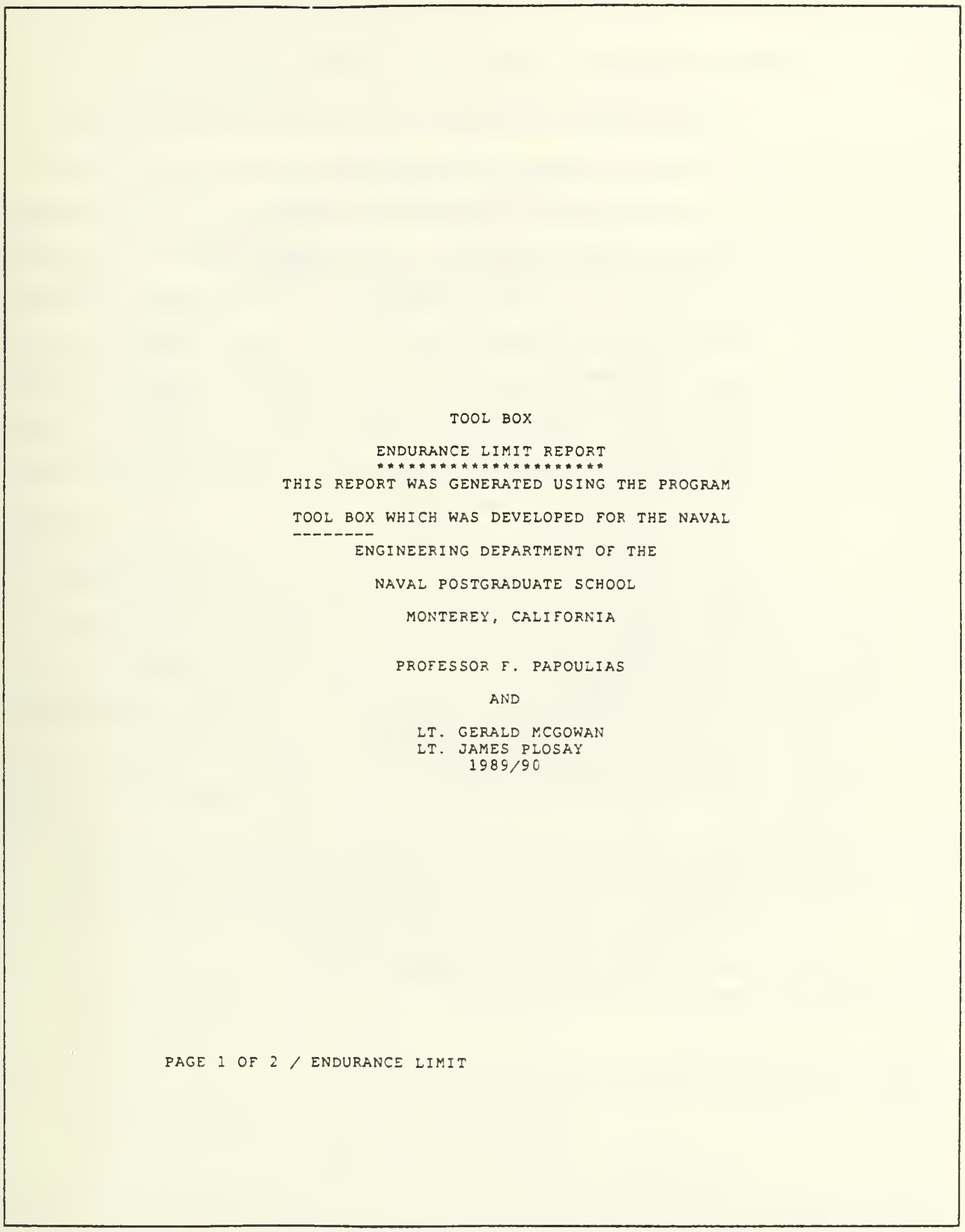


```
****    HELP SYSTEM SETTINGS    ****
      CHANGING HELP LEVEL FROM
         HELP WINDOWS ON
               TO
         HELP WINDOWS OFF
PLEASE WAIT.........................
................PROGRAM WILL RESUME
```

```
9.  CRUISING OTHER FUEL RATE....    15.000
10.*EST.ENDURANCE RANGE (MILES).    10000.
11.*EST.ENDURANCE SPEED (KNOTS).    20.000
********************************  ************

      **RESULTS FROM CALC**
********************************  ************
 EST. ENDURANCE FUEL LOAD (TONS)    2435.1
 ELAPSED TRIP TIME (HRS):.......    500.00
 FUEL STORAGE REQUIREMENTS(FT^3)    92532.
 FUEL WEIGHT ALLOWANCE (%-FL):..    11.028
********************************  ************
```

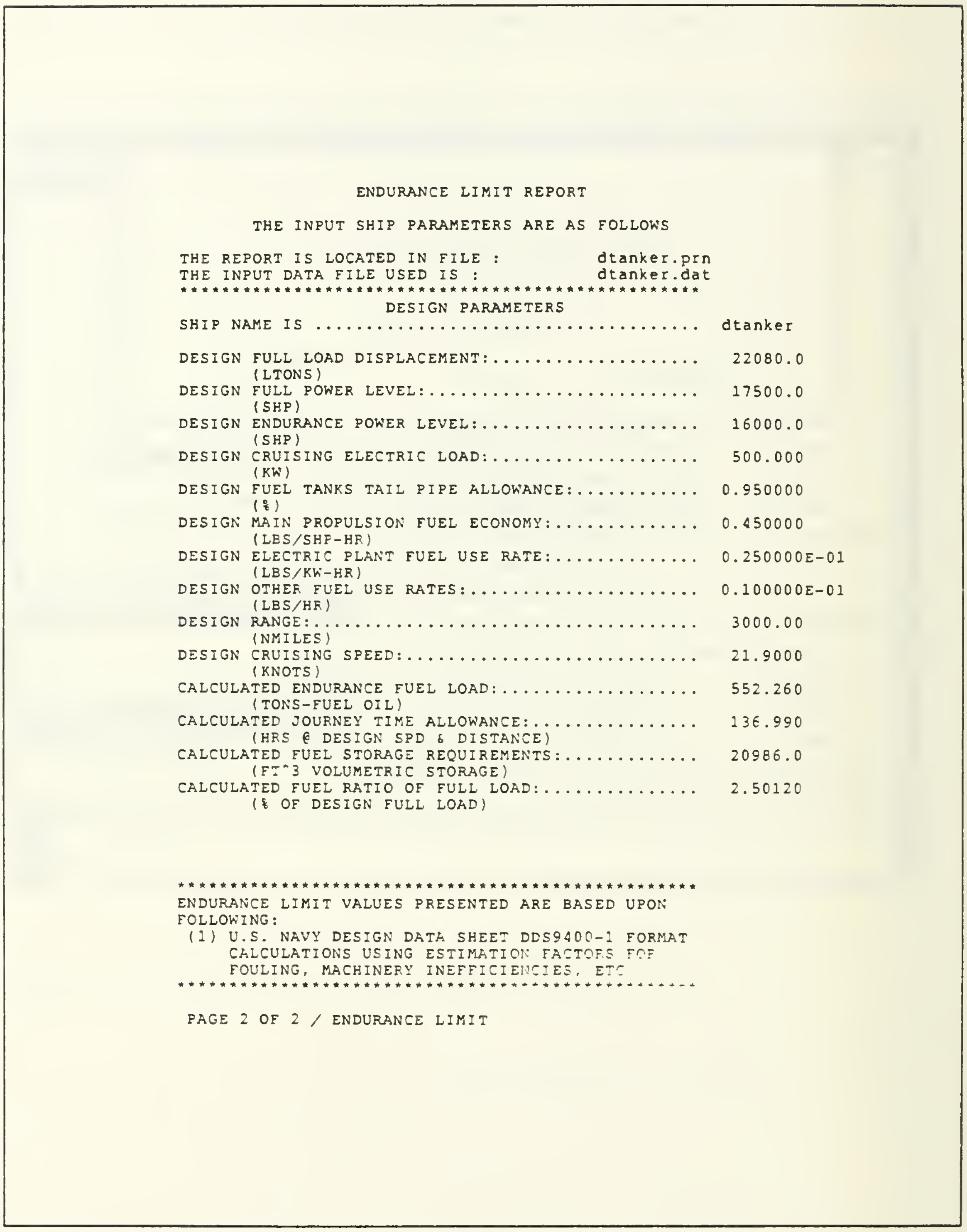
```
$
$RUN TOOLBOX
```

Figure 28.   User Display during Resetting of the Help Level



```
                    TOOL BOX

              ENDURANCE LIMIT REPORT
              **********************
     THIS REPORT WAS GENERATED USING THE PROGRAM

      TOOL BOX WHICH WAS DEVELOPED FOR THE NAVAL
      --------
            ENGINEERING DEPARTMENT OF THE

              NAVAL POSTGRADUATE SCHOOL

                MONTEREY, CALIFORNIA

               PROFESSOR F. PAPOULIAS

                        AND

                 LT. GERALD MCGOWAN
                 LT. JAMES PLOSAY
                     1989/90




PAGE 1 OF 2 / ENDURANCE LIMIT
```

Figure 29.   Endurance Estimation Report, page 1 of 2



```
                 ENDURANCE LIMIT REPORT

       THE INPUT SHIP PARAMETERS ARE AS FOLLOWS

THE REPORT IS LOCATED IN FILE :          dtanker.prn
THE INPUT DATA FILE USED IS :            dtanker.dat
**************************************************
                     DESIGN PARAMETERS
SHIP NAME IS ..................................... dtanker

DESIGN FULL LOAD DISPLACEMENT:....................  22080.0
       (LTONS)
DESIGN FULL POWER LEVEL:..........................  17500.0
       (SHP)
DESIGN ENDURANCE POWER LEVEL:.....................  16000.0
       (SHP)
DESIGN CRUISING ELECTRIC LOAD:....................  500.000
       (KW)
DESIGN FUEL TANKS TAIL PIPE ALLOWANCE:............ 0.950000
       (%)
DESIGN MAIN PROPULSION FUEL ECONOMY:.............. 0.450000
       (LBS/SHP-HR)
DESIGN ELECTRIC PLANT FUEL USE RATE:.............. 0.250000E-01
       (LBS/KW-HR)
DESIGN OTHER FUEL USE RATES:...................... 0.100000E-01
       (LBS/HR)
DESIGN RANGE:.....................................  3000.00
       (NMILES)
DESIGN CRUISING SPEED:............................  21.9000
       (KNOTS)
CALCULATED ENDURANCE FUEL LOAD:...................  552.260
       (TONS-FUEL OIL)
CALCULATED JOURNEY TIME ALLOWANCE:................  136.990
       (HRS @ DESIGN SPD & DISTANCE)
CALCULATED FUEL STORAGE REQUIREMENTS:.............  20986.0
       (FT^3 VOLUMETRIC STORAGE)
CALCULATED FUEL RATIO OF FULL LOAD:...............  2.50120
       (% OF DESIGN FULL LOAD)



**************************************************
ENDURANCE LIMIT VALUES PRESENTED ARE BASED UPON
FOLLOWING:
 (1) U.S. NAVY DESIGN DATA SHEET DDS9400-1 FORMAT
     CALCULATIONS USING ESTIMATION FACTORS FOR
     FOULING, MACHINERY INEFFICIENCIES, ETC
**************************************************

 PAGE 2 OF 2 / ENDURANCE LIMIT
```

Figure 30.   Endurance Estimation Report, page 2 of 2

V. CONCLUSIONS AND RECOMMENDATIONS

A. WINDOWING GUI'S AND PRELIMINARY SHIP DESIGN

We have seen in the original paper [Ref. 3] that we could design and program a Windowing GUI to accomplish preliminary ship design calculations using the capabilities of the VAX/VMS User Interface Services (UIS) routines and the FORTRAN programming language. Further, we have shown here that we could use these capabilities to provide context sensitive HELP information to the user, and enhance the user interface by the addition of Dialog boxes that provide information at appropriate moments during the execution of the program, thus negating a requirement to have a Users Manual handy at all times, or to require specific and lengthy training in the use of the program.

B. CONCLUSIONS

These capabilities provide us with the ability to develop a powerful and easy to use in-house development environment for working with preliminary design problems that is easily expanded with new Modules and calculational routines as the needs of the Students and Faculty evolve, and that places no limitations on the developer. Specifically, we can summarize the attributes of *TOOL BOX* in the following manner:

- Easily utilized by design engineers and students, without lengthy training or complicated users manuals, to provide basic ship design parameters and study the effects of changing those parameters in an interactive design environment using modern graphical programming techniques.
- Follows accepted progression along an Iterative Design Spiral commonly in use in the Marine Industry today, thus affording the developer a logical progression thru the Preliminary Design process.
- Provides valuable information in a timely fashion to the developer in an easily accessed manner, thus avoiding the constraints of current Computer Aided Design (CAD) programs that do not utilize modularity.
- Easily enhanced by the addition or modification of any module.
- Easily programmed in whatever development language is most familiar or appropriate. Here we have used FORTRAN; the UIS routines however are non-specific, and can be utilized by any language supported on the VAX Network, including C and PASCAL.

C. RECOMMENDATIONS

Throughout the design and programming of the additions to TOOL BOX that we have made, we have kept a fairly lengthy ***wish list*** of things that we would do better or different given the opportunity. Things that I feel need to be accomplished in the next version include the following:

- Continue the progression along the Iterative Design Spiral, continuing to add functionality and utility to the core TOOL BOX code.
- Figure out a method of having more AST functions enabled simulaneously than we now use. It has been reported by the original author that too many of these functions enabled simultaneously causes a system slow-down and sluggishness. There should be a means to get around this capability.
- Reprogram the opening menu, and perhaps the Module menus, to accomodate a wider range of user choices. As it stands currently, the Opening Menu has a single RESERVE module left for use. This needs to be increased if we are going to accomodate the entire design spiral of Figure 1 on page 3 into TOOL BOX. The mouse boxes could be made smaller or grouped into two or more columns to accomodate this feature.
- Another possibility would be to group the Module functions into sub-groups, perhaps having a single Main Menu box used to select the two functions we have accomplished here, instead of each Module having its own selection box on the Main menu. Then there would need to be sub-menus to select which of the two functions was desired from those available to the sub-group.
- Enhanced use of color would make the system more user friendly and visually attractive. The HELP windows could easily be made different in color from the rest of the program, thus distinguishing them from the main program.
- Find a means of adding a FILE-NOT-FOUND recovery capability to the READ DATA FROM FILE selections that appear on all of the Module menus, thus avoiding program crashes due to a simple spelling mistake in data entry procedures.
- Find a means, perhaps thru some assembly language routines, to significantly increase the speed of the program. There are a few places where the system response has diminished as the code size grew. This program now accounts for roughly 8000 lines of FORTRAN source code, a language that has never been known for its speed.

Finally, our last recommendation is that the development of TOOL BOX continue along these lines and those already established. This program could become a unique and easily adaptable *tool* for the Total Ship Systems Engineering (TSSE) curriculum at NPS Monterey. This program would provide a friendly computer-based development environment for the TSSE students and greatly aid in preliminary design problems, thus allowing the students to concentrate on new technologies and methods, rather than laboriously computing oft-required, and oft-changing variable parameters such as those provided here.

Complete copies of the TOOL BOX source code, including parts not covered in this document, may be obtained from Professor F.A. Papoulias, Code ME/Pa, Naval Postgraduate School, Monterey CA 94943-5000.

APPENDIX A. TYPICAL DIALOG BOX SOURCE CODE

A. VISUAL DISPLAY INFORMATION BOX

This is the source code for the pop-up information display boxes utilized in the program.

```
      SUBROUTINE SHOW_SAVE
C*********************************************************************
C                                                                    *
C     THIS SUBROUTINE POPS UP A DIALOG BOX TO INSTRUCT THE USER NOT  *
C     TO USE THE SAME DATA FILE NAME AS USED IN A PREVIOUS PROGRAM   *
C     SECTION [SINCE IT WILL OVERWRITE THE OLD ONE CAUSING ERRORS    *
C     OR LOST DATA].  IT IT CALLED BY THE SUBROUTINE THAT SAVES THE  *
C     DATA FILES                                                     *
C                                                                    *
C*********************************************************************
C
      IMPLICIT INTEGER(A-Z)
      INCLUDE 'SYS$LIBRARY:UISENTRY'
      INCLUDE 'SYS$LIBRARY:UISUSRDEF'
      CHARACTER*34 SAVE1/'*** SAVING DEFINED DATA FILE  *** '/
      CHARACTER*34 SAVE2/' DO NOT USE SAME NAME & EXTENSION '/
      CHARACTER*34 SAVE3/'AS DATA FILE FOR OTHER SUB-SECTION'/
      CHARACTER*34 SAVE4/'PLEASE WAIT.......................'/
      CHARACTER*34 SAVE5/'...............PROGRAM WILL RESUME'/
      REAL Y_POSN
C*****CREATE THE DISPLAY FOR THE DIALOG BOX
      VD_SAVE=UIS$CREATE_DISPLAY(0.0,0.0,10.0,5.0,20.0,6.0)
C*****CREATE THE WINDOW TO DISPLAY THE TEXT IN
      WD_SAVE=UIS$CREATE_WINDOW(VD_SAVE,'SYS$WORKSTATION','HELP BOX')
C*****COPY ATTRIBUTE BLOCK '0' AS BLOCK '21' AND CHANGE THE FONT SIZE
      CALL UIS$SET_FONT(VD_SAVE,0,21,'DTABER0R03WK00GG0001UZZZZ02A000')
C*****SIGNAL THE USER TO GET THEIR ATTENTION
      CALL UIS$SOUND_BELL('SYS$WORKSTATION',4)
C*****WRITE THE TEXT INTO THE WINDOW
        Y_POSN=4.0
        CALL UIS$TEXT(VD_SAVE,21,SAVE1,0.2,Y_POSN)
        Y_POSN=Y_POSN-.8
        CALL UIS$TEXT(VD_SAVE,21,SAVE2,0.2,Y_POSN)
        Y_POSN=Y_POSN-.8
        CALL UIS$TEXT(VD_SAVE,21,SAVE3,0.2,Y_POSN)
        Y_POSN=Y_POSN-.8
        CALL UIS$TEXT(VD_SAVE,21,SAVE4,0.2,Y_POSN)
        Y_POSN=Y_POSN-.8
        CALL UIS$TEXT(VD_SAVE,21,SAVE5,0.2,Y_POSN)
C*****SPAWN A NEW PROCESS TO WAIT 15 SECONDS AND COMPLETE PROCESS
      CALL LIB$SPAWN('WAIT 00:00:15')
      CALL UIS$SOUND_BELL('SYS$WORKSTATION',4)
      CALL UIS$DELETE_DISPLAY(VD_SAVE)
      RETURN
      END
```

B. CONVERSATIONAL DIALOG BOX

This segment of code produces the dialog boxes that require user input to continue.

```
        SUBROUTINE NOWHERE
C*********************************************************************
C                                                                    *
C      THIS SUBROUTINE IS CALLED BY A MOUSE BUTTON SELECTION OF AN   *
C      OPTION THAT IS NOT ENABLED YET.  IT POPS UP A DIALOG 'HELP'   *
C      BOX THAT INSTRUCTS THE USER AND GIVES THEM THE OPTION OF      *
C      CONTINUING BACK TO THE PROGRAM AT THE POINT OF INTERRUPTION   *
C      OR ENDING THE PROGRAM ENTIRELY.                               *
C                                                                    *
C*********************************************************************
C
        IMPLICIT INTEGER(A-Z)
        INCLUDE 'SYS$LIBRARY:UISENTRY'
        INCLUDE 'SYS$LIBRARY:UISUSRDEF'
        CHARACTER*34 HINT1/'*** THIS FEATURE NOT AVAILABLE ***'/
        CHARACTER*34 HINT2/'    TYPE "CONTINUE" TO RESUME     '/
        CHARACTER*34 HINT3/'    OR "EXIT" TO END PROGRAM      '/
        REAL Y_POSN
C******CREATE THE DISPLAY WINDOW FOR THE DIALOG BOX
        VD_TEST=UIS$CREATE_DISPLAY(0.0,0.0,10.0,5.0,20.0,5.0)
C******CREATE THE WINDOW TO DISPLAY THE HELP TEXT
        WD_TEST=UIS$CREATE_WINDOW(VD_TEST,'SYS$WORKSTATION','HELP BOX')
C******SET THE SCREEN FONTS WE DESIRE FROM ATTRIBUTE BLOCK '0' TO BLOCK
C******'21' AND CALL THE FILENAME OF THE FONT
        CALL UIS$SET_FONT(VD_TEST,0,21,'DTABER0R03WK00GG0001UZZZZ02A000')
C******SIGNAL THE USER ON PROCESS PAUSE FOR DISPLAY OF HELP BOX
        CALL UIS$SOUND_BELL('SYS$WORKSTATION',4)
C******PRINT THE TEXT AS WE WANT IT
        Y_POSN=4.0
        CALL UIS$TEXT(VD_TEST,21,HINT1,0.4,Y_POSN)
        Y_POSN=Y_POSN-1
        CALL UIS$TEXT(VD_TEST,21,HINT2,0.4,Y_POSN)
        Y_POSN=Y_POSN-1
        CALL UIS$TEXT(VD_TEST,21,HINT3,0.4,Y_POSN)
        PAUSE
C******SIGNAL THE USER PROCESS RESTARTED
        CALL UIS$SOUND_BELL('SYS$WORKSTATION',4)
C******RETURN TO CALLING PROCESS AT POINT OF INTERRUPT
        CALL UIS$DELETE_DISPLAY(VD_TEST)
        RETURN
        END
```

APPENDIX B. TOOL BOX POWER PREDICTION

A. SUBROUTINE POWER SOURCE CODE

This is the source code for the **TOOL BOX** Power Prediction module and all the subroutines called by the module. The INCLUDE File TOP_POWER.FOR follows this section.

```
      SUBROUTINE POWER
C***********************************************************************
C                                                                      *
C   THIS SUBROUTINE IS THE MAIN PART OF PROPULSIVE POWER CALCULATIONS. *
C   IT SETS UP SCREENS AND CONTROLS POWER PROGRAM ACTIONS.             *
C                                                                      *
C   IT IS CALLED FROM THE MAIN TOOL BOX MENU BY THE "POWER PREDICTION" *
C   SELECTION USING THE MOUSE.  IT REQUIRES THE FOLLOWING INPUTS FOR   *
C   OPERATION AFTER INITIALIZATION:                                    *
C      PW_FNAME   SHIP NAME GIVEN TO SAVED DATA FILES         CHAR     *
C      PW_LPP     SHIP LENGTH BETWEEN PERPENDICULARS (FT)     REAL     *
C      PW_NSHAFT  NUMBER OF PROPULSION SHAFTS ( 1 OR 2 )      REAL     *
C      PW_DISP    DISPLACEMENT (TONS STD SEAWATER)            REAL     *
C      PW_VK      SHIP SPEED (KTS)                            REAL     *
C      PW_B       SHIP BEAM (FT)                              REAL     *
C      PW_T       SHIP DRAFT (FT)                             REAL     *
C      PW_CB      SHIP BLOCK COEFFICIENT                      REAL     *
C      PW_NRPM    SHAFT SPEED (RPM)                           REAL     *
C                                                                      *
C   AND CALCULATES THE FOLLOWING INTERMEDIATE VALUES:                  *
C      PW_NRPS    SHAFT SPEED (RPS)                                    *
C      PW_LPPM    SHIP LENGTH BETWEEN PERPENDICULARS (METERS)          *
C      PW_VKM     SHIP SPEED (METERS PER SECOND)                       *
C      PW_DISPM   SHIP DISPLACEMENT (METRIC TONS)                      *
C      PW_VB      S&D BOUNDARY SPEED VALUE                             *
C      PW_SLR     S&D SPEED-LENGTH RATIO                               *
C      PW_BTR     S&D BEAM-DRAFT RATIO                                 *
C      PW_LBR     S&D LENGTH-BEAM RATIO                                *
C      &&&&&&&    AND OTHER CALCULATION-SPECIFIC VARIABLES             *
C                                                                      *
C   AND OUTPUTS THE FOLLOWING VALUES:                                  *
C      PW_ADME    ADMIRALTY POWER PREDICTION                           *
C      PW_SDNB    S&D PREDICTION FOR NO BULBOUS BOW                    *
C      PW_SDBB    S&D PREDICTION FOR SHIP INCLUDING A BULBOUS BOW      *
C      *******    ALL INPUT DESIGN VARIABLES                           *
C      &&&&&&&    UPDATED VALUES OF INPUTS IF ANY PARAMETER CHANGES    *
C                                                                      *
C   ALL PARAMETERS ARE PASSED IN COMMON BY THE INCLUDE 'TOP_POWER.FOR' *
C      FILE THAT HAS ALL THE PARAMETER DEFINITIONS, TYPE STATEMENTS,   *
C      AND SET UP.                                                     *
C                                                                      *
C   CALLED BY MAIN MODULE 'TOOL_BOX'                                   *
```

```
C   CALLS SUBROUTINES 'POWER_KB_DATA_IN','PW_READ_FILE',PW_RECORD',      *
C         'PW_CALC_1','POWER_GRAPH','TO_MAIN',AND 'EXIT'                 *
C                                                                        *
C************************************************************************
        INCLUDE 'GENERAL.FOR'
        INCLUDE 'TOP_POWER.FOR'
        EXTERNAL TO_MAIN,EXIT,DARK2,LIGHT2,NOWHERE,PW_CALC_2
        EXTERNAL POWER_KB_DATA_IN,PW_RECORD,PW_READ_FILE,PW_CALC_1
        EXTERNAL PW_SDI_LABEL,PW_SORT,PW_FAIL,POWER_GRAPH
        EXTERNAL G_DISK,G_SCREEN
C*****INITIALIZE THE PARAMETERS TO AVOID DIVIDE BY ZERO ERRORS.  ALSO
C     THIS INSURES THAT WHEN WE INITIALLY CALL THE KEYBOARD DATA ENTRY
C     ROUTINE THAT THE DATA PASSES ALL THE 'PW_CALC_2' FAILURE
C     CRITERION.  OTHERWISE WE HAVE THE POTENTIAL TO TAKE "FOREVER"
C     WAITING THROUGH THE HELP SCREENS AS WE SLOWLY ENTER OUR DESIGN
C     DATA.  ALSO, IT GIVES THE USER AN IDEA OF WHAT KIND OF SHIP
C     WILL GENERATE VALID DATA.
        IF ( PW_TIMES .EQ. 0 ) THEN
           PW_KB_UP(1)='GENCARGO'
           PW_INFILE='GENCARGO'
           PW_FNAME='GENCARGO.DAT'
           PW_NSHAFT=1.0
           PW_KB_UP(9)='1.0'
           PW_VK=20.80
           PW_KB_UP(7)='20.80'
           PW_LPP=582.50
           PW_KB_UP(2)='582.50'
           PW_T=35.0
           PW_KB_UP(3)='35.0'
           PW_B=82.0
           PW_KB_UP(4)='82.0'
           PW_DISP=31995.0
           PW_KB_UP(5)='31995.0'
           PW_CB=0.6698
           PW_KB_UP(6)='0.66980'
           PW_NRPM=120.0
           PW_KB_UP(8)='120.0'
        ENDIF
        PW_TIMES=PW_TIMES+1
C*****ERASE THE DATA AREA
        CALL UIS$ERASE(VD_ID,3.6,10.1,9.9,18.5)
C*****REMOVE THE MOUSE POINTER TO SOME OTHER AREA
        STA = UIS$SET_POINTER_POSITION(VD_ID,WD_MAIN,9.9,4.5)
C*****CREATE POWER WINDOW
        WD_POWER=UIS$CREATE_WINDOW(VD_ID,'SYS$WORKSTATION','POWER
     &  PREDICTION WINDOW',-.5,9.9,10.1,19.,40.,30.)
        DO 10 Y_COOR = 13.1,18.4,.8
          DY = Y_COOR + .2
          CALL UIS$SET_POINTER_AST(VD_ID,WD_POWER,DARK2,,X0,Y_COOR,X1,
     &    DY,LIGHT2)
 10     CONTINUE
C*****SET MENU TITLES
        OPTION(11) = OPTION(21)
        OPTION(12) = OPTION(20)
C*****WRITE SCREEN
        CALL UIS$TEXT(VD_ID,0,OPTION(11),.3,15.1)
```

```
      CALL UIS$TEXT(VD_ID,0,OPTION(12),.3,15.9)
      CALL UIS$SET_BUTTON_AST(VD_ID,WD_POWER,EXIT,,,X0,13.0,
     &     X1,13.4)
      CALL UIS$SET_BUTTON_AST(VD_ID,WD_POWER,TO_MAIN,,,X0,13.8,
     &     X1,14.2)
      CALL UIS$SET_BUTTON_AST(VD_ID,WD_POWER,G_SCREEN,,,X0,14.6,
     &     X1,15.0)
      CALL UIS$SET_BUTTON_AST(VD_ID,WD_POWER,G_DISK,,,X0,15.4,
     &     X1,15.8)
      CALL UIS$SET_BUTTON_AST(VD_ID,WD_POWER,PW_RECORD,,,X0,16.2,
     &     X1,16.6)
      CALL UIS$SET_BUTTON_AST(VD_ID,WD_POWER,PW_READ_FILE,,,X0
     &     ,17.,X1,17.4)
      CALL UIS$SET_BUTTON_AST(VD_ID,WD_POWER,POWER_KB_DATA_IN,
     &     ,,X0,17.8, X1,18.2)
 19   WNDOW = WD_POWER
 20   RETURN
      END
C
C
     SUBROUTINE POWER_KB_DATA_IN
C*********************************************************************
C                                                                    *
C    THE POWER_KB_DATA_IN ROUTINES LOAD IN KB DATA AND ALLOWS        *
C    THE OPERATOR TO CHANGE INPUTS AND SEE REAL TIME EFFECTS         *
C    OF THOSE CHANGES ON THE POWERING CHARACTERISTICS OF THE         *
C    PRELIMINARY DESIGN.                                             *
C                                                                    *
C    CALLED BY SUBROUTINE 'POWER'                                    *
C    CALLS SUBROUTINES 'PW_SDI_LABEL','KEY_READ','PW_SORT',          *
C                      'PW_CALC_1'                                   *
C*********************************************************************
C
      INCLUDE 'TOP.FOR'
      INCLUDE 'TOP_POWER.FOR'
      REAL PW_LINE_NO
      STA=UIS$SET_POINTER_POSITION(VD_ID,WD_POWER,9.9,14.5)
      KB_ID=UIS$CREATE_KB('SYS$WORKSTATION')
      CALL UIS$ENABLE_KB(KB_ID,WD_POWER)
C*****FIRST, WE WE WRITE UP THE LINE LABLES AND FIRST INSTRUCTIONS
      CALL PW_SDI_LABEL(VD_ID)
C*****SECOND, CALCULATE AND DISPLAY STARTING VALUES
      CALL PW_CALC_1
C*****THIRD, ALLOW USER TO CHANGE INPUT THEN
C*****RECALCULATE RESULTS IN REAL TIME
 1    CALL UIS$ERASE(VD_ID,-0.4,10.1,3.4,10.9)
      CALL UIS$TEXT(VD_ID,7,'ENTER A LINE NUMBER',0.0,11.0)
      CALL UIS$TEXT(VD_ID,7,'OR [RETURN] TO EXIT',0.0,10.6)
 3    CALL  KEY_READ (PW_LINE,.TRUE.,0.0,12.0,*300)
      READ (PW_LINE,FMT='(F2.0)',ERR = 5) PW_LINE_NO
      PLINE_NO = INT(PW_LINE_NO)
      GO TO (10,20,30,40,50,60,70,80,90) PLINE_NO
C
C*****IF IT GETS TO HERE A MISTAKE HAS BEEN MADE ********************
C
 5    CALL UIS$ERASE(VD_ID,-0.4,10.1,3.4,10.9)
```

```
      CALL UIS$TEXT(VD_ID,7,'IMPROPER LINE NUMBER',0.0,11.0)
      CALL UIS$TEXT(VD_ID,7,'ENTER A NEW NUMBER PLEASE',0.0,10.6)
      GOTO 3
C
 10   CALL PW_SORT(PLINE_NO,DEL_Y,PW_LINE,*300)
      PW_KB_UP(1) = PW_LINE
      PW_INFILE = PW_LINE
      READ (PW_LINE, FMT='(G)',ERR = 5)
      WRITE (PW_KB_UP(1), FMT='(A)') PW_INFILE
      CALL UIS$ERASE(VD_ID,8.2,17.9,9.9,18.3)
      CALL UIS$TEXT(VD_ID,7,PW_KB_UP(1),8.2,18.2)
      GOTO 1
C
 20   CALL PW_SORT(PLINE_NO,DEL_Y,PW_LINE,*300)
      PW_KB_UP(2) = PW_LINE
      READ (PW_LINE, FMT='(G)',ERR = 5) PW_LPP
      PW_DISP = PW_CB*PW_LPP*PW_B*PW_T/35
      WRITE (PW_KB_UP(5), FMT='(G12.5)') PW_DISP
      CALL UIS$ERASE(VD_ID,8.2,16.3,9.9,16.7)
      CALL UIS$TEXT(VD_ID,7,PW_KB_UP(5),8.2,16.6)
      CALL PW_CALC_1
      GOTO 1
C
 30   CALL PW_SORT(PLINE_NO,DEL_Y,PW_LINE,*300)
      PW_KB_UP(3) = PW_LINE
      READ (PW_LINE, FMT='(G)',ERR = 5) PW_T
      PW_DISP = PW_CB*PW_LPP*PW_B*PW_T/35
      WRITE (PW_KB_UP(5), FMT='(G12.5)') PW_DISP
      CALL UIS$ERASE(VD_ID,8.2,16.3,9.9,16.7)
      CALL UIS$TEXT(VD_ID,7,PW_KB_UP(5),8.2,16.6)
      CALL PW_CALC_1
      GOTO 1
C
 40   CALL PW_SORT(PLINE_NO,DEL_Y,PW_LINE,*300)
      PW_KB_UP(4) = PW_LINE
      READ (PW_LINE, FMT='(G)',ERR = 5) PW_B
      PW_DISP = PW_CB*PW_LPP*PW_B*PW_T/35
      WRITE (PW_KB_UP(5), FMT='(G12.5)') PW_DISP
      CALL UIS$ERASE(VD_ID,8.2,16.3,9.9,16.7)
      CALL UIS$TEXT(VD_ID,7,PW_KB_UP(5),8.2,16.6)
      CALL PW_CALC_1
      GOTO 1
C
 50   CALL PW_SORT(PLINE_NO,DEL_Y,PW_LINE,*300)
      PW_KB_UP(5) = PW_LINE
      READ (PW_LINE, FMT='(G)',ERR = 5) PW_DISP
      PW_CB=35*PW_DISP/(PW_LPP*PW_B*PW_T)
      WRITE (PW_KB_UP(6), FMT='(G12.5)') PW_CB
      CALL UIS$ERASE(VD_ID,8.2,15.9,9.9,16.3)
      CALL UIS$TEXT(VD_ID,7,PW_KB_UP(6),8.2,16.2)
      CALL PW_CALC_1
      GOTO 1
C
 60   CALL PW_SORT(PLINE_NO,DEL_Y,PW_LINE,*300)
      PW_KB_UP(6) = PW_LINE
      READ (PW_LINE, FMT='(G)',ERR = 5) PW_CB
      PW_DISP = PW_CB*PW_LPP*PW_B*PW_T/35
      WRITE (PW_KB_UP(5), FMT='(G12.5)') PW_DISP
      CALL UIS$ERASE(VD_ID,8.2,16.3,9.9,16.7)
      CALL UIS$TEXT(VD_ID,7,PW_KB_UP(5),8.2,16.6)
      CALL PW_CALC_1
      GOTO 1
C
 70   CALL PW_SORT(PLINE_NO,DEL_Y,PW_LINE,*300)
      PW_KB_UP(7) = PW_LINE
      READ (PW_LINE, FMT='(G)',ERR = 5) PW_VK
      WRITE (PW_KB_UP(7), FMT='(G12.5)') PW_VK
      CALL PW_CALC_1
      GOTO 1
C
 80   CALL PW_SORT(PLINE_NO,DEL_Y,PW_LINE,*300)
      PW_KB_UP(8) = PW_LINE
      READ (PW_LINE, FMT='(G)',ERR = 5) PW_NRPM
      WRITE (PW_KB_UP(8), FMT='(G12.5)') PW_NRPM
      CALL PW_CALC_1
      GOTO 1
C
 90   CALL PW_SORT(PLINE_NO,DEL_Y,PW_LINE,*300)
      PW_KB_UP(9) = PW_LINE
      READ (PW_LINE, FMT='(G)',ERR = 5) PW_NSHAFT
      WRITE (PW_KB_UP(9), FMT='(G12.5)') PW_NSHAFT
      CALL PW_CALC_1
      GOTO 1
C
 300  CALL UIS$DISABLE_KB(KB_ID)
      CALL UIS$ERASE(VD_ID,-.4,10.1,3.4,10.9)
      CALL UIS$TEXT(VD_ID,7,'SELECT AN OPTION',0.,11.)
      CALL UIS$TEXT(VD_ID,7,' WITH THE MOUSE ',0.,10.6)
      RETURN
      END
C
C
      SUBROUTINE PW_SORT(PLINE_NO,DEL_Y,PW_LINE,COUNT,*)
C*********************************************************************
C                                                                    *
C     SUBROUTINE PW_SORT SORTS OUT THE LINE NUMBER TO THE            *
C     VARIABLE INVOLVED AND CALLS THE APPROPRIATE SCREEN             *
C     INSTRUCTION TO BE WRITTEN IN THE INSTRUCTION BOX.              *
C                                                                    *
C     CALLS SUBROUTINE 'KEY_READ'                                    *
C*********************************************************************
C
      INCLUDE 'TOP.FOR'
      INCLUDE 'TOP_POWER.FOR'
      REAL DEL_Y, DY, DOT_Y
      LOGICAL PW_FCHAR
      CHARACTER PW_KB_INST(9)*25
C*****SET THE SCREEN INSTRUCTIONS STRINGS
      PW_KB_INST(1) = 'ENTER NEW SHIP NAME'
      PW_KB_INST(2) = 'ENTER LENGTH (LPP) IN FT'
      PW_KB_INST(3) = 'ENTER DRAFT (T) IN FT.'
      PW_KB_INST(4) = 'ENTER BEAM (B) IN FT.'
      PW_KB_INST(5) = 'DISPLACEMENT IN LTONS'
      PW_KB_INST(6) = 'ENTER BLOCK COEFICIENT'
      PW_KB_INST(7) = 'ENTER SHIP SPEED IN KNOTS'
      PW_KB_INST(8) = 'ENTER PROPELLER RPM'
      PW_KB_INST(9) = 'ENTER NUMBER OF SHAFTS'
C*****ERASE THE INSTRUCTION AREA AND FIND ITS DATA POSITION
      CALL UIS$ERASE(VD_ID,-0.4,10.1,3.4,10.9)
      DEL_Y = 18.6 - .4 * PLINE_NO
C*****WRITE THE APPROPRIATE INSTRUCTION STRING
      CALL UIS$TEXT(VD_ID,7,PW_KB_INST(PLINE_NO),0.0,11.0)
      CALL UIS$TEXT(VD_ID,7,'[RETURN] TO EXIT',0.0,10.6)
C*****DECIDE IF WE WANT THE FIRST LINE (NAME) OR A VALUE
      IF (PLINE_NO .EQ. 1) THEN
          PW_FCHAR = .FALSE.
      ELSE
          PW_FCHAR = .TRUE.
      ENDIF
      CALL KEY_READ(PW_LINE,PW_FCHAR,0.0,12.0,*300)
      DY = DEL_Y - .35
      CALL UIS$ERASE(VD_ID,8.2,DY,9.9,DEL_Y)
      CALL UIS$TEXT(VD_ID,7,PW_LINE,8.2,DEL_Y)
      RETURN
 300  RETURN 1
      END
C
C
       SUBROUTINE PW_SDI_LABEL(VD_ID)
C*******************************************************************
C                                                                  *
C  PERFORM DISPLAY VARIABLES IS A ROUTINE WHICH READS UP THE       *
C  DATA ONTO THE DISPLAY AREA ALONG WITH THE SCREEN LABELS.        *
C                                                                  *
C  CALLED BY SUBROUTINES 'POWER','POWER_KB_DATA_IN'                *
C*******************************************************************
C
        IMPLICIT INTEGER (A-Z)
        INCLUDE 'TOP_POWER.FOR'
        INCLUDE 'SYS$LIBRARY:UISENTRY'
        INCLUDE 'SYS$LIBRARY:UISUSRDEF'
        REAL DEL_Y, DOT_Y
        CHARACTER P_LINE_LBL(20)*35
        CALL UIS$ERASE(VD_ID,4.,10.1,9.9,18.3)
        P_LINE_LBL(1) =  '1.  SHIP NAME ..................'
        P_LINE_LBL(2) =  '2.  LENGTH (Lpp)..........(FT)..'
        P_LINE_LBL(3) =  '3.  DESIGN DRAFT .........(FT)..'
        P_LINE_LBL(4) =  '4.  BEAM .................(FT)..'
        P_LINE_LBL(5) =  '5.  DISPLACEMENT ........(LTONS)'
        P_LINE_LBL(6) =  '6.  BLOCK COEFICIENT......(Cb)..'
        P_LINE_LBL(7) =  '7.  SHIP SPEED............(KTS).'
        P_LINE_LBL(8) =  '8.  PROPELLER SPEED.......(RPM).'
        P_LINE_LBL(9) =  '9.  NUMBER OF SHAFTS............'
        P_LINE_LBL(10) = ' *****END OF REQD INPUTS********'
        P_LINE_LBL(11) = ' '
        P_LINE_LBL(12) = ' '
        P_LINE_LBL(13) = '       ***RESULTS***'
        P_LINE_LBL(14) = ' ******HORSEPOWER.(HP)**********'
        P_LINE_LBL(15) = ' .METHOD:........................'
        P_LINE_LBL(16) = ' ........ADMIRALTY...............'
        P_LINE_LBL(17) = ' ........S.&.D.W/.NO BB.........'
        P_LINE_LBL(18) = ' ........S.&.D.W/....BB.........'
        P_LINE_LBL(19) = ' .BB=.WELL.DESIGNED.BULB.BOW....'
        P_LINE_LBL(20) = ' ********************************'
        DO 10 I = 1,20,1
           DEL_Y = 18.6  - .4*I
           CALL UIS$TEXT(VD_ID,7,P_LINE_LBL(I),4.,DEL_Y)
10      CONTINUE
        DO 20 I = 1,12,1
           DEL_Y = 18.6  - .4*I
           CALL UIS$TEXT(VD_ID,7,PW_KB_UP(I),8.2,DEL_Y)
20      CONTINUE
        RETURN
        END
C
C
        SUBROUTINE PW_CALC_1
C*********************************************************************
C                                                                    *
C  SUBROUTINE PW_CALC_1                                              *
C  PW_CALC_1 CALLS PW_CALC_2 WHICH DOES THE POWER (HP)               *
C  CALCULATIONS.  ON RETURN, PW_CALC_1 DISPLAYS THE RESULTS          *
C  OF THE CALCULATIONS AND THEN RETURNS TO THE ROOT PROGRAM.         *
C                                                                    *
C  CALLED BY SUBROUTINES 'POWER','POWER_KB_DATA_IN','POWER_GRAPH'   *
C  CALLS SUBROUTINE 'PW_CALC_2'                                      *
C ********************************************************************
C
        INCLUDE 'TOP.FOR'
        INCLUDE 'TOP_POWER.FOR'
        REAL DY
        CALL UIS$ERASE(VD_ID,8.1,9.95,9.9,13.)
        CALL PW_CALC_2
        WRITE( PW_KB_UP(16), FMT='(G12.5)') PW_ADME
        WRITE( PW_KB_UP(17), FMT='(G12.5)') PW_SDNB
        WRITE( PW_KB_UP(18), FMT='(G12.5)') PW_SDBB
        PW_KB_UP(10) = '************'
        PW_KB_UP(11) = '  '
        PW_KB_UP(12) = '  '
        PW_KB_UP(13) = '  '
        PW_KB_UP(14) = '************'
        PW_KB_UP(15) = '  '
        PW_KB_UP(19) = '  '
        PW_KB_UP(20) = '************'
        DO 10 I = 10,20,1
           DY = 18.6 -.4*I
           CALL UIS$TEXT(VD_ID,7,PW_KB_UP(I),8.2,DY)
10      CONTINUE
        RETURN
        END
C
C
      SUBROUTINE PW_CALC_2
C*********************************************************************
```

```
C                                                                       *
C  SUBROUTINE PW_CALC_2 JUST DOES THE CALCULATION OF THE POWER          *
C  REQUIREMENTS FROM THE INPUT DATA.  IT IS BASED UPON TWO              *
C  ALGORITHMS FOR POWER PREDICTION: THE ADMIRALTY COEFFICIENT,          *
C  AND THE METHOD OF SILVERLEAF AND DAWSON AS MODIFIED BY STIAN         *
C  ERICHSEN AT UNIV. OF MICHIGAN.  IT IS ALSO USED IN THE REPORT        *
C  SECTION FOR GENERATION OF THE WRITTEN REPORT OF THE DESIGN.          *
C                                                                       *
C  CALLED BY SUBROUTINE 'PW_CALC_1'                                     *
C  CALLS SUBROUTINE 'PW_FAIL'                                           *
C************************************************************************
C
      INCLUDE 'TOP.FOR'
      INCLUDE 'TOP_POWER.FOR'
C*****FIRST CONVERT ENGLISH UNITS TO METRIC FOR ADMIRALTY CALCULATION
C*****KNOTS->M/S ; FT->M ; LTONS->MTONS ;
      PW_NRPS = PW_NRPM/60.0
      PW_LPPM = PW_LPP/3.280
      PW_VKM  = PW_VK*0.514444
      PW_DISPM = PW_DISP/1.016
      I_PW_NSHAFT=INT(PW_NSHAFT)
C*****BEGIN ERROR CHECKING LOOPS FOR ADM CALCULATION
C     THIS SECTION CHECKS THE ADM FORMULA DATA FOR VALIDITY BEFORE
C     THE CALCULATION BEGINS.  IF A PARAMETER IS OUT OF RANGE, THE
C     ERROR CODE IS SET AS 'PW_CAUSE' AND PASSED TO THE SUBROUTINE
C     'PW_FAIL' WHICH DISPLAYS A HELP WINDOW SHOWING THE SPECIFIC
C     REASON THE DATA WAS REJECTED.  THE ADM VALUES ARE SET TO
C     ZERO AND THE PROGRAM CONTINUES TO THE S&D CALCULATION.
      IF ( PW_LPPM .GT. 1162.196 ) THEN
         PW_CAUSE=1
         CALL PW_FAIL(PW_CAUSE)
         PW_ADME=0.0
         GOTO 100
      ENDIF
      IF ( PW_NRPS*(PW_LPPM**0.5) .GT. 136.364 ) THEN
         PW_CAUSE=2
         CALL PW_FAIL(PW_CAUSE)
         PW_ADME=0.0
         GOTO 100
      ENDIF
C*****CALCULATE POWER BY METHOD OF ADMIRALTY COEFFICIENTS
      PW_ADM=5.0*((PW_DISPM**(2.0))**(1.0/3.0))*(PW_VKM**3.0)
     &     *(33.0-0.017*PW_LPPM)/(15000.0-110.0*PW_NRPS*PW_LPPM**0.5)
C*****CONVERT METRIC KILOWATTS BACK TO HORSEPOWER
      PW_ADME = PW_ADM/0.7456999
C*****NOW WE HAVE ADMIRALTY POWER IN HORSEPOWER (HP)
C
 100  CONTINUE
C
C*****NOW CALCULATE S&D POWER AND REQUIRED CORRECTIONS
C
C*****CALCULATE THE BOUNDARY SPEED AND SOME COEFFICIENTS
      PW_VB = (1.70-1.40*PW_CB)*(PW_LPP)**0.5
      PW_SLR = PW_VK/(PW_LPP**0.5)
      PW_BTR = PW_B/PW_T
      PW_LBR = PW_LPP/PW_B
C*****BEGIN ERROR CHECKING LOOPS FOR S&D CALCULATION
C     THIS SECTION CHECKS THE S&D FORMULA DATA FOR VALIDITY BEFORE
C     THE CALCULATION BEGINS.  IF A PARAMETER IS OUT OF RANGE, THE
C     ERROR CODE IS SET AS 'PW_CAUSE' AND PASSED TO THE SUBROUTINE
C     'PW_FAIL' WHICH DISPLAYS A HELP WINDOW SHOWING THE SPECIFIC
C     REASON THE DATA WAS REJECTED.  THE S&D VALUES ARE SET TO
C     ZERO AND THE PROGRAM CONTINUES.
      IF (I_PW_NSHAFT .EQ.  1 ) THEN
         IF (PW_SLR .GT.  1.20 .OR.  PW_SLR .LT.  0.40 ) THEN
            PW_CAUSE=3
            CALL PW_FAIL(PW_CAUSE)
            PW_SDNB=0.0
            PW_SDBB=0.0
            GOTO 200
         ENDIF
         IF (PW_CB .GT.  0.86 .OR.  PW_CB .LT.  0.50 ) THEN
            PW_CAUSE=4
            CALL PW_FAIL(PW_CAUSE)
            PW_SDNB=0.0
            PW_SDBB=0.0
            GOTO 200
         ENDIF
         IF (PW_BTR .GT.  4.50 .OR.  PW_BTR .LT.  2.0 ) THEN
            PW_CAUSE=5
            CALL PW_FAIL(PW_CAUSE)
            PW_SDNB=0.0
            PW_SDBB=0.0
            GOTO 200
         ENDIF
         IF (PW_LBR .GT.  9.50 .OR.  PW_LBR .LT.  3.33 ) THEN
            PW_CAUSE=6
            CALL PW_FAIL(PW_CAUSE)
            PW_SDNB=0.0
            PW_SDBB=0.0
            GOTO 200
         ENDIF
      ELSEIF (I_PW_NSHAFT .EQ.  2 ) THEN
         IF (PW_SLR .GT.  1.20 .OR.  PW_SLR .LT.  0.40 ) THEN
            PW_CAUSE=3
            CALL PW_FAIL(PW_CAUSE)
            PW_SDNB=0.0
            PW_SDBB=0.0
            GOTO 200
         ENDIF
         IF (PW_CB .GT.  0.80 .OR.  PW_CB .LT.  0.54 ) THEN
            PW_CAUSE=4
            CALL PW_FAIL(PW_CAUSE)
            PW_SDNB=0.0
            PW_SDBB=0.0
            GOTO 200
         ENDIF
         IF (PW_BTR .GT.  4.50 .OR.  PW_BTR .LT.  2.0 ) THEN
            PW_CAUSE=5
            CALL PW_FAIL(PW_CAUSE)
            PW_SDNB=0.0
            PW_SDBB=0.0
            GOTO 200
         ENDIF
         IF (PW_LBR .GT. 11.50 .OR. PW_LBR .LT. 3.80 ) THEN
            PW_CAUSE=6
            CALL PW_FAIL(PW_CAUSE)
            PW_SDNB=0.0
            PW_SDBB=0.0
            GOTO 200
         ENDIF
      ELSE
         PW_CAUSE=7
         CALL PW_FAIL(PW_CAUSE)
         PW_SDNB=0.0
         PW_SDBB=0.0
         GOTO 200
      ENDIF
C*****CORRECT FOR SHIP LENGTH
      IF ( PW_LPP .GT. 1000.0 ) PW_CX=0.850
      IF ( PW_LPP .LE. 1000.0 ) PW_CX=0.850+0.00185*
     &                          ((1000.0-PW_LPP)/100.0)**2.5
C*****CORRECT FOR BEAM/DRAFT RATIO
      IF ( PW_B/PW_T .LT. 2.40 ) C_KBT=0.982
      IF ( PW_B/PW_T .GE. 2.40 ) C_KBT=0.960+
     &          0.00054*10.0**((2.0*PW_B)/(3.0*PW_T))
C*****CORRECT FOR SHIP SPEED NOT EQUAL TO BOUNDARY SPEED
      IF (PW_VK/PW_VB .LE. 1.0 ) C_KV=2.75-7.25*(PW_VK/PW_VB)+
     &                           5.50*((PW_VK/PW_VB)**2.0)
      IF (PW_VK/PW_VB .GT. 1.0 ) C_KV=21.2-43.2*(PW_VK/PW_VB)+
     &                           23.0*((PW_VK/PW_VB)**2.0)
C*****CORRECT FOR HYDRODYNAMIC EFFICIENCY AT LPP=400.0 FT
      IF ( I_PW_NSHAFT .EQ. 1 ) C_H400=2.60-0.2917*(PW_VK/
     &                      (PW_DISP**(1.0/6.0)))
      IF ( I_PW_NSHAFT .EQ. 2 ) C_H400=2.38-0.2917*(PW_VK/
     &                      (PW_DISP**(1.0/6.0)))
C*****CORRECT FOR LPP NOT EQUAL TO 400.0 FT
      IF ( PW_LPP .EQ. 400.0 ) C_KLBP=1.0
      IF ( PW_LPP .NE. 400.0 ) C_KLBP=0.9196+(PW_LPP*2.31E-04)-
     &                         (7.5E-08*(PW_LPP**2.0))
C*****CORRECT FOR OPEN WATER EFFICIENCY AT PROP SPEED=120 RPM
      IF ( I_PW_NSHAFT .EQ. 1 ) C_ETA_0120=0.98-0.55*PW_CB
      IF ( I_PW_NSHAFT .EQ. 2 ) C_ETA_0120=0.90-0.33*PW_CB
C*****CORRECT FOR ACTUAL PROP SPEED RPM
      IF ( I_PW_NSHAFT .EQ. 1 ) C_DETA_0=0.360-0.0029*PW_NRPM
      IF ( I_PW_NSHAFT .EQ. 2 ) C_DETA_0=0.135-0.0011*PW_NRPM
C*****ADD THE TWO ABOVE FOR TOTAL PROP SPEED CORRECTION
      PW_ETA_0=C_ETA_0120+C_DETA_0
C*****CALCULATE S&D FOR NO BULBOUS BOW
      TOP_ANS=PW_CX*(PW_DISP**(2.0/3.0))*(PW_VB**3.0)*C_KBT*C_KV
      BOT_ANS=427.1*PW_ETA_0*C_H400*C_KLBP
      PW_SDNB=TOP_ANS/BOT_ANS
C*****ADJUST S&D FOR SHIP WITH BULBOUS BOW
      PW_SDBB=PW_SDNB*0.9500
C*****NOW WE HAVE S&D POWER IN HORSEPOWER (HP) FOR BOTH BOW SHAPES
C     AND THE ADMIRALTY POWER.
C
```

```
 200  CONTINUE
C
      RETURN
      END
C
C
        SUBROUTINE PW_RECORD
C*********************************************************************
C                                                                    *
C    SUBROUTINE 'PW_RECORD' SAVES THE KBD INPUT DATA TO A USER       *
C    SPECIFIED FILE.  FILE TYPE IS '.DAT' AUTOMATICALLY              *
C                                                                    *
C    CALLED BY SUBROUTINE 'POWER'.                                   *
C    CALLS SUBROUTINE 'SHOW_SAVE'                                    *
C*********************************************************************
C
        INCLUDE 'TOP.FOR'
        INCLUDE 'TOP_POWER.FOR'
        STA = UIS$SET_POINTER_POSITION(VD_ID,WD_POWER,0.,10.)
        CALL SHOW_SAVE
        OPEN(21,FILE=PW_INFILE,STATUS='UNKNOWN',blank='null')
        WRITE (21,200) PW_INFILE
        WRITE (21,210) PW_LPP
        WRITE (21,210) PW_T
        WRITE (21,210) PW_B
        WRITE (21,210) PW_DISP
        WRITE (21,210) PW_CB
        WRITE (21,210) PW_VK
        WRITE (21,210) PW_NRPM
        WRITE (21,210) PW_NSHAFT
        WRITE (21,210) PW_ADME
        WRITE (21,210) PW_SDNB
        WRITE (21,210) PW_SDBB
        ENDFILE (21)
        CLOSE(21)
C
C*****AND REPORT OUT WHEN DONE
C
        PW_FNAME = PW_INFILE(:INDEX(PW_INFILE,' '))//'.DAT'
        CALL UIS$ERASE(VD_ID,-.4,10.1,3.4,10.9)
        CALL UIS$TEXT(VD_ID,7,'FILE SAVED AS',0.,11.)
        CALL UIS$TEXT(VD_ID,7,PW_FNAME,0.,10.6)
 200    FORMAT (BN,2X,A)
 210    FORMAT (G12.5)
        RETURN
        END
C
C
       SUBROUTINE PW_READ_FILE
C*********************************************************************
C                                                                    *
C     READ FILE IS A SUBROUTINE WHICH READS THE DATA INPUT FILE      *
C     SPECIFIED BY KEYBOARD ENTRY                                    *
C                                                                    *
C     CALLED BY SUBROUTINE 'POWER'                                   *
C     CALLS SUBROUTINES 'PW_SDI_LABEL','PW_CALC_1'                   *
```

```
C*********************************************************************
C
       INCLUDE 'TOP.FOR'
       INCLUDE 'TOP_POWER.FOR'
       STA=UIS$SET_POINTER_POSITION(VD_ID,WD_POWER,0.,10.2)
       KB_ID=UIS$CREATE_KB('SYS$WORKSTATION')
       CALL UIS$ENABLE_KB(KB_ID,WD_POWER)
       CALL UIS$ERASE(VD_ID,-.4,10.1,2.3,10.9)
       CALL UIS$TEXT(VD_ID,7,'ENTER THE FILE NAME',0.,11.)
       CALL UIS$TEXT(VD_ID,7,'AND FILE EXTENSION',0.,10.6)
       CALL KEY_READ(PW_LINE,'TRUE',0.,12.,*150)
       OPEN(21,FILE=PW_LINE,STATUS='UNKNOWN')
       READ (21,200) PW_INFILE
       READ (21,210) PW_LPP
       READ (21,210) PW_T
       READ (21,210) PW_B
       READ (21,210) PW_DISP
       READ (21,210) PW_CB
       READ (21,210) PW_VK
       READ (21,210) PW_NRPM
       READ (21,210) PW_NSHAFT
       REWIND 21
C**** NOW READ IT IN AS CHARACTER DATA TO PRINT TO SCREEN
       READ (21,200) PW_KB_UP(1)
       DO 10 I = 2,9,1
           READ (21,220) PW_KB_UP(I)
 10    CONTINUE
C
 200   FORMAT (2X,A)
 210   FORMAT (G12.5)
 220   FORMAT (2X,A)
 150   CLOSE(21)
       CALL PW_SDI_LABEL(VD_ID)
       CALL UIS$DISABLE_KB(KB_ID)
       CALL PW_CALC_1
       CALL UIS$ERASE(VD_ID,-.4,10.1,3.4,10.9)
       CALL UIS$TEXT(VD_ID,7,'SELECT AN OPTION',0.,11.)
       CALL UIS$TEXT(VD_ID,7,' WITH THE MOUSE ',0.,10.6)
       RETURN
       END
C
C
C
      SUBROUTINE SAVE_GRAPH
C**********************************************************************
C                                                                     *
C     THIS SUBROUTINE POPS UP A DIALOG BOX TO INSTRUCT THE USER THAT *
C     THE POWER PREDICTION CURVE GRAPH PLOT HAS BEEN WRITTEN TO DISK *
C     UNDER THE NAME 'POWER_PLOT.UIS' AND INSTRUCTS THEM ON HOW TO    *
C     RETRIEVE THE FILE IN HARDCOPY FORM.  IT IS CALLED BY THE SUB-   *
C     ROUTINE 'POWER' SUBMENU SELECTION 'GRAPH POWER' AND IS          *
C     DISPLAYED AFTER THE GRAPH HAS BEEN CREATED, VIEWED, AND SENT    *
C     TO DISK.                                                        *
C                                                                     *
C     CALLED BY SUBROUTINE 'POWER_GRAPH'                              *
```

```
C***************************************************************************
C
      IMPLICIT INTEGER(A-Z)
      INCLUDE 'SYS$LIBRARY:UISENTRY'
      INCLUDE 'SYS$LIBRARY:UISUSRDEF'
      INCLUDE 'TOP_POWER.FOR'
      CHARACTER*34 PLOTA/'*** SAVING DEFINED GRAPH FILE  ***'/
      CHARACTER*34 PLOTB/' GRAPH IS BEING WRITTEN TO DISK   '/
      CHARACTER*34 PLOTC/' USING THE FILENAME:              '/
      CHARACTER*34 PLOTD/'        "STD00001.DAT"            '/
      CHARACTER*34 PLOTE/'   YOU CAN OBTAIN HARDCOPY PLOTS  '/
      CHARACTER*34 PLOTF/' BY THE DCL COMMAND:              '/
      CHARACTER*34 PLOTG/'  "PRINT/QUEUE=LASER STD00001.DAT"'/
      CHARACTER*34 PLOTH/' AFTER TERMINATION OF TOOLBOX.    '/
      CHARACTER*34 PLOTI/'PLEASE WAIT.......................'/
      CHARACTER*34 PLOTJ/'...............PROGRAM WILL RESUME'/
      REAL Y_POSN
C******CREATE THE DISPLAY FOR THE DIALOG BOX
      VD_PLOT=UIS$CREATE_DISPLAY(-5.0,-2.5,10.0,5.0,26.0,14.0)
C******CREATE THE WINDOW TO DISPLAY THE TEXT IN
      WD_PLOT=UIS$CREATE_WINDOW(VD_PLOT,'SYS$WORKSTATION','HELP WINDOW')
C******COPY ATTRIBUTE BLOCK '0' AS BLOCK '23' AND CHANGE THE FONT SIZE
      CALL UIS$SET_FONT(VD_PLOT,0,23,'DTABER0R03WK00GG0001UZZZZ02A000')
C******COPY ATTRIBUTE BLOCK '23' AS BLOCK '24' AND CHANGE TO BOLD FONT
      CALL UIS$SET_FONT(VD_PLOT,23,24,'DTABER0R03WK00PG0001UZZZZ02A000')
C******SIGNAL THE USER TO GET THEIR ATTENTION
      CALL UIS$SOUND_BELL('SYS$WORKSTATION',4)
C******WRITE THE TEXT INTO THE WINDOW AND SPAWN PROCESS TO 'WAIT' 15 SEC
        Y_POSN=4.0
        CALL UIS$TEXT(VD_PLOT,24,PLOTA,-2.5,Y_POSN)
        Y_POSN=Y_POSN-.6
        CALL UIS$TEXT(VD_PLOT,23,PLOTB,-2.5,Y_POSN)
        Y_POSN=Y_POSN-.6
        CALL UIS$TEXT(VD_PLOT,23,PLOTC,-2.5,Y_POSN)
        Y_POSN=Y_POSN-.6
        CALL UIS$TEXT(VD_PLOT,24,PLOTD,-2.5,Y_POSN)
        Y_POSN=Y_POSN-.6
        CALL UIS$TEXT(VD_PLOT,23,PLOTE,-2.5,Y_POSN)
        Y_POSN=Y_POSN-.6
        CALL UIS$TEXT(VD_PLOT,23,PLOTF,-2.5,Y_POSN)
        Y_POSN=Y_POSN-.6
        CALL UIS$TEXT(VD_PLOT,24,PLOTG,-2.5,Y_POSN)
        Y_POSN=Y_POSN-.6
        CALL UIS$TEXT(VD_PLOT,23,PLOTH,-2.5,Y_POSN)
        Y_POSN=Y_POSN-.6
        CALL UIS$TEXT(VD_PLOT,23,PLOTI,-2.5,Y_POSN)
        Y_POSN=Y_POSN-.6
        CALL UIS$TEXT(VD_PLOT,23,PLOTJ,-2.5,Y_POSN)
      CALL LIB$SPAWN('WAIT 00:00:15')
      CALL UIS$SOUND_BELL('SYS$WORKSTATION',4)
      CALL UIS$DELETE_DISPLAY(VD_PLOT)
      RETURN
      END
C
C
C
```

```
      SUBROUTINE PW_FAIL(PW_CAUSE)
C*********************************************************************
C                                                                    *
C     THIS SUBROUTINE POPS UP A DIALOG BOX TO INSTRUCT THE USER THAT *
C     THE POWER PREDICTION ALGORITHM WILL FAIL TO ACURATELY PREDICT  *
C     PROPULSIVE POWER REQUIREMENTS DUE TO THE INPUTED DATA BEING    *
C     OUTSIDE OF THE PARAMETERS ALLOWED BY THE SILVERLEAF & DAWSON   *
C     OR ADMIRALTY FORMULAS.  THE SPECIFIC CAUSE IS PASSED AS THE    *
C     PARAMETER 'PW_CAUSE' TO THE SUBROUTINE FROM THE ERROR CHECKING *
C     LOOPS IN SUBROUTINE PW_CALC_2.  THIS PARAMETER ENABLES THIS    *
C     SUBROUTINE TO BE WRITTEN ONCE FOR ALL CASES AND STILL BE       *
C     SPECIFIC FOR EACH INDIVIDUAL CASE.                             *
C                                                                    *
C     THE ALGORITHMS USED ARE DETAILED IN SUBROUTINE PW_CALC_2       *
C                                                                    *
C     CALLED BY SUBROUTINE 'PW_CALC_2'                               *
C*********************************************************************
C
      IMPLICIT INTEGER(A-Z)
      INCLUDE 'SYS$LIBRARY:UISENTRY'
      INCLUDE 'SYS$LIBRARY:UISUSRDEF'
      INCLUDE 'TOP_POWER.FOR'
      CHARACTER*34 FAILA/'***  POWER PREDICTION FAILURE  ***'/
      CHARACTER*34 FAILB/' TOOL BOX MAY FAIL TO ACCURATELY  '/
      CHARACTER*34 FAILC/' PREDICT POWER REQUIREMENTS FOR   '/
      CHARACTER*34 FAILD/' ADMIRALTY OR S & D METHOD DUE TO '/
      CHARACTER*34 FAILE/'PLEASE WAIT.......................'/
      CHARACTER*34 FAILF/'...............PROGRAM WILL RESUME'/
      REAL Y_POSN
      CHARACTER*36 FAIL(7)
C*****INITIALIZE THE FAILURE MESSAGES ARRAY
      FAIL(1) = ' SHIP LENGTH OUT OF RANGE            '
      FAIL(2) = ' SHIP LENGTH OR NRPM OUT OF RANGE '
      FAIL(3) = ' SPEED/LENGTH RATIO OUT OF RANGE   '
      FAIL(4) = ' SHIP BLOCK COEFFICIENT OUT OF RANGE'
      FAIL(5) = ' BEAM/DRAFT RATIO OUT OF RANGE     '
      FAIL(6) = ' LENGTH/BEAM RATIO OUT OF RANGE    '
      FAIL(7) = ' NUMBER OF SHAFTS INCORRECT (1 OR 2)'
C*****CREATE THE DISPLAY FOR THE DIALOG BOX
      VD_FAIL=UIS$CREATE_DISPLAY(-5.0,-2.0,10.0,4.0,26.0,10.0)
C*****CREATE THE WINDOW TO DISPLAY THE TEXT IN
      WD_FAIL=UIS$CREATE_WINDOW(VD_FAIL,'SYS$WORKSTATION','HELP WINDOW')
C*****COPY ATTRIBUTE BLOCK '0' AS BLOCK '25' AND CHANGE THE FONT SIZE
      CALL UIS$SET_FONT(VD_FAIL,0,25,'DTABER0R03WK00GG0001UZZZZ02A000')
C*****COPY ATTRIBUTE BLOCK '25' AS BLOCK '26' AND CHANGE TO BOLD FONT
      CALL UIS$SET_FONT(VD_FAIL,25,26,'DTABER0R03WK00PG0001UZZZZ02A000')
C*****SIGNAL THE USER TO GET THEIR ATTENTION
      CALL UIS$SOUND_BELL('SYS$WORKSTATION',4)
C*****WRITE THE TEXT INTO THE WINDOW AND SPAWN PROCESS TO 'WAIT' 15 SEC
        Y_POSN=2.8
        CALL UIS$TEXT(VD_FAIL,26,FAILA,-2.5,Y_POSN)
        Y_POSN=Y_POSN-.6
        CALL UIS$TEXT(VD_FAIL,25,FAILB,-2.5,Y_POSN)
        Y_POSN=Y_POSN-.6
        CALL UIS$TEXT(VD_FAIL,25,FAILC,-2.5,Y_POSN)
        Y_POSN=Y_POSN-.6
```

```
          CALL UIS$TEXT(VD_FAIL,25,FAILD,-2.5,Y_POSN)
          Y_POSN=Y_POSN-.6
          CALL UIS$TEXT(VD_FAIL,26,FAIL(PW_CAUSE),-2.5,Y_POSN)
          Y_POSN=Y_POSN-.6
          CALL UIS$TEXT(VD_FAIL,25,FAILE,-2.5,Y_POSN)
          Y_POSN=Y_POSN-.6
          CALL UIS$TEXT(VD_FAIL,25,FAILF,-2.5,Y_POSN)
        CALL LIB$SPAWN('WAIT 00:00:15')
        CALL UIS$SOUND_BELL('SYS$WORKSTATION',4)
        CALL UIS$DELETE_DISPLAY(VD_FAIL)
        RETURN
        END
C
C
C
      SUBROUTINE POWER_GRAPH(TYPE)
C*****************************************************************************
C                                                                            *
C     THIS SUBROUTINE CREATES A CA-DISSPLA METAFILE GRAPH OF THE             *
C     PROPULSIVE POWER CURVE OF THE SHIP DESIGN FROM ZERO (0)                *
C     KNOTS TO THE OPERATING SPEED 'PW_VK' GIVEN AT THE STAGE                *
C     OF DESIGN WHEN THE SUBROUTINE WAS CALLED.  IT IS CALLED BY             *
C     A MOUSE CLICK ON THE MAIN POWER MENU 'GRAPH POWER' SO IT               *
C     IS IMPORTANT THAT THE DESIGN PARAMETERS BE FINALIZED AT                *
C     THAT POINT SINCE ALL THE REQUIRED PARAMETERS WILL BE PASSED            *
C     IN COMMOM BLOCKS.  IT ALSO CALLS THE SUBROUTINE 'SAVE_GRAPH'           *
C     THAT ANNOUNCES THE METAFILE NAME TO THE USER BEFORE IT                 *
C     WRITES THE FILE TO DISK.                                               *
C                                                                            *
C     CALLED BY SUBROUTINE 'POWER'                                           *
C     CALLS SUBROUTINES 'SAVE_GRAPH' AND MANY CA-DISSPLA ROUTINES            *
C*****************************************************************************
        INCLUDE 'TOP.FOR'
        INCLUDE 'TOP_POWER.FOR'
        CHARACTER*23 DATETIME,PWG_NAME
        REAL LM,LF,BF,TF,N,DELTAM,DELTAT,VM,VK(0:15),PHP_AD,VKP,
     &      CB,C_VB,C_X,KBT,H400,KV,ETA0120,DETA0,C_ETA0,VMTK(0:15),
     &      GPW_SD(0:15),GPW_SDB(0:15),KLBP,C_VL,C_BT,C_LB,VMT(0:15),
     &      TOP_SD,TOP_SD1,BOT_SD,NRPM,GPW_ADM(0:15),GPW_ADMP,VMG,
     &      MIN_POWER1,MIN_POWER2,MAX_PWR
       INTEGER I_NSHAFT,INDEX1,INDEX2,I,J,K,L,PLOT_TYPE,TYPE
       INTEGER*4 STATUS
       PLOT_TYPE=TYPE
C*****ZERO OUT ALL MATRICES AND VARIABLES
      DO 5 I=0,15,1
         VK(I)=0.0
         VMTK(I)=0.0
         GPW_SD(I)=0.0
         GPW_SDB(I)=0.0
         VMT(I)=0.0
         GPW_ADM(I)=0.0
 5    CONTINUE
      MIN_POWER1=0.0
      MIN_POWER2=0.0
      INDEX1=0
      INDEX2=0
      MAX_POWER=0.0
C*****RECALCULATE COMMON VALUES FOR ACCURACY
      CALL PW_CALC_1
C*****TELL WORLD TO SIT TIGHT AND WAIT
      CALL UIS$ERASE(VD_ID,-0.4,10.1,3.4,10.9)
      CALL UIS$TEXT(VD_ID,7,'CALCULATING PLOT...',0.0,11.0)
      CALL UIS$TEXT(VD_ID,7,'PLEASE RELAX & WAIT',0.0,10.6)
C*****TRANSFER VALUES VIA COMMON BLOCKS
      PWG_NAME=PW_INFILE
      I_NSHAFT=INT(PW_NSHAFT)
      LF=PW_LPP
      BF=PW_B
      TF=PW_T
      DELTAT=PW_DISP
      VKP=PW_VK
      CB=PW_CB
      NRPM=PW_NRPM
C*****SET UP CA-DISSPLA ENVIRONMENT***********************************
C  'LN03I' SENDS OUTPUT TO A LASER PRINTER CAPABLE FILE; 'PAGE'  *
C  SETS PAGE SIZE AS STANDARD,'BLOWUP' INCREASES PICTURE SIZE;   *
C  'AREA2D' SETS 2D PLOT SIZE; 'TRIPLX' SELECTS THE FONT FILE;   *
C  'X/Y NAME'S SET AXIS NAMES; 'HEADIN' SETS THE GRAPH TITLE;    *
C  'GRAF' SETS AXIS SIZES(ORIGIN,MAX X,Y ,UNITS,SCALE,ETC);      *
C  'THKFRM' DRAWS A THICK LINE AROUND THE SUBPLOT AREA; 'FRAME'  *
C  FRAMES THE ENTIRE PAGE; 'GRID' SELECTS GRAPH GRIDDING:        *
C  'RASPLN' SELECTS TYPE OF CURVE SMOOTHING/INTERPOLATION TO USE;*
C  'HEIGHT' SETS CHARACTER HEIGHTS AS WE WANT THEM TO BE;        *
C*****************************************************************
C*****DETERMINE IF DESIRE PLOT TO DISK (1) OR TO SCREEN (2)
      IF ( PLOT_TYPE .EQ. 1 ) THEN
          CALL PGPX
      ELSEIF ( PLOT_TYPE .EQ. 2 ) THEN
          CALL LN03I
      ELSE
          CONTINUE
      ENDIF
      CALL PAGE(8.5,11.0)
      CALL BLOWUP(1.15)
      CALL AREA2D(6.0,8.0)
      CALL TRIPLX
      CALL HEIGHT(0.200)
      CALL XNAME('SPEED (KTS)$',100)
      CALL YNAME('HORSEPOWER (HP)$',100)
      CALL HEIGHT(0.375)
      CALL HEADIN('SPEED-POWER RELATIONSHIP$',-24,-1.,1)
      CALL HEIGHT(0.200)
      MAX_PWR=MAX(PW_ADME,PW_SDBB,PW_SDNB)
      CALL GRAF(0.0,2.0,PW_VK,0.0,5000.0,(MAX_PWR+2000.0))
      CALL THKFRM(0.04)
      CALL FRAME
      CALL GRID(-2,-1)
      CALL RASPLN(2.0)
C*****SET UP ADMIRALTY POWER ARRAY IN 15 INCREMENTS
      LM=LF/3.28
      DELTAM=DELTAT/1.016
      VM=PW_VK*0.514444/15.0
      N=NRPM/60.0
      DO 10 I=0,15,1
          VMT(I)=I*VM
          VMTK(I)=PW_VK*I/15.0
          GPW_ADMP=5.0*((DELTAM**2.0)**(1.0/3.0))*(VMT(I)**3.0)*
     &              (33.0-(0.017*LM))/(15000.0-110.0*N*LM**0.5)
          GPW_ADM(I)=GPW_ADMP/0.7456999
 10   CONTINUE
C*****ALSO, ACCOUNT FOR CASE THAT ADMIRALTY METHOD HAS FAILED
      IF ( PW_ADME .LT. 1.0 ) THEN
         DO 12 J=1,15,1
            GPW_ADM(J)=0.0
 12      CONTINUE
      ENDIF
C*****GRAPH ADMIRALTY POWER CURVE
      CALL MARKER(16)
      CALL DOT
      CALL CURVE(VMTK,GPW_ADM,16,1)
      CALL HEIGHT(0.070)
      CALL LINES('ADM. POWER',ID,1)
C*****FILL IN S&D ARRAY IN 15 INCREMENTS TO PLOT CURVE
      DO 20 I=0,15,1
        VK(I)=PW_VK*I/15.0
        VMTK(I)=PW_VK*I/15.0
        C_VB=(1.70-1.4*CB)*LF**0.5
        IF ( LF .GT. 1000.0 ) C_X=0.85
        IF ( LF .LE. 1000.0 ) C_X=0.85+0.00185*
     &                          ((1000.0-LF)/100.0)**2.5
        IF ( BF/TF .LT. 2.40 ) KBT=0.982
        IF ( BF/TF .GE. 2.40 ) KBT=0.96+
     &                         0.00054*10.0**((2.0*BF)/(3.0*TF))
        IF ( VK(I)/C_VB .LE. 1.0 ) KV=2.75-7.25*(VK(I)/C_VB)+
     &                          5.50*((VK(I)/C_VB)**2.0)
        IF ( VK(I)/C_VB .GT. 1.0 ) KV=21.2-43.2*(VK(I)/C_VB)+
     &                          23.0*((VK(I)/C_VB)**2.0)
        IF (I_NSHAFT.EQ.1) H400=2.60-0.2917*VK(I)/(DELTAT**(1.0/6.0))
        IF (I_NSHAFT.EQ.2) H400=2.38-0.2917*VK(I)/(DELTAT**(1.0/6.0))
        IF ( LF .EQ. 400.0) KLBP=1.0
        IF ( LF .NE. 400.0) KLBP=0.9196+2.31E-04*LF-7.50E-08*LF**2.0
        IF ( I_NSHAFT .EQ. 1 ) ETA0120=0.98-0.55*CB
        IF ( I_NSHAFT .EQ. 2 ) ETA0120=0.90-0.33*CB
        IF ( I_NSHAFT .EQ. 1 ) DETA0=0.360-0.0029*NRPM
        IF ( I_NSHAFT .EQ. 2 ) DETA0=0.135-0.0011*NPRM
        C_ETA0=ETA0120+DETA0
C
        TOP_SD=C_X*(DELTAT**(2.0/3.0))*(C_VB**3.0)*KBT*KV
        BOT_SD=427.10*C_ETA0*H400*KLBP
C
        GPW_SD(I) =TOP_SD/BOT_SD
        GPW_SDB(I) = 0.9500*(TOP_SD/BOT_SD)
C
 20   CONTINUE
C*****SEARCH THE POWER ARRAYS TO FIND THE MINIMUMS AND THEIR INDICES.
C     THIS HAS TO BE DONE DUE TO THE INVALIDITY OF THE S&D FORMULAS
C     AT SPEEDS << BOUNDARY SPEED.  THIS ERROR CAUSES THE UN-
C     CORRECTED GRAPHS TO START AT HIGH HORSEPOWER AT LOW SPEED
```

```
C     (AN UNREALISTIC ASSUMPTION), DROP TO A MINIMUM (THAT WE FIND),
C     AND THEN GROW ASYMPTOTICALLY TO THE TRUE PREDICTION.  THUS WE
C     WILL SIMPLY FIND THE MINIMUM AND SET ALL LOW SPEED VALUES TO THE
C     LEFT OF THIS POINT EQUAL TO THAT MINIMUM FOR EACH CURVE.
C*****ALSO, ACCOUNT FOR CASE THAT S&D METHOD HAS FAILED
      IF ( PW_SDNB .LT. 1.0 .AND. PW_SDBB .LT. 1.0 ) THEN
         DO 22 J=1,15,1
             GPW_SD(J)=0.0
             GPW_SDB(J)=0.0
 22      CONTINUE
      ENDIF
      MIN_POWER1=GPW_SD(0)
      MIN_POWER2=GPW_SDB(0)
      DO 25 J=1,15,1
         IF ( GPW_SD(J) .LT. MIN_POWER1 ) THEN
            MIN_POWER1=GPW_SD(J)
            INDEX1=J
         ELSE
            CONTINUE
         ENDIF
         IF ( GPW_SDB(J) .LT. MIN_POWER2 ) THEN
            MIN_POWER2=GPW_SDB(J)
            INDEX2=J
         ELSE
            CONTINUE
         ENDIF
 25   CONTINUE
C*****SET ANY POWER TO LEFT OF S&D MINIMUM EQUAL TO MINIMUM
      DO 26 K=INDEX1,0,-1
        GPW_SD(K)=MIN_POWER1
 26   CONTINUE
      DO 27 L=INDEX2,0,-1
        GPW_SDB(L)=MIN_POWER2
 27   CONTINUE
C*****GRAPH BOTH S&D CURVES
      CALL MARKER(15)
      CALL DASH
      CALL CURVE(VMTK,GPW_SD,16,1)
      CALL HEIGHT(0.070)
      CALL LINES('S&D POWER W/O BB',ID,2)
      CALL MARKER(17)
      CALL CHNDOT
      CALL CURVE(VMTK,GPW_SDB,16,1)
      CALL HEIGHT(0.070)
      CALL LINES('S&D POWER W/  BB',ID,3)
      CALL HEIGHT(0.150)
      CALL LEGNAM('***POWER CURVES***$',18)
      CALL LEGEND(ID,3,1.75,6.8)
C*****STAMP FILENAME ON GRAPH SO ORIGIN IS KNOWN
      CALL HEIGHT(0.100)
      CALL MESSAG(PWG_NAME,23,4.5,-0.75)
C*****GET SYSTEM DATE&TIME AND STAMP ON GRAPH
      STATUS=LIB$DATE_TIME(DATETIME)
      CALL HEIGHT(0.100)
      CALL MESSAG(DATETIME,23,4.5,-0.90)
C*****DONE
```

```
      CALL ENDPL(0)
      CALL DONEPL
C*****DETERMINE IF PLOT WENT TO DISK (1) OR TO SCREEN (2)
      IF ( PLOT_TYPE .EQ.  1 ) THEN
          CONTINUE
      ELSEIF ( PLOT_TYPE .EQ.  2 ) THEN
          CALL SAVE_GRAPH
      ELSE
          CONTINUE
      ENDIF
C*****CLEAR THE INSTRUCTION BOX SO THE USERS KNOWS ITS DONE
      CALL UIS$ERASE(VD_ID,-0.4,10.1,3.4,10.9)
 50   RETURN
      END
C
C
C
      SUBROUTINE G_DISK
C*****************************************************************
C                                                                *
C     THIS SUBROUTINE CALLS THE CA-DISSPLA ROUTINE TO PLOT TO    *
C     DISK FILE.                                                 *
C                                                                *
C     CALLED BY SUBROUTINE MAIN MENU 'POWER'                     *
C     CALLS SUBROUTINE 'POWER_GRAPH'                             *
C*****************************************************************
      INCLUDE 'TOP.FOR'
      TYPE = 1
      CALL POWER_GRAPH(TYPE)
      RETURN
      END
C
C
C
      SUBROUTINE G_SCREEN
C*****************************************************************
C                                                                *
C     THIS SUBROUTINE CALLS THE CA-DISSPLA ROUTINE TO PLOT TO    *
C     SCREEN.                                                    *
C                                                                *
C     CALLED BY SUBROUTINE MAIN MENU 'POWER'                     *
C     CALLS SUBROUTINE 'POWER_GRAPH'                             *
C*****************************************************************
      INCLUDE 'TOP.FOR'
      TYPE = 2
      CALL POWER_GRAPH(TYPE)
      RETURN
      END
C
C
C
C
```

B. SUBROUTINE POWER VARIABLE DECLARATIONS FILE

This file is the INCLUDE File TOP_POWER.FOR that is common to every subroutine in the Power Prediction module and Power Prediction Report subroutine. This commonality is made to ensure that all data is available to every subroutine without needing to pass every parameter as an argument.

```
*****THE DECLARATIONS FOR THE PROPULSIVE POWER SECTION IN ONE FILE
C
      CHARACTER PW_LINE*12,PW_INFILE*12
      CHARACTER PW_KB_UP(20)*12,PW_FNAME*12

      INTEGER PW_TIMES,PW_CAUSE,I_PW_NSHAFT,PLINE_NO

      REAL PW_LPP,PW_T,PW_B,PW_DISP,PW_CB,PW_VK
      REAL PW_NRPM,PW_SDNB,PW_SDBB,PW_NSHAFT
      REAL PW_VB,PW_CX,C_KBT,C_KV,C_H400,C_KLBP
      REAL C_ETA_0120,C_DETA_0,PW_ETA_0,PW_TEMP
      REAL PW_ADME,PW_ADM,PW_SLR,PW_BTR,PW_LBR
      REAL PW_NRPS,PW_LPPM,PW_VKM,PW_DISPM
      REAL TOP_ANS,BOT_ANS

      COMMON /PW_NAMES/ PW_LPP,PW_T,PW_B,PW_DISP,PW_CB,PW_VK
      COMMON /PW_VARS/ PW_NRPM,PW_NSHAFT,PW_TIMES,PW_KB_UP
      COMMON /PW_COEFF/ PW_VB,PW_CX,C_KBT,C_KV,C_H400,C_KLBP
      COMMON /COEFFS2/ C_ETA_0120,C_DETA_0,PW_ETA_0,PW_TEMP
      COMMON /PW_RESULTS/ PW_SDNB,PW_SDBB,PW_ADME,PW_ADM
      COMMON /PW_METRIC/ PW_NRPS,PW_LPPM,PW_VKM,PW_DISPM
      COMMON /PW_FILES/ PW_FNAME,PW_INFILE
```

APPENDIX C. POWER PREDICTION TEST ROUTINE

A. SOURCE CODE

This code was used to verify and correct the Silverleaf and Dawson prediction formulas. It also makes an excellent 'stand- alone' program for general power calculations outside of the *TOOL BOX* environment.

```
      PROGRAM TRIAL_POWER
C*********************************************************************
C                                                                    *
C     THIS PROGRAM TESTS THE SILVERLEAF & DAWSON PROPULSIVE POWER    *
C     PREDICTION FORMULA FOR SOME REALISTIC DESIGNS WITH KNOWN       *
C     PROPULSION REQUIREMENTS.  THE ADMIRALTY PREDICTION IS ALSO     *
C     CALCULATED FOR COMPARISON AND THE DATA IS PRESENTED IN         *
C     TABLE 1 OF CHAPTER 3.                                          *
C                                                                    *
C     REQUIRED INPUTS ARE AS FOLLOWS:                                *
C        SHIP LENGTH(LPP) IN FEET                                    *
C        SHIP BEAM(B) IN FEET                                        *
C        SHIP DRAFT(T) IN FEET                                       *
C        SHIP DISPLACEMENT(DELTA) IN TONS STD SEAWATER               *
C        NUMBER OF PROPULSION SHAFTS ( 1 OR 2 )                      *
C        PROPELLER BLADE SPEED(N) IN REV/SEC                         *
C        SHIP SPEED(V) IN KNOTS                                      *
C                                                                    *
C     THIS PROGRAM IS WRITTEN IN VAX FORTRAN AND WILL ALSO COMPILE   *
C     UNDER LAHEY FORTRAN LP77 V3.0 AND MICROSOFT FORTRAN 4.01 FOR   *
C     MS-DOS BASED SYSTEMS WITH A CHANGE TO THE OUTPUT FILENAME TO   *
C     ADHERE TO MS-DOS CONVENTIONS.                                  *
C                                                                    *
C*********************************************************************
C
      REAL LM,LF,BF,TF,N,DELTAM,DELTAT,VM,VK,PHP_AD,PKW,NRPM,
     &     CB,C_VB,C_X,C_KBT,C_KV,C_H400,ETA0120,DETA0,C_ETA0,
     &     PHP_SD,C_KLBP,PHP1SD,C_SL,C_BT,C_LB,
     &     TOP_SD,BOT_SD
      INTEGER NSHAFT,IANSWER
C*****OPEN AN OUTPUT FILE
      OPEN (UNIT=8,FILE='TRIAL_POWER.TXT',STATUS='UNKNOWN')
 1    CONTINUE
C*****INITIALIZE ( RE-INITIALIZE ) 'CORRECTING' VARIABLES
      VK=0.0
      C_KV=0.0
C
C*****QUERY FOR REQUIRED DATA
C
      PRINT *,'ENTER SHIP LENGTH IN FEET : '
      READ *,LF
      PRINT *,'ENTER SHIP BEAM IN FEET : '
      READ *,BF
      PRINT *,'ENTER SHIP DRAFT IN FEET : '
      READ *,TF
      PRINT *,'ENTER DISPLACEMENT IN TONS : '
      READ *,DELTAT
      PRINT *,'ENTER NUMBER OF SHAFTS AS AN INTEGER : '
      READ *,NSHAFT
      PRINT *,'ENTER PROPELLER SPEED IN REV/SEC : '
      READ *,N
      PRINT *,'ENTER SHIP SPEED IN KNOTS : '
      READ *,VK
      LM=LF/3.28
      DELTAM=DELTAT/1.016
      VM=VK*0.514444
C
C*****ERROR CHECK LENGTH(M),N*SQRT[LENGTH(M)] FOR ADMIRALTY METHOD
C
      IF ( LM .GT. 1162.196 ) THEN
          PRINT *,'******LENGTH OUT OF RANGE(ADM)*****************'
          GOTO 13
      ELSE
          CONTINUE
      ENDIF
      IF ( N*(LM**0.5) .GT. 136.364 ) THEN
          PRINT *,'******PRODUCT N*L(1/2) OUT OF RANGE(ADM)*********'
          GOTO 13
      ELSE
          CONTINUE
      ENDIF
C
C*****CALCULATE POWER BY ADMIRALTY METHOD
C
      PRINT *,'......STARTING ADMIRALTY CALCULATION'
C
      PKW=5.0*((DELTAM**2.)**(1.0/3.0))*(VM**3.0)*(33.0-(0.017*LM)
     &     )/(15000.0-110.0*N*LM**0.5)
C*****TRANSLATE TO HORSEPOWER FROM KILOWATTS
      PHP_AD=PKW/0.7456999
C*****OUTPUT TO FILE AND SCREEN
      WRITE (8,7)
      WRITE (8,28)
      WRITE (8,8)
      WRITE (8,9) PHP_AD
      WRITE (8,7)
      WRITE (8,91)
      WRITE (8,92)
      WRITE (8,10) LM,DELTAM
      WRITE (8,11)
      WRITE (8,12) VM,N
      PRINT *,'......POWER IN HP BY ADMIRALTY METHOD = ',PHP_AD
C
 13   WRITE (8,7)
C
C*****ERROR CHECK BLOCK COEFFICIENT,SPEED LENGTH RATIO,BEAM/DRAFT RATIO
C     FOR SILVERLEAF AND DAWSON METHOD
C
      NRPM=N*60.0
      CB=35.0*DELTAT/(LF*BF*TF)
      C_SL=VK/LF**0.5
      C_BT=BF/TF
      C_LB=LF/BF
C
      IF ( NSHAFT .EQ. 1 ) THEN
         IF ( C_SL .GT. 1.20 .OR. C_SL .LT. 0.40 ) THEN
             PRINT *,'*******VK/SQRT(L) RATIO OUT OF RANGE(1)********'
             GOTO 30
         ELSE
             CONTINUE
         ENDIF
         IF ( CB .GT. 0.86 .OR. CB .LT. 0.50 ) THEN
             PRINT *,'*******BLOCK COEFFICIENT OUT OF RANGE(1)*******'
             GOTO 30
         ELSE
             CONTINUE
         ENDIF
         IF ( C_BT .GT. 4.50 .OR. C_BT .LT. 2.0 ) THEN
             PRINT *,'********BEAM/DRAFT RATIO OUT OF RANGE(1)*******'
             GOTO 30
         ELSE
             CONTINUE
         ENDIF
         IF ( C_LB .GT. 9.50 .OR. C_LB .LT. 3.33 ) THEN
             PRINT *,'********LENGTH/BEAM RATIO OUT OF BOUNDS(1)*****'
             GOTO 30
         ELSE
             CONTINUE
         ENDIF
      ELSEIF ( NSHAFT .EQ. 2 ) THEN
         IF ( C_SL .GT. 1.20 .OR. C_SL .LT. 0.40 ) THEN
             PRINT *,'*******VK/SQRT(L) RATIO OUT OF RANGE(2)********'
             GOTO 30
         ELSE
             CONTINUE
         ENDIF
         IF ( CB .GT. 0.80 .OR. CB .LT. 0.54 ) THEN
             PRINT *,'*******BLOCK COEFFICIENT OUT OF RANGE(2)*******'
             GOTO 30
         ELSE
             CONTINUE
         ENDIF
         IF ( C_BT .GT. 4.50 .OR. C_BT .LT. 2.0 ) THEN
             PRINT *,'********BEAM/DRAFT RATIO OUT OF RANGE(2)*******'
             GOTO 30
         ELSE
             CONTINUE
         ENDIF
         IF ( C_LB .GT. 11.50 .OR. C_LB .LT. 3.80 ) THEN
             PRINT *,'********LENGTH/BEAM RATIO OUT OF RANGE(2)******'
             GOTO 30
         ELSE
             CONTINUE
         ENDIF
      ELSE
         PRINT *,'**********NUMBER OF SHAFTS OUT OF RANGE(1,2)*******'
         GOTO 30
      ENDIF
C
      WRITE (8,141)
      PRINT *,'......BLOCK COEFFICIENT = ',CB
      WRITE (8,15) CB
      PRINT *,'......SPEED-LENGTH RATIO = ',C_SL
      WRITE (8,16) C_SL
      PRINT *,'......BEAM-DRAFT RATIO = ',C_BT
      WRITE (8,17) C_BT
      PRINT *,'......LENGTH-BEAM RATIO = ',C_LB
      WRITE (8,18) C_LB
      WRITE (8,141)
C
      WRITE (8,7)
C
C*****NOW CALCULATE BY SILVERLEAF & DAWSON METHOD
C
      PRINT *,'......STARTING S & D CALCULATION'
      WRITE (8,14)
C
      C_VB=(1.70-1.4*CB)*LF**0.5
      IF ( LF .GT.  1000.0 ) C_X=0.85
      IF ( LF .LE.  1000.0 ) C_X=0.85+0.00185*((1000.0-LF)/100.0)**2.5
      IF ( BF/TF .LT.  2.40 ) C_KBT=0.982
      IF ( BF/TF .GE.  2.40 ) C_KBT=0.96+
     &                        0.00054*10.0**((2.0*BF)/(3.0*TF))
      IF ( VK/C_VB .LE.  1.0 ) C_KV=2.75-7.25*(VK/C_VB)+
     &                         5.50*((VK/C_VB)**2.0)
      IF ( VK/C_VB .GT.  1.0 ) C_KV=21.2-43.2*(VK/C_VB)+
     &                         23.0*((VK/C_VB)**2.0)
      IF ( NSHAFT .EQ.  1 ) C_H400=2.60-0.2917*VK/(DELTAT**(1.0/6.0))
      IF ( NSHAFT .EQ.  2 ) C_H400=2.38-0.2917*VK/(DELTAT**(1.0/6.0))
      IF ( LF .EQ.  400.0 ) C_KLBP=1.0
      IF ( LF .NE.  400.0 ) C_KLBP=0.9196+2.31E-04*LF-7.50E-08*LF**2.0
      IF ( NSHAFT .EQ.  1 ) ETA0120=0.98-0.55*CB
      IF ( NSHAFT .EQ.  2 ) ETA0120=0.90-0.33*CB
      IF ( NSHAFT .EQ.  1 ) DETA0=0.360-0.0029*NRPM
      IF ( NSHAFT .EQ.  2 ) DETA0=0.135-0.0011*NRPM
      C_ETA0=ETA0120+DETA0
C
C*****CALCULATE S & D POWER NUMERATOR & DENOMINATOR FOR
C     NORMAL AND 'CORRECTED' VERSIONS
C
      TOP_SD=C_X*(DELTAT**(2.0/3.0))*(C_VB**3.0)*C_KBT*C_KV
      BOT_SD=427.10*C_ETA0*C_H400*C_KLBP
C*****VALUE
      PHP_SD =TOP_SD/BOT_SD
C*****OUTPUT VALUES
      PRINT *,'......POWER IN HP BY S&D METHOD FOR NO B/B  = ',PHP_SD
      WRITE (8,19) PHP_SD
C*****CORRECT IF WELL-DESIGNED BULBOUS BOW IS PRESENT & OUTPUT
      PHP1SD = 0.95*PHP_SD
      PRINT *,'......POWER IN HP BY S&D METHOD FOR B/B     = ',PHP1SD
      WRITE (8,20) PHP1SD
      WRITE (8,7)
      WRITE (8,91)
      WRITE (8,23)
      WRITE (8,24) LF,BF,TF
      WRITE (8,25)
      WRITE (8,26) DELTAT,N,NRPM
      WRITE (8,27)
      WRITE (8,271) VK,C_VB,NSHAFT
      WRITE (8,28)
      WRITE (8,28)
C
C*****SET 'BEGIN AGAIN' CONDITIONS TO EXIT OR CONTINUE
C
 29   PRINT *,' '
 30   CONTINUE
      PRINT *,' DO YOU WANT TO PROCESS AGAIN(??):    INSTRUCTIONS:'
      PRINT *,' ENTER ZERO (0) TO TERMINATE OR ONE (1) TO PROCEED:'
      PRINT *,' RESPONSE: (???) : '
      READ *,IANSWER
      IF ( IANSWER .EQ. 0 ) THEN
          CLOSE (UNIT=8)
          GOTO 31
      ELSEIF ( IANSWER .EQ. 1 ) THEN
          GOTO 1
      ELSE
          GOTO 29
      ENDIF
C
C*****FORMAT STATEMENTS
C
  7   FORMAT(2X,' ')
  8   FORMAT (2X,'ADMIRALTY CALCULATION')
  9   FORMAT (2X,'POWER IN HORSEPOWER FOR ADMIRALTY METHOD....',F12.2)
  91  FORMAT (2X,'SHIP DESIGN CHARACTERISTICS')
  92  FORMAT (10X,'LENGTH(M)...........DISPL(MTONS)')
 10   FORMAT (2X,2F19.2)
 11   FORMAT (10X,'SPEED(M/S)..........PROP SPD(RPS)')
 12   FORMAT (2X,2F19.2)
 141  FORMAT(2X,'            ******************')
 15   FORMAT (2X,'BLOCK COEFFICIENT.......',F15.6)
 16   FORMAT (2X,'SPEED LENGTH RATIO......',F15.6)
 17   FORMAT (2X,'BEAM/DRAFT RATIO........',F15.6)
 18   FORMAT (2X,'LENGTH/BEAM RATIO.......',F15.6)
 14   FORMAT (2X,'SILVERLEAF & DAWSON CALCULATIONS')
 19   FORMAT (2X,'POWER IN HP BY S&D METHOD FOR NO B/B......',F15.2)
 20   FORMAT (2X,'POWER IN HP BY S&D METHOD FOR WD B/B......',F15.2)
 23   FORMAT (10X,'LENGTH(FT)...........BEAM(FT)...........DRAFT(FT)')
 24   FORMAT (2X,3F19.2)
 25   FORMAT (10X,'DISPL(TONS).........SHAFT RPS...........SHAFT RPM')
 26   FORMAT (2X,3F19.2)
 27   FORMAT (10X,'DESIGN SPD(KTS)....BOUN SPD(KTS).........# SHAFTS')
 271  FORMAT (2X,2F19.2,13X,I3)
 28   FORMAT (2X,'*****************************************************')
C
 31   CONTINUE
      STOP
      END
```

B. TEST ROUTINE OUTPUT

This output was used to verify the algorithms and is used in part in Table 1 of Chapter 3.

- LARGE GENERAL CARGO Design

```
**************************************************
ADMIRALTY CALCULATION
POWER IN HORSEPOWER FOR ADMIRALTY METHOD....    20351.91

SHIP DESIGN CHARACTERISTICS
        LENGTH(M)...........DISPL(MTONS)
             177.59            31491.14
        SPEED(M/S)..........PROP SPD(RPS)
             10.70                 2.00

            *****************
BLOCK COEFFICIENT.......       0.669842
SPEED LENGTH RATIO......       0.861818
BEAM/DRAFT RATIO........       2.342857
LENGTH/BEAM RATIO.......       7.103659
            *****************

SILVERLEAF & DAWSON CALCULATIONS
POWER IN HP BY S&D METHOD FOR NO B/B......       23780.55
POWER IN HP BY S&D METHOD FOR WD B/B......       22591.52

SHIP DESIGN CHARACTERISTICS
        LENGTH(FT)...........BEAM(FT)...........DRAFT(FT)
             582.50              82.00              35.00
        DISPL(TONS).........SHAFT RPS...........SHAFT RPM
           31995.00               2.00             120.00
        DESIGN SPD(KTS)....BOUN SPD(KTS).........# SHAFTS
             20.80              18.40                1
**************************************************
**************************************************
```

- CONTAINER A Design

```
**************************************************
ADMIRALTY CALCULATION
POWER IN HORSEPOWER FOR ADMIRALTY METHOD....    14127.90

SHIP DESIGN CHARACTERISTICS
        LENGTH(M)...........DISPL(MTONS)
             176.83            21732.28
        SPEED(M/S)..........PROP SPD(RPS)
             10.29                 2.00

            *****************
```

```
BLOCK COEFFICIENT.......         0.542293
SPEED LENGTH RATIO......         0.830455
BEAM/DRAFT RATIO........         2.476191
LENGTH/BEAM RATIO.......         7.435897
            ********************

SILVERLEAF & DAWSON CALCULATIONS
POWER IN HP BY S&D METHOD FOR NO B/B......          11503.22
POWER IN HP BY S&D METHOD FOR WD B/B......          10928.06

SHIP DESIGN CHARACTERISTICS
        LENGTH(FT)...........BEAM(FT)...........DRAFT(FT)
             580.00              78.00                31.50
        DISPL(TONS).........SHAFT RPS...........SHAFT RPM
           22080.00                2.00              120.00
        DESIGN SPD(KTS)....BOUN SPD(KTS).........# SHAFTS
              20.00              22.66                 1
**************************************************
**************************************************
```

- CONTAINER B Design

```
**************************************************
ADMIRALTY CALCULATION
POWER IN HORSEPOWER FOR ADMIRALTY METHOD....     30511.80

SHIP DESIGN CHARACTERISTICS
        LENGTH(M)...........DISPL(MTONS)
             206.40            38090.55
        SPEED(M/S)..........PROP SPD(RPS)
              11.73                2.00

            ********************
BLOCK COEFFICIENT.......         0.619424
SPEED LENGTH RATIO......         0.876275
BEAM/DRAFT RATIO........         2.794118
LENGTH/BEAM RATIO.......         7.126316
            ********************

SILVERLEAF & DAWSON CALCULATIONS
POWER IN HP BY S&D METHOD FOR NO B/B......          29480.57
POWER IN HP BY S&D METHOD FOR WD B/B......          28006.54

SHIP DESIGN CHARACTERISTICS
        LENGTH(FT)...........BEAM(FT)...........DRAFT(FT)
             677.00              95.00                34.00
        DISPL(TONS).........SHAFT RPS...........SHAFT RPM
           38700.00                2.00              120.00
        DESIGN SPD(KTS)....BOUN SPD(KTS).........# SHAFTS
              22.80              21.67                 1
**************************************************
**************************************************
```

- ROLL ON/ROLL OFF Design

```
**************************************************
ADMIRALTY CALCULATION
POWER IN HORSEPOWER FOR ADMIRALTY METHOD....    28573.58

SHIP DESIGN CHARACTERISTICS
        LENGTH(M)...........DISPL(MTONS)
             195.12           33233.27
        SPEED(M/S)..........PROP SPD(RPS)
              11.83               2.00

                 *****************
BLOCK COEFFICIENT.......       0.565724
SPEED LENGTH RATIO......       0.909155
BEAM/DRAFT RATIO........       3.187500
LENGTH/BEAM RATIO.......       6.274510
                 *****************

SILVERLEAF & DAWSON CALCULATIONS
POWER IN HP BY S&D METHOD FOR NO B/B......        27435.27
POWER IN HP BY S&D METHOD FOR WD B/B......        26063.51

SHIP DESIGN CHARACTERISTICS
        LENGTH(FT)...........BEAM(FT)...........DRAFT(FT)
             640.00             102.00               32.00
        DISPL(TONS).........SHAFT RPS...........SHAFT RPM
           33765.00               2.00              120.00
        DESIGN SPD(KTS)....BOUN SPD(KTS)..........# SHAFTS
              23.00              22.97                 1
**************************************************
**************************************************
```

- LASH BARGE CARRIER Design

```
**************************************************
ADMIRALTY CALCULATION
POWER IN HORSEPOWER FOR ADMIRALTY METHOD....    26203.87

SHIP DESIGN CHARACTERISTICS
        LENGTH(M)...........DISPL(MTONS)
             220.73           32135.83
        SPEED(M/S)..........PROP SPD(RPS)
              11.57               2.00

                 *****************
BLOCK COEFFICIENT.......       0.563709
SPEED LENGTH RATIO......       0.836206
BEAM/DRAFT RATIO........       3.571429
LENGTH/BEAM RATIO.......       7.240000
                 *****************

SILVERLEAF & DAWSON CALCULATIONS
POWER IN HP BY S&D METHOD FOR NO B/B......        23787.66
POWER IN HP BY S&D METHOD FOR WD B/B......        22598.28

SHIP DESIGN CHARACTERISTICS
```

```
                LENGTH(FT)...........BEAM(FT)...........DRAFT(FT)
                     724.00             100.00              28.00
                DISPL(TONS).........SHAFT RPS...........SHAFT RPM
                  32650.00               2.00              120.00
                DESIGN SPD(KTS)....BOUN SPD(KTS).........# SHAFTS
                      22.50             24.51                1
        **************************************************
        **************************************************
```

- SEABEE BARGE CARRIER Design

```
        **************************************************
        ADMIRALTY CALCULATION
        POWER IN HORSEPOWER FOR ADMIRALTY METHOD....    26772.95

        SHIP DESIGN CHARACTERISTICS
                LENGTH(M)...........DISPL(MTONS)
                     220.34           56387.79
                SPEED(M/S)..........PROP SPD(RPS)
                      10.29               2.00

                      *****************
        BLOCK COEFFICIENT.......       0.670064
        SPEED LENGTH RATIO......       0.743962
        BEAM/DRAFT RATIO........       2.708440
        LENGTH/BEAM RATIO.......       6.824363
                      *****************

        SILVERLEAF & DAWSON CALCULATIONS
        POWER IN HP BY S&D METHOD FOR NO B/B......      21918.27
        POWER IN HP BY S&D METHOD FOR WD B/B......      20822.36

        SHIP DESIGN CHARACTERISTICS
                LENGTH(FT)...........BEAM(FT)...........DRAFT(FT)
                     722.70             105.90              39.10
                DISPL(TONS).........SHAFT RPS...........SHAFT RPM
                  57290.00               2.00              120.00
                DESIGN SPD(KTS)....BOUN SPD(KTS).........# SHAFTS
                      20.00             20.48                1
        **************************************************
        **************************************************
```

- TANKER A Design

```
        **************************************************
        ADMIRALTY CALCULATION
        POWER IN HORSEPOWER FOR ADMIRALTY METHOD....    12045.93

        SHIP DESIGN CHARACTERISTICS
                LENGTH(M)...........DISPL(MTONS)
                     201.22           46536.41
                SPEED(M/S)..........PROP SPD(RPS)
                       8.23               2.00

                      *****************
```

```
BLOCK COEFFICIENT.......        0.795976
SPEED LENGTH RATIO......        0.622799
BEAM/DRAFT RATIO........        2.571429
LENGTH/BEAM RATIO.......        7.333333
            *****************

SILVERLEAF & DAWSON CALCULATIONS
POWER IN HP BY S&D METHOD FOR NO B/B......        11077.69
POWER IN HP BY S&D METHOD FOR WD B/B......        10523.80

SHIP DESIGN CHARACTERISTICS
        LENGTH(FT)...........BEAM(FT)...........DRAFT(FT)
            660.00              90.00              35.00
        DISPL(TONS).........SHAFT RPS...........SHAFT RPM
           47281.00              2.00             120.00
        DESIGN SPD(KTS)....BOUN SPD(KTS).........# SHAFTS
             16.00              15.05               1
**************************************************
**************************************************
```

- TANKER B Design

```
**************************************************
ADMIRALTY CALCULATION
POWER IN HORSEPOWER FOR ADMIRALTY METHOD....     53068.73

SHIP DESIGN CHARACTERISTICS
        LENGTH(M)...........DISPL(MTONS)
            348.48          443809.00
        SPEED(M/S)..........PROP SPD(RPS)
              8.18              2.00

            *****************
BLOCK COEFFICIENT.......        0.818361
SPEED LENGTH RATIO......        0.470299
BEAM/DRAFT RATIO........        3.081081
LENGTH/BEAM RATIO.......        5.013158
            *****************

SILVERLEAF & DAWSON CALCULATIONS
POWER IN HP BY S&D METHOD FOR NO B/B......        36030.91
POWER IN HP BY S&D METHOD FOR WD B/B......        34229.36

SHIP DESIGN CHARACTERISTICS
        LENGTH(FT)...........BEAM(FT)...........DRAFT(FT)
           1143.00             228.00              74.00
        DISPL(TONS).........SHAFT RPS...........SHAFT RPM
          450910.00              2.00             120.00
        DESIGN SPD(KTS)....BOUN SPD(KTS).........# SHAFTS
             15.90              18.74               1
**************************************************
**************************************************
```

- LNG TANKER Design

```
*****************************************************************
ADMIRALTY CALCULATION
POWER IN HORSEPOWER FOR ADMIRALTY METHOD....     39727.77

SHIP DESIGN CHARACTERISTICS
        LENGTH(M)...........DISPL(MTONS)
             273.48          93110.23
        SPEED(M/S)..........PROP SPD(RPS)
              10.49                2.00

            *********************
BLOCK COEFFICIENT.......        0.717015
SPEED LENGTH RATIO......        0.681136
BEAM/DRAFT RATIO........        3.972222
LENGTH/BEAM RATIO.......        6.272727
            *********************

SILVERLEAF & DAWSON CALCULATIONS
POWER IN HP BY S&D METHOD FOR NO B/B......        37910.40
POWER IN HP BY S&D METHOD FOR WD B/B......        36014.88

SHIP DESIGN CHARACTERISTICS
        LENGTH(FT)...........BEAM(FT)...........DRAFT(FT)
             897.00           143.00                36.00
        DISPL(TONS).........SHAFT RPS...........SHAFT RPM
           94600.00             2.00               120.00
        DESIGN SPD(KTS)....BOUN SPD(KTS).........# SHAFTS
              20.40            20.85                 1
*****************************************************************
*****************************************************************
```

- BULK CARRIER Design

```
*****************************************************************
ADMIRALTY CALCULATION
POWER IN HORSEPOWER FOR ADMIRALTY METHOD....     10940.75

SHIP DESIGN CHARACTERISTICS
        LENGTH(M)...........DISPL(MTONS)
             178.05          31594.49
        SPEED(M/S)..........PROP SPD(RPS)
               8.69                2.00

            *********************
BLOCK COEFFICIENT.......        0.645051
SPEED LENGTH RATIO......        0.699327
BEAM/DRAFT RATIO........        2.912500
LENGTH/BEAM RATIO.......        6.266095
            *********************

SILVERLEAF & DAWSON CALCULATIONS
POWER IN HP BY S&D METHOD FOR NO B/B......         8578.57
POWER IN HP BY S&D METHOD FOR WD B/B......         8149.64

SHIP DESIGN CHARACTERISTICS
```

```
        LENGTH(FT)...........BEAM(FT)...........DRAFT(FT)
             584.00              93.20                32.00
        DISPL(TONS).........SHAFT RPS...........SHAFT RPM
          32100.00               2.00               120.00
        DESIGN SPD(KTS)....BOUN SPD(KTS).........# SHAFTS
              16.90              19.26                 1
**************************************************
**************************************************
```

▪ ORE/BULK/OIL CARRIER Design

```
**************************************************
ADMIRALTY CALCULATION
POWER IN HORSEPOWER FOR ADMIRALTY METHOD....     21700.58

SHIP DESIGN CHARACTERISTICS
        LENGTH(M)...........DISPL(MTONS)
             260.67          97647.63
        SPEED(M/S)..........PROP SPD(RPS)
               8.49              2.00

                ********************
BLOCK COEFFICIENT.......        0.838120
SPEED LENGTH RATIO......        0.564288
BEAM/DRAFT RATIO........        2.310044
LENGTH/BEAM RATIO.......        8.081285
                ********************

SILVERLEAF & DAWSON CALCULATIONS
POWER IN HP BY S&D METHOD FOR NO B/B......        18981.25
POWER IN HP BY S&D METHOD FOR WD B/B......        18032.19

SHIP DESIGN CHARACTERISTICS
        LENGTH(FT)...........BEAM(FT)...........DRAFT(FT)
             855.00             105.80                45.80
        DISPL(TONS).........SHAFT RPS...........SHAFT RPM
          99210.00               2.00               120.00
        DESIGN SPD(KTS)....BOUN SPD(KTS).........# SHAFTS
              16.50              15.40                 1
**************************************************
**************************************************
```

APPENDIX D. POWER PREDICTION REPORT MODULE SOURCE CODE

This is the source code for the subroutines that were added to the Report Module to produce the Power Prediction Reports.

```
      SUBROUTINE POWER_RPT
C***********************************************************************
C                                                                      *
C     POWER REPORT SECTION READS IN A DATA FILE CREATED BY THE         *
C     POWER PREDICTION MODULE OF THE MAIN PROGRAM AND WRITES A         *
C     DETAILED HARD COPY REPORT OF THE RESULTS.  THE FIRST PART OF     *
C     THE REPORT ECHOS THE INPUT DATA AND THE CALCULATES THE           *
C     RESULTS FOR THE REPORT.                                          *
C                                                                      *
C     CALLED BY SUBROUTINE 'REPORT'                                    *
C     CALLS SUBROUTINE 'PW_CALC_2'                                     *
C***********************************************************************
      INCLUDE 'TOP.FOR'
      INCLUDE 'TOP_POWER.FOR'
      CHARACTER PW_INPUT_DATA*12, PW_PRFILE*12
C*****MOVE THE MOUSE POINTER AWAY FROM THE SELECTION
      STA = UIS$SET_POINTER_POSITION(VD_ID,WD_RPT,9.9,23.5)
C*****SET UP THE FIRST SCREEN INSTRUCTIONS
      CALL UIS$ERASE(VD_ID, 4.1,20.1,8.4,24.9)
      CALL UIS$TEXT(VD_ID,7,'ENTER THE FILE NAME',4.5,24.)
      CALL UIS$TEXT(VD_ID,7,'AND EXTENSION OF THE',4.5,23.6)
      CALL UIS$TEXT(VD_ID,7,'DATA FILE TO BE USED',4.5,23.2)
      CALL UIS$TEXT(VD_ID,7,'OR [RETURN] TO EXIT',4.5,22.8)
C*****READ THE DATA INPUT FILE NAME
      CALL KEY_READ (PW_INPUT_DATA,.FALSE.,5.,21.2,*30)
C*****NOW WE NEED THE NAME OF THE OUTPUT REPORT FILE
      CALL UIS$ERASE(VD_ID, 4.1,20.1,8.4,24.9)
      CALL UIS$TEXT(VD_ID,7,' ENTER THE FILE NAME',4.5,24.)
      CALL UIS$TEXT(VD_ID,7,' AND EXTENSION TO BE ',4.5,23.6)
      CALL UIS$TEXT(VD_ID,7,'ASSIGNED TO THE REPORT',4.5,23.2)
      CALL UIS$TEXT(VD_ID,7,' OR [RETURN] TO EXIT',4.5,22.8)
      CALL KEY_READ (PW_PRFILE,.FALSE.,5.,21.2,*30)
C*****READ THE DATA IN
 1    OPEN (21,FILE = PW_INPUT_DATA, STATUS = 'OLD')
      OPEN (22,FILE = PW_PRFILE, STATUS = 'NEW',BLANK='NULL')
      READ (21,100) PW_INFILE
      READ (21,110) PW_LPP
      READ (21,110) PW_T
      READ (21,110) PW_B
      READ (21,110) PW_DISP
      READ (21,110) PW_CB
      READ (21,110) PW_VK
      READ (21,110) PW_NRPM
      READ (21,110) PW_NSHAFT
      READ (21,110) PW_ADME
      READ (21,110) PW_SDNB
      READ (21,110) PW_SDBB
      REWIND(21)
      CLOSE (21)
C*****RE-CALCULATE FOR ACCURACY
      CALL PW_CALC_2
      PW_SLR = PW_VK/(PW_LPP**0.5)
      PW_BTR = PW_B/PW_T
      PW_LBR = PW_LPP/PW_B
C*****GENERATE THE COVER PAGE OF THE REPORT
      DO 5 I = 1,18,1
         WRITE (22,130) ' '
 5    CONTINUE
      WRITE (22,120) '                              TOOL BOX '
      WRITE (22,120) ' '
      WRITE (22,120) '                      POWER PREDICTION
     & REPORT'
      WRITE (22,120) '                      ***************************'
      WRITE (22,120) '             THIS REPORT WAS GENERATED USING THE
     &PROGRAM'
      WRITE (22,120) ' '
      WRITE (22,120) '              TOOL BOX WHICH WAS DEVELOPED FOR THE
     &NAVAL  '
      WRITE (22,120) '              --------'
      WRITE (22,120) '                    ENGINEERING DEPARTMENT OF
     &THE  '
      WRITE (22,120) ' '
      WRITE (22,120) '                      NAVAL POSTGRADUATE SCHOOL'
      WRITE (22,120) ' '
      WRITE (22,120) '                         MONTEREY, CALIFORNIA'
      WRITE (22,120) ' '
      WRITE (22,120) ' '
      WRITE (22,120) '                        PROFESSOR F. PAPOULIAS '
      WRITE (22,120) ' '
      WRITE (22,120) '                                AND  '
      WRITE (22,120) ' '
      WRITE (22,120) '                          LT. GERALD MCGOWAN '
      WRITE (22,120) '                          LT. JAMES PLOSAY'
      WRITE (22,120) '                               1989/90 '
      WRITE (22,120) ' '
      WRITE (22,120) ' '
      WRITE (22,120) ' '
      DO 6 I = 1,13,1
         WRITE (22,120) ' '
 6    CONTINUE
      WRITE (22,120) ' '
      WRITE (22,120) ' PAGE 1 OF 2 / POWER PREDICTION'
      WRITE (22,120) ' '
C*****IN THIS SECTION, THE ECHO OF SHIP PARAMETERS AND CALCULATED
C     DATA IS PRESENTED.
C
      DO 10 I = 1,5,1
         WRITE (22,130) ' '
 10   CONTINUE
 WRITE(22,120) '                  POWER PREDICTION REPORT '
 WRITE(22,130) ' '
 WRITE(22,120) '       THE INPUT SHIP PARAMETERS ARE AS FOLLOWS'
 WRITE(22,130) ' '
 WRITE(22,150) 'THE REPORT IS LOCATED IN FILE :        ',PW_PRFILE
 WRITE(22,150) 'THE INPUT DATA FILE USED IS :          ',
& PW_INPUT_DATA
 WRITE(22,160) '**************************************************'
 WRITE(22,130) '                     DESIGN PARAMETERS'
 WRITE(22,150) 'SHIP NAME IS ..........................',PW_INFILE
 WRITE(22,130) ' '
 WRITE(22,140) 'LENGTH BETWEEN PERPENDICULARS ....FT....',PW_LPP
 WRITE(22,130) ' '
 WRITE(22,140) 'DESIGN DRAFT ....................FT....',PW_T
 WRITE(22,130) ' '
 WRITE(22,140) 'DESIGN BEAM .....................FT....',PW_B
 WRITE(22,130) ' '
 WRITE(22,140) 'DESIGN DISPLACEMENT ............LTONS..',PW_DISP
 WRITE(22,130) ' '
 WRITE(22,160) 'COEFFICIENTS OF FORM********************'
 WRITE(22,140) 'BLOCK COEFFICIENT: ',PW_CB
 WRITE(22,140) 'SPEED-LENGTH RATIO:',PW_SLR
 WRITE(22,140) 'BEAM-DRAFT RATIO:  ',PW_BTR
 WRITE(22,140) 'LENGTH-BEAM RATIO: ',PW_LBR
 WRITE(22,160) '**************************************'
 WRITE(22,130) ' '
 WRITE(22,140) 'DESIGN OPERATING SPEED (NOM) ....KNOTS..',PW_VK
 WRITE(22,130) ' '
 WRITE(22,140) 'PROPULSION SHAFT SPEED ...........RPM..',PW_NRPM
 WRITE(22,130) ' '
 WRITE(22,140) 'NUMBER OF PROPULSION SHAFTS ...........',PW_NSHAFT
 WRITE(22,130) ' '
 WRITE(22,130) '**************************************************'
 WRITE(22,130) 'DESIGN RESULTS (HORSEPOWER)'
 WRITE(22,130) ' '
 WRITE(22,140) 'ADMIRALTY POWER.........................',PW_ADME
 WRITE(22,130) ' '
 WRITE(22,130) 'SILVERLEAF & DAWSON POWER:'
 WRITE(22,130) ' '
 WRITE(22,140) '       DESIGN W/O BULBOUS BOW...........',PW_SDNB
 WRITE(22,130) ' '
 WRITE(22,140) '       DESIGN W/  BULBOUS BOW...........',PW_SDBB
 WRITE(22,130) ' '
 WRITE(22,130) ' '
 WRITE(22,130) '**************************************************'
 WRITE(22,130) 'POWER ESTIMATIONS PRESENTED ARE BASED UPON FOLLOWI
&NG: '
 WRITE(22,130) ' (1) METHOD OF ADMIRALTY COEFFICIENTS'
 WRITE(22,130) '     HARVALD, Sv.Aa.,"RESISTANCE AND PROPULSION O
&F'
 WRITE(22,130) '     SHIPS", JOHN WILEY & SONS, NEW YORK,N.Y.; 19
&83'
 WRITE(22,130) ' (2) METHOD OF SILVERLEAF AND DAWSON, AS MODIFIED
& BY S.'
 WRITE(22,130) '     ERICHSEN: ERICHSEN,S. , REPORT No.123,'
 WRITE(22,130) '     "OPTIMUM CAPACITY OF SHIPS AND PORT TERMINAL
```

```
     &S"'
      WRITE(22,130) '     UNIV. OF MICHIGAN, ANN ARBOR,MI.; 1971'
      WRITE(22,130) '****************************************************'
      WRITE (22,120) ' '
      WRITE (22,120) ' PAGE 2 OF 2 / POWER PREDICTION'
      CLOSE (22)
 100  FORMAT (A)
 110  FORMAT (G12.5)
 120  FORMAT (10X,A)
 130  FORMAT (10X,A)
 135  FORMAT (10X,A,'..........',A)
 140  FORMAT (10X,A,'..........',G14.6)
 150  FORMAT (10X,2A)
 160  FORMAT (10X,A)
C*****EXIT FROM THE REPORT AND ENABLE MOUSE
 30   CALL UIS$ERASE(VD_ID, 4.1,20.1,8.4,24.9)
      CALL UIS$TEXT(VD_ID,7,' SELECT AN OPTION WITH',4.5,24.)
      CALL UIS$TEXT(VD_ID,7,'      THE MOUSE',4.5,23.6)
      RETURN
      END
C
```

APPENDIX E.   TOOL BOX ENDURANCE ESTIMATION

A.   SUBROUTINE ENDURANCE SOURCE CODE

This program is the TOOL BOX Endurance Estimation module that is called from the Main Menu, and all of its associated subroutines.

```
      SUBROUTINE ENDURANCE
C***********************************************************************
C                                                                      *
C   THIS SUBROUTINE IS THE MAIN PART OF THE ENDURANCE CALCULATIONS.    *
C   IT SETS UP SCREENS AND CONTROLS ENDURANCE PROGRAM ACTIONS.         *
C                                                                      *
C   IT CALCULATES ENDURANCE USING THE USN DDS9400-1 FORMAT
C   IT IS CALLED FROM THE MAIN TOOL BOX MENU BY THE "ENDURANCE"        *
C   SELECTION USING THE MOUSE.  IT REQUIRES THE FOLLOWING INPUTS FOR   *
C   OPERATION AFTER INITIALIZATION:                                    *
C                                                                      *
C      ND_DISP      FULL LOAD DISPLACEMENT (TONS)    REAL              *
C      ND_PWR       CRUISING POWER (HP)              REAL              *
C      ND_NDPWR     AVERAGE ENDURANCE POWER REQD FOR CRUISING          *
C                                                    REAL              *
C      ND_ELEC      CRUISING ELECTRIC LOAD (KW)      REAL              *
C      ND_MFR       MAIN PROPULSION FUEL RATE (LBS/HR)   REAL          *
C      ND_EFR       CRUISING ELECTRIC PLANT FUEL RATE (LBS/KW-HR)      *
C                                                    REAL              *
C      ND_OTH       OTHER FUEL CONSUMPTION RATES     REAL              *
C      ND_TPA       FUEL STORAGE TAIL PIPE ALLOW (%) REAL              *
C                                                                      *
C   USING THE FOLLOWING ITERATIVE VARIABLES:                           *
C      ND_RANGE     ENDURANCE RANGE (NMILES)         REAL              *
C      ND_SPD       ENDURANCE SPEED (KNOTS)          REAL              *
C      ND_FUEL      VOYAGE FUEL CAPACITY (TONS)      REAL              *
C                                                                      *
C   AND CALCULATES THE FOLLOWING SOLUTION VALUES:                      *
C      ND_TIME      TIME (HRS) FOR JOURNEY OF ND_RANGE AT ND_SPD       *
C                                                    REAL              *
C      ND_VOL       FUEL STORAGE CAPACITY REQD       REAL              *
C      ND_PCT       FUEL PERCENT OF FULL LOAD DISPL  REAL              *
C                                                                      *
C   ALL PARAMETERS ARE PASSED IN COMMON BY THE INCLUDE 'TOP_ENDU.FOR'  *
C      FILE THAT HAS ALL THE PARAMETER DEFINITIONS, TYPE STATEMENTS,   *
C      AND SET UP.                                                     *
C                                                                      *
C   CALLED BY MAIN MODULE 'TOOL_BOX'                                   *
C   CALLS SUBROUTINES 'ND_KB_DATA_IN','ND_READ_FILE',ND_RECORD',       *
C         'ND_CALC_1','TO_MAIN',AND 'EXIT'                             *
C                                                                      *
C***********************************************************************
        INCLUDE 'GENERAL.FOR'
        INCLUDE 'TOP_ENDU.FOR'
        EXTERNAL TO_MAIN,EXIT,DARK2,LIGHT2,NOWHERE,ND_CALC_2
        EXTERNAL ND_KB_DATA_IN,ND_RECORD,ND_READ_FILE,ND_CALC_1
        EXTERNAL ND_SDI_LABEL,ND_SORT,MUCH_TOO_MUCH,TOO_MUCH,ND_LIST
        EXTERNAL KEY_READ,SET_HELP_LEVEL
C*****SET HELP LEVEL FOR PARAMETER WINDOWS TO 'ON' UPON ENTRY
        ND_HELP=1
C*****INITIALIZE THE PARAMETERS TO AVOID DIVIDE BY ZERO ERRORS.  ALSO
C     THIS INSURES THAT WHEN WE INITIALLY CALL THE KEYBOARD DATA ENTRY
C     ROUTINE THAT THE DATA PASSES W/O DIVIDE BY ZERO ERRORS
        IF ( ND_ENTERS .EQ. 0 ) THEN
C*****INPUT VARIABLES
           ND_SOLVE=1
           ND_INFILE='GENERIC'
           ND_KB_UP(1)='GENERIC'
           ND_DISP=1.0
           ND_KB_UP(2)='1.0'
           ND_PWR=0.0001
           ND_KB_UP(3)='0.0001'
           ND_NDPWR=0.0001
           ND_KB_UP(4)='0.0001'
           ND_ELEC=0.0001
           ND_KB_UP(5)='0.0001'
           ND_TPA=0.0001
           ND_KB_UP(6)='0.0001'
           ND_MFR=0.0001
           ND_KB_UP(7)='0.0001'
           ND_EFR=0.0001
           ND_KB_UP(8)='0.0001'
           ND_OTH=0.0001
           ND_KB_UP(9)='0.0001'
           ND_RANGE=0.0001
           ND_KB_UP(10)='0.0001'
           ND_SPD=0.0001
           ND_KB_UP(11)='0.0001'
           ND_FUEL=0.0001
           ND_KB_UP(12)='0.0001'
           ND_WGT=0.0001
           ND_KB_UP(18)='0.0001'
           ND_VOL=0.0001
           ND_KB_UP(19)='0.0001'
           ND_PCT=0.0001
           ND_KB_UP(20)='0.0001'
        ENDIF
        ND_ENTERS=ND_ENTERS+1
C*****ERASE THE DATA AREA
        CALL UIS$ERASE(VD_ID,3.6,10.1,9.9,18.5)
C*****REMOVE THE MOUSE POINTER TO SOME OTHER AREA
        STA = UIS$SET_POINTER_POSITION(VD_ID,WD_MAIN,9.9,4.5)
C*****CREATE POWER WINDOW
        WD_NDUR=UIS$CREATE_WINDOW(VD_ID,'SYS$WORKSTATION','ENDURANCE
     & PREDICTION WINDOW',-.5,9.9,10.1,19.,40.,30.)
        DO 10 Y_COOR = 13.1,18.4,.8
          DY = Y_COOR + .2
          CALL UIS$SET_POINTER_AST(VD_ID,WD_NDUR,DARK2,,X0,Y_COOR,X1,
     &    DY,LIGHT2)
 10     CONTINUE
C*****SET MENU TITLES
```

```
        OPTION(11) = OPTION(23)
        OPTION(12) = OPTION(22)
C*****WRITE SCREEN
        CALL UIS$TEXT(VD_ID,0,OPTION(11),.3,15.1)
        CALL UIS$TEXT(VD_ID,0,OPTION(12),.3,15.9)
        CALL UIS$SET_BUTTON_AST(VD_ID,WD_NDUR,EXIT,,,X0,13.0,
     &      X1,13.4)
        CALL UIS$SET_BUTTON_AST(VD_ID,WD_NDUR,TO_MAIN,,,X0,13.8,
     &      X1,14.2)
        CALL UIS$SET_BUTTON_AST(VD_ID,WD_NDUR,SET_HELP_LEVEL,,,X0,14.6,
     &      X1,15.0)
        CALL UIS$SET_BUTTON_AST(VD_ID,WD_NDUR,ND_CALC_1,,,X0,15.4,
     &      X1,15.8)
        CALL UIS$SET_BUTTON_AST(VD_ID,WD_NDUR,ND_RECORD,,,X0,16.2,
     &      X1,16.6)
        CALL UIS$SET_BUTTON_AST(VD_ID,WD_NDUR,ND_READ_FILE,,,X0
     &      ,17.,X1,17.4)
        CALL UIS$SET_BUTTON_AST(VD_ID,WD_NDUR,ND_KB_DATA_IN,
     &      ,,X0,17.8, X1,18.2)
        WNDOW = WD_NDUR
20      RETURN
        END
C
C
      SUBROUTINE ND_KB_DATA_IN
C*******************************************************************
C                                                                  *
C    THE ND_KB_DATA_IN ROUTINES LOAD IN KB DATA AND ALLOWS         *
C    THE OPERATOR TO CHANGE INPUTS AND SEE REAL TIME EFFECTS       *
C    OF THOSE CHANGES ON THE ENDURANCE CHARACTERISTICS OF THE      *
C    PRELIMINARY DESIGN.                                           *
C                                                                  *
C    CALLED BY SUBROUTINE 'ENDURANCE'                              *
C    CALLS SUBROUTINES 'ND_SDI_LABEL','KEY_READ','ND_SORT',        *
C                      'ND_CALC_1', 'ND_LIST'                      *
C*******************************************************************
C
       INCLUDE 'TOP.FOR'
       INCLUDE 'TOP_ENDU.FOR'
       REAL ND_LINE_NO
       STA=UIS$SET_POINTER_POSITION(VD_ID,WD_NDUR,9.9,14.5)
       KB_ID=UIS$CREATE_KB('SYS$WORKSTATION')
       CALL UIS$ENABLE_KB(KB_ID,WD_NDUR)
C*****FIRST, WE WE WRITE UP THE LINE LABLES AND FIRST INSTRUCTIONS
       CALL ND_SDI_LABEL(VD_ID)
C*****SECOND, CALCULATE AND DISPLAY STARTING VALUES
       CALL ND_CALC_1
C*****THIRD, ALLOW USER TO CHANGE INPUT THEN
C*****RECALCULATE RESULTS IN REAL TIME
 1     CALL UIS$ERASE(VD_ID,-0.4,10.1,3.4,10.9)
       CALL UIS$TEXT(VD_ID,7,'ENTER A LINE NUMBER',0.0,11.0)
       CALL UIS$TEXT(VD_ID,7,'OR [RETURN] TO EXIT',0.0,10.6)
 3     CALL  KEY_READ(ND_LINE,.TRUE.,0.0,12.0,*300)
       READ (ND_LINE,FMT='(F2.0)',ERR = 5) ND_LINE_NO
       NLINE_NO = INT(ND_LINE_NO)
      GO TO (10,20,30,40,50,60,70,80,90,100,110) NLINE_NO
C
C*****IF IT GETS TO HERE A MISTAKE HAS BEEN MADE ***********************
C
 5    CALL UIS$ERASE(VD_ID,-0.4,10.1,3.4,10.9)
      CALL UIS$TEXT(VD_ID,7,'IMPROPER LINE NUMBER',0.0,11.0)
      CALL UIS$TEXT(VD_ID,7,'ENTER A NEW NUMBER PLEASE',0.0,10.6)
      GOTO 3
C
 10   CALL ND_SORT(NLINE_NO,DEL_Y,ND_LINE,*300)
      ND_KB_UP(1) = ND_LINE
      ND_INFILE = ND_LINE
      READ (ND_LINE, FMT='(G)',ERR = 5)
      WRITE (ND_KB_UP(1), FMT='(A)') ND_INFILE
      CALL UIS$ERASE(VD_ID,8.2,17.9,9.9,18.3)
      CALL UIS$TEXT(VD_ID,7,ND_KB_UP(1),8.2,18.2)
      GOTO 1
C
 20   CALL ND_SORT(NLINE_NO,DEL_Y,ND_LINE,*300)
      ND_KB_UP(2) = ND_LINE
      READ (ND_LINE, FMT='(G)',ERR = 5) ND_DISP
      WRITE (ND_KB_UP(2), FMT='(G12.5)') ND_DISP
      CALL UIS$ERASE(VD_ID,8.2,17.5,9.9,17.9)
      CALL UIS$TEXT(VD_ID,7,ND_KB_UP(2),8.2,17.8)
      GOTO 1
C
 30   ND_NDX=1
      IF ( ND_HELP .EQ.1 ) CALL ND_LIST(ND_NDX)
      CALL ND_SORT(NLINE_NO,DEL_Y,ND_LINE,*300)
      ND_KB_UP(3) = ND_LINE
      READ (ND_LINE, FMT='(G)',ERR = 5) ND_PWR
      WRITE (ND_KB_UP(3), FMT='(G12.5)') ND_PWR
      CALL UIS$ERASE(VD_ID,8.2,17.1,9.9,17.5)
      CALL UIS$TEXT(VD_ID,7,ND_KB_UP(3),8.2,17.4)
      GOTO 1
C
 40   ND_NDX=2
      IF ( ND_HELP .EQ.1 ) CALL ND_LIST(ND_NDX)
      CALL ND_SORT(NLINE_NO,DEL_Y,ND_LINE,*300)
      ND_KB_UP(4) = ND_LINE
      READ (ND_LINE, FMT='(G)',ERR = 5) ND_NDPWR
      WRITE (ND_KB_UP(4), FMT='(G12.5)') ND_NDPWR
      CALL UIS$ERASE(VD_ID,8.2,16.7,9.9,17.1)
      CALL UIS$TEXT(VD_ID,7,ND_KB_UP(4),8.2,17.0)
      GOTO 1
C
 50   ND_NDX=3
      IF ( ND_HELP .EQ.1 ) CALL ND_LIST(ND_NDX)
      CALL ND_SORT(NLINE_NO,DEL_Y,ND_LINE,*300)
      ND_KB_UP(5) = ND_LINE
      READ (ND_LINE, FMT='(G)',ERR = 5) ND_ELEC
      WRITE (ND_KB_UP(5), FMT='(G12.5)') ND_ELEC
      CALL UIS$ERASE(VD_ID,8.2,16.3,9.9,16.7)
      CALL UIS$TEXT(VD_ID,7,ND_KB_UP(5),8.2,16.6)
      GOTO 1
C
```

```
 60     ND_NDX=4
        IF ( ND_HELP .EQ. 1 ) CALL ND_LIST(ND_NDX)
        CALL ND_SORT(NLINE_NO,DEL_Y,ND_LINE,*300)
        ND_KB_UP(6) = ND_LINE
        READ (ND_LINE, FMT='(G)',ERR = 5) ND_TPA
        WRITE (ND_KB_UP(6), FMT='(G12.5)') ND_TPA
        CALL UIS$ERASE(VD_ID,8.2,15.9,9.9,16.3)
        CALL UIS$TEXT(VD_ID,7,ND_KB_UP(6),8.2,16.2)
        GOTO 1
C
 70     ND_NDX=5
        IF ( ND_HELP .EQ. 1 ) CALL ND_LIST(ND_NDX)
        CALL ND_SORT(NLINE_NO,DEL_Y,ND_LINE,*300)
        ND_KB_UP(7) = ND_LINE
        READ (ND_LINE, FMT='(G)',ERR = 5) ND_MFR
        WRITE (ND_KB_UP(7), FMT='(G12.5)') ND_MFR
        CALL UIS$ERASE(VD_ID,8.2,15.5,9.9,15.9)
        CALL UIS$TEXT(VD_ID,7,ND_KB_UP(7),8.2,15.8)
        GOTO 1
C
 80     ND_NDX=6
        IF ( ND_HELP .EQ. 1 ) CALL ND_LIST(ND_NDX)
        CALL ND_SORT(NLINE_NO,DEL_Y,ND_LINE,*300)
        ND_KB_UP(8) = ND_LINE
        READ (ND_LINE, FMT='(G)',ERR = 5) ND_EFR
        WRITE (ND_KB_UP(8), FMT='(G12.5)') ND_EFR
        CALL UIS$ERASE(VD_ID,8.2,15.1,9.9,15.5)
        CALL UIS$TEXT(VD_ID,7,ND_KB_UP(8),8.2,15.4)
        GOTO 1
C
 90     ND_NDX=7
        IF ( ND_HELP .EQ. 1 ) CALL ND_LIST(ND_NDX)
        CALL ND_SORT(NLINE_NO,DEL_Y,ND_LINE,*300)
        ND_KB_UP(9) = ND_LINE
        READ (ND_LINE, FMT='(G)',ERR = 5) ND_OTH
        WRITE (ND_KB_UP(9), FMT='(G12.5)') ND_OTH
        CALL UIS$ERASE(VD_ID,8.2,14.7,9.9,15.1)
        CALL UIS$TEXT(VD_ID,7,ND_KB_UP(9),8.2,15.0)
        GOTO 1
C
 100    CALL ND_SORT(NLINE_NO,DEL_Y,ND_LINE,*300)
        ND_KB_UP(10) = ND_LINE
        READ (ND_LINE, FMT='(G)',ERR = 5) ND_RANGE
        WRITE (ND_KB_UP(10), FMT='(G12.5)') ND_RANGE
        CALL UIS$ERASE(VD_ID,8.2,14.3,9.9,14.7)
        CALL UIS$TEXT(VD_ID,7,ND_KB_UP(10),8.2,14.6)
        CALL ND_CALC_1
        GOTO 1
C
 110    CALL ND_SORT(NLINE_NO,DEL_Y,ND_LINE,*300)
        ND_KB_UP(11) = ND_LINE
        READ (ND_LINE, FMT='(G)',ERR = 5) ND_SPD
        WRITE (ND_KB_UP(11), FMT='(G12.5)') ND_SPD
        CALL UIS$ERASE(VD_ID,8.2,13.9,9.9,14.3)
        CALL UIS$TEXT(VD_ID,7,ND_KB_UP(11),8.2,14.2)
        CALL ND_CALC_1
```

```
       GOTO 1
C
C
 300   CALL UIS$DISABLE_KB(KB_ID)
       CALL UIS$ERASE(VD_ID,-.4,10.1,3.4,10.9)
       CALL UIS$TEXT(VD_ID,7,'SELECT AN OPTION',0.,11.)
       CALL UIS$TEXT(VD_ID,7,' WITH THE MOUSE ',0.,10.6)
       RETURN
       END
C
C
       SUBROUTINE ND_SORT(NLINE_NO,DEL_Y,ND_LINE,COUNT,*)
C*********************************************************************
C                                                                    *
C     SUBROUTINE ND_SORT SORTS OUT THE LINE NUMBER TO THE            *
C     VARIABLE INVOLVED AND CALLS THE APPROPRIATE SCREEN             *
C     INSTRUCTION TO BE WRITTEN IN THE INSTRUCTION BOX.              *
C                                                                    *
C     CALLS SUBROUTINE 'KEY_READ'                                    *
C*********************************************************************
C   .
      INCLUDE 'TOP.FOR'
      INCLUDE 'TOP_ENDU.FOR'
      REAL DEL_Y, DY, DOT_Y
      LOGICAL ND_FCHAR
      CHARACTER ND_KB_INST(11)*25
C*****SET THE SCREEN INSTRUCTIONS STRINGS
      ND_KB_INST(1) = 'ENTER NEW SHIP NAME      '
      ND_KB_INST(2) = 'ENTER FULL LOAD DISPL    '
      ND_KB_INST(3) = 'ENTER FULL RATED POWER   '
      ND_KB_INST(4) = 'ENTER ENDURANCE POWER    '
      ND_KB_INST(5) = 'ENTER NOMINAL ELEC LOAD  '
      ND_KB_INST(6) = 'ENTER TPA AS 0.XY%       '
      ND_KB_INST(7) = 'ENTER FUEL RATE (LBS/HR)'
      ND_KB_INST(8) = 'ENTER FUEL USE(LBS/KwHR)'
      ND_KB_INST(9) = 'ENTER FUEL USE (LBS/HR) '
      ND_KB_INST(10) = 'ENTER ENDURANCE RANGE  '
      ND_KB_INST(11) = 'ENTER ENDURANCE SPEED  '
C*****ERASE THE INSTRUCTION AREA AND FIND ITS DATA POSITION
      CALL UIS$ERASE(VD_ID,-0.4,10.1,3.4,10.9)
      DEL_Y = 18.6 - .4 * NLINE_NO
C*****WRITE THE APPROPRIATE INSTRUCTION STRING
      CALL UIS$TEXT(VD_ID,7,ND_KB_INST(NLINE_NO),0.0,11.0)
      CALL UIS$TEXT(VD_ID,7,'[RETURN] TO EXIT',0.0,10.6)
C*****DECIDE IF WE WANT THE FIRST LINE (NAME) OR A VALUE
      IF (NLINE_NO .EQ. 1) THEN
           ND_FCHAR = .FALSE.
      ELSE
           ND_FCHAR = .TRUE.
      ENDIF
      CALL KEY_READ(ND_LINE,ND_FCHAR,0.0,12.0,*300)
      DY = DEL_Y - .35
      CALL UIS$ERASE(VD_ID,8.2,DY,9.9,DEL_Y)
      CALL UIS$TEXT(VD_ID,7,ND_LINE,8.2,DEL_Y)
      RETURN
 300  RETURN 1
```

```
      END
C
C
       SUBROUTINE ND_SDI_LABEL(VD_ID)
C***********************************************************************
C                                                                      *
C  PERFORM DISPLAY VARIABLES IS A ROUTINE WHICH READS UP THE           *
C  DATA ONTO THE DISPLAY AREA ALONG WITH THE SCREEN LABELS.            *
C                                                                      *
C  CALLED BY SUBROUTINES 'ENDURANCE','ND_KB_DATA_IN'                   *
C***********************************************************************
C
        IMPLICIT INTEGER (A-Z)
        INCLUDE 'TOP_ENDU.FOR'
        INCLUDE 'SYS$LIBRARY:UISENTRY'
        INCLUDE 'SYS$LIBRARY:UISUSRDEF'
        REAL DEL_Y, DOT_Y
        CHARACTER N_LINE_LBL(20)*35
        CALL UIS$ERASE(VD_ID,4.,10.1,9.9,18.3)
        N_LINE_LBL(1) =  '1.   SHIP NAME ..................'
        N_LINE_LBL(2) =  '2.   FULL LOAD DISPLACEMENT(TONS)'
        N_LINE_LBL(3) =  '3.   FULL RATED PLANT POWER (SHP)'
        N_LINE_LBL(4) =  '4.   AVG ENDURANCE POWER (SHP)...'
        N_LINE_LBL(5) =  '5.   CRUISING ELEC LOAD.....(KW).'
        N_LINE_LBL(6) =  '6.   TAIL PIPE ALLOWANCE....(%)..'
        N_LINE_LBL(7) =  '7.   CRUISING MAIN FUEL RATE.....'
        N_LINE_LBL(8) =  '8.   CRUISING ELEC FUEL RATE.....'
        N_LINE_LBL(9) =  '9.   CRUISING OTHER FUEL RATE....'
        N_LINE_LBL(10) = '10.*EST.ENDURANCE RANGE (MILES).'
        N_LINE_LBL(11) = '11.*EST.ENDURANCE SPEED (KNOTS).'
        N_LINE_LBL(12) = '*********************************'
        N_LINE_LBL(13) = ' '
        N_LINE_LBL(14) = '     **RESULTS FROM CALC**'
        N_LINE_LBL(15) = '*********************************'
        N_LINE_LBL(16) = ' EST. ENDURANCE FUEL LOAD (TONS)'
        N_LINE_LBL(17) = ' ELAPSED TRIP TIME (HRS):.......'
        N_LINE_LBL(18) = ' FUEL STORAGE REQUIREMENTS(FT
        N_LINE_LBL(19) = ' FUEL WEIGHT ALLOWANCE (%-FL):..'
        N_LINE_LBL(20) = '*********************************'
        DO 10 I = 1,20,1
           DEL_Y = 18.6  - .4*I
           CALL UIS$TEXT(VD_ID,7,N_LINE_LBL(I),4.,DEL_Y)
10      CONTINUE
        DO 20 I = 1,9,1
           DEL_Y = 18.6  - .4*I
           CALL UIS$TEXT(VD_ID,7,ND_KB_UP(I),8.2,DEL_Y)
20      CONTINUE
        RETURN
        END
C
C
      SUBROUTINE ND_CALC_1
C***********************************************************************
C                                                                      *
C  ND_CALC_1 CALLS ND_CALC_2 WHICH DOES THE ACTUAL ENDUARNCE           *
C  CALCULATIONS.  ON RETURN, ND_CALC_1 DISPLAYS THE RESULTS            *
```

```
C  OF THE CALCULATIONS AND THEN RETURNS TO THE ROOT PROGRAM.           *
C                                                                      *
C  CALLED BY SUBROUTINES 'ENDURANCE','ND_KB_DATA_IN'                   *
C  CALLS SUBROUTINE 'ND_CALC_2'                                        *
C *********************************************************************
C
         INCLUDE 'TOP.FOR'
         INCLUDE 'TOP_ENDU.FOR'
         REAL DY
         CALL UIS$ERASE(VD_ID,8.1,9.95,9.9,14.7)
         CALL ND_CALC_2
         WRITE( ND_KB_UP(10), FMT='(G12.5)') ND_RANGE
         WRITE( ND_KB_UP(11), FMT='(G12.5)') ND_SPD
         ND_KB_UP(12) = '*************'
         ND_KB_UP(13) = '  '
         ND_KB_UP(14) = '  '
         ND_KB_UP(15) = '*************'
         ND_KB_UP(20) = '*************'
         WRITE( ND_KB_UP(16), FMT='(G12.5)') ND_FUEL
         WRITE( ND_KB_UP(17), FMT='(G12.5)') ND_TIME
         WRITE( ND_KB_UP(18), FMT='(G12.5)') ND_VOL
         WRITE( ND_KB_UP(19), FMT='(G12.5)') ND_PCT
         DO 10 I = 10,20,1
            DY = 18.6 -.4*I
            CALL UIS$TEXT(VD_ID,7,ND_KB_UP(I),8.2,DY)
 10      CONTINUE
         RETURN
         END
C
C
      SUBROUTINE ND_CALC_2
C**********************************************************************
C                                                                      *
C  SUBROUTINE ND_CALC_2 JUST DOES THE CALCULATION OF THE ENDURANCE*
C  REQUIREMENTS FROM THE INPUT DATA.  IT IS BASED UPON THE NAVY   *
C  DDS9400-1 ALGORITHM FOR ENDURANCE ESTIMATION. IT IS ALSO USED  *
C  IN THE REPORT SECTION FOR GENERATION OF THE WRITTEN REPORT OF  *
C  THE DESIGN.                                                         *
C                                                                      *
C  CALLED BY SUBROUTINE 'ND_CALC_1'                                    *
C  CALLS SUBROUTINE 'ND_FAIL'                                          *
C**********************************************************************
C
      INCLUDE 'TOP.FOR'
      INCLUDE 'TOP_ENDU.FOR'
C
C*****CALCULATE AVERAGE ENDURANCE POWER
      ND_C_AVG = ND_NDPWR * 1.10
C*****CALCULATE POWER RATIO FOR USE LATER
      ND_RATIO = ND_C_AVG / ND_PWR
C*****CALCULATE PROPULSION F/O CONSUMPTION RATE
      ND_PFRATE = ND_MFR * ND_C_AVG
C*****CALCULATE AUXILIARY GENERATOR F/O RATE
      ND_GFRATE = ND_EFR * ND_ELEC
C*****CALCULATE TOTAL F/O USEAGE RATE
```

```
      ND_TFR = ND_PFRATE + ND_GFRATE + ND_OTH
C*****CALCULATE ALL PURPOSE F/O RATE
      ND_APFR = ND_TFR / ND_C_AVG
C*****SET FUEL RATE CORRECTION FACTOR
      IF ( ND_C_AVG .LE.  0.3333*ND_PWR ) THEN
          FRCORR = 1.04
      ELSEIF ( ND_C_AVG .GT.  0.3333*ND_PWR .AND.
     &         ND_C_AVG .LE.  0.6667*ND_PWR ) THEN
          FRCORR = 1.03
      ELSEIF ( ND_C_AVG .GT.  0.6667*ND_PWR .AND.
     &         ND_C_AVG .LE.  ND_PWR ) THEN
          FRCORR = 1.02
      ELSE
          FRCORR = 1.03
      ENDIF
C*****CALCULATE SPECIFIED F/O RATE
      ND_SFR = FRCORR * ND_APFR
C*****CALCULATE AVERAGE ENDURANCE F/O RATE
      ND_AVGFR = ND_SFR * 1.05
C*****DETERMINE BURNABLE FUEL LOAD
      ND_WGT =   ( ND_RANGE * ND_C_AVG * ND_AVGFR )
     &           / ( ND_SPD * 2240.0 )
C*****ADJUST FOR UNUSEABLE FUEL BELOW SUCTION
      ND_FUEL = ND_WGT / ND_TPA
C*****CALCULATE FINAL OUTPUT VARIABLES
      ND_VOL = ND_FUEL * 38.00
      ND_PCT = ND_FUEL * 100.0 / ND_DISP
      ND_TIME = ND_RANGE / ND_SPD
C*****WHAT IF WE ARE OVER DRASTICALLY OR JUST A LITTLE?
      IF ( ND_FUEL .GT.  0.150*ND_DISP ) CALL TOO_MUCH
      IF ( ND_FUEL .GT.  ND_DISP ) CALL MUCH_TOO_MUCH
      RETURN
      END
C
C
        SUBROUTINE ND_RECORD
C*******************************************************************
C                                                                  *
C    SUBROUTINE 'ND_RECORD' SAVES THE KBD INPUT DATA TO A USER     *
C    SPECIFIED FILE.  FILE TYPE IS '.DAT' AUTOMATICALLY            *
C                                                                  *
C    CALLED BY SUBROUTINE 'ENDURANCE'                              *
C    CALLS SUBROUTINES 'SHOW_SAVE'                                 *
C*******************************************************************
C
        INCLUDE 'TOP.FOR'
        INCLUDE 'TOP_ENDU.FOR'
        STA = UIS$SET_POINTER_POSITION(VD_ID,WD_NDUR,0.,10.)
        CALL SHOW_SAVE
        OPEN(23,FILE=ND_INFILE,STATUS='UNKNOWN',blank='null')
        WRITE (23,200) ND_INFILE
        WRITE (23,210) ND_DISP
        WRITE (23,210) ND_PWR
        WRITE (23,210) ND_NDPWR
        WRITE (23,210) ND_ELEC
        WRITE (23,210) ND_TPA
```

```
       WRITE (23,210) ND_MFR
       WRITE (23,210) ND_EFR
       WRITE (23,210) ND_OTH
       WRITE (23,210) ND_RANGE
       WRITE (23,210) ND_SPD
       WRITE (23,210) ND_FUEL
       WRITE (23,210) ND_TIME
       WRITE (23,210) ND_VOL
       WRITE (23,210) ND_PCT
       ENDFILE (23)
       REWIND 23
       CLOSE(23)
C
C*****AND REPORT OUT WHEN DONE
C
       ND_FNAME = ND_INFILE(:INDEX(ND_INFILE,' '))//'.DAT'
       CALL UIS$ERASE(VD_ID,-.4,10.1,3.4,10.9)
       CALL UIS$TEXT(VD_ID,7,'FILE SAVED AS',0.,11.)
       CALL UIS$TEXT(VD_ID,7,ND_FNAME,0.,10.6)
 200   FORMAT (BN,2X,A)
 210   FORMAT (G12.5)
       RETURN
       END
C
C
      SUBROUTINE ND_READ_FILE
C*******************************************************************
C                                                                  *
C     READ FILE IS A SUBROUTINE WHICH READS THE DATA INPUT FILE    *
C     SPECIFIED BY KEYBOARD ENTRY                                  *
C                                                                  *
C     CALLED BY SUBROUTINE 'ENDURANCE'                             *
C     CALLS SUBROUTINES 'ND_SDI_LABEL','ND_CALC_1'                 *
C*******************************************************************
C
      INCLUDE 'TOP.FOR'
      INCLUDE 'TOP_ENDU.FOR'
      STA=UIS$SET_POINTER_POSITION(VD_ID,WD_NDUR,0.,10.2)
      KB_ID=UIS$CREATE_KB('SYS$WORKSTATION')
      CALL UIS$ENABLE_KB(KB_ID,WD_NDUR)
      CALL UIS$ERASE(VD_ID,-.4,10.1,2.3,10.9)
      CALL UIS$TEXT(VD_ID,7,'ENTER THE FILE NAME',0.,11.)
      CALL UIS$TEXT(VD_ID,7,'AND FILE EXTENSION',0.,10.6)
      CALL KEY_READ(ND_LINE,'TRUE',0.,12.,*150)
      OPEN(23,FILE=ND_LINE,STATUS='UNKNOWN')
      READ (23,200) ND_INFILE
      READ (23,210) ND_DISP
      READ (23,210) ND_PWR
      READ (23,210) ND_NDPWR
      READ (23,210) ND_ELEC
      READ (23,210) ND_TPA
      READ (23,210) ND_MFR
      READ (23,210) ND_EFR
      READ (23,210) ND_OTH
      READ (23,210) ND_RANGE
      READ (23,210) ND_SPD
```

```
       READ (23,210) ND_FUEL
       READ (23,210) ND_TIME
       READ (23,210) ND_VOL
       READ (23,210) ND_PCT
       REWIND 23
C**** NOW READ IT IN AS CHARACTER DATA TO PRINT TO SCREEN
       READ (23,200) ND_KB_UP(1)
       DO 10 I = 2,12,1
           READ (23,220) ND_KB_UP(I)
 10    CONTINUE
C
 200   FORMAT (2X,A)
 210   FORMAT (G12.5)
 220   FORMAT (2X,A)
 150   CLOSE(23)
       CALL ND_SDI_LABEL(VD_ID)
       CALL UIS$DISABLE_KB(KB_ID)
       CALL ND_CALC_1
       CALL UIS$ERASE(VD_ID,-.4,10.1,3.4,10.9)
       CALL UIS$TEXT(VD_ID,7,'SELECT AN OPTION',0.,11.)
       CALL UIS$TEXT(VD_ID,7,' WITH THE MOUSE ',0.,10.6)
       RETURN
       END
C
C
C
      SUBROUTINE ND_LIST(ND_NDX)
C*******************************************************************************
C                                                                             *
C     THIS SUBROUTINE POPS UP A DIALOG BOX TO INSTRUCT THE USER OF            *
C     THE AVAILABLE OPTIONS FOR EACH '...NOT QUITE OBVIOUS...' PARA-          *
C     METER THAT IS REQUESTED BY THE 'ENDURANCE' SUBROUTINE.  THE             *
C     SPECIFIC PARAMETER IS PASSED AS 'ND_CAUSE' AND INDICATES WHICH          *
C     LINES OF INSTRUCTION ARE TO BE DISPLAYED TO THE USER.                   *
C                                                                             *
C     CALLED BY SUBROUTINE 'ND_CALC_2'                                        *
C*******************************************************************************
C
      IMPLICIT INTEGER(A-Z)
      INCLUDE 'SYS$LIBRARY:UISENTRY'
      INCLUDE 'SYS$LIBRARY:UISUSRDEF'
      INCLUDE 'TOP_ENDU.FOR'
      CHARACTER*34 TOP_A/'*****  PARAMETER DEFINITION  *****'/
      CHARACTER*34 TOP_B/'PLEASE ENTER FOLLOWING INFORMATION'/
      CHARACTER*34 GO_A /'PLEASE WAIT........................'/
      CHARACTER*34 GO_B /'................PROGRAM WILL RESUME'/
      CHARACTER*34 GO_C /'OR PRESS [HOLD SCREEN] TO SAVE....'/
      REAL Y_POSN
      CHARACTER*36 TELL(7),LIST(7),MLIST(7),XLIST(7)
C*****INITIALIZE THE HEADINGS MESSAGES ARRAY
      TELL(1) = '     THE FULL POWER RATING AT      '
      TELL(2) = '   THE CRUISING POWER (TYPICAL)    '
      TELL(3) = '   THE CRUISING ELECTRICAL LOAD    '
      TELL(4) = '  STORAGE TANK TAIL PIPE ALLOWANCE'
      TELL(5) = '  MAIN PROPULSION MACH FUEL USEAGE'
      TELL(6) = '  GENERATOR FUEL CONSUMPTION RATE '
      TELL(7) = 'OTHER (MISC) FUEL CONSUMPTION RATE'
C*****INITIALIZE THE INSTRUCTIONS MESSAGES ARRAY
      LIST(1) = ' DESIGN MAXIMUM FULL POWER (SHP)  '
      LIST(2) = 'FOR STEADY CRUISING, UNDER NORMAL '
      LIST(3) = 'TO OPERATE NORMAL SHIPS FUNCTIONS '
      LIST(4) = 'FOR %-FUEL ABOVE THE SUCTION POINT'
      LIST(5) = 'FOR NORMAL CONDITIONS AT CRUISING '
      LIST(6) = 'AT CRUISING ELEC LOAD, WITH NORMAL'
      LIST(7) = 'FOR AUX. EQUIP, BOATS, WINCHES,ETC'
C*****INITIALIZE THE SECONDARY INSTRUCTIONS MESSAGES ARRAY
      MLIST(1) = 'NEEDED FOR PROPULSION REQUIREMENTS'
      MLIST(2) = 'WEATHER CONDITIONS & SEA STATE.   '
      MLIST(3) = 'AND REQUIRED AUX/HAB/DECK GEAR.   '
      MLIST(4) = 'OF XFER SYS THAT IS USEABLE.      '
      MLIST(5) = 'SPEED AND POWER SELECTED ABOVE.   '
      MLIST(6) = 'AUXILIARY MACHINERY IN USE.       '
      MLIST(7) = 'NORMALLY REQUIRED BY THE CREW.    '
C*****INITIALIZE EXAMPLES ARRAY
      XLIST(1) = 'EX: ADMIRALTY POWER + 25% MARGIN  '
      XLIST(2) = 'EX: 0.75 * FULL POWER (TYPICAL)   '
      XLIST(3) = 'EX: 500KW (NOMINAL) CRUISING      '
      XLIST(4) = 'EX: 0.95 FOR BROAD, SHALLOW TANKS '
      XLIST(5) = 'EX: 0.40 FOR GAS TURBINE @ 20K SHP'
      XLIST(6) = 'EX: 0.02 FOR DIESEL GENERATOR     '
      XLIST(7) = 'EX: 0.05 FOR OTHER SHIP SERVICES  '
C*****CREATE THE DISPLAY FOR THE DIALOG BOX
      VD_SCR=UIS$CREATE_DISPLAY(-5.0,-2.6,10.0,4.0,26.0,10.0)
C*****CREATE THE WINDOW TO DISPLAY THE TEXT IN
      WD_SCR=UIS$CREATE_WINDOW(VD_SCR,'SYS$WORKSTATION','HELP WINDOW')
C*****COPY ATTRIBUTE BLOCK '0' AS BLOCK '27' AND CHANGE THE FONT SIZE
      CALL UIS$SET_FONT(VD_SCR,0,27,'DTABER0R03WK00GG0001UZZZZ02A000')
C*****COPY ATTRIBUTE BLOCK '27' AS BLOCK '28' AND CHANGE TO BOLD FONT
      CALL UIS$SET_FONT(VD_SCR,27,28,'DTABER0R03WK00PG0001UZZZZ02A000')
C*****SIGNAL THE USER TO GET THEIR ATTENTION
      CALL UIS$SOUND_BELL('SYS$WORKSTATION',4)
C*****WRITE THE TEXT INTO THE WINDOW AND SPAWN PROCESS TO 'WAIT' 10 SEC
        Y_POSN=3.0
        CALL UIS$TEXT(VD_SCR,28,TOP_A,-2.5,Y_POSN)
        Y_POSN=Y_POSN-.6
        CALL UIS$TEXT(VD_SCR,28,TOP_B,-2.5,Y_POSN)
        Y_POSN=Y_POSN-.6
        CALL UIS$TEXT(VD_SCR,27,TELL(ND_NDX),-2.5,Y_POSN)
        Y_POSN=Y_POSN-.6
        CALL UIS$TEXT(VD_SCR,27,LIST(ND_NDX),-2.5,Y_POSN)
        Y_POSN=Y_POSN-.6
        CALL UIS$TEXT(VD_SCR,27,MLIST(ND_NDX),-2.5,Y_POSN)
        Y_POSN=Y_POSN-.6
        CALL UIS$TEXT(VD_SCR,27,XLIST(ND_NDX),-2.5,Y_POSN)
        Y_POSN=Y_POSN-.6
        CALL UIS$TEXT(VD_SCR,28,GO_A,-2.5,Y_POSN)
        Y_POSN=Y_POSN-.6
        CALL UIS$TEXT(VD_SCR,28,GO_B,-2.5,Y_POSN)
        Y_POSN=Y_POSN-.6
        CALL UIS$TEXT(VD_SCR,28,GO_C,-2.5,Y_POSN)
      CALL LIB$SPAWN('WAIT 00:00:10')
      CALL UIS$SOUND_BELL('SYS$WORKSTATION',4)
      CALL UIS$DELETE_DISPLAY(VD_SCR)
      RETURN
      END
C
C
      SUBROUTINE MUCH_TOO_MUCH
C***********************************************************************
C                                                                      *
C     THIS SUBROUTINE POPS UP A DIALOG BOX TO INSTRUCT THE USER        *
C     THAT THE CALCULATED WEIGHT OF FUEL IS LARGER THAN THE VESSELS    *
C     OWN DISPLACEMENT AND THAT CLEARLY SOME CHANGES HAVE TO BE        *
C     MADE TO THE INPUT DATA.                                          *
C                                                                      *
C     CALLED BY SUBROUTINE 'ND_CALC_2'                                 *
C***********************************************************************
C
      IMPLICIT INTEGER(A-Z)
      INCLUDE 'SYS$LIBRARY:UISENTRY'
      INCLUDE 'SYS$LIBRARY:UISUSRDEF'
      INCLUDE 'TOP_ENDU.FOR'
      CHARACTER*34 MTM_A/'** PARAMETER ERROR HAS OCCURRED **'/
      CHARACTER*34 MTM_B/'  PLEASE NOTE THAT THE SHIPS FUEL '/
      CHARACTER*34 MTM_C/'  REQUIREMENTS EXCEED ITS STORAGE '/
      CHARACTER*34 MTM_D/'  ABILITY. ADJUST ITEMS 2 THRU 12 '/
      CHARACTER*34 MTM_E/'PLEASE WAIT........................'/
      CHARACTER*34 MTM_F/'...............PROGRAM WILL RESUME'/
      REAL Y_POSN
C*****CREATE THE DISPLAY FOR THE DIALOG BOX
      VD_MTM=UIS$CREATE_DISPLAY(-5.0,-2.0,10.0,4.0,26.0,10.0)
C*****CREATE THE WINDOW TO DISPLAY THE TEXT IN
      WD_MTM=UIS$CREATE_WINDOW(VD_MTM,'SYS$WORKSTATION','HELP WINDOW')
C*****COPY ATTRIBUTE BLOCK '0' AS BLOCK '29' AND CHANGE THE FONT SIZE
      CALL UIS$SET_FONT(VD_MTM,0,29,'DTABER0R03WK00GG0001UZZZZ02A000')
C*****COPY ATTRIBUTE BLOCK '29' AS BLOCK '30' AND CHANGE TO BOLD FONT
      CALL UIS$SET_FONT(VD_MTM,29,30,'DTABER0R03WK00PG0001UZZZZ02A000')
C*****SIGNAL THE USER TO GET THEIR ATTENTION
      CALL UIS$SOUND_BELL('SYS$WORKSTATION',4)
C*****WRITE THE TEXT INTO THE WINDOW AND SPAWN PROCESS TO 'WAIT' 10 SEC
        Y_POSN=2.8
        CALL UIS$TEXT(VD_MTM,30,MTM_A,-2.5,Y_POSN)
        Y_POSN=Y_POSN-.6
        CALL UIS$TEXT(VD_MTM,29,MTM_B,-2.5,Y_POSN)
        Y_POSN=Y_POSN-.6
        CALL UIS$TEXT(VD_MTM,29,MTM_C,-2.5,Y_POSN)
        Y_POSN=Y_POSN-.6
        CALL UIS$TEXT(VD_MTM,29,MTM_D,-2.5,Y_POSN)
        Y_POSN=Y_POSN-.6
        CALL UIS$TEXT(VD_MTM,30,MTM_E,-2.5,Y_POSN)
        Y_POSN=Y_POSN-.6
        CALL UIS$TEXT(VD_MTM,30,MTM_F,-2.5,Y_POSN)
      CALL LIB$SPAWN('WAIT 00:00:10')
      CALL UIS$SOUND_BELL('SYS$WORKSTATION',4)
      CALL UIS$DELETE_DISPLAY(VD_MTM)
      RETURN
      END
C
```

```
C
C
      SUBROUTINE TOO_MUCH
C*************************************************************************
C                                                                       *
C     THIS SUBROUTINE POPS UP A DIALOG BOX TO INSTRUCT THE USER         *
C     THAT THE CALCULATED WEIGHT OF FUEL EXCEEDS THE NORMAL ALLOW-      *
C     ANCE FOR SHIPS OF APPROXIMATELY 15% OF TOTAL DISPLACEMENT.        *
C     THUS, CLEARLY SOME CHANGES MUST BE MADE TO THE INPUT DATA.        *
C                                                                       *
C     CALLED BY SUBROUTINE 'ND_CALC_2'                                  *
C*************************************************************************
C
      IMPLICIT INTEGER(A-Z)
      INCLUDE 'SYS$LIBRARY:UISENTRY'
      INCLUDE 'SYS$LIBRARY:UISUSRDEF'
      INCLUDE 'TOP_ENDU.FOR'
      CHARACTER*34 TM_A/'** PARAMETER ERROR HAS OCCURRED **'/
      CHARACTER*34 TM_B/'  PLEASE NOTE THAT THE SHIPS FUEL '/
      CHARACTER*34 TM_C/'  REQUIREMENTS EXCEED THE NORMAL  '/
      CHARACTER*34 TM_D/'  RANGE OF 15% TOTAL DISPLACEMENT.'/
      CHARACTER*34 TM_E/'PLEASE WAIT.......................'/
      CHARACTER*34 TM_F/'...............PROGRAM WILL RESUME'/
      REAL Y_POSN
C*****CREATE THE DISPLAY FOR THE DIALOG BOX
      VD_TM=UIS$CREATE_DISPLAY(-5.0,-2.0,10.0,4.0,26.0,10.0)
C*****CREATE THE WINDOW TO DISPLAY THE TEXT IN
      WD_TM=UIS$CREATE_WINDOW(VD_TM,'SYS$WORKSTATION','HELP WINDOW')
C*****COPY ATTRIBUTE BLOCK '0' AS BLOCK '31' AND CHANGE THE FONT SIZE
      CALL UIS$SET_FONT(VD_TM,0,31,'DTABER0R03WK00GG0001UZZZZ02A000')
C*****COPY ATTRIBUTE BLOCK '31' AS BLOCK '32' AND CHANGE TO BOLD FONT
      CALL UIS$SET_FONT(VD_TM,31,32,'DTABER0R03WK00PG0001UZZZZ02A000')
C*****SIGNAL THE USER TO GET THEIR ATTENTION
      CALL UIS$SOUND_BELL('SYS$WORKSTATION',4)
C*****WRITE THE TEXT INTO THE WINDOW AND SPAWN PROCESS TO 'WAIT' 10 SEC
        Y_POSN=2.8
        CALL UIS$TEXT(VD_TM,32,TM_A,-2.5,Y_POSN)
        Y_POSN=Y_POSN-.6
        CALL UIS$TEXT(VD_TM,31,TM_B,-2.5,Y_POSN)
        Y_POSN=Y_POSN-.6
        CALL UIS$TEXT(VD_TM,31,TM_C,-2.5,Y_POSN)
        Y_POSN=Y_POSN-.6
        CALL UIS$TEXT(VD_TM,31,TM_D,-2.5,Y_POSN)
        Y_POSN=Y_POSN-.6
        CALL UIS$TEXT(VD_TM,32,TM_E,-2.5,Y_POSN)
        Y_POSN=Y_POSN-.6
        CALL UIS$TEXT(VD_TM,32,TM_F,-2.5,Y_POSN)
      CALL LIB$SPAWN('WAIT 00:00:10')
      CALL UIS$SOUND_BELL('SYS$WORKSTATION',4)
      CALL UIS$DELETE_DISPLAY(VD_TM)
      RETURN
      END
C
C
C
```

```
      SUBROUTINE SET_HELP
C***********************************************************************
C                                                                     *
C     SET_HELP ALLOWS THE USER TO DEFINE WHETHER OR NOT PARAMETER     *
C     DEFINITION WINDOWS ARE DISPLAYED DURING KB DATA ENTRY BY        *
C     SELECTING THIS OPTION ON THE MAIN 'ENDURANCE' MENU.  THIS       *
C     FEATURE SIMPLY TOGGLES THE HELP LEVEL TO ON OR OFF AND CALLS    *
C     THIS WINDOW DISPLAY ACCORDINGLY.                                *
C                                                                     *
C     CALLED BY SUBROUTINE 'SET_HELP_LEVEL'                           *
C***********************************************************************
C
      IMPLICIT INTEGER(A-Z)
      INCLUDE 'SYS$LIBRARY:UISENTRY'
      INCLUDE 'SYS$LIBRARY:UISUSRDEF'
      INCLUDE 'TOP_ENDU.FOR'
      CHARACTER*34 HL(2)
      CHARACTER*34 SH_A/'****   HELP SYSTEM SETTINGS   ****'/
      CHARACTER*34 SH_B/'     CHANGING HELP LEVEL FROM     '/
      CHARACTER*34 SH_C/'                TO                '/
      CHARACTER*34 SH_D/'PLEASE WAIT.......................'/
      CHARACTER*34 SH_E/'...............PROGRAM WILL RESUME'/
      REAL Y_POSN
      HL(1) = '         HELP WINDOWS ON          '
      HL(2) = '         HELP WINDOWS OFF         '
C*****CREATE THE DISPLAY FOR THE DIALOG BOX
      VD_SH=UIS$CREATE_DISPLAY(-5.0,-2.0,10.0,4.0,26.0,10.0)
C*****CREATE THE WINDOW TO DISPLAY THE TEXT IN
      WD_SH=UIS$CREATE_WINDOW(VD_SH,'SYS$WORKSTATION','HELP WINDOW')
C*****COPY ATTRIBUTE BLOCK '0' AS BLOCK '33' AND CHANGE THE FONT SIZE
      CALL UIS$SET_FONT(VD_SH,0,33,'DTABER0R03WK00GG0001UZZZZ02A000')
C*****COPY ATTRIBUTE BLOCK '33' AS BLOCK '34' AND CHANGE TO BOLD FONT
      CALL UIS$SET_FONT(VD_SH,33,34,'DTABER0R03WK00PG0001UZZZZ02A000')
C*****SIGNAL THE USER TO GET THEIR ATTENTION
      CALL UIS$SOUND_BELL('SYS$WORKSTATION',4)
C*****WRITE THE TEXT INTO THE WINDOW AND SPAWN PROCESS TO 'WAIT' 10 SEC
        Y_POSN=2.5
        CALL UIS$TEXT(VD_SH,34,SH_A,-2.5,Y_POSN)
        Y_POSN=Y_POSN-.6
        CALL UIS$TEXT(VD_SH,33,SH_B,-2.5,Y_POSN)
        Y_POSN=Y_POSN-.6
        IF ( ND_HELP .EQ. 0 ) THEN
           CALL UIS$TEXT(VD_SH,34,HL(1),-2.5,Y_POSN)
        ELSEIF ( ND_HELP .EQ. 1 ) THEN
           CALL UIS$TEXT(VD_SH,34,HL(2),-2.5,Y_POSN)
        ENDIF
        Y_POSN=Y_POSN-.6
        CALL UIS$TEXT(VD_SH,33,SH_C,-2.5,Y_POSN)
        Y_POSN=Y_POSN-.6
        IF ( ND_HELP .EQ. 0 ) THEN
           CALL UIS$TEXT(VD_SH,34,HL(2),-2.5,Y_POSN)
        ELSEIF ( ND_HELP .EQ. 1 ) THEN
           CALL UIS$TEXT(VD_SH,34,HL(1),-2.5,Y_POSN)
        ENDIF
        Y_POSN=Y_POSN-.6
        CALL UIS$TEXT(VD_SH,33,SH_D,-2.5,Y_POSN)
```

```
          Y_POSN=Y_POSN-.6
          CALL UIS$TEXT(VD_SH,33,SH_E,-2.5,Y_POSN)
        CALL LIB$SPAWN('WAIT 00:00:10')
        CALL UIS$SOUND_BELL('SYS$WORKSTATION',4)
        CALL UIS$DELETE_DISPLAY(VD_SH)
        RETURN
        END
C
C
C
        SUBROUTINE SET_HELP_LEVEL
C***********************************************************************
C                                                                      *
C       THIS SUBROUTINE ALLOWS THE USER TO HAVE THE SYSTEM PARAMETER   *
C       DEFINITION WINDOWS DISPLAYED OR NOT DISPLAYED, DEPENDING UPON  *
C       THEIR KNOWLEDGE AND PREFERENCES AND FAMILIARITY WITH TOOL BOX. *
C                                                                      *
C       CALLED BY SUBROUTINE 'ENDURANCE' MAIN MENU                     *
C       CALL SUBROUTINE 'SET_HELP'                                     *
C***********************************************************************
C
        IMPLICIT INTEGER(A-Z)
        INCLUDE 'SYS$LIBRARY:UISENTRY'
        INCLUDE 'SYS$LIBRARY:UISUSRDEF'
        INCLUDE 'TOP_ENDU.FOR'
        IF ( ND_HELP .EQ. 1 ) THEN
           ND_HELP=0
           CALL SET_HELP
        ELSEIF ( ND_HELP .EQ. 0 ) THEN
           ND_HELP=1
           CALL SET_HELP
        ELSE
           ND_HELP=1
           CALL SET_HELP
        ENDIF
        RETURN
        END
C
C
C
```

B. SUBROUTINE ENDURANCE VARIABLES DECLARATIONS FILE

This file is INCLUDE'd in each of the subroutines of the ENDURANCE program to porovide commonality of variable declarations.

```
C*****THE DECLARATIONS FOR THE ENDURANCE CALCULTIONS SECTION IN ONE FILE
C
        CHARACTER ND_LINE*12,ND_INFILE*12
        CHARACTER ND_KB_UP(20)*12,ND_FNAME*12

        INTEGER ND_ENTERS,NLINE_NO,ND_NDX,ND_HELP

        REAL ND_RANGE,ND_SPD,ND_DISP,ND_PWR,ND_ELEC,ND_MFR
        REAL ND_EFR,ND_FUEL,ND_TPA,ND_WGT,ND_VOL,ND_PCT
        REAL ND_NDPWR,ND_OTH,ND_C_AVG,ND_RATIO,ND_PFRATE,ND_GFRATE
```

```
      REAL ND_TFR,ND_APFR,FRCORR,ND_SFR,ND_AVGFR,ND_TIME

      COMMON /NDUR_VARS1/ ND_TIMES,ND_KB_UP,ND_HELP
      COMMON /NDUR_VARS2/ ND_RANGE,ND_SPD,ND_DISP,ND_PWR,ND_ELEC,ND_MFR
      COMMON /NDUR_NAMES/ND_EFR,ND_FUEL,ND_TPA,ND_WGT,ND_VOL,ND_PCT
      COMMON /NDUR_RESULTS/ND_NDPWR,ND_OTH,ND_C_AVG,ND_RATIO,ND_PFRATE
      COMMON /NDUR_FILES/ ND_FNAME,ND_INFILE,ND_GFRATE,ND_TIME
      COMMON /NDUR_COEFF/ND_TFR,ND_APFR,FRCORR,ND_SFR,ND_AVGFR

C*****KEEP POWER RESULTS HANDY TO PASS IN INCASE WE NEED THEM
      COMMON /PW_RESULTS/ PW_SDNB,PW_SDBB,PW_ADME,PW_ADM
```

APPENDIX F. TOOL BOX ENDURANCE ESTIMATION REPORT

A. ENDURANCE ESTIMATION REPORT SOURCE CODE

This program contains the Source Code for the Endurance Estimation report. This code has been added to the overall TOOL BOX program REPORT.FOR, and can be called from the main Report menu.

```
      SUBROUTINE ENDU_RPT
C*********************************************************************
C                                                                    *
C    SUBROUTINE ENDU_RPT READS IN A STORED DATA FILE FROM THE         *
C    ENDURANCE SECTION AND PRINTS A DETAILED REPORT OF THE DESIGN    *
C    PARAMETERS AND THEIR UNITS.                                     *
C                                                                    *
C    CALLED BY SUBROUTINE 'REPORT'                                   *
C*********************************************************************
       INCLUDE 'TOP.FOR'
       INCLUDE 'TOP_ENDU.FOR'
       CHARACTER ND_INPUT_DATA*12, ND_PRFILE*12
C*****MOVE THE MOUSE POINTER AWAY FROM THE SELECTION
       STA = UIS$SET_POINTER_POSITION(VD_ID,WD_RPT,9.9,23.5)
C*****SET UP THE FIRST SCREEN INSTRUCTIONS
       CALL UIS$ERASE(VD_ID, 4.1,20.1,8.4,24.9)
       CALL UIS$TEXT(VD_ID,7,'ENTER THE FILE NAME',4.5,24.)
       CALL UIS$TEXT(VD_ID,7,'AND EXTENSION OF THE',4.5,23.6)
       CALL UIS$TEXT(VD_ID,7,'DATA FILE TO BE USED',4.5,23.2)
       CALL UIS$TEXT(VD_ID,7,'OR [RETURN] TO EXIT',4.5,22.8)
C*****READ THE DATA INPUT FILE NAME
       CALL KEY_READ (ND_INPUT_DATA,.FALSE.,5.,21.2,*30)
C*****NOW WE NEED THE NAME OF THE OUTPUT REPORT FILE
       CALL UIS$ERASE(VD_ID, 4.1,20.1,8.4,24.9)
       CALL UIS$TEXT(VD_ID,7,' ENTER THE FILE NAME',4.5,24.)
       CALL UIS$TEXT(VD_ID,7,' AND EXTENSION TO BE ',4.5,23.6)
       CALL UIS$TEXT(VD_ID,7,'ASSIGNED TO THE REPORT',4.5,23.2)
       CALL UIS$TEXT(VD_ID,7,' OR [RETURN] TO EXIT',4.5,22.8)
       CALL KEY_READ (ND_PRFILE,.FALSE.,5.,21.2,*30)
C*****READ THE DATA IN
 1     OPEN (23,FILE = ND_INPUT_DATA, STATUS = 'OLD')
       OPEN (24,FILE = ND_PRFILE, STATUS = 'NEW',BLANK='NULL')
       READ (23,100) ND_INFILE
       READ (23,110) ND_DISP
       READ (23,110) ND_PWR
       READ (23,110) ND_NDPWR
       READ (23,110) ND_ELEC
       READ (23,110) ND_TPA
       READ (23,110) ND_MFR
       READ (23,110) ND_EFR
       READ (23,110) ND_OTH
       READ (23,110) ND_RANGE
       READ (23,110) ND_SPD
      READ (23,110) ND_FUEL
      READ (23,110) ND_TIME
      READ (23,110) ND_VOL
      READ (23,110) ND_PCT
      REWIND(23)
      CLOSE (23)
C*****RE-CALCULATE FOR ACCURACY
C*****GENERATE THE COVER PAGE OF THE REPORT
      DO 5 I = 1,18,1
         WRITE (24,130) ' '
 5    CONTINUE
      WRITE (24,120) '                             TOOL BOX '
      WRITE (24,120) ' '
      WRITE (24,120) '                      ENDURANCE LIMIT REPORT'
      WRITE (24,120) '                      ***********************'
      WRITE (24,120) '             THIS REPORT WAS GENERATED USING THE
     &PROGRAM'
      WRITE (24,120) ' '
      WRITE (24,120) '              TOOL BOX WHICH WAS DEVELOPED FOR THE
     &NAVAL  '
      WRITE (24,120) '              --------'
      WRITE (24,120) '                    ENGINEERING DEPARTMENT OF
     &THE  '
      WRITE (24,120) ' '
      WRITE (24,120) '                      NAVAL POSTGRADUATE SCHOOL'
      WRITE (24,120) ' '
      WRITE (24,120) '                         MONTEREY, CALIFORNIA'
      WRITE (24,120) ' '
      WRITE (24,120) ' '
      WRITE (24,120) '                        PROFESSOR F. PAPOULIAS '
      WRITE (24,120) ' '
      WRITE (24,120) '                                AND  '
      WRITE (24,120) ' '
      WRITE (24,120) '                          LT. GERALD MCGOWAN '
      WRITE (24,120) '                          LT. JAMES PLOSAY'
      WRITE (24,120) '                               1989/90 '
      WRITE (24,120) ' '
      WRITE (24,120) ' '
      WRITE (24,120) ' '
      DO 6 I = 1,13,1
         WRITE (24,120) ' '
 6    CONTINUE
      WRITE (24,120) ' '
      WRITE (24,120) ' PAGE 1 OF 2 / ENDURANCE LIMIT'
      WRITE (24,120) ' '
C*****IN THIS SECTION, THE ECHO OF SHIP PARAMETERS AND CALCULATED
C     DATA IS PRESENTED.
C
      DO 10 I = 1,5,1
         WRITE (24,130) ' '
 10   CONTINUE
      WRITE(24,120) '                 ENDURANCE LIMIT REPORT '
      WRITE(24,130) ' '
      WRITE(24,120) '       THE INPUT SHIP PARAMETERS ARE AS FOLLOWS'
      WRITE(24,130) ' '
      WRITE(24,150) 'THE REPORT IS LOCATED IN FILE :          ',ND_PRFILE
      WRITE(24,150) 'THE INPUT DATA FILE USED IS :            ',
     &ND_INPUT_DATA
      WRITE(24,160) '*****************************************************'
      WRITE(24,130) '                    DESIGN PARAMETERS'
      WRITE(24,135) 'SHIP NAME IS ............................',ND_INFILE
      WRITE(24,130) ' '
      WRITE(24,140) 'DESIGN FULL LOAD DISPLACEMENT:..........',ND_DISP
      WRITE(24,130) '       (LTONS)'
      WRITE(24,140) 'DESIGN FULL POWER LEVEL:................',ND_PWR
      WRITE(24,130) '       (SHP)'
      WRITE(24,140) 'DESIGN ENDURANCE POWER LEVEL:...........',ND_NDPWR
      WRITE(24,130) '       (SHP)'
      WRITE(24,140) 'DESIGN CRUISING ELECTRIC LOAD:..........',ND_ELEC
      WRITE(24,130) '       (KW)'
      WRITE(24,140) 'DESIGN FUEL TANKS TAIL PIPE ALLOWANCE:..',ND_TPA
      WRITE(24,130) '       (%)'
      WRITE(24,140) 'DESIGN MAIN PROPULSION FUEL ECONOMY:....',ND_MFR
      WRITE(24,130) '       (LBS/SHP-HR)'
      WRITE(24,140) 'DESIGN ELECTRIC PLANT FUEL USE RATE:....',ND_EFR
      WRITE(24,130) '       (LBS/KW-HR)'
      WRITE(24,140) 'DESIGN OTHER FUEL USE RATES:............',ND_OTH
      WRITE(24,130) '       (LBS/HR)'
      WRITE(24,140) 'DESIGN RANGE:...........................',ND_RANGE
      WRITE(24,130) '       (NMILES)'
      WRITE(24,140) 'DESIGN CRUISING SPEED:..................',ND_SPD
      WRITE(24,130) '       (KNOTS)'
      WRITE(24,140) 'CALCULATED ENDURANCE FUEL LOAD:.........',ND_FUEL
      WRITE(24,130) '       (TONS-FUEL OIL)'
      WRITE(24,140) 'CALCULATED JOURNEY TIME ALLOWANCE:......',ND_TIME
      WRITE(24,130) '       (HRS @ DESIGN SPD & DISTANCE)'
      WRITE(24,140) 'CALCULATED FUEL STORAGE REQUIREMENTS:...',ND_VOL
      WRITE(24,130) '       (FT 3 VOLUMETRIC STORAGE)'
      WRITE(24,140) 'CALCULATED FUEL RATIO OF FULL LOAD:.....',ND_PCT
      WRITE(24,130) '       (% OF DESIGN FULL LOAD)'
      WRITE(24,130) ' '
      WRITE(24,130) ' '
      WRITE(24,130) ' '
      WRITE(24,130) ' '
      WRITE(24,130) '*****************************************************'
      WRITE(24,130) 'ENDURANCE LIMIT VALUES PRESENTED ARE BASED UPON'
      WRITE(24,130) 'FOLLOWING: '
      WRITE(24,130) ' (1) U.S. NAVY DESIGN DATA SHEET DDS9400-1 FORMAT'
      WRITE(24,130) '     CALCULATIONS USING ESTIMATION FACTORS FOR '
      WRITE(24,130) '     FOULING, MACHINERY INEFFICIENCIES, ETC'
      WRITE(24,130) '*****************************************************'
      WRITE (24,120) ' '
      WRITE (24,120) ' PAGE 2 OF 2 / ENDURANCE LIMIT'
      CLOSE (24)
100   FORMAT (A)
110   FORMAT (G12.5)
120   FORMAT (10X,A)
130   FORMAT (10X,A)
135   FORMAT (10X,A,'..........',A)
140   FORMAT (10X,A,'..........',G14.6)
150   FORMAT (10X,2A)
```

```
 160  FORMAT (10X,A)
C*****EXIT FROM THE REPORT AND ENABLE MOUSE
 30   CALL UIS$ERASE(VD_ID, 4.1,20.1,8.4,24.9)
      CALL UIS$TEXT(VD_ID,7,' SELECT AN OPTION WITH',4.5,24.)
      CALL UIS$TEXT(VD_ID,7,'      THE MOUSE',4.5,23.6)
      RETURN
      END
```

APPENDIX G. OTHER SOURCE CODE

A. TOOL BOX MAIN PROGRAM

This source code is presented for completeness, although much of it belongs to the original author, [Ref. 3], it is still included here since some of the UIS function calls had to be changed to support the new Modules added in this development.

```
      PROGRAM TOOLBOX
C     VERSION B18.01
C     VERSION DATE: 09/03/90
C*************************************************************************
C     THE GOAL OF VERSION 18 IS TO ADD IN THE SCREEN DISPLAY PLOT
C     TO THE PROPULSIVE POWER SECTION, AND TO CLEAN UP THE REMAINING
C     CODE SECTIONS FOR FINALIZATION
C
C     REVISION HISTORY:
C       VERSION B13.6 : AS RECEIVED FROM G.MCGOWAN
C       VERSION B14.0 : ADDED SEVERAL HELP WINDOWS TO EXISTING CODE
C                     : FINALIZED IN B14.03
C       VERSION B15.0 : ADDED POWER PREDICTION MODULE
C                     : FINALIZED IN B15.88
C       VERSION B16.0 : ADDED POWER PREDICTION REPORT
C                     : FINALIZED IN B16.09
C       VERSION B17.0 : ADDED ENDURANCE MODULE
C                     : ADDED USER SELECTION OF HELP LEVEL
C                     : ADDED ENDURANCE ESTIMATION REPORT
C                     : FINALIZED IN B17.42
C       VERSION B18.0 : ADDED POWER GRAPH TO SCREEN OPTION
C                     : FINALIZED IN B18.12
C*************************************************************************
C*****GENERAL HEADER INFORMATION
        INCLUDE 'GENERAL.FOR/LIST'
        EXTERNAL NOWHERE,REPORT,POWER,ENDURANCE
        EXTERNAL DARK1,DARK2,LIGHT1,STATIC,EXIT,MANEUVER,UTIL
        REAL DY2
C
C*****SINCE THE SUBROUTINES ARE PASSED AS ARGUMENTS IN THE AST ROUTINES
C     THEY MUST BE CALLED IN EXTERNAL STATEMENTS
C
C*****CREATE THE VIRTUAL DISPLAY WITH TITLES
C
C       FIRST, WE SHRINK THE CALLING WINDOW.
C
C       CALL LIB$SPAWN('SET TERMINAL/WIDTH=5')
C       CALL LIB$SPAWN('SET TERMINAL/PAGE=2')
C*****THEN CREATE A NEW VIRTUAL DISPLAY
        VD_ID=UIS$CREATE_DISPLAY(-1.,-1.,20.,25.5,40.0,30.0)
C*****AND ADD COLORS TO IT
C*****THESE TWO LINES COMMENTED OUT TO ACHIEVE GOOD B&W SCREEN COPIES
C     FOR OVERHEADS.  REMOVE COMMENTS IN FINAL VERSION
C       CALL UIS$SET_COLOR(VD_ID,1,.75,1.,1.)
```

```
C       CALL UIS$SET_COLOR(VD_ID,0,0.,0.,.5)
C******SET UP SOME CHARACTER TYPES
        CALL UIS$SET_WRITING_MODE(VD_ID,0,1,UIS$C_MODE_REPLN)
        CALL UIS$SET_WRITING_MODE(VD_ID,1,2,UIS$C_MODE_REPL)
        CALL UIS$SET_WRITING_MODE(VD_ID,0,4,UIS$C_MODE_REPLN)
        CALL UIS$SET_CHAR_SIZE(VD_ID,4,5,,.138,.39)
        CALL UIS$SET_CHAR_SIZE(VD_ID,0,7,,.125,.3)
C******INITIALIZE
        X0 = 0.0
        X1 = 3.0
        Y_LINE = .7
        FL1 = .FALSE.
        FL2 = .FALSE.
        YPOS = 1
C******DRAW THE MAIN MENU
C******THIS ROUTINE DRAWS THE TITLE BOXES FOR THE MAIN MENU
        DO 10 Y_COOR = 0.2,5.8,.8
           DY = Y_COOR + .4
           CALL UIS$PLOT(VD_ID,0,0.,Y_COOR,3.,Y_COOR,3.,DY,0.,
     &         DY,0.,Y_COOR)
 10     CONTINUE
C******SET TITLES FOR MAIN MENU BLOCKS
        BLOCK(1) = '     RESPONSE AREA      '
        BLOCK(2) = '     INSTRUCTIONS        '
        BLOCK(3) = '     OPTIONS MENU        '
        BLOCK(4) = '  '
        BLOCK(5) = '     SELECTIONS         '
        BLOCK(6) = ' '
        BLOCK(7) = ' '
        BLOCK(8) = ' '
        BLOCK(9) = ' '
        BLOCK(10) = ' '
        BLOCK(11) = ' '
        LONG_TITLE(1) = '      DATA DISPLAY AREA    '
        LONG_TITLE(2) = ' '
        LONG_TITLE(3) = ' '
        LONG_TITLE(4) = ' '
        LONG_TITLE(5) = ' '
C******SET TITLES FOR OPTIONS BLOCKS
        OPTION(1) = '  EXIT THE PROGRAM'
        OPTION(2) = '   GENERATE REPORT'
        OPTION(3) = '   FILE UTILITIES'
        OPTION(4) = '   RESERVE MODULE'
        OPTION(5) = '   ENDURANCE'
        OPTION(6) = '   POWER PREDICTION'
        OPTION(7) = '   MANEUVERING'
        OPTION(8) = '   STATIC STABILITY'
        OPTION(9) = '   EXIT THE PROGRAM'
        OPTION(10) = '  RETURN TO MAIN'
        OPTION(11) = ' '
        OPTION(12) = ' '
        OPTION(13) = '  STORE DISPLAYED DATA'
        OPTION(14) = '  INPUT DATA FROM FILE'
        OPTION(15) = '  ENTER DATA FROM KB'
        OPTION(16) = '  RUN TURNING CIRCLE '
        OPTION(17) = '  RUN DYNAMIC PERFORMANCE '
        OPTION(18) = '  LONGITUDINAL STABILITY  '
        OPTION(19) = '  INITIAL TRANS. STABILITY'
C*****ADD OPTIONS FOR VERSIONS 15 & 18
        OPTION(20) = '  PLOT GRAPH TO SCREEN  '
        OPTION(21) = '  PLOT GRAPH TO DISK  '
C*****ADD OPTIONS FOR VERSION 17
        OPTION(22) = '  CALC ENDURANCE LIMIT'
        OPTION(23) = '  CHANGE HELP LEVEL ON/OFF'
C*****NOW FILL THE BLOCKS OF THE MAIN MENU WITH THE STORED TITLES
        DO 20 I = 1,8,1
          CALL UIS$TEXT(VD_ID,0,OPTION(I),.3,Y_LINE)
          Y_LINE = Y_LINE + .8
 20     CONTINUE
C*****TURN ON THE MAIN MENU VIEWPORT
        WD_MAIN=UIS$CREATE_WINDOW(VD_ID,'SYS$WORKSTATION','MAIN MENU',
     &  -.5,-.5,10.,7.,40.,30.)
C*****SET UP THE AST'S FOR HIGHLIGHTING
        DO 50 Y_COOR = .2,5.8,.8
          DY = Y_COOR + .2
          CALL UIS$SET_POINTER_AST(VD_ID,WD_MAIN,DARK1,,X0,Y_COOR,X1,
     &    DY,LIGHT1)
 50     CONTINUE
C*****SET MAIN MENU AST TRAPS FOR MOUSE BUTTONS
        CALL UIS$SET_BUTTON_AST(VD_ID,WD_MAIN,EXIT,,X0,.2,X1,.6)
        CALL UIS$SET_BUTTON_AST(VD_ID,WD_MAIN,REPORT,,X0,1.0,X1,1.4)
        CALL UIS$SET_BUTTON_AST(VD_ID,WD_MAIN,UTIL,,X0,1.8,X1,2.2)
        CALL UIS$SET_BUTTON_AST(VD_ID,WD_MAIN,NOWHERE,,X0,2.6,X1,3.0)
        CALL UIS$SET_BUTTON_AST(VD_ID,WD_MAIN,ENDURANCE,,X0,3.4,X1,3.8)
        CALL UIS$SET_BUTTON_AST(VD_ID,WD_MAIN,POWER,,X0,4.2,X1,4.6)
        CALL UIS$SET_BUTTON_AST(VD_ID,WD_MAIN,MANEUVER,,X0,5.0,X1,5.4)
        CALL UIS$SET_BUTTON_AST(VD_ID,WD_MAIN,STATIC,,X0,5.8,X1,6.6)
C*****PUT IN THE TITLE AND AUTHORS BLOCK TEXT
        INCLUDE 'HEADER.FOR/LIST'
C*****DRAW THE STATIC/MANEUVERING/POWER/ENDURANCE INTERACTIVE DISPLAY
        CALL UIS$PLOT(VD_ID,0,-.5,10.,10.,10.,10.,18.8,-.5,
     &                 18.8,-.5,10.)
        CALL UIS$LINE(VD_ID,0,3.5,10.,3.5,18.8,-.5,11.,3.5,11.,
     &    -.5,11.4,3.5,11.4,-.5,12.4,3.5,12.4,-.5,12.8,3.5,12.8,
     &    -.5,18.4,10.,18.4)
        DO 30 Y_COOR = 13.  , 18.4, .8
           DY = Y_COOR + .4
           DY2 = DY +.1
           Z = 9 + INT((Y_COOR - 12.9)/.8)
           CALL UIS$TEXT(VD_ID,0,OPTION(Z),.3,DY2)
           CALL UIS$PLOT(VD_ID,0,X0,Y_COOR,X1,Y_COOR,X1,DY,X0,
     &     DY,X0,Y_COOR)
 30     CONTINUE
C*****FILL THE TITLE BLOCKS OF WORKING DISPLAY
        CALL UIS$TEXT(VD_ID,5,BLOCK(1),-.5,12.85)
        CALL UIS$TEXT(VD_ID,5,BLOCK(2),-.5,11.4)
        CALL UIS$TEXT(VD_ID,5,BLOCK(3),-.5,18.8)
        CALL UIS$TEXT(VD_ID,5,LONG_TITLE(1),4.,18.8)
C*****THIS LINE CREATES A HARCOPY METAFILE WHEN RUN
C       CALL HCUIS$WRITE_DISPLAY(VD_ID,'HARD.UIS')
C*****NOW DRAW UP THE REPORT SCREEN
```

```
        CALL RPT_GRAPH
C*****HAVE SYSTEM WAIT FOR USER ACTION
        CALL SYS$HIBER()
40      END
C
C
C*****EACH MAIN SUBROUTINE IS AN 'INCLUDE' FILE TO EASE EDITING AND
C      MODIFICATIONS
        INCLUDE 'REPORT.FOR/LIST'
        INCLUDE 'MANUEVER.FOR/LIST'
        INCLUDE 'STATIC.FOR/LIST'
        INCLUDE 'POWER.FOR/LIST'
        INCLUDE 'ENDURANCE.FOR/LIST'
        END
C
C
        SUBROUTINE DARK1
C***********************************************************************
C                                                                      *
C    THIS SUBROUTINE TURNS THE MENU ITEM IN THE BOX TO REVERSE VIDEO  *
C    WHEN THE MOUSE POINTER ENTERS THE REGION DEFINED INSIDE THE      *
C    BOX.  IT IS USED TO INDICATE TO THE USER THAT THE SELECTION IS   *
C    ABLE TO BE SELECTED SINCE THEY ARE IN THE REGION                 *
C                                                                      *
C***********************************************************************
C
        INCLUDE 'GENERAL.FOR'
C*****TEST TO SEE IF WE HAVE BEEN HERE BEFORE AND DONE THE JOB.
        POSI1 = UIS$GET_POINTER_POSITION(VD_ID,WD_MAIN,XPOS,YPOS)
        Z = INT(YPOS/.8)+1
        Y0 = .8 * INT(YPOS/.8) + .4
        Y1 = Y0 + .3
        IF (FL1) RETURN
        FL1 = .TRUE.
C*****TURN OFF THE LIGHTS
        CALL UIS$TEXT(VD_ID,1,OPTION(Z),.3,Y1)
        RETURN
        END
C
C
        SUBROUTINE LIGHT1
C***********************************************************************
C                                                                      *
C    THIS SUBROUTINE TURNS THE MENU ITEM IN THE BOX TO REVERSE VIDEO  *
C    WHEN THE MOUSE POINTER ENTERS THE REGION DEFINED INSIDE THE      *
C    BOX.  IT IS USED TO INDICATE TO THE USER THAT THE SELECTION IS   *
C    EBLE TO BE SELECTED SINCE THEY ARE IN THE REGION                 *
C                                                                      *
C***********************************************************************
C
        INCLUDE 'GENERAL.FOR'
        FL1 = .FALSE.
C*****TURN ON THE LIGHTS
        CALL UIS$TEXT(VD_ID,2,OPTION(Z),.3,Y1)
        RETURN
        END
C
C
        SUBROUTINE DARK2
C***********************************************************************
C                                                                      *
C    THIS SUBROUTINE TURNS THE MENU ITEM IN THE BOX TO REVERSE VIDEO  *
C    WHEN THE MOUSE POINTER ENTERS THE REGION DEFINED INSIDE THE      *
C    BOX.  IT IS USED TO INDICATE TO THE USER THAT THE SELECTION IS   *
C    ABLE TO BE SELECTED SINCE THEY ARE IN THE REGION                 *
C                                                                      *
C***********************************************************************
C
       INCLUDE 'GENERAL.FOR'
C  TEST TO SEE IF WE HAVE BEEN HERE BEFORE AND DONE THE JOB.
       POSI2 = UIS$GET_POINTER_POSITION(VD_ID,WNDOW,XPOS,YPOS)
       Z2 = INT(YPOS/.8)-7
       Y0 = .8 * INT(YPOS/.8) + .3
       Y1 = Y0 + .4
       IF (FL2) RETURN
       FL2 = .TRUE.
C******TURN OFF THE LIGHTS
       CALL UIS$TEXT(VD_ID,1,OPTION(Z2),.3,Y1)
       RETURN
       END
C
C
        SUBROUTINE LIGHT2
C***********************************************************************
C                                                                      *
C    THIS SUBROUTINE TURNS THE MENU ITEM IN THE BOX TO REVERSE VIDEO  *
C    WHEN THE MOUSE POINTER ENTERS THE REGION DEFINED INSIDE THE      *
C    BOX.  IT IS USED TO INDICATE TO THE USER THAT THE SELECTION IS   *
C    ABLE TO BE SELECTED SINCE THEY ARE IN THE REGION                 *
C                                                                      *
C***********************************************************************
C
        INCLUDE 'GENERAL.FOR'
C******TELL THE WORLD WE ARE LEAVING
        IF (Z2 .LE. .01) RETURN
        FL2 = .FALSE.
C******TURN ON THE LIGHTS
        CALL UIS$TEXT(VD_ID,2,OPTION(Z2),.3,Y1)
        RETURN
        END
C
C
        SUBROUTINE EXIT
C***********************************************************************
C                                                                      *
C    THIS SUBROUTINE IS THE GLOBAL SYSTEM 'EXIT' ROUTINE SELECTED     *
C    FROM THE MAIN OR ANY SUB-MENU TO EXIT THE SYSTEM AND RETURN      *
C    TO THE CALLING PROCESS                                           *
C                                                                      *
C***********************************************************************
C
```

```
C*****MAKE THE DISPLAY BIG AGAIN SINCE WE SHRUNK IT TO START OFF
C
        CALL LIB$SPAWN('SET TERMINAL/WIDTH=80')
        CALL LIB$SPAWN('SET TERMINAL/PAGE=24')
        CALL SYS$WAKE(,)
        RETURN
        END
C
C
        SUBROUTINE NOWHERE
C************************************************************************
C                                                                       *
C      THIS SUBROUTINE IS CALLED BY A MOUSE BUTTON SELECTION OF AN     *
C      OPTION THAT IS NOT ENABLED YET.  IT POPS UP A DIALOG 'HELP'     *
C      BOX THAT INSTRUCTS THE USER AND GIVES THEM THE OPTION OF        *
C      CONTINUING BACK TO THE PROGRAM AT THE POINT OF INTERRUPTION     *
C      OR ENDING THE PROGRAM ENTIRELY.                                 *
C                                                                       *
C************************************************************************
C
      IMPLICIT INTEGER(A-Z)
      INCLUDE 'SYS$LIBRARY:UISENTRY'
      INCLUDE 'SYS$LIBRARY:UISUSRDEF'
      CHARACTER*34 HINT1/'*** THIS FEATURE NOT AVAILABLE ***'/
      CHARACTER*34 HINT2/'    TYPE "CONTINUE" TO RESUME     '/
      CHARACTER*34 HINT3/'    OR "EXIT" TO END PROGRAM      '/
      REAL Y_POSN
C*****CREATE THE DISPLAY WINDOW FOR THE DIALOG BOX
      VD_TEST=UIS$CREATE_DISPLAY(0.0,0.0,10.0,5.0,20.0,5.0)
C*****CREATE THE WINDOW TO DISPLAY THE HELP TEXT
      WD_TEST=UIS$CREATE_WINDOW(VD_TEST,'SYS$WORKSTATION','HELP WINDOW')
C*****SET THE SCREEN FONTS WE DESIRE FROM ATTRIBUTE BLOCK '0' TO BLOCK
C     '20' AND CALL THE FILENAME OF THE FONT
      CALL UIS$SET_FONT(VD_TEST,0,20,'DTABER0R03WK00GG0001UZZZZ02A000')
C*****SIGNAL THE USER ON PROCESS PAUSE FOR DISPLAY OF HELP BOX
      CALL UIS$SOUND_BELL('SYS$WORKSTATION',4)
C*****PRINT THE TEXT AS WE WANT IT
        Y_POSN=4.0
        CALL UIS$TEXT(VD_TEST,20,HINT1,0.4,Y_POSN)
        Y_POSN=Y_POSN-1
        CALL UIS$TEXT(VD_TEST,20,HINT2,0.4,Y_POSN)
        Y_POSN=Y_POSN-1
        CALL UIS$TEXT(VD_TEST,20,HINT3,0.4,Y_POSN)
      PAUSE
C*****SIGNAL THE USER PROCESS RESTARTED
      CALL UIS$SOUND_BELL('SYS$WORKSTATION',4)
C*****RETURN TO CALLING PROCESS AT POINT OF INTERRUPT
      CALL UIS$DELETE_DISPLAY(VD_TEST)
      RETURN
      END
C
C
        SUBROUTINE TO_MAIN
C************************************************************************
C                                                                       *
C      THIS SUBROUTINE IS CALLED TO RETURN TO THE MAIN MENU FROM A     *
```

```
C      SUB-MENU.                                                         *
C                                                                        *
C*************************************************************************
C
         INCLUDE 'GENERAL.FOR'
C*****DISABLE THE AST'S
         STA=UIS$SET_POINTER_POSITION(VD_ID,Wndow,9.7,14.)
         CALL LIGHT2
         CALL UIS$SET_BUTTON_AST(VD_ID,WNDOW,,,X0,13.0,X1,13.4)
         CALL UIS$SET_BUTTON_AST(VD_ID,WNDOW,,,X0,13.8,X1,14.2)
         CALL UIS$SET_BUTTON_AST(VD_ID,WNDOW,,,X0,14.6,X1,15.0)
         CALL UIS$SET_BUTTON_AST(VD_ID,WNDOW,,,X0,15.4,X1,15.8)
         CALL UIS$SET_BUTTON_AST(VD_ID,WNDOW,,,X0,16.2,X1,16.6)
         CALL UIS$SET_BUTTON_AST(VD_ID,WNDOW,,,X0,17.0,X1,17.4)
         CALL UIS$SET_BUTTON_AST(VD_ID,WNDOW,,,X0,17.8,X1,18.2)
C*****NOW CALL UP MAIN MENU
         CALL UIS$DELETE_WINDOW(WNDOW)
         RETURN
         END
C
C
      SUBROUTINE SHOW_SAVE
C************************************************************************
C                                                                       *
C     THIS SUBROUTINE POPS UP A DIALOG BOX TO INSTRUCT THE USER NOT    *
C     TO USE THE SAME DATA FILE NAME AS USED IN A PREVIOUS PROGRAM     *
C     SECTION [SINCE IT WILL OVERWRITE THE OLD ONE CAUSING ERRORS      *
C     OR LOST DATA].  IT IT CALLED BY THE SUBROUTINE THAT SAVES THE    *
C     DATA FILES                                                        *
C                                                                       *
C************************************************************************
C
      IMPLICIT INTEGER(A-Z)
      INCLUDE 'SYS$LIBRARY:UISENTRY'
      INCLUDE 'SYS$LIBRARY:UISUSRDEF'
      CHARACTER*34 SAVE/'*** SAVING DEFINED DATA FILE  *** '/
      CHARACTER*34 SAVE2/' DO NOT USE SAME NAME & EXTENSION '/
      CHARACTER*34 SAVE3/'AS DATA FILE FOR OTHER SUB-SECTION'/
      CHARACTER*34 SAVE4/'PLEASE WAIT........................'/
      CHARACTER*34 SAVE5/'...............PROGRAM WILL RESUME'/
      REAL Y_POSN
C*****CREATE THE DISPLAY FOR THE DIALOG BOX
      VD_SAVE=UIS$CREATE_DISPLAY(0.0,0.0,10.0,5.0,20.0,6.0)
C*****CREATE THE WINDOW TO DISPLAY THE TEXT IN
      WD_SAVE=UIS$CREATE_WINDOW(VD_SAVE,'SYS$WORKSTATION','HELP WINDOW')
C*****COPY ATTRIBUTE BLOCK '0' AS BLOCK '20' AND CHANGE THE FONT SIZE
      CALL UIS$SET_FONT(VD_SAVE,0,20,'DTABER0R03WK00GG0001UZZZZ02A000')
C*****COPY ATTRIBUTE BLOCK '20' AS BLOCK '21' AND CHANGE TO BOLD FONT
      CALL UIS$SET_FONT(VD_SAVE,20,21,'DTABER0R03WK00PG0001UZZZZ02A000')
C*****SIGNAL THE USER TO GET THEIR ATTENTION
      CALL UIS$SOUND_BELL('SYS$WORKSTATION',4)
C*****WRITE THE TEXT INTO THE WINDOW AND SPAWN PROCESS TO 'WAIT' 15 SEC
         Y_POSN=4.0
         CALL UIS$TEXT(VD_SAVE,21,SAVE,0.2,Y_POSN)
         Y_POSN=Y_POSN-.8
         CALL UIS$TEXT(VD_SAVE,20,SAVE2,0.2,Y_POSN)
         Y_POSN=Y_POSN-.8
         CALL UIS$TEXT(VD_SAVE,20,SAVE3,0.2,Y_POSN)
         Y_POSN=Y_POSN-.8
         CALL UIS$TEXT(VD_SAVE,21,SAVE4,0.2,Y_POSN)
         Y_POSN=Y_POSN-.8
         CALL UIS$TEXT(VD_SAVE,21,SAVE5,0.2,Y_POSN)
      CALL LIB$SPAWN('WAIT 00:00:15')
      CALL UIS$SOUND_BELL('SYS$WORKSTATION',4)
      CALL UIS$DELETE_DISPLAY(VD_SAVE)
      RETURN
      END
C
C
      SUBROUTINE UTIL
*********************************************************************
*                                                                   *
*     UTILITY IS A SUBROUTINE WHICH ALLOWS THE USER                 *
*     TO SHELL BACK TO THE CALLING WINDOW WITHOUT LEAVING THE       *
*     PROGRAM ENVIRONMENT. THIS ALLOWS A USER TO RUN A DCL          *
*     COMMAND ( SAY A 'DIR' ) WITHOUT LEAVING THE PROGRAM.          *
*     WHEN FINISHED, THE USER ENTERS THE COMMAND 'EXIT' OR          *
*     A RETURN ON A BLANK LINE AND CONTROL RETURNS TO THE TOOL      *
*     BOX MENU.                                                     *
*                                                                   *
*********************************************************************
      INCLUDE 'GENERAL.FOR'
      CHARACTER*64  CMMD
      STA=UIS$SET_POINTER_POSITION(VD_ID,WD_MAIN,9.9,5.)
C******RETURN THE SCREEN TO DEFAULT SIZE
      CALL LIB$SPAWN('SET TERMINAL/WIDTH=80')
      CALL LIB$SPAWN('SET TERMINAL/PAGE=24')
C******PUSH THAT SCREEN TO THE FOREFRONT
      CALL UIS$PUSH_VIEWPORT(WD_MAIN)
 1    TYPE *, 'ENTER THE DCL COMMAND ...OR [RETURN] TO EXIT'
      PRINT *,' '
      READ (*,FMT='(BN,A)')  CMMD
      PRINT *, CMMD
C******IF COMMAND IS A 'RETURN', XFER BACK TO 'TOOL BOX'
      IF (CMMD .EQ. ' ')  THEN
         CALL UIS$POP_VIEWPORT(WD_MAIN)
         RETURN
      ENDIF
C******SPAWN A NEW PROCESS FOR THE ENTERED COMMAND
      CALL LIB$SPAWN(CMMD)
      PRINT *,' '
      GOTO 1
      END
C
C
```

B. TOOL BOX MAIN PROGRAM VARIABLE DECLARATIONS

This file GENERAL.FOR is INCLUDE'ed in each subroutine of the main core code to provide commonality of variable names and types.

```
C******GENERAL.FOR*****************************************************************
C     GENERAL HEADER INFORMATION
C**********************************************************************************
        IMPLICIT INTEGER(A-Z)
        INCLUDE 'SYS$LIBRARY:UISENTRY'
        INCLUDE 'SYS$LIBRARY:UISUSRDEF'
        COMMON /KB/ KB_ID
        COMMON /DISPLAY_1/ VD_ID,WD_MAIN,WD_STATIC,WD_MANU,WNDOW,VD_TEST
        COMMON /DISPLAY_2/ WD_RPT,WD_POWER,WD_FAIL,VD_FAIL,VD_SCR
        COMMON /DISPLAY_3/ WD_NDUR,WD_TM,WD_MTM,WD_SCR
        COMMON /DISPLAY_4/ WD_SH,VD_SH
        REAL*4 X0,Y0,X1,Y1,XPOS,YPOS,Y_COOR,Y_LINE,DY
        COMMON /REAL_N/ X0,X1,Y0,Y1,Y_COOR,Y_LINE,DY,YPOS
        COMMON /INTEG_R/ Z,Z2
        LOGICAL*1 FL1,FL2,KEYBUF(4)
        COMMON /LOGI/FL1,FL2
        CHARACTER*26 BLOCK*29, OPTION, LONG_TITLE*40
        COMMON /CHAR/ BLOCK(11),OPTION(23),LONG_TITLE(5)
```

C. TOOL BOX MAIN SUB-PROGRAM VARIABLE DECLARATIONS

This file TOP.FOR is INCLUDE'ed in each of the subroutines throughout the program that require commonality of only a few of the main program window and device variable declarations.

```
*********************** TOP.FOR ***********************************************
        IMPLICIT INTEGER (A-Z)
        INCLUDE 'SYS$LIBRARY:UISENTRY'
        INCLUDE 'SYS$LIBRARY:UISUSRDEF'
        COMMON /KB/ KB_ID
        COMMON /DISPLAY_1/ VD_ID,WD_MAIN,WD_STATIC,WD_MANU,WNDOW,VD_TEST
        COMMON /DISPLAY_2/ WD_RPT,WD_POWER,WD_FAIL,VD_FAIL,VD_SCR
        COMMON /DISPLAY_3/ WD_NDUR,WD_TM,WD_MTM,WD_SCR
        COMMON /DISPLAY_4/ WD_SH,VD_SH
```

LIST OF REFERENCES

1. Coombs, M.J. and Alty, J.L., *Computing Skills and the User Interface*, 1st Edition, Academic Press, 1981.

2. Smith, D.K. and Alexander, R.C., *Fumbling the Future: How XEROX Invented, then Ignored, the First Personal Computer*, 1st Edition, William Morrow and Co., Inc., 1988.

3. McGowan, G.K., *Application of VAX/VMS Graphics for Solving Preliminary Ship Design Problems*, Master's Thesis, Naval Postgraduate School, Monterey, California, March, 1990.

4. Taggart, R., editor, *Ship Design and Construction* , published by the Society of Naval Architects and Marine Engineers, New York, N.Y., 1980

5. Digital Equipment Corporation, *MicroVMS Programmer's Manual*, April, 1986.

6. Taylor, D.W., *The Speed and Power of Ships*, 2nd Edition, United States Maritime Commission, Washington, D.C., 1943

7. Harvald, Sv.Aa., *Resistance and Propulsion of Ships*, John Wiley & Sons, Inc., New York, New York, 1983

8. Watson, D.G.M., response to *The Effective Horsepower of Single Screw Ships* by D. I. Moor and V. F. Small, contained in *Quarterly Transactions*, Royal Institute of Naval Architects, London, United Kingdom, July 1960, Vol 102, No. 3

9. Silverleaf, A. and Dawson, J., *Hydrodynamic Design of Merchant Ships for High Speed Operation*, Royal Institute of Naval Architects, London, United Kingdom, 1967

10. Erichsen, S., Report Number 123, *Optimum Capacity of Ships and Port Terminals*, Department of Naval Architecture and Naval Engineering, College of Engineering, University of Michigan; April, 1971

11. Harrington, R., editor, *Marine Engineering* , published by the Society of Naval Architects and Marine Engineers, New Yrok, N.Y., 1971

12. Baker, E., *Introduction to Steel Shipbuilding*, 2nd Edition, McGraw-Hill, New York, N.Y., 1953

13. U.S. Department of the Navy, *Calculation of Surface Ship Endurance Requirements*, Design Data Sheet, DDS9400-1, Washington D.C., 1963

BIBLIOGRAPHY

The following articles concerning the rise to predominance of the Graphical User Interface (GUI) are particularily well written as an introduction to the subject :

Hayes, F. and Baran, N., "A Guide to GUI's", *BYTE* , Vol. 14 Number 7, July 1989.

Seymour, J., "The GUI: An Interface You Won't Outgrow", *PC Magazine* , Vol. 8 Number 15, 12 September 1989.

General information on ship design and Naval Architecture can be found in the following references :

Munro-Smith, R., *Ships and Naval Architecture (SI Units)* ,The Institute of Marine Engineers, London, U.K., 1973

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Department Chairman, Code ME Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93943-5000	1
4.	Naval Engineering Curricular Office, Code 34 Naval Postgraduate School Monterey, California 93943-5000	1
5.	Professor Fotis A. Papoulias, Code Me/Pa Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93943-5000	2
6.	Professor James F. Hallock, Code ME/Hl Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93943-5000	2
7.	Professor Robert E. Ball, Code AA/Bp Department of Aeronautics and Astronautics Naval Postgraduate School Monterey, California 93943-5000	1
8.	Gerald K. McGowan, LT/USN c/o Avis Grindstaff P.O. Box 112 Dent, Minnesota 56520	1
9.	James R. Plosay, LT/USN 4110 Oxford Rd. York Springs, Pennsylvania 17372	2