NOISE AND SONIC BOOM IMPACT TECHNOLOGY

BOOMAP2 Computer Program for Sonic Boom Research:
Program Maintenance Manual

Volume III of III Volumes

Philip J. Day
Thomas M. Reilly
Harry Seidman

BEN Laboratories, Incorporated
21120 Vanowen Street
Canoga Park, CA 91303

August 1988

Final Report for Period July 1986 - November 1987

Approved for public release; distribution is unlimited.

Noise and Sonic Boom Impact Technology Program
Systems Acquisition Division
Human Systems Division
Brooks Air Force Base, TX 78235-5000
NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility nor any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise as in any manner construed, as licensing the holder, or any other person or corporation; or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed and it is releasable to the National Technical Information Service (NTIS), where it will be available to the general public, including foreign nationals.

This report has been reviewed and is approved for publication.

GERALD L. LONG, LT COL, USAF
NSBIT Program Manager

FOR THE COMMANDER

MICHAEL G. MACNAUGHTON, COL, USAF
Deputy Commander Development & Acquisition

Please do not request copies of this report from the Human Systems Division. Copies may be obtained from DTIC. Address your request for additional copies to:

Defense Technical Information Center
Cameron Station
Alexandria VA 22301-6145

If your address has changed, if you wish to be removed from our mailing list, or if your organization no longer employs the addressee, please notify NSBIM/SORT, Brooks AFB TX 78235-5000, to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.
Air Combat Maneuvering Instrumentation/Tactical Air Combat Training Systems (ACMI/TACTS) are used at several Military Operating Areas (MOA) in the United States and abroad as a post-flight pilot debriefing aid in training for air-to-air combat. Engineering flight data are acquired and recorded from several radar facilities simultaneously during flights in appropriately instrumented MOAs. These data are used to generate the information required for subsequent graphical replays of the aircraft position, airspeed, g-value, attitude, climb/dive angle, etc. of the training sorties at post-flight debriefings. The BCOMAP2 and MOAPS computer programs analyze noise from supersonic aircraft operations by extracting information from the ACMI/TACTS computer tapes. The MOAPS program extracts information from a TACTS ACII mission standard data tape and compiles a computer library of information concerning the supersonic operations. The BCOMAP2 program utilizes the library produced by the MOAPS program. The program calculates various statistics on the supersonic operations, and calculates expected sonic boom levels on the ground based on the extracted information.
19. Continued

information, BOCMAP2 can: (1) generate various spatial/temporal distribution statistics; (2) interface with sonic boom generation and propagation models; (3) calculate the intensity and location of sonic booms reaching the ground; and (4) provide the data file used by commercial graphical software package, GCP, to plot contours of boom exposure in units of average peak overpressure or C-weighted day-night average sound level (CINL).

These two programs, when used with an adequate library of aircraft sorties from Military Operating Areas, can be an invaluable tool for environmental planning purposes to predict boom intensity, frequency, and distribution.

This program maintenance manual provides computer programming personnel with the information to maintain the BOCMAP2 program software developed under this contract. The BOCMAP2 program utilizes a sophisticated acoustic ray theory model for predicting the sonic boom overpressures and noise levels on the ground. The model is a modified version of the TRAPS computer program earlier developed by Dr. Albion Taylor. This BOCMAP2 program replaces the earlier BOCM-MAP program, which could not provide accurate predictions of the booms resulting from non-steady supersonic aircraft flight.
The BOOMAP2 computer program is the result of efforts by several individuals. In particular, the authors of the maintenance manual would like to thank Mr. Dwight Bishop, Ms. Emma Wilby, and Mr. Jerold Haber for their technical assistance.

The support and encouragement of the NSBIT Technical Staff is also gratefully acknowledged, as is the continuing support by Mr. Jerry D. Speakman of the Biodynamics and Bionics Division, Aerospace Medical Research Laboratory, Wright-Patterson AFB.

We also thank Ms. Dorothy Miller and Mr. Kenneth Williams of Tyndall AFB for their aid in helping with the installation of the code on the local Cyber computer.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREFACE</td>
<td>i</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>v</td>
</tr>
<tr>
<td>1.0 GENERAL</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Purpose</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Program History and Overview</td>
<td>1</td>
</tr>
<tr>
<td>1.3 Terms and Abbreviations</td>
<td>4</td>
</tr>
<tr>
<td>2.0 SYSTEM DESCRIPTION</td>
<td>7</td>
</tr>
<tr>
<td>2.1 Functional Overview</td>
<td>7</td>
</tr>
<tr>
<td>2.2 Relationship and Description of Subroutines</td>
<td>7</td>
</tr>
<tr>
<td>2.2.1 Main Program</td>
<td>7</td>
</tr>
<tr>
<td>2.2.1.1 BOOMAP2</td>
<td>7</td>
</tr>
<tr>
<td>2.2.1.2 RNGALT</td>
<td>7</td>
</tr>
<tr>
<td>2.2.1.3 RNGLL</td>
<td>10</td>
</tr>
<tr>
<td>2.2.1.4 SOUND</td>
<td>10</td>
</tr>
<tr>
<td>2.2.1.5 STOREC</td>
<td>10</td>
</tr>
<tr>
<td>2.2.2 Input Routines</td>
<td>10</td>
</tr>
<tr>
<td>2.2.2.1 LEXPACK</td>
<td>10</td>
</tr>
<tr>
<td>2.2.2.1.1 GETLINE</td>
<td>10</td>
</tr>
<tr>
<td>2.2.2.1.2 GETCHR</td>
<td>10</td>
</tr>
<tr>
<td>2.2.2.1.3 ADDCHR</td>
<td>12</td>
</tr>
<tr>
<td>2.2.2.1.4 GETOKEN</td>
<td>12</td>
</tr>
<tr>
<td>2.2.2.2 PRSPACK</td>
<td>12</td>
</tr>
<tr>
<td>2.2.2.2.1 LDATE</td>
<td>12</td>
</tr>
<tr>
<td>2.2.2.2.2 LTIME</td>
<td>13</td>
</tr>
<tr>
<td>2.2.2.2.3 LOOKUP</td>
<td>13</td>
</tr>
<tr>
<td>2.2.2.2.4 PARSE</td>
<td>13</td>
</tr>
<tr>
<td>2.2.2.3 SCHPACK</td>
<td>13</td>
</tr>
<tr>
<td>2.2.2.3.1 FILBUF</td>
<td>13</td>
</tr>
<tr>
<td>2.2.2.3.2 STRMCH</td>
<td>14</td>
</tr>
<tr>
<td>2.2.2.3.3 INTMCH</td>
<td>14</td>
</tr>
<tr>
<td>2.2.2.3.4 GETREC/GETINX</td>
<td>14</td>
</tr>
<tr>
<td>2.2.3 Ray Tracing Driver Routines</td>
<td>14</td>
</tr>
<tr>
<td>2.2.3.1 RTTRACE</td>
<td>14</td>
</tr>
<tr>
<td>2.2.3.2 SPLINE</td>
<td>16</td>
</tr>
<tr>
<td>2.2.3.3 GETSEG</td>
<td>16</td>
</tr>
<tr>
<td>2.2.3.4 SCREEN</td>
<td>16</td>
</tr>
<tr>
<td>2.2.3.5 SORTPHI</td>
<td>16</td>
</tr>
<tr>
<td>2.2.3.6 STORE</td>
<td>16</td>
</tr>
<tr>
<td>2.2.3.7 SIDAT</td>
<td>17</td>
</tr>
<tr>
<td>2.2.3.8 EXTRPR</td>
<td>17</td>
</tr>
<tr>
<td>2.2.3.9 FFUNC</td>
<td>17</td>
</tr>
<tr>
<td>2.2.3.10 LSQUAR</td>
<td>17</td>
</tr>
<tr>
<td>2.2.3.11 GAUSJR</td>
<td>17</td>
</tr>
<tr>
<td>2.2.3.12 SORT</td>
<td>18</td>
</tr>
<tr>
<td>2.2.3.13 RBRAYS</td>
<td>18</td>
</tr>
</tbody>
</table>
2.2.4  Focal Zone Calculation Routines  

<table>
<thead>
<tr>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2.4.1  FOCMAP</td>
<td>18</td>
</tr>
<tr>
<td>2.2.4.2  CSTGND</td>
<td>18</td>
</tr>
<tr>
<td>2.2.4.3  FOCUS</td>
<td>18</td>
</tr>
<tr>
<td>2.2.4.4  FOCAL</td>
<td>20</td>
</tr>
<tr>
<td>2.2.4.5  INTERP</td>
<td>20</td>
</tr>
<tr>
<td>2.2.4.6  GETDILT</td>
<td>20</td>
</tr>
</tbody>
</table>

2.2.5  TRAPS Input Routines  

<table>
<thead>
<tr>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2.5.1  SETUP</td>
<td>20</td>
</tr>
<tr>
<td>2.2.5.2  ATMSIN</td>
<td>20</td>
</tr>
<tr>
<td>2.2.5.3  PTDHIN</td>
<td>20</td>
</tr>
<tr>
<td>2.2.5.4  RACOBWK</td>
<td>22</td>
</tr>
<tr>
<td>2.2.5.5  WINDIN</td>
<td>22</td>
</tr>
</tbody>
</table>

2.2.6  TRAPS Ray Tracing Routines  

<table>
<thead>
<tr>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2.6.1  ACMOVE</td>
<td>23</td>
</tr>
<tr>
<td>2.2.6.2  TACMOV</td>
<td>23</td>
</tr>
<tr>
<td>2.2.6.3  FILIMS</td>
<td>23</td>
</tr>
<tr>
<td>2.2.6.4  RAYORG</td>
<td>25</td>
</tr>
<tr>
<td>2.2.6.5  RAYTRK</td>
<td>25</td>
</tr>
<tr>
<td>2.2.6.6  RATES</td>
<td>25</td>
</tr>
<tr>
<td>2.2.6.7  ADVANS</td>
<td>25</td>
</tr>
<tr>
<td>2.2.6.8  RCRVIT</td>
<td>26</td>
</tr>
<tr>
<td>2.2.6.9  RECORD</td>
<td>26</td>
</tr>
<tr>
<td>2.2.6.10 ARTUBE</td>
<td>26</td>
</tr>
<tr>
<td>2.2.6.11 DIST</td>
<td>26</td>
</tr>
<tr>
<td>2.2.6.12 RCSPCL</td>
<td>26</td>
</tr>
</tbody>
</table>

2.2.7  Signature Aging Routines  

<table>
<thead>
<tr>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2.7.1  SIGNUR</td>
<td>28</td>
</tr>
<tr>
<td>2.2.7.2  FREAD</td>
<td>28</td>
</tr>
<tr>
<td>2.2.7.3  AGING</td>
<td>28</td>
</tr>
<tr>
<td>2.2.7.4  HILBRT</td>
<td>28</td>
</tr>
<tr>
<td>2.2.7.5  CPVAL</td>
<td>29</td>
</tr>
<tr>
<td>2.2.7.6  SORTEM</td>
<td>29</td>
</tr>
<tr>
<td>2.2.7.7  SIGPRT</td>
<td>29</td>
</tr>
<tr>
<td>2.2.7.8  TSGPRT</td>
<td>29</td>
</tr>
<tr>
<td>2.2.7.9  FOCALP</td>
<td>29</td>
</tr>
<tr>
<td>2.2.7.10 CALSEL</td>
<td>30</td>
</tr>
<tr>
<td>2.2.7.11 FFT</td>
<td>30</td>
</tr>
<tr>
<td>2.2.7.12 CURVE</td>
<td>30</td>
</tr>
<tr>
<td>2.2.7.13 FINDT</td>
<td>30</td>
</tr>
</tbody>
</table>

2.2.8  Utility Routines  

<table>
<thead>
<tr>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2.8.1 AIR</td>
<td>32</td>
</tr>
<tr>
<td>2.2.8.2 PHLELV</td>
<td>32</td>
</tr>
<tr>
<td>2.2.8.3 PHAZIM</td>
<td>32</td>
</tr>
<tr>
<td>2.2.8.4 EAMENU</td>
<td>32</td>
</tr>
<tr>
<td>2.2.8.5 UNITS</td>
<td>32</td>
</tr>
<tr>
<td>2.2.8.6 TIMCVR</td>
<td>33</td>
</tr>
<tr>
<td>2.2.8.7 GETLYR</td>
<td>33</td>
</tr>
<tr>
<td>2.2.8.8 FNDLYR</td>
<td>33</td>
</tr>
<tr>
<td>2.2.8.9 DOTP</td>
<td>33</td>
</tr>
<tr>
<td>2.2.8.10 RNORM</td>
<td>33</td>
</tr>
<tr>
<td>2.2.8.11 CROSS</td>
<td>33</td>
</tr>
<tr>
<td>2.2.8.12 UNIT</td>
<td>34</td>
</tr>
<tr>
<td>2.2.8.13 MDOT</td>
<td>34</td>
</tr>
</tbody>
</table>
2.2.8.14 SAVRAY ............. 34
2.2.8.15 SPTRAY ............. 34
2.2.8.16 SORTRY ............. 34
2.2.8.17 DDOTP ............. 34
2.2.8.18 DRNORM ............. 34
2.2.8.19 DCROSS ............. 35
2.2.8.20 DUNIT ............. 35

2.2.9 Graphics Routines ............. 35
2.2.9.1 STOREC ............. 35
2.2.9.2 PLOTDR ............. 35
2.2.9.3 SCRFL ............. 35
2.2.9.4 CPBMTR ............. 37
2.2.9.5 CPSSTR ............. 37
2.2.9.6 CPINIT ............. 37
2.2.9.7 CTERM ............. 37

2.2.10 Contouring Routines ............. 37
2.2.10.1 CONTUR ............. 39
2.2.10.2 CPCONT ............. 39
2.2.10.3 GRIDPW ............. 39
2.2.10.4 DIVARR ............. 39

2.2.11 Scratch Pad Routines ............. 40
2.2.11.1 SCRPAD ............. 40
2.2.11.2 CCONVL ............. 40
2.2.11.3 FNDCNT ............. 40
2.2.11.4 CONPTS ............. 40
2.2.11.5 PSETUP ............. 42
2.2.11.6 PLOTIT ............. 42
2.2.11.7 CLSPLT ............. 42

3.0 ENVIRONMENT ............. 43
3.1 Hardware ............. 43
3.2 Database ............. 43

  3.2.1 Input Index and Library Files ............. 43
  3.2.1.1 Index File, 'INDEX' ............. 43
  3.2.1.2 Library File, 'LIBRY' ............. 43

  3.2.2 Output Index and Library Files ............. 44
  3.2.2.1 Ray Index File, 'CINDEX' ............. 44
  3.2.2.2 Ray Library File, 'CLIBRY' ............. 45

4.0 PROGRAM MAINTENANCE PROCEDURES ............. 49
4.1 Conventions ............. 49
4.2 Verification Techniques ............. 49
4.3 Updating Site and Aircraft Information ............. 49

REFERENCES ............. 51

Appendix A: PROGRAM LISTINGS
## LIST OF FIGURE

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. FUNCTIONAL RELATIONSHIP BETWEEN ELEMENTS OF BOOMAP2 COMPUTER PROGRAM</td>
<td>5</td>
</tr>
<tr>
<td>2. OVERVIEW OF BOOMAP2 ROUTINES</td>
<td>8</td>
</tr>
<tr>
<td>3. MAIN PROGRAMS</td>
<td>9</td>
</tr>
<tr>
<td>4. INPUT ROUTINES</td>
<td>11</td>
</tr>
<tr>
<td>5. RAY TRACING DRIVER ROUTINES</td>
<td>15</td>
</tr>
<tr>
<td>6. FOCAL ZONE CALCULATION ROUTINES</td>
<td>19</td>
</tr>
<tr>
<td>7. TRAPS INPUT ROUTINES</td>
<td>21</td>
</tr>
<tr>
<td>8. TRAPS RAY TRACING ROUTINES</td>
<td>24</td>
</tr>
<tr>
<td>9. TRAPS SIGNATURE AGING ROUTINES</td>
<td>27</td>
</tr>
<tr>
<td>10. UTILITY ROUTINES</td>
<td>31</td>
</tr>
<tr>
<td>11. GRAPHICS ROUTINES</td>
<td>36</td>
</tr>
<tr>
<td>12. CONTOURING ROUTINES</td>
<td>38</td>
</tr>
<tr>
<td>13. SCRATCH PAD ROUTINES</td>
<td>41</td>
</tr>
</tbody>
</table>
1.0 GENERAL

1.1 Purpose

The objective of writing this Program Maintenance Manual is to provide the maintenance programmer personnel with the information necessary to effectively maintain the software.

1.2 Program History and Overview

The major purpose of the BOOMAP2 and the accompanying MOAOPS programs is to extract and analyze information from the Tactical Air Crew Combat Training System/Air Combat Maneuvering Instrumentation (TACTS/ACMI) system installed at various combat training military operating areas. This information is then used to predict the location and magnitude of sonic boom overpressures on the ground in the vicinity of supersonic flights.

Real time flight information is transmitted to the TACTS/ACMI systems on the ground. Among the data is real time information on aircraft position, velocity and acceleration, updated at intervals of 100 to 200 milliseconds. The MOAOPS program extracts these data for the sonic boom analysis from the tapes at approximately 1.5 second intervals in order to minimize both the time taken to read the tapes and the quantity of information to be stored.

The MOAOPS program is in two parts: a data extraction program EXTRCT, and an index deletion and modification program DELETE. The data extraction program reads the ACMI tapes, extracting relevant information and appending this information to either a new or existing database. This library file accumulates
the information from all the mission tapes analyzed. The library file is indexed so that a particular mission, aircraft type, etc. can be accessed by the sonic boom analysis programs.

The BOOMAP2 data analysis program accesses the MOAOPS library tapes as selected by the user. The data analysis program produces statistical and graphical output describing the aircraft positions parameters as various measures of predicted boom strength. The BOOMAP2 program produces tabular output of various statistics that are sent directly to a line printer. In addition, for those situations where focused sonic booms are produced, individual plots of the maximum overpressures together with other technical information are produced in the form of a "scratch pad". These "scratch pads" can be plotted for each situation in which focused booms occur.

When a mission is selected from the MOAOPS Library and used as input to the BOOMAP2 computer program, the rays traced by BOOMAP2 are saved in the RAYS Library. If that same mission is then selected at a future time, the necessary ray information is recalled from the library thus saving substantial computer time.

To produce graphic output, BOOMAP2 creates a file which is compatible with California Computer Products' (CALCOMP) General Purpose Contouring Program (GPCP II) (Reference 1). GPCP II reads this file and generates the necessary plotter directives to produce hard copy graphic output.

The user controls the database subset to be extracted from the MOAOPS Library through the use of an input data file. Through this file, the user specifies: a) the name(s) of the MOA ranges to be considered; b) mission names or dates; c) bounding times of day; d) aircraft types (specific tail numbers optional).
Users also specify the desired output products. These include:

1. A statistical summary of position, speed, and boom strength variables. This summary includes distribution functions of range x-coordinates and y-coordinates, and the aircraft z-coordinate (height above the range), all in feet. It also includes a distribution function of effective height ($h_e$). Distribution functions of Mach number, cutoff Mach number, and effective Mach number are also presented. Estimated boom strength distribution functions include peak overpressure (in pounds per square foot), the peak overpressure (in dB, re: 20 microPascals), the C-weighted sound exposure level (in dB), and the A-weighted sound exposure level (in dB). The estimated boom strength are those calculated directly below the extended aircraft flight trajectory using Carlson’s Simplified Sonic Boom Prediction Model. Also included are root mean square values for effective height, Mach number, effective Mach number, and cutoff Mach number.

2. A flight track map depicting ground projections of flight paths during supersonic activity.

3. A flight track map depicting ground projections of flight paths during sonic boom producing activity.

4. A noise contour map of average C-weighted sound exposure levels (CSEL).

5. A noise contour map of C-weighted day-night average levels (CLDN). The map requires input of the reference number of daytime operations which is used to convert CSEL to CLDN.

6. A noise contour map of flight-averaged peak over pressures in pounds per square foot, in OASPL or CSEL.
7. A map showing geographic location of maximum overpressures due to focused sonic booms.

The functional relationship between major program elements is shown in Figure 1. Information on executing the MOAOPS programs can be found in Reference 2. This manual discusses the maintenance of the BOOMAP2 program and associated graphics packages. A technical discussion of the algorithms is provided in Reference 3. A user’s and computer operator’s manual is provided in Reference 4.

1.3 Terms and Abbreviations

In this report, overpressure will typically mean the "magnitude" of the sonic boom at a given point expressed in terms of the maximum overpressure in pounds per square foot (psf) or in terms of the overall sound pressure level (OASPL) in dB, or in terms of the C-weighted sound exposure level (CSEL) in dB. Program options allow a choice of either of these three metrics for the contour presentations.

Except as noted in the separate routines all units are in meters, seconds, degrees, and pascals. Atmospheric pressures in the traps routines are millibars.
Figure 1. Functional Relationship Between Elements of BOOMAP2 Computer Program
2.0 SYSTEM DESCRIPTION

2.1 Functional Overview (Figure 2)

The main routine, BOOMAP2, is responsible for the input of user directives and flight information from the MOAOPS library. It also outputs a file with the directives for the plotting routines. Although the plotting routines require a separate job step, BOOMAP2 must be run in order to create this file. The ray tracing routines are called from BOOMAP2. These consist of routines to trace the actual rays, calculate the location of a focus, and calculate the aging of the signatures.

2.2 Relationship and Description of Subroutines

2.2.1 Main Program (Figure 3)

2.2.1.1 BOOMAP2

BOOMAP2 calls the parser, which interprets the user directives, and processes the information accordingly. It calls the routines that set up the tables which are necessary for the TRAPS routines. It then reads in a flight track and outputs those portions which are supersonic (Unit 3) and which are boom producing (Unit 4) for the plotting routines. It then calls RTRACE to calculate the overpressures on the ground. BOOMAP2 also does a statistical summary of all of the flights requested in one run. Units are feet, and psf.

2.2.1.2 RNGALT

Given a site, RNGALT returns the mean altitude of that site. Units are feet.
Figure 2. Overview of BOOMAP2 Routines
Figure 3. Main Programs

- BOOMAP2
  - STOREC
  - SOUND
  - RNGLL
  - RNGALT
  - RTRACE
  - SETUP
  - OPFIND
  - GETREC
  - PARSE
2.2.1.3 RNGLL

RNGLL, when supplied with a site, returns the latitude and longitude.

2.2.1.4 SOUND

SOUND returns the local speed of sound in feet/second at a given altitude. It uses the utility routines FNDLYR and GETLYR.

2.2.1.5 STOREC

STOREC saves the data necessary for the plotting routines. The data are saved on a temporary file (Unit 39).

2.2.2 Input Routines (Figure 4)

2.2.2.1 LEXPACK

This package is used to perform the lexical analysis necessary for parsing. The purpose for combining these procedures into a package is to reduce the scope of data communication to the subroutines contained within the package.

2.2.2.1.1 GETLINE

This subroutine is used to fill the input buffer with one line of input data from INFILE. The line of code is then echoed to the output file, LSTFILE. When the end of file is reached, the flag ENDFLAG is set to true.

2.2.2.1.2 GETCHR

This function subroutine is used to return the current character from the input buffer.
Figure 4. Input Routines
2.2.2.1.3 ADDCHR

This subroutine is used to concatenate the given character with the token string being built. The length of the string stored in TKNLEN is also incremented by one representing the current length of the token string.

2.2.2.1.4 GETOKEN

This subroutine will lexically analyze an input stream of characters, returning the string composing the token, the length of the string, and a token value.

2.2.2.2 PRSPACK

This package is used to perform the parsing of the source file INFILE. The method is a simple table driven parse. The parse table is initialized in the block data subroutine PRSDATA. The parse table consists of the state transitions for input tokens. Each time an input token is returned by LEXPACK\GETOKEN, subroutine PRSPACK\LOOKUP is called to determine the next state to go to by referencing the parse table with the current state versus the current input token value. Program execution then transfers to a statement label representing that state, where various semantic actions are performed to store the information in internal data structures.

2.2.2.2.1 LDATE

This function subroutine is used to check if a certain part (i.e., MONTH, DAY, YEAR) is legal according the integer bounds pertaining to that part of the date.
2.2.2.2.2 LTIME

This subroutine function is used to test if the time specified is within the military bounds of 0001-2400 hours.

2.2.2.2.3 LOOKUP

This subroutine is used to access the parsing table based on the current input token value and the current state of the parser. The current state of the parser is updated and then an alternate return is processed based on the current state of the parser.

2.2.2.2.4 PARSE

This subroutine is used to parse the input file INFILE by means of a table-driven parser. Upon reaching the current state of the parser various semantic actions are performed to store the data in INFILE in internal data structures for later use.

2.2.2.3 SCHPACK

This package is used to perform the process of searching the data tables created during the parse stage. This search is used to find records of subsonic and supersonic flight data records in the library file by finding their location through the use of an index file. This index file is similar to a card catalog.

2.2.2.3.1 FILBUF

This function subroutine is used to read in one record from the INDEX FILE. If there are no more records, the flag ENDRECS is set TRUE.
2.2.2.3.2 STRMCH

This function subroutine is used to see if an input string matches any string in the current row of an input table. If 'ALL' is found then the search is considered successful.

2.2.2.3.3 INTMCH

This function subroutine is used to test if an integer passed in matches an integer in the current row of the table passed in. If '9999' is found then the test is considered successful.

2.2.2.3.4 GETREC/GETINX

These subroutines are used to search the index file according to the user specifications stored during the parse stage. When invoked these subroutines will read records from the index file until a match is found. When a match is found the subroutine will return the starting record number and the number of records occurring after the starting record. If a match is not found then the end of record flag ENDRECS is set true.

2.2.3. Ray Tracing Driver Routines (Figure 5)

2.2.3.1 RTRACE

This is the initial driver for the ray tracing routines. RTRACE calls the routines that break the flight track up into segments, create the splines, do maneuver screening and phi angle selection, and do the ray tracing. The flight track information is converted from feet to meters before it is processed. (Note: All routines for Ray Tracing are in meters.)
2.2.3.2 SPLINE

This routine computes the matrix for finding the coefficients of a cubic spline through a set of data. The system is then solved to obtain the second derivative values. Using the second derivatives the spline coefficients are calculated.

2.2.3.3 GETSEG

GETSEG divides the flight into segments where the points are above the critical Mach number. The first two and the last two points of a segment can be below critical, which is done in order to improve the spline interpolation. There can also be subcritical points in the track; however, there can be at most only 5.5 seconds between critical points. If there is a 4.5 second gap between data points, the segment is also terminated. It passes back two arrays of pointers that point to the start and end of each flight segment.

2.2.3.4 SCREEN

SCREEN calculates the matrix of phi angles that rays are to be traced for each emission time. If the aircraft is below 1500 feet then rays are traced every two degrees. Otherwise, they are traced every degree.

2.2.3.5 SORTPHI

This subroutine puts the array of phi angles in ascending order.

2.2.3.6 STORE

This routine stores the aircraft locations and the ground coordinates and overpressures of the boom in the ray library file "CLIBRY". (Meters, pascals.)
2.2.3.7 SIDATT

SIDATT retrieves the information necessary from the "CLIBRY" to do overpressure extrapolation at the sidelines of the boom.

2.2.3.8 EXTRPR

This subroutine is designed to extrapolate outside the margin of the last ray down to the threshold of approximately 80 dB. It performs this task by receiving two rays and three angles. (If the last ray does not hit the ground then its termination point is calculated and used as the last ray). The last and next to the last rays are used to extrapolate outside the margin to calculate the new ray's termination points and overpressure.

2.2.3.9 FFUNC

FFUNC generates F-Functions for the various aircraft in the table. The F-Function is generated from a table of $k_s$ factors and it is 104 points long. If the aircraft type is not found in the table, an error message is printed, and processing for that flight is aborted.

2.2.3.10 LSQUAR

This routine is designed to get the acceleration information from the cubic spline coefficients and calculate quadratic coefficients using a weighted linear least squares method.

2.2.3.11 GAUSJR

This routine does a Gauss-Jordan reduction and a determinate evaluation of a matrix.
2.2.3.12 SORT

SORT uses a Heap Sort to sort the input array.

2.2.3.13 RBRAYS

RBRAYS is a filter used at the end of processing of each time hack. It checks the rays and removes any anomalous ones from the database.

2.2.4 Focal Zone Calculation Routines (Figure 6)

2.2.4.1 FOCMAP

Given an approximate angle where the maximum over pressure on the ground will occur, FOCMAP traces rays on either side of that angle to get a good sampling of the overpressures on the ground. It terminates processing in a given direction when the focus is no longer between +1000 and -1500 feet of the ground.

2.2.4.2 CSTGND

CSTGND identifies the phi angle(s) at the point(s) when a caustic surface intersects, or is closest to, the ground.

2.2.4.3 FOCUS

FOCUS locates a caustic surface and its relative curvature at the ray intersection. It calculates an initial ray tube by tracing three other rays and uses this bundle to calculate the direction in which to trace three auxiliary ray tubes. The initial ray and the auxiliary tubes are used to map out the caustic surface in space.
Figure 6. Focal Zone Calculation Routines

FOCMAP

FOCAL

FOCUS

GETDIT

CSTGNRD

INTERP

SPLINE
2.2.4.4 FOCAL

FOCAL is used to calculate the focal zone width when a focus is between +1000 ft and -1500 ft of the ground. It then determines whether the ground is in the focal zone and whether to use the TRAPS or FOB OOM signature and overpressure.

2.2.4.5 INTERP

INTERP does a linear interpolation between two points.

2.2.4.6 GETDLT

To get the delta increment for the phi angles to be traced in a caustic-ground intersection, GETDLT takes 1/10th of the delta of normal ray tracing.

2.2.5 TRAPS Input Routines (Figure 7)

2.2.5.1 SETUP

SETUP sets various flags for TRAPS ray tracing. It also calls the atmospheric table routines.

2.2.5.2 ATMSIN

ATMSIN performs overall control of the routines written to input atmospheric data. It merges the results of PTDHIN and WINDIN and a pre-selected set of altitudes at which ray-trace output is wanted. It then uses the above to create a single overall data table for use by subroutines AIR and RAYTRK.

2.2.5.3 PTDHIN (This routine is presently used to set up a standard atmosphere table only.)
Figure 7. TRAPS Input Routines
PTDHIN reads the RAOB File. It converts all data into Standard International (S.I.) units, interpolates dewpoint data as needed and calculates virtual or molecular scale temperatures from the temperature and dewpoint data. It supplies hydrostatically valid height or pressure data, as appropriate, and returns a table of virtual temperatures, pressures, and heights. It can also print out all input data, together with the calculated pressure and height information in original units for comparison with other sources.

2.2.5.4 RAOBWK

RAOBWK is called by PTDHIN to create an RAOB; i.e., to calculate from the given temperatures\(^1\) and pressures the "thicknesses" (i.e., height of the column of air between each pair of pressure levels) and then, by keeping a running total of "thicknesses", calculate heights. Conversely, if given thickness, it calculates the pressure drop.

2.2.5.5 WINDIN (This routine is currently bypassed to produce no winds.)

WINDIN is called by ATMSIN to read the WINDS File and convert to S.I. units. It produces an internal table of wind speeds, directions, and "turning rates"; i.e., the rate of direction change with height between the levels in the WINDS File. The turning rate is provided to assist AIR in linear interpolation of wind direction; it has the sign and magnitude to cause the smallest rate of direction change meeting the given directions. Where the wind speed is zero on one side of a layer, the turning rate is taken to be the same as that of an adjacent

\(^1\) The virtual or molecular-scale temperature is the temperature at which dry air of mean tropospheric chemical composition would have the same pressure-density relationship as the actual air. It is the appropriate temperature for calculating both thicknesses and sound speeds.
layer. The routine also prints out the speed and direction data in the original units for documentation.

2.2.6 TRAPS Ray Tracing Routines (Figure 8)

The function of emitting and tracking rays from aircraft to ground is performed by the Ray Tracing routines. These routines are based in the program identified in Reference 5.

2.2.6.1 ACMOVE

ACMOVE interpolates aircraft track coefficients to current value of emission time. It computes and stores in COMMON block the position, velocity, acceleration, and jerk of the aircraft, the local speed of sound and wind, the air speed and its rate of change, the Mach number and its rate of change, the climb and bank angle and the wing loading, and the direction cosines of a ray cone coordinate system and their rates of change. It can also print out the information on the aircraft position and motion, both in an airborne reference frame and a ground reference frame.

2.2.6.2 TACMOV

TACMOV calls acmove and saves the aircraft state vector as required.

2.2.6.3 FILIMS

Given the information from the ACMOVE subroutine, and the wind velocity and speed of sound at the ground, FILIMS computes the limits of the phi angle of the admittance ellipse for the ground level. It can print out the limiting phi angles for the arcs inside the admittance ellipse, if any.
Figure 8. TRAPS Ray Tracing Routines
2.2.6.4 RAYORG

For each emission time and for each value of phi lying within the admittance ellipse, RAYORG computes the initial values of position, ray normals, "frequencies", and their rates of change. It sets current time equal to emission time. The rates of change are with respect to not only current time, but also to the ray parameters of phi angle and of emission time. If ray trace printing is selected, it prints out the initial ray trace values.

2.2.6.5 RAYTRK

From the initial values supplied from RAYORG, RAYTRK traces the ray to the ground level and reflects as many times as necessary. RAYTRK has been modified to trace the ray to either a caustic or 2000 feet below the ground depending on the value of TRACE selected. It controls the computation of the change in not only the position of the ray, but associated terms such as the ray normals, the ray tube area terms, and the age(s). If ray trace printing is selected, it also prints a record of position, ray tube area, and time at selected altitudes.

2.2.6.6 RATES

RATES computes the local rate of change of the ray position, the ray normals, and the associated derivations with respect to the ray parameters and emission time.

2.2.6.7 ADVANS

ADVANS utilizes information from RATES to compute the advance in current time, and the change in ray position and associated variables corresponding to it.
2.2.6.8 RCRVIT

When a tentative advance brings the ray beyond a reversal layer, RCRVIT locates the exact position of the reversal layer.

2.2.6.9 RECORD

When the ray has been traced to ground in a selected carpet, RECORD will record the location and all the associated variables required to compute signatures on a temporary file (FORTRAN Unit 9).

2.2.6.10 ARTUBE

ARTUBE computes the Jacobian defining the ray tube area.

2.2.6.11 DIST

DIST computes the distance between two points in space.

2.2.6.12 RCSPCL

RCSPCL outputs the positions and times for each "special point" in the ray's path. "Special points" include reversal layer encounters, ground encounters, and the encounters with the caustic surfaces. (This routine is presently unused.)

2.2.7 Signature Aging Routines (Figure 9)

After all rays have been traced, it is the task of the signature aging routines to perform the final calculations and determine the actual overpressures to be expected.
Figure 9. TRAPS Signature Aging Routines
2.2.7.1 SIGNUR

SIGNUR has overall control of the aging and printout process. For each ray terminus recorded by RECORD, it reads, interprets and prints out the information on ray tape, including Mach number of aircraft, initiation time, phi angle, location, elevation and azimuth of the ray normals, and the conversion factors from F-function normalized coordinates to time (TFACT) and pressure (PFACk). It combines the F-functions according to this information and controls the evolution of the signature. (Printout has been inhibited.)

2.2.7.2 FREAD (Presently unused)

FREAD determines whether the necessary F-function tables are in main memory, and if not, reads them into main memory. It enables the use of lift and area coefficients to determine the F-function.

2.2.7.3 AGING

AGING shifts the abscissa values (phase) of the F-functions according to the age value, determines the total area of the resulting figure, and fits discontinuities as appropriate. It replaces the input F-function with the result.

2.2.7.4 HILBRT

HILBRT has overall responsibility for calculating the Hilbert Transform. It replaces the input F-function, as modified by AGING and possibly containing shocks, by its Hilbert Transform. It computes the transform at a selection of points determined by the overall structure of the function, which includes a set of points exponentially converging to each shock (terminating within a distance of the shock equal to $6 \times 10^{-7}$ times the overall scale of the input F-function). It also includes a
set of points which are centered on the mean abscissa value of
the input F-function and which are spaced at increasing
increments to cover an interval several times the abscissa scale
of the input F-function.

2.2.7.5 CPVAL

CPVAL computes the value of the integral defining the
Hilbert Transform, as a Cauchy Principal Value, at each point
directed by HILBRT.

2.2.7.6 SORTEM

SORTEM sorts the values calculated by HILBRT and CPVAL into
ascending order of abscissa values, as required by AGING.

2.2.7.7 SIGPRT

SIGPRT prints out the final signature, as directed on the
CONTROL File cards. (Printout is currently inhibited.)

2.2.7.8 TSIGPT

TSIGPT sets the appropriate variables and calls SIGPRT.

2.2.7.9 FOCALP

This subroutine applies 'GILL AND SEEBAAS' (Reference 6)
focused shock wave solution to each shock in a sonic boom at a
caucistic. It first converts the input signature from values PP at
arbitrarily spaced XX to 100 evenly spaced points. Pressure
positions are diddled slightly, but no more than half of one
percent of the signature length. The focus solution is applied
to each shock. The published signature is linearly extrapolated
until zero, space permitting. Space not permitting, focus
solutions are carried out to the midpoints between successive shocks.
NOTE: The units used in this routine are feet and PSF.

2.2.7.10 CALSEL

CALSEL calculates the CSEL from the signature, which is done by first removing the shocks from the input pressure array and then interpolating the signature to every 0.5 millisecond. It then passes the array to a FFT routine and the CSEL is computed from the frequency spectra.

2.2.7.11 FFT

FFT calculates a multivariate complex Fourier Transform.

2.2.7.12 CURVE

This subroutine fits a circle through a set of points and returns the radius vector and curvature.

2.2.7.13 FINDT

FINDT finds a point on a ray at a specific time. It then sets the arrays so that the locations on each ray stored correspond to the time of the earliest caustic.

2.2.8 Utility Routines (Figure 10)

These utility routines are called from various subroutines throughout the code. Some of these routines deal with physical problems and vector arithmetic. Others are used by TRAPS for housekeeping and table lookup.
2.2.8.1 AIR

AIR is called to produce, at a specified altitude within a specified layer, the values of the speed of sound and wind velocity, the first and second derivatives of those quantities with respect to height, and the density of the atmosphere. It uses linear interpolation of wind speed, wind direction, and virtual temperature, with respect to geopotential height. All other quantities are derived by algebraic manipulation and a hydrostatic assumption.

2.2.8.2 PHELEV

Given the components of the wave-number vector, PHELEV calculates the elevation angle of the normals to the phase surfaces of the wave.

2.2.8.3 PHAZIM

Given the components of the wave-number vector, PHAZIM calculates the azimuth angle of the normals to the phase surfaces of the wave.

2.2.8.4 EAMENU

Given the elevation angle, azimuth angle, and magnitude of a vector, EAMENU calculates the east, north, and upward components of that vector.

2.2.8.5 UNITS

Given a character string for unit type, a table of possible unit names, a default unit index, and a character string, UNITIS determines the appropriate unit index or prints appropriate error messages.
2.2.8.6 TIMCVR

If TRACK File chose HHMMSS units, TIMCVR converts hhmmss time units to seconds and vice-versa. Otherwise, it leaves time units unchanged.

2.2.8.7 GETLYR

Given a numeric value and a pre-sorted table of numeric values, GETLYR performs a binary table search to determine between which two table entries the given value is located. If the given numeric value is not covered by the table, a non-standard return is performed.

2.2.8.8 FNDLYR

FNDLYR defines the location of a layer in the atmosphere in which a given altitude is located. It sets numeric variables to top and bottom of layer. It is called just prior to calling AIR by all routines except RAYTRK, which manages layer definition for itself.

2.2.8.9 DOTP

Function DOTP computes the dot product of 2 N-dimensional vectors.

2.2.8.10 RNORM

Function RNORM calculates the vector norm.

2.2.8.11 CROSS

CROSS calculates the cross product of two 3-dimensional vectors.
2.2.8.12 UNIT

UNIT calculates a unit vector in the same direction as the input vector.

2.2.8.13 MDOT

MDOT converts zero values in the 'STATS' output to dots, for easier reading.

2.2.8.14 SAVRAY

Subroutine SAVRAY saves the ray times, locations, and ages at each mesh step in a temporary array.

2.2.8.15 SRTRAY

SRTRAY accepts a random access file and a specified field. It then sorts the random access file on the selected field using a heapsort method.

2.2.8.16 SORTRY

SORTRY accepts a two dimensional array and a specified column of that array. It then sorts the array on the selected column using a heapsort method.

2.2.8.17 DDOTP

Double precision function DOTP computes the dot product of 2 N-dimensional vectors.

2.2.8.18 DRNORM

Double precision function RNORM calculates the vector norm.
2.2.8.19 DCROSS

CROSS calculates the cross products of two 3-dimensional double precision vectors.

2.2.8.20 DUNIT

UNIT calculates a double precision unit vector in the same direction as the input vector.

2.2.9 Graphics Routines (Figure 11)

The following routines are used to initialize the needed variables, open the necessary files, and sort the data on a selected field in order to generate the selected graphical output.

2.2.9.1 STOREC

STOREC reads in the necessary variables stored in file HOLDVAR, FORTRAN unit 76, to select the specified flight tracks and create the selected output, which allows the BOOMAP2 program to be run as a two-step process. This routine then calls PLOTDR to invoke the plotting routines.

2.2.9.2 PLOTDR

PLOTDR opens the files needed to produce the selected graphical output. It then calls CONTUR, CPBMTR, CPSSTR, CPINIT, CPTERM, and SCRPAD to produce the selected output.

2.2.9.3 SCRHFL

SCRHFL is designed to read records from the ray library, CLIBRY FORTRAN unit 52, and create a temporary file called SCRCHFL, Fortran unit 32. In doing so the routine also creates a
Figure 11. Graphics Routines
temporary file called HOLDF, FORTRAN unit 34. The flight tracks read from CLIBRY are stored in file SCRCHFL if the flight track contains caustic rays. If a flight track has a gap of 4.5 seconds or greater it is broken up into several flight segments. Each has a scratch pad plot generated if it contains caustic rays.

2.2.9.4 CPBMTR

CPBMTR reads in the boom producing flight tracks from FORTRAN file 4 and produces a plot file, TAPE11, to be processed later by GPCP II.

2.2.9.5 CPSSTR

CPSSTR reads in the supersonic flight tracks from file three and produces a plot file, TAPE11, to be processed later by GPCP II.

2.2.9.6 CPINIT

This routine initializes TAPE11 which is used to store the GPCP II compatible plot files.

2.2.9.7 CPTERM

This routine is used to close file eleven which contains the plot file for GPCP II.

2.2.10 Contouring Routines (Figure 12)

The following routines are used to calculate the contour grid, output the contour plots, and to output the supersonic and boom-producing flight tracks.
Figure 12. Contouring Routines

- CONTUR
  - SCRHFL
  - GETINX
    - GRIDPW
    - DIVARR
  - CPCONT
2.2.10.1 CONTUR

CONTUR calls GETINX which returns pointers to the ray database for a selected flight track. It then calls SCRCFL which creates the scrchpad file. CONTUR reads in ray data from CLIBRY, FORTRAN unit 52, to produce temporary file TEMPFL, FORTRAN unit 33. TEMPFC contains the rays, sorted on termination time, to be entered into the contour grid. In producing file TEMPFL, another temporary file is used called TMP2FL, FORTRAN unit 35. After a flight track is processed, CONTUR calls GRIDPW and then looks for another flight track to process. When all the selected flight tracks have been processed, CONTUR calls CPCONT which outputs the contour grid to be processed by GPCP II.

2.2.10.2 CPCONT

CPCONT accepts a grid array which it then outputs to TAPE11 which will be processed later by GPCP II to produce the selected contour plots.

2.2.10.3 GRIDPW

GRIDPW is designed to read data from file TEMPFL, FORTRAN unit 33, that was created by CONTUR. It then calculates the slant distance from the aircraft to the ray termination point. With the slant distance it calculates the weights for each of the four grid points closest to the ray termination. It then adds these values to the scratch array and increments the counter array. It then calls DIVARR when there is an elapsed time of 4.5 seconds.

2.2.10.4 DIVARR

This routine takes the scratch array and divides it by the counter array and then adds it to the master array which is later output to CPCONT.
2.2.11 Scratch Pad Routines (Figure 13)

The following routines are designed to set up the file used to create the scratch pad plots, select the proper contour levels, and generate the scratch pad plots.

2.2.11.1 SCRPAD

SCRPAD is designed to read in data from the file, SCRCHFL, FORTRAN unit 32, created in SCRHFL. It then calls CCONVL, FNDCNT, CONPTS, PSETUP, PLOTIT, and CLSPLT to output the scratch pad plots. It also calculates the area for each contour level. For each scratch pad, three or less contour levels are generated.

2.2.11.2 CCONVL

CCONVL accepts an array of pressures which it converts to either PSF or dB. It also selects ten contour levels which will be searched for in the other routines.

2.2.11.3 FNDCNT

FNDCNT accepts three arrays for each time hack in the flight segment; they are a pressure array and the x and y location of the pressure value. It then searches for four or less points in the pressure array which coincide with the selected contour level. If it finds a point, it does a linear interpolation to find the approximate x and y locations for that contour level.

2.2.11.4 CONPTS

CONPTS is designed to connect up the points found in FNDCNT to form a contour level.
Figure 13. Scratch Pad Routines
2.2.11.5 PSETUP

PSETUP sets up the plot page. It plots the aircraft flight track, flight information, flight identification, maximum overpressure location, range center, carpet boom level, and map annotation. It also calculates a map scale.

2.2.11.6 PLOTIT

PLOTIT plots the contour levels one at a time and puts the area and level of the contour on the scratch pad plot.

2.2.11.7 CLSPLT

The CLSPLT, this routine is designed to create a new plot page after a scratch pad plot has been generated.
3.0 ENVIRONMENT

3.1 Hardware

The BOOMAP2 software was designed to execute on Control Data Corporation (CDC) computers using the NOS 2.4 operating system. Access to California Computer Products (CALCOMP) General Purpose Contouring Program (GPCP II) is needed to do contour maps and flight-track maps. CDC's UNIPILOT Library is needed to do scratch pad plots. To produce all plots an appropriate pen plotter (that can be driven by both GPCP II and UNIPILOT) must be attached to the computer systems.

3.2 Database

3.2.1 Input Index and Library files

These two input files contain the aircraft flight information necessary for the ray tracing calculations.

3.2.1.1 Index File, 'INDEX'

This is a formatted, direct access file. It contains the mission information and access information necessary to locate entries in the flight library file (Reference 2).

3.2.1.2 Library File, 'LIBRY'

This is a formatted, direct access file. It contains the dynamic database for each aircraft flying each mission (Reference 2).
3.2.2 Output Index and Library Files

There are two files used for the storing of ray information after processing. They are CLIBRY and CINDEX. These files are accessed by the plotting programs to produce focal boom plots and full contours.

3.2.2.1 Ray Index File, 'CINDEX'

CINDEX is a random access file which contains the information necessary to uniquely identify each flight. Each record is 110 characters long. It also contains the addresses in CLIBRY for the rays and maximum overpressure records.

First record: Format(I10)

I10 NUMREC: Number of records in the file.

Second record onwards: Format(A16, A8, I2, A10, I8, I8, I2, A6, A8, I10, I10, I10, I10, L1)

A16 Mission name
A8 Date of mission
I2 Site number
A10 Site location
I8 Starting time of the mission
I8 Ending time of the mission
I2 Aircraft type number
A6 Aircraft type
A8 Aircraft tail number
I10 Starting record of the corresponding ray database in 'CLIBRY' file
I10 Total number of ray records
I10 Starting record of maximum overpressure records in 'CLIBRY'
I10 Total number of the maximum overpressure records
L1  Caustic flag set true if there are any caustics in the flight.

The range times are in the form Hour, Minute, Second, 1/100 Second.

3.2.2.2 Ray Library File, 'CLIBRY'

CLIBRY contains the ray information for all of the flights. It is also a random access file and each record is 10 - 100 characters long. Each section of this file consists of three parts: the header record, the ray records, and the maximum overpressure records.

In addition to information about the flight, the header record also contains the altitude of the ground.

The ray records contain all of the pertinent information about the rays for each super critical time analyzed. The flag at the start of each record contains either 0, 21, or 11. A zero indicates that these overpressures are attenuated at the sidelines. A 21 indicates that this ray has a focus near the ground and a scratch pad plot has not been produced. The flag setting of 11 indicates that the ray has no focus, or that the focus is not near the ground. When the scratch pad plots for this flight have been processed the 21s are changed to 11s.

The maximum overpressure records contain the information about the ray with the largest overpressure, for each time processed, that had a focus near the ground.
First record: Format(I10)

I10 ENDREC: Record number of the last record in the file.

For each aircraft, for each mission segment.

Header record: Format(A16, A8, I2, A10, A6, A8, F10.2)

A16 Mission name
A8 Date of mission
I2 Site number
A10 Site location
A6 Aircraft type
A8 Aircraft tail number
F10.2 Altitude of the ground in meters

Ray records: Format(I2, F8.2, 3F8.0, F8.2, 2F8.0, F8.3, F10.4, F10.4, F10.4)

I2 Processing flag
F8.2 Ray emission time in seconds
3F8.0 Range coordinates (x, y, and z) of the aircraft in meters
F8.2 Ground arrival time of the ray in seconds
2F8.0 Ground coordinates of the ray in meters
F8.3 Emission phi angle
F10.4 Overpressure in pascals
F10.4 SEL
F10.4 Effective aircraft Mach number

Maximum overpressure record: Format(F8.2, 3F8.0, F8.2, 2F8.0, F10.4, F10.4, F10.4)
F8.2 Ray emission time in seconds
3F8.0 Range coordinates (x, y, and z) of the aircraft in meters
F8.2 Ground arrival time of the ray in seconds
2F8.0 Ground coordinates of the ray in meters
F8.3 Emission phi angle
F10.4 Overpressure in pascals
F10.4 SEL
F10.4 Overpressure calculated using Carlson's method in PSF
4.0 PROGRAM MAINTENANCE PROCEDURES

4.1 Conventions

See the program listings for a detailed description of the variables associated with each routine.

4.2 Verification Techniques

Verification of the code was accomplished by comparison of the output with that of the TRAPS and FOBOOM programs. In comparison with TRAPS, overpressures differed by no more than 5%. These differences were due to the fact that TRAPS and BOOMAP2 calculate the aircraft vectors using different interpolation methods. The differences with FOBOOM were slightly larger, about 10%. FOBOOM cannot calculate overpressures beyond the focus; therefore, a detailed comparison was not possible.

4.3 Updating Site and Aircraft Information

There are two routines that define site altitude and latitude and longitude information. They are GETALT and GETLL. To add another site to the program, the data statements in these routines must be modified accordingly.

The subroutine FFUNC contains the aircraft information. In it are data statements for aircraft type, a $K_s$ value, aircraft weight, and aircraft length. A new aircraft type may be added to the program by placing the appropriate information into these data statements and altering the dummy values at the end, accordingly.

The TRAPS subroutine FREAD allows the user to input lift and area factors for the F-functions. If lift and area inputs are desired, the calls to FFUNC must be replaced with calls to FREAD.
Report No. 6489

For a more detailed description of the input to FREAD see Reference 5.
REFERENCES


Appendix A
PROGRAM LISTINGS
PROGRAM BOOMAP2 (INPUT, OUTPUT, TAPE6=OUTPUT,
   1)

COMMON /GRID/ GRIDXO, XGS, GRIDMX, GRIDY, YGS, GRIDMY,
   1 LIMAXO, LIMAYO, LIMBXY, LIMBYO,
   2 LIMAXY1, LIMAYY1, LIMBXY1, LIMBY1

COMMON /STATS/ STATFG, BOOMFG, MACHFG, CONTFG, BOOMVL,
  + MACHVL, CONTVL(5,20), CONTYP(5), WIDTH, FFT,
  + SIGNAT, RAYTRC, SCRPAD, SCRPSF, SCROLL

LOGICAL LMFG, LMCFL1, LMCFL0, PLYFG, MRFG, PERFG
LOGICAL STATFG, MACHFG, BOOMFG, CONTFG
LOGICAL RAYTRC, SIGNAT, FFT, SCRPAD, SCRPSF, SCROLL
LOGICAL GPCPFL, GPCPMK, GPCPM, GPCPCN, RAYINX

COMMON /FLIGHT/ NFP, FTIME, FX, FY, FZ, VX, VY, VZ, FMACH, CA
DIMENSION FTIME(1156),FX(1156),FY(1156),FZ(1156)
DIMENSION VX(1156),VY(1156),VZ(1156),FMACH(1156),CA(1156)
DIMENSION MX(52), MY(52), MZ(SZ), NM(59), MMC(S9), MME(S9)
DIMENSION MOP(59), MLPK(79), MLCE(79), MLAE(79), MHE(52)
DIMENSION MXY(52,52)
DIMENSION GRIDA(102,102), GRIDB(10,10), GRIDT(10,10)

COMMON /ACIDNT/ ACTYP
CHARACTER*8 ACTYP

COMMON /ACWIEG/ ACWT

COMMON /INDEXR/ INDXR

CHARACTER*10 SITELC, TDATE, TTIME, LAT, LONG
CHARACTER*16 MNNAME
CHARACTER*8 MDATE, ACTAIL, STARTT, ENDT
CHARACTER*6 ACTYPE
CHARACTER*20 IDX, IDY, IDZ, IMD, IDMC, IDME, IDOP,
1 IDLPK, IDLCE, IDLAE
CHARACTER*70 TITLE

COMMON /UNITS/ WTUNIT,HTUNIT
CHARACTER*8 WTUNIT,HTUNIT

COMMON /GROUND/ GLAYER,ZGRND,CGRNO,UGRNO,VGRNO,REFLFC
INTEGER GLAYER

REAL MACH, MACHVL
INTEGER CONTYP

EQUIVALENCE TO SAVE SPACE

EQUIVALENCE (GRIDA(1,1),FTIME(1))

GET THE TIME AND DATE

TDATE = DATE()
TTIME = TIME()

HTUNIT = 'FT'
WTUNIT = 'KG'

PI = 3.1415926535
FTOELM = 999999.
TWOPI = 2.0 * PI
STATFG = .FALSE.
MACHFG = .FALSE.
BOOMFG = .FALSE.
CONTFG = .FALSE.
RMHE = 0.
RMMSN = 0.
RMSEMN = 0.
RMSHE = 0.
DO B I=1,20
DO B J=1,5
B CONTVL(J,1) = 0.
C C ZERO THE STATISTICAL MATRICES
C
DO 12 I=1,NX
MX(I) = 0
12 MY(I) = 0
DO 13 I=1,NZ
MHE(I) = 0
13 MZ(I) = 0
DO 14 I=1,WM
MM(I) = 0
MMC(I) = 0
14 MME(I) = 0
DO 15 I=1,WO
15 MOP(I) = 0
DO 16 I=1,ML
MLPK(I) = 0
MLCE(I) = 0
16 MLAE(I) = 0
DO 17 I=1,NX
DO 17 J=1,NY
MXY(I,J) = 0
C C GET USER SUPPLIED INPUT DIRECTIVES
C
CALL PARSE(TITLE, PERFG)
IF (PERFG) STOP ' ** INPUT DIRECTIVE SYNTAX ERROR **'
C C INITIALIZE POINTERS AND COUNTERS
C
FTINC = 1.0
NMTCH = 0
NSUP = 0
NSUB = 0
NBOOM = 0
LINCNT = 0
TSECS = 0.
TSECBR = 0.
NME = 0
NCE = 0
NMET = 0
NCET = 0
C C OPEN INDEX AND LIBRARY FILES
C
OPEN (1, FILE='INDEX', STATUS='OLD', ACCESS='DIRECT',
1     FORM='FORMATTED', RECL=98, BLANK='NULL')
OPEN (2, FILE='LIBRARY', STATUS='OLD', ACCESS='DIRECT',
1     FORM='FORMATTED', RECL=70, BLANK='NULL')
OPEN(3, FILE='FIL3')
OPEN(4, FILE='FIL4')
OPEN(7, FILE='FIL7')
OPEN(12,FILE='FIL12')
OPEN(20,FILE='SIGNAT')

OPEN CONTOUR INDEX AND LIBRARY FILES

OPEN (51, FILE='CINDEX', STATUS='UNKNOWN', ACCESS='DIRECT',
1     FORM='FORMATTED', RECL=110, BLANK='NULL')
OPEN (52, FILE='CLIBRY', STATUS='UNKNOWN', ACCESS='DIRECT',
1     FORM='FORMATTED', RECL=110, BLANK='NULL')
OPEN (50, FILE='RAYDAT',STATUS='UNKNOWN')
RAYINDEX = .FALSE.

C----------------------- TOP OF LOOP -----------------------

C GET THE NEXT ENTRY FROM THE LIBRARY INDEX

40 CALL GETREC (NSTREC, NREC, NSUREC, MRFG, 11)
   INDXR = 11

C CHECK FOR END OF MATCHING RECORDS

C IF (MRFG) GOTO 500
IF (NREC.LE.0) GOTO 40

C READ TIME HISTORY FROM DISK TO MEMORY

50 NMTCH = NMTCH + 1
   SECS = 0.
   SECB = 0.
   IR = NSTREC
   READ (2,2001,REC=IR) MNAME, MDATE, STARTT, ACTYPE, ACTAIL
   IR = IR + 1
   READ (2,2002,REC=IR) SITELC, MSEG
   IR = IR+1

C FIND OVERPRESSURE FACTOR AND WEIGHT IN KG

C CALL OPFIND (ACTYPE, OPFACT, ACWT)
   ACTYP = ACTYPE
   IF (OPFACT .EQ. 0.0 .OR. NSUREC .LE. 1) GOTO 70

C LOOK UP TEST RANGE ALTITUDE, LATITUDE AND LONGITUDE

C CALL RNGALT (SITELC, RZMIN)
CALL RONGL (SITELC, LAT, LONG)
   ZGRND = RZMIN
C
INITIALIZE RAY TRACING ROUTINES
CALL SETUP

READ FLIGHT TIME HISTORY INTO ARRAY

NFP = NREC - 3
IF (NFP .LE. 1) GOTO 70
IF (NFP .GT. 1156) STOP2

DO 60 I=1,NFP
READ (2,2003,REC=IR) FHR, FMIN, FSEC, FMACH(I), FX(I),
FY(I), FZ(I), CA(I), VX(I), VY(I), VZ(I)
IR = IR + 1
FTIME(I) = FHR*3600. + FMIN*60. + FSEC/100.
60 CONTINUE

READ FINAL RECORD

70 IR = NSTREC + NREC - 1
READ (2,2004,REC=IR) ENDT, NRECD, NMACH
IF (OPFACT .EQ. 0.0 .OR. NSUREC .LE. 1) THEN
NSUB = NSUB + 1
GOTO 310
ENDIF

GO THRU THE FLIGHT AT ONE SECOND INTERVALS

IF (NFP .LE. 1) GO TO 40
FTO = FTIME(1)
FTN = FTIME(NFP)

PRESSET CURRENT MACH .GT. 1.0 AND MACH .GT. CUTOFF AS .FALSE.

LMFG = .FALSE.
LMCFL1 = .FALSE.

JO = 2
FT = FTO
105 DO 110 J=JO,NFP
10 = J - 1
IF (FT .LE. FTIME(J)) GO TO 115
110 CONTINUE

RAN OUT OF DATA POINTS, ALL DONE WITH THIS FLIGHT
WRITE (3,3001) FTDELm
WRITE (4,4001) FTDELm
GOTO 300

115 JO = 10 + 1
    FTIO = FTME(IO)
    FTJO = FTME(JO)
C
C REMEMBER MACH STATUS AND LATERAL BOOM PROPAGATION
C POINTS OF PREVIOUS DATA POINT ONE SECOND AGO
C
LM CFLO = LMCFL1
C ELXO = ELX1
C ELYO = ELY1
C ERXO = ERX1
C ERYO = ERY1
C XCO = XC
C YCO = YC
C BXO = BX
C BYO = BY
C NEO = HE
C DYCO = DYC
C DLATO = DLAT
C XLCEO = XLCE
C
C TEST FOR 5.0 SECOND OR GREATER GAP. IF TRUE ASSUME
C A/C WENT SUBSONIC IN THE INTERIM
C
IF ((FTJO-FTIO) .LE. 5.0) GOTO 117
IF (LMFG) WRITE (3,3001) FTDELM
LMFG = .FALSE.
IF (LMCFL1) WRITE (4,4001) FTDELM
LMCFL1 = .FALSE.
FT = FTJO + 0.01
GOTO 105
C
C ELSE CALCULATE NEXT X-COORD, Y-COORD, Z-COORD, CLIMB ANGLE,
C MACH #, GAMMA, HEIGHT ABOVE GROUND
C
117 CONTINUE
C
XC = (FX(JO) - FX(IO))* (FT - FTIO) / (FTJO - FTIO) + FX(IO)
YC = (FY(JO) - FY(IO))* (FT - FTIO) / (FTJO - FTIO) + FY(IO)
ZC = (FZ(JO) - FZ(IO))* (FT - FTIO) / (FTJO - FTIO) + FZ(IO)
XCA = (CA(JO) - CA(IO))* (FT - FTIO) / (FTJO - FTIO) + CA(IO)
XMT = (FMACH(JO) - FMACH(IO))* (FT - FTIO) / (FTJO - FTIO) +
     1    FMACH(IO)
C
CALL SOUND(ZC, CO)
VXC = (VX(JO) - VX(IO))* (FT-FTIO) / (FTJO - FTIO) + VX(IO)
VYC = (VY(JO) - VY(IO))* (FT-FTIO) / (FTJO - FTIO) + VY(IO)
VZC = (VZ(JO) - VZ(IO))* (FT-FTIO) / (FTJO - FTIO) + VZ(IO)
VEL = (VXC**2 + VYC**2 + VZC**2)**0.5
XM = VEL/CO
C
GAMMA = XCA * 1.743329E-2
HG = RZMIN
HA = ZC
H = HA - HG
IF (XMT - 1.) 120, 23, 25
C
FOR MACH NO. TELEMETERED FROM A/C, XMT
C
IF CURRENT MACH # LESS THAN 1
C
23 NMET = NMET + 1
GO TO 120
C
IF CURRENT MACH # GREATER THAN 1
C
25 NMET = NMET + 1
XMET = 1. / (SIN(GAMMA + ATAN(1./SQRT(XMT**2-1.))))
XMCT = EXP (4.035E-6 * AMIN1(HA,35300.))
IF (XMET .LE. XMCT .OR. XMT .LE. XMCT) GO TO 120
C
IF CURRENT MACH # GREATER THAN CUTOFF
C
NCET = NCET + 1
C
120 IF (XM - 1.) 122,123,125
C
FOR MACH NO. CALCULATED FROM VELOCITIES.
C
122 IF (LMFG) WRITE (3,3001) FTDLM
LMFG = .FALSE.
IF (LMCFL1) WRITE (4,4001) FTDELM
LMCFL1 = .FALSE.
GOTO 290
C
IF CURRENT MACH # BETWEEN 1 AND CUTOFF
C
123 WRITE (3,3001) XC, YC
NME = WME + 1
SECS = SECS + FTINC
LMFG = .TRUE.
124 IF (LMCFL1) WRITE (4,4001) FTDELM
LMCFL1 = .FALSE.
GOTO 290
C
IF CURRENT MACH # GREATER THAN 1
C
125 XME = 1. / (SIN(GAMMA + ATAN(1./SQRT(XM**2-1.))))
XMC = EXP(4.035E-6 * AMIN1(HA, 35300.))
LMFG = .TRUE.
WRITE (3,3001) XC, YC
NME = WME + 1
SECS = SECS + FTINC
IF (XME .LE. XMC .OR. XM .LE. XMC) GO TO 124
C
IF CURRENT MACH # GREATER THAN CUTOFF
C
`LMCFL1 = .TRUE.
WRITE (4,4001) XC, YC
SCCB = SECB + FTINIC
XKD1C = 2. + 4.53E-6 * HA
IF (HA .GT. 35300.) XKD1C = 2.3929 - 6.6E-6 * HA

130 XND = 0.22 + 1.6E-6 * HA
XKD = XKD1C + (1.04 - XKD1C) * ((XME - XMC) / (XME - 1.)) ** XND
DX = XKD * H / SQRT(XME**2 - 1.)
DYM = H * ((1.+XMC) / XM) * SQRT((XME**2 - XMC**2) / (XME**2 - 1.))
HE = H * COS(GAMMA) + DX * SIN(GAMMA)
DA = (1. + 6.8756E-6 * HA) ** 5.2559
DG = (1. + 6.8756E-6 * H) ** 5.2559
OP = (8.4E3 * SQRT(DA*DG) * (XM**2 - 1.) / HE) ** 0.75
OP = OP * OPFACT

XLPK = 20. * ALOG10(OP) + 127.6
XLCE = XLPK - 26.0
XLAE = 188.7 * ALOG(XLPK) - 825.6

UPDATE RMS VALUES
RMSHE = RMSHE + HE*HE
RMSN = RMSN + XM*XM
RMSCN = RMSCN + XM*XMC
RMSEN = RMSEN + XM*XME

UPDATE STATISTICAL MATRICES
IX = IFIX((XC - RXMIN) / RXCL) + 2
IX = MAXO(MINO(IX, WX), 1)
IY = IFIX((YC - RYMIN) / RYCL) + 2
IY = MAXO(MINO(IY, WY), 1)
IZ = IFIX((ZC - RZMIN) / RZCL) + 2
IZ = MAXO(MINO(IZ, WZ), 1)
IHE = IFIX((HE - RHEMIN) / RHCL) + 2
IHE = MAXO(MINO(IHE, NH), 1)
IM = IFIX((XM - RMMIN) / RMCL) + 2
IM = MAXO(MINO(IM, WM), 1)
IMC = IFIX((XMC - RMMIN) / RMCL) + 2
IMC = MAXO(MINO(IMC, WM), 1)
IME = IFIX((XME - RRMIN) / RMCL) + 2
IME = MAXO(MINO(IME, WM), 1)
IOP = IFIX((OP - ROMIN) / ROCL) + 2
IOP = MAXO(MINO(IOP, WO), 1)
ILPK = IFIX((XLPK - RLPMIN) / RLCL) + 2
ILPK = MAXO(MINO(ILPK, WL), 1)
ILCE = IFIX((XLCE - RLCMIN) / RLCL) + 2
ILCE = MAXO(MINO(ILCE, WL), 1)
ILAE = IFIX((XLAE - RLAGMIN) / RLCL) + 2
ILAE = MAXO(MINO(ILAE, WL), 1)

MXY(IX, IY) = MXY(IX, IY) + 1`
MX(IX) = MX(IX) + 1
MY(IY) = MY(IY) + 1
MZ(IZ) = MZ(IZ) + 1
MNE(IHE) = MNE(IHE) + 1
M(NIM) = MM(IM) + 1
MNC(INC) = MNC(INC) + 1
MME(IHE) = MME(IHE) + 1
MOP(MOP) = MOP(MOP) + 1
MLPK(ILPK) = MLPK(ILPK) + 1
MLCE(ILCE) = MLCE(ILCE) + 1
MLAE(ILAE) = MLAE(ILAE) + 1
NCE = NCE + 1
C
290 FT = FT + FTINC
GOTO 105
300 IF (SECS .GT. 0.0) THEN
   NBOOM = NBOOM + 1
END IF
IF (SECS .GT. 0.0) THEN
   NSUP = NSUP + 1
END IF
C
310 IF (LINCNT .GE. 0) THEN
   LINCNT = -50
   WRITE (6,6001) TITLE
END IF
C
LINCNT = LINCNT + 1
WRITE (6,6002) NTMCH, NNAME, MDATE, SITEC,
1     STARTT(1:2), STARTT(3:4), STARTT(5:6), STARTT(7:8),
2     ENDT(1:2), ENDT(3:4), ENDT(5:6), ENDT(7:8),
3     ACTYPE, ACTAIL, SECS, SEC
TSECS = TSECS + SECS
TSECB = TSECB + SEC
C
C
C............................................................................
C- CALL RAY TRACING ROUTINES
C
7878 CONTINUE
   IF (.NOT.RAYTRC) CALL RTRACE
C
   GOTO 40
C
C
500 CONTINUE
C
   REPORT TOTALS
C
   WRITE (6,6003) NSUP, TSECS, TSECB, NBOOM
   WRITE (6,6004) NME, NCE, NMET, NCE
C
C IF (.NOT. STATFG) OR. (SEC .EQ. 0.0)) GOTO 600
C
C PRINT STATISTICAL SUMMARY
C
FNCE = FLOAT(NCE)
RMSSH = SORT(RMSHE/FNCE)
RMSSMN = SORT(RMSSMN/FNCE)
RMSCMN = SORT(RMSCMN/FNCE)
RMSEM = SORT(RMSEM/FNCE)

WRITE (6,6000) TITLE
WRITE (6,6010) IDX, RXMIN, RXCL, MX
WRITE (6,6010) IDY, RYMIN, RYCL, MY
WRITE (6,6010) IDZ, RZMIN, RZCL, MZ
WRITE (6,6010) IDHE, RHMIN, RHCL, RHEHE, MHE
WRITE (6,6010) IDM, RMMIN, RMCL, RMHM, MM
WRITE (6,6010) IDMC, RMMIN, RMCL, RMSCMN, MMC
WRITE (6,6010) IDME, RMMIN, RMCL, RNSEMN, MME
WRITE (6,6010) IDOP, ROMIN, ROCL, MOP
WRITE (6,6010) IDLPK, RLPMIN, RLCL, MLPK
WRITE (6,6010) IDLCE, RLCMIN, RLCL, MICE
WRITE (6,6010) IDLAE, RLAMIN, RLCL, MLAE
WRITE (6,6011) NME, NCE
C
DO 831 J = 1,52
DO 832 I = 1,NY
IF (MXY(I,J,K).EQ.0.0) THEN
AMXY(I,J,K) = '   '
ELSE
WRITE(C8F(14),(14)) MXY(I,J,K)
READ(C8F(14),(14)) AMXY(I,J,K)
ENDIF
832 CONTINUE
831 CONTINUE

WRITE (6,6000) TITLE
WRITE (6,6010) IDX, RXMIN, RXCL
WRITE (6,6010) IDY, RYMIN, RYCL
DO 510 JY=1,NY
IY = NY - JY + 1
510 WRITE (6,6021) IY, (MXY(IY,JY), IY=1,52)
WRITE (6,6000) TITLE
WRITE (6,6010) IDX, RXMIN, RXCL
WRITE (6,6010) IDY, RYMIN, RYCL
DO 520 JY=1,NY
IY = NY - JY + 1
520 WRITE (6,6021) IY, (MXY(IY,JY), IY=31,NX)
C
SET FLAGS FOR VARIOUS GCP OUTPUTS
C
600 CONTINUE
GCPMH = MACHFG
GPCPSM = BOOMFG
GPCPCN = CONTFG
GPCPFL = GPCPMH .OR. GPCPSM .OR. GPCPCN

CLOSE(51)
CLOSE(52)
CALL STOREC(TITLE,GPCPFL, GPCPMH, GPCPSM)
WRITE (6,6000)
STOP

C FORMAT STATEMENTS
C
2001 FORMAT (A16, A8, A8, A6, A8)
2002 FORMAT (2X, A10, 2X, 12)
2003 FORMAT (2X, 2F2.0, F4.0, F6.0, 3F8.0, F6.0, 3F6.0)
2223 FORMAT (2X, F12.2, 2X, 2F3.0, F5.0, F8.2, 3F10.2, F8.2, 3F8.2)
2004 FORMAT (A8, 218)

C

3001 FORMAT (2F10.0)

C

6000 FORMAT (1H1)
6001 FORMAT ('1' // ' ', 10X, 'TITLE: ', A70 //
1 ' ', 5EX, 'STARTING FINISHING' /
2 ' ', 35X, 'MISSION SITE TIME ' TIME
3 A/C A/C SUPERSONIC BOOM' /
4 ' ', 12X, 'NO MISSION NAME DATE LOCATION HR
5 M.SEC'S HR M.SEC'S TIME TIME TITLE: ' TIME ' TIME
6 ' ', 10X, '--- ----------------- --- ---------
7------------------ -------- ------ ---------
8')
6002 FORMAT (' ', 10X, I5, 2X, A16, 2X, A8, 2X, A10, 2(2X, A2, ' '):
1 A2, ' ', A2, ' ', A2), 2X, A6, 2X, A8, 2X, F7.1, 5X,
2 F7.1)
6003 FORMAT ('0', 101X, '---------- ------ ----------' /
1 ' ', 17X, 'NUMBER OF SUPERSONIC SORTIES(FLIGHTS)': I5,
2 ' ', 33X, 'TOTAL: ', F9.1, 3X, F9.1 /
3 ' ', 17X, 'NUMBER OF BOOM PRODUCING SORTIES(FLIGHTS)': I5
6004 FORMAT ('0', 17X, 'USING MACH NO CALCULATED FROM GROUND'
1 ' ', 'VELOCITIES', /
2 ' ', 21X, 'TOTAL SUPERSONIC TIME =', I10, ' SECONDS', /
3 ' ', 21X, 'TOTAL BOOM PRODUCING TIME =', I10, ' SECONDS', /
4 ' ', 17X, 'USING TELEMETERED MACH NO CALCUATED FROM AIRSPEED', /
5 ' ', 21X, 'TOTAL SUPERSONIC TIME =', I10, ' SECONDS', /
3 ' ', 21X, 'TOTAL BOOM PRODUCING TIME =', I10, ' SECONDS', /
6006 FORMAT (1HO, ' NON-SUPERSONIC A/C')
6010 FORMAT (1HO, 10X, A20, 'LOWER BOUND CELL 2 =', F10.1,
1 ' CELL SIZE =', F9.3 / (1H, 10X, 2015))
6011 FORMAT (1HO, 10X, 'TIME GREATER THAN MACH 1.0 (SEC) =', I6,
1 ' TIME GREATER THAN CUTOFF MACH NO (SEC) =', I6)
6012 FORMAT (1H, 10X, 'TITLE: ', A)
6014 FORMAT (1HO, 10X, A20, 'LOWER BOUND CELL 2 =', F10.1,
1 ' CELL SIZE =', F9.3, ' RMS =', F9.3 /
DATA STATEMENTS

DATA IDX, IDY, IDZ / 'X-COORD', 'Y-COORD', 'Z-COORD' /  
DATA IDNE, IDM / 'EFFECTIVE HEIGHT', 'MACH NUMBER' /  
DATA IDMC, IDME / 'CUTOFF MACH NO.', 'EFFECTIVE MACH NO.' /  
DATA IDLPK / 'OVERPRESSURE (PSI)', 'PEAK LEVEL' /  
DATA IDLCE, IDLAE / 'C-LEVEL', 'A-LEVEL' /  
DATA MX, NY, NZ, MM, NM, ML, NO / 52, 52, 52, 52, 59, 79, 59 /  
DATA RXMIN, RXCL / -132000., 5280. /  
DATA RYMIN, RYCL / -132000., 5280. /  
DATA RZMIN, RZCL / 0., 1000. /  
DATA RHEMIN, RHECL / 0., 1000. /  
DATA RMIN, RMCL / 1.00, 0.02 /  
DATA ROMIN, ROCCL / 0.0, 0.25 /  
DATA RLPMIN, RLCMIN, RLMIN, RLC / 115.0, 90.0, 80.0, 0.5 /  
END
SUBROUTINE SOUND (ZC, CO)

C ROUTINE TO OBTAIN SPEED OF SOUND, CO (FT/SEC), AT ALTITUDE ZC (FT)

COMMON /ATMSPH/ GAM, C, U, V, DCDZ, DUDZ, DUDC, D2CDZ2, D2UDZ2, D2VDZ2, RHO

ZCM = 0.3048 * ZC
CALL FNDLYR (ZCM, 1)
CALL AIR (ZCM)
CO = C / 0.3048
RETURN
END
SUBROUTINE OPFIND (ACTYPE,OPFACT,ACWT)

ROUTINE TO LOOK UP A/C OVERPRESSURE FACTOR

CHARACTER*6 ACTYPE, ACTABL(30)

REAL OPFACT, OPTABL(30), ACWT, WTABL(30)

OPFACT = 0.0

DO 20 I=1,30
IF (ACTABL(I) .EQ. '') RETURN
IF (ACTYPE .NE. ACTABL(I)) GOTO 20
OPFACT = OPTABL(I)
ACWT = WTABL(I)
RETURN
20 CONTINUE
RETURN

DATA ACTABL / 'A-4', 'A-6', 'A-7', 'A-10', 'AV-8', 'F-4',
1 'F-5', 'F-8', 'F-14', 'F-15', 'F-16', 'F-18',
2 'F-104', 'F-106', 'F-111', 'OV-10', '14*1' /

DATA OPTABL / 1.0, 1.0, 1.0, 1.0, 0.93,
1 0.76, 0.86, 1.00, 1.00, 0.80, 0.91,
2 0.89, 1.08, 1.11, 1.0, 14*1.0 /

DATA WTABL / 1.0, 1.0, 1.0, 1.0, 1.0,
1 1.0, 1.0, 1.0, 1.0, 16040.0, 1.0,
2 16040.0, 1.0, 1.0, 1.0, 14*1.0 /

END
SUBROUTINE RNGALT (SITLC, RZMIN)
C
DIMENSION SITE(20), SITALT(20)
C
CHARACTER*10 SITE
CHARACTER*(*) SITLC
C
DO 20 I=1,20
RZMIN = SITALT(I)
IF (SITE(I) .EQ. ' ') RETURN
IF (SITLC .EQ. SITE(I)) RETURN
20 CONTINUE
C
RZMIN = 0.
RETURN
C
DATA SITE/'OCEANA', 'TYNDALL', 'LUKE', 'HOLLOMAN', 'NELLIS'
1 , 'YUMA', 'LLLL', 'BBBB', 'TTTT', 10**1 '/
DATA SITALT / 0., 0., 750., 4500., 5500., 800., 1000.0, 10000.0,
1 3800.0, 0.0, 10*0. '/
END
SUBROUTINE RNGLL (SITLC, LAT, LONG)

DIMENSION SITE(20), SITLAT(20), SITLON(20)

CHARACTER*10 SITE, SITLAT, SITLON, LAT, LONG
CHARACTER*(*) SITLC

DO 20 I=1,20
LAT = SITLAT(I)
LONG = SITLON(I)
IF (SITE(I) .EQ. ' ') RETURN
IF (SITLC .EQ. SITE(I)) RETURN
20 CONTINUE

LAT = ' '
LONG = ' '
RETURN

DATA SITE/'OCEANA', 'TYNDALL', 'LUKE', 'HOLLoman', 'NELLIS'
   1 , 'YUMA', 'UNKNOWN' /
DATA SITLAT/' 36 00.0 W', ' 29 32.0 W', ' 32 23.48N', ' 33 48.0 N'
   1 , ' 36 50.29N', ' 32 29.24N', 'UNKNOWN' /
DATA SITLON/' 75 10.0 W', ' 84 37.0 W', '113 15.0 W', '106 25.0 W'
   1 , '115 25.36W', '113 52.56W', 'UNKNOWN' /
END
REAL FUNCTION DISADJ (DLAT, DYC, HE)

C

FUNCTION TO CALCULATE DISTANCE ADJUSTMENT IN Db
FROM THE EFFECTIVE HEIGHT (HE) TO A LATERAL
SIDE LINE POINT AT A DISTANCE (DLAT) FROM
THE FLIGHT TRACK

C

DISRAT = SQRT(DLAT**2 + HE**2) / DYC
HEDYC = 15. * ALOG10(HE/DYC)

C

IF (DISRAT .LT. 0.8) THEN
DISADJ = -15. * ALOG10(DISRAT) + HEDYC
RETURN
C

ELSEIF (DISRAT .LT. 1.0) THEN
DISADJ = -118.1885 * ALOG10(DISRAT) - 10. + HEDYC
RETURN
C

ELSE
DISADJ = -25. * ALOG10(DISRAT) - 10. + HEDYC
RETURN
ENDIF
C

END
SUBROUTINE GETALF (X,Y,XG,YGALPHA,BETA)

SUBROUTINE TO CALCULATE ALPHA AND BETA FACTORS FOR EXPOSURE LEVEL INTERPOLATION

DIMENSION ALF(2), X(4), Y(4)
LOGICAL FG1, FG2

P = Y(3) - Y(4)
Q = X(2) - X(1) - X(3) + X(4)
R = X(3) - X(4)
S = Y(2) - Y(1) - Y(3) + Y(4)
T = X(1) - X(4)
U = Y(4) - YG
V = Y(1) - Y(4)
W = X(4) - XG

A = P*Q - R*S
B = P*T + Q*U - R*V - S*W
C = U*T - W*V

IF (ABC(A) .LE. 1.E-6) THEN
  ALF(1) = -C/B
  ALF(2) = 0.0
  GOTO 30
ELSE
  RAD = B**2 - 4.0*A*C
  IF (RAD .LT. 0.0) STOP91
  IF (RAD .LT. 0.0) RAD = ABS(RAD)
  RAD = SQRT(RAD)
  ALF(1) = (-B + RAD) / (2.*A)
  ALF(2) = (-B - RAD) / (2.*A)
  GOTO 30
ENDIF

FG1 = (ALF(1) .GE. 0.0 .AND. ALF(1) .LE. 1.0)
FG2 = (ALF(2) .GE. 0.0 .AND. ALF(2) .LE. 1.0)

IF (FG1 .AND. FG2) GOTO 40
IF (FG1) ALPHA = ALF(1)
IF (FG2) ALPHA = ALF(2)
GOTO 60

DO 50 I=1,2
YA = Y(4) + ALF(1) * (Y(3) - Y(4))
XA = X(4) + ALF(1) * (X(3) - X(4))
YB = Y(1) + ALF(1) * (Y(2) - Y(1))
XB = X(1) + ALF(1) * (X(2) - X(1))
IF (YB .EQ. YA) THEN
  IF (YG .NE. YA) GOTO 50
ELSE
  Z = (YG-YA)/(YB-YA)
  IF (Z .LT. 0.0 .OR. Z .GT. 1.0) GOTO 50
ENDIF
C
IF (XB .EQ. XA) THEN
  IF (XG .NE. XA) GOTO 50
ELSE
  Z = (XG-XA)/(XB-XA)
  IF (Z .LT. 0.0 .OR. Z .GT. 1.0) GOTO 50
ENDIF
C
ALPHA = ALF(1)
GOTO 60
50 CONTINUE
STOP92
C
60 YBMYA = ALPHA * S + V
XBMXA = ALPHA * Q + T
IF (ABS(YBMYA) .GT. ABS(XBMXA)) THEN
  BETA = (TG - ALPHA*P - Y(4)) / YBMYA
  GOTO 70
ELSE
  BETA = (XG - ALPHA*R - X(4)) / XBMXA
  GOTO 70
ENDIF
C
70 ALF1 = ALF(1)
ALF2 = ALF(2)
RETURN
C
END
SUBROUTINE MAKETR (X, Y, XT, YT)

SUBROUTINE TO MAKE TWO TRIANGLES OUT OF A FOUR-SIDED POLYGON

DIMENSION X(4), Y(4), XT(3,2), YT(3,2)

X1 = X(1)
X2 = X(2)
X3 = X(3)
X4 = X(4)
Y1 = Y(1)
Y2 = Y(2)
Y3 = Y(3)
Y4 = Y(4)

ADJUST POINTS SO SLOPES CAN'T BE INFINITE

IF (ABS(X2 - X1) .LT. 0.01) X2 = X2 + 0.1
IF (ABS(X3 - X4) .LT. 0.01) X3 = X3 + 0.1
IF (ABS(X2 - X3) .LT. 0.01) X3 = X3 + 0.1

FM1 = (Y2 - Y1) / (X2 - X1)
FB1 = Y1 - FM1*X1
FM2 = (Y3 - Y4) / (X3 - X4)
FB2 = Y3 - FM2*X3

IF (FM1 .EQ. FM2) GOTO 40

XC = (FB1 - FB2) / (FM2 - FM1)
YC = (FM2*FB1 - FM1*FB2) / (FM2 - FM1)
TEMP1 = (XC - X1) / (X2 - X1)

IF (TEMP1 .GE. 0.0 .AND. TEMP1 .LE. 1.0) THEN

XT(1,1) = X1
YT(1,1) = Y1
XT(2,1) = XC
YT(2,1) = YC
XT(3,1) = X4
YT(3,1) = Y4
XT(1,2) = X3
YT(1,2) = Y3
XT(2,2) = XC
YT(2,2) = YC
XT(3,2) = X2
YT(3,2) = Y2
RETURN
ENDIF

V12V14 = (X2-X1)*(Y4-Y1) - (Y2-Y1)*(X4-X1)
\[ v_{23V21} = (X_3 - X_2)(Y_1 - Y_2) - (Y_3 - Y_2)(X_1 - X_2) \]
\[ v_{34V32} = (X_4 - X_3)(Y_2 - Y_3) - (Y_4 - Y_3)(X_2 - X_3) \]
\[ v_{41V43} = (X_1 - X_4)(Y_3 - Y_4) - (Y_1 - Y_4)(X_3 - X_4) \]

\[ v_{12V14} = \text{SIGN}(1., v_{12V14}) \]
\[ v_{23V21} = \text{SIGN}(1., v_{23V21}) \]
\[ v_{34V32} = \text{SIGN}(1., v_{34V32}) \]
\[ v_{41V43} = \text{SIGN}(1., v_{41V43}) \]

\[ v_{SUM} = v_{12V14} + v_{23V21} + v_{34V32} + v_{41V43} \]

IF (ABS(v_{SUM}) .EQ. 4.) GOTO 40

IF (v_{SUM} .GT. 0.) GOTO 30
\[ v_{12V14} = -v_{12V14} \]
\[ v_{23V21} = -v_{23V21} \]
\[ v_{34V32} = -v_{34V32} \]
\[ v_{41V43} = -v_{41V43} \]

30 IF (v_{12V14} .EQ. -1. .OR. v_{34V32} .EQ. -1.) GOTO 60
IF (v_{23V21} .EQ. -1. .OR. v_{41V43} .EQ. -1.) GOTO 50
STOP 93

40 D13 = ((X_3 - X_1)^2 + (Y_3 - Y_1)^2) +
D24 = ((X_2 - X_4)^2 + (Y_2 - Y_4)^2)

C

50 XT(1,1) = X1
YT(1,1) = Y1
XT(2,1) = X2
YT(2,1) = Y2
XT(3,1) = X4
YT(3,1) = Y4
XT(1,2) = X2
YT(1,2) = Y2
XT(2,2) = X3
YT(2,2) = Y3
XT(3,2) = X4
YT(3,2) = Y4
RETURN

C

60 XT(1,1) = X1
YT(1,1) = Y1
XT(2,1) = X2
YT(2,1) = Y2
XT(3,1) = X3
YT(3,1) = Y3
XT(1,2) = X1
YT(1,2) = Y1
XT(2,2) = X3
YT(2,2) = Y3
XT(3,2) = X4
YT(3,2) = Y4
RETURN

C
BLOCK DATA DICK

COMMON /GRID/ GRD XO, XGS, GRDXMX, GRDYO, YGS, GRDYMX,
   LIMAXO, LIMAYO, LIMBXO, LIMBYO,
   LIMAX1, LIMAY1, LIMBX1, LIMBY1

DATA GRD XO, XGS, GRDXMX / -126250., 2500., 126250. /
DATA GRDYO, YGS, GRDYMX / -126250., 2500., 126250. /
END
MODULE NAME: LEXPACK
MODULE TYPE: PACKAGE

OVERVIEW:

THIS PACKAGE IS USED TO PERFORM THE LEXICAL ANALYSIS NECESSARY FOR PARSING. THE MAIN PURPOSE FOR COMBINING THESE PROCEDURES IN THIS PACKAGE IS TO REDUCE THE SCOPE OF DATA COMMUNICATION TO THE SUBROUTINES CONTAINED IN THE PACKAGE.

INTERFACE:

GETOKN ( P1, P2, P3 )

P1 ::= [CHARACTER "(*)] STRING COMPOSING THE TOKEN
P2 ::= [INTEGER] LENGTH OF THE CHARACTER STRING
P3 ::= [INTEGER] VALUE OF THE TOKEN

INTERNAL SUBROUTINES & FUNCTIONS:

GETLIN() ; READS ONE LINE FROM SOURCE FILE
GETCHR() ; RETURNS THE CURRENT INPUT CHARACTER FROM BUFFER
ADDCHR() ; ADDS CURRENT INPUT CHARACTER TO THE TOKEN STRING
FILEDT() ; NON-EXECUTABLE- INITIALIZES FILE UNIT NUMBERS

PROGRAMMER: BRUCE B. LACEY
DATE : 10-OCT-85

SHARED DATA:

BLOCK DATA FILEDT

COMMON /IOC0M/ INFILE, LISTFL

INTEGER INFILE, LISTFL

DATA INFILE /5/, LISTFL/6/

END
MODULE NAME: LEXPACK\GETLIN
MODULE TYPE: SUBROUTINE

OVERVIEW:

THIS SUBROUTINE IS USED TO FILL THE INPUT BUFFER WITH ONE LINE OF SOURCE CODE FROM 'INFILE'. THE LINE OF CODE IS THEN ECHOED TO THE OUTPUT FILE 'LISTFL'. WHEN THE END OF FILE IS REACHED, THE FLAG, 'ENDFLG' IS SET TO TRUE.

INVOCATION:

[CALL] GETLIN ( P1, P2, P3 )

P1 ::= [CHARACTER (*{*})] STRING CONTAINING A LINE OF SOURCE CODE
P2 ::= [INTEGER] POINTER TO THE CURRENT POSITION IN P1
P3 ::= [LOGICAL] FLAG SIGNALING THE END OF FILE ON 'INFILE'

VARIABLE DICTIONARY:

BUFFER ; P1
BUFPOS ; P2
ENDFLG ; P3
INFILE ; INPUT FILE CONTAINING SOURCE CODE FOR PARSING
LISTFL ; OUTPUT FILE FOR ECHOING 'INFILE' WITH ERRORS

CALLER MODULES:

[FUNCTION] LEXPACK\GETCHR()

CALLED MODULES:

...NONE...

PROGRAMMER: BRUCE B. LACEY
DATE : 10-OCT-85
REVISIONS :

SUBROUTINE GETLIN ( BUFFER, BUFPOS, ENDFLG )

COMMON /IOCOM/ INFILE, LISTFL

CHARACTER *{*)) BUFFER
LOGICAL ENDFLG
INTEGER BUFPOS, INFILE, LISTFL, LINENM

SAVE LINENM

DATA LINENM /0/
C IF FIRST LINE THEN WRITE THE HEADER CARD.
   IF (LINENM.EQ.0) THEN
     WRITE(LISTFL,01)
01   FORMAT('1', ' SOURCE LISTING://)
   END IF
C GET A LINE OF CODE FROM INFILE
READ(UNIT=INFILE,FMT='(A)',ERR=10,END=20)BUFFER
C NOW ECHO THE INPUT LINE TO THE LISTING FILE
   LINENM = LINENM + 1
   WRITE(UNIT=LISTFL,FMT=05) LINENM, BUFFER
05   FORMAT(1X,I3,: ',A)
   BUFFPOS = 1
C IF EVERYTHING WENT OK THEN RETURN
RETURN
C **-EXCEPTIONS-**
C RAISE I/O ERROR
10  WRITE(*,FMT=15)
15  FORMAT('I/O ERROR WHILE READING SOURCE FILE...')
STOP
C SET END OF FILE FLAG
20  ENDFLG = .TRUE.
RETURN
END
MODULE NAME: LEXPACK\GETCHR

MODULE TYPE: CHARACTER FUNCTION SUBROUTINE

OVERVIEW:

This function subroutine is used to return the current input character from the input buffer, updating the pointer "BUFPOS" accordingly.

INVOCATION:

[X = ] GETCHR( P1, P2 )

P1 ::= [INTEGER] pointer to the current position in P1
P2 ::= [LOGICAL] flag signaling end of file on 'INFILE'

VARIABLE DICTIONARY:

BUFFER ; STRING*80 containing current line of source
BUFPOS ; P1
ENDFLG ; P2
INFILE ; INPUT file containing source code to be parsed
LISTFL ; OUTPUT file for echoing source code and errors

CALLER MODULES:

[SUBROUTINE] LEXPACK\GETOKN()

CALLED MODULES:

[SUBROUTINE] LEXPACK\GETLIN()

PROGRAMMER: BRUCE B. LACEY
DATE : 10-OCT-85

REVISIONS :

CHARACTER*1 FUNCTION GETCHR ( BUFPOS, ENDFLG )

CHARACTER*80 BUFFER
LOGICAL ENDFLG
INTEGER BUFPOS, INFILE, LISTFL

SAVE BUFFER

IF (BUFPOS.GE.LEN(BUFFER)) THEN
  C FILL THE BUFFER UP
  CALL GETLIN(BUFFER,BUFPOS,ENDFLG)
END IF

C NOW PULL A CHARACTER FROM THE BUFFER
GETCHR = BUFFER ( BUFPOS : BUFPOS )
BUFPOS = BUFPOS + 1
RETURN
END
MODULE NAME: LEXPACK\ADDCHR
MODULE TYPE: SUBROUTINE

OVERVIEW:

THIS SUBROUTINE IS USED TO CONCATENATE THE GIVEN CHARACTER
WITH THE TOKEN STRING BEING BUILT. THE LENGTH OF THE STRING STOR
IN 'TKNLEN' IS ALSO INCREMENTED BY ONE REPRESENTING THE CURRENT
LENGTH OF THE TOKEN STRING.

INVOCATION:

[CALL] ADDCHR ( P1, P2, P3 )

P1 ::= [CHARACTER*1] TO BE CONCATENATED TO P2
P2 ::= [CHARACTER*(*)] TOKEN STRING
P3 ::= [INTEGER] LENGTH OF P2

VARIABLE DICTIONARY:

CC ; P1
TKNLEN ; P3
TKNSTR ; P2

CALLER MODULES:

[SUBROUTINE] LEXPACK\GETOKN()

CALLED MODULES:

[INTRINSIC FUNCTION] LEN()

PROGRAMMER: BRUCE B. LACEY
DATE : 10-OCT-85
REVISIONS :

SUBROUTINE ADDCHR (CC, TKNSTR, TKNLEN )

INTEGER TKNLEN
CHARACTER*1 CC
CHARACTER*(*) TKNSTR

TKNLEN = TKNLEN + 1
IF(TKNLEN.LE.LEN(TKNSTR)) THEN
    TKNSTR(TKNLEN:TKNLEN) = CC
END IF

RETURN
END
MODULE NAME: LEXPACK\GETOKN

MODULE TYPE: SUBROUTINE

OVERVIEW:

**OVERVIEW:**

THIS SUBROUTINE WILL LEXICALLY ANALYZE AN INPUT STREAM OF CHARACTERS, RETURNING THE STRING COMPOSING THE TOKEN, THE LENGTH OF THE STRING, AND A TOKEN VALUE. THE TOKEN VALUE WILL CORRESPOND TO THE FOLLOWING:

```

<table>
<thead>
<tr>
<th>TOKEN</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>' '</td>
<td>1</td>
</tr>
<tr>
<td>'/'</td>
<td>2</td>
</tr>
<tr>
<td>','</td>
<td>3</td>
</tr>
<tr>
<td>'.'</td>
<td>4</td>
</tr>
<tr>
<td>E.O.F.</td>
<td>5</td>
</tr>
<tr>
<td>LEX. ERROR</td>
<td>6</td>
</tr>
<tr>
<td>IDENTIFIER</td>
<td>7</td>
</tr>
<tr>
<td>INTEGER</td>
<td>8</td>
</tr>
<tr>
<td>REAL</td>
<td>9</td>
</tr>
<tr>
<td>'ACWTN'</td>
<td>10</td>
</tr>
<tr>
<td>'AIRCRAFT'</td>
<td>11</td>
</tr>
<tr>
<td>'ALL'</td>
<td>12</td>
</tr>
<tr>
<td>'BOOMTRK'</td>
<td>13</td>
</tr>
<tr>
<td>'CLDN'</td>
<td>14</td>
</tr>
<tr>
<td>'CONTOUR'</td>
<td>15</td>
</tr>
<tr>
<td>'CSSEL'</td>
<td>16</td>
</tr>
<tr>
<td>'DATE'</td>
<td>17</td>
</tr>
<tr>
<td>'MACHTRK'</td>
<td>18</td>
</tr>
<tr>
<td>'MISSION'</td>
<td>19</td>
</tr>
<tr>
<td>'PKOP'</td>
<td>20</td>
</tr>
<tr>
<td>'SITE'</td>
<td>21</td>
</tr>
<tr>
<td>'STAT'</td>
<td>22</td>
</tr>
<tr>
<td>'TIME'</td>
<td>23</td>
</tr>
<tr>
<td>'TITLE'</td>
<td>24</td>
</tr>
<tr>
<td>'WIDTH'</td>
<td>25</td>
</tr>
<tr>
<td>'FFT'</td>
<td>26</td>
</tr>
<tr>
<td>'STATS'</td>
<td>27</td>
</tr>
<tr>
<td>'SIGNAT'</td>
<td>28</td>
</tr>
<tr>
<td>'STATONLY'</td>
<td>29</td>
</tr>
<tr>
<td>'SCRCHPAD'</td>
<td>30</td>
</tr>
<tr>
<td>'OB'</td>
<td>31</td>
</tr>
<tr>
<td>'PSF'</td>
<td>32</td>
</tr>
<tr>
<td>'NEW'</td>
<td>33</td>
</tr>
</tbody>
</table>
```

**INVOCATION:**

```

CALL GETOKN ( P1, P2, P3, P4 )
```

**P1** := [CHARACTER(*)] STRING COMPOSING THE TOKEN

**P2** := [INTEGER] LENGTH OF THE CHARACTER STRING (P1)

**P3** := [INTEGER] VALUE OF THE TOKEN
VARIABLE DICTIONARY:

ACPTST ; SYMBOL REPRESENTING STATEMENT LABEL 100
BUFSO ; CURRENT POSITION IN THE INPUT BUFFER
CHAD ; DUMMY-ARGUMENT FOR LOGICAL STATEMENT FUNCTIONS
ENDFLG ; FLAG SIGNALING THE END OF FILE ON 'INFILE'
INCHAR ; CURRENT INPUT CHARACTER FROM BUFFER
INFILE ; SOURCE FILE CORRRAING CODE TO BE PARSED
KWORDS ; TABLE CONTAINING CERTAIN KEYWORDS TO BE RECOGNIZED
LISTFL ; OUTPUT FILE FOR ECHOING SOURCE CODE AND ERRORS
PRSCTR ; P4
STATE1 ; SYMBOL REPRESENTING STATEMENT LABEL 1
STATE2 ; SYMBOL REPRESENTING STATEMENT LABEL 2
STATE3 ; SYMBOL REPRESENTING STATEMENT LABEL 3
STATE4 ; SYMBOL REPRESENTING STATEMENT LABEL 4
STATE5 ; SYMBOL REPRESENTING STATEMENT LABEL 5
STATE6 ; SYMBOL REPRESENTING STATEMENT LABEL 6
STATE7 ; SYMBOL REPRESENTING STATEMENT LABEL 7
STATE8 ; SYMBOL REPRESENTING STATEMENT LABEL 8
TBLIDX ; LOOP CONTROL VARIABLE FOR INDEXING TABLE 'KWORDS'
TKNLEN ; P2
TKNSTR ; P1
TKNVAL ; P3

CALLER MODULES:

[SUBROUTINE] PRSPACK\PARSE()
INTEGER BUFPOS, INFILE, LISTFL
LOGICAL ENDFLG, PRSCTR
CHARACTER*8 KWORDS(MAXKEYS)
CHARACTER*1 GETCHR

INTEGER STATE1, STATE2, STATE3, STATE4, STATE5
INTEGER STATE6, STATE7, STATE8, ACPTST, TBLIDX
INTEGER TKNLEN, TKNVAL
LOGICAL LETTER, DIGIT, COMMA, SLASH, HYPHEN, BLANK
LOGICAL PERIOD
CHARACTER*1 INCHAR, CHAR
CHARACTER*(*) TKNSTR

SAVE KWORDS, BUFPOS, ENDFLG

DATA BUFPOS /80/, ENDFLG /.FALSE./
DATA KWORDS /'ACWTN', 'AIRCRAFT', 'ALL', +
  'BOOMTRK', 'CLDN', 'CONTOJR', +
  'CSEL', 'DATE', 'MACHTRK', +
  'MISSION', 'PKOP', 'SITE', +
  'STAT', 'TIME', 'TITLE', +
  'WIDTH', 'FFT', 'STATS', +
  'SIGNAT', 'STATIONLY', 'SCRCPAD', +
  'DB', 'PSF', 'NEW' /

LETTER(CHAR) = ((CHAR.GE.'A').AND.(CHAR.LE.'Z'))
DIGIT (CHAR) = ((CHAR.GE.'0').AND.(CHAR.LE.'9'))
COMMA (CHAR) = ( CHAR.EQ.',' )
SLASH (CHAR) = ( CHAR.EQ. '/' )
HYPHEN(CHAR) = ( CHAR.EQ. '-' )
PERIOD(CHAR) = ( CHAR.EQ. '.' )
BLANK (CHAR) = ( CHAR.EQ. ' ' )

ASSIGN 1 TO STATE1
ASSIGN 2 TO STATE2
ASSIGN 3 TO STATE3
ASSIGN 4 TO STATE4
ASSIGN 5 TO STATE5
ASSIGN 6 TO STATE6
ASSIGN 7 TO STATE7
ASSIGN 8 TO STATE8
ASSIGN 100 TO ACPTST

TKNLEN = 0
TKNSTR = '

C STATE 1: START STATE
  INCHAR = GETCHR(BUFPOS,ENDFLG)
  TEST TO MAKE SURE THAT WE ARE NOT AT THE END OF THE FILE
  IF (ENDFLG.EQV..TRUE.) THEN
    TKNVAL = 5
    RETURN
  END IF
IF (BLANK(INCHAR).EQV..TRUE.) THEN
  GO TO STATE1
ELSE IF (LETTER(INCHAR).EQV..TRUE.) THEN
  GO TO STATE2
ELSE IF (DIGIT(INCHAR).EQV..TRUE.) THEN
  GO TO STATE3
ELSE IF (COMMA(INCHAR).EQV..TRUE.) THEN
  GO TO STATE4
ELSE IF (SLASH(INCHAR).EQV..TRUE.) THEN
  GO TO STATE5
ELSE IF (HYPHEN(INCHAR).EQV..TRUE.) THEN
  GO TO STATE6
ELSE IF (PERIOD(INCHAR).EQV..TRUE.) THEN
  GO TO STATE7
ELSE IF (PERIOD(INCHAR).EQV..TRUE.) THEN
  GO TO STATE8
ELSE
  WRITE(LISTFL,10)INCHAR
  10  FORMAT('=',+---WARMING: UNKNOWN CHARACTER IN INPUT --> ('
    +',A,''),---')
  TKNVAL = 6
  RETURN
END IF

C STATE 2: IDENTIFIER STATE
2  TKNVAL = 7
  CALL ADDCHR(INCHAR,TKNSTR,TKNLEN)
  INCHAR = GETCHR(BUFPOS,ENDFLG)
  IF (.NOT.PRSCTR) THEN
    IF ((LETTER(INCHAR).EQV..TRUE.).OR.
      + (DIGIT(INCHAR).EQV..TRUE.).OR.
      + (HYPHEN(INCHAR).EQV..TRUE.)) THEN
      GO TO STATE2
    ELSE
      GO TO ACPTST
    END IF
  ELSE
    IF (.NOT.BLANK(INCHAR)).AND. (.NOT.COMMA(INCHAR))) THEN
      GO TO STATE 2
    ELSE
      GO TO ACPTST
    END IF
  END IF

C STATE 3: INTEGER STATE
3  TKNVAL = 8
  IF (PRSCTR .EQV..TRUE.) THEN
    GO TO STATE2
  END IF
  CALL ADDCHR(INCHAR,TKNSTR,TKNLEN)
  INCHAR = GETCHR(BUFPOS,ENDFLG)
  IF (DIGIT(INCHAR).EQV..TRUE.) THEN
    GO TO STATE3
  ELSE IF (PERIOD(INCHAR).EQV..TRUE.) THEN
    GO TO STATE4
  ELSE

GO TO ACPTST
END IF

C STATE 4: REAL NUMBER STATE
4 TKNVAL = 9
CALL ADDCHR(INCHAR, TKNSTR, TKNLEN)
INCHAR = GETCHR(BUFPOS, EDFLG)
IF (DIGIT(INCHAR).EQV..TRUE.) THEN
   GO TO STATE4
ELSE
   GO TO ACPTST
END IF

C STATE 5: COMMA STATE
5 TKNVAL = 1
CALL ADDCHR(INCHAR, TKNSTR, TKNLEN)
GO TO ACPTST

C STATE 6: SLASH STATE
6 TKNVAL = 2
CALL ADDCHR(INCHAR, TKNSTR, TKNLEN)
GO TO ACPTST

C STATE 7: HYPHEN STATE
7 TKNVAL = 3
CALL ADDCHR(INCHAR, TKNSTR, TKNLEN)
GO TO ACPTST

C STATE 8: PERIOD STATE
8 TKNVAL = 4
CALL ADDCHR(INCHAR, TKNSTR, TKNLEN)
GO TO ACPTST

C ACPTST : ACCEPT STATE

C CHECK IF THE BUFFER POINTER SHOULD BE RETRACTED.
100 IF (.NOT.BLANK(INCHAR)).AND.(TKNVAL.GE.7)) THEN
   BUFPOS = BUFPOS - 1
END IF

IF (TKNVAL.EQ.7) THEN
   DO 101 TBLIDX = 1, MAXKEYS
      IF (KEYWORDS(TBLIDX).EQ.TKNSTR(1:8)) THEN
         TKNVAL = TBLIDX + 9
      END IF
   C CHECK IF THE TOKEN IS 'STATS' IF SO THEN TKNVAL EQUALS 22
   C THIS IS DUE TO THE FACT THAT 'STAT' AND 'STATS' ARE EQUAL
   IF (TKNVAL .EQ. 27) THEN
      TKNVAL = 22
   END IF
101 CONTINUE
END IF

PRSCTR = .FALSE.
MODULE NAME: PRSPACK

MODULE TYPE: PACKAGE

OVERVIEW:

THIS PACKAGE IS USED TO PERFORM THE PARSING OF THE SOURCE
FILE 'INFIL'. THE METHOD OF PARSE IS A SIMPLE TABLE DRIVEN
PARSE. THE PARSE TABLE IS INITIALIZED IN THE BLOCK DATA SUB-
ROUTINE 'PRSDAT'. THE PARSE TABLE CONSISTS OF THE STATE TRANS-
ITIONS FOR INPUT TOKENS. EACH TIME AN INPUT TOKEN IS RETURNED
BY 'LEXPACK\GETOK()', SUBROUTINE 'PRSPACK\LOOKUP()' IS CALLED
TO DETERMINE THE NEXT STATE TO GO TO BY REFERENCING THE PARSE TABL
WITH THE CURRENT STATE VERSUS THE CURRENT INPUT TOKEN VALUE.
PROGRAM EXECUTION THEN TRANSFERS TO A STATEMENT LABEL REPRESENTIN
THAT STATE, WHERE VARIOUS SEMANTIC ACTIONS ARE PERFORMED TO STORE T
THE INFORMATION IN INTERNAL DATA STRUCTURES.

INTERFACE:

PARSE ( P1 )

P1 ::= [LOGICAL] FLAG REPRESENTING A FATAL PARSE ERROR

INTERNAL SUBROUTINE & FUNCTIONS:

LDATE() ; RETURNS TRUE IF A MONTH, DAY, OR YEAR IS LEGAL
LTIME() ; RETURNS TRUE IF A TIME IS LEGAL
LOOKUP() ; TRANSFERS PROGRAM CONTROL TO THE NEXT STATE
PRSDAT ; NON-EXECUTABLE. SETS THE PARSE TABLE VALUES

PROGRAMMER: BRUCE B. LACEY
DATE : 14-OCT-85
REVISIONS : 11-NOV-86 NEW STATES 'FFT','SIGNAT','STATONLY'

SHARED DATA:

BLOCK DATA PRSDAT
COMMON /PRSCOM/ P1
INTEGER P1(59,33)
MODULE NAME: PRSPACK
MODULE TYPE: PACKAGE

OVERVIEW:

THIS PACKAGE IS USED TO PERFORM THE PARSING OF THE SOURCE FILE 'INFILE'. THE METHOD OF PARSE IS A SIMPLE TABLE DRIVEN PARSE. THE PARSE TABLE IS INITIALIZED IN THE BLOCK DATA SUBROUTINE 'PRSDAT'. THE PARSE TABLE CONSISTS OF THE STATE TRANSITIONS FOR INPUT TOKENS. EACH TIME AN INPUT TOKEN IS RETURNED BY 'LEXPACK\GETOKN()', SUBROUTINE 'PRSPACK\LOOKUP()' IS CALLED TO DETERMINE THE NEXT STATE TO GO TO BY REFERENCING THE PARSE TABLE WITH THE CURRENT STATE VERSUS THE CURRENT INPUT TOKEN VALUE. PROGRAM EXECUTION THEN TRANSFERS TO A STATEMENT LABEL REPRESENTING THAT STATE, WHERE VARIOUS SEMANTIC ACTIONS ARE PERFORMED TO STORE THE INFORMATION IN INTERNAL DATA STRUCTURES.

INTERFACE:

PARSE ( P1 )

P1 ::= [LOGICAL] FLAG REPRESENTING A FATAL PARSE ERROR

INTERNAL SUBROUTINE & FUNCTIONS:

LTE() ; RETURNS TRUE IF A MONTH, DAY, OR YEAR IS LEGAL
LTIME() ; RETURNS TRUE IF A TIME IS LEGAL
LOOKUP() ; TRANSFERS PROGRAM CONTROL TO THE NEXT STATE
PRSDAT ; NON-EXECUTABLE. SETS THE PARSE TABLE VALUES

PROGRAMMER: BRUCE B. LACEY
DATE : 14-OCT-85
REVISIONS : 11-NOV-86 NEW STATES 'FFT', 'SIGNAT', 'STATONLY'

SHARED DATA:

BLOCK DATA PRSDAT
COMMON /PRSCON/ PT
INTEGER PT(59,33)

C: INITIALIZE THE PARSE TABLE.

DATA PT( 1,21) / 2/, PT( 1,24) /47/
DATA PT( 2,12) / 5/, PT( 2, 7) / 3/
DATA PT( 3,17) / 6/, PT( 3,19) / 7/, PT( 3, 1) / 4/
DATA PT( 4, 7) /3/
DATA PT( 5,17) / 6/, PT( 5,19) / 7/
DATA PT( 6,12) /10/, PT( 6, 8) /11/
DATA PT( 7,12) /23/, PT( 7, 7) / 8/
MODULE NAME: PRSPACK\LDATE
MODULE TYPE: LOGICAL FUNCTION SUBROUTINE

OVERVIEW:

THIS FUNCTION SUBROUTINE IS USED TO CHECK IF A CERTAIN PART (I.E. MONTH, DAY, YEAR) IS LEGAL ACCORDING THE INTEGER BOUNDS PERTAINING TO THAT PART OF THE DATE.

INVOCATION:

[X = ] LDATE ( P1, P2 )

P1 := [CHARACTER*2] STRING SPECIFYING SEGMENT TO TEST
P2 := [CHARACTER*(*)] STRING HOLDING THE DATE

VARIABLE DICTIONARY:

FRAG ; P1
FRGSTR ; P2
TSTINT ; INTEGER VARIABLE USED TO COMPARE BOUNDS

CALLER MODULES:

{SUBROUTINE} PRSPACK\PARSE()

CALLED MODULES:

...NONE...

PROGRAMMER: BRUCE B. LACEY
DATE : 14-OCT-85

LOGICAL FUNCTION LDATE(FRAG, FRGSTR)

CHARACTER*2 FRAG
CHARACTER(*)(*) FRGSTR
INTEGER TSTINT

C CONVERT THE CHARACTER STRING TO AN INTEGER
READ(FRGSTR(1:2),FMT='(12)')TSTINT

IF (FRAG.EQ.'MM') THEN
C CHECK IF A LEGAL MONTH
LDATE = ((TSTINT.GE.1).AND.(TSTINT.LE.12))
ELSE IF (FRAG.EQ.'DD') THEN
C CHECK IF A LEGAL DAY
LDATE = ((TSTINT.GE.1).AND.(TSTINT.LE.31))
ELSE IF (FRAG.EQ., 'YY') THEN
  C CHECK IF A LEGAL YEAR
  LDATE = ((TSTINT.GE.1).AND.(TSTINT.LE.99))
END IF
RETURN
END
MODULE NAME: PRSPACK\LTIME

MODULE TYPE: LOGICAL FUNCTION SUBROUTINE

OVERVIEW:

THIS SUBROUTINE FUNCTION IS USED TO TEST IF THE TIME SPECIFIED IS WITHIN THE MILITARY BOUNDS OF 0001-2400 HOURS.

INVOCATION:

[X = ] LTIME ( P1 )

P1 ::= [INTEGER] TIME INPUT FOR TESTING

VARIABLE DICTIONARY:

INTIME ; P1

CALLER MODULES:

[SUBROUTINE] PRSPACK\PARSE()

CALLED MODULES:

...NONE...

PROGRAMMER: BRUCE B. LACEY

DATE : 14-OCT-85

REVISIONS : 

LOGICAL FUNCTION LTIME(INTIME)

LTIME = ((INTIME.GE.0001).AND.(INTIME.LE.2400))

RETURN

END
MODULE NAME: PRSPACK\LOOKUP
MODULE TYPE: SUBROUTINE

OVERVIEW:

THIS SUBROUTINE IS USED TO ACCESS THE PARSING TABLE BASED ON
THE CURRENT INPUT TOKEN VALUE AND THE CURRENT STATE OF THE PARSER
THE CURRENT STATE OF THE PARSER IS UPDATED AND THEN AN ALTERNATE
RETURN IS PROCESSED BASED ON THE CURRENT STATE OF THE PARSER.

INVOCATION:

[CALL] LOOKUP ( P1, P2, AR )

P1 ::= CURRENT STATE OF THE PARSER
P2 ::= CURRENT INPUT TOKEN VALUE
AR ::= ALTERNATE RETURNS TO NEXT STATE

VARIABLE DICTIONARY:

CURST ; CURRENT STATE OF THE PARSER
INPUTV ; P2
NEXTST ; P1
PT ; PARSE TABLE

CALLER MODULES:

[SUBROUTINE] PRSPACK\PARSE()

CALLED MODULES:

...NONE...

PROGRAMMER: BRUCE B. LACEY
DATE : 14-OCT-85
REVISIONS : 11-NOV-86 NEW STATES FOR 'FFT','SIGNAT','STATONLY'

SUBROUTINE LOOKUP(NEXTST, INPUTV, *
+   *, *, *, *, *, *
+   *, *, *, *, *, *
+   *, *, *, *, *, *
+   *, *, *, *, *, *
+   *, *, *, *, *, *
+   *, *, *, *, *, *

COMMON /PRSCOM/ PT

INTEGER PT(59,33)
INTEGER NEXTST, INPUTV, CURST

CURST = NEXTST
NEXTST = PT(CURST,INPUTV)
RETURN PT(CURST, INPUTV)
END
BLOCK DATA BGETRC

INTEGER TIMEWM, MXXDATE, MXXSSN, MXPLNS, MXREPS
INTEGER MXSITE, MXTIME, TNKVAL, TNKLEN, LISTFL

PARAMETER(MXXDATE=10, MXXSSN=10, MXPLNS=10,
          MXREPS=5, MXSITE=20, MXTIME=10,
          MXCONT=20)

C NAMED COMMON: CHRTABS- DATA COMMUNICATION TO SCHPACK\GETREC
COMMON /CHRTABS/ ARCRT, MSSNS, SITES, TAILNM

C NAMED COMMON: INTTABS- DATA COMMUNICATION TO SCHPACK\GETREC
COMMON /INTTABS/ ENDATE, ENTIME, STDATE, STTIME, REPMUM

INTEGER REPMUM
CHARACTER*6 ARCRT (MXREPS, MXPLNS)
INTEGER ENDATE (MXREPS, MXXDATE)
INTEGER ENTIME (MXREPS, MXTIME)
CHARACTER*16 MSSNS (MXREPS, MXXSSN)
CHARACTER*10 SITES (MXREPS, MXSITE)
INTEGER STDATE (MXREPS, MXXDATE)
INTEGER STTIME (MXREPS, MXTIME)
CHARACTER*8 TAILNM (MXREPS, MXPLNS)

DATA ARCRT /50** '/
DATA ENDATE /50*0 /
DATA ENTIME /50*359/
DATA MSSNS /50** '/
DATA SITES /100** '/
DATA STDATE /50*0 /
DATA STTIME /50*0 /
DATA TAILNM /50** '/

END
MODULE NAME: PRSPACK\PARSE

MODULE TYPE: SUBROUTINE

OVERVIEW:

THIS SUBROUTINE IS USED TO PARSE THE INPUT FILE 'INFILE'
BY MEANS OF A TABLE DRIVEN PARSER. UPON REACHING THE CURRENT
STATE OF THE PARSER VARIOUS SEMANTIC ACTIONS ARE PERFORMED TO
STORE THE DATA IN 'INFILE' IN INTERNAL DATA STRUCTURES FOR LATER
USE.

INVOCATION:

[CALL] PARSE ( P1, P2 )

P1 := (CHARACTER*()) STRING CONTAINING THE RUN TITLE
P2 := (LOGICAL) FLAG SIGNALING A FATAL PARSE ERROR

VARIABLE DICTIONARY:

ACNUM ; NUMBER OF AIRCRAFT LISTED
ARCFIT ; TABLE CONTAINING AIRCRAFT TYPES
BOOMFL ; .TRUE. IF BOOMVAL IS SPECIFIED
BOOMVA ; CONTAINS THE BOOM VALUE SPECIFIED IN INFIE
CONTFL ; .TRUE. IF CONVAL IS SPECIFIED
CONTRL ; SWITCH TO CONTROL THE LEXICAL ANALYZER
CURST ; CURRENT STATE OF THE TABLE DRIVEN PARSER
DTENUM ; NUMBER OF DATES LISTED
ENDATE ; TABLE CONTAINING THE END DATES
ENTIME ; TABLE CONTAINING THE END TIMES
ERRORF ; P2
FFT ; BOOLEAN FLAG FOR FFT VALUES.
INTEST ; VARIABLE USED TO TEST INTEGER VALUES
LOOP ; SYMBOL REPRESENTING STATEMENT LABEL #1
LISTFL ; OUTPUT FILE FOR ERRORS AND SOURCE CODE ECHO
MACHFL ; .TRUE. IF MACHVAL IS SPECIFIED
MACHVA ; CONTAINS 2-10 MACH VALUES
MSSNS ; TABLE CONTAINING MISSION/EXERCISE NAMES
MSSNUM ; NUMBER OF MISSIONS LISTED
MXDATE ; MAXIMUM NUMBER OF DATES ALLOWED
MXMSSN ; MAXIMUM NUMBER OF MISSIONS ALLOWED
MXPLNS ; MAXIMUM NUMBER OF PLANES ALLOWED
MXREPS ; MAXIMUM NUMBER OF REPETITIONS OF INPUT UNITS
MXSITE ; MAXIMUM NUMBER OF SITES ALLOWED
MXTIME ; MAXIMUM NUMBER OF TIMES ALLOWED
RAYTRA ; BOOLEAN FLAG FOR STATONLY VALUES
SIGNAT ; BOOLEAN FLAG FOR SIGNATURE VALUES
SITES ; TABLE CONTAINING THE SITE LOCATIONS
STATFL ; .TRUE. IF STATS ARE TO BE PRINTED
SITENM ; NUMBER OF SITE LOCATIONS LISTED
STDAT ; TABLE CONTAINING THE START DATES
STTIME ; TABLE CONTAINING THE START TIMES
TAILNM ; TABLE CONTAINING THE AIRCRAFT TAIL NUMBERS
TIMENM ; NUMBER OF START AND END TIMES LISTED
.

CALLER MODULES:
.

MAIN DRIVER ROUTINE
.

CALLED MODULES:
.

[SUBROUTINE] LEXPACK\GETOKN()
[SUBROUTINE FUNCTION] PRSPACK\LDATE()
[INTRINSIC FUNCTION ] LEN()
[SUBROUTINE] LEXPACK\LOOKUP()
[SUBROUTINE FUNCTION] PRSPACK\LTIME()
.

PROGRAMMER: BRUCE B. LACEY
DATE : 16-OCT-85
REVISIONS : 31-OCT-85 IMPLEMENTED HANDLING OF TITLE CARD
11-NOV-86 IMPLEMENTED NEW STATES 'FFT','SIGNAT',
'STONALY'.

SUBROUTINE PARSE ( TITLE , ERRORF )

C EXTERNAL LDATE, LTIME

INTEGER TIMENM, MXDATE , MXMSSN , MXPLNS , MXREPS
INTEGER MXSITE , MXTIME , TXNVAL , TXNLEN , LISTFL

PARAMETER(MXDATE=10, MXMSSN=10, MXPLNS=10,
+ MXREPS=5 , MXSITE=20, MXTIME=10,
+ MXCONT=20 )

C NAMED COMMON: STATS- DATA COMMUNICATION TO MAIN DRIVER ONLY!
COMMON /STATS/ STATFL, BOOMFL, MACNFL, CONTFL, BOOMVA,
+ MACHVA, CONTVA, CONTYP, WIDTH, FFT, SIGNAT,
+ RAYTRC, SCRPA D, SCRPSF, SCRALL

C NAMED COMMON: CHRTABS- DATA COMMUNICATION TO SCHPACK\GETREC
COMMON /CHRTABS/ ARCRFT, MSSNS, SITES, TAILNM

C NAMED COMMON: INTTABS- DATA COMMUNICATION TO SCHPACK\GETREC
COMMON /INTTABS/ ENDATE, ENTIME, STDATE, STTIME, REPNUM

INTEGER ACNUM , DTNUM, MSSNUM, REPNUM, SITEMN
INTEGER CURST , TBLNUM , COL , PRNTROW, CONTREP
INTEGER CONTYP(MXREPS)
REAL BOOMVA, MACHVA, WIDTH , CONTVA(MXREPS,MXCONT)
CHARACTER*20 TKNSTR
CHARACTER*91 LSTBUFF
CHARACTER*(*) TITLE
LOGICAL LDATE , LTIME , ERRORF, CONTRL
LOGICAL STATFL, BOOMFL, MACHFL, CONTFL, FFT, SIGNAT,
+ RAYTRC, TCSEL, SCRPA D, SCRPSF, SCRALL
CHARACTER*6 ARCRFT (MXREPS, MXPLNS)
INTEGER ENDATE (MXREPS, MXDATE)
INTEGER ENTIME (MXREPS, MXTIME)
CHARACTER*16 MSSNS (MXREPS, MXMSSN)
CHARACTER*10 SITES (MXREPS, MXSITE)
INTEGER STDATE (MXREPS, MXDATE)
INTEGER STTIME (MXREPS, MXTIME)
CHARACTER*8 TAILNM (MXREPS, MXPLNS)

DATA LISTFL "/6/

ASSIGN 1 TO LOOP
ASSIGN 1550 TO PRNTROW

TCSEL = .FALSE.
ERRORF = .FALSE.
CONTRL = .FALSE.
FFT = .FALSE.
SIGNAT = .FALSE.
RAYTRC = .FALSE.
SCR PAD = .FALSE.
SCRPSF = .FALSE.
SCR ALL = .FALSE.
CURST = 1
REPNUM = 0
TITLE = ''
CONTREP = 0
WIDTH = 30.0
DO 7 1 = 1, MXREPS
   DO 6 J = 1, MXCONT
      CONTVA(I,J) = 0.0
6 CONTINUE
7 CONTINUE

C LOOP: PARSE AND STORE UNTIL THE ACCEPT STATE HAS BEEN REACHED.
1 IF (ERRORF.EQV..TRUE.) RETURN
CALL GETOKN(TKNSTR, TKNLEN, TKNVAL, CONTRL)
CALL LOOKUP(CURST, TKNVAL, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53)

C STATE 0 : ERROR STATE.
WRITE(LISTFL,FMT=3) TKNSTR(1:TKNLEN)
3 FORMAT(5X,"***-FATAL SYNTAX ERROR ON INPUT ",A, 
      
      *** (PARSE TERMINATED)-***")
   ERRORF = .TRUE.
   GO TO LOOP

C STATE 1 : START STATE.
10    GO TO LOOP

C STATE 2: RECOGNIZED 'SITE'
20    REPNUM = REPNUM + 1
21    IF (REPNUM.GT.MXREPS) THEN
22       WRITE(LISTFL,21)
23       FORMAT(' ***WARNING: TOO MANY SITE CARD REPETITIONS***')
24       ERRORF = .TRUE.
25    ELSE
26       ACNUM = 0
27       DTENUM = 0
28       MSSNUM = 0
29       SITENM = 0
30       TIMENM = 0
31    END IF
32    GO TO LOOP

C STATE 3: FOUND IDENTIFIER AFTER 'SITE'
30    IF (TKNLEN.GT.LEN(SITES(1,1))) THEN
31       WRITE(LISTFL,FMT=31)
32       FORMAT(' **-ERROR: SITE NAME TOO LONG-**')
33       ERRORF = .TRUE.
34    ELSE IF (SITENM.GE.MXSITE) THEN
35       WRITE(LISTFL,FMT=32)
36       FORMAT(' **-WARNING: TOO MANY SITES LISTED-**')
37       ERRORF = .TRUE.
38    ELSE
39       STORE THE SITE NAME IN THE SITE LOCATION TABLE
40       SITENM = SITENM + 1
41       SITES(REPNUM,SITENM) = TKNSTR(1:LEN(SITES(1,1)))
42    END IF
43    GO TO LOOP

C STATE 4: IDENTIFIER LIST FOR SITE LOCATIONS BEING PROCESSED
40    GO TO LOOP

C STATE 5: 'ALL' SITE LOCATIONS ARE TO BE INCLUDED
40    STORE 'ALL' IN SITE LOCATION TABLE
50    SITES(REPNUM, 1) = 'ALL'
51    GO TO LOOP

C STATE 6: 'DATE' RECOGNIZED
50    THEREFORE ALL MISSION NAMES ARE VALID
60    MSSNS(REPNUM,1) = 'ALL'
61    GO TO LOOP

C STATE 7: 'MISSION' RECOGNIZED
70    CONTRL = .TRUE.
71    ALL DATES ARE VALID.
72    STDATE(REPNUM,1) = 9999
73    ENDATE(REPNUM,1) = 9999
GO TO LOOP

C STATE 8: FOUND AN IDENTIFIER AFTER 'MISSION'
80 IF (TKNLEN.GT.LEN(MSSNS(1,1))) THEN
   WRITE(LISTFL,FMT=81)
81 FORMAT(' **ERROR: MISSION NAME TOO LONG-**')
   ERRORF = .TRUE.
ELSE IF (MSSNUM.GE.MXMESSN) THEN
   WRITE(LISTFL,FMT=82)
82 FORMAT(' **WARNING: TOO MANY MISSIONS LISTED-**')
   ERRORF = .TRUE.
ELSE
   STORE THE MISSION NAME IN THE TABLE
   MSSNUM = MSSNUM + 1
   MSSNS(REPNUM, MSSNUM) = TKNSTR(1:LEN(MSSNS(1,1)))
END IF
GO TO LOOP

C STATE 9: IDENTIFIER LIST FOR MISSION BEING PROCESSED.
90 CONTNL = .TRUE.
GO TO LOOP

C STATE 10: 'ALL' DATES ARE TO BE INCLUDED.
100 STDATE(REPNUM, 1) = 9999
ENDATE(REPNUM, 1) = 9999
GO TO LOOP

C STATE 11: MONTH OF START DATE RECOGNIZED.
110 IF (.NOT.LDATE('MM',TKNSTR)) THEN
   WRITE(LISTFL,FMT=111)
111 FORMAT(' **ERROR: ILLEGAL MONTH IN START DATE-**')
   ERRORF = .TRUE.
ELSE IF (DTENUM.GE.MXDATE) THEN
   WRITE(LISTFL,FMT=112)
112 FORMAT(' **WARNING: TOO MANY DATES LISTED-**')
   ERRORF = .TRUE.
ELSE
   DTENUM = DTENUM + 1
   READ(TKNSTR(1:2),FMT='(12)') INTEST
   STDATE(REPNUM, DTENUM) = STDATE(REPNUM, DTENUM) +
   (INTEST * 100)
END IF
GO TO LOOP

C STATE 12: MM '//' OF START DATE RECOGNIZED.
120 GO TO LOOP

C STATE 13: DAY OF START DATE RECOGNIZED
130 IF (.NOT.LDATE('DD',TKNSTR)) THEN
   WRITE(LISTFL,FMT=131)
131 FORMAT(' **ERROR: ILLEGAL DAY IN START DATE-**')
   ERRORF = .TRUE.
ELSE
   READ(TKNSTR(1:2),FMT='(12)') INTEST
   +
STDATE(REPNUM,DTENUM) = STDATE(REPNUM,DTENUM) + INTEST
END IF
GO TO LOOP

C STATE 14: MM/DD '/' OF START DATE RECOGNIZED.
140 GO TO LOOP

C STATE 15: YEAR OF START DATE RECOGNIZED.
150 IF (.NOT.LDATE('YY',TKNSTR)) THEN
WRITE(LISTFL,FMT='151)
151 FORMAT(' **ERROR: ILLEGAL YEAR IN START DATE-**')
ERRORF = .TRUE.
ELSE
READ(TKNSTR(1:2),FMT='(12)1') INTEST
STDATE(REPNUM,DTENUM) = STDATE(REPNUM,DTENUM) +
(INTEST * 10000)
C ASSUME THAT NO END DATE IS SPECIFIED BY USER
ENDATE(REPNUM,DTENUM) = STDATE(REPNUM,DTENUM)
END IF
GO TO LOOP

C STATE 16: '·' BETWEEN START AND END DATE RECOGNIZED.
160 GO TO LOOP

C STATE 17: MONTH OF END DATE RECOGNIZED.
170 IF (.NOT.LDATE('MM',TKNSTR)) THEN
WRITE(LISTFL,FMT='171)
171 FORMAT(' **ERROR: ILLEGAL MONTH IN END DATE-**')
ERRORF = .TRUE.
ELSE
READ(TKNSTR(1:2),FMT='(12)1') INTEST
ENDATE(REPNUM,DTENUM) = 0
ENDATE(REPNUM,DTENUM) = ENDATE(REPNUM,DTENUM) +
(INTEST * 100)
END IF
GO TO LOOP

C STATE 18: MM '/' OF END DATE RECOGNIZED.
180 GO TO LOOP

C STATE 19: DAY OF END DATE RECOGNIZED.
190 IF (.NOT.LDATE('DD',TKNSTR)) THEN
WRITE(LISTFL,FMT='191)
191 FORMAT(' **ERROR: ILLEGAL DAY IN END DATE-**')
ERRORF = .TRUE.
ELSE
READ(TKNSTR(1:2),FMT='(12)1') INTEST
ENDATE(REPNUM,DTENUM) = ENDATE(REPNUM,DTENUM) + INTEST
END IF
GO TO LOOP

C STATE 20: MM/DD '/' OF END DATE RECOGNIZED.
200 GO TO LOOP
C STATE 21: YEAR OF END DATE RECOGNIZED.
210 IF (.NOT.LDATE('YY',TKNSTR)) THEN
   WRITE(LISTFL,FMT=211)
211 FORMAT(' ***-ERROR: ILLEGAL YEAR IN END DATE-***')
   ERRORD = .TRUE.
ELSE
   READ(TKNSTR(1:2),FMT='(I2)') INTEST
   ENDATE(REPNUM,DTENUM) = ENDATE(REPNUM,DTENUM) +
   (INTEST * 10000)
   END IF
GO TO LOOP

C STATE 22: DATE LIST FOR 'DATE' BEING PROCESSED.
220 GO TO LOOP

C STATE 23: 'ALL' MISSIONS ARE TO BE INCLUDED.
230 MSSNS(REPNUM, 1) = 'ALL'
GO TO LOOP

C STATE 24: 'TIME' RECOGNIZED.
240 GO TO LOOP

C STATE 25: START TIME RECOGNIZED.
250 READ(TKNSTR(1:8),FMT='(I8)') INTEST
   IF (.NOT.LTIME(INTEST)) THEN
      WRITE(LISTFL,FMT=251)
251 FORMAT(' ***-ERROR: ILLEGAL START TIME-***')
   ELSE IF (TIMENM.GE.MXTIME) THEN
      WRITE(LISTFL,FMT=252)
252 FORMAT(' ***-WARNING: TOO MANY TIMES SPECIFIED-***')
      ERRORD = .TRUE.
   ELSE
      TIMENM = TIMENM + 1
      STTIME(REPNUM,TIMENM) = INTEST
   END IF
GO TO LOOP

C STATE 26: '···' BETWEEN START TIME AND END TIME RECOGNIZED.
260 GO TO LOOP

C STATE 27: END TIME RECOGNIZED.
270 READ(TKNSTR(1:8),FMT='(I8)') INTEST
   IF (.NOT.LTIME(INTEST)) THEN
      WRITE(LISTFL,FMT=271)
271 FORMAT(' ***-ERROR: ILLEGAL END TIME-***')
      ERRORD = .TRUE.
   ELSE
      ENTIME(REPNUM,TIMENM) = INTEST
   END IF
GO TO LOOP

C STATE 28: TIME LIST FOR 'TIME' BEING PROCESSED.
280 GO TO LOOP
C STATE 29: 'ALL' TIMES ARE TO BE INCLUDED
290      STIME(REPNUM,1) = 9999
          EN TIME(REPNUM,1) = 9999
          GO TO LOOP

C STATE 30: 'ACWTN' RECOGNIZED.
300      GO TO LOOP

C STATE 31: 'AIRCRAFT' RECOGNIZED.
C THEREFORE ALL TAIL NUMBERS ARE VALID
310      TAILNM(REPNUM,1) = 'ALL'
          GO TO LOOP

C STATE 32: AIRCRAFT TYPE RECOGNIZED.
320      IF (TKNLEN.GT.LEN(ARCRFT(1,1))) THEN
              WRITE(LISTFL,FMT=321)
321          FORMAT( ' **-ERROR: AIRCRAFT TYPE TOO MANY CHARACTERS-**')
              ERRORF = .TRUE.
          ELSE IF (ACNUM.GE.MXPLNS) THEN
              WRITE(LISTFL,FMT=322)
322          FORMAT( ' **-WARNING: TOO MANY AIRCRAFT TYPES LISTED-**')
              ERRORF = .TRUE.
          ELSE
              ACNUM = ACNUM + 1
              ARCRFT(REPNUM,ACNUM) = TKNSTR(1:LEN(ARCRFT(1,1)))
          END IF
          GO TO LOOP

C STATE 33: IDENTIFIER LIST FOR AIRCRAFT TYPES BEING PROCESSED
330      GO TO LOOP

C STATE 34: AIRCRAFT TYPE RECOGNIZED.
340      IF (TKNLEN.GT.LEN(ARCRFT(1,1))) THEN
              WRITE(LISTFL,FMT=321)
          ELSE IF (ACNUM.GE.MXPLNS) THEN
              WRITE(LISTFL,FMT=322)
          ELSE
              ACNUM = ACNUM + 1
              ARCRFT(REPNUM,ACNUM) = TKNSTR(1:LEN(ARCRFT(1,1)))
          END IF
          CONTRL = .TRUE.
          GO TO LOOP

C STATE 35: TAIL NUMBER 'ACWTN' RECOGNIZED.
350      IF (TKNLEN.GT.LEN(TAILNM(1,1))) THEN
              WRITE(LISTFL,FMT=351)
351          FORMAT( ' **-ERROR: ILLEGAL TAIL NUMBER SPECIFIED-**')
              ERRORF = .TRUE.
          ELSE
              TAILNM(REPNUM,ACNUM) = TKNSTR(1:TKNLEN)
          END IF
          GO TO LOOP

C STATE 36: IDENTIFIER LIST FOR 'ACWTN' BEING PROCESSED.
360  GO TO LOOP

C STATE 37: 'ALL' 'ACWTN' ARE TO BE INCLUDED.
370  ARCRFT(REPNUM, 1) = 'ALL'
     TAILNM(REPNUM, 1) = 'ALL'
     GO TO LOOP

C STATE 38: 'ALL' AIRCRAFT ARE TO BE INCLUDED.
380  ARCRFT(REPNUM, 1) = 'ALL'
     GO TO LOOP

C STATE 39: 'STAT' RECOGNIZED.
390  STATFL = TRUE.
     GO TO LOOP

C STATE 40: 'BOOMTRK' RECOGNIZED.
400  BOOMFL = TRUE.
     GO TO LOOP

C STATE 41: INTEGER OR REAL FOUND AFTER 'BOOMTRK'
C CONVERT THE TOKEN STRING TO A REAL NUMBER AND STORE
410  READ(TKNSTR(1:12), FMT='(F10.0)') BOOMVA
     GO TO LOOP

C STATE 42: 'MACHTRK' RECOGNIZED.
420  MACHFL = TRUE.
     GO TO LOOP

C STATE 43: INTEGER OR REAL FOUND AFTER 'MACHTRK'
C CONVERT THE TOKEN STRING TO A REAL NUMBER AND STORE
430  READ(TKNSTR(1:12), FMT='(F10.0)') MACHVA
     GO TO LOOP

C STATE 44: 'CONTOUR' RECOGNIZED.
440  CONTFL = TRUE.
     INTEST = 0
     CONTREP = CONTREP + 1
     IF (CONTREP.GT.MXREPS) THEN
         WRITE(LISTFL,3572)
     3572   FORMAT('*-WARNING: TO MANY CONTOUR SPECS.*')
         ERRORF = TRUE.
         END IF
     GO TO LOOP

C STATE 45: INTEGER OR REAL FOUND AFTER 'CONTOUR XXXX'
C CONVERT THE TOKEN STRING TO A REAL NUMBER AND STORE
450  INTEST = INTEST + 1
     IF (INTEST.GT.MXCONT) THEN
         WRITE(LISTFL,*') '++-WARNING: TOO MANY CONTOUR VALUES---'
         ERRORF = TRUE.
         ELSE
         READ(TKNSTR(1:12), FMT='(F10.0)') CONTVA(CONTREP, INTEST)
         END IF
     GO TO LOOP
C STATE 46: SCALE LIST FOR 'CONTOUR' BEING PROCESSED
460 GO TO LOOP

C STATE 47: 'TITLE' RECOGNIZED
470 CONTRL = .TRUE.
   COL = 1
   GO TO LOOP

C STATE 48: CONCATENATE ALL TITLE PIECES TOGETHER.
480 IF ((COL+TKNLEN).LE.(LEN(TITLE))) THEN
   TITLE(COL:(COL+TKNLEN-1))=TKNSTR(1:TKNLEN)
   END IF
   COL = COL + TKNLEN + 1
   CONTRL = .TRUE.
   GO TO LOOP

C STATE 49: CONTOUR TYPE RECOGNIZED AFTER 'CONTOUR'
490 IF (TKNSTR(1:TKNLEN).EQ. 'CSEL') THEN
   CONTPY(CONTREP) = 1
   TCSL = .TRUE.
   ELSE IF (TKNSTR(1:TKNLEN).EQ. 'CLDM') THEN
   CONTPY(CONTREP) = 2
   ELSE IF (TKNSTR(1:TKNLEN).EQ. 'PKUP') THEN
   CONTPY(CONTREP) = 3
   END IF
   GO TO LOOP

C STATE 50: 'WIDTH' CARD RECOGNIZED
500 GO TO LOOP

C STATE 51: INTEGER OR REAL RECOGNIZED AFTER 'WIDTH'
510 READ(TKNSTR(1:8),FMT='(F6.2)') WIDTH
   IF ((WIDTH.LE.8).OR.(WIDTH.GE.48)) THEN
   WRITE(LISTFL,FMT='(A)') '++-WARNING: ILLEGAL'
   WRITE(LISTFL,FMT='(A)') '- -WARNING: ILLEGAL'
   WIDTH = 30.0
   END IF
   GO TO LOOP

C STATE 52: ',' RECOGNIZED AFTER 'CONTOUR XXXX'
520 GO TO LOOP

C STATE 54: 'FFT' RECOGNIZED.
525 FFT = .TRUE.
   GO TO LOOP

C STATE 55: 'SIGNAT' RECOGNIZED.
527 SIGNAT = .TRUE.
   GO TO LOOP

C STATE 56: 'STATONLY' RECOGNIZED.
528 RAYTRC = .TRUE.
STATFL = .TRUE.
GOTO LOOP

C STATE 57: 'SCRCHPAD' RECOGNIZED.
535  SCRPAD = .TRUE.
GOTO LOOP

C STATE 58: 'DB OR PSF' RECOGNIZED.
537  IF (TKNSTR(1:TKNLEN) .EQ. 'DB') THEN
     SCRPSF = .FALSE.
ELSE
     SCRPSF = .TRUE.
ENDIF
GOTO LOOP

C STATE 59: 'ALL OR NEW' RECOGNIZED.
538  IF (TKNSTR(1:TKNLEN) .EQ. 'ALL') THEN
     SCRALL = .TRUE.
ENDIF
GOTO LOOP

C STATE 53: ACCEPT STATE
C PRINT OUT A TABLE OF THE INFORMATION STORED DURING THE PARSE
530  DO 600 TBLNUM = 1, REPNRM
     WRITE(LISTFL,550) TBLNUM, REPNRM
550  FORMAT(1X,,3X,'TABLE:',2X,12,'/',12,/)     WRITE(LISTFL,560)
560  FORMAT(2X,91('=I'))
     WRITE(LISTFL,570)
570  FORMAT(2X,'!',4X,'SITE',4X,'!',5X,'EXERCISE',5X,'!',6X,
     + 'DATE',7X,'!',4X,'TIME',5X,'AIRCRAFT',3X,'TAIL',3X,
     + 'LOCATION',2X,'!',7X,'NAME',7X,
     + '! [YYYY-MM-DD] ![HHH-MMM] !!',2X,'TYPE',
     + 'NUMBER',2X,'!',5X,'!/',2X,'!',89('!'),'!!')
     COL = 1
C PRNTROW:
1550  LSTBUFF = ' '
     LSTBUFF(1:1) = '!'
     LSTBUFF(3:12) = SITES(TBLNUM,COL)
     LSTBUFF(14:14) = '!
     LSTBUFF(16:31) = MSSNS(TBLNUM,COL)
     LSTBUFF(33:33) = '!
     IF (STDATE(TBLNUM,1).EQ.9999) THEN
          IF (COL.EQ.1) LSTBUFF(35:37) = 'ALL'
     ELSE IF (STDATE(TBLNUM,COL).EQ.0) THEN
          LSTBUFF(35:49) = '!
     ELSE
          LSTBUFF(35:35) = '!
          WRITE(LSTBUFF(36:41),'(16)')STDATE(TBLNUM,COL)
          LSTBUFF(42:42) = '!
          WRITE(LSTBUFF(43:48),'(16)')ENDATE(TBLNUM,COL)
     LSTBUFF(49:49) = '!
     END IF
     LSTBUFF(51:51) = '!'
IF (STTIME(TBLNUM, 1) .EQ. 9999) THEN
  IF (COL .EQ. 1) LSTBUFF(53:55) = 'ALL'
ELSE IF (STTIME(TBLNUM, COL) .EQ. 0) THEN
  LSTBUFF(53:63) = 'I'
ELSE
  LSTBUFF(53:53) = 'C
WRITE(LSTBUFF(54:57), '(14)') STTIME(TBLNUM, COL)
LSTBUFF(58:58) = '..
WRITE(LSTBUFF(59:62), '(14)') ENTIME(TBLNUM, COL)
LSTBUFF(63:63) = 'I'
END IF
LSTBUFF(65:65) = 'I'
LSTBUFF(67:72) = ARCRFT(TBLNUM, COL)
LSTBUFF(74:74) = 'I'
LSTBUFF(76:83) = TAILNM(TBLNUM, COL)
LSTBUFF(85:85) = 'I'
WRITE(LSTBUFF(87:90), '(13)') COL
LSTBUFF(91:91) = 'I'
WRITE(LISTFL, FMT = '2X,A91') LSTBUFF

C - CHECK IF ANOTHER ROW SHOULD BE PRINTED.
    COL = COL + 1
    IF (SITES(TBLNUM, COL) .NE. ' ' .OR. MSSWS(TBLNUM, COL) .NE. ' ' + .OR. STDATE(TBLNUM, COL) .NE. ' ' .OR. STTIME(TBLNUM, COL) + .NE. ' ' .OR. ARCRFT(TBLNUM, COL) .NE. ' ' ) GOTO PRNTROW
    WRITE(LISTFL,580)
580  FORMAT(2X,'*')

600  CONTINUE
    IF ((FFT) .AND. (TCSEL)) THEN
      DO 650 I = 1,5
        IF ((CONTYP(I) .EQ. 2) .OR. (CONTYP(I) .EQ. 3)) THEN
          WRITE (LISTFL,3575)
          FORMAT( '---WARNING : OVERPRESSURE AND CSEL NO LONGER
+ RELATED CONTOUR ABORTED')
          CONTYP(I) = 0
          DO 620 J = 1, 20
            CONTVA(I,J) = 0.0
          620        CONTINUE
        END IF
      650      CONTINUE
      IT = 1
      DO 670 I = 1,5
        IF ((CONTYP(I) .EQ. 1) .AND. (IT .NE. 1)) THEN
          CONTYP(IT) = 1
          CONTYP(I) = 0
          DO 660 J = 1, 20
            CONTVA(IT,J) = CONTVA(I,J)
            CONTVA(I,J) = 0
          660        CONTINUE
          IT = IT + 1
        END IF
      670    CONTINUE
END IF
RETURN

*---------------------------------------------------------------------*
*---------------------------------------------------------------------*
*---------------------------------------------------------------------*
*---------------------------------------------------------------------*
**PRSPACK**
*---------------------------------------------------------------------*
**---------------------------------------------------------------------*
END
MODULE NAME: SCHPACK

MODULE TYPE: PACKAGE

OVERVIEW:

THIS PACKAGE IS USED TO PERFORM THE PROCESS OF SEARCHING
THE DATA TABLES CREATED DURING THE PARSE STAGE. THIS SEARCH
IS USED TO FIND RECORDS OF SUBSONIC AND SUPersonic FLIGHT DATA
RECORDS IN THE LIBRARY FILE BY FINDING THEIR LOCATION THROUGH
THE USE OF AN INDEX FILE. THIS INDEX FILE IS SIMILAR TO A CARD
CATALOG.

INTERFACE:

GETREC ( P1, P2, P3 )

P1 ::= (INTEGER) POINTER TO THE STARTING RECORD
P2 ::= (INTEGER) TOTAL OF RECORDS STARTING AT P1
P3 ::= (LOGICAL) FLAG SIGNALING NO MORE RECORDS LEFT

INTERNAL SUBROUTINES & FUNCTIONS

FILBUF() ; READS ON RECORD FROM THE INDEX FILE INTO A BUFFER
STRMCH() ; RETURNS TRUE IF A STRING MATCHES WITH TABLE STR
INTMCH() ; RETURNS TRUE IF AN INT. MATCHES WITH TABLE INTE

PROGRAMMER: BRUCE B. LACEY
DATE : 23-OCT-85

REVISIONS :

MODULE NAME: SCHPACK\FILBUF

MODULE TYPE: CHARACTER FUNCTION SUBROUTINE

OVERVIEW:

THIS FUNCTION SUBROUTINE IS USED TO READ IN ON RECORD FROM T
INDEX FILE. IF THERE ARE NO MORE RECORDS THEN THE FLAG .ENDREC.
SET TRUE.

INVOCATION:

[X = ] FILBUF ( P1, P2, P3, P4 )

P1 ::= (INTEGER) CURRENT RECORD NUMBER
P2 ::= (INTEGER) NUMBER OF RECORDS IN INDEX FILE
P3 ::= (INTEGER) UNIT NUMBER CORRESPONDING TO INDEX FIL

P4 ::= [LOGICAL] FLAG SIGNALING THE END OF RECORDS

VARIABLE DICTIONARY:

ENDREC ; P4
IDXFIL ; P3
NUMREC ; P2
RECNUM ; P1

CALLER MODULES:

[SUBROUTINE] SCHPACK\GETREC

CALLED MODULES:

...NONE...

PROGRAMMER: BRUCE B. LACEY
DATE : 22-OCT-85
REVISIONS :

CHARACTER*(*) FUNCTION FILBUF(
 + RECNUM, NUMREC, IDXFIL, ENDREC)

INTEGER RECNUM, NUMREC, IDXFIL
LOGICAL ENDREC

IF (RECNUM.LE.NUMREC) THEN
  RECNUM = RECNUM + 1
  READ(IDXFIL,FMT='(A)',REC=RECNUM) FILBUF
ELSE
  ENDREC = .TRUE.
END IF

RETURN
END
MODULE NAME: SCHPACK\STRMCH

MODULE TYPE: LOGICAL FUNCTION SUBROUTINE

OVERVIEW:

THIS FUNCTION SUBROUTINE IS USED TO SEE IF A STRING PASSED IN MATCHES ANY STRING IN THE CURRENT ROW OF A TABLE PASSED IN. IF 'ALL' IS FOUND THEN THE SEARCH IS CONSIDERED SUCCESSFUL.

INVOCATION:

(K = ) STRMCH ( P1, P2, P3, P4, P5 )

P1 ::= [CHARACTER*(*)] STRING TO SEARCH FOR
P2 ::= [INTEGER] REPETITION BEING TESTED
P3 ::= [CHARACTER*(*)] (P4,P5) TABLE TO SEARCH THROUGH
P4 ::= [INTEGER] BORDER FOR THE ROW SIZE
P5 ::= [INTEGER] BORDER FOR THE COLUMN SIZE

VARIABLE DICTIONARY:

COL ; CURRENT COLUMN IN THE SEARCH TABLE
CURREP ; P2
EXTLOP ; SYMBOL REPRESENTING STATEMENT LABEL 200
MXCOL ; P5
MXROW ; P4
SRCFOR ; P1
TABLE ; P3

CALLER MODULES:

(SUBROUTINE) SCHPACK\GETREC

CALLED MODULES:

...NONE...

PROGRAMMER: BRUCE B. LACEY

DATE : 22-OCT-85

REVISIONS :

LOGICAL FUNCTION STRMCH( +
    SRCFOR, CURREP, TABLE, MXROW, MXCOL)

INTEGER CURREP, MXROW, MXCOL, COL, EXTLOP
CHARACTER*(*) SRCFOR, TABLE(MXROW,MXCOL)

ASSIGN 200 TO EXTLOP

STRMCH = .FALSE.
DO 100 COL = 1, MXCOL
    IF ((TABLE(CURREP, COL).EQ.'ALL').OR.
        (TABLE(CURREP, COL).EQ.SRCFOR)) THEN
        IF (TABLE(OJREPP, COL).NE.' ') THEN
            STRMCH = .TRUE.
            GO TO EXTLOP
        ENDIF
    ENDIF
100    CONTINUE
C EXTLOP:
200    RETURN
END
MODULE NAME: SCHPACK\INTMCH
MODULE TYPE: LOGICAL FUNCTION SUBROUTINE

OVERVIEW:

THIS FUNCTION SUBROUTINE IS USED TO TEST IF AN INTEGER PASSED IN MATCHES AN INTEGER IN THE CURRENT ROW OF THE TABLE PASSED IN. IF 9999 IS FOUND THEN THE TEST IS CONSIDERED SUCCESSFUL.

INVOCATION:

[X = INTMCH ( P1, P2, P3, P4, P5, P6 )]

P1 ::= [INTEGER] VALUE TO BE TESTED
P2 ::= [INTEGER] ROW CURRENTLY BEING TESTED
P3 ::= [INTEGER(P5,P6)] TABLE FOR LOWER BOUND
P4 ::= [INTEGER(P5,P6)] TABLE FOR UPPER BOUND
P5 ::= [INTEGER] LOWER BOUND FOR P3 AND P4
P6 ::= [INTEGER] UPPER BOUND FOR P3 AND P4

VARIABLE DICTIONARY:

COL ; CURRENT COLUMN IN SEARCH TABLES
CURREP ; P2
ETABLE ; P3
EXTLOP ; SYMBOL REPRESENTING STATEMENT LABEL 200
MXCOL ; P6
MXROW ; P5
SRCFOR ; P1
STABLE ; P4

CALLER MODULES:

[SUBROUTINE] SCHPACK\GETREC

CALLED MODULES:

...NONE...

PROGRAMMER: BRUCE B. LACEY
DATE : 22-OCT-85
REVISION :

LOGICAL FUNCTION INTMCH(
   SRCFOR, CURREP, STABLE, ETABLE, MXROW, MXCOL)

INTEGER SRCFOR, CURREP, MXROW, MXCOL, EXTLOP
INTEGER COL, LOOP
INTEGER STABLE(MXROW,MXCOL), ETABLE(MXROW,MXCOL)
ASSIGN 100 TO LOOP
ASSIGN 200 TO EXTLOP

INTMCH = .FALSE.
IF (STABLE(CURREP,1).EQ.9999) THEN
  INTMCH = .TRUE.
ELSE
  COL = 1

C LOOP:
100 IF((SRCFOR.GE.STABLE(CURREP,COL)).AND.
+ (SRCFOR.LE.ETAULE(CURREP,COL))) THEN
  INTMCH = .TRUE.
  GO TO EXTLOP
END IF
  COL = COL + 1
  IF (COL.LE.XCOL) GO TO LOOP

C EXTLOP:
200 END IF
  RETURN
END
MODULE NAME: SCHPACK\GETREC

MODULE TYPE: SUBROUTINE

OVERVIEW:

THIS SUBROUTINE IS USED TO SEARCH THE INDEX FILE ACCORDING TO THE USER SPECIFICATIONS STORED DURING THE PARSE STAGE. WHEN INVO

THIS SUBROUTINE WILL READ RECORDS FROM THE INDEX FILE UNTIL A MATCH IS FOUND. WHEN A MATCH IS FOUND THE SUBROUTINE WILL RETURN THE STARTING RECORD NUMBER AND THE NUMBER OF RECORDS OCCURRING AFTER THE STARTING RECORD. IF A MATCH IS NOT FOUND THEN THE END OF REC FLAG .ENDREC. IS SET TRUE.

INVOCATION:

(CALL) GETREC ( P1, P2, P3, P4 )

P1 ::= [INTEGER] STARTING RECORD NUMBER
P2 ::= [INTEGER] COUNT OF RECORDS FOLLOWING P1
P3 ::= [INTEGER] COUNT OF SUPersonic RECORDS
P4 ::= [LOGICAL] FLAG SIGNALING THE END OF RECORDS

VARIABLE DICTIONARY:

ARCRFT ; TABLE CONTAINING AIRCRAFT TYPES
CURREC ; THE CURRENT RECORD NUMBER FROM FILE 'INDEX'
ENDATE ; TABLE CONTAINING THE END DATES
ENDREC ; P3
ENTIME ; TABLE CONTAINING THE END TIMES
IDXFIL ; UNIT NUMBER FOR THE INDEX FILE
INTDAT ; INTEGER REPRESENTING YMD DATE
LOOP ; SYMBOL REPRESENTING STATEMENT LABEL 1
MSSNS ; TABLE CONTAINING THE MISSION/EXERCISE NAMES
MXDATE ; MAXIMUM NUMBER OF DATE ALLOWED
MXMSNS ; MAXIMUM NUMBER OF MISSION ALLOWED
MXPLNS ; MAXIMUM NUMBER OF AIRCRAFT ALLOWED
MXREPS ; MAXIMUM NUMBER OF REPETITIONS OF SITE CARDS ALLOWED
MXSITE ; MAXIMUM NUMBER OF SITES LOCATIONS ALLOWED
MXTIME ; MAXIMUM NUMBER OF START/END TIMES ALLOWED
NUMREC ; NUMBER OF RECORDS IN THE INDEX FILE
NUMREP ; NUMBER OF REPETITIONS STORED DURING PARSE
RECBUF ; BUFFER TO HOLD ONE RECORD FROM FILE 'INDEX'
RECTOT ; P2
SITES ; TABLE CONTAINING THE SITE LOCATIONS
STREC ; P1
STTIME ; TABLE CONTAINING THE STARTING TIMES
TAILNM ; TABLE CONTAINING THE AIRCRAFT TAIL NUMBERS
TBLIDX ; CURRENT REPETITION BEING COMPARED
TIME1 ; STARTING TIME FROM RECORD IN 'INDEX'
TIME2 ; ENDING TIME FROM RECORD IN 'INDEX'
CALLER MODULES:
- MAIN DRIVER ROUTINE
- CALLED MODULES:
  - [SUBROUTINE FUNCTION] SCHPACK\FILBUF()
  - [SUBROUTINE FUNCTION] SCHPACK\SCHMCH()
  - [SUBROUTINE FUNCTION] SCHPACK\INTMCH()

PROGRAMMER: BRUCE B. LACEY
DATE: 22-OCT-85
REVISIONS:

SUBROUTINE GETREC(STREC, RECTOT, SUPREC, ENDREC, 11)

EXTERNAL STRMCH, INTMCH, FILBUF
PARAMETER(MXDATE=10, MXMSSN=10, MXPLNS=10,
          MXREPS=5, MXSITE=20, MXTIME=10)
COMMON /CHRTABS/ ARCRFT, MSSNS, SITES, TAILNM
COMMON /INTTABS/ ENDATE, ENTIME, STDATE, STTIME, NUMREP

INTEGER ENDATE(MXREPS,MXDATE)
INTEGER ENTIME(MXREPS,MXTIME)
INTEGER STDATE(MXREPS,MXDATE)
INTEGER STTIME(MXREPS,MXTIME)
INTEGER NUMREP, STREC, RECTOT, CURREC,
          INTDAT, LOOP
INTEGER TIME1, TIME2, IDXFIL,
          NUMREC, TBLIDX, SUPREC

CHARACTER*6 ARCRFT(MXREPS,MXPLNS)
CHARACTER*16 MSSNS(MXREPS,MXMSN)
CHARACTER*10 SITES(MXREPS,MXSITE)
CHARACTER*8 TAILNM(MXREPS,MXPLNS)
CHARACTER*98 FILBUF, RECBUF

LOGICAL ENDREC, STRMCH, INTMCH

SAVE CURREC

DATA IDXFIL /1/
DATA CURREC /0/
ASSIGN 1 TO LOOP

IF (CURREC.LE.1) THEN
  READ THE HEADER RECORD TO GET THE NUMBER OF RECORDS.
  CURREC = CURREC + 1
  READ(IDXFIL,FMT='(I6)',REC=CURREC) NUMREC
  ENDREC = .FALSE.
END IF
TBLIDX = 0

C- GET THE STARTING RECORD
RECBUF = FILBUF(CURREC,NUMREC,IDXFILE,ENDREC)

C LOOP:
1 IF (ENDREC.EQ..TRUE.) RETURN
   TBLIDX = TBLIDX + 1
   IF (TBLIDX.GT.NUMREP) THEN
   C- READ IN ANOTHER RECORD FOR TESTING
      RECBUF = FILBUF(CURREC,NUMREC,IDXFILE,ENDREC)
      TBLIDX = 1
   END IF

C- CHECK IF SITE LOCATIONS MATCH
IF (STRMCH(RECBUF(27:36),TBLIDX,SITES,NUMREPS,MXSITE).
   .EQV..TRUE.) THEN
   C- CHECK IF THE MISSION NAMES MATCH
   IF (STRMCH(RECBUF(1:16),TBLIDX,MSSIONS,NUMREPS,MXMISSION).
      .EQV..TRUE.) THEN
   C- CHECK IF THE DATE INTERVALS CORRESPOND.
   C- FIRST CONVERT THE DATE TO YYMMDD INTEGER
      READ(RECBUF(17:18),FMT='(I2)') INTDAT
      INTDAT = INTDAT * 100
      READ(RECBUF(20:21),FMT='(I2)') I
      INTDAT = INTDAT + I
      READ(RECBUF(23:24),FMT='(12)') I
      INTDAT = INTDAT + (I * 10000)
   IF (INTMCH(INTDAT,TBLIDX,STDAT,ENDAT,NUMREPS,MXDATE).
      .EQV..TRUE.) THEN
   C- CHECK IF THE TIME INTERVALS CORRESPOND.
      READ(RECBUF(37:40),FMT='(I4)') TIME1
      READ(RECBUF(45:48),FMT='(14)') TIME2
      IF ((INTMCH(TIME1,TBLIDX,STTIME,ENTIME,NUMREPS,MXTIME).
         .EQV..TRUE.).OR.
         (INTMCH(TIME2,TBLIDX,STTIME,ENTIME,NUMREPS,MXTIME).
         .EQV..TRUE.)) THEN
   C- CHECK IF AIRCRAFT TYPES MATCH.
   IF (STRMCH(RECBUF(55:60),TBLIDX,ARCRFT,NUMREPS,MXPLOTS).
      .EQV..TRUE.) THEN
   C- CHECK IF THE AIRCRAFT TAIL NUMBERS MATCH
      IF (STRMCH(RECBUF(61:68),TBLIDX,TAILNM,NUMREPS,MXPLOTS).
         .EQV..TRUE.) THEN
      WE HAVE A SUCCESSFUL MATCH
      READ(RECBUF(69:78),FMT='(110)') STREC
      READ(RECBUF(79:88),FMT='(110)') RECTOT
      READ(RECBUF(99:98),FMT='(110)') SUPREC
11 = CURRENT
RETURN
ELSE
GO TO LOOP
END IF
ELSE
GO TO LOOP
END IF
ELSE
GO TO LOOP
END IF
ELSE
GO TO LOOP
END IF
ELSE
GO TO LOOP
END IF
ELSE
GO TO LOOP
END IF

===============================================

END SCNPACK

===============================================

END
MODULE NAME: SCHPACK\GETOMC

MODULE TYPE: SUBROUTINE

OVERVIEW:

THIS SUBROUTINE IS USED TO SEARCH THE INDEX FILE ACCORDING TO THE USER SPECIFICATIONS STORED DURING THE PARSE STAGE. WHEN INVOKED, THIS SUBROUTINE WILL READ RECORDS FROM THE INDEX FILE UNTIL A MATCH IS FOUND. WHEN A MATCH IS FOUND THE SUBROUTINE WILL RETURN THE STARTING RECORD NUMBER AND THE NUMBER OF RECORDS OCCURRING AFTER THE STARTING RECORD. IF A MATCH IS NOT FOUND THEN THE END OF RECORDS FLAG .ENDREC. IS SET TRUE.

INVOCATION:

[CALL] GETINX ( P1, P2, P3, P4 )

P1 :: INTEGER. STARTING RECORD NUMBER
P2 :: INTEGER. COUNT OF RECORDS FOLLOWING P1
P3 :: INTEGER. COUNT OF SUPersonic records
P4 :: LOGICAL. FLAG SIGNALING THE END OF RECORDS

VARIABLE DICTIONARY:

ARCRFT : TABLE CONTAINING AIRCRAFT TYPES
CURREC : THE CURRENT RECORD NUMBER FROM FILE 'INDEX'
ENDATE : TABLE CONTAINING THE END DATES
ENDREC : P3
ENTIME : TABLE CONTAINING THE END TIMES
INDEXFIL : UNIT NUMBER FOR THE INDEX FILE
INTDAT : INTEGER REPRESENTING YMDMD DATE
LOOP : SYMBOL REPRESENTING STATEMENT LABEL 1
MSSNS : TABLE CONTAINING THE MISSION/EXERCISE NAMES
MXDATE : MAXIMUM NUMBER OF DATE ALLOWED
MXMSSN : MAXIMUM NUMBER OF MISSION ALLOWED
MXPLNS : MAXIMUM NUMBER OF AIRCRAFT ALLOWED
MXREPS : MAXIMUM NUMBER OF REPETITIONS OF SITE CARDS ALLOWED
MXSITE : MAXIMUM NUMBER OF SITES LOCATIONS ALLOWED
MXTIME : MAXIMUM NUMBER OF START/END TIMES ALLOWED
MOPREC : NUMBER OF OVERPRESSURE RECORDS.
NUMREC : NUMBER OF RECORDS IN THE INDEX FILE
NUMREP : NUMBER OF REPETITIONS STORED DURING PARSE
OPR : TRUE IF THE TRACK CONTAINS OVERPRESSURE RECORD
RECBUF : BUFFER TO HOLD ONE RECORD FROM FILE 'INDEX'
RECTOT : P2
SITES : TABLE CONTAINING THE SITE LOCATIONS
STREC : P1
STTIME : TABLE CONTAINING THE STARTING TIMES
SUPREC : STARTING OVERPRESSURE RECORD
TAILNM : TABLE CONTAINING THE AIRCRAFT TAIL NUMBERS
SUBROUTINE GETINX(STREC, RECTOT, SUPREC, ENDREC, 11, NOPREC, OPR)

C

EXTERNAL STRMCH, INTMCH, FILBUF

PARAMETER(MXDATE=10, MXSSN=10, MXPLNS=10,
MXREPS=5, MXSITE=20, MXTIME=10)

COMMON /CHRTABS/ ARCRFT, MSSNS, SITES, TAILNM
COMMON /INTTABS/ ENDATE, ENTIME, STDATE, STTIME, NUMREP

INTEGER ENDATE(MXREPS, MXDATE)
INTEGER ENTIME(MXREPS, MXTIME)
INTEGER STDATE(MXREPS, MXDATE)
INTEGER STTIME(MXREPS, MXTIME)
INTEGER NUMREP, STREC, RECTOT, CURREC, INTDAT, LOOP
INTEGER TIME1, TIME2, IDXFIL,
NUMREC, TBLIDX, SUPREC

CHARACTER*6 ARCRFT(MXREPS, MXPLNS)
CHARACTER*16 MSSNS(MXREPS, MXSSN)
CHARACTER*10 SITES(MXREPS, MXSITE)
CHARACTER*8 TAILNM(MXREPS, MXPLNS)
CHARACTER*110 FILBUF, RECSUF

LOGICAL ENDREC, STRMCH, INTMCH, OPR

SAVE CURREC

DATA CURREC /0/
DATA IDXFIL /51/
ASSIGN 1 TO LOOP

IF (CURREC.LE.1) THEN

C.. READ THE HEADER RECORD TO GET THE NUMBER OF RECORDS.
C.
C.. CURREC = CURREC + 1
READ(IDXFIL,FMT='(16)',REC=CURREC) NUMREC
ENDREC = .FALSE.
NUMREC = 2
NUMREC = NUMREC - 1
END IF

TBLIDX = 0

C-- GET THE STARTING RECORD
RECBUF = FILBUF(CURREC,NUMREC,IDXFIL,ENDREC)

C LOOP:
1 IF (ENDREC.EQV..TRUE.) RETURN
TBLIDX = TBLIDX + 1
IF (TBLIDX.GT.NUMREP) THEN
C-- READ IN ANOTHER RECORD FOR TESTING
RECBUF = FILBUF(CURREC,NUMREC,IDXFIL,ENDREC)
TBLIDX = 1
END IF

C-- CHECK IF SITE LOCATIONS MATCH
IF (STRMCH(RECBS(27:36),TBLIDX,SITES,MXREPS,MXSITE) + .EQV..TRUE.) THEN
C-- CHECK IF THE MISSION NAMES MATCH
IF (STRMCH(RECBS(1:16),TBLIDX,MSSNS,MXREPS,MXMSSN) + .EQV..TRUE.) THEN
C-- CHECK IF THE DATE INTERVALS CORRESPOND.
C-- FIRST CONVERT THE DATE TO YYMMDD INTEGER
READ(RECBS(7:18),FMT='(12)') INTOAT
INTDAT = INTOAT * 100
READ(RECBS(20:21),FMT='(12)') I
INTDAT = INTOAT + I
READ(RECBS(23:24),FMT='(12)') I
INTDAT = INTOAT + (I * 10000)

IF (INTMCH(INTDAT,TBLIDX,STDATE,ENDATE,MXREPS,MXDATE) + .EQV..TRUE.) THEN
C-- CHECK IF THE TIME INTERVALS CORRESPOND.
READ(RECBS(37:40),FMT='(14)') TIME1
READ(RECBS(45:48),FMT='(14)') TIME2
IF ((INTMCH(TIME1,TBLIDX,STTIME,ENTIME, + MXREPS,MXTIME).EQV..TRUE.).OR. + (INTMCH(TIME2,TBLIDX,STTIME,ENTIME, + MXREPS,MXTIME).EQV..TRUE.) THEN
C-- CHECK IF AIRCRAFT TYPES MATCH.
IF (STRMCH(RECBS(55:60),TBLIDX,ARCRFT, + MXREPS,MXPLNS).EQV..TRUE.) THEN
C-- CHECK IF THE AIRCRAFT TAIL NUMBERS MATCH
IF (STRMCH(RECSUF(61:68), TBLIDX, TAILNM, MXREPS, MXPLNS).EQV., TRUE.) THEN
WE HAVE A SUCCESSFUL MATCH
READ(RECBUF(69:78), FMT='(110)') STREC
READ(RECBUF(79:88), FMT='(110)') RECTOT
READ(RECBUF(89:98), FMT='(110)') SUPREC
READ(RECBUF(99:108), FMT='(110)') NOPREC
READ(RECBUF(109:109), FMT='(L1)') OPR
 11 = CURREC
RETURN
ELSE
  GO TO LOOP
END IF
ELSE
  GO TO LOOP
END IF
ELSE
  GO TO LOOP
END IF
ELSE
  GO TO LOOP
END IF
ELSE
  GO TO LOOP
END IF
ELSE
  GO TO LOOP
END IF
ELSE
  GO TO LOOP
END IF
SUBROUTINE: RTRACE

PROGRAMMER: PHILIP J. DAY
XONTECH INC.
BBN LABORATORIES
DATE: OCTOBER 22, 1986

PURPOSE: INITIAL DRIVER FOR THE TRAPS ROUTINES. RTRACE CALLS
THE ROUTINES BREAKS THE FLIGHT TRACK UP INTO SEGMENTS,
CREATE THE SPLINES, DO MANEUVER SCREENING AND PHI ANGLE
SELECTION, AND DO THE RAY TRACING. THE FLIGHT TRACK
INFORMATION IS CONVERTED FROM FEET TO METERS BEFORE IT IS
PROCESSED.

SUBROUTINE RTRACE

COMMON /FLIGHT/NFP,FTIME,FX,FY,FZ,VX,VY,VZ,FMACH,CA
DIMENSION FTIME(1156),FX(1156),FY(1156),FZ(1156)
DIMENSION VX(1156),VY(1156),VZ(1156),FMACH(1156),CA(1156)

COMMON /SPLINE/NSP,S(100,3),A(100,3),B(100,3),C(100,3),D(100,3)
REAL S,A,B,C,D
INTEGER NSP

COMMON /RAYLIM/ NLIMS,BEG(2),END(2)

COMMON /PRINTS/TITLE(30),TIMLBL
CHARACTER*4 TITLE
CHARACTER*8 TIMLBL

COMMON /PRINTC/KTPSIG,VCRTIM
LOGICAL CVRTIM

COMMON /ACIDNT/ IDENT
CHARACTER*8 IDENT

COMMON /ACWEIG/ ACWT,ACL

COMMON /UNI:/,UNITX,UNITH
CHARACTER*8 UNITX,UNITH

COMMON /RYCTRL/NORAYS,STND,UL,UR,LL,LR,PRTRAY,TIMBEG,DELTIM,WT:MS,
+PHIBEG,DELPHI,NPHIS
LOGICAL NORAYS,STND,UL,UR,LL,LR,PRTRAY,LOGIC(2,2)
EQUIVALENCE (LOGIC(1,1),UL)
REAL PHIB(8),DPHI(8),SGN(2)
INTEGER MDX(2,2)
COMMON /RAYNIT/ KGMH,NDCRVS,NUCRVS,1UPOWN,T0,PHIO,XO,YO,ZO,
+P10,P20,P30,OMEGA,DELTAO,P1FO,P2FO,P3FO,OMEGA,F,XTO,YTO,ZTO,
+P1F0,P2F0,P3F0,OMEGA,F,XS0,YS0,ZS0,P3G0,RH00,PCONST,AGES,AGES(20)
INTEGER KGMH,NDCRVS,NUCRVS,1UPOWN
LOGICAL BETWEEN

COMMON /GROUND/ GLAYER,ZGRND,UGRND,REFLFC
INTEGER GLAYER

COMMON /CAUSTC/ NMC,TRACE,CT(360),CPHI(360),CXYZ(360,3)
REAL CXYZ
LOGICAL TRACE

COMMON /RPOSN/ NPTR,CPOSN,RT(200),RXYZ(200,3),RAGE(200),
+RPFACT(200),RLIFT,REMEM
REAL RT,RXYZ,RAGE,RPFACT,RLIFT
INTEGER NPTR,CPOSN
LOGICAL REMEM

COMMON /INDEXR/ INDREC

COMMON /STSPLN/ ISTT
INTEGER ISTT

REAL V(100,3),DISP(100,3),CONV,TIME,T(100),ANG(360),OP(360)
REAL CLANG
INTEGER NSEG,START(100),ENDD(100),N,CINDFL,SSR,SER
LOGICAL FLAG,RCBLG,STRT,SCRFLG,CFLAG,CAUSTC,STATT,FOJND
CHARACTER BUFF1*128,BUFF2*128

DATA CONV/.3048/,TMAX/2.2/
DATA INDFIL/1/,CINDFL'51/

IF (NFP.LE.0) RETURN
REWIND 8
CALL FFUNC(IDENT,FOJND)
IF (.NOT.FOUND) RETURN
C-

CHECK TO SEE IF THE FLIGHT HAS BEEN PROCESSED
C
READ(INDFL,5000,REC=INDREC) BUFF1
READ(CINDFL,'(I11)',REC=1,ERR=8001) KREC
K = 1
8000 CONTINUE
READ(CINDFL,5001,REC=K,ERR=8001) BUFF2
IF (BUFF2(1:68).EQ.BUFF2(1:68)) THEN
    RETURN
ENDIF
K = K + 1
IF (K.GT.KREC) GOTO 8001
GOTO 8000
8001 CONTINUE

REFLFC = 1.0
HTUNIT = 'FT'
WTUNIT = 'KG'
TIMLBL = 'SSSSSS'

C GET THE FLIGHT SEGMENTS
CALL GETSEG(FTIME,FMACH,CA,FZ,NFP,START,ENDO,NSEG)

C CONVERT EACH TRACK INTO METRIC UNITS
DO 50 J = 1,NFP
  VX(J) = VX(J)*CONV
  VY(J) = VY(J)*CONV
  VZ(J) = VZ(J)*CONV
  FX(J) = FX(J)*CONV
  FY(J) = FY(J)*CONV
  FZ(J) = FZ(J)*CONV
50 CONTINUE

STRT = .TRUE.
CAUSTC = .FALSE.

DO 100 I = 1,NSEG
  L = 1

C CALCULATE A CUBIC SPLINE ABOUT THE VELOCITY VECTORS
ISTT = START(I)
DO 110 J = START(I),ENDO(I)
  V(L,1) = VX(J)
  V(L,2) = VY(J)
  V(L,3) = VZ(J)
  DISP(L,1) = FX(J)
  DISP(L,2) = FY(J)
  DISP(L,3) = FZ(J)
  T(L) = FTIME(J)
  L = L + 1
110 CONTINUE

NSP = ENDO(I) - START(I) + 1
IF (NSP.LE.2) GOTO 100
CALL SPLINE(T,V,NSP,100,S,A,B,C,D)
DO 13 I14 = 1,NSP
13 CONTINUE
CALL LSQUAR(T)
DO 14, I14 = 1,NSP
CONTINUE
j = 1
FLAG = .TRUE.

120 CONTINUE
C
C- STEP THROUGH SEGMENT AT EACH TRACK POINT.
C- IF THERE IS A GAP OF 2.2 SECONDS OR MORE THEN INTERPOLATE
C- AT A TIME HALF WAY BETWEEN POINTS.
C
IF (J.GT.1) THEN
IF ((T(J) - T(J-1)).GT.TMAX.AND.FLAG) THEN
   TO = T(J-1) + (T(J) - T(J-1))/2.0
   FLAG = .FALSE.
   NODE = J - 1
ELSE
   TO = T(J)
   FLAG = .TRUE.
   NODE = J
ENDIF
ELSE
   TO = T(J)
   FLAG = .TRUE.
   NODE = J
ENDIF
NODEC = NODE + START(I) - 1
END IF

C- CALCULATE AIRCRAFT MOVEMENT PARAMETERS AND LIMITING ANGLES
C
NUMC = 0
KATTER = 0
KATTSR = 0
CALL TACMOV(TO,NODE,NODEC,.TRUE.)
IF (NODE.LE.O) GOTO 126
CALL FILIMS("126")
C
C- DO MANEUVER STRENGTH SCREENING
C
CALL SCREEN(ANG,NANGZGRND)
C
C- TRACE RAYS AT APPROPRIATE ANGLES
C
TRACE = .FALSE.
SCRFLG = .TRUE.
STATT = .TRUE.
KATTSR = 0
KATTER = 0

DO 125 K = 1,NANG
   REWIND 9
   WRITE(9,"(*") IDENT
MPTR = 0
OP(K) = 0.0
PHIO = ANG(K)
CALL RAYORG(*125)
CALL RAYTRK(.FALSE.,RCELG,CFLAG,*124)

DO 6543 III = 1,NPTR
   CALL RDSPCL
   CALL SIGNUR(OP(K))
   CALL SAVRAY
   CALL STORE(11,STRT,.FALSE.,.FALSE.,IREC)
   IF (STATT) THEN
      KATTSR = IREC
      SSR = IREC
      STATT = .FALSE.
   ELSE
      KATTER = IREC
   ENDIF
   STRT = .FALSE.
   124 IF (CFLAG) THEN
      CAUSTC = .TRUE.
   ENDIF
   125 CONTINUE
   126 CONTINUE
C GO INTO FOCUS DRIVER
C IF (((NUMC.GT.0) .AND. (J .GT. 1)) THEN
   TRACE = .TRUE.
   REWIND 12
   WRITE(12,*')
   REWIND 12
   CALL FOCMAP(TO,NCOE,NODEC,ANG,NANG,RRCURV,CAUSTC)
   CALL RBRAYS
   ENDIF
C SIDELINE ATTENUATION TO 80 DB
C IF (((KATTER-KATTSR) .GT. 2) THEN
   CALL SIDATT(KATTSR,KATTER,ANG,NANG,SSR)
   ENDIF
J = J + 1
IF (J.LE.NSP) GOTO 120
100 CONTINUE
C ENTIRE TRACK IS PROCESSED. CLOSE THE SECTION IN THE OUTPUT FILES
C CALL STORE(0,.FALSE.,.TRUE.,CAUSTC,IREC)
C - CONVERT EACH TRACK BACK FROM METRIC UNITS
C
DO 55 J = 1, NFP
   VX(J) = VX(J)/CONV
   VY(J) = VY(J)/CONV
   VZ(J) = VZ(J)/CONV
   FX(J) = FX(J)/CONV
   FY(J) = FY(J)/CONV
   FZ(J) = FZ(J)/CONV
55 CONTINUE
RETURN
5000 FORMAT(A90)
5001 FORMAT(A110)
6000 FORMAT('NUMBER OF ANGLES = ',I4)
6001 FORMAT(3X,I4,3X,F10.4,3X,F10.4)
6002 FORMAT(5X,F10.4)
6005 FORMAT('******** RAY RECURVES BELOW THE GROUND ',
     + '********,///)
END
C=============================================================================
C SUBROUTINE: SPLINE
C PROGRAMMER: CURTIS F. GERALD
C MODIFIED BY: PHILIP J. DAY
C XONTECH INC.
C BBN LABORATORIES
C DATE: OCTOBER 20, 1986
C
C THIS ROUTINE COMPUTES THE MATRIX FOR FINDING THE COEFFICIENTS
C OF A CUBIC SPLINE THROUGH A SET OF DATA.
C THE SYSTEM IS THEN SOLVED TO OBTAIN THE SECOND DERIVATIVE VALUES.
C USING THE SECOND DERIVATIVES THE SPLINE COEFFICIENTS ARE CALCULATED.
C
C PARAMETERS:
C THE SECOND DIMENSION OF THE ARRAYS CORRESPOND
C TO X, Y, AND Z DIRECTIONS.
C INPUT: TYPE
C T(SIZE,3) R : ARRAY OF TIME VALUES
C V(SIZE,3) R : ARRAY OF VELOCITY VALUES
C N I : NUMBER OF POINTS
C
C OUTPUT:
C S(100,3) R : ARRAY OF SECOND DERIVATIVES
C A(100,3) R : SPLINE COEFFICIENT
C B(100,3) R : SPLINE COEFFICIENT
C C(100,3) R : SPLINE COEFFICIENT
C D(100,3) R : SPLINE COEFFICIENT
C
C VARIABLES:
C M(N,4,3) R : AUGMENTED MATRIX OF COEFFICIENTS A
C R.H.S. FOR FINDING S
C DT1 R : DELTA TIME
C DT2 R : DELTA TIME
C DT R : DELTA TIME
C DV1 R : DELTA VELOCITY
C DV2 R : DELTA VELOCITY
C NM1 I : N MINUS ONE
C NM2 I : N MINUS TWO
C
C=============================================================================
C SUBROUTINE SPLINE (T, V, N, SIZE, S, A, B, C, D)

INTEGER SIZE
REAL T(SIZE), V(SIZE,3), S(SIZE,3), A(SIZE,3), B(SIZE,3)
REAL C(SIZE,3), D(SIZE,3)
REAL DT1, DT2, DV1, DV2, M(1000,4,3)

INTEGER NM1, NM2

C COMPUTE FOR THE N-2 ROWS
NM2 = N - 2
NM1 = N - 1

DO 1000 K = 1,3
  DT1 = T(2) - T(1)
  DV1 = (V(2,K) - V(1,K))/DT1*6.0

DO 10 I = 1, NM2
  DT2 = T(I+2) - T(I+1)
  DV2 = (V(I+2,K) - V(I+1,K))/DT2*6.0
  M(I,1,K) = DT1
  M(I,2,K) = 2.0*(DT1 + DT2)
  M(I,3,K) = DT2
  M(I,4,K) = DV2 - DV1

  DT1 = DT2
  DV1 = DV2
  10 CONTINUE

C SET UP PARABOLIC END CONDITION FOR THE END OF THE SPLINE.

M(1,2,K) = M(1,2,K) + V(2,K) - V(1,K)
M(NM2,2,K) = M(NM2,2,K) + V(N,K) - V(NM1,K)

C NOW WE SOLVE THE TRIDIAGONAL SYSTEM. FIRST REDUCE.

DO 110 I = 2, NM2
  M(I,2,K) = M(I,2,K) - M(I,1,K)/M(I-1,2,K)*M(I-1,3,K)
  M(I,4,K) = M(I,4,K) - M(I,1,K)/M(I-1,2,K)*M(I-1,4,K)
  110 CONTINUE

C NOW WE BACK SUBSTITUTE

M(NM2,4,K) = M(NM2,4,K)/M(NM2,2,K)
DO 120 I = 2, NM2
  J = NM1 - I
  M(J,4,K) = (M(J,4,K) - M(J,3,K)*M(J+1,4,K))/M(J,2,K)
  120 CONTINUE

C NOW PUT THE VALUES INTO THE S VECTOR

DO 130 I = 1, NM2
  S(I+1,K) = M(I,4,K)
  130 CONTINUE

C FOR LINEAR ENDS, S(1) = 0, S(N) = 0.

S(1,K) = S(2,K)
S(N,K) = S(NM1,K)
1000 CONTINUE
C
C CALCULATE THE SPLINE COEFFICIENTS
C
DO 200 I = 1,NM1
   DT = T(I+1) - T(I)
   DO 210 K = 1,3
      A(I,K) = (S(I+1,K) - S(I,K))/(6.0*DT)
      B(I,K) = S(I,K)/2.0
      C(I,K) = (V(I+1,K) - V(I,K))/DT -
               (2.0*DT*S(I,K) + DT*S(I+1,K))/6.0
      D(I,K) = V(I,K)
210 CONTINUE
200 CONTINUE
C
C INTERPOLATE THE LAST C COEFFICIENT USING THE PREVIOUS
C SPLINE POINT
C
   DT = T(N) - T(NM1)
   DO 300 K = 1,3
      A(N,K) = 0.0
      B(N,K) = S(N,K)/2.0
      C(N,K) = 3.0*A(NM1,K)*DT*DT + 2.0*B(NM1,K)*DT + C(NM1,K)
      D(N,K) = V(N,K)
300 CONTINUE
C
C END OF SPLINE ROUTINE
C
END
SUBROUTINE GETSEG

PROGRAMMER: PHILIP J. DAY
KONTECH INC.
BBN LABORATORIES
DATE: OCTOBER 23, 1986

PURPOSE: TO DIVIDE THE FLIGHT INTO SEGMENTS WHERE THE POINTS ARE
ABOVE THE CRITICAL MACH NUMBER. THE FIRST TWO AND THE
LAST TWO POINTS OF A SEGMENT CAN BE BELOW CRITICAL. THIS
IS DONE IN ORDER TO IMPROVE THE SPLINE INTERPOLATION.
THERE CAN ALSO BE SUBCRITICAL POINTS IN THE TRACK; HOWEV
THERE CAN ONLY BE AT MOST 5.5 SECONDS BETWEEN CRITICAL
POINTS. IF THERE IS A 4.5 SECOND GAP BETWEEN DATA POINT
THE SEGMENT IS ALSO TERMINATED.

PARAMETERS:

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>INPUT:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td>R</td>
<td>ARRAY OF TIMES (S)</td>
</tr>
<tr>
<td>MACH</td>
<td>R</td>
<td>ARRAY OF MACH NUMBERS</td>
</tr>
<tr>
<td>CA</td>
<td>R</td>
<td>ARRAY OF CLIMB ANGLES (DEG)</td>
</tr>
<tr>
<td>Z</td>
<td>R</td>
<td>ARRAY OF HEIGHTS (FEET)</td>
</tr>
<tr>
<td>NPTS</td>
<td>I</td>
<td>NUMBER OF DATA POINTS</td>
</tr>
</tbody>
</table>

OUTPUT:           |
| START           | I    | INDEX ARRAY FOR START OF S|
| ENDD            | I    | INDEX ARRAY FOR END OF SEGMENT|
| NSEG            | I    | NUMBER OF SEGM |

SUBROUTINE GETSEG(TIME,MACH,CA,Z,NPTS,START,ENDD,NSEG)

REAL TIME(NPTS),MACH(NPTS),CA(NPTS),Z(NPTS)
INTEGER NPTS

INTEGER START(NPTS),ENDD(NPTS),NSEG

COMMON /ACIDNT/ IDENT
CHARACTER*8 IDENT

COMMON /ACWIEG/ ACWT,ACL

COMMON /CARL/ BOMFCT
REAL BOMFCT, BOMF

COMMON /GROUND/ GLAYER,ZGRND,CGRND,UGRND,VGRND,REFLFC
INTEGER GLAYER

REAL MC,ME,PI,GAMMA
REAL BRMAX, MCMAX  
INTEGER I,F,BKPT  
LOGICAL STARTED,BREAK,FIRST  

C- BRMAX FOR BREAK IN DATA  
C- MCMAX FOR TIME BELOW CRITICAL  

DATA BRMAX/4.5/, MCMAX/5.5/, PI/3.1415927/  

NSEG = 0  
I = 1  
BKPT = 1  
STARTED = .FALSE.  
BOMFCT = 0.0  
FIRST = .TRUE.  

100 CONTINUE  
IF (Z(I).GE.6.0E6) THEN  
    NSEG = 0  
    NPTS = 0  
    RETURN  
ENDIF  

C- CALCULATE CRITICAL MACH AND AIRCRAFT'S MACH  

MC = EXP(4.033E-06*ANIN1(Z(1X,35300.)))  
GAMMA = CA(I)*PI/180.0  
IF (MACH(I).GT.1.0) THEN  
    ME = 1.0/SIN(GAMMA + ATAN(1.0/SQRT(MACH(I)**2 - 1.0)))  
ELSE  
    ME = 0.0  
ENDIF  

C- CALCULATE THE PEAK OVERPRESSURE USING CARLESON'S METHOD  

IF (ME.GE.MC) THEN  
    PRINT *, 'ZGRND Z(I) GAMMA MACH(I) HE', ZGRND, Z(I), GAMMA, MACH(I), ME  
    HE = (Z(I)-ZGRND)*COS(GAMMA)  
    ASIGA = (1.0 - 6.8756E-06*Z(I))**5.2559  
    ASIGG = (1.0 - 6.8756E-06*ZGRND)**5.2559  
    BOMF = (8400.0 * SQRT(ASIGA*ASIGG)*  
        + (MACH(I)**2 - 1)**0.125)/(HE**0.75)  
    CALL OPFIND(IDENT,OPFACT,ACWT)  
    BOMF = BOMF*OPFACT  
ELSE  
    BOMF = 0.0  
ENDIF  

BOMFCT = MAX(BOMFCT, BOMF)  

C- START A TRACK  
C
IF (ME.GT.MC.AND..NOT.STARTED) THEN
   STARTED = .TRUE.
   NSEG = NSEG + 1
   IJ = MAX0(I-1,2)
   DO 110 IJ = 1,IJ,-1
      IF ((TIME(IJ) - TIME(IJ-1)) .GT. BRMAX) THEN
         BKPT = IJ
         GOTO 120
      END IF
   110 CONTINUE
   CONTINUE
   START(NSEG) = MAX0(BKPT,1-2)
ENDIF

C- IS THERE A GAP IN THE DATA ?
C
   IF (I.LT.NPTS) BREAK = (TIME(I+1) - TIME(I)).GT.BRMAX)
   IF (STARTED.AND.BREAK) THEN
      STARTED = .FALSE.
      FIRST = .TRUE.
      ENDD(NSEG) = 1
      BKPT = 1 + 1
   ENDIF

C- POINT BELOW CRITICAL ?
C
   IF (STARTED.AND.ME.LE.MC) THEN
      IF (FIRST) THEN
         FIRST = .FALSE.
      ENDIF
   ENDIF

C- IS MACH BELOW CRITICAL LONG ENOUGH ?
C
   IF (.NOT.FIRST) THEN
      BREAK = (TIME(I) - TIME(F)).GT.MCMAX)
      IF (BREAK) THEN
         STARTED = .FALSE.
         FIRST = .TRUE.
         ENDD(NSEG) = MIN0(F+1,I)
         I = F - 1
      ELSE
         C- ABOVE CRITICAL AGAIN
         C
            IF (ME.GT.MC) FIRST = .TRUE.
         ENDF
      ENDIF
   C
   C- LOOP CONTROLE
   C
   I = I + 1
IF (I.LE.NPTS) GOTO 100

IF (NSEG.GT.0.AND.STARTED) ENDD(NSEG) = NPTS

CLOSE(55)
DO 3333 III = 1,NSEG
3333 CONTINUE

RETURN
END
SUBROUTINE SCREEN

PROGRAMMER: PHILIP J. DAY
BBN LABORATORIES

DATE: NOVEMBER 3, 1986

PURPOSE: TO DECIDE WHETHER THE MANEUVER IS STRONG OR WEEK AND
BASED ON THIS TO CALCULATE AN ARRAY OF PHI ANGLES
INDICATING THE RAYS TO BE TRACED. FOR STRONG MANEUVE
THE GROUND SPACING IS APPROXIMATELY 500 FT. FOR WEEK
MANEUVERS IT IS 2000 FT.

PARAMETERS:

NAME     TYPE    DESCRIPTION
--------------
INPUT: MONE

OUTPUT: PHI (400)  R : ARRAY OF PHI ANGLES
FIRST ELEMENT CONTAINS INDEX FOR THE FIRST MA
LAST ELEMENT IS LAST M
NPNI  I : NUMBER OF ANGLES

SUBROUTINE SCREEN(PHI, NPNI, ZGRND)

COMMON /ACSPOT/ TIME, XR, YR, ZR, XDOT, YDOT, ZDOT, AIRSPD, ASPDOT,
+ CO, UO, VOT, XMACH, XMADOT, XU, XMUDOT, COSMU,
+ SINMU, EX(3,3), EKDOT(3,3), GLOAD, HEADIN, CLIMB, BANK,
+ XDDOT, YDDOT, ZDDOT, XDODDOT, YDOTDOT, ZDOTDOT

COMMON /RAYLIM/ NLIMS, BEG(2), END(2)

COMMON /PHILIM/ MINPHI, MAXPHI
REAL MINPHI, MAXPHI

COMMON /CARL/ BOMFCT
REAL BOMFCT

REAL PHI(360)
INTEGER NPNI
REAL ENDPHI, TMU, DELPHI, PI
REAL THETA1, THETA2, DPHI, PMIT, XMOT, YNOT, XT, YT, XLAST, YLAST
REAL V(3), A(3), VHAT(3), AHAT(3), WHAT(3), TEMP(3), MACH, MACHP
REAL ALPDOT, MUDOT, THETAS, THETSP, ADOTP, DRAY, MAXD, CLANG
INTEGER MNPHI

EQUIVALENCE (V(1), XDOT), (A(1), XDOT)
LOGICAL STRONG,LAST

C DPHI IS ONE DEGREE
C
DATA DELPHI/1.0/,PI/3.1415927/,MINPHI/400/

CLANG = CLIMB*PI/180.0
C
C CALCULATE MAXPHI 0.0 <= MAXPHI <= PI
C CALCULATE MIXPHI 0.0 >= MIXPHI >= PI
C
MAXPHI = AMOD(AMOD(END(1),360.)-540.,360.)-180.
MINPHI = AMOD(AMOD(BEG(1),360.)-540.,360.)-180.
C
C CHECK TO SEE IF THERE IS ONLY ONE ANGLE
C
PHI0 = 0.0
PHI(2) = PHI0
NPHI = 2
IF (MINPHI.EQ.MAXPHI) THEN
PHI(1) = 0.0
NPHI = 1
RETURN
ENDIF
C
C DO MANEUVER SCREENING AND SELECT A PHI WITH MAXIMUM OVER PRESSURE
C
MACH = SQRT(DOTP(V,V,3)/(CD**2))
MACHP = DOTP(A,V,3)/(CD*RNORM(V,3))
MUDOT = .10*MACHP/(MACH*MACH - 1.0))

CALL UNIT(V,VHAT,3)
CALL UNIT(A,AHAT,3)

CALL CROSS(VHAT,AHAT,TEMP)
CALL CROSS(VHAT,TEMP,WHAT)

ALPDOT = DOTP(A,WHAT,3)/RNORM(V,3)
THETAS = ASIN(-1.0*WHAT(3))
C
C CALCULATE PHI ANGLE WITHIN MARGINS WITH THE LARGEST OVER PRESSURE
C
IF (ABS(MUDOT).LT.ABS(ALPDOT+MUDOT)) THEN
IF (THETAS.GT.MAXPHI) THEN
THETSP = MAXPHI
ELSE
THETSP = THETAS
ENDIF
ELSE
THETSP = THETAS - PI
IF (THETSP.LT.MINPHI) THEN
THETSP = MINPHI
ENDIF

ENDIF
END IF
END IF

ADOTP = DOTP(A, WHAT, 3)/RNORM(V, 3)*COS(THETSP-THETAS)

DRAY = MUDOT+ADOTP

IF (ZRO.GT.4572) THEN
  DPHI = DELPHI
ELSE
  DPHI = 2.*DELPHI
ENDIF

PHI(1) = MINPHI
PHI(2) = AINT(MINPHI)
J = 2

C  CALCULATE THE REST OF THE PHI ANGLES
C
1000 CONTINUE
  J = J + 1
  PHI(J) = PHI(J-1) + DPHI
  IF (PHI(J).LT.MAXPHI) GOTO 1000
  PHI(J) = MAXPHI
  NPHI = J

RETURN
END
SUBROUTINE: STORE
PROGRAMMER: PHILIP J. DAY
XOMTECH INC.
BBN LABORATORIES
DATE: DECEMBER 15, 1986

PURPOSE: TO STORE GROUND LOCATIONS AND OVERPRESSURES OF THE BOOM IN THE CONTOUR LIBRARY FILE "CLIBRY".

PARAMETERS:

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>IFLAG</td>
<td>I</td>
<td>CAUSTIC POINT INDICATOR</td>
</tr>
<tr>
<td>START</td>
<td>L</td>
<td>TRUE FOR THE FIRST POINT IN NEW TRACK</td>
</tr>
<tr>
<td>TERM</td>
<td>L</td>
<td>TRUE FOR THE LAST POINT IN TRACK</td>
</tr>
<tr>
<td>CAUSTIC</td>
<td>L</td>
<td>SET IF THERE WAS A FOCUS IN GROUND REGION</td>
</tr>
<tr>
<td>IKREC</td>
<td>I</td>
<td>RECORD NUMBER OF THE CURRENT STORED RAY DATA</td>
</tr>
</tbody>
</table>

VARIABLES:

| ENDREC | I    | LAST RECORD OF THE CLIBRY BE ADDITIONS |
| FFTFLG | L    | FLAG SET TRUE IF AN FFT WAS CALCULATED FOR THE SEL |
| FILBUF | C    | CHARACTER BUFFER |
| I1 | I    | INTEGER DUMMY VARIABLE |
| I2 | I    | INTEGER DUMMY VARIABLE |
| I3 | I    | INTEGER DUMMY VARIABLE |
| INDREC | I    | RECORD NUMBER OF THE ENTRY INDEX FILE |
| IREC | I    | RECORD NUMBER OF THE CURRENT IN THE CINDEX FILE |
| KREC | I    | CURRENT RECORD TO BE WRITTEN |
| L1 | L    | LOGICAL DUMMY VARIABLE |
| NOPREC | I    | NUMBER OF PEAK OVER PRESSURE RECORDS |
| NREC | I    | NUMBER OF GROUND POINT RECORDS |
| OREC | I    | STARTING LOCATION FOR PEAK O PRESSURE RECORDS |
| OPC | R    | PEAK OVER PRESSURE OF A CAUSTIC |
| OPG | R    | OVER PRESSURE ON THE GROUND |
| PHIO | R    | PHI ANGLE OF THE RAY |
| SEL | R    | SOUND EVENT LEVEL |
| STREC | I    | STARTING RECORD IN THE CLIBRY |
SUBROUTINE STORE(IFLAG,START,TERM,CAUSTC,KREC)

LOGICAL CAUSTC,START,TERM
INTEGER KREC

COMMON /GROUND/ GLAYER,ZGRND,GGRND,UGRND,VGRND,REFLFC
INTEGER GLAYER

COMMON /HACSPT/ HACM(41)

COMMON /RAYOUT/ T0,PHIO,XK(3),OPG,SEL
COMMON /INDEXR/ INDREC

REAL TC,XC,YC,OPC

INTEGER ENDFIL,STREC,NREC,OPREC,NOPREC,IFLAG
INTEGER EN1,12,13,INDREC,IREC

CHARACTER FBUF*98
COMMON /PHILIM/ MINPHI,MAXPHI
REAL MINPHI,MAXPHI

LOGICAL 1,STRTED
DATA INDFIL/1/,STRTED/.FALSE./

KREC = 0
IF (START.OR.(.NOT.STRTED.AND.TERM)) THEN

STRTED = .TRUE.
ENREC = 1
WINDEX = 1
READ(52,5000,REC=1,ERR=1O) ENREC
GOTO 20
10 ENREC = 1
20 READ(51,5000,REC=1,ERR=30) WINDEX
GOTO 40
30 WINDEX = 1
40 CONTINUE

STREC = ENREC + 1
NREC = 0
READ(INDFIL,FMT=*(A),REC=INDREC) FILBUF
WRITE(52,5005,REC=STREC) FILBUF(1:36),FILBUF(55:68),ZGRND
IF (.TRUE.) THEN
  WRITE(50,5006) FILBUF(1:36),FILBUF(55:68)
  WRITE(50,5007) ZGRND/0.3048
  WRITE(50,5020)
ENDIF

ENDIF

PHIO = AMOD(AMOD(PHIO,360.)*-540.,360.)*-180.

IF (.NOT.TERM) THEN
  WREC = WREC + 1
  KREC = STREC + WREC
  IKREC = KREC

  WRITE(52,5001,REC=KREC) IFLAG,HACM(1),HACM(2),HACM(3),HACM(4),
       TO,XK(1),XK(2),PHIO,OPG,SEL,HACM(14)

  IF (.TRUE.) THEN
    CTOUT = TIMVCVR(HACM(1),5)
    FX = HACM(2)/0.3048
    FY = HACM(3)/0.3048
    FZ = HACM(4)/0.3048
    FFX = XK(1)/0.3048
    FFFY = XK(2)/0.3048
    AOPG = OPG/47.85
    WRITE(50,5004) IFLAG,HACM(1),FX,FY,FZ,PHIO,TO,
                  FFX,FFY,AOPG,SEL,HACM(14)
  ENDIF
ELSE
  OUTPUT THE MAXIMUM OVER PRESSURE FOR TIMES WITH CAUSTICS
  STRTED = .FALSE.
  OPREC = 0
  NOPREC = 0
 REWIND 8

  IF (CAUSTC) THEN
    CONTINUE
    READ(8,5002,END=501) TIM,AX,AY,AZ,TC,XC,YC,PHIO,OPG,SEL,COP
    OPREC = STREC + WREC + 1
    NOPREC = NOPREC + 1
    JOPREC = STREC + WREC + NOPREC
    WRITE(52,5022,REC=JOPREC) TIM,AX,AY,AZ,TC,XC,YC,
                  OPG,SEL,COP
    GOTO 500
  ENDIF
C BRANCH TO HERE IF THERE ARE NO CAUSTICS IN THE FLIGHT TRACK
C
501 CONTINUE
READ(INDFIL,FMT='(A)',REC=INDREC) FILBUF
NIDX = NIDX + 1
WRITE(S1,5003,REC=NIDX) FILBUF(1:68),STREC,NREC,OPREC,
+ NOPREC,CAUSTC
ENDREC = ENDREC + NREC + NOPREC + 1
WRITE(S1,5000,REC=1) NIDX
WRITE(S2,5000,REC=1) ENDREC
REWIND 8
ENDIF
RETURN

801 FORMAT(5F10.4)
5000 FORMAT(L10)
5001 FORMAT(12,F8.2,3F8.0,F8.2,2F8.0,F8.3,F10.4,F10.4,F10.4)
5002 FORMAT(F8.2,3F8.0,F8.2,2F8.0,F8.3,F10.4,F10.4,F10.4)
5003 FORMAT(A68,4110,L1)
5004 FORMAT(12,1X,F10.2,3F8.0,F9.3,F10.2,2F8.0,F11.4,F10.4,F10.4)
5005 FORMAT(3X,A36,2X,A14,F61,F10.2)
5006 FORMAT(3X,A36,2X,A14)
5007 FORMAT(3X,'ALTITUDE OF THE GROUND IS',F12.2,' FT.')
5200 FORMAT('FLAG',3X,'TO',7X,'XO',6X,'YO',6X,'ZO',6X,'PHIO',
+ 7X,'TG',7X,'XG',6X,'YG',7X,'OP',7X,'CSEL',
+ 7X,'MACH')

END
SUBROUTINE SIDATT(KATTSR,KATTER,ANG,NWANG,SSR)

INTEGER KATTSR,KATTER,NWANG,SSR
REAL ANG(400)
REAL HACM(4),XK(3),PI
INTEGER RAYCT
CHARACTER*80 BUFF
LOGICAL RGRND

COMMON /ACSPT/ HACM

COMMON /RAYPTS/ RAY1P,RAY1X,RAY1Y,RAY2P,RAY2X,RAY2Y,GRNDZ,
+ SNOSPD

COMMON /GROUND/ GLAYER,ZGRND,CGRND,UGRND,VGRND,REFLFC
INTEGER GLAYER

COMMON /PHILZM/ MINPHI,MAXPHI
REAL MINPHI,MAXPHI

DATA PI/3.1415926/
GRNDZ = ZGRND
SNOSPD = CGRND

IF (KATTSR.GT.0) THEN
  READ(52,5001,REC=KATTSR) IFLAG,HACM(1),HACM(2),HACM(3),HACM(4),
+ TO,XK(1),XK(2),PHIO,OPG,SEL,MACH

K = 1
100 CONTINUE

RAY2X = XK(1)
RAY2Y = XK(2)
RAY2P = OPG

READ(52,5001,REC=KATTSR+1) IFLAG,TTT,HACM(2),HACM(3),
+ HACM(4),TO,XK(1),XK(2),PHI01,OPG,SEL,MACH
C  
C CHECK TO SEE THAT THE RAYS START AT THE SAME TIME
C
RAY1X = XK(1)
RAY1Y = XK(2)
RAY1P = OPG
C
WRITE(BUFF,1000) ANG(1)
READ(BUFF,1000) AAA
IF (PHIO .GT. AAA) THEN
   ANGLE3 = ANG(1)*PI/180.
   ANGLE2 = PHIO*PI/180.
   ANGLE1 = PHIO*PI/180.
   RGRND = .FALSE.
ELSE
   ANGLE3 = ANG(1)*PI/180.
   ANGLE2 = PHIO*PI/180.
   ANGLE1 = PHIO*PI/180.
   RGRND = .TRUE.
ENDIF
C DO THE EXTRAPOLATION
C
CALL EXTRPR(ANGLE1,ANGLE2,ANGLE3,RGRND,RAYCT)
C
ENDIF
C REPETE FOR THE OTHER SIDELINE
C
IF (KATTER.GT.0) THEN
   READ(52,5001,REC=KATTER) IFLAG,HACM(1),HACM(2),HACM(3),HACM(4),
   + TO,XK(1),XK(2),PHIO,OPG,SEL,MACH
   K = MANG
   CONTINUE
   200
   RGRND = .TRUE.
   KK = K
RAYZX = XK(1)
RAYZY = XK(2)
RAYZP = OPG

READ(52,5001,REC=KATTER-1) IFLAG,TTT,HACM(2),HACM(3),
   + HACM(4),TO,XK(1),XK(2),PHIO,OPG,SEL,MACH
C
C CHECK TO SEE THAT THE RAYS START AT THE SAME TIME
C
RAY1X = XK(1)
RAY1Y = XK(2)
RAY1P = OPG

WRITE(BUFF,1000) ANG(1)
READ(BUFF,1000) AAA
IF (PHI0 .LT. AAA) THEN
  ANGLE3 = ANG(KK)*PI/180.
  ANGLE2 = PHI0*PI/180.
  ANGLE1 = PHI0*PI/180.
  RGRND = .FALSE.
ELSE
  ANGLE3 = ANG(1)*PI/180.
  ANGLE2 = PHI0*PI/180.
  ANGLE1 = PHI0*PI/180.
  RGRND = .TRUE.
ENDIF
C- DO THE EXTRAPOLATION
C
CALL EXTRPR(ANGLE1,ANGLE2,ANGLE3,RGRND,RAYCT)
END IF
1000 FORMAT(F8.3)
RETURN
5001 FORMAT(I2,F8.2,3F8.0,F8.2,2F8.0,F8.3,F10.4,F10.4,F10.4)
5200 FORMAT(I10)
6000 FORMAT(/' !!!!!! NOT ENOUGH INFORMATION FOR EXTRAPOLATION',
     + '/' !!!!!!',/',' TIME =',F12.3)
END
**MODULE NAME**: EXTRPR

**MODULE TYPE**: SUBROUTINE

**PROGRAMMER**: THOMAS REILLY

**XONTECH INC.**

**BBN LABORATORIES**

**DATE**: DECEMBER 9, 1986

**DESCRIPTION**:

This subroutine is designed to extrapolate outside the margin of the last ray down to the threshold of approximat seventy dB. It accomplishes this by receiving two rays and three angles. If the last ray does not hit the ground then it's termination point is calculated and used for the last. The last and next to the last rays are used to extrapolate outside the margin to calculate the new ray's termination and overpressure.

**VARIABLE DICTIONARY**:

- **ANGLE1**: Angle of the first ray (N-1).
- **ANGLE2**: Angle of the second ray (N).
- **ANGLE3**: Angle of the third ray (M).
- **AIRT**: Time the aircraft is at coordinate AIRX, AIRY, AIRZ.
- **AIRX**: X coordinate of the aircraft.
- **AIRY**: Y coordinate of the aircraft.
- **AIRZ**: Z coordinate of the aircraft.
- **DM**: Slant distance of the last ray.
- **DN**: Slant distance of the next to last ray.
- **GRNDZ**: Z coordinate for the ground termination point of the ray.
- **LE**: New rays calculated overpressure in db.
- **LMC**: Modified overpressure values in db, to calculate new ray.
- **PE**: New rays calculated overpressure.
- **PMC**: Modified overpressure values for calculating the new ray.
- **RAY1P**: Pressure of the first ray.
- **RAY1X**: X coordinate of where ray one terminates.
- **RAY1Y**: Y coordinate of where ray one terminates.
- **RAY2P**: Pressure of the second ray.
- **RAY2X**: X coordinate of where ray two terminates.
- **RAY2Y**: Y coordinate of where ray two terminates.
- **RAYCT**: Number of rays in the array RAYPT.
- **RGND**: Boolean flag, true if the last ray reached the ground.
- **SNSPD**: Speed of sound value.
- **TIME**: Termination time of the newly calculated ray.
- **XE,YE**: X and Y coordinates of the extrapolated rays.
- **XM,YM**: X and Y coordinates of the last ray.
*- **XN,YN** - X AND Y COORDINATES OF THE NEXT TO LAST RAY.

*- **MODIFIED**: MARCH 20, 1987

*- **PROGRAMMER**: PHILIP J. DAY

*- **COMMON RAYOUT ADDED**

*- **CALL TO STORE ADDED**

```
SUBROUTINE EXTRPR(ANGLE1, ANGLE2, ANGLE3, RGRND, RAYCT)

COMMON /ACSPT/ AIRT, AIRX, AIRY, AIRZ
COMMON /RAYPTS/ RAY1P, RAY1X, RAY1Y, RAY2P, RAY2X, RAY2Y, GRNDZ,
   + SNDSPO

REAL  LN, PN, LMC, PMC, LE, PE
INTEGER RAYCT, OUTFILE
LOGICAL RGRNO

COMMON /RAYOUT/ TO,PHIO,XK(3),OPG,SEL

PARAMETER (OUTFILE = 4)

RAYCT = 0

*- **CHECK IF THE LAST RAY REACHED THE GROUND.**
*IF (.NOT. RGRNO) THEN

*- **CALCULATE THE RAYS ESTIMATED COORDINATES FROM THE RAY BEFORE I**
XN = RAY2X + (RAY2X - RAY1X) * ((TAN(ANGLE3) - TAN(ANGLE2))
   + / (TAN(ANGLE2) - TAN(ANGLE1)))
YM = RAY2Y + (RAY2Y - RAY1Y) * ((TAN(ANGLE3) - TAN(ANGLE2))
   + / (TAN(ANGLE2) - TAN(ANGLE1)))

*- **CALCULATE THE SLANT DISTANCE FROM THE A/C.**
DN = (XN - AIRX)**2.0 + (YM - AIRY)**2.0 + (GRNDZ - AIRZ)
   + ** 2.0)** 0.5
DN = (RAY2X - AIRX)**2.0 + (RAY2Y - AIRY)**2.0 + (GRNDZ -
   + AIRZ)**2.0)** 0.5

*- **ASSIGN RAY2 TO XN AND YN COORDINATE VARIABLES.**
XN = RAY2X
YN = RAY2Y

*- **CHECK TO SEE IF SLANT DISTANCE LIES BETWEEN .8 AND 1.0**
IF (((0.8 * DM) .LT. DN) .AND. (DN .LT. DM)) THEN
   LN = 10 * LOG10((RAY2P**2.0)) + 68.0
   LN = LN - 10 - 103.2 * LOG10((DN/DM))
   PN = RAY2P * 3.162*((DN/DM)** 5.12)
ELSE
   LN = 10 * LOG10((RAY2P**2.0)) + 68.0
   PN = RAY2P

endif
```

F\text{calculate the LNC and the PMC.}\n\text{LNC} = \text{LW} + 15 \times \log_{10}(\text{DM/DM}) - 10.0\n\text{PMC} = 0.3162 \times \text{PM} \times (\text{DM/DM})^{0.75}\n\text{store the limit point}\n\text{to} = \text{AIRT} \div (\text{DM} / \text{SNOSPO})\n\text{PHIO} = 0.0\n\text{XX}(1) = \text{XM}\n\text{XX}(2) = \text{YM}\n\text{OPG} = \text{PMC}\n\text{SEL} = 0.0\n\text{call store}(0, \text{false}, \text{false}, \text{false}, \text{false}, \text{IREC})\n\text{if the last ray did terminate then}\n\text{else}\n\text{DM} = ((\text{RAYZX} - \text{AIRX})^2 + (\text{RAYZY} - \text{AIRY})^2 + (\text{GRNDZ} + \text{AIRZ}))^{0.5}\n\text{DN} = ((\text{RAY1X} - \text{AIRX})^2 + (\text{RAY1Y} - \text{AIRY})^2 + (\text{GRNDZ} + \text{AIRZ}))^{0.5}\n\text{LNC} = 10 \times \log_{10}(\text{RAYZP}^{2.0}) + 68.0 - 10.0\n\text{PMC} = 0.3162 \times \text{RAYZP}\n\text{assign RAYZ to XM, YM coordinate variables and RAY1 to XN, YN.}\n\text{XN} = \text{RAY1X}\n\text{XM} = \text{RAY2X}\n\text{YN} = \text{RAY1Y}\n\text{YM} = \text{RAY2Y}\n\text{endif}\n\text{INC} = 914.4\n\text{loop until LE \leq 80.}\n\text{continue}\n\text{XE} = \text{INC} \times ((\text{XM} - \text{XM}) + ((\text{XM} - \text{XM})^2 + (\text{YM} - \text{YM})^2 + (\text{GRNDZ} + \text{AIRZ}))^{0.5}) \div \text{XM}\n\text{YE} = \text{INC} \times ((\text{YM} - \text{YM}) + ((\text{XM} - \text{XM})^2 + (\text{YM} - \text{YM})^2 + (\text{GRNDZ} + \text{AIRZ}))^{0.5}) \div \text{YM}\n\text{DE} = ((\text{XE} - \text{AIRX})^2 + (\text{YE} - \text{AIRY})^2 + (\text{GRNDZ} + \text{AIRZ}))^2 \div 0.5\n\text{LE} = \text{LNC} + 25 \times \log_{10}(\text{DM/DE})\n\text{PE} = \text{PMC} \times (\text{DM/DE})^{1.25}\n\text{calculate the time the ray hits the ground.}\n\text{TIME} = \text{AIRT} \div (\text{DE} / \text{SNOSPO})\n\text{check LE value before entering the new ray in the file.}\n\text{if } (\text{LE} \geq 80.0) \text{ then}\n\text{RAYCT} = \text{RAYCT} + 1\n\text{assign variables in the common to store values and call store.}\n\text{TO} = \text{TIME}\n\text{PHIO} = 0.0
XX(1) = XE
XX(2) = YE
OPG = PE
SEL = 0.0
IF (((XX(1) .GT. -39624) .AND. (XX(1) .LT. 39624))
1 .AND. ((XX(2) .GT. -39624) .AND. (XX(2) .LT. 39624))) THEN
    CALL STORE(O,.FALSE.,.FALSE.,.FALSE.,IREC)
ENDIF

C     CALL ENTRAY(XE, YE, PE, TIME, RAYCT)
ENDIF
INC = INC + 914.4

*.- LOOP UNTIL LE * 80 DB.
  IF (LE .GT. 80.0) GO TO 10
CLOSE (OUTFILE)
RETURN
END
*.- END OF SUBROUTINE EXTRRAY.
SUBROUTINE FFUNC

PROGRAMMER: PHILIP J. DAY
XONTECH INC.
BBN LABORATORIES

DATE: MARCH 25, 1987

PURPOSE: TO GENERATE F-FUNCTIONS FOR THE VARIOUS AIRCRAFT IN
THE TABLE. IF THE AIRCRAFT TYPE IS NOT FOUND IN
THE TABLE, AN ERROR MESSAGE IS PRINTED AND PROCESSING
FOR THAT FLIGHT IS ABORTED.

PARAMETERS:

NAME TYPE DESCRIPTION

INPUT:

TYPE C'S AIRCRAFT TYPE

OUTPUT:

FOUND L TRUE IF THE AIRCRAFT TYPE IS FOUND

VARIABLES:

ACTYPE C'S(30) ARRAY OF AIRCRAFT TYPE
ACWT R AIRCRAFT WEIGHT
AKS R(30) ARRAY OF KS VALUES
FAC R(500) ARRAY OF F-FUNCTION A COEFFICIENTS
FLC R(500) ARRAY OF F-FUNCTION L COEFFICIENTS
KS R KS FACTOR
KSSQ R (KS**2) * 3.46
LEGNTH R(30) ARRAY OF AIRCRAFT LEG
LGNSR R AIRCRAFT LEGTH
LGNSQR R SQUARE ROOT OF LEGNTH
STEP R SIZE OF EACH OF THE 1 INTERVALS IN THE F-F
TAU R(500) ARRAY IF F-FUNCTION L
WEIGHT R(30) ARRAY OF AIRCRAFT WEI

SUBROUTINE FFUNC(TYPE, FOUND)

PARAMETERS
CHARACTER*8 TYPE
LOGICAL FOUND

REAL AKS(30), LEGNTH(30), WEIGHT(30)
REAL KS, LGN, LGNSQR, ACWT, KSSQ, STEP
CHARACTER*8 ACTYPE(30)

COMMON /FFTAB/ KRCAC, NSPDS, SPEEDS(11), LOCSPD(10), KTABL,
+ NTAU, TAU(200), FAC(200), FLC(200)
COMMON /CFFTAB/ ACIDNT
CHARACTER*8 ACIDNT
COMMON /ACWEIG/ ACWT,ACL
FOUND = .FALSE.
C
C- FIND THE AIRCRAFT TYPE KS, LENGTH, AND WEIGHT
DO 100 I = 1,30
IF (TYPE.EQ.ACTYPE(I)) THEN
   KS = AKS(I)
   LGN = LENGTH(I)
   ACWT = WEIGHT(I)
   ACL = LENGTH(I)*0.3048
   FOUND = .TRUE.
ENDIF
100 CONTINUE
IF (FOUND) THEN
C
C- GENERATE THE F-FUNCTION
C
   STEP = LGN*0.01*0.3048
   KSSQ = 3.46*KS**2
   LGNSQR = SQRT(LGN*0.3048)
C
C- START WITH A LEADING 0
TAU(1) = 0.0
FAC(1) = 0.0
FLC(1) = 0.0
C
C- COMPUTE THE 101 POINTS
DO 200 I = 2,102
   TAU(I) = STEP*REAL(I-1)
   FAC(I) = (KSSQ*(52.0 - REAL(I))/50.0)*LGNSQR
   FLC(I) = 0.0
200 CONTINUE
C
C- ADD TWO TRAILING ZEROS
TAU(103) = STEP*102.0
FAC(103) = 0.0
FLC(103) = 0.0
TAU(104) = STEP*103.0
FAC(104) = 0.0
FLC(104) = 0.0
NTAU = 104
ELSE
C
C- ACTYPE NOT FOUND PRINT ERROR MESSAGE
C
WRITE(6,6000) TYPE
NTAU = 0

ENDIF

6000 FORMAT(1X,
+ ',/10X,'*****************************:****',
+ ',/10X,'**** AIRCRAFT TYPE ','AB',' NOT FOUND ****',
+ ',/10X,'**** FLIGHT ABORTED ****',
+ ',/10X,'****     ',
+ ',/10X,'*****************************:****',
+ ')

C C ADD A NEW AIRCRAFT TYPE TO THE END OF THE LIST.
C C ENTER THE APPROPRIATE KS, LEGNTH, AND WEIGHT VALUES IN THE
C C FOLLOWING DATA STATEMENTS. REMEMBER TO DECRIMENT THE
C C DUMMY VALUES IN ORDER TO KEEP THE ARRAY SIZE CONSTANT.
C
DATA ACTYPE /'B-1 ','F-4 ','RF-4 ','F-5 ',
+ 'F-14 ','F-15 ','F-16 ','F-18 ',
+ 'F-20 ','F-101 ','F-104 ','F-105 ',
+ 'F-106 ','F-111 ','SR-71 ','T-38 ',
+ 14'X'XXXXXXXX'/

DATA AKS / 0.0910, 0.0880, 0.0880, 0.0642,
+ 0.0873, 0.0838, 0.0838, 0.0900,
+ 0.0643, 0.0860, 0.0690, 0.0860,
+ 0.0840, 0.0892, 0.0870, 0.0642,
+ 14*0.0/

C C LEGNTH IS IN FEET
C
DATA LEGNTH / 147.0, 58.2, 63.0, 46.6,
+ 62.7, 63.8, 47.6, 56.0,
+ 46.5, 71.1, 54.8, 64.2,
+ 70.8, 75.3, 107.4, 46.3,
+ 14*0.0/

C C WEIGHT IS IN KLBS.
C
DATA WEIGHT / 453.0, 56.0, 55.1, 19.1,
+ 56.7, 42.3, 23.3, 49.3,
+ 26.1, 48.4, 21.4, 42.7,
+ 34.2, 95.0, 161.0, 11.2,
+ 14*0.0/

RETURN
END
SUBROUTINE: FOC14AP

PROGRAMMER: PHILIP J. DAY

XONTECH INC.

BBN LABORATORIES

DATE: DECEMBER 21, 1986

PURPOSE: GIVEN A APPROXIMATE ANGLE WHERE THE MAXIMUM OVER PRESS
ON THE GROUND WILL OCCUR, TRACE RAYS ON EITHER SIDE OF
ANGLE TO GET A GOOD SAMPLING OF THE OVER PRESSURES ON T
GROUND.

SUBROUTINE FOCMAP(TO,NODE,CNODE,ANG,NANG,RRCURV,CSTFLG)

REAL TO,ANG(400)
DOUBLE PRECISION RRCURV
INTEGER NODE,CNODE,NANG

COMMON /HRPOSN/ HNPTR,HCPOSN,HRT(200),HRXYZ(200,3),
+ HRAGE(200),HRPFAC(200),HRVLFT,HREMEN
REAL HRT,HRXYZ,HRAGE,HRPFAC,HRVLFT
INTEGER HNPTR,HCPOSN
LOGICAL HREMEN

COMMON /GROUND/ GLAYER,ZGRND,CGRND,UGRND,VRGRD,REFLFC
INTEGER GLAYER

COMMON /NACSPT/ HACN(41)

COMMON /CAUSTC/ NUMC,TRACE,CT(360),CPHI(360),CXYZC360,3)
REAL CXYZ
LOGICAL TRACE

COMMON /PHILIM/ PINPHI,MXPHI
REAL MINPHI,MXPHI

COMMON /HHHH/ HOMEGA,PPK(3)
REAL CPK10(2)
INTEGER NCPI,IREC,OPREC
LOGICAL SAVE

COMMON /FFTAB/ KRCAC,NPDOS,PEPDE(11),LOCSPD(10),KTABL,
+ WTAU,TAU(200),FAC(200),FLC(200)
COMMON /CFFTAB/ ACIDNT
CHARACTER*8 ACIDNT

COMMON /BASEAG/ NTTERMS,XILEAD(2),XI(500),XTAIL(502),
+ VLEAD(2),V(500),VTAIL(502)
DIMENSION XI(1004),V(1004)
EQUIVALENCE (XI(1),XILEAD(1),(V(1),VLEAD(1)))
COMMON /RAYOUT/ TTO, TPHIO, TXK(3), TOPG, CSEL

COMMON /CARL/ BOMFCT
REAL BOMFCT
REAL MAXOP

COMMON /ACIDMT/ IDENT
CHARACTER*8 IDENT

COMMON /ACWEIG/ ACWT, ACL
LOGICAL CSTFLG
CSTFLG = .FALSE.
SAVE = .TRUE.
MAXOP = 0.0
OPREC = 0

C SAVE THE ORIGINAL AIRCRAFT FLIGHT VECTOR
C CALL TACMOV(TO, NOOE, CNOOE, SAVE)
IF (NOOE.LE.O) RETURN
TTO = TO

C FIND THE ANGLE OF CAUSTIC INTERCEPTION OF THE GROUND
C CALL CSTGND(ANG, NANG, CPHIO, NCPHI)

IF (NCPHI.GT.O) THEN
CSTFLG = .TRUE.
DO 100 I = 1, NCPHI
   KK = 0
   TPHI = CPHIO(I)
   CALL GETDLT(TPHI, ANG, NANG, DELTA)
100 CONTINUE
CALL FOCUS(TO, NODE, CNODE, TPHI, RRCURV,*195,*190,*197)

TPHIO = TPHI
TTO = TO

CALL FOCAL(RRCURV, RAYOP, PMAXP, CCSEL,*195)
CSEL = CCSEL
TOPG = RAYOP
TPHIO = TPHI
IGND = 0
CONTINUE
IGND = IGND + 1
IF (HRXYZ(IGND, 3).NE.ZGRNO) GOTO 140
TXK(1) = HRXYZ(IGND, 1)
TXK(2) = HRXYZ(IGND,2)
TXK(3) = HRXYZ(IGND,3)
TTO = HRT(IGND)

FX = HACM(2)
FY = HACM(3)
FZ = HACM(4)
FFX = TXK(1)
FFY = TXK(2)
AOPG = TOPG
IFLAG = 21
TPHI = AMOD(AMOD(TPHI,360.)+540.,360.)*-180.
WRITE(12,5044) IFLAG,HACM(1),FX,FY,FZ,TPHI,TTO,
+ FFX,FFY,AOPG,CSEL,HACM(14)
GOTO 195

C  CALCULATE GROUND SOLUTION VIA THE TRAPS METHOD
C
190  CONTINUE
DO 410 K=1,NTAU
   X(K)=TAU(K)
   V(K)=FAC(K)+NRVLFT*FLC(K)
410  CONTINUE
   NTERMS=NTAU
C
C  FIND THE INDEX OF THE GROUND POINT AND SAVE LOCATION FOR OUTPUT
C
IGND = 0
150  CONTINUE
   IGND = IGND + 1
IF (HRXYZ(IGND,3).NE.ZGRND) GOTO 150
TXK(1) = HRXYZ(IGND,1)
TXK(2) = HRXYZ(IGND,2)
TXK(3) = HRXYZ(IGND,3)
IF (CXYZ(1,3).GT.ZGRND) THEN
   CALL AGING(HRAGE(MCPOSN 1))
   CALL HILBRT
END IF
   CALL AGING(HRAGE(IGND))
   CALL FNDLYR(HRXYZ(IGND,3),*555)
555  CALL AIR(HRXYZ(IGND,3))
   TPHI0 = TPHI
   TTO = TO
   CALL TSIGPT(RAYOP,IGND)
   TOPG = RAYOP
   TPHI0 = TPHI
   TTO = HRT(IGND)
   FX = HACM(2)
   FY = HACM(3)
   FZ = HACM(4)
FFX = TXX(1)
FFY = TXX(2)
AOPG = TOPG
IFLAG = 21
TPHI = AMOD(AMOD(TPHI,360.)*540.,360.)*180.
WRITE(12,5004) IFLAG,MACM(1),FX,FY,FZ,TPHI,TTO,
FFX,FFY,AOPG,CSEL,HAC4(14)
GOTO 195
199 WRITE(7,6006)
195 TPHI = TPHI + DELTA
IF (TPHI.GE.MINPHI.AND.TPHI.LE.MAXPHI) GOTO 110
197 DELTA = -1.0*DELTA
TPHI = CPHI0(I)
KK = KK + 1
IF (KK.LE.1) THEN
TPHI = TPHI + DELTA
GOTO 110
ENDIF
100 CONTINUE
ENDIF
RETURN
5001 FORMAT(12,F8.2,3F7.0,F8.2,2F7.0,F8.3,F10.4,F10.4,F10.4)
5002 FORMAT(F8.2,3F7.0,F8.2,2F7.0,F8.3,F10.4,F10.4,F10.4)
5004 FORMAT(12,1X,F10.2,3F8.0,F9.3,F10.2,2F8.0,F11.4,F10.4,F10.4)
6006 FORMAT( ' ***** FOCAL ZONE BEYOND START OF RAY *****' ,/,
+ ' ***** ABORT PROCESSING *****')
END
SUBROUTINE: TSIGPT

PROGRAMMER: PHILIP J. DAY
XONTECH INC.
BBN LABORATORIES
DATE: DECEMBER 21, 1986

PURPOSE: TO RETRIEVE PRESSURE INFORMATION AND CALL SIGPRT FOR OVERPRESSURE CALCULATIONS

SUBROUTINE TSIGPT(MAXOP,J)

REAL MAXOP
INTEGER J

COMMON /PRINTS/ TITLE(30),TIMLBL
CHARACTER*4 TITLE
CHARACTER*8 TIMLBL

COMMON /PRINTC/ KTSPGT,CVRTIM
LOGICAL CVRTIM

COMMON /BASEAG/ NTERMS,XILEAD(2),XI(500),XITAIL(502),
+ VLEAD(2),V(500),VTAIL(502)
DIMENSION XII(1004),VI(1004)
EQUIVALENCE (XII(1),XI(1)),(VI(1),VLEAD(1))

COMMON /SIGPAR/ KGMH,NRCURV,IPDNW,XMACH,VLIFT,T0,
+ PH10,SIGMA,XX(3),OMEGA,PK(3),XKS(3),XKT(3),
+ XKF(3),FACT,NAGES,AGES(20)
COMMON /SIGPAC/ IDENT,RAYNAM
CHARACTER*8 IDENT,RAYNAM
INTEGER KGMH,NRCURV,IPDNW

COMMON /HRPOSN/ HNPTR,HCPOSN,HRT(200),HRXYZ(200,3),
+ HRAGE(200),HRPFAC(200),HRVLFT,HREMEN
REAL HRT,HRXYZ,HRAGE,HRPFAC,HRVLFT
INTEGER HNPTR,HCPOSN
LOGICAL HREMEN

COMMON /GROUND/ GLAYER,2GRND,2GRDN,UGRND,VGRND,REFLFC
INTEGER GLAYER

COMMON /CAUSTC/ NUMC,TRACE,CT(360),CPhi(360),CXYZ(360,3)
REAL CXYZ
LOGICAL TRACE

COMMON /HHHH/ HOMEGA,PPK(3)

DO 100 I = 1,3
XX(I) = HRXYZ(J,1)
PK(I) = PPK(I)
100 CONTINUE

OMEGA = HOMEGA
PFAC = HRPFA(J)

CALL SIGPRT(MAXOP)

RETURN
END
SUBROUTINE CSTGND

PROGRAMMER: PHILIP J. DAY

SSW LABORATORIES

DATE: DECEMBER 12, 1986

PURPOSE: TO IDENTIFY THE PHI ANGLE(S) AT THE POINT(S) WHEN A CURVE INTERSECTS THE GROUND.

SUBROUTINE CSTGND(PHI,NPHI,CPHIO,NCPHI)

REAL PHI(400), CPHIO(2)
INTEGER NPHI, NCPHI

PARAMETER (NDIVS=20)

COMMON /GROUND/ GLAYER, ZGRND, CGRND, UGGRND, VGRND, REFLFC
INTEGER GLAYER

COMMON /PHILIM/ MINPHI, MAXPHI
REAL MINPHI, MAXPHI

COMMON /CAUSTC/ NUMC, TRACE, CT(360), CPHI(360), CXYZ(360,3)
REAL CXYZ
LOGICAL TRACE

REAL ZMIN1, ZMIN2, S(360,3), A(360,3), B(360,3), C(360,3)
REAL D(360,3)

INTEGER IMIN1, IMIN2, IZRO(2)
LOGICAL POSITV

- STATEMENT FUNCTIONS

REAL SPLN
LOGICAL OPSIGN

SPLN(DT, I, J) = A(I, J)*DT^3 + B(I, J)*DT^2 + C(I, J)*DT + D(I, J)
OPSIGN(AAA, BBB) = ((AAA*BBB).LE.0.0)

NCPHI = 0

IF (NUMC.LE.1) THEN

IF (CXYZ(1,3).GE.ZGRND-457.2.AND.
+ CXYZ(1,3).LE.ZGRND+304.8) THEN

CPHIO(1) = PHI(1)
NCPHI = 1
ENDIF
RETURN
IF (NUMC.LE.2) THEN
  CPHI0(1) = REAL(INTERP(CPHI(1),CPHI(2),CXYZ(1,3),CXYZ(2,3), ZGRND))
  IF (CPHI(1).GE.MINPHI.AND.CPHI(1).LE.MAXPHI) THEN
    NCPHI = 1
  ENDIF
ENDIF
RETURN
ENDIF

CALL SPLINE(CPHI,CXYZ,NUMC,360,S,A,B,C,D)
POSITV = .FALSE.
NCPHI = 0
ZMIN1 = 1000000.0
DO 100 I = 1,NUMC
  IF (CXYZ(I,3).LT.ZMIN1) THEN
    ZMIN1 = CXYZ(I,3)
    IMINI = I
  ENDIF
  IF (CXYZ(I,3).GT.ZGRND) POSITV = .TRUE.
100 CONTINUE

C-- CASE 1: IDENTIFIED CAUSTIC POINTS ARE ALL ABOVE THE GROUND
C
IF (ZMIN1.GT.ZGRND) THEN
  IF (IMINI.EQ.1) THEN
    IF (CPHI(1).GT.PHI(1)) THEN
      NCPHI = NCPHI + 1
      CPHI0(NCPHI) = (CPHI(1) - CPHI(2))/(CXYZ(1,3) - CXYZ(2,3)) * (ZGRND - CXYZ(1,3)) + CPHI(1)
    ELSE
      IF (CXYZ(1,3) .LT. ZMIN1) THEN
        ZMIN1 = CXYZ(1,3)
        IMINI = 1
      ENDIF
      IF (CXYZ(I,3).GT.ZGRND) POSITV = .TRUE.
    ENDIF
  ELSE
    IF (IMINI.EQ.NUMC) THEN
      IF (CPHI(NUMC).LT.PHI(NPHI)) THEN
        NCPHI = NCPHI + 1
        CPHI0(NCPHI) = PHI(1)
      ENDIF
    ELSE
      IF (ZMIN1.GT.ZGRND+304.8) THEN
        RETURN
      ELSE
        NCPHI = NCPHI + 1
        CPHI0(NCPHI) = PHI(1)
      ENDIF
    ENDIF
  ELSE
    IF (IMINI.EQ.NUMC) THEN
      IF (CPHI(NUMC).LT.PHI(NPHI)) THEN
        NCPHI = NCPHI + 1
        CPHI0(NCPHI) = (CPHI(NUMC) - CPHI(NUMC-1))/(CXYZ(NUMC,3) - CXYZ(NUMC-1,3)) * (ZGRND - CXYZ(NUMC,3)) + CPHI(NUMC)
      ELSE
        IF (ZMIN1.GT.ZGRND+304.8) THEN
          RETURN
        ELSE
          NCPHI = NCPHI + 1
          CPHI0(NCPHI) = PHI(NPHI)
        ENDIF
      ENDIF
    ELSE
      IF (ZMIN1.GT.ZGRND+304.8) THEN
        RETURN
      ELSE
        NCPHI = NCPHI + 1
        CPHI0(NCPHI) = PHI(NPHI)
      ENDIF
    ENDIF
END
RETURN
ELSE
   NCPHI = NCPHI + 1
   CPHIO(NCPHI) = PHI(NPHI)
ENDIF
ENDIF
ELSE
   C
   PHI ANGLE IS NOT AT THE END OF THE ARRAY SO WE CAN USE A CUBIC
   SPLINE FOR INTERPOLATION.
   C
   IF (CXYZ(IMIN1-1,3).LT.CXYZ(IMIN1+1,3)) THEN
      IMIN2 = IMIN1
      IMIN1 = IMIN1 - 1
   ELSE
      IMIN2 = IMIN1 + 1
   ENDIF
   DPHI = ABS(PHI(IMIN2) - PHI(IMIN1))/REAL(NDIVS)
   ZMIN = ZMIN1
   TPHIO = CPHI(IMIN1)
   DO 200 I = 1,NDIVS
      DT = DPHI * REAL(I)
      ZMIN2 = SPLN(DT,IMIN1,3)
      IF (ZMIN2.LT.ZMIN) THEN
         ZMIN = ZMIN2
         TPHIO = CPHI(IMIN1) + DT
      ENDIF
   200 CONTINUE
   IF (ZMIN.LE.ZGRND+304.8) THEN
      IF (ZMIN.GE.ZGRND) THEN
         NCPHI = NCPHI + 1
         CPHIO(NCPHI) = TPHIO
      ELSE
         C- WE HAVE THE CURVE INTERSECTING THE GROUND IN TWO PLACES.
         C- WE CHECK TO SEE IF THE LOWEST POINT ON THE CURVE IS BELOW
         C- OR ABOVE GROUND-1000 FT.
         C- IF IT IS BELOW THEN FIND THE ZERO ON EITHER SIDE USING THE
         C- BISECTION METHOD OF ROOT FINDING
         C
         IF (ZMIN.LT.ZGRND-457.2) THEN
            DO 250 I = 1,2
            IF (I.EQ.1) THEN
               PHIA = CPHI(IMIN1)
               PHIB = TPHIO
            ELSE
               PHIA = TPHIO
               PHIB = CPHI(IMIN2)
            ENDIF
            250 CONTINUE
      ENDIF
      ELSE
ENDIF
PA = PHIA
PB = PHIB
PC = (PA+PB)/2.0
ERR = ABS(PA-PB)/(2000.0*REAL(NDIVS))

255 CONTINUE
DPB = PB - CPHI(IMIN1)
DPC = PC - CPHI(IMIN1)
SP1 = SPLH(DPB,IMIN1,3) - ZGRND
SP2 = SPLH(DPC,IMIN1,3) - ZGRND
IF (OPSIGN(SP1,SP2)) THEN
  PA = PC
ELSE
  PB = PC
ENDIF
PC = (PA+PB)/2.0
IF ((PB-PC).GT.ERR) GOTO 255

CPHIO(I) = PC

250 CONTINUE
NCPHI = 2
ELSE
  MIN Z IS ABOVE GROUND-1000 FT. SET MCPHI NEGATIVE AS A FLAG
  NCPHI = NCPHI + 1
  CPHIO(NCPHI) = TPHIO
  NCPHI = -1*NCPHI
ENDIF
ENDIF
ELSE
  RETURN
ENDIF
ENDIF
RETURN

C: CASE 2: CAUSTIC SURFACE IS BELOW THE GROUND

C
IF (.NOT.POSITV) THEN
  IF (CPHI(1).EQ.PHI(1)) THEN
    NCPHI = NCPHI + 1
    CPHIO(NCPHI) = CPHI(1)
  ENDIF
  IF (CPHI(NUMC).EQ.PHI(NPHI)) THEN
    NCPHI = NCPHI + 1
    CPHIO(NCPHI) = CPHI(NUMC)
  ENDIF
ENDIF
ELSE
  RETURN
ENDIF
C: CASE 2: CAUSTIC SURFACE IS BELOW THE GROUND

C
IF (.NOT.POSITV) THEN
  IF (CPHI(1).EQ.PHI(1)) THEN
    NCPHI = NCPHI + 1
    CPHIO(NCPHI) = CPHI(1)
  ENDIF
  IF (CPHI(NUMC).EQ.PHI(NPHI)) THEN
    NCPHI = NCPHI + 1
    CPHIO(NCPHI) = CPHI(NUMC)
  ENDIF
ENDIF
ELSE
  RETURN
ENDIF
ENDIF

RETURN
ENDIF

C--------
C
C CASE 3: CAUSTIC SURFACE INTERCECTS THE GROUND
C
DO 300 I = 1,NUMC-1
   IF (CPSIGN(CXYZ(I,3)-ZGRND,CXYZ(I+1,3)-ZGRND)) THEN
      NCPHI = NCPHI + 1
      IZRO(NCPHI) = 1
   ENDIF
300 CONTINUE
C
C CHECK TO SEE IF CAUSTIC SURFACE IS ABOVE GROUND-1000 FT.
C
IF (IMIN1.EQ.NUMC) THEN
   IMIN2 = IMIN1
   IMIN1 = IMIN1 - 1
ELSE
   IF (IMIN1.EQ.1) THEN
      IMIN2 = 2
   ELSE
      IF (CXYZ(IMIN1-1,3).LT.CXYZ(IMIN1+1,3)) THEN
         IMIN2 = IMIN1
         IMIN1 = IMIN1 - 1
      ELSE
         IMIN2 = IMIN1 + 1
      ENDIF
   ENDIF
ENDIF

DCPHI = ABS(PHI(IMIN1) - PHI(IMIN2))/REAL(NDIVS)
ZMIN = ZMIN1

DO 500 I = 1,NDIVS
   DT = DPHI * REAL(I)
   ZMIN2 = SPLN(DT,IMIN1,3)
   IF (ZMIN2.LT.ZMIN) THEN
      ZMIN = ZMIN2
      TPHIO = CPHI(IMIN2) + DT
   ENDIF
500 CONTINUE
IF (ZMIN.GE.ZGRND-457.2) THEN
   IF ((ABS(CPHI(IMIN1) - CPHI(IMIN2))) .LE. (ABS(PHI(2) - PHI(3)))) NCPHI = 1
ENDIF

C USE THE BISECTION METHOD OF ROOT FINDING TO FIND THE ZERO
C
DO 400 I = 1,NCPHI
PA = CPHI(I)ZRO(I))
PB = CPHI(IZRO(I)+1)
PC = (PA+PB)/2.0
ERR = ABS(PA-PB)/(2000.0*REAL(NDIVS))

410 CONTINUE
  DPB = PB - CPHI(I)ZRO(I))
  DPC = PC - CPHI(I)ZRO(I))
  SP1 = SPLN(DPB,I)ZRO(I),3) - ZGRND
  SP2 = SPLN(DPC,I)ZRO(I),3) - ZGRND
  IF (OPSIGN(SP1,SP2)) THEN
    PA = PC
  ELSE
    PB = PC
  ENDIF
  PC = (PA+PB)/2.0

  IF ((PB-PC).GT.ERR) GOTO 410
  CPHI0(I) = PC

400 CONTINUE

RETURN

END
SUBROUTINE: FOCUS
PROGRAMMER: PHILIP J. DAY
XONTECH INC.
BBN LABORATORIES
DATE: DECEMBER 16, 1986

PURPOSE: TO LOCATE A CAUSTIC SURFACE AND ITS RELATIVE CURVATURE
AT THE RAY LOCATION.

PARAMETERS:

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DISCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>NODE</td>
<td>I</td>
<td>INDEX INTO THE FLIGHT TRACK AR</td>
</tr>
<tr>
<td>CNOE</td>
<td>I</td>
<td>INDEX INTO THE SPLINE</td>
</tr>
<tr>
<td>ANG</td>
<td>R</td>
<td>PHI ANGLE OF THE ORIGINAL RAY</td>
</tr>
</tbody>
</table>

OUTPUT:

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DISCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>RRCURV</td>
<td>R</td>
<td>RADIUS OF CURVATURE OF THE CAUSTIC SURFACE</td>
</tr>
<tr>
<td>ALT RETURN 1</td>
<td>*</td>
<td>TAKEN IF ONE OF THE RAYS RECURS ABOVE THE GROUND</td>
</tr>
<tr>
<td>ALT RETURN 2</td>
<td>*</td>
<td>TAKEN IF ONE OF THE AUXILIARY RAYS HAS NO CAUSTIC</td>
</tr>
<tr>
<td>ALT RETURN 3</td>
<td>*</td>
<td>TAKEN IF THE ORIGINAL RAY HAS A CAUSTIC, OR THE CAUSTIC IS MORE THAN 500 FT. ABOVE THE GROUND</td>
</tr>
</tbody>
</table>

SUBROUTINE FOCUS(TO, NODE, CNOE, ANG, RRCURV, *, *, *)

REAL TO, ANG
DOUBLE PRECISION RRCURV
INTEGER NODE, CNOE
COMMON /RPOSN/ NPTR,CPOSN,RT(200),RXYZ(200,3),RAGE(200), + RPFACT(200),RVLIFT,REMEM
REAL RT,RXYZ,RAGE,RPFACT,RVLIFT
INTEGER NPTR,CPOSN
LOGICAL REMEM

COMMON /AC'.EIG/ AC .T,ACL

DOUBLE PRECISION RRAY1(3),RRAY2(3),RRAY3(3),RRR(3),RRRR
DOUBLE PRECISION RCURV(3),CCURV(3),RELCUR(3)
DOUBLE PRECISION XXX
DOUBLE PRECISION DDOTP,DRNORM

REAL DELR1(3),DELR2(3),VELO,DELV
REAL V(3),A(3),WHAT(3),AHAT(3),WHAT(3),MACH,MACHP,MUDDOT
REAL ALPDOT,EDOT,DLTAT,TTO,PNIO,TT(3),THETAS,TE4P(3),PI
REAL DL(7)

LOGICAL SAVE,NOSAVE,RCBLG,CFLAG

EQUIVALENCE (V,XDOT),(A,XDDOT)

DATA SAVE/.TRUE./,NOSAVE/.FALSE./,DOLPHI/.5/,DLTAS/243.84/
DATA P1/3.14159263/,KKTR/O/,KKTB/O/

NODE = NNODE
CNODE = WCNODE
CALL TACMOV(TO,NODE,CHNODE,NOSAVE)
IF (NODE.LE.0) RETURN !

NPTR = 0
NUMC = 0
TRACE = .TRUE.
REMEM = .TRUE.

VELO = RNORM(V,3)
PHIO = ANG
IF (ANG.NE.0.0) THEN
   DLPHI = DDLPHI*ANG/ABS(ANG)
ELSE
   DLPHI = DDLPHI
ENDIF
CALL RAYORG(*5001)
CALL RAYTRK(.FALSE..RCBLG,CFLAG,*5001)
CALL SAVRAY

C
C- NO CAUSTIC ON THE RAY
C
IF (NUMC.LE.0) THEN
   KKT8 = KKT8 + 1
   IF (KKT8.GT.1) THEN
      KKT8 = 0
      RETURN 3
   ELSE
      KKT8 = KKT8 + 1
      IF (KKT8.GT.1) THEN
         KKT8 = 0
         RETURN 3
      ELSE
         CALL SAVRAY
         IF (NUMC.LE.0) THEN
            KKT8 = KKT8 + 1
            IF (KKT8.GT.1) THEN
               KKT8 = 0
               RETURN 3
            ELSE
               CALL SAVRAY
         ELSE
            IF (KKT8.GT.1) THEN
               KKT8 = 0
               RETURN 3
            ELSE
               CALL SAVRAY
         ELSE
            CALL SAVRAY
         ELSE
            CALL SAVRAY
         ELSE
            CALL SAVRAY
         ELSE
            CALL SAVRAY
      ELSE
         CALL SAVRAY
      ELSE
         CALL SAVRAY
   ELSE
      CALL SAVRAY
   ELSE
      CALL SAVRAY
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
END
RETURN 2
ENDIF
ENDIF
KKTB = 0

C- CAUSTIC 500 FT. OR MORE ABOVE THE GROUND
C
IF (RXYZ(CPOSN,3).GT.ZGRND+304.8) THEN
  KKTR = KKTR + 1
IF (KKTR.GT.1) THEN
  KKTR = 0
  RETURN 3
ELSE
  RETURN 2
ENDIF
ENDIF

C- CALCULATE RAY CURVATURE
C
IF (CPOSN.GE.5) THEN
  J1 = 2
ELSE
  J1 = 1
ENDIF
J2 = 2*J1

DO 100 I = 1,3
  RRAY1(I) = DBLE(RXYZ(CPOSN,I))
  RRAY2(I) = DBLE(RXYZ(CPOSN-J1,I))
  RRAY3(I) = DBLE(RXYZ(CPOSN-J2,I))
100 CONTINUE

CALL CURVE(RRAY1,RRAY2,RRAY3,RRR,RRRR)

DO 200 I = 1,3
  RCURV(I) = -1.0*(DBLE(CXYZ(1,1)) - RRR(I))/(RRRR**2)
200 CONTINUE

C- TRACE THE RAYS FOR THE PHIINC CALCULATIONS
C
C- RAY 2
C
1000 PHIO = ANG + DLPHI
CALL RAYORG(*5001)
CALL RAYTRK(*FALSE., RCBLG, CFLAG, *5001)
IF (ABS(CT(2)-CT(1)).GT.0.5) RETURN 1
CALL FINDT(2,*1000)

MACH = SQRT(DOTP(V,V,3)/(CD**2))
MACHP = DOTP(A,V,3)/(CD*RNRM(V,3))
MUDOT = -1.0*MACHP/(MACH*SQRT(MACH*MACH - 1.0))

CALL UNIT(V,VHAT,3)
CALL UNIT(A,AHAT,3)
CALL CROSS(VHAT,AHAT,TEMP)
CALL CROSS(VHAT,TEMP,WHAT)

THETAS = ASIN(-1.0*WHAT(3))
EDOT = DOTP(A,WHAT,3)/RNORM(V,3)*COS((ANG*PI/180.0)-THETAS)

C DLAT = ABS(DLPHI/((MUDOT + EDOT)**2.0/PI)**2)
DLAT = ABS(DLPHI/PI**2.0)*4.6*ACL/CO/SRT(MACH*MACH + 1.0))

TTO = TO + DLAT

NODE = MNODE
CNODE = NCNODE
CALL TAMOV(TTO,NODE,CNODE,NOSAVE)
IF (NODE.LE.0) RETURN

C RAY 3
C
PHIO = ANG
CALL RAYORG(*5001)
CALL RAYTRK(.FALSE.,RCBLG,CFLAG,*5001)
IF (ABS(CT(3)-CT(1)).GT.0.5) RETURN 1
CALL FINOT(3,*1000)

C RAY 4
C
PHIO = ANG + DLPHI
CALL RAYORG(*5001)
CALL RAYTRK(.FALSE.,RCBLG,CFLAG,*5001)
IF (ABS(CT(4)-CT(1)).GT.0.5) RETURN 1
CALL FINOT(4,*1000)

C CALCULATE PHIINC
C
DO 300 1 = 1,3
DELR1(I) = CXYZ(3,I) - CXYZ(1,I)
DELR2(I) = CXYZ(4,I) - CXYZ(3,I)
300 CONTINUE

DO 302 1 = 1,6
DL(I) = 0.0
302 CONTINUE

DO 301 1 = 1,3
DL(1) = (CXYZ(2,1) - CXYZ(1,1))**2 + DL(1)
DL(2) = (CXYZ(3,1) - CXYZ(1,1))**2 + DL(2)
DL(3) = (CXYZ(4,1) - CXYZ(1,1))**2 + DL(3)
DL(4) = (CXYZ(3,1) - CXYZ(2,1))**2 + DL(4)
DL(5) = (CXYZ(4,1) - CXYZ(2,1))**2 + DL(5)
DL(6) = (CXYZ(4,1) - CXYZ(3,1))**2 + DL(6)

301 CONTINUE
301 CONTINUE

\[
\text{PHIINC} = -1.0*(\text{DOTP}(\text{DELR1}, \text{DELR2}, 3) \times \text{DLPHI}) + \frac{\text{RNORM}(\text{DELR2}, 3)^2}{(\text{VELO} \times \text{DLTAT})}
\]

\[
\text{TT}(1) = \text{TO} - \text{DLTAS}/(\text{MACH} \times \text{CD})
\]
\[
\text{TT}(2) = \text{TO} + \text{DLTAS}/(\text{MACH} \times \text{CD})
\]
\[
\text{TT}(3) = \text{TO} - 2.0 \times \text{DLTAS}/(\text{MACH} \times \text{CD})
\]

\text{NUMC} = 1

\text{CALL FINOT}(1, *5001)

\text{DO 400 } I = 1, 3

\text{NODE} = \text{NNODE}

\text{CNODE} = \text{CNODE}

\text{CALL TACMOV(} \text{TT}(I), \text{NODE}, \text{CNODE}, \text{MOSAVE})

\text{IF (} \text{NODE} \leq 0 \text{) RETURN}

\text{DELV} = (\text{VELO} + \text{RNORM}(V, 3))/2.0

\text{PHIO = ANG + PHIINC*REAL(DELV)} * (\text{TT}(I) - \text{TO})

\text{TRACE} = .FALSE.

\text{CALL RAYORG(*400)}

\text{CALL RAYTRK(.TRUE., RCBLG, CFLAG, *400)}

400 CONTINUE

C

\text{IF ONE OF THE AUXILIARY RAY TUBES HAS NO CAUSTIC. USE TRAPS}

\text{SIGNATURE CALCULATIONS}

\text{IF (NUMC.LT.4) THEN}

\text{RETURN 1}

\text{ENDIF}

C

\text{CHECK FOR CUSP STRADLE}

\text{IF ((CT(4).GE.CT(2)).OR.(CT(2).GE.CT(1)).OR.(CT(1).GE.CT(3)))}

\text{THEN}

\text{RETURN 1}

\text{ENDIF}

C

\text{CALCULATE RELATIVE CURVATURE OF THE CAUSTIC SURFACE}

\text{DO 500 } I = 1, 3

\text{RRAY1(I) = CXYZ(1, I)}

\text{RRAY2(I) = CXYZ(2, I)}

\text{RRAY3(I) = CXYZ(3, I)}

500 CONTINUE

\text{CALL CURVE(} \text{RRAY1, RRAY2, RRAY3, RRR, RRRR})

\text{DO 600 } I = 1, 3

\text{CCURV(I)} = -1.0D0*(DBLE(} \text{CXYZ(1, I)}) - \text{RRR(I))}/(\text{RRRR}^2)

600 CONTINUE

\text{XXX} = 1.0D0 - \text{DOTP(} \text{RCURV, CCURV, 3})/(\text{DRNORM(} \text{CCURV, 3})^2)
DO 700 I = 1,3
   RELCUR(I) = XXX*CCURV(I)
700 CONTINUE

RRCURV = 1.000/DRNORM(RELCUR,3)

RETURN
5000 RETURN

C  ONE OF THE RAYS RECURVES BEFORE IT REACHES THE GROUND. WE CAN NOT
C  PROCESS THIS CASE
C
5001 RETURN 1
END
SUBROUTINE TACMOV

PROGRAMMER: PHILIP J. DAY

XONTECH INC.

BBN LABORATORIES

DATE: DECEMBER 16, 1986

PURPOSE: TO CALL ACMOVE AND TO SAVE (OR NOT) THE STATE VECTOR AS REQUIRED.

PARAMETERS:

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>INPUT:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TO</td>
<td>R</td>
<td>TIME USED TO COMPUTE THE AIRC Position</td>
</tr>
<tr>
<td>NODE</td>
<td>I</td>
<td>INDEX INTO THE FLIGHT TRACK AR</td>
</tr>
<tr>
<td>CNOE</td>
<td>I</td>
<td>INDEX INTO THE SPLINE</td>
</tr>
<tr>
<td>SAVE</td>
<td>L</td>
<td>FLAG FOR SAVING THE STATE VECT</td>
</tr>
</tbody>
</table>

OUTPUT: NONE

SUBROUTINE TACMOV(TO,NODE,CNOE,SAVE)

REAL TO
INTEGER NODE,CNOE
LOGICAL SAVE

COMMON /ACSPOT/ TIME,XRO,YRO,ZRO,XDOT,YDOT,ZDOT,AIRSPD,ASPDOT,
+ CO,UO,VO,CDOT,XMACH,XMADOT,XMU,XMUOOG,CSMU,
+ SIMU,EK(3,3),EKDOT(3,3),GLOAD,HEADIN,CLIMB,BANK,
+ XDDOT,YDOT,ZDDOT,XDDDOT,YDODOT,ZDDDOT

COMMON /HACSPT/ HACM

REAL ACM(41),HACM(41)

EQUIVALENCE (ACM(1),TIME)

CALL ACMOVE(TO,NODE,CNOE)

IF (SAVE) THEN
    DO 100 I = 1,41
        HACM(I) = ACM(I)
    100 CONTINUE
ENDIF

RETURN
END
SUBROUTINE SAVRAY
COMMON /RPOSN/ WPTR,CPOSN,RT(200),RXYZ(200,3),RAGE(200),
+ RPFAC(200),RVLF,REM
REAL RT,RXYZ,RAGE,RPFAC,RVLFT
INTEGER WPTR,CPOSN
LOGICAL REM

COMMON /HRPOSN/ HNPTR,HCPOSN,HRT(200),HRXYZ(200,3),
+ HRPFAC(200),HRVLFT,HREM
REAL HRT,HXYZ,HAGE,HRPFAC,HRVLFT
INTEGER HNPTR,HCPOSN
LOGICAL HREM

COMMON /RAYNIT/ KGMH,NDCRVS,NUCRVS,IUPDOWN,T0,PHIO,XO,Y0,Z0,
+ P10,P20,P30,OMEGA,DETAO,P1FO,P2FO,P3FO,
+ OMEGAF,XTO,YTO,ZTO,P1TO,P2TO,P3TO,OMEGAT,XSO,
+ YSO,ZSO,P3SO,RH00,PCONST,NAGES,AGE(20)
INTEGER KGMH,NDCRVS,NUCRVS,IUPDOWN
REAL XX0(3),PK0(3),PKFO(3),XKTO(3),PETO(3),P3SO(3)
REAL PPJ(3)
EQUIVALENCE (PPJ(1),P10)

COMMON /HHHH/ HOMGA,PPK(3)
HOMGA = OMEGA
HNPTR = WPTR
HCPOSN = CPOSN
HRVLFT = RVLF
HREM = REM

DO 100 I = 1,WPTR
   HAGE(I) = RAGE(I)
   HRPFAC(I) = RPFAC(I)
   HRT(I) = RT(I)
   DO 110 J = 1,3
      HRXYZ(I,J) = RXYZ(I,J)
110   CONTINUE
100 CONTINUE
DO 200 I = 1,3
   PPK(I) = P PJ(I)
200 CONTINUE
RETURN
END
SUBROUTINE FOCAL

PROGRAMMER: PHILIP J. DAY
XONTECH INC.

DATE: DECEMBER 18, 1986

PURPOSE: TO COMPUTE THE OVER PRESSURE AT THE GROUND WHEN A CAUSTIC BETWEEN 500 FT AND 1000 FT ALTITUDE.

SUBROUTINE FOCAL(RRCURV,MAXOP, PAXP, CSEL, *)

DOUBLE PRECISION RRCURV
REAL MAXOP

COMMON /GROUND/ GLAYER,ZGRND,CGRN0,UGRND,REFL/C
INTEGER GLAYER

COMMON /FFTAB/ KRCAC,NSPOS,SPEEDS(11),LOCSPD(10),KTABLE,
+ TAU,TAU(200),FAC(200),FLC(200)
COMMON /FFTAB/ ACIDNT
CHARACTER*8 ACIDNT

COMMON /BASEAG/ NTERMS,XILEAD(2),XI(500),XITAIL(502),
+ VLEAD(2),V(500),VTAIL(502)
DIMENSION XI1(1004),VI1(1004)
EQUIVALENCE (XI1(l),XILEAD(l)),(VI1(l),VLEAD(l))

COMMON /PRESUR/ PRS,C
REAL PRS,C

COMMON /CAUSTC/ NUMC,TRACE,CT(360),CPHI(360),CXYZ(360,3)
REAL CXYZ
LOGICAL TRACE

COMMON /RPOSN/ NPTR,COPSN,RT(200),RXYZ(200,3),RAGE(200),
+ RPFACT(200),RVLFIT,REMEM
REAL RT,RXYZ,RAGE,RPFACT,RVLIFT
INTEGER NPTR,COPSN
LOGICAL REMEM

COMMON /HRPOSN/ HNPTR,HCPSN,HRT(200),HRXYZ(200,3),
+ HRAGE(200),HRPFAC(200),HRVLIFT,HRREMEM
REAL HRT,HRXYZ,HRAGE,HRPFAC,HRVLIFT
INTEGER HNPTR,HCPSN
LOGICAL HRREMEM

COMMON /HHHH/ HOMEGA,PK(3)
REAL  INTERP,:IST,MAXOPG,MAXOP1
REAL  XXT(504),PPT(504),PMX,CSEL

C- INITIALIZE THE F-FUNCTION
C
DO 200 K=1,NTAU
  XI(K)=TAU(K)
  V(K)=FAC(K)+HRVLFT*FLC(K)
200 CONTINUE
  NTERMS=NTAU
C
C- FIND THE GROUND POSITION
C
  IGNO = 0
100 CONTINUE
    IGNO = IGNO + 1
    IF (HRXYZ(IGNO,3).NE.ZGRNO) GOTO 100
C
C- CALCULATE THE OVERPRESSURE AND CP ON THE GROUND USING THE TRAPS
C- METHOD
C
  IF (CXYZ(1,3).GT.ZGRND) THEN
    CALL AGING(HRAGE(HCPOSN-1))
    CALL HILBRT
    ENDIF
    CALL AGING(HRAGE(IGNO))
    CALL FNDLYR(HRXYZ(IGNO,3),*556)
    CALL AIR(HRXYZ(IGNO,3))

  PMAX = 0.0
  PMIN = 100.0
  DO 220 K=1,NTERMS
    V(K)=V(K)*HRPFAC(IGNO)/2.0
    PMAX=AMAX1(PMAX,V(K))
    PMIN=AMIN1(PMIN,V(K))
 220 CONTINUE

  MAXOPG = AMAX1(ABS(PMIN),ABS(PMAX))
  DDPP1 = 0.0
  DO 300 K = 1,NTERMS+1
  C    IF (XI(K+1).GE.XI(K+2).AND.VI(K+1).LT.VI(K+2)) THEN
  C      DDPP1 = AMAX1(DDPP1,(VI(K+2)-VI(K+1)))
  C  ENDIF
 300 CONTINUE
  IF (DDPP1.EQ.0.0) DDPP1 = MAXOPG

  CPGND=DDPP1/1.40/PRS*2.0*0.001
C
C- CALCULATE THE FOCAL ZONE AND THE FOCUS OVERPRESSURE
C
  I = HCPOSN
CONTINUE

I = I + 1
IF (1.0.E.0) RETURN 1

SSS = DIST(HCPSOIN, I, HRXYZ)

DO 410 K = 1, NTAU
    XI(K) = TAU(K)
    V(K) = FAC(K) * HRVLT * FLC(K)

410 CONTINUE

NTERMS = NTAU

CALL AGING(HRAGE(I))
CALL FNOLYR(HRXYZ(1, N), 557)

557 CALL AIR(HRXYZ(1, 3))

PMAX = 0.0
PMIN = 100.0

DO 420 K = 1, NTERMS
    V(K) = V(K) * HRPFAC(I) / 2.0
    PMAX = AMAX1(PMAX, V(K))
    PMIN = AMIN1(PMIN, V(K))

420 CONTINUE

MAXOP1 = AMAX1(ABS(PMIN), ABS(PMAX))
DDPP1 = 0
IF (DDPP1.EQ.0.0) DDPP1 = MAXOP1

CPREF = DDPP1 / 1.40 / PRS = 2.0 * 0.001
YREF = SSS / SSS / REAL(RRCURV) / 2.
YSTAR = (0.1 ** (4. / 5.) / 0.39) ** 4 * (0.60 * CPREF * YREF ** 0.25 * 
REAL(RRCURV)) ** (4. / 5.)
XSTAR = SQRT(2.0 * REAL(RRCURV) * YSTAR)
IF (SSS.LT.XSTAR) GOTO 400

PFOCUS = PMAX * SQRT(SSS / XSTAR)

PMA XP = PFOCUS

PSIG = 0.05 * (PMA XP - PMIN)
KMIN = NTERMS
KMAX = 1

DO 810 III = 1, NTERMS
    IF (ABS(V(III)).GE.PSIG) THEN
        KMIN = MIN(KMIN, III)
        KMAX = MAX(KMAX, III + 2)
    ENDIF
810 CONTINUE

NX = KMAX - KMIN + 1
K = 0
XSHFT = 0.0
DO 800 III = KMIN,KMAX
   K = K + 1
   XXT(K) = (XII(III+1)-XII(KMIN+1)+XSHFT)/0.3048
   PPT(K) = VI(III+1)/47.8803
800 CONTINUE
   SREF = SSS/0.3048
   RRC = RRCURV/0.3048
   CCT = C/0.3048
C
C - PRS in mb to psf
C
   PRST = PRS/0.478803E-01
C
   NSIG = NX
C
C - SELECT THE PROPER OVERPRESSURE TO USE
C
   IF (HRXYZ(1,3).LE.ZGRND) THEN
      MAXOP = MAXOPG
   ELSE
      IF (CXYZ(1,3).GT.ZGRND) THEN
         IF (CPGND.GT.CPREF) THEN
            CALL FOCALP(XXT,PPT,NXSREFRRC,PRST,CCT,PMX,NSIG)
            MAXOP = PMX*47.8803
         ELSE
            MAXOP = MAXOPG
         ENDIF
      ELSE
         CALL FOCALP(XXT,PPT,NX,SREF,RRC,PRST,CCT,PMX,NSIG)
         MAXOP = PMX*47.8803
      ENDIF
   ENDIF
   MAXOP = MAXOP*2.0
C
C - CONVERT BACK TO METRIC
C
   XSHFT = XXT(1)
   DO 850 III = 1,NSIG
      XXT(III) = (XXT(III) - XSHFT)*0.3048*1000./C
      PPT(III) = PPT(III)*47.8803
850 CONTINUE
C
C - CALCULATE THE CSEL
C
   CALL PRTSNG(XXT,PPT,NSIG,C)
   CALL CALSEL(PPT,XXT,NSIG,CSEL)
9000 FORMAT(1H1)
RETURN
END
C
C---------------------------------------------------------------------
C
C FUNCTION: DIST
C PROGRAMMER: PHILIP J. DAY
C XONTECH INC.
C BBN LABORATORIES
C DATE: DECEMBER 18, 1986
C
C PURPOSE: TO COMPUTE THE DISTANCE BETWEEN TWO POINTS IN SPACE
C
C
FUNCTION DIST(I,J,RAY)

REAL RAY(200,3),TMP
INTEGER I,J

TMP = 0.0

DO 100 K = 1,3
   TMP = TMP + (RAY(I,K) - RAY(J,K))**2
100 CONTINUE

DIST = SQRT(TMP)

RETURN

END
FUNCTION: INTERP
PROGRAMMER: PHILIP J. DAY
BBN LABORATORIES
DATE: DECEMBER 18, 1986

PURPOSE: TO DO A LINEAR INTERPOLATION BETWEEN TWO POINTS

FUNCTION INTERP(P1,P2,Z1,Z2,ZG)

REAL P1,P2,Z1,Z2,ZG

INTERP = (P1 - P2)/(Z1 - Z2)*(ZG-Z1) + P1

RETURN
END
SUBROUTINE: CURVE

PROGRAMMER: TAKEN FROM THE FOSOOM ROUTINES

SUBROUTINE CURVE (X1, X2, X3, RRR, RRRR)

THIS SUBROUTINE FITS A CIRCLE THROUGH POINTS X123, AND RETURNS THE
RADIUS VECTOR AND CURVATURE AS RRR(3) AND RRRR.

IMPLICIT DOUBLE PRECISION (A-H,O-Z)
DOUBLE PRECISION X1(3), X2(3), X3(3), RVC(3), DELR1(3), DELR2(3)
1, DELR3(3), DETR, DETRX, DETRY, DETRZ, RRR(3), RRRR, X1J(3), Y1J(3)
2, Z1J(3)

T(ITE(7,*)'CURVE 1'
WRITE(7,*)'X1,X2,X3=,X1,X2,X3
XIJ(1)=X1(1)-X3(1)
YIJ(1)=X1(2)-X3(2)
ZIJ(1)=X1(3)-X3(3)
XIJ(2)=X2(1)-X3(1)
YIJ(2)=X2(2)-X3(2)
ZIJ(2)=X2(3)-X3(3)
RVC(1)=0.0
RVC(2)=0.0
DO 6002 J=1,3
RVC(J)=(X1(J)**2+X3(J)**2)**0.5 RVC(1)
RVC(2)=(X2(J)**2+X3(J)**2)**0.5 RVC(2)
6002 CONTINUE

WRITE(7,*)'CURVE 2'
DO 6003 I=1,3
DELR1(I)=X3(I)-X1(I)
DELR2(I)=X2(I)-X1(I)
XIJ(3)=DELR1(2)*DELR2(3)-DELR1(3)*DELR2(2)
YIJ(3)=DELR1(3)*DELR2(1)-DELR1(1)*DELR2(3)
ZIJ(3)=DELR1(1)*DELR2(2)-DELR1(2)*DELR2(1)
RVC(3)=X1J(1)*X1J(3)+X1J(2)*X1J(3)+X1J(3)*X1J(2)
DETR=X1J(1)*Y1J(2)*Z1J(3)-Y1J(3)*Z1J(2)+Y1J(1)*X1J(3)*Z1J(2)
1X1J(2)*Z1J(3)+Z1J(1)*X1J(2)*Y1J(3)-Y1J(2)*X1J(3))
DETRX=RVC(1)*Y1J(2)*Z1J(3)-Y1J(3)*RVC(2)+Y1J(1)*RVC(3)*Z1J(2)
1RVC(2)*Z1J(3)+Z1J(1)*RVC(2)*Y1J(3)-Y1J(2)*RVC(3))
DETRY=X1J(1)*RVC(2)*Z1J(3)-RVC(3)*Z1J(2)+RVC(1)*X1J(3)*Z1J(2)
1X1J(2)*Z1J(3)+Z1J(1)*X1J(2)*RVC(3)-RVC(2)*X1J(3))
DETRZ=X1J(1)*Y1J(2)*RVC(3)-Y1J(3)*RVC(2)+Y1J(1)*X1J(3)*RVC(2)
1X1J(2)*RVC(3)+RVC(1)*X1J(2)*Y1J(3)-Y1J(2)*X1J(3))

WE NOW HAVE THE CENTER RRR AND RADIUS RRRR OF THE FITTED CIRCLE:

WRITE(7,*)'DETR,DETRX,DETRY,DETRZ=IDETR,DETRX,DETRY,DETRZ
RRR(1)=DETRX/DETR
RRR(2)=DETRY/DETR
RRR(3)=DETRZ/DETR

WRITE(7,*)'CURVE 3'

NOTE THAT X1 OR X3 WOULD WORK EQUALLY WELL IN THE FOLLOWING LINE
RRRR=SQRT((RRR(1)-X2(1))**2+(RRR(2)-X2(2))**2+(RRR(3)-
1X2(3))**2)
C     WRITE(7,'CURVE 4')
RETURN
END
SUBROUTINE FOCALP(XX,PP,NX,SREF,R,PO,AO,PMAXNSIG)

C

C

C THIS SUBROUTINE APPLIES GILL AND SEEBASS' FOCUSED SHOCK WAVE SOLUTION TO EACH SHOCK IN A SONIC BOOM AT A CAUSTIC. IT FIRST CONVERTS THE INPUT SIGNATURE FROM VALUES PP AT ARBITRARILY SPACED XX TO 100 EVENLY SPACED POINTS. PRESSURE POSITIONS ARE DODDLED SLIGHTLY, BUT NOT MORE THAN HALF OF ONE PERCENT OF THE SIGNATURE LENGTH. THE FOCUS SOLUTION IS APPLIED TO EACH SHOCK. THE PUBLISHED SIGNATURE IS LINEARLY EXTRAPOLATED 'TIL ZERO, SPACE PERMITTING. SPACE NOT PERMITTING, FOCUS SOLUTIONS ARE CARRIED OUT TO THE MIDPOINTS BETWEEN SUCCESSIVE SHOCKS. CURRENTLY, A MESSAGE IS PRINTED OUT IDENTIFYING SUCH COMPUTATIONAL DISCONTINUITIES; SOME KIND OF SMOOTHING MUST BE ADDED.

REAL XX(504), PP(504)
DIMENSION IDP(50), PSH(50), FPOS(21), FNEG(21)
DATA FNEG/2.38, 1.19, 0.59, 0.55, 0.52, 0.50, 0.48, 0.45, 0.43, 0.42, 0.41, 0.39, 0.36, 0.35, 0.34, 0.33, 0.32, 0.31, 0.30, 0.29, 0.28/ 
DATA FPOS/2.78, 1.75, 1.30, 1.01, 0.83, 0.70, 0.60, 0.54, 0.48, 0.43, 0.40, 0.36, 0.31, 0.30, 0.27, 0.25, 0.22, 0.20, 0.16, 0.13, 0.10/ 
C WRITE( 7, 900) 
900 FORMAT(1H1) 
C

C FIRST MAJOR OPERATION IS TO RE-DISTRIBUTE THE SIGNATURE. IF THERE IS NO LEADING SIGNATURE, IT STARTS AT THE HEAD OF THE ARRAY.

JMOVE=0 
C IF THERE IS A LEADING SHOCK, LEAVE 20 BLANKS AT THE START IF((XX(2)-XX(1)) .LE. XX(2)*1.0E-05) JMOVE=20

J=0 
NXX=NX-1 
C THE FOLLOWING LOOP LOOKS FOR SHOCKS, SAVES THEIR VALUE, AND MARKS THE POSITION ON THE INPUT ARRAYS (POSITION = SECOND ASSOCIATED MESH POINT)

DO 1 I=1,NXX 
ADADAD = XX(I+1) - XX(I) 
DADADA = XX(I+1)*1.0E-05 
IF((XX(I+1)-XX(I)).GT.XX(I+1)*1.0E-05) GO TO 1 
J=J+1 
IDP(J)=I+1 
PSH(J)=ABS(PP(I+1)-PP(I)) 
1 CONTINUE 

C WE NOW KNOW THERE ARE NSH SHOCKS, AT IDP() WITH JUMPS PSH()

C THE NEW SIGNATURE WILL HAVE 100 INTERVALS (101 POINTS), PLUS DOUBLED C POINTS FOR EACH SHOCK, PLUS THE ALLOWED SPACE AT THE BEGINNING.

NSIG=101+NSH+JMOVE 
XDEL=(XX(NX)-XX(1))/100. 
C THE FOLLOWING IS A LOOP (ENDING JUST ABOVE LABEL 7) WHICH RUNS I FROM C 1 TO NX (SIZE OF INPUT SIGNATURE), WITH BACKWARD-RUNNING INDEX IND C GOING FROM NX TO 1. (IND HAS THIS ROLE OVER THE TOP HALF OF THE LOOP;
C IT IS USED AS A LOCAL POINTER AFTER LABEL 2.) THIS STRUCTURE, RATHER C THAN A DO, IS USED BECAUSE C I AND IND ARE DOUBLE-BUMPED AT SHOCKS.

I=0
6 I=I+1
IND=NX-I+1
C SET SHOCK FLAG. NOTE THAT J IS THE LAST SHOCK; IT WILL BE DECREMENTED C AS EACH IS HANDLED.

ISH=1
IF(IDP(J).EQ.IND) ISH=2
C MOVE INPUT POINT TO NEAREST ROUNDED-FORWARD POSITION ON NEW ARRAY.
C NOTE THAT INPUT ARRAY SHOULD HAVE NO MORE THAN 100 POINTS, SO THAT BY C WORKING FROM THE REAR OF AN ARRAY WITH MORE THAN 100 POINTS, WE WILL C CERTAINLY NOT OVERWRITE THE FIRST POINT. OVERWRITING CAN OCCUR ONLY C IF TWO ORIGINAL POINTS ARE LESS THAN XDEL APART; THIS IS OK BECAUSE C OUR ROUNDING-DOWN-TO-THE-100-INTERVAL ALGORITHM DOES NOT SUPPORT THAT C RESOLUTION.

N=N+IF(XX(IND)/XDEL)+J+JMOVE+1
XX(N)=XX(IND)
PP(N)=PP(IND)
C I LOW POINTS TO MESH POINT JUST ABOVE THE ONE WE JUST FILLED; THERE C USUALLY WILL BE SOME EMPTIES TO BE FILLED, SINCE GENERALLY INPUT C POINTS ARE SPACED MUCH GREATER THAN XDEL
C LOW=N+1
GO TO (2,3),ISH
C FOR A SHOCK, WE MOVE THE UPSTREAM POINT AS WELL AND RE-BUMP C THE POINTERS
3 XX(N-1)=XX(IND-1)
PP(N-1)=PP(IND-1)
IDP(J)=N
J=N-1
I=I+1
2 IF(IND.EQ.NX) GO TO 4
C IND NOW POINTS TO THE CURRENT POINT (DOWNSTREAM POINT OF SHOCK, C IF THERE IS ONE)
IND=LOW-1
C SET UP PROPORTIONALITIES FOR LINEAR FILL-IN BETWEEN THIS NEW POINT C AND LAST ONE
C C BRANCH TO AVOID ABEND
IF (IHGH.LT.ILOW) GOTO 4

XDELT=(XX(IHGH+1)-XX(IND))/FLOAT(IHGH-ILOW+2)
POELT=(PP(IHGH+1)-PP(IND))/FLOAT(IHGH-ILOW+2)
DO 5 IDX=LOW,IHGH
XX(IDX)=XX(IDX-1)+XDELT
5 PP(IDX)=PP(IDX-1)+POELT
4 IHGH=LOW-2
C IHGH POINTS TO UPSTREAM OF CURRENT POINT; NEXT PASS THROUGH, IT WILL C CORRESPOND TO UPSTREAM OF LAST POINT.
IF(IHGH.EQ.2) IHGH=IHGH-1
IF(1.EQ.NX) GO TO 7
GO TO 6
7 CONTINUE
C SIGNATURE RE-DISTRIBUTED. NOW APPLY G-S SOLUTION TO SHOCKS.
P1=0.
FF=1.
P1=0.
C START OF MAIN LOOP, CYCLING THROUGH SHOCKS STARTING AT FRONT.
DO 10 J=1,NSH
   INDX=IDP(J)
   P1=P1+FF*(PP(J)-P1)
   CPREF=2.*PSH(J)/1.4/PO
   YREF=SREF/SREF/2./R
C FF AND FL ARE AMPLITUDE AND LENGTH SCALE FACTORS; SEE WR 75-7
C EQUATIONS 62 AND 63
C*******NEED TO MAKE SURE FL CORRESPONDS TO LENGTH COORDINATE ALONG
C RAY; IT ISN'T RIGHT FOR GROUND COORDINATE BLUNDER **********
FF=0.74*(YREF/1.2/CPREF/R)*0.2
FL=(12.*CPREF*(YREF**0.25)*(R/2.)**(7.12.10.2))**1.2
P1=PP(INDX-1)
C APPLY G-S SOLUTION TO SHOCK-POINT
   PP(INDX-1)=FF*PSH(J)*FNEG(1)
IF(J.GT.1)GO TO 20
IF(JMOVE.EQ.0)GO TO 26
C FOR LEADING SHOCK, APPLY G-S UPSTREAM ELEMENT WITHOUT SCALING;
C XX VALUES (NOT SPACED XDEL) AS WELL AS PP VALUES ARE CREATED
C IN THE 20 POINTS ADDED FOR THIS PURPOSE
DO 25 II=1,20
   XX(II)=FLOAT(II-21)*FL/10.
25 PP(II)=FF*(PSH(1)*FNEG(22-II)-P1)
C WE'RE DONE WITH UPSTREAM PART HERE, SO...
   GO TO 31
26 ILNG=INDX-2
C FIRST SHOCK, BUT NOT LEADING. NOTE HOW MANY POINTS ARE AHEAD OF THIS
C SHOCK, THEN GO TO UPSTREAM G-S APPLER, WITHOUT WORRYING ABOUT
C MID-POINT TO SHOCK AHEAD (SINCE THERE ISN'T ANY)...
   GO TO 32
C BEGIN GENERAL CASE OF APPLYING UPSTREAM G-S SOLUTION WHEN THERE IS A
C SHOCK AHEAD OF CURRENT ONE. FIRST FIND MID-POINT, THEN SET ILNG TO
C THE NUMBER OF POINTS JUST WITHIN THE HALF-DISTANCE.
   JINDX=IDP(J-1)
   XINT=(XX(INDX)+XX(JINDX))/2.
   ILNG=0
   DO 21 II=1,100
      IF(XX(INDX-II-1).LT.XX(1)-XINT)GO TO 22
21 ILNG=ILNG+1
22 CONTINUE
C LET HIM KNOW WHERE THE BREAK IS BETWEEN THESE TWO SHOCKS
200 FORMAT(1X,4HPOSSIBLE INVALID PRESSURE CORRECTION FROM SHOCK 12,
   18H AT X = ,F10.2)
C NOW APPLY THE UPSTREAM SOLUTION, GOING TO EITHER THE MID-POINT (GEV-
C ERAL CASE) OR THE FRONT (FIRST-BUT-NOT-LEADING SHOCK CASE)
C THERE ARE THREE ALGORITHMS BELOW:
C * UP TO LABEL 40, WHERE FNEG TABLE IS OVER IRREGULAR X/L INTERVALS.
C * FROM LABEL 40 UP TO 41, WHERE X/L = 0.1
C
C - LABEL 41 ONWARD, WHERE A LINEAR EXTRAPOLATION FORMULA APPLIES.
32 DO 30 II=1,ILNG
   XDIST=(XX(INDX-1)-XX(INDX-II-1))*10./FL
   INDX=IF(XDIST.GT.1)
   GO TO (50,50,50)
   IF(XDIST.GT.0.3).GO TO 55
   PP(INDX-I-I)=IPSH(J)*(FNEG(1)+(2.21-FNEG(1))/0.3*XDIST)+PP(INDX-111-I)-P1)*FF+P11
   GO TO 30
55 PP(INDX-I-I)=(PSH(J)*(2.21*(FNEG(2)-2.21))/0.7*(XDIST-0.3))+
   1PP(INDX-1-I-I)-P1)*FF+P11
   GO TO 30
60 IF(XDIST.GT.1.8).GO TO 65
   PP(INDX-1-I-I)=(PSH(J)*(0.6-FNEG(2))/0.8*(XDIST-1.1))+
   1PP(INDX-1-I-I)-P1)*FF+P11
   GO TO 30
65 PP(INDX-1-I-I)=(PSH(J)*(0.6-FNEG(3)-0.60)/0.2*(XDIST-1.8))+
   1PP(INDX-1-I-I)-P1)*FF+P11
   GO TO 30
40 IF(XDIST.GE.20.).GO TO 41
   PP(INDX-1-I-I)=(PSH(J)*(FNEG(IND)+<FNEG(IND+1)-FNEG(IND)))*
   1(XDIST-FLAT(IND-111)))+PP(INDX-1-I-I)-P1)*FF+P11
   GO TO 30
41 FN=0.28-0.01*(XDIST-20.)
   IF(FN<0.7).FN=0.
   PP(INDX-1-I-I)=(PSH(J)*FN+PP(INDX-1-I-I)-P1)*FF+P11
   CONTINUE
30 CONTINUE
C DOWNSTREAM G-S SOLUTION IS NOW APPLIED FOLLOWING THE SHOCK. LOGIC
C IS SIMILAR TO ABOVE, EXCEPT WE NOW WORRY ABOUT THE STATUS TO THE
C REAR, I.E. NOW FAR TO THE NEXT SHOCK BACK (IF ANY), IS THE LAST SHOCK
C AT THE END OF THE SIGNATURE, ETC. THE APPLICATION OF FPOS IS A BIT
C SIMPLER, SINCE THE FPOS ARRAY IS ALL ON A UNIFORM X/L - 0.1 MESH.
C THERE ARE THUS ONLY A SINGLE INTERPOLATION FORMULA, PLUS EXTRAPOL
C ATION BEYOND THE TABLE.
31 PP(INDX)={PSH(J)}*FPOS(1)={PP(INDX)}*P1)*FF+P11
   I(F.J.LT.NSH).GO TO 80
   IF.FD(J).LT.NSIG).GO TO 81
   DO 70 II=1,10
   XX(NSIG+II)=XX(NSIG)+FLOAT(11)*FL/10.
   70 PP(NSIG+II)=(PSH(NSH)*FPOS(I+1)-P1)*FF+P11
   NSIG=NSIG+20
   GO TO 95
81 ILNG=NSIG-IDP(NSH)
   GO TO 82
80 JINDX=IDP(J+1)
   XINT=(XX(JINDX)-XX(INDX))/2.
   ILNG=0
   DO 83 II=1,100
   IF(XX(INDX+XINT).LT.XX(INDX+II)).GO TO 84
   83 ILNG=ILNG+1
84 CONTINUE
C WRITE( 7,200)J,XX(INDX+ILNG)
82 DO 90 II=1,ILNG
  XDIST=(XX(INDX+II)-XX(INDX))*.10./FL
  IF(XDIST.GE.20.)GO TO 91
  INDP=IFIX(XDIST)+1
  PP(INDX+II)=(PSH(J)*(FPOS(INDP)+(FPOS(INDP+1)-FPOS(INDP)))*XDIST-
    1FLOAT(INDP-1))=PP(INDX+II)-P1)*FF=P11
  GO TO 90
91 FPSS=0.10-0.03*(XDIST-20.)
    IF(FPSS.LT.0.)FPSS=0.
    PP(INDX+II)=(PSH(J)*FPSS+PP(INDX+II)-P1)*FF=P11
90 CONTINUE
10 CONTINUE
C ALL DONE. PRINT SIGNATURE AND GO HOME
  NSIG = NSIG - 1
95 CONTINUE
C 95 WRITE( 7,300)
  300 FORMAT(1HO,11X,18HF0CUSSED SIGNATURE)
    TFAC=1000./AO
C WRITE( 7,400)XX(I)*TFAC,PP(I),I=1,NSIG)
  400 FORMAT(1HO, (12X,7HT, MSEC,14X,6HP, PSF)/// (F20.2,F20.3))
C P?4AX TO SUMMARY FILE
  PMAX=PP(1)
    DO 500 II=2,NSIG
       PMAX=AMAX1(PMAX,PP(II))
500 CONTINUE
510 FORMAT( PMAX(FOC)=':,F10.3)
    RETURN
END
SUBROUTINE GETDLT
PROGRAMMER: PHILIP J. DAY
XOMTECH INC.
BBN LABORATORIES
DATE: DECEMBER 29, 1986
PURPOSE: TO GET THE DELTA INCREMENT FOR THE PHI ANGLES IN A CAU
GROUND INTERCECTION.

SUBROUTINE GETDLT(PHIO, PHI(NPHI), DELTA)

REAL PHIO, PHI(NPHI), DELTA
INTEGER NPHI

C FIND THE TWO ANGLES PHIO IS BETWEEN AND TAKE 1/10 OF THEIR DIFFERENCE
DO 100 I = 1, NPHI - 1
   IF (PHIO.GE.PHI(I).AND.PHIO.LE.PHI(I+1)) THEN
      DELTA = ABS(PHI(I+1) - PHI(I))/10.0
      RETURN
   ENDIF
100 CONTINUE

C SET DEFAULT VALUE FOR DELTA
DELTA = 0.5
RETURN
END
SUBROUTINE FINDT(N,*)
COMMON /CAUSTC/ NUMC, TRACE, CT(360), CPHI(360), CXYZ(360, 3)
REAL CXYZ
LOGICAL TRACE

COMMON /RPOSN/ NPT, COPSN, RT(200), RXYZ(200, 3), RAGE(200),
+ RPFACT(200), RVLF, REMEM
REAL RT, RXYZ, RAGE, RPFACT, RVLF
INTEGER NPT, COPSN
LOGICAL REMEM

COMMON /HRPOSN/ HNPT, HCPSN, HRT(200), HRXYZ(200, 3),
+ HRAGE(200), HRPFAC(200), HRLFT, HREMEM
REAL HRT, HRXYZ, HAGE, HRPFAC, HRLFT
INTEGER HNPT, HCPSN
LOGICAL HREMEM

C C- RESET THE THE ORIGION C
C
IF (N.EQ.1) THEN
CT(1) = HRT(HCPSN)
CXYZ(1,1) = HRXYZ(HCPSN, 1)
CXYZ(1,2) = HRXYZ(HCPSN, 2)
CXYZ(1,3) = HRXYZ(HCPSN, 3)
RETURN
ENDIF

IF (CT(1).GT.CT(N)) THEN
DO 100 I = 2, NPT
IF (HRT(I).GE.CT(N)) THEN
DO 110 J = 1, 3
CXYZ(I,J) = HRXYZ(I-1, J) + (HRXYZ(I,J)-HRXYZ(I-1,J))*
+ (CT(N) - HRT(I-1))/(HRT(I) - HRT(I-1))
CT(I) = CT(N)
110 CONTINUE
ELSEIF (CT(N).GT.CT(1)) THEN
ENDIF
100 CONTINUE
RETURN
END
DO 200 I = 2, NPTR
    IF (RT(I) .GE. CT(1)) THEN
        DO 210 J = 1, 3
            CXYZ(N, J) = RXYZ(I-1, J) + (RXYZ(I, J) - RXYZ(I-1, J)) * 
            (CT(1) - RT(I-1))/(PT(1) - RT(I-1))
            CT(N) = CT(1)
        CONTINUE
    END IF
210 CONTINUE
RETURN
END IF
200 CONTINUE
END IF
RETURN
C
C====================================================================================================
C====================================================================================================
C====================================================================================================
C
ENO
OF
FOCUS
ROUTINES
ENO
C/ADD NAME=SONICSOM

C==========================================================================
C
C-- MODIFIED TRAPS CODE FOR USE WITH BOOKMAP AND FOGGROM PROGRAMS --
C
C==========================================================================
C
C ************************************************************************
C
C **** T.R.A.P.S. - SONIC BOM MODELING PROGRAM
C **** T.RACING RAYS AND AIR PRESSURE SIGNATURES
C **** (SEE NOAA TECHNICAL MEMORANDUM ERL ARL-87)
C
C ************************************************************************
C
C ************************************************************************
C
C DR. ALBION O. TAYLOR,
C NOAA/AIR RESOURCES LABORATORIES R/E/AR
C RM. 921, GRAMAX BUILDING
C 8060 13TH STREET
C SILVER SPRING, MD 20910
C JULY, 1980
C
C ************************************************************************
C
C==========================================================================
C
C/ADD NAME=BLKDATA

C BLOCK DATA TRPRAY

C

C COMMON BLOCKS

C COMMON /PUNITS/ PTABL,TTABL,HTABL,STABL,TIMTAB,LTABL,FTABL
CHARACTER*8 PTABL(6),TTABL(4),HTABL(6),STABL(9),TIMTAB(2),LTABL(6)
CHARACTER*5 FTABL(5)

C COMMON /CPUNIT/ CPTABL,CCTABL,CHTABL,CLTABL,CFTABL,
+ ATMPO,ACPOT
REAL CPTABL(5),CCTABL(2,4),CHTABL(6),CLTABL(9),CFTABL(5)
REAL ATMPO(6),ACPOT(6)

C COMMON /ATMCON/ REARTH,G0,RSTAR,RMO,ROGOMO

C COMMON /CLASES/ CNAMES(30)
CHARACTER*8 CNAMES
COMMON /CLASSS/ NRCURV(2,2),TYPRAY(3,2,2),DIRECT,LOFT,UP,DOWN
LOGICAL TYPRAY,DIRECT,LOFT,UP,DOWN
C DEFAULT ON HEIGHT = GEOPOTENTIAL(ATMOSPHERE), GEOMETRIC(AIRCRAFT)
C INTERNAL PROGRAM UNITS HEIGHT GEOPOTENTIAL METERS(ATMOSPHERE),
C GEOMETRIC METERS ALL OTHER HEIGHTS AND LENGTHS, TEMPERATURE
C DEGREES KELVIN, PRESSURE KILOPASCALS (KPA), SPEED METERS PER SECOND.
C KNOTS AND NAUTICAL MILE CONVERSIONS BASED ON THE INTERNATIONAL
C NAUTICAL MILE OF 1852. METERS EXACTLY (6076.1155FT), AS ADOPTED
C BY THE U.S. IN 1954, RATHER THAN THE BRITISH ADMIRALTY NAUTICAL
C MILE OF 6080 FT. (1853.18METERS) OR THE U.S. NAUTICAL MILE PRIOR
C TO 1954 OF 6080.21 FT. (1853.250METERS)

DATA PTABL/'KPA','MB','NSM','PSF','PSI'/
DATA CPTABL/1.,0.1,1.E3,1.E3,4.78803E-2,6.89476/
DATA TTABL/'C','F','K','R'/
DATA CTTABL/ 1.,273.150, 1.80,459.670, 1.,0., 1.80,0./
DATA HTABL/'FT','GMFT','GMM','GPFT','GPM','METERS'/
DATA CHTABL/0.3048,.3O48,1.,.XO48,1.,1./
DATA ATMPOT/.TRUE.,2-.FALSE.,3-.TRUE./
DATA ACPOT/3*.FALSE.,2-.TRUE.,.FALSE./
DATA STABL/'FPS','FTPS','KNOTS','KT','KPH','MPH','MPS','NMPH',
A 'SNPH'/
DATA CSTABL/2*.3048,2*.5144444,.2777778,4470400,1.,5144444,
A .4470400/
DATA TIMTAB/'HHMMSS','SSSSSSSS'/
DATA LTABL/'FT','KM','METERS','MILES','NM','SMI'/
DATA CLTABL/.3048,1E3,1.,1609.344,1852.,1609.344/
DATA FTABL/'GM','GRAMS','KG','LS','POUNDS'/
DATA CFTABL/211E-3,1.,2*.45359237/
C
C ******************************************
C
C REARTH=RADIUS OF EARTH FOR CONVERSION GEOMETRIC TO GEOPOTENTIAL
C METERS. ROMO=STAR/MO AND ROGOMO=STAR/(GO*MO)
C WHERE RSTAR=UNIVERSAL GAS CONSTANT=8.31432E3 JOULES / (KMOL-DEGK),
C GO=9.80665M/SEC''2, AND MO=MEAN MOLECULAR WEIGHT OF STANDARD DRY
C AIR (28.9644 KG/KMOL) (SEE U.S. STANDARD ATMOSPHERE 1976)

DATA GD/9.80665/,ROMO/287.0531/
C
C ******************************************
C
C RAY CLASSES. G=GROUND, M=MID HEIGHT (ABOUT 50KM) H=EXTREME
C WEIGHT (>100KM OR MORE). RAY CLASSES DEFINED IN THE ORDER IN WHICH
C A RAY TOUCHES AND RETURNS FROM ANY OF THESE LAYERS. Thus, A GMG
C RAY HAS REFLECTED FROM THE GROUND, CURVED FROM THE MID LEVEL, AND
C TOUCHED THE GROUND AGAIN. A MG (OR M) RAY ROSE DIRECTLY FROM
C AIRCRAFT TO MID LAYER AND CURVED TO TOUCH GROUND.
DATA CHANES/ENDCLASS,'FULL','G','GH','GHG','GHGH','GHGHG',
+\'GHGHGH','GHGHGHG','W\',\'GMG\',\'GSGMG','GIGMGM','GMG14GIG',
+\'HG','1GH','HGHG',\'HGH',\'HGHGHG','N','MG',\'4GM','MGP4G',\'NGMGN','MGNGMG','NGPRINT','SHOCKS','SUMMARY'/

END
SUBROUTINE SETUP INITIALIZES THE F-FUNCTIONS AND ALL OF THE FLAG
THAT THE T.R.A.P.S ROUTINES USE. IT CALLS ATMSIN TO SET UP THE
ATMOSPHERE TABLE.

SUBROUTINE SETUP

COMMON /ACIONT/ IDENT
CHARACTER*8 IDENT

COMMON /ACWEG/ ACV,ACL

COMMON /ACSPOT/ TIME,XRO,YRO,ZRO,XDOT,YDOT,ZDOT,AIRSPD,ASPDOT,
+ CO,VO,CDOT,XMACH,XMADOT,XMU,XMUDOT,COSMU,
+ SIND, Ek(3,3), Ekdot(3,3), GLOAD, HEADING, CLIMB, BANK,
+ XDOT,YDOT,ZDOT,XDDOT,YDDOT,ZDDOT

COMMON /UNITS/ WTUNIT,HTUNIT
CHARACTER*8 WTUNIT,HTUNIT

COMMON /PRINTS/ TITLE(30),TIMLBL
CHARACTER*4 TITLE
CHARACTER*8 TIMLBL

COMMON /PRINTC/ KTSPIN,CRTIM
LOGICAL CRTIM

COMMON /GROUND/ GLAYER, ZGRNO, GGRNO, UGRNO, VGRNO, REFLECTION
INTEGER GLAYER

COMMON /LYRDEF/ NLAYER, GMZA(200), INDPTH(200), INDWND(200),
+LYRPR(200), KAYER, ZTOP, ZBOT
INTEGER INDPTH, INDWND
LOGICAL LYPRT

COMMON /ATMSPH/ GAM,C,U,V,DC0Z,DUDZ,DVDZ,D2C0Z2,D2UDZ2,D2VDZ2,RHO
REAL GAM,C,U,V

COMMON /PUNIT/ PTABL,TTABL,HTABL,STABL,TMTAB,LTABL,FTABL
CHARACTER*8 PTABL(6), TTABL(4), HTABL(6), STABL(9), TMTAB(2), LTABL(6)
CHARACTER*8 F TABL(5)

COMMON /CPUNIT/ CPTABL,CTTABL,CHTABL,CSTABL,CLTABL,CFTABL,
+ ATMPOT,ACPOT
REAL CPTABL(6),CTTABL(2,4),CHTABL(6),CSTABL(9),CLTABL(6)
REAL CFTABLE(5)
LOGICAL ATMPOT(6), ACPOT(6)

C DEFAULT ON HEIGHT = GEOPOTENTIAL(ATMOSPHERE), GEOMETRIC(AIRCRAFT)
C INTERNAL PROGRAM UNITS HEIGHT GEOPOTENTIAL METERS(ATMOSPHERE),
C GEOMETRIC METERS ALL OTHER HEIGHTS AND LENGTHS, TEMPERATURE
C DEGREES KELVIN, PRESSURE KILOPASCALS (KPA), SPEED METERS PER SECOND.
COMMON /ATMCON/ REARTH, GO, RSTAR, ROMO, ROGOMO

COMMON /CLASSES/ CNAMES(30)
CHARACTER*8 CNAMES

COMMON /CLASSES/ WARCURV(2,2), TYPRAY(3,2,2), DIRECT, LOFT, UP, DOWN
LOGICAL TYPRAY, DIRECT, LOFT, UP, DOWN

C RAY CLASSES. G=GROUNA, M=MID HEIGHT (ABOUT 50KM) H=EXTREME
C HEIGHT (100KM OR MORE). RAY CLASSES DEFINED IN THE ORDER IN WHICH
C A RAY TOUCHES AND RETURNS FROM ANY OF THESE LAYERS. THUS, A GMG
C RAY HAS REFLECTED FROM THE GROUND, RECURRED FROM THE MID LEVEL, AND
C TOUCHED THE GROUND AGAIN. A NG (OR M) RAY ROSE DIRECTLY FROM
C AIRCRAFT TO MID LAYER AND RECURRED TO TOUCH GROUND.
LOGICAL FND
INTEGER KTRANS(29)
CHARACTER*8 PRYP(4)
CHARACTER*8 BUF(9), TPUNIT(3)
CHARACTER*8 SBUF(9)
CHARACTER*120 TITLE1
EQUIVALENCE (IUF(1), SBUF(1)), (TITLE1, TITLE(1))

DATA KTRANS/4, 0, 24*-1, 1, 3, 2/
DATA PRYP/' NO', 'SUMMARY', 'SHOCKS', ' FULL'/
DATA TPUNIT/'WEIGHT', 'HEIGHT', 'RAYCLASS' /

C - INPUT FILES
C
C OPEN(5, FILE='TITLE', STATUS='OLD')
TITLE1 = '------------------- BOOM2 TESTING -------------------'
C
C - OUTPUT FILES
C
C - TEMPORARY FILES
C
C OPEN(9, STATUS='SCRATCH')
C OPEN(11, STATUS='SCRATCH')

C READ 2 TITLE CARDS (1-72 ON FIRST CARD, 1-24 ON 2ND)
C READ(5, 10) TITLE
10 FORMAT(18A4)
C
CALL FFUNC(IDENT, FND)
CALL UNITIS(WTUNIT, FTABL, 5, IMUNIT, TPUNIT(1), 4)
ACWT=ACWT*CFTABL(IMUNIT)
BUFF=ACWT/CFTABL(IMUNIT)

CALL UNITIS(CHTUNIT,HTABL,6,IGUNIT,TPUNIT(2),6)

HEIGHT = ZGRND

HEIGHT=HEIGHT*CFTABL(IGUNIT)
C----------------- CHANGED FROM / TO *
IF(ACPOT(IGUNIT)) HEIGHT=HEIGHT/(1.-HEIGHT/REARTH)
ZGRND=HEIGHT
IF(ACPOT(IGUNIT)) HEIGHT=HEIGHT/(1.+HEIGHT/REARTH)

HEIGHT=HEIGHT*CFTABL(IGUNIT)
C----------------- CHANGED FROM * TO /

C
C READ RAY TYPES TO BE RECORDED.
    DO 412 K=1,2
    DO 42 L=1,2
    DO 41 M=1,3
       TYPRAY(M,K,L)=.FALSE.
        41 CONTINUE
        NRCURV(K,L)=-1
        42 CONTINUE
        412 CONTINUE
KTPSIG=.1
UP=.FALSE.
DOWN=.FALSE.
DIRECT=.FALSE.
LOFT=.FALSE.
    43 CONTINUE
BUF(1) = 'G'
BUF(2) = '1'

    DO 55 K=1,9
    CALL UNITIS(BUF(K),CHAMES,30,LCUNIT,TPUNIT(3),1)
    IF(LCUNIT.LE.1) GO TO 60
    IF(KTRNS(LCUNIT-1)) 47,45,50
    45 DIRECT=.TRUE.
    DOWN=.TRUE.
    GO TO 55
    47 IF(LCUNIT.GE.28) GO TO 50
    LOFT=LOFT. OR. (LCUNIT.LE.14)
    LCUNIT=(LCUNIT-4)/2
    KDWNUP=LCUNIT/6
    KHM=LCUNIT/3-KDWNUP*2
    KMH=2-KHM
    KRCRV=LCUNIT-3*KHM-6*KDWNUP
    TYPRAY(KRCRV+1,KDWNUP+1,KMH)=.TRUE.
    NRCURV(KDWNUP+1,KMH)=MAXO(NRCURV(KDWNUP+1,KMH),KRCRV+1)
UP=UP.OR.(KDWNUP.NE.0)
DOWN=DOWN.OR.(KDWNUP.EQ.0)
GO TO 55

50 KTPSIG=MAX0(KTPSIG,KTRNS(LCUNIT-1)-1)
55 CONTINUE

C
GO TO 43
60 CONTINUE
DO 75 K=1,2
   DO 75 L=1,2
      DO 75 M=1,3
         IF(.NOT.TYPRAY(M,K,L)) GO TO 75
         KTABLE=12*(K-1)+6*(L-1)+2*M+3
      75 CONTINUE
   IF(KTPSIG.EQ.-1) KTPSIG=3
   CALL ATMSIN
   CALL FNOLYR(ZGRND,'80')
80 CALL AIR((ZGRND))

CGRND=C
UGRND=U
VGRND=V

DO 888 II = 1,4
   ZZZ = GMZA(II)
   CALL AIR(ZZZ)
888 CONTINUE
OPEN(8,STATUS='SCRATCH')
RETURN
END
SUBROUTINE ATMSIN

COMMON /PTN/ NPTH,PRESS(97),TMPMOL(97),GPHC(97),GAMMA(97)
COMMON /WINDS/ NWINDS,GPHW(80),DIR(80),TURN(79),SPEED(80)
COMMON /ATMCON/ REARTH,GO,RSTAR,ROMO,ROGOMO
COMMON /GROUND/ GLAYER,ZGRND,CGRND,VGRND,REFLFC
INTEGER GLAYER
COMMON /LYRDEF/NLAYER,GMZA(200),INOPTN(200),INDWND(200),
*LYRPRT(200),XLYR,ZTOP,ZBOT

INTEGER INOPTN,INDWND
LOGICAL LYRPRT
REAL PRINTL(24)
LOGICAL REC


ZFROMH(H)=H/(1.-H/REARTH)

REC = .FALSE.
CALL PTNIN
CALL WINDIN

ZPTH=ZFROMH(GPHC(1))
ZWIND=ZFROMH(GPHW(1))
ZM1 = PRINTL(1)

NLAYER=1
GMZA(NLAYER)=AMAX1(ZM1,ZPTH,ZWIND)

DO 2 KPRT=1,24
   IF(PRINTL(KPRT).GT.GMZA(1)) GO TO 3
   LPRT=MINO(KPRT,23)
2 CONTINUE

3 DO 4 KPTH=1,NPTH
ZPTH=ZFROH(GPHC(KPTH))
IF(ZPTH.GT.GMZA(1)) GO TO 5
LPTH=MINO(KPTH,NPTH-1)
4 CONTINUE

5 INDPTH(NLAYER)=LPTH

DO 6 KWIND=1,NWINDS
    ZWIND=ZFROH(GPHC(KWIND))
    IF(ZWIND.GT.GMZA(1)) GO TO 7
    LWIND=MINO(KWIND,NWINDS-1)
6 CONTINUE

7 INDWIND(NLAYER)=LWIND

10 KPRT=MINO(LPRT+1,24)
    KPTH=MINO(LPTH+1,NPTH)
    KWIND=MINO(LWIND+1,NWINDS)

ZPRT=PRINTL(KPRT)
ZPTH=ZFROH(GPHC(KPTH))
ZWIND=ZFROH(GPHC(KWIND))
ZLEVEL=MINT(ZPRT,ZWIND,ZPTH)

IF(ZLEVEL.LE.QMZA(NLAYER).OR.(NLAYER.GE.200)) GO TO 200

NLAYER=NLAYER+1

IF (ZGRND.EQ.ZLEVEL) THEN
    GLAYER = NLAYER
    REC = .TRUE.
END IF

IF (.NOT.REC.AND.ZGRND.LT.ZLEVEL) THEN
    GMZA(NLAYER) = ZGRND
    GLAYER = NLAYER
    REC = .TRUE.
ELSE
    GMZA(NLAYER)=ZLEVEL
END IF

LYRPRT(NLAYER)=.FALSE.
IF(GMZA(NLAYER).LT.ZPRT) GO TO 30
LYRPRT(NLAYER)=.TRUE.
LPRT=MINO(KPRT,25)

30 IF(GMZA(NLAYER).EQ.ZPTH) LPTH=MINO(KPTH,NPTH-1)
   INDPTH(NLAYER)=LPTH
   IF(GMZA(NLAYER).EQ.ZWIND) LWIND=MINO(KWIND,NWINDS-1)
   INDWIND(NLAYER)=LWIND
GO TO 10

200 LYRPRT(1)=.TRUE.
LYRPRT(NLAYER)=.TRUE.
SUBROUTINE PTDHIN READS THE RAOB FILE. IT CONVERTS ALL DATA TO S.I. UNITS, INTERPOLATES DEWPOINT DATA AS NEEDED AND CALCULATES VIRTUAL OR MOLECULAR SCALE TEMPERATURES FROM THE TEMPERATURE AND DEWPOINT DATA AS APPROPRIATE, RETURNS A TABLE OF VIRTUAL TEMPERATURES, Pressures, A NIGHTS.

SUBROUTINE PTDHIN
COMMON /PRINTS/ TITLE(30),TIMLBL
CHARACTER*4 TITLE
CHARACTER*B TIMLBL

COMMON /PRINTC/ KTPSIG,CVRTIM
LOGICAL CVRTIM

COMMON /PTH/ WPTh,PRES(97),TMPHOL(97),GPHC(97),GAMMA(97)

C :: TMPMOL= 'MOLECULAR SCALE TEMPERATURE' = VIRTUAL TEMPERATURE
C :: GPH = GEOPOTENTIAL HEIGHT

REAL GPH(80),TEMPK(80),DEWPHT(80)
REAL STANHT(21)
REAL STANTP(21)
REAL STANGN(21)

COMMON /PUNITs/ PTABLE,TTABLE,HTABLE,TIMTABLE,LTABLE,FTABLE
CHARACTER*B TABLE(6),TTABLE(4),HTABLE(6),STABLE(9),TIMTABLE(2),LTABLE(6)
CHARACTER*B FTABLE(5)

COMMON /CPUNIT/ CPTABLE,CTABLE,CHTABLE,CSTABLE,CLTABLE,CFTABLE,
+ ATMPOT,ACPOT
REAL CPTABLE(6),CTABLE(2,4),CHTABLE(6),CSTABLE(9),CLTABLE(6)
REAL CFTABLE(5)
LOGICAL ATMPOT(6),ACPOT(6)

C DEFAULT ON HEIGHT = GEOPOTENTIAL(ATMOSPHERE), GEOMETRIC(AIRCRAFT)
C INTERNAL PROGRAM UNITS HEIGHT GEOPOTENTIAL METERS(ATMOSPHERE),
C GEOMETRIC Meters ALL OTHER HEIGHTS AND LENGTHS, TEMPERATURE
C DEGREES KELVIN, PRESSURE KILOPASCALS (KPA), SPEED METERS PER SECOND.

CHARACTER*B STANRD,FINISH,BLANK
CHARACTER*B TPCODE(4)
CHARACTER*B BUF(4)
REAL DUMMY(7),DEFARY(4)

COMMON /WKRAOB/ HMISS(97),PMISS(97)
LOGICAL HMISS,PMISS
LOGICAL DMISS(80), GEOMET, GEOPOT, SOMEHT, TRUE
LOGICAL FALSE
COMMON /ATMCON/ REARTH, GO, RSTAR, ROMO, ROGOMO

DATA STANHT/-5E3,11E3,20E3,32E3,47E3,51E3,71E3,84E3,97716.,
+ 94572.,97442.,99420.,102326.,104196.,107162.,108129.,
+ 117777.,121627.,125473.,130274./
DATA STANTP/3Z0.65,216.65,216.65,228.65,270.65,270.65,214.65,
+ 186.95,187.16,189.35,194.28,204.65,213.22,221.65,234.19,242.86,
+ 254.27,397.09,453.89,508.05,571.42/
DATA STANGN/7*-1.401,2-1.402,1.404,1.406,1.408,1.411,1.413,
+ 1.416,1.417,1.419,1.432,1.436,1.441,1.446/
DATA STANRD/STANO,ARO/,FINISH/END/,BLANK/I
DATA TPUNIT/IPRESS/, TEMPI, DEW PT., HEIGHT/
DATA DEFARY/4*-1.66/
DATA TRUE/. TRUE., FALSE./, FALSE./, FALSE./

C REARTH=RADIUS OF EARTH FOR CONVERSION GEOMETRIC TO GEOPOTENTIAL

C METERS.
VAPRS(DWPT)=6105*EXP(25.22*(1.-273./DWPT)-5.31*ALOG(DWPT/273.))
RATMPX(PRS,DWPT)=0.622/(PRS/VAPRS(DWPT)-1.)
VIRTMP(TMP,RTMIX)=TMP*(1.0.61653*RTMIX)
GNW(RTMIX)=1.401*(1.0.899*RTMIX)/(1.2.016*RTMIX)
PRESS(1)=177.68
PMISS(1)=FALSE
TMPMOL(1)=STANTP(1)

C READ TITLE/STANDARD CARD AND INTERPRET
C
C READ(5,5,END=200) BUF
C 5 FORMAT(9A8)
C
C BRANCH AROUND TO SETUP STANDARD ATMOSPHERE
C
GOTO 200
C CALL LJUST(8,4,KARD,BUF)
C
CALL LOKUP(8,1,STANRD,BUF(1),ISTND,*6,*7)
6 IF(BUF(1).EQ.BLANK) GO TO 7
GO TO 200
7 WRITE(7,8) TITLE
WRITE(7,9) BUF
8 FORMAT('1',30A4)
9 FORMAT('O',9A8)
C
C READ UNITS CARD AND INTERPRET
C
C READ(5,5,END=200) BUF
C
C CALL LJUST(8,4,KARD,BUF)
C
CALL UNITIS(BUF(1),PTABL,6,IPUNIT,TPUNIT(1),2)
CALL UNITIS(BUF(2),TTABL,4,ITUNIT,TPUNIT(2),0)
CALL UNITIS(BUF(3),TTABL,4,ITUNIT,TPUNIT(3),ITUNIT)
IF(ITUNIT.EQ.0) ITUNIT=IDUNIT
IF(ITUNIT.NE.0) GO TO 30
ITUNIT=1
IDUNIT=1
30 CALL UNITIS(BUF(4),HTABL,6,IDUNIT,TPUNIT(4),5)
     GEOMET=.NOT.ATPOT(IDUNIT)
C READ IN DATA VALUES P-T-O-N. CONVERT TO INTERNAL UNITS.
C CHECK FOR MISSING VALUES OF DEWPOINT AND HEIGHT.
C
   TEMPK(1)=STANTP(1)
   HMISS(1)=TRUE
   DMISS(1)=TRUE
   SOMEHT=FALSE
   DO 50 N=2,80
   C READ(10,5,END=55) KARD
   C CALL LJUST(8,4,KARD,BUF)
   C READ(5,555,END=55) DUMMY
   555 FORMAT(4F8.0)
   DO 556 II=1,4
   IF (DUMMY(II).LE.-999.) DUMMY(II) = DEFARY(II)
   556 CONTINUE
   C IF(BUF(1).EQ.FINISH) GO TO 55
   C CALL FFAZN(KARD,1,8,4,DUMMY,DEFARY,KERR)
   PRESS(N)=DUMMY(1)*CTABT(1,ITUNIT)
   TEMPK(N)=(DUMMY(2)+CTTABL(2,ITUNIT))/CTTABL(1,ITUNIT)
   DEWPWT(N)=(DUMMY(3)+CTTABL(2,IDUNIT))/CTTABL(1,IDUNIT)
   GPH(N)=DUMMY(4)*CTTABL(1,IDUNIT)
   IF(GEOMET) GPH(N)=GPH(N)/(1.+GPH(N)/REARTH)
   TMPW(1)=TEMNK(N)
   GPHC(N)=GPH(N)
   DMSS(N)=DEWPWT(N).LT.0.
   PMISS(N)=PRESS(N).LE.0.
   HMSS(N)=GPH(N).LT.-1.E4
   IF(TEMPK(N).LT.0..OR.(PMISS(N).AND.HMSS(N))) GO TO 65
   IF(HMSS(N).OR.PMISS(N)) GO TO 50
   IF(SOMEHT) GO TO 50
   IFPTN=FALSE
   50 CONTINUE
   N=81
   WRITE(6,51)
   51 FORMAT(' P-T-O-N READING TERMINATED AFTER 79 ITEMS. ')
   55 NPTN=N-1
   IF(SOMEHT) GO TO 70
   WRITE(6,60)
60 FORMAT(' AT NO LEVEL IS BOTH HEIGHT AND PRESSURE GIVEN. CANNOT EVALUATE ATMOSPHERIC PROFILE. RUN ABORTED.')
STOP 650

65 WRITE(6,67) BUF
67 FORMAT(' INSUFFICIENT DATA ON CARD:','9A8,1''/ RUN ABORTED.')
STOP 650

70 CALL RAOBK(1,IPHTN,-1)
CALL RAOBK(IPHTN,NPTH,1)

C WORK DOWN TO OBTAIN VIRTUAL TEMPERATURES. BEFORE TOPMOST DEW POINT,
C MIXING RATIO IS ZERO. DEW POINT INTERPOLATED LINEARLY ACROSS GAPS
C W.R.T. DRY GPH, CONSTANT BELOW LOWEST INPUT DEW POINT.

DO 71 NN=1,NPTH
    N=NPTH-NN+1
    IF (.NOT.DMISS(N)) GO TO 72
    GAMMA(N)=1.401
71 CONTINUE
GO TO 80

N2=N
DOLD=DEWPNT(N2)
HOLD=GPHC(N2)
DO 77 NN=2,N2
    N=N2-NN+2
    DO 73 N3=2,N
        N4=N3+1
        IF (.NOT.DMISS(N4)) GO TO 74
73 CONTINUE
    DNEW=DOLD
    GO TO 75

74 DNEW=DEWPNT(N4)
75 HNEW=GPHC(N4)
    N4=N4+1
    DO 76 N5=N4,N
        D=((GPHC(N5)-HOLD)*DNEW+(HNEW-GPHC(N5))*DOLD)/(HNEW-HOLD)
        RTMIX=RATMIX(PRESS(N5),D)
        GAMMA(N5)=GMW(RTMIX)
        TMPMOL(N5)=VIRTMP(TEMPK(N5),RTMIX)
76 CONTINUE
    DOLD=DNEW
    HOLD=HNEW
77 CONTINUE

C RTMIX=RATMIX(PRESS(1),DOLD)
    GAMMA(1)=GMW(RTMIX)
    TMPMOL(1)=VIRTMP(TEMPK(1),RTMIX)
80 CALL RAOBK(1,IPHTN,-1)
    CALL RAOBK(IPHTN,NPTH,1)

C PRINT OUT WORKED UP VALUES IN ORIGINAL UNITS
C
WRITE(6,100) PTABL(IPUNIT),TTASL(ITUNIT),TTABL(IDUNIT),
+HTABL(IUNIT),HTABL(IUNIT),TTABL(IUNIT)
100 FORMAT('0',T17,'TEMPERATURE',T35,'HEIGHT',T49,'VIRTUAL',T60,
+ 'SOUND'/2X,'PRESSURE',T13,'KINETIC',T21,'DEW POINT',T32,'INPUT',
+ T39,'COMPUTED',T50,'TEMP.',T60,'SPEED'/5X,6A9,T62,'MPS')

GEOPOT=.NOT.GEOMET
DO 110 N=2,NPTH
   DUMMY(1)=PRESS(N)/CPTABL(IUNIT)
   DUMMY(2)=DEWPK(N)*CTTABL(1,ITUNIT)-CTTABL(2,ITUNIT)
   DUMMY(3)=DEWPK(N)*CTTABL(1,ITUNIT)-CTTABL(2,ITUNIT)
   H1PRNT=GPH(N)
   H2PRNT=GPHC(N)
   IF(GEOPOT) GO TO 105
   H1PRNT=H1PRNT/(1.-H1PRNT/REARTH)
   H2PRNT=H2PRNT/(1.-H2PRNT/REARTH)
105 DUMMY(4)=H1PRNT/CPTABL(IUNIT)
   DUMMY(5)=H2PRNT/CPTABL(IUNIT)
   DUMMY(6)=TMPHOL(N)*CTTABL(1,ITUNIT)-CTTABL(2,ITUNIT)
   DUMMY(7)=SORT(RHOMO*GAMMA(N)*TMPMOL(N))

C   CALL FN2A(KARD7,1,-8,4,7,DUMMY)
C   IF(HMISS(N)) KARD7(4)=BLANK
C   IF(DMISS(N)) KARD7(3)=BLANK
   WRITE(7,107) KARD7
   107   FORMAT(1X,7A9)

110 CONTINUE
   GO TO 300
C
C STANDARD ATMOSPHERE BASIS PREPARATION
C
200 GPHC(1)=STANHT(1)
   HMISS(1)=FALSE
   GAMMA(1)=STANGM(1)
   NPTH=1

C   WRITE(6,210)
   210 FORMAT('STANDARD ATMOSPHERE TABLE SELECTED.')
C
C MERGE IN STANDARD ATMOSPHERE (1976)
C
300 STANLO=GPHC(NPTH)+1000.
   IF(GPHC(NPTH).GT.STANHT(21)) RETURN
   DO 310 K=2,21
      IF(STANLO.LE.STANHT(K)) GO TO 320
   310   CONTINUE
   K=21
   320 L2=MINO(22-K,97-NPTH)
      IF(L2.LT.1) RETURN
      DO 350 L=1,L2
      NL=NPTH+L
      I10 CONTINUE
      GO TO 300
C
LK=L+K-1
GPNC(NL)=STANHT(LK)
TMPMOL(NL)=STANTP(LK)
GAMMA(NL)=STANGM(LK)
PMISS(NL)=TRUE
HMISS(NL)=FALSE
350 CONTINUE
CALL RAOBK(NPTH,NPTH+L2,1)
NPTH=NPTH+L2
RETURN
END
SUBROUTINE RAOBWI(IS)

IS is called to "work up a RAO"; i.e., to calculate
from the given temperatures and pressures the "thickness" (i.e., the
inght of the column of air between each pair of pressure levels) and
then calculate heights. Conversely, if given thickness, calculate
pressure drop.

SUBROUTINE RAOBWL(ILow,IHigh,IDIR)

COMMON /WTTH/ WPTH,PRESS(97),TMPMOL(97),GPHC(97),GAMMA(97)

C .. TMPMOL = 'MOLECULAR SCALE TEMPERATURE' = VIRTUAL TEMPERATURE
C .. GPH = GEOPOTENTIAL HEIGHT

COMMON /WKRAOB3/ MISS(97),PMISS(97)

LOGICAL MISS,PMISS

COMMON /ATMCON/ REARTH,GO,RSTAR,ROMO,ROGOMO

ROMO=RSTAR/MO AND ROGOMO=RSTAR/(GO*MO)

WHERE RSTAR=UNIVERSAL GAS CONSTANT=8.31432E3 JOULES / (KMOL-DEGK),
GO=9.80665M/SEC**2, AND MO=MEAN MOLECULAR WEIGHT OF STANDARD DRY
AIR (28.9644 KG/KMOL) (SEE U.S. STANDARD ATMOSPHERE 1976)

FIS(TAU)=(((TAU/5..1.)*TAU/4..1.)*TAU/3.*1.)*TAU/.+1.

F1A(TAU)=(EXP(TAU)-1./TAU

IF(ILow.GE.IHIGH) RETURN

IDIR=ISIGN(1,IDIR)

IF (IDIR.EQ.0) RETURN

KOFSET=0

IF(IDIR.LT.0) KOFSET=ILow-IHIGH

ISTOP=IHIGH-1

DO 9 NN=ILow,ISTOP

N=KOFSET+ISIGN(NN,IDIR)

TAU=ALOG(TMPMOL(N)/TMPMOL(N+IDIR))

IF(ABS(TAU).GT. .1) GO TO 2

FACTOR=TMPMOL(N)*FIS(TAU)

GO TO 3

2 FACTOR=TMPMOL(N)*FIA(TAU)

3 IF(PMISS(N+IDIR)) GO TO 5

THICK=ALOG(PRESS(N)/PRESS(N+IDIR))*FACTOR*ROGOMO

GPHC(N+IDIR)=GPHC(N)+THICK

GO TO 9

5 PRESS(N+IDIR)=PRESS(N)*EXP((GPHC(N)-GPHC(N+IDIR))/

FACTOR*ROGOMO)

9 CONTINUE
SUBROUTINE WINDIN
COMMON /PRINTS/ TITLE(30),TIMLBL
CHARACTER*4 TITLE
CHARACTER*8 TIMLBL
COMMON /PRINTC/ KTPSIG,CVRTIM
LOGICAL CVRTIM
COMMON /WINDS/ NWINDS,GPHW(80),DIR(80),TURN(79),SPEED(80)
COMMON /PUNITS/ PTABL,TTABL,HTABL,STABL,TITMBL,LTABL,FTABL
CHARACTER*8 PTABL(6),TTABL(4),HTABL(6),STABL(9),TIMTAB(2),LTABL(6)
CHARACTER*5 FTABL(5)
COMMON /CPUNIT/ CPTABL,CTTABL,CHTABL,CSTABL,CLTABL,CFTABL,
+ ATMPOT,ACPOT
REAL CPTABL(6),CTTABL(2,4),CHTABL(6),CSTABL(9),CLTABL(6)
REAL CFTABL(5)
LOGICAL ATMPOT(6),ACPOT(6)

C DEFAULT ON HEIGHT = GEOPOTENTIAL(ATMOSPHERE), GEOMETRIC(AIRCRAFT)
C INTERNAL PROGRAM UNITS HEIGHT GEOPOTENTIAL METERS(ATMOSPHERE),
C GEOMETRIC METERS ALL OTHER HEIGHTS AND LENGTHS,
C DEGREES KELVIN, PRESSURE KILOPASCALS (KPA), SPEED METERS PER SECOND.
COMMON /ATMCON/ REARTH,GO,RSTAR,ROMO,ROGOMO
C REARTH=RADIUS OF EARTH FOR CONVERSION GEOMETRIC TO GEOPOTENTIAL
C METERS. RSTAR=RSTAR/MO AND ROGOMO=RSTAR/(GO*MO)
C WHERE RSTAR=UNIVERSAL GAS CONSTANT=8.31432E3 JOULES / (KMOL-DEGK),
C GO=9.80665M/SEC**2, AND MO=MEAN MOLECULAR WEIGHT OF STANDARD DRY
C AIR (28.9644 KG/KMOL) (SEE U.S. STANDARD ATMOSPHERE 1976)

CHARACTER*8 NOWIND,FINISH
CHARACTER*3 TPUNIT(3)
CHARACTER*6 BUF(4),BLANK
REAL DUMMY(3),DEFARY(3)
LOGICAL GEOMET,NOCAP,TRUE,FALSE

DATA NOWIND/'NOIND'/,FINISH/ENDD/
DATA TPUNIT/'HEIGHT','DIRECT','SPEED'/
DATA BLANK/1/
DATA DEFARY/3*0./
DATA TRUE/.TRUE./,FALSE/.FALSE./
GPHW(1)=-9E3
DIR(1)=0.
SPEED(1)=0.

READ TITLE/NOWINDS CARD AND INTERPRET
READ(5,5,END=200) BUF
5 FORMAT(9A8)
BRANCH AROUND FOR NO WINDS
GOTO 200

CALL LJUST(8,3,KARD,BUF)
CALL LOKUP(8,1,NOWIND,BUF(1),1STND,*6,*7)
6 IF(BUF(1).EQ.BLANK) GO TO 7
GO TO 200
7 WRITE(7,8) TITLE
WRITE(7,9) BUF
8 FORMAT('1V,30A4)
9 FORMAT('10f,5X,9A8)

READ UNITS CARD AND INTERPRET
READ(5,5,END=200) BUF

CALL LJUST(8,3,KARD,BUF)
CALL UNITIS(BUF(1),HTABL,6,HUNIT,TPUNIT(1),5)
CALL UNITIS(BUF(3),STABL,9,ISUNIT,TPUNIT(3),3)
GEOMET.NOT. ATMPOT(IHUNIT)

READ IN DATA VALUES H-DIR-SPD. CONVERT TO INTERNAL UNITS.
COMPUTE TURN (RATE OF DIRECTION CHANGE PER METER)
OLDTRN=0.
NOCAP=FALSE
DO 40 N=2,80
READ(5,555,END=45) DUMMY
555 FORMAT(3F8.0)

CALL LJUST(8,3,KARD,BUF)
IF(BUF(1).EQ.FINISH) GO TO 45

CALL FFAN(KARD,1,8,3,DUMMY,DEFARY,KERR)
GPHW(N)=DUMMY(1)*CHTABL(HUNIT)
DIR(N)=DUMMY(2)
SPEED(N)=DUMMY(3)*CSTABL(ISUNIT)
IF(GEOMET) GPHW(N)=GPHW(N)/(1.+GPHW(N)/REARTH)
IF(SPEED(N).EQ.0.) GO TO 35
OLDTRN = ANO(ANO(DIR(N) - DIR(N-1), 360.) + 540., 360.) - 180. / 
+ (GPHW(N) - GPHW(N-1))
TURN(N-1) = OLDTRN
GO TO 40
35 TURN(N-1) = OLDTRN
OLDTRN = 0.
40 CONTINUE

N = 81
NOCAP = TRUE
WRITE(7, 41)
41 FORMAT(' N-DIR-SPD READING TERMINATED AFTER 79 ITEMS.')

45 NWINDS = N-1
IF(NWINDS.EQ.80) GO TO 50
NWINDS = NWINDS + 1
GPHW(NWINDS) = 130274.
SPEED(NWINDS) = 0.
DIR(NWINDS) = 0.
TURN(NWINDS-1) = OLDTRN

C WORK DOWN TURN AND DIR FOR THE CASE SPEED=0.
C
50 DO 60 NN = 2, NWINDS
N = NWINDS - NN + 2
IF(SPEED(N).EQ.0.) GO TO 60
IF(SPEED(N+1).EQ.0.) GO TO 55
OLDTRN = TURN(N-1)
GO TO 60
55 TURN(N-1) = OLDTRN
DIR(N) = DIR(N) - OLDTRN * (GPHW(N) - GPHW(N-1))
OLDTRN = 0.
60 CONTINUE

C PRINT OUT IN ORIGINAL UNITS
C
WRITE(7, 90) NTABL(INUNIT), STABL(ISUNIT)
90 FORMAT(' O', T4, 'HEIGHT', T16, 'DIR', T23, 'SPEED'/SX, A9, T16, 'DEG', T25, + A9)

NPRNT = NWINDS-1
IF(NOCAP) NPRNT = NPRNT + 1
DO 100 N = 2, NPRNT
HPRNT = GPHW(N)
IF(GEOMET) HPRNT = HPRNT / (1. - HPRNT / REARTH)
HPRNT = HPRNT * CHTABL(INUNIT)
SPRNT = SPEED(N) / CSTABL(ISUNIT)
WRITE (6, 95) HPRNT, DIR(N), SPRNT
95 FORMAT(1X, 3F9.0)
100 CONTINUE
RETURN
C NOWINDS SELECTED

C

200 NOWINDS=2
   GPHW(2)=130274.
   SPEED(2)=0.
   DIR(2)=0.
   TURN(1)=0.

C WRITE (6,210)
210 FORMAT('NOWINDS SELECTED.')

RETURN
END
**RAY TRACE ROUTINES - TIMPHI, ACMOVE, FILIMS, RAYORG, RAYTRK, RATES**

**ADVAMS, ARTUBE, RCRVIT, RECORD, RCPCL**

The ray tracing routines are responsible for emitting and tracking rays from aircraft to ground, under control of the main program SONBOM.

---

**SUBROUTINE ACMOVE(T, NODE, NODEC)**

ACMOVE interpolates aircraft track spline to current value of emission time. Computes and stores in common block the position and velocity of the aircraft, the local sound speed and wind, the airspeed and its rate of change, the Mach number and its rate of change, the climb and bank angle and the wing loading, the direction cosines of a "ray cone coordinate system" and their rates of change. Prints out the information on the aircraft position and motion, both in an airborne reference frame and a ground reference frame.

**INTEGER** SKEW(4)

**DIMENSION** RAC(1156,3), VXYZ(1156,3)

**DIMENSION** RO(3), RDOT(3), RDDOT(3), RDDDOT(3), RLWDOT(3), OMEGA(3)

**COMMON /ACSPOT/** TIME, XRO, YRO, ZRO, XDOT, YDOT, ZDOT, AIRSPD, ASPDOT, 
+ QC, UO, VO, CDOT, XMACH, XMACHDOT, XMACHD, COSMU, 
+ SINMU, EX(3,3), EKDOT(3,3), GLOAD, HEADIN, CLIMB, BANK, 
+ XDDOT, YDDOT, ZDDOT, XDDDDOT, YDDDDOT, ZDDDDOT

**EQUIVALENCE** (RO(1), XRO), (XDOT, RDOT(1)), (RDDDOT(1), XDDDOT), 
+ (RDDDOT(1), XDDDOT)

**COMMON /ATMCON/** REARTH, GO, RSTAR, ROMO, ROGOMO

**COMMON /FLIGHT/** NFIXES, TIMEAC(1156), YAC(1156), ZAC(1156), FMACH(1156), 
+ VX(1156), VY(1156), VZ(1156), CA(1156)

**EQUIVALENCE** (RAC(1,1), XAC(1))

**EQUIVALENCE** (VXYZ(1,1), VX(1))

**COMMON /ATMSPH/** GAM, C, U, V, DCDZ, DUDZ, DVDZ, D2CDZ2, D2UDZ2, D2VDZ2, RHO

**REAL** GAM, C, U, V

**COMMON /PRINTS/** TITLE(30), TIMLBL

**CHARACTER*4** TITLE

**CHARACTER*8** TIMLBL
COMMON /PRINTC/ KTPSIG, CVTIM
LOGICAL CVTIM

REAL CVTIM

COMMON /SPLINE/ NXP, SS(100,3), AS(100,3), BS(100,3), CS(100,3),
+ DS(100,3)
REAL SS, AS, BS, CS, DS
INTEGER NXP

COMMON /STSPIN/ ISTT
INTEGER ISTT

DATA DGPRAD/57.295780/
DATA SKEW/2,3,1,2/

TIME=T
C RETURN IF THE TIME IS BEYOND THE START OF THE SPLINE
C 334 IF (NODE.LE.0) RETURN

DT = TIME - TIMEAC(NODEC)
DT2 = DT**2
DT3 = DT2*DT
DT4 = DT3*DT

DO 10 K=1,3
  RD(K) = AS(NODE,K)/12.*DT4 + BS(NODE,K)/6. * DT
  1 + CS(NODE,K)/2.0 * DT2 + VXYZ(NODEC,K)**DT
  2 + RAC(NODEC,K)
RDOT(K) = AS(NODE,K)/3. * DT3 + BS(NODE,K)/2. * DT2
  1 + CS(NODE,K) * DT + VXYZ(NODEC,K)
RDOT(K) = AS(NODE,K) * DT2 + BS(NODE,K) * DT
  1 + CS(NODE,K)
RDDDOT(K) = 2.0 * AS(NODE,K) * DT + BS(NODE,K)
10 CONTINUE

CALL FNDLYR(ZRO,*250)
CALL AIR((ZRO))

CD=CD
UD=U
VD=V
CDOT=CD*ZDOT
UAS=U*DOT-U
VAS=YDOT-V
ASP=UAS**2+VAS**2
AIRSPD=SQRT(ASP+ZDOT**2)
ASP=SQRT(ASP)
RLWDOT(1)=RDOT(1)-DU2*ZDOT
RLWDOT(2)=RDOT(2)-DV2*ZDOT
RLWDOT(3)=RDDOT(3)
ASPDOT=(RLWDOT(1)*UAS+RLWDOT(2)*VAS+ZDOT*RLWDOT(3))/AIRSPD
XMACH=AIRSPD/C0
XMADOT=(ASPDOT*C0-AIRSPD*CDOT)/C0**2
IF (XMACH.GT.1.) GO TO 15
XMU=90.
XMUDOT=0.
SINU=1.
COSMU=0.
GO TO 20
15 SINU=.5/XMACH
COSMU=SQRT(1.-SINU**2)
XMU=GRPRAD*ASIN(SINU)
XMUDOT=-GRPRAD*XMDOT*SINU**2/COSMU
20 EK(1,1)=UAS/AIRSPD
EK(2,1)=VAS/AIRSPD
EK(3,1)=ZDOT/AIRSPD
EK(1,2)=VAS/ASPH
EK(2,2)=UAS/ASPH
EK(3,2)=0.
DO 30 K=1,3
   K1=SKW(K)
   K2=SKW(K+1)
   EK(K,3)=EK(K1,K2)*EK(K2,1)-EK(K2,K1)*EK(K1,1)
   OMEGA(K)=(RLWDOT(K1)*EK(K2,1)-RLWDOT(K2)*EK(K1,1))/AIRSPD
30 CONTINUE
FACT=(OMEGA(1)*EK(1,1)+OMEGA(2)*EK(2,1))/(EK(1,1)**2+EK(2,1)**2)
HLOAD=.0.
VLOAD=GF*EK(3,3)/(1.+20/REARTH)**2
DO 40 K=1,3
   OMEGA(K)=OMEGA(K)-FACT*EK(K,1)
   HLOAD=HLOAD+RDDOT(K)*EK(K,2)
   VLOAD=VLOAD+RDDOT(K)*EK(K,3)
40 CONTINUE
GLOAD=SQRT(HLOAD**2+VLOAD**2)/GO
BANK=GRPRAD*ATAN2(HLOAD,VLOAD)
HEAD=GRPRAD*ATAN2(-EK(3,1),-EK(2,1))=.180.
CLIM=GRPRAD*ASIN(EK(3,1))
DO 50 K=1,3
   K1=SKW(K)
   K2=SKW(K+1)
   OM1=OMEGA(K1)
   OM2=OMEGA(K2)
   DO 50 L=1,3
      EKDOT(K,L)=EK(K1,L)*OM2-EK(K2,L)*OM1
50 CONTINUE
C WRITE(7,60) TITLE
60 FORMAT('1,30A4)
C TPRINT=TIMCMVR(TIME),2)
C WRITE(7,65) TIMLBL,TPRINT,XRO,YRO,ZRO,XMACH,GLOAD,BANK
65 FORMAT('OARMAUT MANEUVER DATA'/I0,T4,'TIME',I0,'X',I6,'Y',I6,'Z',I6
+
,'2',I4,'MACH',I4,'LOAD',I4,'BANK',I4,'A8',I15,'MET',I25,'MET'
+',I35,'MET',I46,'NO.',I55,'G',I5,'S',I64,'DEGS.'/1X,4F10.0,2F10.5,F10.1 +
)
HEADG=OGPRAD*ATAN2(-XDOT,-YDOT)+180.
GNTSP=SQRT(XDOT**2+YDOT**2+ZDOT**2)
GCLMB=OGPRAD*ASIN(ZDOT/GNTSP)
C        WRITE(7,70)AIRSPD,UAS,VAS,ZDOT,CLIMB,HEADIN,GNTSPD,XDOT,YDOT,ZDOT,
C    +GCLMB,HEADG
C 70 FORMAT('0',T11,10(' '),T25,'SPEED MPS',T41,10(' '),1X,5(' '),ZX,'A
C  +NGLE',3X,5(' ')T14,'TOTAL',T23,'X-COMP',T33,'Y-COMP',T43,'Z-COMP'
C  +',T54,'CLIMB',T64,'HEADING'/ AIR',T11,4F10.0,F10.2,F10.1/GROUND
C  +',T11,4F10.0,F10.2,F10.1)
RETURN
200 WRITE(6,210)T,TIMEAC(T),TIMEAC(MFIXES)
210 FORMAT(' IN CALL TO ACMOVE, TIME',F10.1,' AIRCRAFT IS OUTSIDE RANGE',
C  +F10.1,' TO',F10.1)
RETURN
250 WRITE(6,260)T,ZRO
260 FORMAT(' IN CALL TO ACMOVE AT TIME',F10.1,' AIRCRAFT IS AT ALT
C  +ITUDE Z=',F10.2,' METERS AND OUTSIDE ATMOSPHERE TABLE.'
STOP 600
END
SUBROUTINE FILIMSC(*)

C FILIMS, GIVEN THE INFORMATION FROM THE ACMOVE SUBROUTINE, AND THE WIN
C VELOCITY AND SOUND SPEED AT THE GROUND, COMPUTES THE LIMITS OF PHI AN
C AT THE ADMITTANCE ELLIPSE FOR THE GROUND LEVEL. PRINTS OUT THE LIMITI
C PHI ANGLES FOR THE ARCS INSIDE THE ADMITTANCE ELLIPSE, IF ANY.
C ALTERNATE RETURN TAKEN IF RAYS DO NOT TOUCH THE GROUND OR AIRCRAFT I
C SUBSONIC.

DIMENSION ZRO(5),TRN0(5)

COMMON /ACSPOT/ TIME,XRO,YRO,ZRO,XDOT,YDOT,ZDOT,AIRSPD,ASPDOT,
+ CO,UD,VO,CDOT,XMACH,XMUADOT,XMUADOT,COSMU,
+ SINMU,EK(3,3),EKDOT(3,3),GLOAD,HEADIN,CLMS,BANK,
+ XDDOT,YDDOT,ZDDOT,XDDDOT,YDDDOT,ZDDDOT
EQUIVALENCE (SINGAM,EK(3,1)),(COSGAM,EK(3,3))

COMMON /GROUND/ GLAYER,ZGRNO,CGRNO,UGRND,VGRND,REFLFC
INTEGER GLAYER

COMMON /RAYLIM/ NLIMS,BEG(2),END(2)
DATA DGPRAD/57.295780/,TWOPI/6.28318531/

IF (XMACH.LT.1.) GO TO 102
UOMG=UD-UGRND
VOMG=VO-VGRND
ALPHA1=1.+SINMU*(UOMG*EK(1,1)+VOMG*EK(2,1))/CO
ALPHA2=COSMU*(UOMG*EK(1,2)+VOMG*EK(2,2))/CO
ALPHA3=COSMU*(UOMG*EK(1,3)+VOMG*EK(2,3))/CO
AO=ALPHA1**2+5.(ALPHA2**2+ALPHA3**2)
A1=2.*ALPHA1*ALPHA3
A2=2.*ALPHA1*ALPHA2
A3=-.5*(ALPHA3**2-ALPHA2**2)
A4=ALPHA2*ALPHA3
SINMU2=SINMU**2
COSMU2=COSMU**2
CGFACT=(CGRND/CO)**2
CSGM2=COSGAM**2
AO=AO*CGFACT*(SINMU2*CSGM2+(1.-.5*CSGM2)*COSMU2)
A1=A1*CGFACT**2.*SINGAM*COSGAM*SINMU*COSMU
A3=A3*CGFACT**.5*(CSGM2)*COSMU
A3=A3**2+A4**2
DFULM=SQRT(A1**2+A2**2)+A34
CPPM=DFULM*3.*A34
EPS=5E.-6*DFULM
PHI=-90./DGPRAD
FO=AO-A2-A3
PHI=PHI
PHIBEG=PHI
KZRO=1
IF (DFULIM.LT.ABS(AO).OR.DFULIM.EQ.0.) GO TO 100
5 IF (PHIO.GT.PHIBEG+TWOPHI) GO TO 100
SINPHI=SIN(PHI)
COSPHI=COS(PHI)
TWOPHI=PHI+PHI
COS2FI=COS(TWOPHI)
SIN2FI=SIN(TWOPHI)
F=AO*A1*COSPHI+A2*SINPHI+A3*COS2FI+A4*SIN2FI
IF (ABS(F).LT.EPS) GO TO 25
IF (F*FO.LE.0.) GO TO 10
C CASE NO ZERO CROSSING. ADVANCE PHI
FPR=A1*SINPHI-A2*COSPHI-2.*(A3*SIN2FI-A4*COS2FI)
FPPMX=SIGN(CPMX,F)
DPHII=FPR/FPPMX
DPHI2=SQRT(FPR**2+2.*F*FPPMX)/CPMX
PHIO=PHI
FO=F
PHI=PHI+DPHI
GO TO 5
C CASE ZERO IS CROSSED. LOCATE ZERO BY HALVES.
10 PHI1=PHI
FHI=F
15 PHI=.5*(PHI1+PHI0)
SINPHI=SIN(PHI)
COSPHI=COS(PHI)
TWOPHI=PHI+PHI
COS2FI=COS(TWOPHI)
SIN2FI=SIN(TWOPHI)
F=AO*A1*COSPHI+A2*SINPHI+A3*COS2FI+A4*SIN2FI
IF (ABS(F).LT.EPS) GO TO 25
IF (F*FHI.GT.0.) GO TO 20
PHIO=PHI
FO=F
GO TO 15
20 PHI1=PHI
FHI=F
GO TO 15
25 FPR=A1*SINPHI-A2*COSPHI-2.*(A3*SIN2FI-A4*COS2FI)
DPHI=ABS(FPR/CPMX)
DF=.5*ABS(FPR)*DPHI
IF (DF.LT.EPS) GO TO 30
MULT=1
SGN=SIGN(1.,FPR)
GO TO 50
30 FPPR=A1*COSPHI-A2*SINPHI-4.*(A3*COS2FI-A4*SIN2FI)
DPHI=ABS(2.*FPPR)/(CPMX**4.*A34)
DF=ABS(FPPR)*DPHI/6.
IF (DF.LT.EPS) GO TO 35
MULT=2
SGN=SIGN(1.,FP3R)
GO TO 50
35 FP3R=A1*SINPNI-A2*COSPNI*8.*(A3*SIN2FI-A4*COS2FI)
DPHI=ABS(3.*FP3R)/(CPPNX+12.*A34)
DF=ABS(FP3R)*DPHI*DPHI/24.
IF (DF.LT.EPS) GO TO 40
MULT=5
SGN=SIGN(1.,FP3R)
GO TO 50
40 FP4R=A1*COSPHI+A2*SINPHI+16.*(A3*COS2FI+A4*SIN2FI)
MULT=4
DPHI=TWOPHI
DO 55 K=1,MULT
ZRO(KZRO)=PHI
TRND(KZRO)=SGN
KZRO=KZRO/2
IF (KZRO.GT.5) GO TO 100
55 CONTINUE
PHI=PHI+DPHI
PHI=PH1O
TWOPHI=PHI+PHI
GO TO 5
100 IF (KZRO.GT.1) GO TO 110
IF (FO.GE.0) GO TO 105
102 NLIMS=0
GO TO 130
105 NLIMS=1
BEG(1)=-90.
END(1)=270.
GO TO 130
110 KZRO=KZRO-1
IF (MOD(KZRO,2).EQ.1) GO TO 115
KZRO=KZRO+1
ZRO(KZRO)=ZRO(1)+TWOPHI
TRND(KZRO)=TRND(1)
115 NLIMS=(KZRO-1)/2
L=1
IF (TRND(1).LT.0.) L=2
DO 120 N=1,NLIMS
BEG(N)=ZRO(N*L-2)*DGPRAD
END(N)=ZRO(N*L-1)*DGPRAD
120 CONTINUE
130 IF (NLIMS.GT.0) GO TO 150
C WRITE(7,145)
145 FORMAT(10 RAYS WILL NOT TOUCH GROUND OR AIRCRAFT IS SUBSONIC.)
RETURN
150 CONTINUE
C 150 WRITE(7,155) NLIMS
155 FORMAT(10',T10,12,' PHI-ANGLE INTERVALS:')
DO 165 N=1,NLIMS
BEG(N)=AMOD(AMOD(BEG(N),360.)+450.,360.)*-90.
END1=AMOD(AMOD(END(N),360.),450.,360.)*90.

WRITE(7,160) N,BEG1,END1

160 FORMAT('INTERVAL',12,' FROM','F7.2',' DEGREES TO','F7.2,'DEGREES."

165 CONTINUE
RETURN
END
SUBROUTINE RAYORG(*)

RAYORG, FOR EACH EMISSION TIME AND FOR EACH VALUE OF PHI LYING WITHIN
THE ADMITTANCE ELLIPSE, COMPUTES THE INITIAL VALUES OF POSITION, RAY
NORMALS, "FREQUENCIES", AND THEIR RATES OF CHANGE. SETS CURRENT TIME
EQUAL TO EMISSION TIME. THE RATES OF CHANGE ARE WITH RESPECT TO NOT
ONLY CURRENT TIME, BUT ALSO THE RAY PARAMETERS OF PHI ANGLE AND OF
EMISSION TIME. IF RAY TRACE PRINTING IS SELECTED, PRINTS OUT THE INIT
RAY TRACE VALUES.

COMMON /ACIDNT/ IDENT
CHARACTER*8 IDENT

COMMON /ACWEIG/ ACWT,ACL

COMMON /RAYVAR/ ZDIR,PKK,RTPAAO,ATTEN,SIGMA,X,Y,Z,DAGE,XF,GF,ZF,
+ XT,YT,XTS,YS,SZ,XSS,SSS,YSSS,SSSS,PS,S,
+ PR,P3S,XFS,YFS,XTS,YTS,ZTPS,XFTZ,YFTZ,ZFTZ,
+ ZTA,ZTA,P3FTA,P3FA,P3TA,AREA,DAGDS
REAL SIGMA,X,Y,Z,DAGE,XF,GF,ZF,XT,YT,XTS
REAL XX(3),XXF(3),XXFT(3)
REAL RENORM
REAL XXX(3),XXFSS(2),XXSTS(2),XXFTSTS(3),XXSSS(3)
EQUIVALENCE (XX(1),X),(XXF(1),XF),(XXFT(1),XF)
EQUIVALENCE (XXF(1),XF),(XXSTS(1),XTS),(XXFTSTS(1),XFTSTS)
EQUIVALENCE (XXSSS(1),XXSSS)

COMMON /RAYNIT/ KGMH,NDCRVS,NUCRVS,IPOWN,T0,PHIO,XO,YO,ZO,
+ P10,P20,P30,OMEGA,DELTA0,PI10,P210,P310,OMEGAT,XSO,
+ YSO,2SO,P3SO,ROHO,PCONST,NAGES,AGES(20)
INTEGER KGMH,NDCRVS,NUCRVS,IPOWN
REAL XK(3),PK(3),PKFO(3),XKTO(3),PKTO(3),XKSO(3)
EQUIVALENCE (XK(1),XO),(PK(1),PO),(PKTO(1),PKTO), (XKTO(1),XTO)
EQUIVALENCE (PKTO(1),PKTO), (XKSO(1),XSO)

COMMON /ATMCON/ REARTH,G0,STAR,RMO,ROGOMO

COMMON /ACSPOT/ TIME,XXO,YRO,ZRO,XXDOT,YYDOT,ZDDOT,AIRSPD,ASPDOT,
+ CO,UO,VO,CDOT,XMACH,XMADOT,XMU,XMUDOT,COSMU,
+ SINC,EC(3,3),EKDOT(3,3),GLOAD,HEADDN,CLMB,BANK,
+ YDDOT,YYDDOT,ZDDDOT,YDDDDOT,ZDDDDOT
REAL XXKO(3),XXDOT(3)
EQUIVALENCE (XXKO(1),XXO),(XXDOT(1),XXDOT)

COMMON /ATMSPH/ GAM,C,U,V,DDDO,DDZ,D2V,D2DD2,D2UD22,D2DD22,RHOP
REAL GAM,C,U,V
COMMON /CLASSES/ CNAI4ES(30)
CHARACTER*8 CNAI4ES

COMMON /CLASSS/ NRCURV(2,2),TYPRAY(3,2,2),DIRECT,LOFT,UP,DOWN
LOGICAL TYPRAY,DIRECT,LOFT,UP,DOWN

COMMON /PRINTS/ TITLE(30),TMLBL
CHARACTER*4 TITLE
CHARACTER*8 TMLBL

COMMON /PRINTC/ KTPSIG,CVRTIM
LOGICAL CVRTIM
REAL CVRTIM

COMMON /RYCTRL/ NORAYS,STND,UL,UR,LL,LR,PRTRAY,TIMBEG,DELTIM,
+ NTIMS,PHIBEG,DELPHI,NPHIS
LOGICAL NORAYS,STND,UL,UR,LL,LR,PRTRAY

COMMON /RPOSN/ NPTCR,CPOSN,RT(200),RXYZ(200,3),RAGE(200),
+ RPFACT(200),RVLIFT,REMEM
REAL RT,RXYZ,RAGE,RPFACT
INTEGER NPTCR,CPOSN
LOGICAL REMEM

DATA DGPRAD/57.295780/

NPTCR = 0
NAGES=1
DAGE=0.0
MEDH1=1
COSH=0=0.0
SINH=0.0
RENM=0.0
DO 10 K=1,3
XK0(K)=XK0(K)
EH=0.0
EH=0.0
EH=0.0
PK0(K)=PK0(K)
PKT0(K)=PKT0(K)
RENORM=RENORM+ (PK0(K)
10 CONTINUE
RENORM=SINWWURSRT(RENORM)

CALL FNOLYR(ZO,*20)
20 CALL AIR((Z0))

RHO0=RHO
C=
PCONST=AIRSPD*2*SQR(.5*RHO0)
PCONST=AIRSPD*SQR(0.5*RHO0)
DELTA=AIRSPD
CS0=0*SINWW
DO 25 K=1,3
   PKO(K)=PKO(K)/RENUM
   XXSO(K)=CSQD*PKO(K)
   XKTO(K)=XXD0(K)*-XXSO(K)
25 CONTINUE
XSO=XSO+UO
YSO=YSO+V0
XTO=XTO+U0
YTO=YTO+V0
RTPAAO=SQRT(P10**2+P20**2)
DELTAO=CSQD*RTPAAO**2
OMEGA=DELTAO*U*P10*V*P20
OMEGAF=M*U*P10*V*P20
P300=-DDZ*DELTAO*U*P10*V*P20
P30=P300
P3T=P3T0=P30
IUP0UN=1
IF (P30.LT.0.) IUP0UN=2
IF (IUP0UN.EQ.1.AND..NOT.UP) RETURN
IF (IUP0UN.EQ.2.AND..NOT.DOWN) RETURN
SIGMA=0
ZDIR=SIGN(1.,P30)
DO 50 K=1,3
   XX(K)=XXO(K)
   XXF(K)=0.0
   XKT(K)=XXTO(K)
50 CONTINUE
P3F=P3F0
P3T=P3T0
CALL RATES(*100,*100)
AREA=0.0
ATTEN=1.
RVLFT=(ACWT*GLOAD*CO*COSMU*COS((PHI0-BANK)/DGPRAD))/
       (RHO0*SINMU*AIRSPD**2))
IF (.NOT.PRTRAY) RETURN
WRITE(7,60) TITLE
60 FORMAT(1',30A4)
WRITE(7,65) TPRINT,TIMLBL,FIPRNT,P10,P20,AZIM
65 FORMAT(1',01,T20,'DATA FOR RAY DEPARTING AIRCRAFT TIME=',F10.0,'TX=',F10.0,'THETA=',F7.2,' DEGREES.'
    'PHI ANGLE=',F7.2,' DEGREES.'
    'NORMAL AZIMUTH=',F6.0,' DEGREES.'
)
WRITE(7,70) TIMLBL,TPRINT,X,Y,Z,P3,ELEV,C,DS,AREA
70 FORMAT(1',01,T5,'SIGMA',T18,'X',T28,'Y',T39,'Z',T45,'P3',T53,'PHASE'
    '+',T65,'C',T73,'O2/DS',T81,'AREA',T91,'AGE/T5',A8,T17,'MET',T27,
    '+',T38,'MET',T4,'ELEV',T63,'M/SEC',T73,'M/SEC',T79,'M**2/SEC',
    'IMETI',T38,'MET?',T54,ELEV,T63,MSEC,T79,'M*2/SEC',
    'IMETI',T38,'MET?',T54,ELEV,T63,MSEC,T79,'M*2/SEC',
    'IMETI',T38,'MET?',T54,ELEV,T63,MSEC,T79,'M**2/SEC',
    'IMETI',T38,'MET?',T54,ELEV,T63,MSEC,T79,'M**2/SEC',
    'IMETI',T38,'MET?'.
C  +TB9,'MET**.5'/
C  +1X,F10.1,3F10.0,210.3,F6.1,2F10.1,2G10.4)
RETURN
100 WRITE(6,101)
101 FORMAT(' IMPROPER RETURN FROM RATES IN RAYORG')
RETURN
END
SUBROUTINE RAYTRK(GFLAG, RCBLG, CFLAG, *)


PARAMETERS:

INPUT:

OUTPUT:

LOGICAL RCBLG

COMMON /ATMCON/ REARTH,GO,RSTAR,ROMO,ROGOMO

COMMON /ACSPOT/ TIME,XRO,YRO,ZRO,XDOT,YDOT,ZDOT,AIRSPD, ASPDOT, + CO,UC,VO,COT,CMACH,CMADOT,CMU,CMUDOT,CSMU, + SIMMU,EK(3,3),EKDOT(3,3),GLOAD,HEADIN,CLIMB,BANK, + XDOT,YDOT,ZDOT,XDDOT,YDDOT,ZDDDOT

COMMON /RAYVAR/ ZDIR,PKK,RTPAAO,ATTEN,SIGMA,X,Y,Z,DAGE, XF,YF,ZF, + XT,YT, ZT, XS,YS,ZS, XSS,XSSS, YSS,YSSS, ZSS, ZSSS, P3, + P3F,P3T,P3S,XFS,YFS,XTS,YTS,ZFT3,XFT2,YFT2,ZFT2, + ZFA,ZTA,P3FT2,PSFA,PSSTA,AREA,DAGDS

REAL SIGMA,X,Y,Z,DAGE, XF,YF,ZF, XT, YT, ZT
REAL XK(3),XKF(3),XKT(3)
REAL XKS(3),XKFS(2),XKTS(2),XKFTZ(3),XKSSS(3)
EQUIVALENCE (XK(1),X),(XKF(l),XF),(XKT(l),XT),(XKS(l),XS)
EQUIVALENCE (XKFS(1),XFS),(XKTS(l),XTS),(XKFTZ(1),XFTZ)
EQUIVALENCE (XKSSS(1),XSSS),(XKSSS(1),XSSS)

COMMON /RAYHLD/ HSIGMA,HX,HY,HZ,DAGE, HXF,HYF,HZF, HXT,HYT,HZT, + HXS,HYS,HZS,HXSS,HYSS,HZSS,HXSSS, HYSSS,HZSSS, + HP3,HP3F,HP3T,HP3S,XFS,YFS,HXTS, HYTS, HZFT3, + HXFT2,HYFT2,HZFT2,HZFA,HZTA,HP3FT2,HP3FA,HP3TA, + AREA,DAGDS

REAL HSIGMA,HX,HY,HZ,DAGE, HXF,HYF,HZF, HXT,HYT,HZT
REAL HX(3),HXF(3),HXT(3)
REAL HBVR(11),HBND(11),DELZ
REAL HXKS(3),HXKFS(2),HXKTS(2),HXKFTZ(3),HXKSS(3),HXKSSS(3)
REAL HOLDVR(28),HOLDHD(28)
EQUIVALENCE (HOLDVR(1),XS),(HOLDHD(1),HXS)
EQUIVALENCE (HBVR(1),SIGMA),(HBND(1),HSIGMA)
EQUIVALENCE (HXK(1),HX),(HXKFS(1),HXF),(HXKTS(1),HXT),(HXKFTZ(1),HXFTZ)
EQUIVALENCE (HXKSS(1),HXSS),(HXKSSS(1),HXS)

COMMON /CLASSES/ CNAMES(30)
CHARACTER*8 CNAMES

COMMON /CLASSS/ WRCURV(2,2),TYPRAY(3,2,2),DIRECT,LOFT,UP,DOWN
LOGICAL TYPRAY,DIRECT,LOFT,UP,DOWN

COMMON /RYCTRL/ NORAYS,STND,UL,UR,LL,LR,PRTRAY,TIMBEG,DELTIM,
+ NTIMS,PHIBEG,DELPHI,NPHIS
LOGICAL NORAYS,STND,UL,UR,LL,LR,PRTRAY

COMMON /RAYNIT/ KGMM,NDCRVS,NUCRVS,IUPDN,TO,PM10,XO,YO,ZO,
+ P10,P20,P30,OMEGA,DELTAO,P1FO,P2FO,P3FO,OMEAGAF,
+ XTO,YTO,ZTO,P1TO,P2TO,P3TO,OMEGAT,XSO,YSO,ZSO,
+ P3SO,RHOO,PCONST,NAGES,AGES(ZO)
INTEGER KGMM,NDCRVS,NUCRVS,IUPDN
REAL PK(2),PKF(2),PKT(2)
EQUIVALENCE (PK(1),PIO),(PKF(1),P1FO),(PKT(1),P1TO)

COMMON /ATMSPH/ GAM,C,U,V,DCDZ,UDUZ,UDVZ,DZD2Z,D2UD2Z,D2VD2Z,RHO
REAL GAM,C,U,V

COMMON /GROUND/ GLAYER,ZGRND,CGRNO,UGRNO,VGRND,REFLFC
INTEGER GLAYER

COMMON /LYRDEF/ NLAYER,GMZA(200),IIDPTH(200),INOWNO(200),
+ LYRPRT(200),KLAYER,ZTOP,ZBOT
LOGICAL LYRPRT
INTEGER TIMCVR,TPRINT

COMMON /CAUSTC/ NUMC,TRACE,CT(360),CPHI(360),CXYZ(360,3)
REAL CXYZ
LOGICAL TRACE

COMMON /RPOSN/ NPTR,CPOSN,RT(200),RXYZ(200,3),RAGE(200),
+ RPFAC(200),RVLIFT,REMEM
REAL RT,RXYZ,RAGE,RPFAC
INTEGER NPTR,CPOSN
LOGICAL REMEM

LOGICAL PTHDR
COMMON /JJJ/ PTHDR
LOGICAL CFLAG,GFLAG
REAL IFACT
IF (GFLAG) THEN
  IFACT = 17.
ELSE
  IFACT = 1.5
ENDIF

PTHDR = .TRUE.
RCBLG = .FALSE.
CFLAG = .FALSE.

NDCRVS=0
NUCRVS=0
KDMIN=1

1 CONTINUE
DO 2 L=1,28
  HOLDH0(L)=HOLDVR(L)
2 CONTINUE
DO 3 L=1,11
  H8HD(L)=H8VR(L)
3 CONTINUE
TDLSIG=.30

C - CHECK FOR RAY CURVING UP
C
IF (ZDIR.GT.0.) RETURN 1

IF (ZS*ZSS*TDLSIG.GT.0.) TDLSIG=AMAX1(0.,-ZS/ZSS)
DELZ=AMAX1(-50.0,ZBOT-Z,AMIN1(-1.0,(ZS+.50*ZSS*TDLSIG)*TDLSIG))
IF (DELZ.LT.0.0) GO TO 15
LPRNT=KLAYER
KLAYER=KLAYER-1

C - CHECK FOR RAY AT GROUND - 1000 FT
C
IF (KLAYER.LE.0) GO TO 451

C - CHECK FOR RAY AT GROUND
C
IF (KLAYER.EQ.GLAYER-1) GOTO 450

GO TO 400

15 Z=HZ+OELZ
IF (Z.LT.ZGRND - IFACT*304.8) GOTO 451

CALL RATES(*320,*300)
CALL ADVANS(CFLAG)

C- IF A CAUSTIC IS ENCOUNTERED BELOW 500 FT AND WE ARE LOOKING FOR
C- THE FOCUS THEN CONTINUE ELSE GO ON TO THE NEXT RAY
C
IF (CFLAG.AND.Z.LE.ZGRND+304.80) THEN
  IF (.NOT.TRACE) THEN
    Z = Z
  ELSE
    RETURN 1
  ENDIF
ENDIF
GO TO 1

300 CALL RCRVIT
305 CALL RATES(*320,*420)
320 CALL ADVANS(CFLAG)

IF (CFLAG.AND.Z.LE.ZGRND+304.80) THEN
  IF (.NOT.TRACE) THEN
    Z = Z
  ELSE
    RETURN 1
  ENDIF
ENDIF

C- RECURVATURE BEFORE -1000 FT
C
350 IF (Z.LT.ZGRND-IFACT*304.8) GO TO 451
C
C- RECURVATURE BETWEEN 0 & -1000 FT
C
  IF (Z.LE.ZGRND) THEN
    IF (Z.EQ.ZGRND) THEN
TPRINT=TIMCVR(SIGMA,2)
ELEV=THELEV(0.)
C IF (PRTRAY)WRITE(7,60) TPRINT,X,Y,Z,P3,ELEV,C,ZS,AREA,DAGE
NUCRVS=NUCRVS+1
C CALL RCSPCL('GROUND 1',SIGMA,XK,P3,XKF,'**T',XKS,AREA)
 Call RECORD(*600)
ENDIF
600 RCBLG = .TRUE.
RETURN
ENDIF
C - RECURVATURE ABOVE THE GROUND
C
NUCRVS=NUCRVS+1
C CALL RCSPCL('RAY LOW 1',SIGMA,XK,P3,XKF,XKT,XKS,AREA)
C WRITE(7,355)
C 355 FORMAT(' RAY RECURVING UPWARD; WILL NEVER TOUCH GROUND.')
RETURN 1

IF (Z-ZGRND.GE.1.) RETURN
GO TO (370,380,380),KG4H
370 IF (LOFT) GO TO 480
RETURN
380 IF (NUCRVS.GE.NRCURV(3,UPOWN,KGMH-1)) RETURN
GO TO 480
400 ZBOT=GMZA(KLAYER)
ZTOP=GMZA(KLAYER+1)
CALL RATES(*410,*420)
FCTJMP=(P3S-HP3S)/ZS
P3F=HP3F+F*FCTJMP
P3T=HP3T+ZT*FCTJMP
C IF (.NOT.(LYRPRT(LPRNT).AND.PRTRAY)) GO TO 1
C TPRINT=TIMCVR(SIGMA,2)
C ELEV=THELEV(0.)
C WRITE(7,60) TPRINT,X,Y,Z,P3,ELEV,C,ZS,AREA,DAGE
C 60 FORMAT(1X,9(1D10.4))
GO TO 1
410 IF (ZDIR.LT.0.) GO TO 350
KAYER=KAYER-1
ZBOT=GMZA(KLAYER)
ZTOP=GMZA(KLAYER+1)
GO TO 305
420 WRITE(7,421)
421 FORMAT(' IMPROPER RETURN FROM RATES IN RAYTRK')
RETURN
C
C - OUTPUT GROUND
C
450 TPRINT=TIMCVR(SIGMA,2)
ELEV=PHELEV(0.)
C IF (PRTRAY)WRITE(7,60) TPRINT,X,Y,Z,P3,ELEV,C,ZS,AREA,DAGE
    NUCRVS=NUCRVS+1
C CALL RCSPCL(' GROUND',SIGMA,XX,P3,XXF,XXT,XXS,AREA)
    CALL RECORD(*452)

452 GOTO 400

C OUTPUT GROUND - 1000 FT
C
451 TPRINT=TIMCVR(SIGMA,2)
    ELEV=PHELEV(0.)
C IF (PRTRAY)WRITE(7,60) TPRINT,X,Y,Z,P3,ELEV,C,ZS,AREA,DAGE
C CALL RCSPCL(' G-1000 ',SIGMA,XX,P3,XXF,XXT,XXS,AREA)
    RETURN
C
C...............................................................
C- MUST MAKE PROVISION FOR TRACING OF OTHER THEN G RAYS HERE
C- IF THAT IS TO BE ADDED
C...............................................................
C
480 ZDIR=1.
    KLAYER=1
    IF (ZS.EQ.0.) GO TO 495
    FCTJMP=2.*HP3F/ZS
    P3F=HP3F+FCTJMP*HZF
    P3T=HP3T+FCTJMP*HZT
    ZF=HZF
    ZT=HZT
    AREA=HAREA
    ATTEN=ATTEN*REFLFC
    CALL RATES(*495,*420)

    IF (PRTRAY)WRITE(7,490)
490 FORMAT(' *************** REFLECTING FROM GROUND ***************')
    GO TO 1
495 IF (PRTRAY)WRITE (6,496)
496 FORMAT(' **********RAY TANGENT AT GROUND LEVEL**********')
    GO TO 1
500 IF (PRTRAY)WRITE(6,505)
505 FORMAT(' STOPPING AT TOP OF ATMOSPHERE.')
    RETURN
END
SUBROUTINE RATES(*,*)

RATES computes the local rate of change of the ray position, the ray normals, and the associated derivatives with respect to the ray parameters PHI and EMISSION TIME.

COMMON /RAYNIT/ KGMH,NDCRVS,NUCRVS,IUPDWN,T0,P10,X0,Y0,Z0,
+ P10,P20,P30,OMEGA,DELTA0,P1FO,P2FO,P3FO,OMEGAF,
+ XTO,YTO,ZTO,P1TO,P2TO,P3TO,OMEGAT,XSO,YSO,ZSO,
+ PSSO,RHOQ,PCONST,MAGES,AGES(20)
INTEGER KGMH,NDCRVS,NUCRVS,IUPDWN
REAL PK(2),PKF(2),PKT(2)
EQUIVALENCE (PKC1),P1O),CPKF(1),PlFO),(PKT(l),PlTO)

COMMON /RAYVAR/ ZDIR,PKK,RTPAAO,ATTEN,SIGMA,X,Y,Z,DAGE,XY,FZ,
+ XT,YT,XT,YS,YS,XSS,YSS,YSSS,ZSSS,P3,
+ P3F,P3T,P3S,XFS,YFS,XTS,YTS,2FTP3,XFTZ,YFTZ,ZFTZ,
+ ZFA,ZTA,P3FTZ,P3FA,P3TA,AREA,DAGDS
REAL XKS(3),XKFS(2),XKTS(2),XKFTZC3),XKSS(3),XKSSS(3)
REAL SIGMA,X,Y,Z,DAGE,XY,FZ,XT,YT,XT
REAL XK(3),XKF(3),XKTC3)
EQUIVALENCE (XKC1),X),(XKF(l),XF),(XKT(l),XT),(XKS(l),XS)
EQUIVALENCE (XKFS(l),XFS),(XKTSC1),XTS),CXKFTZ(1),XFTZ)
EQUIVALENCE (XKSS(l),)CSS),(XKSSS(l),XSSS)
COMMON /ATMSPH/ GAM,C,U,V,DCDZ,DUDZ,DVDz,D2CMZ2,D2UDZ2,D2VDZ2,RHO
C REAL UK(2) GARBAGE
REAL UK(2),UKDZ(2),D2UKDZ(2)
REAL GAM,C,U,V
REAL DEDELTA,RTPKK
EQUIVALENCE (UK(1),U),(UKDZ(1),UDZ2),(D2UKDZ(1),D2UDZ2)

CALL AIRMZ

DDELTAS=(OMEGA)-U*(P10)-V*(P20)
DELTAS=DEDELTA
RTPKK=DEDELTA/C
PKK=RTPKK**2
IF (RTPKK.LT.RTPAAO) RETURN 2
CSCOD=C*C/ODELTA
P3=SQRT((RTPKK-RTPAAO)*(RTPKK+RTPAAO))
P3=SIGN(P3,ZDIR)
ZS=P3*CSCOD
DO 20 K=1,2
XKS(K)=CSCOD*PK(K)+UK(K)
20 CONTINUE
OELTAF=OMEGAT-UP1FO-V*P2FO
DELTA=OMEGAT-UP1TO-V*P2TO
DELTAZ=(P10*DUDZ+P20*DUDZ)
DLTASS=(P10*DUDZ+P20*DUDZ)
DLNDLZ=DELTAZ/DELTA
DLNCDZ=0CDZ/C
DLNCDZ=0CDZ/C-DLNDLZ**2
CSQCDZ=CSQCD*(2.*DLNCDZ-DLNDLZ)
CSQDZZ=CSQDZ*(2.*DLNCDZ-DLNDLZ)+CSQCD*(2.*D2LCDZ-02LDNL)
P3S=DELTA*(DLNCDZ-DLNDLZ)
P3SZ=DELTA*D2LCDZ-DELTAZ*DLNDLZ+DLTASS
ZSS=CSQCD*P3S+CSQDZ*P3*ZS
P3SS=P3SZ*ZS
ZSSS=CSQCD*P3SS+ZSS=CSQDZ*P3S*ZS+
P3S*(CSQDZZ*ZS*ZSS=CSQDZ*ZSS)
P3TA-=(P1TO*OUDZ+P2TO*OUDZ)-DELTAZ*DLNCDZ
P3FA-=(P1TO*OUDZ+P2TO*OUDZ)-DELTAZ*DLNCDZ
P3FTZ=P3SZ
ZFTP3=CSQCD
ZFTZ=P3*CSQCDZ
ZFA-=DELTA*P3/PKK
ZTA-=DELTA*P3/PKK
DO 40 K=1,2
XKFTZ(K)=0KDUZ(K)+CSQCDZ*PK(K)
XKFTZ(K)=ZFTP3*PKF(K)*PK(K)*DELTA/PKK
XKFS(K)=ZFTP3*PKF(K)*PK(K)*DELTA/PKK
XKSS(K)=ZS*(CSQDZ*PK(K)+0KDUZ(K))
XKSSS(K)=ZSS*(CSQDZ*PK(K)+0KDUZ(K))+
40 CONTINUE
C--------------------------------------------------
C  DAGDS=PCONST*.5*(1.+GAM)*((RTPK)/DELTA0)**1.5)/SQRT(RHO)
DAGDS=PCONST*.5*(1.+GAM)*((RTPK)/DELTA0)**1.5)/SQRT(RHO)
IF (ZS.EQ.0.) RETURN 1
RETURN
END
SUBROUTINE ADVANS(CFLAG)

ADVANS UTILIZES INFORMATION FROM RATES TO COMPUTE ADVANCE IN CURRENT TIME, AND THE CHANGE IN RAY POSITION AND ASSOCIATED VARIABLES CORRESPONDING TO IT.

COMMON /RYCTRL/ MORAYS,STND,UL,LL,LR,PRTRAY,TIMBEG,DELTIM, + NTIMS,PHIBEG,DELPHI,NPHIS
LOGICAL MORAYS,STND,UL,LL,LR,PRTRAY

COMMON /GROUND/ GLAYER,ZGRND,CGRND,UGRND,VGRND,REFLFC
INTEGER GLAYER

COMMON /RAYNIT/ KGMH,NDCRVS,NUCRVS,IUPDOWN,TO,PHIO,XO,YO,ZO, + P1O,P2O,P3O,OMEGA,DELTAO,P1FO,P2FO,P3FO,OMEGAF, + XTO,YTO,ZTO,P1TO,P2TO,P3TO,OMEGAT,XTSO,YTSO,ZTSO, + P3SO,RHO0,PCONST,:AGES,AGES(20)
INTEGER KGMH,NDCRVS,NUCRVS

COMMON /RAYNIT/ PKK,PKFO,PKTO

EQUIVALENCE (PKO(1),PlO),(PKFO(1),PlFO),(PKTO(1),P1TO)

COMMON /ACSPOT/ TIME,XRO,YRO,ZRO,XDOT,YDOT,ZDOT,AIRSPD,ASPOOT,
+ CO,VO,COT,XMACH,XMADOT,XMU,XMUDOT,COSMU,
+ SINMU,Ek(3,3),Ekdot(3,3),GLOAD,HEADIN,CLIMB,BANK,
+ XDOT,YDOT,ZDOT,XDDOT,YDROT,ZDZDOT
REAL XRO(3),XDOT(3)

EQUIVALENCE(XKRO(l),XRO),(XKDOT(1),XDOT)

COMMON /ATMSPH/ GAM,C,U,V,DCDZ,DUDZ,DVDZ,D2CDZ2,02UDZ2,02VDZ2,RMO
REAL GAM,C,U,V

COMMON /RAYVAR/ ZDIR,PKK,RTPAO,ATTEN,SIGMA,X,Y,Z,DAGE,XF,YF,ZF,
+ XT,YY,XT,YS,ZS,SS,SSS,SSSS,SSSS,P3,
+ PSF,P3F,PSF,XFO,YFO,ZFO,XTF2,YFZ,XTF, + ZF,AZTA,P5FZ,P3A,APHA,AREA,DAGDS
REAL XSS(3),XXS(2),XXFTZ(3),XXS(3),XXS(3)

REAL SIGMA,X,Y,Z,DAGE,XF,YF,ZF,XT,YY,XT
REAL X(3),XXF(3),XFT(3)

EQUIVALENCE (XX(1),X),(XX(1),X),(XX(1),X),XKSS(1),XSS(3)
EQUIVALENCE (XXF(1),X),(XFT(1),X),(XFT(1),X)

COMMON /GRAYVR/ GATEN,DPFACT

COMMON /RAYMLD/ HSIGMA,HX,HY,HZ,HDAGE,HXF,HYF,HZF,HXT,HYT,HZT, + NXS,HYS,HZS,HXSS,HYSS,HZSS,HXSSS,HYSSS,HZSSS,
LOGICAL TONE
REAL HXXS(3), HXXFS(2), HXXTS(2), HXXFTZ(3), HXXSS(3), HXXXSSS(3)
REAL RS(3)
REAL MSIGMA, RX, HY, HZ, HDAGDS, HXF, HYF, HZF, HXT, HYT, HZT
REAL HXX(3), HXXFS(3), HXXT(3)
REAL TPRINT, TETIMEVR
REAL RF(3), RT(3), RX(3), SIG
EQUIVALENCE (HXX(1), RX), (HXXFS(1), HXF), (HXXT(1), HXT), (HXXXS(1), HXS)
EQUIVALENCE (HXXFS(1), HXF), (HXXTS(1), HXTS), (HXXFTZ(1), HXXFTZ)
EQUIVALENCE (HXXXSSS(1), HXXXSS), (HXXXSSS(1), HXXXSSS)

COMMON /CAUSTC/ NUMR, CTRACE, CT(360), CPHI(360), CXYZ(360, 3)
REAL CXYZ
LOGICAL TRACE

COMMON /RPOSN/ N PTR, CPOSN, RTT(200), RXY(200, 3), RAGE(200),
+ RPFAC(200), RVLIFT, REMEM
REAL RTT, RXY, RAGE, RPFAC, RVLIFT
INTEGER N PTR, CPOSN
LOGICAL REMEM

LOGICAL OPSIGN, CFLAG

DATA MAXR/200/

OPSIGN(A,B)=((A.LT.0.).AND..(B. GE.0.).OR.((A.GT.0.).AND..(B.LT.0.)))

CFLAG = .FALSE.
AA=Z-HZ
BB=5*(ZS+H2S)
CC=(ZSS-H2SS)/10.
DD=(ZSSS+H2SSS)/120.
IF (AA.EQ.0.) RETURN
DELSIG=AA/BB
DO 10 K=1,5
  ENUM=(-2*DELSIG*CC)*DELSIG*AA
  DEN=(-3*DD*DELSIG+2.*CC)*DELSIG*BB
  IF (DEN*AA. GE.0.) GO TO 12
  DELSIG=DELSIG-ENUM/DEN
10 CONTINUE
GO TO 15
12 WRITE(7,14)
14 FORMAT( ' TOLSIG TOO LARGE. ')
15 SIGMA=MSIGMA+DELSIG
HDSLIG=.5*DELSIG
DLSIG=DELSIG/6.
DO 20 K=1,2
  XZ(K)=HXX(K)+((HXXXSSS(K)+HXXSSS(K))**DELSIG/12.-
& (HXXSSS(K)+HXXSSS(K))**DELSIG+2*(HXXS(K)+HXXS(K))**HDSLIG
20 CONTINUE
EM11=1.-DLSIG6*(2.*ZFTZ+H2FTZ)
EM12=*DLSIG6*(2.*ZFTP3+H2FTP3)
EM21=*DLSIG6*(2.*P3FTZ+NP3FTZ)
EM22=1.
DET=EM11*EM22-EM12*EM21
HEM11=*DLSIG6*(ZFTPZ+2.*H2FTP3)
HEM12=*DLSIG6*(2.ZFTPZ+2.*H2FTP3)
HEM21=*DLSIG6*(P3FTZ+2.*NP3FTZ)
HEM22=1.
AZ=HEM11*HZF+HEM12*NP3F+HDLSIG*(ZFA+H2FA)
BZ=HEM21*HZF+HEM22*NP3F+HDLSIG*(P3FA+HP3FA)
ZF=(EM22*AZ-EM12*BZ)/DET
P3F=(-EM21*AZ+EM11*BZ)/DET
AZ=HEM11*HZT+HEM12*NP3T+HDLSIG*(ZTA+H2TA)
BZ=HEM21*HZT+HEM22*NP3T+HDLSIG*(P3TA+HP3TA)
ZT=(EM22*AZ-EM12*BPZ)/DET
P3T=(-EM21*AZ+EM11*BZ)/DET
DO 40 K=1,2
  XXF(K)=XXF(K)+HDLSIG*(XXFS(K)+XXKFS(K))+DLSIG6*
  & (ZF=(2.*XXFTZ(K)+XXKFTZ(K)+2.*XXKFTZ(K)))
  XKT(K)=XXKFTZ(K)+HDLSIG*(XXKS(K)+XXKKS(K))+DLSIG6*
  & (ZT=(2.*XXFTZ(K)+XXKFTZ(K)+2.*XXKFTZ(K)))
40 CONTINUE
AREA=ARTUBE(P3,XXF,XKT)
C PFAC=C/CSQRT(RHOPK/(DELTAS*(ABSAREA)+1.E-12)))
ARFCT=CSQRT(ABSAREA+1.E-12)
HARFCT=CSQRT(MAREA+1.E-12)
IF (OPSIGN(MAREA,AREA)) GO TO 70
DAGE=HDAGE*ATTEN*DELSIG*(DAGOS*(2.*ARFCT+ARFCT)+HDAGDS* 
  + (HARFCT-2.*ARFCT))/1.5*(ARFCT+HARFCT)**2)
C SAVE RAY PARAMETERS
C IF (Z.EQ.ZGRND) .GT.762.0 THEN
  IF (NPRTR.LT.MAXR) NPRTR = NPRTR + 1
  DO 1011 K = 1,3
    RXY2(NPTR,K) = XX(K)
1011 CONTINUE
RTT(NPTR) = SIGMA
RAGE(NPTR) = DAGE
C SAVE RAY PARAMETERS
C IF (Z.EQ.ZGRND) THEN
  GATTEN = ATTEN
  GPFACT = PFAC
ENDIF
IF (Z.EQ.ZGRND) THEN
RETURN
C CAUSTIC ENCOUNTERED
C
70 AR1=HAREA
   AR2=AREA
   TONE=AR2,GT,AR1
   TAU1=0.
   TAU2=1.
100 TAU=.5*(TAU1+TAU2)
   TAUPR=1.-TAU
   DO 110 K=1,3
      RF(K)=HXXF(K)*TAUPR+XXF(K)*TAU
      RT(K)=HXXT(K)*TAUPR+XXT(K)*TAU
110 CONTINUE
   PZ=TAUPR*HP3+TAU*P3
   IF (TAU2-TAU1.LT.1.E-6) GO TO 160
   ARM=ARTUBE(PZ,RF,RT)
   IF (ARM) 120,160,140
120 IF (TONE) GO TO 150
130 TAU2=TAU
   GO TO 100
140 IF (TONE) GO TO 130
150 TAU1=TAU
   GO TO 100
160 SIG=TAUPR*SIGMA+TAU*SIGMA
   DAGE=DAGE
   IF (HAREA.NE.0.) DAGE=DAGE+ATTEN*DSIG6*8.*(HDAGDS*(1.5-TAU)+
      + DAGDS*TAU)*TAU/HARFCT
   DO 170 K=1,3
      RK(K)=TAU*XK(K)+TAUPR*HXK(K)
      RS(K)=TAU*XKS(K)+TAUPR*HXKS(K)
170 CONTINUE
   IF (.NOT.PRTRAY) GO TO 200
   C
   WRITE(7,180)
180 FORMAT(' CAUSTIC POINT CROSSED.'
   TPRINT=TIMEVR(SIGMA,Z)
   ELEV=PELEV(0.)
   C
   WRITE(7,190) TPRINT,X,Y,Z,P3,ELEV,ZS,DAGE
190 FORMAT(1X,F10.1,3F10.0,G10.3,F6.1,10X,'0.'),6X,6F10.4
200 CONTINUE
   CALL RCPCL('CAUSTIC','SIG,RK,P3,RF,RT,R5,0.)
   C
   SAVE THE POSITION OF THE CAUSTIC WHEN NOT LOOKING FOR THE FOCI
   C
   IF (NUMC.LT.360) NUMC = NUMC + 1
   CPHI(NUMC) = PH10
   CT(NUMC) = SIG
   DO 1010 K = 1,3
      CXYZ(NUMC,K) = RK(K)
1010 CONTINUE
   AGES(NAGES)=DAGE
   NAGES=NAGES+1
   DAGE=0.0
   IF (Z.LE.(ZGRND)+762.00) THEN
IF (NPTR.LT.MAXR) NPTR = NPTR + 1
CPOSN = NPTR
DO 1012 K = 1,3
RXYZ(NPTR,K) = RK(K)
1012 CONTINUE
RTT(NPTR) = SIG
RAGE(NPTR) = DAGE
RPFACT(NPTR) = 0.0
ENDIF
IF (AREA.NE.0.) DAGE = ATTEN*DSIG6*B.*(DAGDS*(1.5-TAUPR) +
+ HAGDS*TAUPR)/ARFC
CFLAG = .TRUE.

IF (XK(3).EQ.ZGRN) THEN
IF (NPTR.LT.MAXR) NPTR = NPTR + 1
DO 1013 K = 1,3
RXYZ(NPTR,K) = XK(K)
1013 CONTINUE
RTT(NPTR) = SIGMA
RAGE(NPTR) = DAGE
RPFACT(NPTR) = PFAC
ENDIF

IF (Z.EQ.ZGRND) THEN
GATTEN = ATTEN
GPFAC = PFAC
ENDIF
RETURN
END
SUBROUTINE RCRVIT

RCRVIT, when a tentative advance brings Ray beyond a reversal layer, will locate the exact position of the reversal layer.

COMMON /RAYNIT/ KGMH,NDCRVS,NUCRVS,IUPOWN,T0,P0ID,X0,Y0,Z0,
+ P10,P20,P30,OMEGA,Delta0,PIFO,PHIF0,PHIF0,OMEGAF,
+ XTO,YTO,ZTO,P1TO,P2TO,P3TO,OMEGAT,XSO,YSO,ZSO,
+ P3SO,RH00,PCONST,RAGES,RAGES(20)
INTEGER KGMH,NDCRVS,NUCRVS,IUPOWN

COMMON /RAYVAR/ ZDIR,PKX,RTPAAO,ATTEN,SIGMA,X,Y,Z,DAGE,XY,DF,DF,
+ X,Y,Z,YS,ZS,XSS,ZSS,XSSS,ZSSS,P3,
+ P3F,P3T,P3S,XFS,YFS,XTS,YTS,XFT3,XFT3,YFT3,2FT3,
+ ZFA,2TA,P3FTZ,P3FA,P3TA,AREA,DAGDS
REAL SIGMA,X,Y,Z,DAGE,XY,DF,DF,X,Y,Z

COMMON /ATMSPH/ GAN,C,U,V,OCOZ,DUDZ,DVDZ,D2CDZ,D2V0Z,D2U0Z,
REAL GAN,C,U,V

COMMON /RAYHLO/ HSIGMA,HX,NY,HZ,HOAG,E,HXF,HYF,HZF,HXT,HYT,HZT,
+ HXS,HYS,HZS,XYSS,XYSSS,XYSSS,P3,
+ HP3,HP3F,HP3T,HP3S,XFS,YFS,XTS,YTS,HZFT3,
+ HXFTZ,HYFTZ,HZFTZ,HZFTZ,HZTA,HP3FTZ,HP3FA,HP3TA,
+ HAREA,HADGDS
REAL HSIGMA,HX,NY,HZ,HOAGE,HXF,HYF,HZF,HXT,HYT,HZT

REAL ZA,ZB,ZMID
REAL DDELTA,RTPKK

ZA=HZ
ZB=Z
5 ZMID=.5*(ZB+ZA)
GG1 = ABS(ZMID-ZA)
GG2 = ABS(ZMID-ZB)

IF (MIN1(ABS(ZMID-ZA),ABS(ZB-ZMID)).LT.1.E-4) GO TO 100

CALL AIR(ZMID)

DDELTA=(OMEGA)-U*(P10)-V*(P20)
RTPKK=DDELTA/C
XLXL = RTPKK-RTPAAO

C IF (ABS(XLXL).LT.1.0E-6) GOTO 90
IF (XLXL) 10,90,20

10 ZB=ZMID
GO TO 5
20 ZA=ZMID
GO TO 5
90 Z = ZMID
RETURN
100 Z = ZA
RETURN
END
SUBROUTINE RECORD(*)

RECORD, WHEN THE RAY HAS BEEN TRACED TO GROUND IN A SELECTED CARPET, WILL RECORD THE LOCATION AND ALL THE ASSOCIATED VARIABLES REQUIRED TO COMPUTE SIGNATURES ON A TEMPORARY FILE (FORTRAN UNIT 9).

COMMON /ATMCON/ REARTH,GO,RSTARR,ROMO,ROGORMO
COMMON /ACI DNT/ IDENT
CHARACTER*8 IDENT
COMMON /ACI WEIG/ ACW1,ACL
COMMON /C LASES/ CNAMES(30)
CHARACTER*8 CNAMES
COMMON /C LASS/ NR1RVS(2,2), TYPRAV(3,2,2),DIRECT,LOFT,UP,DOWN
LOGICAL TYPRAV,DIRECT,LOFT,UP,DOWN
COMMON /G ROUN D/ GLAYER,ZGRND,CGRND,UGRND,VGRND,REFLFC
INTEGER GLAYER
COMMON /ACSPOT/ TIME,XRO,YRO,ZRO,XDOT,YDOT,ZDST,ARTPO,ASPDOT,
  + CO,UC,VC,CDOT,CMACH,CMADOT,CMU,CMUDOT,COSMU,
  + SIMMU,EM(3,3),EMDOT(3,3),GLOAD,HEADM,CLIMB,BANK,
  + XMUDOT,YDUDOT,ZDUDOT,XDDDOT,YDDDOT,ZDDDOT
COMMON /ATMSPH/ GAM,C,U,V,DCDZ,DUODZ,DVDZ,D2C0Z2,D2UDZ2,D2VDZ2,RH0
REAL GAM,C,U,V
COMMON /RAYNIT/ KGMH,NDCRV55,NURCVS,IPUQDNW,T0,P10,X0,Y0,Z0,
  + P10,P20,P30,OMGADEL,PAF0,P2F0,P3F0,OMGAF,
  + XTO,YTO,ZTO,PT0,PT0,PT0,OMEGAT,X50,Y50,Z50,
  + P50,RH0,PCONS2,AGES,AGES(20)
INTEGER KGMH,NDRCV5,NUCRV5,IPUQDNW
REAL PK(2),PKF(2),PKT(2)
EQUIVALENCE (PK(1),P10),(PKF(1),P1F0),(PKT(1),P1TO)
COMMON /RATVAR/ ZDIR,PKX,RPTAAO,ATTEN,SIGMA,X,Y,Z,DAGE,XF,YF,ZF,
  + XT,YT,ZT,XS,ZS,XS,XS,XS,XS,XS,XS,P3,
  + P3F,P3T,P3S,XFS,YFS,XTS,ZFTP3,XFT2,YFT2,ZFT2,
  + P3F,P3T,P3S,XFS,YFS,XTS,ZFTP3,XFT2,YFT2,ZFT2,
  + ZFA,TRA,P3FT2,P3FA,PA,AREA,DAGDS
REAL XKS(3),XKXS(2),XKTS(2),XKTFZ(3),XKSS(3),XKSSS(3)
REAL RX(3),RXF(3),RXT(3)
REAL SIGMA,X,Y,Z,DAGE,XF,YF,ZF,XT,YT,ZT
REAL XK(3),XKF(3),XKT(3)
EQUIVALENCE (XK(1),X),(XXF(1),XF),(XKT(1),XT),(XXS(1),XS)
EQUIVALENCE (XXFS(1),XFS),(XCTS(1),XTS),(XXFTZ(1),XFTZ)
EQUIVALENCE (XXSS(1),XSS),(XXS(1),XSS)

COMMON /GRAYVR/ GATTEN,GPFACT

DATA DGPRAD/57.295780/

C

NCLASS=3
IF (KGMH.EQ.1) GO TO 10
NCLASS=2*NDCRV+3*(KGMH+2*(NDCRV-WPWN))
IF (.NOT.TYPRAY(NDCRV,3-IUPWN,KGMH-1)) GO TO 20
GO TO 15
10 IF (.NOT.DIRECT) GO TO 20
C

RTPK=SQR(PKK)
15 DELTA = C*SQR(PKK)
C

PFAC=PCONST*C*SQR(RHO*RTPK/(DELTAO*(ABS(AREA)*1.E-12)))
PFAC=PCONST*C*SQR(RHO*DELTA/(ABS(AREA)*1.E-12))
PFAC=PFAC*ATTEN*(1.+REFLFC)
VLIFT=ACWT*GLOAD*GDOSMUCOS(PHI0-BANK)/DGPRAD/
+ (PHO*SINM)*AIRSPD=2
RECPHI=AMOD(AMOD(PHIO,360.)*450.,360.)-90.
RSIGN=SIGNA
DO 17 K=1,3
RX(K)=KK(K)
RF(K)=KF(K)
RT(K)=KT(K)
17 CONTINUE
AGES(NAGES)=OAGE
WRITE(9,99) NAMES(NCLASS)

99 FORMAT(ABA)
WRITE(9,*) KGMH,NDCRV,1UPWN,XMACH,VLIFT,TO,RECPHI,
+RSIGN,RX,OMEGA,P3,XS,RT,RF,PFAC,NAGES,(AGES(K),K=1,NAGES)
20 IF (KGMH.EQ.1) GO TO 30
IF (NDCRV.GE.NCRCRV(3-IUPWN,KGMH-1)) RETURN
RETURN 1
30 IF (LOFT) RETURN 1
RETURN
END
FUNCTION ARTUBE(PZ,RF,RT)
C
ARTUBE COMPUTES THE JACOBIAN DEFINING THE RAY TUBE AREA.
C
COMMON /RAYNIT/ KGMM,NDCRVS,NUCRVS,UPDWN,T0,P10,X0,Y0,Z0,+
+ P10,P20,P30,OMEGA,DELTAO,P1FO,P2FO,P3FO,OMEGAF,+
+ XO,YTO,ZTO,P1TO,P2TO,P3TO,OMEGAT,XSO,YSO,ZSO,+
+ P3SO,RHOO,PCONST,NAGES,AGES(20)
INTEGER KGMM,NDCRVS,NUCRVS,UPDWN
INTEGER INDET(3)
REAL PK(2),PKF(2),PKT(2)
REAL RF(3),RT(3)
EQUIVALENCE (PK(1),P10), (PKF(1),P1FO), (PKT(1),P1TO)
DATA INDET/2,3,1/
C
ARTUBE=PZ*(RF(1)*RT(2)-RF(2)*RT(1))
PKK=PZ**2
DO 10 K=1,2
   ARTUBE=ARTUBE+PK(K)*(RF(INDET(K))*RT(INDET(K+1))-+
& RF(INDET(K+1))*RT(INDET(K)))+PKK=PKK=PK(1)**2
10 CONTINUE
C
ARTUBE=ARTUBE/SQRT(PKK)
ARTUBE=ARTUBE/PKK
RETURN
END
SUBROUTINE RCSPCL(TYPE,SIG,RK,RF,RT,RS,AREA)

RCSPCL RECORDS ON A TEMPORARY FILE (FORTRAN UNIT 11) THE POSITIONS AND TIMES FOR EACH "SPECIAL POINT" IN THE RAY'S PATH. "SPECIAL POINTS" INCLUDE REVERSAL LAYER ENCOUNTERS, GROUND ENCOUNTERS, AND THE ENCOUNTER WITH THE CAUSTIC SURFACES.

CHARACTER*8 TYPE
REAL RK(3),RF(3),RT(3),SIG
REAL RS(3),AREA,RF4(3),RT4(3),RK4(3)

COMMON /CLASES/ CNAI4ES
CHARACTER*8 CNAI4ES

COMMON /CLASSS/ NRCURV(2,2),TYPRAY(3,2,2),DIRECT,LOFT,UP,DOWN
LOGICAL TYPRAY,DIRECT,LOFT,UP,DOWN

COMMON /PRINTS/ TITLE(30),TIMLBL
CHARACTER*4 TITLE
CHARACTER*8 TIMLBL

COMMON /PRINTC/ KTPSIG,CVRTIM
LOGICAL CVRTIM
LOGICAL PTHDR
COMMON /JJJ/ PTHDR

COMMON /RAYZIT/ KGMH,NDCRVS,NUCRVS,IPOWN,X0,Y0,Z0,
+ P10,P20,P30,OMEGA,DELTAO,P1FO,P2FO,P3FO,OMEGAF,
+ XTO,YTO,ZTO,P1TO,P2TO,P3TO,OMEGAT,XSO,YSO,ZSO,
+ P3SO,RHOO,PCONST,NAGES,AGES(20)
INTEGER PK(2),PKF(2),PKT(2)
EQUIVALENCE (PKF(1),P1FO),(PKT(1),P1TO)
KTPSIG=(SIG)
DO 20 K=1,3
RK4(K)=RK(K)
RF4(K)=RF(K)
RT4(K)=RT(K)

IF (TYPE.NE.'CAUSTIC') RETURN

NCLAS=3
IF (KGMH.EQ.1) GO TO 10
NCLAS=2*(NDCRVS+3*(3-KGMH+2*IPOWN))>3
10 RCTPHI=AMOD(AMOD(P1FO,360.)+450.,360.)*90.
RTSIG=(SIG)
DO 20 K=1,3
RK4(K)=RK(K)
RF4(K)=RF(K)
RT4(K)=RT(K)
20 CONTINUE

IF (PTHDR) THEN
  C WRITE(7,5) TITLE
  5 FORMAT('1',30A4)
  C WRITE(7,6) TIMBL, TIMB
  6 FORMAT('0 POINT', T11, 'WHICH #LOW', T22, 'RAY', T34, 'TIME', T44, 'PHI',
        + 'TYPE', T21, 'CLASS', T32, 'REINITIAL', T42, 'REINITIAL', T93, 'AZIMUTH'
        + 'ELEV', T33, A8, T33, A8, T65, 'MET', T75, 'MET', T85, 'MET', T93, 'MET', T100,
        + 'DEG', T106, 'MET**2/SEC')
ENDIF

PTHDR = .FALSE.
TPRN = TIMCVR(T0, 2)
SIGPN = TIMCVR(RSIGM, 2)

CALL EAMNUL(ELEV, AZIM, PMAG, PK(1), PK(2), P2)

220 FORMAT(1X, A8, 215, T22, A8, T31, 2F10.5, T48, F10.1, 2F10.0, F10.1,
        + F7.0, F7.1, G12.4)
100 RETURN
END
C
C **************************************************
C * SIGNATURE CALCULATIONS - RDSPCL,SIGNUR,FREAD,AGING,HILBERT *
C * SIGPRN,CPVAL,SORTEM *
C * (DREAD,FFA2F,FFA2I) *
C **************************************************
C
C AFTER ALL RAYS HAVE BEEN TRACED, IT IS THE TASK OF THE SIGNATURE AGING
C Routines to perform the final calculations and determine the actual
C overpressures to be expected.
C
C SUBROUTINE RDSPL
C
C RDSPL IS ACTUALLY BETWEEN THE RAY TRACING ROUTINES AND THE SIGNATURE
C CALCULATIONS PER SE. IT LISTS ALL THE SPECIAL POINTS RECORDED BY RCSR
C
C CHARACTER*8 PTTYPE,RYCLAS
C REAL SIGPRN,TPRN,TIMCVR
C REAL RK(3),XF(3),XT(3),PK(3)
C INTEGER NHIGH,NLOW
C
C COMMON /PRINTS/ TITLE(30),TIMLBL
C CHARACTER*4 TITLE
C CHARACTER*8 TIMLBL
C
C COMMON /PRINTC/ KTPSIG,CVRTIM
C LOGICAL CVRTIM
C
C WRITE(7,5) TITLE
C 5 FORMAT(I1',30A4)
C WRITE(7,6) TIMLBL,TIMLBL
C 6 FORMAT('O POINT',T11,'#HIGH #LOW',T22,'RAY',T34,'TIME',T44,'PHI',
C + T54,'TIME',T66,'X',T76,'Y',T86,'Z',T93,'RAY NORMAL',T109,'AREA'/
C + T3,'TYPE',T21,'CLASS',T32,'(INITIAL)',T42,'(INITIAL)',T53,'AZIMUTH
C + T63,'ELEV',T73,'AZ',T83,'MET',T93,'MET',T109,'DEG',T109,'T100,
C + 'DEG',T106,'MET**2/SEC')
10 READ(9,200,END=100) PTTYPE,RYCLAS
200 FORMAT(2A8)
   READ(9,*,END=100) NHIGH,NLOW,TIMO,PHI0,SIGMA,RK,XF,
   * XT,AREA,PK
   TPRN=TIMCVR((TIMO),2)
   SIGPRN=TIMCVR((SIGMA),2)
   CALL EAMENU(ELEV,AZIM,PMAG,PK(1),PK(2),PK(3))
C WRITE(7,20) PTTYPE,NHIGH,NLOW,RYCLAS,TMR,PHI0,SIGMA,RK,AZIM,
C + ELEV,AREA
C 20 FORMAT(1X,A6,215,T22,A8,T31,F10.1,F8.2,T50,F10.1,2F10.0,F10.1,
C +F7.0,F7.1,G12.4)
   GO TO 10
100 RETURN
END
SUBROUTINE SIGNUR(MAXOP)

SIGNUR has overall control of the aging and printout process. For each ray terminus recorded by RECORD, it reads, interprets and prints out the information on ray type, Mach number of aircraft, initiation time and phi angle, location, elevation and azimuth of the ray normals, as well as the conversion factors from F-function normalized coordinates normalized coordinates to time (TFACT) and pressure (PFACT). It combines the F-functions according to this information and controls the evolution of the signature.

PARAMETERS:

NAME TYPE DISCRIPTION

INPUT:

NONE

OUTPUT:

MAXOP : R maximum overpressure

REAL MAXOP

COMMON /FFTAB/ KRCAC,KSPDS,SPEDS(11),LOCSPD(10),KTABL,
NTAU,TAU(200),FAC(200),FLC(200)
COMMON /CFFTAB/ ACIDNT
CHARACTER*8 ACIDNT

COMMON /BASEAG/ NTERMS,XILEAD(2),XI(500),XITAIL(502),
VLEAD(2),V(500),VTAIL(502)
DIMENSION XII(1004),VI(1004)
EQUIVALENCE (XII(1),XILEAD(1)),(VI(1),VLEAD(1))

COMMON /PRINTS/ TITLE(30),TIMLBL
CHARACTER*4 TITLE
CHARACTER*8 TIMLBL

COMMON /PRINTC/ KTPSIG,CVRTIM
LOGICAL CVRTIM
REAL SIGD,TDO
REAL TIMCVR

COMMON /SIGPAR/ KGMH,NRCURV,IUPDWN,XJACH,VLIFT,TO,
PHIO,SIGMA,XK(3),OMEGA,PK(3),XKS(3),XKT(3),
XKF(3),PFACT,NAGES,AGES(20)
COMMON /SIGPAC/ IDENT,RAYNAM
CHARACTER*8 IDENT,RAYNAM

INTEGER KGMH,NRCURV,IUPDWN

COMMON /RAYOUT/ SSIGMA,SPHIO,SKX(3),OPG,CSEL
IF (KTPSIG.LE.0) RETURN
REWIND 9
READ(9,'(AB)') IDENT
C WRITE(7,15) TITLE
C IF (KTPSIG.GE.2) CALL FREAD
IF (KTPSIG.GT.1) GO TO 10
C WRITE(7,16) IDENT
C WRITE(7,20) TIMLBL,TIMLBL
10 CONTINUE
READ(9,*9,END=500) RAYNAM
99 FORMAT(AB)
READ(9,*,END=500) KGMH,KRCURV,IPUPDN,XMACH,VLIFT,TO,PHIO,
+ SIGMA,XK,OMEGA,PK,XKS,XKT,XKF,PFACT,NAGES,
+ (AGES(K),K=1,NAGES)
CALL EAMENU(ELEV,AZIM,FMAG,PK(1),PK(2),PK(3))
IF (KTPSIG=2) 25,17,11
11 CONTINUE
C WRITE(7,15) TITLE
15 FORMAT(1',30A4)
C WRITE(7,16) IDENT
16 FORMAT(0A/C IDENT=',AB)
17 CONTINUE
C WRITE(7,20) TIMLBL,TIMLBL
C 20 FORMAT(' RAY TYPE MACH#,T2O,'TIMIT',T2B,'PHIO','TIME',T50,'X'
C + ,T60,'Y',T67,'Z',T72,'RAY NORMAL',TB4,'TFACT',T91,'PFAC',T104,
C + 'VLIFT'/T2O,AB,T2B,'DEG.',T36,A8,T49,'MET',T59,'MET',T66,'MET',
C + T77,'AZIMUTH ELEV',T83,'MS/MET',T90,'PA/MET**5',T104,'MET**2')
25 TFACT=1000./OMEGA
TDD=TIMCVR((TO),2)
SIGD=TIMCVR((SIGMA),2)
C WRITE(7,100) RAYNAM,XMACH,TO,PHIO,SIGD,XK,AZIM,ELEV,TFACT,
C + PFAC,VLIFT,(AGES(K),K=1,NAGES)
C 100 FORMAT(*',AB,IX,F5.3,F7.2,F10.1,F7.2,F10.1,F7.2,F10.1,F7.2,F10.1,F7.2,
C + 211.4,9F9.2)
C IF (KTPSIG.EQ.1) GO TO 10
C CALL NEWTAB
DO 200 K=1,NTAU
XI(K)=TAU(K)
V(K)=FAC(K)+VLIFT*FLC(K)
200 CONTINUE
NTERMS=NTAU
CALL AGING(AGES(1))
IF (NAGES.LE.1) GO TO 215
DO 210 K=2,NAGES
CALL HILBRT
CALL AGING(AGES(K))
210 CONTINUE

215 CALL SIGPRT(MAXOP)
500 RETURN
END
SUBROUTINE FREAD

FREAD determines whether the necessary F-function tables are in main memory, and if not, reads them into main memory.

HAS ENTRY POINT AT NEWTAB

COMMON /SIGPAR/ KGMH,NRCURV,IPDOWN,XMACH,VLIFT,TQ,
+ PHIO,SIGMA,XK(3),OMEGA,PK(3),XKST(3),
+ XKF(3),PFACT,NAGES,AGES(20)
COMMON /SIGPAC/ IDENT,RAYNAM
  CHARACTER*8 IDENT,RAYNAM
  INTEGER KGMH,NRCURV,IPDOWN

COMMON /FFTAB/ KRCAC,NSPDS,SPEEDS(11),LOCSPD(10),XTABL,
+ NTAU,TAU(200),FAC(200),FLC(200)
COMMON /CFTAB/ ACIDNT
  CHARACTER*8 ACIDNT
  CHARACTER*15 FORM(4)

COMMON /FERSM/MESG(26)
  LOGICAL OPENED

VARIABLE FORMAT SPECIFIER

DATA FORM/"(T28,F5.2,F5.0)'","(T38,F5.2,F5.0)'","(T48,F5.2,F5.0)'",
+ "(T58,F5.2,F5.0)'",
DATA OPENED/.FALSE./

IF (.NOT.OPENED) THEN
  OPEN(90,FILE='FFUNC.DIR',STATUS='OLD',ACCESS='DIRECT',
+ FORM='FORMATTED',RECL=80)
  OPENED = .TRUE.
ENDIF

CALL LJUST(8,1,IDENT,ACIDNT)
CALL ACCVRT(IDENT, ACIONT)
KRCAC = 1
C 10 CALL DREAD(90, KRCAC, BUFFER, *900)
10 CONTINUE
C READ FOR FORTRAN V
C READ(90, 991, REC=KRCAC, ERR=900) ACIO, KINCR, NSPOS
991 FORMAT(A8, T18, I5, T26, I2)
C IF (ACIONT.EQ. BUFFER(1)) GO TO 20
IF (ACIONT.EQ. ACIO) GOTO 20
C CALL FFA2N(BUFFER, 18, 5, 1, DUMMY, 0., KERR)
READ(90, 995, REC=KRCAC, ERR=900) DUMMY
995 FORMAT(T18, F5.0)
KINCR = DUMMY + 0.5
IF (KINCR.EQ. 0) GO TO 950
C KRCAC = KRCAC + KINCR
C KRCAC = KRCAC + KINCR
GO TO 10
C 20 CALL FFA2N(BUFFER, 26, 2, 1, DUMMY, 1., KERR)
20 CONTINUE
READ(90, 996, REC=KRCAC, ERR=900) DUMMY
996 FORMAT(T26, F2.0)
NSPOS = DUMMY + 0.5
NCARDS = (NSPOS + 3) / 4
DO 40 I = 1, NCARDS
  K1 = I
  K2 = MIN0(4, NSPOS - 4*(I-1))
  DO 30 K = K1, K2
    KK = K + 4*(I-1)
C CALL FFA2N(BUFFER, 18+10*K, 5, 1, SPEEDS(KK), 0., KERR)
C CALL FFA2N(BUFFER, 23+10*K, 5, 1, DUMMY, 0., KERR)
C
C READ WITH A DIFFERENT FORMAT TO GET EACH MACH NUMBER AND
C RELATIVE ADDRESS PAIR IN ORDER
C READ(90, FORM(K), REC=KRCAC+I-1, ERR=900) SPEEDS(KK), DUMMY
LOCSPO(KK) = DUMMY + 0.5
LOCSPO(KK) = LOCSPO(KK) - KRCAC
CONTINUE

CALL DREAD(90, KRCAC+1, BUFFER, *900)

CONTINUE

WRITE(7, 50) IDENT,(SPEEDS(K), K=1, NSPOS)

FORMAT('OF-FUNCTION TABLES FOR ' ,A8,' AIRCRAFT.'/)

SPEEDS(NSPC+1)=SPEEDS(NSPOS)

DO 60 M=1, NSPOS

SPEEDS(NSPOS+M+2)=0.5*(SPEEDS(NSPOS*M+2)+SPEEDS(NSPOS*M+1))

CONTINUE

SPEEDS(1)=1.

LTABL=1

GO TO 150

ENTRY POINT

ENTRY NEWTAB

IF (NSPOS.EQ.1) RETURN

DO 100 K=1, NSPOS

IF (AMIN1(XMACH-SPEEDS(K), SPEEDS(K+1)-XMACH).GE.0.) GO TO 120

CONTINUE

IF (XMACH.GT.SPEEDS(NSPOS+1)) WRITE(7, 110) XMACH, SPEEDS(NSPOS+1)

FORMAT('MACH NUMBER ',F5.2,' IS GREATER THAN MAXIMUM IN TABLES ')

K=NSPOS

LTABL=K

IF (LTABL.EQ.KTABL) RETURN

150 LTABL=KTABL

MREC=LOCSPD(KTABL)

CALL DREAD(90, MREC, BUFFER, *900)

CALL FFA2N(BUFFER, 16, 6, 1, XLAC, 0., KERR)

CALL FFA2N(BUFFER, 22, 7, 1, STEP, 0., KERR)

CALL FFA2N(BUFFER, 13, 3, 1, DUMMY, 0., KERR)

READ(90, 992, REC=MREC, ERR=900) DUMMY, XLAC, STEP

992 FORMAT(T13, F3.0, T16, F6.2, T22, F7.2)

NTAU=DUMMY+.5

XLR=SQRT(XLAC)

CONST=1./(XLR*XLAC)

DO 200 K=1, NTAU

CALL FFA2N(BUFFER, 48, 3, 1, EXP10, 0., KERR)

CALL FFA2N(BUFFER, 35, 12, 1, FAC(K), 0., KERR)

FAC(K)=FAC(K)*XLR*(10.**EXP10)

READ(90, 993, REC=MREC, K-1, ERR=900) FAC(K), FLC(K)

993 FORMAT(T35, E16.9, T54, E16.9)

FAC(K)=FAC(K)*XLR
CALL FFA2N(BUFFER,67,3,1,EXP10,0.,KERR)
CALL FFA2N(BUFFER,54,12,1,FLC(K),0.,KERR)
FLC(K)=FLC(K)*CONST*(10.**EXP10)
FC(K)=FLC(K)*CONST
TAU(K)=(K-1)*STEP*XLAC
CALL DREAD90,MREC+K,BUFFER,*900)
200 CONTINUE
RETURN
900 WRITE(7,910) MEG
910 FORMAT(' DA/10 ERROR ON UNIT 90.'/1X,18,8A4,419)
   STOP 900
950 WRITE(7,960) IDENT
960 FORMAT(' AERIAL ID ',8A8,1A,1A,1A,1A,8A4,419)
   STOP 960
END
SUBROUTINE AGING(AGE)

AGING SHIFTS THE ABSCISSA VALUES (PHASE) OF THE F-FUNCTIONS ACCORDING TO THE AGE VALUE, DETERMINES THE TOTAL AREA OF THE RESULTING FIGURE, AND FITS JUMP DISCONTINUITIES AS APPROPRIATE. REPLACES THE INPUT F-FUNC WITH THE RESULT.

COMMON /BASEAG/ NTERMS,XILEAD(2),XI(500),XITAIL(502),
                  VLEAD(2),V(500),VTAIL(502)
LOGICAL JUMP
REAL SA,SB,SC,SD,SE1,SE2
DIMENSION XII(1004),VI(1004)
EQUIVALENCE (XII(1),XILEAD(1)),(VI(1),VLEAD(1))

DO 2 K=1,2
   XII(K)=XI(1)
   XII(NTERMS+K+2)=XI(NTERMS)
   VI(K)=0.
   VI(NTERMS+K+2)=0.
2 CONTINUE

LTERMS=2
K=2
XIB=XII(2)
VB=0.
SB=0.0
L=2
V0=0.
SD=0.0
XID=XII(2)
JUMP=.FALSE.

5 K=K+1
IF (K.GT.NTERMS+4) GO TO 200
XIA=XIB
VA=VB
SA=SB
VB=VI(K)
XIB=XII(K)-AGE*VB
SB=SA+((.50*(XIB-XIA))*(VB+VA)
XII(1)=AMIN1(XII(1),XIB)
XII(NTERMS+4)=AMAX1(XII(NTERMS+4),XIB)
IF (.NOT.JUMP) GO TO 15
IF (XIB.LT.XIA) GO TO 10
LTERMS=LTERMS+1
XII(LTERMS)=XIB
VI(LTERMS)=VB
GO TO 5
10 JUMP=.TRUE.
   GO TO 5
15 IF (XIB.LE.XIA) GO TO 5
17 XIC=XII(L-1)
   VC=Vl(L-1)
   SC=SD+(-.50*(VC+VD))*(XID-XIC)
   IF (XIC.LE.XIA) GO TO 21
   L=L-1
   VD=VC
   XID=XIC
   SD=SC
   GO TO 17
20 L=L+1
   XIC=XID
   VC=VD
   SC=SD
   XID=XII(L)
   VD=Vl(L)
   SD=SC+(-.50*(VC+VD))*(XID-XIC)
21 IF (XIB.LE.XIC) GO TO 5
   IF (XIC.LE.XIC) GO TO 20
   IF (XIA.GT.XID) GO TO 20
   XIE=AMIN1(XIB,XID)
   VE1=(VB*(XIE-XIA)+VA*(XIB-XIE))/(XIB-XIA)
   VE2=(VD*(XIE-XIC)+VC*(XID-XIE))/(XID-XIC)
   SE1=SA+(-.50*(VE1+VA))*(XIE-XIA)
   SE2=SE+(-.50*(VE2+VC))*(XIE-XIC)
   C=SE1-SE2
   IF (C) 25,40,30
25 IF (XIC-XIB) 20,5,5
30 A=(VB-VA)/(XIB-XIA)-(VD-VC)/(XID-XIC)
   B=VE1-VE2
   XIE=XII-2.*C/(B+SQR(B**2-2.*A*C))
   VE1=(VB*(XIE-XIA)+VA*(XIB-XIE))/(XIB-XIA)
   VE2=(VD*(XIE-XIC)+VC*(XID-XIE))/(XID-XIC)
   SE2=SC+(-.50*(VC+VE2))*(XIE-XIC)
40 SB=SE2+(-.50*(VE1+VB))*(XIB-XIE)
   XII(L)=XIE
   /II(L)=VE2
   XII(L+1)=XIE
   VI(L+1)=VE1
   XII(L+2)=XIB
   VI(L+2)=VB
   L=L+2
   SD=SB
   XID=XIB
   VD=VB
   LTERMS=L
   JUMP=.FALSE.
   GO TO 5
200 LL=1
   DO 220 L=3,LTERMS
      IF (XII(LL).EQ.XII(L)) GO TO 220
      IF (XII(LL).LT.XII(L-1)) GO TO 210
IF (VI(LL).EQ.VI(LL-1)) GO TO 220

210 XII(LL+1)=XII(LL-1)
   VI(LL+1)=VI(LL-1)
   LL=LL+1

220 CONTINUE
   XII(LL+1)=XII(LTERMS)
   VI(LL+1)=VI(LTERMS)
   NTERMS=LL-3
   RETURN
END
SUBROUTINE HILBRT

HILBERT HAS OVERALL RESPONSIBILITY FOR CALCULATING THE HILBERT TRANSFORM. REPLACES THE INPUT F-FUNCTION, AS MODIFIED BY AGING AND POSSIBLY CONTAINING SHOCKS, BY ITS HILBERT TRANSFORM. COMPUTES THE TRANSFORM AT A SELECTION OF POINTS DETERMINED BY THE OVERALL STRUCTURE OF THE FUNCTION. THIS INCLUDES A SET OF POINTS EXPONENTIALLY CONVERGING TO EACH SHOCK (TERMINATING WITHIN A DISTANCE OF THE SHOCK EQUAL TO 6*10E TIMES THE OVERALL SCALE OF THE INPUT F-FUNCTION). IT ALSO INCLUDES A SET OF POINTS WHICH ARE CENTERED ON THE MEAN ABCISSA VALUE OF THE INPUT F-FUNCTION AND WHICH ARE SPACED AT INCREASING INCREMENTS TO COV AN INTERVAL SEVERAL TIMES THE ABCISSA SCALE OF THE INPUT F-FUNCTION.

COMMON /BASEAG/ NTERMS,XILEAD(2),XI(500),XITAIL(502),
+ VLEAD(2),V(500),VTAIL(502)
DIMENSION XII(1004),VI(1004)
EQUIVALENCE (XII(1),XILEAD(1)),(VI(1),VLEAD(1))

COMMON /XISAVE/ NSTRMS,XIS(502),VS(502)

WEIGHT=0.
XIMEAN=0.
XIVAR=0.
XIS(1)=XII(2)
VS(1)=0.
NSTRMS=NTERMS+2
DO 10 K=2,NSTRMS
   XIS(K)=XIS(K-1)
   VS(K)=VS(K-1)
   WA1=VS(K-1)**2
   WAIT2=VS(K-1)*VS(K)
   WAIT3=VS(K)**2
   DELXI=XIS(K)-XIS(K-1)
   WAV=DELXI*(WAIT1+WAIT2+WAIT3)/3.
   WAVX=XIS(K-1)*WAV+((WAIT1+2.*WAIT2+3.*WAIT3)*DELXI**2)/12.
   WAVX2=(WAV*XIS(K-1)+2.*WAVX)*XIS(K-1)+(WAIT1+3.*
   & WAIT2+6.*WAIT3)*DELXI*DELXI**2/30.
   WEIGHT=WEIGHT+WAV
   XIMEAN=XIMEAN+WAVX
   XIVAR=XIVAR+WAVX2
10 CONTINUE
   XIMEAN=XIMEAN/WEIGHT
   XIVAR=XIVAR/WEIGHT*XIMEAN**2
   XILNG=SORT(XIVAR)
   LTRMHF=40
   LTERMS=LTRMHF*2+1
   NTERMS=0
DO 100 L=1,LTERMS
XINEW=XILNG*(LTERMS*(L-LTRMHF)/(L*(LTERMS+1.-L)))+XIMEAN

CALL CPVAL(XINEW,VV,*100)

NTERMS=NTERMS+1
V(NTERMS)=VV
XI(NTERMS)=XINEW

100 CONTINUE
XI(NTERMS+1)=2.*(XI(1)-XIMEAN)+XIMEAN
XI(NTERMS+2)=2.*(XI(NTERMS)-XIMEAN)+XIMEAN
V(NTERMS+1)=0.
V(NTERMS+2)=0.
NTERMS=NTERMS+2
DO 200 K=2,NTERMS
   IF (XI(K).GT.XI(K-1)) GO TO 200
   IF (VS(K).EQ.VS(K-1)) GO TO 200
   DELXI=XILMG
   DO 190 M=1,10
      DELXI=DELXI*.3
      CALL CPVAL(XI(K)-DELXI,VV,*180)
   NTERMS=NTERMS+1
   XI(NTERMS)=XI(K)-DELXI
   V(NTERMS)=VV
180 CALL CPVAL(XI(K)+DELXI,VV,*190)
   NTERMS=NTERMS+1
   XI(NTERMS)=XI(K)+DELXI
   V(NTERMS)=VV
190 CONTINUE
200 CONTINUE

CALL SORTEM

RETURN
END
SUBROUTINE SIGPRT(MAXOP)

SIGPRT PRINTS OUT THE FINAL SIGNATURE, AS DIRECTED ON THE CONTROL FILE CARDS.

PARAMETERS:

NAME       TYPE   DESCRIPTION

INPUT:

NONE

OUTPUT:

MAXOP : R  MAXIMUM OVERPRESSURE

REAL MAXOP

COMMON /PRINTS/ TITLE(30), TIMLBL
CHARACTER*4 TITLE
CHARACTER*8 TIMLBL

COMMON /PRINTC/ KTPSIG, CVRTIM
LOGICAL CVRTIM

COMMON /BASEAG/ NTERMS, XILEAD(2), XI(500), XITAIL(502),
+ VLEAD(2), V(S00), VTAIL(502)
DIMENSION XII(1004), VI(1004)
EQUIVALENCE (XII(1), XILEAD(1)), (VI(1), VLEAD(1))

COMMON /SIGPAR/ KGMH, NRCURV, IUPOWN, XMACH, VLIFT, TO,
+ PH10, SIGMA, XX(3), OMEGA, PK(3), XXS(3), XK(3),
+ XKF(3), PFACT, NAGES, AGES(20)
COMMON /SIGPAC/ IDENT, RAYNAM
CHARACTER*8 IDENT, RAYNAM
INTEGER KONK, NRCURV, IUPOWN

COMMON /RAYOUT/ SSIGMA, SPHIO, SXK(3), OPG, CSSEL
INTEGER NSIG
REAL CCSSEL, XXT(504), PPT(504)

DATA DGPRAD/57.295780/

IF (KTPSIG.LE.1) RETURN
TFACT=1000./OMEGA
PMAX=0.
PMIN=0.
DO 220 K=1, NTERMS
  V(K)=V(K)*PF
  PMAX=AMAX1(PMAX, V(K))
  PMIN=AMIN1(PMIN, V(K))
220 CONTINUE
PSIG = 0.05*(PMAX - PMIN)
KMAX = 1
KMIN = NTERMS
DO 225 K = 1, NTERMS
  IF (ABS(V(K)).LT.PSIG) GO TO 225
  KMIN = MIN0(KMIN, K)
  KMAX = MAX0(KMAX, K+2)
225 CONTINUE
DIR = OGP*GRAD*ATAN2(PK(1), PK(2)) + 180.
PN = SQRT(PK(1)**2 + PK(2)**2)
NSHOCK = 0
NN = NTERMS + 1
DO 250 K = 1, NN
  IF (XI(K+1).LT.XI(K+2)) GO TO 250
  IF (VI(K+1).GE.VI(K+2)) GO TO 250
  NSHOCK = NSHOCK + 1
  AXI = XI(K+1)
  TPR = AXI**TFACT
  XPR = AXI/PN
  PONE = VI(K+1)
  PTWO = VI(K+2)
  KMIN = MIN0(KMIN, K)
  KMAX = MAX0(KMAX, K+1)
250 CONTINUE
C
C - CALCULATE ABSOLUTE MAXIMUM OVERPRESSURE
C
MAXOP = AMAX1(ABS(PMIN), ABS(PMAX))

QPG = MAXOP
SIGMA = SIGMA
SPHIO = PHIO
DO 823 I = 1, 3
  SXX(1) = XX(1)
823 CONTINUE
IIJ = 0
DO 867 III = KMIN, KMAX
  IIJ = IIJ + 1
  XXT(IIJ) = (XI(III) - XI(KMIN))*TFACT
  PPT(IIJ) = VI(III)
867 CONTINUE
NSIG = KMAX - KMIN + 1
CALL CALSEL(PPT, XXT, NSIG, CCSEL)
CSEL = CCSEL
CALL PRTSNG(XXT, PPT, NSIG, OMEGA)
500 RETURN
END
SUBROUTINE CPVAL(XIARG,V,*)

CPVAL computes the value of the integral defining the Hilbert transform, as a Cauchy principal value, at each point directed by Hilbert.

COMMON /XISAVE/ NSTRMS,XIS(502),VS(502)
REAL SUM,ALPHA,RATIO,VSA,VS8,DIF1,DIF2,DIF3,P1,DIFA,DIFB
DATA P1/3.141592653589790+0/

SUM=0.0
DO 50 K=2,NSTRMS
  DIFA=(XIS(K))-XIARG
  DIFB=(XIS(K-1))-XIARG
  DIF2=DIFA
  DIF1=DIFB
  IF (ABS(DIF2).GE.ABS(DIF1)) GO TO 5
  DIF3=0IF2
  DIF2=0IF1
  DIF1=0IF3
  5 IF (DIF1.NE.O.) GO TO 15
  IF (DIF2.NE.O.) GO TO 7
  IF (VS(K).EQ.VS(K-1)) GO TO 50
  RETURN
  7 ALPHA=-ALOG(ABS(DIF2))/DIF2
  GO TO 30
  15 RATIO=(DIF1-DIF2)/DIF2
  IF (ABS(RATIO).LT.50-4) GO TO 20
  ALPHA=ALOG(ABS(DIF1/DIF2))/(DIF1-DIF2)
  GO TO 30
  20 ALPHA=((.250*RATIO+1.0/3.0)*RATIO+.50)*RATIO+& 1.0)/DIF2
  30 VSA=VS(K)
  VS8=VS(K-1)
  SUM=SUM+(-VSA*DIFB+VS8*DIFA)*ALPHA
  50 CONTINUE
  V=SUM/P1
  RETURN
END
SUBROUTINE SORTEP4

SORTEM SORTS THE VALUES CALCULATED BY HILBERT AND CPVAL INTO
ASCENDING ORDER OF ABSCISSA VALUES, AS REQUIRED BY AGING.

COMMON /BASEAG/ NTERMS,XILEAD(2),XI(1000),XITAIL(2),
+ VLEAD(2),V(1000),VTAIL(2)

LSTRT2=1
LSIZE=1
10 LSTRT1=LSTRT2
LSTOP1=LSTRT1+NTERMS-1
LSTRT2=NTERMS+2-LSTRT2
KC=LSTRT2
KSTOPB=LSTRT1-1
20 KSTRTA=KSTOPB+1
KSTOPA=MINO(KSTRTA+LSIZE-1,LSTOP1)
KSTRTB=KSTOPA+1
KSTOPB=MINO(KSTRTB+LSIZE-1,LSTOP1)
IF (KSTRTA.GT.KSTOPA) GO TO 90
30 IF (KSTRTB.GT.KSTOPB) GO TO 70
IF (KSTRTA.GT.KSTOPA) GO TO 50
IF (XI(KSTRTA)-XI(KSTRTB)) 36,33,40
33 IF (V(KSTRTA).GT.V(KSTRTB)) GO TO 40
36 XI(KC)=XI(KSTRTA)
V(KC)=V(KSTRTA)
KC=KC+1
KSTRTA=KSTRTA+1
GO TO 30
40 XI(KC)=XI(KSTRTB)
V(KC)=V(KSTRTB)
KC=KC+1
KSTRTB=KSTRTB+1
GO TO 30
50 IF (KSTRTB.GT.KSTOPB) GO TO 20
DO 60 K=KSTRTB,KSTOPB
XI(KC)=XI(K)
V(KC)=V(K)
KC=KC+1
60 CONTINUE
GO TO 20
70 IF (KSTRTA.GT.KSTOPA) GO TO 20
DO 80 K=KSTRTA,KSTOPA
XI(KC)=XI(K)
V(KC)=V(K)
KC=KC+1
80 CONTINUE
GO TO 20
90 LSIZE=LSIZE+LSIZE
IF (LSTRTZ.NE.1) GO TO 10
IF (LSIZE.LT.WTERMS) GO TO 10
RETURN
END
**PHYSICAL UTILITY ROUTINES - AIR,PHELEV,PHAZIM,EAMENU**

SUBROUTINE AIR(Z)

AIR is called to produce, at a specified altitude within a specified layer, the values of the sound speed and wind velocity, the first and second derivatives of those quantities with respect to height, and the density of the atmosphere. Uses linear interpolation of wind speed, wind direction, virtual temperature, and gamma with respect to geopotential height; the other quantities are derived from algebra and a hydrostatic assumption.

REAL Z,H,ZFACT
REAL DH,H1,H2,T,DMW,H1W,H2W,SPD,THETA,ST,CT,DTH,DH

COMMON /PTH/ NPTH,PRESS(97),TMPMOL(97),GPHC(97),GAMMA(97)
COMMON /WINDS/ WWINDS,GPHW(80),DIR(80),TURN(79),SPEED(80)
COMMON /LYRDEF/ NLAGH,GMZA(200),INDPTH(200),INDWND(200),
+ LYPRT(200),KLAYER,ZTOP,ZBOT
LOGICAL LYPRT
INTEGER INDPTH,INDWND

COMMON /ATMCON/ REALNH,GO,STAR,ROMO,ROGOMO
COMMON /ATMSPH/ GAM,C,U,V,CDZ,OLZ,DVOZ,D2CD2Z,D2VD2Z,RHO
REAL GAM,C,U,V
REAL RADPOG
COMMON /PRESUR/ CC
REAL PRSSU,CC

DATA RADPOG/1.74532925199433D-2/

F1A(TAU)=((TAU/5.+1.)*TAU/4.+1.)*TAU/3.+1.)*TAU/2.+1.
F1C(TAU)=EXP(TAU)-1./TAU

NLPTH=INDPTH(KLAYER)
NLWND=INDWND(KLAYER)
ZFACT=1.0+2/REALNH
H=Z/ZFACT
DH=1.0/ZFACT**2
D2H=2.*DH/(REALNH*ZFACT)
DH=GPHC(NLPTH-1)-GPHC(NLPTH)
H1=(H-GPHC(NLPTH))/DH
H2=(GPHC(NLPTH+1)-H)/DH
T=M*TMPMOL(NLPTH+1)+H2*TMPMOL(NLPTH)
DTH=(TMPMOL(NLPTH+1)-TMPMOL(NLPTH))/DH
GAM = H1*GAMMA(NLPTH + 1) + H2*GAMMA(NLPTH)
DGANDN = (GAMMA(NLPTH + 1) - GAMMA(NLPTH)) / DH
C = SQRT(GAMMA*ROMO*T)
DCDH = 0.5*C*(DH/T + DGANDN/GAM)
D2CDH2 = 0.25*C*(DGANDN/GAM - DH/T)**2
D2CDZ = DCDH*DHOZ
D2CDZ2 = DCDH*D2HOZ2 + D2CDH2*(DHOZ**2)
TAU = ALOG(T / TMPMOL(NLPTH))
IF (TAU .GT. 1) GO TO 5
FACTOR = TMPMOL(NLPTH)*F1S(TAU)
GO TO 10
5 FACTOR = TMPMOL(NLPTH)*F1A(TAU)
10 PRS = PRESS(NLPTH)*EXP(-H1*DH/(ROGOMO*FACTOR))
PRSS = (PRS)
C =-
RHO = PRS/(T*ROMO)
RHO = PRS*1.0E3/(T*ROMO)
DHW = GPHW(NLWND + 1) - GPHW(NLWND)
H1W = (H - GPHW(NLWND))/DHW
H2W = (GPHW(NLWND + 1) - H)/DHW
SPD = H1W*SPEED(NLWND + 1) + H2W*SPEED(NLWND)
DSDH = (SPEED(NLWND + 1) - SPEED(NLWND))/DHW
DTNDH = TURK(NLWND)**RADPG
THETA = DIR(NLWND)**RADPG + OTHDH*H1W*DHW
CT = COS(THETA)
ST = SIN(THETA)
U = SPD*ST
V = SPD*CT
DUDH = SPD*CT*OTHDH - DSDH*ST
DVDH = SPD*ST*OTHDH - DSDH*CT
D2UDH2 = OTHDH*(SPD*ST*OTHDH - 2.*DSDH*CT)
D2VHDH2 = OTHDH*(SPD*CT*OTHDH + 2.*DSDH*ST)
DUDZ = OUDH*DHOZ
DVDZ = OVDH*DHOZ
D2UDZ2 = OUDH*D2HOZ2 + OUDH2*(DHOZ**2)
D2VDZ2 = OVDH*D2HOZ2 + OVDH2*(DHOZ**2)
CC = C
RETURN
END
FUNCTION PHELEV(DUMMY)

PHELEV, GIVEN THE COMPONENTS OF THE WAVE NUMBER VECTOR, CALCULATES THE ELEVATION ANGLE OF THE NORMALS TO THE PHASE SURFACES OF THE WAVE.

COMMON /RAYVAR/ ZDIR,PKK,RTPAAO,ATTEN,SIGMA,X,Y,Z,DAGE,XF,YF,ZF,
+ XT,YT,ZT,XS,YS,ZS,XSS,YSS,ZSSS,XXSSS,YSSS,ZSSS,P3,
+ PSF,P3T,P3S,XFS,YFS,XTS,YTS,2FTP3,XFTZ,YFTZ,2FTZ,
+ ZFA,ZTA,P3FTZ,P3FA,P3TA,AREA,DAGDS

REAL SIGMA,X,Y,Z,DAGE,XF,YF,ZF,XT,YT,ZT
DATA DGPRAD/57.295780/

PHELEV=DGPRAD*ATAN2(P3,RTPAAO)
RETURN
END
FUNCTION PHAZIM(DUMMY)

PHAZIM, GIVEN THE COMPONENTS OF THE WAVE NUMBER VECTOR, CALCULATES THE AZIMUTH ANGLE OF THE NORMALS TO THE PHASE SURFACES OF THE WAVE.

COMMON /RAYINIT/ KGMH,NDCRVS,NUCRVS,IUPDWN,TO,PH10,XO,YO,ZO,
+ P10,P20,P30,OMEGA,DELTAO,P1FO,P2FO,P3FO,OMEGAF,
+ XT0,YT0,ZT0,P1T0,P2T0,P3T0,OMEGAT,XS0,YS0,ZS0,
+ P3SO,RHOO,PCONST,NAGES,AGES(20)
INTEGER KGMH,NDCRVS,NUCRVS,IUPDWN
DATA DGPRAD/57.295780/

PHAZIM=DGPRAD*ATAN2(-P10,-P20)+180.
RETURN
END
SUBROUTINE EAMENU(ELEV, AZIM, MAG, EAST, NORTH, UP)

C EAMENU, GIVEN THE ELEVATION ANGLE, AZIMUTH ANGLE, AND MAGNITUDE OF A
C VECTOR, CALCULATES THE EAST, NORTH, AND UPWARD COMPONENTS OF THAT
C VECTOR.

REAL MAG, NORTH, DGPRAD
DATA DGPRAD/57.295780/

HSQ=EAST**2+NORTH**2
IF (HSQ.NE.0.) GO TO 5
AZIM=0.
GO TO 10
5 AZIM=DGPRAD*ATAN(-EAST,-NORTH)+180.
IF (AZIM.LE.0.) AZIM=360.
10 MAG=SQRT(HSQ+UP**2)
IF (MAG.LE.0.) GO TO 20
HORIZ=SQRT(HSQ)
ELEV=DGPRAD*ATAN2(UP,HORIZ)
RETURN
20 ELEV=0.
RETURN
END
SUBROUTINE UNITIS(GIVEN,TABLE,NTABS,LCUNIT,TYPE,IDEFLT)

    CHARACTER*8 GIVEN, TABLE(NTABS), TYPE, BLANK
    DATA BLANK/' '/

    CALL LOKUP(8, NTABS, TABLE, GIVEN, LCUNIT, *5, *10)
    RETURN

5 IF(GIVEN.EQ.BLANK) GO TO 15

    WRITE(6,7) GIVEN, TYPE, TABLE(LCUNIT)
7 FORMAT(' AMBIGUOUS ABBREVIATION ' ,A8,' FOR ' ,A8,' UNIT. ' , +A8,' ASSUMED. ')
    RETURN

10 WRITE(7,12) TYPE, GIVEN
12 FORMAT(' INVALID ' ,A8,' UNIT SPECIFIED - ' ,A8,'. ')
    STOP 650

15 LCUNIT=IDEFLT
    RETURN
END
SUBROUTINE LOKUP(NCHAR,NTERMS,KTABL,KTEST,KTERM,-,**)

CHARACTER*8 KTABL(NTERMS), KTEST
CHARACTER BLANK,G

DATA BLANK/'I'/

GET FIRST CHARACTER OF STRING AND TEST FOR BLANK

G = KTEST(1:1)
IF (G.EQ.BLANK) RETURN 1

SEARCH TABLE FOR MATCH

KTERM = 0
DO 100 I = 1,NTERMS
   IF (KTEST.EQ.KTABL(I)) THEN
      KTERM = I
      RETURN
   ENDIF
100 CONTINUE

NO MATCH FOUND

RETURN 2
END
C
C
C
C
C./     ADD  NAME=FNDLYR
C
C    SUBROUTINE FNDLYR DEFINES THE LOCATION OF THE LAYRE IN THE ATMOSPHERIC LAYER IN WHICH A GIVEN ALTITUDE IS LOCATED.
C
C    SUBROUTINE FNDLYR(Z,*)
C
    COMMON /LYRDEF,NLAYER,GMZA(200),INDPHTH(200),INDOMWD(200),
    +LYRPRT(200),KLAYER,ZTOP,ZBOT
    INTEGER INDPHTH,INDOMWD
    LOGICAL LYRPRT

    CALL GETLYR(Z,GMZA,KLAYER,KLAYER,*50)
    ZBOT=GMZA(KLAYER)
    ZTOP=GMZA(KLAYER+1)

    RETURN

50 RETURN 1
END
SUBROUTINE GETLYR DOES A BINARY TABLE SEARCH OF THE ATMOSPHERIC TABLE TO FIND THE PROPER LAYRE INDEX.

SUBROUTINE GETLYR(XXTABL,NITEMS,NLAYR,*)

DIMENSION XTABL(NITEMS)

IF(X.LT.XTABL(1)) RETURN 1
IF(XTABL(NITEMS).LT.X) RETURN 1

N1=1
N2=NITEMS-1

IF(N2.LT.N1) RETURN 1

2 CONTINUE
NLAYR=(N1+N2+1)/2

5 N1=NLAYR
GO TO 2

10 N2=NLAYR-1
GO TO 2

40 RETURN
END
CONVERT TIME FROM HHHMMSS TO SSSSSS AND VICE-VERSA.

FUNCTION TIMCVR(T,KDIR)

COMMON /PRINTS/ TITLE(30),TIMLBL
CHARACTER*4 TITLE
CHARACTER*8 TIMLBL

COMMON /PRINTC/ KTPSIG,CVRTIM
LOGICAL CVRTIM

REAL T,HMS,SS,SSS,HMMSS
REAL ROUND,X,XNEAR,TIMCVR

C STATEMENT FUNCTIONS

SSS(HHMSS)=.2400.*AINT(HMS/1E4) .40.0*AINT(MS/100.0).HMS
HHMMSS(SS)=4000.0*AINT(SS/3600.0)+60.0*AINT(SS/60.0)+SS
ROUND(X,XNEAR)=SIGN(XNEAR*AINT(ABS(X/XNEAR)+.50),X)

IF (.NOT.CVRTIM) GO TO 50
IF (KDIR.LE.1) GO TO 30
TIMCVR=HHMMSS(ROUND(T,.1))
RETURN

30 TIMCVR=SSS(T)
RETURN

50 TIMCVR=T
RETURN

END
SUBROUTINE ACCVRT(ACTYP, CACTYP)

CHARACTER*8 ACTYP, CACTYP, TAC, TCAC
CHARACTER*1 AC(8), CAC(8)

EQUIVALENCE (TAC, AC), (TCAC, CAC)

TAC = ACTYP
TCAC = '1'

I = 0
J = 0

100 CONTINUE
  I = I + 1
  J = J + 1
  IF (AC(I).EQ.'--') THEN
    I = I + 1
  ENDIF
  CAC(J) = AC(I)
  IF (I.LT.8) GOTO 100

CACTYP = TCAC

RETURN
END
VECTOR MANIPULATION Routines

FUNCTION: DOTP

PROGRAMMER: PHILIP J. DAY

PURPOSE: TO COMPUTE THE SCALAR PRODUCT OF TWO N-DIMENSIONAL VECTORS

FUNCTION DOTP(V1, V2, N)
REAL V1(N), V2(N)
INTEGER N

REAL DP

DP = 0.0

DO 100 I = 1, N
   DP = DP + V1(I)*V2(I)
100 CONTINUE

DOTP = DP

RETURN
END
C
C----------------------------------------------------------------------------------
C
C FUNCTION: RNORM
C PROGRAMMER: PHILIP J. DAY
C
C PURPOSE: TO COMPUTE THE VECTOR NORM OF A N-DIMENTATIONAL VECTOR
C
FUNCTION RNORM(V1,N)
REAL V1(N)
INTEGER N
REAL D
D = 0.0
DO 100 I = 1,N
D = D + V1(I)**2
100 CONTINUE
RNORM = SQRT(D)
RETURN
END
SUBROUTINE: CROSS
PROGRAMMER: PHILIP J. DAY

PURPOSE: TO COMPUTE THE VECTOR PRODUCT OF TWO 3-DIMENSIONAL VECTORS

SUBROUTINE CROSS(V1,V2,CP)

REAL V1(3),V2(3),CP(3)

CP(1) = V1(2)*V2(3) - V1(3)*V2(2)
CP(2) = V1(3)*V2(1) - V1(1)*V2(3)
CP(3) = V1(1)*V2(2) - V1(2)*V2(1)

RETURN
END
SUBROUTINE: UNIT

PROGRAMMER: PHILIP J. DAY

PURPOSE: TO COMPUTE A UNIT VECTOR, VHAT, IN THE SAME DIRECTION AS V1.

SUBROUTINE UNIT(V1,VHAT,N)

REAL V1(N),VHAT(N)
INTEGER N

REAL A

A = RNORM(V1,N)

IF (A.EQ.0.0) A = 1.0

DO 100 I = 1,N
   VHAT(I) = V1(I)/A
100 CONTINUE

RETURN
END
C
C========================================================================================================================================
C========================================================================================================================================
C========================================================================================================================================
C========================================================================================================================================
C========================================================================================================================================
C========================================================================================================================================
C========================================================================================================================================
C========================================================================================================================================
C
C DOUBLE PRECISION VECTOR MANIPULATION ROUTINES
C
C========================================================================================================================================
C
C FUNCTION:  DDOTP
C PROGRAMMER: PHILIP J. DAY
C
C PURPOSE:  TO COMPUTE THE DOUBLE PRECISION SCALER PRODUCT OF TWO
C N-DIMENTIONAL VECTORS
C
C DOUBLE PRECISION FUNCTION DDOTP(V1,V2,N)
DOUBLE PRECISION V1(N),V2(N)
INTEGER N

DOUBLE PRECISION DP

DP = 0.0

DO 100 I = 1,N
   DP = DP + V1(I)*V2(I)
100 CONTINUE

DDOTP = DP

RETURN

END
PROGRAMMER: PHILIP J. DAY

PURPOSE: TO COMPUTE THE DOUBLE PRECISION VECTOR NORM OF A N-DIMENSIONAL VECTOR

DOUBLE PRECISION FUNCTION DRNORM(V1,N)

DOUBLE PRECISION V1(N)
INTEGER N

DOUBLE PRECISION D

D = 0.0

DO 100 I = 1,N
      D = D + V1(I)**2
100 CONTINUE

DRNORM = DSQRT(D)

RETURN
END
SUBROUTINE DCROSS

PROGRAMMER: PHILIP J. DAY

PURPOSE: TO COMPUTE THE DOUBLE PRECISION VECTOR PRODUCT OF TWO 3-DIMENSIONAL VECTORS

SUBROUTINE DCROSS(V1,V2,CP)

DOUBLE PRECISION V1(3),V2(3),CP(3)

CP(1) = V1(2)*V2(3) - V1(3)*V2(2)
CP(2) = V1(1)*V2(3) - V1(3)*V2(1)
CP(1) = V1(1)*V2(2) - V1(2)*V2(1)

RETURN
END
SUBROUTINE: DUNIT

PROGRAMMER: PHILIP J. DAY

PURPOSE: TO COMPUTE A DOUBLE PRECISION UNITVECTOR, VHAT, IN THE SAME DIRECTION AS V1.

SUBROUTINE DUNIT(V1, VHAT, N)

DOUBLE PRECISION V1(N), VHAT(N)
INTEGER N

DOUBLE PRECISION A

A = RNORM(V1, N)

DO 100 I = 1, N
   VHAT(I) = V1(I)/A
100 CONTINUE

RETURN

END
SUBROUTINE FFT (A, B, NTOT, N, NSPAN, ISN)

PURPOSE:

MULTIVARIATE COMPLEX FOURIER TRANSFORM.

MULTIVARIATE COMPLEX FOURIER TRANSFORM, COMPUTED IN PLACE USING
MIXED-RADIX FAST FOURIER TRANSFORM ALGORITHM. MULTIVARIATE DAT
INDEXED ACCORDING TO THE FORTRAN ARRAY ELEMENT SUCCESSOR FUNCTI
WITHOUT LIMIT ON THE NUMBER OF IMPLIED MULTIPLE SUBSCRIPTS. TH
SUBROUTINE IS CALLED ONCE FOR EACH VARIATE. THE CALLS FOR A MU
VARIATE TRANSFORM MAY BE IN ANY ORDER.

CATEGORIES:

CFT MIXED_RADIX_FFT FAST_FOURIER_TRANSFORM

REFERENCES:

CALL FFT (A, B, NTOT, N, NSPAN, ISN)

ARGUMENTS:

"A" (1:NTOT) REAL
ARRAYS A AND B ORIGINALLY HOLD THE REAL AND IMAGINARY COMPO
OF THE DATA, AND RETURN THE REAL AND IMAGINARY COMPONENTS OF
RESULTING FOURIER COEFFICIENTS.

"B" (1:NTOT) REAL
SEE ARRAY A, ABOVE.

-NTOT INTEGER
TOTAL NUMBER OF COMPLEX DATA VALUES IN ARRAYS A AND B.

-N INTEGER
THE LENGTH OF DIMENSION ALONG WHICH IT IS DESIRED TO TRANSFO

-NSPAN INTEGER
THE LENGTH OF THE TRANSFORM TIMES THE SPACING BETWEEN ELEMEN

-ISN INTEGER
DETERMINES THE SIGN OF THE COMPLEX EXPONENTIAL. THE MAGNITUD
OF ISN IS NORMALLY ONE.

"-" INDICATES AN INPUT PARAMETER;
"**" INDICATES BOTH AN INPUT AND OUTPUT PARAMETER.

**EXTERNALS:**

NONE

**FILES:**

- DEFAULT DEFAULT SEQUENTIAL FORMATTED
  ERROR MESSAGE IF HIGHEST PRIME FACTOR OF N IS GREATER THAN 2

"**" INDICATES OUTPUT ONLY.

**COMMONS:**

NONE

**DEPENDENCIES:**

IF THE INPUT DATA IS REAL, SEE 'REALTR' PROLOGUE.

**EXAMPLES:**

A TRI-VARIATE TRANSFORM WITH \( A(n_1, n_2, n_3), B(n_1, n_2, n_3) \) IS COMPUTED BY:

```
CALL FFT (A, B, n_1*n_2*n_3, n_1, n_1, 1)
CALL FFT (A, B, n_1*n_2*n_3, n_2, n_1*n_2, 1)
CALL FFT (A, B, n_1*n_2*n_3, n_3, n_1*n_2*n_3, 1)
```

FOR A SINGLE-VARIATE TRANSFORM WITH \( N_{TOT} = N = N_{SPAN} = \text{(NUMBER COMPLEX DATA VALUES)} \), FOR EXAMPLE:

```
CALL FFT (A, B, N, N, N, 1)
```

THE DATA MAY ALTERNATELY BE STORED IN A SINGLE COMPLEX ARRAY A, THE MAGNITUDE OF ISN CHANGED TO TWO TO GIVE THE CORRECT INDEX INCREMENT AND A(2) USED TO PASS THE INITIAL ADDRESS FOR THE SEQUENCE OF IMAGINARY VALUES; FOR EXAMPLE:

```
CALL FFT (A, A(2), N_{TOT}, N, N_{SPAN}, 2)
```

**NOTES:**

ARRAYS AT(MAXF), CK(MAXF), BT(MAXF), SK(MAXF), AND WP(MAXP) ARE USED FOR TEMPORARY STORAGE. IF THE AVAILABLE STORAGE IS INSUFFICIENT, THE PROGRAM IS TERMINATED BY A STOP.

MAXF MUST BE .GE. THE MAXIMUM PRIME FACTOR OF N.
MAXP MUST BE .GE. THE NUMBER OF PRIME FACTORS OF N.

IN ADDITION, IF THE SQUARE-FREE PORTION K OF N HAS TWO OR MORE PRIME FACTORS, THEN MAXP MUST BE .GE. K - 1.

ARRAY STORAGE IN NFAC FOR A MAXIMUM OF 11 FACTORS OF N. IF N HAS MORE THAN ONE SQUARE-FREE FACTOR, THE PRODUCT OF THE SQUARE-FRE
FACTORS MUST BE .LE. 210.

LIMITATIONS:

THE HIGHEST PRIME FACTOR OF N MUST NOT EXCEED 23.

HISTORY:

R. SINGLETON  01OCT68  STANFORD RESEARCH INSTITUTE.
M. FORSTER    16NOV76  ADDED TO LIBRARY.

DIMENSION A(1024),B(1024)
DIMENSION NFAC(11),NP(255)
DIMENSION AT(23),CK(23),BT(23),SK(23)
EQUIVALENCE (1,11)

THE FOLLOWING TWO CONSTANTS SHOULD AGREE WITH THE ARRAY DIMENSIONS.

MAXF=23
MAXP=255
IF(N .LT. 2) RETURN
INC=1SN
RAD=8.0*ATAN(1.0)
$72=RAD/5.0
CT2=COS($72)
$72=SIN($72)
$120=SQRT(0.75)
IF(ISN .GE. 0) GO TO 10
$72=$72
$120=$120
RAD=RAD
INC=INC
10 NT=INC*NTOT
KSN=INC*NSPAN
KSPAN=KSN
NN=NT-INC
JC=KSN/N
RADF=RAD*FLOAT(JC)*0.5
I=0
JF=0

DETERMINE THE FACTORS OF N

M=0
K=N
GO TO 20
15 M=M+1
NFAC(M)=4
K=K/16
20 IF(K-K/16)*16 .EQ. 0 GO TO 15
J=3
JJ=9
GO TO 30
25 M=M+1
   NFAC(M)=J
   K=K/J
30 IF(MOD(K,JJ) .EQ. 0) GO TO 25
   J=J+2
   JJ=J+2
   IF(JJ .LE. K) GO TO 30
   IF(K .GT. 4) GO TO 40
   KT=M
   NFAC(K+1)=K
   IF(K .NE. 1) M=M+1
   GO TO 80
40 IF(K<(K/4)*4 .NE. 0) GO TO 50
   M=M+1
   NFAC(M)=2
   K=K/4
50 KT=M
   J=2
60 IF(MOD(K,J) .NE. 0) GO TO 70
   M=M+1
   NFAC(M)=J
   K=K/J
70 J=(J-1)/2+1
   IF(J .LE. K) GO TO 60
80 IF(KT .EQ. 0) GO TO 100
   J=KT
90 M=M+1
   NFAC(M)=NFAC(J)
   J=J-1
   IF(J .NE. 0) GO TO 90
C COMPUTE FOURIER TRANSFORM
C 100 SD=RADF/FLOAT(KSPAN)
   CD=2.0*SIN(SD)**2
   SD=SIN(SD+SD)
   KK=1
   I=I+1
   IF(NFAC(I) .NE. 2) GO TO 400
C TRANSFORM FOR FACTOR OF 2 (INCLUDING ROTATION FACTOR)
C 210 K2=KK*KSPAN
   AK=A(K2)
   BK=B(K2)
   A(K2)=A(KK)-AK
   B(K2)=B(KK)-BK
   A(KK)=A(KK)+AK
   B(KK)=B(KK)+BK
   KK=K2*KSPAN
   IF(KK .LE. MM) GO TO 210
IF(KK .LE. JC) GO TO 210
IF(KK .GT. KSPAN) GO TO 800

220 C1=1.0-CD
S1=SD

230 K2=KK+KSPAN
AK=A(KK)-A(K2)
BK=B(KK)-B(K2)
A(KK)=A(KK)+A(K2)
B(KK)=B(KK)+B(K2)
A(K2)=C1*AK-S1*BJ
B(K2)=S1*AK+C1*BJ

KX=K2+KSPAN
IF(KK .LT. NT) GO TO 230
K2=KK+NT
C1=C1
KK=K1+K2

IF(KK .GT. K2) GO TO 230
AK=C1-(CD*C1+SD*S1)
S1=(SD*C1+CD*S1)+S1

THE FOLLOWING THREE STATEMENTS COMPENSATE FOR TRUNCATION ERROR. IF ROUNDED ARITHMETIC IS USED, SUBSTITUTE
C=AK

C1=0.5/(AK**2+S1**2)+0.5
S1=C1*S1
C1=C1*AK
KK=KK+JC

IF(KK .LT. K2) GO TO 230
K1=K1+INC+INC
KK=(K1-KSPAN)/2+JC
IF(KK .LE. JC+JC) GO TO 220
GO TO 100

TRANSFORM FOR FACTOR OF 3 (OPTIONAL CODE)
C

320 K1=KK+KSPAN
K2=K1+KSPAN
AK=A(KK)
BK=B(KK)
AJ=A(K1)+A(K2)
BJ=B(K1)+B(K2)
A(KK)=AK+AJ
B(KK)=BK+BJ
AK=-0.5*AJ+AK
BK=-0.5*BJ+BK
AJ=(A(K1)+A(K2))*S120
BJ=(B(K1)+B(K2))*S120
A(K1)=AK-BJ
B(K1)=BK+AJ
A(K2)=AK+BJ
B(K2)=BK-AJ
KK=K2+KSPAN
IF(KK .LT. NN) GO TO 320
KK=KK-NN
IF(KK .LE. KSPAN) GO TO 320
GO TO 700

C
C TRANSFORM FOR FACTOR OF 4
C
400 IF(NFAC(I) .NE. 4) GO TO 600
KSPNN=KSPAN
KSPAN=KSPAN/4
410 C1=1.0
S1=0
420 K1=KK+KSPAN
K2=K1+KSPAN
K3=K2+KSPAN
AKP=A(KK)+A(K2)
AKM=A(KK)-A(K2)
AJP=A(K1)+A(K3)
AJM=A(K1)-A(K3)
A(KK)=AKP+AJP
AJP=AKP-AJP
BKP=B(KK)+B(K2)
BKM=B(KK)-B(K2)
BJP=B(K1)+B(K3)
BJM=B(K1)-B(K3)
B(KK)=BKP-BJP
BJP=BKP-BJP
IF(ISM .LT. 0) GO TO 450
AKP=AKM-BJM
AKM=AKM+BJM
BKP=AKM-AJM
BKM=AKM-AJM
IF(S1 .EQ. 0.0) GO TO 460
430 A(K1)=AKP*C1-BKP*S1
B(K1)=AKM*S1+BKP*C1
A(K2)=AJP*C2-BJP*S2
B(K2)=AJM*S2+BJP*C2
A(K3)=AKM*C3-BKM*S3
B(K3)=AJM*S3+BKM*C3
KK=K3+KSPAN
IF(KK .LE. NT) GO TO 420
440 C2=C1-(CD*C1+SD*S1)
S1=(SD*C1-CD*S1)+S1

C
C THE FOLLOWING THREE STATEMENTS COMPENSATE FOR TRUNCATION ERROR. IF ROUNDED ARITHMETIC IS USED, SUBSTITUTE
C
C C2=C2
C
C C1=0.5/(C2**2+S1**2)+0.5
S1=C1*S1
C1=C1*C2
C2=C1**2-S1**2
S2=2.0*C1*S1
C3=C2*C1-S2*S1
S3+C2*S1+S2*C1
KK=KK+NT+JC
IF(KK.LE.KSPAN) GO TO 420
KK=KK-KSPAN+INC
IF(KK.LE.JK) GO TO 410
IF(KSPAN.EQ.JC) GO TO 800
GO TO 100

450 AKP=AKM+BJM
AKM=AKM-BJM
BKP=BKM-AJM
BKM=BKM+AJM
IF(S1. NE. 0.0) GO TO 430

460 A(K1)=AKP
B(K1)=BKP
A(K2)=AJP
B(K2)=BJP
A(K3)=AKM
B(K3)=BKM
KK=K3+KSPAN
IF(KK.LE.WT) GO TO 420
GO TO 460

C
C TRANSFORM FOR FACTOR OF 5 (OPTIONAL CODE)
C
510 C2=C72**2-S72**2
S2=2.0*C72*S72
520 K1=KK+KSPAN
K2=K1+KSPAN
K3=K2+KSPAN
K4=K3+KSPAN
AKP=A(K1)+AK(K4)
AKM=A(K1)-AK(K4)
BKP=B(K1)+B(K4)
BKM=B(K1)-B(K4)
AJP=A(K2)+A(K3)
AJM=A(K2)-A(K3)
BJP=B(K2)+B(K3)
BJM=B(K2)-B(K3)
AA=A(KK)
BB=B(KK)
A(KK)=AA+AKP+AJP
B(KK)=BB+BKP+BJP
AK=AKP*C72+AJP*C2+AA
BK=BKP*C72+BKP*C2+BB
AJ=AKM*S72-AJM*S2
BJ=BKM*S72+BJM*S2
A(K1)=AK-BJ
A(K4)=AK-BJ
B(K1)=BE-AJ
B(K4)=BE-AJ
AK=AKP*C2+AJP*C72+AA
BK=BKP*C2+BKP*C72+BB
AJ=AKM*S2-AJM*S72
BJ=BKM*S2+BJM*S72
```
A(K2) = AK + BJ
A(K3) = AK + BJ
B(K2) = BK + AJ
B(K3) = BK - AJ
KK = K4 + KSPAN
IF(KK .LT. NN) GO TO 520
KK = KK - NN
IF(KK .LE. KSPAN) GO TO 520
GO TO 700
C
C TRANSFORM FOR ODD FACTORS
C
600 K = NFAC(I)
   KSPHN = KSPAN
   KSPAN = KSPAN / K
   IF(K .EQ. 3) GO TO 320
   IF(K .EQ. 5) GO TO 510
   IF(K .EQ. JF) GO TO 640
   JF = K
   S1 = RAD / FLOAT(K)
   C1 = COS(S1)
   S1 = SIN(S1)
   IF(JF .GT. MAXF) GO TO 998
   CX(JF) = 1.0
   SK(JF) = 0.0
   J = 1
630 CX(J) = CK(K) * C1 + SK(K) * S1
   SK(J) = CK(K) * S1 - SK(K) * C1
   K = K - 1
   CK(K) = CK(J)
   SK(K) = -SK(J)
   J = J + 1
   IF(J .LT. K) GO TO 630
640 K1 = K
   K2 = KK + KSPHN
   AA = A(KK)
   BB = B(KK)
   AK = AA
   BK = BB
   J = 1
   K1 = K1 + KSPAN
650 K2 = K2 + KSPAN
   J = J + 1
   AT(J) = A(K1) + A(K2)
   AK = AT(J) + AK
   BT(J) = B(K1) + B(K2)
   BK = BT(J) + BK
   J = J + 1
   AT(J) = A(K1) - A(K2)
   BT(J) = B(K1) - B(K2)
   K1 = K1 + KSPAN
   IF(K1 .LT. K2) GO TO 650
   A(KK) = AK
   B(KK) = BK
```
K1=KK
K2=KK+KSPAN
J=1
660 K1=K1+KSPAN
K2=K2-KSPAN
JJ=J
AK=AA
BK=BB
AJ=0.0
BJ=0.0
K=1
670 KK=1
AK=AT(K)*CK(JJ)+AK
BK=BT(K)*CK(JJ)+BK
K=K+1
AJ=AT(K)*SK(JJ)+AJ
BJ=BT(K)*SK(JJ)+BJ
JJ=JJ+J
IF(JJ .GT. JF) JJ=JJ-JF
IF(K .LT. JF) GO TO 670
K=JF-J
A(KT)=AK-BJ
B(KT)=BK+AJ
A(K2)=AK+BJ
B(K2)=BK-AJ
J=J+1
IF(J .LT. K) GO TO 660
KK=KK+KSPAN
IF(KK .LE. M) GO TO 700
IF(KK .LT. KSPAN) GO TO 640
KK=KK+KSPAN
IF(KK .LE. KSPAN) GO TO 640
C
C MULTIPLY BY ROTATION FACTOR (EXCEPT FOR FACTORS OF 2 AND 4)
C
700 IF(I .EQ. M) GO TO 800
KK=JC+1
710 C2=1.0-C
S1=SD
720 C1=C2
S2=S1
KK=KK+KSPAN
730 AK=A(KK)
A(KK)=C2*AK-S2*B(KK)
B(KK)=S2*AK+C2*B(KK)
KK=KK+KSPAN
IF(KK .LE. NT) GO TO 730
AK=S1*S2
S2=S1*C2+C1*S2
C2=C1*C2-AK
KK=KK-NT+KSPAN
IF(KK .LE. KSPAN) GO TO 730
C2=C1-(CD*C1+SD*S1)
S1=S1-(SD*C1-CD*S1)
C THE FOLLOWING THREE STATEMENTS COMPENSATE FOR TRUNCATION ERROR. IF ROUNDED ARITHMETIC IS USED, THEY MAY BE DELETED.

C

C C1=0.5/(C2**2+S1**2)+0.5
S1=C1*S1
C2=C1*C2
KK=KK+KSPAN+JC
IF(KK .LE. KSPAN) GO TO 720
KK=KK-KSPAN+JC+INC
IF(KK .LE. JC+JC) GO TO 710
GO TO 100

C C PERMUTE THE RESULTS TO NORMAL ORDER---DONE IN TWO STAGES
C PERMUTATION FOR SQUARE FACTORS OF N

C

800 NP(1)=KS
IF(KT .EQ. 0) GO TO 890
KT=KT+1
IF(M .LT. K) K=K-1
J=1
NP(K+1)=JC
810 NP(J+1)=NP(J)/NFAC(J)
NP(K)=NP(K+1)/NFAC(J)
J=J+1
K=K-1
IF(J .LT. K) GO TO 810
K3=NP(K+1)
KSPAN=NP(2)
KK=JC+1
K2=KSPAN+1
J=1
IF(N .NE. NTOT) GO TO 850
C
C PERMUTATION FOR SINGLE-VARIATE TRANSFORM (OPTIONAL CODE)

C

820 AK=A(KK)
A(KK)=A(K2)
A(K2)=AK
BK=B(KK)
B(KK)=B(K2)
B(K2)=BK
KK=KK+INC
K2=KSPAN+K2
IF(K2 .LT. KS) GO TO 820
830 K2=K2-NP(J)
J=J+1
K2=NP(J+1)+K2
IF(K2 .GT. NP(J)) GO TO 830
J=1
840 IF(KK .LT. K2) GO TO 820
KK=KK+INC
K2=KSPAN+K2
IF(K2 .LT. KS) GO TO 840
IF(KK .LT. KS) GO TO 830
JC=K3
GO TO 890

C PERMUTATION FOR MULTIVARIATE TRANSFORM

850 K=KK+JC
860 AK=A(KK)
A(KK)=A(K2)
A(K2)=AK
BK=B(KK)
B(KK)=B(K2)
B(K2)=BK
KK=KK+INC
K2=K2+INC
IF(KK .LT. K) GO TO 860
KK=KK+KS-JC
K2=K2+KS-JC
IF(KK .LT. NT) GO TO 850
K2=K2+KSPAN+K2
KK=KK+KSPAN
IF(K2 .LT. KS) GO TO 850

870 K2=K2-NP(J)
J=J+1
K2=NP(J+1)+K2
IF(K2 .GT. NP(J)) GO TO 870
J=1

880 IF(KK .LT. K2) GO TO 850
KK=KK+JC
K2=KSPAN+K2
IF(K2 .LT. KS) GO TO 880
IF(KK .LT. KS) GO TO 870
JC=K3

890 IF(2*KT+1 .GE. M) RETURN
KSPNN=NP(KT+1)

C PERMUTATION FOR SQUARE-FREE FACTORS OF N

J=MM-KT
NFAC(J+1)=1
900 NFAC(J)=NFAC(J)*NFAC(J+1)
J=J+1
IF(J .NE. KT) GO TO 900
KT=KT+1
NN=NFAC(KT)-1
IF(NN .GT. MAXP) GO TO 998
JJ=0
J=0
GO TO 906
902 JJ=JJ-K2
K2=KK
K=K+1
KK=NFAC(K)
904 JJ=KK+JJ
IF(JJ .GE. K2) go to 902

NP(J)=JJ

906 K2=MFACT(KT)
    K=KT+1
    KK=MFACT(K)
    J=J+1
    IF(J .LE. NN) go to 904

C DETERMINE THE PERMUTATION CYCLES OF LENGTH GREATER THAN 1

C

J=0
    GO to 914

910 K=KK
    KK=NP(K)
    NP(K)=KK
    IF(KK .NE. J) go to 910
    KS=KK

914 J=J+1
    KK=NP(J)
    IF(KK .LT. 0) go to 914
    IF(KK .NE. J) go to 910
    NP(J)=J
    IF(J .NE. NN) go to 914
    MAXF=1NC*MAXF

C REORDER A AND B, FOLLOWING THE PERMUTATION CYCLES

C

GO to 950

924 J=J-1
    IF(NP(J) .LT. 0) go to 924
    JJ=JC

926 KSPAN=JJ
    IF(JJ .GT. MAXF) KSPAN=MAXF
    JJ=JJ-KSPAN
    K=NP(J)
    KK=JC*K+1+JJ
    K1=KK+KSPAN
    K2=0

928 K2=K2+1
    AT(K2)=A(K1)
    BT(K2)=B(K1)
    K1=K1-INC
    IF(K1 .NE. KK) go to 928

932 K1=KK+KSPAN
    K2=K1-JC*(K+NP(K))
    K=NP(K)

936 A(K1)=A(K2)
    B(K1)=B(K2)
    K1=K1-INC
    K2=K2-INC
    IF(K1 .NE. KK) go to 936
    KK=K2
    IF(K .NE. J) go to 932
    K1=KK+KSPAN
K2=0
940    K2=K2+1
       A(K1)=AT(K2)
       B(K1)=BT(K2)
       K1=K1-INC
       IF(K1 .NE. KK) GO TO 940
       IF(JJ .NE. 0) GO TO 926
       IF(J .NE. 1) GO TO 924
950    J=K3+1
       NT=NT-KSPMN
       I=NT-INC+1
       IF(NT .GE. 0) GO TO 924
       RETURN

C ERROR FINISH, INSUFFICIENT ARRAY STORAGE
C
998    ISN=0
       PRINT 999
       STOP
999    FORMAT(' ERROR FFT, INSUFFICIENT ARRAY STORAGE')
END
SUBROUTINE CALSEL

MODULE NAME: CALSEL

MODULE TYPE: SUBROUTINE

PROGRAMMER: THOMAS REILLY

DATE: MARCH 4, 1987

DESCRIPTION:

THIS SUBROUTINE IS DESIGNED TO CALCULATE THE CSEL FROM THE SIGNATURE. THIS IS DONE BY FIRST REMOVING THE SHOCKS FROM THE INPUTED PRESSURE ARRAY AND THEN DOING AN FFT ON THE ARRAY. THE CSEL VALUE IS THEN COMPUTED FROM THE FFT OUTPUT.

MODULE I/O: CALSEL(PRESS, PTIME, NPTS, CSEL)

INPUTS:

NPTS - INTEGER : NUMBER OF POINTS IN PRESS AND PTIME.
PRESS - REAL(504) : ARRAY CONTAINING THE SIGNATURE PRESSURES
PTIME - REAL(504) : ARRAY CONTAINING THE TIME ASSOCIATED W/P

OUTPUTS:

CSEL - REAL : CALCULATED SOUND EXPOSURE LEVEL IN DB.

VARIABLE DICTIONARY:

CF - REAL : USED TO COMPUTE THE C-WEIGHTED SPECTRUM
EIND - INTEGER : VALUE OF THE LAST INDICE NEEDED IN THE S TIME ARRAY.
ETIME2 - INTEGER : END OF SECONUD TIME ARRAY WITH A VALUE L THAN PTIME(NPTS).
GF - REAL : USED TO COMPUTE THE MEAN SQUARE SOUND PR LEVEL AT FREQUENCY F FOR BANDWIDTH (1/T)
ICNT - INTEGER : LOOP CONTOL VARIABLE.
LCE - REAL : USED FOR SUMATION OF THE C-WEIGHTED SOUND EXPOSURE LEVEL.
MAXIND - INTEGER : ARRAY INDICE FOR THE MAX PRESSURE.
MPRESS - REAL : MAX PRESSURE VALUE.
PRESS2 - REAL(1024) : SECONUD PRESSURE ARRAY WITH INTERPOLATE
PTIME2 - REAL(1024) : SECONUD TIME ARRAY WITH T INCREMENTS OF SCOUNT - INTEGER : NUMBER OF SHOCKS THAT HAVE BEEN FOUND.
SIND - INTEGER : STARTING INDICE FOR THE SECONUD TIME ARR
STIME2 - INTEGER : START OF SECONUD TIME ARRAY WITH A VALUE GREATER THAN PTIME(1).
TCNT1 - INTEGER : COUNTER FOR THE PTIME2 ARRAY.
TCNT2 - INTEGER : COUNTER FOR THE PTIME ARRAY.
TCNT3 - INTEGER : COUNTER FOR THE PTIME ARRAY.

CALLING MODULES:

CALLED MODULES:

FFT(PRESS2, PRESS, NPTS, NPTS, MPRESS, 1);

This subroutine performs a Fourier transform on the time in array PRESS2, with NPTS having a value of 512 or 1024.

SUBROUTINE CALSEL(TPRESS, TPTIME, NPTS, CSEL)

DECLARE SUBROUTINE PARAMETER INPUT/OUTPUT VARIABLES.
REAL TPRESS(504), TPTIME(504), CSEL
INTEGER NPTS

DECLARE SUBROUTINE DEPENDANT VARIABLES.
REAL PRESS(2048), PTIME(2048)
REAL PRESS2(2048), PTIME2(2048), MPRESS
REAL LCE, CF, CF1, CF2, CF3
INTEGER TCNT1, MAXIND, SCOUNT, SIND, EIND, ETIME2, STIME2, TCNT1, 1
TCNT2, TCNT3
COMPLEX IP(2048)

ASSIGN ARRAY TO SIZE 1024.
DO 3, I = 1, NPTS
    PRESS(I) = TPRESS(I)
    PTIME(I) = TPTIME(I)
3 CONTINUE

PAD WITH 0.0.
DO 4, I = NPTS+1, 2048
    PRESS(I) = 0.0
    PTIME(I) = 0.0
4 CONTINUE

INITIALIZE VARIABLES.
MPRESS = PRESS(1)
MAXIND = 0
SCOUNT = 0
* IF THE ARRAY DOESN'T START WITH A VALUE OF ZERO ADD ONE.
  IF ((PRESS(1) .NE. 0.0)) THEN
    NPTS = NPTS + 1
  DO 5, ICNT = NPTS, 2, -1
    PRESS(ICNT) = PRESS(ICNT - 1)
    PTIME(ICNT) = PTIME(ICNT - 1)
  5 CONTINUE
  PRESS(1) = 0.0
  PTIME(1) = PTIME(2) - 0.5
ENDIF

* IF LAST ELEMENT OF THE ARRAY ISN'T ZERO THEN ADD ONE.
  IF ((PRESS(NPTS) .NE. 0.0)) THEN
    NPTS = NPTS + 1
    PRESS(NPTS) = 0.0
    PTIME(NPTS) = PTIME(NPTS - 1) + 0.5
ENDIF

* TRAVERSE PRESSURE ARRAY FINDING MAX PRESS AND REMOVING SHOCKS.
  DO 10, ICNT = 2, NPTS
    IF A SHOCK IS FOUND INCREMENT SHOCK COUNTER BY ONE.
    IF (PTIME(ICNT - 1) .EQ. PTIME(ICNT) + (SCOUNT * 0.5))
      SCOUNT = SCOUNT + 1
    INCREASE TIME ARRAY BY A FACTOR OF .5 FOR EACH SHOCK FOUND.
    PTIME(ICNT) = PTIME(ICNT) + (SCOUNT * 0.5)
  FIND MAX PRESSURE.
  IF (PRESS(ICNT) .GT. MPRESS) THEN
    MPRESS = PRESS(ICNT)
    MAXIND = ICNT
  ENDIF

* END LOOP.
10 CONTINUE

* CALCULATE THE START AND END INDICES FOR THE SECOND TIME ARRAY.
  SIND = INT((PTIME(MAXIND) - PTIME(1)) / 0.5) + 1
  EIND = INT((PTIME(NPTS) - PTIME(MAXIND)) / 0.5) + (2 + SIND)
  IF (SIND.LT.1.OR.EIND.GT.2048) THEN
    CSEL = -1.0
    RETURN
  ENDIF
  PTIME2(SIND) = PTIME(MAXIND)
  PRESS2(SIND) = PRESS(MAXIND)

* TRAVERSE TIME ARRAY STARTING FROM THE MAX PRESS TO THE LAST ELEMENT
  DO 20, ICNT = (SIND + 1), EIND
* INCREMENT TIME FROM MAX PRESS IN 0.5 SECOND INCREMENTS.
  PTIME2(ICNT) = PTIME(MAXIND) + ((ICNT - SIND) * 0.5)

* END LOOP.
TRVERSE TIME ARRAY STARTING FROM MAX PRESS DOWN TO FIRST ELEMENT.
DO 30, ICNT = (SIND - 1), 1, -1

DECREMENT TIME FROM MAX PRESS IN 0.5 SECOND INCREMENTS.
PTIME2(ICNT) = PTIME(MAXIND) - ((SIND - ICNT) * 0.5)

END LOOP.
30 CONTINUE

STIME2 = 1
ETIME2 = EIND

FIND THE ELEMENT IN PTIME2 THAT IS GREATER THAN PTIME( ).
40 IF (PTIME2(STIME2) .LT. PTIME(1)) THEN
   STIME2 = STIME2 + 1
   GOTO 40
ENDIF

FIND THE ELEMENT IN PTIME2 THAT IS LESS THAN PTIME(NPTS).
50 IF (PTIME2(ETIME2) .GT. PTIME(NPTS)) THEN
   ETIME2 = ETIME2 - 1
   GOTO 50
ENDIF

TCNT1 = STIME2
TCNT3 = 2

TRVERSE TIME AND PRESSURE ARRAY INTERPOLATING PRESS FOR DELTA T.
60 IF (TCNT1 .LE. ETIME2) THEN

IF THE PRESENT ARRAY ELEMENT IS THE MAX PRESSURE THEN SKIP IT.
IF (TCNT1 .EQ. SIND) THEN
   TCNT2 = MAXIND
   TCNT3 = MAXIND + 1
   TCNT1 = TCNT1 + 1
ENDIF

FIND AN ELEMENT OF PTIME THAT IS LARGER THAN PTIME2.
70 IF (PTIME(TCNT3) .LT. PTIME2(TCNT1)) THEN
   TCNT3 = TCNT3 + 1
ENDIF

INTERPOLATE A VALUE FOR THE PRESS FOR PTIME2(TCNT1).
PRESS2(TCNT1) = (((PRESS(TCNT3) - PRESS(TCNT2)) / (PTIME
1 (TCNT3) - PTIME(TCNT2))) * (PTIME2(TCNT1) -
2 PTIME(TCNT2))) + PRESS(TCNT2)

MOVE TO THE NEXT ELEMENT OF PTIME2.
TCNT1 = TCNT1 + 1

END OF LOOP.
GOTO 60
END IF

*.
START PRESS2 AND TIME2 ARRAYS AT THE FIRST ELEMENT.
DO 80, ICNT = STIME2, ETIME2

PRESS2(ICNT) = (STIME2 - 1)) = PRESS2(ICNT)

PTIME2(ICNT) = (STIME2 - 1)) = PRESS2(ICNT)

80 CONTINUE

NPTS = (ETIME2 - STIME2) + 1

IF (NPTS.GT.2048) THEN

CSEL = -1.0

RETURN
END IF

*.
PAD THE ARRAY WITH ZERO'S
DO 90, ICNT = NPTS+1, 2048

PRESS2(ICNT) = 0.0

90 CONTINUE

DO 100, ICNT = 1, 2048

PTIME2(ICNT) = 0.0

100 CONTINUE

IF (NPTS .LE. 512) NPTS = 512
IF (NPTS .GT. 512) NPTS = 1024
IF (NPTS .GT. 1024) NPTS = 2048

*.
DO THE FFT.

CALL FFT(PRESS2, PTIME2, NPTS, NPTS, NPTS, 1)

*.
COMPUTE THE C WEIGHTED SOUND EXPOSURE LEVEL.

LCE = 0.0

T = NPTS * 0.0005

I = NPTS

NPTS = NPTS / 2

DO 102 ICNT = 1,I

IP(ICNT) = CMPLX(PRESS2(ICNT),PTIME2(ICNT))

102 CONTINUE

DO 105 ICNT = 1,NPTS

IP(ICNT) = IP(ICNT-1) * (0.5/I)

PRESS(ICNT) = ICNT/0.5

105 CONTINUE

DO 110, ICNT = 1, NPTS

GF = REAL((2/T) * IP(ICNT) * CONJG(IP(ICNT))))

IF (GF .NE. 0.0) GF = 10 * LOG10(GF/0.00002)**2

CF1 = 2.242881E16 * PRESS(ICNT)**4

CF2 = PRESS(ICNT)**2 + 20.598997**2

CF3 = (PRESS(ICNT)**2 + 12194.22**2)

CF = CF1 / ((CF2 * CF3)**2)

IF (CF .NE. 0.0) CF = 10 * LOG10(CF)
IF ((CF+GF) .NE. 0.0) LCE = LCE + 10 **((GF +CF) / 10.)
CONTINUE

* CONVERT THE OUTPUT INTO DB.
IF (LCE .NE. 0.0) THEN
    CSEL = 10 * ALOG10(LCE)
ENDIF

* END SUBROUTINE.
RETURN
END
SUBROUTINE PRTEGG(XXT,PPT,NPTS,CC)

REAL XXT(504),PPT(504)
INTEGER NPTS

COMMON /RAYOUT/ SSIGMA,SPHIO,SKX(3),OPG,CSEL

COMMON /STATS/ STATFG, BOOMFG, MACHFG, CONTFG, BOOMVL,
+ MACHVL, CONTVL(5,20), CONTYP(5), WIDTH, FFT,
+ SIGNAT, RAYTRC, SCRPAD, SCRPSF, SCRALL

LOGICAL LMFG, LMCF1, LMCF0, PLYFG, MRFG, PERFG
LOGICAL STATFG, MACHFG, BOOMFG, CONTFG
LOGICAL RAYTRC, SIGNAT, FFT, SCRPAD, SCRPSF, SCRALL
LOGICAL GPCPFL, GPCPMH, GPCPBM, GPCPCN, RAYINX

IF (.NOT.SIGNAT) RETURN

TFACT = 1000./CC

WRITE(20,2000) SSIGMA,SPHIO
WRITE(20,2001)
DO 100 I = 1,NPTS
   WRITE(20,2002) XXT(I)*TFACT,PPT(I)/47.85
100 CONTINUE

2000 FORMAT(' ! TO =',F12.3,' PHIO =',F8.3)
2001 FORMAT(' ******** PRESSURE SIGNATURE ********',//,6X,
+ 'TIME (MS)',9X,'PRESSURE (PSF)',//)
2002 FORMAT(4X,F12.3,8X,F10.4)

RETURN
END
SUBROUTINE RBRAYS

*.
*  MODULE NAME : RBRAYS
*  MODULE TYPE : SUBROUTINE
*.
*  PROGRAMMER : THOMAS REILLY
*  DATE : APRIL 20, 1987
*  REVISIONS :
*.
*  DESCRIPTION :
*  THIS SUBROUTINE WAS DESIGNED TO READ IN THE RAYS FROM
*  FILE, SORT THE RAYS ON THE PHI ANGLE AND THEN REMOVE THE 8A
*  RAYS. ONCE THIS IS DONE THE RAYS ARE WRITTEN BACK INTO THE
*  SEQUENTIAL ACCESS FILE.
*.
*  MODULE I/O : NONE.
*.
*.
*  FILE I/O :
*  ........
*.
*  ONE SEQUENTIAL ACCESS FILE NAME (?).
*.
*  VARIABLE DICTIONARY :
*  ---------------
*  TARR    - REAL(13,200) : ARRAY THE RAYS ARE SORTED IN.
*  RECNT   - INTEGER : NUMBER OF RECORDS IN TARR.
*  RFLAG   - INTEGER : FLAG FOR WHAT TYPE OF RAY IT IS.
*.

SUBROUTINE RBRAYS

COMMON /CARL/ BOMFCT
COMMON /RAYOUT/ TTO, TPHIO, TXK(3), TOPG, CSEL

INTEGER RECNT, K, RFLAG
REAL TARR(13,500)

REWIND(12)
RECNT = 1

*  READ INFORMATION FROM THE FILE INTO TARR.
10 CONTINUE
READ(12, 15, END=100) RFLAG, (TARR(K,RECNT), K=2,12)
TARR(1,RECNT) = RFLAG
TARR(10,RECNT) = TARR(10,RECNT) / 47.85
TARR(13,RECNT) = 0.0
RECNT = RECNT + 1
IF (RECNT.GT.500) THEN
    WRITE(6,*) ' MORE THEN 500 CAUSTIC RAYS IN RBRAY'S'
    GOTO 100
ENDIF
GOTO 10
15 FORMAT (12,1X,F10.2,3F8.0,F9.3,F10.2,2F8.0,F11.4,2F10.4)

*.-  SORT THE ARRAY TARR ON PHI ANGLE.
100 CONTINUE
    RECNT = RECNT - 1
    IF ((RECNT .LE. 1) .AND. (TARR(10,1) .EQ. 0.0)) RETURN
    IF (RECNT .LT. 3) GOTO 700
    CALL SORT(RECNT, TARR, 6)

*.-  SEARCH ARRAY FROM THE BEGINNING LOOKING FOR BAD RAYS.
DO 200, I = 1, (RECNT-2)
    IF (((TARR(10,1) .LT. TARR(10,1+1)) .AND. (TARR(10,1+1)
        1 .LT. TARR(10,1+2))) .OR. ((TARR(10,1) .LT. TARR(10,1+1))
        2 .AND. (TARR(10,1+1) .LT. TARR(10,1+2))) OR TARR(10,1+1)
        .LT. TARR(10,1)) .AND. (TARR(10,1) .LT. TARR(10,1+2)))
    AVRG = (TARR(10,1) + TARR(10,1+1) + TARR(10,1+2))/3.0
    DIF1 = ABS(TARR(10,1) - AVRG)
    DIF2 = ABS(TARR(10,1+1) - AVRG)
    DIF3 = ABS(TARR(10,1+2) - AVRG)
    XBAD = MAX(DIF1,DIF2,DIF3)
    XBAD = MAX(XBAD,ABS(TARR(10,1) - TARR(10,1+1))
    IF (XBAD .EQ. DIF1) IBAD = 1
    IF (XBAD .EQ. DIF2) IBAD = 2
    IF (XBAD .EQ. DIF3) IBAD = 3
    IF (IBAD .EQ. 2) THEN
        DEV1 = ABS(TARR(10,1) - TARR(10,1+1))
    DEV2 = ABS(TARR(10,1+1) - TARR(10,1+2))
    RDEV = MAX(DEV1,DEV2)
    IF (RDEV .EQ. 0) IBCNT = 0
    IF (RDEV .EQ. 2) IBCNT = 2
    DPRI = ABS(TARR(6,1+IBCT) - TARR(6,1+1))
    OSEC = RDEV / DPRI
    IF (OSEC .GE. 15.0) THEN
        TARR(13,1+1) = TARR(13,1+1) + 1.0
    ENDIF
    ENDIF
    ENDIF
200 CONTINUE
DO 300 I = RECNT, 3, -1
    IF (((TARR(10,1) .LT. TARR(10,1+1)) .AND. (TARR(10,1+1)
        1 .LT. TARR(10,1+2))) .OR. ((TARR(10,1) .LT. TARR(10,1+1))
        2 .AND. (TARR(10,1+1) .LT. TARR(10,1+2)))
    AVRG = (TARR(10,1) + TARR(10,1+1) + TARR(10,1+2))/3
    DIF1 = ABS(TARR(10,1) - AVRG)
    DIF2 = ABS(TARR(10,1+1) - AVRG)
    DIF3 = ABS(TARR(10,1+2) - AVRG)
PROGRAM XBAD

IMPLICIT NONE

EXTERNAL TARR, A, B, C

DIMENSION XBAD(10), YBAD(10)

DATA XBAD, YBAD / 0.0, 0.0 /

IF (XBAD .EQ. 1.0) THEN
  XBAD = XBAD + 1.0
END IF

IF (YBAD .EQ. 1.0) THEN
  YBAD = YBAD + 1.0
END IF
SUBROUTINE SORT(NPTS, TEMPAR, SRTFLD)

INTEGER NPTS, I, J, K, L, ARRL, SRTFLD

PARAMETER (ARRL = 13)
REAL TRAY(ARRL)
REAL TEMPAR(ARRL, 500)

* SET UP INITIALIZATION FOR HEAPSORT.
L = NPTS / 2 + 1
IR = NPTS

* HEAP CREATION PHASE.
30 CONTINUE
IF (L .GT. 1) THEN
    L = L - 1

* INITIALIZE RRA TO ELEMENT RA(L) IN THE RAY ARRAY.
DO 32, K = 1, ARRL
  TRAY(K) = TEMPAR(K,L)
32  CONTINUE
ELSE
  *- PLACE TOP OF HEAP AT THE END OF THE ARRAY.
  DO 34, K = 1, ARRL
  TRAY(K) = TEMPAR(K,IR)
  TEMPAR(K,IR) = TEMPAR(K,1)
34  CONTINUE
  IR = IR - 1
  *- PLACE SMALLEST ELEMENT AT THE BEGINING OF THE ARRAY.
  IF (IR .EQ. 1) THEN
    DO 36, K = 1, ARRL
    TEMPAR(K,1) = TRAY(K)
36  CONTINUE
  *- EXIT LOOP AND WRITE ARRAY BACK INTO THE RAY FILE.
    GO TO 100
  ENDIF.
ENDIF
I = L
J = L + L
*- SET UP TO SHIFT DOWN RRA TO ITS PROPER LEVEL
70  IF (J .LE. IR) THEN
    IF (J .LT. IR) THEN
      *- COMPARE THE RAY TERMINATION TIMES.
      IF (TEMPAR(SRTFLD,J) .LT. TEMPAR(SRTFLD,(J+1))) J = J + 1
      ENDIF
    *- COMPARE THE RAY TERMINATION TIMES.
    IF (TRAY(SRTFLD) .LT. TEMPAR(SRTFLD,J)) THEN
      DO 72, K = 1, ARRL
      TEMPAR(K,I) = TEMPAR(K,J)
72  CONTINUE
      I = J
      J = J + J
    ELSE
      *- THIS IS RRA'S LEVEL. SET J TO TERMINATE SHIFT DOWN
      J = IR + 1
    ENDIF
    *- LOOP WHILE J LESS THAN OR EQUAL TO IR.
    GO TO 70
  ENDIF
  *- PUT RRA INTO ITS SLOT.
  DO 74, K = 1, ARRL
  TEMPAR(K,1) = TRAY(K)
74  CONTINUE
LOOP UNTIL ARRAY IS SORTED.
GO TO 30

100 CONTINUE
RETURN
END
**SUBROUTINE LSQUAR**

**MODULE NAME**: LSQUAR
**MODULE TYPE**: SUBROUTINE
**PROGRAMMER**: THOMAS REILLY
**DATE**: MAY 7, 1987

**DESCRIPTION**

This subroutine is designed to accept a velocity vector which it then weights and computes a least square approximation for a polynomial of N degrees with NPTS.

```fortran
SUBROUTINE LSQUAR(TIME)

COMMON /SPLINE/ NPTS, S(100,3), A(100,3), B(100,3), C(100,3),
1 D(100,3)
PARAMETER (N = 2)
DIMENSION ACCVEC(100,3), TIME(100), T(6), ACWGHT(6,3)
DIMENSION E(20,21)
INTEGER NPTS, DET

IF (NPTS .LT. 3) RETURN

**READ ACCELERATION VECTOR INTO ACCVEC.**

DO 5 I = 1, 100
  DO 5 J = 1, 3
    ACCVEC(I,J) = C(I,J)
 5 CONTINUE

DO 10 I = 1, 3
  A(I,1) = 0.0
  B(I,1) = 0.0
  D(I,1) = 0.0
10 CONTINUE

**MOVE THE WEIGHT WINDOW ONE ELEMENT AT A TIME.**

DO 1000 I = 2, NPTS

**IF ITS THE FIRST OR LAST ELEMENT WEIGHT IT 1 2 1.**

IF ((I .EQ. 2) .OR. (I .EQ. NPTS)) THEN
  IF (I .EQ. 2) THEN
    K = 1
    L = 4
  ELSE
```

(CODE CONTINUES)

```fortran
ELSE

(CODE CONTINUES)

ENDIF
```
K = NPTS - 2
L = NPTS
ENDIF
IF ((J .EQ. K) .OR. (J .EQ. L)) THEN
    W = 1
ELSE
    W = 2
ENDIF
DO 50 J2 = 1,3
ACWGHT(1,J2) = ACCVEC(K,J2)
T(1) = TIME(K) - TIME(K+1)
DO 60 J = 2,3
    ACWGHT(J,J2) = ACCVEC(K+1,J2)
    CONTINUE
T(J) = 0.0
60 CONTINUE
DO 80 J2 = 1,3
ACWGHT(4,J2) = ACCVEC(K+2,J2)
T(4) = TIME(K+2) - TIME(K+1)
80 CONTINUE
CAPN = 4
*
IF ITS NOT THE FIRST OR LAST ELEMENT WEIGHT IT 1 2 2 1.
ELSE
K = 1 - 2
IF ((J .EQ. (1-2)) .OR. (J .EQ. (1+1))) THEN
    W = 1
ELSE
    W = 2
ENDIF
DO 150 J2 = 1,3
ACWGHT(1,J2) = ACCVEC(K,J2)
T(1) = TIME(K) - TIME(K+2)
DO 160 J = 2,3
    ACWGHT(J,J2) = ACCVEC(K+1,J2)
    T(J) = TIME(K+1) - TIME(K+2)
160 CONTINUE
DO 180 J = 4,5
    ACWGHT(J,J2) = ACCVEC(K+2,J2)
    CONTINUE
T(J) = 0.0
180 CONTINUE
DO 195 J2 = 1,3
    ACWGHT(6,J2) = ACCVEC(K+3,J2)
    T(6) = TIME(K+3) - TIME(K+2)
195 CONTINUE
CAPN = 6
ENDIF
DO 900 K1 = 1,3
DET = 2
NP1 = N + 1
DO 300 I1 = 1, NP1
   E(I1,N+2) = 0.0
   DO 300 J = 1, I1
      E(I1,J) = 0.
   300 CONTINUE
DO 500 K = 1, CAPN
   P1 = 1.0
   DO 500 I1 = 1, NP1
      P2 = 1.0
      DO 400 J = 1, I1
         E(I1,J) = E(I1,J) + P1 * P2
      P2 = P2 * T(K)
   400 CONTINUE
   E(I1,N+2) = E(I1,N+2) + P1 * ACWGHTE(K,K1)
   P1 = P1 * T(K)
500 CONTINUE
DO 600 I1 = 1, NP1
   DO 600 J = 1, I1
      E(J,I1) = E(I1,J)
   600 CONTINUE
CALL GAUSJR(N+1, DET, E, DELTA)
   A(I,K1) = E(3,N+2)
   B(I,K1) = E(2,N+2)
   C(I,K1) = E(1,N+2)
900 CONTINUE
1000 CONTINUE
RETURN
END
SUBROUTINE GAUSJR(N, DET, E, DELTA)

INTEGER DET
DIMENSION E(20,21)

M = N + 1
GOTO (3,6), DET

3 M = N

* INITIALIZE THE DETERMINANT.
6 DELTA = 1
DO 12 K = 1,N
   DELTA = DELTA * E(K,K)
   KP1 = K + 1
   DO 9 J = KP1,N
      E(K,J) = E(K,J) / E(K,K)
   CONTINUE
9 IF (KP1.GT. N) GOTO 13
   DO 12 I = KP1,N
      DO 12 J = KP1,M
         E(I,J) = E(I,J) - E(I,K) * E(K,J)
      CONTINUE
12 CONTINUE
13 IF (DET .EQ. 1) GOTO 18
   MM1 = N - 1
   DO 15 IND = 1, MM1
      K = N + 1 - IND
      KM1 = K - 1
      DO 15 I = 1, KM1
         E(I,M) = E(I,M) - E(I,K) * E(K,M)
      CONTINUE
15 CONTINUE
18 RETURN
END

SUBROUTINE STOREC(TITLE, GPCPFL, GPCPMH, GCPBM)

* THIS SUBROUTINE IS DESIGNED TO STORE NEEDED VARIABLES IN
  A TEMPORARY FILE SO THE PLOTTING/RAYTRACING CAN BE RUN AS
A two step process.

COMMON /CHRTABS/ ARCRFT, MSSNS, SITES, TAILNM

COMMON /INTTABS/ ENDATE, ENTIME, STDATE, STTIME, NUMREP

COMMON /STATS/ STATFG, BOOMFG, MACHFG, CONTFG, BOOMVL,
1 MACHVL, CONTVL, CONTYP, WIDTH, FFT,
2 SIGNAT, RAYTRC, SCRPAO, SCRPSF, SCRALL

INTEGER ENDATE(5,10), ENTIME(5,10), STDATES(5,10)
INTEGER STTIME(5,10), CONTYP(5), NUMREP

REAL CONTVL(5,20), BOOMVL, MACHVL, WIDTH

LOGICAL STATFG, BOOMFG, MACHFG, CONTFG, SIGNAT, RAYTRC,
1 SCRPAO, SCRPSF, SCRALL, GPCPFL, FFT, GPCPMH, GPCPBM

CHARACTER*70 TITLE
CHARACTER*6 ARCRFT(5,10)
CHARACTER*16 MSSNS(5,10)
CHARACTER*10 SITES(5,20)
CHARACTER*8 TAILNM(5,10)

OPEN(76,FILE='HOLDVAR',STATUS='UNKNOWN')

WRITE(76,FMT='(A)') TITLE
DO 10 I = 1, 5
 WRITE(76,FMT='(A)') (ARCRFT(I,J), J=1,10)
 WRITE(76,FMT='(A)') (MSSNS(I,J), J=1,10)
 WRITE(76,FMT='(A)') (SITES(I,J), J=1,20)
 WRITE(76,FMT='(A)') (TAILNM(I,J), J=1,10)
 WRITE(76,FMT='(18)') (ENTIME(I,J), J=1,10)
 WRITE(76,FMT='(18)') (ENDATE(I,J), J=1,10)
 WRITE(76,FMT='(18)') (STTIME(I,J), J=1,10)
 WRITE(76,FMT='(18)') (STDATE(I,J), J=1,10)
 WRITE(76,FMT='(F20.4)') (CONTVL(I,J), J=1,20)
10 CONTINUE
 WRITE(76,FMT='(5L1)') STATFG, BOOMFG, MACHFG, CONTFG, FFT
 WRITE(76,FMT='(6L1)') SIGNAT, RAYTRC, SCRPAO, SCRPSF, SCRALL,
1 GPCPFL
 WRITE(76,FMT='(3F20.4)') BOOMVL, MACHVL, WIDTH
 WRITE(76,FMT='(2L1,18)') GPCPMH, GPCPBM, NUMREP
 CLOSE(76)

RETURN
END
SUBROUTINE SCRPAD

MODULE NAME: SCRPAD

MODULE TYPE: SUBROUTINE

PROGRAMMER: HARRY SEIDMAN

DATE: DECEMBER 1986

REVISIONS: MARCH 1987, UPDATED TO BE CONSISTENT WITH BOOMAP2.

DESCRIPTION:
THE PURPOSE OF THIS SUBROUTINE IS TO READ DATA FROM THE SCRCHPAD FILE TO PLOT CONTOURS. IT ALSO CALCULATES AREA OF CONTOUR LEVELS AND CALLS THE APPROPRIATE SUBROUTINES TO CALCULATE THE CONTOUR LEVELS.

MODULE I/O: SCRPAD(SCRPSF)

INPUTS:
SCRPSF - LOGICAL FLAG : FLAG TRUE IF CONTOURS ARE TO BE IN PSF E

OUTPUTS: NONE.

FILE I/O:

SCRCHFL - FILE CONTAINING THE SORTED SCRCHPAD RAY DATA.

VARIABLE DICTIONARY:

ACTYPE : CHAR(8) : AIRCRAFT TYPE.
AR : REAL : AREA OF THE CONTOUR LEVEL.
ATIME : REAL(1000) : TIME ASSOCIATED WITH A/C X,Y,Z COORDINATE.
AX : REAL(1000) : X COORDINATE ASSOCIATED WITH THE A/C.
AY : REAL(1000) : Y COORDINATE ASSOCIATED WITH THE A/C.
AZ : REAL(1000) : Z COORDINATE ASSOCIATED WITH THE A/C.
CBLEV : REAL : CARPET BOOM LEVEL USING CARLSON'S METHOD.
CONVAL : REAL(10) : 10 CONTOUR VALUES TO ATTEMPT TO MAKE 3 CO
EMACHN : REAL : ENDING MACH NUMBER.
IPATH : INT(1000) : THE PATH TO CONNECT THE CONTOUR PTS.
IPTS : INT(1000) : NUMBER OF PTS FROM EACH TIME HACK ACCUR
JPTR : INTEGER : NUMBER OF PTS IN IPATH.
MAXOP : REAL : MAXIMUM OVERPRESSURE.
**MOATE**

**CHAR()**

**MISSION DATE.**

**MNAME**

**CHAR(16)**

**MISSION NAME.**

**MSITE**

**CHAR(10)**

**MISSION SITE.**

**NPPH**

**INT(1000)**

**UPPER BOUND IN ARRAY'S FOR EACH TIME HACK.**

**NPTS**

**INTEGER**

**NUMBER OF PTS READ IN FROM THE FILE.**

**NTH**

**INTEGER**

**NUMBER OF TIME HACKS.**

**PRESS**

**REAL(1000)**

**PRESS ASSOCIATED WITH RX, RY.**

**RX**

**REAL(1000)**

**X COORDINATE OF WHERE THE RAY TERMINATES**

**RY**

**REAL(1000)**

**Y COORDINATE OF WHERE THE RAY TERMINATES**

**SMACHN**

**REAL**

**STARTING MACH NUMBER.**

**TAILN**

**CHAR(8)**

**TAIL NUMBER OF THE A/C.**

**XCOORD**

**REAL**

**X COORDINATE OF THE MAXIMUM OVERPRESSURE.**

**XINT**

**REAL(1000)**

**X COORDINATE OF THE INTERPOLATED CONTOUR**

**YCOORD**

**REAL**

**Y COORDINATE OF THE MAXIMUM OVERPRESSURE.**

**YINT**

**REAL(1000)**

**Y COORDINATE OF THE INTERPOLATED CONTOUR**

**XPLT**

**REAL(1000)**

**X COORDINATE OF THE CONNECTED PTS TO PLO**

**YPLT**

**REAL(1000)**

**Y COORDINATE OF THE CONNECTED PTS TO PLO**

**ZCOORD**

**REAL**

**Z COORDINATE OF THE MAXIMUM OVERPRESSURE.**

**CALLED MODULES :**

* * *

**CCONVL** **(PRESS, NPTS, CONVAL, SCRPSF);** **CONVERTS FROM PASCALS TO PS**

**PRESS** **REAL(1000)**

**PRESSURE IN PASCALS RETURNED IN DB OR PS**

**NPTS** **INTEGER**

**NUMBER OF POINTS IN PRESS.**

**CONVAL** **REAL(1000)**

**CONTOUR LEVELS TO LOOK FOR.**

**SCRPSF** **LOGICAL**

**FLAG TRUE IF PRESSURE TO BE IN PSF.**

* * *

**PSETUP** **(AX, AY, AZ, NPTS, XTEMP, YTEMP, XBASE, YBASE, ATIME, FTS;**

**SETS UP THE PLOT PAGE WITH THE FLIGHT TRACK, FLIGHT INFORMATION**

**AND SCALING THE CONTOURS AND FLIGHT TRACK.**

**ATIME** **REAL(1000)**

**TIME ASSOCIATED WITH A/C X,Y,Z COORDINATES**

**AX** **REAL(1000)**

**X COORDINATE ASSOCIATED WITH THE A/C.**

**AY** **REAL(1000)**

**Y COORDINATE ASSOCIATED WITH THE A/C.**

**AZ** **REAL(1000)**

**Z COORDINATE ASSOCIATED WITH THE A/C.**

**NPTS** **INTEGER**

**NUMBER OF PTS READ IN FROM THE FILE.**

**XTEMP** **REAL**

**STARTING X COORDINATE OF THE FLIGHT TRACK**

**YTEMP** **REAL**

**STARTING Y COORDINATE OF THE FLIGHT TRACK**

**XBASE** **REAL**

**PLACE FROM WHICH TO BASE THE FLIGHT TRACK**

**YBASE** **REAL**

**PLACE FROM WHICH TO BASE THE CONTOUR.**

**ATIME** **REAL(1000)**

**TIME ASSOCIATED WITH X,Y,Z OF THE A/C.**

**FTS** **REAL**

**SLOPE OF THE FLIGHT TRACK.**

**SCALE** **REAL**

**SCALE OF THE FLIGHT TRACK AND CONTOUR.**

* * *

**FNDCTN** **(PRESS(LPTR), RX(LPTR), RY(LPTR), NPPH(JPLT),**

**CONVAL(IPLT), IPTR, XINT, YINT, ICNT);**

**FINDS X AND Y COORDINATES ASSOCIATED WITH THE CONTOUR LEVELS**

**EACH TIME HACK.**

**PRESS** **REAL(1000)**

**PRESSURE IN PASCALS RETURNED IN DB OR PS**

**RX** **REAL(1000)**

**X COORDINATE OF WHERE THE RAY TERMINATES**

**RY** **REAL(1000)**

**Y COORDINATE OF WHERE THE RAY TERMINATES**

* * *
**NPPH** - INTEGER : NUMBER OF PTS IN THIS CURRENT TIME HACK.

**CONVAL** - REAL : CURRENT CONTOUR VALUE BEING SEARCHED FOR

**IPTP** - INTEGER : NUMBER OF PTS FOUND FOR THIS CONTOUR LEVEL

**XINT** - REAL(1000) : X COORDINATE OF INTERPOLATED PTS FOR CON

**YINT** - REAL(1000) : Y COORDINATE OF INTERPOLATED PTS FOR CON

**ICNT** - INTEGER : NUMBER OF PTS FOUND FOR CURRENT TIME HACK

**COMPTS** (IPTS, NTH, IPATH, JPTR) : CONNECTS THE PTS THAT ARE RETURN

* BY FINDNT TO CREATE A CONTOUR.

**IPTS** - INT(1000) : NUMBER OF PTS FROM EACH TIME HACK ACCUMULATED

**NTH** - INTEGER : NUMBER OF TIME HACKS.

**IPATH** - INT(1000) : THE PATH TO CONNECT THE CONTOUR PTS.

**JPTR** - INTEGER : NUMBER OF PTS IN IPATH.

**PLOTIT** (XPLT, YPLT, JPTR, XTEMP, YTEMP, XBASE, YBASE, AR, CONVAL, FT$T, SCALE, SCRPSF) : PLOTS EACH OF THE THREE CONTOUR LEVELS.

**XINT** - REAL(1000) : X COORDINATE OF INTERPOLATED PTS FOR CON

**YINT** - REAL(1000) : Y COORDINATE OF INTERPOLATED PTS FOR CON

**JPTR** - INTEGER : NUMBER OF PTS IN IPATH.

**XTEMP** - REAL : STARTING X COORDINATE OF THE FLIGHT TRACK.

**YTEMP** - REAL : STARTING Y COORDINATE OF THE FLIGHT TRACK.

**XBASE** - REAL : PLACE FROM WHICH TO BASE THE CONTOUR.

**YBASE** - REAL : PLACE FROM WHICH TO BASE THE CONTOUR.

**AR** - REAL : AREA OF THE CONTOUR LEVEL.

**CONVAL** - REAL : CURRENT CONTOUR VALUE BEING SEARCHED FOR

**FTS** - REAL : SLOPE OF THE FLIGHT TRACK.

**SCALE** - REAL : SCALE OF THE FLIGHT TRACK AND CONTOUR.

**SCRPSF** - LOGICAL : FLAG TRUE IF CONTOURS ARE TO BE IN PSF E

**CLSPLT();** CREATES A NEW PLOT PAGE.

**CALLING MODULE :**

**------------------------**

**SUBROUTINE SCRPAD(SCRPSF, TEMPAR)**

**DIMENSION**

ATIME(4000), AX(4000), AY(4000), AZ(4000)

RX(4000), RY(4000), PRESS(4000), ACTIME(4000)

NPPH(1500), IPTS(1500), YPLT(1500), ACT(1500)

XINT(1500), YINT(1500), IPATH(1500), XPLT(1500)

CONVAL(10), TACT(1500), TEMPAR(11,1000)

**COMMON /HEADER/ MNAME, MDATE, MSITE, ACTYPE, TAILN**

**COMMON /OPREC/ SMACHN, CBEV, MAXOP, XCOORD, YCOORD, ZCOORD, EMA

**CHARACTER *8 ACTYPE, MDATE, TAILN**
CHARACTER *10 MSITE
CHARACTER *16 MNAME
REAL MAXOP, EMACHN, SMACHN
LOGICAL SCRPSF, SMFLG, NEWPLT

C INITIALIZE COUNTER ETC.
DATA ITPTR, ITH/Z'0'/

C REWIND(32)
10 CONTINUE
I = 1
PH11 = 0.0
PH12 = 0.0
SMFLG = .FALSE.
NEWPLT = .TRUE.

C READ THE MISSION INFORMATION FROM THE HEADER
READ(32,15,END=200) MNAME, MOATE, MSITE, ACTYPE, TAILN, ZCOORD
15 FORMAT (3X,A16, A8,2X, A8,2X,2A8,2X,F9.2)

20 CONTINUE

* READ A RECORD FROM THE SCRCHPAD FILE.
READ(32,3201,END = 200) ATIME(I), AX(I), AY(I), AZ(I),
1 ACTIME(I), RX(I), RY(I), TEMP2, PRESS(I), TEMP3, TEMP4
3201 FORMAT (F8.2, 3F8.0, F8.2, 2F8.0, F8.3, 3F10.4)
IF (I .EQ. 1) THEN
   PH11 = TEMP2
   PH12 = TEMP2
ELSE
   PH11 = PH12
   PH12 = TEMP2
ENDIF

* IF ITS THE FIRST RECORD STORE THE STARTING MACH NUMBER.
IF (I .EQ. 1) SMACHN = TEMP4

* CONVERT FROM METERS TO FEET.
AX(I) = AX(I) / 0.3048
AY(I) = AY(I) / 0.3048
AZ(I) = AZ(I) / 0.3048
RX(I) = RX(I) / 0.3048
RY(I) = RY(I) / 0.3048

* IF ITS THE LAST RECORD STORE THE ENDING MACH NUMBER.
IF ((ATIME(I) .NE. 88.) .AND. (ATIME(I) .NE. 99.)) THEN
   I = I + 1
   IF (I .GT. 4000) THEN
      PRINT *, 'EXCEEDING ARRAY BOUNDS GT 4000'
   ENDIF
   EMACHN = TEMP4
GO TO 20
END IF

* READ THE MAXIMUM OVERPRESSURE RECORD.
READ (32, 3202, END=200) TEMP, TEMP2, TEMP3, TEMP4, TEMP5, 
1 XCOORD, YCOORD, MAXOP, TEMP6, CLEVEL

3202 FORMAT (F8.2,3F8.0,F8.2,2F8.0,3F10.4)

* CONVERT FROM METERS TO FEET.
  XCOORD = XCOORD / 0.3048
  YCOORD = YCOORD / 0.3048
  ZCOORD = ZCOORD / 0.3048

200 CONTINUE
NPTS = 1+1
IF (I .LE. 1) GO TO 400

* CONVERT PRESSURE FROM PASCALS TO DB OR PSF.
CALL CCONVL( PRESS, NPTS, CONVAL, SCRPSF)

* SET UP PLOT, PLOT FLIGHT TRACK AND FLIGHT TRACK INFORMATION.
CALL PSETUP (AX, AY, AZ, NPTS, XTEMP, YTEMP, XBASE, YBASE, ATIME, 
1 FTS, SCALE)

* FIND CONTOUR INFORMATION FOR UP TO 3 CONTOURS.
* FIND THE MIN X,Y COORD OF THE RAYS INCASE IT IS A STRAIGHT ACCELER
* FLIGHT.
CUTMIN = 99999999
DO 250, I=1,NPTS
  IF (CUTMIN .GT. (ABS(RX(I)) + ABS(RY(I)))) THEN
    XHOLD = RX(I)
    YHOLD = RY(I)
    CUTMIN = ABS(RX(I)) + ABS(RY(I))
  ENDIF
250 CONTINUE
CUTMIN = 99999999
DO 255 I = NPTS,1,-1
  IF (CUTMIN .GT. (ABS(RX(I)) + ABS(RY(I)))) THEN
    XHOLD2 = RX(I)
    YHOLD2 = RY(I)
    CUTMIN = ABS(RX(I)) + ABS(RY(I))
  ENDIF
255 CONTINUE
ITH = 0
DO 256, N=1,1000
  NPHM(I) = 0
256 CONTINUE

* FIND THE UPPER BOUND OF EACH TIME HIST.
PRETIM = 0.
DO 300 I = 1, NPTS
IF( PRETIM .EQ. ATIME(I)) THEN  
     NPPH(I) = NPPH(I) + 1
ELSE  
     ITH = ITH + 1  
     NPPH(I) = 1  
     PRETIM = ATIME(I)  
ENDIF
300 CONTINUE

** LOOP UNTIL WE GET THREE GOOD PLOTS AND MAKE TEN ATTEMPTS.  
NMGOOD = 0  
DO 1000 JPLT = 1,10  
    ITH = 1  
    ITPTR = 2  
    LPTR = 1
** TRAVERSE THROUGH EACH TIME HACK.  
DO 900 JPLT = 1, ITH  
    IF(NPPH(JPLT) .LT. 3) THEN  
        LPTR = LPTR + NPPH(JPLT)  
        GO TO 900  
    END IF
** FIND PTS FOR THIS CONTOUR LEVEL.  
CALL FMDCNT(PRESS(LPTR),RX(LPTR),RY(LPTR),NPPH(JPLT),NTH,CONVAL(JPLT),ITPTR,XINT,YINT,ICNT,ACTIME(LPTR),ACT)  
DO 41 JPLT = LPTR, NPPH(JPLT)+LPTR  
   41 CONTINUE  
C LPTR = LPTR + NPPH(JPLT)  
IF( ICNT .LE. 1 .OR. ICNT .EQ. 3) GO TO 900  
C ITPS(NTH) = ICNT
 IF (((NMGOOD .EQ. 0) .AND. (NTH .EQ. 2))) THEN  
    IMHAC = JPLT  
    XT1 = XINT(3)  
    XT2 = XINT(ICNT+2)  
    YT1 = YINT(3)  
    YT2 = YINT(ICNT+2)
 C XT = (XINT(3) + XINT(ICNT+2)) /2  
 C YT = (YINT(3) + YINT(ICNT+2))/2  
 C IF ((XT .LT. XINT(3)) .OR. (XT .GT. XINT(ICNT+2)))  
 C 1 XT = (XINT(3) + XINT(ICNT+2))/2 + XINT(3)  
 C IF ((YT .LT. YINT(3)) .OR. (YT .GT. YINT(ICNT+2)))  
 C 1 YT = (YINT(3) + YINT(ICNT+2))/2 + YINT(3)
 ELSE IF (JPLT .EQ. IMHAC) THEN  
    ITEMP = ITPTR * (ICNT - 1)  
    XT1 = XINT(ITEMP)  
    XT2 = XINT(ITPTR)
YT1 = YINT(ITEMP)
YT2 = YINT(ITPTR)

C
XT = (XINT(ITEMP) + XINT(ITPTR)) /2
C
YT = (YINT(ITEMP) + YINT(ITPTR))/2
C
IF ((XT .LT. XINT(ITEMP)) .OR. (XT .GT. XINT(ITPTR)))
C 1
XT = (XINT(ITEMP) + XINT(ITPTR))/2 + XINT(ITEMP)
C
IF ((YT .LT. YINT(ITEMP)) .OR. (YT .GT. YINT(ITPTR)))
C 1
YT = (YINT(ITEMP) + YINT(ITPTR))/2 + YINT(ITEMP)
C
ENDIF
900 CONTINUE
IF (((ITPTR .LT. 6) .OR. (NTH .LE. 2)) THEN
GO TO 1000
ELSE
NMG00 = NMG00 + 1
ENDIF

IF ((SMFLG) .AND. (IPTS(NTH) .NE. 4))
1 GOTO 1100
IF((IPTS(NTH) .EQ. 4) .AND.
1 (NTH .GT. 2)) THEN
SMFLG = .TRUE.
IPTS(1) = 2
XINT(1) = XHOLD
YINT(1) = YHOLD
XINT(2) = XHOLD2
YINT(2) = YHOLD2
ITPTR = ITPTR + 3
NTH = NTH + 1
IPTS(NTH) = 3
IF (FTS .EQ. 999999) THEN
XINT(ITPTR-2) = X1
XINT(ITPTR) = XT2
IF (XBASE .EQ. 0.5) THEN
YINT(ITPTR-2) = YT1 + (NMG00*(1./20.*SCALE))
YINT(ITPTR) = YT2 - (NMG00*(1./20.*SCALE))
XINT(ITPTR-1) = (XHOLD+XHOLD2)/2. + (NMG00
1 * (1./20. * SCALE))
YINT(ITPTR-1) = (YHOLD+YHOLD2)/2.
ELSE
YINT(ITPTR-2) = YT1 - (NMG00*(1./20.*SCALE))
YINT(ITPTR) = YT2 + (NMG00*(1./20.*SCALE))
XINT(ITPTR-1) = (XHOLD+XHOLD2)/2. - (NMG00
1 * (1./20. * SCALE))
YINT(ITPTR-1) = (YHOLD+YHOLD2)/2.
ENDIF
ELSE
IF (FTS .EQ. 0.0) THEN
YINT(ITPTR-2) = YT1
YINT(ITPTR) = YT2
IF (XBASE .EQ. 0.5) THEN
XINT(ITPTR-2) = X1 - (NMG00*(1./20.*SCALE))
\[
\begin{align*}
XINT(ITPTR) &= XT2 + (NMGOOD*(1./20.*SCALE)) \\
XINT(ITPTR-1) &= (XHOLD+XHOLD2)/2. \\
YINT(ITPTR-1) &= (YHOLD+YHOLD2)/2. - (NMGOOD \\
&\quad * (1./20. * SCALE)) \\
\end{align*}
\]

ELSE
\[
\begin{align*}
XINT(ITPTR-2) &= XT1 + (NMGOOD*(1./20.*SCALE)) \\
XINT(ITPTR) &= XT2 - (NMGOOD*(1./20.*SCALE)) \\
YINT(ITPTR-1) &= (YHOLD+YHOLD2)/2. - (NMGOOD \\
&\quad * (1./20. * SCALE)) \\
\end{align*}
\]

ENDIF

ELSE
\[
\begin{align*}
XINT(ITPTR-2) &= XT1 \\
XINT(ITPTR) &= XT2 \\
\end{align*}
\]

IF (XBASE .EQ. 0.5) THEN
\[
\begin{align*}
YINT(ITPTR-2) &= YT1 + (NMGOOD*(1./20.*SCALE)) \\
YINT(ITPTR) &= YT2 - (NMGOOD*(1./20.*SCALE)) \\
\end{align*}
\]

ELSE
\[
\begin{align*}
YINT(ITPTR-2) &= YT1 - (NMGOOD*(1./20.*SCALE)) \\
YINT(ITPTR) &= YT2 + (NMGOOD*(1./20.*SCALE)) \\
\end{align*}
\]

ENDIF

FTSY = ABS((AY(NPTS)-AY(1))/(AX(NPTS)-AX(1)))

IF (FTSY .LT. 1.0) THEN
\[
\begin{align*}
\text{IF (XBASE .EQ. 0.5) THEN} \\
\quad XINT(ITPTR-1) &= (XHOLD+XHOLD2)/2. + (NMGOOD \\
&\quad * (1./20. * SCALE)) \\
\end{align*}
\]

ELSE
\[
\begin{align*}
XINT(ITPTR-1) &= (XHOLD+XHOLD2)/2. - (NMGOOD \\
&\quad * (1./20. * SCALE)) \\
\end{align*}
\]

ENDIF

IF (((YHOLD+YHOLD2)/2.) .LT. 0.0) THEN
\[
\begin{align*}
\text{IF (((YHOLD+YHOLD2)/2.) .LT. 0.0) THEN} \\
\quad YINT(ITPTR-1) &= (YHOLD+YHOLD2)/2. - \\
&\quad (NMGOOD * (1./20. * SCALE))**FTSY \\
\end{align*}
\]

ELSE
\[
\begin{align*}
YINT(ITPTR-1) &= (YHOLD+YHOLD2)/2. + \\
&\quad (NMGOOD * (1./20. * SCALE))**FTSY \\
\end{align*}
\]

ENDIF

ELSE
\[
\begin{align*}
\text{IF (XBASE .EQ. 0.5) THEN} \\
\quad XINT(ITPTR-1) &= (XHOLD+XHOLD2)/2. + (NMGOOD \\
&\quad * (1./20. * SCALE))**FTS \\
\end{align*}
\]

ELSE
\[
\begin{align*}
XINT(ITPTR-1) &= (XHOLD+XHOLD2)/2. - (NMGOOD \\
&\quad * (1./20. * SCALE))**FTS \\
\end{align*}
\]

ENDIF

IF (((YHOLD+YHOLD2)/2.) .LT. 0.0) THEN
\[
\begin{align*}
\text{IF (((YHOLD+YHOLD2)/2.) .LT. 0.0) THEN} \\
\quad YINT(ITPTR-1) &= (YHOLD+YHOLD2)/2. - \\
&\quad (NMGOOD * (1./20. * SCALE)) \\
\end{align*}
\]

ELSE
\[
\begin{align*}
YINT(ITPTR-1) &= (YHOLD+YHOLD2)/2. + (NMGOOD \\
&\quad * (1./20. * SCALE)) \\
\end{align*}
\]

ENDIF

ENDIF

ENDIF

ENDIF
ENDIF
ELSE
DO 910 I = 2,NTH
   IPTS(I-1) = IPTS(I)
910 CONTINUE
NTH = NTH + 1
DO 920 I = 3,ITPTR
   XINT(I-2) = XINT(I)
   YINT(I-2) = YINT(I)
   ACT(I-2) = ACT(I)
920 CONTINUE
ITPTR = ITPTR + 2
ENDIF

* CONNECT UP THE POINTS FOR THIS CONTOUR
CALL CONPTS(IPTS,NTH,IPATH,JPTR)

YMIN = 999999.
DO 950 ICON = 1, JPTR
   XPLT(ICON) = XINT(IPATH(ICON))
   YPLT(ICON) = YINT(IPATH(ICON))
   TACT(ICON) = ACT(IPATH(ICON))
   IF (YPLT(ICON) .LT. YMIN) YMIN = YPLT(ICON)
950 CONTINUE

* PLOT THIS CONTOUR
* SORT THE PTS. SO THEY ARE IN ORDER ACCORDING TO TERM TIME.
IF (.NOT. SNFLG) THEN
   ITMP = (JPTR-1)/2
   DO 600 I = 1, ITMP
      TEMPAR(1,I) = XPLT(I)
      TEMPAR(2,I) = YPLT(I)
      TEMPAR(3,I) = TACT(I)
   600 CONTINUE
   IF (ITMP .GT. 1) CALL SORTBY(ITMP,TEMPAR,3)
   DO 610 I = 1, ITMP
      XPLT(I) = TEMPAR(1,(ITMP+I-1))
      YPLT(I) = TEMPAR(2,(ITMP+I-1))
   610 CONTINUE
   DO 615 I = 1, ITMP
      TEMPAR(1,I) = XPLT(I+ITMP)
      TEMPAR(2,I) = YPLT(I+ITMP)
      TEMPAR(3,I) = TACT(I+ITMP)
   615 CONTINUE
   IF (ITMP .GT. 1) CALL SORTBY(ITMP,TEMPAR,3)
   DO 620 I = 1, ITMP
      XPLT(I+ITMP) = TEMPAR(1,I)
      YPLT(I+ITMP) = TEMPAR(2,I)
   620 CONTINUE
XPLT(JPTR) = XPLT(1)
YPLT(JPTR) = YPLT(1)
ELSE

ITMP = (JPTR-6)/2
DO 630 I = 1,ITMP
   TEMPAR(1,I) = XPLT(I)
   TEMPAR(2,I) = YPLT(I)
   TEMPAR(3,I) = TACT(I)
630   CONTINUE
   IF (ITMP .GT. 1) CALL SORTRY(ITMP,TEMPAR,3)

DO 640 I = 1,ITMP
   XPLT(I) = TEMPAR(1,(ITMP+1-I))
   YPLT(I) = TEMPAR(2,(ITMP+1-I))
640   CONTINUE
   DO 645 I = 1,ITMP
      TEMPAR(1,I) = XPLT(I+ITMP+2)
      TEMPAR(2,I) = YPLT(I+ITMP+2)
      TEMPAR(3,I) = TACT(I+ITMP+2)
645   CONTINUE
   IF (ITMP .GT. 1) CALL SORTRY(ITMP,TEMPAR,3)

DO 650 I = 1,ITMP
   XPLT(I+ITMP+2) = TEMPAR(1,I)
   YPLT(I+ITMP+2) = TEMPAR(2,I)
650   CONTINUE
ENDIF

* CALCULATE THE AREA OF THE CONTOUR PLOT.
AR = 0.0
DO 975 ICNT = 2,JPTR
   BASE = (XPLT(ICNT-1) - XPLT(ICNT))
   HEIGHT = YPLT(ICNT-1) - YPLT(ICNT)
   TAR = 0.5 * (BASE * HEIGHT)
   YM = MIN(YPLT(ICNT),YPLT(ICNT-1))
   AR = AR + (TAR + (BASE * (YM - YM)))
975   CONTINUE
AR = ABS( AR/27878400.00)

IF (.NOT. SMFLG) JPTR = JPTR - 1

CALL PLOTIT(XPLT, YPLT, JPTR, XTEMP, YTEMP, XBASE, YBASE, AR, + CONVAL(IPLT), FTS, SCALE, SCRPSF, NEWPLT)

IF (NMGOOD .EQ. 3) GO TO 1100
1000 CONTINUE

1100 CONTINUE

* CREATE ANOTHER PLOT PAGE.
   CALL CLSPLT

* IF NOT END OF THE DATA THEN MAKE ANOTHER CONTOUR.
IF (ATIME(NPTS+1) .EQ. 88) THEN
    GO TO 10
END IF

* - CLOSE THE PLOT FILE
400 CONTINUE

RETURN
END
SUBROUTINE CCONVL(PRESS,NPTS,CONVAL,SCRPSF)

LOGICAL SCRPSF
DIMENSION PRESS(NPTS), CONVAL(10), TVAL1(40), TVAL2(40), TVAL(40)

DATA TVAL1/70.,72.5,75.,77.5,80.,82.5,85.,87.5,
     90.,92.5,95.,97.5,100.,102.5,105.,107.5,/
CONVERTING DATA FROM PASCALS TO DB.

CONVERT THE INPUTED PRESSURE TO DB OR PSF.

DO 100 1 = 1,NPTS
  IF (SCRPSF) THEN
    PRESS(I) = PRESS(I) / 47.85
  ELSE
    PRESS(I) = 10. * ALOG10(PRESS(I) * PRESS(M)) + 94.0
  ENDIF
  PMAX = AMAX1(PMAX,PRESS(I))
100 CONTINUE

INITIALIZE TVAL WITH THE CORRECT CONTOUR LEVELS IE. PSF/DB.

DO 110 I = 1,40
  IF (SCRPSF) THEN
    TVAL(I) = TVAL2(I)
  ELSE
    TVAL(I) = TVAL1(I)
  ENDIF
110 CONTINUE

IDENTIFY THE MAXIMUM CONTOUR, I.E. THE FIRST CONTOUR VALUE LESS THAN OR EQUAL TO THE MAX PRESSURE.

DO 200 I = 1, 40
  IC = 40 - I + 1
  IF(PMAX .GE. TVAL(IC)) GO TO 250
200 CONTINUE
250 CONTINUE

SET THE TEN CONTOUR VALUES

DO 300 I = 1,10
  CONVAL(I) = TVAL( IC - I + 1)
300 CONTINUE
RETURN
END
SUBROUTINE FNDCNT

* MODULE NAME : FNDCNT
* MODULE TYPE : SUBROUTINE
* 
* PROGRAMMER : HARRY SEIDMAN
* DATE : DECEMBER 1986
* REVISIONS : APRIL 1987, UPDATE TO SEARCH FROM FAR END OF ARRAY FOR CONTOUR VALUES IF TWO ARE ALREADY FOUND.
* 
* DESCRIPTION :
* 
* THE PURPOSE OF THIS ROUTINE IS TO TAKE AN ARRAY OF PRESSURES, INTERPOLATE THE DATA TO FIND THE NUMBER OF TIMES THAT THE DESIRED CONTOUR VALUE EXISTS FOR EACH THIS TIME HACK, AND FIND THE THE INTERPOLATED X AN Y LOCATION WHERE THE CONTOUR BELONGS.
* 
* MODULE I/O :
* 
* INPUTS :
* PRESS : REAL(1000) : PRESSURE IN PASCALS RETURNED IN DB OR PS
* RX : REAL(1000) : X COORDINATE OF WHERE THE RAY TERMINATES
* RY : REAL(1000) : Y COORDINATE OF WHERE THE RAY TERMINATES
* NPPH : INTEGER : NUMBER OF PTS IN THIS CURRENT TIME HACK.
* CONVAL : REAL : CURRENT CONTOUR VALUE BEING SEARCHED FOR
* 
* OUTPUTS:
* ITPTR : INTEGER : NUMBER OF PTS FOUND FOR THIS CONTOUR LEVEL
* XINT : REAL(1000) : X COORDINATE OF INTERPOLATED PTS FOR CON
* YINT : REAL(1000) : Y COORDINATE OF INTERPOLATED PTS FOR CON
* ICNT : INTEGER : NUMBER OF PTS FOUND FOR CURRENT TIME HACK
* 
* 
* VARIABLE DICTIONARY :
* 
* CALLED MODULES : NONE
* 
* CALLING MODULE :
* 
* SCRPAD : SUBROUTINE THAT DRIVES THE SCRACHPAD PLOTTING ROUTINE
* 
* SUBROUTINE FNDCNT(PRESS, X, Y, NPTS, CONVAL, ITPTR, XINT, YINT, ICNT, 
* 1 ACTIME, ACT)
DIMENSION PRESS(NPTS), X(NPTS), Y(NPTS), ACTIME(NPTS)
DIMENSION XINT(1500), YINT(1500), ACT(1500)

C C SET UP A COUNTER TO MAKE SURE WE DO NOT FIND MORE THAN
C FOUR OCCURANCES OF A CONTOUR VALUE DURING ONE TIME HACK
C
ICNT= 0
C
C START AT THE BEGINNING OF THE TIME HACK.
DO 100 I = 2, NPTS
  C IF(CONVAL .GT. PRESS(I-1) .AND. CONVAL .LT. PRESS(I)) THEN
    FACTOR = (CONVAL .GT. PRESS(I-1)) / (PRESS(I) .GT. PRESS(I-1))
    ITPTR = ITPTR + 1
    XINT(ITPTR) = X(I-1) + ( X(I) - X(I-1) ) * FACTOR
    YINT(ITPTR) = Y(I-1) + ( Y(I) - Y(I-1) ) * FACTOR
    ACT(ITPTR) = ACTIME(I-1)+(ACTIME(I-1)-ACTIME(I-1))*FACTOR
    ICNT = ICNT+ 1
    IF(ICNT.EQ. 2) THEN
      IHOLO = I
      GO TO 110
    END IF
  ELSEIF((CONVAL .LT. PRESS(I-1)) .AND. (CONVAL .GT. PRESS(I))) THEN
    FACTOR = (CONVAL .LT. PRESS(I-1)) / (PRESS(I) .LT. PRESS(I-1))
    ITPTR = ITPTR + 1
    XINT(ITPTR) = X(I-1) + ( X(I) - X(I-1) ) * FACTOR
    YINT(ITPTR) = Y(I-1) + ( Y(I) - Y(I-1) ) * FACTOR
    ACT(ITPTR) = ACTIME(I-1)+(ACTIME(I-1)-ACTIME(I-1))*FACTOR
    ICNT = ICNT+ 1
    IF(ICNT.EQ. 2) THEN
      IHOLO = I
      GO TO 110
    END IF
  ENDIF
C 100 CONTINUE
C
RETURN
*
C SEARCH FROM THE FAR END OF THE TIME HACK TO SEE IF ANY MORE PTS EX
C 110 CONTINUE
  I = 1
  DO 200 I = (NPTS-1), IHOLO, -1
    IF((CONVAL .GT. PRESS(I+1)) .AND.(CONVAL .LT. PRESS(I))
      1 .AND. (ICNT.EQ. 3)) THEN
      FACTOR = (CONVAL .GT. PRESS(I+1)) / (PRESS(I) .LT. PRESS(I+1))
      ITPTR = ITPTR + 1
    END IF
C
XINT(ITPTR + IT) = X(I+1) + ( X(I) - X(I+1) ) * FACTOR
YINT(ITPTR + IT) = Y(I+1) + ( Y(I) - Y(I+1) ) * FACTOR
ACT(ITPTR+IT)= ACTIME(I+1)+ACTIME(I+1)/FACTOR
ICNT= ICNT+ 1
IF (ICNT .EQ. 3) IT = -1
IF (ICNT.EQ. 4) RETURN
ELSEIF(CONVAL .LT. PRESS(I+1) .AND. CONVAL .GT. PRESS(I)) THEN
   FACTOR = (CONVAL - PRESS(I+1)) / (PRESS(I) - PRESS(I+1))
   ITPTR = ITPTR + 1
   XINT(ITPTR + IT) = X(I+1) + ( X(I) - X(I+1) ) * FACTOR
   YINT(ITPTR + IT) = Y(I+1) + ( Y(I) - Y(I+1) ) * FACTOR
   ACT(ITPTR+IT)= ACTIME(I+1)+ACTIME(I+1)/FACTOR
   ICNT= ICNT+ 1
   IF (ICNT .EQ. 3) IT = -1
   IF (ICNT.EQ. 4) RETURN
ENDIF
200 CONTINUE
RETURN
END
SUBROUTINE PSETUP

MODULE NAME: PSETUP

MODULE TYPE: SUBROUTINE

PROGRAMMER: THOMAS REILLY

DATE: DECEMBER 12, 1986

DESCRIPTION:

THE PURPOSE OF THIS SUBROUTINE IS TO SET UP THE PLOT PAD. THE FLIGHT TRACK IS PLOTTED, FLIGHT INFORMATION IS PLOTTED, AND FLIGHT IDENTIFICATION IS PLOTTED.

MODULE I/O: PSETUP (AX, AY, AZ, NPTS, XTEMP, YTEMP, XBASE, YBASE, ATIME, FTS, SCALE).

INPUTS: INCLUDING COMMON BLOCKS.

- ACTYPE: CHAR(8): TYPE OF AIRCRAFT THAT IS BEING FLOWN.
- AX: REAL(1000): ARRAY OF X COORDINATES OF THE FLIGHT TRACK.
- AY: REAL(1000): ARRAY OF Y COORDINATES OF THE FLIGHT TRACK.
- AZ: REAL(1000): ARRAY OF Z COORDINATES OF THE FLIGHT TRACK.
- CBLEV: REAL: CARPET BOOM LEVEL USING CARLSON'S METHOD.
- EMACHN: REAL: ENDING MACH NUMBER.
- MAXOP: REAL: MAXIMUM OVERPRESSURE VALUE.
- MDATE: CHAR(8): DATE OF THE MISSION.
- MNAME: CHAR(16): MISSION NAME.
- MSITE: CHAR(10): SITE OF THE MISSION.
- NPTS: INTEGER: NUMBER OF POINTS IN THE FLIGHT TRACK ARRAY.
- SMACHN: REAL: STARTING MACH NUMBER.
- TAILN: CHAR(8): TAIL NUMBER OF THE AIRCRAFT.
- XCOORD: REAL: X COORDINATE OF THE MAXIMUM OVERPRESSURE.
- YCOORD: REAL: Y COORDINATE OF THE MAXIMUM OVERPRESSURE.
- ZCOORD: REAL: Z COORDINATE OF THE MAXIMUM OVERPRESSURE.

OUTPUTS: INCLUDING COMMON BLOCKS.

- FTS: REAL: SLOPE OF THE FLIGHT TRACK.
- SCALE: REAL: SCALE OF THE FLIGHT TRACK AND CONTOURS.
- XBASE: REAL: X COORD OF THE BASE TO PLOT THE CONTOUR.
* XTEMP  - REAL  : X COORD OF THE FIRST ELEMENT OF THE FLIGHT
* YBASE  - REAL  : Y COORD OF THE BASE TO PLOT THE CONTOUR F
* YTEMP  - REAL  : Y COORD OF THE FIRST ELEMENT OF THE FLIGHT

* VARIABLE DICTIONARY :
  
  ALT  - REAL  : ALTITUDE OF THE AIRCRAFT.
  HR   - REAL  : USED TO CONVERT INPUTED TIME INTO HOURS.
  NMAX - INTEGER : INDICE OF THE LARGEST Y COORDINATE.
  MIN  - REAL  : USED TO CONVERT INPUTED TIME INTO MINUTES.
  SEC  - REAL  : USED TO CONVERT INPUTED TIME INTO SECONDS.
  T1   - REAL  : DUMMY VARIABLE USED TO PLOT A 0.
  X    - REAL  : USED TO CALCULATE REAL X COORD INTO INCHES.
  Y    - REAL  : USED TO CALCULATE REAL Y COORD INTO INCHES.
  YMAX - REAL  : LARGEST Y COORDINATE VALUE.

* CALLED MODULES :
  
  SORTFT (ATIME, AX, AY, NPTS);
  THIS SUBROUTINE IS USED TO SORT THE FLIGHT TRACK ON TIME.
  
  PLOT (X, Y, PENSTATE); PLOTS A STRAIGHT LINE.
  * X - REAL  : X COORDINATE ON PLOT PAGE IN INCHES.
  * Y - REAL  : Y COORDINATE ON PLOT PAGE IN INCHES.
  * PENSTATE - INTEGER : STATUS OF THE PEN, (UP, DOWN, NEW ORIG
  
  SYMBOL (X, Y, SIZE, STRING, ANGEL, NCHAR); PLOTS A CHARACTER STRING.
  * SIZE - REAL : SIZE OF THE CHARACTER IN INCHES.
  * STRING  - CHAR(*) : CHARACTER STRING TO BE PLOTTED.
  * ANGEL   - REAL : ANGEL STRING IS TO BE PLOTTED AT.
  * NCHAR   - INTEGER : NUMBER OF CHARACTERS TO BE PLOTTED.
  
  NUMBER (X, Y, SIZE, NUM, ANGEL, DPLACE); PLOTS A NUMBER.
  * NUM - REAL : NUMBER TO BE PLOTTED.
  * DPLACE - INTEGER : NUMBER OF DECIMAL PLACES TO BE PLOTTED.

* CALLING MODULE :
  
  SCRPAD ();
  SUBROUTINE THAT DRIVES THE SCRATCH PAD PLOTTING.

SUBROUTINE PSETUP (AX, AY, AZ, NPTS, XTEMP, YTEMP, XBASE, YBASE,
                   ATIME, FTS, SCALE)
DECLARATION OF SUBROUTINE INPUT/OUTPUT VARIABLES.
COMMON /HEADER/ MNAME, MDATE, MSITE, ACTYPE, TAILN
COMMON /OPREC/ SNA_CN, CBLEV, MAXOP, XCOORD, YCOORD, ZCOORD, EMACHN
REAL MAXOP, CBLEV, XCOORD, YCOORD, ZCOORD
REAL XBASE, YBASE, XTEMP, YTEMP, AX(4000)
REAL AY(4000), AZ(4000), ATIME(4000)
CHARACTER*8 ACTYPE, MOATE, TAILN
CHARACTER*10 MSITE, LAT, LONG
CHARACTER*16 MNAME

DECLARATION OF SUBROUTINE DEPENDANT VARIABLES.
REAL HR, MIN, SEC, ALT, T1, X, Y, SCALE

DETERMINE WHETHER THE FLIGHT TRACK'S SLOPE IS (\(-\)) OR (\(\ast\))
CALL PLOTS
CALL PLOT(8.5,0.,3)
XBASE = 0.0
YBASE = 0.0
IF (AX(1).EQ. AX(NPTS)) THEN
  FTS = 0.0
ELSE
  IF (AY(1).EQ. AY(NPTS)) THEN
    FTS = 999999.
  ELSE
    FTS = ABS(((AX(NPTS) - AX(1)) / (AY(NPTS) - AY(1))))
  ENDIF
ENDIF
CY = AY(NPTS) - AY(1)
CX = AX(NPTS) - AX(1)
CALL NEWPEN(2)
CALL PLOT(2.0,7.63,3)
CALL PLOT(2.2,7.63,2)
CALL NEWPEN(1)
CALL SYMBOL(2.3,7.6,0.09,' - FLIGHT TRACK',0.0,15)
CALL SYMBOL(7.25,7.55,0.20,'+',0.0,1)
CALL SYMBOL(7.35,7.6,0.09,' - MAX OVERPRESSURE',0.0,19)
CALL PLOT(0.25,2.3,3)
CALL PLOT(0.25,7.5,2)
CALL PLOT(10.5,7.5,2)
CALL PLOT(10.5,2.3,2)
CALL PLOT(0.25,2.3,2)

CALCULATE THE BASE FOR THE FLIGHT TRACK.
IF (FTS .GE. 1.0) THEN
  IF (CY .LT. 0) THEN
IF ((FTS .GE. 0.0) .AND. (FTS .LE. 0.5)) YBASE = 0.0
IF ((FTS .GT. 0.5) .AND. (FTS .LE. 1.0)) YBASE = 0.0
ENDIF
IF (CY .GT. 0) THEN
IF ((FTS .GE. 0.0) .AND. (FTS .LE. 0.5)) YBASE = 0.0
IF ((FTS .GT. 0.5) .AND. (FTS .LE. 1.0)) VSASE = YBASE
ENDIF
IF (CY .GT. 0) THEN
XBASE = 0.5
YBASE = YBASE + 5.0
ELSE
XBASE = 10.0
YBASE = YBASE + 5.0
ENDIF

*.
PLOT NORTH UP,
CALL PLOT(5.2,7.90,3)
CALL PLOT(5.3,7.95,2)
CALL PLOT(5.4,7.9,2)
CALL PLOT(5.3,7.75,2)
CALL SYMBOL(5.25,7.55,14,'N',0.0,1)
SCALE = ABS(100*(CX/30000)) + 1 - 30000
*.
PLOT THE FLIGHT TRACK OF THE AIRCRAFT.
CALL NEWPEN(2)
CALL PLOT(XBASE,YBASE,3)
DC 30 I = 2, NPTS
  Y = (AY(1) - AY(I))/SCALE + YBASE
  X = (AX(I) - AX(1))/SCALE + XBASE
  CALL PLOT(X,Y,2)
30 CONTINUE
CALL NEWPEN(1)
X = (((XCOORD - AX(1))/SCALE) + XBASE) * 0.075
Y = (((YCOORD - AY(1))/SCALE) + YBASE) * 0.075
*.
PLOT A (+) AT THE COORDINATES OF THE MAXIMUM OVERPRESSURE.
CALL SYMBOL(X,Y,.20,1+1,0.0,1)
ELSE
PLACE NORTH EAST
CALL SYMBOL(4.95,7.75,14,'W',0.0,1)
CALL PLOT(5.1,7.8,3)
CALL PLOT(5.3,7.8,2)
CALL PLOT(5.25,7.9,2)
CALL PLOT(5.25,7.7,3)
CALL PLOT(5.3,7.7,2)
IF (CX .LT. 0) THEN
IF ((FTS .GT. 0.0) .AND. (FTS .LE. 0.5)) YBASE = 0.0
IF ((FTS .GT. 0.5) .AND. (FTS .LE. 1.0)) YBASE = 0.0
ENDIF
IF (CX .GT. 0) THEN
  IF ((FTS .GT. 0.0) .AND. (FTS .LE. 0.5)) YBASE = 0.0
  IF ((FTS .GT. 0.5) .AND. (FTS .LE. 1.0)) YBASE = 0.0
ENDIF

IF (CY .GT. 0) THEN
  XBASE = 0.5
  YBASE = YBASE + 5.0
ELSE
  XBASE = 10.0
  YBASE = YBASE + 5.0
ENDIF

SCALE = (ABS(INT(CY/30000)) + 1) * 30000

* PLOT THE FLIGHT TRACK OF THE AIRCRAFT.
CALL NEWPEN(2)
CALL PLOT(XBASE,YBASE, 3)
DO 35 I = 2, NPTS
  Y = (AY(I) - AY(1))/ SCALE
  X = (AX(I) - AX(1))/ SCALE
  Y1 = Y
  Y = X * (-1) + YBASE
  X = Y1 + XBASE
  CALL PLOT (X,Y,2)
35 CONTINUE
CALL NEWPEN(1)
X = ((XCOORD - AX(1))/SCALE)
Y = ((YCOORD - AY(1))/SCALE)
Y1 = Y
Y = (X * (-1) + YBASE) - 0.075
X = (Y1 + XBASE) - 0.075

* PLOT A (+) AT THE COORDINATES OF THE MAXIMUM OVERPRESSURE.
CALL SYMBOL(X,Y,.20,'+',0.0,1)

ENDIF

* PLOT THE A/C TYPE AND TAIL NUMBER.
CALL SYMBOL(0.6,2.1,.09,'A/C TYPE ',0.0,15)
CALL SYMBOL(1.9,2.1,.09,'ACTYPE,0.0,8)
CALL SYMBOL(3.7,2.1,.09,'TAIL # ',0.0,13)
CALL SYMBOL(4.8,2.1,.09,'TAILN,0.0,8)

* CALCULATE AND PLOT THE STARTING TIME OF THE FLIGHT TRACK.
HR = INT(ATIME(1)/3600)
MIN = INT((ATIME(1) - (HR * 3600)) / 60)
SEC = ATIME(1) - ((HR * 3600) + (MIN * 60))
CALL SYMBOL(0.6,1.9,.09,'START TIME ',0.0,15)
IF (HR .LT. 10) THEN
  T1 = 0.0
  CALL NUMBER(1.9,1.9,.09,T1,0.0,-1)
  CALL NUMBER(2.0,1.9,.09,HR,0.0,-1)
ELSE
   CALL NUMBER(1.9,1.9,.09,HR,0.0,-1)
END IF
CALL SYMBOL(2.1,1.9,0.09,':',0.0,1)
IF (MIN .LT. 10) THEN
   TI = 0.0
   CALL NUMBER(2.2,1.9,.09,TI,0.0,-1)
   CALL NUMBER(2.3,1.9,.09,MIN,0.0,-1)
ELSE
   CALL NUMBER(2.2,1.9,0.09,MIN,0.0,-1)
END IF
IF (SEC .LT. 10.0) THEN
   CALL SYMBOL(2.4,1.9,0.09,'0',0.0,2)
   CALL NUMBER(2.6,1.9,0.09,SEC,0.0,2)
ELSE
   CALL SYMBOL(2.4,1.9,0.09,'1',0.0,1)
   CALL NUMBER(2.5,1.9,0.09,SEC,0.0,2)
ENDIF

* PLOT THE ALTITUDE OF THE FLIGHT TRACK AT THE STARTING POINT.
   CALL SYMBOL(0.6,1.7,0.09,'START ALT : ',0.0,15)
   ALT = AZ(1)/1000
   CALL NUMBER(1.9,1.7,0.09,ALT,0.0,2)
   CALL SYMBOL(2.5,1.7,0.09,'K FEET',0.0,16)

* PLOT THE MAP SCALE.
   CALL NUMBER(7.7, 2.1, 0.09,SCALE,0.0,-1)
   CALL SYMBOL(8.2,2.1,0.09,'FT',0.0,2)
   CALL PLOT (7.5, 2.0,3)
   CALL PLOT (8.5, 2.0,2)
   CALL PLOT (7.5, 1.95,3)
   CALL PLOT (7.5, 2.05,2)
   CALL PLOT (8.5, 1.95,3)
   CALL PLOT (8.5, 2.05,2)

* CALCULATE AND PLOT THE ENDING TIME OF THE FLIGHT TRACK.
   HR = INT(ATIME(NPTS)/3600)
   MIN = INT((ATIME(NPTS) - (HR * 3600)) / 60)
   SEC = ATIME(NPTS) - ((HR * 3600) + (MIN * 60))
   CALL SYMBOL(3.7,1.9,0.09,'END TIME : ',0.0,13)
IF (HR .LT. 10) THEN
   TI = 0.0
   CALL NUMBER(4.8,1.9,.09,TI,0.0,-1)
   CALL NUMBER(4.9,1.9,.09,HR,0.0,-1)
ELSE
   CALL NUMBER(4.8,1.9,.09,HR,0.0,-1)
ENDIF
CALL SYMBOL(5.0,1.9,0.09,'1',0.0,1)
IF (MIN .LT. 10) THEN
   TI = 0.0
   CALL NUMBER(5.1,1.9,.09,TI,0.0,-1)
   CALL NUMBER(5.2,1.9,.09,MIN,0.0,-1)
ELSE
CALL NUMBER(5.1,1.9,0.09,MIN,0.0,1)
ENDIF
IF (SEC .LT. 10.0) THEN
   CALL SYMBOL(5.3,1.9,0.09,'0',0.0,2)
   CALL NUMBER(5.5,1.9,0.09,SEC,0.0,2)
ELSE
   CALL SYMBOL(5.3,1.9,0.09,',',0.0,1)
   CALL NUMBER(5.4,1.9,0.09,SEC,0.0,2)
ENDIF

*-- PLOT THE ALTITUDE OF THE END OF THE FLIGHT TRACK.
CALL SYMBOL(3.7,1.7,0.09,'END ALT : ',0.0,13)
ALT = AZ(NPTS)/1000
CALL NUMBER(4.8,1.7,0.09,ALT,0.0,2)
CALL SYMBOL(5.4,1.7,0.09,'K FEET ',0.0,7)

*-- PLOT THE MACH NUMBER, CARPET BOOM LEVEL, AND MAXIMUM OVERPRESSURE.
CALL SYMBOL(0.6,1.5,0.09,'START MACH #: ',0.0,15)
CALL NUMBER(1.9,1.5,0.09,SMACHN,0.0,4)
CALL SYMBOL(3.7,1.5,0.09,'END MACH #: ',0.0,13)
CALL NUMBER(4.8,1.5,0.09,EMACHN,0.0,4)
CALL SYMBOL(0.6,1.15,0.09,'CARPET BOOM LEVEL : ',0.0,23)
CALL NUMBER(2.6,1.15,0.09,CBLEV,0.0,2)
CALL SYMBOL(3.1,1.15,0.09,'PSF',0.0,3)
CALL SYMBOL(0.6,0.95,0.09,'MAXIMUM OVERPRESSURE : ',0.0,23)
MAXOP = MAXOP / 47.85
CALL NUMBER(2.6,0.95,0.09,MAXOP,0.0,2)
CALL SYMBOL(3.1,0.95,0.09,'PSF',0.0,3)
CALL SYMBOL(0.6,0.75,0.09,'ENHANCEMENT FACTOR : ',0.0,23)
IF (CBLEV .GT. 0.0) EFACT = MAXOP / CBLEV
CALL NUMBER(2.6,0.75,0.09,EFACT,0.0,2)
CALL SYMBOL(3.7,0.95,0.09,'LAT : ',0.0,7)
CALL SYMBOL(4.3,0.95,0.09,LYCOORD,0.0,10)
CALL SYMBOL(3.7,0.75,0.09,'LONG : ',0.0,7)
CALL SYMBOL(4.3,0.75,0.09,LONG,0.0,10)

*-- PLOT THE COORDINATE VALUES OF THE MAXIMUM OVERPRESSURE.
CALL SYMBOL(6.5,1.55,0.09,'COORDINATES OF MAXIMUM OVERPRESSURE'
   1
   ,0.0,35)
CALL SYMBOL(6.8,1.35,0.09,'X COORDINATE : ',0.0,17)
CALL NUMBER(8.3,1.35,0.09,(XCOORD/1000),0.0,2)
CALL SYMBOL(9.0,1.35,0.09,'K FEET ',0.0,7)
CALL SYMBOL(6.8,1.15,0.09,'Y COORDINATE : ',0.0,17)
CALL NUMBER(8.3,1.15,0.09,(YCOORD/1000),0.0,2)
CALL SYMBOL(9.0,1.15,0.09,'K FEET ',0.0,7)
CALL SYMBOL(6.8,0.95,0.09,'ALTITUDE : ',0.0,17)
CALL NUMBER(8.3,0.95,0.09,(ZCOORD/1000),0.0,2)
CALL SYMBOL(9.0,0.95,0.09,'K FEET ',0.0,7)

*-- PLOT THE FLIGHT SEGMENT IDENTIFICATION.
CALL SYMBOL(6.5,0.6,.09,'FLIGHT SEGMENT IDENTIFICATION',0.0,29)
CALL SYMBOL(6.8,0.4,.09,'MISSION NAME ',0.0,17)
CALL SYMBOL(8.3,0.4,.09,'MNAME',0.0,16)
CALL SYMBOL(6.8,0.2,.09,'MISSION DATE ',0.0,17)
CALL SYMBOL(8.3,0.2,.09,'MDATE',0.0,8)
CALL SYMBOL(6.8,0.0,.09,'MISSION SITE ',0.0,17)
CALL SYMBOL(8.3,0.0,.09,'MSITE',0.0,10)
XTEMP = AX(1)
YTEMP = AY(1)
RETURN
END
END OF SUBROUTINE PSETUP.
SUBROUTINE PLOTIT

MODULE NAME: PLOTIT
MODULE TYPE: SUBROUTINE

PROGRAMMER: THOMAS REILLY
DATE: JANUARY 5, 1986
REVISIONS:

DESCRIPTION:

This subroutine is designed to plot three contour levels for the scratch pad plots. It also plots the area of each of the contour levels.

MODULE I/O: PLOTIT (XPLT, YPLT, JPTR, XTEMP, YTEMP, XBASE, YBASE, CONVAL, SCALE, FTS, SCRPSF)

INPUTS:
- AREA (AR) - REAL: AREA OF THE CONTOUR LEVEL.
- CONVAL (REAL): CONTOUR LEVEL VALUE.
- JPTR (INTEGER): NUMBER OF POINTS IN THE ARRAYS YPLT, XPLT.
- FTS (REAL): SLOPE OF THE FLIGHT TRACK.
- SCALE (REAL): SCALE OF THE FLIGHT TRACK AND CONTOURS.
- SCRPSF (LOGICAL): TRUE IF THE CONTOURS ARE IN PSF.
- XBASE (REAL): X COORDINATE BASE FOR THE CONTOUR LEVELS.
- XPLT (REAL(1000)): X COORD OF THE CONTOUR POINTS TO BE PLOT.
- XTEMP (REAL): X COORD USED TO CALCULATE DISTANCE FROM XBASE.
- YBASE (REAL): Y COORDINATES BASE FOR THE CONTOUR LEVEL.
- YPLT (REAL(1000)): ARRAY OF Y COORDINATES OF THE CONTOUR POINTS TO BE PLOTTED.
- YTEMP (REAL): Y COORDINATE USED TO CALCULATE DISTANCE FROM YBASE.

OUTPUTS:
- NONE.

VARIABLE DICTIONARY:
- CCNT (INTEGER): COUNTER FOR PRESENT CONTOUR LEVEL.
- X (REAL): X COORDINATE IN INCHES.
- Y (REAL): Y COORDINATE IN INCHES.

CALLED MODULES:
*.* PLOT (X, Y, PENSTATE) plots a straight line.

*.* X - REAL : X coordinate on plot page in inches.

*.* Y - REAL : Y coordinate on plot page in inches.

*.* PENSTATE - INTEGER : status of the pen, (up, down, new orig

*.* SYMBOL (X, Y, SIZE, STRING, ANGEL, NCHAR) plots a character string.

*.* SIZE - REAL : size of the character in inches.

*.* STRING - CHAR*: character string to be plotted.

*.* ANGEL - REAL : angle string is to be plotted at.

*.* NCHAR - INTEGER : number of characters to be plotted.

*.* NUMBER (X, Y, SIZE, NUM, ANGEL, DPLACE) plots a number.

*.* NUM - REAL : number to be plotted.

*.* DPLACE - INTEGER : number of decimal places to be plotted.

*.*

*.* CALLING MODULE:

*.*

*.* SCRPAD()

*.* subroutine which acts as a driver for the scratch pad plots

SUBROUTINE PLOTIT(XPLT, YPLT, JPTR, XTEMP, YTEMP, XBASE, YBASE, +
    AR, CONVAL, FTS, SCALE, SCRPSF, NEWPLT)

*.* declaration of subroutine dependent variables.

REAL XPLT(1500), YPLT(1500), AR, CONVAL, XBASE, YBASE, XTEMP

REAL YTEMP

INTEGER JPTR

LOGICAL SCRPSF, NEWPLT

*.* declaration of subroutine dependent variables.

REAL X, Y, SCALE

INTEGER CCNT

SAVE CCNT

*.* IF THE SLOPE IS POSITIVE THEN PLOT THIS CONTOUR LEVEL.

IF (NEWPLT) THEN
    CCNT = 0
    NEWPLT = .FALSE.
ENDIF

IF (FTS .GE. 1.0) THEN
    X = ((XPLT(1) - XTEMP)/SCALE) + XBASE
    Y = ((YPLT(1) - YTEMP)/SCALE) + YBASE
    CALL PLOT(X, Y, 3)
    DO 10, I = 1, JPTR
        X = ((XPLT(I) - XTEMP)/SCALE) + XBASE
        Y = ((YPLT(I) - YTEMP)/SCALE) + YBASE
        CALL PLOT(X, Y, 2)
CONTINUE

ELSE

\[ X = \frac{(XPLT(1) - XTEMP)}{SCALE} \]
\[ Y = \frac{(YPLT(1) - YTEMP)}{SCALE} \]

\[ Y1 = Y \]
\[ Y = X * (-1) + YBASE \]
\[ X = Y1 + XBASE \]
CALL PLOT(X,Y,3)

DO 20, I = 1, JPTR

\[ X = \frac{(XPLT(I) - XTEMP)}{SCALE} \]
\[ Y = \frac{(YPLT(I) - YTEMP)}{SCALE} \]

\[ Y1 = Y \]
\[ Y = X * (-1) + YBASE \]
\[ X = Y1 + XBASE \]
CALL PLOT(X,Y,2)

20 CONTINUE
ENDIF

*.- CALCULATE WHICH CONTOUR LEVEL IT IS.

IF (CCNT .EQ. 3) THEN

CCNT = 1
ELSE

CCNT = CCNT + 1
END IF

Y = 0.6 - (CCNT * .2)

**.- PLOT THE AREA OF THE CONTOUR THAT WAS JUST PLOTED.

IF (CCNT .EQ. 1) THEN

IF (SCRPSF) THEN

CALL SYMBOL(0.6,Y,.09,'AREA OF PSF CONTOUR LEVEL : ',
+ 0.0,36)

CALL SYMBOL(4.6,Y,.09,'SQ. MILES',0.0,9)
ELSE

CALL SYMBOL(0.6,Y,.09,'AREA OF DB CONTOUR LEVEL : ',
+ 0.0,36)

CALL SYMBOL(4.6,Y,.09,'SQ. MILES',0.0,9)
ENDIF
ENDIF

IF (CCNT .EQ. 2) THEN

IF (SCRPSF) THEN

CALL SYMBOL(0.6,Y,.09,'AREA OF PSF CONTOUR LEVEL : ',
+ 0.0,36)

CALL SYMBOL(4.6,Y,.09,'SQ. MILES',0.0,9)
ELSE

CALL SYMBOL(0.6,Y,.09,'AREA OF DB CONTOUR LEVEL : ',
+ 0.0,36)

CALL SYMBOL(4.6,Y,.09,'SQ. MILES',0.0,9)
ENDIF
ENDIF

IF (CCNT .EQ. 3) THEN

IF (SCRPSF) THEN

CALL SYMBOL(0.6,Y,.09,'AREA OF PSF CONTOUR LEVEL : ',
+ 0.0,36)
CALL SYMBOL(4.6,Y,.09,'SQ. MILES',0.0,9)
ELSE
CALL SYMBOL(0.6,Y,.09,'AREA OF CONTOUR LEVEL : ',0.0,36)
CALL SYMBOL(4.6,Y,.09,'SQ. MILES',0.0,9)
ENDIF
END IF
CALL NUMBER (1.35,Y,.09,CONVAL,0.0,2)
CALL NUMBER (3.8,Y,.09,AR,0.0,4)
RETURN
END
* END OF SUBROUTINE PLOTIT.
SUBROUTINE CLSPLT

*.
* MODULE NAME : CLSPLT
* MODULE TYPE : SUBROUTINE
*.
* PROGRAMMER : THOMAS REILLY
* DATE : JANUARY 5, 1986
* REVISIONS :
*.
* DESCRIPTION :
*.
* THIS SUBROUTINE IS DESIGNED TO MOVE TO THE NEXT PLOT PAGE.
*.
* CALLED MODULES :
* .
*.
* PLOT (X, Y, PENSTATE); SKIPS TO THE NEXT PLOT PAGE.
*.
* X - REAL : X COORDINATE ON PLOT PAGE IN INCHES.
* Y - REAL : Y COORDINATE ON PLOT PAGE IN INCHES.
* PENSTATE - INTEGER : SKIPS TO THE NEXT PLOT PAGE.
*.
* CALLING MODULE :
* .
* .
* SCRPAD() - SUBROUTINE TO DRIVE THE SCRATCH PAD PLOTHING.
*.

SUBROUTINE CLSPLT
CALL PLOT(0.,0.,999)
RETURN
END
*.
END OF SUBROUTINE CLSPLT.
**MODULE NAME**: CONPTS
**MODULE TYPE**: SUBROUTINE

**PROGRAMMER**: HARRY SEIDMAN
**DATE**: DECEMBER 1986
**REVISIONS**: APRIL 1987, CHANGE FOR CONSISTENCY WITH NEW TEST DATA.

**DESCRIPTION**: THE PURPOSE OF THIS ROUTINE IS TO CONNECT ALL THE POINTS AROUND A CONTOUR. IT IS ASSUMED THAT AT ANY TIME UP TO FOUR POINTS ON THE CONTOUR MAY EXIST.

**MODULE I/O**:

**INPUTS**:
- **IPTS** :: INT(1000) : NUMBER OF PTS FROM EACH TIME HACK ACCUR
- **NTH** :: INTEGER : NUMBER OF TIME HACKS.

**OUTPUTS**:
- **IPATH** :: INT(1000) : THE PATH TO CONNECT THE CONTOUR PTS.
- **JPTR** :: INTEGER : NUMBER OF PTS IN IPATH.

**VARIABLE DICTIONARY**:

**CALLED MODULES**: NONE.

**CALLING MODULE**:
- **SCRPAK** : SUBROUTINE THAT DRIVES THE SCRATCHPAD PLOTTING ROUTINE

SUBROUTINE CONPTS(IPTS,NPTS,IPATH,JPTR)

PARAMETER (MAXPTS = 1500)
DIMENSION IPTS(NPTS)
DIMENSION IPATH(MAXPTS)
DIMENSION ISUMPTS(MAXPTS)
IF( NPTS .GT. MAXPTS) THEN
  PRINT *, 'MUST INCREASE THE DIMENSION OF ISUMPTS IN',
  1 'SUBROUTINE COMPTS'
  STOP
ENDIF

* - IF ONLY ONE TIME HACK JUST CONNECT THE POINTS
IF(NPTS .EQ. 1) THEN
  DO 20 I = 1, IPTS(I)
    IPATH(I) = I
  20 CONTINUE
  JPTR = IPTS(I)
  RETURN
ENDIF

* - DETERMINE THE TOTAL NUMBER OF POINTS THAT EXIST BEFORE EACH TIME H
ISUMPTS(1) = 0
 DO 100 I = 2, NPTS
   ISUMPTS(I) = ISUMPTS(I-1) + IPTS(I-1)
  100 CONTINUE
  GOTO 400

120 CONTINUE

* - CONNECT UP THE LEFT SIDE OF THE CONTOUR
DO 200 I = 1, NPTS
  IPATH(JPTR) = ISUMPTS(I) + IPTS(I)
  JPTR = JPTR + 1
  200 CONTINUE

* - CONNECT THE BOTTOM OF THE CONTOUR
IF ( IPTS(NPTS) .EQ. 1 ) RETURN
 DO 250 I = 2, IPTS(NPTS)
   IPATH(JPTR) = ISUMPTS(NPTS) + IPTS(NPTS) - (I-1)
   JPTR = JPTR + 1
  250 CONTINUE
  JPTR = JPTR - 1
  RETURN

* - CONNECT UP THE RIGHT SIDE OF THE CONTOUR.
400 CONTINUE
  IPATH(JPTR) = ISUMPTS(NPTS) + 1
  JPTR = JPTR + 1
  DO 500 I = NPTS+1,1,-1
    IPATH(JPTR) = ISUMPTS(I)+1
    JPTR = JPTR + 1
  500 CONTINUE

* - CONNECT UPU THE TOP OF THE CONTOUR.
 DO 550, I = 2,(ISUMPTS(2)-1)
IPATH(JPTR) = 1
JPTR = JPTR + 1

550 CONTINUE
GOTO 120
END
SUBROUTINE SCRHF

**********W**********W**********W

MODULE NAME: SCRHF

MODULE TYPE: SUBROUTINE

PROGRAMMER: THOMAS REILLY

DATE: DECEMBER 10, 1986

DESCRIPTION:

THIS SUBROUTINE HAS BEEN DESIGNED TO CREATE A TEMPORARY FILE WHICH CONTAINS THE CAUSTIC RAYS WHICH HAVE NOT HAD A SCRATCH PAD CREATED FOR THEM. THIS SUBROUTINE ALSO CALLS THE SUBROUTINE SORTRY, WHICH SORTS THE RAYS ON THE PHI ANGLE.

MODULE I/O:

SCRHF (SREC, NREC, SOPR, NOPR)

INPUTS:

NOPR - INTEGER: NUMBER OF OVERPRESSURE RECORDS.
NREC - INTEGER: NUMBER OF RAYS IN THE FLIGHT SEGMENT.
SOPR - INTEGER: STARTING RECORD OF THE OVERPRESSURE RECORD.
SREC - INTEGER: STARTING RECORD OF THE FLIGHT SEGMENT.

OUTPUTS: THE OUTPUTED FILE FOR THE SCRATCH PAD PLOTS, \scrhf'

FILE/IO DICTIONARY:

INFILE: THIS FILE IS THE RAY DATABASE FILE.
OUTFL: THIS IS A SEQUENTIAL FILE USED TO STORE THE SORTED RAYS FOR USE BY THE SCRATCH PAD SUBROUTINE, "SCRHF"
TEMPFL: THIS FILE IS USED TO SORT THE ARRAYS ON THE PHI ANGLE.

VARIABLE DICTIONARY:

ARRL - INTEGER: NUMBER OF FIELDS IN THE FILES.
BUF - CHAR(90): USED TO TRANSFER AN ENTIRE RECORD FROM ONE TO ANOTHER.
CASTIC - LOGICAL: TRUE IF A CAUSTIC RAY HAS BEEN FOUND.
CTIME - REAL: CONTAINS A RAY TERMINATION TIME, USED TO FLIGHT SEGMENTS WITHIN A FLIGHT TRACK.
ERECS - INTEGER: RECORD NUMBER OF LAST RAY IN FILE.
ETIME - REAL: ENDING TIME OF THE FLIGHT SEGMENT.

FILE - INTEGER: FILE NUMBER OF THE RAY DATABASE FILE.
CALLING MODULES:

SUBROUTINE SCRHF(SREC, NREC, SOPR, NOPR, ACTAIL, SCRALL,

1                   TEMPAR)
FRECFL = .FALSE.
NOPR = (NOPR + SOPR) - 1
TTIME = 0.0
STIME = 0.0

* LOOP UNTIL END OF THE FLIGHT TRACK.
10 CONTINUE
IF (RECNT .LE. EREC) THEN
READ (INFILE, 15, REC=RECNT) RTYPE, (RAYS(I), I=1, ARRL)
15 FORMAT (12, F8.2, 3F8.0, F8.2, 2F8.0, F8.3, 3F10.4)
RECNT = RECNT + 1
IF (RTYPE .EQ. 00) THEN
END IF
IF (CTIME .EQ. 0.0) THEN
CTIME = RAYS(1)
ENDIF
IF (STIME .EQ. 0.0) THEN
STIME = RAYS(1)
ENDIF
IF (TTIME .EQ. 0.0) TTIME = RAYS(1)

* CHECK IF THE CAUSTIC RAY HAS A TIME DIFFERENCE GREATER THAN 4.5
IF ((RAYS(1) - TTIME) .LT. 4.5) THEN

* UPDATE THE DATABASE THAT THE RAY HAS BEEN PLOTTED.
IF (RTYPE .EQ. 21) CASTIC = .TRUE.
IF ((RTYPE .EQ. 12) .AND. (SCROLL)) CASTIC = .TRUE.
IF (RTYPE .EQ. 21) RTYPE = 12
IF (RTYPE .EQ. 01) RTYPE = 01
WRITE(INFILE, 15, REC=(RECNT-1)) RTYPE, (RAYS(I), I=1, ARRL)

* IF IT'S THE FIRST UNPLOTTED CAUSTIC RAY THEN.
IF (.NOT. FRECFL) .AND. (CASTIC) THEN
READ (INFILE, FMT='(A)'), REC=SREC
BUF
WRITE(BUF(48:49), FMT='(A2)') I'
WRITE(BUF(50:57), FMT='(A8)') ACTAIL
WRITE(OUTFL, FMT='(4)') BUF
FRECFL = .TRUE.
ENDIF

* APPEND THE CAUSTIC RAY TO TEMPAR.
TTIME = RAYS(1)
DO 18, I = 1, ARRL
TEMPAR(I, SCHCNT) = RAYS(I)
18 CONTINUE
SCHCNT = SCHCNT + 1

. IF (CTIME .NE. RAYS(1)) THEN
   SCHCNT = SCHCNT - 2
   RECNT = RECNT - 1
   IF (SCHCNT .GT. 1) THEN
CALL SORTRY(SCHCNT, TEMPAR, 8)
ENDIF

APPEND THE SORTED RAYS TO A SEQUENTIAL ACCESS FILE.
DO 20 I = 1, SCHCNT
   WRITE(34,25) (TEMPAR(K,I), K=1,ARRL)
   CONTINUE
20 FORMAT (F8.2, 3F8.0, F8.2, 2F8.0, F8.3, 3F10.4)
SCHCNT = 1
CTIME = 0.0
GO TO 10
ENDIF
CTIME = RAYS(1)
GO TO 10

*.* IF TIME DIFFERENCE IS GREATER THAN 4.5'. ELSE
   FRECFL = .FALSE.
   TTIME = 0.0
   ETIME = CTIME
   CTIME = 0.0
   IF (.NOT. CASTIC) THEN
      SCHCNT = 1
      REWIND(34)
      STIME = 0.0
      GO TO 10
      ENDIF
      CASTIC = .FALSE.
      SORT THE RAYS ON PHI ANGLE.
      RECNT = RECNT - 1
      SCHCNT = SCHCNT - 1
      IF (SCHCNT .GT. 1) THEN
         CALL SORTRY(SCHCNT, TEMPAR, 8)
      ENDIF
   ENDIF

APPEND THE SORTED RAYS TO A SEQUENTIAL ACCESS FILE.
DO 30 I = 1, SCHCNT
   WRITE(34,25) (TEMPAR(K,I), K=1,ARRL)
   CONTINUE
30 SCHCNT = 1
   REWIND(34)
   READ(34,25,END=34) (RAYS(I), I=1,ARRL)
   WRITE(OUTFL,25) (RAYS(I), I = 1,ARRL)
   GOTO 32
32 CONTINUE
   REWIND(34)
   WRITE *'88.0' TO THE SEQUENTIAL ACCESS FILE.
   WRITE (OUTFL,FMT='(A)') '88.0'
   SEEK THE MAX OVERPRESSURE AND WRITE IT TO THE FILE.
   MAXOP = -.999999.
READ(INFILE,FMT='(A)',REC=NPCNT) BUF
READ(BUF(52:61),FMT='(F10.4)') CUROP
READ(BUF(1:8),FMT='(F8.2)') CURTIM
IF ((CURTIM .GE. STIME) .AND. (CURTIM .LE. ETIME)) THEN
  IF (CUROP .GT. MAXOP) THEN
    MOPREC = NPCNT
    MAXOP = CUROP
  ENDIF
  NPCNT = NPCNT + 1
  IF (NPCNT .LE. NOPR) THEN
    GOTO 35
  ENDIF
ENDIF
READ(INFILE,FMT='(A)',REC=MOPREC) BUF
WRITE(OUTFL,FMT='(A)') BUF
STIME = 0.0
GO TO 10
END IF

IF SCRCHCNT GREATER THAN ONE THAN
IF ((SCHCNT .GT. 1) .AND. (CASTIC)) THEN
  ETIME = CTIME
  CTIME = 0.0
  SCHCNT = SCHCNT - 1
  IF (SCHCNT .GT. 1) THEN
    CALL SORTRY(SCHCNT, TEMPAR, 8)
  ENDIF
  APPEND THE SORTED RAYS TO A SEQUENTIAL ACCESS FILE.
  DO 50 I = 1, SCHCNT
      WRITE(34, 25) (TEMPAR(K, I), K = 1, ARLR)
  CONTINUE
  REWIND(34)
  52 READ(34, 25, END=54) (RAYS(I), I = 1, ARLR)
  WRITE(OUTFL, 25) (RAYS(I), I = 1, ARLR)
  GOTO 52
  CONTINUE
  REWIND(34)
  SORT LAST SCRCH PAD AND WRITE '88.0' TO OUTFILE.
  WRITE (OUTFL, FMT='(A)') '88.0'
  SEEK THE MAX OVERPRESSURE AND WRITE IT TO THE FILE.
  MAXOP = .9999999
  READ(INFILE,FMT='(A)',REC=NPCNT) BUF
  READ(BUF(52:61),FMT='(F10.4)') CUROP
  READ(BUF(1:8),FMT='(F8.2)') CURTIM
  IF ((CURTIM .GE. STIME) .AND. (CURTIM .LE. ETIME)) THEN
    IF (CUROP .GT. MAXOP) THEN
      MOPREC = NPCNT
      MAXOP = CUROP
    ENDIF
  ENDIF
END IF
ENDIF
NPCNT = NPCNT + 1
IF (NPCNT .LE. NOPR) THEN
    GOTO 55
ENDIF
ENDIF
READ(INFILE,FMT='(A)',REC=MOPREC) BUF
WRITE(OUTFL,FMT='(A)') BUF
END IF

60 CONTINUE

*   END OF THE SUBROUTINE SCRCNFL.
RETURN
END

*   END OF SUBROUTINE SCRCNFL.
SUBROUTINE SORTRY(NPTS,TEMPAR,SRTFLD)

INTEGER NPTS, I, J, K, L, ARRL, SRTFLD

PARAMETER (ARRL = 11)

REAL TRAY(ARRL)

REAL TEMPAR(ARRL,1000)

SET UP INITIALIZATION FOR HEAPSORT.

L = NPTS / 2 + 1
IR = NPTS

*· HEAP CREATION PHASE.
30 CONTINUE
   IF (L .GT. 1) THEN
      L = L - 1
   *
   INITIALIZE RRA TO ELEMENT RA(L) IN THE RAY ARRAY.
   DO 32, K = 1, ARL
      TRAY(K) = TEMPAR(K,L)
   32 CONTINUE
   ELSE
   *
   PLACE TOP OF HEAP AT THE END OF THE ARRAY.
   DO 34, K = 1, ARL
      TRAY(K) = TEMPAR(K,IR)
      TEMPAR(K,IR) = TEMPAR(K,1)
   34 CONTINUE
   IR = IR - 1
   *
   PLACE SMALLEST ELEMENT AT THE BEGINING OF THE ARRAY.
   IF (IR .EQ. 1) THEN
      DO 36, K = 1, ARL
         TEMPAR(K,1) = TRAY(K)
   36 CONTINUE
   EXIT LOOP AND WRITE ARRAY BACK INTO THE RAY FILE.
   GO TO 100
   ENDIF
   ENDIF
   I = L
   J = L + L
   *
   SET UP TO SHIFT DOWN ELEMENT RRA TO ITS PROPER LEVEL
   70 IF (J .LE. IR) THEN
      IF (J .LT. IR) THEN
      *
      COMPARE THE RAY TERMINATION TIMES.
      IF (TEMPAR(SRTFLD,J) .LT. TEMPAR(SRTFLD,(J+1))) J = J + 1
      ENDF
      *
      COMPARE THE RAY TERMINATION TIMES.
      IF (TRAY(SRTFLD,J) .LE. TEMPAR(SRTFLD,J)) THEN
         DO 72, K=1,ARRL
            TEMPAR(K,1) = TEMPAR(K,J)
      72 CONTINUE
         I = J
         J = J + J
      ELSE
      *
      THIS IS RRA'S LEVEL. SET J TO TERMINATE SHIFT DOWN
         J = IR + 1
      ENDIF
**-** LOOP WHILE J LESS THAN OR EQUAL TO IR.
   GO TO 70
ENDIF
**-** PUT RRA INTO ITS SLOT.
   DO 74, K=1,ARRL
       TEMPAR(K,J) = TRAY(K)
   74 CONTINUE
**-** LOOP UNTIL ARRAY IS SORTED.
   GO TO 30
100 CONTINUE
   RETURN
   END
**MODULE NAME : DIVARR**

**MODULE TYPE : SUBROUTINE**

**PROGRAMMER : THOMAS REILLY**

**DATE : DECEMBER 17, 1986**

**DESCRIPTION :**

THIS SUBROUTINE IS DESIGNED TO ACCEPT A SCRATCH ARRAY, A SCRATCH COUNTER ARRAY, AND A SARR ARRAY. THIS SUBROUTINE THEN DIVIDES THE SCRATCH ARRAY BY THE COUNTER ARRAY AND ENTERS IT INTO THE MASTER ARRAY, WHEN 4.5 SECONDS HAS ELAPSED BETWEEN RAYS OR IT THE END OF A FLIGHT SEGMENT.

**MODULE I/O : DIVARR (MASTER, SCRCNT, SCRCH)**

**INPUTS :**

- MASTER : REAL(102,102) : MASTER ARRAY CONTAINING THE POWER VAL
- SCRCNT : INT(102,102) : COUNTER ARRAY OF THE NUMBER OF RAYS THAT HIT A GRID POINT.
- SCRCH : REAL(102,102) : SCRATCH PAD FOR THE RAY POWERS AT EACH GRID POINT.

**OUTPUTS :**

- THE MASTER ARRAY IS RETURNED WITH SCRCH/SCRCNT + MASTER.

**VARIABLE DICTIONARY :**

**CALLING MODULES :**

- GRIDPW (GRNDZ, TEMPCT, MASTER, SCRCNT, SCRCH) ;
  THIS SUBROUTINE CALCULATES THE GRID POWERS AND CREATES THE ARRAYS SCRCNT, SCRCH.

**SUBROUTINE DIVARR (MASTER, SCRCNT, SCRCH)**
DETECTION OF SUBROUTINE INPUT/OUTPUT VARIABLES.

INTEGER LOWBND, UPBND
PARAMETER (LOWBND = -51, UPBND = 50)
REAL SCRCH(LOWBND:UPBND, LOWBND:UPBND)
REAL MASTER(LOWBND:UPBND, LOWBND:UPBND), SCRCH(LOWBND:UPBND, 1)

L LOWBND:UPBND)

DETECTION OF SUBROUTINE DEPENDENT VARIABLES.

INTEGER I, J

DIVIDE THE SCRATCH ARRAY BY THE SCRATCH COUNTER ARRAY.

ENTER THE VALUE INTO THE MASTER ARRAY.

DO 20 J = LOWBND, UPBND
DO 10 I = LOWBND, UPBND
IF (SCRCH(I,J) .NE. 0) THEN
MASTER(I,J) = MASTER(I,J) + (SCRCH(I,J) / SCRCH(I,J))
SCRCH(I,J) = 0.0
SCRCH(I,J) = 0
END IF
10 CONTINUE
20 CONTINUE
RETURN
END

END OF SUBROUTINE DIVARR.
**SUBROUTINE GRIDPW**

* MODULE NAME: GRIDPW
* MODULE TYPE: SUBROUTINE
*
* PROGRAMMER: THOMAS REILLY
* DATE: NOVEMBER 20, 1986
* REVISIONS:
*
* DESCRIPTION:

* MODULE I/O: GRIDPW (GRNDZ, TEMPCT, MASTER, SCRCNT, SCRCH)
* *
* INPUTS: INCLUDING COMMON BLOCKS.
* CONTYP - INTEGER(5) : TYPE OF CONTOURS THE USER WANTS.
* FFT - LOGICAL : TRUE IF CSEL LEVEL IS TO BE USED ON CONTOMAP
* GRNDZ - REAL : Z COORDINATE OF RAY WHERE IT TERMINATES.
* TEMPCT - INTEGER : NUMBER OF RECORDS IN THE INPUT FILE,
*
* OUTPUTS: INCLUDING COMMON BLOCKS.
* MASTER - REAL(102,102) : MASTER ARRAY CONTAINING POWER OF INTI
* LIMAXO - INTEGER : LOWER LIMIT OF THE X INDICE FOR THE GRID
* LIMAYO - INTEGER : LOWER LIMIT OF THE Y INDICE FOR THE GRID
* LIMAX1 - INTEGER : UPPER LIMIT OF THE X INDICE FOR THE GRID
* LIMAY1 - INTEGER : UPPER LIMIT OF THE Y INDICE FOR THE GRID
*
* THE FOLLOWING VARIABLE ARE INPUTED AND OUTPUTED ONLY TO ALLOW THE MEMORY THEY OCCUPY TO BE EQUIVALENCED.
* SCRCH - REAL(102,102) : SCRATCH ARRAY CONTAINING THE POWERS AT EACH GRID POINT.
* SCRCNT - INT(102,102) : COUNTER OF NUMBER OF RAYS HITING EACH GRID POINT.
*
* FILE/IO DICTIONARY:
TEMPFL - TEMPORARY FILE CONTAINING SORTED RAYS.

VARIABLE DICTIONARY:

- AIRT : REAL TIME THE RAY LEAVES THE AIRCRAFT.
- AIRX : REAL X COORDINATE OF RAY WHERE IT LEFT THE A/C
- AIRY : REAL Y COORDINATE OF RAY WHERE IT LEFT THE A/C
- AIRZ : REAL Z COORDINATE OF RAY WHERE IT LEFT THE A/C
- ETIME : REAL CONSTANT OF 4.5, DIFFERENCE IN SEGMENTS.
- GRNDT : REAL TIME THE RAY TERMINATES.
- GRNDX : REAL X COORDINATE OF RAY WHERE IT TERMINATES.
- GRNDY : REAL Y COORDINATE OF RAY WHERE IT TERMINATES.
- GSEC : INTEGER CONSTANT WITH THE VALUE OF THE GRID SEGMENT.
- POWER1 : REAL POWER OF RAY FOR GRID POINT [I,J].
- POWER2 : REAL POWER OF RAY FOR GRID POINT [I+1,J].
- POWER3 : REAL POWER OF RAY FOR GRID POINT [I,J+1].
- POWER4 : REAL POWER OF RAY FOR GRID POINT [I+1,J+1].
- PRESS : REAL PRESSURE OF THE RAY AT TERMINATION.
- RECNT : INTEGER COUNTER FOR THE RECORD IN THE TEMPORARY FILE.
- SLANT : REAL SLANT DISTANCE BETWEEN A/C AND RAY TERMINATION.
- SLANT1 : REAL SLANT DISTANCE BETWEEN A/C AND GRID POINT.
- SLANT2 : REAL SLANT DISTANCE BETWEEN A/C AND GRID POINT.
- SLANT3 : REAL SLANT DISTANCE BETWEEN A/C AND GRID POINT.
- SLANT4 : REAL SLANT DISTANCE BETWEEN A/C AND GRID POINT.
- TIME : REAL CONTAIN TIME TO CHECK FOR 4.5 SECOND SEGMENTS.
- WGHT1 : REAL WEIGHT OF RAY FOR GRID POINT [I,J].
- WGHT2 : REAL WEIGHT OF RAY FOR GRID POINT [I-1,J].
- WGHT3 : REAL WEIGHT OF RAY FOR GRID POINT [I,J+1].
- WGHT4 : REAL WEIGHT OF RAY FOR GRID POINT [I+1,J-1].
- X, Y : INTEGER TEMPORARY VARIABLE FOR COMPUTING GRID POINTS.

CALLING MODULES:

- CONTUR (MASTER, SCRCH, SCRCH) ;

THIS SUBROUTINE CREATES THE TEMPORARY FILES AND INVOKES THIS SUBROUTINE.

CALLED MODULES:

- DIVARR (MASTER, SCRCH, SCRCH) ;

THIS SUBROUTINE DIVIDES THE SCRATCH ARRAY BY THE SCRATCH COUNTER ARRAY AND ENTERS IT INTO THE MASTER ARRAY.

SUBROUTINE GRIDPW (GRNOZ, TEMPCT, MASTER, SCRCH, SCRCH)
* DECLARATION OF SUBROUTINE INPUT/OUTPUT VARIABLES.

COMMON /GRID/ GRDXO, XGS, GRDYM, GDYO, YGS, GRDYM,
1 LIMAXO, LIMAYO, LIMAXO, LIMBYO,
2 LIMAX1, LIMAY1, LIMAX1, LIMBY1

COMMON /STATS/ STATFL, BOOFL, MACHFL, CONTFL, BOOMVA,
+ MACHVA, CONTVA, CONTYP, WIDTH, FFT, SIGNAT,
+ RAYTRA, SCRPA, SCRPS, SCRALL

INTEGER LOWSN, UPBND, LIMAXO, LIMAYO, LIMAX1, LIMAY1
PARAMETER (LOWBND = -51, UPBND = 50)
REAL SCRNT(LOWBND:UPBND, LOWBND:UPBND)
INTEGER CONTYP(5)
REAL MASTER(LOWBND:UPBND, LOWBND:UPBND), SCRCH(LOWBND:
1 UPBND, LOWBND:UPBND), CONTVA(5, 20), MACHVA,
2 WIDTH
LOGICAL FFT, RAYTRA, SIGNAT, STATFL, BOOFL, MACHFL,
1 CONTFL

* DECLARATION OF SUBROUTINE DEPENDANT VARIABLES.

INTEGER GSEC, WPOWER, X, Y, RECNT, TEMPCT
REAL ETIME, PRESS, SLANT, SLANT1, SLANT2, SLANT3,
1 SLANT4, GWHT1, GWHT2, GWHT3, GWHT4, POWER1, POWER2,
2 POWER3, POWER4, AIRX, AIRY, AIRZ, GRNX, GRNY,
3 GRNZ
PARAMETER (GSEC = 2500, WPOWER = 3, ETIME = 5.5)

TIME = 0.0
RECNT = 1

* READ A RAY FROM THE TEMPORARY FILE.
10 CONTINUE
IF (RECNT .LE. TEMPCT) THEN
READ (33, 15, REC=RECNT) AIRT, AIRX, AIRY, AIRZ, GRNDT,
+ GRNDX, GRNHY, TEMP2, PRESS, CSEL
15 FORMAT (F8.2, 3F8.0, F8.2, 2F8.0, F8.3, 2F10.4)
RECNT = RECNT + 1
IF ((CONTYP(1) .EQ. 1) .AND. (FFT) .AND. (CSEL .EQ. 0.0)
1 .OR. (CSEL .EQ. -1.0)) GOTO 10

AIRX = AIRX / 0.3048
ARY = AIRY / 0.3048
AIRZ = AIRZ / 0.3048
GRNDX = GRNDX / 0.3048
GRNHY = GRNHY / 0.3048

* CALCULATE NEAREST GRID POINTS, CHECK THAT RAYS ARE IN BOUNDS.
X = INT(GRNDX / GSEC)
Y = INT(GRNHY / GSEC)
IF (TIME .EQ. 0.0) THEN
TIME = GRNDT
ENDIF

CHECK IF RAY IS WITHIN GRID BOUNDS.
IF (((X .LE. LOUBND) .OR. (X .GE. UPPBD)) .OR. (Y .LE. LOWNB) .OR. (Y .GE. UPPND))) THEN
   GO TO 10
ENDIF

IF ((GRNDT - TIME) .LT. ETIME) THEN
   CALCULATE SLANT DISTANCE FROM A/C TO RAY TERMINATION.
   SLANT = ((AIRX - GRNDX)**2.0 + (AIRY - GRNDY)**2.0 + 
            (AIRZ - GRNDZ)**2.0)**0.5
   SLANT1 = ((AIRX - (X * GSEC))**2.0 + (AIRY - (Y * GSEC))**2.0 + (AIRZ - GSEC)**2.0)**0.5
   SLANT2 = ((AIRX - ((X+1) * GSEC))**2.0 + (AIRY - (Y + GSEC))**2.0 + (AIRZ - GRNDZ)**2.0)**0.5
   SLANT3 = ((AIRX - (X * GSEC))**2.0 + (AIRY - ((Y+1) * GSEC))**2.0 + (AIRZ - GRNDZ)**2.0)**0.5
   SLANT4 = ((AIRX - ((X+1) * GSEC))**2.0 + (AIRY - ((Y + 1) * GSEC))**2.0 + (AIRZ - GRNDZ)**2.0)**0.5
   CALCULATE WEIGHTS FOR EACH OF THE FOUR GRID POINTS.
   WGHT1 = (SLANT / SLANT1)**WPOWER
   WGHT2 = (SLANT / SLANT2)**WPOWER
   WGHT3 = (SLANT / SLANT3)**WPOWER
   WGHT4 = (SLANT / SLANT4)**WPOWER
   CALCULATE THE POWERS OF EACH OF THE FOUR GRID POINTS.
   IF (((CONTP(1) .EQ. 1) .AND. (FFT)) THEN
      POWER1 = WGHT1 * (CSEL**2.0)
      POWER2 = WGHT2 * (CSEL**2.0)
      POWER3 = WGHT3 * (CSEL**2.0)
      POWER4 = WGHT4 * (CSEL**2.0)
   ELSE
      POWER1 = WGHT1 * (PRESS**2.0)
      POWER2 = WGHT2 * (PRESS**2.0)
      POWER3 = WGHT3 * (PRESS**2.0)
      POWER4 = WGHT4 * (PRESS**2.0)
   END IF
   ADD THE POWERS TO THE APPROPRIATE SCRATCH PAD CELL AND INCR
   THE APPROPRIATE SCRATCH COUNTER CELL.
   SCRCH(X,Y) = SCRCH(X,Y) + POWER1
   SCRCH((X+1),Y) = SCRCH((X+1),Y) + POWER2
   SCRCH(X,(Y+1)) = SCRCH(X,(Y+1)) + POWER3
   SCRCH((X+1),(Y+1)) = SCRCH((X+1),(Y+1)) + POWER4
**SCRCNT(X,Y)** = SCRCNT(X,Y) + 1
**SCRCNT(X+1,Y)** = SCRCNT((X+1),Y) + 1
**SCRCNT(X,(Y+1))** = SCRCNT(X,(Y+1)) + 1
**SCRCNT((X+1),(Y+1))** = SCRCNT((X+1),(Y+1)) + 1

LIMAXO = MINO(LIMAXO,(X+52))
LIMAYO = MINO(LIMAYO,(Y+52))
LIMAX1 = MAXO(LIMAX1,(X+52))
LIMAY1 = MAXO(LIMAY1,(Y+52))

ELSE IF END OF FLIGHT SEGMENT OR ELAPSED TIME GREATER THAN 4.5
ELSE

IF ((GRNDT - TIME) .GE. ETIME) THEN
RECNT = RECNT - 1
ENDIF

DIVIDE SCRATCH ARRAY BY COUNTER SCRATCH ARRAY AND ADD RESULT TO THE MASTER ARRAY.
CALL DIVARR( MASTER, SCRCNT, SCRCNT)

TIME = 0.0
ENDIF

END OF LOOP.
TIME = GRNDT
GO TO 10
ENDIF

CALL DIVARR( MASTER, SCRCNT, SCRCNT)
RETURN
END

END OF SUBROUTINE GRIDPOW.
SUBROUTINE SRTRAY

* MODULE NAME : SRTRAY
* MODULE TYPE : SUBROUTINE
*
* PROGRAMMER : THOMAS REILLY
* DATE : DECEMBER 1, 1986
* REVISIONS :
*
* DESCRIPTION :
*
* THIS SUBROUTINE IS DESIGNED TO SORT THE TEMPORARY FILE TEMPFL, ON THE RAY'S TERMINATION TIME. THE NUMBER OF RECORDS TO BE SORTED, IN THE FILE TEMPFL, IS PASSED INTO THE SUBROUTINE. THE FILE IS THEN SORTED USING A HEAP SORT, ON TERMINATION TIMES.
*
* MODULE I/O : SRTRAY (TEMPCT, INFILE, SRTFLD)
* ----------
* INPUTS :
* INFILE INTEGER : NUMBER OF THE FILE TO BE SORTED.
* TEMPCT INTEGER : NUMBER OF RECORDS IN THE SEGMENT TO BE SORTED.
* SRTFLD INTEGER : FIELD THE FILE IS TO BE SORTED ON.
* OUTPUTS : THE SORTED INPUT FILE INFILE.
*
* FILE/I/O DICTIONARY :
* -----------------------
* TEMPFL : DATA FILE THAT CONTAINS THE RAY'S START POINTS, END POINTS, TERMINATION TIMES, AND PRESSURE.
*
* VARIABLE DICTIONARY :
* ---------------------
* I INTEGER : USED TO TRAVERSE THE ARRAYS TO SORT.
* IR INTEGER : USED TO TRAVERSE THE ARRAYS TO SORT.
* J INTEGER : USED TO TRAVERSE THE ARRAYS TO SORT.
* K INTEGER : LOOP CONTROL VARIABLE.
* L INTEGER : USED TO TRAVERSE THE ARRAYS TO SORT.
* TRAY REAL(11) : TEMPORARY VARIABLE TO HOLD AN ELEMENT OF DATA WHILE IT IS BEING SWITCHED WITH ANOTHER.
* TRAY2 REAL(11) : TEMPORARY VARIABLE TO HOLD AN ELEMENT OF DATA WHILE IT IS BEING SWITCHED WITH ANOTHER.
* TRAY3 REAL(11) : TEMPORARY VARIABLE TO HOLD AN ELEMENT OF DATA WHILE IT IS BEING SWITCHED WITH ANOTHER.
SUBROUTINE SRTRAY(TEMPCT, INFILE, SRTFLD)
  
  DECLARATION OF SUBROUTINE INPUT/OUTPUT VARIABLES.
  INTEGER TEMPCT, INFILE, SRTFLD

  DECLARATION OF SUBROUTINE DEPENDANT VARIABLES.
  INTEGER I, J, K, L, ARRL
  PARAMETER (ARRL = 11)
  REAL TRAY(ARRL), TRAY2(ARRL), TRAY3(ARRL)

  SET UP INITIALIZATION FOR HEAPSORT.
  L = TEMPCT / 2 + 1
  IR = TEMPCT

  HEAP CREATION PHASE.
  30 CONTINUE
  IF (L .GT. 1) THEN
    L = L - 1
  ENDIF

  INITIALIZE TRAY TO RECORD L IN THE PAY FILE.
  READ (INFILE, 120, REC=L ) (TRAY(K), K=1,ARRL)
  ELSE

  PLACE TOP OF HEAP AT THE END OF THE FILE.
  READ (INFILE, 120, REC=IR ) (TRAY(K), K=1,ARRL)
  READ (INFILE, 120, REC=1 ) (TRAY2(K), K=1,ARRL)
  WRITE (INFILE, 120, REC=IR ) (TRAY2(K), K=1,ARRL)
  IR = IR - 1

  PLACE SMALLEST ELEMENT AT THE BEGINING OF THE FILE.
  IF (IR .EQ. 1) THEN
    WRITE (INFILE, 120, REC=1 ) (TRAY(K), K=1,ARRL)
  ENDIF

  EXIT LOOP AND RETURN TO CALLING MODULE.
  GO TO 100

ENDIF
ENDIF
I = L
J = L + L
SET UP TO SHIFT DOWN ELEMENT TRAY TO ITS PROPER LEVEL

70 IF (J .LE. IR) THEN
   IF (J .LT. IR) THEN
     *- COMPARISON OF RAY TERMINATION TIMES.
     READ (INFILE, 120, REC=J ) (TRAY2(K), K=1,ARRL)
     READ (INFILE, 120, REC=(J+1) ) (TRAY3(K), K=1,ARRL)
     IF (TRAY2(SRTFLD) .LT. TRAY3(SRTFLD)) J = J + 1
     ENDIF
   ENDIF
   *- COMPARISON OF RAY TERMINATION TIMES.
   READ (INFILE, 120, REC=J ) (TRAY2(K), K=1,ARRL)
   IF (TRAY(SRTFLD) .LT. TRAY2(SRTFLD)) THEN
     WRITE (INFILE, 120, REC=I ) (TRAY2(K), K=1,ARRL)
     I = J
     J = J + 1
   ELSE
     J = I + 1
   ENDF
   *- LOOP WHILE J LESS THAN OR EQUAL TO IR.
   GO TO 70
   ENDF

*- PUT TRAY INTO ITS SLOT.
WRITE (INFILE, 120, REC=I ) (TRAY(K), K=1,ARRL)
*- LOOP UNTIL ARRAY IS SORTED.
GO TO 30

100 CONTINUE
RETURN

120 FORMAT (FB.2, 3FB.0, FB.2, 2FB.0, FB.3, 3F10.4)
END
*- END OF SUBROUTINE SORTRAY.
* * *

SUBROUTINE CONTUR

* * *

MODULE NAME: CONTUR

* * *

MODULE TYPE: SUBROUTINE

* * *

PROGRAMMER: THOMAS REILLY

* * *

DATE: DECEMBER 5, 1986

* * *

REVIONS

* * *

DESCRIPTION:

* * *

THIS SUBROUTINE IS USED TO INVOKE 'GETINX', WHICH RETU
THE STARTING RECORD, AND NUMBER OF RECORDS FROM THE INDEX F
FOR A VALID FLIGHT SEGMENT. THIS INFORMATION IS THEN U
TRANSFER THE DATA FROM THE FILE CLIBRY TO A TEMPORARY DATA
THE SUBROUTINE SORTRAY IS THEN CALLED TO SORT THE TEMPORARY
ON RAY TERMINATION TIME. NEXT THE SUBROUTINE GRIDPOW IS CA
WHICH CALCULATES THE GRID POWERS FOR EACH RAY.

* * *

MODULE I/O:

CONTUR (MASTER, SCRCHT, SCRCH, CONTFL, SCRPAD, SCRALL)

* * *

INPUTS:

CONTFL - LOGICAL : TRUE IF CONTOURS ARE TO BE PLOTTED.

SCRPAD - LOGICAL : TRUE IF SCRCHPADS ARE TO BE PLOTTED.

SCRALL - LOGICAL : TRUE IF ALL SCRCHPADS ARE TO BE PLOTTED.

* * *

OUTPUTS:

INCLUDING COMMON BLOCKS.

MASTER - REAL(102,102) : MASTER ARRAY CONTAINING POWER OF INTI

FLIGHT TRACK.

LIMAXO - INTEGER : LOWER LIMIT OF THE X INDICE FOR THE GRID

LIMAYO - INTEGER : LOWER LIMIT OF THE Y INDICE FOR THE GRID

LIMAX1 - INTEGER : UPPER LIMIT OF THE X INDICE FOR THE GRID

LIMAY1 - INTEGER : UPPER LIMIT OF THE Y INDICE FOR THE GRID

* * *

THE FOLLOWING VARIABLE ARE INPUTED AND OUTPUTED ONLY TO ALLOW
THE MEMORY THEY OCCUPY TO BE EQUIVALANCED.

SCRCH - REAL(102,102) : SCRATCH ARRAY CONTAINING THE POWERS A

EACH GRID POINT.

SCRNT - INT(102,102) : COUNTER OF NUMBER OF RAYS HITING EACH

GRID POINT.

* * *

FILE/I0 DICTIONARY:

* * *
CINDEX - INDEX FILE FOR THE RAY DATA FILE CLIBRY.
CFILE - DATA FILE FOR THE RAY INFORMATION.
TEMPFL - TEMPORARY FILE USED TO SORT THE RAYS AND CALCULATE TH
GRID POWERS.

VARIABLE DICTIONARY:

EOFFLG - LOGICAL : TRUE IF EOF OF THE INDEX FILE.
EOFSEG - LOGICAL : TRUE IF END OF FLIGHT TRACK.
GRNDZ - REAL : Z COORDINATE OF THE RAY TERMINATION AT TH
NREC - INTEGER : NUMBER OF RAY RECORDS FOR THIS FLIGHT SEG
OPR - LOGICAL : TRUE IF THERE ARE CAUSTICS IN THE FLIGHT
RAYS - REAL(11) : ARRAY USED TO TRANSFER THE RAY DATA TO TH
FILE.
RSTIME - REAL : CONTAINS THE PREVIOUS RAYS START TIME.
SREC - INTEGER : STARTING RECORD OF THE RAY RECORDS FOR FL
TEMP - REAL : DUMMY VARIABLE USED FOR AN UNNEEDED RETUR
PARAMETER.
TEMPCT - INTEGER : COUNTER OF THE NUMBER OF RAYS IN THE TEMP

CALLED MODULES:

GETINX (SREC, NREC, SOPR, EOFFLG, TEMP, NOPR, RAYINX) ;
SELECTS VALID FLIGHT, RETURNS SREC, NREC, EOFFLG.

SRTRAY (TEMPCT, INFILE, SRTFLD) ;
SORTS RAYS ON THEIR GROUND TERMINATION TIME.
INFILE - INTEGER : NUMBER OF FILE TO BE SORTED.
SRTFLD - INTEGER : FIELD THE FILE IS TO BE SORTED ON.

GRIDPW (MASTER, SCRCNT, SCRCH) ;
CALCULATES THE GRID POWERS OF THE RAYS.

CALLING MODULE:

SUBROUTINE CONTUR( MASTER, SCRCNT, SCRCH, CONTFL, SCRPA, SCRALL,
1 TEMPAR, TITLE, LAT, LONG)
INTEGER LOWBND, UPBND
PARAMETER (LOWBND = -51, UPBND = 50)
INTEGER LIMAXO, LIMAYO, LIMAX1, LIMAY1
REAL SCRCNT(LOWBND:UPBND, LOWBND:UPBND), SCRCH(LOWBND:UPBND)
REAL MASTER(LOWBND:UPBND, LOWBND:UPBND), SCRC(LOWBND:UPBND)
INTEGER LIMAXO, LIMAYO, LIMAX1, LIMAY1
LOGICAL CONTFL, SCRPAZ, SCRALL
DIMENSION TEmPAR(11,1000)

*.- DECLARATION OF SUBROUTINE DEPENDANT VARIABLES/
*.- INTEGER NREC, SREC, TEMPCT, NOPR, SOPR, J, ARRL, RTYPE
*.- INTEGER LOOP, TDAT, ELOOP
*.- PARAMETER (ARRL = 11)
*.- REAL RAYS(ARRL), RAYS2(ARRL), RAYS3(ARRL), RAYS4(ARRL)
*.- LOGICAL EOFFLAG, EOFSEG, OPR, EFILE
*.- CHARACTER*110 BUF
*.- CHARACTER*10 TOATE, TTIME
*.- CHARACTER*70 TITLE
*.- CHARACTER*8 ACTAIL
*.- CHARACTER*10 MSITE, LAT, LONG

ASSIGN 10 TO LOOP
ASSIGN 20 TO TDAT
ASSIGN 30 TO ELOOP

*.- LOOP UNTIL END OF THE INDEX FILE.
*.- EFILE = .TRUE.
*.- EOFSEG = .FALSE.
*.- LIMAXO = 102
*.- LIMAYO = 102
*.- LIMAX1 = 1
*.- LIMAY1 = 1
DO 13, I = -51, 50
   DO 12, J = -51,50
      MASTER(J,I) = 0.0
      SCRCH(J,I) = 0.0
      SCRCNT(J,I) = 0
12 CONTINUE
13 CONTINUE

CONTINUE

GET STARTING RECORD AND NUMBER OF RECORDS FROM INDEX FILE.
 CALL GETINX(SREC, NREC, SOPR, EOFFLAG, 11, NOPR, OPR)
 SOPR1 = SOPR
 NOPR1 = NOPR
 IF (NREC.LE.0.AND..NOT.EOFFLAG) GOTO LOOP
 EREC = (SREC + NREC) - 1

*.- CHECK IF END OF INDEX FILE.
*.- IF ((EOFFLAG).AND. (EFILE)) RETURN
EFILE = .FALSE.
IF (EOFFLG) GO TO ELOOP

*.
ENTER DATA INTO SCRATCH PAD FILE TO BE PLOTTED LATER.
IF ((OPR) .AND. (SCRPAD)) THEN
  READ(S1,FMT='(A)',REC=I1)
  SUF
  READ(BUF(61:68),FMT=('A8')) ACTAIL
  CALL SCRNFL(SREC, NREC, SOPR, NOPR, ACTAIL, SCRALL, TEMPAR)
ENDIFTEMPAR

IF (.NOT.(CONTFL)) THEN
  GOTO 10
ENDIF

*.
READ THE Z GROUND COORDINATE.
READ(S2, 15, REC=SREC) GRNDZ

*TDAT START LOOP TO TRANSFER RAY DATA.
20 CONTINUE
READ(S2,25,REC=SREC) RTYPE,(TEMPAR(I,TEMPCT),I=1,ARRL)

31 IF (CURTIM .NE. TEMPAR(1,TEMPCT)) .OR. (RTYPE .NE. 0)) THEN
  TEMPCT = TEMPCT - 1
  IF (TEMPCT .GT. 1) THEN
    CALL SORTRY(TEMPCT,TEMPAR,8)
  END IF
  WRITE(33,35,REC=NCNdT) (TEMPAR(K,1),X=1,ARRL)
  NCNT = NCNT + 1
DO 200 I = 2, TEMPCT
  DPHI = INT(ABS(TEMPAR(B,1) - TEMPAR(B,1-1))) - 1
  CPHI = TEMPAR(B,1-1)
  IF ((DPHI .GT. 2) .AND. (ABS(TEMPAR(B,1)) .LT. 180.)
    .AND. (ABS(TEMPAR(B,1-1)) .LT. 180.) THEN
    DO 250 L = 1, DPHI
      CPHI = CPHI + 1.0
      FACTOR = (CPHI - TEMPAR(B,1-1))/(TEMPAR(B,1) - TEMPAR(B,1-1))
      RAYS2(1) = TEMPAR(1,1)
      RAYS2(2) = TEMPAR(2,1)
      RAYS2(3) = TEMPAR(3,1)
RAYS2(4) = TEMPAR(4,1)
RAYS2(5) = TEMPAR(5,1-1) + (TEMPAR(5,1) - TEMPAR(5,1-1))*FACTOR
RAYS2(6) = TEMPAR(6,1-1) + (TEMPAR(6,1) - TEMPAR(6,1-1))*FACTOR
RAYS2(7) = TEMPAR(7,1-1) + (TEMPAR(7,1) - TEMPAR(7,1-1))*FACTOR
RAYS2(8) = CPHI
RAYS2(9) = TEMPAR(9,1-1) + (TEMPAR(9,1) - TEMPAR(9,1-1))*FACTOR
RAYS2(10) = TEMPAR(10,1-1) + (TEMPAR(10,1) - TEMPAR(10,1-1))*FACTOR ELSE
RAYS2(10) = -1.0
ENDIF
RAYS2(11) = TEMPAR(11,1-1) + (TEMPAR(11,1) - TEMPAR(11,1-1))*FACTOR
WRITE(33,35,REC=NCNT) (RAYS2(K),K1,ARRL)
NCNT = NCNT + 1
CONTINUE
ENDIF
WRITE(33,35,REC=NCNT) (TEMPAR(K,1),K1,ARRL)
NCNT = NCNT + 1
CONTINUE
TEMPCT = 0
CURTIM = 0.0
215 IF (RSTYPE .EQ. 0) THEN
READ(52,25,REC=SREC) RTYPE,(RAYS2(K),K1,ARRL)
IF ((RSTYPE .EQ. 0) .AND. (SREC .LE. EREC)) THEN
WRITE(33,35,REC=NCNT) (RAYS2(K),K1,ARRL)
NCNT = NCNT + 1
SREC = SREC + 1
IF (SREC .GT. EREC) GOTO 215
ENDIF
IF (SREC .GT. EREC) GOTO 45
SREC = SREC + 1
ENDIF
END IF
IF ((RAYS(1) - RSTYPE) .GT. 5.5) THEN
RSTYPE = RAYS(1)
ELSE
IF (SREC .LT. EREC) THEN
SREC = SREC + 1
TEMPCT = TEMPCT + 1
RSTYPE = RAYS(1)
GO TO TDAT
ELSE
END IF
25 FORMAT (12, F8.2, 3F8.0, F8.2, 2F8.0, F8.3, 3F10.4)
35 FORMAT (F8.2, 3F8.0, F8.2, 2F8.0, F8.3, 3F10.4)
IF (RSTYPE .EQ. 0.0) THEN
RSTYPE = RAYS(1)
ENDIF
IF (TEMPCT .GT. 1) THEN
    TEMPCT = TEMPCT + 1
    CURTIM = 0.0
    GOTO 31
ENDIF
END IF
END IF

* - END OF TRANSFER LOOP.

45 CONTINUE

* - SORT THE FLIGHT SEGMENT ON TERMINATION TIME.
CALL SRTRAY((NCNT-1), 33, 5)

* - CALCULATE THE GRID POWERS.
IF (SREC .GE. EREC) THEN
    EOFSEG = .TRUE.
ENDIF
CALL GRIDPW (GRNDZ, NCNT-1, MASTER, SCRCNT, SCRCH)
OPVAL = -.99999.0
DO 500 IJ = SOPR1, ((NOPR1+SOPR1)-1)
    READ(52,1001,REC=IJ) T1,T2,T3,T4,T5,XC,YC,OPV,T6,T7
1001 FORMAT(F8.2,3F8.0,F8.2,2F8.0,3F10.4)
    IF (OPVAL .LT. OPV) THEN
        OPVAL = OPV
        XCOORD = XC
        YCOORD = YC
    ENDIF
500 CONTINUE

XCOORD = INT(XCOORD/0.3048)
YCOORD = INT(YCOORD/0.3048)
WRITE(5,FMT='(2FIO.0)') XCOORD,YCOORD

IF (.NOT. EOFSEG) THEN
    GO TO TDAT
ELSE
    GO TO LOOP
ENDIF

*ELOOP END OF INDEX FILE OUTPUT MASTER ARRAY.
30 CONTINUE

* - PUT MASTER ARRAY INTO DB.
DO 28 I=-51,49
    DO 26 J = -51,49
        IF (MASTER(J+1,I+1) .GT. 0.0) THEN
            MASTER(J,I) = 20 * LOG10(MASTER(J+1,I+1)) + 68
        ENDIF
26 CONTINUE
28 CONTINUE

C
C WRITE GRID ON LINE PRINTER
C
C*E*E*
TO OUTPUT THE LINE PRINTER PLOTS REG-JVE THE FOLLOWING GOTO.

GOTO 830
WRITE (15,6920) (1, I=1,25)
WRITE (15,6921) (50-J, (MASTER(I,50-J), I=50,-26), J=50,49)
WRITE (15,6920) (1, I=26,50)
WRITE (15,6921) (50-J, (MASTER(I,50-J), I=25,-1), J=50,49)
WRITE (15,6920) (1, I=51,75)
WRITE (15,6921) (50-J, (MASTER(I,50-J), I=0,24), J=50,49)
WRITE (15,6920) (1, I=76,100)
WRITE (15,6921) (50-J, (MASTER(I,50-J), I=25,49), J=50,49)

830 CONTINUE
C
IF (CONTFL) THEN
   CALL CPCONT(TITLE,TDATE,TTIME,LAT,LONG,MASTER)
   CALL CPX4OP(TITLE,TDATE,TTIME,LAT,LONG)
ENDIF

6920 FORMAT ('1', 5X,25I5)
6921 FORMAT ('0', I4, 1X, 25F5.1)
RETURN
END
*-
END OF SUBROUTINE CONTOUR.
SUBROUTINE PLOTDR

* MODULE NAME : PLOTDR
* MODULE TYPE : SUBROUTINE
*
* PROGRAMMER : THOMAS REILLY
* DATE : APRIL 22, 1987
* REVISIONS :
*
* DESCRIPTION :
** THIS SUBROUTINE WAS DESIGNED TO DRIVE THE SCRCHPAD
** PLOTTING SUBROUTINES AND THE CONTOURING SUBROUTINES.
*
* MODULE I/O : PLOTDR(MASTER, SCRCH, SCRCNT)
*
* INPUTS :
* CONTFL LOGICAL : TRUE IF CONTOURS ARE TO BE PLOTTED.
* MASTER REAL(102,102) : MASTER ARRAY FOR THE CONTOUR GRID.
* SCRCH REAL(102,102) : SCRCH ARRAY FOR THE CONTOURING.
* SCRCNT INT (102,102) : SCRCH COUNTER ARRAY FOR THE CONTOURING.
* SCRPAD LOGICAL : TRUE IF THE SCRCHPAD PLOTS ARE TO BE PL
* SCRPSF LOGICAL : TRUE IF THE SCRCHPAD PLOTS ARE TO BE IN
* SCRALL LOGICAL : TRUE IF ALL THE SCRCHPAD PLOTS ARE TO B
*
* OUTPUTS : NONE.
*
* FILE DICTIONARY :
* ............
*
* CLIBRY - CONTAINS THE RAY LIBRARY.
* CINDEX - INDEX TO THE DIRECT ACCESS FILE CLIBRY.
* HOLDFL - TEMPORARY FILE USED IN THE SCRCHPAD PLOTTING.
* TEMPFL - TEMPORARY FILE USED IN THE CONTOUR PLOTTING
* SCRCHFL - FILE USED IN THE SCRCHPAD PLOTTING.
*
* SUBROUTINE PLOTDR(TITLE,GPCPFL, GPCPMH, GPCPBM)

COMMON /STATS/ STATFL, BOOMFL, MACHFL, CONTFL, BOOMVA,
1 MACHVA, CONTVA, CONTYP, WIDTH, FFT, SIGNAT,
2 RAYTRC, SCRFL, SCRPSF, SCRALL

INTEGER CONTYP(5)
REAL BOOMVA, MACHVA, WIDTH, CONTVA(5,20)
LOGICAL STATFL, BOOMFL, MACHFL, CONTFL, FFT, SIGNAT,
1 RAYTRC, SCRFL, SCRPSF, SCRALL, GPCPFL,GPCPMH,
1 GPCPBM

CHARACTER*70 TITLE
CHARACTER*10 LAT, LONG, TDATE, TTIME
REAL SCRCNT(-51:50,-51:50)
REAL MASTER(-51:50,-51:50), SCRCH(-51:50,-51:50)
DIMENSION TEMPAR(11,1000)

OPEN(33,FILE='TEMPFL',ACCESS='DIRECT',FORM='FORMATTED',
    1 RECL=100)
OPEN(32,FILE='SCRCHFL',ACCESS='SEQUENTIAL',FORM='FORMATTED')
OPEN(51,FILE='CINDEX',STATUS='UNKNOWN',ACCESS='DIRECT',
    1 FORM='FORMATTED',RECL=110,BLANK='NULL')
OPEN(52,FILE='CLIBRY',STATUS='UNKNOWN',ACCESS='DIRECT',
    1 FORM='FORMATTED',RECL=110,BLANK='NULL')
OPEN(34,FILE='HOLDFL',ACCESS='SEQUENTIAL',FORM='FORMATTED')
OPEN(35,FILE='TMPL2',ACCESS='DIRECT',FORM='FORMATTED',
    1 RECL=100)
OPEN(11,FILE='TAPEFL',ACCESS='SEQUENTIAL',FORM='FORMATTED')
OPEN(5,FILE='FILS')

CALL CONTUR TO CREATE CONTOUR PLOTS AND SET UP SCRPAF FILE.
IF (GPCPFL) CALL CPINIT(TITLE)
CALL CONTUR(MASTER, SCRCNT, SCRCH, CONTFL, SCRPL, SCRALL,
    1 TEMPAR, TITLE,LAT,LONG)
IF (GPCPFL) THEN
    IF (GPCPMH) CALL CPSSTR(TITLE, TOATE, TTIME, LAT,LONG)
    IF (GPCPMH) CALL CPBMTR(TITLE, TOATE, TTIME, LAT,LONG)
    CALL CPTERM
END IF

CHECK IF SCRCH PAD PLOTS ARE TO BE PLOTTED.
IF (.NOT. (SCRPL)) RETURN
CALL SCRPAF(SCRPSF,TEMPAR)

RETURN
END
SUBROUTINE RNGLL (SITLC, LAT, LONG)
C
DIMENSION SITE(20), SITLAT(20), SITLON(20)
C
CHARACTER*10 SITE, SITLAT, SITLON, LAT, LONG
CHARACTER*(*) SITLC
C
DO 20 I=1,20
LAT = SITLAT(I)
LONG = SITLON(I)
IF (SITE(I) .EQ. ' ') RETURN
IF (SITLC .EQ. SITE(I)) RETURN
20 CONTINUE
C
LAT = ''
LONG = ''
RETURN
C
DATA SITE/"OCEANA", "TYNDALL", "LUKE", "HOLLOMAN", "NELLIS"
1, "YUMA", 14*" /
DATA SITLAT/ 36 00.0 N', 29 32.0 N', 32 23.48N', 33 48.0 N'
1, 36 50.29N', 32 29.24N', 14" UNKNOWN ' /
DATA SITLON/ 75 10.0 W', 84 37.0 W', 113 15.0 W', 106 25.0 W'
1, 115 25.36W', 113 52.56W', 14" UNKNOWN ' /
END
SUBROUTINE CPTERM
C
C ROUTINE TO TERMINATE GPCP FILE
C
WRITE (11,1101)
ENDFILE 11
REWIND 11
RETURN
C
1101 FORMAT ('STOP')
END
SUBROUTINE CPINIT (TITLE)

ROUTINE TO INITIALIZE OUTPUT TO GPCC

CHARACTER(*) TITLE

REWIND 11
RETURN

END
SUBROUTINE CPCONT (TITLE, TDATE, TTIME, LAT, LONG, GRID)

ROUTINE TO OUTPUT GPCP CARDS TO PLOT CONTOURS

COMMON /GRID/ GROXO, XGS, GROXM, GROYO, YGS, GROYM,
1 LIMAXO, LIMAYO, LIMBXO, LIMBYO,
2 LIMAX1, LIMAY1, LIMBX1, LIMBY1

COMMON /STATS/ STATF3, 800MFG, MACNFG, CONTFG,
1 SOOMYL, MACHVL,
2 CONTVL(5,20), CONTYP(5), WIDTH, FFT,
3 SIGNAT, RAYTRC, SCRPAO, SCRPSF, SCRALL

DIMENSION IAN(5), GRID(102,102)

 CHARACTER(*) TITLE, TDATE, TTIME, LAT, LONG
 CHARACTER*70 MAPANO, XANO, YANO, MAP(3)
 CHARACTER MAP1*30, MAP2*30, MAP3*10

INTEGER CONTYP
LOGICAL CONFG, FFT, SIGNAT, RAYTRC, SCRPAO, SCRPSF, SCRALL

DATA IAN /-100000,-50000,0,50000,100000/
DATA MAP(1) /'CONTOURS OF AVERAGE C-WEIGHTED SOUND EXPOSURE LEVEL /
1 (CSEL), IN DB'/
DATA MAP(2) /'CONTOURS OF C-WEIGHTED DAY/NIGHT AVERAGE LEVEL (DNL /
1), IN DB'/
DATA MAP(3) /'CONTOURS OF AVERAGE PEAK OVERPRESSURE IN POUNDS PER /
1 SQUARE FOOT'/
DATA XANO /'RANGE X·COORDINATE IN FEET'/
DATA YANO /'RANGE Y·COORDINATE IN FEET'/

CONFG = .FALSE.

DO 100 KK=1,5
IF (CONTYP(KK) .EQ. 0) RETURN
CONTYP = CONTYP(KK)
IF (CONTYP .LT. 1 .OR. CONTYP .GT. 3) STOP80
MAPANO = MAP(CONTYP)

IF (CONTVL(KK,1) .LE. 0. .OR. CONTVL(KK,2) .LE. 0.) GOTO 100

WRITE (11,1101) TITLE
WIDTH = AMAX1(8.0, AMIN1(WIDTH, 48.))
WRITE (11,1102) WIDTH
WRITE (11,1103) 16

SCALE = CONTVL(KK,1) / 12.0
ISCALE = IFIX(SCALE + 0.5)
IGS = IFIX(XGS + 0.5)
XMIN = GROXO + XGS/2.0
XMAX = GRDXX - XGS/2.0
YMIN = GRDYO + YGS/2.0
YMAX = GRDXX + YGS/2.0

IF THIS IS LDN CONTOUR, EXTRACT REFERENCE NUMBER OF OPS

IF (NCTYPE .EQ. 2) THEN
  FNOPS = CONTVL(KK,2)
  OPSADJ = 10. * ALOG10(FNOPS)
  IPTR = 3
ELSE
  IPTR = 2
ENDIF

WRITE (11,1104) ISCALE, ISCALE, 0., 1.50, XMIN, IGS, XMAX, 1
                      YMIN, IGS, YMAX

IF THIS IS NOT THE FIRST CONTOUR MAP THEN SIMPLY
RESTORE THE GRID ARRAY, OTHERWISE OUTPUT THE CONTROL
POINTS SO ARRAY CAN BE GENERATED

IF (COMFG) THEN
  WRITE (11,1111)
ELSE
  WRITE (11,1112)
  WRITE (11,1105) 0.17, 0.17, 1, 2, 21, 21
  IO = MAXO(1, LIMAXO-2)
  II = MINO(100, LIMAX1+2)
  JO = MAXO(1, LIMAYO-2)
  JJ = MINO(100, LIMAY1+2)

  DO 40 I=IO,II
  XP = GRDXX + XGS*FLOAT(I-1)
  DO 40 J=JO,JJ
  YP = GRDYO + YGS*FLOAT(J-1)
  40 WRITE (11,1106) XP, YP, GRIDA(I,J), 2

  WRITE (11,1107)
  CONFG = .TRUE.
ENDIF

OUTPUT CONTOUR VALUES TO BE PLOTTED

OSETLB = 0.
DO 50 I=IPTR,20
  IF (CONTVL(KK,I) .LE. 0.) GOTO 60
  IF (NCTYPE .EQ. 1) THEN
    CONLAB = CONTVL(KK,I)
    CONLEV = CONLAB
    NOCHR5 = 6
    IFMT = 0
  ELSEIF (NCTYPE .EQ. 2) THEN
    CONLAB = CONTVL(KK,I)
  ENDIF
  OSETLB = OSETLB + 0.17
  GOTO 60
50                             I=I+1

60                                 I=I+1

ENDIF
CONLEV = CONLAB * OPSADJ + 49.3651
NOCHR$ = 4
IFMT = 0
ELSEIF (NCTYPE .EQ. 3) THEN
CONLAB = CONTVL(KX,I)
CONLEV = 20. * ALOG10(CONLAB) + 101.6
NOCHR$ = 4
IFMT = 1
ENDIF
WRITE (11,1108) CONLEV, CONLAB, OSETLB, 0.125, 1, NOCHR$, IFMT
OSETLB = OSETLB + 1.50
50 CONTINUE
60 CONTINUE
WRITE (11,1109)
   CALCULATE HEIGHT OF 'TITLE' CHARACTERS (MAX=0.25 IN)
   AND PLOT TITLE
   BRORLN = (YMAX - YMIN) / SCALE
   CHGHT = AMIN1(0.25, (BRORLN-2.5) / 80.)
   WRITE (11,1120) 0.500, 1.000, 0.0, CHGHT, TITLE(1:30),
   (TITLE(31:60), TITLE(61:70)
   CALCULATE HEIGHT OF 'MAP TYPE' AND 'SCALE' CHARACTERS
   (MAX=0.20 IN) AND PLOT 2 LINES OF TEXT
   CHGHT = AMIN1(0.20, (BRORLN-3.0) / 80.)
   MAP1 = MAPANO(1:30)
   MAP2 = MAPANO(31:60)
   MAP3 = MAPANO(61:70)
   WRITE (11,1120) 1.000, 0.625, 0.0, CHGHT, MAP1,
   (MAP2, MAP3
   WRITE (11,1122) 1.000, 0.250, 0.0, CHGHT, ISCALE, LAT, LONG
   DRAW BOX AROUND TITLE BLOCK
   WRITE (11,1121) 0.00, 1.50, 0.00, 0.00,
   (0.00, 0.00, BRORLN, 0.00,
   2, BRORLN, 0.00, BRORLN, 1.50,
   3, BRORLN-1.5, 1.50, BRORLN-1.5, 0.0
   PUT TIC MARKS ON MAP EDGE
   YTICO = 1.5
   YTIC1 = YTICO + 0.15
   DO 82 I=1,9
   XTIC = FLOAT(I) * BRORLN / 10.
   WRITE (11,1121) XTIC, YTICO, XTIC, YTIC1
82 CONTINUE
   YTICO = BRORLN - 1.5
   YTIC1 = YTICO - 0.15
   DO 84 I=1,9
XTIC = FLOAT(I) * BRDRLN / 10.
WRITE (11,1121) XTIC, YTIC0, XTIC, YTIC1
84 CONTINUE
C
XTICO = 0.0
XTIC1 = XTICO + 0.15
DO 86 I=1,9
YTIC = FLOAT(I) * BRDRLN / 10. + 1.5
WRITE (11,1121) XTICO, YTIC, XTIC1, YTIC
86 CONTINUE
C
XTICO = BRDRLN
XTIC1 = BRDRLN - 0.15
DO 88 I=1,9
YTIC = FLOAT(I) * BRDRLN / 10. + 1.5
WRITE (11,1121) XTICO, YTIC, XTIC1, YTIC
88 CONTINUE
C
C PUT COORDINATE ANNOTATION ON SIDES OF MAP
C
XP = BRDRLN/2. - 2.6
YP = BRDRLN + 1.5 + 0.35
MAP1 = XANO(1:30)
MAP2 = XANO(31:60)
WRITE (11,1140) XP, YP, 0., 0.1, MAP1, MAP2
YP = BRDRLN + 1.6
DO 92 I=1,5
XP = (FLOAT(IAN(I)) - XMIN) * BRDRLN / (XMAX-XMIN) - 0.8
92 WRITE (11,1130) XP, YP, 0., 0.1, 10, IAN(I)
C
XP = -0.35
YP = BRDRLN/2. + 1.5 - 2.6
MAP1 = YANO(1:30)
MAP2 = YANO(31:60)
WRITE (11,1140) XP, YP, 90., 0.1, MAP1, MAP2
XP = -0.1
DO 94 I=1,5
YP = (FLOAT(IAN(I)) - YMIN) * BRDRLN / (YMAX-YMIN) + 0.7
94 WRITE (11,1130) XP, YP, 90., 0.1, 10, IAN(I)
C
C DRAW A '+' AT THE RANGE CENTER (COORDINATES 0,0)
C
XORG = (0.0 - XMIN) / SCALE + 0.0
YORG = (0.0 - YMIN) / SCALE + 1.5
WRITE (11,1121) XORG, YORG-0.25, XORG, YORG+0.25, 1
XORG-0.25, YORG, XORG+0.25, YORG
C
END OF MAP
C
WRITE (11,1110)
100 CONTINUE
C
RETURN
C
1101 FORMAT ('JOBX ', A70)
1102 FORMAT ('PAGE ', F4.1)
1103 FORMAT ('REF ', I2)
1104 FORMAT ('SIZX ', 2F5.1, 2F5.1, 2(F10.0, IS, F10.0))
1105 FORMAT ('CNTL ', 2F5.2, 2I5, 45X, 2I5)
1106 FORMAT ('CNTL ', 2F10.0, F10.3, 35X, I2)
1107 FORMAT ('BEND')
1108 FORMAT ('LEV ', 2F5.1, 2F5.2, 20X, 3I5)
1109 FORMAT ('BRDR')
1110 FORMAT ('END')
1111 FORMAT ('RESA', 26X, ' 2')
1112 FORMAT ('SAVA')
1120 FORMAT ('SYMB: 0', 4F5.3, ' 30', 15X, A30 /
 1 'ETCS ', 25X, ' 30', 15X, A30 /
 2 'ETCS ', 25X, ' 10', 15X, A10 )
1121 FORMAT ('LINE: 0', 4F5.2, 19X, '1')
1122 FORMAT ('SYMB: 0', 4F5.3, ' 30', 15X, 'SCALE: 1 INCH =', 18,
 1 ' FEET /
 3 'ETCS ', 25X, ' 17', 15X, ' LONG ', A10 )
1130 FORMAT ('SYMB: 0', 4F5.2, 15, 15X, 110)
1140 FORMAT ('SYMB: 0', 4F5.2, ' 30', 15X, A30 /
 1 'ETCS ', 25X, ' 30', 15X, A30)
END
SUBROUTINE CPSSTR (TITLE, TDATE, TTIME, LAT, LONG)

ROUTINE TO OUTPUT GPCP CARDS FOR PLOTTING SUPERSONIC FLIGHT TRACKS

COMMON /GRID/ GROXO, XGS, GROXMX, GROYO, YGS, GROYMX,
1   LIMAXO, LIMAYO, LIMBXO, LIMBYO,
2   LIMAX1, LIMAY1, LIMBX1, LIMBY1

COMMON /STATS/ STATFG, 8 OOM FG, MACHFG, CONTFG,
+   IOOI4VL, MACHVL, CONTVL(5,20), CONTYP(5), WIDTH, FFT,
+   SIGNAT, RAYTRC, SCRPAD, SCRPSF, SCRALL

INTEGER CONTYP
LOGICAL FFT, SIGNAT, RAYTRC, SCRPAD, SCRPSF, SCRALL

DIMENSION IAN(5)

CHARACTER*(*)(*) TITLE, TDATE, TTIME, LAT, LONG
CHARACTER MAP1*30, MAP2*30, MAP3*10
CHARACTER*70 MAPANO, XANO, YANO

REAL MACHVL
DATA IAN / -100000, -50000, 0, 50000, 100000 /
DATA MAPANO /'FLIGHT TRACK SEGMENTS OF SUPERSONIC AIRCRAFT ACTIVITY (MACH > 1)/
DATA XANO /'RANGE X-COORDINATE IN FEET'/
DATA YANO /'RANGE Y-COORDINATE IN FEET'/

FTDELM = 999999.

WRITE (11,1101) TITLE
PWIDTH = AMAX1(8.0, AMIN1(WIDTH, 48.))
WRITE (11,1102) PWIDTH

SCALE = MACHVL / 12.0
ISCALE = IFIX(SCALE + 0.5)
IGS = IFIX(XGS + 0.5)
XMIN = GROXO + XGS/2.0
XMAX = GROXMX - XGS/2.0
YMIN = GROYO + YGS/2.0
YMAX = GROYMX - YGS/2.0
WRITE (11,1104) ISCALE, ISCALE, 0., 1.50, XMIN, IGS, XMAX,
1   YMIN, IGS, YMAX

WRITE (11,1105)

REWIND UNIT 3 WITH FLIGHT TRACK X/Y COORDINATES
REWIND 3

X1 = FTDELM
Y1 = 0.

KEEP TRACK OF PREVIOUS COORDINATES

10 XO = X1
   YO = Y1

READ NEXT COORDINATE PAIR

15 READ (3,3001,END=100) X1, Y1

CONVERT COORDINATES TO PLOTTER INCHES IF THIS
COORDINATE PAIR IS VALID, AND LIMIT TO BOARDER
(GPCP INPUT CARD FIELD LENGTH LIMITATION)

IF (X1 .EQ. FTDELM) GOTO 10
X1 = AMAX1(XMIN, AMIN1(X1, XMAX))
Y1 = AMAX1(YMIN, AMIN1(Y1, YMAX))
X1 = (X1 - XMIN) / SCALE
Y1 = (Y1 - YMIN) / SCALE + 1.5

OUTPUT A LINE SEGMENT ONLY IF CURRENT AND PREVIOUS
COORDINATES ARE VALID AND AT LEAST ONE IS INSIDE BOARDER

IF (XO .EQ. FTDELM) GOTO 10
IF (ABS(X1-XO) .LT. 0.015 .AND. ABS(Y1-YO) .LT. 0.015) GOTO 15
WRTE (11,1106) XO, YO, X1, Y1
GOTO 10

CALCULATE HEIGHT OF 'TITLE' CHARACTERS (MAX=0.25 IN)
AND PLOT TITLE

100 CONTINUE

BRDRLN = (YMAX - YMIN) / SCALE
CHGHT = AMIN1(0.25, (BRDRLN-2.5) / 80.)
WRTE (11,1120) 0.500, 1.000, 0.0, CHGHT, TITLE(1:30), 1
   TITLE(31:60), TITLE(61:70)

CALCULATE HEIGHT OF 'MAP TYPE' AND 'SCALE' CHARACTERS
(MAX=0.20 IN) AND PLOT 2 LINES OF TEXT

CHGHT = AMIN1(0.20, (BRDRLN-3.0) / 80.)
MAP1 = MAPANO(1:30)
MAP2 = MAPANO(31:60)
MAP3 = MAPANO(61:70)
WRTE (11,1122) 1.000, 0.625, 0.0, CHGHT, MAP1, 1
   MAP2, MAP3
WRTE (11,1122) 1.000, 0.250, 0:0, CHGHT, ISCALE, LAT, LONG

DRAW BOX AROUND TITLE BLOCK
WRITE (11,1121) 0.00, 1.50, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.50, 0.00, 0.00, 0.00, 1.50, 0.00, 0.00, 0.00
1
2
3

C
C PUT TIC MARKS ON MAP EDGE

C

YTICO = 1.5
YTICO1 = YTICO + 0.15
DO 82 I=1,9
XTIC = FLOAT(I) * BRDRLN / 10.
WRITE (11,1121) XTIC, YTICO, XTIC, YTIC1
82 CONTINUE

C

YTICO = BRDRLN + 1.5
YTICO1 = YTICO - 0.15
DO 84 I=1,9
XTIC = FLOAT(I) * BRDRLN / 10.
WRITE (11,1121) XTIC, YTICO, XTIC, YTIC1
84 CONTINUE

C

XTICO = 0.0
XTICO1 = XTICO + 0.15
DO 86 I=1,9
YTIC = FLOAT(I) * BRDRLN / 10. + 1.5
WRITE (11,1121) XTICO, YTIC, XTICO1, YTIC
86 CONTINUE

C

XTICO = BRDRLN
XTICO1 = BRDRLN - 0.15
DO 88 I=1,9
YTIC = FLOAT(I) * BRDRLN / 10. + 1.5
WRITE (11,1121) XTICO, YTIC, XTICO1, YTIC
88 CONTINUE

C
C PUT COORDINATE ANNOTATION ON SIDES OF MAP

C

XP = BRDRLN/2. - 2.6
YP = BRDRLN + 1.5 + 0.35
MAP1 = XANO(1:30)
MAP2 = XANO(31:60)
WRITE (11,1140) XP, YP, 0., 0.1, MAP1, MAP2
YP = BRDRLN + 1.6
DO 92 I=1,5
XP = (FLOAT(IAN(I)) / XMAX - XMIN) * BRDRLN / (XMAX-XMIN) * 0.8
WRITE (11,1130) XP, YP, 0., 0.1, 10, IAN(I)
92 CONTINUE

C

XP = -0.35
YP = BRDRLN/2. + 1.5 - 2.6
MAP1 = YANO(1:30)
MAP2 = YANO(31:60)
WRITE (11,1140) XP, YP, 90., 0.1, MAP1, MAP2
XP = 0.1
DO 94 I=1,5
YP = (FLOAT(IAN(I)) - YMIN) * BRORD / (YMAX - YMIN) + 0.7

94 WRITE (11,1130) XP, YP, 90., 0.1, 10, IAN(I)

C DRAW A '+' AT THE RANGE CENTER (COORDINATES 0,0)

C XORG = (0.0 - XMIN) / SCALE + 0.0
YORG = (0.0 - YMIN) / SCALE + 1.5

WRITE (11,1121) XORG, YORG-0.25, XORG, YORG+0.25,
XORG-0.25, YORG, XORG+0.25, YORG

C IF EOF THEN END THE FRAME

C WRITE (11,1107)
RETURN

C

3001 FORMAT (2F10.0)
1101 FORMAT ('JOEX ', A70)
1102 FORMAT ('PAGE ', F4.1)
1104 FORMAT ('SIZX ', 2I5, 2F5.1, 2(F3.0, I5, F10.0))
1105 FORMAT ('BRORD')
1106 FORMAT ('LINE 0', 4F5.2, 19X, '1')
1107 FORMAT ('END')
1120 FORMAT ('SYMB 0', 4F5.3, ' 30', 15X, A30 /
1 'ETCS ', 25X, ' 30', 15X, A30 /
2 'ETCS ', 25X, ' 10', 15X, A10 )
1121 FORMAT ('LINE 0', 4F5.2, 19X, '1 ')
1122 FORMAT ('SYMB 0', 4F5.3, ' 30', 15X, 'SCALE: 1 INCH =', 18,
1 ' FEET /
3 'ETCS ', 25X, ' 17', 15X, ' LONG ', A10 )
1130 FORMAT ('SYMB 0', 4F5.2, 15, 15X, A10)
1140 FORMAT ('SYMB 0', 4F5.2, ' 30', 15X, A30 /
1 'ETCS ', 25X, ' 30', 15X, A30)
END
SUBROUTINE CPMXOP (TITLE, TDATE, TTIME, LAT, LONG)

ROUTINE TO OUTPUT GPCP CARDS FOR PLOTTING SUPERSONIC FLIGHT TRACKS

COMMON /GRID/ GROXO, XGS, GRDXXH, GRDYO, YGS, GRDYMH,
  1 LIMAXO, LIMAYO, LIMBHX, LIMBYO,
  2 LIMAX1, LIMAY1, LIMBHX1, LIMBY1

COMMON /STATS/ STATFG, *OCMFG, MACHFG, CONTFG,
  MACHVL, CONTVL(5,20), CONTYP(5), WIDTH, FFT,
  SIGNAT, RAYTRC, SCRPAD, SCRPSF, SCRALL

INTEGER CONTYP
LOGICAL FFT, SIGNAT, RAYTRC, SCRPAD, SCRPSF, SCRALL

DIMENSION IAN(5)

CHARACTER*(*) TITLE, TDATE, TTIME, LAT,
  LONG, MAPI*30, MAP2*30, MAP3*10

CHARACTER*70 MAPANO, XANO, YANO

REAL CONTVL
DATA IAN / -100000, -50000, 0, 50000, 100000 /
DATA MAPANO / 'I'
DATA XANO / 'RANGE X-COORDINATE IN FEET'/
DATA YANO / 'RANGE Y-COORDINATE IN FEET'/

FTDELM = 999999.

WRITE (11,1101) TITLE
PWIDTH = AMAX1(8.0, AMIN1(WIDTH, 48.))
WRITE (11,1102) PWIDTH

SCALE = CONTVL(1,1) / 12.0
ISCALE = IFIX(SCALE + 0.5)
IGS = IFIX(XGS + 0.5)
XMIN = GRDXXO + XGS/2.0
XMAX = GRDXXH - XGS/2.0
YMIN = GRDYO + YGS/2.0
YMAX = GRDYMH - YGS/2.0
WRITE (11,1104) ISCALE, ISCALE, 0., 1.50, XMIN, IGS, XMAX,
  YMIN, 'GS', YMAX

WRITE (11,1105)

REWIND UNIT 5 WITH FLIGHT TRACK X/Y COORDINATES
REWIND 5

KEEP TRACK OF PREVIOUS COORDINATES

READ NEXT COORDINATE PAIR

15 READ (5,3001,END=100) X1, Y1

CONVERT COORDINATES TO PLOTTER INCHES IF THIS
COORDINATE PAIR IS VALID, AND LIMIT TO BOARDER
(GPCP INPUT CARD FIELD LENGTH LIMITATION)

X1 = AMAX1(XMIN, AMIN1(X1, XMAX))
Y1 = AMAX1(YMIN, AMIN1(Y1, YMAX))
X1 = (X1 - XMIN) / SCALE
Y1 = (Y1 - YMIN) / SCALE + 1.5

OUTPUT A LINE SEGMENT ONLY IF CURRENT AND PREVIOUS
COORDINATES ARE VALID AND AT LEAST ONE IS INSIDE BOARDER

WRITE (11,1121) X1 , Y1 -0.25 , X1 , Y1 +0.25,
1 X1 -0.25, Y1 , X1 +0.25, Y1
GOTO 15

CALCULATE HEIGHT OF 'TITLE' CHARACTERS (MAX=0.25 IN)
AND PLOT TITLE

100 CONTINUE
BRORLN = (YMAX - YMIN) / SCALE
CHGHT = AMIN1(0.25, (BRORLN - 2.5) / 80.)
WRITE (11,1120) 0.500, 1.000, 0.0, CHGHT, TITLE(1:30),
1 TITLE(31:60), TITLE(61:70)

CALCULATE HEIGHT OF 'MAP TYPE' AND 'SCALE' CHARACTERS
(MAX=0.20 IN) AND PLOT 2 LINES OF TEXT

CHGHT = AMIN1(0.20, (BRORLN - 3.0) / 80.)
MAP1 = MAPANO(1:30)
MAP2 = MAPANO(31:60)
MAP3 = MAPANO(61:70)
WRITE (11,1120) 1.000, 0.625, 0.0, CHGHT, MAP1,
1 MAP2, MAP3
WRITE (11,1122) 1.000, 0.250, 0.0, CHGHT, ISCALE, LAT, LONG

DRAW BOX AROUND TITLE BLOCK

WRITE (11,1121) 0.00, 1.50, 0.00, 0.00,
1 0.00, 0.00, BRORLN, 0.00,
2 BRORLN, 0.00, BRORLN, 1.50,
3 BRORLN-1.5, 1.50, BRORLN-1.50, 0.0
PUT TIC MARKS ON MAP EDGE

YTICO = 1.5
YTIC1 = YTICO + 0.15
DO 82 I=1,9
XTIC = FLOAT(I) * BRDRLN / 10.
WRITE (11,1121) XTIC, YTICO, XTIC, YTIC1
82 CONTINUE

YTICO = BRDRLN + 1.5
YTIC1 = YTICO - 0.15
DO 84 I=1,9
XTIC = FLOAT(I) * BRDRLN / 10.
WRITE (11,1121) XTIC, YTICO, XTIC, YTIC1
84 CONTINUE

XTICO = 0.0
XTIC1 = XTICO + 0.15
DO 86 I=1,9
YTIC = FLOAT(I) * BRORLN / 10. + 1.5
WRITE (11,1121) XTICO, YTICO, XTIC1, YTIC
86 CONTINUE

PUT COORDINATE ANNOTATION ON SIDES OF MAP

XP = BRDRLN/2. - 2.6
YP = BRDRLN + 1.5 + 0.35
MAP1 = XANO(1:30)
MAP2 = XANO(31:60)
WRITE (11,1140) XP, YP, 0., 0.1, MAP1, MAP2
YP = BRDRLN + 1.6
DO 92 I=1,5
XP = (FLOAT(IAN(I)) - XMIN) * BRDRLN / (XMAX-XMIN) - 0.8
WRITE (11,1130) XP, VP, 90., 0.1, 10, IAN(I)
92 WRITE (11,1130) XP, VP, 90., 0.1, 10, IAN(I)

DRAW A '+' AT THE RANGE CENTER (COORDINATES 0,0)
XORG = (0.0 - XMIN) / SCALE + 0.0
YORG = (0.0 - YMIN) / SCALE + 1.5
WRITE (11,1121) XORG, YORG-0.25, XORG, YORG+0.25,
1 XORG-0.25, YORG, XORG+0.25, YORG
C C
IF EOF THEN END THE FRAME
C
WRITE (11,1107)
RETURN
C
C
3001 FORMAT (2F10.0)
1101 FORMAT ('JOBX ', A70)
1102 FORMAT ('PAGE ', F4.1)
1104 FORMAT ('SIZX ', 215, 2F5.1, 2(F10.0, 15, F10.0))
1105 FORMAT ('BRDR')
1106 FORMAT ('LINE 0', 4F5.2, 19X, '1')
1107 FORMAT ('END')
1120 FORMAT ('SYMB 0', 4F5.3, ' 30', 15X, A30 /
1 'ETCS ', 25X, ' 30', 15X, A30 /
2 'ETCS ', 25X, ' 30', 15X, A30
1121 FORMAT ('LINE 0', 4F5.2, 19X, '1')
1122 FORMAT ('SYMB 0', 4F5.3, ' 30', 15X, 'SCALE: 1 INCH =', I8,
1 ' FEET '/
3 'ETCS ', 25X, ' 17', 15X, ' LONG ', A10)
1130 FORMAT ('SYMB 0', 4F5.2, 15, 15X, 110)
1140 FORMAT ('SYMB 0', 4F5.2, ' 30', 15X, A30 /
1 'ETCS ', 25X, ' 30', 15X, A30)
END
SUBROUTINE CPBMTR (TITLE, TDATe, TTIME, LAT, LONG)

ROUTINE TO OUTPUT GPCP CARDS FOR PLOTTING SONIC BOOM FLIGHT TRACKS

COMMON /GRID/ GROXO, XGS, GROXMX, GROYO, YGS, GROYMX,
1 LIMAXO, LIMAYO, LIMAXD, LIMAYD,
2 LIMAXI, LIMAYI, LIMAXW, LIMAYW

COMMON /STATS/ STATFG, BOOMFG, MACHFG, CONTFG, BOOMVL,
+ MACHVL, CONTVL(5,20), CONTTYP(5), WIDTH, FFT,
+ SIGNAT, RAYTRC, SCRPAD, SCRPSF, SCRALL

INTEGER CONTTYP
LOGICAL FFT, SIGNAT, RAYTRC, SCRPAD, SCRPSF, SCRALL

DIMENSION IAN(5)

CHARACTER*(*) TITLE, TDATe, TTIME, LAT, LONG
CHARACTER MAP1*30, MAP2*30, MAP3*10
CHARACTER*70 MAPANO, XANO, YANO

REAL BOOMVL
DATA IAN / -100000, -50000, 0, 50000, 100000 /
DATA MAPANO /'FLIGHT TRACK SEGMENTS OF SONIC BOOM PRODUCING AIRCRFT ACTIVITY' />
DATA XANO /'RANGE X COORDINATE IN FEET' /
DATA YANO /'RANGE Y COORDINATE IN FEET' /

FTDEL = 999999.

WRITE (11,1101) TITLE
PWIDTH = AMAX1(8.0, AMIN1(WIDTH, 48.))
WRITE (11,1102) PWIDTH

SCALE = BOOMVL / 12.0
ISCALE = IFIX(SCALE + 0.5)
IGS = IFIX(XGS + 0.5)
XMIN = GRDXO + XGS/2.0
XMAX = GRDXMX - XGS/2.0
YMIN = GRDYO + YGS/2.0
YMAX = GROYMX - YGS/2.0
WRITE (11,1104) ISCALE, ISCALE, 0., 1.50, XMIN, IGS, XMAX,
1 YMIN, IGS, YMAX

WRITE (11,1105)

REWIND UNIT 4 WITH FLIGHT TRACK X/Y COORDINATES

REWIND 4
**C**

```
X1 = FTDELM
Y1 = 0.

C KEEP TRACK OF PREVIOUS COORDINATES

10 XO = X1
    Y0 = Y1

C READ NEXT COORDINATE PAIR

15 READ (4,4001,END=100) X1, Y1

C CONVERT COORDINATES TO PLOTTER INCHES IF THIS
COORDINATE PAIR IS VALID, AND LIMIT TO BOARDER
(GPCP INPUT CARD FIELD LENGTH LIMITATION)

IF (X1 .EQ. FTDELM) GOTO 10
X1 = AMAX1(XMIN, AMIN1(X1, XMAX))
Y1 = AMAX1(YMIN, AMIN1(Y1, YMAX))
X1 = (X1 - XMIN) / SCALE
Y1 = (Y1 - YMIN) / SCALE + 1.5

C OUTPUT A LINE SEGMENT ONLY IF CURRENT AND PREVIOUS
COORDINATES ARE VALID AND AT LEAST ONE IS INSIDE BOARDER

IF (X0 .EQ. FTDELM) GOTO 10
IF (ABS(X1-X0) .LT. 0.015 .AND. ABS(Y1-Y0) .LT. 0.015) GOTO 15
WRITE (11,1106) XO, Y0, X1, Y1
GOTO 10

C CALCULATE HEIGHT OF 'TITLE' CHARACTERS (MAX=0.25 IN) AND PLOT TITLE

100 CONTINUE
BRORLN = (YMAX - YMIN) / SCALE
CHGHT = AMIN1(0.25, (BRORLN-2.5) / 80.)
WRITE (11,1120) 0.500, 1.000, 0.0, CHGHT, TITLE(1:30),
    TITLE(31:60), TITLE(61:70)

C CALCULATE HEIGHT OF 'MAP TYPE' AND 'SCALE' CHARACTERS
(CMAX=0.20 IN) AND PLOT 2 LINES OF TEXT

CHGHT = AMIN1(0.20, (BRORLN-3.0) / 80.)
MAP1 = MAPANO(1:30)
MAP2 = MAPANO(31:60)
MAP3 = MAPANO(61:70)
WRITE (11,1122) 1.000, 0.625, 0.0, CHGHT, MAP1,
    MAP2, MAP3
WRITE (11,1122) 1.000, 0.250, 0.0, CHGHT, ISCALE, LAT, LONG

C DRAW BOX AROUND TITLE BLOCK

WRITE (11,1121) 0.00, 1.50, 0.00, 0.00,
```
1 0.00, 0.00, BRDRLN, 0.00,
2 BRDRLN, 0.00, BRDRLN, 1.50,
3 BRDRLN-1.5, 1.50, BRDRLN-1.50, 0.0

C
C  PUT TIC MARKS ON MAP EDGE
C
YTICO = 1.5
YTIC1 = YTICO - 0.15
DO 82 I=1,9
XTIC = FLOAT(I) * BRDRLN / 10.
WRITE (11,1121) XTIC, YTICO, XTIC, YTIC1
82 CONTINUE
C
YTICO = BRDRLN + 1.5
YTIC1 = YTICO - 0.15
DO 84 I=1,9
XTIC = FLOAT(I) * BRDRLN / 10.
WRITE (11,1121) XTIC, YTICO, XTIC, YTIC1
84 CONTINUE
C
XTICO = 0.0
XTIC1 = XTICO + 0.15
DO 86 I=1,9
YTIC = FLOAT(I) * BRDRLN / 10. + 1.5
WRITE (11,1121) XTICO, YTIC, XTIC1, YTIC
86 CONTINUE
C
XTICO = BRDRLN
XTIC1 = BRDRLN - 0.15
DO 88 I=1,9
YTIC = FLOAT(I) * BRDRLN / 10. + 1.5
WRITE (11,1121) XTICO, YTIC, XTIC1, YTIC
88 CONTINUE
C
C  PUT COORDINATE ANNOTATION ON SIDES OF MAP
C
XP = BRDRLN/2. - 2.6
YP = BRDRLN + 1.5 + 0.35
MAP1 = XANO(1:30)
MAP2 = XANO(31:60)
WRITE (11,1140) XP, YP, 0., 0.1, MAP1, MAP2
YP = BRDRLN + 1.6
DO 92 I=1,5
XP = (FLOAT(IAN(I)) - XMIN) * BRDRLN / (XMAX-XMIN) - 0.8
92 WRITE (11,1130) XP, YP, 0., 0.1, 10, IAN(I)
C
XP = 0.35
YP = BRDRLN/2. + 1.5 - 2.6
MAP1 = YANO(1:30)
MAP2 = YANO(31:60)
WRITE (11,1140) XP, YP, 90., 0.1, MAP1, MAP2
XP = -0.1
DO 94 I=1,5
YP = (FLOAT(IAN(I)) - YMIN) * BRDRLN / (YMAX-YMIN) + 0.7
94 CONTINUE
94 WRITE (11,1130) XP, YP, 90., 0.1, 10, IAM(I)
C
   DRAW A '+' AT THE RANGE CENTER (COORDINATES 0,0)
C
   XORG = (0.0 - XMIN) / SCALE + 0.0
   YORG = (0.0 - YMIN) / SCALE + 1.5
C
   WRITE (11,1121) XORG, YORG-0.25, XORG, YORG-0.25;
   1 XORG-0.25, YORG, XORG+0.25, YORG
C
   IF EOF THEN END THE FRAME
C
   WRITE (11,1107)
   RETURN
C
   4001 FORMAT (2F10.0)
1101 FORMAT ( 'JOBX ', A70)
1102 FORMAT ( 'PAGE ', F4.1)
1104 FORMAT ( 'szx ', 215, 2F5.1, 2(F10.0, I5, F10.0))
1105 FORMAT ( 'BRDR')
1106 FORMAT ( 'LINE 0', 4F5.2, 19X, '1')
1107 FORMAT ('END')
1120 FORMAT ('SYMB 0', 4F5.3, '30', 15X, A30 /
      1 'ETCS ', 25X, '30', 15X, A30 /
      2 'ETCS ', 25X, '10', 15X, A10)
1121 FORMAT ( 'LINE 0', 4F5.2, 19X, '1')
1122 FORMAT ('SYMB 0', 4F5.3, '30', 15X, 'SCALE: 1 INCH = ', I8,
      1 ' FEET /
      3 'ETCS ', 25X, '17', 15X, ' LONG ', A10)
1130 FORMAT ('SYMB 0', 4F5.2, I5, 15X, I10)
1140 FORMAT ('SYMB 0', 4F5.2, '30', 15X, A30 /
      1 'ETCS ', 25X, '30', 15X, A30)
END
BLOCK DATA DICK

COMMON /GRID/ GROXO, XGS, GRDXXM, GROYO, YGS, GRDYYM,
   1  LIMAXO, LIMAYO, LIMBXO, LIMBYO,
   2  LIMAX1, LIMAY1, LIMBX1, LIMBY1

DATA GROXO, XGS, GRDXXM / -126250., 2500., 126250. /
DATA GROYO, YGS, GRDYYM / -126250., 2500., 126250. /
END
**MODULE NAME:** SCHPACK

**MODULE TYPE:** PACKAGE

**OVERVIEW:**

This package is used to perform the process of searching the data tables created during the parse stage. This search is used to find records of subsonic and supersonic flight data records in the library file by finding their location through the use of an index file. This index file is similar to a card catalog.

**INTERFACE:**

- **GETREC (P1, P2, P3)**
  - P1 ::= [INTEGER] pointer to the starting record
  - P2 ::= [INTEGER] total of records starting at P1
  - P3 ::= (LOGICAL) flag signaling no more records left

**INTERNAL SUBROUTINES & FUNCTIONS**

- FILBUF() ; reads on record from the index file into a buffer
- STRMCH() ; returns true if a string matches with table STR
- INTMCH() ; returns true if an int. matches with table INTE

**PROGRAMMER:** BRUCE S. LACEY

**DATE:** 23-OCT-85

**REVISIONS:**

---

**MODULE NAME:** SCHPACK\FILBUF

**MODULE TYPE:** CHARACTER FUNCTION SUBROUTINE

**OVERVIEW:**

This function subroutine is used to read in one record from the index file. If there are no more records then the flag .ENDREC. set true.

**INVOCATION:**

- \( (X = 1) \) FILBUF (P1, P2, P3, P4)
  - P1 ::= [INTEGER] current record number
  - P2 ::= [INTEGER] number of records in index file
P3 := [INTEGER] UNIT NUMBER CORRESPONDING TO INDEX FIL
P4 := [LOGICAL] FLAG SIGNALING THE END OF RECORDS

VARIABLE DICTIONARY:

- ENDREC ; P4
- IDXFIL ; P3
- NUMREC ; P2
- RECNUM ; P1

CALLER MODULES:

- [SUBROUTINE] SCNPACK\GETREC

CALLED MODULES:

- ...NONE...

PROGRAMMER: BRUCE B. LACEY
DATE : 22-OCT-85

REVISIONS :

CHARACTER(*) FUNCTION FILBUF(
   + RECNUM, NUMREC, IDXFIL, ENDREC)

   INTEGER RECNUM, NUMREC, IDXFIL
   LOGICAL ENDREC

   IF (RECNUM.LE.NUMREC) THEN
      RECNUM = RECNUM + 1
      READ(IDXFIL,FMT='(A)',REC=RECNUM) FILBUF
   ELSE
      ENDREC = .TRUE.
   END IF

   RETURN

END
MODULE NAME: SCHPACK\STRMCH

MODULE TYPE: LOGICAL FUNCTION SUBROUTINE

OVERVIEW:

THIS FUNCTION SUBROUTINE IS USED TO SEE IF A STRING PASSED IN MATCHES ANY STRING IN THE CURRENT ROW OF A TABLE PASSED IN. IF 'ALL' IS FOUND THEN THE SEARCH IS CONSIDERED SUCCESSFUL.

INVOCATION:

[X =] STRMCH ( P1, P2, P3, P4, P5 )

P1 ::= [CHARACTER*(*)] STRING TO SEARCH FOR
P2 ::= [INTEGER] REPETITION BEING TESTED
P3 ::= [CHARACTER*(*) (P4,P5)] TABLE TO SEARCH THROUGH
P4 ::= [INTEGER] BOUND FOR THE ROW SIZE
P5 ::= [INTEGER] BOUND FOR THE COLUMN SIZE

VARIABLE DICTIONARY:

COL ; CURRENT COLUMN IN THE SEARCH TABLE
CURREP ; P2
EXTLOP ; SYMBOL REPRESENTING STATEMENT LABEL 200
MXCOL ; P5
MXROW ; P4
SRCFOR ; P1
TABLE ; P3

CALLER MODULES:

[SUBROUTINE] SCHPACK\GETREC

CALLED MODULES:

...NONE...

PROGRAMMER: BRUCE B. LACEY
DATE : 22-OCT-85
REVISIONS :

LOGICAL FUNCTION STRMCH(
+ SRCFOR, CURREP, TABLE, MXROW, MXCOL)

INTEGER CURREP, MXROW, MXCOL, COL, EXTLOP
CHARACTER*(*) SRCFOR, TABLE(MXROW,MXCOL)

ASSIGN 200 TO EXTLOP

STRMCH = .FALSE.
DO 100 COL = 1, MXCOL
   IF ((TABLE(CURREP, COL).EQ.'ALL').OR.
       (TABLE(CURREP, COL).EQ.SRCHFOR)) THEN
     IF (TABLE(CURREP, COL).NE.' ') THEN
       STRMCH = .TRUE.
       GO TO EXTLOP
     ENDIF
  ENDIF
END IF
100    CONTINUE

C EXTLOP:
200    RETURN
END
MODULE NAME: SCHPACK\INTMCH
MODULE TYPE: LOGICAL FUNCTION SUBROUTINE

OVERVIEW:

THIS FUNCTION SUBROUTINE IS USED TO TEST IF AN INTEGER PASSED IN MATCHES AN INTEGER IN THE CURRENT ROW OF THE TABLE PASSED IN. IF 9999 IS FOUND THEN THE TEST IS CONSIDERED SUCCESSFUL.

INVOCATION:

CALL INTMCH (P1, P2, P3, P4, P5, P6)

P1 := [INTEGER] VALUE TO BE TESTED
P2 := [INTEGER] ROW CURRENTLY BEING TESTED
P3 := [INTEGER(P5,P6)] TABLE FOR LOWER BOUND
P4 := [INTEGER(P5,P6)] TABLE FOR UPPER BOUND
P5 := [INTEGER] LOWER BOUND FOR P3 AND P4
P6 := [INTEGER] UPPER BOUND FOR P3 AND P4

VARIABLE DICTIONARY:

COL ; CURRENT COLUMN IN SEARCH TABLES
CURREP ; P2
ETABLE ; P3
EXTLOP ; SYMBOL REPRESENTING STATEMENT LABEL 200
MXCOL ; P6
MXROW ; P5
SRCFOR ; P1
STABLE ; P4

CALLER MODULES:

[SUBROUTINE] SCHPACK\GETREC

CALLED MODULES:

...NONE...

PROGRAMMER: BRUCE B. LACEY
DATE : 22-OCT-85
REVISION :

LOGICAL FUNCTION INTMCH(
 + SRCFOR, CURREP, STABLE, ETABLE, MXROW, MXCOL)

INTEGER SRCFOR, CURREP, MXROW, MXCOL, EXTLOP
INTEGER COL, LOOP
INTEGER STABLE(MXROW,MXCOL), ETABLE(MXROW,MXCOL)

ASSIGN 100 TO LOOP
ASSIGN 200 TO EXTL0P

INTMCH = .FALSE.
IF (STABLE(CURREP,1).EQ.9999) THEN
  INTMCH = .TRUE.
ELSE
  COL = 1
  C LOOP:
  100 IF((SRCFOR.GE.STABLE(CURREP,COL)).AND.
          (SRCFOR.LE.ETABLE(CURREP,COL))) THEN
          INTMCH = .TRUE.
          GO TO EXTL0P
  END IF
  COL = COL + 1
  IF (COL.LE.MXC0L) GO TO LOOP
C EXTL0P:
  200 END IF
  RETURN
END
* MODULE NAME: SCHPACK\GETREC
* MODULE TYPE: SUBROUTINE
*
* OVERVIEW:
*
* THIS SUBROUTINE IS USED TO SEARCH THE INDEX FILE ACCORDING
* THE USER SPECIFICATIONS STORED DURING THE PARSE STAGE. WHEN INVO
* THIS SUBROUTINE WILL READ RECORDS FROM THE INDEX FILE UNTIL A MAT
* IS FOUND. WHEN A MATCH IS FOUND THE SUBROUTINE WILL RETURN THE
* STARTING RECORD NUMBER AND THE NUMBER OF RECORDS OCCURRING AFTER
* THE STARTING RECORD. IF A MATCH IS NOT FOUND THEN THE END OF REC
* FLAG .ENDREC. IS SET TRUE.
*
* INVOCATION:
*
* [CALL] GETREC ( P1, P2, P3, P4 )
*
* P1 := [INTEGER] STARTING RECORD NUMBER
* P2 := [INTEGER] COUNT OF RECORDS FOLLOWING P1
* P3 := [INTEGER] COUNT OF SUPersonic RECORDS
* P4 := [LOGICAL] FLAG SIGNALING THE END OF RECORDS
*
* VARIABLE DICTIONARY:
*
* ARCRFT ; TABLE CONTAINING AIRCRAFT TYPES
* CURREC ; THE CURRENT RECORD NUMBER FROM FILE 'INDEX'
* ENDATE ; TABLE CONTAINING THE END DATES
* ENREC ; P3
* ENTIME ; TABLE CONTAINING THE END TIMES
* IDXFIL ; UNIT NUMBER FOR THE INDEX FILE
* INTDAT ; [INTEGER] REPRESENTING YYMDM DATE
* LOOP ; SYMBOL REPRESENTING STATEMENT LABEL 1
* MSSNS ; TABLE CONTAINING THE MISSION/EXERCISE NAMES
* MXDATE ; MAXIMUM NUMBER OF DATE ALLOWED
* MXMSSN ; MAXIMUM NUMBER OF MISSION ALLOWED
* MXPLNS ; MAXIMUM NUMBER OF AIRCRAFT ALLOWED
* MXREPS ; MAXIMUM NUMBER OF REPETITIONS OF SITE CARDS ALLOWE
* MXSITE ; MAXIMUM NUMBER OF SITES LOCATIONS ALLOWED
* MXTIME ; MAXIMUM NUMBER OF START/END TIMES ALLOWED
* NUMREC ; NUMBER OF RECORDS IN THE INDEX FILE
* NUMREP ; NUMBER OF REPETITIONS STORED DURING PARSE
* RECBUF ; BUFFER TO HOLD ONE RECORD FROM FILE 'INDEX'
* RECTOT ; P2
* SITES ; TABLE CONTAINING THE SITE LOCATIONS
* STREC ; P1
* STTIME ; TABLE CONTAINING THE STARTING TIMES
* TAILNM ; TABLE CONTAINING THE AIRCRAFT TAIL NUMBERS
* TBLIDX ; CURRENT REPETITION BEING COMPARE
* TIME1 ; STARTING TIME FROM RECORD IN 'INDEX'
* TIME2 ; ENDING TIME FROM RECORD IN 'INDEX'
CALLER MODULES:
MAIN DRIVER ROUTINE

CALLED MODULES:

SUBROUTINE FUNCTION] SCHPACK\FILBUF()
SUBROUTINE FUNCTION] SCHPACK\SCHMCH()
SUBROUTINE FUNCTION] SCHPACK\INTMCH()

PROGRAMMER: BRUCE B. LACEY
DATE: 22-OCT-85

REVISIONS:

SUBROUTINE GETREC(STREC, RECTOT, SUPREC, ENDREC, IT)

EXTERNAL STRMC4, INTMCH, FILBUF
PARAMETER(MXDATE=10, MXMSSN=10, MXPLNS=10,
MXREPS=5, MXSITE=20, MXTIME=10)
COMMON /CHRTABS/ ARCRFT, MSSNS, SITES, TAILNM
COMMON /INTTABS/ ENDATE, ENTIME, STDATE, STTIME, NUMREP
INTEGER ENDATE(MXREPS, MXDATE)
INTEGER ENTIME(MXREPS, MXTIME)
INTEGER STDATE(MXREPS, MXDATE)
INTEGER STTIME(MXREPS, MXTIME)
INTEGER NUMREP, STREC, RECTOT, CURREC, INTDAT, LOOP
INTEGER TIME1, TIME2, IDXFIL,
NUMREC, TBLIDX, SUPREC
CHARACTER*6 ARCRFT(MXREPS, MXPLNS)
CHARACTER*16 MSSNS(MXREPS, MXMSSN)
CHARACTER*10 SITES(MXREPS, MXSITE)
CHARACTER*8 TAILNM(MXREPS, MXPLNS)
CHARACTER*98 FILBUF, RECSUF
LOGICAL ENDREC, STRMCH, INTMCH

SAVE CURREC
DATA IDXFIL /1/
DATA CURREC /0/
ASSIGN 1 TO LOOP

IF (CURREC.LE.1) THEN
   READ THE HEADER RECORD TO GET THE NUMBER OF RECORDS.
   CURREC = CURREC + 1
   READ(IDXFIL,FMT='(I6)',REC=CURREC) NUMREC
   ENDREC = .FALSE.
END IF
TBLIDX = 0

C::
  GET THE STARTING RECORD
  RECBUF = FILBUF(CURREC,NUMREC,IDXFIL,ENDREC)

C LOOP:
  1   IF (ENDREC.EQV..TRUE.) RETURN
       TBLIDX = TBLIDX + 1
   IF (TBLIDX.GT.NUMREP) THEN
     RECBUF = FILBUF(CURREC,NUMREC,IDXFIL,ENDREC)
     TBLIDX = 1
     END IF

C::
  CHECK IF SITE LOCATIONS MATCH
  IF (STRMCH(RECBUF(27:36),TBLIDX,SITES,MXREPS,MXSITES)
       .EQV..TRUE.) THEN
    C::
      CHECK IF THE MISSION NAMES MATCH
      IF (STRMCH(RECBUF(1:16),TBLIDX,MSSNS,MXREPS,MXMSSN)
           .EQV..TRUE.) THEN
        C::
          FIRST CONVERT THE DATE TO YYMMDD INTEGER
          READ(RECBUF(17:18),FMT=1'(I2)' ) INTOAT
          INTOAT = INTOAT * 100
          READ(RECBUF(20:21),FMT=1'I2' ) I
          INTOAT = INTOAT + I
          READ(RECBUF(23:24),FMT=1'I2' ) I
          INTOAT = INTOAT + (I * 10000)
      C::
        IF (INTMCH(INTDAT,TBLIDX,STDAT,ENDAT,MXREPS,MXDATE)
             .EQV..TRUE.) THEN
          C::
            CHECK IF THE TIME INTERVALS CORRESPOND
            READ(RECBUF(37:40),FMT=1'(I4)' ) TIME1
            READ(RECBUF(45:48),FMT=1'I4' ) TIME2
            IF ((INTMCH(TIME1,TBLIDX,STTIME,ENTIME,
                         MXREPS,MXTIME).EQV..TRUE.).OR.
                 (INTMCH(TIME2,TBLIDX,STTIME,ENTIME,
                         MXREPS,MXTIME).EQV..TRUE.)) THEN
              C::
                CHECK IF AIRCRAFT TYPES MATCH
                IF (STRMCH(RECBUF(55:60),TBLIDX,ARCRFT.,
                           MXREPS,MXPLNS).EQV..TRUE.) THEN
                  C::
                    CHECK IF THE AIRCRAFT TAIL NUMBERS MATCH
                    IF (STRMCH(RECBUF(61:68),TBLIDX,TAILNM,
                                 MXREPS,MXPLNS).EQV..TRUE.) THEN
                      C::
                        WE HAVE A SUCCESSFULL MATCH
                        READ(RECBUF(69:78),FMT=1'(I10)' ) STREC
                        READ(RECBUF(79:88),FMT=1'(I10)' ) RECTOT
                        READ(RECBUF(89:98),FMT=1'(I10)' ) SUPREC
I1 = CURREC
RETURN
ELSE
    GO TO LOOP
END IF
ELSE
    GO TO LOOP
END IF
ELSE
    GO TO LOOP
END IF
ELSE
    GO TO LOOP
END IF
ELSE
    GO TO LOOP
END IF
ELSE
    GO TO LOOP
END IF
ELSE
    GO TO LOOP
END IF
ELSE
    GO TO LOOP
END IF

*=================================================================
*...........> END SCHPACK <<<<<<------------------------
*=================================================================

END
MODULE NAME: SCHPACK\GETOMC

MODULE TYPE: SUBROUTINE

OVERVIEW:

This subroutine is used to search the index file according to the user specifications stored during the parse stage. When invoked, this subroutine will read records from the index file until a match is found. When a match is found, the subroutine will return the starting record number and the number of records occurring after the starting record. If a match is not found, then the END REC flag .ENDREC is set true.

INVOCATION:

[CALL] GETINX ( P1, P2, P3, P4 )

P1 ::= INTEGER starting record number
P2 ::= INTEGER count of records following P1
P3 ::= INTEGER count of supersonic records
P4 ::= LOGICAL flag signaling the end of records

VARIABLE DICTIONARY:

ARCRFT; table containing aircraft types
CURREC; the current record number from file 'INDEX'
ENDATE; table containing the end dates
ENDREC; P3
ENTIME; table containing the end times
IDXFIL; unit number for the index file
INTDAT; integer representing YYMMDD date
LOOP; symbol representing statement label 1
MSSNS; table containing the mission/exercise names
MAXDATE; maximum number of date allowed
MAXSSN; maximum number of mission allowed
MAXPLNS; maximum number of aircraft allowed
MAXREPS; maximum number of repetitions of site cards allowed
MAXSITE; maximum number of sites allowed
MXTIME; maximum number of start/end times allowed
NOPREC; number of overpressure records
NUMREC; number of records in the index file
NUMREP; number of repetitions stored during parse
OPR; flag .TRUE. if the track contains overpressure rec
RECBUF; buffer to hold one record from file 'INDEX'
RECTOT; P2
SITES; table containing the site locations
STREC; P1
STTIME; table containing the starting times
SUPREC; starting overpressure record
TAILNM; table containing the aircraft tail numbers
TBLIDX; current repetition being compared
- TIME1 ; STARTING TIME FROM RECORD IN 'INDEX'
- TIME2 ; ENDING TIME FROM RECORD IN 'INDEX'

CALLER MODULES:

MAIN DRIVER ROUTINE

CALLED MODULES:

- [SUBROUTINE FUNCTION] SCHPACK\FILBUF()
- [SUBROUTINE FUNCTION] SCHPACK\SCHMCH()
- [SUBROUTINE FUNCTION] SCHPACK\INTMCH()

PROGRAMMER: BRUCE B. LACEY

DATE : 22-OCT-85

REVISIONS:

SUBROUTINE GETINX(STREC, RECTOT, SUPREC, ENDREC, 11, NOPREC, OPR)

C EXTERNAL STRMCH, INTMCH, FILBUF

PARAMETER(MXDATE=10, MXMSSN=10, MXPLNS=10,
+ MXREPS=5, MXSITE=20, MXTIME=10)

COMMON /CHRTABS/ ARCRFT, MSSNS, SITES, TAILNM
COMMON /INTTABS/ ENDATE, ENTIME, STDATE, STTIME, NUMREP

INTEGER ENDATE(MXREPS, MXDATE)
INTEGER ENTIME(MXREPS, MXTIME)
INTEGER STDATE(MXREPS, MXDATE)
INTEGER STTIME(MXREPS, MXTIME)
INTEGER NUMREP, STREC, RECTOT, CURREC, INTDAT, LOOP
INTEGER TIME1, TIME2, IDXFIL, NUMREC, TBLIDX, SUPREC

CHARACTER*6 ARCRFT(MXREPS, MXPLNS)
CHARACTER*16 MSSNS(MXREPS, MXMSSN)
CHARACTER*10 SITES(MXREPS, MXSITE)
CHARACTER*8 TAILNM(MXREPS, MXPLNS)
CHARACTER*110 FILBUF, RECBUF

LOGICAL ENDREC, STRMCH, INTMCH, OPR

SAVE CURREC

DATA CURREC /0/
DATA IDXFIL /51/
ASSIGN 1 TO LOOP

IF (CURREC.LE.1) THEN
  READ THE HEADER RECORD TO GET THE NUMBER OF RECORDS.
  CURREC = CURREC + 1
  READ(IDXFIL,FMT=’(4X,16)’,REC=CURREC) NUMREC

ENDREC = .FALSE.
NUMREC = NUMREC - 1
END IF

TBLIDX = 0

C--

GET THE STARTING RECORD
RECBUF = FILBUF(CURREC,NUMREC,IDXFIL,ENDREC)

C LOOP:
1 IF (ENDREC.EQV..TRUE.) RETURN
TBLIDX = TBLIDX + 1
IF (TBLIDX.GT.NUMREP) THEN
C--
READ IN ANOTHER RECORD FOR TESTING
RECBUF = FILBUF(CURREC,NUMREC,IDXFIL,ENDREC)
TBLIDX = 1
END IF

C--
CHECK IF SITE LOCATIONS MATCH
IF (TBLIDX.EQV..TRUE.) THEN
C--
CHECK IF THE MISSION NAMES MATCH
IF (TBLIDX.EQV..TRUE.) THEN
C--
FIRST CONVERT THE DATE TO YYMMDD INTEGER
READ(RECBUF(17:18),FMT*1(12)) INTDAT
INTDAT = INTDAT * 100
READ(RECBUF(20:21),FMT*1(12)) INTDAT + INTDAT * 10000
IF (INTMCH(INTDAT,TBLIDX,STDATE,ENDATE,MXREPS,MXDATE) .EQV..TRUE.) THEN
C--
CHECK IF THE TIME INTERVALS CORRESPOND.
READ(RECBUF(37:40),FMT*1(14)) TIME1
READ(RECBUF(45:48),FMT*1(14)) TIME2
IF ((INTMCH(TIME1,TBLIDX,STTIME,ENTIME,MXREPS,MXTIME).EQV..TRUE.) .OR. (INTMCH(TIME2,TBLIDX,STTIME,ENTIME,MXREPS,MXTIME).EQV..TRUE.)) THEN
C--
CHECK IF AIRCRAFT TYPES MATCH.
IF (TBLIDX.EQV..TRUE.) THEN
C--
CHECK IF THE AIRCRAFT TAIL NUMBERS MATCH
IF (TBLIDX.EQV..TRUE.) THEN
C:

WE HAVE A SUCCESSFUL MATCH
READ(RECBUF(69:78),FMT=*(1D0)) STREC
READ(RECBUF(79:88),FMT=*(1D0)) RECTOT
READ(RECBUF(89:98),FMT=*(1D0)) SUPREC
READ(RECBUF(99:108),FMT=*(1D0)) NOPREC
READ(RECBUF(109:109),FMT='(L1)') OPR
I1 = CURREC
RETURN
ELSE
   GO TO LOOP
END IF
ELSE
   GO TO LOOP
END IF
ELSE
   GO TO LOOP
END IF
ELSE
   GO TO LOOP
END IF
ELSE
   GO TO LOOP
END IF
ELSE
   GO TO LOOP
END IF
ELSE
   GO TO LOOP
END IF
END
PROGRAM STOREC(OUTPUT, TAPE6=OUTPUT, TAPE11=0)
.

THIS SUBROUTINE IS DESIGNED TO STORE NEEDED VARIABLES IN A TEMPORARY FILE SO THE PLOTING/RAYTRACING CAN BE RUN AS A TWO STEP PROCESS.

COMMON /CHRTASS/ ARCRFT, MSSNS, SITES, TAILNM
COMMON /INTTABS/ ENDATE, ENTIME, STDATE, STTIME, NUMREP
COMMON /STATS/ STATFG, BOOMFG, MACHFG, CONTFG, BOOMVL, 1 MACHVL, CONTVL, CONTYP, WIDTH, FFT, 2 SIGMAT, RAYTRC, SCRPAO, SCRPSF, SCRALL

INTEGER ENDATE(5,10), ENTIME(5,10), STDATE(5,10)
INTEGER STTIME(5,10), CONTYP(5), NUMREP

REAL CONTVL(5,20), BOOMVL, MACHVL, WIDTH

LOGICAL STATFG, BOOMFG, MACHFG, CONTFG, SIGMAT, RAYTRC, 1 SCRPAO, SCRPSF, SCRALL, GPCPFL, FFT, GPCPMH, GPCPBM

CHARACTER*70 TITLE
CHARACTER*6 ARCRFT(5,10)
CHARACTER*16 MSSNS(5,10)
CHARACTER*10 SITES(5,20)
CHARACTER*8 TAILNM(5,10)

OPEN(76,FILE='HOLDVAR',STATUS='UNKNOWN')

REWIND(76)
READ(76,FMT='(A)') TITLE
DO 10 I = 1, 5
READ(76,FMT='(A)') (ARCRFT(I,J),J=1,10)
READ(76,FMT='(A)') (MSSNS(I,J),J=1,10)
READ(76,FMT='(A)') (SITES(I,J),J=1,20)
READ(76,FMT='(A)') (TAILNM(I,J),J=1,10)
READ(76,FMT='(A)') (ENTIME(I,J),J=1,10)
READ(76,FMT='(A)') (STDATE(I,J),J=1,10)
READ(76,FMT='(A)') (STTIME(I,J),J=1,10)
READ(76,FMT='(A)') (CONTYP(I,J),J=1,10)
READ(76,FMT='(F20.4)') (CONTVL(I,J),J=1,20)
CONTINUE
READ(76,FMT='(SL)') STATFG, BOOMFG, MACHFG, CONTFG, FFT
READ(76,FMT='(SL)') SIGMAT, RAYTRC, SCRPAO, SCRPSF, SCRALL, 1 GPCPFL
READ(76,FMT='(3F20.4)') BOOMVL, MACHVL, WIDTH
READ(76,FMT='(2L1,1B)') GPCPMH, GPCPBM, NUMREP

C CALL THE DRIVER
CALL PLOTDR(TITLE, GPCPFL, GPCPMH, GPCPBM)
CLOSE(11)
CLOSE(33)
CLOSE(34)
CLOSE(51)
CLOSE(52)
CLOSE(78)
CLOSE(35)
CLOSE(3)
CLOSE(4)
STOP
END