AN INVESTIGATION INTO THE USE OF DATA BASES IN COMPUTER-AIDED ETC (U)

JUN 81 R C CELOTTO
**Title:** An Investigation into the Use of Data Bases in Computer-Aided Naval Ship Design (Vol I & II)

**Author:** Celotto, Richard C.

**Performing Organization Name and Address:** Mass. Inst. of Technology, Cambridge, MA 02139

**Controlling Office Name and Address:** Code 031, Naval Postgraduate School, Monterey, CA 93940

**Security Classification (of this report):** Unclassified

**Distribution Statement (of this Report):** Approved for public release; distribution unlimited

**Distribution Statement (of the abstract entered in Block 20, if different from Report):**

**Supplementary Notes:**

**Key Words:** Naval ship design, computer-aided ship design

**Abstract:**

**Attached:**

**DTIC File Copy**
A design-oriented, interactive computer system which makes possible the dynamic loading of programs at the user's request throughout the operating session has been developed. This system, which is referred to as DEX, also allows the user to select various types of files as the source and destination of information during the session. With respect to one type of file, databases, DEX introduces a more versatile form of organization and use.

An extended DEX library of subroutines is developed which enables the user to read and write integer scalar, real scalar and one-dimensional real array variables and to edit from the terminal integer and real scalar values. It also enables the user to employ during input and output sequences the unit system of his choice.

A proposal is offered for the organization of DEX databases for the preliminary design of naval ships. Suggestions are made, based on a demonstration computer program, for employing existing ship databases to support a generalized ship synthesis model.
AN INVESTIGATION
INTO THE USE OF DATA BASES
IN COMPUTER-AIDED NAVAL SHIP DESIGN
by
RICHARD CHARLES CELOTTO
Lieutenant, United States Navy
B.S., Webb Institute of Naval Architecture
and Marine Engineering
(1973)
SUBMITTED TO THE DEPARTMENT OF
OCEAN ENGINEERING
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS OF THE DEGREES OF
OCEAN ENGINEER
and
MASTER OF SCIENCE IN
NAVAL ARCHITECTURE AND MARINE ENGINEERING
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
June 1981
© Richard Charles Celotto 1981
The author hereby grants to M.I.T. permission to reproduce and
to distribute copies of this thesis document in whole or in part.
Signature of Author
Certified by C. Chryssostomides, Thesis Supervisor
Accepted by A. Douglas Carmichael
Chairman, Departmental Graduate Committee
AN INVESTIGATION
INTO THE USE OF DATA BASES
IN COMPUTER-AIDED NAVAL SHIP DESIGN

by
RICHARD CHARLES CELOTTI

Submitted to the Department of Ocean Engineering
on May 8, 1981 in partial fulfillment of
the requirements for the degrees of
Master of Science in
Naval Architecture and Marine Engineering
and
Ocean Engineer

ABSTRACT

A design-oriented, interactive computer system which
makes possible the dynamic loading of programs at the
user’s request throughout the operating session has been
developed. This system, which is referred to as DEX, also
allows the user to select various types of files as the
source and destination of information during the session.
With respect to one type of file, databases, DEX introduces
a more versatile form of organization and use.

An extended DEX library of subroutines is developed
which enables the user to read and write integer scalar,
real scalar and one-dimensional real array variables and
to edit from the terminal integer and real scalar values.
It also enables the user to employ during input and output
sequences the unit system of his choice.

A proposal is offered for the organization of DEX
databases for the preliminary design of naval ships.
Suggestions are made, based on a demonstration computer
program, for employing existing ship databases to support
a generalized ship synthesis model.

Thesis Supervisor: Chryssostomos Chryssostomidis
Title: Associate Professor of Ocean Engineering
ACKNOWLEDGEMENTS

The author wishes to express his appreciation to Professor Chryssostomos Chryssostomidis for his guidance, advice and unfailing encouragement which made this thesis possible. The author is especially grateful for the generous sharing of his time throughout the undertaking.

The author would also like to thank his lovely fiancée, Kathy, for her patience and perspective.
TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>2</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>3</td>
</tr>
<tr>
<td>TABLE OF CONTENTS</td>
<td>4</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>8</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>9</td>
</tr>
<tr>
<td>CHAPTER 1    INTRODUCTION TO DEX.</td>
<td>10</td>
</tr>
<tr>
<td>1.1 Background</td>
<td>10</td>
</tr>
<tr>
<td>1.2 Description of DEX.</td>
<td>12</td>
</tr>
<tr>
<td>1.2.1 Theory</td>
<td>12</td>
</tr>
<tr>
<td>1.2.2 Organization</td>
<td>21</td>
</tr>
<tr>
<td>1.3 The Extended DEX Library.</td>
<td>25</td>
</tr>
<tr>
<td>1.3.1 Environment</td>
<td>26</td>
</tr>
<tr>
<td>1.3.2 Readers</td>
<td>26</td>
</tr>
<tr>
<td>1.3.3 Editors</td>
<td>27</td>
</tr>
<tr>
<td>1.3.4 Writers</td>
<td>27</td>
</tr>
<tr>
<td>1.3.5 Units</td>
<td>28</td>
</tr>
<tr>
<td>1.4 DEX Databases</td>
<td>30</td>
</tr>
<tr>
<td>1.4.1 Philosophy</td>
<td>30</td>
</tr>
<tr>
<td>1.4.2 Format of Database Entries.</td>
<td>30</td>
</tr>
<tr>
<td>CHAPTER 2    THE CUBE MODULE SAMPLE PROGRAM</td>
<td>33</td>
</tr>
<tr>
<td>2.1 General Description</td>
<td>33</td>
</tr>
<tr>
<td>2.1.1 Function of the Module</td>
<td>33</td>
</tr>
<tr>
<td>2.1.2 Module Subprograms</td>
<td>33</td>
</tr>
<tr>
<td>2.1.3 Typical Operation</td>
<td>34</td>
</tr>
<tr>
<td>2.2 Frequently Used Subroutines.</td>
<td>42</td>
</tr>
<tr>
<td>2.2.1 BLOCK DATA</td>
<td>42</td>
</tr>
<tr>
<td>2.2.2 MAINPG</td>
<td>45</td>
</tr>
<tr>
<td>2.2.3 MODIO.</td>
<td>47</td>
</tr>
<tr>
<td>2.2.4 MXUNIT and the &quot;All&quot; Logic.</td>
<td>47</td>
</tr>
<tr>
<td>2.3 The Input Series</td>
<td>51</td>
</tr>
<tr>
<td>2.3.1 INPUT.</td>
<td>51</td>
</tr>
<tr>
<td>2.3.2 DIMENS</td>
<td>51</td>
</tr>
<tr>
<td>2.4 The Output Series Subprograms</td>
<td>56</td>
</tr>
<tr>
<td>2.5 General Programming Comments</td>
<td>56</td>
</tr>
<tr>
<td>CHAPTER 3    THE EXTENDED DEX LIBRARY ENVIRONMENT</td>
<td>59</td>
</tr>
<tr>
<td>SETTING ROUTINES</td>
<td></td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS (cont’d)

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Introduction</td>
<td>59</td>
</tr>
<tr>
<td>3.2 Subroutine DIALOG</td>
<td>59</td>
</tr>
<tr>
<td>3.2.1 Menu and Calling Parameter</td>
<td>59</td>
</tr>
<tr>
<td>3.3 Subroutine SOURCE</td>
<td>62</td>
</tr>
<tr>
<td>3.3.1 Menu and Calling Sequence</td>
<td>62</td>
</tr>
<tr>
<td>3.3.2 Operation of SOURCE</td>
<td>64</td>
</tr>
<tr>
<td>3.4 Subroutine DESTIN</td>
<td>65</td>
</tr>
<tr>
<td>3.4.1 Menu and Calling Sequence</td>
<td>65</td>
</tr>
<tr>
<td>3.4.2 Operation of DESTIN</td>
<td>65</td>
</tr>
<tr>
<td>3.5 Subroutine MDMODE</td>
<td>66</td>
</tr>
<tr>
<td>3.6 Subroutine CHKRNG</td>
<td>67</td>
</tr>
<tr>
<td><strong>CHAPTER 4 THE EXTENDED DEX LIBRARY READING ROUTINES</strong></td>
<td>68</td>
</tr>
<tr>
<td>4.1 General Description</td>
<td>68</td>
</tr>
<tr>
<td>4.1.1 Function</td>
<td>68</td>
</tr>
<tr>
<td>4.1.2 Organization</td>
<td>68</td>
</tr>
<tr>
<td>4.2 Integer Scalar Series</td>
<td>70</td>
</tr>
<tr>
<td>4.2.1 ISCLDR</td>
<td>70</td>
</tr>
<tr>
<td>4.2.2 ISCPMP</td>
<td>73</td>
</tr>
<tr>
<td>4.2.3 ISREAD</td>
<td>75</td>
</tr>
<tr>
<td>4.3 Real Scalar Series</td>
<td>77</td>
</tr>
<tr>
<td>4.3.1 Brief Description</td>
<td>77</td>
</tr>
<tr>
<td>4.3.2 RSCLDR</td>
<td>77</td>
</tr>
<tr>
<td>4.3.3 RVAPMP</td>
<td>79</td>
</tr>
<tr>
<td>4.3.4 RSREAD</td>
<td>80</td>
</tr>
<tr>
<td>4.4 Real Array Series</td>
<td>82</td>
</tr>
<tr>
<td>4.4.1 Brief Description</td>
<td>82</td>
</tr>
<tr>
<td>4.4.2 RAILDR</td>
<td>82</td>
</tr>
<tr>
<td>4.4.3 RAIRED</td>
<td>83</td>
</tr>
<tr>
<td><strong>CHAPTER 5 THE EXTENDED DEX LIBRARY EDITING ROUTINES</strong></td>
<td>87</td>
</tr>
<tr>
<td>5.1 General Description</td>
<td>87</td>
</tr>
<tr>
<td>5.2 Logical Function ISCEDT</td>
<td>88</td>
</tr>
<tr>
<td>5.2.1 Calling Sequence</td>
<td>88</td>
</tr>
<tr>
<td>5.2.2 Operation</td>
<td>90</td>
</tr>
<tr>
<td>5.3 Logical Function RSCEDT</td>
<td>91</td>
</tr>
<tr>
<td>5.3.1 Calling Parameters</td>
<td>91</td>
</tr>
<tr>
<td>5.3.2 Operation</td>
<td>91</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS (cont'd)

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>THE EXTENDED DEX LIBRARY WRITING ROUTINES.</td>
<td>93</td>
</tr>
<tr>
<td>6.1</td>
<td>General Description</td>
<td>93</td>
</tr>
<tr>
<td>6.2</td>
<td>Integer Scalar Series</td>
<td>95</td>
</tr>
<tr>
<td>6.2.1</td>
<td>ISCDMP.</td>
<td>95</td>
</tr>
<tr>
<td>6.2.2</td>
<td>ISDSCR</td>
<td>97</td>
</tr>
<tr>
<td>6.2.3</td>
<td>ISRITE</td>
<td>98</td>
</tr>
<tr>
<td>6.3</td>
<td>Real Scalar Series</td>
<td>100</td>
</tr>
<tr>
<td>6.3.1</td>
<td>RSCDMP.</td>
<td>101</td>
</tr>
<tr>
<td>6.3.2</td>
<td>RVDSR</td>
<td>102</td>
</tr>
<tr>
<td>6.3.3</td>
<td>RSRITE</td>
<td>103</td>
</tr>
<tr>
<td>6.4</td>
<td>Real Array Series</td>
<td>105</td>
</tr>
<tr>
<td>6.4.1</td>
<td>RARDMP.</td>
<td>105</td>
</tr>
<tr>
<td>6.4.2</td>
<td>RARITE</td>
<td>108</td>
</tr>
<tr>
<td>7</td>
<td>THE EXTENDED DEX LIBRARY UNIT ROUTINES.</td>
<td>112</td>
</tr>
<tr>
<td>7.1</td>
<td>General Description</td>
<td>112</td>
</tr>
<tr>
<td>7.2</td>
<td>The I/O Unit Specifiers</td>
<td>113</td>
</tr>
<tr>
<td>7.2.1</td>
<td>General Description</td>
<td>113</td>
</tr>
<tr>
<td>7.2.2</td>
<td>Characteristics of a Typical Subroutine</td>
<td>117</td>
</tr>
<tr>
<td>7.3</td>
<td>The Basic Unit Series</td>
<td>117</td>
</tr>
<tr>
<td>7.3.1</td>
<td>General Description</td>
<td>118</td>
</tr>
<tr>
<td>7.3.2</td>
<td>Calling Parameters</td>
<td>118</td>
</tr>
<tr>
<td>7.3.3</td>
<td>Execution</td>
<td>120</td>
</tr>
<tr>
<td>7.4</td>
<td>Derived I/O Unit Series</td>
<td>121</td>
</tr>
<tr>
<td>7.4.1</td>
<td>Series Description</td>
<td>121</td>
</tr>
<tr>
<td>7.4.2</td>
<td>UPRESS Calling Parameters</td>
<td>123</td>
</tr>
<tr>
<td>7.4.3</td>
<td>UPRESS Operation</td>
<td>124</td>
</tr>
<tr>
<td>8</td>
<td>DEVELOPMENT OF A CRUISER-DESTROYER DATABANK AT M.I.T.</td>
<td>127</td>
</tr>
<tr>
<td>8.1</td>
<td>Considerations in Database Design</td>
<td>127</td>
</tr>
<tr>
<td>8.1.1</td>
<td>Function</td>
<td>128</td>
</tr>
<tr>
<td>8.1.2</td>
<td>Types of Databases</td>
<td>128</td>
</tr>
<tr>
<td>8.2</td>
<td>Organization of the MIT Cruiser-Destroyer Databases</td>
<td>131</td>
</tr>
<tr>
<td>8.2.1</td>
<td>General Databases</td>
<td>131</td>
</tr>
<tr>
<td>8.2.2</td>
<td>Mass Properties Databases</td>
<td>139</td>
</tr>
<tr>
<td>8.3</td>
<td>Independent and Dependent Variables</td>
<td>142</td>
</tr>
<tr>
<td>8.3.1</td>
<td>Concept</td>
<td>142</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS (cont'd)

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4 Application of DEX: An Example.</td>
<td>145</td>
</tr>
<tr>
<td>8.4.1 Function of MACHWT.</td>
<td>145</td>
</tr>
<tr>
<td>8.4.2 List of Subprograms and Menus</td>
<td>146</td>
</tr>
<tr>
<td>8.4.3 Description of the Subprograms.</td>
<td>148</td>
</tr>
<tr>
<td>8.4.4 Results from the MACHWT Module.</td>
<td>153</td>
</tr>
<tr>
<td>8.4.5 Future Developments</td>
<td>154</td>
</tr>
<tr>
<td>CHAPTER 9 CONCLUSIONS AND RECOMMENDATIONS</td>
<td>157</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>160</td>
</tr>
<tr>
<td>APPENDIX A CUBE MODULE LISTING</td>
<td>162</td>
</tr>
<tr>
<td>APPENDIX B UNIT SUBROUTINES ABBREVIATIONS AND CALLING SEQUENCES</td>
<td>201</td>
</tr>
<tr>
<td>APPENDIX C SAMPLE GENERAL DATABASE LISTING</td>
<td>205</td>
</tr>
<tr>
<td>APPENDIX D GENERAL DATABASE ENTRY CODES</td>
<td>210</td>
</tr>
<tr>
<td>APPENDIX E SAMPLE WEIGHT AND VERTICAL CENTER OF GRAVITY DATABASE LISTINGS</td>
<td>219</td>
</tr>
<tr>
<td>APPENDIX F MACHWT MODULE LISTING</td>
<td>226</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Page</th>
<th>Figure Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>Menu &quot;LEN.UNIT&quot;</td>
</tr>
<tr>
<td>1-2</td>
<td>Two Consecutive Operation Menus</td>
</tr>
<tr>
<td>1-3</td>
<td>DEX Menus</td>
</tr>
<tr>
<td>2-1</td>
<td>Cube Module Menus</td>
</tr>
<tr>
<td>2-2</td>
<td>Cube Module Subprograms Access Routes</td>
</tr>
<tr>
<td>2-3</td>
<td>Sample Cube Module Input Session</td>
</tr>
<tr>
<td>2-4</td>
<td>Sample Cube Module Output</td>
</tr>
<tr>
<td>2-5</td>
<td>Comparison of Module Input/Output Flow</td>
</tr>
<tr>
<td>3-1</td>
<td>Menu &quot;MOD.ALTR&quot;</td>
</tr>
<tr>
<td>3-2</td>
<td>Menu &quot;SOURCE&quot;</td>
</tr>
<tr>
<td>3-3</td>
<td>Menu &quot;DESTINAT&quot;</td>
</tr>
<tr>
<td>8-1</td>
<td>General Database Variable Relationships</td>
</tr>
<tr>
<td>8-2</td>
<td>MACHWT Module Menus</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-1</td>
<td>I/O Unit Specifier Subroutines, Menu Names and Units Available</td>
<td>114</td>
</tr>
<tr>
<td>7-2</td>
<td>I/O Unit Specifier Calling Parameters</td>
<td>116</td>
</tr>
<tr>
<td>7-3</td>
<td>Basic Unit Calling Parameters</td>
<td>116</td>
</tr>
<tr>
<td>7-4</td>
<td>Derived I/O Unit Series</td>
<td>122</td>
</tr>
<tr>
<td>7-5</td>
<td>Special Unit Names</td>
<td>126</td>
</tr>
<tr>
<td>8-1</td>
<td>General Ship Characteristics Database Variables</td>
<td>133</td>
</tr>
<tr>
<td>8-2</td>
<td>Input for MACHWT</td>
<td>147</td>
</tr>
<tr>
<td>8-3</td>
<td>Sample MACHWT Results</td>
<td>154</td>
</tr>
<tr>
<td>B-1</td>
<td>Angle Unit Abbreviations</td>
<td>202</td>
</tr>
<tr>
<td>B-2</td>
<td>Force Unit Abbreviations</td>
<td>202</td>
</tr>
<tr>
<td>B-3</td>
<td>Length Unit Abbreviations</td>
<td>202</td>
</tr>
<tr>
<td>B-4</td>
<td>Temperature Unit Abbreviations</td>
<td>203</td>
</tr>
<tr>
<td>B-5</td>
<td>Time Unit Abbreviations</td>
<td>203</td>
</tr>
<tr>
<td>B-6</td>
<td>Calling Sequences of Derived Unit Subroutines</td>
<td>204</td>
</tr>
<tr>
<td>D-1</td>
<td>Payload Shopping List</td>
<td>211</td>
</tr>
<tr>
<td>D-2</td>
<td>Supplemental Payload Shopping List</td>
<td>217</td>
</tr>
<tr>
<td>D-3</td>
<td>Index to Non-Payload Reed Code</td>
<td>218</td>
</tr>
</tbody>
</table>
CHAPTER 1
INTRODUCTION TO DEX

1.1 Background

Significant improvements in the capability and efficiency of computer-aided design systems have been achieved in the past decade by the introduction of interactive computer programs. This development was a major advance in providing more flexibility to the user at the input end of the process. However, too many of the innumerable design programs, and the design systems that incorporate collections of them, suffer from several bothersome problems:

(i) Less, but still excessive, restrictions on the flexibility of the programs to respond to the individual user's needs.

(ii) Incompatibility of input and output amongst programs and especially between programs and databases.

(iii) Excessive training time required for users to learn how to use the programs.

(iv) Inability to be transported to different facilities.

In 1974, researchers at the Massachusetts Institute of Technology and the University of Michigan felt that an investment in the "front end" of computer-aided design systems could eliminate or reduce these characteristics.
and result in design tools of greatly enhanced capability [1], [2], [3]. One of their goals was to develop a system that provides the designer almost as much flexibility at the computer terminal as he/she* has when sitting at a desk with pencil, paper, calculator and imagination at their disposal. The system would be easy to use because of a consistent approach to the interface between the user and the programs. Further, it would incorporate a more intelligent approach to the use of databases. They named this concept "DEX", for Design Executive System.

DEX is a self-contained software package that can be adapted to almost any computer system that supports Fortran. It provides an environment for running task-oriented programs, called "modules". It supports primarily, but not exclusively, interactive programs where there is communication between essentially five "parties": the user, the computer, the computation program, the source of input and the destination of output.

The purpose of the work reported in this thesis was to continue the development of the system at the intermediate level in the DEX hierarchy between what is referred to as "(the) DEX" and the "module". This intermediate

*The use throughout this thesis of the proper pronoun "he" and its derivatives when referring to the programmer, user or designer is for the smoothness of the text and is not to imply any presumption of those persons's sex.
category of subprograms is called the "extended DEX library". (The collection of all the design program modules is called the "DEX library".) The function of the extended DEX library is to enable the user to accomplish the following:

(i) Establish an environment in which the type of dialogue and the source and destination of information is defined.
(ii) Specify the system of units to be used for input and output.
(iii) Read information.
(iv) Edit information.
(v) Write information.

This investigator's motivation was to advance the development of an extremely capable tool for the field of ship design, but it should be stressed that DEX can be employed by any discipline involving computer-aided design.

1.2 Description of DEX

1.2.1 Theory. Reference [3] provides an original and excellent description of DEX, but this writer felt that some discussion should be presented here to assist the reader in relating to the information offered in this thesis.
There are five characteristics of DEX which reflect the design philosophy of the system:

(i) The user is in the design loop.

(ii) The system allows the design process to be executed in more than one sequence.

(iii) The system talks with the user in plain English.

(iv) The system is forgiving.

(v) The system has multiple capabilities for input and output.

1.2.1.1 The user in the design loop. Design processes are typically iterative ones. This is especially true in the ship design process as vividly illustrated by the ship design spiral. Computer programs allow many and/or complicated calculations to be performed quickly. The faster that the results can be analyzed and a new set of calculations initiated, the better. Even more advantageous is the ability to do only part of the calculations and analyze those results to decide whether or not to proceed. DEX extends the degree of spontaneity characteristic of interactive programs by enabling the dynamic starting of modules of the user's choice, by providing more options for sources of information immediately available to the user and by allowing the user to edit and insert information before it is used in calculations or written to its final destination.
1.2.1.2 **Flexibility.** The increased flexibility offered by DEX manifests itself in two ways. The first, implemented to allow any computer program acceptable to the operating system to be operated in a system incorporating DEX, is that the degree of involvement with DEX is completely within the prerogative of the module author. The term "module author" was introduced in reference [3] and will be adopted here for consistency. It applies to the person who writes the particular application program and who chooses which DEX services to use. As a minimum, the module author can arrange for the user to issue only the following commands to execute the program:

* start main (this activates DEX)
* begin module2 (this starts the user's program)

There would not really be any point to such an exercise but it can be done by fulfilling only two requirements. First, the name of the file containing the module name (e.g., module2 above) must be introduced in the DEX's

---

*The symbol "." will be used to indicate user-typed commands, "$" will indicate DEX messages and "*" will indicate module messages. These symbols are automatically inserted by DEX.*
library file. Secondly, the main program of the module must be a subroutine appearing first in a listing of the module.

The second aspect of the DEX's flexibility is the use of "menus" to provide the user with a wide choice of paths to follow to accomplish his goal. A menu is a list of options (a maximum of twelve per menu is possible) from which the user chooses to either define a value or proceed onto the next step of the operation. Some examples should prove helpful.

Figure 1-1 depicts a typical units menu which illustrates the first type of menu. The user would type in sufficient characters to identify the length unit to be used for input or output, e.g., "foot" (or just "f"). The subprogram which includes this menu would accept the choice, if proper, and return control to the subprogram which called it.

Menus are normally not displayed. The user will likely be aware of the menu options and is simply prompted to make a choice. For this example, one would see:

*ENTER AN ITEM FROM MENU - LEN.UNIT

However, if the menu choices are unknown, the user can type
Figure 1-1. Menu "LEN.UNIT"

Figure 1-2. Two Consecutive Operational Menus
. $display menu len.unit

to have the menu displayed by the DEX. Note that in
this case the user himself types "$". The word "dis-
play" is a selection from menu "DEX.MAIN". After re-
viewing the menu, by typing

    .continue

he is returned to the prompting message for the length
unit menu.

The second type of menu option directs the program
to proceed to the next operation. Figure 1-2 shows two
successive "operational" menus from a theoretical program
that estimates horsepower from the Taylor Standard
Series. The user would select item "input" from menu
"MOD.IO" in order to pass onto the subprogram containing
the menu "INPUT". Any of the items from that menu would
pass him onto another subprogram.

There is a subtle difference between the menus of
Figure 1-1 and 1-2. Observe that the second shows the
item "done" in both menus which is absent from the first.
A subprogram with a menu containing "done" returns con-
trol to the subprogram which called it only when that
entry is selected, whereas for the other, without a
"done", control returns automatically once a selection is made. The latter is used in menus where only one choice would be made at any time.

The user is free to choose any item from a menu that is meaningful to him. There is, therefore, no one predetermined path that must be followed when executing a group of menus. Logic, however, may dictate that one specifies units before reading in water properties.

1.2.1.3 Plain English. The messages and queries to the user provided by DEX have been designed to be as clear as possible. The instructions or responses that the user must supply are natural and logical words that would be used in an oral dialogue. An important aspect of these practices is the uniformity of dialogue encountered by the user under DEX. This reduces the effort required to learn how to operate a new program, which is not an insignificant advantage.

1.2.1.4 Forgiving. By extending the capabilities of the conventional design programs with DEX, the user can accomplish more during a session, but this entails more thinking on his part. The probability that errors will occur is therefore higher.

Even the most experienced user makes mistakes. It may be as simple as depressing the wrong key when typing a menu selection or as improper as supplying an integer
when the program wants a real number. When developing the DEX and extended DEX library routines, and the Machinery Weight Estimating Module of Chapter 8, as many potential errors as possible were anticipated and diagnostic messages, in plain English, were provided. In some cases they advise the user of the error and allow him to try again at that same point. In others, especially where a problem is caused by a programming error, control is returned to the user several sequential subroutines prior to where the error occurred, with a message issued to assist in determining where to search for the mistake.

1.2.1.5 Input/Output Capability. DEX enables the communication of information by the dynamic allocation of databases and files, which are the only two storage locations it recognizes. In practice we distinguish between two types of files, such that the list of locations is as follows:

(i) databases

(ii) input locations (which are the terminal for alphanumeric characters and a graphics screen for x-y coordinates) and output locations (the terminal screen in the form of menus)

(iii) disk files.

Now, for the ease of understanding of the user, the environment in which he operates is described as
having a total of five "sources" of information and four "destinations". The term "information" is preferred here to "input" and "output" to preclude a limiting misconception by the reader. The tendency to think of input as data read and output as answers written should be avoided. In fact, the user may need to "write" input to the terminal for inspection, or "read" an output value from the terminal in order to "write" it to a database. For this reason this writer will often apply the term "information" to both input and output variables as values that have a source or destination.

The sources and destinations provided for in the operating environment of the DEX system described in this thesis are listed here:

(i) DEX-created databases
(ii) the terminal using DEX routines to read or write alphanumeric characters
(iii) the terminal or a plotter using graphics routines to read or create x-y plots
(iv) sequential files
(v) module default data (source only)

The third capability does not yet exist in the present version of DEX at MIT because all the necessary enabling routines have not yet been implemented. If the user tries to employ it, error messages advise him of this situation.
The user is not confined to using the same type of destination as source, or the same source for all the information of a program. He may read information from one or more databases, for example, and write it to another, or to the terminal or to a file, or all three in succession. The only restriction is that the module can be pointed toward only one source or destination at one time.

1.2.2 Organization. The hierarchy of DEX consists of three levels of programs: the DEX, the extended DEX library and the module. The first two categories comprise a permanent, portable set of subprograms which provides an interface between the computer and the user-supplied module.

1.2.2.1 DEX. The category called DEX consists of several hundred subroutines, each with a very specific function, which provide the foundation, or "umbrella" if you will, for the DEX System. The employment of these subroutines by the module authors results in a uniform appearance of the system to the user of the various modules exercised. The DEX services provide input/output and data utilities and, eventually at MIT, graphics utilities for the module authors. Although the module author and user need not be aware of most of these subprograms, in two areas they draw directly on the features of routines in this category.
The first area, of interest to the module author, is a set of 37 subroutines and functions which the programmer may incorporate into his module to perform certain tasks. References [4] and [5] describe these subprograms and how they are used. Briefly, they are grouped into five categories: control of execution, input, output, database management and character manipulation. Subsequent chapters will refer to these as "DEX routines".

The second area of overt involvement with the DEX category is encountered by the user during operation of a module. When the command

```
.start main
```

is issued, the user enters the DEX environment and is presented with a prompting message for menu "DEX.MAIN". There are six menus at this level of DEX, and these are listed in Figure 1-3. The first instruction given to the user is:

```
$ENTER AN ITEM FROM MENU - DEX.MAIN
```

These items are listed in the rightmost column in the table. Note menu "DEX.DISP" which allows the user to display any menu by typing

```
.display menu menuname
```
Figure 1-3. DEX Menus (as printed on terminal)
The user's module is activated when he types, from menu "DEX.MAIN",

.begim modulename

He then enters the environment of the module, but he can return to the DEX level by using the symbol "$" and then an item from "DEX.MAIN". Similarly, he can transfer temporarily from the DEX level to the local computer operating system level by the option "system". To get back into DEX he uses the command

.return

and then to get back to the module he uses the menu option "continue".

When a module execution is complete, the user returns to the DEX level and issues the command

.quit

to return permanently to the operating system.

1.2.2.2 Extended DEX Library. The extended DEX library is not a level of operation like DEX on the module, but rather a collection of 45 subroutines and functions which the module author can adopt in constructing his module. The bulk of this investigator's work involved the development of this library, and it will be discussed further in Section 1.3 and Chapters 3-7.
1.2.2.3 Module. A module is a complete set of subprograms written by the module author and executed by the user to perform a specific task. A module may consist of only one program which does the actual calculations, or it may have many additional subprograms employing menus to take advantage of the flexibility offered by DEX and the extended DEX library. Chapter 2 describes in detail an actual module used during the testing of the extended DEX library subprograms, and Chapter 8 describes the Machinery Weight Estimating Module written to demonstrate the use of the cruiser-destroyer databases.

1.3 The Extended DEX Library

In order to convey information from a storage location to the program doing the calculations and from there to another storage location or display, five capabilities are required by the user. These five, listed here, are provided by the extended DEX library:

(i) Setting and reviewing the module environment for type of dialogue and sources and destinations of information.

(ii) Reading information.

(iii) Editing information.

(iv) Writing information.

(v) Converting the user-preferred input/output units to the module author-preferred units and back again.
The five categories will be briefly outlined here and described in detail in Chapters 3-7.

1.3.1 Environment. Four subroutines provide or display the module environment defined by the user. They are:

**DIALOG:** enables user to specify terse or verbose dialogue

**SOURCE:** enables user to specify source of information, either a DEX-created database, the terminal, a sequential file or the module's default data.

**DESTIN:** enables user to specify the destination of information, either a DEX-created database, the terminal or a sequential file

**MDMODE:** displays the status of the module environment, including type of dialogue, reading source, writing destination, and reference numbers for files to be read from or written to.

1.3.2 Readers. Eight logical functions allow the user to read information from the designated source. They are:

**ISCLDR:** invokes ISCPMP and ISREAD

**ISCPMP:** prepares a prompting message for reading an integer from the terminal

**ISREAD:** reads an integer value from the source.

**RSCLDR:** invokes RVAPMP and RSREAD

**RVAPMP:** prepares a prompting message for reading a real scalar or a real array from the terminal.

26
RSREAD: reads a real scalar value from the source
RAILLDR: invokes RVAPMP and RAIRED
RAIRED: reads a single one-dimensional array from the source.

1.3.3 Editors. The editing routines are still undergoing development. Eventually they will enable the user to perform various editing functions on the working variables of the module. Two preliminary routines were completed during this investigation:

ISCEDT: enables the user to change the value of an integer scalar variable from the terminal.
RSCEDT: enables the user to change the value of a real scalar variable from the terminal.

1.3.4 Writers. Eight logical functions allow the user to write information to the designated destination. They are:

ISCMDMP: calls ISDSCR and ISRITE
ISDSCR: prepares a description message for writing an integer to the terminal
ISRITIE: writes an integer scalar to the destination
RSCMDMP: calls RVDISCR and RSRITE
RVDSCR: prepares a description message for writing a real scalar or a one-dimensional real array to the terminal
RARDMP: calls RVDSCR and RARITE
RARITE: writes one-dimensional real arrays to the destination

1.3.5 Units. The units subprograms are divided into three categories. The first category contains five subroutines which read, edit or write the input/output units that the user wishes to employ. There is one for each of the five basic types of units adopted by the system: plane angle, force, length, temperature and time. The names of these subroutines are:

AUNIT
FUNIT
LUNIT
TPUNIT
TUNIT

The second category contains five logical functions, one for each of the five basic unit types, which determine the conversion factors for converting to i/o units to the program standard units and vice versa. Additionally, they prepare unit names and abbreviations of unit names which are used in database comments and prompting and description messages. The names of these functions are:
The last category contains twelve logical functions which perform the same function as those in the second category, but for derived units formed by combining basic units. They are:

- UAACC: angular acceleration
- UACCEL: linear acceleration
- UAREA: area
- UFREQ: frequency
- UKVISC: kinematic viscosity
- UMASS: mass
- UMPOWER: mechanical power
- UPRESS: pressure
- UPSPEC: power spectrum
- URHO: density
- USPEED: speed
- UVOL: volume
1.4 Dex Databases

1.4.1 Philosophy. The DEX philosophy includes as a fundamental feature a new and more capable approach to database manipulation. This revolves around the concept of identifying a variable within a database by its name and not by its location. For example, if a user needs the value of an entry in the database signifying the ship's draft, he retrieves that value at the address "DRAFT", or whatever name it has been assigned by the database creator, and not by specifying that the value is the fourth or twentieth entry in the database.

In order to use the capabilities of DEX a database must be constructed via either the commands of menu "DBEDCMDS" or the database management routines in reference [4]. These entail a specific format for the entry of the variable, but there is no sequential order required for arranging the different variables in the database.

1.4.2 Format of Database Entries. In the present version of DEX a database can contain up to 200 variables. Three types of variables are allowed: integer scalars, real scalar, and one-dimensional real arrays. A real array can contain up to 200 elements.

A variable entry in a database consists of four parts. First is the database name, which is formed by
up to eight alphanumeric characters (including blanks), e.g., "LBP", "WEIGHT17", "TYPSONAR". Second is the type of variable - integer, real scalar or real array - and third is the value of the variable itself. The final part is the database comment statement, a string of words up to 64 characters long which describes the variable. If the variable is either a real scalar or real array and has units, the comment statement contains a twelve-character (including blanks) version of the units enclosed in parentheses. Appendix B contains edited listings of a warship general database which illustrates database entries.

Accompanying each database will be a database dictionary which lists for each variable its database name, type, array size, if applicable, and official database comment, including units, if applicable. Future versions of DEX will separate the units from the comment as a distinct fifth part of a variable entry. Further, development has started to create positional databases which will allow database elements to be related to each other, e.g., a database containing a ship's compartments where two compartments are adjacent.

This chapter has attempted to give a brief introduction to the concept and organization of DEX. For the
reader who is confused at this point by the large number of new ideas, terms and subprograms mentioned, Chapter 2 has been included to provide an example of a simple module which employs many of the concepts and subroutines described. It should give the reader a practical awareness of how this all ties together. This will prove helpful in reading the next five chapters which discuss the design of the extended DEX library subprograms. Chapter 8 discusses an application of DEX to computer-aided ship design. Finally, Chapter 9 offers recommendations for future work.
2.1 General Description

2.1.1 Function of the Module. The Cube Module was developed to test the subprograms of the extended DEX library written during this investigation. The module calculated the volume and weight of a block of salt water given the length, width, and height (note that "cube" is a slight misnomer). While that function itself was elementary, the combination of single, scalar values for length and width and an array for height (and also, therefore, for volume and weight) permitted the testing of the reading, editing and writing routines for real scalars and the reading and writing routines for real arrays. The subprogram for specifying input and output units employed the routines for integer scalars. The subroutines for determining the conversion factors for length, force and volume were also exercised. Finally, as a matter of course, the environment setting routines were also tested.

Appendix A contains a listing of the module.

2.1.2 Module Subprograms. Although no single, correct sequence of operating the module subprograms existed, there was a logical path one would follow to
execute the program when not attempting to test every available option of the module. This path is a convenient order in which to list the module subprograms and those extended DEX library subprograms involving menus:

- MAINPG (C-M)
- DIALOG (DL-M)
- SOURCE (DL-M)
- DESTIN (DL-M)
- MDMODE (DL)
- MODIO (C-M)
- INPUT (C-M)
- MXUNIT (C-M)
- LUNIT (DL-M)
- FUNIT (DL-M)
- DIMENS (C-M)
- COMPUT (C)
- OUTPUT (C-M)
- VANDWT (C-M)
- BLOCK DATA (C)

The "C" indicates a Cube Module subprogram, the "DL" indicates an extended DEX library subprogram and the "M" indicates that the subprogram included a menu. Figure 2-1 illustrates the menus encountered in this model.

2.1.3 Typical Operation. A description of a typical execution of the Cube Module should prove helpful. To assist the reader, Figure 2-2 shows the various access and return routes of the subprograms. Once the user entered the DEX level environment he activated the Cube Module via the "begin" option of menu "DEX.MAIN", that is

- .begin cube
Figure 2-1. Menus Encountered in Executing the Cube Module
Figure 2-2. Access routes of the various subprograms encountered in executing the Cube Module. Return routes are back along the arrows.
where "cube" was the assigned module identification name. This command placed him in module subroutine MAINPG where he encountered the message:

*ENTER AN ITEM FROM MENU-MOD.MAIN

The user typed "mod.mode" and was presented with the status of the module environment. The initialized values for the dialogue, source and destination were verbose, terminal and terminal respectively and he found these satisfactory. He then typed

.read

which sent him to subroutine MODIO and the message

*SELECT WHICH MODULE VARIABLE SEGMENT TO READ
*ENTER AN ITEM FROM MENU-MOD.IO

From this menu he selected item "input" by typing it:

.input

Now in subroutine INPUT he received the instruction

*SELECT WHICH INPUT VARIABLE SEGMENT TO READ
*ENTER AN ITEM FROM MENU-INPUT

At this point, to save time, he made two sequential selections. From menu "INPUT" he selected "units" and anticipating menu "UNIT" he chose "length". He accomplished this by typing

.units length

DEX recognized the space between the two words as a delimiter between two commands. It acted on the first by invoking subroutine MXUNIT. It then located item
"length" on that subroutine's menu and called subroutine LUNIT which issued the command

*ENTER LENGTH UNIT TO BE USED DURING INPUT OUTPUT
*ENTER AN ITEM FROM MENU-LEN.UNIT

The user specified "foot". LUNIT then passed control back to subroutine MXUNIT which printed

*SELECT WHICH UNIT TO READ
*ENTER AN ITEM FROM MENU-UNIT

Again to save time, the user typed in two sequential menu selections:

.force fpound

This sent him to FUNIT and then back to MXUNIT. This time from menu "UNIT he chose item "done" which returned him to subroutine INPUT. Figure 2-3 illustrates this sequence.

At this point the user had defined the length and weight units he intended to use for input and output. In response to INPUT's request for a menu selection he typed

.dimensio

This invoked subroutine DIMENS and caused the following to be printed.

*SELECT THE DESIRED DIMENSION TO READ
*ENTER AN ITEM FROM MENU-DIMENSIO

The user intended to read in all three dimensions (he could have used any or all of the initialized values built into the module in BLOCK DATA) so he typed item
*ENTER AN ITEM FROM MENU - MOD.MAIN
.mod.mode
*MODULE DIALOGUE : VERBOSE
*MODULE INPUT SOURCE : THE TERMINAL (ALPHANUMERIC)
*MODULE OUTPUT SOURCE : THE TERMINAL (ALPHANUMERIC)
*REFERENCE NUMBER FORM MODULE
*FORTRAN READ FROM A FILE : 3
*FORTRAN WRITE TO A FILE : 4
*ENTER AN ITEM FROM MENU - MOD.MAIN
.read
*SELECT WHICH MODULE VARIABLE SEGMENT TO READ.
*ENTER AN ITEM FROM MENU - MOD.IO
.input
*SELECT WHICH INPUT VARIABLE SEGMENT TO READ.
*ENTER AN ITEM FROM MENU - INPUT
.units length
*ENTER AN ITEM FROM MENU - LEN.UNIT
.foot
*SELECT WHICH UNIT TO READ.
*ENTER AN ITEM FROM MENU - UNIT
.force fpound
*SELECT WHICH UNIT TO READ.
*ENTER AN ITEM FROM MENU - UNIT
.done
*SELECT WHICH INPUT VARIABLE SEGMENT TO READ.
*ENTER AN ITEM FROM MENU - INPUT
.dimension
*SELECT THE DESIRED DIMENSION TO READ.
*ENTER AN ITEM FROM MENU - DIMENSION
.all
*ENTER LENGTH OF CUBE (FOOT )
*ENTER UP TO 1 REAL NUMBERS
1.0
*ENTER WIDTH OF CUBE (FOOT )
*ENTER UP TO 1 REAL NUMBERS
1.0
*ENTER HEIGHT OF CUBE (FOOT )
*ENTER UP TO 4 REAL NUMBERS
1. 2. 3. 4.
*SELECT WHICH INPUT VARIABLE SEGMENT TO READ.
*ENTER AN ITEM FROM MENU - INPUT

Figure 2-3. Sample Cube Module Input Sequence
"all". The computer responded with:

*ENTER LENGTH OF CUBE (FOOT )
*ENTER UP TO 1 REAL NUMBERS

The user typed in "1.0" and the computer proceeded to the next instruction

*ENTER WIDTH OF CUBE (FOOT )
*ENTER UP TO 1 REAL NUMBERS

The process was repeated, except that for height the computer requested up to four real numbers (height was defined as an array having up to four elements). When the desired number (say four) of heights were entered, control returned to INPUT. The user then typed in .done done to return to subroutine MODIO and from there to subroutine MAINPG.

In response to this subroutine’s instruction the user typed .compute

This invoked the COMPUT subroutine to actually calculate the volumes and weights. Control was then returned to MAINPG as evidenced by the instruction

*ENTER AN ITEM FROM MENU-MOD.MAIN

By now comfortable with the operation of the module, the user decided to display his results on the terminal. He decided to retain "foot" and "poundforce" as the units, although he could have selected any desired
units by accessing MXUNIT again. The volumes and weights returned by COMPUT were in metric units, that being the system in which COMPUT was written. The conversions took place at the input/output locations. More on this later.

Now observe closely. The user issued the following commands:

```
WRITE output results all
```

These were selections from menus "MOD.MAN", "MOD.IO", "OUTPUT" and "RESULTS" respectively. The computer printed on the terminal the four volumes and four weights. Figure 2-4 is a copy of that printing.

Satisfied with his answers, the user responded to the current instruction from subroutine OUTPUT with

```
DONE done
```

to get back to MOD.MAN. Choice "quit" at this point caused him to leave the module level and return to the DEX level, facing menu "DEX.MAN". The "quit" selection allowed him to exit DEX and reenter the operating system level in order to log off.

The rest of this chapter will describe the subprograms of the Cube Module to assist the reader in understanding how to write a module program.
2.2 Frequently Used Subroutines

2.2.1 Block Data. We will commence the discussion of the module with the subroutines which may be used with only slight changes in form and/or content in several modules. The user may wish to refer frequently to the listings of these subroutines in Appendix A.

First is a special subprogram, BLOCK DATA, used as standard practice in Fortran to initialize all the labeled common blocks used throughout the module. With respect to input/output variables, a typical labeled

<table>
<thead>
<tr>
<th>INDEX</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.999999E+00</td>
</tr>
<tr>
<td>2</td>
<td>0.200000E+01</td>
</tr>
<tr>
<td>3</td>
<td>0.299999E+01</td>
</tr>
<tr>
<td>4</td>
<td>0.399999E+01</td>
</tr>
</tbody>
</table>

Figure 2-4. Sample Cube Results (as printed on terminal)
common block appearing in BLOCK DATA and the pertinent subprograms is as follows:

    COMMON /VINFO/ V(4), VOLNAM, VOLCMT, NDEVF, DEFALV(4)

From this we can identify the following information which should appear in BLOCK DATA:

1. the variable (dimensioned for its maximum size if an array)
2. the variable database name
3. the variable database comment
4. the number of default values (if it is an array)
5. the default value (or dimensioned default array)

These values would all be initialized in the BLOCK DATA. Locating the initialization and dimensioning of all input/output variables in one subprogram facilitates checking and is a highly recommended practice.

The variables are grouped under the subroutines in which they first appeared. MAINPG included four common blocks - REFNOS, INOUTF, DIALFG and MDNCPW. All of these are used in several other subprograms. REFNOS included the variables RNRFIL and RNWFIL which represented respectively the file reference numbers for reading from a sequential file using the Fortran READ command and writing to a sequential file via the Fortran WRITE
command. INOUTF included the variables IMODE and OMODE. The first denoted the source of reading information (1 = database, etc.) and the second denoted the destination for writing information (1 = database, etc.). DIALGF contained the logical variable MTERSE to denote the type of module dialogue (terse or verbose). Finally labeled common MDNCDW contained the integer variable NCPW which was the number of characters per word assumed by the DEX routines. This value was site dependent. For example, on IBM computers it was 4. For this reason it was flagged in BLOCK DATA as site-dependent.

The other subroutines which represented the first occurrences of labeled common blocks were LUNIT, TUNIT, FUNIT, AUNIT, TPUNIT, DIMENS and VANDWT. Let us examine DIMENS. It contained nine labeled common blocks. The first four were mentioned above in MAINPG. LUNITS will be described in Chapter 7. The remaining four were listed under subroutine DIMENS in BLOCK DATA. LINFO, WINFO and HINFO contained the variable, database name, comment, default values and number of default values, where applicable, for the length, width and height dimensions respectively. H and DEFALH were dimensioned because they were arrays. The type declarations and dimensions of all the variables were followed by the
data declarations of their values. The remaining common block, DIMFRM, represented the formats to be used when reading from or writing to sequential files. One format was for real scalars and the other for arrays.

2.2.2 MAINPG. Subroutine MAINPG, via menu "MOD.MAIN", allowed the user to select the environment of the module and the desired paths to follow to operate it. The capabilities it gave the user were:

1. To set the style of module dialogue. This choice invoked subroutine DIALOG.

2. To select the source of module input. This choice invoked subroutine SOURCE. Remember that "input" here means any information to be read, even output variables.

3. To select the destination of module output. This choice invoked subroutine DESTIN.

4. To list the module flags. This invoked subroutine MDMODE which printed the status of the dialogue, the source, the destination, and the file reference numbers.

5. To access the module reading routines. This choice set the variable IOFLAG=1 and then called MODIO.

6. To access the module editing routines. This choice set IOFLAG=2 and then called MODIO.
(7) To access the module computing routine by calling COMPUT to execute the calculations.

(8) To access the module writing routines. This choice set IOFLAG=3 and then called MODIO.

(9) To return control to the DEX level and menu "DEX.MAIN". It did this by invoking the DEX routine ENDIT.

The menu name and the items were declared in MAINPG with data statements. DEX routine MENUIN was used to convert the user's selection to an integer flag ITEM for branching to the correct point in MAINPG. MENUIN was the routine that actually printed the prompting instruction to enter a menu item.

The user can make his subroutine MAINPG as simple or extensive as desired within the constraint that the maximum number of menu items (because of MENUIN) is twelve. To show how simple it could be, consider a user who has a program called STRENGTH and for some reason he wanted to start it with the DEX. All he would need in his module besides STRENGTH is the following subroutine:

```
SUBROUTINE MAINPG
CALL STRENGTH
RETURN
END
```

If STRENGTH used menus, DEX routine ENDIT would also have to be called to clear the buffer afterwards.
2.2.3 MODIO. Subroutine MODIO had one parameter in its calling sequence - IOFLAG - which indicated if the operation to be performed was reading, editing and writing. MODIO had two labeled commons - DIALGF and MDNCPW - discussed in BLOCK DATA. DIALGF determined whether the verbose or terse menu prompting message would be issued. NCPW was needed for calling DEX routines STRPAK and LMOVEC.

MODIO presented the user with the menu selections shown in Figure 2-1. The "all" option allowed the user to read, edit or write all the input and output variables. The "all" option was implemented by a logical variable "ALLFLG" which was set to .TRUE if that menu item was chosen. This variable was subsequently passed on through the module. If it was returned .FALSE. to MODIO, it would cause the execution of the last command to be halted and the prompting message for menu "MOD.IO" to be issued so that the user could take corrective action.

2.2.4 MXUNIT and the "ALL" Logic. A subprogram similar to MXUNIT will be needed in any module which allows the user to specify i/o units different from those in which the computing routines were written. MXUNIT permitted the user to read, edit or write the module units he wished to use for input and output. For example, if the user selected "force" from menu "UNIT", subroutine FUNIT
would have been called to present the menu choices for force units. FUNIT and the others in that series will be described in Chapter 7, but some detail is required here because of all the calling parameters involved.

The calling sequence for FUNIT, typical for all five in that series, was as follows:

```
CALL FUNIT(UIFUN, LOCALL, IOFLAG, IOMODE, MTERSE, NCPW, DBFUNN, DBFUNC, PMPREP, PMES, RNFILE, FUNFRM, DEFFUN)
```

The first parameter was an output variable, LOCALL was both input and output, and the rest were all input variables provided by MXUNIT.

UIOFUN would have been assigned a value between 1 and 7 depending on what force unit the user selected.

LOCALL carried the information concerning the "ALL" option. One of the calling parameters for MXUNIT, CALALL, passed on the "ALL" option from subroutine INPUT. If it was .TRUE. then LOCALL would have immediately been assigned .TRUE. also, and the prompting for menu "UNIT" would have been bypassed. Each of the i/o unit-defining routines, such as FUNIT, would have been called and the user asked to specify the desired choice of the particular type of unit. Even if CALALL was .FALSE., the user could have selected "all" from menu "UNIT" with similar results.
If any of the i/o unit-defining subroutines was unsuccessful, LOCALL would have been returned .FALSE.. The program would have checked the value of CALALL. If it too was .FALSE., the menu "UNIT" would have been presented for another selection. However, if CALALL was .TRUE., it would have meant that there had been a change in the value of LOCALL (i.e., a failure in the called subprogram). MXUNIT would have set CALALL to .FALSE. and returned control through subroutine INPUT back to MODIO. This was because INPUT incorporated the same "ALL" logic as MXUNIT.

Note from the listing of MXUNIT that LOCALL is initialized as .FALSE.. If CALALL was .FALSE. and the user did not choose "all" from menu "UNIT", he would have been asked to select from the menu each time after a unit was specified, until he typed "done". This "ALL" logic was used extensively throughout the module as a means of expediting the reading, editing or writing of many variables.

Returning to the calling sequence for FUNIT, we have already mentioned IOFLAG, indicating the operation to be performed. PMPREP was set locally by a data declaration. It indicated to the subprograms called whether or not a prompting message for the unit to be selected was to be prepared by the program. If .TRUE., PMES would later be the storage location for the prompting message. If
PMPREP was .FALSE., PMES would already have to have the prompting message in it. Since PMES first appeared here in MXUNIT it was dimensioned here for the maximum allowable size of 16 "words". This number came from the limitation on PMES to be at most 64 characters long, divided into 4-character "words".

All the other variables were obtained by MXUNIT from other subprograms, including BLOCK DATA, by labeled common blocks. An inspection of the listing of MXUNIT reveals the large number of commons required because of five types of units.

INOUTF passed the indicator of the source of information to be read (IMODE) and the destination of information to be written (OMODE). Depending on IOFLAG, IOMODE was set equal to one of these two. This told FUNIT from where UIOFUN was to be read or to where UIOFUN was to be written.

REPNOS passed the values of the file reference numbers for Fortran READ (RNRFIL) and Fortran WRITE (RNWFIL). Depending on IOFLAG, RNFILE was set equal to one or the other. These values are assigned to files during the execution of extended DEX library routines SOURCE and DESTIN by DEX routine SETDEV if the user had designated files as the source or destination.

The other labeled commons will be described in Chapter 7.
2.3 The Input Series

2.3.1 INPUT. Subroutine INPUT provided the user with a menu by which he could access subroutines MXUNIT and DIMENS and return control to subroutine MODIO. It contained the labeled common blocks DIALGF and MDNCPW for use in calling STRPAK and LMOVEC for character manipulation. INPUT had two variables in its calling sequence - CALALL and IOFLAG - as did MXUNIT described above. The logic for the use of CALALL and LOCALL was identical to that described in 2.2.4. Briefly, if LOCALL was set to .TRUE. when INPUT was invoked because CALALL=.TRUE., and it was returned by MXUNIT or DIMENS as .FALSE., CALALL would have been set equal to .FALSE. and control passed back to MODIO. If CALALL was .FALSE., LOCALL would have been .FALSE. unless the user selected "all" from menu "INPUT".

Besides LOCALL, INPUT passed IOFLAG to MXUNIT and DIMENS to indicate reading, editing or writing.

2.3.2 DIMENS. Subroutine DIMENS allowed the user to read, edit, or write the value of the block dimensions. It had two calling parameters, ALLFLG and IOFLAG. The former carried the "ALL" option information and behaved like CALALL. The "ALL" option was passed on to the called logical functions by the usual variable LOCALL.

DIMENS, like MXUNIT, also contained quite a few labeled common blocks. Five have already been seen in
MXUNIT: DIALGF, INOUTF, MDNCPW, REFNOS, and LUNITS. LINFO, WINFO, and HINFO contained the variable, database name, comment and default values for the length, width and height of the block. DIMFRM contained the format information for reading from or writing to files.

Before menu "DIMENSIO" was presented to the user, two actions occurred. The first was the calling of the logical function UNITLF by the statement

```
LOGVAL=UNITLF(CONVLM,NAML02,NAML06,NAML12,ALLFLG, PSTLUN,UIOLUN,NCPW)
```

UNITLF, discussed in Chapter 7, produced the multiplicative conversion factor CONVLM for converting the dimensions in the input length unit to values in the program standard length unit (meter). The second through fourth variables in UNITLF's calling sequence represented various abbreviations of the length unit—the user had selected for input/output. UNITLF was able to provide this information because of the values of PSTLUN and UIOLUN. The first indicated program standard length unit, the second user i/o length unit. As a pair they were an index to locating the other information.

The other action which occurred immediately in DIMENS was the setting of LOCALL equal to ALLFLG. If this made LOCALL equal to .TRUE., menu "DIMENSIO" was not presented and DIMENS immediately started operating on all the menu choices.
If the user selected "width" from the menu, the program next referred to IOFLAG. If IOFLAG=1 (reading), the program branched to the statement

```
LOGVAL=RSCLDR(W,LOCALL,MTERSE,IMODE,NCPW,WIDNAM,
               CONVLM,CONVLA,NAML12,.FALSE.,.TRUE.,
               PMES,WMORGN,RNRFIL,LFIRM,DEFALW)
```

Logical function RSCLDR is the first of three functions for reading real scalars. The value read in for width, in program standard units, was stored in W. IMODE indicated the source of the reading. WIDNAM was the 8-character database name for width. CONVLM and CONVLA were respectively the multiplicative and additive conversion factors for changing the read in width to the value in program standard units via the expression

```
W=W * CONVLM + CONFLA
```

CONVLA was locally declared 0. NAML12 was the 12-character version of the i/o length unit and was used in prompting message preparation and database comment comparison. The parameter .FALSE. represented the variable VITAL which indicated if the variable W was essential for input continuation. The parameter .TRUE. was for PMPREP, indicating that the reading subprograms would prepare the prompting message PMES, at this point undefined. Since PMES was a local variable not passed to DIMENS by the calling sequence or a labeled common block it was dimensioned "16" in DIMENS. WMORGN contained the description
of the width, including space allotted for inserting the units. RNRFIL and LWPRM were the reference number and format respectively for reaching width from a file. DEFLAW was the default value for width if the user wished to set W equal to that.

If IOFLAG=2, the subprogram called the editing routines by the expression

\[ \text{LOGVAL=RS CedT}(W, \text{LOCALL}, \text{MTERSE}, \text{NCPW}, \text{WIDNAM}, \text{CONVLM}, \text{CONVLA}, \text{NAML}2, .\text{TRUE.}, \text{Pmes}, \text{WMORGN}, \text{RNRFIL}, \text{LWFRM}, \text{DEFLAW}) \]

This was very similar to the reading function except that IMODE was not specified. This was because RS CedT itself defined and employed a variable EDMODE, of value 2, for the terminal, in lieu of IMODE, when it called RSCLDR.

Finally, if IOFLAG=3, the real scalar writing routines were invoked by the statement

\[ \text{LOGVAL=RS CDMp}(\text{LOCALL}, \text{MTERSE}, \text{OMODE}, \text{NCPW}, W, \text{WIDNAM}, \text{CONVLM}, \text{CONVLA}, \text{NAML}2, .\text{TRUE.}, \text{Pmes}, \text{WMORGN}, \text{RNWFIL}, \text{LWFRM}) \]

Observe that OMODE was used to indicate destination of the value of W for writing. Also note the use of RNWFIL to indicate the file reference number for writing with Fortran WRITE, using the format supplied by LWFRM.

Because the height variable H represented a four-element array, the logical functions called were different. For reading (IOFLAG=1) DIMENS branched to the statement
Several new variables appear here. MXTOGT represented the maximum number of elements to extract from the source when the source was a database or the terminal. NGOT represented the actual number of elements read from the source. Both MXTOGT and NGOT were initialized as 4 in DIMENS. VITAL was defined by the .FALSE. and PMPREP by the .TRUE.. HFRM contained the format for reading the array from a file. NDEFH and DEFALH were respectively the number of default values and a four-element array containing the default values for height.

Because the array editing routines had not yet been written, provision for calling a dummy routine, RAREDT, was included in DIMENS.

For writing (IOFLAG=3), the program branched to

LOGVAL=RARDMP (LOCALL, MTERSE, OMODE, NCPW, H, HEINAM, NFROM, NGOT, CONVLM, CONVLA, NAML2, .TRUE., PMES, HMORGN, RNWFIL, HFRM)
2.4 The Output Series Subprograms

The output series consisted of two subprograms - OUTPUT and VANDWT - for working with the volumes and weights calculated by COMPUT. These had direct parallels to INPUT and DIMENS and need not be discussed in detail. OUTPUT offered the user the capability of invoking MXUNIT, via the menu item "unit", in case the user wished to write the volume and weight in different units from those he had used for reading in the block dimensions.

2.5 General Programming Comments

This chapter will conclude with some comments about writing modules. It is hoped that the reader has some understanding of the calling parameters used by module subprograms when invoking extended DEX library routines. In some cases large numbers of variables appear in the calling sequences. This is due to the fact, as will be pointed out in the next chapter, that the library routines use no common blocks.

One suggestion already emphasized is the use of a BLOCK DATA subprogram to initialize all input/output variables and their associated variables. This adds some extra work by increasing the number of common block variables, such as the database names, comments and default values. However, the improved efficiency for checking values and dimensions is considered to outweigh this disadvantage.
When writing a module, to determine the menus required it is easiest to work backwards in the opposite direction to the order of use. Identify the input and output variables and place them in menus. Then establish a supervisory input menu and a supervisory output menu. For example, one may contain the group name for the input variables, such as "dimensions" in Cube, plus a units item to allow the user the choice of i/o units. At this point the module author may be almost done with the module design phase, because he can frequently incorporate the standard routines MAINPG and MODIO to complete his set. MXUNIT is also offered as a very adaptable, comprehensive routine for handling units. In a situation like the Machinery Weight Estimating Module in Chapter 8, one menu suffices for both input and output variables. Figure 2-5 illustrates the difference in flow between the main subroutine and the i/o variables for the two modules. Yet both use identical MAINPG and MODIO subprograms and only slightly different versions of MXUNIT.

When establishing menus a few rules must be kept in mind. The maximum number of menu items is twelve. When constructing the eight character menu item names, no blanks are allowed between characters but are acceptable after all the characters. If each menu item begins with a different character, only that one character has to be
entered by the user to enable DEX routine MENUIN to identify the selection.

Figure 2-5. Comparison of Module Input/Output Flow
3.1 Introduction

Upon entering the module level of DEX operation, the user needs to establish or verify the environment which suits his requirements. Four extended DEX library subroutines give him the capability to do this. They are:

1. DIALOG, which sets the type of module dialogue
2. SOURCE, which defines the location of information to be read
3. DESTIN, which defines the location to which information is to be written
4. MDMODE, which displays on the terminal the current status of the type of dialogue, the source, the destination and the file reference numbers for Fortran READ and WRITE to sequential files.

Each of these will be discussed in this chapter. Logical function CHKRNG, revised during this investigation, is also discussed here, although it will eventually become a DEX routine and not an extended DEX library function.

3.2 Subroutine DIALOG

3.2.1. Menu and Calling Parameter. Subroutine DIALOG would normally be called by a subroutine similar to the
subroutine MAINPG of the Cube Module. In that specific case it was done by selecting item "dialogue" from menu "MOD.MAIN". DIALOG has its own menu, and this is illustrated in Figure 3-1. Note the absence of an item "done".

$ + ------ +
$ + MENU +
$ + MOD. ALTR +
$ + ------ +
$ 1 + TERSE +
$ + ------ +
$ 2 + VERBOSE +
$ + ------ +

Figure 3-1. Menu "MOD. ALTR"
(for Subroutine DIALOG)
which appeared in most of the Cube Module menus. This is
typical for the subroutines of this chapter. Since the
user only makes one choice from these menus, the subpro-
grams automatically return control to the calling program
(e.g. MAINPG) without further action by the user.

It should also be called to the reader's attention
that the DEX library subroutines employ no labeled common
blocks. Rather, all variable values are transmitted by
means of the calling sequences. This was done to minimize
the possibility of inadvertently passing improper values
among the many subprograms which share the same commons.
The calling sequence presents a readily-checked format for
tracing errors.

The only variable in the calling sequence for DIALOG
is the logical variable MTERSE. If MTERSE=.TRUE., the
dialogue type is terse; it is .FALSE. the dialogue is
verbose. These two types of dialogues are offered to sat-
isfy the needs of the individual user. For the new user,
the verbose dialogue provides longer messages from the
module or DEX to facilitate learning how to use them. The
terse dialogue allows more rapid operations by the experi-
enced user.

Initialized in BLOCK DATA, MTERSE can have its value
changed by the correct selection from menu "MOD.ALTR".
This is accomplished by subroutine MENUIN. MENUIN is a
DEX integer function (see reference [5]) which converts a menu selection into an integer value. In DIALOG, the following statements are involved in this process:

```plaintext
DATA LMS/8/
DATA MENUNM/4HMOD.,4HALTR/
DATA NITEMS/2/
DATA ITEMS/4HTERS,4HE,
1   /4HVERB,4HOSE/
CALL STRPAK(MESS,LMS,4H<29HSELECT TYPE OF MOD
1ULE DIALOGUE<)
ITEM=MENUIN(MENUNM,NITEMS,ITEMS,MESS)
```

MENUIN prints both the prompting message MESS shown above and the message

*ENTER AN ITEM FROM MENU - MOD.ALTR

MENUIN sets ITEM equal to 1 if the user selects TERSE and 2 if he selects VERBOSE. ITEM is then used to branch to statements which make MTERSE either .TRUE. or .FALSE. as appropriate. Both branches return control to the calling program.

3.3 Subroutine SOURCE

3.3.1 Menu and Calling Sequence. Subroutine SOURCE, normally accessed from a subprogram like MAINPG, allows the user to select from the menu shown in Figure 3-2 the source of information to be read. Subroutine MENUIN assigns a value between 1 and 5 to IMODE depending on the user's choice.

* STRPAK is a DEX routine which inserts character strings into an integer array in the DEX format of four characters per word. See ref [5].
The calling sequence for SOURCE is as follows:

```
SUBROUTINE SOURCE(IMODE, DBFNME, IFILNM, RNRFIL, MTERSE, NCPW)
```

DBFNME is the database filename when the source is a database. IFILNM is the name of the sequential file if the user intends to read from a file. RNRFIL is an input variable defined in BLOCK DATA which represents the device number to be assigned to file IFILNM. The information in the file will be read in by one of the reading routines using the statement of the form

```
READ (RNRFIL, FORMAT) X
```

where X represents the variable being read. MTERSE and NCPW are required here for the preparation of the menu prompting message and other error messages supplied by subroutine source.

```
$ + ------- +
$ + MENU +
$ + SOURCE +
$ + ------- +
$ 1 + DATABASE +
$ + ------- +
$ 2 + TERMINAL +
$ + ------- +
$ 3 + SCREEN +
$ + ------- +
$ 4 + FILE +
$ + ------- +
$ 5 + DEFAULT +
$ + ------- +
```

Figure 3-2. Menu "SOURCE"
3.3.2. Operation of SOURCE. SOURCE commences execution by calling MENUIN to prompt the user to make a menu selection. MENUIN assigns an appropriate value to IMODE for branching.

If IMODE=1, the subroutine requires a database via DEX logical function DBOPEN. If there already is an open database, DBOPEN asks the user if he wants to use it, and if so, if he wants to save it ("save" has the usual meaning of writing a copy into memory from the buffers) before using it. If a database is "active", meaning a copy is in the buffers but there is also a saved copy in memory, the user is asked only if he wants to use it. If the user answers "no" to either of these questions, or if there is no active or open database, DBOPEN prompts the user for the name of either a new one to be created or an existing one to be opened. DBFNME is assigned that name. When this is done control is returned to the calling program.

If IMODE=2 (terminal) or 5 (default values) control simply returns to the calling program. If IMODE=3, indicating that the user hoped to employ DEX routines to read X-Y coordinates from a screen plot, he is informed that this mode of module input has not been implemented yet.

IMODE=4 causes the calling of subroutine GETFLNM which asks the user the name of the sequential file to be
read. Then logical function SETDEV assigns RNRFIL to IFILNM before control returns to the calling program.

3.4 Subroutine DESTIN

3.4.1 Menu and Calling Sequence. Subroutine DESTIN is similar to SOURCE except that there are only four possible choices, as seen in the menu illustration in Figure 3-3. The calling sequence contains six parameters:

```
SUBROUTINE DESTIN(OMODE, DBFNME, OFILNM, RNWFIL, MTERSE, NCPW)
```

OMODE is assigned a value between 1 and 4 by MENUIN. DBFNME is the same as in SOURCE. OFILNM is the name of the sequential file to which output is to be written. RNWFIL is an input variable defined in BLOCK DATA which represents the file reference number to be assigned to OFILNM for Fortran WRITE.

3.4.2 Operation of DESTIN. This subroutine behaves very similarly to SOURCE. If OMODE=1, DBOPEN checks for and opens as necessary a database and assigned DBFNME its name. If OMODE=2, control simply returns to the calling program. If the user selected the screen in order to do plotting, he is informed that this mode of output has not yet been implemented and is asked to make another selection. If OMODE=4, the user is asked the file name and it is assigned to OFILNM. Then logical function SETDEV assigns RNWFIL to the file and control is returned to the calling program.
3.5 Subroutine MDMODE

3.5.1 Function. Subroutine MDMODE informs the user of the current value of the following variables:

(1) MTERSE, which denotes the type of module dialogue

(2) IMODE, which denotes the source of information to be read

(3) OMODE, which denotes the destination of information to be written

(4) RNRFIL, which denotes the file reference number for module Fortran READ from a sequential file

(5) RNWFIL, which denotes the file reference number for module Fortran WRITE to a sequential file
After displaying this information, or an error message if a failure occurs, MDMODE returns control to the calling program.

3.5.2 Operation of MDMODE. The calling sequence for MDMODE contains only variables already discussed.

SUBROUTINE MDMODE(IMODE,OMODE,RNRFIL,RNWFIL,MTERSE,NCPW)

MDMODE does not display the numerical values of IMODE and OMODE but rather, for the user's convenience, it prints character strings defining the source and destination, e.g. "a dex created database". Similarly, for MTERSE, it prints "terse" and "verbose" vice ".TRUE." or ".FALSE.".

3.6 Logical Function CHKRNG

Logical function CHKRNG determines if an integer number is within the range defined by two other integer numbers. If not, an appropriate error message is issued and CHKRNG is returned .FALSE.. The calling sequence is as follows:

LOGICAL FUNCTION CHKRNG(NUMBER,MNEMON,MINNUM, MAXNUM,NCPW)

The routine checks if NUMBER is between the lower number MINNUM and the higher number MAXNUM. MNEMON is an 8-character mnemonic for NUMBER used in the error message. The use of CHKRNG avoids the need for the module author to include a message in his program.
CHAPTER 4
THE EXTENDED DEX LIBRARY READING RUTINES

4.1 General Description

4.1.1 Function. Information, which includes both input values and previously calculated output values, resides in four locations: DEX-created databases, the user himself, requested files and default data stored in the module. The user actually presents two distinct sources of information to the computer because of the two ways in which he can transmit data at the terminal: the first in the form of alphanumeric characters and the other by the location of a pen-pointer or cross-hairs on a plot on the screen. The latter method has not yet been implemented in DEX at MIT.

The function of the extended DEX library reading routines is to permit the transmission of that information to the module so that it can be written to other locations, such as the terminal for inspection, and/or used as input for the calculations being performed.

4.1.2 Organization. Eight logical functions comprise the reading routines group. They are listed here by the type of variable on which the operate:
<table>
<thead>
<tr>
<th>Integer Scalar</th>
<th>Real Scalar</th>
<th>Real Array</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISCLDR</td>
<td>RSCLDR</td>
<td>RAILDRA</td>
</tr>
<tr>
<td>ISCPMP</td>
<td>RVAPMP</td>
<td>RAIRED</td>
</tr>
<tr>
<td>ISREAD</td>
<td>RREAD</td>
<td>RAIREDD</td>
</tr>
</tbody>
</table>

The real scalar and array categories share RVAPMP.

These routines could be categorized horizontally by their specific jobs, as evidenced by the similarities in their names. The top three are called "loaders". These are the functions that appear in the module sub-programs where it is desired to read variables, and to the module author, for all intents and purposes, they are the reading routines. He does not need to be aware that they actually call the others to do the work.

If the user intends to input from the terminal he needs to be prompted for the correct information. This prompting message can be supplied by the module or it can be prepared by two routines in this group called "prompters". The loaders call the prompters as they are required. Having ensured that a prompting message exists, the loaders then call the third category, the "readers", to actually read in and assign the values to the input, output and working variables.
4.2 Integer Scalar Series

4.2.1 ISCLDR

4.2.1.1 Calling sequence. The calling sequence for logical function ISCLDR includes 17 variables listed here:

```
LOGICAL FUNCTION ISCLDR(IVAR, ALLFLG, MTERSE, IOMODE, NCPW, DBNAME, VITAL, MENUFL, MENUNM, NITEMS, ITEMS, PMPREP, PMES, PMORGN, RNFILE, FORMAT, DEFLT)
```

From the point of view of the function, IVAR is an output variable, ALLFLG is both input and output, and the remainder are all input variables.

IVAR is where the value of the integer sought will be stored. VITAL should be discussed next. An essential input variable, that is one required when reading input so that subsequent values can be entered correctly, is indicated when the logical variable VITAL has a value .TRUE.. This is important to the value of ALLFLG. ALLFLG indicates the status of the calling program "all" option (active when ALLFLG=.TRUE.). If both VITAL and ALLFLG are .TRUE. and IVAR is not read successfully, or the prompting message is not prepared successfully, ALLFLG will be changed to .FALSE.. ALLFLG's value can also be changed to .FALSE. during a reading evolution if IOMODE=4 (file source) and a premature end-of-file is encountered. Otherwise ALLFLG will retain its input value.
MTERSE indicates whether the module dialogue is terse or verbose. IOMODE denotes the source of input and corresponds to IMODE in the calling program. NCPW is the number of characters per word assumed by the DEX routines. DBNAME is where the database name of the integer is stored. Eight characters (including blanks) must be defined for this variable.

MENUFL is a logical variable which indicates if the integer sought will be a menu selection. If it is .TRUE., the next three variables must be defined. MENUNM is the eight-character name of the menu from which the selection will be made. NITEMS is the number of items in the menu. In the current version of DEX, the maximum number of menu items allowed is 12. ITEMS are the menu choices. Each choice is described by a name of eight characters (including blanks). Therefore, with four characters per word, ITEMS must be dimensioned as twice NITEMS where it first appears. The reader may wish to refer to the AUNIT series routines of Chapter 7 as examples of where menus are used to define integer values.

PMPREP is a logical variable which, when .TRUE., indicates that function ISCPMP will prepare the prompting message for the menu or INTIN. INTIN is a DEX routine which reads integer values entered from the terminal.
If PMPREP=.FALSE., the calling program supplies the prompting message in PMES.

PMES can be up to 64 characters long, and if less than 64 characters it must include the trim character, "<", last to signal the end of the string. Note that if PMPREP=.TRUE., PMES is undefined in the calling sequence of ISCLDR.

PMORGN ("prompting message origin") is a character string, usually the database comment, which identifies the variable sought. Like PMES, it must be 64 characters or less long and must include the trim character at its end if less than 64. When IOMODE=1, PMORGN will be compared with the database comment and differences brought to the user's attention.

RNFILE, corresponding to RNRFILE of the calling program, is the device number for Fortran READ if the integer is to be read from a file. FORMAT is the format for reading from the file. DEFAULT is the default value of the integer sought when IOMODE=5.

4.2.1.2 Characteristics. When the source of the integer is the terminal, ISCLDR invokes routine CHKRRNG to verify that NITEMS is between 1 and 12 inclusive when MENUL=.TRUE.. If PMPREP=.TRUE., ISCLDR invokes ISCPMP to prepare the prompting message. If ISCPMP is returned .FALSE., ISCLDR is also set to .FALSE. and
control returns to the calling program. Otherwise, for terminal input, as well as database, file and default value input, ISCLDR calls logical function ISREAD to do the reading. ISCLDR is set to .TRUE. or .FALSE. depending on the similar value of ISREAD and control returns to the calling program.

ISCLDR becomes .FALSE. if ISCPMP or ISREAD fails. It also becomes .FALSE. if the programmer has bypassed previous checks and set IOMODE=3. The user is informed by ISCLDR that integer scalars cannot be read in from a screen supplying x-y coordinates. The subprogram then checks VITAL and if .TRUE., it advises the user that the variable is essential for program continuation and must be corrected. Then, if ALLFLG=.TRUE., it is changed to .FALSE. and the user is advised that the calling program "all" option is aborted. ISCLDR is set to .FALSE. and control returns to the calling program.

4.2.2 ISCPMP

4.2.2.1 Calling sequence. Logical function ISCPMP is used to prepare a prompting message suitable for identifying the integer being sought when the source is the terminal (IOMODE=2) and PMPREP=.TRUE.. The calling sequence for this function is as follows:

\[
\text{LOGICAL FUNCTION ISCPMP(PMES,ALLFLG,TERSE,NCPW, DBNAME,VITAL,MENUFL,PMORGN)}
\]
These parameters have been described in section 4.2.1.1.
PMES is the output variable where the prompting message sought is stored. MTERSE is needed here to determine whether a brief or long message is to be prepared.
MENUFL is included here but presently it is not used by ISCPMP.

4.2.2.2 Characteristics. ISCPMP creates the prompting message by inserting the word "ENTER", followed by a short or long description of the variable depending on MTERSE, into PMES. When the dialogue is verbose, PMORGN is used as the indentifying string. The program scans it for the location of the trim character to determine its length. If the string is 51 characters or less, the message can be fit on one line. "ENTER" and "PMORGN" are copied into PMES. If, however, PMORGN has more than 64 characters, the prompting message must be two lines long. The word "ENTER" is printed immediately and PMORGN alone is copied onto PMES, to be printed by ISREAD.

When the dialogue is terse, PMES is created from "ENTER" and the database name, DBNAME, with a trim character added at the end.

If a failure occurs in preparing the prompting message, an error message is used. Then, when VITAL=.TRUE., the user is advised that the variable is
necessary for program input continuation and the problem must be corrected. When both VITAL and ALLFLG are .TRUE., the later changes to .FALSE. and the program advises the user that the calling program "all" option is aborted. Finally, ISCPMP is set to .FALSE. and control returns to ISCLDR. If the message preparation is successful, ISCPMP is set to .TRUE. before returning control to ISCLDR.

4.2.3 ISREAD

4.2.3.1 Calling sequence. Logical function ISREAD actually does the reading of the integer from the source defined by IOMODE. The calling sequence for ISREAD is as follows:

LOGICAL FUNCTION ISREAD(IVAR, ALLFLG, MTERSE, IOMODE, NCPW, DBNAME, VITAL, MENUFL, MENUVM, NITEMS, ITEMS, PMES, RNFILE, FORMAT, DEFLXT)

It is almost identical to ISCLDR, the difference being that PMPREP is no longer needed. PMES should be defined here and it must include the trim character if not 64 characters long.

4.2.3.2 Characteristics. ISREAD branches depending on the value of IOMODE. If the source of the integer is a database (IOMODE=1), ISREAD extracts the value using the DEX integer function IGET (see reference [5]). IGET provides error codes which, if other than
success, cause appropriate error feedback messages to be issued.

When IOMODE=2 (terminal input), and MENUFL=.TRUE., routine MENUIN is invoked to obtain IVAR from the menu selection. When a menu is not employed, DEX routine INTIN prompts the user to supply the integer value and reads what is entered at the terminal.

For IOMODE=4 (file source), ISREAD uses a Fortran READ statement, involving RNFILE and FORMAT, to read from the file. The program issues a warning if a premature end-of-file is encountered.

Whenever an error occurs in the reading, or if the user insists on trying IOMODE=3, VITAL and ALLFLG are checked. A warning that the variable is essential for program continuation is issued when VITAL=.TRUE.. ALLFLG will then be set to .FALSE. if not already. ALLFLG's value will also be changed, even if the variable is not essential, if either no open database was found (IOMODE=1) or a premature end-of-file is encountered (IOMODE=4), since further attempts at reading from these sources would be fruitless. In all cases of errors, ISREAD is set to .FALSE. and control is returned to the calling program. When the reading is successful, ISREAD is set to .TRUE..
4.3 **Real Scalar Series**

4.3.1 **Brief description.** The function of this group of routines is to permit the user to read real scalar values from any of the designated sources. Three logical functions make up this series: RSCLDR, RVAPMP and RSREAD. RVAPMP prepared prompting messages for reading both real scalars and real arrays from the terminal.

4.3.2 **RSCLDR**

4.3.2.1 **Calling sequence.** Logical function RSCLDR is invoked from the module. Its calling sequence is as follows:

```plaintext
LOGICAL FUNCTION RSCLDR(RVAR, ALLFLG, MTERSE, IOMODE, NCPW, DBNAME, UNITFM, UNITFA, UNITNM, VITAL PMPREP, PMES, PMORGN, RNFILE, FORMAT, DEFAULT)
```

Many of these variables are identical to those used in ISCLDR.

**RVAR** will be assigned the value, in program standard units, of the real number to be read. **UNITFM** and **UNITFA** are respectively the multiplicative and additive conversion factors which convert the real scalar read from the user input/output units to the program units. The conversion occurs in RSREAD. **UNITNM** is a 12-character version (including blanks) of the user i/o units, which is used in preparing the prompting message. If the real
scalar is not dimensional, UNITFM must be equal to 1.0, UNITFA must equal 0.0 and UNITNM is undefined.

PMORGN contains a string of 64 characters or less which identify the variable sought. If it has less than 64 characters it must have the trim character at the end of the string. If the real variable has units, PMORGN should contain the string "(?????????????)" into which UNITNM is inserted at the appropriate time. PMES must be dimensioned sufficiently large to accommodate PMORGN plus 6 characters (for "ENTER ") or 64 characters, whichever is less.

4.3.2.2 Characteristics. RSCLDR behaves similarly to ISCLDR. If the source is the terminal and PMPREP=.TRUE., it calls RVAPMP to prepare a prompting message. When the preparation is unsuccessful, RVAPMP is returned .FALSE.. RSCLDR becomes .FALSE. also and control returns to the calling program. If, however, the message preparation is successful, or for IOMODE=1, 4 or 5, RSCLDR calls RSREAD to read the real value. RSCLDR is set to .TRUE. or .FALSE. depending on the success or failure of RSREAD.

When IOMODE=3, RSCLDR informs the user that reading x-y coordinates from a graph on the screen in inappropriate for real scalar input. If VITAL=.TRUE., it issues an
advisory message to the user that the variable is essential for program continuation. ALLFLG is changed to .FALSE. if not already, with a warning that the "all" option is aborted, RSCLDR is set to .FALSE., and control returns to the calling program.

4.3.3 RVAPMP

4.3.3.1 Calling sequence. Both RSCLDR and RAILDR invoke RVAPMP to prepare a prompting message for real scalars and real arrays respectively. The calling sequence is:

    LOGICAL FUNCTION RVAPMP(PMES, ALLFLG, MTERSE, NCPW, DBNAME, UNITNM, VITAL, PMORGN)

These parameters are identical to those for ISCPMP except for UNITNM which carries the user i/o unit to be inserted into PMES.

4.3.3.2 Characteristics. When RVAPMP is invoked, PMORGN is scanned for the location of the trim character and for the string "(?????????????)" called UNITPT. The presence of UNITPT indicates that the variable is dimensional.

When the dialogue is verbose, the prompting message will be one line long if the trim character is found before the 59th position. The word "ENTER " and PMORGN are copied into PMES, and UNITNM is inserted
into UNITPT if applicable. This is why UNITNM must be 12 characters long. If PMORGN is longer than 58 characters, the prompting message will be two lines long. The string "ENTER" is printed immediately and PMORGN alone is copied into PMES, corrected by UNITNM if necessary. PMES will be the second line of the prompting message.

When the dialogue is terse, PMES is created from the word "ENTER ", followed by DBNAME, UNITPT adjusted by UNITNM, and lastly a trim character.

If the message preparation is successful, RVAPMP is set to .TRUE. and control returns to RSCLDR or RALLDR as applicable. If, however, an error occurs, VITAL is checked. The user is advised that the variable is essential when VITAL=.TRUE.. If ALLFLG also is .TRUE., it is changed to .FALSE. and the user told the "all" option is no longer in effect. In all cases of error, RVAPMP is returned as .FALSE. to the calling program. When successful, RVAPMP is set to .TRUE. before returning control.

4.3.4 RSREAD

4.3.4.1 Calling sequence. Logical function RSREAD reads the real scalar sought from one of the four valid sources of information. It includes 13 parameters in its calling sequence, almost all of which are identical to those in RSCLDR. The sequence is listed here:
LOGICAL FUNCTION RSREAD(RVAR, ALLFLG, MTERSE, IOMODE, NCPW, DBNAME, UNITFM, UNITFA, VITAL, PMES, RNFILE, FORMAT, DEFAULT)

Note the absence of UNITNM, PMPREP and PMORGN. These variables have either been used or incorporated into PMES, which is defined at this point.

4.3.4.2 Characteristics. If the source of information is indicated to be the user hoping to input x-y coordinates from the screen, he is informed that this mode of input is inappropriate for real scalar input.

The usual checks of VITAL and ALLFLG and messages occur.

The DEX routine RGET is used to read the real scalar from the database and it returns error codes for either success or the problems which can occur. The latter are brought to the user's attention. DEX routine REALIN (reference [5]) is used to read the real scalar from the terminal.

In cases where an error occurs, the user is informed if the variable is essential for input continuation.

When VITAL and ALLFLG are both .TRUE., ALLFLG is set to .FALSE. and the user is informed. Further, if IOMODE=1 but there is no open database, or IOMODE=4 but a premature end-of-file is encountered, ALLFLG is set to .FALSE. regardless of the value of VITAL.
RSREAD is set to .TRUE. if successful and .FALSE. if an error occurs, and control is then returned to the calling program. When successful, prior to returning control, the value read is converted into program standard units by the expression
\[ RVAL = RVAL \times UNITFM + UNITFA \]

4.4 Real Array Series

4.4.1 Brief description. The real array reading routines include RALLDR and RAIRED, plus RVAPMP which is shared with the real scalar series. Their function is to read one dimension real arrays, up to the current DEX limit of 200 elements, from any of the four valid sources of input. The reading of x-y coordinates from the screen, while legitimate for an array since it can store a pair of real numbers, has not been implemented yet and the user is advised of this. The next generation of DEX at MIT will include routines for reading and writing two arrays simultaneously for graphics tasks.

4.4.2 RALLDR.

4.4.2.1 Calling sequence. Logical function RALLDR invokes RAIRED to read a real array from the designated source. It also calls RVAPMP as necessary to prepare a prompting message for terminal input. Its calling sequence, listed here, has a few parameters not seen in RSCLDR:
LOGICAL FUNCTION RAILDR(RLARR,ALLFLG,NGOT,
MTERSE,IOMODE,NCPW,DBNAME,
MXTOGT,UNITFM,UNITFA,UNITNM,
VITAL
PMPREP,PMES,PMORGN,RNFILE,FORMAT,
NDEF,DEFAULT)

RLARR is the real array that will store the elements, in
program standard units, that are read. MXTOGT repre-
sents the maximum number of elements to be read into
RLARR, and is the dimensioned size of RLARR. NGOT is
the number of elements actually read and can be less than
or equal to MXTOGT. NGOT need not be defined when RAILDR
is invoked. NDEF is the number of default values and is
provided to allow the default condition to be smaller
than the maximum capability of the array.

4.4.2.2 Characteristics. When IOMODE=2
and PMPREP=.TRUE., RAILDR invokes RVAPMP to prepare
PMES. When RVAPMP is successful, and for the other
valid sources of input (IOMODE=1, 4 or 5), RAILDR calls
RAIRED. If either RVAPMP or RAIRED are not successful,
RAILDR is set equal to .FALSE.. This also occurs when
IOMODE=3. It is set to .TRUE. otherwise and control is
returned to the calling program.

4.4.3 RAIRED

4.4.3.1 Calling sequence. The calling
sequence for RAIRED is as follows:
LOGICAL FUNCTION RA1RED(R1ARR, ALLFLG, NGOT, MTERSE, IOMODE, NCPW, DBNAME, MXTOGT, UNITFM, UNITFA, VITAL, PMES, RNFILE, FORMAT, NDEF, DEFLT)

Like in RSREAD, UNITNM, PMPREP and PMORGN are no longer required. MXTOGT represents the maximum number of elements to be read into R1ARR, always starting at position 1. NGOT is defined in RA1RED.

4.4.3.2 Characteristics. RA1RED first employs DEX routine CHKRNG to verify that MXTOGT is not greater than the maximum DEX array size (currently 200 elements).

If IOMODE=3, the real array is not read and the appropriate checks of VITAL and ALLFLG and message issuing occur.

The reading of an array from a database is accomplished by DEX routine AGET, which seeks MXTOGT elements from the database array. AGET returns six possible result codes. RCODE=0 is simple success, that is, there were MXTOGT elements stored in the database array. NGOT is set equal to MXTOGT. If RCODE=1, there was no open database. This causes ALLFLG to change to .FALSE. if it was .TRUE. and RA1RED to be set to .FALSE.. RA1RED is also set to .FALSE. if the variable does not exist in the database (RCODE=2), if it is not an array (RCODE=3), or if it was undefined (RCODE=4). When the number of
elements requested exceeds the number stored (RCODE=5), the extra elements in RLARR are set equal to 0.0 but NGOT is set equal to the number stored. When the number of elements requested is less than the number of elements stored (RCODE=6), the first MXTOGT elements are read into RLARR and NGOT is set equal to MXTOGT. The user is advised in these circumstances of what has occurred.

When reading from the terminal, DEX logical function REALIN is invoked with the following statement:

\[
\text{LOGVAL} = \text{REALIN}(\text{MXTOGT}, \text{NEED}, \text{RLARR}, \text{PMES})
\]

A prompting message asks the user to input up to MXTOGT values. NEED represents the difference between the number of elements read in and MXTOGT. If no elements are read (NEED=MXTOGT) the reading is considered a failure. The reading is considered successful if at least one number is entered. NGOT is set equal to the number of elements entered.

The user is advised if a premature end-of-file is encountered when reading from a file.

When failures occur, VITAL and ALLFLG are processed as usual. Then RAIRED is set to .FALSE. and control returns to the calling program. If the reading is successful, the elements are converted into program
standard units by a DO loop for NGOT iterations with the statement

\[ \text{RLARR}(I) = \text{RLARR}(I) \times \text{UNITFM} + \text{UNITFA} \]

Then RAIRED is set to .TRUE. and control returns to the calling program.
CHAPTER 5
THE EXTENDED DEX LIBRARY EDITING Routines

5.1 General Description
At some point in the operation of a program the user may decide that he wants to change the value of one or many variables. It may be that the value read as input is incorrect, or he wants to see the effect of changing one variable on the output. He may even decide he does not like the answer given by his program and, before storing it in a database or file, wishes to exchange it for another value he has. For whatever reason, the user requires the capability to enter the new value at the terminal. The editing routines were developed for this purpose.

Currently there are two logical functions in this category, ISCEDT and RSCEDT, with a third RARED, scheduled to be developed for the next version of the extended DEX library. ISCEDT allows the editing of integer scalar variables, RSCEDT allows the editing of real scalars, and RARED will allow the changing of elements in a real array.
5.2 Logical Function ISCEDT

5.2.1 Calling sequence. Logical function ISCEDT would be invoked by a module subprogram, normally when IFLAG is set equal to 2 in a subroutine like MAINPG described in Chapter 2. The calling sequence includes 15 parameters listed here:

\[
\text{LOGICAL FUNCTION ISCEDT(NEWVAR,ALLFLG, MTERSE,NCPW,DBNAME, MENUFL,MENUM,NITEMS,ITEMS, PMPREP,PMES,PMORGN,RNFILE, FORMAT,DEFAUL)}
\]

For ISCEDT, the first parameter is an output variable, ALLFLG is both input and output, and the remainder are all input variables.

NEWVAR will store the new value of the integer variable being changed. ALLFLG indicates the status of the calling program "all" option. Its value may be changed from .TRUE. to .FALSE. during the editing sequence.

Logical variable MTERSE indicates the type of module dialogue: terse or verbose. NCPW represents the number of characters per word assumed by DEX routines, and is dependent upon the particular computer in use DBNAME is the 8-character database name of the integer sought.

When MENUFL=.TRUE., the integer being sought is a menu selection. The eight-character menu name is stored
in MENUNM. NITEMS is the number of menu items, and it cannot exceed 12. ITEMS is where the menu choices are stored. Each item is described by an eight-character name (including blanks), so that, at four characters per word, ITEMS should be dimensioned as 2*NITEMS.

Since the new integer value will be entered at the terminal, a prompting message is required. PMPREP is a logical variable which indicates if the program is to prepare the message (.TRUE.) or if it is supplied by the calling program (.FALSE.). PMES is where the prompting message is stored. It can be up to 64 characters long, and if less it must include the trim character, "<", at its end. If PMPREP=.TRUE., PMES is undefined in ISCEDT.

PMORGN stores the information describing the variable in question. It is typically the database comment. PMORGN can be up to 64 characters long and requires the trim character if less. Because PMORGN is used to prepare the prompting message when the dialogue is verbose, if PMPREP=.FALSE., it need not be defined in ISCEDT.

RNFILE is the reference number for reading from a sequential file using Fortran READ and corresponds to RNRFILE in the calling program. FORMAT stores the format for reading from a file. DEFAUT is the default value of the integer in question.
5.2.2 Operation. The task of ISCEDT is actually quite simple. In order to read a new integer from the terminal, ISCEDT merely invokes ISCLDR, using the variable EDMODE, which has a value of 2, in place of IMODE. ISCLDR then prepares a prompting message, if necessary, and calls ISREAD to read the value entered. ISCEDT is set to the same value with which ISCLDR is returned (i.e., .TRUE. for success), and control returns to the calling program.

In calling ISCLDR, ISCEDT defines the parameter VITAL as .TRUE. in all cases. This stems from the policy that if the user wishes to correct a value, he really wants to correct it for program continuation. Failure of any of the integer reading routines would change ALLFLG if it was .TRUE. when ISCEDT was invoked.

It may not be readily apparent to the reader, but because the source will always be defined as the terminal by ISCEDT, the calling parameters RNFILE, FORMAT and DEFALT, used only when IMODE=4 or 5, need not be defined here. In fact, dummy variables could have been used. It was decided to use the correct variables in the calling sequence to avoid potential errors by creating more variables. The ones used should already be available in the module, being needed for reading and writing integer values.
5.3 **Logical Function RSCEDT**

5.3.1 **Calling parameters.** Logical function RSCEDT is invoked by the module to permit the editing of a real scalar variable. Its calling sequence is as follows:

```
LOGICAL FUNCTION RSCEDT(NEWVAR, ALLFLG, MTERSE, NCPW, DBNAME, UNITFM, UNITFA, UNITNM, PMPREP, PMES, PMORGN, RNFILE, FORMAT, DEFAUL)
```

All of the parameters except NEWVAR are input variables, and ALLFLG is both an input and output variable.

Most of these parameters are identical to those used in ISCEDT. Three are new. UNITFM and UNITFA are respectively the multiplicative and additive conversion factors for converting the real scalar read in user i/o units to the value NEWVAR in program standard units. UNITNM is a 12-character version (including blanks) of the user input/output units of the variable, and is used in preparing the prompting message when PMPREP=.TRUE..

5.3.2 **Operation.** RSCEDT behaves very similarly to ISCEDT. It calls RSCLDR with the variable EDMODE, defined as 2, in lieu of IMODE (terminal source) and VITAL specified as .TRUE.. The real scalar reading routines accept a value entered at the terminal, convert it to program standard units and store it in NEWVAR. If a failure occurs, ALLFLG changes to .FALSE. if it was .TRUE. when RSCEDT was invoked.
RSCEDT is set to the same value as RSCLDR (.TRUE. if NEWVAR is successfully read in, .FALSE. if an error occurred in the reading sequence) and control is returned to the calling program.

One can see that the editing routines are essentially another version of the reading routines. Current plans for the array editor anticipate extending this capability to include reorganizing the entries and inserting new values for whichever element or group of elements is specified by the user. Further, it is hoped that through the editor, it will be possible to execute the calculation subprograms only for those elements changed in order to reduce computing costs. It may be that these options will be operated by the user by means of editing menus also. In short, the goal will be to simulate the editor capability of an interactive system such as CMS at MIT.
CHAPTER 6
THE EXTENDED DEX LIBRARY WRITING Routines

6.1 General Description

Once information has been read, edited or computed, unless it is to be used as input for computations, it is necessary to transmit it to one of three possible destinations: a DEX-created database, the terminal or a sequential file. Further, for our purposes, a distinction shall be made between writing alphanumeric characters on the terminal screen via DEX routines and plotting the information on the screen as a graph. In the current version of DEX at MIT the plotting option is not yet implemented.

The function of the extended DEX library writing routines, described in this chapter, is to permit the transmission of the information to the three valid destinations. Eight logical functions comprise this group of routines, and, like the reading routines, they can be categorized by either type of variable handled or by function. They are listed here by the first method:

<table>
<thead>
<tr>
<th>Integer</th>
<th>Real</th>
<th>Real</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scalar</td>
<td>Scalar</td>
<td>Array</td>
</tr>
<tr>
<td>ISCDMP</td>
<td>RSCDMP</td>
<td>RARDMP</td>
</tr>
<tr>
<td>ISDSCR</td>
<td>RVDSCR</td>
<td></td>
</tr>
<tr>
<td>ISRITE</td>
<td>RSRITE</td>
<td>RARITE</td>
</tr>
</tbody>
</table>

Both the real scalar and real array series share RVDSCR.
The top routines, called the "dumpers", serve a function similar to that of the loaders of the reading routines. They screen out requests to perform plotting, call the "descriptors", if necessary, to prepare description messages for identifying the values when writing on the terminal, and invoke the "writers" to do the actual writing to the destination.

One general observation concerning the writing routines which is best made here is that the concept of "essential" variables, which was introduced in the chapter on the reading routines, is not employed. This affects the execution of a calling program all option. The premise is that when writing all the values from a menu, the failure to write one should not prevent the writing of the remainder. The user can go back and analyze why the one was not successful without having to rewrite all of the variables from the menu. The only case where the all option is aborted is where the destination is a database and no database is found open. Rather than getting a string of similar messages announcing this fact, the writing sequence is halted.
AN INVESTIGATION INTO THE USE OF DATA BASES IN COMPUTER-AIDED N--ETC(U)

JUN 81 R C CELOTTO
6.2 Integer Scalar Series

6.2.1 ISCDMP

6.2.1.1 Calling sequence. Logical function ISCDMP is the supervisory subroutine for writing integer scalar values and is the subroutine which appears in module subprograms. The calling sequence contains eleven parameters and is listed here:

\[
\text{LOGICAL FUNCTION ISCDMP(ALLFLG, MTERSE, IOMODE, NCPW, IVAR, DBNAME, DMORGN, DMPREP, DMES, RNFILE, FORMAT)}
\]

In accordance with DEX practices, the output variables come first in the calling sequence. ALLFLG is both an input and output variable for ISCDMP, being defined at the invocation and capable of having a different value when ISCDMP is returned to the module subprogram. ALLFLG contains the information about the calling program all option.

The remaining parameters are exclusively input variables from the point of view of ISCDMP. MTERSE indicates the type of module dialogue. NCAW is the number of characters per word assumed by the DEX routines. IOMODE corresponds to OMODE in the module calling program and represents the destination of the information to be written. As a reminder, its values are repeated here:

IOMODE=1: a DEX-created database
IOMODE=2: the terminal using DEX routines
IOMODE=3: the screen using plotting routines
IOMODE=4: a sequential file using a formatted WRITE statement

IVAR is the integer value which is to be written.

DBNAME is the 8-character database name of the variable.

DMORGN is a string of up to 64 characters which describes the variables. It is usually the database comment. If it has less than 64 characters it must include the trim character "<". The routines in this series assume that integer variables have no units.

DMPREP is a logical variable which, if .TRUE., means that the description message is to be prepared by ISDSCR. In this case, DMES is undefined. DMES is where the description message is stored. If the dialogue is verbose it can be up to 64 characters by, whereas if the dialogue is terse it will only contain DBNAME. If DBPREP=.FALSE., DMES must be provided by the module and must include the trim character if it is less than 64 characters long.

RNFILE is the file reference number for writing to a sequential file. It corresponds to RNWFIL in the calling program. FORMAT contains the format to be used to write the integer available to the file.
6.2.1.2 Characteristics. ISCDMP first checks the destination pointer. If IOMODE=3, the user is informed that plotting is an improper mode of output for an integer scalar value and ISCDMP is set to .FALSE. before returning control to the calling program.

If IOMODE=2 and DMPREP=.TRUE., ISCDMP calls ISDSCR to prepare the description message for identifying the integer when it is written to the terminal. When this is successfully accomplished, or when IOMODE=1 or 4, ISCDMP invokes ISRITE to actually perform the writing.

If either ISDSCR or ISRITE is returned .FALSE., ISCDMP is also set to .FALSE. and control is returned to the calling program. Otherwise, it is set to .TRUE. prior to returning control.

When invoking ISRITE, a new logical variable, DBCHNG, is introduced. It is included in anticipation of future capabilities of DEX and indicates when a change is made to a database value. This will alert the user to check other variables which are dependent on the value of the integer being written. Currently DBCHNG is initialized as .FALSE. in ISCDMP.

6.2.2 ISDSCR

6.2.2.1 Calling sequence. Logical function ISDSCR prepares a description message suitable for
identifying the integer available when it is to be written on the terminal. Its calling sequence lists the pertinent parameters provided by ISCDMP:

LOGICAL FUNCTION ISDSCR(DMES,MTERSE,NCPW,DBNAME,DMORGN)
The value of logical variable MTERSE dictates whether the description message, to be stored in DMES, will be brief or long. NCPW is used by the DEX routines which manipulate character strings to produce the message.

6.2.2.2 Characteristics. If the dialogue is verbose, DMORGN, which contains a string of up to 64 characters describing the integer variable in question, is inserted into DMES. For terse dialogue, DBNAME is copied into the description message. If the insertion is successful, ISDSCR is set to .TRUE. and control returns to ISCDMP. If not, an error message is issued and ISDSCR is returned .FALSE..

6.2.3 ISRITE

6.2.31. Calling sequence. Logical function ISRITE actually writes the integer available to the specified destination. Its calling sequence is as follows:

LOGICAL FUNCTION ISRITE(ALLFLG,DBCHNG,MTERSE,IOMODE,
                          NCPW,
                          IVAR,DBNAME,DMORGN,DMES,RNFILE,
                          FORMAT)
Logical variables ALLFLG and DBCHNG are defined when ISRITE is invoked. DBCHNG is always .FALSE., and will become .TRUE. if a change is made to the integer variable DBNAME in the database. DMES is always defined when ISRITE is invoked so DMPREP is no longer needed. DMORGN is required because it may be used for comparison with the database comment.

6.2.3.2 Characteristics. If the destination of the integer available is a database, ISRITE first attempts to extract an existing value by DEX routine IGET (reference [5]). If no database is open, ALLFLG is changed to .FALSE. if it was not already, with an appropriate message being issued to the user. If the variable is defined in the database, and it is different from the integer available, both are presented for the user's inspection. The user then specifies which is to be placed in the database. The new value is inserted by DEX routine IPUT (Reference [5]) and DBCHNG becomes .TRUE.. If the variable was not defined, the new value is automatically inserted. Once this is accomplished, the database comment is compared to DMORGN and, if different, they are presented for the user's inspection. Again, he specifies which is to be the final database comment.
When writing to the terminal, an output message is created from DMES, the string " = " and the integer value. The entire message is then printed by DEX routine MESOUT.

If IOMODE=3, the user is informed that plotting is an improper mode of output for an integer scalar. In this case, and other cases where the writing is unsuccessful, ISRITE is returned .FALSE. to the calling program. Otherwise it is set to .TRUE. prior to returning control.

6.3 Real Scalar Series

6.3.1 RSCDMP

6.3.1.1 Calling Sequence. RSCDMP is the subprogram that normally appears in the module for writing a real scalar value to the designated source. Its calling sequence includes 14 parameters:

LOGICAL FUNCTION RSCDMP(ALLFLG,MTERSE,IOMODE,NCPW, RVAR,DBNAME,UNITFM,UNITFA, UNITNM, DMPREP,DMES,DMORGN,RNFIL, FORMAT)

All of these parameters are input variables with respect to RSCDMP except ALLFLG, whose value may be changed during the writing sequence. RVAR stores the value of the real scalar available to be written. UNITFM and UNITFA are respectively the multiplicative and additive
conversion factors for converting the real value in program standard units to input/output units prior to the writing. UNITNM is a 12-character version (including blanks) of the input/output unit name. DMPREP indicates if the description message, DMES, is to be prepared by RVDSCR.

DBNAME is an 8-character name of the real scalar and DMORGN is a string of up to 64 characters which identifies the variable. If the variable is dimensioned, DMORGN contains the string "(?????????????)", referred to as UNITPT, into which UNITNM will be inserted. DMORGN must include the trim character if it is less than 64 characters long. RNFILE, corresponding to RNWFIL of the module calling program, is the file writing device number. FORMAT contains the format for writing to a file with a formatting Fortram WRITE statement.

6.3.1.2 Characteristics. RSCDMP has three tasks: to screen out requests to plot a real scalar, to call RVDSCR if DMPREP=.TRUE. and IOMODE=2, and to call RSRITE to perform the actual writing. If IOMODE=3, RSCDMP issues a message informing the user that this is not possible. In this case, if either RSDSCR or RSRITE are returned .FALSE., RSCDMP is set to .FALSE. and control is returned to the calling program. If the called
functions are returned .TRUE., RSCDMP is also set to .TRUE. before returning control to the calling program.

In invoking logical function RSRITE, the logical variable DBCHNG is introduced. It is initialized in RSCDMP as .FALSE.. If, when executing RSRITE the variable value is changed in the database, DBCHNG is returned to RSCDMP as .TRUE.. In future versions of DEX, RSCDMP will pass the value of DBCHNG back to the calling program to alert the user to check other variables which are dependent on RVAR.

6.3.2 RVDSCR

6.3.2.1 Calling sequence. In the same manner as RVAPMP, RVDSCR is shared by both the real scalar and real array series. Its function is to prepare a description message suitable for identifying the values being written on the terminal. It is invoked by either RSCDMP or RARDMP using the following calling sequence:

LOGICAL FUNCTION RVDSCR(DMES,MTERSE,NCPW,DBNAME,DMORGN,UNITNM)

DMES is undefined when RVDSCR is invoked. The other parameters are all input variables from this function's point of view. They have all been described in either section 6.2.1.1 or 6.3.1.1.
6.3.2.2 Characteristics. If the module dialogue is verbose (MTERSE=.FALSE.), the description message is formed by copying DMORGN into DMES. If the real scalar being written has units, RVDSCR inserts the 12-character unit name UNITNM into the string UNITPT, "(?????????????)", which is now in DMES by virtue of having been in DMORGN. If the dialogue is terse, then RVDSCR copies DBNAME into DMES. It then scans DMORGN for UNITPT, and if it finds it, copies UNITPT into DMES following DBNAME and replaces the question marks with UNITNM. If DMES is successfully prepared, RVDSCR is returned .TRUE.. Otherwise it issues an error message, is set to .FALSE., and returns control to the calling program (either RSCDMP or RARDMP).

6.3.3 RSRITE

6.3.3.1 Calling sequence. Logical function RSRITE is used to write a real scalar to the valid specified destination. Its calling sequence is as follows:

```
LOGICAL FUNCTION RSRITE(ALLFLG, DBCHNG, MTERSE, IOMODE, NCPW, RVAR, DBNAME, UNITFM, UNITFA, UNITNM, DMORGN, DMES, RNFILE, FORMAT)
```

This is similar to RSCDMP with three exceptions. DBCHNG is a logical variable which is always .FALSE. when RSRITE is invoked. DMPREP is no longer needed since in all
cases DMES is now defined. DMORGN is still required because it will be compared with the database comment.

6.3.3.2 Characteristics. RSRITE first converts the real scalar value from program standard units to user input/output units, stored in variable TVAR, by the statement:

$$TVAR = (RVAR-UNITFA)/UNITFM$$

If IOMODE=1, the database is first checked to see if a value in question already exists. If the database is found closed, RSRITE alerts the user and changes ALLFLG to .FALSE. if it was not already so. If the variable does not exist in the database, it is inserted using DBNAME, TVAR and DMORGN (corrected for units if applicable) as its name, value and database comment. If the variable exists but is not a real scalar, RSRITE informs the user and is set to .FALSE..

If the variable exists and is defined, its value is compared to TVAR and the user is presented with both if there is a difference. He then is asked which value he wants in the database and the chosen one is written (or left) in. RSRITE then compares the existing database comment to the one specified by "RGN" and writes them both for the user's inspection if they are different. The user then specifies which one is to be the
database comment. This step is crucial in insuring that 
the correct units for TVAR exist in the database comment.

When writing to the terminal, an output message is 
constructed using DMES, the string " = " and the real 
value. This message is then printed by DEX routine 
MESOUT.

If IOMODE=3 despite the previous checks, the user 
is informed that a plot cannot be used for writing a 
real scalar value, and RSRITE is returned .FALSE. . 
RSRITE is set to .FALSE. in all cases where the real 
value is not successfully written and control is then 
returned to the calling program. Otherwise it is 
returned .TRUE. to RSCDMP.

6.4 Real Array Series
6.4.1 RARDMP

6.4.1.1 Calling sequence. Logical function 
RARDMP is used in the module subprogram for writing a 
real array. It has the same three functions as RSCDMP 
and ISCDMP: to screen out requests to plot graphs, to 
call RVDSGR if needed to prepare a description message, 
and to call RARITE to actually do the writing. It has 
the following calling sequence:
LOGICAL FUNCTION RARDMP(ALLFLG, MTERSE, IOMODE, NCPW, RIARR, DBNAME, NFROM, NTO, UNITFM, UNITFA, UNITNM, DMPREP, DMES, DMORGN, RNFILE, FORMAT)

A few new parameters deserve explanation. RIARR stores the array elements, in program standard units if they have dimensions. The array corresponding to RIARR in the module should be dimensioned as large as the value MXTOGT used in RAILDR for reading the array.

NFROM represents the position in the array at which writing commences. It should always have a value of 1, specified in the module calling program, except possibly when writing to the terminal. If the user is in the "editing" versus the "writing" mode of operation, he may desire to write only part of an array on the terminal. In this case the editing routine specifies NFROM to be a value from 1 to NTO inclusive prior to invoking RARDMP. NFROM should be 1 when writing the array on the terminal when not in the "editing" mode.

NOT represents the number of elements to be written in all cases but one. This exception occurs when writing to the terminal in "editing" mode, when NTO indicates the last element in the array to be written. Other than this case, NTO corresponds to the value NGOT obtained when reading the array with the reading routines (i.e.,
the actual number of elements read into the array represented by RARR), and may be less than MXTOGT.

The other parameters in the calling sequence are the same as those in RSCDMP.

6.4.1.2 Characteristics. If IOMODE=2 and DMPREP=.TRUE., RARDMP invokes RVDSCR to prepare a description message suitable for identifying the array being written to the terminal. If RVDSCR is successful, and for IOMODE=1 and 4, RARDMP invokes RARITE with the statement

\[
\text{LOGVAL=RARITE(ALLFLG,DBCHNG,MTERSE,IMODE,NCPW, RARR,DBNAME,UNITF,UNITFM,UNITFA, UNITNM, DMORGN,DMES,RFILE,FORMAT)}
\]

In this version of DEX, DBCHNG is initialized .FALSE. in RARDMP. If a change is made to a database array by RARITE it will be changed to .TRUE.. In future versions RARDMP will pass the value of DBCHNG back to the user via its calling sequence, to alert the user who may wish to verify other variables dependent upon this array.

If IOMODE=3, RARDMP informs the user that it cannot be used to write a real array. In this case, or if RVDSCR or RARITE is returned .FALSE., RARDMP is set to .FALSE. prior to returning control to the calling program. If the two called functions are successful, RARAMP is also set to .TRUE..
6.4.2 RARITE. Since the calling sequence for RARITE has been described above, this section will only discuss RARITE's characteristics. The task of this logical function is to write the real array elements available to the proper specified destination. The first action it takes is to convert the elements in program standard units to input/output units and store them in a temporary array. This is done with a DO loop from NFROM to NTO and the statement

\[RTARR(I) = (R_\text{LARR}(I) - \text{UNITFA}) / \text{UNITFM}\]

The elements in RTARR are in the units described by UNITNM.

If the destination is a database (IOMODE=1), it is desired to compare the existing array with the new one. RARITE first attempts to extract the existing array and store it in a working array RXARR using DEX routine AGET by the calling sequence

\[\text{LOGVAL} = \text{AGET(DBNAME, RXARR, NTO, NSTORD, RCODE)}\]

There are six possible result codes returned by AGET. If RCODE=0, AGET was completely successful in that the number of elements stored in the database array (NSTORD) is equal to the number requested (NTO), which is also the number of new elements to be stored. If RCODE=1, the database was not open. This will cause ALLFLG to change to .FALSE. if it was not already, aborting the calling program all option.
If DBNAME does not exist in the database (RCODE=2),
it is created with DEX function DBVINS, and RTARR is
stored in it. RTARR is also immediately stored if the
array DBNAME exists but has no datum stored in it (RCODE=
4). If DBNAME is not a real array (RCODE=3), the user
is informed.

The final two result codes are more diabolical.
If the number of elements stored in the database array
is less than the number requested by AGET (RCODE=5), the
elements that do exist, plus zeros up to NTO elements,
are stored in RXARR. The user is advised that this has
occurred, that comparison of the existing values in
RXARR and the new values in RTARR can be accomplished,
but that the new values cannot be stored if the user
decided they are the ones desired. This is because the
storing of an array is performed by DEX routine APUT via
the statement

\[ \text{LOGVAL}=\text{APUT(DBNAME,RTARR,NTO,NSTORD,RCODE)} \]

Unless NTO=NSTORD, the storing will not occur. All is
not lost, however! The user can proceed back to the
module calling program, exit to the DEX level via the
"$" command and delete the array with the DEX editing
capability. He can then return to the module and write
the array into the database when AGET returns RCODE=2.
The other problem occurs when the number requested is less than the number stored (RCODE=6). In this case NTO elements are stored in RXARR for comparison, but for the same reason as above, the user is advised that storing the new values will not be possible.

Once RXARR is established (RCODE=0, 5 or 6), a comparison between its elements and those of RTARR is conducted. The criteria for difference is $1.0 \times 10^{-6}$. The user is informed of how many differences were found and asked if an inspection of all the values is desired. A partial review is not possible. If the user responds affirmatively, the values are listed. UNITNM is printed in the heading. RARITE then asks the user to specify which group of values (all old or all new) is desired. If the user chooses to insert the new values, DBCHNG is set to .TRUE. and APUT is called to store the values. Error messages will be issued if NTO does not equal NSTORD.

If the writing is successful, or if the old values are retained, RARITE proceeds to compare the database comment to DMORGN, corrected for units, if applicable. When not the same, it prints both and asks the user to decide which is correct, storing the one chosen as the database comment. If there was no comment already in the database, DMORGN is automatically inserted.
When the destination is the database, the writing is considered successful only when all the new values are successfully stored or the old values retained. All the other possibilities result in RARITE being returned to RARDMP as .FALSE..

If IOMODE=2, the description message, DMES, is printed on the terminal, and then the array is listed from position NFROM to NTO. If IOMODE=3, the user is informed that plotting the array cannot be accomplished and RARITE is set to .FALSE..

When IOMODE=4, the array is written to a sequential file by a DO loop from 1 to NTO with the statement

```
WRITE(RNFILE,FORMAT) NTO, (RTARR(I), I=1,NTO)
```

In the cases where the writing is successful RARITE is set to .TRUE. and control is returned to the calling program.
CHAPTER 7
THE EXTENDED DEX LIBRARY UNIT ROUTINES

7.1 General Description

The module author will invariably write the computational subprograms of the module in the unit system with which he is most familiar. Frequently, it is not the system that the user of the module prefers. The tenet of the DEX philosophy to make modules convenient to use dictated that this problem be addressed. The result was the development of a group of subprograms in the extended DEX library which allow the user to choose from a reasonable selection, the units for input and output purposes for five basic types of measurement, plane angle, force, length, temperature, and time — and combinations thereof.

The twenty-two extended DEX library unit routines can be divided into three categories:

(i) Five subroutines which enable the user to choose from the options available the preferred input/output (i/o) units.

(ii) Five logical functions which enable the module to obtain conversion factors which convert the five basic user-specified (i/o) units into the program standard units (p.s.u.) and to obtain the unit names of the i/o units for use in prompting and des-
cription messages on the terminal and database comments.

(iii) Twelve logical functions which enable the module to obtain the conversion factors and unit names or special names for combinations of the basic units.

This chapter will examine each category.

7.2 The I/O Unit Specifiers

7.2.1 General Description. The extended DEX library includes five subroutines which enable the user to read, edit, or write the five basic units he wishes to use for input and output. These are listed here:

- AUNIT (plane angle)
- FUNIT (force)
- LUNIT (length)
- TPUNIT (temperature)
- TUNIT (time)

The user must choose from the units offered by the particular subroutine menu options. These choices were included in anticipation of the possible needs of most users. Table 7-1 lists the choices available.

7.2.2 Characteristics of a Typical Subroutine.

In execution, the five subroutines simply call ISCLDR to read the input/output unit indicator, ISCEDT to edit the i/o unit indicator, or ISCDMP to write the i/o unit indicator. Because all five subroutines are structured identically, only one, AUNIT, will be described in detail.
Table 7-1. I/O Unit Specifier Subroutines, Menu Names and Units Available

<table>
<thead>
<tr>
<th>AUNIT</th>
<th>FUNIT</th>
<th>LUNIT</th>
<th>TPUNIT</th>
<th>TUNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANG.UNIT</td>
<td>FOR.UNIT</td>
<td>LEN.UNIT</td>
<td>TEMPUNIT</td>
<td>TIMEUNIT</td>
</tr>
<tr>
<td>1 cycle</td>
<td>poundal</td>
<td>inch</td>
<td>Celsius deg.</td>
<td>second</td>
</tr>
<tr>
<td>2 radian</td>
<td>poundforce</td>
<td>foot</td>
<td>Fahrenheit deg.</td>
<td>minute</td>
</tr>
<tr>
<td>3 degree (ang)</td>
<td>short ton</td>
<td>statute mile</td>
<td>Kelvin deg.</td>
<td>hour</td>
</tr>
<tr>
<td>4 minute (ang)</td>
<td>long ton</td>
<td>nautical mi.</td>
<td>Rankine deg.</td>
<td>week</td>
</tr>
<tr>
<td>5 second (ang)</td>
<td>dyne</td>
<td>millimeter</td>
<td></td>
<td>month (30 day)</td>
</tr>
<tr>
<td>6</td>
<td>newton</td>
<td>centimeter</td>
<td></td>
<td>year (360 day)</td>
</tr>
<tr>
<td>7</td>
<td>kilopond</td>
<td>meter</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>kilometer</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Its calling sequence is listed here:

```
SUBROUTINE AUNIT(UIOAUN,CALALL,IOFLAG,MODE,
MTERSE,DBAUNN,DBAUNC,NCPW,PMPREP,
PMES,RNFILE,AUNFILE,ISCLDR,
ISCEDT,ISCDMP)
```

The calling sequences for the others are similar. Table 7-2 lists the comparable distinctive parameters.

UIOAUN denotes the i/o angle unit and can be either an output variable (when reading or editing) or an input variable (when writing). It has the following integer values depending on the specific i/o angle unit:

1: cycle
2: radian
3: degree (angular)
4: minute (angular)
5: second (angular)

CALALL is a logical variable which indicates the status of the calling program "all" option. Recall from the Cube Module that subroutine MXUNIT was the calling program for this series. IFLAG indicates whether the operation is reading, editing, or writing (IOFLAG = 1, 2, or 3 respectively) and dictates whether ISCLDR, ISCEDT or ISCDMP will be invoked. IOMODE indicates the source when reading and the destination when writing. MTERSE, NCPW, PMPREP, PMES, and RNFILE fulfill the same roles as described in previous chapters.

DBAUNN is where the database name of the angle unit, "UIOAUN", is stored. DBAUNC is a character
Table 7-2. I/O Unit Specifier Subroutine Calling Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>AUNIT</th>
<th>FUNIT</th>
<th>LUNIT</th>
<th>TPNUNIT</th>
<th>TUNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O Unit Indicator</td>
<td>UIOAUN</td>
<td>UIOFUN</td>
<td>UIOLUN</td>
<td>UIOTPU</td>
<td>UIOTUN</td>
</tr>
<tr>
<td>Database name</td>
<td>DBAUNN</td>
<td>DBFUNN</td>
<td>DBLUNN</td>
<td>DBTPUN</td>
<td>DBTUNN</td>
</tr>
<tr>
<td>Database comment</td>
<td>DBAUNC</td>
<td>DBFUNC</td>
<td>DBLUNC</td>
<td>DBTPUC</td>
<td>DBTUNC</td>
</tr>
<tr>
<td>File format</td>
<td>AUNFRM</td>
<td>FUNFRM</td>
<td>LUNFRM</td>
<td>TPUFRM</td>
<td>TUNFRM</td>
</tr>
<tr>
<td>Default variable</td>
<td>DEFAUN</td>
<td>DEFFUN</td>
<td>DEFLUN</td>
<td>DEFTPU</td>
<td>DEFTUN</td>
</tr>
</tbody>
</table>

Table 7-3. Basic Unit Series Calling Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>AUNIT</th>
<th>FUNIT</th>
<th>LUNIT</th>
<th>TPNUNIT</th>
<th>TUNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conversion factor</td>
<td>CONVA</td>
<td>CONVF</td>
<td>CONVL</td>
<td>CNVTPM</td>
<td>CNVT</td>
</tr>
<tr>
<td>Two letter abbrev.</td>
<td>NAMF02</td>
<td>NAML02</td>
<td>NAMT02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Three letter abbr.</td>
<td>NAMA03</td>
<td>NAMF03</td>
<td>NAMT03</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Five letter abbrev.</td>
<td>NAMA06</td>
<td>NAML06</td>
<td>NAMT06</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Six letter abbrev.</td>
<td>NAMA08</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eight letter abbr.</td>
<td>NAMA12</td>
<td>NAMF12</td>
<td>NAML12</td>
<td>NAMT12</td>
<td></td>
</tr>
<tr>
<td>Twelve letter abb.</td>
<td>PSTAUN</td>
<td>PSTFUN</td>
<td>PSLUN</td>
<td>PSTPUN</td>
<td>PSTUN</td>
</tr>
<tr>
<td>Program standard unit indicator</td>
<td>PSTAUN</td>
<td>PSTFUN</td>
<td>PSLUN</td>
<td>PSTPUN</td>
<td>PSTUN</td>
</tr>
<tr>
<td>I/O unit indicator</td>
<td>UIOAUN</td>
<td>UIOFUN</td>
<td>UIOLUN</td>
<td>UIOTPU</td>
<td>UIOTUN</td>
</tr>
</tbody>
</table>

116
string which identifies the angle unit variable. It is used as the database comment and in the preparation of the prompting and description messages. AUNFRM is the format to be used if the angle unit indicator is to be read from or written to a sequential file. DEFAUN is the default value of the angle unit indicator if that is chosen as the source.

In operation, AUNIT branches depending on the value of IOFLAG and calls ISCLDR, ISCEDT, and ISCDMP. When the user wishes to read the angle unit, AUNIT provides menu "ANG.UNIT" with its five choices to ISCLDR. This is an example of when a menu is used to input an integer value. The reader should understand that is is not the name of the unit which is read or written by these subroutines, but rather an integer value which denotes the i/o unit to be used.

7.3 The Basic Unit Series

7.3.1 Series Description. Since the module author knows and provides indicators for the units in which he has written his program, once the user specifies the units he wishes to use during input and output, it is possible to determine the conversion factors for relating the i/o units to the program.
standard units. These conversion factors can then be passed on to the loaders when reading variables (real scalars or arrays) to convert their values in i/o units to p.s.u. for the module computing subprograms. Similarly, these factors can be passed on to the dumpers when writing variables to convert their values in p.s.u. to i/o units. The determination of the conversion factors for the five basic types of units is accomplished by five logical functions listed here:

- UNITAF (angle units)
- UNITFF (force units)
- UNITLF (length units)
- UNITMP (temperature units)
- UNITTF (time units)

These functions accomplish one other task: they prepare various alphabetic character versions of the input/output unit names, up to twelve characters long, which are used in database comments and prompting and description messages. This is explained further below.

7.3.2 Calling Parameters. Once again, because all five subroutines are essentially structured the same, only one, UNITFF, will be described in detail. The calling sequence for UNITFF, as it would appear in a module subprogram for reading, editing, or writing input/output variables, is as follows:

```
LOGVAL=UNITFF(CONVF,NAMF02,NAMF03,NAMF12,
```
The sequence is similar for all five functions with two exceptions. The number of versions of the unit names for some is different and UNITMP includes two conversion factors instead of one. Table 7-3 lists the comparable calling parameters for the five functions.

The first four calling parameters are defined by UNITFF and the four are input variables to the function. CONVF is the multiplicative conversion factor which partially converts the force values from i/o units to p.s.u. when reading or editing and does the reverse when writing. The conversion also requires an additive conversion factor which, in all cases except with temperature units, is equal to zero and is provided to the loader, editor, or dumper by the module. The conversion that takes place in the reading routines is of the form:

\[ \text{VARIABLE(p.s.u.)} = \text{VARIABLE(i/o unit)} \times \text{UNITFM} + \text{UNITFA} \]

where UNITFM is the multiplicative conversion factor determined by one of these functions and UNITFA is the additive conversion factor.

NAMF02, NAMF03, and NAMF12 are respectively two-, three-, and twelve-character abbreviations of the
force unit used during input and output. NAMF12 is used as UNITNM in prompting and description messages and database comments for force variables (recall that UNITNM must be a twelve character version of the relevant unit). Tables B-1 through B-5 in Appendix B list the various abbreviations of the five basic units.

PSTFUN and UIOFUN denote the program standard force unit and the input/output force unit respectively. They can each be an integer between 1 and 7 inclusive, corresponding to the seven permissible force units listed in Table 7-1.

7.3.3 Execution. When invoked, UNITFF calls routine CHKRNG to verify that PSTFUN and UIOFUN are within the permissible range 1-7. UNITFF then uses the pair (PSTFUN, UIOFUN) as an index to a data table included within the function to locate the conversion factor appropriate for converting an input value in the i/o force unit denoted by UIOFUN to the program standard force unit denoted by PSTFUN.

UIOFUN is also used as an index to another data table in the function which contains the various abbreviations of the seven force units. UNITFF employs DEX routine LMOVEC to copy the characters from the data table into the strings NAMF02, NAMF03, and NAMF12.
If a failure occurs in defining either the force unit conversion factor or the unit name, the user is informed that the appropriate variable has not been defined, is essential for i/o continuation, and must be corrected before continuing. ALLFLG is changed to .FALSE. if it was .TRUE. and UNITFF is set to .FALSE.. If successful in accomplishing both tasks it is set to .TRUE..

7.4 Derived I/O Unit Series

7.4.1 Series Description. The third series in the units category contains twelve logical functions for defining conversion factors and unit names for units of measurement formed by combining basic units. These are listed in Table 7-4.

In order to operate these functions, the module author must first have either specified or allowed the user to specify the basic i/o units which are building blocks for these derived unit functions. The module program must then have used the appropriate basic unit series function or functions to obtain the various multiplicative conversion factors and abbreviations. These are then used as calling parameters for the derived units in this series.
<table>
<thead>
<tr>
<th>Function</th>
<th>Type of Measurement</th>
<th>Units of Measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>UAACC</td>
<td>angular acceleration plane</td>
<td>plane angle/(time)$^2$</td>
</tr>
<tr>
<td>UACCEL</td>
<td>linear acceleration</td>
<td>length/(time)$^2$</td>
</tr>
<tr>
<td>UAREA</td>
<td>area</td>
<td>(length)$^2$</td>
</tr>
<tr>
<td>UFREQ</td>
<td>frequency</td>
<td>plane angle/time</td>
</tr>
<tr>
<td>UKVISC</td>
<td>kinematic viscosity</td>
<td>(length)$^2$/time</td>
</tr>
<tr>
<td>UMASS</td>
<td>mass</td>
<td>force-(time)$^2$/length</td>
</tr>
<tr>
<td>UMPOWR</td>
<td>mechanical power</td>
<td>force-length/time</td>
</tr>
<tr>
<td>UPRESS</td>
<td>pressure</td>
<td>force/(length)$^2$</td>
</tr>
<tr>
<td>UPSPEC</td>
<td>power spectrum</td>
<td>(length)$^2$-time</td>
</tr>
<tr>
<td>URHO</td>
<td>mass density</td>
<td>force-(time)$^2$/(length)$^4$</td>
</tr>
<tr>
<td>USPEED</td>
<td>speed</td>
<td>length/time</td>
</tr>
<tr>
<td>UVOL</td>
<td>volume</td>
<td>(length)$^3$</td>
</tr>
</tbody>
</table>
There is considerably more diversity in the calling sequences of the twelve functions. Appendix B, Table B-6, lists them for reference. In these functions only multiplicative conversion factors are used to determine the combined conversion factors because none involve temperature units. It should, therefore, be easy to identify CONVA, CONVF, CONVL, and CONVT as the angular, force, length, and time multiplicative conversion factors. The abbreviations of the basic units used in the calling sequences were shown in Tables B-1 through B-5.

One of the functions, UPRESS, will be described in more detail as an example of how they operate.

7.4.2 UPRESS Calling Parameters. UPRESS allows its users to define the unit conversion factor and name for a variable that has the units of pressure (force/area). The calling sequence for UPRESS is as follows:

```
LOGICAL FUNCTION UPRESS(UFPRESS, UNPRESS, ALLFLG, CONVF, CONVL, NAMF03, NAMF02, NCPW)
```

The pressure conversion factor UFPRESS converts the input/output pressure unit to the program standard pressure unit by multiplication when reading or editing and converts the p.s. pressure unit to i/o pressure unit when writing by division. The unit name UNPRESS
is used to identify the units of the variable in question for messages and the database comment. UNPRES is a twelve-character string (including blanks). ALLFLG indicates the calling program "all" option. NAMFØ3 is a three-character force unit abbreviation and NAMLØ2 is a two-character length unit abbreviation.

7.4.3 UPRESS Operation. UPRESS first defines the pressure unit conversion factor by the statement

\[ \text{UPPRESS} = \text{CONVF/CONVL}^2 \]

In order to form the pressure unit name, UPRESS defines a twelve-character dummy name variable UXPRES printed here:

" \_ / \_ * \_ 2 \_ / \_ "

UPRESS inserts, via DEX routine LMOVEC, NAMFØ3 into the first three blank spaces and NAMLØ2 into the fifth and sixth spaces. The three "words" (four characters per word) of UXPRES are then set equal to the three words of UNPRES. As an example, if the force unit was poundforce and the length unit was inches, the final version of UNPRES would be

"LBF/IN"^2"

If a failure occurs in preparing the unit name, a message advises the user and informs him that the problem must be corrected, because it is essential for input/
output continuation. If ALLFLG was .TRUE. it is set equal to .FALSE. and the user is informed that the "all" option is aborted. UPRESS is then set equal to .FALSE. If it is successful, UPRESS is set equal to .TRUE..

Certain combinations of basic units have special universally recognized names used to identity the measurement unit. Where possible, the logical functions provide these names rather than creating a name by its constituents, such as UNPRES was formed in the above example. Table 7-5 lists these special names.

Although there are only twelve types of measurements listed the derived unit series have more versatility than first meets the eye. They can be used for units that have different names but the same basic units. For example, UPRESS can be used for stress units as well as pressure. In addition, they can be used for units that have different basic units but the same format. An example is provided by UAACC and UACCEL, which could be used for any unit type requiring one basic unit in the numerator and a basic unit squared in the denominator. The module author must be careful to supply the correct special parameters in the function calling sequence in the module calling subprogram.
Table 7-5. Special Unit Names

<table>
<thead>
<tr>
<th>Function</th>
<th>Special Name</th>
<th>Meaning</th>
<th>Occurrence</th>
</tr>
</thead>
<tbody>
<tr>
<td>UFREQ</td>
<td>hertz</td>
<td>cycle/second</td>
<td>UIOAUN=1 and UIOTUN=1</td>
</tr>
<tr>
<td>UKVISC</td>
<td>stoke</td>
<td>centimeter²/second</td>
<td>UIOLUN=6 and UIOTUN=1</td>
</tr>
<tr>
<td>UMASS</td>
<td>slug</td>
<td>lbf-second²/foot</td>
<td>UIOFUN=2 and UIOLUN=2 and</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>UIOTUN=1</td>
</tr>
<tr>
<td>UMASS</td>
<td>kilogram</td>
<td>newton-sec²/meter</td>
<td>UIOFUN=6 and UIOLUN=7 and</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>UIOTUN=1</td>
</tr>
<tr>
<td>UMPWR</td>
<td>watt</td>
<td>newton-meter/sec</td>
<td>UIOFUN=6 and UIOLUN=7 and</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>UIOTUN=1</td>
</tr>
<tr>
<td>URHO</td>
<td>slug/ft³</td>
<td>slug/foot³</td>
<td>UIOFUN=2 and UIOLUN=2 and</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>UIOTUN=1</td>
</tr>
<tr>
<td>URHO</td>
<td>kg/m³</td>
<td>kilogram/meter³</td>
<td>UIOFUN=6 and UIOLUN=7 and</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>UIOTUN=1</td>
</tr>
<tr>
<td>USPEED</td>
<td>knot</td>
<td>naut. mi./hour</td>
<td>UIOLUN=4 and UIOTUN=3</td>
</tr>
</tbody>
</table>
CHAPTER 8

DEVELOPMENT OF A CRUISER-DESTROYER DATABANK AT M.I.T.

8.1 Considerations in Database Design

8.1.1 Function. When designing a database, the developer must not only consider for what immediate function it is intended, but must also try and anticipate other future demands and organize it accordingly. One solution to this problem, in a sense an avoidance of it, is to create very specialized databases containing information about only one aspect of the overall project involved. The project has a databank comprised of many databases. Physical limitations on the database size, such as the limit of 200 variables in a DEX-created database, suggest this practice. These smaller databases may be more efficient from the point of view of computer costs when it comes to manipulating them. However, the situation can arise where a computer program requires as input data from several different databases, entailing the time consuming effort of opening and closing them all. Only experience in using the databases can reveal the deficiencies in their design.

The function of the cruiser-destroyer databases developed and/or envisioned in the Department of Ocean

127
Engineering at MIT is to support the naval architect during the concept and preliminary design phases of a ship design. During these phases a variety of products are developed, including the overall vessel dimensions and hull definition, hydrostatic and Bonjean curves, weight and volume estimates, longitudinal weight distribution, propulsion and electrical powering requirements, transverse stability and floodable length checks and general arrangements. The tasks to produce several of these, notably the determination of ship dimensions, weight and volume estimates, powering requirements and transverse stability, can be accomplished with the aid of a computer synthesis model. The REED Model [6] used at MIT is an excellent example of this design tool, and it was the anticipated support of that model that strongly influenced the databases designed in this investigation. The naval architect who chooses to use a synthesis model must carefully determine his input if he desires to use the model efficiently. Being able to draw upon a supply of existing ship information is invaluable to this effort, and this was one of the reasons for developing the cruiser-destroyer databases.

An effective database is one that can be shared by many different engineers involved in the ship design
project, each of whom has a different task to perform. Data should be stored in a form that allows each one to extract the information required and use it directly without having to pass it through some form of interpretation process. An example is a table of offsets database. Ideally, it contains sufficient offsets properly organized such that each one of the programs for hydrostatics, Bonjean curves, cross curves of stability, floodable length, structures and seakeeping can directly access it and obtain the input required without having to go through a "black box" interface program.

The development of a comprehensive computer-aided ship design system that ensures such program/database design requires a "top down" approach to the problem, as described in reference [7]. One starts with the overall objective and works down through functional specifications to complete system design. If successfully accomplished, as a result of strict discipline during the process, no unnecessary capabilities need be developed along the way. One proceeds from each level to the next lower by answering the question of how to provide for the needs of the higher one. This contrasts directly with the traditional method of many individuals writing programs for their specific task, and only after-
wards determining if these programs can be integrated for some higher objective.

8.1.2 Types of Databases. Accepting the concept of a bank of databases to describe a ship, either existing or being designed, we can list the types which will be useful:

1. General description
2. Weights and centers of gravity
3. Longitudinal weight distribution
4. Volumes, areas, and centroids
5. Offsets
6. Equipment specifications and locations
7. Power-speed data
8. Seakeeping data
9. Internal arrangements
10. Topsides arrangements

This list is similar to that of the computer-aided ship design system implemented in the Ship Department of the British Ministry of Defense [8].

Storing in a computer databank several of these databases for many classes of ship is extremely helpful as a research resource during the concept design of a new vessel. Taking this one step further, as described in reference [8], is to establish "base" ship databanks made up of all of the database types. If a new vessel is similar to one of these, a copy of the databank provides an excellent starting point to begin defining the new design and can save much redundant work. This is
predicated on the assumption that all the databases of a particular type for all ships are identical in structure and differ only in content. Such a practice is essential to the efficient use of the databanks.

8.2 Organization of the MIT Cruiser-Destroyer Databases

8.2.1 General Databases. During this investigation work was conducted to establish the first two types of databases listed in Section 8.1.2 for eleven classes of U.S. cruisers, destroyers, and frigates. These classes are as follows:

- FF-1040
- FF-1052
- FFG-1
- FFG-7
- DD-931
- DD-963
- DDG-2
- DDG-40
- CG-16
- CG-26
- CG-47

This section will describe the organization of the general databases and the next section will describe the weights and centers of gravity databases.

The general databases are so named because they provide a general and not-too-detailed description of the ship class which would be useful to a researcher seeking to determine first estimates for a new design. The information was gleaned from various sources in the open literature, and the respective weight and moment reports and booklets of general plans [9,10,11,12].
The database contains eight categories of variables.

These are:

1. Hull characteristics
2. Propulsion and powering
3. Transverse and directional stability
4. Weapons payload
5. Electronics, fire control, and sensors
6. Aviation capability
7. Complement
8. Gross mass properties

Appendix C is an example of a general database. The individual entries are what would appear if one issued the "dump" command from the DEX level with a particular database open. The order of the listing would not be as they appear here because of a "hashing" function built into the DEX which distributes entries to a database in the memory randomly in order to store them more efficiently.

There are actually 78 variables, listed in Table 8-1 which constitute the eight categories. Each one has a number assigned on the left hand margin. These serve as a convenient indicator for the creation of Fortran names for the various variables associated with each element used in a DEX program. For example, in the module MACHWT described later in this chapter, the program names for the default value and comment statement for propulsion plant type (Item #20) are DEF20 and
TABLE 8.1
GENERAL SHIP CHARACTERISTICS DATABASE
FOR U.S. NAVY CRUISER-DESTROYERS

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>COMMENT</th>
<th>UNITS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>HULL CHARACTERISTICS</strong></td>
<td></td>
</tr>
<tr>
<td>1. LOA</td>
<td>R</td>
<td>Length overall</td>
<td>feet</td>
</tr>
<tr>
<td>2. LBP</td>
<td>R</td>
<td>Length between perpendiculars</td>
<td>feet</td>
</tr>
<tr>
<td>3. BEAMDL</td>
<td>R</td>
<td>Molded beam at design waterline</td>
<td>feet</td>
</tr>
<tr>
<td>4. BEAMAX</td>
<td>R</td>
<td>Maximum beam</td>
<td>feet</td>
</tr>
<tr>
<td>5. T</td>
<td>R</td>
<td>Molded draft to keel</td>
<td>feet</td>
</tr>
<tr>
<td>6. CP</td>
<td>R</td>
<td>Prismatic coefficient</td>
<td></td>
</tr>
<tr>
<td>7. CX</td>
<td>R</td>
<td>Midship coefficient</td>
<td></td>
</tr>
<tr>
<td>8. CB</td>
<td>R</td>
<td>Block coefficient</td>
<td></td>
</tr>
<tr>
<td>9. CWP</td>
<td>R</td>
<td>Waterplane coefficient</td>
<td></td>
</tr>
<tr>
<td>10. LCB</td>
<td>R</td>
<td>Longitudinal center of buoyancy as a fraction of LBP aft FP</td>
<td></td>
</tr>
<tr>
<td>11. LCF</td>
<td>R</td>
<td>Longitudinal center of flotation as fraction of LBP aft FP</td>
<td></td>
</tr>
<tr>
<td>12. DEPTHOl</td>
<td>R</td>
<td>Depth amidships at centerline</td>
<td>feet</td>
</tr>
<tr>
<td>13. DRAFTSON</td>
<td>R</td>
<td>Draft of sonar dome</td>
<td>feet</td>
</tr>
<tr>
<td>14. DISPMLD</td>
<td>R</td>
<td>Molded displacement</td>
<td>tons</td>
</tr>
<tr>
<td>15. DISPTOT</td>
<td>R</td>
<td>Total displacement including appendages</td>
<td>tons</td>
</tr>
<tr>
<td>16. WETSURF</td>
<td>R</td>
<td>Wetted surface</td>
<td>sq. ft</td>
</tr>
<tr>
<td>17. FOCSL</td>
<td>I</td>
<td>Raised forecastle (0 = no, 1 = yes)</td>
<td></td>
</tr>
<tr>
<td>18.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20. PPTYP</td>
<td>I</td>
<td>Type of propulsion plant</td>
<td>Reed</td>
</tr>
<tr>
<td>21. SHP</td>
<td>R</td>
<td>Total installed shaft horsepower</td>
<td>hp</td>
</tr>
<tr>
<td>22. NSHAFT</td>
<td>I</td>
<td>Number of propeller shafts</td>
<td></td>
</tr>
<tr>
<td>23. NE</td>
<td>I</td>
<td>Number of engines</td>
<td></td>
</tr>
<tr>
<td>24. NB</td>
<td>I</td>
<td>Number of boilers</td>
<td></td>
</tr>
</tbody>
</table>
### TABLE 8.1 (cont’d)

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>COMMENT</th>
<th>UNITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>25. VSUS</td>
<td>R</td>
<td>Maximum continuous sustained speed</td>
<td>knots</td>
</tr>
<tr>
<td>26. VEND</td>
<td>R</td>
<td>Endurance speed</td>
<td>knots</td>
</tr>
<tr>
<td>27. ENDUR</td>
<td>R</td>
<td>Endurance range</td>
<td>n. mi.</td>
</tr>
<tr>
<td>28. PRPTYP</td>
<td>I</td>
<td>Type of propeller (1 = FP; 2 = CRP)</td>
<td></td>
</tr>
<tr>
<td>29. DPROP</td>
<td>R</td>
<td>Propeller diameter</td>
<td>feet</td>
</tr>
<tr>
<td>30. RPM</td>
<td>R</td>
<td>Propeller rpm at full power</td>
<td>rpm</td>
</tr>
<tr>
<td>31. SSEPTYP</td>
<td>I</td>
<td>Type of primary ship service electrical plant</td>
<td></td>
</tr>
<tr>
<td>32. NSSG</td>
<td>I</td>
<td>Number of primary ship service generators</td>
<td></td>
</tr>
<tr>
<td>33. KWSSER</td>
<td>R</td>
<td>Installed primary ship service generator capacity</td>
<td>kw</td>
</tr>
<tr>
<td>34. EMETYP</td>
<td>A(2)</td>
<td>Type of secondary or emergency electrical plant</td>
<td></td>
</tr>
<tr>
<td>35. NEMG</td>
<td>A(2)</td>
<td>Number of secondary or emergency generators of each type</td>
<td></td>
</tr>
<tr>
<td>36. KWEER</td>
<td>R</td>
<td>Installed secondary or emergency generator capacity</td>
<td>kw</td>
</tr>
<tr>
<td>37. KPENG</td>
<td>A(2)</td>
<td>Capacity per secondary or emergency generator</td>
<td>kw</td>
</tr>
</tbody>
</table>

#### TRANSVERSE AND DIRECTIONAL STABILITY

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>COMMENT</th>
<th>UNITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>41. GM</td>
<td>R</td>
<td>Metacentric height uncorrected</td>
<td>feet</td>
</tr>
<tr>
<td>42. FSUFCOR</td>
<td>R</td>
<td>Free surface correction</td>
<td>feet</td>
</tr>
<tr>
<td>43. CI</td>
<td>R</td>
<td>Waterplane moment of inertia coefficient</td>
<td></td>
</tr>
<tr>
<td>44. FINSTABL</td>
<td>I</td>
<td>Pin stabilizers installed (9 = no; 1 = yes)</td>
<td></td>
</tr>
<tr>
<td>45.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>46. NRUDDER</td>
<td>I</td>
<td>Number of rudders</td>
<td></td>
</tr>
<tr>
<td>47.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>48.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>49.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### WEAPONS PAYLOAD

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>COMMENT</th>
<th>UNITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>50. TYPGUNS</td>
<td>A(3)</td>
<td>Type of guns</td>
<td>Reed</td>
</tr>
<tr>
<td>51. NGUNS</td>
<td>A(3)</td>
<td>Number of guns of each type</td>
<td>Reed</td>
</tr>
<tr>
<td>52. TYPMSL</td>
<td>I</td>
<td>Type of missile launchers</td>
<td></td>
</tr>
<tr>
<td>53. NMSL</td>
<td>I</td>
<td>Number of missile launchers</td>
<td></td>
</tr>
</tbody>
</table>
**Table 8.1 (cont'd)**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>COMMENT</th>
<th>UNITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>54. TYPWMS</td>
<td>I</td>
<td>Type of close-in weapon system</td>
<td>Reed</td>
</tr>
<tr>
<td>55. NCIWS</td>
<td>I</td>
<td>Number of close-in weapon systems</td>
<td>Reed</td>
</tr>
<tr>
<td>56. TYPBDMS</td>
<td>I</td>
<td>Type of basic point defense missile system</td>
<td>Reed</td>
</tr>
<tr>
<td>57. NBPDM</td>
<td>I</td>
<td>Number of basic point defense missile launchers</td>
<td>Reed</td>
</tr>
<tr>
<td>58. TYPDORPL</td>
<td>I</td>
<td>Type of torpedo launchers</td>
<td>Reed</td>
</tr>
<tr>
<td>59. NTDORPL</td>
<td>I</td>
<td>Number of torpedo launchers</td>
<td>Reed</td>
</tr>
<tr>
<td>60. TYPASWL</td>
<td>I</td>
<td>Type of ASW launchers</td>
<td>Reed</td>
</tr>
<tr>
<td>61. NASWL</td>
<td>I</td>
<td>Number of ASW launchers</td>
<td></td>
</tr>
<tr>
<td>62.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>63.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>65.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Electronics, Fire Control and Sensors**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>COMMENT</th>
<th>UNITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>66. TYPSONAR</td>
<td>A(2)</td>
<td>Type of sonar systems</td>
<td>Reed</td>
</tr>
<tr>
<td>67. TYPDOME</td>
<td>I</td>
<td>Type of sonar dome</td>
<td>Reed</td>
</tr>
<tr>
<td>68. TYPSURAD</td>
<td>I</td>
<td>Type of surface search radar</td>
<td>Reed</td>
</tr>
<tr>
<td>69. TYP3AIR</td>
<td>I</td>
<td>Type of 3-D air search radar</td>
<td>Reed</td>
</tr>
<tr>
<td>70. TYP2AIR</td>
<td>I</td>
<td>Type of 2-D air search radar</td>
<td>Reed</td>
</tr>
<tr>
<td>71. TYPGARAD</td>
<td>I</td>
<td>Type of gun fire control radars or directors</td>
<td>Reed</td>
</tr>
<tr>
<td>72. MGARAD</td>
<td>I</td>
<td>Number of gun fire control radars or directors</td>
<td>Reed</td>
</tr>
<tr>
<td>73. TYPMARD</td>
<td>I</td>
<td>Type of missile fire control radars or directors</td>
<td>Reed</td>
</tr>
<tr>
<td>74. NMMLRAD</td>
<td>I</td>
<td>Number of missile fire control radars or directors</td>
<td>Reed</td>
</tr>
<tr>
<td>75. TYPFCSG</td>
<td>I</td>
<td>Type of gun fire control system</td>
<td>Reed</td>
</tr>
<tr>
<td>76. TYPFCM</td>
<td>I</td>
<td>Type of missile fire control system</td>
<td>Reed</td>
</tr>
<tr>
<td>77. TYPASWFL</td>
<td>I</td>
<td>Type of ASW fire control system</td>
<td>Reed</td>
</tr>
<tr>
<td>78. TYPHDS</td>
<td>I</td>
<td>Type of tactical data system</td>
<td></td>
</tr>
</tbody>
</table>
### TABLE 8.1 (cont'd)

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>COMMENT</th>
<th>UNITS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>AVIATION CAPABILITY</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Type of helicopters carried</td>
<td></td>
</tr>
<tr>
<td>81. TYNELO</td>
<td>I</td>
<td>Number of helicopters carried</td>
<td></td>
</tr>
<tr>
<td>82. NHELO</td>
<td>I</td>
<td>Helicopter in-flight refueling capability (1=yes, 0=no)</td>
<td></td>
</tr>
<tr>
<td>83. UIFR</td>
<td>I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>84.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>85.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>86. NOFF</td>
<td>I</td>
<td>Number of ship's officers</td>
<td></td>
</tr>
<tr>
<td>87. NCPO</td>
<td>I</td>
<td>Number of chief petty officers in ship's crew</td>
<td></td>
</tr>
<tr>
<td>88. NCREW</td>
<td>I</td>
<td>Number of enlisted in ship's crew</td>
<td></td>
</tr>
<tr>
<td>89. NFLAGOFF</td>
<td>I</td>
<td>Number of officers on flag staff</td>
<td></td>
</tr>
<tr>
<td>90. NENLISTF</td>
<td>I</td>
<td>Number of enlisted on flag staff</td>
<td></td>
</tr>
<tr>
<td>91. NTOOPS</td>
<td>I</td>
<td>Number of troops</td>
<td></td>
</tr>
<tr>
<td>92.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>93. TYPMATL</td>
<td>A(2)</td>
<td>Type of material for hull and superstructure respectively</td>
<td></td>
</tr>
<tr>
<td>94. WEIGHT17</td>
<td>A(7)</td>
<td>Weights of weight groups 1-7 respectively</td>
<td></td>
</tr>
<tr>
<td>95. VGCI7</td>
<td>A(7)</td>
<td>Vertical centers of gravity of weight groups 1-7 respectively</td>
<td></td>
</tr>
<tr>
<td>96. WTLOADS</td>
<td>R</td>
<td>Weight of Group 8 loads</td>
<td></td>
</tr>
<tr>
<td>97. VCGLOADS</td>
<td>R</td>
<td>Vertical center of gravity for Group 8 loads</td>
<td></td>
</tr>
<tr>
<td>98.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>99.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In each category some space has been left for additional variables. Further, experience with the databases may indicate that some items are not needed and can be deleted.

An inspection of both Table 8-1 and Appendix C reveals that certain items referring to the types of plant or type of equipment have integer values where one would expect a name. The reason for this is because only three types of variables are allowed in the DEX: integer scalar, real scalar, and real array. Alphanumeric words in the "value" part of a database entry are not allowed. A code of integer values was needed to solve this dilemma, and it was decided to adopt the payload shopping list of the REED model because of its comprehensiveness and its widespread use at MIT. Appendix D contains the payload list from reference [6], with some additional items included for this application.

The restriction on arrays that they contain only real values poses a minor problem because they sometimes contain information from the code which should be stored as an integer. It should be obvious to the user from the array name that an integer value is implied. Arrays are used in some not very obvious cases in order to
accommodate the most information. An explanation of the array variables should prove helpful.

TYPMATL has two entries to distinguish between the type of material for the hull and the type of material for the superstructure. The integer values are 1 for steel and 2 for aluminum.

The type of sonar carried (TYPSONAR) is an array because some ships have two systems installed: a bow or keel-mounted sonar, plus a towed array or variable depth sonar.

NGUNS and TYPGUNS are three element arrays to accommodate the most number of distinguishable gun mounts in any of the classes, which exists on the DD-931 class. Not only does this destroyer carry two calibers, 5" and 3", but the REED payload code allows the distinction between a 5" gun mounted on the main-deck (93) and a 5" gun mounted on the 01 level (94).

The emergency or secondary electrical plant includes three array variables: EMETYP, NEMG, and KWPEMG. The CG-26 class cruiser has both a gas turbine-driven and diesel-driven emergency generator. Therefore, the first entries of the three arrays describes the one and the second entries describe the other.

Unfortunately, a great amount of the data available
from the various sources for the general database was conflicting. Where such discrepancies occurred, this investigator made choices based upon the most original source, or the value upon which the most sources agreed. Whenever possible, the original ship equipment is listed in order to correspond to the weights and centers of gravity databases, whose information comes from the original class weight and moment reports.

Any value that was either classified or unavailable was left undefined.

8.2.2 Mass Properties Databases. The general databases include a gross mass properties category which includes two arrays, WEIGHT17 and VCG17. These contain respectively the overall weights and centers of gravity of weight groups 1 through 7. The weight groups conform to the U.S. Navy BSCI organization of ship weights. Although the BSCI system has been replaced by the SWBS (Ship Work Breakdown System) in recent years, it was used for the databases because only the FFG-7 class is sufficiently recent to have its weight and moment report organized with the new system. Further, the REED model is based on BSCI.

The gross mass properties are included in the general databases because they are more frequently used for
estimations than the individual weight items, and their inclusion may save the user from inspecting two different databases.

There are about 150 items comprising the eight weight groups of the BSCI system. Therefore, the combination of weight and center of gravity for each item exceeds the limit of 200 entries in a DEX database. Although the use of arrays offers an apparent solution to this problem, the idea was discarded after careful consideration for several reasons. First, if one wished to store the weight in long tons, the vertical center of gravity in feet above baseline and the longitudinal center of gravity in feet aft of the forward perpendicular or from amidships in a three element array, not only would it be difficult to identify the information in the 64 characters of the database comment, but only one of the two unit names could be stored there. Another possibility was to store all of the weights (or centers of gravity) for one weight group in an array. There would then be eight weight arrays, eight vcg arrays and eight lcg arrays, with the proper units in the database comment. The limit of 200 elements per array would not be a problem because the largest index in any weight group is 51. This was considered
unsatisfactory because it was not felt that the one
database comment for the array was sufficient to iden-
tify the individual weight items and an extra index
would have to be provided to the user. Further, an
additional process would have to be developed for ex-
tracting the particular weight item out of the array,
and avoiding the need to know where a value was stored
in the database was one of the driving principles for
developing DEX databases to begin with.

Instead, it was decided to create a weight data-
base and a vertical center of gravity database, with
each item listed separately. Appendix E illustrates the
listing of each type. No need was felt by this invest-
igator for a longitudinal center of gravity database
for existing ships. The estimating of the transverse
stability of a new ship design can be done effectively
using data from existing ships because the vertical
locations of most items is restricted to a reasonable
degree by physical factors or proven arrangements. The
REED model demonstrates that dependable parametric
equations can be developed for estimating vertical
centers of gravity. However, there is far more flexi-
bility in both theory and practice for the longitudinal
locations of many of the same items. Therefore, it is
more difficult to correlate into acceptably accurate parametric equations the information available on lcg's in existing ships. This does not preclude the need for a database containing the longitudinal weight distribution of a ship design in order to support longitudinal strength and seakeeping analyses. Nor does it preclude the use of a longitudinal center of gravity database for a new ship in order to support longitudinal stability (i.e. trim) analyses.

8.3 Independent and Dependent Variables

8.3.1 Concept. Databases can be both the source and destination of information. A particular program may read its input from a database, calculate values for other variables in the database, and write the new values into those entries. This would be disastrous if uncontrolled. When administering a ship design project that involves multiple uses of the same databases, the ship design manager must have a system whereby he can control changes to the databases that occur as the design progresses around the design spiral. Further, the system should allow all design team members to be alerted to changes which may affect them. It is planned in future versions of DEX to implement a system that supports the concept of independent and dependent
variables.

Certain variables will be defined by the user as independent variables which, either by fact or intention, can not be changed despite changes in other variables. The remaining variables in the program or database are dependent on the former or each other for their values. Each entry in a database will be provided with an index of those variables whose value would be affected by a change in its value. When causing a change to such an entry (i.e. DBCHNG becomes .TRUE.), the user can query this index to determine which other items should be checked.

This task is extremely difficult in ship design because of the interaction of almost all of the variables. Ship design is not a linear process but a spiraling one. Figure 8-1 illustrates an attempt to group the variables of the general database into five levels of dependence. The first column represents those variables which can be considered independent. These might appear as specifications in a Top Level Requirement or they might be the result of trade-off studies during the design phase.

The second group consists of those variables which are most directly affected by the independents or which
Figure 8-1. General Database Variable Relationships
are estimated first in the design process. The third column is dependent upon values in the first and/or second columns and the fourth column on values in the third and possibly first and/or second column. This table allows the database designer to determine what indices to put on each variable to alert him to check dependent ones.

For example, the dependent variables entry for ENDUR may include the following: WTLOAD, LBP, BEAMDWL, T, CP. A check on LBP will then add to the list of affected variables DISPMLD, DISPTOT, LOA, CWP, CI, LCB, LCF, etc. Although this system requires more work by the database designer, it will make the job of the design manager easier.

8.4 Application of DEX: An Example

8.4.1 Function of MACHWT. The Machinery Weight Estimating (MACHWT) Module was written to demonstrate how DEX and the cruiser-destroyer databases could be used in the preliminary design of a new ship. MACHWT has a fairly limited computation capability since it is only a demonstration module. It enables the user to estimate weight items 200, 201 and 203 based on certain existing parametric equations and parametric equations
developed by the user during the module execution.

These three weight groups are respectively the weight of boilers, weight of propulsion units and weight of the propeller, shafting, and bearings. An analysis of 9 ships for which weight data is available reveals that the sum of these three items constitute between 58.0 and 65.5% of the total Group 2 weight.

The first two weight items are estimated by assuming that they are linearly related to installed horsepower. The program fits a straight line to data extracted from the databases chosen by the user and predicts the new ship weights based on the new specified installed SHP. The program calculates the three component weights of item 203 from the input supplied by the user from any of the valid sources, using parametric equations from the REED model. A summary of the input required for each weight is provided in Table 8-2 (the actual database names are used).

8.4.2 List of Subprograms and Menus. The Machinery Weight Estimating Module includes ten subprograms. They are listed below in the order in which they are most
likely encountered during the execution of the module:

MAINPG
MODIO
INPUT
MWUNIT
MWLIST
MWCHRT
MWCOMP
OUTFUT
MWCOEF
BLOCK DATA
LINFIT

There is actually no one correct sequence of listing the subprograms in the module, other than the requirement that MAINPG be first.

---

**TABLE 8-2**

**INPUT FOR MACHWT**

W200: W200 and SHP from at least two steam ships and SHP of new ship

W201: W201 and SHP from at least two ships and SHP of new ship

W203: LBP, PPTYP, NSHAFT, PRPTYP, VSUS and DPROP (optional) of new ship

147
Seven of the subprograms employ menus in their operation. These are illustrated in Figure 8-2. A listing of the module subprograms appears as Appendix F. They are described in the next section.

8.4.3 Description of the Subprograms. A description of a typical execution of the module will serve as a backdrop for the subprogram descriptions. The user leaves the DEX level and activates the module by using the "DEX-MAIN" menu item and module labeled

```
.begin machwt
```

Subprograms MAINPG and menu "MOD.MAIN" are encountered first. MAINPG is identical to the subprogram of the same name used in the Cube Module described in Chapter 2, as is subprogram MODIO, which would be the next one encountered. The menu selections from these two subprograms are

```
.read input
```

These place the user in subroutine INPUT. This subroutine provides the user with a menu permitting him to read, edit, or write the following:

1. All the module input variables.
2. The module input and/or output variables
3. The machinery weight item to be estimated
4. The data from existing ships to be used for curve fitting for weight items $W(200)$ and $W(201)$. 

148
Figure 8-2. Machinery Weight Estimating Menus
5. The characteristics of the new ship design needed as input for the weight calculations.

The machinery weight item must be read first to establish the proper value of a variable WFLAG needed by the subsequent subprograms. This will permit the correct prompting messages to be issued to the user for proper input sequencing.

The user can access MWUNIT to specify the length, force and time units to be used for input and output, but he will normally just use the ones initialized in the module in BLOCK DATA. These values are respectively foot, long ton and second, and were chosen to conform with the units of the database variables used. MWUNIT is a shortened version of MXUNIT from the Cube Module.

Returning from or bypassing MWUNIT, the user types 
.wt.item w200

to access MWLIST and set WFLAG to indicate weight group 200 to be estimated. MWLIST returns him to INPUT and he selects "curvepts". The following prompting message is issued.

*SPECIFY THE SEQUENTIAL NUMBER OF THIS PAIR OF DATA POINTS
*ENTER UP TO 1 INTEGER NUMBERS

He types "1" and is presented with menu "CHARACT." from subroutine MWCHRT.
Subroutine MWCHT allows the user to read, edit, or write the characteristics of the ship in question listed in the menu in Figure 8-2. For data points, the independent variable must be read first and the dependent variable next. In this case the user specifies SHP and then W200 and they are read from the open ship database and then inserted into the first positions of an independent variable array and a dependent variable respectively. The user then issues

.done done done
to get back to MAINPG. Using the "inmode" menu selection the user closes the open general database for one steam warship and opens the other one. He then types

.read input curvepts 2 shp yes w200
to input the second pair of data points into the two arrays. The "yes" responds to a question posed by MWCHRT to ascertain if the user is employing horsepower or kilowatts to measure SHP.

This process is repeated for as many ship class databases from which the user wishes to read data for curve fitting, up to a limit of 10. For the purpose of demonstration, the three weight items were stored in the general databases so that only one database for each ship would have to be opened. Normally they re-
side in the weight databases.

When the user is satisfied with the data points read, he specifies "newship" from menu "INPUT" which causes the following message to be issued:

*TO ESTIMATE W(200) OR W(201) INPUT NEW SHIP SHP.
*SELECT WHICH CHARACTERISTIC TO READ.

The user then selects SHP from menu "CHARACT." to complete the input required. He returns to MAINPG and executes the computing program MWCOMP by the following command:

.done done done compute

Once it completes its calculation, MWCOMP returns control to MAINPG, which issues its menu prompting message.

In order to first inspect the coefficients of the straight line fitted to the data, the user (after ensuring that the destination is the terminal) types

.write output coeffici

These commands invoke MODIO, OUTPUT and MWCOEF successively. The last one causes the two element coefficient array to be printed. The two values which appear are the slope and y-intercept of the straight line.

The user then selects "newship" from menu "OUTPUT" and then "w200" from "CHARACT." and the new estimated boiler weight is printed on the terminal. The user can then return to MAINPG, choose the new ship database
as the destination, and write the estimated $W(200)$ into it. Now, in order to estimate $W(201)$, the user must first exit the module via the "quit" selection from "MOD.MAIN" in order to clear the independent and dependent variable arrays. This is unnecessary if he is going to use at least the same number of data points as for $W(200)$. It is also unnecessary for $W(203)$ which does not require curve fitting.

For $W(203)$, subroutine INPUT prompts the user with the following message when "newship" is chosen:

*TO ESTIMATE $W(203)$ THE FOLLOWING INFORMATION IS REQUIRED: *LBP PPTYPE SHP NSHAFT PRPTYP VSUS DPROP(optional)

If DPROP is not specified MWCOMP estimates it.

Simple as it is, MACHWT is more sophisticated than the Cube Module. It is hoped that the listing in Appendix F can serve as a guide to readers preparing a module for use on the DEX.

8.4.4 Results from the MACHWT Module. The module was exercised to estimate $W(200)$ and $W(201)$ for a nominal new ship design having a 40,000 SHP 1200 psi steam plant installed. In order to estimate the weight of boilers, data from the DDG-2, DDG-40, and FF-1052 classes was used. For estimating the weight of the propulsion units, data from the DDG-2, DDG-40, CG-16, CG-26, FF-1052, and
FFG-1 class databases was used.

The REED Model algorithms for the respective weights are as follows:

\[ W_{200} = 0.00234 \times \text{SHP} + 48.09 \]
\[ W_{201} = 0.00143 \times \text{SHP} + 17.92 \]

The MACHWT Module fits the following equations to the data used:

\[ W_{200} = 0.002585 \times \text{SHP} + 31.94 \]
\[ W_{201} = 0.0017665 \times \text{SHP} + 6.66 \]

The respective estimated weights for the new ship appear in Table 8-3.

<table>
<thead>
<tr>
<th></th>
<th>Reed Model</th>
<th>MACHWT</th>
</tr>
</thead>
<tbody>
<tr>
<td>( W_{200} ) (tons)</td>
<td>141.7</td>
<td>135.3</td>
</tr>
<tr>
<td>( W_{201} ) (tons)</td>
<td>75.1</td>
<td>77.3</td>
</tr>
</tbody>
</table>

**8.4.5 Future Developments.** MACHWT represents a starting point for what is hoped will be a major ship synthesis program incorporating DEX databases and the REED Model. The model as written contains hundreds of parametric equations for estimating weights, volumes, areas and centers of gravity which were derived from
the data available to its author at that time. As new ships are designed by the Navy, say every 4-5 years, a problem arises with respect to incorporating them into the model. It would be a major undertaking to perform the regression analysis for all new equations. Such a task would have questionable merits since it would probably be found that many equations change only slightly, and others that change drastically have insignificant effects on the overall design. Further, the user would still be confined to using equations of a form chosen by some other designer and derived from those ship classes chosen by him, to which the current user may object.

MACHWT demonstrates a program that allows the user to specify the ship data upon which he wishes to perform a regression analysis. There is no reason why the coefficients obtained could not be written into a database which would be accessed by the REED model in order to estimate that weight item. Expanding on this idea, a program could be developed which allows the user to derive his own coefficients for parametric equations for the large, but not all inclusive, set of variables (weights, volumes, etc.) which impact significantly on the ship design. When a new naval ship class design is
finalized, databases could be produced and stored in the design library at MIT. Only after, perhaps, 3-4 designs and 10-15 years would a major revision of the REED model become worthwhile. The cycle could then begin anew. Not only would this approach avoid frequent rewriting of the REED model, but more importantly, it would allow the individual designer much more control over the tool at his disposal. This would greatly support the function of the department to train naval architects.
CHAPTER 9
CONCLUSIONS AND RECOMMENDATIONS

With the completion of the work of this investigation the first truly capable version of DEX at MIT has been implemented. Current plans call for the adaption to DEX of many of the computer programs in the department and the indoctrination of students to the system. These programs cover a wide range of the calculations which occur during the preliminary design phase.

Two areas of the extended DEX library require development. First is the creation of routines for editing real arrays. Several editing capabilities, similar to those of the operating system, are being considered for implementation, possibly operated by the user by means of an editing menu.

The second area is the task of introducing graphics to the DEX at MIT. An idea to develop routines capable of reading or writing a pair of one-dimensional arrays is under consideration as the means for handling plots. One problem that also must be solved is how to allow the plotting of two curves on the same graph on the screen without any intermediate dialogue between program and user. Although some terminals permit both plotting and
dialogue to occur simultaneously on the screen, many do not, and for DEX to be portable it must be suitable for both types of terminals.

For the purpose of performing ship designs at MIT, this writer perceives the most immediate and imperative need to be the development and implementation of programs which will allow the creation of a table of offsets database. Once the hull form can be defined, the existing programs for hydrostatics, cross-curves of stability, floodable length and Bonjeans, adapted to DEX, can be operated using a common offsets database. Actually, with a hull definition database, the door is open for a significant expansion of the use of the computer in the preliminary design phase including seakeeping, general arrangements, longitudinal strength, etc. Therefore, this task is strongly recommended as a fruitful area for further research.

The adoption of the DEX System entails a change in philosophy on the part of the individual author and user. Heretofore, the programmer required the user to learn how to provide the input, to restrict himself to the design path chosen by the author, and to use the units preferred by the author. With DEX, the user should expect some standardization in the means of input,
flexibility in the path to pursue, and choice in the unit system with which to work. It means more work for the module author, but his job is only performed once, while the advantages he can offer by using DEX will be available to countless users.
REFERENCES


REFERENCES (Continued)


APPENDIX A

CUBE MODULE LISTING
C****-----**** C CODE MODULE SUBPROGRAM -----****
SUBROUTINE MAINPG 00000020
C-------------------------------****
C SUBPROGRAM DESCRIPTION:****
C MAINPG ALLOWS THE USER TO SELECT THE DESIRED PATH IN THE****
C TRANSACTION PROGRAM. FOR EXAMPLE, IF THE USER WANTS TO****
C EXECUTION OF THIS MODULE, THE CHOICES ARE****
C 1) SET STYLE OF MODULE DIALOGUE****
C 2) SELECT SOURCE OF MODULE INPUT****
C 3) SELECT DESTINATION OF MODULE OUTPUT****
C 4) INTERROGATE THE MODULE TAGS AND FIND OUT THEIR VALUE****
C 5) ACCESS THE MODULE READING ROUTINES****
C 6) ACCESS THE MODULE EDITING ROUTINES****
C 7) ACCESS THE MODULE COMPUTING ROUTINES****
C 8) ACCESS THE MODULE WRITING ROUTINES****
C 9) RETURN CONTROL TO DEX****
C-------------------------------****
C LUBS INITIALIZED IN BLOCK DATA****
C RUMFILES: FILE REFERENCE NUMBER FOR MODULE FORTRAN READS FROM A****
C SEQUENTIAL FILE****
C RUMFILES: FILE REFERENCE NUMBER FOR MODULE FORTRAN WRITES TO A****
C SEQUENTIAL FILE****
C-------------------------------****
C INOUTI INITIALIZED IN BLOCK DATA****
C IMODE: DENOTES SOURCE OF MODULE INPUT****
C = 1 A DEX CREATED DATABASE****
C = 2 USER EMPLOYING THE TERMINAL TO INPUT ALPHANUMERIC DATA****
C VIA DEX ROUTINES****
C = 3 THE USER EMPLOYING THE SCREEN TO INPUT X-Y COORDINATES VIA****
C DEX ROUTINES****
C = 4 A SEQUENTIAL FILE TO INPUT ALPHANUMERIC DATA WITH FORTRAN****
C READS****
C = 5 THE MODULE DEFAULT DATA****
C OOMODE: DENOTES DESTINATION OF MODULE OUTPUT****
C = 1 A DEX CREATED DATABASE****
C = 2 THE TERMINAL USING DEX ROUTINES****
C = 3 THE SCREEN USING PLOT PRODNG ROUTINES****
C = 4 A SEQUENTIAL FILE USING FORTRAN WRITES****
C = 5 THE MODULE DEFAULT DATA****
C DIERSE: TRUE, IF THE MODULE DIALOGUE IS TRUE****
C DIERSE: FALSE, IF THE MODULE DIALOGUE IS VERBOS****
C-------------------------------****
C SUBPROGRAMS AND FUNCTIONS CALLED:****
C-------------------------------****
C DEX****
C MAINPG****
C DEX****
C END****
C DEX LIBRARY****
C DIALOG****
C SOURCE****
C DISTIN****
C MEMODE
C MODUL
C MOD1
C COMPI
C-PROGRAMMERS AND REPORT----------------------
C PROGRAM VERSION : 1
C PROGRAM DATE : JUNE 1981
C PROGRAMMERS : C. CYRCHSSOTISIMDZI/ AND R. CELOTO
C REPORT : IDEAS ON HOW TO WRITE DEX MODULES
C VOLUME AND WEIGHT OF S.W. PARALLEIPIPED
C CUBE MODUL ROUTINES
C REPORT NUMBER : 81-1
C AUTHORS : C. CYRCHSSOTISIMDZI/ AND R. CELOTO
C PUBLISHER : MASSACHUSETTS INSTITUTE OF TECHNOLOGY
C DEPARTMENT OF OCEAN ENGINEERING
C DESIGN LABORATORY
C CAMBRIDGE MASS 02139, JUNE 1981
C Labeled COMMONS FOR SUBROUTINE MAINPG
C COMMON /MOD1ND/NDEND
C COMMON /KNNUS/ KRUNFIL, KRUNFIL
C COMMON /NATT, INTODE, UMODE
C COMMON /DIALG/ MIERSE
C COMMON /MDNCW/ MCW
C
C VARIABLE AND FUNCTION TYPE DEFINITIONS AND DIMENSIONS
C INTEGER Runfip, Runfip IL
C INTEGER IMODE, UMODE
C INTEGER MCW
C INTEGER ITIM, MNUM, MNUMM, LNTKNS, ITIM(10), MESS(1)
C INTEGER LEMI
C*** START OF SITE DEPENDENT CODE
C INTEGER DIVMIN(12)
C INTEGER DRTMIN(20)
C*** END OF SITE DEPENDENT CODE
C LOGICAL MIERSE
C VARIABLE DATA DEFINITIONS
C ALL VARIABLES IN LABELLED COMMONS ARE INITIALIZED IN BLOCK DATA
C DATA MESS /M1C/
C DATA MNUM/MNUM, RNUMM, WNUMM/
C DATA NITEMS/9/
C DATA MITEMS/4/HIGH
C 1 4HMND, NODE
C 2 4HINH, NODE
C 3 4HMOD, NODE
C 000000500
C 000000510
C 000000520
C 000000530
C 000000540
C 000000550
C 000000560
C 000000570
C 000000580
C 000000590
C 000000600
C 000000610
C 000000620
C 000000630
C 000000640
C 000000650
C 000000660
C 000000670
C 000000680
C 000000690
C 000000700
C 000000710
C 000000720
C 000000730
C 000000740
C 000000750
C 000000760
C 000000770
C 000000780
C 000000790
C 000000800
C 000000810
C 000000820
C 000000830
C 000000840
C 000000850
C 000000860
C 000000870
C 000000880
C 000000890
C 000000900
C 000000910
C 000000920
C 000000930
C 000000940
C 000000950
C 000000960
C 000000970
C 000000980
C SELECT AN ITEM FROM MENU MOD. MAIN AND BRANCH ACCORDINGLY
C
50 CONTINUE
ITEM MENU(ITEM, MENU, ITEM, ITEMS, ITEMS, ITEM, ITEM)
GO TO (100, 200, 300, 400, 500, 600, 700, 800, 900), ITEM
C
C SET STYLE OF MODULE DIALOGUE.
C
100 CONTINUE
CALL DIALOG(MIDRSE)
GO TO 50
C
C SELECT SOURCE OF MODULE INPUT.
C
200 CONTINUE
CALL SOURCE(INODE, DBFILM, DVFLNM, RNRFIL, MTERSE, NCPW)
GO TO 50
C
C SELECT DESTINATION OF MODULE OUTPUT.
C
300 CONTINUE
CALL DEST(INODE, DBFILM, DVFLNM, RNRFIL, MTERSE, NCPW)
GO TO 50
C
C SUPPLY INFORMATION ABOUT THE MODULE FLAGS.
C
400 CONTINUE
CALL MODUDEL(INODE, ONODE, RNRFIL, RNRFIL, MTERSE, NCPW)
GO TO 50
C
C ACCESS THE MODULE READING ROUTINES.
C
500 CONTINUE
IOFLAG-1
CALL MODIO(IOFLAG)
GO TO 50
C
C ACCESS THE MODULE EDITING ROUTINES.
C
600 CONTINUE
IOFLAG-2
CALL MODIO(IOFLAG)
GO TO 50
C ACCESS THE MODULE COMPUTING ROUTINES.
C 700 CONTINUE
    CALL COMPUT
    GO TO 50
C ACCESS THE MODULE WRITING ROUTINES.
C 800 CONTINUE
    I0FLAG-3
    CALL MODI0(I0FLAG)
    GO TO 50
C RETURN CONTROL TO DEX.
C 900 CONTINUE
    CALL END1
    RETURN
    END
--- COMMON MODULE SUBPROGRAM ---

SUBROUTINE MODI(10TAG)
MOD000010

--- PROGRAM DESCRIPTION ---

SUBROUTINE MODI ENABLES ITS USERS TO SELECT THE SEGMENT OR SEGMENTS
OF THE MODUFS VARIABLES THEY WISH TO READ, EDIT OR WRITE.

--- INPUT VARIABLES ---

IOFLAG 1 IF THIS SUBROUTINE WAS INVOKED BY ITEM READ OF MENU MOD.MAIN

2 3

ED1 WRITE

--- LABELLED COMMON VARIABLES ---

--- LABELLED COMMON DIALG HAS BEEN DEFINED IN SUBROUTINE MAINPG.

--- SUBPROGRAMS AND FUNCTIONS CALLED ---

DEX

SIRPAK

IMOVE

MENU

DEX LIBRARY

NONE

MOUNH

INPUT

OUTPUT

--- LABELLED COMMONS ---

COMMON /DIAG1/ MIFSE
COMMON /MDNCMT/ MCM

--- VARIABLE AND FUNCTION TYPE DEFINITIONS AND DIMENSIONS ---

INTEGER IOFLAG

INTEGER MESS(15), IMS, MCM

INTEGER ITEM, MENU, NUMBER(2), NITEMS(16), NITEMS(8)

INTEGER READ(2), EDIT(2), WRITE(2)

LOGICAL MIFSE

LOGICAL MESS, LOGVAL, IMOVE

--- VARIABLE DATA DEFINITIONS ---

DATA IMS /15/
DATA MENU, NUMBER/40 MOD. 41110/
DATA NITEMS/4/
DATA ITEMS /NHINU, NHU /
2 HINPU, NHU /
3 HINH1P, NHU /
4 HINH2H, NHU /
DATA READ /NHREAD, NHU /
DATA EDIT /NHEDIT, NHU /
DATA WRITE /NHWRITE, NHU /
C INITIALIZE MENU.
C MENU = 0.
C PREPARE PROMPTING MESSAGE FOR MENU 'MOD. 10'.
C...
10 CONTINUE
C GOTO (25, 30, 35), 1011AG
C CONTINUE
C GOTO (100, 200, 300, 400), ITEM
C ACTIVATE THE SUBPROGRAM'S ALL OPTION.
C 100 CONTINUE
C MENU = TRUE.
C READ MODULE INPUT DATA.
C
200 CONTINUE
   CALL INPUT(AFITLIG,101LAC)
   IF (.NOT.AFITLIG) GO TO 50
C
C READ MODULE OUTPUT DATA.
C
300 CONTINUE
   CALL OUTPUT(AFITLIG,101LAC)
   IF (.NOT.AFITLIG) GO TO 50
C
C RETURN CONTROL TO THE CALLING PROGRAM.
C
400 CONTINUE
   RETURN
   END
SUBROUTINE INPUT(CALLER, IOFLAG)

SUBROUTINE DESCRIPTION:
SUBROUTINE INPUT PROVIDES THE USER WITH A MENU FROM WHICH TO
CHOOSE WHICH MODULE SEGMENT IT IS DESIRED TO OPERATE NEXT. THE
CHOICES ARE:

ALL MODULE INPUT VARIABLES
THE MODULE UNITS TO BE USED DURING INPUT AND OUTPUT
THE MODULE DIMENSIONS
THE UNITS MODULE ALLOWS THE USER TO SPECIFY THE LENGTH AND FORCE
UNITS TO BE USED DURING INPUT AND OUTPUT. THE DIMENSIONS MODULE
ALLOWS THE USER TO READ, EDIT OR WRITE THE CODE DIMENSIONS: LENGTH,
WIDTH AND HEIGHT.

IF THE ALL OPTION OF THE CALLING PROGRAM IS ACTIVE, THEN
LOCAL ALL OPTION IS SET TO TRUE. UPON INVOKING THIS SUBROUTINE,
IN THIS CASE THE MENU 'INPUT' IS NOT DEFINED.

SUBROUTINE ASSUMPTIONS:
NONE YET.

INPUT VARIABLES:
CALLER: TRUE, IF THE INPUT VALUE OF CALLER WAS TRUE, AND NO ERROR
IOFLAG: IF THE USER WISHES TO READ THE VARIABLES

LABELLED COMMON VARIABLES:
LABELLED COMMON DIALOG HAS BEEN DEFINED IN SUBROUTINE MAINPG.

SUBPROGRAMS AND FUNCTIONS CALLED:
DEX

LIBRARY

NONE

MODULE

MOU严厉打击

DIMENS

LABELLED COMMONS

C C
COMMON /DIALCG/ MILKSE
COMMON /MNP/C/ MPCW
C
C VARIABLE AND FUNCTION TYPE DEFINITIONS AND DIMENSIONS
C
INTEGER IOFLAG
INTEGER MNU, INUM, IITEM, ITM, ITEM
INTEGER MESS(15), LMS, MPCW
INTEGER READ(2), EDIT(2), WRITE(2)
LOGICAL CALAIL, LOCAL
LOGICAL MERSSE, LOGVAL, LMOVEC
C
C VARIABLE DATA DEFINITIONS
C
DATA LMS/15/
DATA MNU/4/ MNU/MNG/4 NETU, NIH /DATA INUM/1/ DATA ITM/10
1
4HNUT, NIH
2
4HIOM, NIH30
3
4HOM, NIH /DATA LOCAL/.FALSE.
 DATA READ /NR/READ, NIH/ / DATA EDIT /WEDIT, NIH/ / DATA WRITE /WHRIT, NIH/.
C
C ACTIVATE THE LOCAL ALL OPTION IF THE CALLING PROGRAM REQUIRES IT.
C NOTE THAT IF THE LOCAL ALL OPTION IS SET IN THIS MANNER, THE MENU
C 'INPUT' IS NOT DEFINED BY THE INVOCATION OF THIS SUBROUTINE.
C
LOCAL=CALLAL
IF (LOCAL) GO 10 200
C
C PREPARE A PROMPTING MESSAGE FOR MENU 'INPUT' AND THEN PROVIDE THE
C MENU TO THE USER.
C
IF (MERSSE) GO 10 40
CALL STRPAK(MESS, LMS, 4NK, 4NSELECT WHICH INPUT VARIABLE SEGMENT INPK0850)
1 104
GO TO (10, 20, 30), IOFLAG
10 LOCAL=1MOVEC(READ, 1, 6, MPCW, MESS, 40, MPCW)
GO TO 10 50
20 LOCAL=1MOVEC(EDIT, 1, 6, MPCW, MESS, 40, MPCW)
GO TO 10 50
30 LOCAL=1MOVEC(WRITE, 1, 7, MPCW, MESS, 40, MPCW)
GO TO 10 50
40 CALL STRPAK(MESS, LMS, 4NK, 21IMATCH INPUT SEGMENT) INPU0850
50 CONTINUE
ITEM MENU IN(MENU, M ITEMS, ITEMS, MESS)
GO TO (100, 200, 300, 400), ITEM
C
C SET THE INPUT ALL OPTION.
C
100 CONTINUE
LOCAL = IFUL.
C
C READ, EDIT OR WRITE THE INPUT/OUTPUT MODULE UNITS.
C
200 CONTINUE
CALL MOUNT(LOCAL, IOFLAG)
S3 (LOCAL) GO TO 300
S4 (.NOT.CAIAL) GO TO 50
CAIAL = FAIS.
GO TO 400
C
C CALL DIMNS TO READ, EDIT OR WRITE THE CUBE DIMENSIONS.
C
300 CONTINUE
CALL DIMNS(LOCAL, IOFLAG)
S3 (LOCAL) GO TO 400
S4 (.NOT.CAIAL) GO TO 50
CAIAL = FAIS.
C
RETURN CONTROL TO CALLING PROGRAM.
C
C
400 CONTINUE
RETURN
END
C------SUBROUTINE CUBI MODULE SUBPROGRAM------

SUBROUTINE MQUILL(CALAI, EDIT)

C------SUBPROGRAM DESCRIPTION------
C SUBROUTINE MQUILL ALLOWS ITS USERS TO SELECT WHICH MODULE UNIT THEY
C WISH TO READ, EDIT OR WRITE. THE CHOICES ARE:
C ALL MODULES
C THE LENGTH UNIT
C THE TIME UNIT
C THE FORCE UNIT
C THE PLANE ANGLE UNIT
C THE TEMPERATURE UNIT

C------SUBPROGRAM ASSUMPTIONS------
C IF THE CALALL PROGRAM ALL OPTION IS ACTIVE, THE LOCAL ALL OPTION IS
C SET TO .TRUE. UPON INVOCING THIS SUBROUTINE. IN THIS CASE THE MNU IS
C NOT DEFINED.
C-------OUTPUT VARIABLES-------
C CALALL: .TRUE. IF THE INPUT VALUE OF CALALL WAS .TRUE. AND NO ERROR
C OCCURRED IN READING OR EDITING A MODULE UNIT
C CALALL: .FALSE. IF THE INPUT VALUE OF CALALL WAS .FALSE. OR AN ERROR
C OCCURRED IN READING OR EDITING A MODULE UNIT

C-------INPUT VARIABLES-------
C EDIT
C CALCALL
C-------LABELLED COMMON VARIABLES-------
C LABELLED COMMON DIALGF, MOUTH, MUNCFW AND RETNO HAVE BEEN DEFINED IN
C SUBROUTINE MAINPG.
C.....UNITS INITIALIZED IN BLOCK DATA
C PSIING: THE PROGRAM STANDARD LENGTH UNIT IS MILL
C UOLUN: DENOTES THE INPUT/OUTPUT LENGTH UNIT
C = 1 IF THE INPUT/OUTPUT UNIT IS INCH
C = 2 FOOT
C = 3 NAUTICAL MILL
C = 4 MILLIMETER
C = 5 CENTIMETER
C = 6 METER
C = 7 KILOMETER

C.....UNHIO INITIALIZED IN BLOCK DATA
C DBLUNW: WHERE THE DATABASE NAME OF THE LENGTH UNIT IS STORED
C DBLUNW: WHERE INFORMATION FOR IDENTIFYING THE LENGTH UNIT VARIABLE
C IS STORED
C UNIM: LOAGAIN TO BE USED TO READ OR WRITE THE LENGTH UNIT FROM A
C SEQUENTIAL FILE
C DEFLUN: THE DEFAULT VALUE OF THE LENGTH UNIT

C

MU00010
C

MU00020
C

MU00040
C

MU00050
C

MU00060
C

MU00070
C

MU00080
C

MU00090
C

MU00100
C

MU00110
C

MU00120
C

MU00130
C

MU00140
C

MU00150
C

MU00160
C

MU00170
C

MU00180
C

MU00190
C

MU00200
C

MU00210
C

MU00220
C

MU00230
C

MU00240
C

MU00260
C

MU00270
C

MU00280
C

MU00290
C

MU00300
C

MU00310
C

MU00320
C

MU00330
C

MU00340
C

MU00350
C

MU00360
C

MU00370
C

MU00380
C

MU00390
C

MU00400
C

MU00410
C

MU00420
C

MU00430
C

MU00440
C

MU00450
C

MU00460
C

MU00470
C SUBPROGRAM BLOCK DATA.
C.......UNITS INITIALIZED IN BLOCK DATA
C PSIUN: 1 THE PROGRAM STANDARD FORC UNIT IS KILOPOUND
C OUN: THE INPUT/OUTPUT FORCE UNIT
C = 1 IF THE INPUT/OUTPUT FORCE UNIT IS POUNDAL
C = 2 POUND (FORCE)
C = 3 SHORT TON
C = 4 LONG TON
C = 5 DYN
C = 6 NEWTON
C = 7 KILOPOUND
C.......UNITS INITIALIZED IN BLOCK DATA
C DBUN: WHERE THE DATABASE NAME OF THE FORCE UNIT IS STORED
C DBUN: WHERE INFORMATION FOR IDENTIFYING THE FORCE UNIT VARIABLE IS STORED
C IS STORED
C FUN: FORMA TO BE USED TO READ OR WRITE THE FORC UNIT TO A SEQUENTIAL FILE
C DEFUN: THE DEFAULT VALUE OF THE FORCE UNIT IS STORED
C.......UNITS INITIALIZED IN BLOCK DATA
C PSIUN: 1 THE PROGRAM STANDARD TIME UNIT IS SECOND
C OUN: THE INPUT/OUTPUT TIME UNIT
C = 1 IF THE INPUT/OUTPUT TIME UNIT IS SECOND
C = 2 MINUTE
C = 3 HOUR
C = 4 DAY
C = 5 WEEK
C = 6 MONTH (30 DAY MONTH)
C = 7 YEAR (360 DAY YEAR)
C.......UNITS INITIALIZED IN BLOCK DATA
C DBUN: WHERE THE DATABASE NAME OF THE TIME UNIT IS STORED
C DBUN: WHERE INFORMATION FOR IDENTIFYING THE TIME UNIT VARIABLE IS STORED
C IS STORED
C FUN: FORMA TO BE USED TO READ THE TIME UNIT FROM OR WRITE IT TO A FILE
C DEFUN: THE DEFAULT VALUE OF THE TIME UNIT IS STORED
C.......UNITS INITIALIZED IN BLOCK DATA
C PSIUN: 1 THE PROGRAM STANDARD PLANE ANGLE UNIT IS CYCLE
C OUN: THE INPUT/OUTPUT ANGLE UNIT
C = 1 IF THE INPUT/OUTPUT ANGLE UNIT IS CYCLE
C = 2 RADIAN
C = 3 DEGREE (ANG)
C = 4 MINUTE (ANG)
C = 5 SECOND (ANG)
C.......UNITS INITIALIZED IN BLOCK DATA
C DBUN: WHERE THE DATABASE NAME OF THE ANGLE UNIT IS STORED
C DBUN: WHERE INFORMATION FOR IDENTIFYING THE ANGLE UNIT VARIABLE IS STORED
C IS STORED
C DEFINE: FORMAT TO BE USED TO READ THE ANGLE UNIT TO OR WRITE IT FROM
C A SEQUENTIAL FILE
C DEFINE: THE DEFAULT VALUE OF THE ANGLE UNIT
C DEFINE: THE PROGRAM STANDARD TEMPERATURE UNIT IS CELSIUS DEGREES
C DEFINE: THE DEFAULT VALUE OF THE TEMPERATURE UNIT
C INITI: DENOTES THE INPUT/OUTPUT TEMPERATURE UNIT
C IF THE INPUT/OUTPUT TEMPERATURE UNIT IS CELSIUS DEGREES
C = 2 FAHRENHEIT DEGREES
C = 3 KELVIN DEGREES
C = 4 RANKINE DEGREES
C...... IPTIO INITIALIZED IN BLOCK DATA
C DBFIP: WHERE THE DATABASE NAME FOR THE TEMPERATURE UNIT IS STORED
C DBFIP: WHERE INFORMATION FOR IDENTIFYING THE TEMPERATURE UNIT
C VARIABLE IS STORED
C IPTIN: FORMAT TO BE USED TO READ THE TEMPERATURE UNIT FROM OR WRITE
C IT TO A SEQUENTIAL FILE
C... DEFIPU: THE DEFAULT VALUE OF THE TEMPERATURE UNIT
C--------------------------------------------SUBPROGRAMS AND FUNCTIONS CALLED----------------------------------------------
C DEX
C SIRPAK
C TMOVC
C MULUN
C DEX LIBRARY
C MUL
C MODUL
C TUNIT
C UUNIT
C AUNIT
C IUNIT
C LABELD COMMONS
C COMMUN /DIALG1/ MTHRS
C COMMUN /INTOUT/ IMODE, OMOUL
C COMMUN /MOUNT/ MCFW
C COMMUN /NEWFS/ RMRFIL, RMRFIL
C COMMUN /TUNITS/ PSTUH, U10UH
C COMMUN /TUNITS/ PSTUH, U10UH
C COMMUN /TUNITS/ PSTUH, U10UH
C COMMUN /TUNITS/ PSTUH, U10UH
C COMMUN /AUNIT/ PSTAUN, U10AUN
C COMMUN /IUNIT/ PSTUH, U10I1
C COMMUN /TUNIT/ DBTUNN, DBTUNC, U10UN, DEFIU
C COMMUN /TUNIT/ DBTUNN, DBTUNC, U10UN, DEFIU
C COMMUN /AUNIT/ DBAUNN, DBAUNC, AUNITM, DEFAUN
C COMMUN /AUNIT/ DBAUNN, DBAUNC, AUNIT, DEFAUN
C COMMUN /AUNIT/ DBAUNN, DBAUNC, AUNITM, DEFAUN
C COMMUN /AUNIT/ DBAUNN, DBAUNC, AUNIT, DEFAUN
COMMON /IPINFO/, DBIPUN, DBIPUC, IPFUN, DEFPUN

C

C VARIABLE AND FUNCTION TYPE DEFINITIONS AND DIMENSIONS

C

INTEGER IOLAG
INTEGER IMODE, OMODE, TMODE
INTEGER MESS(11), IM5, MCPP
INTEGER TMAP, MNUMT, MNUMN(2), NITEMS, NITEMS(14)
INTEGER UIOUN, UIOTUN, UIOUN, UIOUN, DIOC1PU
INTEGER DBOUNN(2), DBOUNN(2), DBAOUNN(2), DBIPUN(2)
INTEGER DBIPUN(16), DBIPUC(16), DBIPUN(2), DBAOUNN(16), DBIPUN(16)
INTEGER TMAPNN(2), TUNERN(2), TUNERN(2), AUNERN(2), APUNERN(2)
INTEGER DEFLAG, DEFUN, DEFUN, DEFAM, DEFPUN
INTEGER RNFIF, RNWFF, RNWAT
INTEGER PSLUN, PSLUN, PSLUN, PSAMN, PSIPUN
INTEGER MNEW(16)
INTEGER READ(2), EDIT(2), WRIT(2)
LOGICAL MERRSE, PMPREP
LOGICAL CAIAT, LOCAL
LOGICAL LOGVAL, LOGVEC

C VARIABLE DATA DEFINITIONS

C

DATA IMS //
DATA MNUMN/64BN1, 4H //
DATA NITEMS//
DATA RMAP //
2 NTMPN, 4H //
3 NITEMN, 4H //
4 NITEMC, 4H //
5 NITEML, 4H //
6 NTMPF, 4H //
7 NTMPF, 4H //
DATA READ /WHREAD, 4H, C //
DATA EDIT /WHEDIT, 4H, C //
DATA WRIT /WHWRIT, 4H, C //

C INITIALIZE THE VALUE OF PMPREP.

C PMPREP = .T.R.

C SET THE VALUE OF TMODE AND RNFILE ACCORDING TO WHETHER THE
C USER WISHES TO READ, EDIT OR WRITE.

C

II (IOLAG,.EQ.3) GO TO 10
TMODE=IMODE
RNFILE=RNFILE

MX001420
MX001430
MX001440
MX001450
MX001460
MX001470
MX001480
MX001490
MX001500
MX001510
MX001520
MX001530
MX001540
MX001550
MX001560
MX001570
MX001580
MX001590
MX001600
MX001610
MX001620
MX001630
MX001640
MX001650
MX001660
MX001670
MX001680
MX001690
MX001700
MX001710
MX001720
MX001730
MX001740
MX001750
MX001760
MX001770
MX001780
MX001790
MX001800
MX001810
MX001820
MX001830
MX001840
MX001850
MX001860
MX001870
MX001880
GO TO 15
10 CONTINUE
LOCAL: CALL
IF (LOCAL) GO TO 200
C
C PREPARE PROMPTING MESSAGE FOR MENU 'UNIT'.
C
20 IF (Mulling) GO TO 40
CALL STRIP(MESS,IN,INK,12HSELICT WHICH UNIT TO)<
GO TO (29,30,35),10 FLAG
25 CONTINUE
LOCAL: MOVIC(RLEAD,1,6,NCPE,MESS,22,NCPE)
GO TO 50
30 CONTINUE
LOCAL: MOVIC(IDIT,1,6,NCPE,MESS,22,NCPE)
GO TO 50
35 CONTINUE
LOCAL: MOVIC(WRIT,1,7,NCPE,MESS,22,NCPE)
GO TO 50
40 CONTINUE
CALL STRIP(MESS,IN,INK,12HSELICT WHICH UNIT?)
C
C SELECT AN ITEM FROM MENU 'UNIT' AND BRANCH ACCORDINGLY.
C
50 CONTINUE
ITEM MENU1(MI1NUM,IN1MS,IN1MS,MESS)
GO TO (100,200,300,400,500,600,700),ITEM
C
C ACTIVATE THE SUBPROGRAM'S ALL OPTION.
C
100 CONTINUE
LOCAL: TRUE.
C
C READ, EDIT OR WRITE THE LENGTH UNIT.
C
200 CONTINUE
CALL UNIT1(WITH UN,LOCAL,
1 10 FLAG, LONG, MULL), NCPE, NMPE, NCW,
2 10 FLAG, LONG, MULL, MUEL, NCW,
3 PMPREP, NMPE, NCW,
4       RUI T, LUNTM,  
5       DITLMN)  
11 IF(LOCAL) GO TO 100  
11 IF (.NOT.CALL) GO TO 20  
  CALL .FALSE.  
     GO TO 100  
C  
C READ, EDIT OR WRITE THE TIME UNIT.  
C  
300 CONTINUE  
CALL TUNIT(JTJUN, LOCAL,  
1       JOU, JINODE, MTHS, NCPE,  
2       DITLMN, DITUNC,  
3       PMPEP, PMES,  
4       RNIT, TUNITM,  
5       DITLMN)  
IF(LOCAL) GO TO 100  
IF (.NOT.CALL) GO TO 20  
  CALL .FALSE.  
     GO TO 100  
C  
C READ, EDIT OR WRITE THE FORCE UNIT.  
C  
400 CONTINUE  
CALL TUNIT(JTJUN, LOCAL,  
1       JOU, JINODE, MTHS, NCPE,  
2       DITLMN, DITUNC,  
3       PMPEP, PMES,  
4       RNIT, TUNITM,  
5       DITLMN)  
IF(LOCAL) GO TO 100  
IF (.NOT.CALL) GO TO 20  
  CALL .FALSE.  
     GO TO 100  
C  
C READ, EDIT OR WRITE THE PLANE ANGLE UNIT.  
C  
500 CONTINUE  
CALL AUNIT(JTAUN, LOCAL,  
1       JOTA, JINODE, MTHS, NCPE,  
2       DITAUN, DITUNC,  
3       PMPEP, PMES,  
4       RNIT, AUNITM,  
5       DITAUN)  
IF(LOCAL) GO TO 600  
IF (.NOT.CALL) GO TO 20  
  CALL .FALSE.  
     GO TO 200
GO TO 700
C
C READ, EDIT OR WRITE THE TEMPERATURE UNIT.
C
600 CONTINUE
   CALL TPUNIT(DT1, TP, LOCAL, 
      1   TOTAC, TORMODE, MIERSE, NCPW, 
      2   DI1PUR, DI1PU, 
      3   PMPREP, PMES, 
      4   RN1IL, TP1FIRM, 
      5   DI1IPU)
   IF (LOCAL) GO TO 700
   IF (.NOT.CALAI) GO TO 20
   CALAI=.FALSE.
C
C RETURN CONTROL TO THE CALLING PROGRAM.
C
700 CONTINUE
   RETURN
   END
G-------------------CUBE MODULE SUBPROGRAM---------------------
SUBROUTINE DIMEN(A, L, LLAG)
G-------------------SUBPROGRAM DESCRIPTION---------------------
SUBROUTINE DIMEN FIRST CALLS UNIT 50 TO OBTAIN THE MULTIPLES-
G CALIBRT LENGTH CONVERSION FACTOR AND THE NAMES OF THE LENGTH UNITS
G TO BE USED DURING INPUT. DIMENS THEN PROVIDES A MENU FROM
G FROM WHICH THE USER SELECTS WHICH DIMENSION IT IS DESIRED TO READ,
G EDIT OR WRITE. THE CHOICES ARE:
G ALL DIMENSIONS
G LENGTH
G WIDTH
G HEIGHT
G OPTION IS SET TO TRUE. UPON INVOKING THIS SUBROUTINE AND THE
G MENU IS NOT DISPLAYED.
G-------------------SUBPROGRAM ASSUMPTIONS---------------------
G NULLE YET
G-------------------OUTPUT VARIABLES-----------------------------
C ALENG: .TRUE. IF THE INPUT VALUE OF ALLIFG WAS .TRUE. AND NO
C ERROR OCCURRED WHEN READING A UNIT OR EDITING ANY
C DIMENSION
C FALSE. IF THE INPUT VALUE OF ALLIFG WAS .FALSE. OR AN
C ERROR OCCURRED WHEN READING A UNIT OR EDITING A
C DIMENSION
G-------------------INPUT VARIABLES-------------------------------
C ALLIFG: .TRUE. THE ALL OPTION OF THE CALLING PROGRAM IS ACTIVE
C .FALSE. THE ALL OPTION OF THE CALLING PROGRAM IS NOT ACTIVE
C LLAG= 1 IF THE USER WISHES TO INVOKE RSLG
C 2 RSEDIT
C 3 RSCOMP
G-------------------LISED COMMON VARIABLES----------------------
C LINED COMMONS DIALOG, INOUT, MONCP, AND REINS HAVE BEEN DEFINED
C IN SUBROUTINE MAING. LISED COMMON UNITS HAS BEEN DEFINED IN
C SUBROUTINE MUNIT.
C.....INFO INITIALIZED IN BLOCK DATA
C L : LENGTH OF THE CUBE IN PROGRAM STANDARD UNITS
C LUNIT: DATABASE NAME FOR LENGTH
C LUNCG: DATABASE COMMENT DESCRIBING LENGTH
C DEFLG: THE DEFAULT VALUE FOR LENGTH
C.....INFO INITIALIZED IN BLOCK DATA
C W : WIDTH OF THE CUBE IN PROGRAM STANDARD UNITS
C WENAM: DATABASE NAME FOR WIDTH
C WNCNG: DATABASE COMMENT DESCRIBING WIDTH
C DEFW: THE DEFAULT VALUE FOR WIDTH
C.....INFO INITIALIZED IN BLOCK DATA
C H : HEIGHT OF THE CUBE IN PROGRAM STANDARD UNITS
C HEINAM: DATABASE NAME FOR HEIGHT
C HEINC: DATABASE NAME FOR HEIGHT
LOGICAL MIERSE , VITAL, PMPIF
LOGICAL LOGVAL, IMOVIC
LOGICAL AIATLG, LOCALI
LOGICAL UNIT1
LOGICAL RSCLDR, RSCLDI, RSCOMP, RAIDLX, NAMESI, RAKNUM, NAMUC, NAMUG, NAMUF,
RALL A, W, H, DETAIL, DEFAULT, DEFAULT
RIAL CONVIM, CONVI

C

C VARIABLE DATA DEFINITIONS

C DATA MNSNUM /MINDME, NMS10/
DATA NALSNS /5/
DATA ITEMS /NIAL, W1 /
1 W1 ENG, W1 H
2 W1 MEXT, W1 H
3 W1 EXP, W1 H
4 W1 HONE, W1 H /
DATA CONVVA /0.0/
DATA ENS /15/
DATA READ /NNREAD, W1 /
DATA EDS /NNEDIT, W1 /
DATA NEN /NENRT, W1 /
DATA NNuge, NENG, NGE, N1, 4/

C

C DELIMIT THE NAMES OF THE UNITS FOR THE INPUT DIMENSIONS AND THE
C MULTIPlicative CONVERSION FACTOR TO CONVERT THE INPUT UNITS TO THE
C PROGRAM STANDARD UNITS.

C LOGVAL=UNIT1( CONVIM, NAM02, NAM06, NAM12, AIATLG,

1 PSIMUN, U1QULM, NCPW)
11 (. NO1. LOGVAL) GO TO 99999

C

C ACTIVATE THE LOCAL ALL OPTION IF THE CALLING PROGRAM REQUESTS IT.
C NOTE THAT IF THE LOCAL ALL OPTION IS SET IN THIS MANNER, MENU
C 'DIMENSION' IS NOT DEFINED BY THE INVOCATION OF THIS SUBROUTINE.

C LOCAL-LATLG
II (LOCALI) GO TO 200

C PREPARE A PROMPTING MESSAGE FOR MENU 'DIMEN'.

C 5 CONTINUE
IF (MIERSI) GO TO 80
CALL STRPAK(MESS, IMS, NMS, 33 , NS, SELECT THE DESired DIMENSION TO<) DIM0180
GO TO (10, 20, 30), IFLAG
10 CONTINUE
LOGVAL=IMOVIC(READ, 1, 6, NCPW, MESS, 33, NCPW)
GO TO 50
20 CONTINUE
LOGVAL-MOVIC(EDIT, 1, 6, NCPW, MESS, 33, NCPW)
GO TO 50
30 CONTINUE
LOGVAL-MOVIC(WRITE, 1, 7, NCPW, MESS, 33, NCPW)
GO TO 50
40 CONTINUE
CALL STRPK(MESS, IMS, 4H, 17, MESS DIMENSION)
50 CONTINUE
ITEM MENUIN(MNNUM, MITEMS, XI(MS, MESS)
GO TO (100, 200, 300, 400, 99999), ITEM
C SET LOCAL ALL OPTION.
C 100 CONTINUE
LOCAL - TRUE.
C DETERMINE WHICH OPERATION TO PERFORM ON THE LENGTH DIMENSION
C AND BRANCH ACCORDINGLY.
C 200 CONTINUE
GO TO (210, 220, 230), IOFLAG
C READ INPUT LENGTH.
C 210 CONTINUE
LOGVAL-RSCDIR(I, LOCAL)
1 MIERSE, NMODE, NCPW,
2 IENMAM, CONVM, CONVI, NAM12, .FALSE.,
3 .TRUE., PMES, TMORN,
4 RNFIL, LWFRM, DETAIL)
11 (LOCAL) GO TO 300
11 (.NOT.IOFLAG) GO TO 5
AILFIC = .FALSE.
GO TO 99999
C EDIT LENGTH.
C 220 CONTINUE
LOGVAL-RSCEDIT(I, LOCAL)
1 MIERSE, NCPW,
2 IENNAM, CONVM, CONVI, NAM12,
3 .TRUE., PMES, TMORN,
4 RNFIL, LWFRM,
5 DETAIL)
11 (LOCAL) GO TO 300
II (.NOT.ALLIG) GO TO 5
ALLIG=.FALSE.
GO TO 99999
C
C WRITE LENGTH.
C
230 CONTINUE
LOGVAL-RSCOMP(LOCAL,
  1 .MRESE, OMODE, NCPW,
  2 .LNNAM, CONV1M, CONVIA, NAML12,
  3 .TRUE., PMES, MPORCH,
  4 RNHF1, IWRM)
II (LOCAL) GO TO 300
II (.NOT.ALLIG) GO TO 5
ALLIG=.FALSE.
GO TO 99999
C
C DETERMINE WHICH OPERATION TO PERFORM ON THE WIDTH DIMENSION AND
C BRANCH ACCORDINGLY.
C
300 CONTINUE
GO TO (310, 320, 330), IOFLAG
C
C READ WIDTH.
C
310 CONTINUE
LOGVAL-RSCURW, ILOCAL,
  1 MRESE, OMODE, NCPW,
  2 .MINNAM, CONV1M, CONVIA, NAML12, .FALSE.,
  3 .TRUE., PMES, MPORCH,
  4 RNHF1, IWRM, DEFAULT)
II (LOCAL) GO TO 400
II (.NOT.ALLIG) GO TO 5
ALLIG=.FALSE.
GO TO 99999
C
C EDIT WIDTH.
C
320 CONTINUE
LOGVAL-RSCEDIW, ILOCAL,
  1 MRESE, NCPW,
  2 .MINNAM, CONV1M, CONVIA, NAML12,
  3 .TRUE., PMES, MPORCH,
  4 RNHF1, IWRM,
  5 DEFAULT)
II (LOCAL) GO TO 400
II (.NOT.ALLIG) GO TO 5
ALLIG., FAISE.
GO TO 99999
C
C WRITE HEIGHT.
C
430 CONTINUE
LOCAL-RANDMP(LOCAL, 1
INJFSE, OMODE, NGPM,
2
II, III, IV, NEN, NG,)
3
CONV, CONV, NAM12,
4
TRUE, PM, S, IMORN,
5
RE, N, IRM, HFIN)
II (LOCAL) GO TO 99999
II (.NOT. ALLIG) GO TO 5
ALLIG., FAISE.
C
C RETURN TO CALLING PROGRAM.
C
99999 CONTINUE
RETURN
END
C-- CUBE MODULE SUBPROGRAM-----------------------------COM0010
C
SUBROUTINE COMPUT
C
C-- SUBPROGRAM DESCRIPTION-----------------------------COM0020
C
THIS SUBROUTINE CALCULATES THE VOLUME AND WEIGHT OF A CUBE OF SALT
C
WATER.
C
THE STANDARD UNITS OF THIS SUBROUTINE ARE METERS AND KILOPONDS.
C
LABELED COMMON VARIATIANTS-----------------------------COM0030
C
LABELED COMMONS LINFO, VINFO, WINFO AND WITINFO HAVE BEEN DEFINED
C
IN SUBROUTINES DIMENS AND VANDWI.
C
C
LABELED COMMONS
C
COMMON /LINFO/ L
COMMON /VINFO/ W
COMMON /WINFO/ WI
COMMON /VITINFO/ VIT
COMMON /WITINFO/ WIT
C
C
VARIABLE AND FUNCTION TYPE DEFINITIONS AND DIMENSIONS
C
REAL L,V,W,VI
REAL RHO
C
VARIABLE DATA DEFINITIONS
C
DATA RHO/1023.2/
C
CALCULATE THE VOLUME IN CUBIC METERS.
C
DO 10 1=1,4
V(1) = L**3
10 CONTINUE
C
CALCULATE THE WEIGHT OF THE CUBE IF THE WEIGHT DENSITY OF SALT
C
WATER IS 1023.2 KILOPONDS/MM**3.
C
DO 20 1=1,4
WIT(1) = V(1) * RHO
20 CONTINUE
RETURN
END
COMMON /MN,CPW/ NCYW
C VARIABLE AND FUNCTION TYPE DEFINITIONS AND DIMENSIONS
C
INTEGER IOITAG
INTEGER MN,NUMM(2),NITEMS,ITEMS(8),ITEM
INTEGER MS(16),IMS,NCYW
INTEGER READ(2),EDIT(2),WRITE(2)
LOGICAL CALAIL,LOCAL
LOGICAL INFUSE,LOGVAL,MOVLC
C
C VARIABLE DATA DEFINITIONS
C
DATA IMS/16/
DATA MINUM/400UPT,4CH1 /
DATA NITEMS/10/
DATA ITEMS/4HNT,4HIS ,
1 4HNT1,4HIS ,
2 4HNT2,4HIS ,
3 4HNT3,4HIS ,
DATA LOCAL/,/FALSE/,/
DATA READ/,/READ,4H< /
DATA EDIT/,/EDIT,4H< /
DATA WRITE/,/WRITE,4H< /
C
C ACTIVATE THE LOCAL ALL OPTION IF THE CALLING PROGRAM REQUIRES IT.
C NOTE THAT IF THE LOCAL ALL OPTION IS SET IN THIS MANNER, MENU
C 'OUTPUT' WILL NOT BE DEFINED BY THE INVOCATION OF THIS SUBROUTINE.
C
LOCAL CALAIL
IF (LOCAL) GO TO 200
C
C PREPARE A PROMPTING MESSAGE FOR MENU 'OUTPUT' AND THEN PROVIDE THE
C MENU TO THE USER.
C
CONTINUE
CALL SIRPAK(MESS,IMS,4HK ,4HSELGET WHICH OUTPUT VARIABLE SEGMENT00100560
10 104)
GO TO (10,20,10),IOITAG
10 LOGVAL=MOVLC(READ,1,6,NCYW,MESS,41,NCYW)
GO TO 50
20 LOGVAL=MOVLC(EDIT,1,6,NCYW,MESS,41,NCYW)
GO TO 50
30 LOGVAL=MOVLC(WRITE,1,7,NCYW,MESS,41,NCYW)
50 CONTINUE
C ITEM MENU(NMINUM,NITEMS,ITEMS,MESS)
GO TO (100,200,300,400),ITEM
C
C SET THE OUTPUT ALL OPTION.
C 100 CONTINUE
    LOCALL = .TRUE.

C READ, EDIT OR WRITE THE INPUT/OUTPUT MODULE UNITS.
C
C 200 CONTINUE
    CALL MUNIT(LOCALI,IOFTAG)
    IF (LOCALI) GO TO 300
    IF (.NOT. CALAI) GO TO 50
    CALAI = .FALSE.
    GO TO 400

C CALL VAMORI TO READ, EDIT OR WRITE THE CUBE VOLUME AND WEIGHT.
C
C 300 CONTINUE
    CALL VAMORI(LOCALI,IOFTAG)
    IF (LOCALI) GO TO 400
    IF (.NOT. CALAI) GO TO 50
    CALAI = .FALSE.

C
C RETURN CONTROI TO CALLING PROGRAM.
C
C 400 CONTINUE
    RETURN
    END
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A.
C-----CUBE MODULE SUBPROGRAM------------------------------------------  VA00010
C-----SUBROUTINE VAMI(NALLIG, IFLAG)---------------------------------  VA00020
C-----SUBPROGRAM DESCRIPTION------------------------------------------  VA00030
C-----SUBROUTINE VAMI FIRST CALLS UNIIT, UNIIIT AND UNIIT TO OBTAIN THE VA00040
C MULTIPlicative LENGTH, FORCE (WEIGHT) AND VOLUME CONVERSION FACTORS VA00050
C AND THE NAMES OF THE FORCE AND VOLUME UNITS TO BE USED DURING OUTPUT.  VA00060
C VAMI THEN PROVIDES A MENU FROM WHICH THE USER SELECTS WHICH VA00070
C PROGRAM COMPUTATION RESULTS IT IS DESIRED TO READ, EDIT OR WRITE.  VA00080
C THE CHOICES ARE:  VA00090
C ALL RESULTS  VA01000
C VOLUME  VA01100
C WEIGHT  VA01200
C VAMI THEN CALLS RAILDR, RAREDT, OR RARUMP AS NECESSARY FOR EACH VA01300
C VARIABLE.
C IF THE ALL OPTION OF THE CALLING PROGRAM IS ACTIVE, THE LOCAL ALL VA01400
C OPTION IS SET TO TRUE IMMEDIATELY AFTER THE UNITS INFORMATION IS VA01500
C OBTAINED. IN THIS CASE, MENU 'RESULTS' IS NOT INVOKED.  VA01600
C-------------------------------------SUBPROGRAM ASSUMPTIONS-------------------  VA01700
C NONE YET
C-------------------------------------OUTPUT VARIABLES----------------------------------  VA02000
C ALLIG: TRUE IF THE INPUT VALUE OF ALLIG WAS TRUE, AND NO ERROR  VA02100
C OCCURRED WHEN READING ANY UNIT OR EDITING ANY PROGRAM  VA02200
C RESULT  VA02300
C .FALSE. IF THE INPUT VALUE OF ALLIG WAS FALSE, OR AN ERROR  VA02400
C OCCURRED WHEN READING A UNIT OR EDITING A PROGRAM  VA02500
C C RESULT  VA02600
C-------------------------------------INPUT VARIABLES----------------------------------  VA02800
C ALLIG: TRUE IF THE ALL OPTION OF THE CALLING PROGRAM IS ACTIVE  VA02900
C .FALSE. THE ALL OPTION OF THE CALLING PROGRAM IS NOT ACTIVE  VA02910
C IFLAG: I IF THE USER WISHES TO INVOKE RAILDR  VA03000
C 2 RAREDT  VA03100
C 3 RARUMP  VA03200
C--------------------------------------Labeled COMMON VARIABLES--------------------------  VA03300
C LABELING COMMONS DIAGL, ENOUT, MNPCW AND REMUS HAVE BEEN DEFINED  VA03400
C IN SUBROUTINE MAINPG; Labeled COMMONS LUNITS AND FUNITS HAVE BEEN  VA03500
C DEFINED IN SUBROUTINE MAINPG.
C------VINO INITIALIZED IN BLOCK DATA  VA03600
C V : VOLUME OF THE CUBE IN PROGRAM STANDARD UNITS  VA03700
C VNAME: DATABASE NAME FOR VOLUME  VA03800
C VNAME: COMMON DESCRIBING VOLUME  VA03900
C NDEFV: THE NUMBER OF VOLUME DEFAULT VALUES  VA04000
C DEFAULT VALUES FOR VOLUME  VA04100
C C------WINFO INITIALIZED IN BLOCK DATA  VA04200
C W : WEIGHT OF THE CUBE IN PROGRAM STANDARD UNITS  VA04300
C WNAME: DATABASE NAME FOR WEIGHT  VA04400
C WNAME: COMMENT DESCRIBING WEIGHT  VA04500
C NDEFW: THE NUMBER OF WEIGHT DEFAULT VALUES  VA04600
C DEFAULT VALUES FOR WEIGHT  VA04700
C C------ANISIA INITIALIZED IN BLOCK DATA  VA04800
COMMON /DIALGF, MIERE
COMMON /INQUT/, MNODE, OMODE
COMMON /KRONCM/, NCPW
COMMON /KRONCM/, NUMIL, RNWIL
COMMON /UNITIS/, PSIUIN, UIOULN
COMMON /UNITIS/, PSIFUN, UIOULN
COMMON /UNITIS/, V4(4), VUHAN, VMORCH, NULLV, DELAV(4)
COMMON /UNITIS/, W4(4), VETHAN, VTHORCH, HOLLWI, DIALV(4)
COMMON /UNITIS/, ANKFRM

COMMON /DIAGF/, MKERL
COMMON /INQUT/, MNODE, OMODE
COMMON /KRONCM/, NCPW
COMMON /KRONCM/, NUMIL, RNWIL
COMMON /UNITIS/, PSIUIN, UIOULN
COMMON /UNITIS/, PSIFUN, UIOULN
COMMON /UNITIS/, V4(4), VUHAN, VMORCH, NULLV, DELAV(4)
COMMON /UNITIS/, W4(4), VETHAN, VTHORCH, HOLLWI, DIALV(4)
COMMON /UNITIS/, ANKFRM

VARIABLE AND FUNCTION TYPE DEFINITIONS AND DIMENSIONS

INTEGER TOFIAG, IMDRE, OMODE, NCPW
INTEGER KMNIL, RNWIL
INTEGER ANINF(3)
INTEGER NM02(1), NAM06(2), NAM12(3)
INTEGER NM02(1), NAM03(1), NAM12(3)
INTEGER UPSUL, UIOULN
INTEGER PSIUIN, UIOULN
INTEGER MESS(3), IMS
INTEGER VUHAN(2), VETHAN(2)
INTEGER PMN(16), VMORCH(16)
INTEGER NULLV, HOLLWI
INTEGER NENRI(2), NIHEA, TLOM(16)
INTEGER READ(2), EDI(2), WRITE(2)
INTEGER KIQUL, KIROM, NTO
INTEGER VGHOT, SIGCH, FROM

VARIOUS: FORMAT TO BE USED FOR THE CODE PROPERTIES WHEN READING FROM
VARIOUS: ON WRITING TO A SEQUENTIAL FILE

SUBPROGRAMS AND FUNCTIONS CALLED

--
LOGICAL INTISE, VITAL, PMPREP
LOGICAL LOGVAL, INOVEC
LOGICAL A11FLG, LOCAL
LOGICAL UNIT11, UNIT12, UNIVOL
LOGICAL BAIKIN, KAREDI, RARDMP
REAL V, W, DELAV, DELATW
REAL CONVIM, CONVFA
REAL CNV10M, CNVF1A
C VARIABLE DATA DEFINITIONS
C
DATA MENUH /HIRESU, HHITS /
DATA ITMS/HAIT, AH /
1 4IIVOM, HHM /
2 4IVOM, HHM /
3 4IVOM, HHM /
DATA CONVI /O.0/8
DATA CONVFA/O.0/
DATA LNS/13/
DATA R/AUD/HIREAD, NIH< /
DATA EDIT /HIREDT, NIH< /
DATA UNH/HM3RT, NIH< /
DATA MXG101, VG01, WGD1, IROR /H4, 4, 1/
C DETERMINE THE NAMES OF THE UNITS FOR THE OUTPUT RESULTS AND THE STANDARD UNITS TO OUTPUT UNITS.
C
LOGVAL=UNIT11(CONVIM, NAM102, NAM106, NAM112, ALLFLG,
1 PSTFUN, U10FUN, NCWM)
II (.NOT. LOGVAL) GO TO 99999
LOGVAL=UNIVOL(CONVI, UNIVOL, ALLFLG,
1 CONVIM, NAM106, NCM)
II (.NOT. LOGVAL) GO TO 99999
LOGVAL=UNIT11(CONVIM, NAM102, NAM103, NAM112, ALLFLG,
1 PSTFUN, U10FUN, NCWM)
II (.NOT. LOGVAL) GO TO 99999
C ACTIVATE THE LOCAL ALL OPTION IF THE CALLING PROGRAM REQUIRES IT.
C NOTE THAT IF THE LOCAL ALL OPTION IS SET IN THIS MANNER, MENU 'RESULTS' WILL NOT BE DEFINED BY THE INVOCATION OF THIS SUBROUTINE.
C
LOCAL = ALLFLG
IF (LOCAL) GO TO 200
C PREPARE A PROMPTING MENU FOR MENU 'RESULTS'.
C
C 5 CONTINUE
  10 (MERSI) GO TO 40
  CALL SIRPAK(MESS,IMS,AHK ,30)SELECT THE DESIRED RESULT TO C
  GO TO (10,20,30),10FLAG
10 CONTINUE
  LOGVAL=MOVLC(READ,1,6,NCPM,MESS,30,NCPM)
  GO TO 50
20 CONTINUE
  LOGVAL=MOVLC(EDIT,1,6,NCPM,MESS,30,NCPM)
  GO TO 50
30 CONTINUE
  LOGVAL=MOVLC(MWRITE,1,7,NCPM,MESS,30,NCPM)
  GO TO 50
40 CONTINUE
  CALL SIRPAK(MESS,IMS,AHK ,40)HIGH RESULT TO C
  GO TO (100,200,300,99999),100
C SET LOCAL ALL OPTION.
C 100 CONTINUE
  LOCALV=.TRUE.
  C
  C DETERMINE WHICH OPERATION TO PERFORM ON THE VOLUME VALUE AND
  C BRANCH ACCORDINGLY.
C 200 CONTINUE
  GO TO (210,220,230),10FLAG
C READ VOLUME.
C 210 CONTINUE
  LOGVAL=RAILDRV,LOCALV,GO1,
  1 MERSI,INODE,NCPM,
  2 VOLUM,INOD,CMVIF,CMVIF,CMVIF,UNVOL,TAI SL,
  3 .TRUE.,PRES,CMVIF,
  4 RUNIF,ANSTP,
  5 NDEFV,DEFALV)
  IF (LOCAL) GO TO 300
11 (.NOT.ALFIG) GO TO 5
  A1Figs=.FALSE.
  GO TO 99999
C EDIT VOLUME.
C 220 CONTINUE
  LOGVAL=RALEDITV,LOCAL,
1  NTERSE, NCPW,  VONH, NIRM, NTO,  CNVYM, CNVIA, UNVOL,  .TRUE., PNES, VMORGN,  RNHIF, ANSFRM,  NUXIV, DELLV)
  II (LOCAL) GO TO 300
  II (.NOT. ALLIG) GO TO 5
  ALLIG=.FALSE.
  GO TO 99999
C
C WRITE VOLUME.
C
230 CONTINUE
  LOCAL-RANKP(LOCAL,  NTERSE, UNMOD, NCPW,  V, VONH, NIRM, FON, VCDT,  CNVYM, CNVIA, UNVOL,  .TRUE., PNES, VMORGN,  RNHIF, ANSFRM)
  II (LOCAL) GO TO 300
  II (.NOT. ALLIG) GO TO 5
  ALLIG=.FALSE.
  GO TO 99999
C
C DETERMINE WHICH OPERATION TO PERFORM ON THE WEIGHT VALUE AND
C BRANCH ACCORDINGLY.
C
300 CONTINUE
  GO TO (310, 320, 330), IOFLAG
C
C READ WEIGHT.
C
310 CONTINUE
  LOCAL-RADMN(W, LOCAL, WIGOT,  NTERSE, INMOD, NCPW,  WIMAM, WMTG, CNVIA, WMAJ 12, .FALSE.,  .TRUE., PNES, WMTI,  RNHIF, ANSFRM,  NOHIF, DELLV)
  II (LOCAL) GO TO 99999
  II (.NOT. ALLIG) GO TO 5
  ALLIG=.FALSE.
  GO TO 99999
C
C EDIT WEIGHT.
C
320 CONTINUE
  LOCAL-RADPMT(LOCAL,  NTERSE, NCPW,  VONH, NIRM, NTO,  CNVYM, CNVIA, UNVOL,  .TRUE., PNES, VMORGN,  RNHIF, ANSFRM,  NUXIV, DELLV)
  II (LOCAL) GO TO 300
  II (.NOT. ALLIG) GO TO 5
  ALLIG=.FALSE.
  GO TO 99999
1      NTERSE, NC:PW,
2      WI, IRAM, IRON, NIO,
3      CONVM, CONVEA, NAM:12,
4      TRU, PMES, MINCH,
5      RMFIL, ANSTRM,
6      NDELT, DIAL:NT
11 (LOCAL) GO TO 99999
11 (.NOT. ALLIG) GO TO 5
ALLIG..FAIL.
GO TO 99999
C
C WRITE WEIGHT.
C
330 CONTINUE
LOCAL-HARDPW(LOCAL,
1      NTERSE, ONOM, NC:PW,
2      WI, WEI:AM, IROM, WTCUT,
3      CONVM, CONVEA, NAM:12,
4      TRU, PMES, MINCH,
5      RMFIL, ANSTRM)
11 (LOCAL) GO TO 99999
11 (.NOT. ALLIG) GO TO 5
ALLIG..FAIL.
C
C RETURN TO CALLING PROGRAM.
C
99999 CONTINUE
RETURN
END
C---------CUBE MODULE SUBPROGRAM---------
B000010

C---------SUBPROGRAM DESCRIPTION---------
B000020

C THIS SUBPROGRAM IS A PROGRAM THAT PERFORMS A SPECIFIC FUNCTION. IT IS PART OF A SYSTEM THAT MANAGES DATA.
B000030

C THE SUBPROGRAM INITIALIZES VARIABLES IN THE LABELLED COMMON BLOCKS OF THE PROGRAM.
B000040

C THE FUNCTION OF THIS MODULE IMPLEMENTS THE LOGIC FOR INITIALIZING THE VARIABLES.
B000050

C EACH LABELLED COMMON AND ALL RELATED STATEMENTS AND DEFINITIONS ARE AS INDICATED IN THE LABELLED COMMON BLOCKS.
B000060

C LISTED UNDER THE SUBPROGRAM NAME WHERE THE LABELLED COMMON FIRST.
B000070

C APPEARS.
B000080

C---------PROGRAMMERS AND REPORT---------
B000090

C PROGRAM VERSION: 1
B000100

C PROGRAM DATE: JUNE 1981
B000110

C PROGRAMMERS: C. CHRYSOSTOMIDIS AND R. CLOUINO
B000120

C REPORT: IDEAS ON HOW TO WRITE DEX MODULES
B000130

C: VOLUME AND WEIGHT OF A S.W. PARALLELEPIPED
B000140

C: CUB Module ROUTINES
B000150

C REPORT NUMBER: 81-1
B000160

C AUTORS: C. CHRYSOSTOMIDIS AND R. CLOUINO
B000170

C PUBLISHER: MASSACHUSSETTS INSTITUTE OF TECHNOLOGY
B000180

C DEPARTMENT OF OCEAN ENGINEERING
B000190

C DESIGN LABORATORY
B000200

C CAMBRIDGE MASS 02139, JUNE 1981
B000210

C---------SUBROUTINE MAINING---------
B000220

COMMON/RMK/INHM1, RNHM1
B000230

COMMON/INHM1/INHM, OMODE
B000240

COMMON/DIA/ENGINE
B000250

COMMON/DIA/INHM1, RNHM1
B000260

INTEGER MKHM1, RNHM1
B000270

INTEGER NDHM1, RNHM1
B000280

LOGICAL WENTS
B000290

DATA RNHM1/3/
B000300

DATA MDHM1/3/
B000310

DATA OMODE/3/
B000320

DATA WENTS/14.
B000330

**** START OF SITE DEPENDENT CODE
B000340

DATA NCPW/N.
B000350

**** END OF SITE DEPENDENT CODE
B000360

COMMON /LUNITS/ PSTLUN, UDULUN
B000370

COMMON /LUNITS/ DBLUN, DBLUNC, LUNHM, DEFLUN
B000380

INTEGER PSTLUN, UDULUN
B000390

INTEGER DBLUNN(2), DBLUNC(16), LUNHM(2), DEFLUN
B000400

DATA PSTLUN/1/
B000410

DATA UDULUN/1/
B000420

DATA DBLUN/0.
B000430

DATA DBLUNC/0.
B000440

DATA LUNHM/0.
B000450

**** END OF SITE DEPENDENT CODE
B000460

DATA DEFLUN/0.
B000470

**** END OF SITE DEPENDENT CODE
B000480

**** END OF SITE DEPENDENT CODE
B000490
INTEGER PSIPUN, UIOTPU
INTEGER DTPUNH(16), TPURH(2), DEF1PU
DATA PSIPUN/1/
DATA UIOTPU/1/
DATA DTPUNH/1/, UIOTPU/1/
DATA DTPUNH/1/, UIOTPU/1/
DATA DTPUNH/1/, UIOTPU/1/
1 4H M/E, 4H SEDE, 4H DIP, 4H U/H, 4H OK1, 4H U/H,
2 4H 1, 4H U/H, 4H OK1, 4H U/H,
3 4H M/E, 4H DIP, 4H U/H,
DATA TPURH/4H(10), 4H/)
DATA DEF1PU/1/
C.................................. SUBROUTINE DIMENS..............................
COMMON / LI1NO/ L, LENDAH, LERG, DEF1AH
COMMON / W1NO/ W, W1D1AH, W1OR1AH, DEF1AH
COMMON / H11NO/ H(4), HEN1AH, W1OR1AH, NDEFH, DEF1AH(4)
COMMON / I11NO/ LLF1M, LEFH, H1N(4)
COMMON / IM1N1/ L1N(2), W1D1AH, HEN1AH(2)
INTEGER LENDAH(16), W1R1AH(16), W1OR1AH(16)
INTEGER L1N(2), LEFH, H1N(4), NDEFH
REAL L, W, DEF1AH, DEF1AH
DATA L, W, H(1), H(2), H(3), H(4) / 1.0, 1.0, 1.0, 1.0, 2.0, 3.0, 4.0/
DATA L1N/H1AH, ENG, 4HT1/
DATA W1R1AH/H1AH, ENG, 4H1F, CU, M/E, (.4H????, .4H????, .4H????, .4H)
1 4H 1, 4H, 4H, 4H, 4H, 4H
2 4H, 4H, 4H /
DATA W1D1AH, M/E, 4H1/
DATA W1R1AH/H1AH, ENG, 4H1F, CU, M/E, (.4H????, .4H????, .4H????),
1 4H 1, 4H, 4H, 4H, 4H, 4H
2 4H, 4H, 4H /
DATA W1D1AH, M/E, 4H1/
DATA W1R1AH/H1AH, ENG, 4H1F, CU, M/E, (.4H????, .4H????, .4H????),
1 4H 1, 4H, 4H, 4H, 4H, 4H
2 4H, 4H, 4H /
DATA NDEFH/4/
DATA DEF1AH, DEF1AH, DEF1AH(1), DEF1AH(2), DEF1AH(3), DEF1AH(4)
1 /
DATA LFR1M/H1M(1), H1M(2), H1M(3), H1M(4) /
DATA LFR1M/H1M(1), H1M(2), H1M(3), H1M(4) /
C.................................. SUBROUTINE VANDMN..............................
COMMON / V1NO/ V(4), W1NAM, W1OR1AH, NDEFH, DEF1AH(4)
COMMON / H11NO/ H(4), W1NAM, W1OR1AH, NDEFH, DEF1AH(4)
COMMON / ANSFRM/ ANSFRM
INTEGER V1NAM(2), W1NAM(2)
INTEGER W1R1AH(16), W1OR1AH(16)
INTEGER NDEFH, NDEFH
INTEGER ANSFRM(3)
REAL V, W, DEF1AH, DEF1AH
DATA V(1), V(2), V(3), V(4) / 1.0, 2.0, 3.0, 4.0/
DATA W(1), W(2), W(3), W(4) /1023.2, 2046.4, 3069.6, 4092.8/
DATA W(1), W(2), W(3), W(4) /1023.2, 2046.4, 3069.6, 4092.8/
APPENDIX B

UNIT SUBROUTINE ABBREVIATIONS AND CALLING SEQUENCES
Table B-1. Angle Unit Abbreviations

<table>
<thead>
<tr>
<th>NAMA03</th>
<th>NAMA06</th>
<th>NAMA08</th>
<th>NAMA12</th>
</tr>
</thead>
<tbody>
<tr>
<td>CYC</td>
<td>CYCLE</td>
<td>CYCLE</td>
<td>CYCLE</td>
</tr>
<tr>
<td>RAD</td>
<td>RADIAN</td>
<td>RADIAN</td>
<td>RADIAN</td>
</tr>
<tr>
<td>DEG</td>
<td>DEGREE</td>
<td>DEG (ANG)</td>
<td>DEGREE (ANG)</td>
</tr>
<tr>
<td>MIN</td>
<td>MINUTE</td>
<td>MIN (ANG)</td>
<td>MINUTE (ANG)</td>
</tr>
<tr>
<td>SEC</td>
<td>SECOND</td>
<td>SEC (ANG)</td>
<td>SECOND (ANG)</td>
</tr>
</tbody>
</table>

Table B-2. Force Unit Abbreviations

<table>
<thead>
<tr>
<th>NAMF02</th>
<th>NAMF03</th>
<th>NAMF12</th>
</tr>
</thead>
<tbody>
<tr>
<td>PL</td>
<td>PDL</td>
<td>POUNDAL</td>
</tr>
<tr>
<td>LB</td>
<td>LBF</td>
<td>POUND (FORCE)</td>
</tr>
<tr>
<td>ST</td>
<td>ST</td>
<td>SHORT TON</td>
</tr>
<tr>
<td>LT</td>
<td>LT</td>
<td>LONG TON</td>
</tr>
<tr>
<td>DY</td>
<td>DYN</td>
<td>DYNE</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>NEWTON</td>
</tr>
<tr>
<td>KP</td>
<td>KGF</td>
<td>KILOPOND</td>
</tr>
</tbody>
</table>

Table B-3. Length Unit Abbreviations

<table>
<thead>
<tr>
<th>NAML02</th>
<th>NAML06</th>
<th>NAML12</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>INCH</td>
<td>INCH</td>
</tr>
<tr>
<td>FT</td>
<td>FOOT</td>
<td>FOOT</td>
</tr>
<tr>
<td>SM</td>
<td>STATMI</td>
<td>STATUTE MILE</td>
</tr>
<tr>
<td>NM</td>
<td>NAUTMI</td>
<td>NAUT. MILE</td>
</tr>
<tr>
<td>MM</td>
<td>MILLIM</td>
<td>MILLIMETER</td>
</tr>
<tr>
<td>CM</td>
<td>CENTIM</td>
<td>CENTIMETER</td>
</tr>
<tr>
<td>MT</td>
<td>METER</td>
<td>METER</td>
</tr>
<tr>
<td>KM</td>
<td>KILOMT</td>
<td>KILOMETER</td>
</tr>
</tbody>
</table>
Table B-4. Temperature Unit Abbreviations

<table>
<thead>
<tr>
<th>NAMTP1</th>
<th>NAMTP5</th>
<th>NMTPL2</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>DEG-C</td>
<td>DEGREES-C</td>
</tr>
<tr>
<td>F</td>
<td>DEG-F</td>
<td>DEGREES-F</td>
</tr>
<tr>
<td>K</td>
<td>DEG-K</td>
<td>DEGREES-K</td>
</tr>
<tr>
<td>R</td>
<td>DEG-R</td>
<td>DEGREES-R</td>
</tr>
</tbody>
</table>

Table B-5. Time Unit Abbreviations

<table>
<thead>
<tr>
<th>NMT02</th>
<th>NMT03</th>
<th>NMT06</th>
<th>NMT12</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC</td>
<td>SEC</td>
<td>SECOND</td>
<td>SECOND</td>
</tr>
<tr>
<td>MN</td>
<td>MIN</td>
<td>MINUTE</td>
<td>MINUTE</td>
</tr>
<tr>
<td>HR</td>
<td>HR</td>
<td>HOUR</td>
<td>HOUR</td>
</tr>
<tr>
<td>DY</td>
<td>DAY</td>
<td>DAY</td>
<td>DAY</td>
</tr>
<tr>
<td>WK</td>
<td>WK</td>
<td>WEEK</td>
<td>WEEK</td>
</tr>
<tr>
<td>MO</td>
<td>MO</td>
<td>MONTH</td>
<td>MONTH</td>
</tr>
<tr>
<td>YR</td>
<td>YR</td>
<td>YEAR</td>
<td>YEAR</td>
</tr>
</tbody>
</table>
Table B-6. Calling Sequences of Derived Units

LOGICAL FUNCTION UAACC(UFAACC, UNAACC, ALLFLG,
CONVA, CONVT, NAMA03, NAMT03, NCPW)

LOGICAL FUNCTION UACCEL(UFACC, UNACC, ALLFLG,
CONVL, CONVT, NAML06, NAMT02, NCPW)

LOGICAL FUNCTION UAREA(UFAREA, UNAREA, ALLFLG,
CONVL, NAML06, NCPW)

LOGICAL FUNCTION UFREQ(UFFREQ, UFREQ, ALLFLG,
CONVA, CONVT, NAMA08, NAMT03,
UIOAUN, UIOTUN, NCPW)

LOGICAL FUNCTION UKVISC(UFKVIS, UNKVIS, ALLFLG,
CONVL, CONVT, NAML02, NAMT03,
UIOLUN, UIOTUN, NCPW)

LOGICAL FUNCTION UMASS(UFMASS, UNMASS, ALLFLG,
CONVF, CONVL, CONVT, NAMF02, NAML02,
NAMT02, UIOFUN, UIOLUN, UIOTUN, NCPW)

LOGICAL FUNCTION UMPower(UFPOWE, UNPOWE, ALLFLG,
CONVF, CONVL, CONVT, NAMF02, NAML02,
NAMT02, UIOFUN, UIOLUN, UIOTUN, NCPW)

LOGICAL FUNCTION UPRESS(UFPPRES, UNPPRES, ALLFLG,
CONVF, CONVL, NAMF03, NAML02, NCPW)

LOGICAL FUNCTION UPSPEC(UFSPSpe, UNPSPE, ALLFLG,
CONVL, CONVT, NAML02, NAMT03, NCPW)

LOGICAL FUNCTION URHO(UFRHO, UNRHO, ALLFLG,
CONVF, CONVL, CONVT, NAMF03, NAML02,
NAMT02, UIOFUN, UIOLUN, UIOTUN, NCPW)

LOGICAL FUNCTION USPEED(UFSSpe, UNSPE, ALLFLG,
CONVL, CONVT, NAML06, NAMT02,
UIOLUN, UIOTUN, NCPW)

LOGICAL FUNCTION UVOL(UFVOL, UNVOL, ALLFLG,
CONVL, NAML06, NCPW)

(204
APPENDIX C

SAMPLE GENERAL DATABASE

Note: This is an edited version of the listing of the database obtained at the terminal. The actual listing of the items is in a random order due to the hashing function employed during the storing of the entries. Group headings also would not appear at the terminal.
<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>VALUE</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA</td>
<td>(R)</td>
<td>0.437000E+03</td>
<td>LENGTH OVERALL (FOOT)</td>
</tr>
<tr>
<td>LBP</td>
<td>(R)</td>
<td>0.420000E+03</td>
<td>LENGTH BETWEEN PERPENDICULARS (FOOT)</td>
</tr>
<tr>
<td>BEAMA</td>
<td>(R)</td>
<td>0.450000E+02</td>
<td>MOLDED BEAM AT DESIGN WATERLINE (FOOT)</td>
</tr>
<tr>
<td>BEAMAX</td>
<td>(R)</td>
<td>0.470000E+02</td>
<td>MAXIMUM BEAM (FOOT)</td>
</tr>
<tr>
<td>T</td>
<td>(R)</td>
<td>0.150000E+02</td>
<td>MOLDED DRAFT TO KEL (FOOT)</td>
</tr>
<tr>
<td>CP</td>
<td>(R)</td>
<td>(UNDEFINED)</td>
<td>PRISOMATIC COEFFICIENT</td>
</tr>
<tr>
<td>CX</td>
<td>(R)</td>
<td>(UNDEFINED)</td>
<td>MIDSHIP COEFFICIENT</td>
</tr>
<tr>
<td>CB</td>
<td>(R)</td>
<td>0.530000E+00</td>
<td>BLOCK COEFFICIENT</td>
</tr>
<tr>
<td>CPW</td>
<td>(R)</td>
<td>0.765000E+00</td>
<td>WATERPLANE COEFFICIENT</td>
</tr>
<tr>
<td>LCB</td>
<td>(R)</td>
<td>(UNDEFINED)</td>
<td>LONGITUDINAL CENTER OF BUOYANCY AS FRACTION OF LBP AFT FP</td>
</tr>
<tr>
<td>LCF</td>
<td>(R)</td>
<td>(UNDEFINED)</td>
<td>LONGITUDINAL CENTER OF FLotation AS FRACTION OF LBP AFT FP</td>
</tr>
<tr>
<td>DFHCN</td>
<td>(R)</td>
<td>0.260000E+02</td>
<td>DEPTH AMIDSHIPS AT CENTERLINE (FOOT)</td>
</tr>
<tr>
<td>DRASHS</td>
<td>(R)</td>
<td>0.240000E+02</td>
<td>DRAFT OF SONAR DOME (FOOT)</td>
</tr>
<tr>
<td>DISPL</td>
<td>(R)</td>
<td>(UNDEFINED)</td>
<td>MOLDED DISPLACEMENT (TON)</td>
</tr>
<tr>
<td>DSN</td>
<td>(R)</td>
<td>0.450000E+04</td>
<td>TOTAL DESIGN DISPLACEMENT INCL. SHELL AND APP. (LONG TON)</td>
</tr>
<tr>
<td>MINIAT</td>
<td>(R)</td>
<td>(UNDEFINED)</td>
<td>WETTED SURFACE (FOOT **2)</td>
</tr>
<tr>
<td>FOCUSL</td>
<td>(R)</td>
<td>0.0</td>
<td>RAISED FORCASTLE (0-NO 1-YES)</td>
</tr>
</tbody>
</table>

**PROPELLATION AND POWERING**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>VALUE</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPP</td>
<td>(1)</td>
<td>2</td>
<td>TYPE OF PROPELLATION PLANT (RED)</td>
</tr>
<tr>
<td>SHP</td>
<td>(R)</td>
<td>0.700000E+05</td>
<td>TOTAL INSTALLED SHIP HORSEPOWER (HP)</td>
</tr>
<tr>
<td>NSHAFT</td>
<td>(1)</td>
<td>2</td>
<td>NUMBER OF PROPELLER SHAFTS</td>
</tr>
<tr>
<td>NE</td>
<td>(1)</td>
<td>2</td>
<td>NUMBER OF ENGINES</td>
</tr>
<tr>
<td>NO</td>
<td>(1)</td>
<td>4</td>
<td>NUMBER OF BOILERS</td>
</tr>
<tr>
<td>VSUS</td>
<td>(R)</td>
<td>0.300000E+02</td>
<td>MAXIMUM CONTINUOUS SUSTAINED SPEED (KNOT)</td>
</tr>
<tr>
<td>VEBR</td>
<td>(R)</td>
<td>0.140000E+02</td>
<td>ENDURANCE SPEED (KNOT)</td>
</tr>
<tr>
<td>ENDHR</td>
<td>(R)</td>
<td>0.600000E+02</td>
<td>ENDURANCE RANGE (NAIL, MILE)</td>
</tr>
<tr>
<td>PPI</td>
<td>(1)</td>
<td>1</td>
<td>TYPE OF PROPELLER (1-IP 2-CRP)</td>
</tr>
<tr>
<td>DPR</td>
<td>(R)</td>
<td>(UNDEFINED)</td>
<td>PROPELLER DIAMETER (FOOT)</td>
</tr>
<tr>
<td>RPM</td>
<td>(R)</td>
<td>(UNDEFINED)</td>
<td>PROPELLER RPM AT FULL POWER (RPM)</td>
</tr>
<tr>
<td>SSPP</td>
<td>(1)</td>
<td>1</td>
<td>TYPE OF PRIMARY SHIP ELECTRICAL PLANT (RED)</td>
</tr>
<tr>
<td>NSSG</td>
<td>(1)</td>
<td>4</td>
<td>NUMBER OF PRIMARY SHIP SERVICE GENERATORS</td>
</tr>
<tr>
<td>RSSGR</td>
<td>(R)</td>
<td>0.200000E+04</td>
<td>INSTALLED PRIMARY SHIP SERVICE GENERATOR CAPACITY (KW)</td>
</tr>
<tr>
<td>LHEMC</td>
<td>(A)</td>
<td>ARRAY(993)</td>
<td>TYPE OF SECONDARY OR EMERGENCY ELECTRICAL PLANT (RED)</td>
</tr>
<tr>
<td>LEMG</td>
<td>(A)</td>
<td>ARRAY(934)</td>
<td>NUMBER OF SECONDARY OR EMERGENCY GENERATORS OF EACH TYPE</td>
</tr>
<tr>
<td>LEMER</td>
<td>(R)</td>
<td>0.200000E+03</td>
<td>INSTALLED EMERGENCY OR SECONDARY GENERATOR CAPACITY (KW)</td>
</tr>
<tr>
<td>KFPP</td>
<td>(A)</td>
<td>ARRAY(915)</td>
<td>CAPACITY PER SECONDARY OR EMERGENCY GENERATORS (KW)</td>
</tr>
</tbody>
</table>
### Transverse and Directional Stability

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GM</td>
<td>0.550000E+01</td>
</tr>
<tr>
<td>FSUCFDR</td>
<td><em>UNDEFINED</em></td>
</tr>
<tr>
<td>CI</td>
<td><em>UNDEFINED</em></td>
</tr>
<tr>
<td>FINSTBL</td>
<td><strong>0</strong></td>
</tr>
<tr>
<td>NRUDDEN</td>
<td>2</td>
</tr>
</tbody>
</table>

### Weapons Payload

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPGUNS</td>
<td><strong>ARRAY[679]</strong></td>
</tr>
<tr>
<td>NMGUNS</td>
<td><strong>ARRAY[859]</strong></td>
</tr>
<tr>
<td>TYPMSL</td>
<td>131</td>
</tr>
<tr>
<td>NMPML</td>
<td>1</td>
</tr>
<tr>
<td>TYPCLNS</td>
<td>0</td>
</tr>
<tr>
<td>NMCWS</td>
<td>0</td>
</tr>
<tr>
<td>TYPPDMS</td>
<td>0</td>
</tr>
<tr>
<td>NMPDMS</td>
<td>0</td>
</tr>
<tr>
<td>TYPTORPL</td>
<td>180</td>
</tr>
<tr>
<td>NMTORPL</td>
<td>2</td>
</tr>
<tr>
<td>TYPASWL</td>
<td>148</td>
</tr>
<tr>
<td>NASWL</td>
<td>1</td>
</tr>
</tbody>
</table>

### Electronics, Fire Control and Sensors

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPSONAR</td>
<td><strong>ARRAY[629]</strong></td>
</tr>
<tr>
<td>TYPSONM</td>
<td>62</td>
</tr>
<tr>
<td>TYPSUAD</td>
<td>9</td>
</tr>
<tr>
<td>TYP3DAIR</td>
<td>15</td>
</tr>
<tr>
<td>TYP2DAIR</td>
<td>16</td>
</tr>
<tr>
<td>TYPGRAD</td>
<td><em>UNDEFINED</em></td>
</tr>
<tr>
<td>NGRAD</td>
<td>1</td>
</tr>
<tr>
<td>TYPFRAD</td>
<td>300</td>
</tr>
<tr>
<td>NMSRAD</td>
<td>2</td>
</tr>
<tr>
<td>TYPFCSU</td>
<td><em>UNDEFINED</em></td>
</tr>
<tr>
<td>TYPFCSN</td>
<td>110</td>
</tr>
<tr>
<td>TYPASFC</td>
<td>184</td>
</tr>
<tr>
<td>TYPIDS</td>
<td>0</td>
</tr>
</tbody>
</table>

### Aviation Capability

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPHEL</td>
<td>0</td>
</tr>
<tr>
<td>NHHEL</td>
<td>0</td>
</tr>
<tr>
<td>NHFR</td>
<td>0</td>
</tr>
</tbody>
</table>
COMPLEMENT

NOFF (1) 20 NUMBER OF SHIP'S OFFICERS
NCPO (1) 17 NUMBER OF CHIEF PETTY OFFICERS IN SHIP'S CREW
NCMCWC (1) 302 NUMBER OF ENLISTED IN SHIP'S CREW
MILAGOFF (1) 4 NUMBER OF OFFICERS ON FLAG STAFF
MILNSTF (1) 11 NUMBER OF ENLISTED ON FLAG STAFF
MINTROOPS (1) 0 NUMBER OF TROOPS

GENERAL MASS PROPERTIES

TYPMAIL (A) ARRAY(1) TYPE OF MATERIAL FOR HULL AND SUPERSTRUCTURE RESPECTIVELY (RED)
WEIGHT1 (A) ARRAY( ) WEIGHTS OF WEIGHT GROUPS 1-7 RESPECTIVELY (LONG TON)
VCGC17 (A) ARRAY( ) VERTICAL CENTERS OF GRAVITY OF MT. GROUPS 1-7 RESPECTIVELY (FOOT)
WLOADS (R) 0.1252E+04 WEIGHT OF GROUP B LOADS (LONG TON)
VCGLOADS (R) 0.99000E+01 VERTICAL CENTER OF GRAVITY OF GROUP B LOADS (FOOT)

THE ARRAYS

$NAME: TYPMAIL, LENGTH= 2 (1)
$ 0.10000E+01 0.20000E+01
$NAME: TYPSONAR, LENGTH= 2 (628)
$ 0.49000E+02 0.0
$NAME: MNGUNS, LENGTH= 3 (859)
$ 0.10000E+01 0.10000E+01 0.0
$NAME: TYPGUNS, LENGTH= 3 (879)
$ 0.93000E+02 0.94000E+02 0.0
$NAME: WMPENG, LENGTH= 2 (915)
$ 0.10000E+03 0.0
$NAME: MENG, LENGTH= 2 (914)
$ 0.70000E+01 0.0
$NAME: ENRTYP, LENGTH= 2 (953)
$ 0.40000E+01 0.0
APPENDIX D

GENERAL DATABASE ENTRY CODES
### TABLE D1

**PAYLOAD SHOPPING LIST**

<table>
<thead>
<tr>
<th>ITEM</th>
<th>DEFINITION</th>
<th>ITEM</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Radio Communications</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>FF (Non ASW Command and Cont)</td>
<td>26</td>
<td>SPS 53</td>
</tr>
<tr>
<td>2</td>
<td>FF (ASW Command and Control)</td>
<td>27</td>
<td>SPS 55 W/IFF (r)</td>
</tr>
<tr>
<td>3</td>
<td>DDG</td>
<td>28</td>
<td>SPS 48 E (n)</td>
</tr>
<tr>
<td>4</td>
<td>Baseline Strike Cruiser</td>
<td>29</td>
<td>SPS 58</td>
</tr>
<tr>
<td>5</td>
<td>CG/CGN</td>
<td>30</td>
<td>SPN 6</td>
</tr>
<tr>
<td>6</td>
<td>DD 963</td>
<td>31</td>
<td>SPN 10</td>
</tr>
<tr>
<td></td>
<td><strong>Radars</strong></td>
<td>32</td>
<td>SPN 12</td>
</tr>
<tr>
<td>7</td>
<td>Target Acquisition System Mod 0</td>
<td>33</td>
<td>SPN 35</td>
</tr>
<tr>
<td>8</td>
<td>Target Acquisition System Mod 3</td>
<td>34</td>
<td>SPN 42</td>
</tr>
<tr>
<td>9</td>
<td>SPS 10</td>
<td>35</td>
<td>Beacon Video Processor</td>
</tr>
<tr>
<td>10</td>
<td>SPS 10 W/IFF</td>
<td>36</td>
<td>SPH 2 (RAVIR Mod 3)</td>
</tr>
<tr>
<td>11</td>
<td>SPS 26</td>
<td>37</td>
<td>SPS 58C</td>
</tr>
<tr>
<td>12</td>
<td>SPS 29</td>
<td>38</td>
<td>SPS 55 (n)</td>
</tr>
<tr>
<td>13</td>
<td>SPS 30</td>
<td>39</td>
<td>IFF (n)</td>
</tr>
<tr>
<td>14</td>
<td>SPS 37</td>
<td>40</td>
<td>SQS 53 (ON 963)</td>
</tr>
<tr>
<td>15</td>
<td>SPS 39A</td>
<td>41</td>
<td>SQS 56</td>
</tr>
<tr>
<td>16</td>
<td>SPS 40</td>
<td>42</td>
<td>SQS 57 (TAR)</td>
</tr>
<tr>
<td>17</td>
<td>SPS 40 W/IFF</td>
<td>43</td>
<td>SQQ 23</td>
</tr>
<tr>
<td>18</td>
<td>SPS 40A W/IFF</td>
<td>44</td>
<td>SQS 26 (.72)</td>
</tr>
<tr>
<td>19</td>
<td>SPS 43</td>
<td>45</td>
<td>EDD 610E</td>
</tr>
<tr>
<td>20</td>
<td>SPS 43 W/IFF</td>
<td>46</td>
<td>SQA 13 (VDS)</td>
</tr>
<tr>
<td>21</td>
<td>SPS 48 (V)</td>
<td>47</td>
<td>SQS 35 (IVDS)</td>
</tr>
<tr>
<td>22</td>
<td>SPS 48 C</td>
<td>48</td>
<td>UQN 4</td>
</tr>
<tr>
<td>23</td>
<td>SPS 49</td>
<td>49</td>
<td>SQS 23</td>
</tr>
<tr>
<td>24</td>
<td>SPS 49 W/IFF</td>
<td>50</td>
<td>TACTLASS</td>
</tr>
<tr>
<td>25</td>
<td>SPS 52B</td>
<td>51</td>
<td>Escort Towed Array System (ETAS)</td>
</tr>
<tr>
<td></td>
<td><strong>Sonars</strong></td>
<td>52</td>
<td>SQS 38 (keel)</td>
</tr>
</tbody>
</table>

### TABLE D1 (Continued)

<table>
<thead>
<tr>
<th>ITEM</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>53</td>
<td>505 Sonar</td>
</tr>
<tr>
<td>54</td>
<td>VDS</td>
</tr>
<tr>
<td>55</td>
<td>SQR-19 Towed Array (n)</td>
</tr>
</tbody>
</table>

**Sonar Domes**

<table>
<thead>
<tr>
<th>ITEM</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>56</td>
<td>23 Keel Dome (Double, Rubber)</td>
</tr>
<tr>
<td>57</td>
<td>23 Bow Dome</td>
</tr>
<tr>
<td>58</td>
<td>26/53 Bow Dome (r)</td>
</tr>
<tr>
<td>59</td>
<td>610E/505 Bow Dome</td>
</tr>
<tr>
<td>60</td>
<td>23 Keel Dome</td>
</tr>
<tr>
<td>61</td>
<td>610E/505 Keel Dome</td>
</tr>
<tr>
<td>62</td>
<td>23 Keel Dome (Rubber)</td>
</tr>
<tr>
<td>63</td>
<td>Blank</td>
</tr>
</tbody>
</table>

**Sonar Liquid**

<table>
<thead>
<tr>
<th>ITEM</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>Sonar Liquid (1 ton)</td>
</tr>
</tbody>
</table>

**Electronic Countermeasure (ECM)**

<table>
<thead>
<tr>
<th>ITEM</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>65</td>
<td>FF/PFG Basic</td>
</tr>
<tr>
<td>66</td>
<td>DD/DDG/CG/CGN Basic (ECM &amp; IR Decoy) (r)</td>
</tr>
<tr>
<td>67</td>
<td>SLQ 32 (V2) (n)</td>
</tr>
<tr>
<td>68</td>
<td>ALR-59</td>
</tr>
<tr>
<td>69</td>
<td>WLR-8</td>
</tr>
<tr>
<td>70</td>
<td>SLR-10 or BRD-4, SLR-11 (r)</td>
</tr>
<tr>
<td>71</td>
<td>DTP Suite Three (n)</td>
</tr>
<tr>
<td>72</td>
<td>Nixie</td>
</tr>
</tbody>
</table>

**Electronic Tactical Data Systems**

<table>
<thead>
<tr>
<th>ITEM</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>73</td>
<td>Conc C and C-FF</td>
</tr>
<tr>
<td>74</td>
<td>Conv C and C-DD</td>
</tr>
<tr>
<td>75</td>
<td>ASW C and C-FF-2C, 7D</td>
</tr>
<tr>
<td>76</td>
<td>NTDS - 4C, 13D - DG/DGN</td>
</tr>
<tr>
<td>77</td>
<td>NTDS - 2C, 9D - DDG</td>
</tr>
<tr>
<td>78</td>
<td>NTDS - 3C, 15 - 18D</td>
</tr>
<tr>
<td>79</td>
<td>NTDS - 3C, 15D - CG</td>
</tr>
<tr>
<td>80</td>
<td>NTDS - 3C, 15D - CGN</td>
</tr>
<tr>
<td>81</td>
<td>SSCDS - 1C</td>
</tr>
<tr>
<td>82</td>
<td>Conv C and C - CG, CGN</td>
</tr>
<tr>
<td>83</td>
<td>DDG C &amp; C (n)</td>
</tr>
</tbody>
</table>

**Guns and Ammo Handling and Stowage**

<table>
<thead>
<tr>
<th>ITEM</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>84</td>
<td>3/50 SM, MK-34 w/Shield Mn DK</td>
</tr>
<tr>
<td>85</td>
<td>3/50 SM, MK-34 w/Shield 01 LV</td>
</tr>
<tr>
<td>86</td>
<td>3/50 SM, MK-34 w/o Shield 01 LV</td>
</tr>
<tr>
<td>87</td>
<td>3/50 TM, MK-33 w/Shield Mn DK</td>
</tr>
<tr>
<td>88</td>
<td>3/50 TM, MK-33 w/Shield 01 LV</td>
</tr>
<tr>
<td>89</td>
<td>3/50 TM, MK-33 w/o Shield Mn DK</td>
</tr>
<tr>
<td>90</td>
<td>3/50 TM, MK-33 w/o Shield 01 LV</td>
</tr>
<tr>
<td>91</td>
<td>5/38 SM, MK-30 Mn DK (r)</td>
</tr>
<tr>
<td>92</td>
<td>5/38 SM, MK-30 01 LV (r)</td>
</tr>
<tr>
<td>93</td>
<td>5/54 SM, MK-42 Mn DK (r)</td>
</tr>
<tr>
<td>94</td>
<td>5/54 SM, MK-42 01 LV (r)</td>
</tr>
<tr>
<td>95</td>
<td>5/54 SM, LW, MK-45/0 Mn DK (r)</td>
</tr>
<tr>
<td>96</td>
<td>8/55 SM, LW, MK-45/0 01 LV (r)</td>
</tr>
<tr>
<td>97</td>
<td>8/55 SM, MK-71 Mod X Mn DK (r)</td>
</tr>
<tr>
<td>98</td>
<td>8/55 SM, MK-71 Mod X 01 LV (r)</td>
</tr>
</tbody>
</table>
TABLE D1 (Continued)

<table>
<thead>
<tr>
<th>ITEM</th>
<th>DEFINITION</th>
<th>ITEM</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>99</td>
<td>50 Cal MG 02 LV</td>
<td>123</td>
<td>40 mm</td>
</tr>
<tr>
<td>100</td>
<td>Vulcan/Phalanx CIWS 01 LV (r)</td>
<td>124</td>
<td>20 mm</td>
</tr>
<tr>
<td>101</td>
<td>35 mm twin 01 LV</td>
<td>125</td>
<td>Vulcan/Phalanx CIWS (1 rd) (n)</td>
</tr>
<tr>
<td>102</td>
<td>76 mm single 01 LV</td>
<td>126</td>
<td>Blank</td>
</tr>
<tr>
<td>103</td>
<td>76 mm single Mm DK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>104</td>
<td>5/54, LW, MK-45, Mod 1 (n)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>105</td>
<td>5/54, LW, MK-45, Mod X (n)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Gun Fire Control Systems**

<table>
<thead>
<tr>
<th>ITEM</th>
<th>DEFINITION</th>
<th>ITEM</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>106</td>
<td>MK-86 (Surface only, no SPG 60, with seafire)</td>
<td>127</td>
<td>NATO Sea Sparrow, MK-2910 (8 boxes)</td>
</tr>
<tr>
<td>107</td>
<td>MK-37</td>
<td>128</td>
<td>VLS Deep Cell, 64 cell (61 missiles)</td>
</tr>
<tr>
<td>108</td>
<td>MK-56</td>
<td>129</td>
<td>Terr-60 MK-10/7-8 Mm DK (n)</td>
</tr>
<tr>
<td>109</td>
<td>MK-63/26</td>
<td>130</td>
<td>VLS Standard Cell, 64 cell (61 missiles)</td>
</tr>
<tr>
<td>110</td>
<td>MK-68/6</td>
<td>131</td>
<td>Tartar, MK-13</td>
</tr>
<tr>
<td>111</td>
<td>GPCS for 5&quot;/54 w/AEGIS ILL.</td>
<td>132</td>
<td>VLS Deep Cell, 32 cell (29 missiles)</td>
</tr>
<tr>
<td>112</td>
<td>MK-86 (w/seafire and SPG 60) (r)</td>
<td>133</td>
<td>VLS Standard Cell, 32 cell (29 missiles)</td>
</tr>
<tr>
<td>113</td>
<td>MK-86 (LFS) Mod 4</td>
<td>134</td>
<td>BPDMs, MK-25</td>
</tr>
<tr>
<td>114</td>
<td>MK-86 with AAW and CWI MOD 5</td>
<td>135</td>
<td>GMLS, MK-26 (MOD 0)</td>
</tr>
<tr>
<td>115</td>
<td>MK-87</td>
<td>136</td>
<td>GMLS, MK-26 (MOD 1)</td>
</tr>
<tr>
<td>116</td>
<td>MK-92 CWI/STIR (r)</td>
<td>137</td>
<td>GMLS, MK-26 (MOD 2)</td>
</tr>
<tr>
<td>117</td>
<td>MK-86 - 5 w/o SPQ 9 (r)</td>
<td>138</td>
<td>5&quot; SSR, MK-105</td>
</tr>
<tr>
<td>118</td>
<td>Seafire</td>
<td>139</td>
<td>CHAFFROC, 4 Launch</td>
</tr>
<tr>
<td></td>
<td></td>
<td>140</td>
<td>Harpoon Launcher (MK-13)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>141</td>
<td>Harpoon (Box) w/4 Missiles</td>
</tr>
<tr>
<td></td>
<td></td>
<td>142</td>
<td>IPMDs Launcher</td>
</tr>
<tr>
<td></td>
<td></td>
<td>143</td>
<td>Redeye</td>
</tr>
<tr>
<td></td>
<td></td>
<td>144</td>
<td>Sea Launched Cruise Missile (SLCM) w/4 missiles</td>
</tr>
<tr>
<td></td>
<td></td>
<td>145</td>
<td>Near Term Laser Pointer Trainer</td>
</tr>
</tbody>
</table>

**Gun Ammo**

<table>
<thead>
<tr>
<th>ITEM</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>119</td>
<td>3&quot;/50 Rd.</td>
</tr>
<tr>
<td>120</td>
<td>5&quot;/38 Rd.</td>
</tr>
<tr>
<td>121</td>
<td>5&quot;/54 Rd.</td>
</tr>
<tr>
<td>122</td>
<td>8&quot;/55 Rd.</td>
</tr>
<tr>
<td>123</td>
<td>40 mm</td>
</tr>
<tr>
<td>124</td>
<td>20 mm</td>
</tr>
<tr>
<td>125</td>
<td>Vulcan/Phalanx CIWS (1 rd) (n)</td>
</tr>
<tr>
<td>126</td>
<td>Blank</td>
</tr>
<tr>
<td>127</td>
<td>NATO Sea Sparrow, MK-2910 (8 boxes)</td>
</tr>
<tr>
<td>128</td>
<td>VLS Deep Cell, 64 cell (61 missiles)</td>
</tr>
<tr>
<td>129</td>
<td>Terr-60 MK-10/7-8 Mm DK (n)</td>
</tr>
<tr>
<td>130</td>
<td>VLS Standard Cell, 64 cell (61 missiles)</td>
</tr>
<tr>
<td>131</td>
<td>Tartar, MK-13</td>
</tr>
<tr>
<td>132</td>
<td>VLS Deep Cell, 32 cell (29 missiles)</td>
</tr>
<tr>
<td>133</td>
<td>VLS Standard Cell, 32 cell (29 missiles)</td>
</tr>
<tr>
<td>134</td>
<td>BPDMs, MK-25</td>
</tr>
<tr>
<td>135</td>
<td>GMLS, MK-26 (MOD 0)</td>
</tr>
<tr>
<td>136</td>
<td>GMLS, MK-26 (MOD 1)</td>
</tr>
<tr>
<td>137</td>
<td>GMLS, MK-26 (MOD 2)</td>
</tr>
<tr>
<td>138</td>
<td>5&quot; SSR, MK-105</td>
</tr>
<tr>
<td>139</td>
<td>CHAFFROC, 4 Launch</td>
</tr>
<tr>
<td>140</td>
<td>Harpoon Launcher (MK-13)</td>
</tr>
<tr>
<td>141</td>
<td>Harpoon (Box) w/4 Missiles</td>
</tr>
<tr>
<td>142</td>
<td>IPMDs Launcher</td>
</tr>
<tr>
<td>143</td>
<td>Redeye</td>
</tr>
<tr>
<td>144</td>
<td>Sea Launched Cruise Missile (SLCM) w/4 missiles</td>
</tr>
<tr>
<td>145</td>
<td>Near Term Laser Pointer Trainer</td>
</tr>
</tbody>
</table>
### TABLE D1 (Continued)

<table>
<thead>
<tr>
<th>ITEM</th>
<th>DEFINITION</th>
<th>ITEM</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>146</td>
<td>Tarpon Canister Launcher w/4 missiles</td>
<td>167</td>
<td>Tomahawk &amp; Cannisters for VLS (n)</td>
</tr>
<tr>
<td>147</td>
<td>Super RBOC (2 launchers)</td>
<td>168</td>
<td>Tomahawk (non VLS) (n)</td>
</tr>
<tr>
<td></td>
<td><em>(ASW Rocket and Missile Launchers)</em></td>
<td>169</td>
<td>RIM 67A-Ter 2 DK</td>
</tr>
<tr>
<td>148</td>
<td>ASROC-8, MK-16/4 w/8 ASROC, 01 level</td>
<td>170</td>
<td>SM-2, ASROC, Harpoon &amp; Cannister for VLS (n)</td>
</tr>
<tr>
<td>149</td>
<td>ASROC-8, MK-16/4 w/8 ASROC, 02 level</td>
<td>171</td>
<td>RIM 7H Sparrow</td>
</tr>
<tr>
<td>150</td>
<td>ASROC Reload, Mag &amp; Handling Gear</td>
<td>172</td>
<td>5&quot; Rocket, MK-10</td>
</tr>
<tr>
<td>151</td>
<td>DASH Launch (Hangar)</td>
<td>173</td>
<td>CHAFFROC</td>
</tr>
<tr>
<td>152</td>
<td>Blank</td>
<td>174</td>
<td>Harpoon, ASROC, SM-2 (not VLS) or Tartar (n)</td>
</tr>
<tr>
<td></td>
<td><em>(Missile Fire Control System)</em></td>
<td>175</td>
<td>Subroc (2 launcher ready service) (n)</td>
</tr>
<tr>
<td>153</td>
<td>Near Term Laser System Less Pointer Trainer</td>
<td>176</td>
<td>MK-25 TT 2 DK</td>
</tr>
<tr>
<td>154</td>
<td>Tomahawk WCS MK 2 (n)</td>
<td>177</td>
<td>MK-25 TT 01 LV</td>
</tr>
<tr>
<td>155</td>
<td>Weapons Direction System MK-13</td>
<td>178</td>
<td>MK-32 Twin Mk DK</td>
</tr>
<tr>
<td>156</td>
<td>MK-74 FCS (1 DIR) (Tartar C)</td>
<td>179</td>
<td>MK-32 Twin 01 LV</td>
</tr>
<tr>
<td>157</td>
<td>MK-74 FCS (2 DIR)</td>
<td>180</td>
<td>MK-32 Triple (r)</td>
</tr>
<tr>
<td>158</td>
<td>MK-115 BPDS FCS</td>
<td>181</td>
<td>Torpedo Countermeasure Launchers (r)</td>
</tr>
<tr>
<td>159</td>
<td>CHAFFROC FCS</td>
<td>182</td>
<td>Hedgehog</td>
</tr>
<tr>
<td>160</td>
<td>MK-99 FCS (2 chan, for SPY-1B) (n)</td>
<td>183</td>
<td>MK-32 Triple, 1 mount (in hull) (n)</td>
</tr>
<tr>
<td>161</td>
<td>MK-91 GMFCS (2 chan)</td>
<td>184</td>
<td>MK-114 UBFCs</td>
</tr>
<tr>
<td>162</td>
<td>AEGIS SPY-1 (n)</td>
<td>185</td>
<td>MK-114/12 UBFCs</td>
</tr>
<tr>
<td>163</td>
<td>AEGIS SPY-1A (n)</td>
<td>186</td>
<td>MK-116 UBFCs (r)</td>
</tr>
<tr>
<td>164</td>
<td>AEGIS SPY-1B (n)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>165</td>
<td>Harpoon FCS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>166</td>
<td>Tartar D FCS (2 chan)</td>
<td>187</td>
<td>Drone Control SRW4C w/URW 15</td>
</tr>
</tbody>
</table>

**Torpedo Tubes, Handling & Stowage**

<table>
<thead>
<tr>
<th>ITEM</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>176</td>
<td>MK-25 TT 2 DK</td>
</tr>
<tr>
<td>177</td>
<td>MK-25 TT 01 LV</td>
</tr>
<tr>
<td>178</td>
<td>MK-32 Twin Mk DK</td>
</tr>
<tr>
<td>179</td>
<td>MK-32 Twin 01 LV</td>
</tr>
<tr>
<td>180</td>
<td>MK-32 Triple</td>
</tr>
<tr>
<td>181</td>
<td>Torpedo Countermeasure Launchers</td>
</tr>
<tr>
<td>182</td>
<td>Hedgehog</td>
</tr>
<tr>
<td>183</td>
<td>MK-32 Triple, 1 mount (in hull)</td>
</tr>
</tbody>
</table>

**ASW and Torpedo Fire Control Systems**

<table>
<thead>
<tr>
<th>ITEM</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>184</td>
<td>MK-114 UBFCs</td>
</tr>
<tr>
<td>185</td>
<td>MK-114/12 UBFCs</td>
</tr>
<tr>
<td>186</td>
<td>MK-116 UBFCs</td>
</tr>
<tr>
<td>187</td>
<td>Drone Control SRW4C w/URW 15</td>
</tr>
<tr>
<td>ITEM</td>
<td>DEFINITION</td>
</tr>
<tr>
<td>------</td>
<td>------------</td>
</tr>
<tr>
<td>188</td>
<td>Drone Control SRW4C</td>
</tr>
<tr>
<td>189</td>
<td>TORP FCS (MK-25) (Hull)</td>
</tr>
<tr>
<td>190</td>
<td>TORP FCS (MK-25) (Supstr)</td>
</tr>
<tr>
<td>191</td>
<td>Tarpon FCS Interface Alterations</td>
</tr>
<tr>
<td>192</td>
<td>MK-309 UBFCS (n)</td>
</tr>
<tr>
<td></td>
<td><strong>ASW Ammo</strong></td>
</tr>
<tr>
<td>193</td>
<td>ASROC Reload (02 Level)</td>
</tr>
<tr>
<td>194</td>
<td>ASROC Reload (01 Level)</td>
</tr>
<tr>
<td>195</td>
<td>MK-37/0,3 2 DK</td>
</tr>
<tr>
<td>196</td>
<td>MK-37/0,3 01 LV</td>
</tr>
<tr>
<td>197</td>
<td>MK-37/1,2 2 DK</td>
</tr>
<tr>
<td>198</td>
<td>MK-44 Mn DK</td>
</tr>
<tr>
<td>199</td>
<td>MK-44 01 LV</td>
</tr>
<tr>
<td>200</td>
<td>MK-46 Mn DK</td>
</tr>
<tr>
<td>201</td>
<td>MK-48 Mn DK</td>
</tr>
<tr>
<td>202</td>
<td>MK-48 2 DK</td>
</tr>
<tr>
<td>203</td>
<td>MK-46 in hull (n)</td>
</tr>
<tr>
<td></td>
<td><strong>Helos</strong></td>
</tr>
<tr>
<td>204</td>
<td>Two LAMPS III (main deck)</td>
</tr>
<tr>
<td>205</td>
<td>HASP-T Helo (main deck)</td>
</tr>
<tr>
<td>206</td>
<td>One SH-3 (n)</td>
</tr>
<tr>
<td></td>
<td><strong>Aviation Ammo</strong></td>
</tr>
<tr>
<td></td>
<td><strong>AVIATION LIQUIDS</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Small Arms</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Helos Support</strong></td>
</tr>
<tr>
<td>208</td>
<td>LAMPS MK III Package</td>
</tr>
<tr>
<td>209</td>
<td>LAMPS Control</td>
</tr>
<tr>
<td>210</td>
<td>LAAV 1</td>
</tr>
<tr>
<td>ITEM</td>
<td>DEFINITION</td>
</tr>
<tr>
<td>------</td>
<td>------------</td>
</tr>
<tr>
<td>Sewage Treatment Plant</td>
<td>230 Sewage Treatment Plant (175 man)</td>
</tr>
<tr>
<td>231 Blank</td>
<td></td>
</tr>
<tr>
<td>Replenishment at Sea &amp; Cargo Handling</td>
<td>232 FAST System - Dest/Cruiser</td>
</tr>
<tr>
<td>233 Blank</td>
<td>254 LVL II Armor for one 64 cell VLS (n)</td>
</tr>
<tr>
<td>Liquids</td>
<td>234 NSFO (100 gal.)</td>
</tr>
<tr>
<td>235 Diesel Oil (100 gal.)</td>
<td>258 Frag Protection SPS-49 Room &amp; Ill. Base (n)</td>
</tr>
<tr>
<td>236 Potable Water (100 gal.)</td>
<td>237 Liquid O₂ (100 gal.)</td>
</tr>
<tr>
<td>Boats</td>
<td>240 14' Punt</td>
</tr>
<tr>
<td>243 28' Personnel</td>
<td>244 33' Utility</td>
</tr>
<tr>
<td>247 40' Utility</td>
<td>248 40' Personnel</td>
</tr>
<tr>
<td>n: new item - 6 December 1979</td>
<td>r: updated item - 6 December 1979</td>
</tr>
<tr>
<td>300+ See TABLE D-2</td>
<td></td>
</tr>
<tr>
<td>ITEM</td>
<td>DEFINITION</td>
</tr>
<tr>
<td>------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>300</td>
<td>SPG-51C</td>
</tr>
<tr>
<td>301</td>
<td>SPG-55</td>
</tr>
<tr>
<td>302</td>
<td>SPG-60</td>
</tr>
<tr>
<td>303</td>
<td>MK-51 Gun Fire Control Director</td>
</tr>
<tr>
<td>304</td>
<td>MK-63 Gun Fire Control Director</td>
</tr>
<tr>
<td>305</td>
<td>Tartar MK-11 GMLS</td>
</tr>
<tr>
<td>306</td>
<td>Tartar MK-22 GMLS</td>
</tr>
<tr>
<td>307</td>
<td>MK-11 MFCS</td>
</tr>
<tr>
<td>308</td>
<td>MK-76 MFCS</td>
</tr>
</tbody>
</table>
## Table D-3

### Index to Non-Payload Reed Code

<table>
<thead>
<tr>
<th>Item</th>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Propulsion Plant Type (PPTYP)</td>
<td>1</td>
<td>600 psi steam</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1200 psi steam</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1200 psi pressure-fired steam</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>nuclear</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>gas turbine first generation</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>gas turbine second generation</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>diesel</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>COGAS</td>
</tr>
<tr>
<td>Ship Service Electric Plant Type (SSEPTYP)</td>
<td>1</td>
<td>steam</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>gas turbine first generation</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>gas turbine second generation</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>low speed diesel</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>medium speed diesel</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>high speed diesel</td>
</tr>
<tr>
<td>Emergency Electric Plant Type (EMETYP)</td>
<td>1</td>
<td>gas turbine first generation</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>gas turbine second generation</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>low speed diesel</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>medium speed diesel</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>high speed diesel</td>
</tr>
<tr>
<td>Type of Material (TYPMATL)</td>
<td>1</td>
<td>steel</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>aluminum</td>
</tr>
</tbody>
</table>

APPENDIX E

SAMPLE WEIGHT AND VERTICAL CENTER OF GRAVITY DATABASES

Note: These are edited versions of the listings of the databases obtained at the terminal. The actual listings of the items in each database is in a random order due to the hashing function employed during the storing of the entries.
<p>| V500 | (R) | 0.47000E+01 | WT. OF HEATING SYSTEMS (LONG TON) |
| V501 | (R) | 0.47200E+02 | WT. OF VENTILATION SYSTEMS (LONG TON) |
| V502 | (R) | 0.21100E+02 | WT. OF AIR COND. SYSTEMS (LONG TON) |
| V503 | (R) | 0.26000E+01 | WT. OF REF. SPACES, PLANT AND EQUIP. (LONG TON) |
| V504 | (R) | 0.20000E+00 | WT. OF GAS, HEAT, AV. I. O., SEWAGE SYSTEMS (LONG TON) |
| V505 | (R) | 0.13700E+02 | WT. OF PLUMBING INST. (LONG TON) |
| V506 | (R) | 0.34000E+02 | WT. OF FIREMAIN, FISHING, SPRKLR, S.M. SVC SYSTEMS (LONG TON) |
| V507 | (R) | 0.90000E+01 | WT. OF FIRE EX. SYSTEMS (LONG TON) |
| V508 | (R) | 0.11300E+02 | WT. OF DRNL. BAILAST, STEEL ZK SYSTEMS (LONG TON) |
| V509 | (R) | 0.64000E+01 | WT. OF FRESH WATER SYSTEMS (LONG TON) |
| V510 | (R) | 0.15000E+01 | WT. OF SCUPPERS AND DECK DRAINS (LONG TON) |
| V511 | (R) | 0.29000E+02 | WT. OF F.O. AND D.O. FILL, VENT, STW AND XFR SYS (LONG TON) |
| V512 | (R) | 0.31000E+01 | WT. OF TANK HEATING SYSTEMS (LONG TON) |
| V513 | (R) | 0.84000E+01 | WT. OF COMPRESSED AIR SYSTEMS (LONG TON) |
| V514 | (R) | 0.68000E+02 | WT. OF AUX. STEAM, EXH. STEAM AND STEAM DRAINS (LONG TON) |
| V515 | (R) | 0.13300E+02 | WT. OF DISTILLING PLANT (LONG TON) |
| V516 | (R) | 0.11300E+02 | WT. OF STIERING SYSTEMS (LONG TON) |
| V517 | (R) | 0.26300E+02 | WT. OF HUDBKS (LONG TON) |
| V518 | (R) | 0.28000E+02 | WT. OF HOERING, WING, ANCH AND AC HNDLG, DK MCARY (LONG TON) |
| V519 | (R) | 0.92000E+01 | WT. OF LILY, STORLDS HANDLING SYSTEMS (LONG TON) |
| V520 | (R) | 0.36000E+01 | WT. OF AUXILIARY SYSTEMS REPAIR PARTS (LONG TON) |
| V521 | (R) | 0.19300E+02 | WT. OF AUXILIARY SYSTEMS OP. FLUIDS (LONG TON) |
| V522 | (R) | 0.16600E+02 | WT. OF HULL FITTINGS (LONG TON) |
| V523 | (R) | 0.22000E+02 | WT. OF BOALS, BOAT STING AND HNDLG (LONG TON) |
| V524 | (R) | 0.13000E+01 | WT. OF RIGGING AND CANVAS (LONG TON) |
| V525 | (R) | 0.22000E+02 | WT. OF LADDERS AND CHATINGS (LONG TON) |
| V526 | (R) | 0.22700E+02 | WT. OF NON-STRUCTUAL BIDS AND DOORS (LONG TON) |
| V527 | (R) | 0.63000E+02 | WT. OF RAINING (LONG TON) |
| V528 | (R) | 0.14000E+02 | WT. OF DECK COVIRING (LONG TON) |
| V529 | (R) | 0.25000E+02 | WT. OF HULL INSULATION (LONG TON) |
| V530 | (R) | 0.24600E+02 | WT. OF STOREROOMS, SUMENTS AND LOCKERS (LONG TON) |
| V531 | (R) | 0.41000E+01 | WT. OF EQUIP FOR UTILITY SPACES (LONG TON) |
| V532 | (R) | 0.10000E+02 | WT. OF EQUIP FOR WSHOPS, LAUS AND TEST ARAS (LONG TON) |
| V533 | (R) | 0.80000E+02 | WT. OF EQUIP. FOR GALLEY, KERY, SCRIRY AND COMSRY (LONG TON) |
| V534 | (R) | 0.31700E+02 | WT. OF FURNISHINGS FOR LIVING SPACES (LONG TON) |
| V535 | (R) | 0.93000E+01 | WT. OF FURNISHINGS FOR OFFICES, CONE. CTS (LONG TON) |
| V536 | (R) | 0.13000E+01 | WT. OF FURNISHINGS FOR MED. AND DEVAL SP. (LONG TON) |
| V537 | (R) | 0.18000E+01 | WT. OF OUTFIT AND FURNISHING REPAIR PARTS (LONG TON) |
| V538 | (R) | 0.23300E+03 | WT. OF GUNS AND GUNMOUNTS (LONG TON) |
| V539 | (R) | 0.76000E+01 | WT. OF AMMUNITION HANDLING SYSTEMS (LONG TON) |
| V540 | (R) | 0.62000E+01 | WT. OF AMMUNITION STORAGE (LONG TON) |
| W704 | (R) | <strong>UNDEFINED</strong> | WT. OF MISSILE LAUNCHERS (STE W/O) (LONG TON) |
| W705 | (R) | <strong>UNDEFINED</strong> | WT. OF ASW ROCKET LAUNCHERS (STE W/O) (LONG TON) |
| W750 | (R) | 0.4700E+01 | WT. OF ARMAMENT REPAIR PARTS (LONG TON) |
| W751 | (R) | 0.3900E+01 | WT. OF ARMAMENT OPERATING FLUIDS (LONG TON) |
| W800 | (R) | 0.3910E+02 | WT. OF SHIPS OFFICERS, CREW AND EFFECTS (LONG TON) |
| W803 | (R) | 0.9630E+02 | WT. OF SHIPS AMMUNITION (LONG TON) |
| W806 | (R) | 0.5890E+02 | WT. OF PROVISIONS AND PERSONNEL STORES (LONG TON) |
| W812 | (R) | 0.5270E+02 | WT. OF PORTABLE WATER (LONG TON) |
| W813 | (R) | 0.7730E+02 | WT. OF RESERVE FEEDWATER (LONG TON) |
| W814 | (R) | 0.1380E+02 | WT. OF SHIPS LUBE OIL (LONG TON) |
| W816 | (R) | 0.8705E+03 | WT. OF FUEL OIL (LONG TON) |
| W817 | (R) | 0.4150E+02 | WT. OF DIESEL OIL (LONG TON) |</p>
<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>VALUE</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCG110</td>
<td>(R)</td>
<td>0.1050E+02</td>
<td>VCG OF SHELL PLATING (FOOT)</td>
</tr>
<tr>
<td>VCG111</td>
<td>(R)</td>
<td>0.1070E+02</td>
<td>VCG OF LUNG. AND TRANS. FRAMING (FOOT)</td>
</tr>
<tr>
<td>VCG112</td>
<td>(R)</td>
<td>0.1670E+02</td>
<td>VCG OF PLATFORMS AND FLATIS (FOOT)</td>
</tr>
<tr>
<td>VCG113</td>
<td>(R)</td>
<td>0.2720E+02</td>
<td>VCG OF ALL DECKS (FOOT)</td>
</tr>
<tr>
<td>VCG114</td>
<td>(R)</td>
<td>0.3810E+02</td>
<td>VCG OF SUPERSTRUCTURE (FOOT)</td>
</tr>
<tr>
<td>VCG115</td>
<td>(R)</td>
<td>0.5300E+01</td>
<td>VCG OF PROPULSION FOUNDATIONS (FOOT)</td>
</tr>
<tr>
<td>VCG116</td>
<td>(R)</td>
<td>0.5700E+02</td>
<td>VCG OF IDN. FOR AUX. AND OTHER EQUIP (FOOT)</td>
</tr>
<tr>
<td>VCG117</td>
<td>(R)</td>
<td>0.1450E+02</td>
<td>VCG OF STRUCTURAL BUIDS (FOOT)</td>
</tr>
<tr>
<td>VCG118</td>
<td>(R)</td>
<td>0.1570E+02</td>
<td>VCG OF TRUNKS AND ENCLOSURES (FOOT)</td>
</tr>
<tr>
<td>VCG119</td>
<td>(R)</td>
<td>0.1150E+02</td>
<td>VCG OF CASTINGS AND FORGINGS (FOOT)</td>
</tr>
<tr>
<td>VCG201</td>
<td>(R)</td>
<td>0.3000E+01</td>
<td>VCG OF SEA CHES (FOOT)</td>
</tr>
<tr>
<td>VCG202</td>
<td>(R)</td>
<td>0.2550E+02</td>
<td>VCG OF DOORS, HATCHES, SCUTTLES (FOOT)</td>
</tr>
<tr>
<td>VCG203</td>
<td>(R)</td>
<td>0.8200E+02</td>
<td>VCG OF MASON AND KINGPOSTS (FOOT)</td>
</tr>
<tr>
<td>VCG204</td>
<td>(R)</td>
<td>0.8000E+02</td>
<td>VCG OF SUNKER DOMES (FOOT)</td>
</tr>
<tr>
<td>VCG250</td>
<td>(R)</td>
<td>0.1760E+02</td>
<td>VCG OF WELDING, RIVETING AND FASTENINGS (FOOT)</td>
</tr>
<tr>
<td>VCG251</td>
<td>(R)</td>
<td>0.1330E+02</td>
<td>VCG OF BOILERS AND ENERGY CONVERTERS (FOOT)</td>
</tr>
<tr>
<td>VCG252</td>
<td>(R)</td>
<td>0.1330E+02</td>
<td>VCG OF PROPULSION UNITS (FOOT)</td>
</tr>
<tr>
<td>VCG253</td>
<td>(R)</td>
<td>0.1330E+02</td>
<td>VCG OF MAIN CONDENSERS AND AIR EJCTORS (FOOT)</td>
</tr>
<tr>
<td>VCG254</td>
<td>(R)</td>
<td>0.5400E+01</td>
<td>VCG OF ENGINEERING AIR SUPPLY (FOOT)</td>
</tr>
<tr>
<td>VCG255</td>
<td>(R)</td>
<td>0.5600E+02</td>
<td>VCG OF UPTAKES AND SMOKEPipes (FOOT)</td>
</tr>
<tr>
<td>VCG256</td>
<td>(R)</td>
<td>0.1600E+02</td>
<td>VCG OF PROPULSION CONTROL EQUIPMENT (FOOT)</td>
</tr>
<tr>
<td>VCG301</td>
<td>(R)</td>
<td>0.2040E+02</td>
<td>VCG OF MAIN STEAM SYSTEM (FOOT)</td>
</tr>
<tr>
<td>VCG302</td>
<td>(R)</td>
<td>0.1660E+02</td>
<td>VCG OF FEEDWATER AND CONDENSATE SYSTEMS (FOOT)</td>
</tr>
<tr>
<td>VCG303</td>
<td>(R)</td>
<td>0.1440E+02</td>
<td>VCG OF CIRC. AND CLG. WATER SYSTEMS (FOOT)</td>
</tr>
<tr>
<td>VCG304</td>
<td>(R)</td>
<td>0.1140E+02</td>
<td>VCG OF F. 0. SERVICE SYSTEM (FOOT)</td>
</tr>
<tr>
<td>VCG305</td>
<td>(R)</td>
<td>0.2800E+01</td>
<td>VCG OF LUBE OIL SYSTEM (FOOT)</td>
</tr>
<tr>
<td>VCG306</td>
<td>(R)</td>
<td>0.1900E+02</td>
<td>VCG OF PROPULSION REPAIR PARTS (FOOT)</td>
</tr>
<tr>
<td>VCG307</td>
<td>(R)</td>
<td>0.1000E+02</td>
<td>VCG OF PLANT REPAIR PARTS (FOOT)</td>
</tr>
<tr>
<td>VCG308</td>
<td>(R)</td>
<td>0.1690E+02</td>
<td>VCG OF PROPULSION OPERATING FLUIDS (FOOT)</td>
</tr>
<tr>
<td>VCG309</td>
<td>(R)</td>
<td>0.1890E+02</td>
<td>VCG OF ELECTRICAL POWER GENERATION (FOOT)</td>
</tr>
<tr>
<td>VCG310</td>
<td>(R)</td>
<td>0.2000E+02</td>
<td>VCG OF POWER DISTRIBUTION SWITCHBOARDS (FOOT)</td>
</tr>
<tr>
<td>VCG311</td>
<td>(R)</td>
<td>0.2000E+02</td>
<td>VCG OF POWER DISTRIBUTION SYSTEM (FOOT)</td>
</tr>
<tr>
<td>VCG312</td>
<td>(R)</td>
<td>0.2510E+02</td>
<td>VCG OF LIGHTING SYSTEM (FOOT)</td>
</tr>
<tr>
<td>VCG313</td>
<td>(R)</td>
<td>0.1440E+02</td>
<td>VCG OF ELECTRICAL PLANT REPAIR PARTS (FOOT)</td>
</tr>
<tr>
<td>VCG314</td>
<td>(R)</td>
<td>0.5360E+02</td>
<td>VCG OF NAVIGATIONAL EQUIPMENT (FOOT)</td>
</tr>
<tr>
<td>VCG315</td>
<td>(R)</td>
<td>0.1830E+02</td>
<td>VCG OF GUN FIRE CONTROL SYSTEMS (FOOT)</td>
</tr>
<tr>
<td>VCG316</td>
<td>(R)</td>
<td>0.3710E+02</td>
<td>VCG OF GUN FIRE CONTROL SYSTEMS (FOOT)</td>
</tr>
<tr>
<td>VCG317</td>
<td>(R)</td>
<td>0.1970E+02</td>
<td>VCG OF CONTROL MEASURES (NON-ELECTRONIC) (FOOT)</td>
</tr>
<tr>
<td>VCG318</td>
<td>(R)</td>
<td>0.2350E+02</td>
<td>VCG OF CONTROL MEASURES (FOOT)</td>
</tr>
<tr>
<td>VCG319</td>
<td>(R)</td>
<td>0.0000E+00</td>
<td>VCG OF ASW FC AND TORPEDO FC SYSTEMS (FOOT)</td>
</tr>
<tr>
<td>VCG320</td>
<td>(R)</td>
<td>0.0000E+00</td>
<td>VCG OF COMM. AND CONT. REPAIR PARTS (FOOT)</td>
</tr>
</tbody>
</table>
VCG DATABASE FOR DOG-2 CLASS (CONTINUED)

VCG500  (R)  0.25200E+02  VCG OF HEATING SYSTEMS (FOOT)
VCG501  (R)  0.30000E+02  VCG OF VENTILATION SYSTEMS (FOOT)
VCG502  (R)  0.18400E+02  VCG OF AIR COND. SYSTEMS (FOOT)
VCG503  (R)  0.18600E+02  VCG OF HEAT. PLANT AND EQUIP. (FOOT)
VCG504  (R)  0.12000E+02  VCG OF GAS, HEAT, AV. 1.O., SEWAGE SYSTEMS (FOOT)
VCG505  (R)  0.26500E+02  VCG OF PLUMBING INST. (FOOT)
VCG506  (R)  0.16800E+02  VCG OF FIREMAIN, FISHING, SYMPH, S.W. SVC SYSTEMS (FOOT)
VCG507  (R)  0.22900E+02  VCG OF FIRE EXT. SYSTEMS (FOOT)
VCG508  (R)  0.10600E+02  VCG OF DREME, BALLAST, SUBZC TK SYSTEMS (FOOT)
VCG509  (R)  0.21900E+02  VCG OF FRESH WATER SYSTEM (FOOT)
VCG510  (R)  0.31000E+02  VCG OF SCUPPERS AND DECK DRAINS (FOOT)
VCG511  (R)  0.11200E+02  VCG OF F.O. AND D.O. FILL, VENT, SWG AND XFR SYS (FOOT)
VCG512  (R)  0.62000E+01  VCG OF TANK HEATING SYSTEMS (FOOT)
VCG513  (R)  0.20300E+02  VCG OF COMPRESSED AIR SYSTEMS (FOOT)
VCG514  (R)  0.18100E+02  VCG OF AUX. SILAM, EXH. SILAM AND SILAM DRAINS (FOOT)
VCG515  (R)  0.11300E+02  VCG OF DISTILLING PLANT (FOOT)
VCG516  (R)  0.18800E+02  VCG OF SIEFING SYSTEMS (FOOT)
VCG517  (R)  0.13100E+02  VCG OF RODDERS (FOOT)
VCG520  (R)  0.94000E+02  VCG OF MOUNTING, MACH. AND AC UNITS, DK MCHRY (FOOT)
VCG521  (R)  0.30500E+02  VCG OF LIV. STORES HANDLING SYSTEMS (FOOT)
VCG525  (R)  0.19800E+02  VCG OF AUXILIARY SYSTEMS REPAIR PARTS (FOOT)
VCG526  (R)  0.16000E+02  VCG OF AUXILIARY SYSTEMS FLUIDS (FOOT)
VCG527  (R)  0.32500E+02  VCG OF HULL FITTINGS (FOOT)
VCG528  (R)  0.40200E+02  VCG OF HOATS, BOAT SING AND MOUNTING (FOOT)
VCG529  (R)  0.37600E+02  VCG OF RIGGING AND CANVAS (FOOT)
VCG530  (R)  0.13200E+02  VCG OF LADDERS AND GRATING (FOOT)
VCG531  (R)  0.28800E+02  VCG OF NON-STRUCTURAL BIDS AND DOOKS (FOOT)
VCG532  (R)  0.29000E+02  VCG OF PAINTING (FOOT)
VCG533  (R)  0.24700E+02  VCG OF DECK COVERING (FOOT)
VCG534  (R)  0.20500E+02  VCG OF DECK INSULATION (FOOT)
VCG535  (R)  0.20900E+02  VCG OF STORAGE ROOMS, STOREAGES AND LOCKERS (FOOT)
VCG536  (R)  0.21400E+02  VCG OF EQUIP FOR UTILITY SPACES (FOOT)
VCG537  (R)  0.23800E+02  VCG OF EQUIP FOR WASHOPS, LABS AND TEST AREAS (FOOT)
VCG538  (R)  0.29700E+02  VCG OF EQUIP. FOR GALLEY, PIRY, SCLRY AND COMRSY (FOOT)
VCG539  (R)  0.26700E+02  VCG OF FURNISHINGS FOR LIVING SPACES (FOOT)
VCG540  (R)  0.30600E+02  VCG OF FURNISHINGS FOR OFFICES, COND. CTRS. (FOOT)
VCG541  (R)  0.29700E+02  VCG OF FURNISHINGS FOR MDL. AND DENTAL SP. (FOOT)
VCG542  (R)  0.15000E+02  VCG OF OUTFIT AND FURNISHING REPAIR PARTS (FOOT)
VCG543  (R)  0.35800E+02  VCG OF GUNS AND GUNMOUNTS (FOOT)
VCG544  (R)  0.33900E+02  VCG OF AMMUNITION HANDLING SYSTEMS (FOOT)
VCG545  (R)  0.19600E+02  VCG OF AMMUNITION STORAGE (FOOT)
APPENDIX F

MACHWT MODULE LISTING

Note: Subroutines MAINPG and MODIO are included in the Cube Module listing in Appendix A and do not appear here.
C---------MACHINERY WEIGHT ESTIMATING MODULE SUBPROGRAM---------

SUBROUTINE INPUT(CALLI, IFLAG)

C---------SUBPROGRAM DESCRIPTION---------
C SUBROUTINE INPUT PROVIDES THE USER WITH A MENU FROM WHICH TO
C CHOOSE WHICH MODULE SEGMENT IT IS DESIRED TO OPERATE NEXT. THE
C CHOICES ARE:
C
C    ALL MODULE INPUT VARIABLES
C    THE MODULE UNITS TO BE USED DURING INPUT AND OUTPUT
C    THE MACHINERY WEIGHT ITEM TO BE ESTIMATED.
C    THE DATA FROM EXISTING SHIPS TO BE USED FOR CURVE FITTING
C    THE CHARACTERISTICS OF THE NEW SHIP DESIGN NEEDED AS INPUT-
C    TO ALLOW THE CALCULATIONS TO BE DONE
C
C THE UNITS MODULE ALLOWS THE USER TO SPECIFY THE LENGTH, FORCE
C AND TIME UNITS TO BE USED DURING INPUT AND OUTPUT. LENGTH UNITS ARE
C NEEDED FOR TBP, PROPPELLER DIAMETER, DRAFT AND POSSIBLY SPEED. THESE
C ARE ALL USED IN CALCULATING W(203). TIME UNITS MAY BE NEEDED FOR
C SPEED. FORCE UNITS ARE NEEDED FOR WEIGHTS.
C
C THE MACHINERY WEIGHT ITEM NAME SHOULD BE READ BEFORE OTHER INPUT
C IS ENTERED.
C
C IN READING DATA FROM EXISTING DESIGNS FOR CURVE FITTING, THE USER
C IS ASKED TO SPECIFY WHICH SEQUENTIAL NUMBER OF DATA POINTS THIS PAIR.
C REPRESENTS.
C
C---------SUBPROGRAM ASSUMPTIONS---------
C
C NOUN YET

C---------INPUT VARIABLES---------
C CALLI : TRUE. IF THE INPUT VALUE OF CALLI WAS .TRUE. AND NO ERROR
C OCCURRED WHEN READING OR EDITING AN ESSENTIAL VARIABLE
C          FALSE. IF THE INPUT VALUE OF CALLI WAS .FALSE. OR AN ERROR
C
C IFLAG: DENOTES THE OPERATION TO BE PERFORMED
C          = 0 IF THE USER WISHES TO READ THE VARIABLES
C          = 1 THEN USE THE EXPRESSIONS
C          = 2 EDIT
C          = 3 WRITE
C
C---------LABELLED COMMON VARIABLES---------
C LABELLED COMMON BLOCKS AND MACRONS HAVE BEEN DEFINED IN SUBROUTINE
C NAMES.
C
C WFLAG : DENOES WHICH WEIGHT ITEM IS TO BE ESTIMATED
C          = 1 FOR W(200)
C          = 2 W(201)
C          = 3 W(203)
C
C MPIS : THE NUMBER OF PAIRS OF POINTS TO BE FITTED
C IND : AN ARRAY WHICH STORES THE VALUES OF THE INDEPENDENT VARIABLE
C FOR CURV FITTING
C DEP : AN ARRAY WHICH STORES THE VALUES OF THE DEPENDENT VARIABLE
C FOR CURV FITTING
C-----------------------------------SUBPROGRAMS AND FUNCTIONS CALLED-------------------
C DEX
C INIT
C IMOVIC
C MNUM
C MSOUL
C SIRPAK
C DEX LIBRARY
C NONE
C MODULE
C MGCHRI
C MNUMT
C MVALI
C C LABELS COMMONS
C COMMON /DIALG/ MIRSE
C COMMON /MUCPW/ MC PW
C COMMON /WILAG/ WFLAG
C COMMON /CVPIS/ NPTS, IND(10), DEP(10)
C C VARIABLE AND FUNCTION TYPE DEFINITIONS AND DIMENSIONS
C INTEGER IOLAG, MC PW
C INTEGER MNUMM(2), NITEMS, ITEMS(12), ITEM
C INTEGER NWSH(16), IMS
C INTEGER NPTS, NOPW, WFLAG
C INTEGER READ(2), EDIT(2), WRITE(2)
C LOGICAL CALLAL, LOCAL
C LOGICAL MIRSE, LOGVAL, LMOVIC, INIT
C REAL IND, DEP
C C VARIABLE DATA DEFINITIONS
C DATA IMS/16/
C DATA MNUMM/NIINPU, NI
C DATA NITEMS/6/
C DATA ITEMS/HIALL, HI
C 1 NHUMT, NHT
C 2 NHRT, NHTER
C 3 NHCRV, NHRTS
C 4 NHRCW, NHRTS
C 5 NHRCW, NHRTS
C DATA READ/NREAD, NHX/
C DATA EDIT/NHEDIT, NHX/
C
DATA WRITE/Untitled4HE/*
C ACTIVATE THE LOCAL ALL OPTION IF THE CALLING PROGRAM REQUIRES IT.
C NOTE THAT IF THE LOCAL ALL FLAG IS ACTIVATED HERE MENU 'INPUT' WILL
C NOT BE DEFINED BY THIS INVOCATION OF SUBROUTINE MM1NP.
C
LOCAL-CALALL
IF (LOCAL) GO TO 200
C PREPARE A PROMPTING MESSAGE FOR MENU 'INPUT' AND THEN PROVIDE THE
C MENU TO THE USER.
C
5 CONTINUE
   IF (MFIRST) GO TO 40
   CALL STRPAK(MESS,IMS,4HK,2INSEL) WHICH INPUT VARIABLE SEGMENT
   1 I004
   GO TO (10,20,40,10FLAG)
   10 LOCAL-1MOVE(READ,1,6,NCPW,MESS,40,NCIPW)
   GO TO 50
   20 LOCAL-1MOVE(EDIT,1,6,NCPW,MESS,40,NCIPW)
   GO TO 50
   30 LOCAL-1MOVE(WRITE,1,7,NCPW,MESS,40,NCIPW)
   GO TO 50
   40 CALL STRPAK(MESS,IMS,4HK,2INSEL) WHICH INPUT SEGMENTS
   GO TO 50 CONTINUE
   ITEM MENU(IN(MNUM,N,ITEMS,ITEMS,MSS))
   GO TO (100,200,300,400,500,600), ITEM

C
C SET THE INPUT ALL OPTION.
C
100 CONTINUE
   LOCAL-1.DATE.
C
C READ, EDIT OR WRITE THE INPUT/INPUT MODULE UNITS.
C
C
200 CONTINUE
   CALL MM1NP(LOCAL,10FLAG)
   IF (LOCAL) GO TO 300
   IF (.NOT.CALALL) GO TO 5
   CALALL-1.FALSE.
   GO TO 600
C
C READ THE WEIGHT ITEM TO BE ESTIMATED.
C
300 CONTINUE
   CALL MM11ST
   IF (LOCAL) GO TO 400
   IF (.NOT.CALALL) GO TO 5
   CALALL-1.FALSE.
GO TO 600
C READ, EDIT OR WRITE THE DATA FROM EXISTING SHIPS TO BE USED FOR CURVE FITTING. IF THE USER IS INPUTTING DATA, FIRST QUERY THE USER TO SPECIFY WHICH SEQUENTIAL PAIR OF DATA POINTS THIS REPRESENTS.
C
400 CONTINUE
   IF (101FLAG.NE.1) GO TO 450
410 CONTINUE
   CALL SIRPAK(MESS,INS,494) 59) SPECIFY THE SEQUENTIAL NUMBER OF DATA POINTS.
   NPTS = NPTS + 1
   IF (NPTS.GT.NUF1) NPTS = NUF1
   IF (NPTS.NQ NUF1) GO TO 440
   CALL SIRPAK(MESS,INS,494), 59) HINT IF READING IN SEQUENTIAL.
   NPTS = NPTS + 1
   IF (NPTS.NQ NUF1) GO TO 440
   CALL SIRPAK(MESS,INS,494), 59) THIS INFORMATION IS ESSENTIAL. PLEASE ENTER ALL DATA.
   GO TO 30
440 CONTINUE
   IF (NUF1.NQ NPTS) NPTS = NUF1
450 CONTINUE
   CALL MNGSHR((LOCAL), 101FLAG, NUF1)
   IF (LOCAL) GO TO 450
   IF (.NOT.(CALL)) GO TO 5
   CALL -1,... '1.'
   GO TO 600
C C READ, EDIT OR WRITE THE NEW SHIP CHARACTERISTICS. IF INPUTTING DATA, PROMPT THE USER AS TO WHICH SUPPLEMENTARY INFORMATION IS NEEDED FOR CALCULATING THE PARTICULAR WEIGHT ITEM.
C
500 CONTINUE
   NUF1 = 0
   IF (101FLAG.LT.3) GO TO 550
   IF (101FLAG.NE.0) GO TO 550
   CALL SIRPAK(MESS,INS,494), 58) UTE OF W(203) THE FOLLOWING INFORMATION IS REQUIRED:
   CALL SIRPAK(MESS,INS,494), 58) IEB OF P PTYPE SHP NSHAFT PFTYPE VBM
   550 CONTINUE
   CALL SIRPAK(MESS,INS,494), 58) W(200) OR W(201) INPUT NEW SHIP DATA.
C C C C
CALL MHCHR1(LOCALN,LOHAG,NUP1)
IF (LOCALN) GO TO 600
IF (.NOT.CALAIL) GO TO 5

CALAIL=.FALSE.

C RETURN CONTROL TO CALLING PROGRAM.

C 600 CONTINUE
RETURN
END
C-----------------MACHINERY WEIGHT MODULE SUBPROGRAM------------------

C SUBROUTINE NAME: CMAWGT

C DESCRIPTION:

C SUBROUTINE NAME ALLOWS ITS USERS TO SELECT WHICH MODULE UNITS THEY

C WISH TO READ, EDIT OR WRITE. THE CHOICES ARE:

C ALL MODULES

C AN EXISTING MODULE

C THE LENGTH UNIT

C THE TIME UNIT

C THE FORCE UNIT

C IF THE CALLING PROGRAM ALL OPTION IS ACTIVE, THE LOCAL ALL OPTION IS

C SET TO .TRUE. UPON INVOKING THIS SUBROUTINE. IN THIS CASE THE MENU IS

C NOT DEFINED.

C-----------------OUTPUT VARIABLES------------------

C CALABI: .TRUE. IF THE INPUT VALUE OF CALABI WAS .TRUE. AND NO ERROR

C OCCURRED IN READING OR EDITING A MODULE UNIT

C .FALSE. IF THE INPUT VALUE OF CALABI WAS .FALSE. OR AN ERROR

C OCCURRED IN READING OR EDITING A MODULE UNIT

C-----------------INPUT VARIABLES------------------

C CALAII: .TRUE. THE ALL OPTION OF THE CALLING PROGRAM IS ACTIVE

C .FALSE. THE ALL OPTION OF THE CALLING PROGRAM IS NOT ACTIVE

C IFLAG: DENOTES THE OPERATION TO BE PERFORMED

C EDT, IF THE USER WISHES TO READ MODULE UNITS

C EDY, IF THE USER WISHES TO EDIT MODULE UNITS

C WR, IF THE USER WISHES TO WRITE MODULE UNITS

C LABELLED COMMON VARIABLES

C LABELLED COMMON DIALOG, INOUT, MDNGFW, AND RENSOS HAVE BEEN DEFINED IN

C SUBROUTINE MAINING.

C LABELLED COMMON UNITS, TUNITS, LINFO, LINFO, AND TFINFO

C TINFO AND TFINFO HAVE BEEN INITIALIZED IN SUBPROGRAM BLOCK DATA.

C-----------------SUBPROGRAMS AND FUNCTIONS CALLED------------------

C DEX

C SIRPAK

C MNUIN

C DEX LIBRARY

C UNIT

C TUNI

C FUNI

C MODUL

C NTEGR

C COMMON /DIALOG/ MIERSE

C COMMON /INOUT/ INOUT, ONOUT

C COMMON /MDNGFW/ MDNGFW

C COMMON /RENSOS/ RNRTFL, RNRTFL

C-----------------SUBPROGRAM ASSUMPTIONS------------------

C MM000010

C MM000020

C MM000030

C MM000040

C MM000050

C MM000060

C MM000070

C MM000080

C MM000090

C MM000100

C MM000110

C MM000120

C MM000130

C MM000140

C MM000150

C MM000160

C MM000170

C MM000180

C MM000190

C MM000200

C MM000210

C MM000220

C MM000230

C MM000240

C MM000250

C MM000260

C MM000270

C MM000280

C MM000290

C MM000300

C MM000310

C MM000320

C MM000330

C MM000340

C MM000350

C MM000360

C MM000370

C MM000380

C MM000390

C MM000400

C MM000410

C MM000420

C MM000430

C MM000440

C MM000450

C MM000460

C MM000470

C MM000480

C MM000490
COMMON /UNITS/, PSILUN, UOSILUN
COMMON /UNITS/, PSILUN, UOSILUN
COMMON /UNITS/, PSILUN, UOSILUN
COMMON /UNITS/, PSILUN, UOSILUN
COMMON /UNITS/, PSIUN, OUNUN, DFLUN
COMMON /UNITS/, PSIUN, OUNUN, DFLUN
COMMON /UNITS/, PSIUN, OUNUN, DFLUN
COMMON /UNITS/, PSIUN, OUNUN, DFLUN

C
C VARIABLE AND FUNCTION TYPE DEFINITIONS AND DIMENSIONS
C
INTEGER IOFLAG
INTEGER IMODE, OMODE, IOMODE
INTEGER MINS(11), IMS, NCPM
INTEGER ITEM, MINUT, NUMNUM(2), NITEMS, ITEMS(10)
INTEGER UOSILUN, PSILUN, UOSILUN
INTEGER PSIUN, UOSILUN, PSIUN
INTEGER DFLUN(2), OUNUN(2), DBFLUN(2)
INTEGER DBFLUN(16), OUNUN(16), DBFLUN(16)
INTEGER LUNUN(2), LUNUN(2), FUNUN(2)
INTEGER DFLUN, OUNUN, DBFLUN
INTEGER RMULT, RWRITE, RWRITE
INTEGER PSILUN, PSIUN, PSILUN
INTEGER PME(16)
INTEGER BLAUN(2), LEI(2), LIRE(2)
LOGICAL MILL, MPMREP
LOGICAL CAIL, LOCAL
LOGICAL LOGVAR, LMOVE

C
C VARIABLE DATA DEFINITIONS
C
DATA IMS /11/
DATA MINUN/MNUNIT, 4H /
DATA NITEMS/5/
DATA ITEMS/MNUN, 4H /
DATA MMUN/MNUN, 4H /
DATA MNUIS/NIUL, 4H /
DATA MNUIS/NIUL, 4H /
DATA READ/RREAD, NHA, MHA /
DATA EDIT/REEDIT, NHA, MHA /
DATA WRITE/WWRITE, NHA, MHA /

C
C INITIALIZE THE VALUE OF MPMREP.
C
PMREP= TRUE.

C
C SET THE VALUE OF IOMODEL AND RWRITE ACCORDING TO WHETHER THE USER
C WISHES TO READ, EDIT OR WRITE.
C
IF (IOFLAG LT 3) GO TO 10

TOMODE=MODE
RMTFILE:RMTFILE
GO TO 15
10 CONTINUE
TOMODE-OMODE
RMTFILE:RMTFILE
C
C ACTIVATE THE LOCAL ALL OPTION IF THE CALLING PROGRAM REQUIRES IT.
C NOTE THAT IF THE LOCAL ALL OPTION IS SET IN THIS MANNER, MENU 'UNIT'
C IS NOT INITIATED BY THE INVOCATION OF THIS ROUTINE.
C
15 CONTINUE
LOCAL-CALL ALL
IF (LOCALALL) GO TO 200
C
C PREPARE PROMPTING MESSAGE FOR MENU 'UNIT'.
C
20 IF (MTHSEL) GO TO 40
CALL SIRPAK(MISS,IMS,4HC,,22HSELECT WHICH UNIT 10C)
GO TO (25,30,35,10FLAG)
25 CONTINUE
IMOVIC(READ,1,6,NCPW,MISS,22,NCPW)
GO TO 50
30 CONTINUE
IMOVIC(EDIT,1,6,NCPW,MISS,22,NCPW)
GO TO 50
35 CONTINUE
IMOVIC(WRITE,1,7,NCPW,MISS,22,NCPW)
GO TO 50
40 CONTINUE
CALL SIRPAK(MISS,IMS,4HC,,12HWHICH UNIT?)
C
C SELECT AN ITEM FROM MENU 'UNIT' AND BRANCH ACCORDINGLY.
C
50 CONTINUE
ITEM MENU(MNUNUM,MTHSEL,11MS,MISS)
GO TO (100,200,300,400,500,11IM)
C
C ACTIVATE THE SUBPROGRAM'S ALL OPTION.
C
100 CONTINUE
LOCALALL-.TRUE.
C
C READ, EDIT OR WRITE THE LENGTH UNIT.
C
200 CONTINUE
CALL LUNIT(U10UN,LOCALALL,
1 10FLAG,OMODE,MTHSEL,NCPW,
2 DBLUNH,DSULNC,
3 DMPREP,MRES,
4    KWTIL, LUNITRM,
5    DETILN
11   (TOCALL) GO TO 300
11   (.NOT.CALL) GO TO 20
   CALL .FALSE.
   GO TO 500
C
C READ, EDIT OR WRITE THE TIME UNIT.
C
300 CONTINUE
   CALL FUNTUITUTION, LOCAL,
1    10IFAC, 10MODE, 10ERSE, NCPW,
2    10ITNN, 10ITNC,
3    10MPPR, PMP.
4    10KWTIE, LUNITRN,
5    DETILN
11   (TOCALL) GO TO 400
11   (.NOT.CALL) GO TO 20
   CALL .FALSE.
   GO TO 500
C
C READ, EDIT OR WRITE THE FORCE UNIT.
C
400 CONTINUE
   CALL FUNTUITUTION, LOCAL,
1    10IFAC, 10MODE, 10ERSE, NCPW,
2    10ITNN, 10ITNC,
3    10MPPR, PMP.
4    10KWTIE, LUNITRN,
5    DETILN
11   (TOCALL) GO TO 500
11   (.NOT.CALL) GO TO 20
   CALL .FALSE.
C
C RETURN CONTROL TO THE CALLING PROGRAM.
C
500 CONTINUE
   RETURN
   END

MMU1480
MMU1490
MMU1500
MMU1510
MMU1520
MMU1530
MMU1540
MMU1550
MMU1560
MMU1570
MMU1580
MMU1590
MMU1600
MMU1610
MMU1620
MMU1630
MMU1640
MMU1650
MMU1660
MMU1670
MMU1680
MMU1690
MMU1700
MMU1710
MMU1720
MMU1730
MMU1740
MMU1750
MMU1760
MMU1770
MMU1780
MMU1790
MMU1800
MMU1810
MMU1820
MMU1830
MMU1840
MMU1850
MMU1860
C------------MACHINERY WEIGHT ESTIMATING MODULE SUBPROGRAM------------------MWI.00010
C-------------------------------------------------------------SUBPROGRAM DESCRIPTION-------------------MWI.00020
C SUBROUTINE MAIST SETS THE VALUE OF WLAG WHEN THE USER SPECIFIES
C ONE OF THE FOLLOWING MACHINERY WEIGHT ITEMS TO BE ESTIMATED.
C W(200) BOILERS AND ENERGY CONVERTERS
C W(201) PROPULSION UNITS
C W(203) PROPELLER, SHUTTING AND BEARINGS
C-------------------------------------------------------------SUBPROGRAM ASSUMPTIONS-------------------MWI.00030
C NONE YET
C---------------------------------------------------------------------LABELLED COMMON VARIABLES-------------------MWI.00040
C LABELLED COMMON DIAGN HAS BEEN DEFINED IN SUBROUTINE MAINT.
C LABELLED COMMON WLAG HAS BEEN DEFINED IN SUBROUTINE MAINT.
C---------------------------------------------------------------------SUBPROGRAMS AND FUNCTIONS CALLED-------------------MWI.00050
C DEX
C SIRPAK
C MIFIN
C DEX LIBRARY
C NONE
C MODUL
C NONE
C---------------------------------------------------------------------LABELLED COMMONS-------------------MWI.00060
C COMMON /DIALG/ MIESE
C COMMON /WLAG/ WLAG
C---------------------------------------------------------------------VARIABLE AND FUNCTION TYPE DEFINITIONS AND DIMENSIONS-------------------MWI.00070
C INTEGER MIESE(2), WLAG, ITEMS(6), ITEM
C INTEGER MISS(12), LMS, WLAG
C LOGICAL MIESE
C---------------------------------------------------------------------VARIABLE DATA DEFINITIONS-------------------MWI.00080
C DATA LMS/12/
C DATA MENU0/MINT, 1, MITEMS/
C DATA MIESE/12/
C DATA ITEMS/4/MIESE, 1
C DATA ITEMS/4/MIESE, 2
C C PREPARE A PROMPTING MESSAGE FOR MENU 'WT. ITEMS' AND THEN PROVIDE THE
C MENU.
C IF (MIESE) GO 10 10
C CALL SIRPAK(MESS, LMS, WLAG) ; ABISELECT THE DESIRED WEIGHT ITEM TO DWI.00080
C IE ESTIMATED Q
C
GO TO 50
10 CONTINUE
   CALL STRPAK(MESS,LMS,NDH,191MHGH,WEIGHT,ITEM7)
50 CONTINUE
   ITEM.MENUIN(MINNUM,MITEMS,MITEMS,MESS)
   GO TO (100,200,300),ITEM
C
C USER WISHES TO ESTIMATE W(200).
100 CONTINUE
   WSAG-1
   GO TO 99999
C
C USER WISHES TO ESTIMATE W(200).
C
200 CONTINUE
   WSAG-2
   GO TO 99999
C
C USER WISHES TO ESTIMATE W(201).
C
300 CONTINUE
   WSAG-3
C
C RETURN TO CALLING PROGRAM
C
99999 CONTINUE
   RETURN
   END
MACHINERY WEIGHT MODUFE SUBPROGRAM

SUBROUTINE WIGHT(CALAI, IOFLAG, NUPF)

DESCRIPTION

THE USER TO READ, EDIT OR WRITE THE FOLLOWING CHARACTERISTICS OF A SHIP DESIGN:

1. LENGTH BETWEEN PERPENDICULARS
2. DRAFT (KELL)
3. TYPE OF PROPULSION PLANT
4. INSTALLED SHAFT MOMEPOWER
5. NUMBER OF PROPELLER SHAFTS
6. MAXIMUM SUSTAINED SPEED
7. TYPE OF PROPELLER (FP OR CRP)
8. PROPELLER DIAMETER
9. WEIGHT OF BOILERS
10. WEIGHT OF Propulsion UNITS
11. WEIGHT OF PROPELLER, SHAPING AND BOOKING

THE FIRST 8 ITEMS CAN BE USED IN THE ESTIMATING OF THE LATTER 3.

FOR A NEW SHIP DESIGN. WHEN OBTAINING DATA FOR THE CALCULATION OF THE INDIVIDUAL
EQUATIONS FOR THE WEIGHS, THE INDEPENDENT VARIABLES MUST BE READ FIRST AND THE DEPENDENT VARIABLE SECOND.

SUBROUTINE ASSUMPTIONS

THIS SUBROUTINE IS ACCESSED THROUGH EITHER ITEM 'CURVE FIT'
OR ITEM 'NEW SHIP' OF SUBROUTINE M189. WHEN READING DATA POINTS
THAT ARE TO BE USED FOR DETERMINING PARAMETRIC EQUATIONS, CHECK
DATA USED VIA 'CURVE FIT'. WHEN READING IN SUPPLEMENTAL
INFORMATION ABOUT THE NEW SHIP DESIGN, IT IS USED TO OBTAIN THE PARTI-
CULAR WEIGHT ESTIMATE. IT SHOULD BE ACCESSED THROUGH 'NEW SHIP'.

CHURCH HAS BEEN ACCESSED BEFORE RETURNING TO THE CALLING PROGRAM.

OUTPUT VARIABLES

CALAI: TRUE. IF THE INPUT VALUE OF CALAI WAS TRUE, AND NO ERROR
OCCURRED WHILE READING OR EDITING A VITAL VARIABLE
FALSE, IF THE INPUT VALUE OF CALAI WAS FALSE, OR AN ERROR
OCCURRED WHILE READING OR EDITING A VITAL VARIABLE

INPUT VARIABLES

CALAI: TRUE. IF THE ALL OPTION OF THE CALLING PROGRAM IS ACTIVE
FALSE, IF THE ALL OPTION OF THE CALLING PROGRAM IS NOT ACTIVE
IOFLAG: DENOTES THE OPERATION BEING PERFORMED
1 = IF THE USER WISHES TO READ THE VARIABLES
2 = EDIT
3 = WRITE
NUPF: A NUMBER WHICH INDICATES EITHER THE SEQUENTIAL NUMBER OF THE DATA POINTS TO BE READ OR THAT NEW SHIP INFORMATION IS
Sought

LABELED COMMON VARIABLES

LABEL COMM, DIABIT, INOUT, MNCMP, AND SCREENS HAVE BEEN DEFINED IN
MACHINER MNCMP.
LABEL COMMONS FNM, LIMITS AND NITERS HAVE BEEN DEFINED IN
MACHINER MNCMP.
SUBROUTINE M189.
C

VARIABLE AND FUNCTION TYPE DEFINITIONS AND DIMENSIONS

INTEGER IOTAG, IMODE, OMODE, NCPW
INTEGER MANFL, MAIN1, MAIN2, RSCIHR(2)
INTEGER PSIQN, UNIQU, PSIQU, UNIQU, PSIQU, UNIQU
INTEGER NPUF
INTEGER IVAR, IDEL, DURNAME(2), PMLS(16), PHORMC(16)
INTEGER LBPNAME(2), IHNAME(2), PFINAM(2), SHIPNAM(2), NSHFM(2),
1 VSUMNM(2), PFINPM(2), DFRPM(2), W200NM(2), W201MN(2),
2 W203NM(2)
INTEGER CMM12(16), CMM15(16), CMM19(8), CMM20(12), CMM21(7),
1 CMM24(16), CMM27(17), CMM28(9), W200C(13), W201C(10),
2 W203C(14)
INTEGER DE19, DE21, DE27
INTEGER FFTNAME, MSHNEW, PRTRY,
INTEGER MISS(16), IMS
INTEGER MNUMNM(2), NITEMS, NITEMS(24), ITEM
INTEGER MNUMYS(2), NITEMY, TIMLY(4), YITEM
INTEGER HEAD(2), ED11(2), WRITI(2)
INTEGER UNIDNM(3), NAMID(2), NAM106(2), NAM102(1), NAM103(1),
1 NAM101(1), NAM12(3), NAM102(1), NAM103(1), NAM106(2),
2 NAM12(3), UNSPE(3), KN00(1), UNSHH(3), UNAR(3)
LOGICAL CALL, NIERSE, LOGCAL, IMOVEG
LOGICAL N11NF, N12NF, N18NF
LOGICAL ISCLUD, ISCED, ISCUMP, RSCDR, RSCDE, RSCUMP
LOGICAL UNIIF, UNIIT, UNIIT, USPEED
REAL DE17, DE19, DE21, DE27, DE120, DE121, DE123
REAL LPNEW, INME, SHIPNEW, VSUMNM, DFRNEW, W200NU, W201NU, W203NU

M001480
M001490
M001500
M001510
M001520
M001530
M001540
M001550
M001560
M001570
M001580
M001590
M001600
M001610
M001620
M001630
M001640
M001650
M001660
M001670
M001680
M001690
M001700
M001710
M001720
M001730
M001740
M001750
M001760
M001770
M001780
M001790
M001800
M001810
M001820
M001830
M001840
M001850
M001860
M001870
M001880
M001890
M001900
M001910
M001920
M001930
M001940
M001950
M001960
REAL IND, DEP
REAL UINT,F, UNITA, CONVIM, CONVIA, CONV12, CONVFA, UFSPE, UFSPLA
REAL RVAR, NDEF

C VARIABLE DATA DEFINITIONS
C
DATA IMS/16/
DATA MNUM/4CHAR, MNACT./
DATA RITTER/N/12/
DATA ITEM/N/UP, 4H,
   1 NWKFA, 4H1,
   2 NWKFA, 4H1,
   3 NWKFA, 4H1,
   4 NWKFA, 4H1,
   5 NWKFA, 4H1,
   6 NWKFA, 4H1,
   7 NWKFA, 4H1,
   8 NWKFA, 4H1,
   9 NWKFA, 4H1,
  * NWKFA, 4H1/
DATA MNUMY/N/HMOYE, MHS, MNO/
DATA ITEMY/2/
DATA ITEMY/1YVES, 4H /
   1 NWKTY, 4H1/
DATA KNO1 /K/KNO1, 4H, 4H /
DATA UNSHP/N/UP, 4H, 4H /
DATA UNKW/N/UP, 4H, 4H /
DATA READ/N/HREAD, 4H, 4H /
DATA EDIT/N/HEDIT, 4H, 4H /
DATA WTH11/N/HWT11, 4HEX, 4H /
DATA CONVIM, CONVFA, CONV12, UFSPE /0.0,0.0,0.0,0.0,0.0/ C
C INITIALIZE PMPREP AND MENUFL.
C
PMPREP = TRUE,
MENUFL = FALSE.
C
C DETERMINE THE NAMES OF THE LENGTH UNITS FOR LBP, PROPELLER DIAMETER,
C SPEED, HORNPOWER AND DRAFT AND THE MULTIPLICATION CONVERSION FACTOR.
C
LOGVAL=UNUNITF(CONVIM, NAM02, NAM06, NAM12, CALALL,
  1 PSTLUN, UTI04UN, NCPW)
   II (.NOT.LOGVAL) GO TO 99999
C
C DETERMINE THE NAMES OF THE TIME UNITS FOR SPEED
C AND THE MULTIPLICATIVE CONVERSION FACTOR.
C
LOGVAL=UNUNITF(CONVIM, NAM02, NAM03, NAM06, NAM12, CALALL,
PSTF0M,UNITCM,NCPW)
II (.NOT. LOGVAL) GO TO 99999
C
C DETERMINE THE NAMES OF THE FORCE UNITS FOR THE WEIGHTS
C OF THE MACHINERY ITEMS AND THE MULTIPLICATIVE CONVERSION FACTOR.
C
LOGVAL-UNIT11(CONVFM,NAM02,NAM03,NAM12,CALALL,
1 PSTF0M,UNITCM,NCPW)
II (.NOT. LOGVAL) GO TO 99999
C
C PREPARE A PROMPTING MESSAGE FOR MENU 'CHARATC.' AND PROVIDE IT
C TO USER.
C
N O
5 CONTINUE
II (FIRST) GO TO 40
CALL SIRPAK(MESS,...,NCW ) 32NS(SELECT WHICH CHARACTERISTIC TO X)
GO TO (10,20,30),IOFLAG
10 CONTINUE
LOGVAL-IMOVIC(READ,1,6,NCPW,MESS,32,NCPW)
GO TO 50
20 CONTINUE
LOGVAL-IMOVIC(EDIT,1,6,NCPW,MESS,32,NCPW)
GO TO 50
30 CONTINUE
LOGVAL-IMOVIC(WRITE,1,7,NCPW,MESS,32,NCPW)
GO TO 50
40 CONTINUE
CALL SIRPAK(MESS,...,NCW ) 22NS(CHARACTERISTIC TO Y)
50 CONTINUE
ITEM MENUIN(MINUM,ITEMS,MESS)
C
C PROVIDE INDICATOR TO TELL IF VARIABLE TO BE READ IS AN INDEPENDENT
C VARIABLE, A DEPENDENT VARIABLE OR A NEW SHIP CHARACTERISTIC.
C
N O
11 (NOFL.EQ.0) N-O
11 (M.0Q.3.AND.11M.NL.12) GO TO 8000
11 (M.QO.3.AND.11M.NL.12) GO TO 8000
GO TO (100,200,300,400,500,600,700,800,900,1000,1100,99999),IEM
C
SUBSTITUTE VARIABLES AND THEN BRANCH ACCORDING TO WHETHER TO READ,
C EDIT OR WRITE THE LDP.
C
100 CONTINUE
II (IOFLAG.EQ.3) KVVAR=LABPERW
DIMNAME(1)=LVBNAM(1)
DIMNAME(2)=LVBNAM(2)
UNITM-CONVF
UNITA-CONVIA

MCO02460
MCO02470
MCO02480
MCO02490
MCO02500
MCO02510
MCO02520
MCO02530
MCO02540
MCO02550
MCO02560
MCO02570
MCO02580
MCO02590
MCO02600
MCO02610
MCO02620
MCO02630
MCO02640
MCO02650
MCO02660
MCO02670
MCO02680
MCO02690
MCO02700
MCO02710
MCO02720
MCO02730
MCO02740
MCO02750
MCO02760
MCO02770
MCO02780
MCO02790
MCO02800
MCO02810
MCO02820
MCO02830
MCO02840
MCO02850
MCO02860
MCO02870
MCO02880
MCO02890
MCO02900
MCO02910
MCO02920
MCO02930
MCO02940
UNITNM(1)-NAME(21)
UNITNM(2)-NAME(22)
UNITNM(3)-NAME(23)
DO 110 I=1,10
PRINT(1)-CMM12(I)
110 CONTINUE
RETURN
GO TO (2100,2200,2300),10FLAG

C SUBSTITUITE VARIABLES AND THEN BRANCH ACCORDING TO WHETHER TO READ,
C EDIT OR WRITE THE DRAFT.

C 200 CONTINUE
IF (10FLAG.4.1) RVAR-HNEW
   Diname(1)-Diname(1)
   Diname(2)-Diname(2)
   UNITNM-CONVM
   UNITNM-CONVIA
   UNITNM(1)-NAME(21)
   UNITNM(2)-NAME(22)
   UNITNM(3)-NAME(23)
   DO 210 I=1,16
   PRINT(1)-CMM15(1)
210 CONTINUE
RETURN
GO TO (2100,2200,2300),10FLAG

C SUBSTITUITE VARIABLES AND BRANCH ACCORDING TO WHETHER TO READ,
C EDIT OR WRITE THE TYPE OF PROPELSION PLAN.

C 300 CONTINUE
IF (10FLAG.4.1) RVAR-PPTNEW
   Diname(1)-PPTNAME(1)
   Diname(2)-PPTNAME(2)
   DO 310 I=1,8
   PRINT(1)-CMM19(1)
310 CONTINUE
RETURN
GO TO (3100,3200,3300),10FLAG

C SUBSTITUITE VARIABLES AND BRANCH ACCORDING TO WHETHER TO READ,
C EDIT OR WRITE THE INSTALLED SHIP.

C 400 CONTINUE
IF (10FLAG.4.1) RVAR-SHPNEW
   Diname(1)-SHPNAME(1)
   Diname(2)-SHPNAME(2)
C PREPARE MESSAGE FOR READING, EDITING OR WRITING SHAFT HORSEPOWER.
   MSG30
CALL STIRPK(MLSS,1MS,4HK, SHAFT HORSEPOWER MUST BE IN UNITS OF HP)
IF EITHER HP OR KW, CALL MSQU(MLSS)
CALL STIRPK(MLSS,1MS,4HK, SHAFT HORSEPOWER IN HP, 1PM)
ITEM MENU(MENUYS,ITEMY,ITEMY,MLSS)
GO TO (410,420), ITEM

THE I/O UNITS OF SHIP ARE HP.

410 CONTINUE
UNITIM.1.0
UNITIM(1).UNSHIP(1)
UNITIM(2).UNSHIP(2)
UNITIM(3).UNSHIP(3)
GO TO 440

THE I/O UNITS OF SHIP ARE KW.

420 CONTINUE
UNITIM.1.3410
UNITIM(1).UNKW(1)
UNITIM(2).UNKW(2)
UNITIM(3).UNKW(3)

440 CONTINUE
UNITIA.0.0
DO 440 1.1,12
PMORM(1).CMN120(1)

450 CONTINUE
RUE 120
GO TO (2100,2200,2300), 1011AG

SUBSTITUTE VARIABLES AND THEN BRANCH ACCORDING TO WHETHER TO READ, EDIT OR WRITE THE SPEED. FIRST PREPARE A PROMPTING MESSAGE FOR READING SPEED.

500 CONTINUE
CALL STIRPK(MLSS,1MS,4HK, SHIP VSUS HAVE UNITS OF 'KNOTS'?)
ITEM MENU(MENUYS,ITEMY,ITEMY,MLSS)
GO TO (510,520), ITEM

510 CONTINUE
CALL STIRPK(MLSS,1MS,4HK, THANK YOU. PROCEEDING)
CALL MSQI(MLSS)
UNITIM(1).KNOT(1)
UNITIM(2).KNOT(2)
UNITIM(3).KNOT(3)
UNITIM.1.0
GO TO 530
520 CONTINUE
CALL SIRPAK(MSS,IMS,NH4),.62H1HE CONVERSION FACTORS FOR CHANGING
1 THE INPUT/OUTPUT UNITS TO
CALL MISOU(MSS)
CALL SIRPAK(MSS,IMS,NH4),.62H1HE PROGRAM STANDARD UNITS OF KNOTS
1 I WILL BE BASED ON THE LENGTH
CALL MISOU(MSS)
CALL SIRPAK(MSS,IMS,NH4),.35H1AND TIME UNITS YOU HAVE SPECIFIED.
1
1) CALL MISOU(MSS)
UIGVAL,USPFL,USPEE,CAAL,
1 CONV1M,CONV1N,NAM06,NAM102,
2 U10UNU20UNU2M,NCPW)
II (.NO1,UIGVAL) Go 10 99999
UNITM*USPFI/1.68
UNITM1.USPFI(1)
UNITM2.USPFI(2)
UNITM3.USPFI(3)
530 CONTINUE
II (10IFLAG.EQ.3) KVAR-VUSMNU
DIMAHE(1)-VUSMN(1)
DIMAHE(2)-VUSMN(2)
UNITA-A.0
DO 590 1=1,16
PHRIN(1)-CM124(1)
550 CONTINUE
IFLAG=124
DO 590 (2100,2200,2300),10IFLAG
C C SUBSTITUTION VARIABLES AND THEN BRANCH ACCORDING TO WHETHER TO READ,
C EDIT OR WRITE THE NUMBER OF PROPELLER SHAFTS.
C C GO 10 (2100,2200,2300),10IFLAG
600 CONTINUE
II (10IFLAG.EQ.3) IVAR-HSHNEW
DIMAHE(1)-HSHNEW(1)
DIMAHE(2)-HSHNEW(2)
DO 610 1=1,7
PHROH(1)-CMH121(1)
610 CONTINUE
IDEF-DEF21
GO 10 (3100,3200,3300),10IFLAG
C C SUBSTITUTION VARIABLES AND THEN BRANCH ACCORDING TO WHETHER TO READ,
C EDIT OR WRITE THE TYPE OF PROPELLER.
C C GO 10 (3100,3200,3300),10IFLAG
700 CONTINUE
II (10IFLAG.EQ.3) IVAR-PHINLW
DIMAHE(1)-PHIN(1)
DIMAHE(2)-PHIN(2)
DO 710 1,1
   PHGRCH(1),CMN127(1)
710 CONTINUE
   DEFT-DEF2
   GO TO (3100, 3200, 3300), I0FLAG

C
C SUBSTITUTE VARIABLES AND THEN BRANCH ACCORDING TO WHETHER TO READ,
C EDIT OR WRITE THE PROPELLER DIAMETER.
C
800 CONTINUE
   IF (I0FLAG.EQ.1) RVAR-DPRNEW
      DNAMES(1), DPRPMM(1)
      DNAMES(2), DPRPMM(2)
      UNITM-CONVM
      UNITA-CONVIA
      UNITNM(1)-NAM12(1)
      UNITNM(2)-NAM12(2)
      UNITNM(3)-NAM12(3)
      DO 810 I-1,9
      PHORCH(1), CMN128(1)
810 CONTINUE
   DEFT-DEF2
   GO TO (3100, 3200, 3300), I0FLAG

C
C SUBSTITUTE VARIABLES AND THEN BRANCH ACCORDING TO WHETHER TO READ,
C EDIT OR WRITE W(200).
C
900 CONTINUE
   IF (I0FLAG.EQ.1) RVAR-WZ00NU
      DNAMES(1), WZ00NM(1)
      DNAMES(2), WZ00NM(2)
      UNITM-CONVM
      UNITA-CONVIA
      UNITNM(1)-NAM12(1)
      UNITNM(2)-NAM12(2)
      UNITNM(3)-NAM12(3)
      DO 910 I-1,13
      PHORCH(1)-WZ00UC(1)
910 CONTINUE
   DEFT-DEF200
   GO TO (3100, 3200, 3300), I0FLAG

C
C SUBSTITUTE VARIABLES AND THEN BRANCH ACCORDING TO WHETHER TO READ,
C EDIT OR WRITE W(201).
C
1000 CONTINUE
   IF (I0FLAG.EQ.1) RVAR-WZ01NU
      DNAMES(1)-WZ01NM(1)
      DNAMES(2)-WZ01NM(2)
UNITI = CONVIM
UNITA = CONVIA
UNITNM(1) = NAM1 12(1)
UNITNM(2) = NAM1 12(2)
UNITNM(3) = NAM1 12(3)
DO 1010 I=1,10
PMOHGN(1) = W013C(1)

1010 CONTINUE
RDEF DEF201
GO TO (2100,2200,2300), IFLAG
C
C SUBSTITUE VARIABLES AND THEN BRANCH ACCORDING TO WHETHER TO READ,
C EDIT OR WRITE W(203).
C
1100 CONTINUE
IF (IFLAG,10,3) RVAR=W203NU
IMBAH(1) = W013NM(1)
IMBAH(2) = W013NM(2)
UNITI = CONVIM
UNITA = CONVIA
UNITNM(1) = NAM1 12(1)
UNITNM(2) = NAM1 12(2)
UNITNM(3) = NAM1 12(3)
DO 1110 I=1,10
PMOHGN(1) = W013C(1)

1110 CONTINUE
RDEF DEF203
GO TO (2100,2200,2300), IFLAG
C
C READ A RAI VARIABLE.
C
2100 CONTINUE
ILDVAT = NCSFDR(RVAR, CATA1, 0)
2
MILRE, IMODE, NCVM,
1
DINAM, UNITI, UNITA, UNITNM, .FALSE.,
4
PPHRE, PMS, PMOHGN,
4
RMM (11), NCSFRM, RDEF)
1 IF (.NOT.ILDVAT) GO TO 2150
11 (NUPT.10,0) GO TO 2150
11 (NUPT.2) GO TO 2125
IND(NUPT) = RVAR
GO TO 5
2125 CONTINUE
DEF(NUPT) = RVAR
GO TO 5
2150 CONTINUE
GO TO (5010,5020,5,5040,5050,5,5080,5090,5100,5110,99999), IEM
C
C EDIT THE RAI VARIABLE.
C
4
C 2200 CONTINUE
LOGVAL-RSCDT(IVAR, CALALL,
   1 MEIRSE, NCPW,
   2 DBNAME, UNITFM, UNITIA, UNITNM,
   3 PMPREP, PMCS, PMORGN)
   II (.NO1, LOGVAL) GO TO 5
   II (NPUT, 10.0) GO TO 2250
   II (N, IQ, 2) GO TO 2225
   IN(NUPT) RVAR
   GO TO 5
2225 CONTINUE
DI(NUPT)-RVAR 
   GO TO 5
2250 CONTINUE
GO TO (5010,5020,5,5080,5080,5,5080,5090,5100,5110,99999), IITEM
C C WRITE THE REAL VARIABLE.
C 2300 CONTINUE
LOGVAL-RSCDFC(CALALL,
   1 MEIRSE, IMODE, NCPW,
   2 TVAR, DBNAME, UNITFM, UNITIA, UNITNM,
   3 PMPREP, PMCS, PMORGN,
   4 RMFIL, RSCFRM)
   GO TO 5
C C READ THE INTEGER VARIABLE.
C 3100 CONTINUE
LOGVAL-ISCDF(IVAR, CALALL,
   1 MEIRSE, IMODE, NCPW,
   2 DBNAME, IFASE,
   3 MNNUL, MNNUM, NITEMS, ITEMS,
   4 PMPREP, PMCS, PMORGN,
   5 RMFIL, INITRM, IDEF)
   II (.NO1, LOGVAL) GO TO 5
   GO TO (5,5,6030,5,5,6080,6070,5,5,5,5,99999), IITEM
C C EDIT THE INTEGER VARIABLE.
C 3200 CONTINUE
LOGVAL-ISCDE(TIVAR, CALALL,
   1 MEIRSE, NCPW,
   2 DBNAME,
   3 MNUFL, MNUMM, NITEMS, ITEMS,
   4 PMPREP, PMCS, PMORGN,
   5 IDEF)
   II (.NO1, LOGVAL) GO TO 5
MCO05400
MCO05410
MCO05420
MCO05430
MCO05440
MCO05450
MCO05460
MCO05470
MCO05480
MCO05490
MCO05500
MCO05510
MCO05520
MCO05530
MCO05540
MCO05550
MCO05560
MCO05570
MCO05580
MCO05590
MCO05600
MCO05610
MCO05620
MCO05630
MCO05640
MCO05650
MCO05660
MCO05670
MCO05680
MCO05690
MCO05700
MCO05710
MCO05720
MCO05730
MCO05740
MCO05750
MCO05760
MCO05770
MCO05780
MCO05790
MCO05800
MCO05810
MCO05820
MCO05830
MCO05840
MCO05850
MCO05860
MCO05870
MCO05880
GO TO (5, 5, 6030, 5, 5, 6060, 6070, 5, 5, 5, 99999), ITEM
C  WRITE THE INTEGER VARIABLE.
C
3300 CONTINUE
   LGOVAI - ESCOMP(CALALL,
      1: MIFASE, ONODE, NSF, NV, S
      2: IVAR, DBNAME, PNOCH, NSVAR, NMIN, NSPL
      3: PMPRE, PMES, NMIN)
   GO TO 5
C  ASSIGN RVAR TO THE NEW SHIP EBP.
C 5010 CONTINUE
   LBPNW-RVAR
   GO TO 5
C  ASSIGN RVAR TO THE NEW SHIP DRAFT.
C 5020 CONTINUE
   INEWW-RVAR
   GO TO 5
C  ASSIGN RVAR TO THE NEW SHIP SHP.
C 5040 CONTINUE
   SHPWV-RVAR
   GO TO 5
C  ASSIGN RVAR TO THE NEW SHIP VSUS.
C 5050 CONTINUE
   VSUSMU-RVAR
   GO TO 5
C  ASSIGN RVAR TO THE NEW SHIP DPROP.
C 5080 CONTINUE
   DPROPW-RVAR
   GO TO 5
C  ASSIGN RVAR TO THE NEW SHIP W(200).
C 5090 CONTINUE
   W200MUR-RVAR
C  ASSIGN RVAR TO THE NEW SHIP W(201).
5100 CONTINUE
W01NU RVAR
   GO TO 5
C
C ASSIGN IVAR TO THE NEW SHIP W{203}.
C 5110 CONTINUE
W01NU RVAR
   GO TO 5
C
C ASSIGN IVAR TO THE NEW SHIP PPTYPE.
C 6030 CONTINUE
PFINW-IVAR
   GO TO 5
C
C ASSIGN IVAR TO THE NEW SHIP NSHIT.
C 6060 CONTINUE
NSHINW-IVAR
   GO TO 5
C
C ASSIGN IVAR TO THE NEW SHIP PRPTYPE.
C 6070 CONTINUE
PRFINW-IVAR
   GO TO 5
C
C ERROR IN READING DATA FOR CURVE FITTING. INFORM USER.
C 8000 CONTINUE
CALL STRPAK(MESS,1MS,41K .5/IYOU HAVE READ IN TOO MANY NUMBERS I'MC06680)
10/I THIS PAIR OF DATA .4)
CALL NESQUT(MESS) I'MC06690
CALL STRPAK(MESS,1MS,41K .5/HERE-SELECT THE CURVEPTS ITEM IN NENUM I'MC06710
1 INPUT AND TRY AGAIN .4)
CALL NESQUT(MESS)
   GO TO 99999
C
C RETURN TO CALLING PROGRAM.
C 99999 CONTINUE
RETURN
END
---MACHINERY WEIGHT ESTIMATING MODULE SUBPROGRAM---

SUBROUTINE MUCMP

---SUBPROGRAM DESCRIPTION---

SUBROUTINE MUCOMP USES THE DATA FROM EXISTING SHIPS TO ESTIMATE
M(200) OR M(201) AS A FUNCTION OF SHIP. M(203) IS ESTIMATED
BASED ON A SET OF FIXED PARAMETRIC CURVES.

THE STANDARD UNITS OF THIS PROGRAM ARE FEET, TONS AND KNOBS.

---SUBPROGRAM ASSUMPTIONS---

THE MAXIMUM NUMBER OF PAIRS OF DATA POINTS TO BE FITTED IS 10.
LABELED COMMON NUMCMP HAS BEEN DEFINED IN SUBROUTINE MAINPG.
LABELED COMMON CURVPS AND WFLAG HAVE BEEN DEFINED IN SUBROUTINE
MUCHP. LABELED COMMON MEXINT HAS BEEN DEFINED IN SUBROUTINE MUCHT.
LABELED COMMON COFFS HAS BEEN DEFINED IN SUBROUTINE MUCOEF.

---SUBPROGRAMS AND FUNCTIONS CALLED---

DEX
SIRPAK
IMOVIC
MISOUT
DEX LIBRARY
MINE
MODUL

---LABELED COMMONS---

COMMON /MUCMP/ NUMCMP
COMMON /CURVP5/ NPIS, IND(10), DFP(10)
COMMON /MUCHT/ LBPN, HH, FFIN, SHPN, NSHN, VSUSN, PRHN, DPHN, M200N,
1 M201N, M203N
COMMON /WFLAG/ WFLAG
COMMON /COFFS/ C(2)

---VARIABLE AND FUNCTION TYPE DEFINITIONS AND DIMENSIONS---

INTEGER MCPM, NPIS, WFLAG
INTEGER FFIN, NSHN, PRHN
INTEGER NSSS(16), LNS
LOGICAL CALLP, LOCAL
LOGICAL MEXSHA, LOGVAL, LMOCV
REAL IND, DFP, C
REAL LBPN, HH, SHPN, VSUSN, DPHN, M200N, M201N, M203N
REAL M, M201S, M203P, M203B
REAL F, FF, RPM

---VARIABLE DATA DEFINITIONS---

DATA LNS/16/

DATA MCPM/0/
C BRANCH ACCORDING TO WHICH WEIGHT ITEM IS TO BE ESTIMATED.
C GO TO (100,100,300),WFLAG
C FOR MACHINERY WEIGHT ITEM W(200) AND W(201), FIT A STRAIGHT LINE TO
C THE DATA POINTS PROVIDED. OBTAIN THE SLOPE AND Y-INTERCEPT COEFFI-
C CIENTS.
C 100 CONTINUE
   DO 150 I=1,NPTS
      WRITE (10,7000) I,IND(I),DEP(I)
   150 CONTINUE
   7000 FORMAT(1X,110,1X,13.6,1X,13.6)
   CALL LIMIT(NPTS,IND,DEP,C)
C C ESTIMATE THE NEW SHIP WEIGHT.
C   W = C(1)*SHIP + C(2)
   IF (WFLAG.EQ.2) GO TO 200
   W = W10N - W
   GO TO 99999
200 CONTINUE
   W = W1N - W
   GO TO 99999
C C FOR W(203), FIRST ESTIMATE THE PROPELLER DIAMETER IF NOT SPECIFIED.
C 300 CONTINUE
   IF (DPHN.NE.0.) GO TO 320
   IF (NM.NE.0.) GO TO 8100
   IF (NSUM.EQ.0.) GO TO 310
   DPHN = 4.28*(IM**0.4283)
   GO TO 320
310 CONTINUE
   DPHN = 2.603*(IM**0.629)
C C ESTIMATE RPM.
C 320 CONTINUE
   IF (VSUSN.EQ.0.) GO TO 8200
   RPM = 96.12*(VSUSN/DPHN) + 52.15
C C ESTIMATE WEIGHT FOR SHAVING FOR FIXED PITCH AND CPP PROPPELLERS
C DEPENDING ON TYPE OF PROPULSION PLANT.
C   IF (PPIN.LT.4) = .36
   IF (PPIN.GT.4) = .20
   IF (PPIN.EQ.2) GO TO 330
C ESTIMATE PROPELLER WEIGHT FOR FP AND CRP TYPES.
C
340 CONTINUE
IF (PRTN.10.2) GO TO 345
W03=M0.001146*(DPRM**3.279)*NSHM
GO TO 350
345 CONTINUE
W03=M0.00114*(DPRM**3.128)*NSHM
C
C ESTIMATE REARING WEIGHT.
C
350 CONTINUE
W03=W2015+W203*W203
C
C ESTIMATE TOTAL W(203)
C
W03=W203+W203+W203
GO TO 99999
C
C ERROR MESSAGE. DRAFT OF NEW SHIP NOT SPECIFIED.
C
8100 CONTINUE
CALL STRPAK(MESS, LMS, 4HK), 55HA VALUE FOR NEW SHIP DRAFT IS NEEDED
ID. RETURN TO INPUT
CALL MESOUT(MESS)
CALL STRPAK(MESS, LMS, 4HK), 14H AND ENTER IT.
GO TO 99999
C
C ERROR MESSAGE. VSUS OF NEW SHIP NOT SPECIFIED.
C
8200 CONTINUE
CALL STRPAK(MESS, LMS, 4HK), 55HA VALUE FOR NEW SHIP SPEED IS NEEDED
ID. RETURN TO INPUT
CALL MESOUT(MESS)
CALL STRPAK(MESS, LMS, 4HK), 14H AND ENTER IT.
CALL MESOUT(MESS)
C
C RETURN TO CALLING PROGRAM
C
99999 CONTINUE
RETURN
END
C--------MACHINERY WEIGHT ESTIMATING MODULE SUBPROGRAM--------

SUBROUTINE OUTPUT(CALLAG, IFLAG)

C--------SUBPROGRAM DESCRIPTION--------

SUBROUTINE OUTPUT provides the user with a menu from which to
choose which module segment it is desired to operate next. The
choices are:

1. All module output variables
2. The module units to be used during input and output
3. The estimated machinery weights
4. The coefficients from linefitting
5. The units module allows the user to specify the length, force,
   and time units to be used during input and output. The length units
   are used for tip, propeller diameter, draft and possibly speed. The
   force units are used for weights and the time units may be used for
   speed. The user has the option of using 'knots' for speed
   even if 'nautical miles' and 'hour' are not the length and time
   units respectively for I/O.

C--------OUTPUT VARIABLES--------

CALLAG: if the input value of CALLAG is TRUE, and no error
          occurred when reading or editing an essential value.
          If the input value of CALLAG was FALSE, or an error
          occurred when reading or editing an essential value.

IFLAG: denoted the operation to be performed

1 if the user wishes to read the variables
2 edit
3 write

C--------LABELLED COMMON VARIABLES--------

LABELLED COMMON DIALOG and MODULI have been defined in subroutines

C--------SUBPROGRAMS AND FUNCTIONS CALLED--------

C DEX
C SIRPAK
C IMOVIC
C MNUM
C DEX LIBRARY
C NONE
C MODULI
C MODULI
C MODULI
C MODULI
C Labeled Commons

C COMM /DIALG/ MVERSE
C COMM /DIALG2/ NCPW
C
C Variable and Function Type Definitions and Definitions
C INTEGER IFLAG,NCPW,NWP
C INTEGER NITEM,NITEMS,TITEMS,TITEM,ITEM
C INTEGER MESS(16),LMS
C INTEGER READ(2),EDIT(2),WRITE(2)
C LOGICAL CALL1,LOCAL1
C LOGICAL MVERSE,LOGVAL,IMOVEC
C
C Variable Data Definitions
C DATA LMS/16/
C DATA MITEMS/4H01P,4H01 /
C DATA NITEMS/5/
C DATA ITEM/1H01, NH
C 1 4H01T1, NHMS
C 2 4H01T1, NHTMS
C 3 4H30E1, NhFCT
C 4 4H31ONE,AH /
C DATA READ /HHAD,AH,< /
C DATA EDIT /HHAD,AH,< /
C DATA WRIT /HHAD,AH,< /
C
C Activate the LOCAL All Option if the calling program requires it.
C Note that if LOCAL is set in this manner, Menu 'Output' is not
C defined by the invocation of this subroutine.
C LOCAL=CALL1
C 11 (LOCAL) GO 10 200
C
C Prepare a prompting message for Menu 'Output' and then provide the
C menu to the user.
C
C 5 CONTINUE
C 11 if (MVERSE) GO 10 40
C CALL STRPAK(MESS,LMS,NHC,4H01) SECT WHICH OUTPUT VARIABLE SEGMENT
C 11 TO 4
C 10 GO 10 (10,20,30), IFLAG
C 10 LOGVAL=IMOVEC(READ,1,6,NCPW,MESS,41,NCPW)
C 10 GO 10 50
C 20 LOGVAL=IMOVEC(EDIT,1,6,NCPW,MESS,41,NCPW)
C 20 GO 10 50
C 30 LOGVAL=IMOVEC(WRITE,1,7,NCPW,MESS,41,NCPW)
C 30 GO 10 50
40 CONTINUE
   CALL STRPAK(MESS, LNS, NRC, 221) WHICH OUTPUT SEGMENT?
50 CONTINUE
   ITEM REQURG(MN, MM, MT, I, MESS)
   GO TO (100, 200, 300, 400, 500), ITEM
C SET THE OUTPUT ALL OPTION.
C 100 CONTINUE
   LOCALL = .TRUE.
C READ, EDIT OR WRITE THE INPUT/OUTPUT MODULE UNITS.
C 200 CONTINUE
   CALL MWRITI(LOCALI, IFLAG)
   IF (LOCALI) GO TO 300
   IF (.NOT.CALAI) GO TO 5
   CALAI = .FALSE.
   GO TO 500
C CALL MWRITI TO READ, EDIT OR WRITE THE ESTIMATED WEIGHT ITEM.
C 300 CONTINUE
   MUPIT 0
   CALL MWRITI(LOCALI, IFLAG, MUPIT)
   IF (LOCALI) GO TO 400
   IF (.NOT.CALAI) GO TO 5
   CALAI = .FALSE.
   GO TO 500
C CALL MWRITI TO READ, EDIT OR WRITE THE LINE EQUATION COEFFICIENTS.
C 400 CONTINUE
   CALL MWRITI(LOCALI, IFLAG)
   IF (LOCALI) GO TO 500
   IF (.NOT.CALAI) GO TO 5
   CALAI = .FALSE.
C RETURN TO CALLING PROGRAM.
C 500 CONTINUE
   RETURN
END
C--------MACHINERY WEIGHT ESTIMATING MODULE SUBPROGRAM--------

C SUBROUTINE MACHT (ALLFG, IOFLG)

C--------SUBPROGRAM DESCRIPTION--------

C COEFFICIENTS OF THE EQUATION OF A STRAIGHT LINE.

C--------SUBPROGRAM ASSUMPTIONS--------

C NEX Y1

C--------OUTPUT VARIABLES--------

C ALLFG: .TRUE. IF THE INPUT VALUE OF ALLFG WAS .TRUE. AND NO ERROR
C OCCURRED WHEN READING OR EDITING A COEFFICIENT
C .FALSE. IF THE INPUT VALUE OF ALLFG WAS .FALSE. OR AN ERROR

C--------INPUT VARIABLES--------

C ALLFG: .TRUE. IF THE ALL OPTION OF THE CALLING PROGRAM IS ACTIVE
C .FALSE. IF THE ALL OPTION OF THE CALLING PROGRAM IS NOT ACTIVE

C IOFLG: DENOTES THE OPERATION TO BE PERFORMED
C 1 IF THE USER WISHES TO READ A COEFFICIENT
C 2 IF THE USER WISHES TO WRITE A COEFFICIENT

C--------LABELLED COMMON VARIABLES--------

C LABELLED COMMON DIALOG/MINCPW/INDOUT, AND REROS HAVE BEEN DEFINED IN
C SUBROUTINE MAINPG.

C SUBROUTINE MAINPG.

C --------COEFFICIENTS INITIALIZED IN BLOCK DATA
C
C C : A TWO-ELEMENT ARRAY WHICH CONTAINS THE SLOPE AND Y-INTERCEPT 
C RESPECTIVELY OF AN EQUATION OF A STRAIGHT LINE
C
C C : CoORD INITIALIZED IN BLOCK DATA
C
C CIENAM: THE DATABASE NAME OF THE ARRAY CONTAINING THE COEFFICIENTS
C OF THE EQUATION OF A STRAIGHT LINE
C CIENAM: THE DATABASE COMMENT OF THE COEFFICIENT ARRAY
C COTERM: THE FORMAT TO BE USED WHEN READING THE ARRAY FROM OR WRITING
C IT TO A SEQUENTIAL FILE
C DEFC : AN ARRAY WHICH CONTAINS THE DEFAULT VALUES OF THE COEFFICIENTS
C NDEFC : THE NUMBER OF DEFAULT VALUES
C
C--------SUBPROGRAMS AND FUNCTIONS CALLED--------

C DEX
C
C NEX
C DEX LIBRARY
C RAILDR
C REROS
C HARDP
C MODUL
C NEX

C--------LABELLED COMMONS--------

C COMMON /DIALG/ MILKSE
C COMMON /MINCPW/ NCWP
C COMMON /INDOUT/ IMODE,OMODE
COMMON /KHLINS/ RNHFILE, RNHFIL
COMMON /CMTS/ C(2)
COMMON /COFINO/ CFNAME, CFCMNT, COFORM, DFC(2), NDFC
C
C VARIABLE AND FUNCTION TYPE DEFINITIONS AND DIMENSIONS
C
INITI: 1010L, INODE, UMONI, NCMY, RNHFIL, RNHFILE.
INTEGER CFNAME(2), CFCMNT(16), COFORM(2), PME(16), UNIINM(3)
INTEGER ENDM, N10, NHT1, CHT, CFORM
LOGICAL ALITERAL, MITERSE, LOGVAL
LOGICAL RAILDIR, RAILDT, RAROMP
REAL C, UNITIM, UNITTA, DFC
C
C VARIABLE DATA DEFINITIONS
C
DATA UNITIM/UNITIMU, .94/, .4F /
DATA CFORM, N10, NHT1, CMGT1, CG01 /1, 2, 3, 2/
DATA UNITIM, UNITTA /1, 0, 0, 0/
C
C BRANCH ACCORDING TO THE OPERATION TO BE PERFORMED
C
GO TO (100, 200, 300), 1010L
C
C READ THE COEFFICIENTS.
C
100 CONTINUE
LOGVAL-RAILDIR(C, ALLFG, CG01,
1) MITERSE, INODE, NCMY,
2) CFNAME, CMGT1, UNITIM, UNITTA, UNIINM, .TRUE.,
3) .TRUE., PME, CFCMNT,
4) RNHFIL, COFORM,
5) NDFC, DFC)
GO TO 99999
C
C EDIT COEFFICIENTS.
C
200 CONTINUE
LOGVAL-RALEDFG(C, ALLFG,
1) CFNAME, CFROM, N10,
2) UNITIM, UNITTA, UNIINM, .TRUE.,
3) .TRUE., PME, CFCMNT,
4) RNHFIL, COFORM,
5) NDFC, DFC)
GO TO 99999
C
C WRITE COEFFICIENTS.
C
300 CONTINUE
LOGVAL-RAROMP(ALLFG,
C------------------MACHINERY WEIGHT ESTIMATING MODULE SUBPROGRAM-----------------

BLOCK DATA

C------------------SUBPROGRAM DESCRIPTION------------------

C THIS SUBPROGRAM Initializes VARIABLES IN THE Labeled COMMON BLOCKS OF

C THIS MODULE.

C EACH Labeled COMMON AND ALL RELATED STATEMENTS AND DEFINITIONS ARE

C LISTED UNDER THE SUBPROGRAM NAME WHERE THE COMMON FIRST APPEARS.

C------------------

C------------------SUBROUTINE MAINPG------------------

COMMON /DATA/ MIRSE
COMMON /MC/ NCMP
COMMON /NCONF/ NCMP
COMMON /INQUIRE/ MIRSE, MIRSE
COMMON /NEPOS, NINFIL, NINFIL
INTEGER MCMP, MIRSE, MIRSE, MIRSE, MIRSE, MIRSE
LOGICAL MIRSE
DATA NCMP/N
DATA MIRSE, MIRSE, MIRSE
DATA INQUIRE, INQUIRE
DATA NINFIL, NINFIL
DATA NINFIL
C------------------SUBROUTINE LUNT1------------------

COMMON /LUNT1/ PSTUN, UTRUN
COMMON /LUNT1O/ DBLUN, DBLUN, UTRUN, DBLUN
INTEGER PSTUN, UTRUN, UTRUN, UTRUN
INTEGER DBLUN, DBLUN, DBLUN, UTRUN, DBLUN
DATA PSTUN, UTRUN
DATA UTRUN, DBLUN
DATA DBLUN, DBLUN, DBLUN, UTRUN
DATA DBLUN, DBLUN, DBLUN, UTRUN, DBLUN
DATA DBLUN, DBLUN, DBLUN, UTRUN, DBLUN
1
2
3
DATA LUNRM, LUNRM
DATA DUTFUN
C------------------SUBROUTINE LUNT1------------------

COMMON /LUNT1/ PSTUN, UTRUN
COMMON /LUNT1O/ DBLUN, DBLUN, DBLUN, DBLUN
INTEGER PSTUN, UTRUN, UTRUN, UTRUN
INTEGER DBLUN, DBLUN, DBLUN, DBLUN
DATA PSTUN, UTRUN
DATA UTRUN, DBLUN
DATA DBLUN, DBLUN, DBLUN, DBLUN
DATA DBLUN, DBLUN, DBLUN, DBLUN
DATA DBLUN, DBLUN, DBLUN, DBLUN
1
2
3
DATA LUNRM, LUNRM
DATA DUTFUN
C. SUBROUTINE UNIT

COMMON /UNIT1/ PTIUN,UNIT1
COMMON /UNIT0/ DBFUNN,DFUNC,UNITHM,DFITUN
INTEGER PSITUN,UNIT1
INTEGER DFUNIT(2),DFUNC(16),UNITHM(2),DFITUN
DATA PSITUN/h,
DATA UNIT1/h,
DATA DBFUNN/NU1OF,NU1UN/, (DATA DBFUN/NU1OF,NU1UN/,NU1UN,NU1UN)
1
2
3
DATA UNITHM/NU1(10,4H)/
DATA DEITUN/h,

C. SUBROUTINE MACHINE

COMMON /VARUN/ UNITM,RSCTUN
COMMON /UNIT1/ DFUNIT,CM12,DE12
COMMON /UNIT0/ UNITM,CM15,DE15
COMMON /UNIT0/ PFINAM,CM19,DE19
COMMON /UNIT0/ SHFINAM,CM20,DE120
COMMON /UNIT1/ NSHFINAM,CM21,DE21
COMMON /UNIT1/ VSUSNM,CM24,DE24
COMMON /UNIT2/ PRPNAM,CM27,DE27
COMMON /UNIT0/ DPPRPNAM,CM28,DE28
COMMON /UNIT0/ M200NM,CM200,DE200
COMMON /UNIT1/ M201NM,CM201,DE201
COMMON /UNIT1/ M202NM,CM202,DE203
COMMON /UNIT1/ M203NM,CM203,DE203
COMMON /UNIT1/ DBPNAM,SHORTM,SHORTM,SHORTM,SHORTM,SHORTM,SHORTM,SHORTM
COMMON /UNIT1/ VSUSNM,SHORTM,SHORTM,SHORTM,SHORTM,SHORTM,SHORTM,SHORTM

INTEGER DFUNIT(2),RSCTUN(2)
INTEGER DFUNIT(2),SHORTM(2),PFINAM(2),SHFINAM(2),VSUSNM(2),DBPNAM(2),M200NM(2),M201NM(2),M202NM(2),M203NM(2)

INTEGER CM12(16),CM15(16),CM19(16),CM20(16),CM21(16),CM24(16),CM27(16),CM28(16),CM200(16),CM201(16),CM202(16),CM203(16)

INTEGER PFINAM,SHORTM,SHORTM
INTEGER DFIT,DE21,DE22
REAL DF2,DE5,DE20,DE24,DE28,DE200,DE201,DE203

DATA UNITHM/NU1(10,4H)/
DATA DEITUN/h,
DATA UNITM/NU1OF,NU1UN/, (DATA DBFUN/NU1OF,NU1UN/,NU1UN,NU1UN)
1
2
3
DATA UNITHM/NU1(10,4H)/
DATA DEITUN/h,
DATA NPTS/0/
DATA IND(1), IND(2), IND(3), IND(4), IND(5), IND(6), IND(7),
1   IND(8), IND(9), IND(10)
2   0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
DATA DEP(1), DEP(2), DEP(3), DEP(4), DEP(5), DEP(6), DEP(7),
1   DEP(8), DEP(9), DEP(10)
2   0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
C. SUBROUTINE NANCOF
COMMON /CSTTS/ C(2)
COMMON /CSTTS/ C(2), C1STTS(C), C1STTS(1), C1STTS(2), NDEFC
INTEGER C1STTN(2), C1STTN(1), C1STTN(2), NDEFC
REAL C, DEFC
DATA C(0, 0.0). /.
DATA C1STTN, C1STTS(1), C1STTS(2), NDEFC
DATA C1STTS(1), C1STTS(2), NDEFC
1   0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
2   NDEFC, DEFC(1), DEFC(2) /2.0, 0.0234, 48.09/
C----------MACHINERY WEIGHT ESTIMATING MODULE SUBPROGRAM----------
SUBROUTINE LIMFIT(N,X,Y,C)
C----------SUBPROGRAM DESCRIPTION----------
C LIMFIT CALCULATES THE COEFFICIENTS FOR A STRAIGHT LINE FIT OF DATA POINTS. THE COEFFICIENTS A AND B AND X AND Y ARE DEFINED AS:
C Y = A*X + B
C THE VALUES OF A AND B ARE RETURNED TO THE CALLING PROGRAM AS THE FIRST AND SECOND ILLIMINS RESPECTIVELY OF ARRAY C.
C----------SUBPROGRAM ASSUMPTIONS----------
CARIABLE YLI
C----------OUTPUT VARIABLES----------
C C : AN ARRAY CONTAINING THE VALUES OF THE SLOPE (A) AND THE Y-INTERCEPT (B) OF THE LINE
C----------INPUT VARIABLES----------
C N : THE NUMBER OF PAIRS OF DATA POINTS
C X : AN ARRAY CONTAINING THE ABSCISSAS OF THE DATA POINTS
C Y : AN ARRAY CONTAINING THE ORDINATES OF THE DATA POINTS
C----------VARIABLE DEFINITIONS AND DIMENSIONS----------
INTEGER N
REAL X(N),Y(N),C(2)
REAL A,B,SMAX,SMAX1,SMAX2,SUMY1,SMXY1
REAL S0,SMAXY,SMAX,SMAY
C----------INITIALIZE VARIABLES----------
SMAX1=0.
SMAX2=0.
SUMY1=0.
S0=SMXY1=0.
C----------CALCULATE THE COEFFICIENTS A AND B----------
DO 10 I=1,N
SMY1=SUMY1*X(I)
SMXY1=SMXY1+X(I)*Y(I)
SMXY12=SMXY12+X(I)*Y(I)
SUMY1=SUMY1+Y(I)
SMXY12=SMXY12+Y(I)*Y(I)
10 CONTINUE
B = (SMXY12*SUMY1 - SMXY1)/(SMXY12 - SMXY1)
A = (SUMY1 - B*SMAX)/SMAX
C----------
RETURN
END