Naval Research Laboratory
Washington, DC 20375-5000

NRL Memorandum Report 6377

A User's Guide to Raytrace

DR. MICHAEL H. REILLY

Ionospheric Effects Branch
E. O. Hulburt Center for Space Research
Space Science Division

DR. ERIC L. STROBEL

Interferometrics, Inc.
8150 Leesburg Pike, Suite 1400
Vienna, VA 22180

February 8, 1989

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT: Approved for public release; distribution unlimited.
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): NRL Memorandum Report 6377	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: Naval Research Laboratory; 6b. OFFICE SYMBOL (If applicable): Code 4180	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code): Washington, DC 20375-5000	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING / SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO; PROJECT NO; TASK NO; WORK UNIT ACCESSION NO

11. TITLE (Include Security Classification)

A User's Guide to Raytrace

12. PERSONAL AUTHOR(S)

Reilly, M.H. and Strobel,* E.L.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Memorandum	FROM 10/87 TO 4/88	1989 February 8	295

16. SUPPLEMENTARY NOTATION

*Interferometrics, Inc. 8150 Leesburg Pike, Suite 1400 Vienra, VA 22180

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Ionosphere; Simulation; Raytrace; Three-dimensional; Raytracing

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

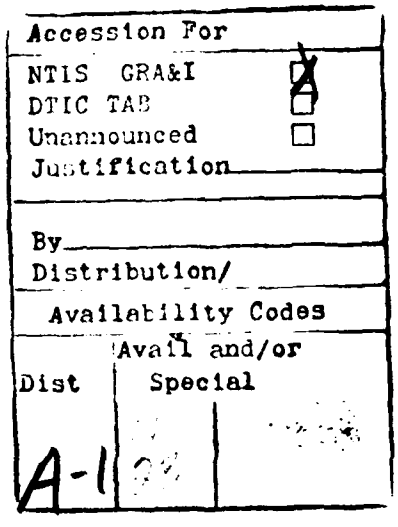
The operation of the RAYTRACE (v4.3) software for ionospheric raytracing is explained. First, the installation and setup of the program are discussed. Instructions for routine operation are given, followed by several tutorial examples. Some suggestions for regular usage are given. The final section and the appendices contain useful reference material, including the source code listing.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT: ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL: Dr. Michael H. Reilly	22b. TELEPHONE (Include Area Code): (202) 767-2891; 22c. OFFICE SYMBOL: Code 4180.2

DD Form 1473, JUN 86 — Previous editions are obsolete. — SECURITY CLASSIFICATION OF THIS PAGE

S/N 0102-LF-014-6603

CONTENTS



A USER'S GUIDE TO RAYTRACE

PART ONE:
INTRODUCTION

Manuscript approved April 22, 1988.

OVERVIEW

This manual is a user's guide to the operation of the program RAYTRACE (Version 4.3) and also covers related topics. As a user's guide, this document is meant to assist with the day to day usage of RAYTRACE. The greater theoretical detail that many researchers may wish to see is contained in a paper that is being published in Radio Science [1]. This user's guide is divided into two sections plus the appendices. The first section is introductory in nature and includes some background material, a 'packing list' of the files, and instructions for installation and setup. The second section is devoted to educating the user in the day to day operation of RAYTRACE. The appendices incorporate a variety of useful information.

The authors of the program RAYTRACE are Dr. Michael H. Reilly of the Naval Research Laboratory, and Dr. Eric L. Strobel of Interferometrics Inc. Dr. Reilly may be reached at the following address:

Dr. Michael H. Reilly
Code 4180.2
Naval Research Laboratory
Washington DC 20375

Phone: (202) 767-2891.

Dr. Strobel may be reached at:

Dr. Eric L. Strobel
Interferometrics Inc.
8150 Leesburg Pike
Vienna VA 22180

Phone: (703) 790-8500.

Summary of capabilities

The program RAYTRACE has a number of distinctive capabilities. It performs fully three-dimensional HF/VHF raytracing through a climatological ionosphere. (Magnetic field effects are, however, not currently implemented.) Single or multiple rays may be traced during a single run, allowing the user greater flexibility in modeling. The

size of the increments used in the raypath calculation is user adjustable so that appropriate compromises may be made between execution time and accuracy. The program can trace rays between points at different altitudes. Cutoff criteria may be set to stop the calculation at a particular range or altitude.

RAYTRACE calculates a number of important quantities about the ray. The change in signal intensity due to distance and ionospheric focusing or defocusing is calculated. The group and phase path length values are both calculated. The location, as well as information on the direction of the ray, are calculated for the end point of the ray and any intervening earth impact points (collectively these are called the data points for the ray).

Technical background

The RAYTRACE program is an implementation of a technique that is much more fully discussed in a paper that will be published in Radio Science [1]. A brief synopsis of that paper's discussion concerning program details is presented here. This section may be skipped with no loss of continuity.

The RAYTRACE algorithm utilizes a set of raypath differential equations that is separable under some approximation assumptions. These assumptions involve expanding the square of the index of refraction in a Taylor series in a local set of Cartesian coordinates and then truncating the series. If the coefficients of this expansion are known, then the raypath differential equations are integrable. The coefficients corresponding to the terms in the horizontal direction are obtained by spatial interpolation of sets of the coefficients defined on a latitude and longitude grid. Those in the vertical direction are obtained by using the functional form assumed by whatever particular ionospheric model the raytracing program has been customized to work with.

Given the truncation of the Taylor series, the equations for the raypath are only valid for a limited distance from the origin of the Taylor expansion. For this reason, the modeled raypath is built up incrementally. The raypath equations, then, give the coordinates of the end point of a ray increment in the local Cartesian system.

The program utilizes an ionospheric specification on a latitude-longitude grid of points. The ray is constructed by successively calculating the coefficients of the series, evaluating the raypath equations, and updating the coordinates and other needed values in preparation for the next increment. The process continues until some cutoff criterion is met.

This raytracing algorithm is quite compact, as it utilizes analytical integration of the raypath differential equations, rather than numerical integration. The storage requirements are smaller than if numerical integration was used because a particular (perhaps piecewise) functional form for the vertical structure of the ionosphere is assumed. This exacts a price, however. If a change is made to the underlying ionospheric model, or if a different model is to be used, portions of the RAYTRACE code need to be changed. The current implementation of RAYTRACE corresponds to that detailed in the paper [1], so the model to which it is currently matched is the RADAR-C ionospheric model [2]. The vertical profile from this model and the number of parameters necessary to specify the profile determine the calculation of the expansion coefficients and the resulting parameters and derivatives.

Operating requirements

The current implementation of RAYTRACE is written in Fortran 77. Subject to very minor compiler dependencies, which are discussed in later sections, the program will run on most computers having Fortran 77 compilers. The total amount of disk storage recommended for use with RAYTRACE is about one megabyte. Given the extra space needed for the compiler and system files, use of a hard disk is virtually essential for microcomputer users of RAYTRACE. If necessary, it is possible to place the executable version of the program, the associated files, and a data file on a floppy disk. A minimum of 512 K of memory is recommended, with 640 K or greater suggested. The presence of a math coprocessor is essentially a requirement for microcomputer users.

An additional requirement that many users will face is a method of generating input to RAYTRACE. This will entail a program that will produce a file, with a format given in Appendix A, from either an iono-

spheric model or experimental data. These data may be entered by hand, but this method of data input to RAYTRACE is only practical for relatively small amounts of data.

Design philosophy

The RAYTRACE program has been developed to meet specific user needs, however the development process has been geared toward maintaining the greatest degree of generality in operation consistent with the original specific needs. For this reason, the program is user driven to the greatest extent possible. This provides some capability to do calculations in a "What if ... ?" manner if desired. The current version of the program represents an attempt to strike a balance between modularity, maintainability of the code, and compactness of both memory space and user interface.

Part of the basic outlook taken while developing RAYTRACE is that this program will most certainly be put to unanticipated uses. The program is therefore set up to be as straightforward as possible (at least as simulation programs like this go) to modify for such uses. Some ideas on customization are given later. Such customization is bound to occur as RAYTRACE becomes more widely disseminated. The program will grow, but some questions will inevitably come back to the original authors of RAYTRACE. It is therefore strongly recommended that any variations, changes, or additions to RAYTRACE be commented as profusely as possible in the source code. Additionally, the program's authors (and, in fact, the research community at large) ought to be kept abreast of such alterations when appropriate. If this occurs, it will not only aid RAYTRACE's authors in responding to questions, but will facilitate the evolutionary improvement of the program as a general research tool.

Conventions used

It is assumed that the user is familiar with the operating system for the machine that is used. Additionally, the user ought to be knowledgeable about the operation of the compiler and editor to be used. Filenames are given in all capital letters (FILE.EXT), with a three letter extension. The total length of filenames is not to exceed ten characters. Prompts are generally displayed in double quotes (" "), with responses given in single quotes (' '). The quotes are not to be typed.

RAYTRACE is currently implemented on a DOS system and a VAX/VMS system, with subtle differences between the two implementations. This manual is written so that those with DOS systems see as little as possible about VMS systems, and vice versa. Although this approach results in some overall repetition, the user may skip over sections not dealing with his/her system without fear of missing key material.

FILES

What's included

The RAYTRACE package is best transported as source code which may then be copied onto a system meeting the requirements stated above for production of the executable version. What follows are brief descriptions of the RAYTRACE source code files and the various supporting files.

The source for the RAYTRACE program itself is broken into six files. The file RAYTRA4.FOR is essentially the user interface shell within which the actual raytracing routines are imbedded. The handling of access of data files is also within this set of routines. RRAYSB.FOR is the actual meat of the raytracing. This contains the routine which implements the raytracing algorithm. The ionospheric parameters and gradients necessary for the raytracing are calculated by routines in RIONO.FOR. Routines to support the calculation of phase path length information are contained in RPHASE.FOR. The boundaries between segments of the vertical profile of the ionosphere will, in general, be tilted with respect to local vertical. The routines to take this tilting into account are contained in RTILT.FOR. Various other supporting routines are located in the remaining source file, RMISC.FOR.

When RAYTRACE is run, it looks to a few text files which contain information on the make up of the menus that the user will see. These are the files with the .MEN suffix. MASTER.MEN, as the name would indicate, is the master file that RAYTRACE looks for to determine what other menu files will be needed. MAIN.MEN contains the information on the construction of the main menu that the user interacts with during a raytracing run. LAUNCH.MEN and ELEV.MEN determine the menus by which the user will input parameters used in the raytracing.

One more file is included. READER2.FOR is the source code for a brief program that may be used to display the results of a raytracing. Since analysis needs will vary from user to user, this program is meant to provide the basis for more sophisticated programs to analyze raytracing results.

Table 1

Table of files

Filename	Size (bytes)
RAYTRA4.FOR	50248
RIONO.FOR	63071
RMISC.FOR	42064
RPHASE.FOR	18811
RRAYSB.FOR	27181
RTILT.FOR	29584
ELEV.MEN	237
LAUNCH.MEN	237
MAIN.MEN	163
MASTER.MEN	47
READER2.FOR	3477

Formats available

RAYTRACE has been, to date, primarily used on a VAX under the VMS operating system, and on an AT-compatible microcomputer running MS-DOS. The program can be furnished in formats compatible with either of these two environments. It may also be possible to furnish it in other formats upon request.

PROGRAM SETUP

VMS environment

Setup and installation of RAYTRACE in the VMS environment will be made more convenient by appropriate planning of directories. What follows is one possible directory setup that has been of use while developing the program. Individual user needs and pre-existing directories will dictate the directory structuring for each site. The pri-

mary point is that the user of RAYTRACE will make usage easier by planning ahead.

The directory scheme used during development and testing of RAYTRACE involved the use of three parallel directories. A working directory was set up containing the source code. Editing, compiling, and linking are performed within this directory. When significant amounts of debugging are anticipated, the .MEN files and a trial input file are moved into the working directory, although by and large such duplication of files is avoided in order to save space.

A simulation directory is maintained for day-to-day usage of RAYTRACE. This directory holds .DAT (input) and the .MEN files, as well as the executable version of RAYTRACE. Numerous input files may be maintained by naming them in a mnemonic fashion.

The simulation directory is maintained in an uncluttered state by having a results directory to which the results of raytracings may be moved. This is particularly important because RAYTRACE simply uses generic filenames when it writes out results. If many runs are to be performed, it is to the user's advantage to rename and move the result files in order to keep them straight.

An additional directory that may be of help to some users is a versions directory. This may be a subdirectory of the working directory, to which older or different versions of the code are moved. Again, these files should be renamed in order to distinguish them at a later date. An advantage of such a subdirectory is keeping the working directory uncluttered.

The source code may now be copied into the desired directory. If the source is uploaded to the VMS system from a DOS system, some things may need to be edited before RAYTRACE will successfully compile. Some DOS-based Fortran compilers need markers at the beginning of subroutines so that subroutines will start on a new page when the compiler is asked to produce a listing file. These markers need to be removed.

Also, different compilers have different means of suppressing generation of a new line after a WRITE statement to the screen. This means that three lines of the subroutine KEYBRD need to be changed from one development environment to the next. This subroutine is located in the file RAYTRA4.FOR. Edit this file and go to the subroutine KEYBRD. There is a comment block below the variable declarations which describes the necessary changes.

RAYTRACE is now ready to be compiled and linked. If default naming is to be used, then RAYTRA4 should be listed first in the link command. The executable version may now be copied to the raytracing directory. The .OBJ files should be left intact. This way, when changes are made to one source file, only that one file need be recompiled and all the object files may be linked. If disk space is a consideration, the executable may certainly be deleted from the working directory.

READER2.FOR should also be compiled and linked. The executable may be copied to the results directory. The object and executable files left in the working directory may be deleted to conserve disk space.

The following files should now exist. The working directory should contain a complete set of both source and object files for RAYTRACE, as well as the source for READER2.FOR. The executable versions of RAYTRACE (default name: RAYTRA4.EXE) and READER2 should be in the locations that the user has chosen as appropriate. The .MEN files should be located in the same directory as the RAYTRACE executable.

The process of compiling, linking, and moving these files will be made considerably easier by making use of the capability of VMS to use .COM files. This is especially true if modifications to the programs are intended.

DOS environment

Setup and installation of RAYTRACE in the DOS environment will be made more convenient by appropriate planning of directories. What follows is one possible directory setup that has been of use while developing the program. Individual user needs and pre-existing directories will dictate the directory structuring for each site. The primary point is that the user of RAYTRACE will make usage easier by planning ahead.

The directory scheme used during development and testing of RAYTRACE involved the use of three parallel directories. A working directory was set up containing the source code. Editing, compiling, and linking are performed within this directory. When significant amounts of debugging are anticipated, the .MEN files and a trial input

file are moved into the working directory, although by and large such duplication of files is avoided in order to save space.

A simulation directory is maintained for day-to-day usage of RAYTRACE. This directory holds .DAT (input) and the .MEN files, as well as the executable version of RAYTRACE. Numerous input files may be maintained by naming them in some mnemonic fashion. This may become crucial, as with a large number of files it is entirely possible to accidentally destroy another file by attempting to have two files of the same name.

The simulation directory is maintained in an uncluttered state by having a results directory to which the results of raytracings may be moved. This is particularly important because RAYTRACE simply uses generic filenames when it writes out results. If many runs are to be performed, it is to the user's advantage to rename and move the result files in order to keep them straight. Again, this helps avoid the unpleasantness of unwanted destruction of result files.

An additional directory that may be of help to some users is a versions directory. This may be a subdirectory of the working directory, to which older or different versions of the code are moved. Again, these files should be renamed in order to distinguish them at a later date. An advantage of such a subdirectory is keeping the working directory uncluttered.

The source code may now be copied into the desired directory. If the source is downloaded to the DOS system from a VMS system, some things may need to be edited before RAYTRACE will successfully compile. Some DOS-based Fortran compilers need markers at the beginning of subroutines so that subroutines will start on a new page when the compiler is asked to produce a listing file. These markers need to be added.

Also, different compilers have different means of suppressing generation of a new line after a WRITE statement to the screen. This means that three lines of the subroutine KEYBRD need to be changed from one development environment to the next. This subroutine is located in the file RAYTRA4.FOR. Edit this file and go to the subroutine KEYBRD. There is a comment block below the variable declarations which describes the necessary changes.

RAYTRACE is now ready to be compiled and linked. If default naming is to be used, then RAYTRA4 should be listed first in the link com-

mand. The executable version may now be copied to the raytracing directory. The .OBJ files should be left intact. This way, when changes are made to one source file, only that one file need be recompiled and all the object files may be linked. If disk space is a consideration, the executable may certainly be deleted from the working directory.

READER2.FOR should also be compiled and linked. The executable may be copied to the results directory. The object and executable files left in the working directory may be deleted to conserve disk space.

The following files should now exist. The working directory should contain a complete set of both source and object files for RAYTRACE, as well as the source for READER2.FOR. The executable versions of RAYTRACE (default name RAYTRA4.EXE) and READER2 should be in the locations that the user has chosen as appropriate. The .MEN files should be located in the same directory as the RAYTRACE executable.

There are several utilities available with some DOS-based compilers that ought to be taken advantage of if available. One of the drawbacks of most DOS Fortrans is the colossal size to which the executable file grows, relative to the size of the source or object files. A necessary utility is therefore some sort of program that compresses the executable to the size that it ought to be.

The other utility is a 'make' utility. This automates the process of building that executable by checking the files that go into constructing the executable to see which ones have changed. Compilation only occurs for those files that need it. If much alteration of RAYTRACE is to be done, a 'make' utility will save a great deal of time. If a 'make' utility is not available, then creative use of DOS batch files may serve as a substitute.

PART TWO:
OPERATOR'S GUIDE

GENERAL INSTRUCTIONS

Overall program flow

The general scheme for using RAYTRACE is much like that of other simulation programs. Input is usually from a data file, and the results are stored in output files. During the course of execution, the user is given the opportunity to change a number of parameters, allowing a variety of situations to be simulated in one session. These settings may be saved for later use. The user is also given the ability to redo entries which were made in error.

Figures 1a & 1b summarize the overall flow of RAYTRACE. When the program is started, the user may choose between doing a completely new problem, or one which has been done before and therefore has an already existing data file with options set. If an old problem is to be worked on, all of the problem's option settings and ionospheric data are loaded in and the user is deposited directly into the main menu. If a new problem is being performed, the ionospheric data must be input. This will usually be by means of a grid file. A grid file is simply a file containing the pertinent ionospheric information at a gridded set of latitudes and longitudes. Grids may be entered from the console, but this is only practical for relatively small grids. Again, after the ionospheric data is entered, the user is placed in the main menu.

FIGURE 1a.

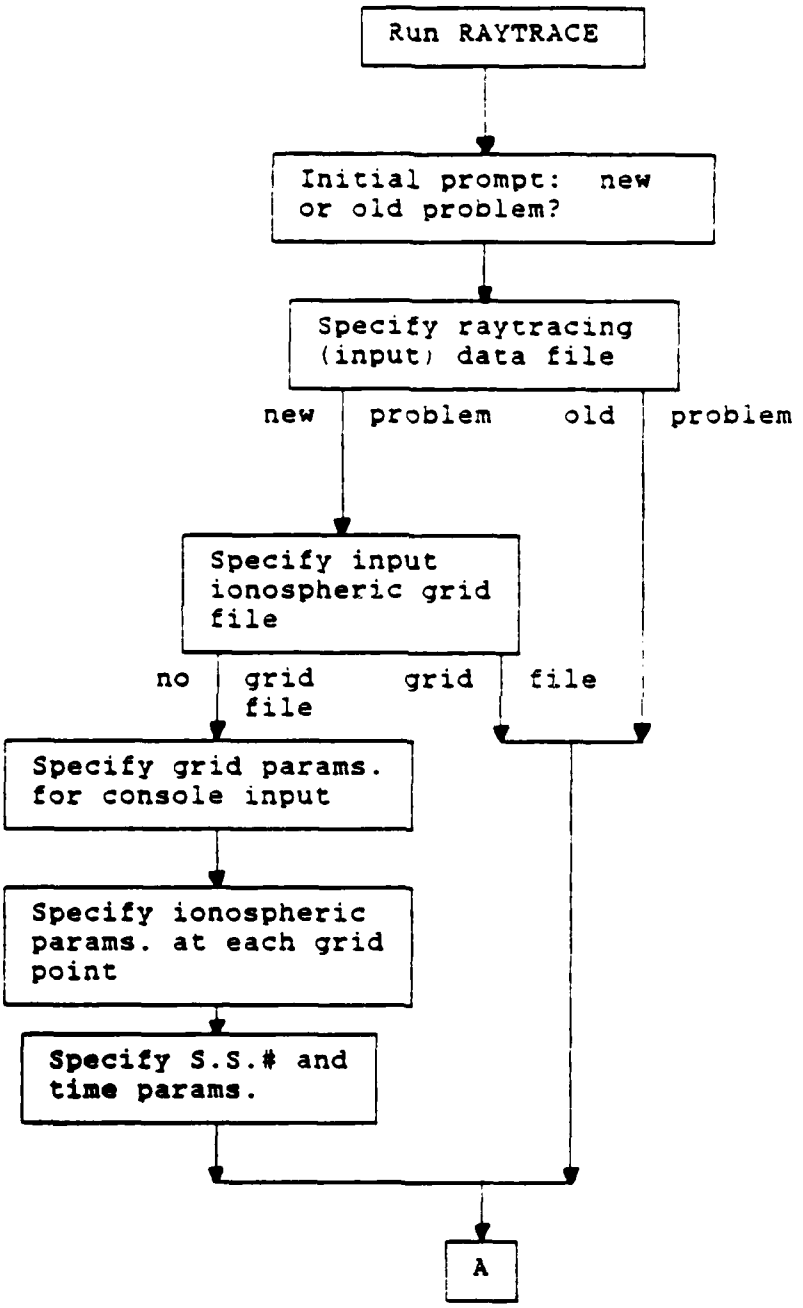
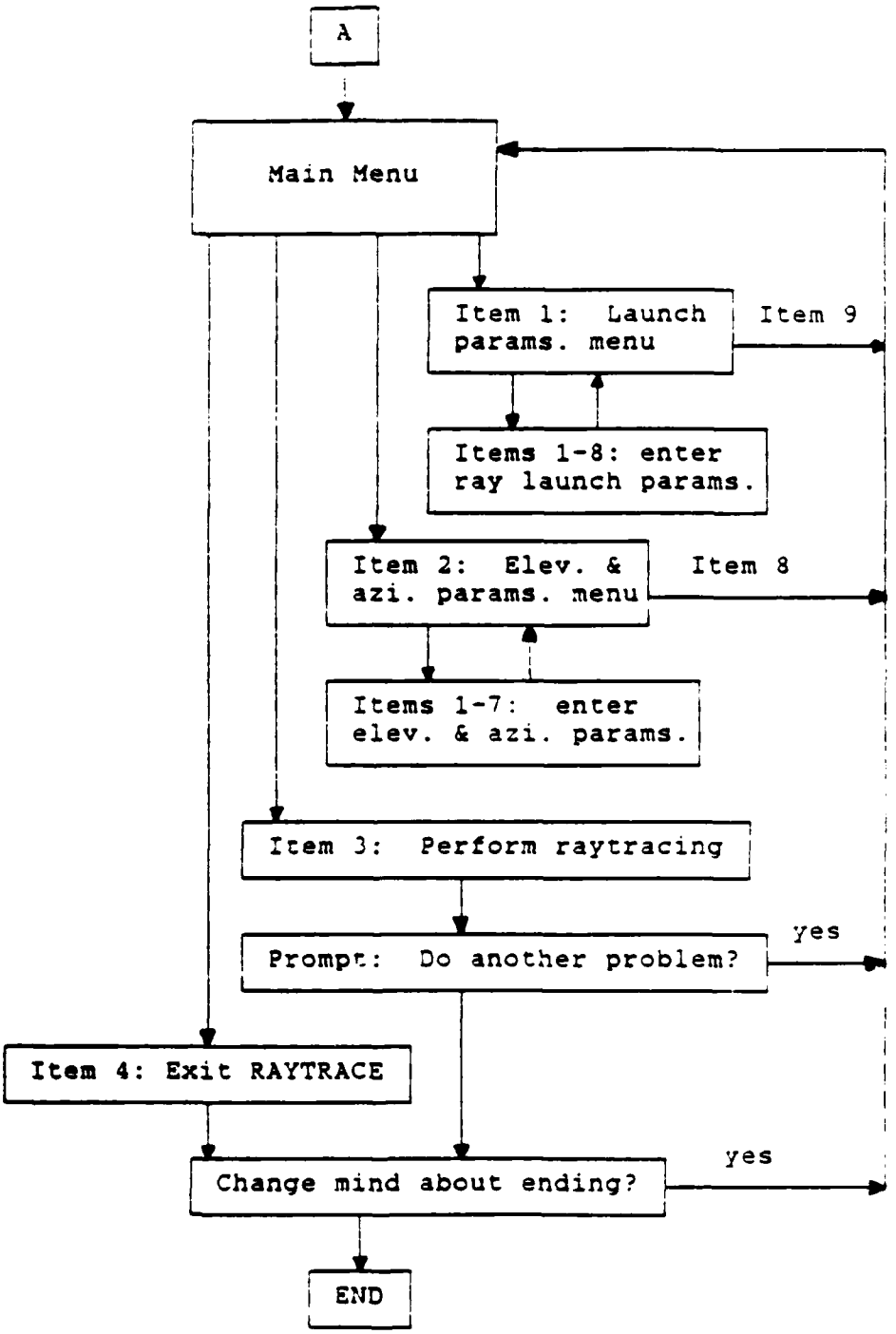

Run RAYTRACE
Initial prompt: new or old problem?
Specify raytracing (input) data file
new problem
old problem
Specify input ionospheric grid file
no grid file
grid file
Specify grid params. for console input
Specify ionospheric params. at each grid point
Specify S.S.# and time params.
A

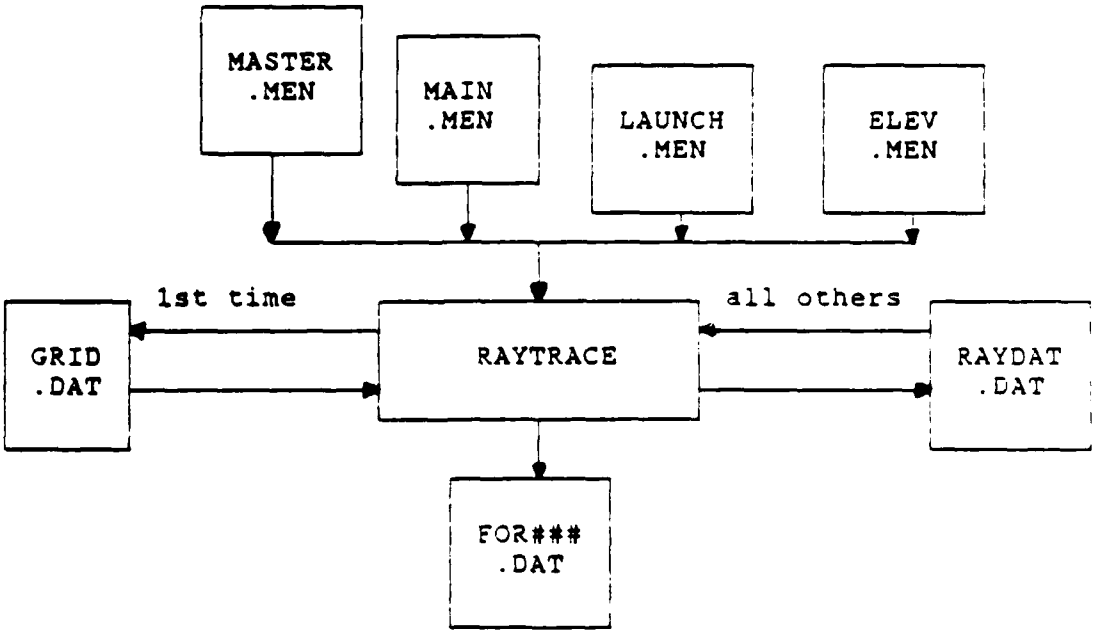
FIGURE 1b.


A
Main Menu
Item 1: Launch params. menu
Item 9
Items 1-8: enter ray launch params.
Item 2: Elev. & azi. params. menu
Item 8
Items 1-7: enter elev. & azi. params.
Item 3: Perform raytracing
Prompt: Do another problem?
yes
Item 4: Exit RAYTRACE
Change mind about ending?
yes
END

The main menu of the program is a central point to which execution will return unless the session is explicitly ended. There are two menu items which lead to other menus. These are for the entry of parameter settings. The other selections are to execute the raytracing using the current parameter settings, and to exit from the program. All of this will be discussed in greater detail below.

During execution, RAYTRACE must deal with a number of other files. The general relationship of the program and these files is summarized in Figure 2. The file GRID.DAT is a grid file, as described briefly above. It contains the ionospheric specification for a given situation and is used as input on fresh problems. This file must be generated by some outside means, although if the specification is entered into RAYTRACE by hand, a GRID.DAT file will be produced. The RAYDAT.DAT file is the file which contains the settings from the previous sessions, as well as the ionospheric specification. The four files with the .MEN suffixes determine the appearance and text of the menus. Results are output into files of the type FOR###.DAT, where ### starts at 040 (This choice is an artifact of earlier versions.) and increments each time a new raytracing is done within a given session.

FIGURE 2.



Ionospheric input

One of the primary benefits of RAYTRACE is the ability to do realistic three-dimensional raytracing with a complicated ionospheric specification. This capability is implemented by means of the grid file(s) described briefly above. The ionospheric grid file may be produced by a numerical model, or the appropriate parameters may be obtained from experimental data and massaged into a grid file. A grid file may also be produced by direct keyboard entry of ionospheric parameters into RAYTRACE.

The grid file, however it is produced, is composed of a number of data arrays. First, the definition of the latitude and longitude grid is recorded. Following this, the six ionospheric parameters required for specification of the vertical profile are given for each point on the grid. Finally, other pertinent data are recorded. The six parameters are the maximum plasma frequencies for the E, F1, and F2 layers, the heights for the maximum frequencies for the F1 and F2 layers, and the semithickness of the F2 layer. The choice of these parameters is determined by the profile model that is built into the current version of RAYTRACE. Currently, RAYTRACE uses the vertical ionospheric profile contained in the RADAR-C model [2]. Because the raytracing is so intimately mated with this model, any input data coming from experiment or some other model must be converted into this form or RAYTRACE must be modified to accommodate the new model profile. The full details on the composition of a grid file are found in Appendix A.

Operating instructions

Before beginning operation of RAYTRACE, be sure that the instructions in the setup section above have been followed. Also, be sure that any input data that are desired exist in a grid file. RAYTRACE may now be executed in the manner appropriate to the system it is operating on.

When RAYTRACE is first run, the following prompt appears:

" Is this a new (1) or old (0) problem? (0): ".

This prompt is typical of those used in RAYTRACE and so it is appropriate to discuss it here in some detail. First, the prompt displays some text which describes the information sought. Any non-intuitive form of response, such as the 0 or 1 above, or other information which will in some way limit the acceptable response, is displayed as part of the prompt text. After the prompt text, the current (or default) value is displayed in parenthesis. The colon marks the end of the prompt, after which the user types his response. If the user merely types a return, then the default value is used as the response. REAL-valued responses may be typed without the decimal point if there is no fractional part to the response, and character responses are case-insensitive.

The above prompt is used to determine whether or not this RAYTRACE run is for an entirely new data set, or a previously used data set. A new problem is simply one for which no RAYDAT-type file exists, or for which no previously existing files of this type will be used. This type of file has been mentioned briefly already and will be discussed more fully below. By selecting the option for a new problem the user is therefore able to start either a genuinely new problem, or use an old ionospheric specification with all new options. An old problem, on the other hand, is one for which the ionospheric specification and the options have been stored in a file of the RAYDAT type. Please note that a response to this prompt MUST be TYPED, i.e. the default cannot be accepted. This is an attempt to ensure that the operator will make the correct choice.

The next prompt that comes up is the following:

```
" Enter filename for storage of ionospheric
  information (10 char max):                (RAYDAT.DAT): '.
```

This prompt asks for the name of the file which will contain both the ionospheric specification and the parameter options settings used in the particular problem. If the problem is a new one and the file given already exists, a Fortran error message will occur, the content of which will depend upon the compiler used. Aside from this, the response to this prompt is straightforward.

The program will now proceed to the main menu if the problem has been declared to be an old one. However, the ionospheric specification still needs to be obtained if this is a new problem. The next prompt asks for the name of the file containing this ionospheric information:

```
" The ionospheric grid file name is?
  (type NONE if none exists)              (GRID.DAT  ): ".
```

The data is retrieved from the grid file whose name is given and the program then proceeds to the main menu. If 'none' is typed (remember, the response is case-insensitive), the program then executes a section of code to read the appropriate information from the console.

Console entry of the ionospheric specification involves three sets of prompts. The first set of prompts asks the user to provide the necessary parameters for specifying the locations of the grid points. The prompts are:

```
"Input lat grid spacing (deg):          (      .000000000    ):
 Input lon grid spacing (deg):          (      .000000000    ):
 Input starting latitude and longitude in degrees.
                  LATITUDE :            (      .000000000    ):
 LONGITUDE (east = positive):           (      .000000000    ):
 Input # of grid points in lat.:        (      .000000000    ):
                  ... in lon.:          (      .000000000    ):
 Grid setup OK (Y/N) ?                  (         Y): ".
```

The first two queries set the latitude and longitude spacing between grid locations. The next two fix the location of the south-west corner of the grid. Longitudes are measured from Greenwich with positive values eastward and negative values westward, so that the range is between -180 and +180 degrees. The last pair of queries gives the extent of the grid in increments of the grid spacing in the appropriate direction. One word of warning is appropriate here. Don't forget about counting BOTH end points when entering the number of grid points! Finally, the user is asked if the values entered are correct. If so, the execution of the program goes on. If not, the user is returned to the beginning of this set of questions and allowed to repeat the data entry. The current settings will be shown as the default values which eases the correction of the errant entry.

The next set of prompts is used to obtain the actual vertical profile data at each grid location. The location is displayed and the user is prompted to enter each of the six values needed to specify the vertical profile used with this version of RAYTRACE, as follows:

```
" GRID PT. - Lat:     ##.####      - Lon:     ###.####

Input  foE**2 :                          (      .000000000    ):
Input    hmF1 :                          (      .000000000    ):
Input foF1**2 :                          (      .000000000    ):
Input    YmF2 :                          (      .000000000    ):
Input    hmF2 :                          (      .000000000    ):
Input foF2**2 :                          (      .000000000    ):
Profile inputs OK (Y/N)?                 (         Y): ".
```

The values requested are the plasma frequencies (squared) for the three ionospheric layers, the E, F1, and F2, at the layer maximum. These are given in units of MHz-squared. The heights of the maximum plasma frequency for the F1 and F2 layers are given in kilometers as is the value of the semithickness of the F2 layer. As with the previous set of prompts, if the user acknowledges that the inputs are OK, the program will continue on. If the values entered are not correct, the user is given the chance to reenter them. These profile entry prompts continue to come up on the console until all grid locations have been exhausted.

The final of the three sets of prompts asks for the time of the simulation and the sunspot number:

```
" Input (integer) sunspot number:        (   0):
  Input year:                            (   0):
  ...  month:                            (   0):
  ...    day:                            (   0):
  Input UT time (hr) :                   (   0):
  ...   UT time (min):                   (   0):
  S.S. # and times OK (Y/N)?             (         Y): ".
```

The sunspot number is the Zurich sunspot number. The year, month, and day are given as their usual integer values. The time is Universal Time and is integer also. After this prompt, the entry of the ionospheric specification is complete and the program proceeds to the main menu.

By whatever route, the program will now have reached the main menu. The main menu serves as the hub of the program as far as the user is concerned. All major actions that may be performed are done from the main menu. The menu contains four items:

```
                    MAIN MENU

        1 - Edit launch parameters

        2 - Edit elevation/azimuth parameters

        3 - Proceed with raytracing

        4 - Quit program

 Your choice (1-4 only, please)?          (   0):
  Press RETURN to continue.               (          ): ".
```

This exhibits the typical structure of a menu in RAYTRACE. The menu is titled, and the items are given with corresponding numbers. Selection is made by entry of an item's number. The entry is checked and if it is out of bounds for that menu, the user is prompted for the choice again. The 'Press RETURN to continue.' prompt is self-explanatory, except for the opportunity it affords the user to abort the previous action. If the user types 'a' (for abort), the menu choice will be aborted and the menu will be presented again.

The first two items of the main menu are used to invoke menus for the entry of parameter values for the particular problem being done. These values are divided up into basically general values in the first menu and elevation/azimuth values in the second. The third and fourth items are self-explanatory.

Selection of item 1 of the main menu brings the Launch Parameters menu to the screen. The parameters which are entered by means of this menu are of a general sort. They include the starting location of the ray, the properties of the ray, and the conditions for cutting off the ray calculation. The Launch Parameters menu is:

" LAUNCH PARAMETERS MENU

1 - Bounce limit

2 - Signal intensity

3 - Conductivity

4 - Launch point

5 - Launch height

6 - Range & height limits

7 - Ray path increment

8 - Wave frequency

9 - Done

Your choice (1-9 only, please)? (0):
Press RETURN to continue. ( ): ".

The first item on the menu is the bounce limit. Selection of this item brings up the following prompt:

" Bounce limit (1): ".

When the ray propagation represents traditional HF communications type raypaths, i.e. rays which are reflected from the bottom side of the ionosphere and which return to earth, it is convenient to place a limit on the number of times that a ray can return to earth. This may be a maximum of 10 currently. In essence, this provides for a kind of fuzzy range cutoff to supplement the hard range cutoff which will be discussed later. Pertinent values for the ray are output to a file at bounce (earth impact) points, and at the end of the ray. This item is somewhat meaningless for raypaths which are not restricted to the space between the bottom of the ionosphere and the earth's surface. One artifact of this is the blank record which may appear at the end of the results file for certain raypaths. This blank record is completely harmless and, to use current terminology, it's not a bug, it's a feature.

The next two items deal with the signal intensity calculations that will be done along the ray. The prompt for item 2:

" Do signal intensity (no = 1)? (0): ",

is used to enter a flag which turns the calculation of signal intensity on and off. Currently, the signal intensity computation that is performed is the geometric spreading of a ray bundle, including focusing and defocusing effects, and reflection loss upon earth impact. The third prompt:

```
" Reflect from ground(1) or water(0)?      (    0): ",
```

is used to determine the conductivity of the reflection surface used in the calculation of reflection loss. At present this is a global choice.

Menu selections 4 and 5 determine the geographic location and altitude of the starting point of the ray. The latitude and longitude are entered at the prompt:

```
" Enter launch pt. latitude                (      .000000000    ):
  Enter launch pt. longitude               (      .000000000    ):".
```

The values are entered in degrees with the longitude ranging from -180 to +180 degrees, positive values to the east of the prime meridian. The starting height is entered at the prompt:

```
" Enter starting ht. (km)                  (      .000000000    ):".
```

This is the height of the starting point above the surface of the earth, defined as the spherical surface having a radius equal to the average earth radius.

Item six provides for the entry of cutoff criteria for the ray-tracing problem:

```
" Enter range limit (km)                   (      3000.00000    ):
  Enter altitude limit (km)                (      36000.0000    ): ".
```

The range in the range limit prompt is great circle range at the earth's surface. It should be thought of as an angular range along the great circle direction defined by the original azimuth of the ray, since this is how it is handled internally. By thinking of this limit as an angular one, it may be readily generalized to problems for which both end points of the ray are above the earth's surface. The altitude limit simply sets a cutoff altitude above the surface of the earth. The default value of 36000 km represents the altitude of geosynchronous satellites.

Selection seven of the Launch Parameters menu allows the user to set the size of the steps taken along the ray as its propagation is modeled. The prompt is as follows:

" Enter raypath increment (km) (4.00000000): ".

Settings for this represent a compromise between accuracy of the results and speed of computation. The default value of 4 km represents a rough optimum. Practically, values much smaller than 0.1 km cause the computation to take much too long and sufficiently small values can begin to cause a decrease in accuracy due to accumulation of roundoff errors from all over the program. Large values are limited by the theory underlying RAYTRACE, and should generally be less than 10-15 km.

The eighth menu item sets the frequency of the electromagnetic wave that the ray represents. The prompt is:

" Enter wave frequency (MHz) (5.00000000): ".

Because effects due to the earth's magnetic field are not implemented in the current version of RAYTRACE, use of frequencies much below about 2 or 3 MHz is questionable. At the high end, frequencies beyond a few GHz begin to cause inaccuracies because of repeated operations involving one minus a small number.

Selection of item nine in the Launch Parameters menu returns the user to the main menu. If the user has decided to alter any of the parameters set in the Launch Parameters menu, the menu may simply be reentered at this point and the values changed. Parameter settings for the raytracing may be changed over and over, the ray will be traced using the current settings.

The second item in the main menu invokes the Elevation Parameters menu. This menu has eight items:

" ELEVATION PARAMETERS MENU

1 - Dimension of problem

2 - Starting azimuth

3 - Starting elevation

4 - Elevation resolution

5 - Azimuth resolution

6 - Elevation limit

7 - Azimuth limit

8 - Done

Your choice (1-8 only, please)? (0):
Press RETURN to continue. ( ): ".

This menu works in an identical fashion to the Launch Parameters menu, but with one twist. It will become clear from discussion of the individual menu items that some of them simply do not apply in all circumstances. In cases where this is true, those items which don't apply are disabled. Attempting to select such a disabled item will return the user to the menu.

Selection one of the Elevation Parameters menu allows entry of a quantity that is known as the dimension of the problem:

" Enter dimension of problem (.000000000): ".

The dimension may have values of 0, 1, or 2. The dimensionality is that of the pattern of rays to be sent out by RAYTRACE. A single ray may be sent (dimension 0). However, multiple rays may be sent in a single run. A vertical fan of rays may be specified, which is to say that the program may be instructed to step in elevation for a fixed azimuth. This is dimension 1. The program may be instructed to step both in elevation and azimuth. This is dimension 2. The aimpoints of the ray(s), if pictured on an elevation versus azimuth plot, then form either a one or two dimensional array, or a single point.

Continuing with this visualization, the array is taken to start in the lower left corner, that is, the lowest value of both elevation and azimuth. For a single ray, the array of values is simply restricted to just this single point and the dimension one problem uses the leftmost

column of the array. All cases require the input values for the starting corner of the array of elevation and azimuth values. Menu items two and three handle this:

" Enter starting azimuth (deg) (.000000000): ",

and,

" Enter starting elevation (deg), use neg.
for values >90 deg from zenith. (.000000000): ".

The azimuth is given in the conventional manner. As the prompt for starting elevation states, elevation angles below horizontal are allowed. Acceptable values range from -90 degrees to +90 degrees, permitting rays which have their origin at altitude to be sent downward.

The next menu item, item 4, is used to enter the elevation spacing between rays,

" Enter elev. resolution (deg) (.000000000): ".

This only applies when a multi-dimensional array of rays is being considered, and so may only be selected when the problem dimension has been declared to be one or two. The elevation values are stepped upward (i.e. toward increasing elevation) by this value. Related to this is item six, where the limiting value for the elevation is given. This prompt,

" Enter elev. limit (deg) (.000000000): ",

is used to define the extent of the pattern of rays in elevation. As with the previous prompt, this may only be selected when the problem is declared to be of dimension one or two.

The fifth prompt is used for entry of the azimuthal spacing in the pattern of rays to be sent out. The prompt is:

" Enter azimuth resolution (deg) (.000000000): ".

This prompt is coupled with the seventh prompt,

" Enter azimuthal limit (deg) (.000000000): "

which provides the program with the information on the azimuthal extent of the ray pattern. These selections only apply for the case when the dimension of the problem is two.

Finally, the eighth item of the Elevation Parameters menu will return the user to the main menu when selected. As with the Launch

Parameters menu, the user may continue to change the values entered in the Elevation Parameters menu until they are satisfactory. These are then the values which will be used in the raytracing.

The main menu's third selection will cause the actual computation of the ray(s) to begin. The values entered in the Launch Parameters and the Elevation Parameters data entry menus are the values that will be used. At this point, if this is the first time through a new problem, the parameters and ionospheric data are written out to the RAYDAT-type file and the ray computation begins immediately. The first time through an old problem, the user is prompted:

" Overwrite existing file (Y/N)? ( ): ",

about overwriting the old parameter values with the new ones. If not, the user is asked:

" Backup the datafile (Y/N)? ( ): ",

whether a backup copy (another RAYDAT-type file) of the settings is desired. After answering these questions, the ray calculation begins. For either case, after the initial time in a problem, every succeeding time the user is asked:

" Update the datafile (Y/N)? ( ): ",

whether or not the data file should be updated with the latest settings. The interface is set up to give the user ample opportunity to preserve the new parameter settings if desired, but the user has the choice of not saving these settings if that is what is desired.

The user is informed of the start and finish of the ray calculations by the messages:

" BEGINNING RAY LOOPS.

RAY LOOPS DONE. ".

Upon completion of the computations, the user is informed of the file that the results have been written to by the following message:

" Results written to file FOR040.DAT ".

The numeric part of the filename will increment each time the user does another raytracing in a single session. Leaving the program resets the numeric portion of the filename.

Now that a particular raytracing is done and the results are stored away, the user is prompted:

" Do another problem (Y/N)? ( ): ",

whether or not additional raytracing is desired. An affirmative answer will return the user to the main menu so that any changes to parameters may be made and more rays may be traced. A negative answer has the same effect as choosing item four from the main menu. First, the prompt:

" Press RETURN to continue. ( ): ",

is issued, allowing the user to abort the exiting of the problem by entering 'a'. If a return is entered, then the program exits with the following final message

" Session done. For safety's sake, copy the data and result files into a separate directory to prevent accidental overwriting. ".

This warning should be heeded. For use on VAX/VMS systems, a large number of virtually indistinguishable files with the same names and different version numbers may result in loss of results through confusion over which version is which. DOS based systems present an even greater problem, because the attempt to save results to a file already in existence will cause the old file to be destroyed.

Output Files

RAYTRACE produces three types of output files. The first is a GRID.DAT type of file, if the ionospheric specification has been entered by hand. This file has been discussed already. The second type of file is the RAYDAT.DAT type. This file is essentially just a GRID.DAT file with the additional user-entered parameters appended to it. First, the definition of the latitude and longitude grid is recorded. Following this, the six ionospheric parameters required for specification of the vertical profile are given for each point on the grid. Other pertinent data are recorded, including the date and time of the ionospheric specification, and the sunspot number. These previous data are just those recorded in the GRID.DAT file. Addition-

ally, the values from the Launch Parameters menu are recorded and finally the values from the Elevation Parameters menu are stored. Full detail of the RAYDAT.DAT type of file is given in Appendix A.

The remaining type of data file is the results file, FOR###.DAT. This file contains the results of a single run of RAYTRACE. First, the bounce limit (plus one for the end point of the ray), the number of azimuths, and the number of elevations are recorded. This enables any program reading the results to break the results out into those for individual rays. For each ray, the azimuth and elevation are recorded. Then the various results are saved for each data point along the ray. These data points are earth impact points and the end point. If the ray is terminated before the bounce limit is reached, blank records will occur and should be ignored. Each succeeding ray is recorded in a similar manner; first, azimuth and elevation, and then the results. The details of this particular file type are presented in Appendix A.

It should be restated that the user ought to be careful and develop defensive file maintenance procedures. After each session files should at the least be given more suitable names that are perhaps indicative of their contents. This will not only reduce the likelihood of accidental destruction of results, but will aid in later analysis. Also, the user should be aware of when the files get written to disk. The RAYTRACE produced GRID.DAT file has its contents written during the entry of the data. The RAYDAT.DAT file is written just before the rays are traced, and the FOR###.DAT (results) file is written during the raytracing.

TUTORIAL EXAMPLES

Although the operation of the various commands of RAYTRACE has now been discussed, this knowledge will be reinforced by going step by step through some examples. These examples will be presented with actual samples of screens, for clarity. The first tutorial involves setting up RAYTRACE to use a spherically symmetric ionosphere with the specification entered by the console. This will create the data files used in the subsequent two tutorials. The second is to perform a new problem using a grid file as input. The last is to perform an old problem. It is hoped that the user will actually perform these tutorials on his/her system, even if operation of the program seems obvious by this time. At the least, the exercise will increase familiarity with the operation of the program, while potential problems or questions may be preempted by use of these tutorials.

New session using keyboard input

To begin this tutorial, run RAYTRACE. This will be a new problem, so respond with '1' to the first prompt. The next prompt will ask about the file to which all the ionospheric and parameter information will be written. The filename that has been chosen for this tutorial is TRIAL.DAT. At this time the screen should look like the following.

Screen 1

```
Is this a new (1) or old (0) problem?    (    0):1

Enter filename for storage of ionospheric
information (10 char max):                (RAYDAT.DAT):trial.dat
```

The next prompt is the prompt which asks about the file from which the ionospheric specification will be read. For this example, there is none, so the appropriate response is 'none'. The screen that corresponds to this is Screen 2.

Screen 2

```
The ionospheric grid file name is?
 (type NONE if none exists)              (GRID.DAT   ):none
```

The next set of prompts that comes up is a set of prompts asking about the definition of the grid on which the ionosphere is to be specified. The first two prompts deal with the latitude and longitude spacing between the grid points. Since this example is a spherically symmetric case, only one grid point will be needed and so the spacing is immaterial. For this reason, a response of '1' is chosen in each case. The next two prompts ask for the starting latitude and longitude for the grid, in this case this is the location of the only grid point. The responses chosen are '45' for latitude and '-90' for longitude. This longitude corresponds to a west longitude. The final two informational prompts here ask about the size of the grid in the latitude and longitude directions. Since there is to be only one point, the number of points in each direction will be '1'. Finally, there is a prompt asking whether the input is correct. Assuming everything has been entered as shown, then typing return will accept the default choice of 'Y'. At this point, the screen should look like Screen 3.

Screen 3

```
Input lat grid spacing (deg):          (      .000000000    ):1
Input lon grid spacing (deg):          (      .000000000    ):1
Input starting latitude and longitude in degrees.
                  LATITUDE :           (      .000000000    ):45
LONGITUDE (east = positive):           (      .000000000    ):-90
Input # of grid points in lat.:        (      .000000000    ):1
                  ...  in lon.:        (      .000000000    ):1
Grid setup OK (Y/N) ?                  (         Y):
```

Now the program begins to prompt for the ionospheric information for the location specificied above. The location of the point is given at the top of the prompt. The parameters prompted for occur ordered in terms of height. It should be noted that the values given here are merely fictitious. First is the plasma frequency squared for the maximum of the E-layer, with a value to be entered as '2'. Next the F1-layer is specified, by the height of the maximum of the F1, '150', and the plasma frequency squared of the F1 maximum, '7'. Finally, three F2-layer parameters are prompted for. They are the semithickness of the F2, '65', the height of the F2 maximum, '300', and the plasma frequency squared at the maximum of the F2, '20'. Again, the program prompts to see if all the input is OK. If it is, the response should be the same as for the similar prompt above. The console screen should now look like Screen 4.

Screen 4

```
GRID PT. - Lat:     45.0000      - Lon:    -90.0000

Input  foE**2 :                          (      .000000000    ):2
Input    hmF1 :                          (      .000000000    ):150
Input foF1**2 :                          (      .000000000    ):7
Input    YmF2 :                          (      .000000000    ):65
Input    hmF2 :                          (      .000000000    ):300
Input foF2**2 :                          (      .000000000    ):20
Profile inputs OK (Y/N)?                 (         Y):
```

The final set of prompts for the environmental specification involves recording the date and time of the problem, and the sunspot number. First the sunspot number is entered, in this example '80'. Next, in descending order of time scale, come the date and time prompts. As indicated by the defaults given, these are expected to be integers. This example uses the following responses in order: '1988', '2', '17', '12', and '00'. After the verification prompt, the screen should look like Screen 5.

Screen 5

```
Input (integer) sunspot number:          (   0):80
Input year:                              (   0):1988
...  month:                              (   0):2
...    day:                              (   0):17
Input UT time (hr) :                     (   0):12
...   UT time (min):                     (   0):00
S.S. # and times OK (Y/N)?               (          Y):
```

At this point, the program brings the user to the main menu. It is from this menu that other menus are invoked for entry of the remaining parameters, and that the actions of tracing rays and quitting are taken. The parameters are currently a blank slate, so both of the data entry menus need to be selected and the parameters entered. Starting at the top, select menu '1'. The program will respond with another prompt. As it says, if the selection is the desired one, the user needs only type a return to proceed. As was mentioned earlier, though, the selection may be aborted by responding with an 'a'. If this is done, the user will be returned to the previous menu. Before typing return, the screen will look like Screen 6.

Screen 6

```
                    MAIN MENU

      1 - Edit launch parameters

      2 - Edit elevation/azimuth parameters

      3 - Proceed with raytracing

      4 - Quit program

Your choice (1-4 only, please)?          (    0):1
 Press RETURN to continue.               (           ):
```

Selection of item one of the main menu brings the user to the Launch Parameters menu. This menu is used for the entry of various parameters relating to the ray calculation. These values include the location of the starting point for the ray(s), settings which are used to halt the calculation, properties of the ray(s), and values which determine whether some calculations are performed and if so, how. This example will simply make the selections from the menu in numerical order, therefore, the first entry will be '1'. This situation is reflected in Screen 7.

Screen 7

LAUNCH PARAMETERS MENU

1 - Bounce limit

2 - Signal intensity

3 - Conductivity

4 - Launch point

5 - Launch height

6 - Range & height limits

7 - Ray path increment

8 - Wave frequency

9 - Done

Your choice (1-9 only, please)? (0):1
Press RETURN to continue. ( ):

This selection from the Launch Parameters menu brings up a query about the limit to be placed on how many times a ray may return to earth. This is useful when dealing with low frequencies which will not penetrate the ionosphere. The problem then may be set up to terminate after a certain number of hops has occurred, regardless of the other termination conditions. For the purpose of this example, accepting the default value of '1' will be sufficient, so the response will be to just type a return. The screen should look like Screen 8. Note that in some of the screens that follow, only the lower portion of the screen is shown.

Screen 8

Bounce limit (1):

Once the entry of the data is made, the user is immediately returned to the Launch Parameters menu. This time the selection will be item '2'. Because the process by which items are selected from menus is uniform, the screen need not be repeated. Menu item two leads to a prompt which allows the program to know whether or not to perform signal intensity calculations. For the sake of speed in running the example, the value entered is '1' which corresponds to no. The screen example is given as Screen 9.

Screen 9

Do signal intensity (no = 1)? (0):1

Returning to the Launch Parameters menu, the next selection made will be '3'. This item allows the user to designate a global value for the conductivity of the earth's surface. These values correspond to soil and water. They are average values. Conductivity values are used when signal intensities are calculated so that the loss due to reflection from the earth's surface may be accounted for. Since no signal intensity will be calculated for this example, this item was selected for instructional purposes only. Because any answer will suffice, the default may be accepted by typing return. This is shown in Screen 10.

Screen 10

```
Reflect from ground(1) or water(0)?      (    0):
```

The next selection from the Launch Parameters menu will be item number '4'. This prompts the user for the latitude and longitude of the launch site of the ray(s). The values are given in degrees. Remember that longitudes are to be given as positive to the east of Greenwich, with the prime meridian being 0 degrees and west longitudes negative. This gives a possible range of longitudes of -180 degrees to +180 degrees. The launch site that is used is south and east of the ionospheric point, with latitude '40' and longitude '-80'. This results in Screen 11.

Screen 11

```
Enter launch pt. latitude                (       .000000000    ):40
Enter launch pt. longitude               (       .000000000    ):-80
```

Following this, item '5' should be selected from the Launch Parameters menu. The purpose of this item is to provide the third component of the location of the ray launch site, namely the starting height. The starting height is given in kilometers above the earth's surface. This exercise will start on the earth's surface, so the response that is needed is to simply type return so that the default value shown is accepted. Before hitting return, the screen will look like Screen 12.

Screen 12

```
Enter starting ht. (km)                    (       .000000000    ):
```

Upon return to the menu, select item '6'. This set of prompts is used by the program to obtain from the user values which will determine when the problem should be terminated. The first parameter is the range limit. This represents great circle range on the earth's surface. Internally this is dealt with as an angular range, but it is easier to think in terms of ground range when entering the value. This must not be confused with the distance traveled along the raypath! The distance along the ray is, in general, longer than the associated ground range. The value selected for this example is '2000' km. The second value to be entered is the altitude cutoff. This gives the height above the surface of the earth of a spherical shell which, when pierced, causes the raytracing to terminate. For this example a reasonable value is '400' km. The resulting screen is Screen 13.

Screen 13

```
Enter range limit (km)                     (       3000.00000    ):2000
Enter altitude limit (km)                  (       36000.0000    ):400
```

The next quantity to be entered is the raypath increment, selection '7'. This value determines how finely the raypath is broken up for the purposes of the calculation. There is an interplay here of the desire to make the path exceedingly fine for the sake of accuracy, with the desire that the raytracing complete in a finite (and preferably small) amount of time. The appropriate range of values has already been discussed. It is sufficient that the default value of '4' km

represents somewhat of a balance and so this is the value chosen. The prompt is displayed in Screen 14.

Screen 14

Enter raypath increment (km) (4.00000000):

The final important selection from the Launch Parameters menu is item number '8', the wave frequency. This is the value of the frequency of the radio wave that the ray represents. The appropriate range of values for this is discussed above in the operating instructions section. For the purposes of this tutorial, it is desired that the ray not penetrate the ionosphere, so a wave frequency of '4' MHz is selected. The default of five MHz is merely a placeholder so that the program doesn't accidently start with frequency equal to zero and promptly crash due to division by zero. The screen containing this prompt is represented by Screen 15.

Screen 15

Enter wave frequency (MHz) (5.00000000):4

After completing this, all the necessary data has been entered in the Launch Parameters menu. The user may therefore enter '9', the selection indicating that the data entry is finished in this menu. The control of the program is returned to the main menu. From the main menu, the other data entry menu may now be entered. This is done by making selection '2' from the main menu. The Elevation Parameters menu now comes on screen.

As with the Launch Parameters menu, this tutorial will now proceed through the items in the Elevation Parameters menu. The first item in this menu deals with the array of azimuth and elevation values for which rays will be sent out. Select '1' as in Screen 16.

Screen 16

ELEVATION PARAMETERS MENU

1 - Dimension of problem

2 - Starting azimuth

3 - Starting elevation

4 - Elevation resolution

5 - Azimuth resolution

6 - Elevation limit

7 - Azimuth limit

8 - Done

Your choice (1-8 only, please)? (0):1
Press RETURN to continue. ( ):

This selection will provide a prompt asking as to the dimension of the problem. This is defined earlier, however a brief rundown is in order. The term dimension refers to the dimension of the array that could be used to represent the elevation and azimuth values to be used. For this reason, dimension 0 represents just a single value of elevation and azimuth. Dimension 1 represents a single azimuth value and multiple elevation values. Finally, dimension 2 corresponds to the case where there are multiple elevations and azimuths to be used. So that all the menu items may be selected, a value of '2' should be entered here. This is represented by Screen 17.

Screen 17

Enter dimension of problem (.000000000):2

As usual, entering the response above returns the user to the menu. Proceeding down the menu in order, the next selection is '2'. This item allows the entry of the initial azimuth value for the problem. The azimuth is given in degrees and is taken relative to geographic north in the usual manner. Because this example involves only a spherically symmetric ionosphere, one direction is as good as another. The default value of '0' is therefore taken, by typing a return. This situation is depicted in Screen 18.

Screen 18

```
Enter starting azimuth (deg)              (       .000000000     ):
```

Item three of the Elevation Parameters menu is the next one selected. This prompts for the starting elevation angle. Elevation is measured in degrees, with the zenith as +90 degrees, and the nadir as -90 degrees. Negative values of elevation angle come into play for cases when the launch point of the ray is at a non-zero altitude. For this example, a reasonable value to use is '10'. Screen 19 shows this.

Screen 19

```
Enter starting elevation (deg), use neg.
for values >90 deg from zenith.           (       .000000000     ):10
```

The fourth selection from the Elevation Parameters menu deals with the elevation separation between rays. Very little coaching can be given about appropriate values for this parameter. The fineness or coarseness of the elevation resolution must be determined in terms of what is appropriate for a given problem. Since this problem is of an instructional nature, an arbitrary value of '0.5' degrees has been chosen. This is shown in Screen 20.

Screen 20

```
Enter elev. resolution (deg)          (       .000000000     ):0.5
```

Next in line is the fifth menu item of the Elevation Parameters menu. The azimuth spacing between rays is entered at the resulting prompt. This is handled in a manner identical to that of the previous prompt. The value which is to be entered is the same as that for elevation, namely '0.5'. Screen 21 shows this.

Screen 21

```
Enter azimuth resolution (deg)        (       .000000000     ):0.5
```

The method of specification for the elevation and azimuth values depends upon having the starting value, the division size, and the ending value. The ending value for elevation is given by selection of item six from the Elevation Parameters menu. Internally, the difference between the start value and the limit is taken. The number of increments of size equal to the resolution value that will fit into this interval is then calculated. This is the number of elevation steps that will be taken. For the sake of completing the example problem quickly, the limit will be taken to be '10.5' so that rays will be sent out at only two elevations. The response is shown in Screen 22.

Screen 22

```
Enter elev. limit (deg)                ( .000000000 ):10.5
```

Lastly, menu item seven is selected. This allows the azimuth limit to be entered in a manner identical to the elevation limit. Again, the value is chosen so that only two values of azimuth will be used. The combination of these elevation and azimuth limits result in just four rays being calculated. The appropriate value to use for the azimuth limit is then '0.5'. It should be remembered in setting the both of the limits that if a specific number of rays is desired in elevation and in azimuth, that the end points must be counted properly. The prompt and response are shown in Screen 23.

Screen 23

```
Enter azimuthal limit (deg)            ( .000000000 ):0.5
```

All of the necessary parameter values have been entered now. Selection of item '8' of the Elevation Parameters menu causes a return to the main menu. Everything is now ready for the raytracing to be performed. Therefore, item '3' is now selected from the main menu. Because this is a fresh problem, no prompting about the disposition of the entered parameters is necessary. These values are automatically written out to the RAYDAT-type file that was specified at the beginning of this exercise, namely TRIAL.DAT. The user is notified that the ray-tracing has begun. When the calculations are complete, the user is notified and the file to which the results have been written is also noted. The user is then given the opportunity to do another raytracing problem. In order to give a feel for what the program will do, the

answer to this prompt is 'y' for yes. The screen should look like that shown in Screen 24.

Screen 24

```
 BEGINNING RAY LOOPS.
 RAY LOOPS DONE.


Results written to file FOR040.DAT

Do another problem (Y/N)?                    (           ):y
```

The affirmative response causes RAYTRACE to return to the main menu. For the second problem, just a single ray will be traced. To do this, select '2' from the main menu in order to proceed to the Elevation Parameters menu. Once at the Elevation Parameters menu, choose '1' to change the current setting for the dimension of the problem. Recall that a problem of dimension zero is a single ray problem. The prompt that comes on the screen for the dimension of the problem contains the current value, which is 2. Enter a value of '0'. This is shown in Screen 25.

Screen 25

```
Enter dimension of problem                   (      2.00000000     ):0
```

It is important to note that no other values in the Elevation Parameters menu need be changed. In fact, it will be very instructive to attempt to change any of the values corresponding to menu items 4-7. The program will not allow these items to be changed, instead the user is returned to the menu. These items are irrelevant to the calculation once the dimension has been set to zero. Because the purpose of this second portion of the problem is simply to do a second portion to the problem, this one change will suffice. Therefore, select '8' to return to the main menu, and then select '3' to proceed with the raytracing.

This time, before the raytracing is started, RAYTRACE notes the possibility that the user has made changes to parameter values. The user is asked whether or not to update the data file. If the answer is yes, the current values are saved out to the RAYDAT-type file. If the answer is no, then if the user chooses to end the session after the current raytracing, the changed parameter values disappear. For the purposes of this example, it doesn't matter whether the values are saved or not: the response of 'n' has been chosen. This may be seen in Screen 26.

Screen 26

```
Update the datafile (Y/N)?                    (          ):n
```

The program then signals that the tracing of the ray has begun. When the end of the calculation is reached, the display is similar to that seen before. This time, however, it is desired to end the problem. The response to the prompt is therefore 'n'. The screen will look like Screen 27.

Screen 27

```
Update the datafile (Y/N)?                    (          ):n

 BEGINNING RAY LOOPS.

 RAY LOOPS DONE.


Results written to file FOR041.DAT

Do another problem (Y/N)?                     (          ):n
 Press RETURN to continue.                    (          ):
```

The final thing that the program does is to display a message. This message is to warn the user to take proper care of any data files that have been produced during the raytracing session. The point cannot be overemphasized that the filenames given are far from descriptive and results may get lost or destroyed if a myriad of such files exists. The closing message is reproduced in Screen 28.

Screen 28

```
Session done. For safety's sake, copy the data
and result files into a separate directory to
prevent accidental overwriting.
```

At this point, RAYTRACE has finished execution and control has returned to whatever system is being used. Several new files should now be in existence. TRIAL.DAT is the file of the RAYDAT type that contains the saved parameter settings and the ionospheric specifica-

tion. A grid file called GRID.DAT was created by entering the ionospheric data from the console. There are also two results files, FOR040.DAT and FOR041.DAT. As the final message warns, these files should either be moved or renamed. Examining the results files is discussed in a later section.

New session using data file input

This second example will build upon the foundation of the first. The grid file that will be used for input is going to be the GRID.DAT that was produced in the previous example. Reference will also be made to the first example for those portions of the two examples which are identical. It is expected that many RAYTRACE runs will be similar to the current example in that most fresh problems will be initiated by use of an input file specifying the ionosphere.

This example begins by starting RAYTRACE as before. The initial prompt comes up on the console. Since this example is to be a new problem, respond with '1'. The prompt asking for the name of the file to store the ionospheric specification and input parameters now appears. Recall that last time the file TRIAL.DAT was used. If this file hasn't been relocated or renamed, entering this name again may cause a problem. So that the files may be distinguished the name that is chosen for this example is 'TRIAL2.DAT'. After this response, the screen should look like Screen 29.

Screen 29

```
Is this a new (1) or old (0) problem?    (    0):1

Enter filename for storage of ionospheric
information (10 char max):                (RAYDAT.DAT):trial2.dat
```

The next prompt is the one asking for the file containing the ionospheric specification. Unlike the first example, one now exists. The file even has the name GRID.DAT. This is the file to be used, so

the default value of the prompt may be accepted by typing a return. This situation is depicted in Screen 30.

Screen 30

```
The ionospheric grid file name is?
 (type NONE if none exists)              (GRID.DAT   ):
```

The user should now notice a difference from the previous example. Because the ionospheric specification already exists, the program has no need to prompt the user for keyboard entry of the ionospheric parameters. RAYTRACE immediately proceeds to the main menu. From here the previous example may be picked up again. Menu item '1' from the main menu should be chosen. The user should now enter values for each of the items in the Launch Parameters menu. The values to be used are just those of the previous example. That example and its discussion may be followed and will not be repeated here.

Following completion of the Launch Parameters menu, the next step is to proceed to the Elevation Parameters menu by selecting '2' from the main menu. Again, the items of this menu will be completed in numerical order. Just as in the first example, enter '1' as the menu choice. For the sake of speed, just a single ray is desired. Therefore, the response to the resulting prompt will be to accept the default value by typing a return. This is shown in Screen 31.

Screen 31

```
Enter dimension of problem                (      .000000000    ):
```

Next, choose item '2' from the Elevation Parameters menu. The value is chosen to be the same as in the previous example. Recall that this means the azimuth value will be 0 degrees. The default response is therefore to be accepted. Screen 32 show this.

Screen 32

```
Enter starting azimuth (deg)              (       .000000000     ):
```

The starting elevation value is the next to be set. Menu item '3' should be selected. At the resulting prompt, a value of '10' degrees should be entered. This ought to lead to a ray with identical results as one of those done in the first example. The results may be compared as a check. The elevation data entry is shown in Screen 33.

Screen 33

```
Enter starting elevation (deg), use neg.
for values >90 deg from zenith.           (       .000000000     ):10
```

Because the dimension of the problem has been selected to be zero, there is no further data entry needed. Item '8' should therefore be selected from the Elevation Parameters menu. Upon return to the main menu, the user may choose item '3' as in the previous example. This will initiate the raytracing without any intervening prompts. Because this is a fresh problem, the data is automatically recorded in the RAYDAT-type file specified at the start. When the raytracing is complete, the program behaves in a manner exactly identical to that shown in the first example. For the purpose of practice, answer the prompt for another problem with 'y'. This is shown in Screen 34.

Screen 34

```
BEGINNING RAY LOOPS.

RAY LOOPS DONE.


Results written to file FOR040.DAT

Do another problem (Y/N)?                    (           ):y
```

The program returns to the main menu for another time through the problem. So that the problem will be somewhat different, a new elevation value will be used. To achieve this, select '2' from the main menu, and then '3' from the Elevation Parameters menu. An elevation angle of '8' degrees will be used. This will result in a screen that looks like Screen 35.

Screen 35

```
Enter starting elevation (deg), use neg.
for values >90 deg from zenith.          (      10.0000000     ):8
```

This is the only change that is necessary, so choose '8' from the Elevation Parameters menu. When control has returned to the main menu, selection '3' may be made so that the calculation may begin. Again, as with the first example, the query about updating the data file is seen. There is no need to preserve the change that has been made, so answer with 'n'. The raytracing then proceeds as usual, displaying the normal information. Because this example is essentially complete, respond with 'n' to the prompt asking about doing another problem. This is

shown in Screen 36 below. RAYTRACE then displays its final message and returns to the system.

Screen 36

```
Update the datafile (Y/N)?                (          ):n

 BEGINNING RAY LOOPS.

 RAY LOOPS DONE.


Results written to file FOR041.DAT

Do another problem (Y/N)?                 (          ):n
 Press RETURN to continue.                (          ):
```

Old (saved) session

This third example illustrates the use of a RAYDAT-type file for input at the beginning of the problem. As with the second example, this example will be built upon the previous ones, and only those portions which are new for this example will get detailed attention. The type of session modeled by this example is representative of perhaps the majority of RAYTRACE sessions. By reading in a RAYDAT-type file, all of the parameters have some pre-existing setting. Only those parameters that are different for the particular problem at hand need be examined and changed.

First, start RAYTRACE in the usual manner. This time, however, at the initial prompt, the response will be '0' to indicate that this is an old problem that will initially use the parameter values stored in a RAYDAT-type file. The second prompt is the same as has been seen before. This time its purpose is to get the name of the file from which the parameters will be read. Any parameter changes may be stored

in this file also. For this example, the file created by the first example, 'TRIAL.DAT', will be used. This is shown in Screen 37.

Screen 37

```
Is this a new (1) or old (0) problem?    (    0):0

Enter filename for storage of ionospheric
information (10 char max):                (RAYDAT.DAT):trial.dat
```

RAYTRACE now continues directly to the main menu. There is no need to read from a grid file because the pertinent data are stored in the RAYDAT-type file along with the parameters. At this point, if the parameters desired are the ones that were saved during the last session, the user could just select the third item from the main menu and proceed directly with the raytracing. For this example, however, recall from the first example that the parameters entered included a dimension of two for the problem. In the interests of speeding the example along, this should be changed to a dimension of zero. To do this, the user should select '2' from the main menu to reach the Elevation Parameters menu. Then '1' should be selected to reach the prompt from which the dimension may be set. Notice that the default value given is indeed two. Enter the new value of '0' as shown in Screen 38.

Screen 38

```
Enter dimension of problem                (      2.00000000    ):0
```

This is the only change that needs to be made, so enter a choice of '8' to return to the main menu. Now select '3' from the main menu to begin the raytracing. The program now presents prompts that have

not been seen until now. Because the parameters may have been changed, even though this is the first time through the problem, the user needs to be presented with the opportunity to preserve these changes. First, the user is asked if the values should be saved in the current file. If they are not to be saved there, the user is then asked if a new data file should be used to backup the new values. A negative answer to both of these prompts will mean that if the problem is ended the altered parameters will not be saved anywhere. It is not necessary that the change that has been made be saved, so answer 'n' to both prompts.

Some additional comments about these prompts are in order. If the old parameters are overwritten, then the data are safe and there is no need for the second prompt. RAYTRACE therefore skips it and goes on with the calculation. If a backup is desired, the user will be prompted for the filename to be used. Also, on succeeding times through the raytracing the prompt asking about updating the data file (as in Screen 36 above) will be referring to the last file that was written to.

After dealing with the above mentioned prompts, the raytracing is performed. The rest of the prompts are just as they have been before. There is really nothing further to be discovered, so 'n' may be entered in response to the query about another problem, as shown in Screen 39. After this the program ends in the usual fashion, returning to the system. This concludes the final example.

Screen 39

```
Overwrite existing file (Y/N)?          (          ):n
Backup the datafile (Y/N)?              (          ):n

 BEGINNING RAY LOOPS.

 RAY LOOPS DONE.


Results written to file FOR040.DAT

Do another problem (Y/N)?               (          ):n
 Press RETURN to continue.              (          ):
```

Remarks

The three examples given serve to illustrate essentially every feature of RAYTRACE. After following all three examples, the user should be familiar enough with the interface of the program to go ahead and "play around" by trying different things. Such "play" may be worth the time if the user feels the need for additional familiarity with the operation of the program. The menu reference sheet given later should also be of some help.

EXAMINING RESULTS

So far, the discussion has dealt with the use of the RAYTRACE program. The program is of very little use, though, unless there is some way of getting at the results from the raytracing runs. This section deals with the program READER2 that is provided with RAYTRACE.

READER2 represents a zeroth-order program for examining the results of RAYTRACE. While this program has proven adequate in the testing and development of RAYTRACE, it is certain that the end users of RAYTRACE will need to write their own, more sophisticated analysis software.

READER2 is very straightforward to use. When the user executes it, the program prompts for the name of the results file that it is to be read. Upon receiving the name, READER2 immediately displays the important values from the result file. This display is done for each pair of elevation and azimuth values. Results for each data point on the ray are displayed with the values for a particular point taking one screen. Recall that data points on the ray are earth impact points and the end point of the ray. After display of each data point, READER2 pauses and awaits the user's response before continuing. When all the data are exhausted, the program simply ends and returns to the system.

The values displayed on the screen are by and large self-explanatory. The elevation and azimuth values are given in degrees. The direction cosines give the orientation and direction of the ray at that particular point in an SEZ coordinate system where X is South, Y is East, and Z is up. The deviation value represents the deviation away from where the ray would have landed had it stayed on a great circle path. This is a measure of the cross-path tilt of the ionosphere.

Several comments on the regular usage of READER2 are in order here. Although the display of the results on screen is useful, it is often more useful to have a hardcopy of these results. This is achieved not by READER2, but by the system that it is operating on and the console used. Many systems and consoles have the capability to either redirect output to a disk file, or perform screen dumps to a printer. Given the rather widespread presence of such a capability, no provision has been made to produce a printable file. Since the user is likely to develop individualized analysis software anyway, there is really no need for this capability.

Another comment concerns the blank record that will occasionally be displayed. This is entirely harmless. It occurs because provision must be made for the possibility that the limit on the number of bounces may be reached and there still needs to be the additional space for the end of ray point's data. If the bounce limit isn't reached, then this extra space is blank.

TIPS, RESTRICTIONS, REMINDERS, ETC.

This section has the purpose of gathering together some of the comments that have either gone unsaid to this point, or bear repeating. First, there are some comments of a general nature. Comments specific to a particular implementation or environment follow. Finally, some usage hints are presented.

General comments

The speed with which RAYTRACE operates will be of some concern to many users. Some benchmarks have been run on various machines. The two primary implementations discussed in this User's Guide are the VAX/VMS and the DOS versions. The machines compared were an IBM AT-compatible (with math coprocessor) and a VAX 11/785. The benchmarked run uses the TRIAL.DAT file from the four ray example of the tutorial examples above. The run was repeatedly made and the results averaged. Timing was done by stopwatch from typing return at the last prompt before the execution of the ray loops until the message of "RAY LOOPS DONE" appeared on the screen. For the VAX, in addition to the apparent execution time as measured by stopwatch, the CPU time was taken by using the <CTRL>-T mechanism of VMS for obtaining the process time statistics. Because the VAX is a multiuser machine, the usage level is of some significance. The benchmark was run early in the morning before the many users were on the system. At the time of the runs, there were only four users, so that the machine could be termed lightly loaded.

The results of the benchmark are as follows:

Table 2

Machine	Time
AT-clone :	40.7 sec
VAX 11/785 :	9.2 sec (user's time)
VAX 11/785 :	5.1 sec (CPU time)

Due to the usage level on the VAX when the benchmark was run, it is fairly clear that the user time could have been somewhat better, but it also could have been much worse. Extrapolating from other experience, the execution time for this benchmark on any of the current (1987-88) crop of 32-bit microcomputers would lie somewhere midway between the VAX's user time and CPU time.

Several restrictions of RAYTRACE should be noted. Filenames are to be ten characters or less in length. The input files need to be located in the same directory as the executable for RAYTRACE, at least the way RAYTRACE is currently written. Another restriction is on the size of the input data. Currently, the grid of points cannot be larger than 30 in latitude by 60 in longitude. This trims the size of the memory allocated to a level tolerable to the DOS specifications stated previously. A third restriction is that rays may not be fired with an elevation angle of 90 degrees. This restriction only applies if the ray will make a reflection from the ionosphere. Rays which penetrate the ionosphere may have elevations of +/- 90 degrees.

It is important to make sure when doing a new problem that ALL of the menu items have been selected and checked to verify that their values are properly set. Failure to do so is perhaps one of the most frustrating of mistakes that can be made while using RAYTRACE, as the program is perfectly willing to use whatever the current values are even though they may not be what the user intends. As a further reminder, the following table lists some parameters and their range of values.

Table 3

Parameter	Range
Latitude	+90 to -90 deg with (+) north
Longitude	-180 to +180 deg with (-) for west longitudes
Plasma frequency squared	Megahertz squared
Heights and distances	Kilometers
Elevation angles	+90 (zenith) to -90 (nadir) degrees
Azimuth angles	0 to 360 degrees with 0 north and values increasing eastward

Some of the parameters that the user enters have either imposed or practical bounds. The wave frequency of the RF transmission that the ray represents is bounded on the lower side to a few MHz. This is because for lower frequencies magnetic field effects, which are not included in the current version of RAYTRACE, play an increasing role. As an upper bound, frequencies beyond a few GHz begin to lose reliability for two reasons. First, as a calculational matter, the ratio of the plasma frequency squared to the wave frequency squared becomes very small. Operations occur of the type (1 - x) where x is a small number, with the resulting loss of accuracy. Second, refraction due to the neutral atmosphere becomes a much more important factor at these frequencies. These limits are then limits on the believability of the results. RAYTRACE will calculate over a much larger range of frequencies, but the results will be far from reliable.

The size of the raypath increment is another quantity that has practical limits. There is a lower limit due to accumulation of error and the duration of the calculation. For raypath increments smaller than a few tenths of kilometers, even modest calculations begin to take intolerable lengths of time. This conclusion is of course subjective and depends upon the computer being used at the time. The problem of duration of the calculation has, however, prevented rigorous testing for the degradation of the results due to accumulation of errors. An upper limit also exists on the size of the raypath increment. Increments which are too large begin to violate an approximation made in the theory underlying the raytracing algorithm. Typically, increment sizes of 10-15 km are about the maximum practical. As with the limits on the wave frequency, values outside these may be used, but the results must be viewed as ranging in quality from rough approximations to completely unreliable.

The cutoff height is a quantity that bears further discussion. The default value initially given is essentially the height of geosynchronous orbit. This value represents a compromise. The ionosphere is considered to end at a height of 2000 km in the model used. Propagation to altitudes beyond this is modeled as straight line propagation through free space. While this is certainly not rigorously correct, it is by and large an accurate assumption. Clearly some cutoff must be

used so that the program doesn't calculate off to infinity. Since most radio operations currently take place within a sphere defined by the radius of geosynchronous orbit the default value seems convenient. The value may be reset to any value the user desires. Be warned, however, that the neglected refraction will eventually add up as the path length increases. Depending upon the application, this error could have a substantial effect.

VMS specific comments

The points that are mentioned here have been discussed in greater detail elsewhere and are referred to here as a reminder. When transporting the source code between a DOS-based and a VAX/VMS system, there are a few lines that may need to be changed. These changes are discussed in detail in the Setup section. The naming and renaming of files needs to be carefully considered. The number of results files having similar names but different version numbers can become bewildering. Also, the user should automate as much of the raytracing process as possible through the use of command files.

When using VMS command files to perform batch RAYTRACE jobs, one drawback becomes obvious, there is no provision in such command files for a response that involves just typing a return. Every line of the command file must contain something. For this reason, any prompt of the 'Press RETURN to continue.' type will accept '.' and return as if it were just a plain return. Therefore, on any line of a command file where the desired response is just a return, a line with a single period on it will serve as the necessary substitute.

DOS specific comments

The points that are mentioned here have been discussed in greater detail elsewhere and are referred to here as a reminder. When transporting the source code between a VAX/VMS system and a DOS-based system, there are a few lines that may need to be changed. These changes are discussed in detail in the Setup section. The naming and renaming of files needs to be carefully considered. Because DOS

provides little protection against accidental destruction of files, the user must exercise extreme care to preserve files against disaster. Also, the user should automate as much of the raytracing process as possible through the use of batch files.

Uses and Usage tips

RAYTRACE is a tool that may be used to accomplish a number of tasks in the realm of propagation modeling. It may not be immediately clear, though, how to do some things with the program. The following is a discussion of a few sample uses.

The instructions for operating RAYTRACE discussed the concept of the dimensionality of the problem. From that discussion, it should be clear how to trace single rays, vertical fans of rays, and arrays of rays. A variation on this theme that might prove useful is the horizontal fan of rays. This may be achieved by using a two dimensional problem with only one allowed elevation angle. The mechanics of doing this involve setting the starting and ending elevation values equal (or nearly equal) and entering a step size that is arbitrary (or large compared to the difference between the starting and ending elevations).

Information on the coverage of an area by a broadcast transmitter may be obtained by tracing a two dimensional set of rays. It should be noted that the resulting array of landing points for these rays will not be evenly spaced in latitude and longitude. It is possible, though, to interpolate on such a grid and produce contours or other desired information. At present the only way to achieve (approximately) the desired spacing between points is by experimentation with a small number of rays.

Tracing rays between two specific points (homing) is also a desirable capability. While such a capability is not currently built into RAYTRACE, it is relatively straightforward to do manually. By tracing a pair of rays that are fairly close in elevation, the results for the end points of the pair of rays may be used in a linear interpolation scheme. Typically it requires only five or so iterations to reach the desired convergence on the specific target point. A similar process may be done for the azimuth, although the azimuth is usually

fairly well approximated by that of the great circle path between the two points. An azimuth search will usually also converge more quickly.

It is sometimes desirable to have detailed information along the raypath. There currently exists some code within RAYTRACE to report the height and range at each increment along the ray, for the purpose of later plotting of the raypath. This code is normally commented out, as the files of coordinates produced may become enormous. This code is located in the routine RAYSUB a few lines below line 30300 (consult the source code listing, Appendix D). It merely consists of a WRITE statement and its corresponding FORMAT statement. Uncommenting these lines and recompiling will yield a version of RAYTRACE that will produce an extra generic data file with pairs of values: ground range, height. These values may then be used in a plot of the trajectory of the ray. Certainly, more sophisticated schemes are possible, but such things are best tailored to a given need.

Ultimately, RAYTRACE is extensible. This is most easily achieved by extending the menu system. Adding to the menu system allows more parameters or flags to be entered by the user. Flag values entered in this manner could be used to activate or deactivate additional sections of code. This code could then operate upon existing and new parameters for the calculation of additional results. Details of adjusting the menu system are given in a comment block in the routine DOxDATA, just after line 12002. The menu files are discussed in Appendix A. The only other requirement is the addition of the new features or calculations at the appropriate locations in the existing program. This may only be determined by study of both RAYTRACE and the theoretical groundwork for the intended addition.

MENU QUICK REFERENCE

The three menus of RAYTRACE are presented here, with brief explanations of most menu items. They are given one per page, so that the user can easily photocopy them for handy reference. There is also ample room for notes. One suggestion, if a copy machine that reduces is available, is to copy the three menus reduced, then paste them on one sheet. Copying this sheet will give a single page "cheat sheet".

MAIN MENU

1 - Edit launch parameters (Go to Launch Parameters menu)

2 - Edit elevation/azimuth parameters (Go to Elevation Parameters menu)

3 - Proceed with raytracing

4 - Quit program

LAUNCH PARAMETERS MENU

1 - Bounce limit	(limit number of earth impacts)
2 - Signal intensity	(flag to turn on/off intensity loss calculation)
3 - Conductivity	(select type of earth surface for reflection loss)
4 - Launch point	(latitude/longitude location of start point)
5 - Launch height	(altitude of start point)
6 - Range & height limits	(set great circle ground range and altitude cutoffs)
7 - Ray path increment	(set step size along the ray)
8 - Wave frequency	(frequency of transmission modeled by the ray)
9 - Done	(return to the main menu)

ELEVATION PARAMETERS MENU

1 - Dimension of problem	(determine multi-ray launch pattern)
2 - Starting azimuth	(set initial launch azimuth)
3 - Starting elevation	(set initial launch elevation)
4 - Elevation resolution	(set elevation step size for multi-ray calculations)
5 - Azimuth resolution	(set azimuth step size for multi-ray calculations)
6 - Elevation limit	(set limiting value on elevation for multi-ray calculations)
7 - Azimuth limit	(set limiting value on azimuth for multi-ray calculations)
8 - Done	(return to main menu)

APPENDICES

Appendix A:
File Formats

GRID.DAT

The grid file is a means by which ionospheric information may be input into RAYTRACE. The grid file is generally produced externally to RAYTRACE, either by use of an ionospheric model program or by processing of actual ionospheric data. The file contains the pertinent ionospheric information at a gridded set of latitudes and longitudes. Grids may be entered from the console, but this is only practical for relatively small grids.

Grid files are composed of a number of data arrays. First, the definition of the latitude and longitude grid is recorded. Following this, the six ionospheric parameters required for specification of the vertical profile are given for each point on the grid. The six parameters are the maximum plasma frequencies for the E, F1, and F2 layers, the heights for the maximum frequencies for the F1 and F2 layers, and the semithickness of the F2 layer. The choice of these parameters is determined by the profile model that is built into the current version of RAYTRACE. Finally, the date, time, and sunspot number are recorded.

The array of values that serves to define the latitude and longitude grid is a real array containing six parameters. They are the latitude and longitude spacing for the grid, followed by the initial latitude and longitude values for the grid. The starting point is taken as the southwest corner of the grid. Values are in degrees and longitude is given as east of Greenwich positive, west negative. This gives longitudes running from -180 to +180 degrees. The last two values in the array are the number of points that the grid is to contain in the latitude direction and in the longitude direction respectively. The grid must be rectangular and if necessary should be either padded with more data (if the grid is being generated from a model or interpolation

of actual data) or some points eliminated so that it becomes rectangular.

The arrays of ionospheric data are written as two dimensional arrays with the first index corresponding to latitude and the second to longitude. The indices represent the integer count of grid points traversed in the given direction to reach the coordinates of the grid point. The first array contains the peak plasma frequency squared values for the E-layer, for the entire grid. The second array holds the height of the peak of the F1-layer and the third holds the plasma frequency squared values for the F1 maximum. The last three arrays contain the values which are used to specify the F2-layer. They are, in order of appearance, the semithickness, the height of the layer maximum, and the plasma frequency squared at the maximum.

The final array is an integer array containing the time, date, and sunspot number for which the problem is being run. These values are contained in the array in this order: sunspot number, year, month, day, hour (UT), and minute. The current configuration of RAYTRACE no longer uses this information directly as the portions of code that use it have been commented out. They have been left in both for the possibility of future calculational use, and so that the information may be included in displays or reports if desired.

The format of the GRID.DAT type of data file is perhaps best illustrated by including fragments of code that RAYTRACE uses to read information from the grid file. The fragmented listing (Listing 1 includes the declaration statements. The file is written unformatted to save time and space. The filename is set either as a default value or from a prompt to the user. The correspondence between the arrays mentioned above and those in Listing 1 is fairly obvious. The variables are named according to the convention that is presented with the source code listing in Appendix D.

Listing 1

```
      INTEGER IVxPARAM(6)
C
      REAL*8 RVxGRID(6)
      REAL*8 RVxION1(30,60), RVxION2(30,60), RVxION3(30,60)
      REAL*8 RVxION4(30,60), RVxION5(30,60), RVxION6(30,60)
C
      CHARACTER*10 KVLxFIL

                              *
                              *
                              *

      OPEN(20, FILE=KVLxFIL,FORM='UNFORMATTED')
      REWIND (20)
      READ(20) RVxGRID
      READ(20) ((RVxION1(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
      READ(20) ((RVxION2(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
      READ(20) ((RVxION3(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
      READ(20) ((RVxION4(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
      READ(20) ((RVxION5(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
      READ(20) ((RVxION6(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
      READ(20) IVxPARAM
      CLOSE(20)
```

RAYDAT.DAT

The RAYDAT.DAT type of file is used by RAYTRACE for the storage and retrieval of both the ionospheric information and parameters of the problem that the user has entered. For the sake of simplicity, the ionospheric information is stored in exactly the same manner as in the case of GRID.DAT type files. Arrays containing the other problem parameters are then added in after the ionospheric values. As a consequence of this file structure, only the arrays that are not repeated from GRID.DAT need be discussed.

The first of these arrays is an integer array containing three values. The bounce limit is stored first. This sets a bound on the number of times that a ray may impact the earth, which is useful in some cases dealing with subionospheric raypaths. Next is a flag that determines whether or not signal loss calculations will be performed.

Finally, there is a flag that determines what sort of surface will be used in reflection loss calculations for earth impacts of the ray.

The second array is a real array containing the latitude and longitude of the launch point. This is followed in the array by the range limit. This parameter sets a bound on the great circle ground range that the ray is allowed to propagate to. The fourth value in the array is the size of the raypath increment. Fifth is the wave frequency. The sixth value is the starting height of the ray above the earth's surface. Finally, the seventh value is the altitude cutoff value.

The final array is a real array that contains the information necessary to specify the elevations and azimuths at which rays will be sent out. The first parameter is the dimensionality of the problem. This is defined by considering the aim points of the ray in an elevation versus azimuth plane. A single ray is a single point, and so has dimension zero. A fan of rays having a particular azimuth forms a line of points, and is considered dimension one. Lastly, a set of such fans produces an array of points on the plane and is therefore considered as dimension two. After the dimensionality of the problem comes the initial azimuth and the initial elevation. Following these values, the azimuth and elevation spacing between rays is stored. Last of all, the final azimuth and elevation values are stored.

As above, the easiest way to make the structure of the data file manifest is to present the fragments of code responsible for writing to the file. The code is presented in Listing 2. Again, the filename is set previously to its proper value. The additional arrays are IVxLAUN, RVxLAUN, and RVxOPTION.

Listing 2

```
      INTEGER IVxPARAM(6), IVxLAUN(3)
C
      REAL*8 RVxGRID(6), RVxLAUN(7)
      REAL*8 RVxOPTION(7)
      REAL*8 RVxION1(30,60), RVxION2(30,60), RVxION3(30,60)
      REAL*8 RVxION4(30,60), RVxION5(30,60), RVxION6(30,60)
C
      CHARACTER*10 KVxDFIL

                                  *
                                  *
                                  *

      OPEN (30, FILE = KVxDFIL, FORM='UNFORMATTED')
      REWIND (30)
      WRITE (30) RVxGRID
      WRITE (30) ((RVxION1(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
      WRITE (30) ((RVxION2(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
      WRITE (30) ((RVxION3(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
      WRITE (30) ((RVxION4(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
      WRITE (30) ((RVxION5(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
      WRITE (30) ((RVxION6(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
      WRITE (30) IVxPARAM
      WRITE (30) IVxLAUN
      WRITE (30) RVxLAUN
      WRITE (30) RVxOPTION
      CLOSE(30)
```

FOR###.DAT

This file is the results file for RAYTRACE. The ### represents a number that is assigned sequentially during the course of a given ray-tracing session so that the different runs during the session may be distinguished. Because results files may need to be of greater portability, the results are written to disk in formatted form. The results are stored by first saving the information necessary to determine the number of data points represented in the file. Data points consist of earth bounce points, and the terminal point on the ray. The azimuth and elevation of a particular ray are saved and then the array

containing the information from all the data points on that ray is stored. This then repeats until the data from all rays has been saved.

The number of data points per ray is determined from the maximum number of bounces allowed by the bounce limit, plus one for the terminal point of the ray. The proper number of rays are then accessed by using the number of azimuths and the number of elevations recorded. These set the limits on two loops, within which the file is accessed.

The array of data for a given ray is two dimensional. The first index is the number of the data point along the ray. The second represents the particular data element for that point. The meaning of these elements is given in a comment block in the source code for the subroutine RAYSUB, just after the variable declarations. The ones that are currently used are as follows. First are the latitude and the longitude. The third element is the group time delay along the ray to that point. The great circle ground range is next, followed by the signal loss in dB, due to spatial loss, focusing or defocusing, and possibly reflection loss. The sixth element contains the amount that the ray has deviated from a great circle path at that point. The ninth element of the array gives the height of the ray at the data point. Element ten contains the phase path length along the ray to that point. The next three elements give the x, y, and z direction cosines of the ray at that point. Here, the x, y, and z directions are in the coordinate system where x is south, y is east, and z is up.

The sections of code for writing the results out to the file have been excerpted from the main routine of RAYTRACE. The code is presented in Listing 3. Also presented is the code that produces the number associated with the file. The integer IVxIND is initialized to 40 and incremented every time a particular run is complete. It is then used to produce the filename of the results file.

Listing 3

```
      INTEGER IVxIND, IVxLAUN(3), IVxAZLIM, IVxELLIM
C
      REAL*8 RVxAZI, RVxELEV, RVxBOUN(11,15)
C
      CHARACTER*2 KVxIND
      CHARACTER*10 KVxOFIL

                                    *
                                    *
                                    *

      WRITE (KVxIND,'(I2)') IVxIND
      KVxOFIL = 'FOR0'//KVxIND//'.DAT'

                                    *
                                    *
                                    *

      OPEN (IVxIND,FILE = KVxOFIL)
      WRITE (IVxIND, 22000) IVxLAUN(1)+1, IVxAZLIM, IVxELLIM

                                    *
                                    *
                                    *

      WRITE(IVxIND, 21003) RVxAZI, RVxELEV

                                    *
                                    *
                                    *

      WRITE(IVxIND, 21003) ((RVxBOUN(L,K),K=1,15), L=1,IVxLAUN(1)+1

                                    *
                                    *
                                    *

      CLOSE(IVxIND)
21003 FORMAT (G24.14)
22000 FORMAT (I4)
```

MENU FILES

The scheme for using and presenting the menus and menu files is detailed in the source code listing in subroutine IOxMENU. The method is reviewed here. The file MASTER.MEN is used to maintain a list of the rest of the menu files. It is merely a text file that contains the names of the other menu files, with each filename on a separate line. RAYTRACE is currently set up to handle up to ten menu files in addition to the master file. The filenames are ordered according to their numbering within RAYTRACE. Filenames are currently configured to be no more than ten characters long.

The other menu files contain the actual information that is displayed on the console as a menu. As with MASTER.MEN, the other .MEN files are just text files. The information needed for the display of a menu is as follows. The first line has two items that are separated by a comma. These are the number of items in the menu, followed by the title of the menu. The number of items is an integer. All of the text information in these files is read in using an A40 format, and so may be no longer than 40 characters long. After this first line, the menu items are given. The text that will be displayed for each item is put in the file with one menu item's text per line. Again, these are to be no longer than 40 characters. The last line of the file contains the text for the prompt that will be displayed at the end of the menu. This information is accessible so that the menu text may be customized at will by the user with just an ordinary text editor.

A more complete understanding of how this all works will only be obtained by examining the source code of subroutine IOxMENU. In particular, this routine should be examined very closely if any extensions to RAYTRACE are to be made. Any additional information that must be entered ought to be entered by means of a menu so that the user interface of the program will remain consistent. Examination of the existing menu files will also be of assistance.

Appendix B:
Inventory of COMMON blocks

This section lists the COMMON blocks used by RAYTRACE, along with the declarations of the variables involved. An explanation is given for some of the variables. A complete explanation of all the variables involved would require describing significant portions of the algorithms used, which is beyond the scope of this User's Guide.

MAINDAT:

This block contains the contents of RAYDAT.DAT. As may be seen from the EQUIVALENCE statements, the array RVxIONPT holds the six sets of ionospheric parameters input from the data file.

```
      INTEGER IVxPARAM(6), IVxLAUN(3)

      REAL*8 RVxGRID(6), RVxIONPT(30,60,6), RVxLAUN(7), RVxOPTION(7
      REAL*8 RVxION1(30,60), RVxION2(30,60), RVxION3(30,60)
      REAL*8 RVxION4(30,60), RVxION5(30,60), RVxION6(30,60)

      EQUIVALENCE (RVxION1,RVxIONPT),(RVxION2,RVxIONPT(1,1,2))
      EQUIVALENCE (RVxION3,RVxIONPT(1,1,3))
      EQUIVALENCE (RVxION4,RVxIONPT(1,1,4))
      EQUIVALENCE (RVxION5,RVxIONPT(1,1,5))
      EQUIVALENCE (RVxION6,RVxIONPT(1,1,6))

      COMMON /MAINDAT/ RVxGRID, RVxIONPT, IVxPARAM, IVxLAUN,
     * RVxLAUN, RVxOPTION
```

RESULTS:

The RESULTS common block contains the array which holds the data for every data point on a particular ray. For each ray that is traced, this data array is recorded. Refer to the discussion above concerning the FOR###.DAT file format for further detail on this array.

```
      REAL*8 RVxBOUN(11,15)

      COMMON /RESULTS/ RVxBOUN
```

LPARM:

```
      INTEGER IVxSSNUM, IVxTIME(5), IFxGRND
      REAL*8 RVxYEAR, RVxA100, RVxTIM
      COMMON /LPARM/ IVxSSNUM,IVxTIME,RVxYEAR,RVxA100,IFxGRND
     *,RVxTIM
```

LOSSES:

This block holds the various components of signal loss that are computed by RAYTRACE. RVxLOSA is the absorption loss. The code to calculate this is based upon empirical formulae which were appropriate to use at earlier stages of the development of RAYTRACE, but cannot be used for general paths. This code is currently commented out. RVxLOSX has suffered a similar fate. This was used to represent the so-called "excess loss" that some earlier ionospheric models use. This value is inappropriate, though, for use in examining specific raypaths in that it becomes possible for two relatively adjacent rays to have anomalously different losses. The remaining two values are actually used if signal intensity calculations are to be performed at all. RVxLOSR gives the reflection loss, and RVxLOSG gives the geometric loss. The geometric loss is the combination of the effects of distance from the source, and also focusing/defocusing effects.

```
      REAL*8 RVxLOSA, RVxLOSR, RVxLOSX, RVxLOSG
      COMMON /LOSSES/ RVxLOSA, RVxLOSR, RVxLOSX, RVxLOSG
```

PRAM:

The values in this block are, in order: pi, a conversion factor to go from degrees to radians, the radius of the earth, and the height considered to be the top of the ionosphere.

```
      REAL*8 RPxPI,RPxDTOR,RPxREARTH,RPxHTOP
      COMMON /PRAM/ RPxPI, RPxDTOR, RPxREARTH, RPxHTOP
```

SCPS1:

This block contains information on the number and location of the grid points. ICxNSCP is the number of grid points, and the other two are arrays containing the latitudes and longitudes, respectively, of the grid points.

```
INTEGER ICxNSCP

REAL*8 RVxLATSC(1800),RVxLONSC(1800)

COMMON /SCPS1/ ICxNSCP, RVxLATSC, RVxLONSC
```

SCPS1A:

These arrays contain the six ionospheric parameters for all the grid points, with all the layer peak plasma frequency squared values in one array and all the height (and semithickness) values in the second.

```
REAL*8 RVxFNSQ(1800,3), RVxH(1800,3)

COMMON /SCPS1A/ RVxFNSQ,RVxH
```

OTHER:

```
REAL*8 RVxLAT1, RVxLON1, RVxHBOT
REAL*8 RVxFSQU, RVxRPI, RVxHCUT

COMMON /OTHER/ RVxLAT1,RVxLON1,RVxHBOT,RVxFSQU,RVxRPI,RVxHCUT
```

MISC:

```
REAL*8 RVxSEZGEC(3,3), RVxX7, RVxY7, RVxZ7
REAL*8 RVxR1, RVxR2, RVxHMIN

COMMON /MISC/ RVxSEZGEC,RVxX7,RVxY7,RVxZ7,RVxR1,RVxR2,RVxHMIN
```

START:

This block contains the initial position information for the raypath increment.

```
REAL*8 RVxXI, RVxYI, RVxZI, RVxLATI, RVxLONI

COMMON /START/ RVxXI, RVxYI, RVxZI, RVxLATI, RVxLONI
```

END:

This block contains information on the end of the raypath increment.

```
REAL*8 RVxH5, RVxXF, RVxYF, RVxZF

COMMON /END/ RVxXF, RVxYF, RVxZF, RVxH5
```

IONO1:

This block stores some ionospheric parameters.

```
REAL*8 RVxALPHA, RVxBETA, RVxETA1, RVxETA2
REAL*8 RVxFNSB(3), RVxHB(3)

COMMON /IONO1/ RVxALPHA,RVxBETA,RVxETA1,RVxETA2,RVxFNSB,RVxHB
```

IONO2:

This block stores ionospheric interpolation results.

```
REAL*8 RVxB5(3), RVxB6(3), RVxA5(3), RVxA6(3)
REAL*8 RVxE5(3), RVxE6(3)

COMMON /IONO2/ RVxA5, RVxA6, RVxB5, RVxB6, RVxE5, RVxE6
```

IONO3:

This holds still other quantities used in the ionospheric calculations.

```
INTEGER IFxCASE
REAL*8 RVxV1, RVxV2, RVxXX, RVxHBND
COMMON /IONO3/ RVxV1,RVxV2,RVxXX,IFxCASE, RVxHBND
```

MORE:

```
INTEGER IVxSX, IVxSY, IVxSZ
REAL*8 RVxCX, RVxCY, RVxCZ
COMMON /MORE/ IVxSX, IVxSY, IVxSZ, RVxCX, RVxCY, RVxCZ
```

TEMP1:

```
REAL*8 RVxF40, RVxF65, RVxK1, RVxHL
COMMON /TEMP1/ RVxF40, RVxF65, RVxK1, RVxHL
```

GORP:

```
REAL*8 RVxDD6, RVxANGLIM
COMMON /GORP/ RVxDD6, RVxANGLIM
```

TEMP2:

These are indices use in accessing grid arrays for interpolation.

```
INTEGER IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3, IFCxN4
COMMON /TEMP2/ IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3, IFCxN4
```

```
TEMP3:

        REAL*8 RTLxD5, RTLxI3, RTLxI4, RTLxI5
        COMMON /TEMP3/ RTLxI3, RTLxI4, RTLxI5, RTLxD5

VAR1:

        REAL*8 RVCxXL, RVCxXU, RVCxA0, RVCxHU. RVCxB0
        COMMON /VAR1/ RVCxXL, RVCxXU, RVCxHU, RVCxA0, RVCxB0

VAR2:

        REAL*8 RVCxHB1, RVCxH2, RVCxYS, RVCxSL1, RVCxSL2, RVCxH1L
        COMMON /VAR2/ RVCxHB1,RVCxH2,RVCxYS.RVCxSL1,RVCxSL2,RVCxH1L

VAR3:

        REAL*8 RVCxA1, RVCxB1, RVCxC1, RVCxH1P, RVCxH2P
        COMMON /VAR3/ RVCxA1, RVCxB1, RVCxC1, RVCxH1P, RVCxH2P

VAR4:

        REAL*8 RVCxA2, RVCxB2, RVCxC2, RVCxHT3, RVCxHT4, RVCxHT5
        COMMON /VAR4/ RVCxA2,RVCxB2,RVCxC2,RVCxHT3,RVCxHT4,RVCxHT5

RAID:

        INTEGER IFCxSN
        COMMON /RAID/ IFCxSN
```

```
TEMP4:

        REAL*8 RTLxF3, RTLxF1, RTLxF2
        COMMON /TEMP4/ RTLxF1, RTLxF2, RTLxF3


LOCAL:

        REAL*8 RVLxELEV
        COMMON /LOCAL/ RVLxELEV
```

Appendix C:
Error Messages

RAYTRACE will detect a few errors internally and generate error messages for these. These are in addition to those that may be generated as run-time errors as specified by the compiler being used, or may be operating system level errors. This section will discuss the error messages generated directly by RAYTRACE.

INVALID CHOICE.

This message indicates that a bad choice has been made from one of the menus. A bad choice is one that is out of the range of the possible selections.

HIT RETURN ONLY TO CONTINUE.

The routine that handles the 'Press RETURN to continue.' prompt will generate this message if anything other than an 'A', '.', or return is typed.

Conversion ERROR, Please RETRY Input

Any given prompt is expecting a particular type of response (REAL, INTEGER, etc.) and this message will be given if the value entered is not of the expected type. Note that if a REAL is expected, and an integer is entered, this will not occur, as a decimal point may be attached to the value.

RAY DOES NOT PROPAGATE!!!!

This message indicates that a ray has been given a starting point at an altitude inside the ionosphere and has parameters that are inconsistent with any propagation for the ray.

EXCEEDINGLY WIERD CASE: POINT #1 (& a few values are also displayed)

If this message ever appears, something has gone quite wrong in the routine which calculates the ionospheric parameters and gradients. Theoretically, this should never be seen. Copy down ALL the values given, along with all other parameters from the menus and the ionospheric specification (if possible). Contact the authors of RAYTRACE with this information.

EXCEEDINGLY WIERD CASE: POINT #2

This message is very similar in nature to the previous one. It, too, should never occur. If it does, however, follow the instructions given for the similar error above.

Appendix D:
Source Code Listing

APPEARS AFTER INDEX

References

[1] Reilly, M.H. and E.L. Strobel, "Efficient ray-tracing through a realistic ionosphere," to be published in Radio Sci., 1988.

[2] Thomason, J., G. Skaggs, and J. Lloyd, "A global ionospheric model," NRL Report 8321 (AD-000-323), August 1979.

Index

RAYTRACE USER'S GUIDE

APPENDIX D:

SOURCE CODE LISTING

Table of Contents

Page

RPHASE.FOR

RTILT.FOR

RMISC.FOR

Naming Convention

Variables in the source code are generally named according to the following convention. In some cases, variables do not fall neatly into an appropriate category and the naming necessarily becomes a bit arbitrary. Cases that do not follow this convention at all are ones contained in code that has been obtained from other sources. Names are defined by:

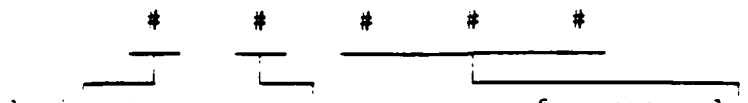
(2 or 3-char prefix)			(separator char)	(mnemonic suffix)
(Type)	(Purpose)	(Context)	x	(Mnemonic)

Type:	I	INTEGER*4
	J	INTEGER*8
	F	REAL*4
	R	REAL*8
	C	COMPLEX*16
	K	CHARACTER
	L	LOGICAL
Purpose:	C	Constant
	V	Variable
	T	Temporary
	P	Parameter
	F	Flag
Context:	none	in MAIN program
	L	local to a subprogram
	S	passed as an argument
	C	passed in a COMMON block

EXAMPLES: **RTLxH** == Real*8, temporary, local variable H
IVxELLIM == Integer*4, variable ELLIM in MAIN

Line Numbering Convention

Line numbers are generally designated as follows:



Flag Variables

IFxEXI:
1 indicates user's choice to quit the program,
0 otherwise

IFxNEW:
1 indicates a new problem is being done,
0 otherwise

IFxSTAY:
1 indicates that the user is continuing a problem,
0 otherwise

IFxBUN:
# of extra ray in ray bundle for calculation of intensity

IFxGEN:
1 = no special condition exists
2 = ray increment has been adjusted so that a layer boundary is not overshot
3 = a z-reflection has occurred
4 = an x-reflection has occurred
5 = a y-reflection has occurred
6 = ray is traveling downward in free space propagation
7 = ray is traveling upward in free space propagation

IFxN4:	1 if the present location is not interior to a triangle of grid points for interpolation 2 otherwise
IFxGRND:	1 if reflection loss calculations are to use soil's conductivity 2 if water's conductivity is to be used
IFxSFL:	0 to activate signal intensity calculations 1 to deactivate them
IFLxOUT:	1 if the present location is outside the grid of points 0 otherwise

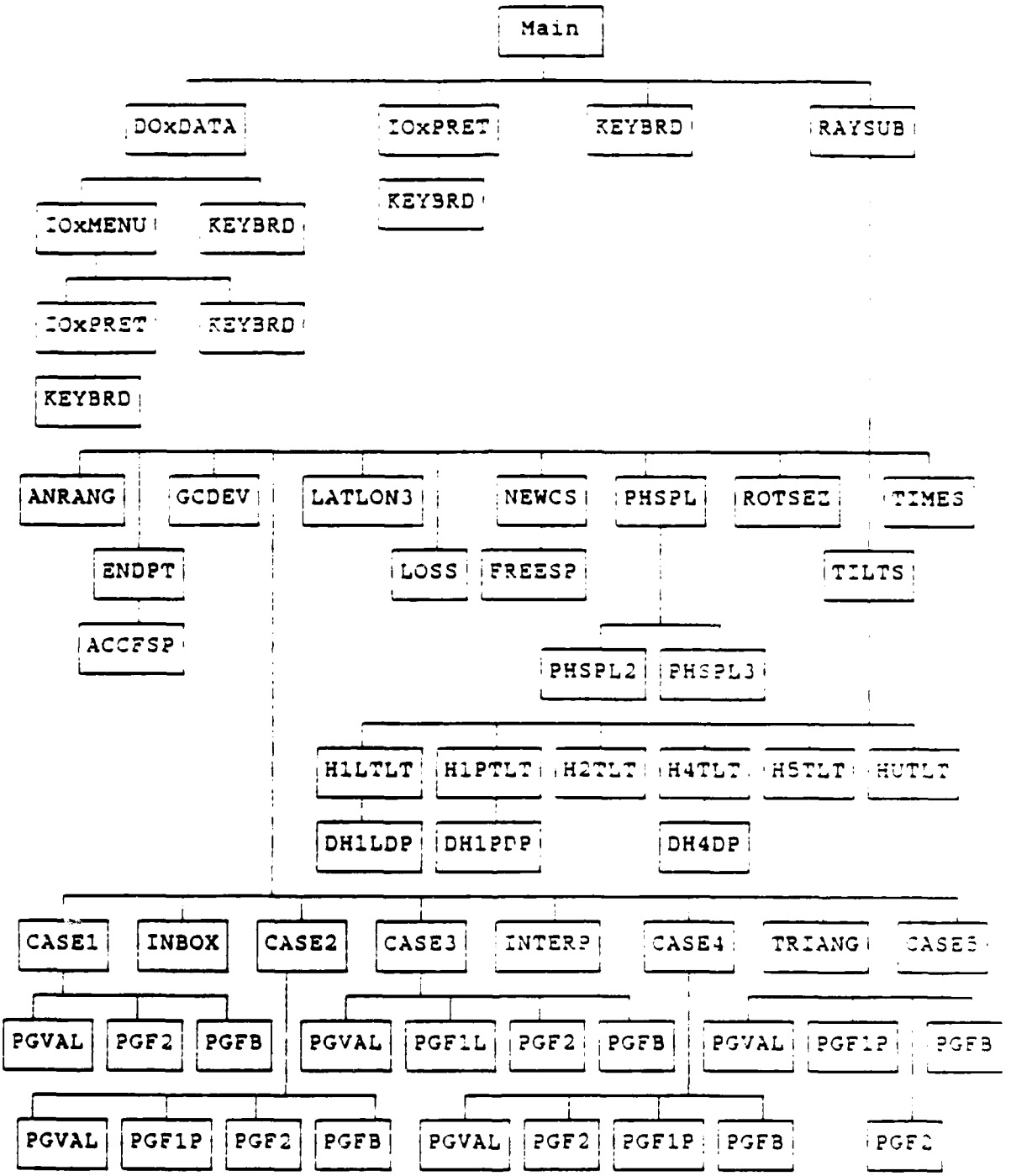
Notes

There will be discrepancies in the consecutive numbering of source code lines within a source code file, as the blank lines between subroutines have been skipped. Also, the consecutive numbering runs to the end of a particular source file, starting over for the next file.

The symbol table after each subroutine contains a column labeled "Class". This corresponds to the portion of the variable prefix denotes as "Context" in the naming convention above. The notable difference being that a Class of 'param' is the same as a prefix of 'S', meaning that the quantity was passed as an argument.

Also in the symbol table, the character 'x', which is used under the variable naming convention to connect the prefix with the mnemonic suffix, has been converted to an upper case letter, 'X'.

Hierarchy of Routines



```
Line#  Source Line

     1          PROGRAM RAYTRACE
     2  C
     3  C -------------------------------------------------------
     4  C
     5  C   RAYTRACE --  3D RAYTRACING WITH ACCELERATED FREE SPACE
     6  C                 PROPAGATION CALCULATION
     7  C
     8  C -------------------------------------------------------
     9  C
    10  C   AUTHORS:         MICHAEL H. REILLY & ERIC L. STROBEL
    11  C
    12  C   DATE:            03/18/88
    13  C
    14  C   VERSION:         4.3
    15  C
    16  C -------------------------------------------------------
    17  C
    18  C   REVISED:       07/25/86 -- Initial revision. Translated
    19  C                  from Tektronix Basic to VAX Fortran by
    20  C                  Eric L. Strobel.
    21  C
    22  C                  07/30/86 -- V1.1.  Change over to use of
    23  C                  REAL*8 precision in the calculations.
    24  C
    25  C                  08/12/86 -- V2.0.  Add the capability to
    26  C                  perform propagation loss calculations.
    27  C                  Also greatly massaged over the output.
    28  C                  A number of minor and MAJOR bugs corrected.
    29  C
    30  C                  09/05/86 -- V2.3.  Add horizon focusing and
    31  C                  handle low angle rays.  Record data for
    32  C                  proper handling of skip focusing.  Record
    33  C                  maximum height.  Calculate deviation from
    34  C                  great circle path.  Minor bug fixes.
    35  C
    36  C                  09/17/86 -- V2.4.  Fix calculation of great
    37  C                  circle path deviation.  Introduce cutoff in
    38  C                  number of bounces.  Automagically obtain
    39  C                  approximate bottom of the ionosphere, elim-
    40  C                  inating the need for the prompt.  Also fixed
    41  C                  rotation matrix used in calculating the new
    42  C                  c-values.
    43  C
    44  C                  10/09/86 -- V3.0.  Lots of changes.  Read &
    45  C                  write to files.  User-friendly means of
    46  C                  getting information into the program.  The
    47  C                  old Raytrace is now a subroutine.  Capability
    48  C                  to do many rays at a time.  Able to handle
    49  C                  large ionospheric specifications.  Able to
    50  C                  change problem parameters without restarting
```

```
Line#  Source Line

   51  C                  the program.
   52  C
   53  C                  04/09/87 -- V3.1.  Corrected error in loss
   54  C                  calculation.
   55  C
   56  C                  09/01/87 -- V4.0.  Major revision.  Now uses
   57  C                  the RADAR-C ionosphere.  Able to launch rays
   58  C                  inside and above the ionosphere.  Calcs. & rpts.
   59  C                  the direction cosines now.  Uses new routine to
   60  C                  obtain user info.  Absorption & excess loss calcs.
   61  C                  commented out.  Geometric loss now done correctly.
   62  C                  Phase path calculated. Data files now BINARY
   63  C                  (vs. ASCII).  Upper bound at radius of GEO dist.
   64  C                  Derivatives of range w.r.t. elev. are incorrect,
   65  C                  but unfixed at this point.  WARNING: Cannot send
   66  C                  rays arbitrarily close to vertical!!!  C-values
   67  C                  are now properly normalized.
   68  C
   69  C                  09/30/87 -- V4.1.  Phase path length calculations
   70  C                  added.  Tilt routines changed to conform to the
   71  C                  requirements of using the RADAR-C model.
   72  C
   73  C                  01/15/88 -- V4.2.  Bug fixes and minor corrections
   74  C                  made over a period of time.  Added feature: will
   75  C                  now stop the ray at the designated range, not at
   76  C                  the end of the next increment, as it previously
   77  C                  did.  Also, some disused variables have been
   78  C                  weeded out and some arrays which were larger than
   79  C                  currently necessary have been redimensioned.  The
   80  C                  code is now smaller and it runs faster.  The source
   81  C                  code is now somewhat universal, in that only the
   82  C                  slightest of changes is needed to convert from
   83  C                  VAX to DOS or back, or between various DOS Fortrans.
   84  C
   85  C                  03/18/88 -- V4.3.  A number of corrections and
   86  C                  improvements have been made.  Some elevation and
   87  C                  azimuth options were being reset when they shouldn't
   88  C                  have been (under some circumstances).  Angular range
   89  C                  is now correctly calculated.  Before, it was based on
   90  C                  the launch azimuth, which isn't correct if the ray
   91  C                  bends.  Now, the actual azimuth between the two points
   92  C                  is calculated and used.  Also, the actual angular
   93  C                  range, and not the cosine of the angular range, is
   94  C                  used for the range cutoff.
   95  C
   96  C ------------------------------------------------------
   97  C
   98  C                   Performs 3-D raytracing of radio propaga-
   99  C         tion through the ionosphere.  Incorporates a spec-
  100  C         ific model for true height profiles of electron
```

```
Line#  Source Line

  101  C        density.  Can utilizes data from ionospheric sounders.
  102  C
  103  C -------------------------------------------------------
  104  C
  105           INTEGER ITxQ, IFxNEW, IFxSTAY, IFxEXI, IVxIND
  106           INTEGER IVxAZLIM, IVxELLIM, IVxPARAM(6), IVxLAUN(3)
  107           INTEGER IVxVAR, IVxTYP
  108  C
  109           REAL*8 RVxGRID(6), RVxIONPT(30,60,6), RVxLAUN(7)
  110           REAL*8 RVxOPTION(7), RVxAZI, RVxELEV
  111           REAL*8 RVxBOUN(11,15), RVxVAR
  112           REAL*8 RVxION1(30,60), RVxION2(30,60), RVxION3(30,60)
  113           REAL*8 RVxION4(30,60), RVxION5(30,60), RVxION6(30,60)
  114  C
  115           CHARACTER*10 KVxDFIL, KTxTMP, KVxOFIL
  116           CHARACTER*2 KVxIND
  117           CHARACTER*1 KTxANS, CH
  118           CHARACTER*40 KVxPMT
  119  C
  120           COMMON /MAINDAT/ RVxGRID, RVxIONPT, IVxPARAM, IVxLAUN,
  121          * RVxLAUN, RVxOPTION
  122           COMMON /RESULTS/ RVxBOUN
  123  C
  124  C         Stuff the big array with pieces of more manageable size.
  125  C           This is in order to compensate for the unfortunate DOS
  126  C           restrictions on record sizes.
  127  C
  128           EQUIVALENCE (RVxION1,RVxIONPT),(RVxION2,RVxIONPT(1,1,2))
  129           EQUIVALENCE (RVxION3,RVxIONPT(1,1,3))
  130           EQUIVALENCE (RVxION4,RVxIONPT(1,1,4))
  131           EQUIVALENCE (RVxION5,RVxIONPT(1,1,5))
  132           EQUIVALENCE (RVxION6,RVxIONPT(1,1,6))
  133  C
  134  C -------------------------------------------------------
  135  C
  136  C        Data setup section
  137  C
  138  C -------------------------------------------------------
  139  C
  140           KVxPMT = ' '
  141           IVxTYP = 0
  142           ITxQ = 0
  143           IVxVAR = 0
  144           RVxVAR = 0.0
  145           KTxTMP = ' '
  146           IVxIND = 40
  147  10000    IVxELLIM = 0
  148           IVxAZLIM = 0
  149           RVxAZI = 0.0D00
  150           RVxELEV = 0.0D00
```

```
Line#  Source Line

  151  C
  152  C       IFxSTAY = 1 indicates that the user has responded with
  153  C         'yes' to the query about performing another problem.
  154  C
  155           IF (IFxSTAY.EQ.1) GO TO 20000
  156           WRITE (6,11001)
  157  10005    KVxPMT = 'Is this a new (1) or old (0) problem? '
  158  C
  159  C       A new problem is one for which no RAYDAT.DAT or similar
  160  C         file exists.  Such a file does exist for an old problem
  161  C         and contains the user inputs from a previous run.
  162  C
  163           CALL KEYBRD(KVxPMT,1,ITxQ,IFxNEW,RVxVAR,KTxTMP)
  164           IF (ITxQ.LT.1.OR...NOT.(IFxNEW.EQ.1.OR.IFxNEW.EQ.0)
  165          * GO TO 10005
  166  C
  167  C ---------------------------------------------------------------
  168  C
  169  C       Section to handle the set up of the problem's data
  170  C         by reading from file and editing some of the
  171  C         data.
  172  C
  173  C ---------------------------------------------------------------
  174  C
  175           IF (IFxNEW.EQ.1) THEN
  176  C
  177  C ---------------------------------------------------------------
  178  C       IFxNEW = 1    -->  new problem, so DOxDATA to get the user
  179  C          options and ionospheric data, then write out to
  180  C          a datafile.
  181  C ---------------------------------------------------------------
  182  C
  183               WRITE (6,11004)
  184               KVxDFIL = 'RAYDAT.DAT'
  185               KVxPMT = 'information (10 char max):'
  186  C
  187  C       Ask the user where to store the problem's information.
  188  C
  189               CALL KEYBRD(KVxPMT,3,ITxQ,IVxVAR,RVxVAR,KVxDFIL,
  190               OPEN (30, FILE=KVxDFIL, FORM='UNFORMATTED',STATUS='NEW'
  191               CALL DOxDATA(IFxSTAY, IFxNEW, IFxEXI,
  192               REWIND (30)
  193               WRITE (30) RVxGRID
  194               WRITE (30) ((RVxION1(I,J),J=1,RVxGRID(6)),I=1,RVxGRID 5
  195               WRITE (30) ((RVxION2(I,J),J=1,RVxGRID(6)),I=1,RVxGRID 5
  196               WRITE (30) ((RVxION3(I,J),J=1,RVxGRID(6)),I=1,RVxGRID 5
  197               WRITE (30) ((RVxION4(I,J),J=1,RVxGRID(6)),I=1,RVxGRID 5
  198               WRITE (30) ((RVxION5(I,J),J=1,RVxGRID(6)),I=1,RVxGRID 5
  199               WRITE (30) ((RVxION6(I,J),J=1,RVxGRID(6)),I=1,RVxGRID 5
  200               WRITE (30) IVxPARAM
```

```
Line#  Source Line

  201              WRITE (30) IVxLAUN
  202              WRITE (30) RVxLAUN
  203              WRITE (30) RVxOPTION
  204              CLOSE(30)
  205  C
  206  C       IFxEXI = 1 indicates that the user has chosen to quit the
  207  C         program.
  208  C
  209              IF (IFxEXI.EQ.1) GO TO 20700
  210            ELSE IF (IFxNEW.EQ.0) THEN
  211  C
  212  C -------------------------------------------------------------
  213  C        IFxNEW = 0  -->  old problem, so read from the datafile
  214  C                then DOxDATA to check for any changes the user
  215  C                wants to make to the options.
  216  C -------------------------------------------------------------
  217  C
  218              WRITE (6,11004)
  219              KVxDFIL = 'RAYDAT.DAT'
  220              KVxPMT = 'information (10 char max):'
  221  C
  222  C       Ask the user where to store the problem's information.
  223  C
  224              CALL KEYBRD(KVxPMT,3,ITxQ,IVxVAR,RVxVAR,KVxDFIL)
  225              OPEN (30, FILE=KVxDFIL, FORM='UNFORMATTED', STATUS='OLD'
  226              REWIND (30)
  227              READ (30) RVxGRID
  228              READ (30) ((RVxION1(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5)
  229              READ (30) ((RVxION2(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5)
  230              READ (30) ((RVxION3(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5)
  231              READ (30) ((RVxION4(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5)
  232              READ (30) ((RVxION5(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5)
  233              READ (30) ((RVxION6(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5)
  234              READ (30) IVxPARAM
  235              READ (30) IVxLAUN
  236              READ (30) RVxLAUN
  237              READ (30) RVxOPTION
  238              CALL DOxDATA(IFxSTAY, IFxNEW, IFxEXI)
  239              IF (IFxEXI.EQ.1) THEN
  240  C
  241  C       The user has chosen to quit, so close the file and quit.
  242  C
  243                  CLOSE (30)
  244                  GO TO 20700
  245              ENDIF
  246  10010       KVxPMT = 'Overwrite existing file (Y/N)? '
  247  C
  248  C -------------------------------------------------------------
  249  C         Give the user the choice of replacing the old problem's
  250  C              data with the (possibly new) current data, or
```

```
Line#  Source Line

  251  C               making a second (backup) datafile.
  252  C -------------------------------------------------------------
  253  C
  254                CALL KEYBRD(KVxPMT,3,ITxQ,IVxVAR,RVxVAR,KTxANS)
  255                IF (KTxANS.EQ.'Y') THEN
  256                    GO TO 10015
  257                ELSE IF (KTxANS.EQ.'N') THEN
  258                    CLOSE(30)
  259                    KTxANS = ' '
  260                    KVxPMT = 'Backup the datafile (Y/N)? '
  261  C
  262  C         If the user has decided not to overwrite the old data, ask
  263  C           if a separate copy is to be made...
  264  C
  265                    CALL KEYBRD(KVxPMT,3,ITxQ,IVxVAR,RVxVAR,KTxANS)
  266                    IF (KTxANS.EQ.'N') GO TO 20000
  267                    KVxPMT = 'New filename (10 char max)? '
  268  C
  269  C         ... and if so, ask what the new file's name will be.
  270  C
  271                    CALL KEYBRD(KVxPMT,3,ITxQ,IVxVAR,RVxVAR,KVxDFIL)
  272                    OPEN (30,FILE=KVxDFIL,FORM='UNFORMATTED',STATUS='NEW'
  273                ELSE
  274  C
  275  C         The user has given an unexpected response, so go back and
  276  C           do it over again.
  277  C
  278                    GO TO 10010
  279                ENDIF
  280  10015         REWIND (30)
  281                WRITE (30) RVxGRID
  282                WRITE (30) ((RVxION1(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5
  283                WRITE (30) ((RVxION2(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5
  284                WRITE (30) ((RVxION3(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5
  285                WRITE (30) ((RVxION4(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5
  286                WRITE (30) ((RVxION5(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5
  287                WRITE (30) ((RVxION6(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5
  288                WRITE (30) IVxPARAM
  289                WRITE (30) IVxLAUN
  290                WRITE (30) RVxLAUN
  291                WRITE (30) RVxOPTION
  292                CLOSE(30)
  293            ELSE
  294  C
  295  C          Else this is an unexpected answer, so re-prompt.
  296  C
  297                GO TO 10005
  298            ENDIF
  299  11001     FORMAT (////////////////////////////////)
  300  11004     FORMAT ('0Enter filename for storage of ionospheric'
```

```
Line#  Source Line

   301  C
   302  C ---------------------------------------------------------------
   303  C
   304  C         End data set up section.
   305  C
   306  C ---------------------------------------------------------------
   307  C
   308  20000    IF (IFxSTAY.EQ.1) THEN
   309  C
   310  C ---------------------------------------------------------------
   311  C         If we've been running and decide to do another problem,
   312  C             (STAY in the program) re-cycle starts here.  DOxDATA
   313  C             to get the (possibly new) user values for the next
   314  C             problem & give the user the opportunity to update
   315  C             the datafile.
   316  C ---------------------------------------------------------------
   317  C
   318               CALL DOxDATA(IFxSTAY, IFxNEW, IFxEXI)
   319  C
   320  C        IFxEXI = 1 indicates that the user has decided to quit.
   321  C
   322               IF (IFxEXI.EQ.1) GO TO 20700
   323               KTxANS = ' '
   324               KVxPMT = 'Update the datafile (Y/N)? '
   325               CALL KEYBRD(KVxPMT,3,ITxQ,IVxVAR,RVxVAR,KTxANS)
   326               IF (KTxANS.EQ.'Y') THEN
   327                   OPEN (30, FILE = KVxDFIL, FORM='UNFORMATTED')
   328                   REWIND (30)
   329                   WRITE (30) RVxGRID
   330                   WRITE (30) ((RVxION1(I,J),J=1,RVxGRID(6)),I=1
   331        #,RVxGRID(5))
   332                   WRITE (30) ((RVxION2(I,J),J=1,RVxGRID(6)),I=1
   333        #,RVxGRID(5))
   334                   WRITE (30) ((RVxION3(I,J),J=1,RVxGRID(6)),I=1
   335        #,RVxGRID(5))
   336                   WRITE (30) ((RVxION4(I,J),J=1,RVxGRID(6)),I=1
   337        #,RVxGRID(5))
   338                   WRITE (30) ((RVxION5(I,J),J=1,RVxGRID(6)),I=1
   339        #,RVxGRID(5))
   340                   WRITE (30) ((RVxION6(I,J),J=1,RVxGRID(6)),I=1
   341        #,RVxGRID(5))
   342                   WRITE (30) IVxPARAM
   343                   WRITE (30) IVxLAUN
   344                   WRITE (30) RVxLAUN
   345                   WRITE (30) RVxOPTION
   346                   CLOSE(30)
   347               ENDIF
   348               KTxANS = ' '
   349            ENDIF
   350  C
```

```
Line#  Source Line

  351  C -------------------------------------------------------------
  352  C
  353  C       Begin actually firing out some rays.  Results are stored
  354  C         in consecutively numbered files, FORnnn.DAT, starting
  355  C         with  nnn = 040.  Beware that if the program is run on
  356  C         DOS or similar systems, starting another run of the
  357  C         program may cause data loss, since the new result files
  358  C         will simply write over the old ones.
  359  C
  360  C -------------------------------------------------------------
  361  C
  362  C        First, select the appropriate limits on the DO loops.
  363  C
  364           IF (IDNINT(RVxOPTION(1)).EQ.0) THEN
  365               IVxELLIM = 1
  366               IVxAZLIM = 1
  367           ELSE IF (IDNINT(RVxOPTION(1)).EQ.1) THEN
  368               IVxAZLIM = 1
  369               IVxELLIM = 1+IDNINT((RVxOPTION(7)-RVxOPTION(3))/
  370       #RVxOPTION(5))
  371           ELSE
  372               IVxELLIM = 1+IDNINT((RVxOPTION(7)-RVxOPTION(3))/
  373       #RVxOPTION(5))
  374               IVxAZLIM = 1+IDNINT((RVxOPTION(6)-RVxOPTION(2))/
  375       #RVxOPTION(4))
  376           ENDIF
  377           PRINT *,' '
  378           PRINT *,' BEGINNING RAY LOOPS.'
  379  C
  380  C        Generate the name of the result file by putting the number
  381  C          part into an 'internal device' (a string).  This string
  382  C          is then concatenated with the rest of the file's name.
  383  C
  384           WRITE (KVxIND,'(I2)') IVxIND
  385           KVxOFIL = 'FOR0'//KVxIND//'.DAT'
  386           OPEN (IVxIND,FILE = KVxOFIL)
  387           WRITE (IVxIND, 22000) IVxLAUN(1)+1, IVxAZLIM, IVxELLIM
  388  C
  389  C        These first numbers are written to the result file so that
  390  C          an analysis program can determine how many points of
  391  C          information there are.
  392  C
  393           DO 20200 I = 1, IVxAZLIM
  394               DO 20100 J = 1, IVxELLIM
  395                   RVxAZI = RVxOPTION(2) + (I-1)*RVxOPTION(4)
  396                   IF (RVxAZI.GE.360.0D00) RVxAZI = RVxAZI - 360.0
  397                   RVxELEV = RVxOPTION(3) + (J-1)*RVxOPTION(5)
  398                   WRITE(IVxIND, 21003) RVxAZI, RVxELEV
  399  C
  400  C        Preface the results for a ray with the azimuth and elevation.
```

```
Line#  Source Line

  401  C
  402  C ---------------------------------------------------------------
  403  C
  404  C       RAYSUB is the routine that actually traces the rays.
  405  C
  406                  CALL RAYSUB(RVxAZI, RVxELEV)
  407  C
  408  C       Now, write the results of the ray.  The elements of the
  409  C         result array are discussed in the comment block near
  410  C         the beginning of RAYSUB.
  411  C
  412                  WRITE(IVxIND, 21003) ((RVxBOUN(L,K),K=1,15), L=1,
  413       # IVxLAUN(1)+1)
  414                  DO 20099 K = 1, 15
  415                      DO 20098 L = 1, IVxLAUN(1)+1
  416  C
  417  C        Clear the result array for the next ray.
  418  C
  419                          RVxBOUN(L,K) = 0.0
  420  20098               CONTINUE
  421  20099           CONTINUE
  422  20100       CONTINUE
  423  20200   CONTINUE
  424          CLOSE(IVxIND)
  425          PRINT *,' '
  426          PRINT *,' RAY LOOPS DONE.'
  427          PRINT *,' '
  428          WRITE (6,21004) KVxOFIL
  429  C
  430  C        Increment the unit number that the results will be
  431  C            written to.
  432  C
  433          IVxIND = IVxIND + 1
  434  C
  435  C ---------------------------------------------------------------
  436  C
  437  C        End section that shoots rays.
  438  C
  439  C ---------------------------------------------------------------
  440  C
  441  20500   KVxPMT = 'Do another problem (Y/N)?'
  442  C
  443  C        Give the user a chance to quit.
  444  C
  445          KTxANS = ' '
  446          CALL KEYBRD(KVxPMT,3,ITxQ,IVxVAR,RVxVAR,KTxANS)
  447          IF (KTxANS.EQ.'Y') THEN
  448              IFxSTAY = 1
  449              GO TO 10000
  450          ELSE IF (ITxQ.LT.1.OR.KTxANS.NE.'N') THEN
```

```
Line#  Source Line

  451              GO TO 20500
  452            ENDIF
  453            KTxANS = ' '
  454  20700     CALL IOxPRET(CH)
  455  C
  456  C          Now, give the user the chance to abort the quit.
  457  C
  458            IF (CH.EQ.'A') GO TO 20500
  459  C
  460  C          Print the closing message.
  461  C
  462            WRITE (6,21007)
  463            WRITE (6,21008)
  464            WRITE (6,21009)
  465  21002     FORMAT (A1)
  466  21003     FORMAT (G24.14)
  467  21004     FORMAT (///,1X,'Results written to file ',A10,//)
  468  21007     FORMAT (1X,'Session done.  For safety''s sake, copy the data'
  469  21008     FORMAT (1X,'and result files into a separate directory to'
  470  21009     FORMAT (1X,'prevent accidental overwriting.')
  471  22000     FORMAT (I4)
  472            END
```

main Local Symbols

Name	Class	Type	Size
ITXQ.	local	INTEGER*4	4
I	local	INTEGER*4	4
J	local	INTEGER*4	4
K	local	INTEGER*4	4
CH.	local	CHAR*1	1
L	local	INTEGER*4	4
IFXEXI.	local	INTEGER*4	4
IFXNEW.	local	INTEGER*4	4
IVXIND.	local	INTEGER*4	4
KVXIND.	local	CHAR*2	2
KVXDFIL	local	CHAR*10	10
KTXANS.	local	CHAR*1	1
IVXVAR.	local	INTEGER*4	4
KVXOFIL	local	CHAR*10	10
RVXAZI.	local	REAL*8	8
KTXTMP.	local	CHAR*10	10
IFXSTAY	local	INTEGER*4	4
RVXVAR.	local	REAL*8	8
KVXPMT.	local	CHAR*40	40
IVXELLIM.	local	INTEGER*4	4
RVXELEV	local	REAL*8	8
IVXTYP.	local	INTEGER*4	4

main Local Symbols

Name	Class	Type	Size
IVXAZLIM.	local	INTEGER*4	4
IVXPARAM.	MAINDAT	INTEGER*4	24
IVXLAUN	MAINDAT	INTEGER*4	12
RVXGRID	MAINDAT	REAL*8	48
RVXIONPT.	MAINDAT	REAL*8	86400
RVXLAUN	MAINDAT	REAL*8	56
RVXOPTION	MAINDAT	REAL*8	56
RVXBOUN	RESULTS	REAL*8	1320
RVXION1	MAINDAT	REAL*8	14400
RVXION2	MAINDAT	REAL*8	14400
RVXION3	MAINDAT	REAL*8	14400
RVXION4	MAINDAT	REAL*8	14400
RVXION5	MAINDAT	REAL*8	14400
RVXION6	MAINDAT	REAL*8	14400

```
Line#  Source Line

  474          SUBROUTINE DOxDATA(IFSxSTA, IFSxNEW, IFSxEXI)
  475  C
  476  C-----------------------------------------------------------------
  477  C
  478  C    DOxDATA -- SUBROUTINE TO MAINTAIN AND ACCESS THE DATA FOR
  479  C          CONVENIENT USE OF THE WHOLE RAYTRACING PROGRAM
  480  C
  481  C          CALLED BY: MAIN
  482  C
  483  C          CALLS: IOxPRET, KEYBRD
  484  C
  485  C-----------------------------------------------------------------
  486  C
  487  C          AUTHOR:       ERIC L. STROBEL
  488  C
  489  C          DATE:         01/15/88
  490  C
  491  C          VERSION:      2.1
  492  C
  493  C-----------------------------------------------------------------
  494  C
  495  C          REVISED:         10/09/86 -- V1.0.  Initial revision.
  496  C
  497  C                           09/01/87 -- V2.0.  Uses the new KEYBRD
  498  C                           routine and unformatted reads and writes.
  499  C
  500  C                           01/15/88 -- V2.1.  Contains necessary
  501  C                           concessions to the limited environment that
  502  C                           DOS provides.
  503  C
  504  C-----------------------------------------------------------------
  505  C
  506  C         USES:    IFSxSTAY          A FLAG THAT INDICATES THAT THE
  507  C                                     USER HAS REMAINED IN THE PROBLEM
  508  C                  IFSxNEW           A FLAG THAT INDICATES THAT THIS
  509  C                                     IS A NEW PROBLEM
  510  C
  511  C          TO DECIDE HOW TO PRESENT AN INTERFACE TO THE USER, IN
  512  C                  ORDER TO MAINTAIN THE DATABASE FILES
  513  C
  514  C         RETURNS:          IFSxEXI          A FLAG THAT INDICATES THAT
  515  C                                             THE USER HAS CHOSEN TO EXIT
  516  C
  517  C-----------------------------------------------------------------
  518  C
  519          INTEGER IFSxSTA, IFSxNEW, IFSxEXI, ITLxQ
  520          INTEGER IVxPARAM(6), ITLxC1, ITLxC2, ITLxC3
  521          INTEGER ITLxTMP, IVxLAUN(3), IVLxTYP, IVLxVAR
  522  C
  523          REAL*8 RVLxLAT, RVLxLON, RVxGRID(6), RVxIONPT(30,60,6
```

```
Line#  Source Line

  524           REAL*8 RTLx1, RTLx2, RVxLAUN(7), RVxOPTION(7), RTxA
  525           REAL*8 RVLxVAR, RPxDTOR
  526           REAL*8 RVxION1(30,60), RVxION2(30,60), RVxION3(30,60)
  527           REAL*8 RVxION4(30,60), RVxION5(30,60), RVxION6(30,60)
  528   C
  529           CHARACTER*10 KVLxFIL, KTLxTMP
  530           CHARACTER*40 KVLxPMT
  531           CHARACTER*1 KVLxANS
  532           CHARACTER*7 KCLxLBL(6)
  533   C
  534           COMMON /MAINDAT/ RVxGRID,RVxIONPT,IVxPARAM,IVxLAUN,
  535        # RVxLAUN,RVxOPTION
  536   C
  537   C          Stuff the big array with pieces of more manageable size.
  538   C            This is in order to compensate for the unfortunate DOS
  539   C            restrictions on record sizes.
  540   C
  541           EQUIVALENCE (RVxION1,RVxIONPT), (RVxION2,RVxIONPT(1,1,2))
  542           EQUIVALENCE (RVxION3,RVxIONPT(1,1,3))
  543           EQUIVALENCE (RVxION4,RVxIONPT(1,1,4))
  544           EQUIVALENCE (RVxION5,RVxIONPT(1,1,5))
  545           EQUIVALENCE (RVxION6,RVxIONPT(1,1,6))
  546   C
  547           DATA KCLxLBL /' foE**2',' hmF1','foF1**2',' YmF2',
  548        #'  hmF2','foF2**2'/
  549   C
  550   C ---------------------------------------------------------
  551   C
  552           KVLxPMT = ' '
  553           IVLxTYP = 0
  554           IVLxVAR = 0
  555           RVLxVAR = 0.0
  556           KTLxTMP = ' '
  557           RPxDTOR = 0.0174532925D00
  558   C
  559   C ---------------------------------------------------------
  560   C          If this is the first time thru on a new problem, read
  561   C             in either an ionospheric grid produced by another
  562   C             program (RADAR-C right now) or a grid that is to be
  563   C             entered by hand.
  564   C ---------------------------------------------------------
  565   C
  566   10000    IF (IFSxNEW.EQ.1.AND.IFSxSTA.EQ.0) THEN
  567               WRITE(6,11001)
  568               WRITE(6,11002)
  569               KVLxFIL = 'GRID.DAT'
  570               KVLxPMT = ' (type NONE if none exists) '
  571               CALL KEYBRD(KVLxPMT,3,ITLxQ,IVLxVAR,RVLxVAR,KVLxFIL)
  572               IF (KVLxFIL.EQ.'NONE') THEN
  573   C
```

```
Line#  Source Line

  574  C          No external grid file exists, so build it by hand.
  575  C
  576                     OPEN(20, FILE='GRID.DAT',FORM='UNFORMATTED'
  577       #, STATUS='NEW')
  578                     REWIND (20)
  579  10020              WRITE (6,'(1X,////////////////////////////////////)')
  580  C
  581  C          RVxGRID gives info needed to obtain the lat's & lon's
  582  C               of the grid.
  583  C
  584                     KVLxPMT = 'Input lat grid spacing (deg): '
  585                     RTLx1 = RVxGRID(1)
  586                     RTLx2 = RVxGRID(2)
  587  C
  588  C         Prompt for the latitude spacing of the lat-lon grid.
  589  C
  590                     CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
  591                     IF (ITLxQ.GE.1) RVxGRID(1) = RTLx1
  592                     KVLxPMT = 'Input lon grid spacing (deg): '
  593  C
  594  C         Prompt for the longitude spacing of the lat-lon grid.
  595  C
  596                     CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx2,KTLxTMP)
  597                     IF (ITLxQ.GE.1) RVxGRID(2) = RTLx2
  598  C
  599                     WRITE(6,11007)
  600                     KVLxPMT = '                    LATITUDE :'
  601                     RTLx1 = RVxGRID(3)
  602  C
  603  C         Prompt for the grid's starting latitude.
  604  C
  605                     CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
  606                     IF (ITLxQ.GE.1) RVxGRID(3) = RTLx1
  607  C
  608                     KVLxPMT = 'LONGITUDE (east = positive):'
  609                     RTLx2 = RVxGRID(4)
  610  C
  611  C         Prompt for the grid's starting longitude.
  612  C
  613                     CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx2,KTLxTMP)
  614                     IF (ITLxQ.GE.1) RVxGRID(4) = RTLx2
  615  C
  616                     KVLxPMT = 'Input # of grid points in lat.: '
  617                     RTLx1 = RVxGRID(5)
  618  C
  619  C         Prompt for the number of grid point in the latitude direction.
  620  C
  621                     CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
  622                     IF (ITLxQ.GE.1) RVxGRID(5) = RTLx1
  623  C
```

```
Line#  Source Line

  624                   KVLxPMT = '                        ...  in lon.: '
  625                   RTLx1 = RVxGRID(6)
  626   C
  627   C       Prompt for the number of grid point in the longitude direction
  628   C
  629                   CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
  630                   IF (ITLxQ.GE.1) RVxGRID(6) = RTLx1
  631   C
  632   C       Verify the grid inputs and give the user a chance to re-enter
  633   C         them to fix errors.
  634   C
  635                   KVLxANS = 'Y'
  636                   KVLxPMT = 'Grid setup OK (Y/N) ? '
  637                   CALL KEYBRD(KVLxPMT,3,ITLxQ,IVLxVAR,RVLxVAR,KVLxANS)
  638                   IF (KVLxANS.EQ.'N'.OR.KVLxANS.EQ.'n') GO TO 10020
  639   C
  640   C       Write the results out into a GRID file.
  641   C
  642                   WRITE(20) RVxGRID
  643                   WRITE (6,'(1X,////////////////////////////////////)')
  644   C
  645   C       DO-loops for the entry of ionospheric parameters at each
  646   C         grid point.
  647   C
  648                   DO 10200 I=1, RVxGRID(5)
  649                     RVLxLAT = RVxGRID(3) + (I-1)*RVxGRID(1)
  650                     DO 10100 J=1, RVxGRID(6
  651                       RVLxLON = RVxGRID(4) + (J-1)*RVxGRID(2)
  652                       WRITE (6,12002) RVLxLAT, RVLxLON
  653   10049               DO 10050 K=1,6
  654   C
  655   C       The elements of KCLxLBL may be found in the DATA statement
  656   C         at the beginning of this routine.
  657   C
  658                         KVLxPMT = 'Input '//KCLxLBL(K)//' :'
  659                         RTLx1 = RVxIONPT(I,J,K)
  660                         CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1
       #,KTLxTMP)
  662                         IF (ITLxQ.GE.1) RVxIONPT(I,J,K) = RTLx1
  663   10050               CONTINUE
  664                       KVLxANS = 'Y'
  665                       KVLxPMT = 'Profile inputs OK (Y/N)? '
  666   C
  667   C       For each grid point, verify correct entry of values and give
  668   C         the user a chance to fix errors.
  669   C
  670                       CALL KEYBRD(KVLxPMT,3,ITLxQ,IVLxVAR,RVLxVAR
       #,KVLxANS)
  672                       IF (KVLxANS.EQ.'N'.OR.KVLxANS.EQ.'n') GO TO 10049
  673                       WRITE (6,'(1X,////////////////////////////////)')
```

```
Line#  Source Line

  674  10100              CONTINUE
  675  10200            CONTINUE
  676  C
  677  C          Write the grid file in the same manageable pieces that
  678  C               everything else expects it to be in.
  679  C
  680                     WRITE (20) ((RVxION1(I,J),J=1,RVxGRID(6)),I=1
  681       #,RVxGRID(5))
  682                     WRITE (20) ((RVxION2(I,J),J=1,RVxGRID(6)),I=1
  683       #,RVxGRID(5))
  684                     WRITE (20) ((RVxION3(I,J),J=1,RVxGRID(6)),I=1
  685       #,RVxGRID(5))
  686                     WRITE (20) ((RVxION4(I,J),J=1,RVxGRID(6)),I=1
  687       #,RVxGRID(5))
  688                     WRITE (20) ((RVxION5(I,J),J=1,RVxGRID(6)),I=1
  689       #,RVxGRID(5))
  690                     WRITE (20) ((RVxION6(I,J),J=1,RVxGRID(6)),I=1
  691       #,RVxGRID(5))
  692  C
  693  C        Do a series of prompts for the sunspot number and time values.
  694  C
  695  10201              KVLxPMT = 'Input (integer) sunspot number: '
  696                     ITLxTMP = IVxPARAM(1)
  697                     CALL KEYBRD(KVLxPMT,1,ITLxQ,ITLxTMP,RVLxVAR,KTLxTMP)
  698                     IF (ITLxQ.GE.1) IVxPARAM(1) = ITLxTMP
  699  C
  700                     KVLxPMT = 'Input year: '
  701                     ITLxTMP = IVxPARAM(2)
  702                     CALL KEYBRD(KVLxPMT,1,ITLxQ,ITLxTMP,RVLxVAR,KTLxTMP
  703                     IF (ITLxQ.GE.1) IVxPARAM(2) = ITLxTMP
  704  C
  705                     KVLxPMT = '...  month: '
  706                     ITLxTMP = IVxPARAM(3)
  707                     CALL KEYBRD(KVLxPMT,1,ITLxQ,ITLxTMP,RVLxVAR,KTLxTMP
  708                     IF (ITLxQ.GE.1) IVxPARAM(3) = ITLxTMP
  709  C
  710                     KVLxPMT = '...    day: '
  711                     ITLxTMP = IVxPARAM(4)
  712                     CALL KEYBRD(KVLxPMT,1,ITLxQ,ITLxTMP,RVLxVAR,KTLxTMP
  713                     IF (ITLxQ.GE.1) IVxPARAM(4) = ITLxTMP
  714  C
  715                     KVLxPMT = 'Input UT time (hr) : '
  716                     ITLxTMP = IVxPARAM(5)
  717                     CALL KEYBRD(KVLxPMT,1,ITLxQ,ITLxTMP,RVLxVAR,KTLxTMP
  718                     IF (ITLxQ.GE.1) IVxPARAM(5) = ITLxTMP
  719  C
  720                     KVLxPMT = '...   UT time (min): '
  721                     ITLxTMP = IVxPARAM(6)
  722                     CALL KEYBRD(KVLxPMT,1,ITLxQ,ITLxTMP,RVLxVAR,KTLxTMP
  723                     IF (ITLxQ.GE.1) IVxPARAM(6) = ITLxTMP
```

```
Line#  Source Line

  724  C
  725                    WRITE (20) IVxPARAM
  726                    KVLxPMT = 'S.S. # and times OK (Y/N)? '
  727                    KVLxANS = 'Y'
  728  C
  729  C         Again, give the user a chance to verify the entries and to
  730  C           correct errors.
  731  C
  732                    CALL KEYBRD(KVLxPMT,3,ITLxQ,IVLxVAR,RVLxVAR,KVLxANS)
  733                    IF (KVLxANS.EQ.'N'.OR.KVLxANS.EQ.'n') GO TO 10201
  734  C
  735  C         Close the GRID file and go on to get the rest of the user's
  736  C           input.
  737  C
  738                    CLOSE(20)
  739                    GO TO 20000
  740                  ENDIF
  741  C
  742  C           Read the externally constructed grid file.
  743  C
  744                  OPEN(20, FILE=KVLxFIL,FORM='UNFORMATTED')
  745                  REWIND (20)
  746                  READ(20) RVxGRID
  747                  READ(20) ((RVxION1(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
  748                  READ(20) ((RVxION2(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
  749                  READ(20) ((RVxION3(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
  750                  READ(20) ((RVxION4(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
  751                  READ(20) ((RVxION5(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
  752                  READ(20) ((RVxION6(I,J),J=1,RVxGRID(6)),I=1,RVxGRID(5))
  753                  READ(20) IVxPARAM
  754                  CLOSE(20)
  755                ENDIF
  756  11001     FORMAT(////////////////////////////////////////)
  757  11002     FORMAT(1X,'The ionospheric grid file name is?')
  758  11007     FORMAT(1X,'Input starting latitude and longitude in degrees.')
  759  12002     FORMAT(/,1X,'GRID PT. - Lat: ',G15.6,'  - Lon: ',G15.6)
  760  C
  761  C -------------------------------------------------------
  762  C
  763  C          Done reading in data, now build/alter database of
  764  C            options, by processing menu choices.
  765  C
  766  C -------------------------------------------------------
  767  C
  768  C          It should be noted that by simply adding to the list
  769  C            of lines contained in the computed GO TOs, additional
  770  C            items may be attached to the existing menus, or additional
  771  C            menus may be added.  If the menu system is extended
  772  C            in this way, the routine IOxMENU is of sufficient
  773  C            generality to handle this.  The appropriate menu
```

```
Line#  Source Line

  774  C           files must be changed/added and care should be
  775  C           taken to make sure that any one menu list doesn't
  776  C           become so large that part of it scrolls up off the
  777  C           screen before the prompt comes up.  The menu file
  778  C           scheme is discussed in a comment block in the
  779  C           routine IOxMENU.
  780  C
  781  C ------------------------------------------------------------
  782  C
  783  C ------------------------------------------------------------
  784  C       MENU 1 -- Main menu:  4 choices
  785  C ------------------------------------------------------------
  786  C
  787  20000   CALL IOxMENU(1,ITLxC1)
  788          GO TO (20010, 20200, 20400, 20600), ITLxC1
  789  C
  790  C ------------------------------------------------------------
  791  C       MENU 2 -- Launch parameters menu:  9 choices
  792  C ------------------------------------------------------------
  793  C
  794  20010   CALL IOxMENU(2, ITLxC2)
  795          GO TO (20020,20030,20040,20050,20060,20070,20080
  796       #,20090,20000), ITLxC2
  797  20020   IF (IVxLAUN(1).EQ.0) IVxLAUN(1)=1
  798          KVLxPMT = 'Bounce limit '
  799          ITLxTMP = IVxLAUN(1)
  800  C
  801  C       Prompt for the limit in the number of times the ray
  802  C         will be allowed to return to earth, if this is in
  803  C         fact an applicable concept.  The default value is
  804  C         1.
  805  C
  806          CALL KEYBRD(KVLxPMT,1,ITLxQ,ITLxTMP,RVLxVAR,KTLxTMP)
  807          IF (ITLxQ.GE.1) IVxLAUN(1) = ITLxTMP
  808          GO TO 20010
  809  20030   KVLxPMT = 'Do signal intensity (no = 1)? '
  810          ITLxTMP = IVxLAUN(2)
  811  C
  812  C       Prompt to find out whether or not the user wants signal
  813  C         intensity to be calculated.
  814  C
  815          CALL KEYBRD(KVLxPMT,1,ITLxQ,ITLxTMP,RVLxVAR,KTLxTMP)
  816          IF (ITLxQ.GE.1) IVxLAUN(2) = ITLxTMP
  817          GO TO 20010
  818  20040   KVLxPMT = 'Reflect from ground(1) or water(0)? '
  819          ITLxTMP = IVxLAUN(3)
  820  C
  821  C       Prompt the user to choose a generic conductivity for the
  822  C         reflection loss calculation at an earth bounce point.
  823  C         Someday this should be replaced with a global conductivity
```

```
Line#  Source Line

  824  C          map.
  825  C
  826           CALL KEYBRD(KVLxPMT,1,ITLxQ,ITLxTMP,RVLxVAR,KTLxTMP)
  827           IF (ITLxQ.GE.1) IVxLAUN(3) = ITLxTMP
  828           GO TO 20010
  829  20050    KVLxPMT = 'Enter launch pt. latitude '
  830  C
  831  C        Do a pair of prompts for the latitude, longitude location
  832  C          of the starting point of the ray.
  833  C
  834           RTLx1 = RVxLAUN(1)
  835           CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
  836           IF (ITLxQ.GE.1) RVxLAUN(1) = RTLx1
  837           KVLxPMT = 'Enter launch pt. longitude '
  838           RTLx2 = RVxLAUN(2)
  839           CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx2,KTLxTMP)
  840           IF (ITLxQ.GE.1) RVxLAUN(2) = RTLx2
  841           GO TO 20010
  842  20060    KVLxPMT = 'Enter starting ht. (km) '
  843           RTLx1 = RVxLAUN(6)
  844  C
  845  C        Prompt for the altitude of the starting point for the ray.
  846  C
  847           CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
  848           IF (ITLxQ.GE.1) RVxLAUN(6) = RTLx1
  849           GO TO 20010
  850  20070    IF (RVxLAUN(3).EQ.0.0) RVxLAUN(3) = 3000.0D00
  851           IF (RVxLAUN(7).EQ.0.0) RVxLAUN(7) = 36000.0D00
  852  C
  853  C        A pair of prompts to establish the range and altitude
  854  C          cutoff values necessary to prevent the problem from
  855  C          running on forever.  The range is the ground range
  856  C          along the ground track of the ray, irregardless of the
  857  C          altitude and actual path of the ray.  The default
  858  C          value for the ground range is 3000 km, and the default
  859  C          for the altitude limit is 36000 km, or approximately
  860  C          geostationary altitude.
  861  C
  862           KVLxPMT = 'Enter range limit (km) '
  863           RTLx1 = RVxLAUN(3)
  864           CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
  865           IF (ITLxQ.GE.1) RVxLAUN(3) = RTLx1
  866           KVLxPMT = 'Enter altitude limit (km) '
  867           RTLx2 = RVxLAUN(7)
  868           CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx2,KTLxTMP)
  869           IF (ITLxQ.GE.1) RVxLAUN(7) = RTLx2
  870           GO TO 20010
  871  20080    IF (RVxLAUN(4).EQ.0.0) RVxLAUN(4) = 4.0D00
  872           KVLxPMT = 'Enter raypath increment (km) '
  873           RTLx1 = RVxLAUN(4)
```

```
Line#  Source Line

  874  C
  875  C       Prompt for the size of the nominal raypath increment.
  876  C         The default value of 4 km represents a compromise
  877  C         between execution time (longer increments = faster
  878  C         execution) and accuracy (shorter increments = more
  879  C         accurate).
  880  C
  881           CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
  882           IF (ITLxQ.GE.1) RVxLAUN(4) = RTLx1
  883           GO TO 20010
  884  20090    KVLxPMT = 'Enter wave frequency (MHz) '
  885           IF (RVxLAUN(5).EQ.0.0) RVxLAUN(5) = 5.0D00
  886           RTLx1 = RVxLAUN(5)
  887  C
  888  C       Prompt for the frequency of the HF transmission that
  889  C         the ray is supposed to represent.  The default value
  890  C         of 5 MHz is somewhat arbitrary.  Since the program
  891  C         performs frequent divisions by wave frequency squared,
  892  C         some sort of non-zero default value is needed just to
  893  C         prevent 'unexplained' crashes of the program for new
  894  C         problems.
  895  C
  896           CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
  897           IF (ITLxQ.GE.1) RVxLAUN(5) = RTLx1
  898           GO TO 20010
  899  C
  900  C ---------------------------------------------------------------
  901  C       MENU 3 -- Elevation parameters menu;  8 choices.
  902  C ---------------------------------------------------------------
  903  C
  904  20200    CALL IOxMENU(3,ITLxC3)
  905           GO TO (20210,20220,20230,20240,20250,20260,20270,20000),ITLxC3
  906  20210    KVLxPMT = 'Enter dimension of problem '
  907           RTLx1 = RVxOPTION(1)
  908  C
  909  C       The dimension of the problem goes as follows (the reason for
  910  C         the term dimension should become obvious):
  911  C
  912  C         0 = a single ray (a single point on the sky)
  913  C         1 = a vertical fan of rays at a single azimuth
  914  C             (a line of points on the sky)
  915  C         2 = a set of fans of rays (an array of points on
  916  C             the sky)
  917  C
  918           CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
  919           IF (ITLxQ.LT.1) THEN
  920              GO TO 20200
  921           ELSE IF (IDNINT(RTLx1).GT.2.OR.IDNINT(RTLx1).LT.0) THEN
  922              GO TO 20210
  923           ELSE
```

```
Line#  Source Line

  924              RVxOPTION(1) = RTLx1
  925              GO TO 20200
  926            ENDIF
  927  20220    KVLxPMT = 'Enter starting azimuth (deg) '
  928            RTLx1 = RVxOPTION(2)
  929  C
  930  C         Prompt for the initial azimuth for the problem.
  931  C
  932            CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
  933            IF (ITLxQ.GE.1) RVxOPTION(2) = RTLx1
  934            GO TO 20200
  935  20230    WRITE (6,22009)
  936            KVLxPMT = 'for values >90 deg from zenith. '
  937            RTLx1 = RVxOPTION(3)
  938  C
  939  C         Prompt for the initial elevation for the problem.
  940  C           For a starting point at greater than zero altitude,
  941  C           elevation angles may be greater than 90 degrees
  942  C           from the zenith.  Such angles are designated by
  943  C           negative values.
  944  C
  945            CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
  946            IF (ITLxQ.GE.1) RVxOPTION(3) = RTLx1
  947            GO TO 20200
  948  20240    IF (IDNINT(RVxOPTION(1)).EQ.0) THEN
  949                IF (RVxOPTION(5).EQ.0.0) RVxOPTION(5) = 1.0D00
  950                GO TO 20200
  951            ENDIF
  952            KVLxPMT = 'Enter elev. resolution (deg) '
  953            RTLx1 = RVxOPTION(5)
  954  C
  955  C         Prompt for the spacing of rays in the elevation
  956  C           direction.  This prompt is only displayed if it
  957  C           is appropriate, namely if the dimension of the
  958  C           problem is 1 or 2.  Otherwise, a default value
  959  C           of 1 is set and the user is returned directly to
  960  C           the elevation parameters menu.
  961  C
  962            CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
  963            IF (ITLxQ.GE.1) RVxOPTION(5) = RTLx1
  964            GO TO 20200
  965  20250    IF (IDNINT(RVxOPTION(1)).LT.2) THEN
  966                IF (RVxOPTION(4).EQ.0.0) RVxOPTION(4) = 1.0D00
  967                GO TO 20200
  968            ENDIF
  969            KVLxPMT = 'Enter azimuth resolution (deg) '
  970            RTLx1 = RVxOPTION(4)
  971  C
  972  C         Prompt for the azimuth spacing of the rays.  This
  973  C           prompt is only displayed if it is appropriate.
```

```
Line#  Source Line

  974  C          namely if the dimension of the problem is 2.
  975  C          Otherwise, a default value of 1 is set and the user
  976  C          is returned directly to the elevation parameters
  977  C          menu.
  978  C
  979             CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
  980             IF (ITLxQ.GE.1) RVxOPTION(4) = RTLx1
  981             GO TO 20200
  982  20260      IF (IDNINT(RVxOPTION(1)).EQ.0) THEN
  983                 IF (RVxOPTION(7).EQ.0.0) RVxOPTION(7) = RVxOPTION(3)
  984                 GO TO 20200
  985             ENDIF
  986             KVLxPMT = 'Enter elev. limit (deg) '
  987             RTLx1 = RVxOPTION(7)
  988  C
  989  C          Prompt to establish the other end of the range of
  990  C            elevation values to be stepped through, the range
  991  C            beginning with the starting elevation from above.
  992  C            This prompt is only displayed if appropriate, namely
  993  C            if there are to be multiple rays.  Otherwise, the
  994  C            limit is set to be equal to the starting elevation and
  995  C            the user is returned directly to the elevation
  996  C            parameters menu.
  997  C
  998             CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
  999             IF (ITLxQ.GE.1) RVxOPTION(7) = RTLx1
 1000             GO TO 20200
 1001  20270      IF (IDNINT(RVxOPTION(1)).LT.2) THEN
 1002                 IF (RVxOPTION(6).EQ.0.0) RVxOPTION(6) = RVxOPTION(2)
 1003                 GO TO 20200
 1004             ENDIF
 1005             KVLxPMT = 'Enter azimuthal limit (deg) '
 1006             RTLx1 = RVxOPTION(6)
 1007  C
 1008  C          Prompt for the other end of the range of azimuth values
 1009  C            which will be stepped through, beginning with the
 1010  C            starting azimuth value.  This prompt is only displayed
 1011  C            when appropriate, i.e. when there the dimension of the
 1012  C            problem is 2.  Otherwise, the limit is set to be equal
 1013  C            to the starting azimuth and the user is returned
 1014  C            directly to the elevation parameters menu.
 1015  C
 1016             CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
 1017             IF (ITLxQ.GE.1) RVxOPTION(6) = RTLx1
 1018             GO TO 20200
 1019  20400      RETURN
 1020  C
 1021  C          Line 20400 is selected when the decides that all necessary
 1022  C            values have been entered and that it is time to start
 1023  C            sending out the rays.
```

```
 Line#  Source Line

  1024  C
  1025  20600    IFSxEXI = 1
  1026  C
  1027  C        This sets the EXIT flag to 1, indicating that the user has
  1028  C          decided to exit the program.
  1029  C
  1030           RETURN
  1031  22009    FORMAT(1X,'Enter starting elevation (deg), use neg.')
  1032           END
```

DOXDATA Local Symbols

Name	Class	Type	Size
IFSXEXI	param		
IFSXNEW	param		
IFSXSTA	param		
IVLXTYP	local	INTEGER*4	4
I	local	INTEGER*4	4
J	local	INTEGER*4	4
K	local	INTEGER*4	4
ITLXQ	local	INTEGER*4	4
KVLXFIL	local	CHAR*10	10
KVLXANS	local	CHAR*1	1
IVLXVAR	local	INTEGER*4	4
RVLXLAT	local	REAL*8	8
ITLXTMP	local	INTEGER*4	4
RPXDTOR	local	REAL*8	8
KTLXTMP	local	CHAR*10	10
RVLXVAR	local	REAL*8	8
RVLXLON	local	REAL*8	8
ITLXC1	local	INTEGER*4	4
KVLXPMT	local	CHAR*40	40
ITLXC2	local	INTEGER*4	4
ITLXC3	local	INTEGER*4	4
RTLX1	local	REAL*8	8
RTLX2	local	REAL*8	8
KCLXLBL	local	CHAR*7	42
IVXPARAM	MAINDAT	INTEGER*4	24
IVXLAUN	MAINDAT	INTEGER*4	12
RVXGRID	MAINDAT	REAL*8	48
RVXIONPT	MAINDAT	REAL*8	86400
RVXLAUN	MAINDAT	REAL*8	56
RVXOPTION	MAINDAT	REAL*8	56
RVXION1	MAINDAT	REAL*8	14400
RVXION2	MAINDAT	REAL*8	14400
RVXION3	MAINDAT	REAL*8	14400
RVXION4	MAINDAT	REAL*8	14400
RVXION5	MAINDAT	REAL*8	14400

DOXDATA Local Symbols

Name	Class	Type	Size
RVXION6	MAINDAT	REAL*8	14400

```
Line#  Source Line

 1034           SUBROUTINE IOxMENU(N, CHOICE)
 1035   C
 1036   C-------------------------------------------------------------
 1037   C
 1038   C    IOxMENU -- SUBROUTINE TO SET UP A MENU BY READING 'TEMPLATES'
 1039   C         FOR THE MENUS FROM FILES, THEN RETURNING THE MENU CHOICE
 1040   C
 1041   C         CALLED BY: DOxDATA
 1042   C
 1043   C         CALLS:  IOxPRET
 1044   C
 1045   C-------------------------------------------------------------
 1046   C
 1047   C         AUTHOR:          ERIC L. STROBEL
 1048   C
 1049   C         DATE:            10/09/86
 1050   C
 1051   C         VERSION:         1.0
 1052   C
 1053   C-------------------------------------------------------------
 1054   C
 1055   C         REVISED:         10/09/86 -- V1.0.  Initial revision.
 1056   C
 1057   C-------------------------------------------------------------
 1058   C
 1059   C         USES:         N          NUMBER DENOTING WHICH MENU RESOURCE
 1060   C                                  TO USE
 1061   C
 1062   C         TO PRESENT THE USER WITH A MENU AND THEN SEND THE USER'S
 1063   C                 CHOICE BACK.  (It should be noted that this routine
 1064   C                 was written before adoption of the variable naming
 1065   C                 convention.  Due to the short length of the routine
 1066   C                 it has never seemed worth the change.)
 1067   C
 1068   C         RETURNS:      CHOICE     THE MENU CHOICE THAT WAS MADE
 1069   C
 1070   C-------------------------------------------------------------
 1071   C
 1072           INTEGER N, CHOICE, ITEMS, Q
 1073           REAL*8 DUMMY
 1074           CHARACTER*10 FILES(10), FILNAM, KDUMMY
 1075           CHARACTER*40 TITLE, ITEMLIST(10), PROMPT
 1076           CHARACTER*1 CH
 1077   C
 1078   C -------------------------------------------------------------
 1079   C
 1080   C        Read the menu resource from a file set.  The scheme for
 1081   C           maintaining menus goes as follows...  The file MASTER.MEN
 1082   C           contains a list of the names of the menu files, in the
 1083   C           order by which they are numbered in DOxDATA.  The indivi-
```

```
Line#  Source Line

 1084  C          dual menu files have the following format:
 1085  C
 1086  C                          # of items, MENU TITLE
 1087  C                          Text
 1088  C                          of
 1089  C                          each
 1090  C                          item
 1091  C                          (line-by-line, one line per item)
 1092  C                          Prompt text
 1093  C
 1094  C          The existing menu files should be consulted as examples.
 1095  C
 1096  C -------------------------------------------------------------
 1097  C
 1098           OPEN(20, FILE='MASTER.MEN', STATUS='OLD')
 1099           READ(20,200) (FILES(I), I=1,10)
 1100  200      FORMAT(A10)
 1101           CLOSE(20)
 1102           FILNAM = FILES(N)
 1103           OPEN(21, FILE=FILNAM, STATUS='OLD')
 1104           READ(21,300) ITEMS, TITLE
 1105  300      FORMAT(I4,A40)
 1106           READ(21,400) (ITEMLIST(J), J=1, ITEMS)
 1107  400      FORMAT(A40)
 1108           READ(21,500) PROMPT
 1109  500      FORMAT(A40)
 1110           CLOSE(21)
 1111  C
 1112  C -----------------------------------------------------
 1113  C
 1114  C         Display the menu's list of items.
 1115  C
 1116  C -----------------------------------------------------
 1117  C
 1118  550      WRITE(6,600)
 1119  600      FORMAT(////////////////////////////////////////////)
 1120           WRITE(6,700) TITLE
 1121  700      FORMAT(20X,A40,//)
 1122           WRITE(6,800) (ITEMLIST(J), J=1, ITEMS)
 1123  800      FORMAT(5X,A40,/)
 1124  C
 1125  C -----------------------------------------------------
 1126  C
 1127  C         Get the user's response and check it.
 1128  C
 1129  C -----------------------------------------------------
 1130  C
 1131           CHOICE = 0
 1132  900      CALL KEYBRD(PROMPT,1,Q,CHOICE,DUMMY,RDUMMY)
 1133           IF (Q.LT.1) GO TO 900
```

```
 Line#  Source Line

  1134          IF (CHOICE.LT.1.OR.CHOICE.GT.ITEMS) THEN
  1135              PRINT *,' INVALID CHOICE.'
  1136              GO TO 550
  1137          ENDIF
  1138          CALL IOxPRET(CH)
  1139          IF (CH.EQ.'A') GO TO 550
  1140          RETURN
  1141          END
```

IOXMENU Local Symbols

Name	Class	Type	Size
CHOICE.	param		
N	param		
TITLE	local	CHAR*40	40
ITEMS	local	INTEGER*4	4
I	local	INTEGER*4	4
J	local	INTEGER*4	4
CH.	local	CHAR*1	1
DUMMY	local	REAL*8	8
Q	local	INTEGER*4	4
KDUMMY.	local	CHAR*10	10
PROMPT.	local	CHAR*40	40
ITEMLIST.	local	CHAR*40	400
FILES	local	CHAR*10	100
FILNAM.	local	CHAR*10	10

```
Line#  Source Line

 1143          SUBROUTINE IOxPRET(XXX)
 1144  C
 1145  C-------------------------------------------------------------
 1146  C
 1147  C    IOxPRET -- SUBROUTINE TO PROMPT THE USER TO PRESS RETURN TO
 1148  C         CONTINUE, AND THEN CLEAR THE SCREEN
 1149  C
 1150  C         CALLED BY: MAIN, IOxMENU
 1151  C
 1152  C         CALLS: KEYBRD
 1153  C
 1154  C-------------------------------------------------------------
 1155  C
 1156  C         AUTHOR:          ERIC L. STROBEL
 1157  C
 1158  C         DATE:            09/01/87
 1159  C
 1160  C         VERSION:         2.0
 1161  C
 1162  C-------------------------------------------------------------
 1163  C
 1164  C         REVISED:         10/09/86 -- V1.0.  Initial revision.
 1165  C
 1166  C                          09/01/87 -- V2.0.  Uses KEYBRD and accepts
 1167  C                          a period '.' as a response (for VAX batching)
 1168  C
 1169  C-------------------------------------------------------------
 1170  C
 1171  C         This routine displays a 'Press RETURN to continue.' prompt.
 1172  C            Upon receiving a RETURN, the screen is cleared.  After
 1173  C            two botched attempts, if the user types something out
 1174  C            of the ordinary, the routine gently reminds the user to
 1175  C            stick to typing RETURN.  IMPORTANT FEATURES:  1) The
 1176  C            user may abort by typing 'A' (for abort) in response to
 1177  C            the 'Press ...' prompt.  This provides a mechanism to
 1178  C            allow the user to have a chance to avoid doing something
 1179  C            irrevocable if the choice was made by mistake.  2) In
 1180  C            an effort to allow the program to be run under VMS in
 1181  C            batch mode, a response of '.' & RETURN has the same effect
 1182  C            as just typing RETURN.  For whatever reason, a VMS batch
 1183  C            file cannot contain a blank line.
 1184  C
 1185  C         Please note that this routine was largely written before the
 1186  C            adoption of the variable naming convention used throughout
 1187  C            most of the rest of the program.  This routine is brief
 1188  C            enough that it has proven so far unnecessary to change it.
 1189  C
 1190  C -------------------------------------------------------------
 1191  C
 1192          CHARACTER*40 PRMT
```

```
Line#  Source Line

 1193           CHARACTER*10 CHTMP
 1194           CHARACTER*1 XXX
 1195           INTEGER YYY, ZZZ, ITMP, ITYPE
 1196           REAL*8 DTMP
 1197           XXX = ' '
 1198    50     PRMT = ' Press RETURN to continue. '
 1199           CALL KEYBRD(PRMT,3,YYY,ITMP,DTMP,XXX)
 1200           IF (YYY.LT.1.OR.XXX.EQ.'A'.OR.XXX.EQ.'.') THEN
 1201               WRITE(6,300)
 1202   300         FORMAT(////////////////////////////////////////////////
 1203               RETURN
 1204           ELSE
 1205               ZZZ = ZZZ + 1
 1206               IF (ZZZ.GT.2) PRINT *,' HIT RETURN ONLY TO CONTINUE.'
 1207               GO TO 50
 1208           ENDIF
 1209           RETURN
 1210           END
```

IOXPRET Local Symbols

Name	Class	Type	Size
XXX	param		
PRMT.	local	CHAR*40	40
YYY	local	INTEGER*4	4
ZZZ	local	INTEGER*4	4
DTMP.	local	REAL*8	8
ITMP.	local	INTEGER*4	4

```
Line#  Source Line

 1212           SUBROUTINE KEYBRD(PSTR,ITYP,LN,IVAR,FVAR,CVAR)
 1213   C
 1214   C-------------------------------------------------------------
 1215   C
 1216   C    KEYBRD -- SUBROUTINE TO TAKE KEYBRD INPUT INTELLIGENTLY AND
 1217   C         PARSE IT.
 1218   C
 1219   C         CALLED BY: MAIN, DOxDATA, IOxPRET
 1220   C
 1221   C-------------------------------------------------------------
 1222   C
 1223   C         AUTHOR:          ERIC L. STROBEL
 1224   C
 1225   C         DATE:            09/01/87
 1226   C
 1227   C         VERSION:         1.0
 1228   C
 1229   C-------------------------------------------------------------
 1230   C
 1231   C         REVISED:         09/01/87 -- V1.0.  Initial revision.
 1232   C
 1233   C-------------------------------------------------------------
 1234   C
 1235   C         USES:         PSTR          The prompt string.
 1236   C
 1237   C                       ITYP          The type of answer expected to be
 1238   C                                     returned.  1 = INT, 2 = FLOAT,
 1239   C                                     3 = CHAR.
 1240   C
 1241   C
 1242   C         To present the user with a prompt and to discern what
 1243   C              sort of response is expected.  This routine attempts
 1244   C              to be somewhat intelligent about parsing the typed
 1245   C              response.  Real values entered without a decimal
 1246   C              point are actually read correctly, and strings are
 1247   C              converted to all CAPS so that the response typed in
 1248   C              appears to be case insensitive.  This is a modification
 1249   C              of a routine authored by Frank Rhoads.  Please note the
 1250   C              comment block below refering to where changes need to
 1251   C              be made in converting from system to system.
 1252   C
 1253   C
 1254   C         RETURNS:      LN            The number of characters typed on
 1255   C                                     the keyboard.
 1256   C
 1257   C                       IVAR          The integer value to be passed.
 1258   C
 1259   C                       FVAR          The floating value to be passed.
 1260   C
 1261   C                       CVAR          The character string to be passed.
```

```
Line#  Source Line

 1262  C
 1263  C-----------------------------------------------------------------
 1264  C
 1265          CHARACTER*(*)CVAR, PSTR
 1266          CHARACTER*20 CVR
 1267          CHARACTER*8 IFMT
 1268          CHARACTER*1 BYTS(20)
 1269          EQUIVALENCE(BYTS,CVR)
 1270          INTEGER*4 IVAR,LN
 1271          REAL*8 FVAR
 1272          CVR='
 1273      9   LN=0
 1274  C
 1275  C ----------------------------------------------------------------
 1276  C     Display the appropriate prompt, including a default value.
 1277  C
 1278  C     This is also where the changes must be made in translating
 1279  C         from system to system.  In the three WRITEs below, the
 1280  C         formatting ends with ... " ,2H):" and then whatever is
 1281  C         the appropriate thing to suppress the return to the
 1282  C         beginning of the next line.  On the VAX (VMS) this is
 1283  C         ",$".  With Microsoft Fortran (DOS) it is ",\", and with
 1284  C         RyanMcFarland Fortran (DOS) it is nothing at all.
 1285  C ----------------------------------------------------------------
 1286  C
 1287          IF (ITYP.EQ.1) THEN
 1288               WRITE(6,'(1X,A40,1H(,I4,2H):,\)') PSTR, IVAR
 1289          ELSE IF (ITYP.EQ.2) THEN
 1290               WRITE(6,'(1X,A40,1H(,G20.9,2H):,\)') PSTR, FVAR
 1291          ELSE
 1292               WRITE(6,'(1X,A40,1H(,A10,2H):,\)') PSTR, CVAR
 1293          ENDIF
 1294  C
 1295  C     Get the user's response as a character string.
 1296  C
 1297          READ(5,'(A20)')CVR
 1298  C
 1299  C     Find the length of the response.
 1300  C
 1301          DO 10 I=1,20
 1302            L=21-I
 1303            IF(BYTS(L).NE.' ')GO TO 11
 1304     10     CONTINUE
 1305            L=0
 1306     11     LN=L
 1307  C
 1308  C     No response, so Return key must have been typed.
 1309  C
 1310          IF(LN.EQ.0)RETURN
 1311          CVAR=CVR
```

```
Line#  Source Line

 1312           IF(ITYP.LT.3)THEN
 1313  C
 1314  C      If the expected answer is a numeric type, then read it
 1315  C          from the internal string.
 1316  C
 1317             IF(ITYP.EQ.1)THEN
 1318               WRITE(IFMT,'(2H(I,I2,1H))')LN
 1319               READ(CVR,IFMT,ERR=12)IVAR
 1320             ELSE
 1321               WRITE(IFMT,'(2H(F,I2,3H.0))')LN
 1322               READ(CVR,IFMT,ERR=12)FVAR
 1323             ENDIF
 1324           ELSE
 1325  C
 1326  C      For a character string answer, convert to all caps.
 1327  C
 1328    20       DO 21 ICI=1,LN
 1329               K=Ichar(BYTS(ICI))
 1330               If(k.GE.96)THEN
 1331                 K=K-32
 1332                 CVAR(ICI:ICI)=Char(K)
 1333               ELSE
 1334                 CVAR(ICI:ICI)=BYTS(ICI)
 1335               ENDIF
 1336    21       CONTINUE
 1337           ENDIF
 1338           RETURN
 1339    12     WRITE(6,'(37H Conversion ERROR, Please RETRY Input)')
 1340           GO TO 9
 1341   100     RETURN
 1342           END


KEYBRD  Local Symbols

Name                          Class    Type              Size

CVAR. . . . . . . . . . . . . param
FVAR. . . . . . . . . . . . . param
IVAR. . . . . . . . . . . . . param
LN. . . . . . . . . . . . . . param
ITYP. . . . . . . . . . . . . param
PSTR. . . . . . . . . . . . . param
I . . . . . . . . . . . . . . local    INTEGER*4          4
K . . . . . . . . . . . . . . local    INTEGER*4          4
L . . . . . . . . . . . . . . local    INTEGER*4          4
ICI . . . . . . . . . . . . . local    INTEGER*4          4
CVR . . . . . . . . . . . . . local    CHAR*20           20
IFMT. . . . . . . . . . . . . local    CHAR*8             8
BYTS. . . . . . . . . . . . . local    CHAR*1            20
```

Global Symbols

Name	Class	Type	Size
DOXDATA	FSUBRT	***	***
IOXMENU	FSUBRT	***	***
IOXPRET	FSUBRT	***	***
KEYBRD.	FSUBRT	***	***
MAINDAT	common	***	86596
RAYSUB.	extern	***	***
RESULTS	common	***	1320
main.	FSUBRT	***	***

Code size = 56db (22235)
Data size = 0a16 (2582)

```
Line#  Source Line

     1          SUBROUTINE RAYSUB(RVxAZI, RVxELEV)
     2  C
     3  C ------------------------------------------------------------
     4  C
     5  C   RAYSUB  --  3D RAYTRACING WITH ACCELERATED FREE SPACE
     6  C                 PROPAGATION CALCULATION
     7  C
     8  C   CALLED BY:    MAIN
     9  C
    10  C   CALLS:        Almost everything else, as this is really
    11  C                 the core of the program.
    12  C
    13  C ------------------------------------------------------------
    14  C
    15  C   AUTHORS:     MICHAEL H. REILLY & ERIC L. STROBEL
    16  C
    17  C   DATE:        03/18/88
    18  C
    19  C   VERSION:     4.3
    20  C
    21  C ------------------------------------------------------------
    22  C
    23  C   REVISED:     07/25/86 -- Initial revision.  Translated
    24  C                from Tektronix Basic to VAX Fortran by
    25  C                Eric L. Strobel.
    26  C
    27  C                07/30/86 -- V1.1.  Change over to use of
    28  C                REAL*8 precision in the calculations.
    29  C
    30  C                08/12/86 -- V2.0.  Add the capability to
    31  C                perform propagation loss calculations.
    32  C                Also greatly massaged over the output.
    33  C                A number of minor and MAJOR bugs corrected.
    34  C
    35  C                09/05/86 -- V2.3.  Add horizon focusing and
    36  C                handle low angle rays.  Record data for
    37  C                proper handling of skip focusing.  Record
    38  C                maximum height.  Calculate deviation from
    39  C                great circle path.  Minor bug fixes.
    40  C
    41  C                09/17/86 -- V2.4.  Fix calculation of great
    42  C                circle path deviation.  Introduce cutoff in
    43  C                number of bounces.  Automagically obtain
    44  C                approximate bottom of the ionosphere, elim-
    45  C                inating the need for the prompt.  Also fixed
    46  C                rotation matrix used in calculating the new
    47  C                c-values.
    48  C
    49  C                10/10/86 -- V3.0.  Former program, now demoted
    50  C                to a subroutine.  Appropriate changes made.
```

```
Line#  Source Line

   51  C                  Duties performed by subroutine GEC now done
   52  C                  elsewhere.  Portion of IONOPAR that searches
   53  C                  for triangles of points now streamlined so
   54  C                  that the search won't outlast the Earth!
   55  C
   56  C                  09/01/87 -- V4.0.  The essentials have been
   57  C                  listed in the comment block at the beginning
   58  C                  of MAIN (RAYTRA4.FOR), so you can read about
   59  C                  the changes there.
   60  C
   61  C                  09/30/87 -- V4.1.  Phase path calcs. added.
   62  C
   63  C                  01/15/88 -- V4.2.  The (incorrect) derivative
   64  C                  values are no longer reported.  One of the rays
   65  C                  in the bundle has been deleted, for speed's sake.
   66  C                  The range cutoff has been tweaked so that it will
   67  C                  occur nearly exactly, not just at the end of the
   68  C                  next increment.  Some of the other changes have
   69  C                  been discussed in the comment block at the
   70  C                  beginning of MAIN (RAYTRA4.FOR).
   71  C
   72  C                  03/18/88 -- V4.3.  Corrections made for angular range.
   73  C                  (see comment block at beginning of MAIN)  In the main
   74  C                  body of this routine, the azimuth is the running
   75  C                  value from the launch point to the previous raypoint.
   76  C                  The C's are now normalized at every step, which should
   77  C                  enhance the accuracy.  Some other minor tweaking.
   78  C
   79  C -------------------------------------------------------
   80  C
   81  C       USES:      RVxAZI           Ray launch azimuth
   82  C
   83  C                  RVxELEV          Ray launch elevation
   84  C
   85  C                  /MAINDAT/        A common block the ionospheric
   86  C                                    and user data that was read in
   87  C                                    the previous routines.
   88  C
   89  C             To perform 3-D raytracing of radio propaga-
   90  C         tion through the ionosphere.  Incorporates a spec-
   91  C         ific model for true height profiles of electron
   92  C         density.  Does propagation loss due to focusing and
   93  C         to reflection from the ground.  Doesn't (yet) incorporate
   94  C         magnetic field effects, or deviative absorption effects.
   95  C
   96  C       RETURNS:  RVxBOUN         The array of results at the
   97  C                                   earth impact and ray end points.
   98  C
   99  C -------------------------------------------------------
  100  C
```

```
Line#  Source Line

  101           INTEGER IVxPARAM(6), IVxLAUN(3)
  102           INTEGER IFxBUN, IVxSSNUM, IVxTIME(5), IVxNBOU, IFxGRND
  103           INTEGER IVxSCSING, IVxSCTRI1, IVxSCTRI2, IVxSCTRI3
  104           INTEGER ICxNSCP, IVxJ, IVxSX, IVxSY, IVxSZ, IFxN4, IFxGEN
  105           INTEGER IVxNBL, IFxSFL, ITxK, IFxEND
  106           INTEGER IFxPPF, IFxCASE, IFxCCAL
  107   C
  108           LOGICAL LVxRCUT, LVxHCUT, LVxEND
  109   C
  110           REAL*8 RVxGRID(6), RVxIONPT(30,60,6), RVxLAUN(7)
  111           REAL*8 RVxOPTION(7), RVxLA, RVxLO, RVxDD6
  112           REAL*8 RVxBNDL(4,3,11), RVxBOUN(11,15), RVxELEV0
  113           REAL*8 RVxAZI0, RVxYEAR, RVxA100, RVxTIM
  114           REAL*8 RPxPI,RPxDTOR,RPxREARTH,RVxLATSC(1800),RVxLONSC(1800)
  115           REAL*8 RVxFNSQ(1800,3), RVxH(1800,3), RVxSEZGEC(3,3)
  116           REAL*8 RVxFNSB(3), RVxHB(3), RVxCOFMAT(2,2), RVxLAT0
  117           REAL*8 RVxLON0, RVxHBOT, RVxRALIM, RVxANGLIM, RVxFREQ
  118           REAL*8 RVxFSQU, RVxELEV, RVxAZI, RVxRPI, RVxCX, RVxCY
  119           REAL*8 RVxCZ, RVxSINEL, RVxXI, RVxYI, RVxZI, RVxH0, RVxH5
  120           REAL*8 RVxGPLTOT, RVxDDSZ1, RTxDZCALC, RVxDELZ, RVxHZ
  121           REAL*8 RVxXF, RVxYF, RVxZF, RTxS4, RVxGPLINC, RVxDDXHB
  122           REAL*8 RVxDDYHB, RVxUDXDS, RVxUDYDS
  123           REAL*8 RVxUDZDS, RVxDXDZ, RVxDYDZ, RVxXGRAD, RVxYGRAD
  124           REAL*8 RVxDET, RTxP1, RTxP2, RTxP3, RVxALPHA
  125           REAL*8 RVxBETA, RVxETA1, RVxETA2, RVxLATI, RVxLONI
  126           REAL*8 RVxLAT1, RVxLON1, RVxX7, RVxY7, RVxZ7
  127           REAL*8 RVxR1, RVxR2, RVxB5(3), RVxB6(3), RVxA5(3), RVxA6(3)
  128           REAL*8 RVxE5(3), RVxE6(3), RVxV1, RVxV2
  129           REAL*8 RVxLOSA, RVxLOSR, RVxLOSX, RVxLOSG
  130           REAL*8 RVxDR, RVxD2R, RVxDEV, RVxCSUM
  131           REAL*8 RVxHCUT, RPxHTOP, RVxHMIN, RTxTEMP
  132           REAL*8 RVxXX, RVxHBND, RVxPPL, RVxPPI, RVxMU, RVxMU2
  133           REAL*8 RVxF40, RVxF65, RVxK1, RVxHL, RVxCOSEL
  134           REAL*8 RVxRTMP, RVxRMAX, RVxDELR, RVxDELR2
  135           REAL*8 RTxAINC, RTxRT1
  136   C
  137   C-----------------------------------------------------------------
  138   C
  139   C     NOTE:  IVxTIME(1) = YEAR; IVxTIME(2) = MONTH
  140   C            IVxTIME(3) = DAY;  IVxTIME(4) = HOUR
  141   C            IVxTIME(5) = MINUTE
  142   C
  143   C            RVxBNDL = RVxBNDL( RAY 1,2,3,OR 4; 1-LAT,2-LON,3-ALT;
  144   C                              # OF BOUNCES)
  145   C
  146   C            RVxBOUN = RVxBOUN(N-TH BOUNCE; PARAMS),
  147   C          PARAMS--          1 -> LAT
  148   C                            2 -> LON
  149   C                            3 -> GROUP PATH LENGTH
  150   C                            4 -> ANGULAR RANGE
```

```
Line#  Source Line

  151  C                          5 -> DB DOWN
  152  C                          6 -> DEVIATION FROM GR. CIRCLE PATH
  153  C                          7 -> UNUSED
  154  C                          8 -> UNUSED
  155  C                          9 -> HEIGHT OF THE DATA POINT
  156  C                         10 -> PHASE PATH LENGTH
  157  C                         11 -> X DIRECTION COSINE OF RAY
  158  C                         12 -> Y DIRECTION COSINE OF RAY
  159  C                         13 -> Z DIRECTION COSINE OF RAY
  160  C                      14,15 -> RESERVED FOR FUTURE USE
  161  C
  162  C-------------------------------------------------------------------
  163  C
  164           COMMON /MAINDAT/ RVxGRID, RVxIONPT, IVxPARAM, IVxLAUN
  165       #,RVxLAUN ,RVxOPTION
  166           COMMON /RESULTS/ RVxBOUN
  167           COMMON /LPARM/ IVxSSNUM,IVxTIME,RVxYEAR,RVxA100,IFxGRND
  168       *,RVxTIM
  169           COMMON /LOSSES/ RVxLOSA, RVxLOSR, RVxLOSX, RVxLOSG
  170           COMMON /PRAM/ RPxPI, RPxDTOR, RPxREARTH, RPxHTOP
  171           COMMON /SCPS1/ ICxNSCP, RVxLATSC, RVxLONSC
  172           COMMON /SCPS1A/ RVxFNSQ,RVxH
  173           COMMON /OTHER/ RVxLAT1,RVxLON1,RVxHBOT,RVxFSQU,RVxRPI,RVxHCUT
  174           COMMON /MISC/ RVxSEZGEC,RVxX7,RVxY7,RVxZ7,RVxR1,RVxR2,RVxHMIN
  175           COMMON /START/ RVxXI, RVxYI, RVxZI, RVxLATI, RVxLONI
  176           COMMON /END/ RVxXF, RVxYF, RVxZF, RVxH5
  177           COMMON /IONO1/ RVxALPHA, RVxBETA,RVxETA1,RVxETA2,RVxFNSB,RVxHB
  178           COMMON /IONO2/ RVxA5, RVxA6, RVxB5, RVxB6, RVxE5, RVxE6
  179           COMMON /IONO3/ RVxV1,RVxV2,RVxXX,IFxCASE, RVxHBND
  180           COMMON /MORE/ IVxSX, IVxSY, IVxSZ, RVxCX, RVxCY, RVxCZ
  181           COMMON /TEMP1/ RVxF40, RVxF65, RVxK1, RVxHL
  182           COMMON /GORP/ RVxDD6, RVxANGLIM
  183  C
  184  C         Initialize things
  185  C
  186           DO 9000 I=1,5
  187               IVxTIME(I) = IVxPARAM(I+1)
  188   9000    CONTINUE
  189           RPxPI = 3.141592654D00
  190           RPxDTOR = 0.0174532925D00
  191           RPxREARTH = 6371.2D00
  192           RPxHTOP = 2000.0D00
  193           RVxHMIN = RVxLAUN(7)
  194           ITxK = 0
  195  C
  196  C        Initialize some things that are specific to the RADAR-C
  197  C          model.
  198  C
  199           RVxHBOT = 40.0D00
  200           RVxF40 = 4.03D-04
```

```
Line#  Source Line

  201           RVxK1 = 0.12D00
  202           RVxF65 = RVxF40*DEXP(25.0D00 * RVxK1)
  203           RVxHL = 120.48384357
  204   C
  205   C    Begin input section, i.e. break out the arrays that will
  206   C     be needed from the large array.
  207   C
  208   10000   ICxNSCP = IDNINT(RVxGRID(5)*RVxGRID(6))
  209           DO 10200 I=1, RVxGRID(5)
  210               RVxLA = RVxGRID(3) + (I-1)*RVxGRID(1)
  211               DO 10100 J = 1, RVxGRID(6)
  212                   RVxLO = RVxGRID(4) + (J-1)*RVxGRID(2)
  213                   ITxK = ITxK + 1
  214                   RVxLATSC(ITxK) = RVxLA
  215                   RVxLONSC(ITxK) = RVxLO
  216                   RVxFNSQ(ITxK,1) = RVxIONPT(I,J,1)
  217                   RVxFNSQ(ITxK,2) = RVxIONPT(I,J,3)
  218                   RVxFNSQ(ITxK,3) = RVxIONPT(I,J,6)
  219                   RVxH(ITxK,1) = RVxIONPT(I,J,2)
  220                   RVxH(ITxK,2) = RVxIONPT(I,J,5)
  221                   RVxH(ITxK,3) = RVxIONPT(I,J,4)
  222                   RVxLATSC(ITxK) = RVxLATSC(ITxK) * RPxDTOR
  223                   RVxLONSC(ITxK) = RVxLONSC(ITxK) * RPxDTOR
  224   10100       CONTINUE
  225   10200   CONTINUE
  226   C
  227   C        More initializing.
  228   C
  229           RVxLAT0 = RVxLAUN(1)
  230           RVxLON0 = RVxLAUN(2)
  231           RVxLAT0 = RVxLAT0 * RPxDTOR
  232           RVxLON0 = RVxLON0 * RPxDTOR
  233           RVxRALIM = RVxLAUN(3)
  234           RVxANGLIM = RVxRALIM/RPxREARTH
  235           RTxS4 = 0.0D00
  236           IVxSSNUM = IVxPARAM(1)
  237           IFxGRND = IVxLAUN(3)
  238           IFxPPF = 1
  239           IVxNBL = IVxLAUN(1)
  240           IF (IVxNBL.GT.10) IVxNBL = 10
  241           IF (IVxNBL.LT.0) IVxNBL = 0
  242           IFxSFL = IVxLAUN(2)
  243           RVxFREQ = RVxLAUN(5)
  244           RVxFSQU = RVxFREQ * RVxFREQ
  245           RVxRPI = RVxLAUN(4)
  246           RVxHCUT = RVxLAUN(7)
  247           IF (RVxELEV.EQ.0.0D00) RVxELEV = 0.01D00
  248   C
  249   C    BEGIN CALCULATIONAL SECTION
  250   C
```

```
Line#  Source Line

  251  20000    RVxELEV = RVxELEV * RPxDTOR
  252           RVxAZI = RVxAZI * RPxDTOR
  253           RVxELEV0 = RVxELEV
  254           RVxAZI0 = RVxAZI
  255           IVxJ = 0
  256           IFxCCAL = 0
  257           RVxDD6 = RVxRPI
  258           RVxLOSR = 0.0D00
  259           RVxLOSG = 0.0D00
  260           RVxLOSA = 0.0D00
  261  20020    IFxBUN = 3
  262           IF (IFxSFL.EQ.1) IFxBUN = 0
  263  C
  264  C         Determine the elevation & azimuth for each ray in the bundle.
  265  C
  266           RTxAINC = 0.00125D00 * RPxDTOR
  267  20050    IF (IFxBUN.EQ.3) THEN
  268               RVxAZI = RVxAZI0 + RTxAINC
  269               RVxELEV = RVxELEV0 + RTxAINC
  270           ELSE IF (IFxBUN.EQ.2) THEN
  271               RVxAZI = RVxAZI0 + RTxAINC
  272               RVxELEV = RVxELEV0 - RTxAINC
  273           ELSE IF (IFxBUN.EQ.1) THEN
  274               RVxAZI = RVxAZI0 - RTxAINC
  275               RVxELEV = RVxELEV0 - RTxAINC
  276           ELSE
  277               RVxELEV = RVxELEV0
  278               RVxAZI = RVxAZI0
  279           ENDIF
  280  C
  281  C         Still more initializing.
  282  C
  283           IVxNBOU = 0
  284           IVxJ = 0
  285           RVxLAT1 = RVxLAT0
  286           RVxLON1 = RVxLON0
  287           RVxH0 = RVxLAUN(6)
  288           RVxXI = 0.0D00
  289           RVxYI = 0.0D00
  290           RVxZI = RVxH0
  291           IVxSZ = INTSIGN(DSIN(RVxELEV))
  292           RVxH5 = RVxH0
  293           RTxTEMP = 0.0D00
  294  C
  295  C         Do the initial coordinate setup and transformations.
  296  C
  297           CALL ROTSEZ
  298           CALL LATLON3
  299           IFxN4 = 1
  300           IFxGEN = 1
```

```
Line#  Source Line

  301  C
  302  C         Get the initial values for the ionospheric parameters
  303  C          and (if any) gradients.
  304  C
  305             CALL IONOPAR(IVxJ, IFxN4, IFxGEN, IVxSCSING, IVxSCTRI1
  306           *, IVxSCTRI2, IVxSCTRI3)
  307             RVxMU2 = 1.0D00 - RVxXX/RVxFSQU
  308             IF (RVxMU2.LT.0.0D00) THEN
  309                 PRINT *, ' RAY DOES NOT PROPAGATE!!!!'
  310                 STOP
  311             ENDIF
  312             RVxCOSEL = DCOS(RVxELEV)
  313             RVxSINEL = DSIN(RVxELEV)
  314             RVxMU = DSQRT(RVxMU2)
  315             RVxCX = -RVxMU * RVxCOSEL * DCOS(RVxAZI)
  316             IVxSX = INTSIGN(RVxCX)
  317             RVxCY = RVxMU * RVxCOSEL * DSIN(RVxAZI)
  318             IVxSY = INTSIGN(RVxCY)
  319             RVxCZ = RVxMU * RVxSINEL
  320             IVxSZ = INTSIGN(RVxCZ)
  321             RVxCX = RVxCX * RVxCX
  322             RVxCY = RVxCY * RVxCY
  323             RVxCZ = RVxCZ * RVxCZ
  324             RVxPPL = 0.0D00
  325             RVxGPLTOT = 0.0D00
  326             IVxIII = 0
  327  C
  328             LVxEND = .FALSE.
  329             LVxHCUT = .FALSE.
  330             LVxRCUT = .FALSE.
  331  C
  332  C          Begin the ray increment by estimating the z increment
  333  C           for the upcoming raypath increment.
  334  C
  335  20106      RVxDDSZ1 = IVxSZ * DSQRT( RVxCZ / (RVxCX + RVxCY + RVxCZ) )
  336             IF (RVxALPHA.NE.0.0D00.AND.RVxBETA.NE.0.0D00) THEN
  337  C
  338  C          In this case, the index of refraction is quadratic in z, so
  339  C            an additional correction is appropriate.
  340  C
  341                 RTxDZCALC = (RVxDDSZ1 - RVxSINEL) / RVxDD6
  342             ELSE
  343                 RTxDZCALC = 0.0D00
  344             ENDIF
  345             RVxDELZ = IVxSZ * RVxRPI * (RVxDDSZ1 + RTxDZCALC*RVxRPI/2.0D00
  346             RVxDELZ = DABS(RVxDELZ)
  347  C
  348  C          This section is here to ensure that the calculation cuts off
  349  C            in range almost exactly when it is supposed to.
  350  C
```

```
Line#  Source Line

  351           RVxDELR = DSQRT(RVxRPI*RVxRPI - RVxDELZ*RVxDELZ)
  352           RVxRTMP = (RTxS4 + RVxDELR/(RVxH0+RPxREARTH)) * RPxREARTH
  353           RVxRMAX = RVxANGLIM * RPxREARTH
  354           IF (RVxRTMP.GE.RVxRMAX) THEN
  355               RVxDELR2 = RVxDELR - ((RVxRTMP-RVxRMAX)*(RVxH0+RPxREARTH)
  356         #/RPxREARTH)
  357               RVxDELZ = RVxDELZ * RVxDELR2 / RVxDELR
  358           ENDIF
  359   C
  360   C       The angle of ray incidence at H = 100 km.  This is used in the
  361   C         (currently commented out) absorption loss calculation.
  362   C
  363   C       IF (DABS(RVxHZ - 100.0D00).LT.(RVxRPI/2.0D00)) RVxA100 =
  364   C     #DCOS(RVxDELZ/RVxRPI)
  365   C
  366           RVxDELZ = DMAX1( RVxDELZ, 1.0D-04)
  367           IFxGEN = 1
  368           RVxHZ = RVxH0 + IVxSZ*RVxDELZ
  369   C
  370   C       Check to see if a boundary has been crossed.
  371   C
  372           IF (IVxSZ.LT.0) THEN
  373               IF (RVxHZ.LT.RVxHBND) THEN
  374                   RVxDELZ = RVxH0 - RVxHBND
  375                   IFxGEN = 2
  376               ENDIF
  377               GO TO 20133
  378           ENDIF
  379           IF (RVxHZ.GT.RVxHBND) THEN
  380               RVxDELZ = RVxHBND - RVxH0
  381               IFxGEN = 2
  382           ENDIF
  383   C
  384   20133   RVxHZ = RVxH0 + IVxSZ*RVxDELZ
  385   C
  386   C        Use the z increment and height to calculate where the
  387   C         ray ends up at.
  388   C
  389           CALL ENDPT(IFxGEN, RVxH0, RVxHZ, RTxS4, RVxGPLINC)
  390           RVxHZ = RVxH0 + RVxZF
  391           RTxRT1 = RPxREARTH + RVxHZ
  392           RVxH5 = DSQRT(RVxXF*RVxXF + RVxYF*RVxYF + RTxRT1*RTxRT1)
  393         #- RPxREARTH
  394           IF (RVxH5.LT.0.0D00) RVxH5 = 0.0D00
  395           IF (RVxH5.LT.RVxHMIN) RVxHMIN = RVxH5
  396   C
  397   C        If desired, calculate the phase path increment.  The
  398   C         value of IFxPPF is currently 'hard-wired' to be 1.
  399   C
  400           IF (IFxPPF.EQ.1) CALL PHSPL(IFxGEN, RVxPPI)
```

```
Line#  Source Line

  401           IVxIII = IVxIII + 1
  402           IFxCCAL = 0
  403           IF (IFxGEN.NE.6.OR.RVxH5.GT.RVxHBOT) GO TO 20150
  404   C
  405   C        Will be FALSE only if the ray is below the bottom of
  406   C         the ionosphere & headed downward.
  407   C
  408           RVxZI = RVxHZ
  409           RVxXI = RVxXF
  410           RVxYI = RVxYF
  411           CALL LATLON3
  412           CALL NEWCS(IFxGEN)
  413           IFxCCAL = 1
  414   C
  415   C   SMALL SECTION TO HANDLE EARTH BOUNCE
  416   C
  417           CALL ANRANG(RVxAZI,RVxLATI,RVxLONI,RVxLAT0,RVxLON0,RTxS4)
  418           IVxNBOU = IVxNBOU + 1
  419   C
  420   C        Check to see if the cutoff in the number of bounces is met.
  421   C
  422           IF (IVxNBOU.GT.IVxNBL) THEN
  423               IVxNBOU = IVxNBOU - 1
  424               LVxEND = .TRUE.
  425               GO TO 20150
  426           ENDIF
  427   C
  428   C ---------------------------------------------------------------
  429   C        If this is the primary ray in the bundle, then all of the
  430   C         information about the bounce point needs to be recorded.
  431   C         otherwise only the parts that will be needed later are
  432   C         stored.
  433   C ---------------------------------------------------------------
  434   C
  435           IF (IFxBUN.EQ.0) THEN
  436               RVxBOUN(IVxNBOU,1) = RVxLATI
  437               RVxBOUN(IVxNBOU,2) = RVxLONI
  438               RVxBOUN(IVxNBOU,3) = RVxGPLTOT + RVxGPLINC
  439               RVxBOUN(IVxNBOU,4) = RTxS4
  440               IF (IFxSFL.NE.1) THEN
  441   C
  442   C       TIMES is a routine which calculates some quantities for the
  443   C         (currently commented out) absorption loss calculation.
  444   C
  445   C                CALL TIMES(RVxBOUN, IVxNBOU, RVxLON0)
  446                    CALL LOSS(RVxBOUN,RVxBNDL,IVxNBOU,RVxLAT0,RVxLON0,
  447        #RVxELEV0,IFxEND)
  448               ENDIF
  449               RVxBOUN(IVxNBOU,5) = RVxLOSR + RVxLOSG
  450               CALL GCDEV(IVxNBOU,RVxAZI0,RVxLAT0,RVxLON0,RVxBOUN,RVxDEV)
```

```
Line#  Source Line

  451              RVxBOUN(IVxNBOU,6) = RVxDEV
  452              RVxBOUN(IVxNBOU,9) = RVxH5
  453              IF (IFxPPF.EQ.1) RVxBOUN(IVxNBOU,10) = RVxPPL + RVxPPI
  454              RVxCSUM = RVxCX + RVxCY + RVxCZ
  455              RVxBOUN(IVxNBOU,11) = IVxSX*DSQRT(RVxCX)/DSQRT(RVxCSUM)
  456              RVxBOUN(IVxNBOU,12) = IVxSY*DSQRT(RVxCY)/DSQRT(RVxCSUM)
  457              RVxBOUN(IVxNBOU,13) = IVxSZ*DSQRT(RVxCZ)/DSQRT(RVxCSUM)
  458          ELSE
  459              RVxBNDL(IFxBUN,1,IVxNBOU) = RVxLATI
  460              RVxBNDL(IFxBUN,2,IVxNBOU) = RVxLONI
  461              RVxBNDL(IFxBUN,3,IVxNBOU) = RVxH5
  462          ENDIF
  463  C
  464  C   END EARTH BOUNCE HANDLING
  465  C
  466  C ----------------------------------------
  467  C
  468  C   Accumulate the group and phase path lengths
  469  C
  470  20150   RVxGPLTOT = RVxGPLTOT + RVxGPLINC
  471          IF (IFxPPF.EQ.1) RVxPPL = RVxPPL + RVxPPI
  472  C
  473  C   TAKE CARE OF BOUNDARY CROSSING
  474  C
  475  C ----------------------------------
  476  C
  477  C        First check for the existence of a boundary crossing.
  478  C
  479          IF (IFxGEN.NE.2) GO TO 30000
  480  C
  481  C        If this is a spherically symmetric case, there are no
  482  C         tilts, so go on.
  483  C
  484          IF (IFxN4.EQ.1) GO TO 30000
  485  C
  486  C        Call the routine that will actually determine the spatial
  487  C         tilt of the boundary.
  488  C
  489          CALL TILTS(RVxDDXHB, RVxDDYHB, IVxJ)
  490  C
  491  C ---------------------------------------------------------------
  492  C        Now, we attempt to correct this particular raypath increment
  493  C         to insure that it ends exactly ON the boundary, and doesn't
  494  C         overshoot it.
  495  C ---------------------------------------------------------------
  496  C
  497          RVxUDXDS = IVxSX * DSQRT(RVxCX - RVxETA1*RVxXF)
  498          RVxUDYDS = IVxSY * DSQRT(RVxCY - RVxETA2*RVxYF)
  499          RVxUDZDS = IVxSZ * DSQRT(RVxCZ - RVxALPHA*RVxZF + RVxBETA*RVxZF*
  500       *RVxZF)
```

```
Line#  Source Line

  501           RVxDXDZ = RVxUDXDS / RVxUDZDS
  502           RVxDYDZ = RVxUDYDS / RVxUDZDS
  503           RVxXGRAD = RVxDDXHB
  504           RVxYGRAD = RVxDDYHB
  505           RVxCOFMAT(1,1) = 1.0D00 - RVxDXDZ * RVxXGRAD
  506           RVxCOFMAT(1,2) = -RVxDXDZ * RVxYGRAD
  507           RVxCOFMAT(2,1) = -RVxDYDZ * RVxXGRAD
  508           RVxCOFMAT(2,2) = 1.0D00 - RVxDYDZ * RVxYGRAD
  509           RVxDET = RVxCOFMAT(1,1)*RVxCOFMAT(2,2) - RVxCOFMAT(1,2)*
  510         *RVxCOFMAT(2,1)
  511           RTxP1 = (RVxXF*RVxCOFMAT(2,2) - RVxYF*RVxCOFMAT(1,2))/RVxDET
  512           RTxP2 = (RVxCOFMAT(1,1)*RVxYF - RVxCOFMAT(2,1)*RVxXF)/RVxDET
  513           RTxP3 = RVxZF + RVxXGRAD*RTxP1 + RVxYGRAD*RTxP2
  514           RVxXF = RTxP1
  515           RVxYF = RTxP2
  516   C
  517   C       Now, correct the group and phase path length totals.
  518   C
  519           RVxGPLTOT = RVxGPLTOT + (RTxP3 - RVxZF)/RVxUDZDS
  520           IF (IFxPPF.EQ.1) THEN
  521               RVxPPL = RVxPPL + (RVxUDXDS*RVxUDXDS + RVxUDYDS*RVxUDYDS
  522         #+ RVxUDZDS*RVxUDZDS) * (RTxP3 - RVxZF)/RVxUDZDS
  523           ENDIF
  524   C
  525           RVxZF = RTxP3
  526           RVxHZ = RVxH0 + RVxZF
  527           RTxRT1 = RPxREARTH + RVxHZ
  528           RVxH5 = DSQRT(RVxXF*RVxXF + RVxYF*RVxYF + RTxRT1*RTxRT1)
  529         #- RPxREARTH
  530           IF (RVxH5.LT.0.0D00) RVxH5 = 0.0D00
  531   C
  532   C   BEGIN UPDATE FOR NEXT ITERATION.
  533   C
  534   30000   RVxXI = RVxXF
  535           RVxYI = RVxYF
  536           RVxZI = RVxHZ
  537   30003   CALL LATLON3
  538           LVxHCUT = (RVxH5.GE.RVxHCUT.AND.IVxSZ.GT.0)
  539           IF (IFxCCAL.EQ.0.OR.LVxHCUT) CALL NEWCS(IFxGEN)
  540   C
  541   C       If it is the end of the problem, there's a bunch of stuff
  542   C         that doesn't need to be done, so skip it.
  543   C
  544           IF (LVxHCUT.OR.LVxEND) GO TO 30300
  545   30203   CALL IONOPAR(IVxJ, IFxN4, IFxGEN, IVxSCSING, IVxSCTRI1
  546         *, IVxSCTRI2, IVxSCTRI3)
  547   C
  548   C        Renormalize the c-values in order to reduce the errors that
  549   C          might otherwise propagate through the program.
  550   C
```

```
Line#  Source Line

  551           RVxCSUM = RVxCX + RVxCY + RVxCZ
  552           RVxMU2 = 1.0D00 - RVxXX/RVxFSQU
  553           RVxCX = RVxCX * RVxMU2 / RVxCSUM
  554           RVxCY = RVxCY * RVxMU2 / RVxCSUM
  555           RVxCZ = RVxCZ * RVxMU2 / RVxCSUM
  556   C
  557           RVxH0 = RVxH5
  558           RVxLAT1 = RVxLATI
  559           RVxLON1 = RVxLONI
  560           RVxSINEL = RVxDDSZ1
  561           CALL ROTSEZ
  562   30300   CALL ANRANG(RVxAZI,RVxLATI,RVxLONI,RVxLAT0,RVxLON0,RTxS4)
  563   C
  564   C       Evaluate the problem cutoff criteria.
  565   C
  566           LVxRCUT = (RTxS4.GE.RVxANGLIM)
  567           LVxHCUT = (RVxH5.GE.RVxHCUT.AND.IVxSZ.GT.0)
  568   C        WRITE(70,30301) RTxS4*RPxREARTH, RVxH5
  569   C30301   FORMAT(1X,G18.8,G18.8)
  570   C
  571   C       Check for whether the ray has reached a cutoff condition.
  572   C
  573           IF (LVxRCUT.OR.LVxHCUT.OR.LVxEND) THEN
  574   C
  575   C -------------------------------------------------------------
  576   C        A cutoff criterion for this problem has been met.  If it
  577   C         is not the primary ray, record the necessary numbers and
  578   C         re-cycle for the next ray in the bundle, otherwise,
  579   C         record all the information on the endpoint and return to
  580   C         MAIN.
  581   C -------------------------------------------------------------
  582   C
  583             IF (IFxBUN.NE.0) THEN
  584                 RVxBNDL(IFxBUN,1,IVxNBOU+1) = RVxLATI
  585                 RVxBNDL(IFxBUN,2,IVxNBOU+1) = RVxLONI
  586                 RVxBNDL(IFxBUN,3,IVxNBOU+1) = RVxH5
  587                 IFxBUN = IFxBUN - 1
  588                 RVxHMIN = RVxLAUN(7)
  589                 GO TO 20050
  590             ENDIF
  591             IFxEND = 1
  592             RVxLAT1 = RVxLATI
  593             RVxLON1 = RVxLONI
  594             CALL ROTSEZ
  595             IVxNBOU = IVxNBOU + 1
  596             RVxBOUN(IVxNBOU,1) = RVxLATI
  597             RVxBOUN(IVxNBOU,2) = RVxLONI
  598             RVxBOUN(IVxNBOU,3) = RVxGPLTOT
  599             RVxBOUN(IVxNBOU,4) = RTxS4
  600             IF (IFxSFL.NE.1) THEN
```

```
 Line#  Source Line

   601  C
   602  C        TIMES is called to calculate some values needed by the
   603  C          (currently commented out) absorption loss calculation.
   604  C
   605  C                CALL TIMES(RVxBOUN, IVxNBOU, RVxLON0)
   606                   CALL LOSS(RVxBOUN,RVxBNDL,IVxNBOU,RVxLAT0,RVxLON0,
   607         #RVxELEV0,IFxEND)
   608                ENDIF
   609                RVxBOUN(IVxNBOU,5) = RVxLOSG + RVxLOSR
   610                CALL GCDEV(IVxNBOU,RVxAZI0,RVxLAT0,RVxLON0,RVxBOUN,RVxDEV)
   611                RVxBOUN(IVxNBOU,6) = RVxDEV
   612                RVxBOUN(IVxNBOU,9) = RVxH5
   613                RVxBOUN(IVxNBOU,10) = RVxPPL
   614                RVxCSUM = RVxCX + RVxCY + RVxCZ
   615                RVxBOUN(IVxNBOU,11) = IVxSX*DSQRT(RVxCX)/DSQRT(RVxCSUM)
   616                RVxBOUN(IVxNBOU,12) = IVxSY*DSQRT(RVxCY)/DSQRT(RVxCSUM)
   617                RVxBOUN(IVxNBOU,13) = IVxSZ*DSQRT(RVxCZ)/DSQRT(RVxCSUM)
   618                DO 30500 I=1,IVxNBOU
   619                    RVxBOUN(I,1) = RVxBOUN(I,1)/RPxDTOR
   620                    RVxBOUN(I,2) = RVxBOUN(I,2)/RPxDTOR
   621                    RVxBOUN(I,3) = RVxBOUN(I,3)/300.0D00
   622                    RVxBOUN(I,4) = RVxBOUN(I,4)*RPxREARTH
   623  30500         CONTINUE
   624                RETURN
   625             ENDIF
   626             GO TO 20106
   627             END
```

RAYSUB Local Symbols

Name	Class	Type	Size
RVXELEV	param		
RVXAZI	param		
RTXP2	local	REAL*8	8
ITXK	local	INTEGER*4	4
RTXP3	local	REAL*8	8
IFXGEN	local	INTEGER*4	4
IVXJ	local	INTEGER*4	4
RVXRTMP	local	REAL*8	8
RTXS4	local	REAL*8	8
RVXUDXDS	local	REAL*8	8
I	local	INTEGER*4	4
RVXUDYDS	local	REAL*8	8
J	local	INTEGER*4	4
RVXUDZDS	local	REAL*8	8
IFXSFL	local	INTEGER*4	4
IFXBUN	local	INTEGER*4	4
RVXLA	local	REAL*8	8

RAYSUB Local Symbols

Name	Class	Type	Size
IFXPPF	local	INTEGER*4	4
RVXLAT0	local	REAL*8	8
LVXEND	local	LOGICAL*4	4
IVXIII	local	INTEGER*4	4
IVXNBL	local	INTEGER*4	4
RVXMU2	local	REAL*8	8
RVXAZI0	local	REAL*8	8
RTXRT1	local	REAL*8	8
RTXAINC	local	REAL*8	8
RVXDELR2	local	REAL*8	8
RVXLON0	local	REAL*8	8
RVXGPLTOT	local	REAL*8	8
RVXLO	local	REAL*8	8
RVXELEV0	local	REAL*8	8
RVXDET	local	REAL*8	8
RVXDEV	local	REAL*8	8
RVXBNDL	local	REAL*8	1056
RVXMU	local	REAL*8	8
RVXHZ	local	REAL*8	8
RVXDDSZ1	local	REAL*8	8
RVXDELR	local	REAL*8	8
RVXPPI	local	REAL*8	8
IFXN4	local	INTEGER*4	4
RVXDDXHB	local	REAL*8	8
RVXDDYHB	local	REAL*8	8
IVXNBOU	local	INTEGER*4	4
RVXPPL	local	REAL*8	8
IVXSCTRI1	local	INTEGER*4	4
RVXFREQ	local	REAL*8	8
LVXHCUT	local	LOGICAL*4	4
IVXSCTRI2	local	INTEGER*4	4
RVXDELZ	local	REAL*8	8
RTXDZCALC	local	REAL*8	8
IVXSCTRI3	local	INTEGER*4	4
RTXTEMP	local	REAL*8	8
RVXRALIM	local	REAL*8	8
RVXCOSEL	local	REAL*8	8
RVXXGRAD	local	REAL*8	8
RVXYGRAD	local	REAL*8	8
RVXRMAX	local	REAL*8	8
RVXCSUM	local	REAL*8	8
RVXH0	local	REAL*8	8
LVXRCUT	local	LOGICAL*4	4
RVXDXDZ	local	REAL*8	8
RVXCOFMAT	local	REAL*8	32

RAYSUB Local Symbols

Name	Class	Type	Size
IFXCCAL	local	INTEGER*4	4
RVXDYDZ	local	REAL*8	8
RVXSINEL	local	REAL*8	8
RVXGPLINC	local	REAL*8	8
IFXEND	local	INTEGER*4	4
IVXSCSING	local	INTEGER*4	4
RTXP1	local	REAL*8	8
RVXH	SCPS1A	REAL*8	43200
RVXV1	IONO3	REAL*8	8
RVXV2	IONO3	REAL*8	8
RVXSEZGEC	MISC	REAL*8	72
RVXLOSA	LOSSES	REAL*8	8
RVXFNSB	IONO1	REAL*8	24
RVXHB	IONO1	REAL*8	24
RVXLOSR	LOSSES	REAL*8	8
RVXLOSX	LOSSES	REAL*8	8
RVXLOSG	LOSSES	REAL*8	8
RVXHBOT	OTHER	REAL*8	8
RVXANGLIM	GORP	REAL*8	8
RVXHCUT	OTHER	REAL*8	8
RVXFSQU	OTHER	REAL*8	8
RPXHTOP	PRAM	REAL*8	8
RVXHMIN	MISC	REAL*8	8
RVXRPI	OTHER	REAL*8	8
RVXCX	MORE	REAL*8	8
RVXXX	IONO3	REAL*8	8
RVXCY	MORE	REAL*8	8
RVXCZ	MORE	REAL*8	8
RVXHBND	IONO3	REAL*8	8
IVXPARAM	MAINDAT	INTEGER*4	24
IVXLAUN	MAINDAT	INTEGER*4	12
RVXXI	START	REAL*8	8
RVXYI	START	REAL*8	8
RVXZI	START	REAL*8	8
IVXSSNUM	LPARM	INTEGER*4	4
RVXF40	TEMP1	REAL*8	8
IVXTIME	LPARM	INTEGER*4	20
RVXH5	END	REAL*8	8
RVXF65	TEMP1	REAL*8	8
RVXK1	TEMP1	REAL*8	8
IFXGRND	LPARM	INTEGER*4	4
RVXHL	TEMP1	REAL*8	8
RVXXF	END	REAL*8	8
ICXNSCP	SCPS1	INTEGER*4	4
RVXYF	END	REAL*8	8

RAYSUB Local Symbols

Name	Class	Type	Size
IVXSX	MORE	INTEGER*4	4
RVXZF	END	REAL*8	8
IVXSY	MORE	INTEGER*4	4
IVXSZ	MORE	INTEGER*4	4
IFXCASE	IONO3	INTEGER*4	4
RVXGRID	MAINDAT	REAL*8	48
RVXALPHA	IONO1	REAL*8	8
RVXIONPT	MAINDAT	REAL*8	86400
RVXLAUN	MAINDAT	REAL*8	56
RVXBETA	IONO1	REAL*8	8
RVXOPTION	MAINDAT	REAL*8	56
RVXETA1	IONO1	REAL*8	8
RVXETA2	IONO1	REAL*8	8
RVXLATI	START	REAL*8	8
RVXDD6	GORP	REAL*8	8
RVXLONI	START	REAL*8	8
RVXLAT1	OTHER	REAL*8	8
RVXBOUN	RESULTS	REAL*8	1320
RVXLON1	OTHER	REAL*8	8
RVXX7	MISC	REAL*8	8
RVXY7	MISC	REAL*8	8
RVXYEAR	LPARM	REAL*8	8
RVXZ7	MISC	REAL*8	8
RVXA100	LPARM	REAL*8	8
RVXR1	MISC	REAL*8	8
RVXTIM	LPARM	REAL*8	8
RVXR2	MISC	REAL*8	8
RVXB5	IONO2	REAL*8	24
RPXPI	PRAM	REAL*8	8
RVXB6	IONO2	REAL*8	24
RPXDTOR	PRAM	REAL*8	8
RVXA5	IONO2	REAL*8	24
RPXREARTH	PRAM	REAL*8	8
RVXLATSC	SCPS1	REAL*8	14400
RVXA6	IONO2	REAL*8	24
RVXLONSC	SCPS1	REAL*8	14400
RVXE5	IONO2	REAL*8	24
RVXE6	IONO2	REAL*8	24
RVXFNSQ	SCPS1A	REAL*8	43200

```
Line#  Source Line

  629           SUBROUTINE ROTSEZ
  630   C
  631   C-----------------------------------------------------------------
  632   C
  633   C    ROTSEZ -- SUBROUTINE TO ESTABLISH THE TRANSFORMATION
  634   C         MATRIX FROM S.E.Z. TO G.E.C. COORDINATES
  635   C
  636   C         CALLED BY: RAYSUB
  637   C
  638   C-----------------------------------------------------------------
  639   C
  640   C         AUTHOR:          MICHAEL H. REILLY & ERIC L. STROBEL
  641   C
  642   C         DATE:            07/30/86
  643   C
  644   C         VERSION:         1.1
  645   C
  646   C-----------------------------------------------------------------
  647   C
  648   C         REVISED:         07/25/86 -- INITIAL REVISION.  TRANSLATED
  649   C                          FROM TEKTRONIX BASIC TO VAX FORTRAN BY
  650   C                          ERIC L. STROBEL.
  651   C
  652   C                          07/30/86 -- V1.1.  Change over to use of
  653   C                          REAL*8 precision in the calculations.
  654   C
  655   C-----------------------------------------------------------------
  656   C
  657   C        USES:    RVCxLA           LAT OF START POINT
  658   C                 RVCxLO           LON OF START POINT
  659   C
  660   C         TO CALCULATE THE ELEMENTS OF THE ROTATION MATRIX.
  661   C
  662   C        RETURNS:         RVCxSEZTX(3,3)  THE ROTATION MATRIX
  663   C
  664   C-----------------------------------------------------------------
  665   C
  666           REAL*8 RVCxLA, RVCxLO, RVCxSEZTX(3,3), RTLx1
  667           REAL*8 RTLx2, RTLx3, RTLx4, RTLx5, RTLx6, RTLx7, RTLx8
  668           REAL*8 RTLx9, RTLx10, RTLx11, RTLx12, RTLx13, RTLx14
  669   C
  670           COMMON /MISC/ RVCxSEZTX,RTLx5,RTLx6,RTLx7,RTLx8,RTLx9,RTLx1
  671           COMMON /OTHER/ RVCxLA, RVCxLO,RTLx11,RTLx12,RTLx13,RTLx14
  672   C
  673           RTLx1 = DSIN(RVCxLA)
  674           RTLx2 = DCOS(RVCxLA)
  675           RTLx3 = DSIN(RVCxLO)
  676           RTLx4 = DCOS(RVCxLO)
  677           RVCxSEZTX(1,1) = RTLx1 * RTLx4
  678           RVCxSEZTX(1,2) = -RTLx3
```

```
Line#  Source Line

   679          RVCxSEZTX(1,3) = RTLx2 * RTLx4
   680          RVCxSEZTX(2,1) = RTLx1 * RTLx3
   681          RVCxSEZTX(2,2) = RTLx4
   682          RVCxSEZTX(2,3) = RTLx2 * RTLx3
   683          RVCxSEZTX(3,1) = -RTLx2
   684          RVCxSEZTX(3,2) = 0.0D00
   685          RVCxSEZTX(3,3) = RTLx1
   686          RETURN
   687          END
```

ROTSEZ Local Symbols

Name	Class	Type	Size
RTLX1	local	REAL*8	8
RTLX2	local	REAL*8	8
RTLX3	local	REAL*8	8
RTLX4	local	REAL*8	8
RVCXLA	OTHER	REAL*8	8
RVCXLO	OTHER	REAL*8	8
RVCXSEZTX	MISC	REAL*8	72
RTLX5	MISC	REAL*8	8
RTLX6	MISC	REAL*8	8
RTLX7	MISC	REAL*8	8
RTLX8	MISC	REAL*8	8
RTLX9	MISC	REAL*8	8
RTLX10	MISC	REAL*8	8
RTLX11	OTHER	REAL*8	8
RTLX12	OTHER	REAL*8	8
RTLX13	OTHER	REAL*8	8
RTLX14	OTHER	REAL*8	8

```
Line#  Source Line

  689           SUBROUTINE LATLON3
  690   C
  691   C-----------------------------------------------------------------
  692   C
  693   C    LATLON3 -- SUBROUTINE TO OBTAIN LAT, LON OF THE INITIAL
  694   C          POINT IN THE RAYPATH INCREMENT.
  695   C
  696   C          CALLED BY: RAYSUB
  697   C
  698   C-----------------------------------------------------------------
  699   C
  700   C          AUTHOR:           MICHAEL H. REILLY & ERIC L. STROBEL
  701   C
  702   C          DATE:             07/30/86
  703   C
  704   C          VERSION:          1.1
  705   C
  706   C-----------------------------------------------------------------
  707   C
  708   C          REVISED:          07/25/86 -- INITIAL REVISION.  TRANSLATED
  709   C                            FROM TEKTRONIX BASIC TO VAX FORTRAN BY
  710   C                            ERIC L. STROBEL.
  711   C
  712   C                            07/30/86 -- V1.1.  Change over to use of
  713   C                            REAL*8 precision in the calculations.
  714   C
  715   C-----------------------------------------------------------------
  716   C
  717   C         USES:    RVCxX,Y,Z         INITIAL X, Y, Z FOR THE RAYPATH
  718   C                                              INCREMENT
  719   C                  RVCxMAT(3,3)      SEZ-GEC TRANSFORMATION MATRIX
  720   C                  RPCxR             EARTH RADIUS
  721   C
  722   C          TO CALCULATE THE CORRESPONDING LAT, LON FOR THE GIVEN
  723   C                   POINT.
  724   C
  725   C         RETURNS:          RVCxLA,LO         LAT, LON FOR THE INITIAL
  726   C                                                RAYPATH POINT.
  727   C
  728   C-----------------------------------------------------------------
  729   C
  730           REAL*8 RVCx1, RVCx2, RVCx3, RVCx4, RVCx5, RVCx6
  731           REAL*8 RPCxPI, RPCxDTOR, RPCxR
  732           REAL*8 RVCxMAT(3,3), RVCxX, RVCxY, RVCxZ
  733           REAL*8 RVCxLA, RVCxLO, RTLx1, RTLx2
  734   C
  735           COMMON /START/ RVCxX, RVCxY, RVCxZ, RVCxLA, RVCxLO
  736           COMMON /PRAM/ RPCxPI, RPCxDTOR, RPCxR, RTLx2
  737           COMMON /MISC/ RVCxMAT,RVCx1,RVCx2,RVCx3,RVCx4,RVCx5,RVCx6
  738   C
```

```
 Line#  Source Line

   739           RTLx1 = RPCxR + RVCxZ
   740           RVCx4 = DSQRT(RVCxX*RVCxX + RVCxY*RVCxY + RTLx1*RTLx1)
   741  C
   742  C        RVCx1, 2, & 3 represent the GEC coordinates.
   743  C
   744           RVCx1 = RVCxMAT(1,1)*RVCxX + RVCxMAT(1,2)*RVCxY + RVCxMAT(1,3
   745         *RTLx1
   746           RVCx2 = RVCxMAT(2,1)*RVCxX + RVCxMAT(2,2)*RVCxY + RVCxMAT(2,3
   747         *RTLx1
   748           RVCx3 = RVCxMAT(3,1)*RVCxX + RVCxMAT(3,2)*RVCxY + RVCxMAT(3,3
   749         *RTLx1
   750  C
   751           RVCx5 = DSQRT( RVCx1*RVCx1 + RVCx2*RVCx2 )
   752           RVCxLA = DASIN( RVCx3 / RVCx4 )
   753           RVCxLO = DASIN( RVCx2 / RVCx5 )
   754           IF (RVCx1.GE.0.0D00) RETURN
   755           RVCxLO = -RVCxLO + INTSIGN(RVCx2)*RPCxPI
   756           IF (INTSIGN(RVCx2).EQ.0) RVCxLO = RPCxPI
   757           RETURN
   758           END


LATLON3  Local Symbols

Name                          Class    Type              Size

RTLX1 . . . . . .             local    REAL*8               8
RVCX1 . . . . . .             MISC     REAL*8               8
RVCX2 . . . . . .             MISC     REAL*8               8
RVCX3 . . . . . .             MISC     REAL*8               8
RVCX4 . . . . . .             MISC     REAL*8               8
RVCX5 . . . . . .             MISC     REAL*8               8
RVCX6 . . . . . .             MISC     REAL*8               8
RPCXPI. . . . . .             PRAM     REAL*8               8
RPCXDTOR. . . . . .           PRAM     REAL*8               8
RPCXR . . . . . .             PRAM     REAL*8               8
RVCXMAT . . . . . .           MISC     REAL*8              72
RVCXX . . . . . .             START    REAL*8               8
RVCXY . . . . . .             START    REAL*8               8
RVCXZ . . . . . .             START    REAL*8               8
RVCXLA. . . . . .             START    REAL*8               8
RVCXLO. . . . . .             START    REAL*8               8
RTLX2 . . . . . .             PRAM     REAL*8               8


Global Symbols

Name                          Class    Type              Size

ANRANG. . . . . .             extern   ***               ***
```

Global Symbols

Name	Class	Type	Size
END	common	***	32
ENDPT	extern	***	***
GCDEV	extern	***	***
GORP	common	***	16
INTSIGN	extern	INTEGER*4	***
IONO1	common	***	80
IONO2	common	***	144
IONO3	common	***	36
IONOPAR	extern	***	***
LATLON3	FSUBRT	***	***
LOSS	extern	***	***
LOSSES	common	***	32
LPARM	common	***	52
MAINDAT	common	***	86596
MISC	common	***	120
MORE	common	***	36
NEWCS	extern	***	***
OTHER	common	***	48
PHSPL	extern	***	***
PRAM	common	***	32
RAYSUB	FSUBRT	***	***
RESULTS	common	***	1320
ROTSEZ	FSUBRT	***	***
SCPS1	common	***	28804
SCPS1A	common	***	86400
START	common	***	40
TEMP1	common	***	32
TILTS	extern	***	***

Code size = 1dce (7630)
Data size = 00bf (191)

```
Line#  Source Line

    1          SUBROUTINE IONOPAR(IVSxJ,IFSxN4,IFSxG,IVSxSCS,IVSxSCT1,IVSxSCT2
    2         *,IVSxSCT3)
    3  C
    4  C-----------------------------------------------------------------
    5  C
    6  C    IONOPAR -- SUBROUTINE TO CALCULATE THE PROPERTIES OF THE
    7  C         IONOSPHERE FOR A GIVEN POSITION.
    8  C
    9  C         CALLED BY: RAYSUB
   10  C
   11  C         CALLS: TRIANG, INBOX, INTERP, CASE1-5
   12  C
   13  C-----------------------------------------------------------------
   14  C
   15  C         AUTHORS:          MICHAEL H. REILLY & ERIC L. STROBEL
   16  C
   17  C         DATE:             03/18/88
   18  C
   19  C         VERSION:          3.1
   20  C
   21  C-----------------------------------------------------------------
   22  C
   23  C         REVISED:          07/25/86 -- INITIAL REVISION.  TRANSLATED
   24  C                           FROM TEKTRONIX BASIC TO VAX FORTRAN BY
   25  C                           ERIC L. STROBEL.
   26  C
   27  C                           07/30/86 -- V1.1.  Change over to use of
   28  C                           REAL*8 precision in the calculations.
   29  C
   30  C                           10/10/86 -- V2.0.  Altered to fit the new
   31  C                           status of the old RAYTRACE program as a
   32  C                           subroutine.  Important change in sorting
   33  C                           out which triangle a pt. is in:  With large
   34  C                           arrays of points the old algorithm would
   35  C                           have taken nearly forever (REALLY!).  Since
   36  C                           the points are now being defined on a grid,
   37  C                           the problem is vastly simpler.
   38  C
   39  C                           09/01/87 -- V3.0.  Now incorporates the
   40  C                           RADAR-C ionosphere.  The interpolation
   41  C                           routine has been modularized.  Estimates
   42  C                           of the nearest boundary are now reported.
   43  C
   44  C                           03/18/88 -- V3.1.  Some corrections have been
   45  C                           made so that the manipulation of grid indices
   46  C                           works when the date line is crossed by the
   47  C                           grid.
   48  C
   49  C-----------------------------------------------------------------
   50  C
```

```
Line#  Source Line

    51  C        USES:    IVSxJ             J-TH IONOSPHERE LAYER
    52  C                 IFSxN4,G          FLAGS
    53  C                 RVCxX,Y,Z7        INTERMEDIATE COORD. VALUES FROM
    54  C                                      LAT, LON COMPUTATION
    55  C                 RVCxR1            ANOTHER INTERMEDIATE VALUE
    56  C                 RVCxLAI,LOI       INITIAL PT'S. LAT, LON
    57  C                 RVCxH5            NEW HEIGHT
    58  C                 ICCxN             NUMBER OF S.C.P.'S
    59  C                 RPCxRE            RADIUS OF THE EARTH
    60  C                 RVCxFN(1800,3)    X
    61  C                 RVCxH(1800,3)     X
    62  C                 RVCxLA(1800)      X-DATA FOR THE S.C.P.'S
    63  C                 RVCxLO(1800)      X
    64  C                 IVCxSZ            SIGN FOR Z
    65  C                 RVCxFREQ          WAVE FREQ. SQUARED
    66  C
    67  C         TO CALCULATE THE VALUES OF THE IONOSPHERIC PARAMETERS
    68  C                  USED TO UPDATE THE RAYPATH.
    69  C
    70  C         RETURNS:
    71  C                 RVCxFB(3)         FN^2 VALUES OF THE LAYERS
    72  C                 RVCxHB(3)         H & Y VALUES OF THE LAYERS
    73  C                 RVCxALPH          X
    74  C                 RVCxBETA          X
    75  C                 RVCxET1           X-COEF. VALUES FOR MU^2
    76  C                 RVCxET2           X
    77  C                 RVCxB5&6          ]
    78  C                 RVCxE5&6          ]
    79  C                 RVCxV1            ]-SOME INTERMEDIATE VALUES
    80  C                 RVCxV2            ]
    81  C                 IVSxSCS           VBL. SIGNIFYING USE OF SINGLE
    82  C                                        S.C.P.
    83  C                 IVSxSCT1-3        VBLS. FOR THE TRIANGLE OF S.C.P.'S
    84  C                                        USED
    85  C
    86  C-----------------------------------------------------------------
    87  C
    88        INTEGER IVSxSCS, IVSxSCT1, IVSxSCT2, IVSxSCT3
    89        INTEGER IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3, IFCxN4
    90        INTEGER IVCxSX, IVCxSY, IVSxJ, IFCxSN
    91        INTEGER IFSxN4, IFSxG, IVCxSZ, ICCxN
    92        INTEGER IFLxOUT, IVLxIN, IVLxFIN, IVCxCASE
    93  C
    94        REAL*8 RVxGRID(6), RPCxRE, RPCxPI, RPCxDR
    95        REAL*8 RTLxD1, RTLxD2, RTLxD5, RTLx1
    96        REAL*8 RVLxTR1, RVLxTR2, RTLxTH
    97        REAL*8 RTLxH, RTLxI3, RTLxI4, RTLxI5
    98        REAL*8 RVCxA5(3), RVCxA6(3), RVCxLAI, RVCxLOI
    99        REAL*8 RVCxFN(1800,3),RVCxH(1800,3),RVCxLA(1800),RVCxLO(1800)
   100        REAL*8 RVCxALPH, RVCxBETA, RVCxHBND, RVCxF40, RVCxF65
```

```
Line#  Source Line

  101           REAL*8 RVCxET1, RVCxET2, RVCxFB(3), RVCxHB(3), RVCxB5(3)
  102           REAL*8 RVCxB6(3), RVCxE5(3), RVCxE6(3)
  103           REAL*8 RVCxV1, RVCxV2, RVCxH5, RVCxFREQ
  104           REAL*8 RVCxL1, RVCxLO1, RVCxHBOT, RVLxHTV, RVLxSLV
  105           REAL*8 RVCxXI, RVCxYI, RVCxZI, RVCxXF, RVCxYF, RVCxZF
  106           REAL*8 RVCxK1, RVCxHL, RVCxXX, RVCxXL, RVCxXU, RVCxA0
  107           REAL*8 RVCxHB1, RVCxHU, RVCxB0, RVCxH2, RVCxYS, RVCxSL1
  108           REAL*8 RVCxSL2, RVCxH1L, RVCxA1, RVCxB1, RVCxC1, RVCxH1P
  109           REAL*8 RVCxA2, RVCxB2, RVCxC2, RVCxHT3, RVCxHT4, RVCxHT5
  110           REAL*8 RVLxLATS, RVLxLONS, RVLxLATE, RVLxLONE, RVLxLOI
  111           REAL*8 RVLxF98, RVLxK2, RVCxH2P, RTCxA, RTCxB, RTCxC
  112   C
  113           COMMON /MAINDAT/ RVxGRID
  114           COMMON /PRAM/ RPCxPI, RPCxDR, RPCxRE, RTCxA
  115           COMMON /SCPS1/ ICCxN,RVCxLA,RVCxLO
  116           COMMON /SCPS1A/ RVCxFN,RVCxH
  117           COMMON /OTHER/ RVCxL1,RVCxLO1,RVCxHBOT,RVCxFREQ,RTCxB,RTCxC
  118           COMMON /START/ RVCxXI, RVCxYI, RVCxZI, RVCxLAI, RVCxLOI
  119           COMMON /END/ RVCxXF, RVCxYF, RVCxZF, RVCxH5
  120           COMMON /IONO1/ RVCxALPH,RVCxBETA,RVCxET1,RVCxET2,RVCxFB,RVCxHB
  121           COMMON /IONO2/ RVCxA5,RVCxA6,RVCxB5,RVCxB6,RVCxE5,RVCxE6
  122           COMMON /IONO3/ RVCxV1, RVCxV2, RVCxXX, IVCxCASE, RVCxHBND
  123           COMMON /MORE/ IVCxSX, IVCxSY, IVCxSZ
  124           COMMON /TEMP1/ RVCxF40, RVCxF65, RVCxK1, RVCxHL
  125           COMMON /TEMP2/ IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3, IFCxN4
  126           COMMON /TEMP3/ RTLxI3, RTLxI4, RTLxI5, RTLxD5
  127           COMMON /VAR1/ RVCxXL, RVCxXU, RVCxHU, RVCxA0, RVCxB0
  128           COMMON /VAR2/ RVCxHB1,RVCxH2,RVCxYS,RVCxSL1,RVCxSL2,RVCxH1L
  129           COMMON /VAR3/ RVCxA1, RVCxB1, RVCxC1, RVCxH1P, RVCxH2P
  130           COMMON /VAR4/ RVCxA2,RVCxB2,RVCxC2,RVCxHT3,RVCxHT4,RVCxHT5
  131           COMMON /RAID/ IFCxSN
  132   C
  133   C-----------------------------------------------------------------------
  134   C
  135   10000   RTLxH = RVCxH5
  136           RVCxHBND = 0.0D00
  137           RVCxLAI = RVCxLAI / RPCxDR
  138           RVCxLOI = RVCxLOI / RPCxDR
  139           RVLxLOI = RVCxLOI
  140           IF (ICCxN.GT.1) THEN
  141   C
  142   C ---------------------------------------------------------------------
  143   C          If the number of points is not 1, then there is a grid.  It
  144   C            calculate the grid boundaries, then call INBOX to determine
  145   C            whether the present location is w/in the grid.  If not in
  146   C            the grid, then do a spherically symmetric ionosphere based
  147   C            upon the nearest grid point's parameters.  Within the grid,
  148   C            call TRIANG to determine the three points that are to be
  149   C            used in the interpolation.
  150   C ---------------------------------------------------------------------
```

```
Line#  Source Line

  151  C
  152                RVLxLATS = RVxGRID(3)
  153                RVLxLONS = RVxGRID(4)
  154                RVLxLATE = RVLxLATS + (RVxGRID(5)-1.0D00)*RVxGRID(1)
  155                RVLxLONE = RVLxLONS + (RVxGRID(6)-1.0D00)*RVxGRID(2)
  156  C
  157  C          If the grid spans the date line, it's longitude values
  158  C            will not be negative.  While this is good for purposes
  159  C            of the interpolation, ray points with west longitudes
  160  C            will spuriously be considered to be outside the grid.
  161  C            Hence, the need for the temporary copy of the longitude.
  162  C            If necessary, the next statement will bring the longitude
  163  C            of the ray point into the same longitude system as the
  164  C            grid.
  165  C
  166                IF (RVLxLOI.LT.RVLxLONS) RVLxLOI = RVLxLOI + 360.0
  167  C
  168                IFLxOUT = -1
  169                CALL INBOX(RVCxLAI,RVLxLOI,RVLxLATS,RVLxLONS,RVLxLATE
  170       #,RVLxLONE,IFLxOUT)
  171  C
  172  C          IFLxOUT = 1 means that the current location is outside the
  173  C            ionospheric specification grid, so only the nearest grid
  174  C            point is used.
  175  C
  176                IF (IFLxOUT.EQ.1) THEN
  177                    IFCxN4 = 1
  178                    IF (RVCxLAI.LT.RVLxLATS) THEN
  179                        ITLxA = 1
  180                    ELSE IF (RVCxLAI.GT.RVLxLATE) THEN
  181                        ITLxA = IDNINT(RVxGRID(5))
  182                    ELSE
  183                        ITLxA =IDNINT(((RVCxLAI-RVLxLATS)/RVxGRID(1))+1.0)
  184                    ENDIF
  185                    IF (RVLxLOI.LT.RVLxLONS) THEN
  186                        ITLxB = 1
  187                    ELSE IF (RVLxLOI.GT.RVLxLONE) THEN
  188                        ITLxB = IDNINT(RVxGRID(6))
  189                    ELSE
  190                        ITLxB =IDNINT(((RVLxLOI-RVLxLONS)/RVxGRID(2))+1.0)
  191                    ENDIF
  192                    IVCxSCS = IDNINT((ITLxA - 1.0D00)*RVxGRID(6) + ITLxB
  193                ELSE
  194                    CALL TRIANG
  195                    IFCxN4 = 2
  196                ENDIF
  197            ELSE
  198  C
  199  C          These are the values to use if only one specification point
  200  C            is given.
```

```
Line#  Source Line

  201  C
  202              IVCxSCS = 1
  203              IFCxN4 = 1
  204              IFLxOUT = 1
  205           ENDIF
  206  C
  207  C         Some initializations.
  208  C
  209           IVSxSCS = IVCxSCS
  210           IVSxSCT1 = IVCxSCT1
  211           IVSxSCT2 = IVCxSCT2
  212           IVSxSCT3 = IVCxSCT3
  213           IFSxN4 = IFCxN4
  214           RVCxET1 = 0.0D00
  215           RVCxET2 = 0.0D00
  216           RVCxALPH = 0.0D00
  217           RVCxBETA = 0.0D00
  218           RVCxXX = 0.0D00
  219           RVCxLOI = RVCxLOI * RPCxDR
  220           RVLxLOI = RVLxLOI * RPCxDR
  221           RVCxLAI = RVCxLAI * RPCxDR
  222  C
  223  C        For calculation purposes here, move slightly off of the
  224  C          boundary, if we're at one.
  225  C
  226           IF (IFSxG.EQ.2.OR.IFSxG.GE.6) RTLxH =
  227       #RTLxH + IVCxSZ*1.0D-02
  228  C
  229  C ---------------------------------------------------------------
  230  C         Now the enumeration of possible height regions begins.
  231  C           First, handle what is needed for the free-space
  232  C           underlying the ionosphere (below 40 km).  Also, the lower
  233  C           part of the D layer is exponential and spherically sym-
  234  C           metric, so do that now too, since it involves no
  235  C           interpolation.  For full details see the RADAR-C report
  236  C           cited in the documentation.
  237  C ---------------------------------------------------------------
  238  C
  239           IF (RTLxH.LE.40.0D00) THEN
  240               IF (IVCxSZ.GT.0) THEN
  241  C
  242  C ---------------------------------------------------------------
  243  C         HBND is the (approximate in some cases) height of the
  244  C            next boundary to be encountered.
  245  C         IVSxJ is sort of an integer counterpart to HBND.
  246  C ---------------------------------------------------------------
  247  C
  248                   RVCxHBND = 40.0D00
  249                   IVSxJ = 1
  250               ELSE
```

```
Line#  Source Line

  251                   RVCxHBND = 0.0D00
  252                   IVSxJ = 0
  253               ENDIF
  254               RETURN
  255           ENDIF
  256   C
  257   C       The lower D region of RADAR-C
  258   C
  259           IF (RTLxH.LE.65.0D00) THEN
  260               RVCxXX = RVCxF40 * DEXP(RVCxK1 * (RTLxH - 40.0D00))
  261               RVCxALPH = RVCxK1 * RVCxXX / RVCxFREQ
  262               RVCxBETA = -RVCxK1 * RVCxALPH / 2.0D00
  263               IF (IVCxSZ.GT.0) THEN
  264                   RVCxHBND = 65.0D00
  265                   IVSxJ = 2
  266               ELSE
  267                   RVCxHBND = 40.0D00
  268                   IVSxJ = 1
  269               ENDIF
  270               RETURN
  271           ENDIF
  272   C
  273   C -------------------------------------------------------------
  274   C        Here is where, if the case is a non-spherically symmetric
  275   C          one, that interpolation begins to be needed.
  276   C
  277   C        Note:   FB(1)  -->  foE ** 2
  278   C                FB(2)  -->  foF1 ** 2
  279   C                FB(3)  -->  foF2 ** 2
  280   C
  281   C                HB(1)  -->  hmF1
  282   C                HB(2)  -->  hmF2
  283   C                HB(3)  -->  YmF2
  284   C
  285   C        The interpolation is done as follows.
  286   C           X = A + B*Lat + E*Lon, where A, B, and E are determined
  287   C           by setting up the three equations at the three grid
  288   C           points surrounding the point in question.  This is done
  289   C           in INTERP.
  290   C -------------------------------------------------------------
  291   C
  292   C       Recall that IFSxN4 = 1 for the spherically symmetric case.
  293   C
  294   C -------------------------------------------------------------
  295   C
  296           IF (IFSxN4.EQ.1) THEN
  297               RVCxFB(1) = RVCxFN(1VSxSCS,1)
  298           ELSE
  299               RVCxV1 = RPCxRE + RTLxH
  300               RVCxV2 = RVCxV1 * DCOS(RVCxLAI)
```

```
Line#  Source Line

   301              RTLxI3 = RVCxLA(IVSxSCT1)*RVCxLO(IVSxSCT2)-RVCxLA(IVSxSCT
   302         #*RVCxLO(IVSxSCT1)
   303              RTLxI4 = RVCxLA(IVSxSCT2)*RVCxLO(IVSxSCT3)-RVCxLA(IVSxSCT
   304         #*RVCxLO(IVSxSCT2)
   305              RTLxI5 = RVCxLA(IVSxSCT3)*RVCxLO(IVSxSCT1)-RVCxLA(IVSxSCT
   306         #*RVCxLO(IVSxSCT3)
   307              RTLxD5 = RTLxI3 + RTLxI4 + RTLxI5
   308   C
   309   C         IVLxIN and FIN are used to get INTERP to operate only on what
   310   C            is needed to do the calculations for the D & E regions.
   311   C
   312              IVLxIN = 1
   313              IVLxFIN = 1
   314              CALL INTERP(IVLxIN, IVLxFIN)
   315              RVCxFB(1) = RVCxA5(1)+RVCxB5(1)*RVCxLAI+RVCxE5(1)*RVLxLOI
   316            ENDIF
   317            IF (RTLxH.LE.98.0D00) THEN
   318   C
   319   C         The upper D layer is exponential.
   320   C
   321              RVLxF98 = 0.64D00 * RVCxFB(1)
   322              RVLxK2 = (DLOG(RVLxF98)-DLOG(RVCxF65))/33.0D00
   323              RVCxXX = RVCxF65 * DEXP(RVLxK2 * (RTLxH - 65.0D00))
   324              RVCxALPH = RVLxK2 * RVCxXX / RVCxFREQ
   325              RVCxBETA = -RVLxK2 * RVCxALPH / 2.0D00
   326              IF (IVCxSZ.GT.0) THEN
   327                  RVCxHBND = 98.0D00
   328                  IVSxJ = 3
   329              ELSE
   330                  RVCxHBND = 65.0D00
   331                  IVSxJ = 2
   332              ENDIF
   333              IF (IFSxN4.NE.1) THEN
   334                  RTLx1 = ((RTLxH - 65.0D00) * RVCxXX)/
   335         #(33.0D00 * RVCxFB(1) * RVCxFREQ)
   336                  RVCxET1 = -RTLx1 * RVCxB5(1) / RVCxV1
   337                  RVCxET2 = RTLx1 * RVCxE5(1) / RVCxV2
   338              ENDIF
   339              RETURN
   340            ENDIF
   341            IF (RTLxH.LE.RVCxHL) THEN
   342   C
   343   C         The E layer is parabolic, with only foE as a non-constant
   344   C            parameter.
   345   C
   346              RVLxHTV = (RTLxH - 110.0D00) / 20.0D00
   347              RVLxSLV = RVCxFB(1) /20.0D00
   348              RVCxXX = RVCxFB(1) * (1.0D00 - RVLxHTV*RVLxHTV)
   349              RVCxALPH = -2.0D00 * RVLxHTV * RVLxSLV / RVCxFREQ
   350              RVCxBETA = RVLxSLV / (20.0D00 * RVCxFREQ)
```

```
Line#  Source Line

  351               IF (IVCxSZ.GT.0) THEN
  352                   RVCxHBND = RVCxHL
  353                   IVSxJ = 4
  354               ELSE
  355                   RVCxHBND = 98.0D00
  356                   IVSxJ = 3
  357               ENDIF
  358               IF (IFSxN4.NE.1) THEN
  359                   RTLx1 = (1.0D00 - RVLxHTV*RVLxHTV) / RVCxFREQ
  360                   RVCxET1 = -RTLx1 * RVCxB5(1) / RVCxV1
  361                   RVCxET2 = RTLx1 * RVCxE5(1) / RVCxV2
  362               ENDIF
  363               RETURN
  364           ENDIF
  365  C
  366  C         Now, get the rest of the values needed to do the
  367  C           calculations in the F1 & F2 layers.
  368  C
  369           IF (IFSxN4.EQ.1) THEN
  370               RVCxFB(2) = RVCxFN(IVSxSCS,2)
  371               RVCxFB(3) = RVCxFN(IVSxSCS,3)
  372               RVCxHB(1) = RVCxH(IVSxSCS,1)
  373               RVCxHB(2) = RVCxH(IVSxSCS,2)
  374               RVCxHB(3) = RVCxH(IVSxSCS,3)
  375           ELSE
  376               IVLxIN = 2
  377               IVLxFIN = 3
  378               CALL INTERP(IVLxIN, IVLxFIN)
  379  C
  380  C ---------------------------------------------------------------
  381  C          The F1 values may be 0 in this model if the F1 isn't
  382  C            seen in the profile. This botches up the interpolation
  383  C            by introducing an artificially large gradient.  This
  384  C            gets fixed by the next snippet of code.
  385  C ---------------------------------------------------------------
  386  C
  387               IF (RVCxH(IVSxSCT1,1).EQ.0.0.OR.RVCxH(IVSxSCT2,1).EQ.
  388       #0.0.OR.RVCxH(IVSxSCT3,1).EQ.0.0) THEN
  389                   RVCxA5(2) = 0.0
  390                   RVCxB5(2) = 0.0
  391                   RVCxE5(2) = 0.0
  392                   RVCxA6(1) = 0.0
  393                   RVCxB6(1) = 0.0
  394                   RVCxE6(1) = 0.0
  395               ENDIF
  396  C
  397               RVCxFB(2) = RVCxA5(2) + RVCxB5(2)*RVCxLAI
  398       #+ RVCxE5(2)*RVLxLOI
  399               RVCxFB(3) = RVCxA5(3) + RVCxB5(3)*RVCxLAI
  400       #+ RVCxE5(3)*RVLxLOI
```

```
Line#  Source Line

  401                RVCxHB(1) = RVCxA6(1) + RVCxB6(1)*RVCxLAI
  402         #+ RVCxE6(1)*RVLxLOI
  403                RVCxHB(2) = RVCxA6(2) + RVCxB6(2)*RVCxLAI
  404         #+ RVCxE6(2)*RVLxLOI
  405                RVCxHB(3) = RVCxA6(3) + RVCxB6(3)*RVCxLAI
  406         #+ RVCxE6(3)*RVLxLOI
  407            ENDIF
  408    C
  409    C ---------------------------------------------------------------
  410    C         From here on, the logic is complicated, but it follows the
  411    C           procedures in the RADAR-C report.  Basically, it must be
  412    C           determined whether the F1 is linear or parabolic, and
  413    C           where the E, valley, F1, F2, and topside profiles fall
  414    C           and where their intersection points are.
  415    C ---------------------------------------------------------------
  416    C
  417            RVCxXL = 0.8516D00 * 0.8516D00 * RVCxFB(1)
  418            RVCxXU = 0.98D00 * 0.98D00 * RVCxFB(1)
  419            RVCxHU = RVCxHB(2) - RVCxHB(3) * DSQRT(1.0D00 -
  420         #(RVCxXU/RVCxFB(3)))
  421            RVCxHB1 = 0.75D00 * RVCxHB(1)
  422            RVCxA0 = (RVCxHU*RVCxXL - RVCxHL*RVCxXU)/(RVCxHU - RVCxHL)
  423            RVCxB0 = (RVCxXU - RVCxXL) / (RVCxHU - RVCxHL)
  424            RVCxHT5 = RVCxHB(2) + 0.25D00*RVCxHB(3)
  425            IF (RVCxFB(2).LE.RVCxXU) THEN
  426                IVCxCASE = 1
  427                GO TO 20000
  428            ENDIF
  429            IF (RVCxFB(2).GT.RVCxFB(3)) GO TO 10750
  430            RVCxH2 = RVCxHB(2) - RVCxHB(3) * DSQRT(1.0D00 -
  431         #(RVCxFB(2)/RVCxFB(3)))
  432            IF (RVCxHB(1).LE.RVCxH2) GO TO 10750
  433            RVCxYS = MAX(1.0D00, RVCxH2 - RVCxHB1)
  434            RVCxSL1 = RVCxFB(2) / RVCxYS
  435            RVCxSL2 = 2.0D00 * RVCxFB(3) * (RVCxHB(2) - RVCxH2)/
  436         #(RVCxHB(3)*RVCxHB(3))
  437            IF (RVCxSL1.GE.RVCxSL2) THEN
  438                RVCxH1L = (RVCxHB1*RVCxSL1 + RVCxA0)/
  439         #(RVCxSL1 - RVCxB0)
  440                IF (RVCxH1L.LT.RVCxHU) THEN
  441                    IVCxCASE = 3
  442                ELSE
  443                    IVCxCASE = 1
  444                ENDIF
  445                GO TO 20000
  446            ENDIF
  447   10750    RVCxA1 = 16.0D00 * RVCxFB(2) / (RVCxHB(1)*RVCxHB(1))
  448            RVCxB1 = (32.0D00 * RVCxFB(2) / RVCxHB(1)) - RVCxB0
  449            RVCxC1 = 15.0D00 * RVCxFB(2) + RVCxA0
  450            RTLxD1 = RVCxB1*RVCxB1 - 4.0D00 * RVCxA1 * RVCxC1
```

```
Line#  Source Line

  451          IF (RTLxD1.LT.0.0D00) THEN
  452              IVCxCASE = 1
  453              GO TO 20000
  454          ENDIF
  455          RVCxH1P = (RVCxB1 - INTSIGN(RVCxB1) * DSQRT(RTLxD1)
  456       #/(2.0D00 * RVCxA1)
  457          RVCxA2 = (RVCxFB(3)/(RVCxHB(3)*RVCxHB(3))) -
  458       #(16.0D00 * RVCxFB(2)/(RVCxHB(1)*RVCxHB(1)))
  459          RVCxB2 = (32.0D00 * RVCxFB(2)/RVCxHB(1)) -
  460       #(2.0D00 * RVCxFB(3) * RVCxHB(2)/(RVCxHB(3)*RVCxHB(3)))
  461          RVCxC2 = RVCxFB(3)*(((RVCxHB(2)*RVCxHB(2))/(RVCxHB(3)*
  462       #RVCxHB(3))) - 1.0D00) - 15.0D00 * RVCxFB(2)
  463          RTLxD2 = RVCxB2 * RVCxB2 - 4.0D00 * RVCxA2 * RVCxC2
  464          IF (RTLxD2.LT.0.0D00) THEN
  465              IVCxCASE = 1
  466              GO TO 20000
  467          ENDIF
  468          RVCxHT3 = (-RVCxB2 + INTSIGN(RVCxB2) * DSQRT(RTLxD2))/
  469       #(2.0D00 * RVCxA2)
  470          RVCxHT4 = (-RVCxB2 - INTSIGN(RVCxB2)*DSQRT(RTLxD2))/
  471       #(2.0D00 * RVCxA2)
  472          IF (RVCxH1P.LE.RVCxHU) THEN
  473              RVCxH2P = (RVCxB1 + INTSIGN(RVCxB1)*DSQRT(RTLxD1))/
  474       #(2.0D00 * RVCxA1)
  475              IF (RVCxH2P.LT.RVCxHU) THEN
  476                  IVCxCASE = 5
  477                  GO TO 20000
  478              ENDIF
  479              IVCxCASE = 2
  480              IF (RVCxB2.GT.0.0D00) THEN
  481                  RVLxTR1 = RVCxHT4
  482                  RVLxTR2 = RVCxHT3
  483                  IFCxSN = 1
  484              ELSE
  485                  RVLxTR1 = RVCxHT3
  486                  RVLxTR2 = RVCxHT4
  487                  IFCxSN = -1
  488              ENDIF
  489              RTLxTH = RVLxTR1
  490  10850       IF (RTLxTH.GT.RVCxHU.AND.RTLxTH.LT.RVCxHT5) THEN
  491                  RVCxHT4 = RTLxTH
  492                  GO TO 20000
  493              ENDIF
  494              IF (RTLxTH.EQ.RVLxTR2) THEN
  495  C
  496  C         Some diagnostics that are of the nature of, "The program
  497  C           can't reach this point, but..."
  498  C
  499                  PRINT *, ' EXCEEDINGLY WIERD CASE: POINT #1'
  500                  PRINT *, ' HT3:',RVCxHT3,'   HT4:',RVCxHT4
```

```
Line#  Source Line

  501                   PRINT *, ' H5 :',RVCxH5,'    B2 :',RVCxB2
  502                   PRINT *, ' LAT:',RVCxLAI/RPCxDR,'    LON:',
  503        #RVCxLOI/RPCxDR
  504                   PRINT *, ' HT5:',RVCxHT5,'    HU :',RVCxHU
  505                   PRINT *, ' H1P:',RVCxH1P
  506                   STOP
  507                ELSE
  508                   RTLxTH = RVLxTR2
  509                   IFCxSN = -IFCxSN
  510                   GO TO 10850
  511                ENDIF
  512             ENDIF
  513             IF (RVCxHT3.LE.RVCxH1P) THEN
  514                IVCxCASE = 1
  515                GO TO 20000
  516             ENDIF
  517             IF (RVCxHT3.LT.RVCxHT4.AND.RVCxHT4.LT.RVCxHT5) THEN
  518                IVCxCASE = 4
  519             ELSE
  520     C
  521     C         Another diagnostic as outlined above.
  522     C
  523                PRINT *, ' EXCEEDINGLY WIERD CASE: POINT #2'
  524                STOP
  525             ENDIF
  526     C
  527     C         These routines continue the calculations, now that
  528     C            it has been decided which path to take.
  529     C
  530     20000   IF (IVCxCASE.EQ.1) THEN
  531                CALL CASE1(RTLxH, IVSxJ)
  532                RETURN
  533             ELSE IF (IVCxCASE.EQ.2) THEN
  534                CALL CASE2(RTLxH, IVSxJ)
  535                RETURN
  536             ELSE IF (IVCxCASE.EQ.3) THEN
  537                CALL CASE3(RTLxH, IVSxJ)
  538                RETURN
  539             ELSE IF (IVCxCASE.EQ.4) THEN
  540                CALL CASE4(RTLxH, IVSxJ)
  541             ELSE
  542                CALL CASE5(RTLxH, IVSxJ)
  543             ENDIF
  544             RETURN
  545             END
```

IONOPAR Local Symbols

Name	Class	Type	Size
IVSXSCT3	param		
IVSXSCT2	param		
IVSXSCT1	param		
IVSXSCS	param		
IFSXG	param		
IFSXN4	param		
IVSXJ	param		
RVLXLATS	local	REAL*8	8
RTLXD2	local	REAL*8	8
RVLXSLV	local	REAL*8	8
ITLXA	local	INTEGER*4	4
ITLXB	local	INTEGER*4	4
RVLXF98	local	REAL*8	8
RVLXLONS	local	REAL*8	8
RVLXK2	local	REAL*8	8
RTLXH	local	REAL*8	8
IVLXIN	local	INTEGER*4	4
IVLXFIN	local	INTEGER*4	4
RVLXTR1	local	REAL*8	8
RVLXTR2	local	REAL*8	8
RTLXTH	local	REAL*8	8
IFLXOUT	local	INTEGER*4	4
RVLXLOI	local	REAL*8	8
RVLXLATE	local	REAL*8	8
RVLXLONE	local	REAL*8	8
RTLX1	local	REAL*8	8
RVLXHTV	local	REAL*8	8
RTLXD1	local	REAL*8	8
RVCXET2	IONO1	REAL*8	8
RVCXFB	IONO1	REAL*8	24
RVCXHB	IONO1	REAL*8	24
RVCXB5	IONO2	REAL*8	24
RVCXB6	IONO2	REAL*8	24
RVCXE5	IONO2	REAL*8	24
RVCXE6	IONO2	REAL*8	24
RVCXV1	IONO3	REAL*8	8
RVCXV2	IONO3	REAL*8	8
RVCXH5	END	REAL*8	8
RVCXFREQ	OTHER	REAL*8	8
RVCXL1	OTHER	REAL*8	8
RVCXLO1	OTHER	REAL*8	8
RVCXHBOT	OTHER	REAL*8	8
RVCXXI	START	REAL*8	8
RVCXYI	START	REAL*8	8
RVCXZI	START	REAL*8	8
RVCXXF	END	REAL*8	8

IONOPAR Local Symbols

Name	Class	Type	Size
RVCXYF	END	REAL*8	8
IVCXSCS	TEMP2	INTEGER*4	4
RVCXZF	END	REAL*8	8
RVCXR1	TEMP1	REAL*8	8
IVCXSCT1	TEMP2	INTEGER*4	4
RVCXHL	TEMP1	REAL*8	8
IVCXSCT2	TEMP2	INTEGER*4	4
RVCXXX	IONO3	REAL*8	8
IVCXSCT3	TEMP2	INTEGER*4	4
IFCXN4	TEMP2	INTEGER*4	4
RVCXXL	VAR1	REAL*8	8
RVCXXU	VAR1	REAL*8	8
IVCXSX	MORE	INTEGER*4	4
RVCXA0	VAR1	REAL*8	8
IVCXSY	MORE	INTEGER*4	4
RVCXHB1	VAR2	REAL*8	8
IFCXSN	RAID	INTEGER*4	4
IVCXSZ	MORE	INTEGER*4	4
RVCXHU	VAR1	REAL*8	8
ICCXN	SCPS1	INTEGER*4	4
RVCXB0	VAR1	REAL*8	8
RVCXH2	VAR2	REAL*8	8
RVCXYS	VAR2	REAL*8	8
RVCXSL1	VAR2	REAL*8	8
IVCXCASE	IONO3	INTEGER*4	4
RVCXSL2	VAR2	REAL*8	8
RVXGRID	MAINDAT	REAL*8	48
RVCXH1L	VAR2	REAL*8	8
RVCXA1	VAR3	REAL*8	8
RPCXRE	PRAM	REAL*8	8
RVCXB1	VAR3	REAL*8	8
RPCXPI	PRAM	REAL*8	8
RVCXC1	VAR3	REAL*8	8
RPCXDR	PRAM	REAL*8	8
RVCXH1P	VAR3	REAL*8	8
RVCXA2	VAR4	REAL*8	8
RVCXB2	VAR4	REAL*8	8
RTLXD5	TEMP3	REAL*8	8
RVCXC2	VAR4	REAL*8	8
RVCXHT3	VAR4	REAL*8	8
RVCXHT4	VAR4	REAL*8	8
RVCXHT5	VAR4	REAL*8	8
RTLXI3	TEMP3	REAL*8	8
RTLXI4	TEMP3	REAL*8	8
RTLXI5	TEMP3	REAL*8	8

IONOPAR Local Symbols

Name	Class	Type	Size
RVCXA5	IONO2	REAL*8	24
RVCXA6	IONO2	REAL*8	24
RVCXLAI	START	REAL*8	8
RVCXLOI	START	REAL*8	8
RVCXH2P	VAR3	REAL*8	8
RVCXFN	SCPS1A	REAL*8	43200
RTCXA	PRAM	REAL*8	8
RVCXH	SCPS1A	REAL*8	43200
RTCXB	OTHER	REAL*8	8
RVCXLA	SCPS1	REAL*8	14400
RTCXC	OTHER	REAL*8	8
RVCXLO	SCPS1	REAL*8	14400
RVCXALPH	IONO1	REAL*8	8
RVCXBETA	IONO1	REAL*8	8
RVCXHBND	IONO3	REAL*8	8
RVCXF40	TEMP1	REAL*8	8
RVCXF65	TEMP1	REAL*8	8
RVCXET1	IONO1	REAL*8	8

```
Line#  Source Line

  547           SUBROUTINE INTERP(IVSxIN, IVSxFIN)
  548   C
  549   C-------------------------------------------------------------
  550   C
  551   C    INTERP -- SUBROUTINE TO PERFORM THE LATITUDE AND LONGITUDE
  552   C         INTERPOLATION REQUIRED BY IONOPAR
  553   C
  554   C         CALLED BY: IONOPAR
  555   C
  556   C-------------------------------------------------------------
  557   C
  558   C         AUTHOR:          MICHAEL H. REILLY & ERIC L. STROBEL
  559   C
  560   C         DATE:            09/01/87
  561   C
  562   C         VERSION:         1.0
  563   C
  564   C-------------------------------------------------------------
  565   C
  566   C         REVISED:         09/01/87 -- V1.0.  Initial revision.
  567   C
  568   C-------------------------------------------------------------
  569   C
  570   C         USES:    IVSxIN & IVSxFIN        TO DETERMINE WHICH SETS
  571   C                           OF INTERPOLATION COEFFICIENTS TO DO.
  572   C
  573   C          The interpolation is done as follows,
  574   C             X = A + B*Lat + E*Lon, where A, B, and E are determined
  575   C             by setting up the three equations at the three grid
  576   C             points surrounding the point in question.  These
  577   C             equations are then solved by use of Kramer's rule.
  578   C
  579   C ------------------------------------------------------------
  580   C
  581   C         RETURNS:        /IONO2/      THE COMMON BLOCK CONTAINING THE
  582   C                          INTERPOLATION COEFFICIENTS (RVCx(A,B,E)(5,6)
  583   C                          WHERE A, B, AND E ARE AS ABOVE, AND THE 5
  584   C                          REPRESENTS A FREQUENCY COEFFICIENT AND THE 6
  585   C                          REPRESENTS A HEIGHT COEFFICIENT.
  586   C
  587   C ------------------------------------------------------------
  588   C
  589           INTEGER IVSxIN, IVSxFIN, IVCxSCT1, IVCxSCT2
  590           INTEGER IVCxSCT3, ICCxN, IVCxSCS, ITLx1
  591   C
  592           REAL*8 RVCxI3, RVCxI4, RVCxI5, RVCxD5
  593           REAL*8 RVCxFN(1800,3), RVCxH(1800,3), RVCxLA(1800)
  594           REAL*8 RVCxLO(1800), RVCxA5(3), RVCxA6(3), RVCxB5(3)
  595           REAL*8 RVCxB6(3), RVCxE5(3), RVCxE6(3)
  596   C
```

```
Line#  Source Line

   597           COMMON /SCPS1/ ICCxN,RVCxLA,RVCxLO
   598           COMMON /SCPS1A/ RVCxFN,RVCxH
   599           COMMON /IONO2/ RVCxA5,RVCxA6,RVCxB5,RVCxB6,RVCxE5,RVCxE6
   600           COMMON /TEMP2/ IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3, ITLx1
   601           COMMON /TEMP3/ RVCxI3, RVCxI4, RVCxI5, RVCxD5
   602  C
   603           DO 10000 I = IVSxIN, IVSxFIN
   604               RVCxA5(I) = (RVCxFN(IVCxSCT1,I)*RVCxI4 + RVCxFN(IVCxSCT2,I)
   605       #*RVCxI5 + RVCxFN(IVCxSCT3,I)*RVCxI3)/RVCxD5
   606               RVCxA6(I) = (RVCxH(IVCxSCT1,I)*RVCxI4 + RVCxH(IVCxSCT2,I)
   607       #*RVCxI5 + RVCxH(IVCxSCT3,I)*RVCxI3)/RVCxD5
   608               RVCxB5(I) = RVCxFN(IVCxSCT1,I)*(RVCxLO(IVCxSCT2)-RVCxLO(
   609       #IVCxSCT3)) + RVCxFN(IVCxSCT2,I)*(RVCxLO(IVCxSCT3)-
   610       #RVCxLO(IVCxSCT1))
   611               RVCxB5(I) = (RVCxB5(I) + RVCxFN(IVCxSCT3,I)*(RVCxLO(
   612       #IVCxSCT1)-RVCxLO(IVCxSCT2)))/RVCxD5
   613               RVCxB6(I) = RVCxH(IVCxSCT1,I)*(RVCxLO(IVCxSCT2)-RVCxLO(
   614       #IVCxSCT3)) + RVCxH(IVCxSCT2,I)*(RVCxLO(IVCxSCT3)-
   615       #RVCxLO(IVCxSCT1))
   616               RVCxB6(I) = (RVCxB6(I) + RVCxH(IVCxSCT3,I)*(RVCxLO(
   617       #IVCxSCT1)-RVCxLO(IVCxSCT2)))/RVCxD5
   618               RVCxE5(I) = RVCxFN(IVCxSCT1,I)*(RVCxLA(IVCxSCT3)-RVCxLA(
   619       #IVCxSCT2)) + RVCxFN(IVCxSCT2,I)*(RVCxLA(IVCxSCT1)-
   620       #RVCxLA(IVCxSCT3))
   621               RVCxE5(I) = (RVCxE5(I) + RVCxFN(IVCxSCT3,I)*(RVCxLA(
   622       #IVCxSCT2)-RVCxLA(IVCxSCT1)))/RVCxD5
   623               RVCxE6(I) = RVCxH(IVCxSCT1,I)*(RVCxLA(IVCxSCT3)-RVCxLA
   624       #IVCxSCT2)) + RVCxH(IVCxSCT2,I)*(RVCxLA(IVCxSCT1)-
   625       #RVCxLA(IVCxSCT3))
   626               RVCxE6(I) = (RVCxE6(I) + RVCxH(IVCxSCT3,I)*(RVCxLA(
   627       #IVCxSCT2)-RVCxLA(IVCxSCT1)))/RVCxD5
   628  10000        CONTINUE
   629           RETURN
   630           END


INTERP  Local Symbols

Name                          Class    Type              Size

IVSXFIN . . . . . param
IVSXIN. . . . . param
I . . . . . local    INTEGER*4            4
IVCXSCT1. . . . . TEMP2    INTEGER*4            4
IVCXSCT2. . . . . TEMP2    INTEGER*4            4
IVCXSCT3. . . . . TEMP2    INTEGER*4            4
ICCXN . . . . . SCPS1    INTEGER*4            4
IVCXSCS . . . . . TEMP2    INTEGER*4            4
ITLX1 . . . . . TEMP2    INTEGER*4            4
RVCXI3. . . . . TEMP3    REAL*8               8
```

INTERP Local Symbols

Name	Class	Type	Size
RVCXI4.	TEMP3	REAL*8	8
RVCXI5.	TEMP3	REAL*8	8
RVCXD5.	TEMP3	REAL*8	8
RVCXFN.	SCPS1A	REAL*8	43200
RVCXH	SCPS1A	REAL*8	43200
RVCXLA.	SCPS1	REAL*8	14400
RVCXLO.	SCPS1	REAL*8	14400
RVCXA5.	IONO2	REAL*8	24
RVCXA6.	IONO2	REAL*8	24
RVCXB5.	IONO2	REAL*8	24
RVCXB6.	IONO2	REAL*8	24
RVCXE5.	IONO2	REAL*8	24
RVCXE6.	IONO2	REAL*8	24

```
Line#  Source Line

  632          SUBROUTINE CASE1(RVSxH, IVSxJ)
  633  C
  634  C-----------------------------------------------------------------
  635  C
  636  C    CASE1 -- SUBROUTINE TO SORT OUT WHERE IN HEIGHT THE PROGRAM
  637  C        IS IN THE CASE 1 PROFILE
  638  C
  639  C        CALLED BY: IONOPAR
  640  C
  641  C        CALLS:  PGVAL, PGF2, PGFB
  642  C
  643  C-----------------------------------------------------------------
  644  C
  645  C        AUTHOR:          ERIC L. STROBEL & MICHAEL H. REILLY
  646  C
  647  C        DATE:            09/01/87
  648  C
  649  C        VERSION:         1.0
  650  C
  651  C-----------------------------------------------------------------
  652  C
  653  C        REVISED:         09/01/87 -- V1.0.  Initial revision.
  654  C
  655  C-----------------------------------------------------------------
  656  C
  657  C        USES:         RVSxH           The current height.
  658  C
  659  C            To determine which part of the CASE 1 profile is being
  660  C         operated on.  The CASE 1 profile consists of (above the E)
  661  C         the linear valley, the parabolic F2, and the topside.
  662  C
  663  C        RETURNS:      IVSxJ           This helps distinguish which
  664  C                                   boundary is being approached.
  665  C
  666  C-----------------------------------------------------------------
  667  C
  668          INTEGER IVSxJ, IVCxSZ, IVCxCASE, IVCxSX, IVCxSY
  669  C
  670          REAL*8 RVSxH, RVCxHU, RVCxHBND, RVCxHL, RVCxHT5
  671          REAL*8 RPCxHTP, RPCxPI, RPCxDR, RPCxRE
  672          REAL*8 RVCxV1, RVCxV2, RVCxXX
  673          REAL*8 RVCxF40, RVCxF65, RVCxK1, RVCxXL, RVCxXU
  674          REAL*8 RVCxA2, RVCxB2, RVCxC2, RVCxHT3, RVCxHT4
  675          REAL*8 RVCxL1, RVCxLO1, RVCxHBOT, RVCxFREQ, RVCxRPI
  676          REAL*8 RVCxHCT, RTLx1, RTLx2
  677  C
  678          COMMON /PRAM/ RPCxPI, RPCxDR, RPCxRE, RPCxHTP
  679          COMMON /IONO3/ RVCxV1, RVCxV2, RVCxXX, IVCxCASE, RVCxHBND
  680          COMMON /MORE/ IVCxSX, IVCxSY, IVCxSZ
  681          COMMON /TEMP1/ RVCxF40, RVCxF65, RVCxK1, RVCxHL
```

```
Line#  Source Line

  682           COMMON /VAR1/ RVCxXL, RVCxXU, RVCxHU, RTLx1, RTLx2
  683           COMMON /VAR4/ RVCxA2,RVCxB2,RVCxC2,RVCxHT3,RVCxHT4,RVCxHT5
  684           COMMON /OTHER/ RVCxL1,RVCxLO1,RVCxHBOT,RVCxFREQ,RVCxRPI,RVCxHCT
  685   C
  686           IF (RVSxH.LE.RVCxHU) THEN
  687   C
  688   C       We're in the valley region, so go calculate the parameters
  689   C         from the valley profile.
  690   C
  691               CALL PGVAL(RVSxH)
  692               IF (IVCxSZ.GT.0) THEN
  693                   RVCxHBND = RVCxHU
  694                   IVSxJ = 5
  695               ELSE
  696                   RVCxHBND = RVCxHL
  697                   IVSxJ = 4
  698               ENDIF
  699               RETURN
  700           ENDIF
  701           IF (RVSxH.LE.RVCxHT5) THEN
  702   C
  703   C       We're in the F2 region, so go calculate the parameters
  704   C         from the F2 profile.
  705   C
  706               CALL PGF2(RVSxH)
  707               IF (IVCxSZ.GT.0) THEN
  708                   RVCxHBND = RVCxHT5
  709                   IVSxJ = 6
  710               ELSE
  711                   RVCxHBND = RVCxHU
  712                   IVSxJ = 5
  713               ENDIF
  714               RETURN
  715           ENDIF
  716           IF (RVSxH.LE.RPCxHTP) THEN
  717   C
  718   C       We're in the topside region, so go calculate the parameters
  719   C         from the topside profile.
  720   C
  721               CALL PGFB(RVSxH)
  722               IF (IVCxSZ.GT.0) THEN
  723                   RVCxHBND = RPCxHTP
  724                   IVSxJ = 7
  725               ELSE
  726                   RVCxHBND = RVCxHT5
  727                   IVSxJ = 6
  728               ENDIF
  729           ELSE
  730   C
  731   C       We're beyond the cutoff of the ionosphere.
```

```
 Line#  Source Line

   732  C
   733                IF (IVCxSZ.GT.0) THEN
   734                    RVCxHBND = RVCxHCT
   735                ELSE
   736                    RVCxHBND = RPCxHTP
   737                ENDIF
   738            ENDIF
   739  C
   740  C         Take care of things if the ray is headed toward a
   741  C            boundary that's above the cutoff height.
   742  C
   743            IF (IVCxSZ.GT.0.AND.RVCxHBND.GE.RVCxHCT) THEN
   744                RVCxHBND = RVCxHCT
   745                IVSxJ = 1
   746            ENDIF
   747  C
   748            RETURN
   749            END
```

CASE1 Local Symbols

Name	Class	Type	Size
IVSXJ	param		
RVSXH	param		
RVCXXX	IONO3	REAL*8	8
RVCXF40	TEMP1	REAL*8	8
RVCXF65	TEMP1	REAL*8	8
RVCXK1	TEMP1	REAL*8	8
RVCXXL	VAR1	REAL*8	8
RVCXXU	VAR1	REAL*8	8
RVCXA2	VAR4	REAL*8	8
RVCXB2	VAR4	REAL*8	8
RVCXC2	VAR4	REAL*8	8
RVCXHT3	VAR4	REAL*8	8
RVCXHT4	VAR4	REAL*8	8
RVCXL1	OTHER	REAL*8	8
RVCXLO1	OTHER	REAL*8	8
RVCXHBOT	OTHER	REAL*8	8
RVCXFREQ	OTHER	REAL*8	8
RVCXRPI	OTHER	REAL*8	8
RVCXHCT	OTHER	REAL*8	8
RTLX1	VAR1	REAL*8	8
RTLX2	VAR1	REAL*8	8
IVCXSZ	MORE	INTEGER*4	4
IVCXCASE	IONO3	INTEGER*4	4
IVCXSX	MORE	INTEGER*4	4
IVCXSY	MORE	INTEGER*4	4
RVCXHU	VAR1	REAL*8	8

CASE1 Local Symbols

Name	Class	Type	Size
RVCXHBND.	IONO3	REAL*8	8
RVCXHL.	TEMP1	REAL*8	8
RVCXHT5	VAR4	REAL*8	8
RPCXHTP	PRAM	REAL*8	8
RPCXPI.	PRAM	REAL*8	8
RPCXDR.	PRAM	REAL*8	8
RPCXRE.	PRAM	REAL*8	8
RVCXV1.	IONO3	REAL*8	8
RVCXV2.	IONO3	REAL*8	8

```
Line#  Source Line

  751          SUBROUTINE CASE2(RVSxH, IVSxJ)
  752  C
  753  C-----------------------------------------------------------------
  754  C
  755  C    CASE2 -- SUBROUTINE TO SORT OUT WHERE IN HEIGHT THE PROGRAM
  756  C        IS IN THE CASE 2 PROFILE
  757  C
  758  C        CALLED BY: IONOPAR
  759  C
  760  C        CALLS:  PGVAL, PGF1P, PGF2, PGFB
  761  C
  762  C-----------------------------------------------------------------
  763  C
  764  C        AUTHOR:          ERIC L. STROBEL & MICHAEL H. REILLY
  765  C
  766  C        DATE:            09/01/87
  767  C
  768  C        VERSION:         1.0
  769  C
  770  C-----------------------------------------------------------------
  771  C
  772  C        REVISED:         09/01/87 -- V1.0.  Initial revision.
  773  C
  774  C-----------------------------------------------------------------
  775  C
  776  C        USES:         RVSxH           The current height.
  777  C
  778  C             To determine which part of the CASE 2 profile is being
  779  C         operated on.  The CASE 2 profile consists of (above the E)
  780  C         the linear valley, a parabolic F1, the parabolic F2,
  781  C         and the topside.
  782  C
  783  C        RETURNS:      IVSxJ           This helps distinguish which
  784  C                                   boundary is being approached.
  785  C
  786  C-----------------------------------------------------------------
  787  C
  788          INTEGER IVSxJ, IVCxSZ, IVCxSX, IVCxSY, IVCxCASE
  789  C
  790          REAL*8 RVSxH, RVCxH1P, RVCxHBND, RVCxHL, RVCxHT4
  791          REAL*8 RVCxHT5, RPCxHTP, RVCxHCT, RPCxPI, RPCxDR, RPCxRE
  792          REAL*8 RVCxL1, RVCxLO1, RVCxHBOT, RVCxFREQ, RVCxRPI
  793          REAL*8 RVCxV1, RVCxV2, RVCxXX
  794          REAL*8 RVCxF40, RVCxF65, RVCxK1, RVCxA1, RVCxB1, RVCxC1
  795          REAL*8 RVCxA2, RVCxB2, RVCxC2, RVCxHT3, RTCxA
  796  C
  797          COMMON /PRAM/ RPCxPI, RPCxDR, RPCxRE, RPCxHTP
  798          COMMON /OTHER/ RVCxL1,RVCxLO1,RVCxHBOT,RVCxFREQ,RVCxRPI,RVCxHCT
  799          COMMON /IONO3/ RVCxV1, RVCxV2, RVCxXX, IVCxCASE, RVCxHBND
  800          COMMON /MORE/ IVCxSX, IVCxSY, IVCxSZ
```

```
Line#  Source Line

  801           COMMON /TEMP1/ RVCxF40, RVCxF65, RVCxK1, RVCxHL
  802           COMMON /VAR3/ RVCxA1, RVCxB1, RVCxC1, RVCxH1P, RTCxA
  803           COMMON /VAR4/ RVCxA2,RVCxB2,RVCxC2,RVCxHT3,RVCxHT4,RVCxHT5
  804   C
  805           IF (RVSxH.LE.RVCxH1P) THEN
  806   C
  807   C       We're in the valley region, so go calculate the parameters
  808   C         from the valley profile.
  809   C
  810               CALL PGVAL(RVSxH)
  811               IF (IVCxSZ.GT.0) THEN
  812                   RVCxHBND = RVCxH1P
  813                   IVSxJ = 5
  814               ELSE
  815                   RVCxHBND = RVCxHL
  816                   IVSxJ = 4
  817               ENDIF
  818               RETURN
  819           ENDIF
  820           IF (RVSxH.LE.RVCxHT4) THEN
  821   C
  822   C       We're in the F1 region, so go calculate the parameters
  823   C         from the F1 profile.
  824   C
  825               CALL PGF1P(RVSxH)
  826               IF (IVCxSZ.GT.0) THEN
  827                   RVCxHBND = RVCxHT4
  828                   IVSxJ = 6
  829               ELSE
  830                   RVCxHBND = RVCxH1P
  831                   IVSxJ = 5
  832               ENDIF
  833               RETURN
  834           ENDIF
  835           IF (RVSxH.LE.RVCxHT5) THEN
  836   C
  837   C       We're in the F2 region, so go calculate the parameters
  838   C         from the F2 profile.
  839   C
  840               CALL PGF2(RVSxH)
  841               IF (IVCxSZ.GT.0) THEN
  842                   RVCxHBND = RVCxHT5
  843                   IVSxJ = 7
  844               ELSE
  845                   RVCxHBND = RVCxHT4
  846                   IVSxJ = 6
  847               ENDIF
  848               RETURN
  849           ENDIF
  850           IF (RVSxH.LE.RPCxHTP) THEN
```

```
Line#  Source Line

  851  C
  852  C       We're in the topside region, so go calculate the parameters
  853  C         from the topside profile.
  854  C
  855                CALL PGFB(RVSxH)
  856                IF (IVCxSZ.GT.0) THEN
  857                    RVCxHBND = RPCxHTP
  858                    IVSxJ = 8
  859                ELSE
  860                    RVCxHBND = RVCxHT5
  861                    IVSxJ = 7
  862                ENDIF
  863           ELSE
  864  C
  865  C       We're beyond the cutoff of the ionosphere.
  866  C
  867                IF (IVCxSZ.GT.0) THEN
  868                    RVCxHBND = RVCxHCT
  869                ELSE
  870                    RVCxHBND = RPCxHTP
  871                ENDIF
  872           ENDIF
  873  C
  874  C       Take care of things if the ray is headed toward a
  875  C          boundary that's above the cutoff height.
  876  C
  877           IF (IVCxSZ.GT.0.AND.RVCxHBND.GE.RVCxHCT) THEN
  878               RVCxHBND = RVCxHCT
  879               IVSxJ = 1
  880           ENDIF
  881  C
  882           RETURN
  883           END


CASE2  Local Symbols

Name                          Class    Type              Size

IVSXJ . . . . . . . . . . . . param
RVSXH . . . . . . . . . . . . param
RVCXLO1 . . . . . . . . . . . OTHER    REAL*8               8
RVCXHBOT. . . . . . . . . . . OTHER    REAL*8               8
RVCXFREQ. . . . . . . . . . . OTHER    REAL*8               8
RVCXRPI . . . . . . . . . . . OTHER    REAL*8               8
RVCXV1. . . . . . . . . . . . IONO3    REAL*8               8
RVCXV2. . . . . . . . . . . . IONO3    REAL*8               8
RVCXXX. . . . . . . . . . . . IONO3    REAL*8               8
RVCXF40 . . . . . . . . . . . TEMP1    REAL*8               8
RVCXF65 . . . . . . . . . . . TEMP1    REAL*8               8
```

CASE2 Local Symbols

Name	Class	Type	Size
RVCXK1	TEMP1	REAL*8	8
RVCXA1	VAR3	REAL*8	8
RVCXB1	VAR3	REAL*8	8
RVCXC1	VAR3	REAL*8	8
RVCXA2	VAR4	REAL*8	8
RVCXB2	VAR4	REAL*8	8
RVCXC2	VAR4	REAL*8	8
RVCXHT3	VAR4	REAL*8	8
RTCXA	VAR3	REAL*8	8
IVCXSZ	MORE	INTEGER*4	4
IVCXSX	MORE	INTEGER*4	4
IVCXSY	MORE	INTEGER*4	4
IVCXCASE	IONO3	INTEGER*4	4
RVCXH1P	VAR3	REAL*8	8
RVCXHBND	IONO3	REAL*8	8
RVCXHL	TEMP1	REAL*8	8
RVCXHT4	VAR4	REAL*8	8
RVCXHT5	VAR4	REAL*8	8
RPCXHTP	PRAM	REAL*8	8
RVCXHCT	OTHER	REAL*8	8
RPCXPI	PRAM	REAL*8	8
RPCXDR	PRAM	REAL*8	8
RPCXRE	PRAM	REAL*8	8
RVCXL1	OTHER	REAL*8	8

```
Line#  Source Line

  885          SUBROUTINE CASE3(RVSxH, IVSxJ)
  886  C
  887  C-----------------------------------------------------------------
  888  C
  889  C    CASE3 -- SUBROUTINE TO SORT OUT WHERE IN HEIGHT THE PROGRAM
  890  C        IS IN THE CASE 3 PROFILE
  891  C
  892  C        CALLED BY: IONOPAR
  893  C
  894  C        CALLS:  PGVAL, PGF1L, PGF2, PGFB
  895  C
  896  C-----------------------------------------------------------------
  897  C
  898  C        AUTHOR:          ERIC L. STROBEL & MICHAEL H. REILLY
  899  C
  900  C        DATE:            09/01/87
  901  C
  902  C        VERSION:         1.0
  903  C
  904  C-----------------------------------------------------------------
  905  C
  906  C        REVISED:         09/01/87 -- V1.0.  Initial revision.
  907  C
  908  C-----------------------------------------------------------------
  909  C
  910  C        USES:         RVSxH           The current height.
  911  C
  912  C            To determine which part of the CASE 3 profile is being
  913  C         operated on.  The CASE 3 profile consists of (above the E
  914  C         the linear valley, a linear F1, the parabolic F2,
  915  C         and the topside.
  916  C
  917  C        RETURNS:      IVSxJ           This helps distinguish which
  918  C                                   boundary is being approached.
  919  C
  920  C-----------------------------------------------------------------
  921  C
  922          INTEGER IVSxJ, IVCxSZ, IVCxCASE, IVCxSX, IVCxSY
  923  C
  924          REAL*8 RVSxH, RVCxH1L, RVCxHBND, RVCxHL, RVCxH2
  925          REAL*8 RVCxHT5, RPCxHTP, RVCxHCT, RPCxPI, RPCxDR
  926          REAL*8 RPCxRE, RVCxL1, RVCxLO1, RVCxHBOT, RVCxFREQ
  927          REAL*8 RVCxRPI, RVCxV1, RVCxV2, RVCxXX, RVCxF40, RVCxF65
  928          REAL*8 RVCxHB1, RVCxYS, RVCxSL1, RVCxSL2, RVCxA2, RVCxH1
  929          REAL*8 RVCxB2, RVCxC2, RVCxHT3, RVCxHT4
  930  C
  931          COMMON /PRAM/ RPCxPI, RPCxDR, RPCxRE, RPCxHTP
  932          COMMON /OTHER/ RVCxL1,RVCxLO1,RVCxHBOT,RVCxFREQ,RVCxRPI,RVCxHCT
  933          COMMON /IONO3/ RVCxV1, RVCxV2, RVCxXX, IVCxCASE, RVCxHBND
  934          COMMON /MORE/ IVCxSX, IVCxSY, IVCxSZ
```

```
Line#  Source Line

  935           COMMON /TEMP1/ RVCxF40, RVCxF65, RVCxK1, RVCxHL
  936           COMMON /VAR2/ RVCxHB1,RVCxH2,RVCxYS,RVCxSL1,RVCxSL2,RVCxH1L
  937           COMMON /VAR4/ RVCxA2,RVCxB2,RVCxC2,RVCxHT3,RVCxHT4,RVCxHT5
  938   C
  939           IF (RVSxH.LE.RVCxH1L) THEN
  940   C
  941   C       We're in the valley region, so go calculate the parameters
  942   C         from the valley profile.
  943   C
  944               CALL PGVAL(RVSxH)
  945               IF (IVCxSZ.GT.0) THEN
  946                   RVCxHBND = RVCxH1L
  947                   IVSxJ = 5
  948               ELSE
  949                   RVCxHBND = RVCxHL
  950                   IVSxJ = 4
  951               ENDIF
  952               RETURN
  953           ENDIF
  954           IF (RVSxH.LE.RVCxH2) THEN
  955   C
  956   C       We're in the F1 region, so go calculate the parameters
  957   C         from the F1 profile.
  958   C
  959               CALL PGF1L(RVSxH)
  960               IF (IVCxSZ.GT.0) THEN
  961                   RVCxHBND = RVCxH2
  962                   IVSxJ = 6
  963               ELSE
  964                   RVCxHBND = RVCxH1L
  965                   IVSxJ = 5
  966               ENDIF
  967               RETURN
  968           ENDIF
  969           IF (RVSxH.LE.RVCxHT5) THEN
  970   C
  971   C       We're in the F2 region, so go calculate the parameters
  972   C         from the F2 profile.
  973   C
  974               CALL PGF2(RVSxH)
  975               IF (IVCxSZ.GT.0) THEN
  976                   RVCxHBND = RVCxHT5
  977                   IVSxJ = 7
  978               ELSE
  979                   RVCxHBND = RVCxH2
  980                   IVSxJ =6
  981               ENDIF
  982               RETURN
  983           ENDIF
  984           IF (RVSxH.LE.RPCxHTP) THEN
```

```
Line#  Source Line

  985  C
  986  C       We're in the topside region, so go calculate the parameters
  987  C         from the topside profile.
  988  C
  989                CALL PGFB(RVSxH)
  990                IF (IVCxSZ.GT.0) THEN
  991                    RVCxHBND = RPCxHTP
  992                    IVSxJ = 8
  993                ELSE
  994                    RVCxHBND = RVCxHT5
  995                    IVSxJ = 7
  996                ENDIF
  997            ELSE
  998  C
  999  C       We're beyond the cutoff of the ionosphere.
 1000  C
 1001                IF (IVCxSZ.GT.0) THEN
 1002                    RVCxHBND = RVCxHCT
 1003                ELSE
 1004                    RVCxHBND = RPCxHTP
 1005                ENDIF
 1006            ENDIF
 1007  C
 1008  C       Take care of things if the ray is headed toward a
 1009  C          boundary that's above the cutoff height.
 1010  C
 1011            IF (IVCxSZ.GT.0.AND.RVCxHBND.GE.RVCxHCT) THEN
 1012                RVCxHBND = RVCxHCT
 1013                IVSxJ = 1
 1014            ENDIF
 1015  C
 1016            RETURN
 1017            END
```

CASE3 Local Symbols

Name	Class	Type	Size
IVSXJ	param		
RVSXH	param		
RVCXHCT	OTHER	REAL*8	8
RPCXPI	PRAM	REAL*8	8
RPCXDR	PRAM	REAL*8	8
RPCXRE	PRAM	REAL*8	8
RVCXL1	OTHER	REAL*8	8
RVCXLO1	OTHER	REAL*8	8
RVCXHBOT	OTHER	REAL*8	8
RVCXFPEQ	OTHER	REAL*8	8
RVCXRPI	OTHER	REAL*8	8

CASE3 Local Symbols

Name	Class	Type	Size
RVCXV1	IONO3	REAL*8	8
RVCXV2	IONO3	REAL*8	8
RVCXXX	IONO3	REAL*8	8
RVCXF40	TEMP1	REAL*8	8
RVCXF65	TEMP1	REAL*8	8
RVCXHB1	VAR2	REAL*8	8
RVCXYS	VAR2	REAL*8	8
RVCXSL1	VAR2	REAL*8	8
RVCXSL2	VAR2	REAL*8	8
RVCXA2	VAR4	REAL*8	8
RVCXK1	TEMP1	REAL*8	8
RVCXB2	VAR4	REAL*8	8
RVCXC2	VAR4	REAL*8	8
RVCXHT3	VAR4	REAL*8	8
RVCXHT4	VAR4	REAL*8	8
IVCXSZ	MORE	INTEGER*4	4
IVCXCASE	IONO3	INTEGER*4	4
IVCXSX	MORE	INTEGER*4	4
IVCXSY	MORE	INTEGER*4	4
RVCXH1L	VAR2	REAL*8	8
RVCXHBND	IONO3	REAL*8	8
RVCXHL	TEMP1	REAL*8	8
RVCXH2	VAR2	REAL*8	8
RVCXHT5	VAR4	REAL*8	8
RPCXHTP	PRAM	REAL*8	8

```
Line#  Source Line

 1019          SUBROUTINE CASE4(RVSxH, IVSxJ)
 1020  C
 1021  C-----------------------------------------------------------------
 1022  C
 1023  C    CASE4 -- SUBROUTINE TO SORT OUT WHERE IN HEIGHT THE PROGRAM
 1024  C        IS IN THE CASE 4 PROFILE
 1025  C
 1026  C        CALLED BY: IONOPAR
 1027  C
 1028  C        CALLS:  PGVAL, PGF1P, PGF2, PGFB
 1029  C
 1030  C-----------------------------------------------------------------
 1031  C
 1032  C        AUTHOR:          ERIC L. STROBEL & MICHAEL H. REILLY
 1033  C
 1034  C        DATE:            09/01/87
 1035  C
 1036  C        VERSION:         1.0
 1037  C
 1038  C-----------------------------------------------------------------
 1039  C
 1040  C        REVISED:         09/01/87 -- V1.0.  Initial revision.
 1041  C
 1042  C-----------------------------------------------------------------
 1043  C
 1044  C        USES:         RVSxH          The current height.
 1045  C
 1046  C            To determine which part of the CASE 4 profile is being
 1047  C         operated on.  The CASE 4 profile consists of (above the E,
 1048  C         the linear valley, a portion of the F2, a parabolic F1,
 1049  C         the rest of the parabolic F2, and the topside.
 1050  C
 1051  C        RETURNS:      IVSxJ          This helps distinguish which
 1052  C                                   boundary is being approached.
 1053  C
 1054  C-----------------------------------------------------------------
 1055  C
 1056          INTEGER IVSxJ, IVCxSZ, IVCxCASE, IVCxSX, IVCxSY
 1057  C
 1058          REAL*8 RVSxH, RVCxHU, RVCxHBND, RVCxHL, RVCxHT3, RVCxHT4
 1059          REAL*8 RVCxHT5, RPCxHTP, RVCxHCT, RPCxPI, RPCxDR, RPCxRE
 1060          REAL*8 RVCxL1, RVCxLO1, RVCxHBOT, RVCxFREQ, RVCxRPI
 1061          REAL*8 RVCxV1, RVCxV2, RVCxXX, RVCxA2, RVCxB2, RVCxC2
 1062          REAL*8 RVCxF40, RVCxF65, RVCxK1, RVCxXL, RVCxXU
 1063          REAL*8 RTLx1, RTLx2
 1064  C
 1065          COMMON /PRAM/ RPCxPI, RPCxDR, RPCxRE, RPCxHTP
 1066          COMMON /OTHER/ RVCxL1,RVCxLO1,RVCxHBOT,RVCxFREQ,RVCxRPI,RVCxHCT
 1067          COMMON /IONO3/ RVCxV1, RVCxV2, RVCxXX, IVCxCASE, RVCxHBND
 1068          COMMON /MORE/ IVCxSX, IVCxSY, IVCxSZ
```

```
Line#  Source Line

 1069           COMMON /TEMP1/ RVCxF40, RVCxF65, RVCxK1, RVCxHL
 1070           COMMON /VAR1/ RVCxXL, RVCxXU, RVCxHU, RTLx1, RTLx2
 1071           COMMON /VAR4/ RVCxA2,RVCxB2,RVCxC2,RVCxHT3,RVCxHT4,RVCxHT5
 1072   C
 1073           IF (RVSxH.LE.RVCxHU) THEN
 1074   C
 1075   C       We're in the valley region, so go calculate the parameters
 1076   C         from the valley profile.
 1077   C
 1078               CALL PGVAL(RVSxH)
 1079               IF (IVCxSZ.GT.0) THEN
 1080                   RVCxHBND = RVCxHU
 1081                   IVSxJ = 5
 1082               ELSE
 1083                   RVCxHBND = RVCxHL
 1084                   IVSxJ = 4
 1085               ENDIF
 1086               RETURN
 1087           ENDIF
 1088           IF (RVSxH.LE.RVCxHT3) THEN
 1089   C
 1090   C       A portion of the F2 parabola is visible under the F1 layer.
 1091   C       (It's unknown how likely this is, but it at least appears
 1092   C       to be possible in the RADAR-C model, so it is safest to
 1093   C       include code for the possibility.)
 1094   C
 1095               CALL PGF2(RVSxH)
 1096               IF (IVCxSZ.GT.0) THEN
 1097                   RVCxHBND = RVCxHT3
 1098                   IVSxJ = 6
 1099               ELSE
 1100                   RVCxHBND = RVCxHU
 1101                   IVSxJ = 5
 1102               ENDIF
 1103               RETURN
 1104           ENDIF
 1105           IF (RVSxH.LE.RVCxHT4) THEN
 1106   C
 1107   C       We're in the F1 region, so go calculate the parameters
 1108   C         from the F1 profile.
 1109   C
 1110               CALL PGF1P(RVSxH)
 1111               IF (IVCxSZ.GT.0) THEN
 1112                   RVCxHBND = RVCxHT4
 1113                   IVSxJ = 7
 1114               ELSE
 1115                   RVCxHBND = RVCxHT3
 1116                   IVSxJ = 6
 1117               ENDIF
 1118               RETURN
```

```
Line#  Source Line

 1119           ENDIF
 1120           IF (RVSxH.LE.RVCxHT5) THEN
 1121  C
 1122  C        We're in the more ordinary portion of the F2 region, so go
 1123  C        calculate the parameters from the F2 profile.
 1124  C
 1125               CALL PGF2(RVSxH)
 1126               IF (IVCxSZ.GT.0) THEN
 1127                   RVCxHBND = RVCxHT5
 1128                   IVSxJ = 8
 1129               ELSE
 1130                   RVCxHBND = RVCxHT4
 1131                   IVSxJ = 7
 1132               ENDIF
 1133               RETURN
 1134           ENDIF
 1135           IF (RVSxH.LE.RPCxHTP) THEN
 1136  C
 1137  C        We're in the topside region, so go calculate the parameters
 1138  C          from the topside profile.
 1139  C
 1140               CALL PGFB(RVSxH)
 1141               IF (IVCxSZ.GT.0) THEN
 1142                   RVCxHBND = RPCxHTP
 1143                   IVSxJ = 9
 1144               ELSE
 1145                   RVCxHBND = RVCxHT5
 1146                   IVSxJ = 8
 1147               ENDIF
 1148           ELSE
 1149  C
 1150  C        We're beyond the cutoff of the ionosphere.
 1151  C
 1152               IF (IVCxSZ.GT.0) THEN
 1153                   RVCxHBND = RVCxHCT
 1154               ELSE
 1155                   RVCxHBND = RPCxHTP
 1156               ENDIF
 1157           ENDIF
 1158  C
 1159  C        Take care of things if the ray is headed toward a
 1160  C           boundary that's above the cutoff height.
 1161  C
 1162           IF (IVCxSZ.GT.0.AND.RVCxHBND.GE.RVCxHCT) THEN
 1163               RVCxHBND = RVCxHCT
 1164               IVSxJ = 1
 1165           ENDIF
 1166  C
 1167           RETURN
 1168           END
```

CASE4 Local Symbols

Name	Class	Type	Size
IVSXJ	param		
RVSXH	param		
RVCXHU	VAR1	REAL*8	8
RVCXHBND	IONO3	REAL*8	8
RVCXHL	TEMP1	REAL*8	8
RVCXHT3	VAR4	REAL*8	8
RVCXHT4	VAR4	REAL*8	8
RVCXHT5	VAR4	REAL*8	8
RPCXHTP	PRAM	REAL*8	8
RVCXHCT	OTHER	REAL*8	8
RPCXPI	PRAM	REAL*8	8
RPCXDR	PRAM	REAL*8	8
RPCXRE	PRAM	REAL*8	8
RVCXL1	OTHER	REAL*8	8
RVCXLO1	OTHER	REAL*8	8
RVCXHBOT	OTHER	REAL*8	8
RVCXFREQ	OTHER	REAL*8	8
RVCXRPI	OTHER	REAL*8	8
RVCXV1	IONO3	REAL*8	8
RVCXV2	IONO3	REAL*8	8
RVCXXX	IONO3	REAL*8	8
RVCXA2	VAR4	REAL*8	8
RVCXB2	VAR4	REAL*8	8
RVCXC2	VAR4	REAL*8	8
RVCXF40	TEMP1	REAL*8	8
RVCXF65	TEMP1	REAL*8	8
RVCXR1	TEMP1	REAL*8	8
RVCXXL	VAR1	REAL*8	8
RVCXXU	VAR1	REAL*8	8
RTLX1	VAR1	REAL*8	8
RTLX2	VAR1	REAL*8	8
IVCXSZ	MORE	INTEGER*4	4
IVCXCASE	IONO3	INTEGER*4	4
IVCXSX	MORE	INTEGER*4	4
IVCXSY	MORE	INTEGER*4	4

```
Line#  Source Line

 1170          SUBROUTINE CASE5(RVSxH, IVSxJ)
 1171  C
 1172  C-----------------------------------------------------------------
 1173  C
 1174  C    CASE5 -- SUBROUTINE TO SORT OUT WHERE IN HEIGHT THE PROGRAM
 1175  C        IS IN THE CASE 5 PROFILE
 1176  C
 1177  C        CALLED BY: IONOPAR
 1178  C
 1179  C        CALLS:  PGVAL, PGF1P, PGF2, PGFB
 1180  C
 1181  C-----------------------------------------------------------------
 1182  C
 1183  C        AUTHOR:          ERIC L. STROBEL & MICHAEL H. REILLY
 1184  C
 1185  C        DATE:            09/01/87
 1186  C
 1187  C        VERSION:         1.0
 1188  C
 1189  C-----------------------------------------------------------------
 1190  C
 1191  C        REVISED:         09/01/87 -- V1.0.  Initial revision.
 1192  C
 1193  C-----------------------------------------------------------------
 1194  C
 1195  C        USES:         RVSxH          The current height.
 1196  C
 1197  C            To determine which part of the CASE 5 profile is being
 1198  C         operated on.  The CASE 5 profile consists of (above the E
 1199  C         the linear valley, a parabolic F1, an additional valley
 1200  C         segment, the parabolic F2, and the topside.
 1201  C
 1202  C        RETURNS:      IVSxJ          This helps distinguish which
 1203  C                                   boundary is being approached.
 1204  C
 1205  C-----------------------------------------------------------------
 1206  C
 1207          INTEGER IVSxJ, IVCxSZ, IVCxCASE, IVCxSX, IVCxSY
 1208  C
 1209          REAL*8 RVSxH, RVCxHU, RVCxHBND, RVCxHL, RVCxHT3, RVCxHT4
 1210          REAL*8 RVCxHT5, RPCxHTP, RVCxHCT, RPCxPI, RPCxDR, RPCxRE
 1211          REAL*8 RVCxL1, RVCxLO1, RVCxHBOT, RVCxFREQ, RVCxRPI
 1212          REAL*8 RVCxV1, RVCxV2, RVCxXX, RVCxA2, RVCxB2, RVCxC2
 1213          REAL*8 RVCxF40, RVCxF65, RVCxK1, RVCxXL, RVCxXU
 1214          REAL*8 RVCxA1, RVCxB1, RVCxC1, RVCxH1P, RVCxH2P
 1215          REAL*8 RTLx1, RTLx2
 1216  C
 1217          COMMON /PRAM/ RPCxPI, RPCxDR, RPCxRE, RPCxHTP
 1218          COMMON /OTHER/ RVCxL1,RVCxLO1,RVCxHBOT,RVCxFREQ,RVCxRPI,RVCxHCT
 1219          COMMON /IONO3/ RVCxV1, RVCxV2, RVCxXX, IVCxCASE, RVCxHBND
```

```
Line#  Source Line

 1220           COMMON /MORE/ IVCxSX, IVCxSY, IVCxSZ
 1221           COMMON /TEMP1/ RVCxF40, RVCxF65, RVCxK1, RVCxHL
 1222           COMMON /VAR1/ RVCxXL, RVCxXU, RVCxHU, RTLx1, RTLx2
 1223           COMMON /VAR3/ RVCxA1, RVCxB1, RVCxC1, RVCxH1P, RVCxH2P
 1224           COMMON /VAR4/ RVCxA2,RVCxB2,RVCxC2,RVCxHT3,RVCxHT4,RVCxHT5
 1225   C
 1226           IF (RVSxH.LE.RVCxH1P) THEN
 1227   C
 1228   C       We're in the valley region, so go calculate the parameters
 1229   C         from the valley profile.
 1230   C
 1231               CALL PGVAL(RVSxH)
 1232               IF (IVCxSZ.GT.0) THEN
 1233                   RVCxHBND = RVCxH1P
 1234                   IVSxJ = 5
 1235               ELSE
 1236                   RVCxHBND = RVCxHL
 1237                   IVSxJ = 4
 1238               ENDIF
 1239               RETURN
 1240           ENDIF
 1241           IF (RVSxH.LE.RVCxH2P) THEN
 1242   C
 1243   C       We're in the F1 region, so go calculate the parameters
 1244   C         from the F1 profile.
 1245   C
 1246               CALL PGF1P(RVSxH)
 1247               IF (IVCxSZ.GT.0) THEN
 1248                   RVCxHBND = RVCxH2P
 1249                   IVSxJ = 6
 1250               ELSE
 1251                   RVCxHBND = RVCxH1P
 1252                   IVSxJ = 5
 1253               ENDIF
 1254               RETURN
 1255           ENDIF
 1256           IF (RVSxH.LE.RVCxHU) THEN
 1257   C
 1258   C       Surprise!  A portion of the valley profile is seen above
 1259   C         the F1.  And yes, this really can in fact happen in the
 1260   C         RADAR-C model.
 1261   C
 1262               CALL PGVAL(RVSxH)
 1263               IF (IVCxSZ.GT.0) THEN
 1264                   RVCxHBND = RVCxHU
 1265                   IVSxJ = 7
 1266               ELSE
 1267                   RVCxHBND = RVCxH2P
 1268                   IVSxJ = 6
 1269               ENDIF
```

```
Line#  Source Line

 1270              RETURN
 1271            ENDIF
 1272            IF (RVSxH.LE.RVCxHT5) THEN
 1273  C
 1274  C         We're in the F2 region, so go calculate the parameters
 1275  C           from the F2 profile.
 1276  C
 1277                CALL PGF2(RVSxH)
 1278                IF (IVCxSZ.GT.0) THEN
 1279                    RVCxHBND = RVCxHT5
 1280                    IVSxJ = 8
 1281                ELSE
 1282                    RVCxHBND = RVCxHU
 1283                    IVSxJ = 7
 1284                ENDIF
 1285                RETURN
 1286            ENDIF
 1287            IF (RVSxH.LE.RPCxHTP) THEN
 1288  C
 1289  C         We're in the topside region, so go calculate the parameters
 1290  C           from the topside profile.
 1291  C
 1292                CALL PGFB(RVSxH)
 1293                IF (IVCxSZ.GT.0) THEN
 1294                    RVCxHBND = RPCxHTP
 1295                    IVSxJ = 9
 1296                ELSE
 1297                    RVCxHBND = RVCxHT5
 1298                    IVSxJ = 8
 1299                ENDIF
 1300            ELSE
 1301  C
 1302  C         We're beyond the cutoff of the ionosphere.
 1303  C
 1304                IF (IVCxSZ.GT.0) THEN
 1305                    RVCxHBND = RVCxHCT
 1306                ELSE
 1307                    RVCxHBND = RPCxHTP
 1308                ENDIF
 1309            ENDIF
 1310  C
 1311  C         Take care of things if the ray is headed toward a
 1312  C            boundary that's above the cutoff height.
 1313  C
 1314            IF (IVCxSZ.GT.0.AND.RVCxHBND.GE.RVCxHCT) THEN
 1315                RVCxHBND = RVCxHCT
 1316                IVSxJ = 1
 1317            ENDIF
 1318  C
 1319            RETURN
```

```
 Line#  Source Line

  1320          END
```

CASE5 Local Symbols

Name	Class	Type	Size
IVSXJ	param		
RVSXH	param		
IVCXSZ	MORE	INTEGER*4	4
IVCXCASE	IONO3	INTEGER*4	4
IVCXSX	MORE	INTEGER*4	4
IVCXSY	MORE	INTEGER*4	4
RVCXHU	VAR1	REAL*8	8
RVCXHBND	IONO3	REAL*8	8
RVCXHL	TEMP1	REAL*8	8
RVCXHT3	VAR4	REAL*8	8
RVCXHT4	VAR4	REAL*8	8
RVCXHT5	VAR4	REAL*8	8
RPCXHTP	PRAM	REAL*8	8
RVCXHCT	OTHER	REAL*8	8
RPCXPI	PRAM	REAL*8	8
RPCXDR	PRAM	REAL*8	8
RPCXRE	PRAM	REAL*8	8
RVCXL1	OTHER	REAL*8	8
RVCXLO1	OTHER	REAL*8	8
RVCXHBOT	OTHER	REAL*8	8
RVCXFREQ	OTHER	REAL*8	8
RVCXRPI	OTHER	REAL*8	8
RVCXV1	IONO3	REAL*8	8
RVCXV2	IONO3	REAL*8	8
RVCXXX	IONO3	REAL*8	8
RVCXA2	VAR4	REAL*8	8
RVCXB2	VAR4	REAL*8	8
RVCXC2	VAR4	REAL*8	8
RVCXF40	TEMP1	REAL*8	8
RVCXF65	TEMP1	REAL*8	8
RVCXK1	TEMP1	REAL*8	8
RVCXXL	VAR1	REAL*8	8
RVCXXU	VAR1	REAL*8	8
RVCXA1	VAR3	REAL*8	8
RVCXB1	VAR3	REAL*8	8
RVCXC1	VAR3	REAL*8	8
RVCXH1P	VAR3	REAL*8	8
RVCXH2P	VAR3	REAL*8	8
RTLX1	VAR1	REAL*8	8
RTLX2	VAR1	REAL*8	8

```
Line#  Source Line

 1323          SUBROUTINE PGVAL(RVSxH)
 1324  C
 1325  C-------------------------------------------------------------
 1326  C
 1327  C     PGVAL -- SUBROUTINE to calculate the ionospheric parameters
 1328  C              for the linear valley profile.
 1329  C
 1330  C
 1331  C        CALLED BY: CASE1, CASE2, CASE3, CASE4, CASE5
 1332  C
 1333  C-------------------------------------------------------------
 1334  C
 1335  C        AUTHOR:          MICHAEL H. REILLY & ERIC L. STROBEL
 1336  C
 1337  C        DATE:            09/01/87
 1338  C
 1339  C        VERSION:         1.0
 1340  C
 1341  C-------------------------------------------------------------
 1342  C
 1343  C        REVISED:         09/01/87 -- V1.0.  Initial revision.
 1344  C
 1345  C-------------------------------------------------------------
 1346  C
 1347  C        USES:            RVSxH         The current height.
 1348  C
 1349  C
 1350  C        To calculate the A, N1, and N2 parameters, as well as the
 1351  C          plasma frequency squared XX.  For more details, see the
 1352  C          Radio Science paper and the RADAR-C report referred to
 1353  C          in the documentation.
 1354  C
 1355  C
 1356  C        RETURNS:         RVCxALPH      A coef. in the index of
 1357  C                                        refraction.
 1358  C
 1359  C                         RVCxET1       A coef. in the index of
 1360  C                                        refraction.
 1361  C
 1362  C                         RVCxET2       A coef. in the index of
 1363  C                                        refraction.
 1364  C
 1365  C                         RVCxXX        The plasma frequency squared.
 1366  C
 1367  C
 1368  C-------------------------------------------------------------
 1369  C
 1370          INTEGER IFCxN4, IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3
 1371          INTEGER ITCxA
 1372  C
```

```
Line#  Source Line

 1373           REAL*8 RVSxH, RVCxXX, RVCxA0, RVCxB0, RVCxALPH, RVCxFREQ
 1374           REAL*8 RVLxHUXE, RVCxHB(3), RVCxFB(3), RVCxHU, RVLxHTR
 1375           REAL*8 RVCxHT3, RVCxHL, RVLxXXXE, RVLxHUX2, RVLxXXH2
 1376           REAL*8 RVLxXXX2, RVLxHUY2, RVLxXXY2, RVCxET1, RVCxB5(3)
 1377           REAL*8 RVCxB6(3), RVCxV1, RVCxET2, RVCxE5(3), RVCxE6(3)
 1378           REAL*8 RVCxV2, RVCxL1, RVCxLO1, RVCxHBOT
 1379           REAL*8 RVCxBETA, RVCxA5(3), RVCxA6(3), RVCxF40, RVCxC2
 1380           REAL*8 RVCxF65, RVCxK1, RVCxXL, RVCxXU, RVCxA2, RVCxB2
 1381           REAL*8 RTCxA, RTCxB, RTCxC, RTCxD, RTCxE
 1382   C
 1383           COMMON /OTHER/ RVCxL1,RVCxLO1,RVCxHBOT,RVCxFREQ,RTCxA,RTCxB
 1384           COMMON /IONO1/ RVCxALPH,RVCxBETA,RVCxET1,RVCxET2,RVCxFB,RVCxHB
 1385           COMMON /IONO2/ RVCxA5,RVCxA6,RVCxB5,RVCxB6,RVCxE5,RVCxE6
 1386           COMMON /IONO3/ RVCxV1, RVCxV2, RVCxXX, ITCxA, RTCxC
 1387           COMMON /TEMP1/ RVCxF40, RVCxF65, RVCxK1, RVCxHL
 1388           COMMON /TEMP2/ IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3, IFCxN4
 1389           COMMON /VAR1/ RVCxXL, RVCxXU, RVCxHU, RVCxA0, RVCxB0
 1390           COMMON /VAR4/ RVCxA2,RVCxB2,RVCxC2,RVCxHT3,RTCxD,RTCxE
 1391   C
 1392           RVCxXX = RVCxA0 + RVCxB0 * RVSxH
 1393           RVCxALPH = RVCxB0 / RVCxFREQ
 1394           IF (IFCxN4.NE.1) THEN
 1395               RVLxHUXE = (0.98D00*RVCxHB(3)) * (0.98D00*RVCxHB(3))
 1396               RVLxHUXE = RVLxHUXE / (2.0D00*RVCxFB(3)*(RVCxHB(2)-RVCxHU)
 1397               RVLxHTR = (RVCxHT3 - RVCxHL)/(RVCxHU - RVCxHL)
 1398               RVLxXXXE = RVCxXX/RVCxFB(1) - RVCxB0*RVLxHTR*RVLxHUXE
 1399               RVLxHUX2 = -RVLxHUXE * RVCxFB(1 /RVCxFB(3)
 1400               RVLxXXH2 = -RVCxB0 * RVLxHTR
 1401               RVLxXXX2 = RVLxXXH2 * RVLxHUX2
 1402               RVLxHUY2 = -(RVCxHB(2) - RVCxHU) / RVCxHB(3)
 1403               RVLxXXY2 = RVLxXXH2 * RVLxHUY2
 1404               RVCxET1 = RVLxXXXE*RVCxB5(1) + RVLxXXX2*RVCxB5(3)
 1405               RVCxET1 = RVCxET1 + RVLxXXH2*RVCxB6(2) + RVLxXXY2*RVCxB6 3
 1406               RVCxET1 = -RVCxET1 / (RVCxV1 * RVCxFREQ)
 1407               RVCxET2 = RVLxXXXE*RVCxE5(1) + RVLxXXX2*RVCxE5(3)
 1408               RVCxET2 = RVCxET2 + RVLxXXH2*RVCxE6(2) + RVLxXXY2*RVCxE6 3
 1409               RVCxET2 = RVCxET2 / (RVCxV2 * RVCxFREQ)
 1410           ENDIF
 1411           RETURN
 1412           END


PGVAL  Local Symbols

Name                          Class    Type                Size

RVSXH . . . . . . . . . . . . param
RVLXXXX2. . . . . . . . . . . local    REAL*8                 3
RVLXHUXE. . . . . . . . . . . local    REAL*8                 8
RVLXXXY2. . . . . . . . . . . local    REAL*8                 8
```

PGVAL Local Symbols

Name	Class	Type	Size
RVLXXXXE	local	REAL*8	8
RVLXHUX2	local	REAL*8	8
RVLXHUY2	local	REAL*8	8
RVLXXXH2	local	REAL*8	8
RVLXHTR	local	REAL*8	8
RVCXA2	VAR4	REAL*8	8
RVCXB2	VAR4	REAL*8	8
RTCXA	OTHER	REAL*8	8
RTCXB	OTHER	REAL*8	8
RTCXC	IONO3	REAL*8	8
RTCXD	VAR4	REAL*8	8
RTCXE	VAR4	REAL*8	8
IFCXN4	TEMP2	INTEGER*4	4
IVCXSCS	TEMP2	INTEGER*4	4
IVCXSCT1	TEMP2	INTEGER*4	4
IVCXSCT2	TEMP2	INTEGER*4	4
IVCXSCT3	TEMP2	INTEGER*4	4
ITCXA	IONO3	INTEGER*4	4
RVCXXX	IONO3	REAL*8	8
RVCXA0	VAR1	REAL*8	8
RVCXB0	VAR1	REAL*8	8
RVCXALPH	IONO1	REAL*8	8
RVCXFREQ	OTHER	REAL*8	8
RVCXHB	IONO1	REAL*8	24
RVCXFB	IONO1	REAL*8	24
RVCXHU	VAR1	REAL*8	8
RVCXHT3	VAR4	REAL*8	8
RVCXHL	TEMP1	REAL*8	8
RVCXET1	IONO1	REAL*8	8
RVCXB5	IONO2	REAL*8	24
RVCXB6	IONO2	REAL*8	24
RVCXV1	IONO3	REAL*8	8
RVCXET2	IONO1	REAL*8	8
RVCXE5	IONO2	REAL*8	24
RVCXE6	IONO2	REAL*8	24
RVCXV2	IONO3	REAL*8	8
RVCXL1	OTHER	REAL*8	8
RVCXLO1	OTHER	REAL*8	8
RVCXHBOT	OTHER	REAL*8	8
RVCXBETA	IONO1	REAL*8	8
RVCXA5	IONO2	REAL*8	24
RVCXA6	IONO2	REAL*8	24
RVCXF40	TEMP1	REAL*8	8
RVCXC2	VAR4	REAL*8	8
RVCXF65	TEMP1	REAL*8	8

PGVAL Local Symbols

Name	Class	Type	Size
RVCXK1.	TEMP1	REAL*8	8
RVCXXL.	VAR1	REAL*8	8
RVCXXU.	VAR1	REAL*8	8

```
Line#  Source Line

 1415          SUBROUTINE PGF1L (RVSxH)
 1416  C
 1417  C-----------------------------------------------------------------
 1418  C
 1419  C     PGF1L -- SUBROUTINE to calculate the ionospheric parameters
 1420  C              for the linear F1 region.
 1421  C
 1422  C
 1423  C         CALLED BY: CASE3
 1424  C
 1425  C-----------------------------------------------------------------
 1426  C
 1427  C         AUTHOR:           MICHAEL H. REILLY & ERIC L. STROBEL
 1428  C
 1429  C         DATE:             09/01/87
 1430  C
 1431  C         VERSION:          1.0
 1432  C
 1433  C-----------------------------------------------------------------
 1434  C
 1435  C         REVISED:          09/01/87 -- V1.0.  Initial revision.
 1436  C
 1437  C-----------------------------------------------------------------
 1438  C
 1439  C         USES:             RVSxH         The current height.
 1440  C
 1441  C
 1442  C         To calculate the A, N1, and N2 parameters, as well as the
 1443  C           plasma frequency squared XX.  For more details, see the
 1444  C           Radio Science paper and the RADAR-C report referred to
 1445  C           in the documentation.
 1446  C
 1447  C
 1448  C         RETURNS:          RVCxALPH      A coef. in the index of
 1449  C                                          refraction.
 1450  C
 1451  C                           RVCxET1       A coef. in the index of
 1452  C                                          refraction.
 1453  C
 1454  C                           RVCxET2       A coef. in the index of
 1455  C                                          refraction.
 1456  C
 1457  C                           RVCxXX        The plasma frequency squared.
 1458  C
 1459  C
 1460  C-----------------------------------------------------------------
 1461  C
 1462          INTEGER IFCxN4, IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3
 1463          INTEGER ITCxA
 1464  C
```

```
Line#  Source Line

 1465           REAL*8 RVCxXX, RVCxSL1, RVLxHTV, RVCxALPH, RVCxFREQ
 1466           REAL*8 RVCxHB1, RVCxYS, RVLxS1X1, RVLxS1H1, RVLxYSX1
 1467           REAL*8 RVCxHB(3), RVCxFB(3), RVCxH2, RVLxXXX1, RVLxXXH1
 1468           REAL*8 RVCxET1, RVCxET2, RVCxB5(3), RVCxB6(3), RVCxE5(3)
 1469           REAL*8 RVCxE6(3), RVCxV1, RVCxV2, RVLxS1X2, RVLxXXX2
 1470           REAL*8 RVLxXXH2, RVLxYSY2, RVLxS1Y2, RVLxXXY2, RVCxA5(3)
 1471           REAL*8 RVCxA6(3),RVCxL1,RVCxLO1,RVCxHBOT,RVCxBETA,RVSxH
 1472           REAL*8 RTCxA, RTCxB, RTCxC, RTCxD, RTCxE
 1473   C
 1474           COMMON /OTHER/ RVCxL1,RVCxLO1,RVCxHBOT,RVCxFREQ,RTCxA,RTCxB
 1475           COMMON /IONO1/ RVCxALPH,RVCxBETA,RVCxET1,RVCxET2,RVCxFB,RVCxHB
 1476           COMMON /IONO2/ RVCxA5,RVCxA6,RVCxB5,RVCxB6,RVCxE5,RVCxE6
 1477           COMMON /IONO3/ RVCxV1, RVCxV2, RVCxXX, ITCxA, RTCxC
 1478           COMMON /TEMP2/ IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3, IFCxN4
 1479           COMMON /VAR2/ RVCxHB1,RVCxH2,RVCxYS,RVCxSL1,RTCxD,RTCxE
 1480   C
 1481           RVLxHTV = RVSxH - RVCxHB1
 1482           RVCxXX = RVCxSL1 * RVLxHTV
 1483           RVCxALPH = RVCxSL1 / RVCxFREQ
 1484           IF (IFCxN4.EQ.1) RETURN
 1485           IF (RVCxYS.EQ.1) THEN
 1486               RVLxS1X1 = 1.0D00
 1487               RVLxS1H1 = 0.0D00
 1488           ELSE
 1489               RVLxYSX1 = RVCxHB(3)*RVCxHB(3)/(2.0D00*RVCxFB(3)*
 1490      #(RVCxHB(2) - RVCxH2))
 1491               RVLxS1X1 = (1.0D00 - (RVCxFB(2)*RVLxYSX1/RVCxYS)/RVCxYS
 1492               RVLxS1H1 = RVCxFB(2) / (RVCxYS*RVCxYS)
 1493           ENDIF
 1494           RVLxXXX1 = RVLxHTV * RVLxS1X1
 1495           RVLxXXH1 = 0.75D00*(-RVCxSL1 + RVLxHTV*RVLxS1H1)
 1496           RVCxET1 = -RVLxXXX1*RVCxB5(2) - RVLxXXH1*RVCxB6(2)
 1497           RVCxET2 = RVLxXXX1 * RVCxE5(2) + RVLxXXH1*RVCxE6(2)
 1498           IF (RVCxYS.EQ.1) THEN
 1499               RVCxET1 = RVCxET1 / (RVCxV1 * RVCxFREQ)
 1500               RVCxET2 = RVCxET2 / (RVCxV2 * RVCxFREQ)
 1501               RETURN
 1502           ENDIF
 1503           RVLxS1X2 = RVLxS1H1 * RVLxYSX1 * RVCxFB(2)/ RVCxFB(3)
 1504           RVLxXXX2 = RVLxHTV * RVLxS1X2
 1505           RVLxXXH2 = -RVLxHTV * RVLxS1H1
 1506           RVLxYSY2 = (RVCxH2 - RVCxHB(2)) / RVCxHB(3)
 1507           RVLxS1Y2 = -RVLxS1H1 * RVLxYSY2
 1508           RVLxXXY2 = RVLxHTV * RVLxS1Y2
 1509           RVCxET1 = RVCxET1 - RVLxXXX2*RVCxB5(3) - RVLxXXH2*RVCxB6(3)
 1510           RVCxET1 = (RVCxET1 - RVLxXXY2*RVCxB6(3)) / (RVCxV1*RVCxFREQ)
 1511           RVCxET2 = RVCxET2 + RVLxXXX2*RVCxE5(3) + RVLxXXH2*RVCxE6(3)
 1512           RVCxET2 = (RVCxET2 + RVLxXXY2*RVCxE6(3)) / (RVCxV2*RVCxFREQ)
 1513           RETURN
 1514           END
```

PGF1L Local Symbols

Name	Class	Type	Size
RVSXH	param		
RVLXYSX1	local	REAL*8	8
RVLXYSY2	local	REAL*8	8
RVLXXXX1	local	REAL*8	8
RVLXXXX2	local	REAL*8	8
RVLXXXY2	local	REAL*8	8
RVLXS1H1	local	REAL*8	8
RVLXS1X1	local	REAL*8	8
RVLXS1X2	local	REAL*8	8
RVLXS1Y2	local	REAL*8	8
RVLXXXH1	local	REAL*8	8
RVLXXXH2	local	REAL*8	8
RVLXHTV	local	REAL*8	8
IFCXN4	TEMP2	INTEGER*4	4
IVCXSCS	TEMP2	INTEGER*4	4
IVCXSCT1	TEMP2	INTEGER*4	4
IVCXSCT2	TEMP2	INTEGER*4	4
IVCXSCT3	TEMP2	INTEGER*4	4
ITCXA	IONO3	INTEGER*4	4
RVCXXX	IONO3	REAL*8	8
RVCXSL1	VAR2	REAL*8	8
RVCXALPH	IONO1	REAL*8	8
RVCXFREQ	OTHER	REAL*8	8
RVCXHB1	VAR2	REAL*8	8
RVCXYS	VAR2	REAL*8	8
RVCXHB	IONO1	REAL*8	24
RVCXFB	IONO1	REAL*8	24
RVCXH2	VAR2	REAL*8	8
RVCXET1	IONO1	REAL*8	8
RVCXET2	IONO1	REAL*8	8
RVCXB5	IONO2	REAL*8	24
RVCXB6	IONO2	REAL*8	24
RVCXE5	IONO2	REAL*8	24
RVCXE6	IONO2	REAL*8	24
RVCXV1	IONO3	REAL*8	8
RVCXV2	IONO3	REAL*8	8
RVCXA5	IONO2	REAL*8	24
RVCXA6	IONO2	REAL*8	24
RVCXL1	OTHER	REAL*8	8
RVCXLO1	OTHER	REAL*8	8
RVCXHBOT	OTHER	REAL*8	8
RVCXBETA	IONO1	REAL*8	8
RTCXA	OTHER	REAL*8	8
RTCXB	OTHER	REAL*8	8

PGF1L Local Symbols

Name	Class	Type	Size
RTCXC	IONO3	REAL*8	8
RTCXD	VAR2	REAL*8	8
RTCXE	VAR2	REAL*8	8

```
Line#  Source Line

 1517          SUBROUTINE PGF1P(RVSxH)
 1518  C
 1519  C-----------------------------------------------------------------
 1520  C
 1521  C     PGF1P -- SUBROUTINE to calculate the ionospheric parameters
 1522  C              for the parabolic F1 region.
 1523  C
 1524  C
 1525  C        CALLED BY: CASE2, CASE4, CASE5
 1526  C
 1527  C-----------------------------------------------------------------
 1528  C
 1529  C        AUTHOR:          MICHAEL H. REILLY & ERIC L. STROBEL
 1530  C
 1531  C        DATE:            09/01/87
 1532  C
 1533  C        VERSION:         1.0
 1534  C
 1535  C-----------------------------------------------------------------
 1536  C
 1537  C        REVISED:         09/01/87 -- V1.0.  Initial revision.
 1538  C
 1539  C-----------------------------------------------------------------
 1540  C
 1541  C        USES:            RVSxH        The current height.
 1542  C
 1543  C
 1544  C        To calculate the A, B, N1, and N2 parameters, as well as the
 1545  C          plasma frequency squared XX.  For more details, see the
 1546  C          Radio Science paper and the RADAR-C report referred to
 1547  C          in the documentation.
 1548  C
 1549  C
 1550  C        RETURNS:         RVCxALPH     A coef. in the index of
 1551  C                                       refraction.
 1552  C
 1553  C                         RVCxBETA     A coef. in the index of
 1554  C                                       refraction.
 1555  C
 1556  C                         RVCxET1      A coef. in the index of
 1557  C                                       refraction.
 1558  C
 1559  C                         RVCxET2      A coef. in the index of
 1560  C                                       refraction.
 1561  C
 1562  C                         RVCxXX       The plasma frequency squared.
 1563  C
 1564  C
 1565  C-----------------------------------------------------------------
 1566  C
```

```
Line#  Source Line

 1567           INTEGER IFCxN4, IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3
 1568           INTEGER ITCxA
 1569  C
 1570           REAL*8 RVSxH, RVLxYF1, RVLxHTV, RVCxHB(3), RVCxXX
 1571           REAL*8 RVCxFB(3), RVCxFREQ, RVCxALPH, RVCxBETA, RVLxXXX1
 1572           REAL*8 RVLxXXH1, RVCxET1, RVCxET2, RVCxB5(3), RVCxB6(3)
 1573           REAL*8 RVCxE5(3), RVCxE6(3), RVCxV1, RVCxV2, RVCxL1, RVCxLO1
 1574           REAL*8 RVCxHBOT, RVCxA5(3), RVCxA6(3), RTCxA, RTCxB
 1575           REAL*8 RTCxC
 1576  C
 1577           COMMON /OTHER/ RVCxL1,RVCxLO1,RVCxHBOT,RVCxFREQ,RTCxA,RTCxB
 1578           COMMON /IONO1/ RVCxALPH,RVCxBETA,RVCxET1,RVCxET2,RVCxFB,RVCxHB
 1579           COMMON /IONO2/ RVCxA5,RVCxA6,RVCxB5,RVCxB6,RVCxE5,RVCxE6
 1580           COMMON /IONO3/ RVCxV1, RVCxV2, RVCxXX, ITCxA, RTCxC
 1581           COMMON /TEMP2/ IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3, IFCxN4
 1582  C
 1583           RVLxYF1 = 0.25D00 * RVCxHB(1)
 1584           RVLxHTV = (RVCxHB(1) - RVSxH) / RVLxYF1
 1585           RVCxXX = RVCxFB(2) * (1.0D00 - RVLxHTV*RVLxHTV)
 1586           RVCxALPH = 2.0D00 * RVCxFB(2) * RVLxHTV / (RVLxYF1 * RVCxFREQ)
 1587           RVCxBETA = RVCxALPH / (2.0D00 * RVLxHTV * RVLxYF1)
 1588           IF (IFCxN4.EQ.1) RETURN
 1589           RVLxXXX1 = RVCxXX / (RVCxFB(2) * RVCxFREQ)
 1590           RVLxXXH1 = -2.0D00 * RVCxBETA * RVSxH *
 1591         #(1.0D00 - RVSxH/RVCxHB(1))
 1592           RVCxET1 = RVLxXXX1*RVCxB5(2) + RVLxXXH1*RVCxB6(1)
 1593           RVCxET1 = -RVCxET1 / RVCxV1
 1594           RVCxET2 = RVLxXXX1*RVCxE5(2) + RVLxXXH1*RVCxE6(1)
 1595           RVCxET2 = RVCxET2 / RVCxV2
 1596           RETURN
 1597           END
```

PGF1P Local Symbols

Name	Class	Type	Size
RVSXH	param		
RVLXXXX1	local	REAL*8	8
RVLXYF1	local	REAL*8	8
RVLXXXH1	local	REAL*8	8
RVLXHTV	local	REAL*8	8
IFCXN4	TEMP2	INTEGER*4	4
IVCXSCS	TEMP2	INTEGER*4	4
IVCXSCT1	TEMP2	INTEGER*4	4
IVCXSCT2	TEMP2	INTEGER*4	4
IVCXSCT3	TEMP2	INTEGER*4	4
ITCXA	IONO3	INTEGER*4	4
RVCXHB	IONO1	REAL*8	24
RVCXXX	IONO3	REAL*8	8

PGF1P Local Symbols

Name	Class	Type	Size
RVCXFB	IONO1	REAL*8	24
RVCXFREQ	OTHER	REAL*8	8
RVCXALPH	IONO1	REAL*8	8
RVCXBETA	IONO1	REAL*8	8
RVCXET1	IONO1	REAL*8	8
RVCXET2	IONO1	REAL*8	8
RVCXB5	IONO2	REAL*8	24
RVCXB6	IONO2	REAL*8	24
RVCXE5	IONO2	REAL*8	24
RVCXE6	IONO2	REAL*8	24
RVCXV1	IONO3	REAL*8	8
RVCXV2	IONO3	REAL*8	8
RVCXL1	OTHER	REAL*8	8
RVCXLO1	OTHER	REAL*8	8
RVCXHBOT	OTHER	REAL*8	8
RVCXA5	IONO2	REAL*8	24
RVCXA6	IONO2	REAL*8	24
RTCXA	OTHER	REAL*8	8
RTCXB	OTHER	REAL*8	8
RTCXC	IONO3	REAL*8	8

```
Line#  Source Line

 1600          SUBROUTINE PGF2(RVSxH)
 1601  C
 1602  C-----------------------------------------------------------------
 1603  C
 1604  C     PGF2 -- SUBROUTINE to calculate the ionospheric parameters
 1605  C             for the parabolic F2 region.
 1606  C
 1607  C
 1608  C        CALLED BY: CASE1, CASE2, CASE3, CASE4, CASE5
 1609  C
 1610  C-----------------------------------------------------------------
 1611  C
 1612  C        AUTHOR:          MICHAEL H. REILLY & ERIC L. STROBEL
 1613  C
 1614  C        DATE:            09/01/87
 1615  C
 1616  C        VERSION:         1.0
 1617  C
 1618  C-----------------------------------------------------------------
 1619  C
 1620  C        REVISED:         09/01/87 -- V1.0.  Initial revision.
 1621  C
 1622  C-----------------------------------------------------------------
 1623  C
 1624  C        USES:            RVSxH        The current height.
 1625  C
 1626  C
 1627  C        To calculate the A, B, N1, and N2 parameters, as well as the
 1628  C          plasma frequency squared XX.  For more details, see the
 1629  C          Radio Science paper and the RADAR-C report referred to
 1630  C          in the documentation.
 1631  C
 1632  C
 1633  C        RETURNS:         RVCxALPH     A coef. in the index of
 1634  C                                       refraction.
 1635  C
 1636  C                         RVCxBETA     A coef. in the index of
 1637  C                                       refraction.
 1638  C
 1639  C                         RVCxET1      A coef. in the index of
 1640  C                                       refraction.
 1641  C
 1642  C                         RVCxET2      A coef. in the index of
 1643  C                                       refraction.
 1644  C
 1645  C                         RVCxXX       The plasma frequency squared.
 1646  C
 1647  C
 1648  C-----------------------------------------------------------------
 1649  C
```

```
Line#  Source Line

 1650          INTEGER IFCxN4, IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3
 1651          INTEGER ITCxA
 1652  C
 1653          REAL*8 RVSxH, RVLxHTV, RVCxHB(3), RTLx1, RVCxXX
 1654          REAL*8 RVCxFB(3), RVCxFREQ, RVCxALPH, RVCxBETA
 1655          REAL*8 RVLxXXX2, RVLxXXH2, RVLxXXY2, RVCxET1, RVCxET2
 1656          REAL*8 RVCxB5(3), RVCxB6(3), RVCxE5(3), RVCxE6(3)
 1657          REAL*8 RVCxV1, RVCxV2, RVCxL1, RVCxLO1, RVCxHBOT
 1658          REAL*8 RVCxA5(3), RVCxA6(3), RTCxA, RTCxB, RTCxC
 1659  C
 1660          COMMON /OTHER/ RVCxL1,RVCxLO1,RVCxHBOT,RVCxFREQ,RTCxA,RTCxB
 1661          COMMON /IONO1/ RVCxALPH,RVCxBETA,RVCxET1,RVCxET2,RVCxFB,RVCxH
 1662          COMMON /IONO2/ RVCxA5,RVCxA6,RVCxB5,RVCxB6,RVCxE5,RVCxE6
 1663          COMMON /IONO3/ RVCxV1, RVCxV2, RVCxXX, ITCxA, RTCxC
 1664          COMMON /TEMP2/ IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3, IFCxN4
 1665  C
 1666          RVLxHTV = (RVCxHB(2) - RVSxH) / RVCxHB(3)
 1667          RTLx1 = 1.0D00 - RVLxHTV*RVLxHTV
 1668          RVCxXX = RVCxFB(3) * RTLx1
 1669          RVCxBETA = RVCxFB(3) / (RVCxHB(3) * RVCxHB(3) * RVCxFREQ)
 1670          RVCxALPH = 2.0D00 * RVCxBETA * ( RVCxHB(2) - RVSxH )
 1671          IF (IFCxN4.EQ.1) RETURN
 1672          RVLxXXX2 = RTLx1
 1673          RVLxXXH2 = -2.0D00 * RVCxFB(3) * RVLxHTV / RVCxHB(3)
 1674          RVLxXXY2 = -RVLxXXH2 * RVLxHTV
 1675          RVCxET1 = RVLxXXX2*RVCxB5(3) + RVLxXXH2*RVCxB6(2)
 1676          RVCxET1 = -(RVCxET1 + RVLxXXY2*RVCxB6(3))/(RVCxV1*RVCxFREQ
 1677          RVCxET2 = RVLxXXX2*RVCxE5(3) + RVLxXXH2*RVCxE6(2)
 1678          RVCxET2 = (RVCxET2 + RVLxXXY2*RVCxE6(3))/(RVCxV2*RVCxFREQ)
 1679          RETURN
 1680          END
```

PGF2 Local Symbols

Name	Class	Type	Size
RVSXH	param		
RVLXXXX2	local	REAL*8	8
RVLXXXY2	local	REAL*8	8
RVLXXXH2	local	REAL*8	8
RTLX1	local	REAL*8	8
RVLXHTV	local	REAL*8	8
RVCXB5	IONO2	REAL*8	24
RVCXB6	IONO2	REAL*8	24
RVCXE5	IONO2	REAL*8	24
RVCXE6	IONO2	REAL*8	24
RVCXV1	IONO3	REAL*8	8
RVCXV2	IONO3	REAL*8	8
RVCXL1	OTHER	REAL*8	8

PGF2 Local Symbols

Name	Class	Type	Size
RVCXLO1	OTHER	REAL*8	8
RVCXHBOT	OTHER	REAL*8	8
RVCXA5	IONO2	REAL*8	24
RVCXA6	IONO2	REAL*8	24
RTCXA	OTHER	REAL*8	8
RTCXB	OTHER	REAL*8	8
RTCXC	IONO3	REAL*8	8
IFCXN4	TEMP2	INTEGER*4	4
IVCXSCS	TEMP2	INTEGER*4	4
IVCXSCT1	TEMP2	INTEGER*4	4
IVCXSCT2	TEMP2	INTEGER*4	4
IVCXSCT3	TEMP2	INTEGER*4	4
ITCXA	IONO3	INTEGER*4	4
RVCXHB	IONO1	REAL*8	24
RVCXXX	IONO3	REAL*8	8
RVCXFB	IONO1	REAL*8	24
RVCXFREQ	OTHER	REAL*8	8
RVCXALPH	IONO1	REAL*8	8
RVCXBETA	IONO1	REAL*8	8
RVCXET1	IONO1	REAL*8	8
RVCXET2	IONO1	REAL*8	8

```
Line#  Source Line

 1683          SUBROUTINE PGFB(RVSxH)
 1684  C
 1685  C-----------------------------------------------------------------
 1686  C
 1687  C     PGFB -- SUBROUTINE to calculate the ionospheric parameters
 1688  C             for the Bent topside region.
 1689  C
 1690  C
 1691  C        CALLED BY: CASE1, CASE2, CASE3, CASE4, CASE5
 1692  C
 1693  C-----------------------------------------------------------------
 1694  C
 1695  C        AUTHOR:          MICHAEL H. REILLY & ERIC L. STROBEL
 1696  C
 1697  C        DATE:            09/01/87
 1698  C
 1699  C        VERSION:         1.0
 1700  C
 1701  C-----------------------------------------------------------------
 1702  C
 1703  C        REVISED:         09/01/87 -- V1.0.  Initial revision.
 1704  C
 1705  C-----------------------------------------------------------------
 1706  C
 1707  C        USES:            RVSxH        The current height.
 1708  C
 1709  C
 1710  C        To calculate the A, B, N1, and N2 parameters, as well as the
 1711  C          plasma frequency squared XX.  For more details, see the
 1712  C          Radio Science paper and the RADAR-C report referred to
 1713  C          in the documentation.
 1714  C
 1715  C
 1716  C        RETURNS:         RVCxALPH     A coef. in the index of
 1717  C                                       refraction.
 1718  C
 1719  C                         RVCxBETA     A coef. in the index of
 1720  C                                       refraction.
 1721  C
 1722  C                         RVCxET1      A coef. in the index of
 1723  C                                       refraction.
 1724  C
 1725  C                         RVCxET2      A coef. in the index of
 1726  C                                       refraction.
 1727  C
 1728  C                         RVCxXX       The plasma frequency squared.
 1729  C
 1730  C
 1731  C-----------------------------------------------------------------
 1732  C
```

```
Line#  Source Line

 1733           INTEGER IFCxN4, IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3
 1734           INTEGER ITCxA
 1735   C
 1736           REAL*8 RVLxALP, RVCxHB(3), RVLxA7, RVCxFB(3), RVSxH
 1737           REAL*8 RVCxHT5, RVCxXX, RVCxALPH, RVCxBETA, RVCxFREQ
 1738           REAL*8 RVLxXXX2, RVLxXXH2, RVLxXXY2, RVCxET1, RVCxET2
 1739           REAL*8 RVCxA5(3), RVCxA6(3), RVCxB5(3), RVCxB6(3)
 1740           REAL*8 RVCxE5(3), RVCxE6(3), RVCxV1, RVCxV2, RVCxL1, RVCxLO1
 1741           REAL*8 RVCxHBOT, RVCxA2, RVCxB2, RVCxC2, RVCxHT3, RVCxHT4
 1742           REAL*8 RTCxA, RTCxB, RTCxC
 1743   C
 1744           COMMON /OTHER/ RVCxL1,RVCxLO1,RVCxHBOT,RVCxFREQ,RTCxA,RTCxB
 1745           COMMON /IONO1/ RVCxALPH,RVCxBETA,RVCxET1,RVCxET2,RVCxFB,RVCxHB
 1746           COMMON /IONO2/ RVCxA5,RVCxA6,RVCxB5,RVCxB6,RVCxE5,RVCxE6
 1747           COMMON /IONO3/ RVCxV1, RVCxV2, RVCxXX, ITCxA, RTCxC
 1748           COMMON /TEMP2/ IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3, IFCxN4
 1749           COMMON /VAR4/ RVCxA2,RVCxB2,RVCxC2,RVCxHT3,RVCxHT4,RVCxHT5
 1750   C
 1751           RVLxALP = 1.0D00 / (1.875D00 * RVCxHB(3))
 1752           RVLxA7 = 15.0D00 * RVCxFB(3) * DEXP(RVLxALP*RVCxHT5)/16.0D00
 1753           RVCxXX = RVLxA7 * DEXP(-RVLxALP * RVSxH)
 1754           RVCxALPH = -RVLxALP * RVCxXX / RVCxFREQ
 1755           RVCxBETA = RVLxALP * RVCxALPH / 2.0D00
 1756           IF (IFCxN4.EQ.1) RETURN
 1757           RVLxXXX2 = RVCxXX / (RVCxFB(3) * RVCxFREQ)
 1758           RVLxXXH2 = -RVCxALPH
 1759           RVLxXXY2 = -RVCxALPH * (0.25D00 - 1.875D00*RVLxALP*
 1760         #(RVCxHT5-RVSxH))
 1761           RVCxET1 = RVLxXXX2*RVCxB5(3) + RVLxXXH2*RVCxB6(2)
 1762           RVCxET1 = -(RVCxET1 + RVLxXXY2*RVCxB6(3)) / RVCxV1
 1763           RVCxET2 = RVLxXXX2*RVCxE5(3) + RVLxXXH2*RVCxE6(2)
 1764           RVCxET2 = (RVCxET2 + RVLxXXY2*RVCxE6(3)) / RVCxV2
 1765           RETURN
 1766           END
```

PGFB Local Symbols

Name	Class	Type	Size
RVSXH	param		
RVLXA7	local	REAL*8	8
RVLXXXX2	local	REAL*8	8
RVLXXXY2	local	REAL*8	8
RVLXALP	local	REAL*8	8
RVLXXXH2	local	REAL*8	8
IFCXN4	TEMP2	INTEGER*4	4
IVCXSCS	TEMP2	INTEGER*4	4
IVCXSCT1	TEMP2	INTEGER*4	4
IVCXSCT2	TEMP2	INTEGER*4	4

PGFB Local Symbols

Name	Class	Type	Size
IVCXSCT3	TEMP2	INTEGER*4	4
ITCXA	IONO3	INTEGER*4	4
RVCXHB	IONO1	REAL*8	24
RVCXFB	IONO1	REAL*8	24
RVCXHT5	VAR4	REAL*8	8
RVCXXX	IONO3	REAL*8	8
RVCXALPH	IONO1	REAL*8	8
RVCXBETA	IONO1	REAL*8	8
RVCXFREQ	OTHER	REAL*8	8
RVCXET1	IONO1	REAL*8	8
RVCXET2	IONO1	REAL*8	8
RVCXA5	IONO2	REAL*8	24
RVCXA6	IONO2	REAL*8	24
RVCXB5	IONO2	REAL*8	24
RVCXB6	IONO2	REAL*8	24
RVCXE5	IONO2	REAL*8	24
RVCXE6	IONO2	REAL*8	24
RVCXV1	IONO3	REAL*8	8
RVCXV2	IONO3	REAL*8	8
RVCXL1	OTHER	REAL*8	8
RVCXLO1	OTHER	REAL*8	8
RVCXHBOT	OTHER	REAL*8	8
RVCXA2	VAR4	REAL*8	8
RVCXB2	VAR4	REAL*8	8
RVCXC2	VAR4	REAL*8	8
RVCXHT3	VAR4	REAL*8	8
RVCXHT4	VAR4	REAL*8	8
RTCXA	OTHER	REAL*8	8
RTCXB	OTHER	REAL*8	8
RTCXC	IONO3	REAL*8	8

Global Symbols

Name	Class	Type	Size
CASE1	FSUBRT	***	***
CASE2	FSUBRT	***	***
CASE3	FSUBRT	***	***
CASE4	FSUBRT	***	***
CASE5	FSUBRT	***	***
END	common	***	32
INBOX	extern	***	***
INTERP	FSUBRT	***	***
INTSIGN	extern	INTEGER*4	***

Global Symbols

Name	Class	Type	Size
IONO1	common	***	80
IONO2	common	***	144
IONO3	common	***	36
IONOPAR	FSUBRT	***	***
MAINDAT	common	***	48
MORE.	common	***	12
OTHER	common	***	48
PGF1L	FSUBRT	***	***
PGF1P	FSUBRT	***	***
PGF2.	FSUBRT	***	***
PGFB.	FSUBRT	***	***
PGVAL	FSUBRT	***	***
PRAM.	common	***	32
RAID.	common	***	4
SCPS1	common	***	28804
SCPS1A.	common	***	86400
START	common	***	40
TEMP1	common	***	32
TEMP2	common	***	20
TEMP3	common	***	32
TRIANG.	extern	***	***
VAR1.	common	***	40
VAR2.	common	***	48
VAR3.	common	***	40
VAR4.	common	***	48

Code size = 356f (13679)
Data size = 019f (415)

```
Line#  Source Line

     1          SUBROUTINE ENDPT(IFSxG, RVSxH0, RVSxHZ, RVSxS4, RVSxGPI)
     2  C
     3  C-----------------------------------------------------------------
     4  C
     5  C   ENDPT -- SUBROUTINE TO CALCULATE THE ENDPOINTS OF THE INTERVAL
     6  C
     7  C         CALLED BY:    RAYSUB
     8  C
     9  C         CALLS:        ACCFSP
    10  C
    11  C-----------------------------------------------------------------
    12  C
    13  C         AUTHOR:           MICHAEL H. REILLY
    14  C
    15  C         DATE:             07/25/86
    16  C
    17  C         VERSION:          1.1
    18  C
    19  C-----------------------------------------------------------------
    20  C
    21  C         REVISED:          07/25/86 -- INITIAL REVISION.  TRANSLATED
    22  C                           FROM TEKTRONIX BASIC TO VAX FORTRAN BY
    23  C                           ERIC L. STROBEL.
    24  C
    25  C                           07/30/86 -- V1.1.  Change over to use of
    26  C                           REAL*8 precision in the calculations.
    27  C
    28  C                           09/01/87 -- V2.0.  Changed to prevent
    29  C                           problems with calculations on small numbers.
    30  C
    31  C-----------------------------------------------------------------
    32  C
    33  C        USES:    IFSxG             A FLAG
    34  C                 RVSxH0            INITIAL HEIGHT
    35  C                 RVSxHZ            HEIGHT
    36  C                 RVCxH5            NEW HEIGHT
    37  C                 RVCxALPH          X
    38  C                 RVCxBETA          X
    39  C                 RVCxET1           X-IONOSPHERIC PARAMETERS
    40  C                 RVCxET2           X
    41  C                 RVCxHB            HEIGHT OF IONOSPHERE BOTTOM
    42  C                 RVCxCX,CY,CZ
    43  C                 IVCxSX,SY,SZ
    44  C
    45  C         TO COMPUTE THE S.E.Z. COORDINATES FOR THE END OF THE
    46  C                  RAYPATH INCREMENT. (See the Radio Science paper
    47  C                  referred to in the documentation.)
    48  C
    49  C         RETURNS:
    50  C                 IFSxG
```

```
Line#  Source Line

    51  C                RVSxGPI          GROUP PATH INCREMENT
    52  C                RVCxXF,YF,ZF     COORDINATES OF THE ENDPOINT
    53  C
    54  C-----------------------------------------------------------------
    55  C
    56            LOGICAL LTLxT1, LTLxT2, LTLxT3, LTLxT4
    57  C
    58            INTEGER IFSxG, IVCxSX, IVCxSY, IVCxSZ
    59  C
    60            REAL*8 RVCxCX, RVCxCY, RVCxCZ, RVCxLAT0, RVCxLON0
    61            REAL*8 RVCxHB, RVCxXF, RVCxYF, RVCxZF, RTLxTEMP
    62            REAL*8 RVCxH5, RVSxH0, RVSxHZ, RVSxGPI, RVCxALPH
    63            REAL*8 RVCxBETA, RVCxET1, RVCxET2, RVLxEPS
    64            REAL*8 RTLxF3, RTLxR2, RTLxF6, RTLxF4, RTLxB1
    65            REAL*8 RTLxA2, RTLxF5, RTLxG3, RTLxC4, RTLxC5, RTLxF1
    66            REAL*8 RTLxG1, RTLxC6, RTLxC7, RTLxF2, RTLxG2
    67            REAL*8 RPCxPI, RPCxDTR, RPCxRE, RPCxHTP, RVSxS4
    68  C
    69            COMMON /OTHER/ RVCxLAT0,RVCxLON0,RVCxHB
    70            COMMON /END/ RVCxXF, RVCxYF, RVCxZF, RVCxH5
    71            COMMON /IONO1/ RVCxALPH,RVCxBETA,RVCxET1,RVCxET2
    72            COMMON /MORE/ IVCxSX, IVCxSY, IVCxSZ, RVCxCX, RVCxCY, RVCxCZ
    73            COMMON /PRAM/ RPCxPI, RPCxDTR, RPCxRE, RPCxHTP
    74            COMMON /TEMP4/ RTLxF1, RTLxF2, RTLxF3
    75  C
    76  C         Initializations.
    77  C
    78            RTLxA2 = 0.0
    79            RTLxB1 = 0.0
    80            RTLxC4 = 0.0
    81            RTLxC5 = 0.0
    82            RTLxC6 = 0.0
    83            RTLxC7 = 0.0
    84            RTLxF4 = 0.0
    85            RTLxF5 = 0.0
    86            RTLxF6 = 0.0
    87            RTLxG1 = 0.0
    88            RTLxG2 = 0.0
    89            RTLxG3 = 0.0
    90            RTLxR2 = 0.0
    91            RVCxXF = 0.0D00
    92            RVCxYF = 0.0D00
    93            RVSxGPI = 0.0D00
    94            RVLxEPS = 1.0D-09
    95  C
    96  C         Check for being in free space.
    97  C
    98            IF (RVCxH5.GE.(RVCxHB - 0.02D00).AND.RVCxH5.LE.
    99           #(RPCxHTP + 0.02D00)) GO TO 10262
   100            IF (IVCxSZ.GE.0) THEN
```

```
Line#  Source Line

  101              IFSxG = 7
  102            ELSE
  103              IFSxG = 6
  104            ENDIF
  105  C
  106  C         ACCFSP is the routine that does the free space propagation.
  107  C
  108            CALL ACCFSP(RVSxH0,RVSxS4,IFSxG)
  109            GO TO 10267
  110  10262     RVCxZF = RVSxHZ - RVSxH0
  111  C
  112  C         Try to prevent loss of precision in the quantity
  113  C            (sqrt(x) - sqrt(x-y)) with y small.  First, define
  114  C            'small', then round the offending terms to zero.
  115  C
  116  10263     IF (RVCxBETA.NE.0.0D00) THEN
  117                LTLxT1 = DABS(RVCxBETA*RVCxZF*RVCxZF).LT.(RVLxEPS*
  118       #RVCxCZ)
  119                LTLxT2 = DABS(RVCxALPH*RVCxZF).LT.(RVLxEPS*RVCxCZ)
  120                IF (LTLxT1.AND.LTLxT2) THEN
  121                  RVCxALPH = 0.0D00
  122                  RVCxBETA = 0.0D00
  123                  GO TO 10267
  124                ENDIF
  125                IF (IFSxG.EQ.3) GO TO 10282
  126                GO TO 10280
  127            ENDIF
  128  10265     IF (RVCxALPH.NE.0.0D00) THEN
  129                LTLxT2 = DABS(RVCxALPH*RVCxZF).LT.(RVLxEPS*RVCxCZ)
  130                IF (LTLxT2) THEN
  131                  RVCxALPH = 0.0
  132                ELSE
  133                  IF (IFSxG.EQ.3) GO TO 10272
  134                  GO TO 10269
  135                ENDIF
  136            ENDIF
  137  C
  138  C         From here on, the code follows the logic outlined in the
  139  C            Radio Science paper referred to in the documentation.
  140  C
  141  10267     RTLxF3 = RVCxSZ*RVCxZF / DSQRT(RVCxCZ)
  142            GO TO 10293
  143  10269     RTLxR2 = RVCxCZ - RVCxALPH*RVCxZF
  144            IF (RTLxR2.GT.0.0D00) THEN
  145              RTLxF3 = DSQRT(RTLxR2)
  146            ELSE
  147              IFSxG = 3
  148              RVCxZF = RVCxCZ / RVCxALPH
  149              RTLxF3 = 0.0D00
  150              GO TO 10265
```

```
Line#  Source Line

  151           ENDIF
  152  10272    RTLxF6 = DSQRT(RVCxCZ)
  153           RTLxF3 = RTLxF6 - RTLxF3
  154           RTLxF3 = 2.0D00*IVCxSZ*RTLxF3 / RVCxALPH
  155           GO TO 10293
  156  10280    RTLxR2 = RVCxBETA*RVCxZF*RVCxZF - RVCxALPH*RVCxZF - RVCxCZ
  157           IF (RTLxR2.GT.0.0D00) THEN
  158               RTLxF4 = DSQRT(RTLxR2)
  159           ELSE
  160               IFSxG = 3
  161               RVCxZF = (RVCxALPH - IVCxSZ*DSQRT(RVCxALPH
  162       #*RVCxALPH - 4.0D00*RVCxBETA*RVCxCZ))/(2.0D00*RVCxBETA)
  163               RTLxF4 = 0.0D00
  164               GO TO 10263
  165           ENDIF
  166  10282    RTLxB1 = DSQRT(ABS(RVCxBETA))
  167           IF (RVCxBETA.LT.0.0) THEN
  168               RTLxF6 = SQRT(RVCxALPH*RVCxALPH - 4.0*RVCxBETA*RVCxCZ)
  169               IF (IFSxG.EQ.3) THEN
  170                   RTLxF4 = INTSIGN(-RVCxALPH)
  171               ELSE
  172                   RTLxF4 = (2.0*RVCxBETA*RVCxZF - RVCxALPH)/RTLxF6
  173               ENDIF
  174               IF (DABS(-RVCxALPH/RTLxF6).GT.1.0) THEN
  175                   RTLxTEMP = DASIN(DFLOAT(INTSIGN(-RVCxALPH/RTLxF6)))
  176               ELSE
  177                   RTLxTEMP = DASIN(-RVCxALPH/RTLxF6)
  178               ENDIF
  179               RTLxF3 = RTLxTEMP - DASIN(RTLxF4)
  180           ELSE
  181               RTLxA2 = RVCxALPH / (2.0D00*RTLxB1)
  182               RTLxF5 = DSQRT(RVCxCZ)
  183               RTLxF6 = -RTLxA2 + RTLxF5
  184               RTLxG3 = (RTLxB1*RVCxZF - RTLxA2 + RTLxF4) / RTLxF6
  185               RTLxF3 = DLOG(RTLxG3)
  186           ENDIF
  187           RTLxF3 = IVCxSZ * RTLxF3 / RTLxB1
  188  10293    RTLxC4 = DSQRT(RVCxCX)
  189           RVSxGPI = RTLxF3
  190           IF (IFSxG.GT.5) GO TO 10352
  191           IF (RVCxET1.EQ.0.0D00) THEN
  192               GO TO 10320
  193           ELSE
  194               LTLxT3 = (DSQRT(RVCxCX)*DABS(RVCxET1*RVCxZF))
  195       #.LT.(RVLxEPS*RVCxCX*DSQRT(RVCxCZ))
  196               IF (LTLxT3) THEN
  197                   RVCxET1 = 0.0D00
  198                   GO TO 10320
  199               ENDIF
  200           ENDIF
```

```
Line#  Source Line

  201           IF (IVCxSX.EQ.0) IVCxSX = -INTSIGN(RVCxET1)
  202           RTLxC5 = RTLxC4 - RVCxET1*RTLxF3/(2.0D00*IVCxSX)
  203           IF (RTLxC5.GT.0.0D00) GO TO 10320
  204           RVCxXF = RVCxCX / RVCxET1
  205           IF (DABS(RVCxET1*RVCxXF).LT.(RVLxEPS*RVCxCX)) THEN
  206               RVCxET1 = 0.0D00
  207               GO TO 10320
  208           ENDIF
  209           IFSxG = 4
  210           RTLxF1 = 2.0D00*RTLxC4*IVCxSX / RVCxET1
  211           RVSxGPI = RTLxF1
  212           IF (RVCxBETA.EQ.0.0D00) THEN
  213               IF (RVCxALPH.EQ.0.0D00) THEN
  214                   RVCxZF = RTLxF1*DSQRT(RVCxCZ)/IVCxSZ
  215               ELSE
  216                   RVCxZF = (RVCxCZ - (RTLxF6 - RVCxALPH*RTLxF1/
  217        *(2.0D00*IVCxSZ))**2.0D00) / RVCxALPH
  218               ENDIF
  219           ELSE
  220               IF (RVCxBETA.LT.0.0) THEN
  221                   IF (DABS(-RVCxALPH/RTLxF6).GT.1.0) THEN
  222                      RTLxTEMP = DASIN(DFLOAT(INTSIGN(-RVCxALPH/RTLxF6)
  223                   ELSE
  224                      RTLxTEMP = DASIN(-RVCxALPH/RTLxF6)
  225                   ENDIF
  226                   RTLxF4 = RTLxTEMP - RTLxB1*RTLxF1*IVCxSZ
  227                   RTLxF4 = DSIN(RTLxF4)
  228                   RVCxZF = (RVCxALPH + RTLxF6*RTLxF4)/(2.0*RVCxBETA
  229               ELSE
  230                   RTLxG1 = RTLxF6*DEXP(RTLxB1*RTLxF1/IVCxSZ) + RTLxA1
  231                   RVCxZF = (RTLxG1*RTLxG1 - RVCxCZ)/(2.0D00*RTLxB1*RTLxG
  232        *- RVCxALPH)
  233               ENDIF
  234           ENDIF
  235           IF (RVCxET2.EQ.0.0D00) THEN
  236               GO TO 10318
  237           ELSE
  238               LTLxT4 = (DSQRT(RVCxCY)*DABS(RVCxET2*RVCxZF)
  239        #.LT.(RVLxEPS*RVCxCY*DSQRT(RVCxCZ))
  240               IF (LTLxT4) THEN
  241                   RVCxET2 = 0.0D00
  242                   GO TO 10318
  243               ENDIF
  244           ENDIF
  245           RTLxC6 = DSQRT(RVCxCY)
  246           RTLxC7 = RTLxC6 - RVCxET2*RTLxF1/(2.0D00*IVCxSY)
  247           IF (RTLxC7.LE.0.0D00) GO TO 10320
  248           RVCxYF = (RVCxCY - RTLxC7*RTLxC7)/RVCxET2
  249           GO TO 10341
  250   10318   RVCxYF = RTLxF1 * DSQRT(RVCxCY) / IVCxSY
```

```
Line#  Source Line

  251           GO TO 10341
  252  10320    RTLxC6 = DSQRT(RVCxCY)
  253           IF (RVCxET2.EQ.0.0D00) THEN
  254               GO TO 10341
  255           ELSE
  256               LTLxT4 = (DSQRT(RVCxCY)*DABS(RVCxET2*RVCxZF))
  257       #.LT.(RVLxEPS*RVCxCY*DSQRT(RVCxCZ))
  258               IF (LTLxT4) THEN
  259                   RVCxET2 = 0.0D00
  260                   GO TO 10341
  261               ENDIF
  262           ENDIF
  263           IF (IVCxSY.EQ.0) IVCxSY = INTSIGN(-RVCxET2)
  264           RTLxC7 = RTLxC6 - RVCxET2*RTLxF3/(2.0D00*IVCxSY)
  265           IF (RTLxC7.GT.0.0D00) GO TO 10341
  266           RVCxYF = RVCxCY / RVCxET2
  267           IF (DABS(RVCxET2*RVCxYF).LT.(RVLxEPS*RVCxCY)) THEN
  268               RVCxET2 = 0.0D00
  269               GO TO 10341
  270           ENDIF
  271           IFSxG = 5
  272           RTLxF2 = 2.0D00 * RTLxC6 * IVCxSY / RVCxET2
  273           RVSxGPI = RTLxF2
  274           IF (RVCxBETA.EQ.0.0D00) THEN
  275               IF (RVCxALPH.EQ.0.0D00) THEN
  276                   RVCxZF = RTLxF2 * DSQRT(RVCxCZ) / IVCxSZ
  277               ELSE
  278                   RVCxZF = (RVCxCZ - (RTLxF6 - RVCxALPH*RTLxF2.(
  279       *2.0D00*IVCxSZ))**2.0D00) / RVCxALPH
  280               ENDIF
  281           ELSE
  282               IF (RVCxBETA.LT.0.0) THEN
  283                   IF (DABS(-RVCxALPH/RTLxF6).GT.1.0) THEN
  284                      RTLxTEMP = DASIN(DFLOAT(INTSIGN(-RVCxALPH/RTLxF6.
  285                   ELSE
  286                      RTLxTEMP = DASIN(-RVCxALPH/RTLxF6)
  287                   ENDIF
  288                   RTLxF4 = RTLxTEMP - RTLxB1*RTLxF2*IVCxSZ
  289                   RTLxF4 = DSIN(RTLxF4)
  290                   RVCxZF = (RVCxALPH + RTLxF6*RTLxF4)/(2.0*RVCxBETA.
  291               ELSE
  292                   RTLxG2 = RTLxF6*DEXP(RTLxB1*RTLxF2/IVCxSZ) + RTLxA2
  293                   RVCxZF = (RTLxG2*RTLxG2 - RVCxCZ)/(2.0D00*RTLxB1*RTLxG2
  294       * - RVCxALPH)
  295               ENDIF
  296           ENDIF
  297           IF (RVCxET1.EQ.0.0D00) THEN
  298               RVCxXF = RTLxF2 * DSQRT(RVCxCX) / IVCxSX
  299           ELSE
  300               RTLxC5 = RTLxC4 - RVCxET1*RTLxF2/(2.0D00*IVCxSX)
```

```
Line#  Source Line

  301                RVCxXF = (RVCxCX - RTLxC5*RTLxC5)/RVCxET1
  302              ENDIF
  303  10341      IF (IFSxG.GT.3) RETURN
  304              IF (RVCxET1.NE.0.0D00) THEN
  305                  RVCxXF = (RVCxCX - RTLxC5*RTLxC5)/RVCxET1
  306              ELSE IF (RVCxCX.NE.0.0D00) THEN
  307                  RVCxXF = RTLxC4 * IVCxSX * RTLxF3
  308              ENDIF
  309              IF (RVCxET2.NE.0.0D00) THEN
  310                  RVCxYF = (RVCxCY - RTLxC7*RTLxC7)/RVCxET2
  311                  RETURN
  312              ENDIF
  313              IF (RVCxCY.EQ.0.0D00) RETURN
  314              RVCxYF = RTLxC6 * IVCxSY * RTLxF3
  315  10352      RETURN
  316              END


ENDPT  Local Symbols

Name                          Class    Type                 Size

RVSXGPI . . . . . . . . . . . param
RVSXS4. . . . . . . . . . . . param
RVSXHZ. . . . . . . . . . . . param
RVSXH0. . . . . . . . . . . . param
IFSXG . . . . . . . . . . . . param
RTLXTEMP. . . . . . . . . . . local    REAL*8                  8
RTLXC4. . . . . . . . . . . . local    REAL*8                  8
RTLXG1. . . . . . . . . . . . local    REAL*8                  8
RTLXC5. . . . . . . . . . . . local    REAL*8                  8
RTLXG2. . . . . . . . . . . . local    REAL*8                  8
RTLXC6. . . . . . . . . . . . local    REAL*8                  8
RTLXC7. . . . . . . . . . . . local    REAL*8                  8
RTLXG3. . . . . . . . . . . . local    REAL*8                  8
RTLXF4. . . . . . . . . . . . local    REAL*8                  8
RTLXF5. . . . . . . . . . . . local    REAL*8                  8
RTLXF6. . . . . . . . . . . . local    REAL*8                  8
LTLXT1. . . . . . . . . . . . local    LOGICAL*4               4
LTLXT2. . . . . . . . . . . . local    LOGICAL*4               4
LTLXT3. . . . . . . . . . . . local    LOGICAL*4               4
LTLXT4. . . . . . . . . . . . local    LOGICAL*4               4
RTLXR2. . . . . . . . . . . . local    REAL*8                  8
RVLXEPS . . . . . . . . . . . local    REAL*8                  8
RTLXA2. . . . . . . . . . . . local    REAL*8                  8
RTLXB1. . . . . . . . . . . . local    REAL*8                  8
IVCXSX. . . . . . . . . . . . MORE     INTEGER*4               4
IVCXSY. . . . . . . . . . . . MORE     INTEGER*4               4
IVCXSZ. . . . . . . . . . . . MORE     INTEGER*4               4
RVCXCX. . . . . . . . . . . . MORE     REAL*8                  8
```

ENDPT Local Symbols

Name	Class	Type	Size
RVCXCY.	MORE	REAL*8	8
RVCXCZ.	MORE	REAL*8	8
RVCXLAT0.	OTHER	REAL*8	8
RVCXLON0.	OTHER	REAL*8	8
RVCXHB.	OTHER	REAL*8	8
RVCXXF.	END	REAL*8	8
RVCXYF.	END	REAL*8	8
RVCXZF.	END	REAL*8	8
RVCXH5.	END	REAL*8	8
RVCXALPH.	IONO1	REAL*8	8
RVCXBETA.	IONO1	REAL*8	8
RVCXET1	IONO1	REAL*8	8
RVCXET2	IONO1	REAL*8	8
RTLXF3.	TEMP4	REAL*8	8
RTLXF1.	TEMP4	REAL*8	8
RTLXF2.	TEMP4	REAL*8	8
RPCXPI.	PRAM	REAL*8	8
RPCXDTR	PRAM	REAL*8	8
RPCXRE.	PRAM	REAL*8	8
RPCXHTP	PRAM	REAL*8	8

```
Line#  Source Line

  318          SUBROUTINE PHSPL(IFSxG, RVSxPPI)
  319  C
  320  C-----------------------------------------------------------------
  321  C
  322  C       PHSPL  -- SUBROUTINE TO CALCULATE THE PHASE PATH LENGTH
  323  C                 INCREMENT FOR THE CURRENT RAY PATH INCREMENT.
  324  C
  325  C
  326  C       CALLED BY:       RAYSUB
  327  C
  328  C       CALLS:           PHSPL2, PHSPL3
  329  C
  330  C-----------------------------------------------------------------
  331  C
  332  C       AUTHOR:          MICHAEL H. REILLY & ERIC L. STROBEL
  333  C
  334  C       DATE:            09/30/87
  335  C
  336  C       VERSION:         1.0
  337  C
  338  C-----------------------------------------------------------------
  339  C
  340  C       REVISED:         09/30/87 -- V1.0.  Initial revision.
  341  C
  342  C-----------------------------------------------------------------
  343  C
  344  C       USES:            RVCxALPH \
  345  C                        RVCxBETA  |____  To decide on which PPI calc.
  346  C                        RVCxET1   |        to use.
  347  C                        RVCxET2  /
  348  C
  349  C                        IVCxSX,Y,Z  \
  350  C                        RVCxCX,Y,Z  |-- To calc. PPI in regions of
  351  C                        RVCxX,Y,ZF  /     constant plasma frequency.
  352  C
  353  C                        IFSxG           A flag variable.
  354  C
  355  C       To either pass off the phase path increment length calculation
  356  C         to PHSPL2,3 or to do it locally for the simplest case.  The
  357  C         phase path calculations are done by considering the (mu)*ds
  358  C         integral, where (mu) is the index of refraction.  (See the
  359  C         Radio Science paper referred to in the documentation, eq. ).
  360  C
  361  C
  362  C       RETURNS:         RVSxPPI         The increment of phase path
  363  C                                          length.
  364  C
  365  C
  366  C-----------------------------------------------------------------
  367  C
```

```
Line#  Source Line

  368  C
  369            INTEGER IFSxG, IVCxSX, IVCxSY, IVCxSZ
  370  C
  371            REAL*8 RVSxPPI, RVCxF1, RVCxF2, RVCxF3, RVCxBETA, RVCxXF
  372            REAL*8 RVCxYF, RVCxZF, RVCxCX, RVCxCY, RVCxCZ, RVCxALPH
  373            REAL*8 RVSxPPIX, RVSxPPIY, RVSxPPIZ, RVCxET1, RVCxET2
  374            REAL*8 RTCxA
  375  C
  376            COMMON /END/ RVCxXF, RVCxYF, RVCxZF, RTCxA
  377            COMMON /MORE/ IVCxSX,IVCxSY,IVCxSZ,RVCxCX,RVCxCY,RVCxCZ
  378            COMMON /IONO1/ RVCxALPH,RVCxBETA,RVCxET1,RVCxET2
  379            COMMON /TEMP4/ RVCxF1, RVCxF2, RVCxF3
  380  C
  381  C         IFSxG = 4  to flag an x-reflection & = 5 to flag a
  382  C           y-reflection.
  383  C
  384            IF (IFSxG.EQ.4) RVCxF3 = RVCxF1
  385            IF (IFSxG.EQ.5) RVCxF3 = RVCxF2
  386            IF (RVCxBETA.NE.0.0D00) THEN
  387  C
  388  C         Call PHSPL3 to calculate the Z phase path increment when
  389  C            beta is not equal to zero.
  390  C
  391                CALL PHSPL3(RVCxZF,RVCxF3,IVCxSZ,RVCxCZ,RVCxALPH,RVCxBETA
  392       #,IFSxG,RVSxPPIZ)
  393            ELSE IF (RVCxALPH.EQ.0.0D00) THEN
  394                RVSxPPIZ = IVCxSZ * RVCxZF * DSQRT(RVCxCZ)
  395            ELSE
  396  C
  397  C         Call PHSPL2 to calculate phase path increment for a direction
  398  C            in which the plasma frequency has a linear dependence.
  399  C
  400                CALL PHSPL2(RVCxZF,IVCxSZ,RVCxCZ,RVCxALPH,RVSxPPIZ)
  401            ENDIF
  402            IF (RVCxET1.EQ.0.0D00) THEN
  403                RVSxPPIX = IVCxSX * RVCxXF * DSQRT(RVCxCX)
  404            ELSE
  405                CALL PHSPL2(RVCxXF,IVCxSX,RVCxCX,RVCxET1,RVSxPPIX)
  406            ENDIF
  407            IF (RVCxET2.EQ.0.0D00) THEN
  408                RVSxPPIY = IVCxSY * RVCxYF * DSQRT(RVCxCY)
  409            ELSE
  410                CALL PHSPL2(RVCxYF,IVCxSY,RVCxCY,RVCxET2,RVSxPPIY)
  411            ENDIF
  412            RVSxPPI = RVSxPPIX + RVSxPPIY + RVSxPPIZ
  413            RETURN
  414            END
```

PHSPL Local Symbols

Name	Class	Type	Size
RVSXPPI	param		
IFSXG	param		
RVSXPPIX	local	REAL*8	8
RVSXPPIY	local	REAL*8	8
RVSXPPIZ	local	REAL*8	8
RVCXCY	MORE	REAL*8	8
RVCXCZ	MORE	REAL*8	8
RVCXALPH	IONO1	REAL*8	8
RVCXET1	IONO1	REAL*8	8
RVCXET2	IONO1	REAL*8	8
RTCXA	END	REAL*8	8
IVCXSX	MORE	INTEGER*4	4
IVCXSY	MORE	INTEGER*4	4
IVCXSZ	MORE	INTEGER*4	4
RVCXF1	TEMP4	REAL*8	8
RVCXF2	TEMP4	REAL*8	8
RVCXF3	TEMP4	REAL*8	8
RVCXBETA	IONO1	REAL*8	8
RVCXXF	END	REAL*8	8
RVCXYF	END	REAL*8	8
RVCXZF	END	REAL*8	8
RVCXCX	MORE	REAL*8	8

```
Line#  Source Line

  417           SUBROUTINE PHSPL3(RVSxZ5,RVSxF3,IVSxSZ,RVSxCZ,RVSxALPH
  418         #,RVSxBETA,IFSxG,RVSxPPIZ)
  419   C
  420   C-----------------------------------------------------------------
  421   C
  422   C       PHSPL3  -- SUBROUTINE TO CALCULATE THE Z INCREMENT OF
  423   C                    PHASE PATH LENGTH FOR BETA EQUALS ZERO.
  424   C
  425   C
  426   C       CALLED BY:      PHSPL
  427   C
  428   C-----------------------------------------------------------------
  429   C
  430   C       AUTHOR:         MICHAEL H. REILLY & ERIC L. STROBEL
  431   C
  432   C       DATE:           09/30/87
  433   C
  434   C       VERSION:        1.0
  435   C
  436   C-----------------------------------------------------------------
  437   C
  438   C       REVISED:        09/30/87 -- V1.0.  Initial revision.
  439   C
  440   C-----------------------------------------------------------------
  441   C
  442   C       USES:           RVSxZ5          SEZ z value at end of RPI.
  443   C
  444   C                       RVSxF3          Complete group path length
  445   C                                          increment.
  446   C
  447   C                       IVSxSZ    \
  448   C                       RVSxCZ    |___  Propagation parameters.
  449   C                       RVSxALPH  |
  450   C                       RVSxBETA  /
  451   C
  452   C                       IFSxG           A flag variable.
  453   C
  454   C
  455   C       To calculate the increment of phase path length in the
  456   C         z-direction when the z dependence of the plasma freq.
  457   C         is quadratic (i.e. beta.NE.0).
  458   C
  459   C
  460   C       RETURNS:        RVSxPPIZ        The z increment of phase
  461   C                                          path length.
  462   C
  463   C-----------------------------------------------------------------
  464   C
  465           INTEGER IVSxSZ,IFSxG
  466   C
```

```
Line#  Source Line

   467           REAL*8 RVSxZ5, RVSxF3, RVSxCZ, RVSxALPH, RVSxBETA
   468           REAL*8 RVSxPPIZ, RVLxR2, RVLxDEL
   469   C
   470           IF (IFSxG.EQ.3) THEN
   471               RVLxR2 = 0.0D00
   472           ELSE
   473               RVLxR2 = DSQRT(RVSxCZ - RVSxALPH*RVSxZ5 + RVSxBETA*
   474         #RVSxZ5*RVSxZ5)
   475           ENDIF
   476           RVLxDEL = 4.0D00 * RVSxBETA * RVSxCZ - RVSxALPH * RVSxALPH
   477           RVSxPPIZ = (2.0D00 * RVSxBETA * RVSxZ5 - RVSxALPH) * RVLxR2
   478           RVSxPPIZ = RVSxPPIZ + RVSxALPH * DSQRT(RVSxCZ)
   479           RVSxPPIZ = (IVSxSZ * RVSxPPIZ + RVLxDEL * RVSxF3/2.0D00)
   480           RVSxPPIZ = RVSxPPIZ / (4.0D00 * RVSxBETA)
   481           RETURN
   482           END


PHSPL3  Local Symbols

Name                             Class    Type              Size

RVSXPPIZ. . . . . . . . . .      param
IFSXG . . . . . . . . . . .      param
RVSXBETA. . . . . . . . . .      param
RVSXALPH. . . . . . . . . .      param
RVSXCZ. . . . . . . . . . .      param
IVSXSZ. . . . . . . . . . .      param
RVSXF3. . . . . . . . . . .      param
RVSXZ5. . . . . . . . . . .      param
RVLXR2. . . . . . . . . . .      local    REAL*8               8
RVLXDEL . . . . . . . . . .      local    REAL*8               8
```

```
Line#  Source Line

  485          SUBROUTINE PHSPL2(RVSxW, IVSxS, RVSxC, RVSxN, RVSxPPIW)
  486  C
  487  C-----------------------------------------------------------------
  488  C
  489  C       PHSPL2  -- SUBROUTINE TO CALCULATE THE PHASE PATH LENGTH
  490  C                  INCREMENT FOR LINEAR PLASMA FREQUENCY PATHS.
  491  C
  492  C
  493  C       CALLED BY:      PHSPL
  494  C
  495  C-----------------------------------------------------------------
  496  C
  497  C       AUTHOR:         MICHAEL H. REILLY & ERIC L. STROBEL
  498  C
  499  C       DATE:           09/30/87
  500  C
  501  C       VERSION:        1.0
  502  C
  503  C-----------------------------------------------------------------
  504  C
  505  C       REVISED:        09/30/87 -- V1.0.  Initial revision.
  506  C
  507  C-----------------------------------------------------------------
  508  C
  509  C       USES:           RVSxW       The generalized SEZ coord.
  510  C                                  value at the end of the RPI.
  511  C
  512  C                       IVSxS   \
  513  C                       RVSxC    |--  The generalized propagation
  514  C                       RVSxN   /    parameters.
  515  C
  516  C
  517  C       To calculate the phase path length increment for the
  518  C         particular coordinate direction of interest.  The PPI
  519  C         is calculated for a plasma frequency that is linearly
  520  C         dependent on the coordinate involved.  This translates
  521  C         to having N = Nx or Ny, or if B = 0, N = A.  The same
  522  C         routine can then be used in any of the coordinate
  523  C         directions.
  524  C
  525  C
  526  C       RETURNS:        RVSxPPIW    The phase path increment for
  527  C                                  the particular coordinate dir
  528  C
  529  C-----------------------------------------------------------------
  530  C
  531          INTEGER IVSxS
  532  C
  533          REAL*8 RVSxW, RVSxC, RVSxN, RVSxPPIW, RVLxR, RVLxR2
  534  C
```

```
Line#  Source Line

   535          RVLxR = RVSxC - RVSxN * RVSxW
   536          IF (RVLxR.GT.0.0D00) THEN
   537              RVLxR2 = -RVLxR * DSQRT(RVLxR)
   538          ELSE
   539              RVLxR2 = 0.0D00
   540          ENDIF
   541          RVSxPPIW = 2.0D00 * IVSxS * (RVLxR2 + RVSxC * DSQRT(RVSxC))
   542          RVSxPPIW = RVSxPPIW / (3.0D00 * RVSxN)
   543          RETURN
   544          END


PHSPL2  Local Symbols

Name                            Class    Type                Size

RVSXPPIW. . . . . . . . . . . . param
RVSXN . . . . . . . . . . . . . param
RVSXC . . . . . . . . . . . . . param
IVSXS . . . . . . . . . . . . . param
RVSXW . . . . . . . . . . . . . param
RVLXR2. . . . . . . . . . . . . local    REAL*8                 8
RVLXR . . . . . . . . . . . . . local    REAL*8                 8


Global Symbols

Name                            Class    Type                Size

ACCFSP. . . . . . . . . . . . . extern   ***                 ***
END . . . . . . . . . . . . . . common   ***                  32
ENDPT . . . . . . . . . . . . . FSUBRT   ***                 ***
INTSIGN . . . . . . . . . . . . extern   INTEGER*4            ***
IONO1 . . . . . . . . . . . . . common   ***                  32
MORE. . . . . . . . . . . . . . common   ***                  36
OTHER . . . . . . . . . . . . . common   ***                  24
PHSPL . . . . . . . . . . . . . FSUBRT   ***                 ***
PHSPL2. . . . . . . . . . . . . FSUBRT   ***                 ***
PHSPL3. . . . . . . . . . . . . FSUBRT   ***                 ***
PRAM. . . . . . . . . . . . . . common   ***                  32
TEMP4 . . . . . . . . . . . . . common   ***                  24

Code size = 155e (5470)
Data size = 0044 (68)
```

```
Line#  Source Line

     1          SUBROUTINE TILTS(RVSxDXHB, RVSxDYHB, IVSxJ)
     2  C
     3  C-----------------------------------------------------------------
     4  C
     5  C    TILTS -- SUBROUTINE TO CALCULATE THE IONOSPHERIC BOUNDARY TILTS
     6  C
     7  C          CALLED BY: RAYSUB
     8  C
     9  C-----------------------------------------------------------------
    10  C
    11  C          AUTHOR:          MICHAEL H. REILLY & ERIC L. STROBEL
    12  C
    13  C          DATE:            09/30/87
    14  C
    15  C          VERSION:         2.0
    16  C
    17  C-----------------------------------------------------------------
    18  C
    19  C          REVISED:         07/25/86 -- INITIAL REVISION.  TRANSLATED
    20  C                           FROM TEKTRONIX BASIC TO VAX FORTRAN BY
    21  C                           ERIC L. STROBEL.
    22  C
    23  C                           07/30/86 -- V1.1.  Change over to use of
    24  C                           REAL*8 precision in the calculations.
    25  C
    26  C                           09/30/87 -- V2.0.  Extensively modified
    27  C                           to accommodate the use of RADAR-C.  This
    28  C                           routine now serves as a dispatcher for
    29  C                           the real processing.
    30  C
    31  C-----------------------------------------------------------------
    32  C
    33  C         USES:             IVSxJ          A boundary index.
    34  C
    35  C                           IVCxCASE       Describes the compo. of
    36  C                                         profiles used.
    37  C
    38  C                           RVCxHBND       Height of the upcoming
    39  C                                         boundary.
    40  C
    41  C                           IFCxSN         A flag parameter needed in
    42  C                                         one of the cases.
    43  C
    44  C          To decide which of the tilt calculation routines is to be
    45  C            used, if at all.  The J-index values have different
    46  C            meanings depending on which CASE applies.
    47  C
    48  C          RETURNS:         RVSxDXHB         (d/dx) Boundary Height
    49  C                           RVSxDYHB         (d/dy) Boundary Height
    50  C
```

```
Line#  Source Line

    51  C-----------------------------------------------------------------
    52  C
    53          INTEGER IVSxJ, IVCxCASE, IFCxSN
    54  C
    55          REAL*8 RVSxDXHB, RVSxDYHB, RPCxPI, RPCxDR, RPCxRE
    56          REAL*8 RVCxV1, RVCxV2, RPCxHTP, RVCxXX, RVCxHBND
    57  C
    58          COMMON /PRAM/ RPCxPI, RPCxDR, RPCxRE, RPCxHTP
    59          COMMON /IONO3/ RVCxV1, RVCxV2, RVCxXX, IVCxCASE, RVCxHBND
    60          COMMON /RAID/ IFCxSN
    61  C
    62          RVSxDXHB = 0.0D00
    63          RVSxDYHB = 0.0D00
    64          IF (IVSxJ.LT.5) RETURN
    65          IF (RVCxHBND.GE.RPCxHTP) RETURN
    66  C
    67  C       Begin calling the routines for the boundary tilt calculation.
    68  C         based upon the CASE and J values.
    69  C
    70          IF (IVCxCASE.EQ.1) THEN
    71              IF (IVSxJ.EQ.5) CALL HUTLT(RVSxDXHB, RVSxDYHB)
    72              IF (IVSxJ.EQ.6) CALL H5TLT(RVSxDXHB, RVSxDYHB)
    73          ELSE IF (IVCxCASE.EQ.2) THEN
    74              IF (IVSxJ.EQ.5) CALL H1PTLT(1,RVSxDXHB, RVSxDYHB)
    75              IF (IVSxJ.EQ.6) CALL H4TLT(IFCxSN, RVSxDXHB, RVSxDYHB)
    76              IF (IVSxJ.EQ.7) CALL H5TLT(RVSxDXHB, RVSxDYHB)
    77          ELSE IF (IVCxCASE.EQ.3) THEN
    78              IF (IVSxJ.EQ.5) CALL H1LTLT(RVSxDXHB, RVSxDYHB)
    79              IF (IVSxJ.EQ.6) CALL H2TLT(RVSxDXHB, RVSxDYHB)
    80              IF (IVSxJ.EQ.7) CALL H5TLT(RVSxDXHB, RVSxDYHB)
    81          ELSE IF (IVCxCASE.EQ.4) THEN
    82              IF (IVSxJ.EQ.5) CALL HUTLT(RVSxDXHB, RVSxDYHB)
    83              IF (IVSxJ.EQ.6) CALL H4TLT(-1, RVSxDXHB, RVSxDYHB)
    84              IF (IVSxJ.EQ.7) CALL H4TLT(1, RVSxDXHB, RVSxDYHB)
    85              IF (IVSxJ.EQ.8) CALL H5TLT(RVSxDXHB, RVSxDYHB)
    86          ELSE IF (IVCxCASE.EQ.5) THEN
    87              IF (IVSxJ.EQ.5) CALL H1PTLT(1,RVSxDXHB,RVSxDYHB)
    88              IF (IVSxJ.EQ.6) CALL H1PTLT(-1,RVSxDXHB,RVSxDYHB)
    89              IF (IVSxJ.EQ.7) CALL HUTLT(RVSxDXHB,RVSxDYHB)
    90              IF (IVSxJ.EQ.8) CALL H5TLT(RVSxDXHB,RVSxDYHB)
    91          ENDIF
    92          RETURN
    93          END


TILTS  Local Symbols

Name                        Class    Type              Size

IVSXJ . . . . . . . . . . . param
```

TILTS Local Symbols

Name	Class	Type	Size
RVSXDYHB.	param		
RVSXDXHB.	param		
IVCXCASE.	IONO3	INTEGER*4	4
IFCXSN.	RAID	INTEGER*4	4
RPCXPI.	PRAM	REAL*8	8
RPCXDR.	PRAM	REAL*8	8
RPCXRE.	PRAM	REAL*8	8
RVCXV1.	IONO3	REAL*8	8
RVCXV2.	IONO3	REAL*8	8
RPCXHTP	PRAM	REAL*8	8
RVCXXX.	IONO3	REAL*8	8
RVCXHBND.	IONO3	REAL*8	8

```
Line#  Source Line

    95          SUBROUTINE HUTLT(RVSxDXHB, RVSxDYHB)
    96  C
    97  C-------------------------------------------------------------
    98  C
    99  C       HUTLT  -- SUBROUTINE TO CALCULATE BOUNDARY TILT FOR
   100  C                    VALLEY-F2 BOUNDARY
   101  C
   102  C       CALLED BY:      TILTS
   103  C
   104  C-------------------------------------------------------------
   105  C
   106  C       AUTHOR:         MICHAEL H. REILLY & ERIC L. STROBEL
   107  C
   108  C       DATE:           09/30/87
   109  C
   110  C       VERSION:        1.0
   111  C
   112  C-------------------------------------------------------------
   113  C
   114  C       REVISED:        09/30/87 -- V1.0.  Initial revision.
   115  C
   116  C-------------------------------------------------------------
   117  C
   118  C       USES:           The parameters from the IONO1,2,3 blocks.
   119  C
   120  C
   121  C       To calculate the rate of change of the boundary in the
   122  C         locally horizontal directions.  The boundary here is
   123  C         the boundary between the linear valley profile and the
   124  C         F2 parabolic profile (for those cases where the inter-
   125  C         section is visible on the final profile).  For further
   126  C         details, see the RADAR-C report referred to in the
   127  C         documentation.
   128  C
   129  C
   130  C       RETURNS:        RVSxDXHB          (d/dx) Height of Boundary.
   131  C
   132  C                       RVSxDYHB          (d/dy) Height of Boundary.
   133  C
   134  C-------------------------------------------------------------
   135  C
   136          INTEGER IVCxCASE
   137  C
   138          REAL*8 RVSxDXHB, RVSxDYHB, RTLxD, RVCxHB(3), RVCxHBND
   139          REAL*8 RVLxHUY2, RVLxHUXE, RVCxFB(3), RVLxHUX2, RVLxHUH2
   140          REAL*8 RVCxB5(3), RVCxB6(3), RVCxE5(3), RVCxE6(3)
   141          REAL*8 RVCxA5(3), RVCxA6(3), RVCxV1, RVCxV2, RVCxXX
   142          REAL*8 RVCxALPH, RVCxBETA, RVCxET1, RVCxET2
   143  C
   144          COMMON /IONO1/ RVCxALPH,RVCxBETA,RVCxET1,RVCxET2,RVCxFB,RVCxHB
```

```
Line#  Source Line

   145           COMMON /IONO2/ RVCxA5,RVCxA6,RVCxB5,RVCxB6,RVCxE5,RVCxE6
   146           COMMON /IONO3/ RVCxV1, RVCxV2, RVCxXX, IVCxCASE, RVCxHBND
   147   C
   148           RTLxD = (RVCxHB(2) - RVCxHBND) / RVCxHB(3)
   149           RVLxHUY2 = -RTLxD
   150           RVLxHUXE = 0.9604D00 * RVCxHB(3) / (2.0D00 * RTLxD * RVCxFB(3)
   151           RVLxHUX2 = -RVLxHUXE * RVCxFB(1) / RVCxFB(3)
   152           RVLxHUH2 = 1.0D00
   153           RVSxDXHB = RVLxHUXE*RVCxB5(1) + RVLxHUX2*RVCxB5(3)
   154           RVSxDXHB = RVSxDXHB + RVLxHUH2*RVCxB6(2) + RVLxHUY2*RVCxB6(3
   155           RVSxDXHB = -RVSxDXHB / RVCxV1
   156           RVSxDYHB = RVLxHUXE*RVCxE5(1) + RVLxHUX2*RVCxE5(3)
   157           RVSxDYHB = RVSxDYHB + RVLxHUH2*RVCxE6(2) + RVLxHUY2*RVCxE6(3
   158           RVSxDYHB = RVSxDYHB / RVCxV2
   159           RETURN
   160           END
```

HUTLT Local Symbols

Name	Class	Type	Size
RVSXDYHB.	param		
RVSXDXHB.	param		
RVLXHUXE.	local	REAL*8	8
RTLXD	local	REAL*8	8
RVLXHUH2.	local	REAL*8	8
RVLXHUX2.	local	REAL*8	8
RVLXHUY2.	local	REAL*8	8
IVCXCASE.	IONO3	INTEGER*4	4
RVCXHB.	IONO1	REAL*8	24
RVCXHBND.	IONO3	REAL*8	8
RVCXFB.	IONO1	REAL*8	24
RVCXB5.	IONO2	REAL*8	24
RVCXB6.	IONO2	REAL*8	24
RVCXE5.	IONO2	REAL*8	24
RVCXE6.	IONO2	REAL*8	24
RVCXA5.	IONO2	REAL*8	24
RVCXA6.	IONO2	REAL*8	24
RVCXV1.	IONO3	REAL*8	8
RVCXV2.	IONO3	REAL*8	8
RVCXXX.	IONO3	REAL*8	8
RVCXALPH.	IONO1	REAL*8	8
RVCXBETA.	IONO1	REAL*8	8
RVCXET1	IONO1	REAL*8	8
RVCXET2	IONO1	REAL*8	8

```
Line#  Source Line

  163          SUBROUTINE H5TLT(RVSxDXHB, RVSxDYHB)
  164  C
  165  C-------------------------------------------------------------
  166  C
  167  C       H5TLT  -- SUBROUTINE TO CALCULATE BOUNDARY TILT FOR
  168  C                    F2-TOPSIDE BOUNDARY
  169  C
  170  C       CALLED BY:      TILTS
  171  C
  172  C-------------------------------------------------------------
  173  C
  174  C       AUTHOR:         MICHAEL H. REILLY & ERIC L. STROBEL
  175  C
  176  C       DATE:           09/30/87
  177  C
  178  C       VERSION:        1.0
  179  C
  180  C-------------------------------------------------------------
  181  C
  182  C       REVISED:        09/30/87 -- V1.0.  Initial revision.
  183  C
  184  C-------------------------------------------------------------
  185  C
  186  C       USES:           The parameters from the IONO2,3 blocks.
  187  C
  188  C
  189  C       To calculate the rate of change of the boundary in the
  190  C         locally horizontal directions.  The boundary here is
  191  C         the boundary between the parabolic F2 profile and the
  192  C         Bent topside profile.  For further details, see the
  193  C         RADAR-C report referred to in the documentation.
  194  C
  195  C
  196  C       RETURNS:        RVSxDXHB          (d/dx) Height of Boundary.
  197  C
  198  C                       RVSxDYHB          (d/dy) Height of Boundary.
  199  C
  200  C-------------------------------------------------------------
  201  C
  202          INTEGER ITCxA
  203  C
  204          REAL*8 RVSxDXHB, RVSxDYHB, RVLxH5Y2, RVLxH5H2
  205          REAL*8 RVCxB5(3), RVCxB6(3), RVCxE5(3), RVCxE6(3)
  206          REAL*8 RVCxA5(3), RVCxA6(3), RVCxV1, RVCxV2
  207          REAL*8 RTCxA, RTCxB
  208  C
  209          COMMON /IONO2/ RVCxA5,RVCxA6,RVCxB5,RVCxB6,RVCxE5,RVCxE6
  210          COMMON /IONO3/ RVCxV1, RVCxV2, RTCxA, ITCxA, RTCxB
  211  C
  212          RVLxH5H2 = 1.0D00
```

```
 Line#  Source Line

   213            RVLxH5Y2 = 0.25D00
   214            RVSxDXHB = -(RVLxH5H2*RVCxB6(2) + RVLxH5Y2*RVCxB6(3)) / RVCxV1
   215            RVSxDYHB = (RVLxH5H2*RVCxE6(2) + RVLxH5Y2*RVCxE6(3)) / RVCxV2
   216            RETURN
   217            END
```

H5TLT Local Symbols

Name	Class	Type	Size
RVSXDYHB.	param		
RVSXDXHB.	param		
RVLXH5H2.	local	REAL*8	8
RVLXH5Y2.	local	REAL*8	8
ITCXA	IONO3	INTEGER*4	4
RVCXB5.	IONO2	REAL*8	24
RVCXB6.	IONO2	REAL*8	24
RVCXE5.	IONO2	REAL*8	24
RVCXE6.	IONO2	REAL*8	24
RVCXA5.	IONO2	REAL*8	24
RVCXA6.	IONO2	REAL*8	24
RVCXV1.	IONO3	REAL*8	8
RVCXV2.	IONO3	REAL*8	8
RTCXA	IONO3	REAL*8	8
RTCXB	IONO3	REAL*8	8

```
Line#  Source Line

  220          SUBROUTINE H1PTLT(IVSxSGN,RVSxDXHB, RVSxDYHB)
  221  C
  222  C-----------------------------------------------------------------
  223  C
  224  C       H1PTLT  -- SUBROUTINE TO CALCULATE BOUNDARY TILT FOR
  225  C                    VALLEY-F1 (PARABOLIC) BOUNDARY
  226  C
  227  C       CALLED BY:       TILTS
  228  C
  229  C       CALLS:           DH1PDP
  230  C
  231  C-----------------------------------------------------------------
  232  C
  233  C       AUTHOR:          MICHAEL H. REILLY & ERIC L. STROBEL
  234  C
  235  C       DATE:            09/30/87
  236  C
  237  C       VERSION:         1.0
  238  C
  239  C-----------------------------------------------------------------
  240  C
  241  C       REVISED:         09/30/87 -- V1.0.  Initial revision.
  242  C
  243  C-----------------------------------------------------------------
  244  C
  245  C       USES:            The parameters from the COMMON blocks.
  246  C
  247  C
  248  C       To calculate the rate of change of the boundary in the
  249  C         locally horizontal directions.  The boundary here is
  250  C         the boundary between the linear valley profile and the
  251  C         parabolic F1 profile.  The routine DH1PDP is just
  252  C         d (H1P) / d (parameter).  For further details, see the
  253  C         RADAR-C report referred to in the documentation.
  254  C
  255  C
  256  C       RETURNS:         RVSxDXHB          (d/dx) Height of Boundary.
  257  C
  258  C                        RVSxDYHB          (d/dy) Height of Boundary.
  259  C
  260  C-----------------------------------------------------------------
  261  C
  262          INTEGER IVSxSGN, ITCxA
  263  C
  264          REAL*8 RVSxDXHB, RVSxDYHB, RVLxD1, RVCxA1, RVCxB1, RVCxC1
  265          REAL*8 RVLxBB, RVCxB0, RVCxHU, RVCxHL, RVCxHB(3), RVCxFB(3
  266          REAL*8 RVLxHUXE, RVLxB1XE, RVLxC1XE, RVCxA0, RVLxHPXE
  267          REAL*8 RVLxA1X1, RVLxB1X1, RVLxHPX1, RVLxA1H1, RVLxB1H1
  268          REAL*8 RVLxHPH1, RVLxHUX2, RVLxB1X2, RVLxC1X2, RVLxHPX2
  269          REAL*8 RVLxB1H2, RVLxC1H2, RVLxHPH2, RVLxHUY2, RVLxB1Y2
```

```
Line#  Source Line

  270           REAL*8 RVLxC1Y2, RVLxHPY2, RVCxB5(3), RVCxB6(3), RVCxE5(3)
  271           REAL*8 RVCxE6(3), RVCxV1, RVCxV2, RVCxALPH, RVCxBETA
  272           REAL*8 RVCxET1, RVCxET2, RVCxA5(3), RVCxA6(3)
  273           REAL*8 RVCxF40, RVCxF65, RVCxK1, RVCxXL, RVCxXU
  274           REAL*8 RTCxA, RTCxB, RTCxC, RTCxD
  275     C
  276           COMMON /IONO1/ RVCxALPH,RVCxBETA,RVCxET1,RVCxET2,RVCxFB,RVCxHB
  277           COMMON /IONO2/ RVCxA5,RVCxA6,RVCxB5,RVCxB6,RVCxE5,RVCxE6
  278           COMMON /IONO3/ RVCxV1, RVCxV2, RTCxA, ITCxA, RTCxB
  279           COMMON /TEMP1/ RVCxF40, RVCxF65, RVCxK1, RVCxHL
  280           COMMON /VAR1/ RVCxXL, RVCxXU, RVCxHU, RVCxA0, RVCxB0
  281           COMMON /VAR3/ RVCxA1, RVCxB1, RVCxC1, RTCxC, RTCxD
  282     C
  283           RVLxD1 = DSQRT(RVCxB1*RVCxB1 - 4.0D00*RVCxA1*RVCxC1)
  284           RVLxBB = RVCxB0 / (RVCxHU - RVCxHL)
  285           RVLxHUXE = (0.98D00 * RVCxHB(3))*(0.98D00 * RVCxHB(3))
  286           RVLxHUXE = RVLxHUXE /(2.0D00 * RVCxFB(3)*(RVCxHB(2)-RVCxHU))
  287           RVLxB1XE = RVCxB0 * (-1.0D00 + (RVCxFB(1)*RVLxHUXE)/
  288         #(RVCxHU - RVCxHL)) / RVCxFB(1)
  289           RVLxC1XE = RVCxA0/RVCxFB(1) + RVCxHL*RVLxBB*RVLxHUXE
  290           CALL DH1PDP(0.0D00, RVLxB1XE, RVLxC1XE, RVLxHPXE, RVLxD1,
  291         #IVSxSGN)
  292           RVLxA1X1 = 16.0D00 / (RVCxHB(1) * RVCxHB(1))
  293           RVLxB1X1 = 32.0D00 / RVCxHB(1)
  294           CALL DH1PDP(RVLxA1X1, RVLxB1X1, 15.0D00, RVLxHPX1, RVLxD1,
  295         #IVSxSGN)
  296           RVLxA1H1 = -32.0D00*RVCxFB(2)/(RVCxHB(1)*RVCxHB(1)*RVCxHB(1
  297           RVLxB1H1 = RVLxA1H1 * RVCxHB(1)
  298           CALL DH1PDP(RVLxA1H1, RVLxB1H1, 0.0D00, RVLxHPH1, RVLxD1,
  299         #IVSxSGN)
  300           RVLxHUX2 = -RVLxHUXE * RVCxFB(1) / RVCxFB(3)
  301           RVLxB1X2 = RVLxBB * RVLxHUX2
  302           RVLxC1X2 = RVCxHL * RVLxB1X2
  303           CALL DH1PDP(0.0D00, RVLxB1X2, RVLxC1X2, RVLxHPX2, RVLxD1,
  304         #IVSxSGN)
  305           RVLxB1H2 = RVLxBB
  306           RVLxC1H2 = RVCxHL * RVLxBB
  307           CALL DH1PDP(0.0D00, RVLxB1H2, RVLxC1H2, RVLxHPH2, RVLxD1,
  308         #IVSxSGN)
  309           RVLxHUY2 = -(RVCxHB(2) - RVCxHU) / RVCxHB(3)
  310           RVLxB1Y2 = RVLxBB * RVLxHUY2
  311           RVLxC1Y2 = RVCxHL * RVLxB1Y2
  312           CALL DH1PDP(0.0D00, RVLxB1Y2, RVLxC1Y2, RVLxHPY2, RVLxD1,
  313         #IVSxSGN)
  314           RVSxDXHB = RVLxHPXE*RVCxB5(1) + RVLxHPX1*RVCxB5(2) +
  315         #RVLxHPX2*RVCxB5(3)
  316           RVSxDXHB = RVSxDXHB + RVLxHPH1*RVCxB6(1) + RVLxHPH2*RVCxB6(2)
  317         #+ RVLxHPY2*RVCxB6(3)
  318           RVSxDXHB = -RVSxDXHB / RVCxV1
  319           RVSxDYHB = RVLxHPXE*RVCxE5(1) + RVLxHPX1*RVCxE5(2) +
```

```
 Line#  Source Line

    320        #RVLxHPX2*RVCxE5(3)
    321           RVSxDYHB = RVSxDYHB + RVLxHPH1*RVCxE6(1) + RVLxHPH2*RVCxE6(2)
    322        #  RVLxHPY2*RVCxE6(3)
    323           RVSxDYHB = RVSxDYHB / RVCxV2
    324           RETURN
    325           END


H1PTLT  Local Symbols

Name                            Class    Type              Size

RVSXDYHB. . . . . . . . . . . . param
RVSXDXHB. . . . . . . . . . . . param
RVSXSGN . . . . . . . . . . . . param
RVLXHPXE. . . . . . . . . . . . local    REAL*8               8
RVLXD1. . . . . . . . . . . . . local    REAL*8               8
RVLXHUXE. . . . . . . . . . . . local    REAL*8               8
RVLXA1X1. . . . . . . . . . . . local    REAL*8               8
RVLXB1X1. . . . . . . . . . . . local    REAL*8               8
RVLXB1X2. . . . . . . . . . . . local    REAL*8               8
RVLXB1Y2. . . . . . . . . . . . local    REAL*8               8
RVLXC1X2. . . . . . . . . . . . local    REAL*8               8
RVLXC1Y2. . . . . . . . . . . . local    REAL*8               8
RVLXBB. . . . . . . . . . . . . local    REAL*8               8
RVLXB1XE. . . . . . . . . . . . local    REAL*8               8
RVLXHPH1. . . . . . . . . . . . local    REAL*8               8
RVLXC1XE. . . . . . . . . . . . local    REAL*8               8
RVLXHPH2. . . . . . . . . . . . local    REAL*8               8
RVLXHPX1. . . . . . . . . . . . local    REAL*8               8
RVLXHPX2. . . . . . . . . . . . local    REAL*8               8
RVLXHPY2. . . . . . . . . . . . local    REAL*8               8
RVLXHUX2. . . . . . . . . . . . local    REAL*8               8
RVLXHUY2. . . . . . . . . . . . local    REAL*8               8
RVLXA1H1. . . . . . . . . . . . local    REAL*8               8
RVLXB1H1. . . . . . . . . . . . local    REAL*8               8
RVLXB1H2. . . . . . . . . . . . local    REAL*8               8
RVLXC1H2. . . . . . . . . . . . local    REAL*8               8
RVCXA0. . . . . . . . . . . . . VAR1     REAL*8               8
RVCXB5. . . . . . . . . . . . . IONO2    REAL*8              24
RVCXB6. . . . . . . . . . . . . IONO2    REAL*8              24
RVCXE5. . . . . . . . . . . . . IONO2    REAL*8              24
RVCXE6. . . . . . . . . . . . . IONO2    REAL*8              24
RVCXV1. . . . . . . . . . . . . IONO3    REAL*8               8
RVCXV2. . . . . . . . . . . . . IONO3    REAL*8               8
RVCXALPH. . . . . . . . . . . . IONO1    REAL*8               8
RVCXBETA. . . . . . . . . . . . IONO1    REAL*8               8
RVCXET1 . . . . . . . . . . . . IONO1    REAL*8               8
RVCXET2 . . . . . . . . . . . . IONO1    REAL*8               8
RVCXA5. . . . . . . . . . . . . IONO2    REAL*8              24
```

H1PTLT Local Symbols

Name	Class	Type	Size
RVCXA6	IONO2	REAL*8	24
RVCXF40	TEMP1	REAL*8	8
RVCXF65	TEMP1	REAL*8	8
RVCXK1	TEMP1	REAL*8	8
RVCXXL	VAR1	REAL*8	8
RVCXXU	VAR1	REAL*8	8
RTCXA	IONO3	REAL*8	8
RTCXB	IONO3	REAL*8	8
RTCXC	VAR3	REAL*8	8
RTCXD	VAR3	REAL*8	8
ITCXA	IONO3	INTEGER*4	4
RVCXA1	VAR3	REAL*8	8
RVCXB1	VAR3	REAL*8	8
RVCXC1	VAR3	REAL*8	8
RVCXB0	VAR1	REAL*8	8
RVCXHU	VAR1	REAL*8	8
RVCXHL	TEMP1	REAL*8	8
RVCXHB	IONO1	REAL*8	24
RVCXFB	IONO1	REAL*8	24

```
Line#  Source Line

  328          SUBROUTINE DH1PDP(RVSxA1P, RVSxB1P, RVSxC1P, RVSxDD, RVSxD1,
  329         #IVSxSGN)
  330    C
  331    C-------------------------------------------------------------
  332    C
  333    C       DH1PDP  -- SUBROUTINE TO CALCULATE THE DERIVATIVE OF THE
  334    C                  BOUNDARY HEIGHT W.R.T. A PARTICULAR PARAMETER.
  335    C
  336    C
  337    C       CALLED BY:      H1PTLT
  338    C
  339    C-------------------------------------------------------------
  340    C
  341    C       AUTHOR:         MICHAEL H. REILLY & ERIC L. STROBEL
  342    C
  343    C       DATE:           09/30/87
  344    C
  345    C       VERSION:        1.0
  346    C
  347    C-------------------------------------------------------------
  348    C
  349    C       REVISED:        09/30/87 -- V1.0.  Initial revision.
  350    C
  351    C-------------------------------------------------------------
  352    C
  353    C       USES:           RVSxA,B,C1P    The partials of the para-
  354    C                                      bolic profile parameters w.r.t.
  355    C                                      the parameter of interest.
  356    C
  357    C                       RVSxD1         The discriminant of the para-
  358    C                                      bolic parameters.
  359    C
  360    C                       IVSxSGN        Determines which of the solns.
  361    C                                      of the quadratic eqn. applies.
  362    C
  363    C       To calculate the partial derivative of H1P with respect to
  364    C         the chosen parameter.  For further details, see the
  365    C         RADAR-C report referred to in the documentation.
  366    C
  367    C
  368    C       RETURNS:        RVSxDD         The aforementioned partial
  369    C                                      derivative.
  370    C
  371    C-------------------------------------------------------------
  372    C
  373          INTEGER IVSxSGN
  374    C
  375          REAL*8 RVSxA1P, RVSxB1P, RVSxC1P, RVSxDD, RVSxD1, RVLxH1
  376          REAL*8 RVCxA1, RVCxB1, RVCxC1, RVCxH1P, RVCxH2P
  377    C
```

```
 Line#  Source Line

    378           COMMON /VAR3/ RVCxA1, RVCxB1, RVCxC1, RVCxH1P, RVCxH2P
    379  C
    380           RVLxH1 = RVCxH1P
    381           RVSxDD = (-RVLxH1 + IVSxSGN*RVCxC1/RVSxD1) * RVSxA1P
    382           RVSxDD = RVSxDD + (1.0D00 - IVSxSGN*RVCxB1/RVSxD1)
    383         #*RVSxB1P/2.0D00
    384           RVSxDD = (RVSxDD + IVSxSGN*(RVCxA1/RVSxD1) * RVSxC1P)
    385         #/ RVCxA1
    386           RETURN
    387           END


DH1PDP  Local Symbols

Name                            Class    Type              Size

IVSXSGN . . . . . . . . . . . . param
RVSXD1. . . . . . . . . . . . . param
RVSXDD. . . . . . . . . . . . . param
RVSXC1P . . . . . . . . . . . . param
RVSXB1P . . . . . . . . . . . . param
RVSXA1P . . . . . . . . . . . . param
RVLXH1. . . . . . . . . . . . . local    REAL*8             8
RVCXA1. . . . . . . . . . . . . VAR3     REAL*8             8
RVCXB1. . . . . . . . . . . . . VAR3     REAL*8             8
RVCXC1. . . . . . . . . . . . . VAR3     REAL*8             8
RVCXH1P . . . . . . . . . . . . VAR3     REAL*8             8
RVCXH2P . . . . . . . . . . . . VAR3     REAL*8             8
```

```
Line#  Source Line

  390          SUBROUTINE H4TLT(IVSxSN, RVSxDXHB, RVSxDYHB)
  391  C
  392  C-------------------------------------------------------------
  393  C
  394  C       H4TLT  -- SUBROUTINE TO CALCULATE BOUNDARY TILT FOR
  395  C                      PARABOLIC F1-F2 BOUNDARY
  396  C
  397  C       CALLED BY:      TILTS
  398  C
  399  C       CALLS:          DH4DP
  400  C
  401  C-------------------------------------------------------------
  402  C
  403  C       AUTHOR:         MICHAEL H. REILLY & ERIC L. STROBEL
  404  C
  405  C       DATE:           09/30/87
  406  C
  407  C       VERSION:        1.0
  408  C
  409  C-------------------------------------------------------------
  410  C
  411  C       REVISED:        09/30/87 -- V1.0.  Initial revision.
  412  C
  413  C-------------------------------------------------------------
  414  C
  415  C       USES:           The parameters from the COMMON blocks.
  416  C
  417  C
  418  C       To calculate the rate of change of the boundary in the
  419  C         locally horizontal directions.  The boundary here is
  420  C         the boundary between the parabolic F1 profile and the
  421  C         parabolic F2 profile.  The routine DH4DP is just
  422  C         d (H4) / d (parameter).  For further details, see the
  423  C         RADAR-C report referred to in the documentation.
  424  C
  425  C
  426  C       RETURNS:        RVSxDXHB          (d/dx) Height of Boundary.
  427  C
  428  C                       RVSxDYHB          (d/dy) Height of Boundary.
  429  C
  430  C-------------------------------------------------------------
  431  C
  432          INTEGER IVSxSN, ITCxA
  433  C
  434          REAL*8 RVSxDXHB, RVSxDYHB, RVLxD2, RVCxA2, RVCxB2, RVCxC2
  435          REAL*8 RVCxHT4, RVCxHB(3), RVCxFB(3), RVLxBB1, RVCxHT3
  436          REAL*8 RVLxBB2, RVLxA2X1, RVLxB2X1, RVLxH4X1, RVLxA2X2
  437          REAL*8 RVLxB2X2, RVLxC2X2, RVLxH4X2, RVLxA2H1, RVLxB2H1
  438          REAL*8 RVLxH4H1, RVLxB2H2, RVLxC2H2, RVLxH4H2, RVLxA2Y2
  439          REAL*8 RVLxB2Y2, RVLxC2Y2, RVLxH4Y2, RVCxB5(3), RVCxB6(3)
```

```
 Line#  Source Line

   440          REAL*8 RVCxE5(3), RVCxE6(3), RVCxV1, RVCxV2, RVCxALPH
   441          REAL*8 RVCxBETA, RVCxET1, RVCxET2, RVCxA5(3), RVCxA6(3)
   442          REAL*8 RTCxA, RTCxB
   443  C
   444          COMMON /IONO1/ RVCxALPH,RVCxBETA,RVCxET1,RVCxET2,RVCxFB,RVCxHB
   445          COMMON /IONO2/ RVCxA5,RVCxA6,RVCxB5,RVCxB6,RVCxE5,RVCxE6
   446          COMMON /IONO3/ RVCxV1, RVCxV2, RTCxA, ITCxA, RTCxB
   447          COMMON /VAR4/ RVCxA2,RVCxB2,RVCxC2,RVCxHT3,RVCxHT4
   448  C
   449             RVLxD2 = DSQRT(RVCxB2*RVCxB2 - 4.0D00*RVCxA2*RVCxC2)*IVSxSN
   450           #*INTSIGN(RVCxB2)
   451             RVCxHT4 = -(RVCxB2 + RVLxD2) / (2.0D00 * RVCxA2)
   452             RVLxBB1 = 16.0D00 / (RVCxHB(1) * RVCxHB(1))
   453             RVLxBB2 = 1.0D00 / (RVCxHB(3) * RVCxHB(3))
   454             RVLxA2X1 = -RVLxBB1
   455             RVLxB2X1 = 32.0D00 / RVCxHB(1)
   456             CALL DH4DP(RVLxA2X1, RVLxB2X1, -15.0D00, RVLxH4X1, RVLxD2)
   457             RVLxA2H1 = 2.0D00 * RVLxBB1 * RVCxFB(2) / RVCxHB(1)
   458             RVLxB2H1 = -RVLxA2H1 * RVCxHB(1)
   459             CALL DH4DP(RVLxA2H1, RVLxB2H1, 0.0D00, RVLxH4H1, RVLxD2)
   460             RVLxA2X2 = RVLxBB2
   461             RVLxB2X2 = -2.0D00 * RVCxHB(2) * RVLxBB2
   462             RVLxC2X2 = (-RVLxB2X2 * RVCxHB(2)/2.0D00) - 1.0D00
   463             CALL DH4DP(RVLxA2X2, RVLxB2X2, RVLxC2X2, RVLxH4X2, RVLxD2)
   464             RVLxB2H2 = -2.0D00 * RVCxFB(3) * RVLxBB2
   465             RVLxC2H2 = - RVLxB2H2 * RVCxHB(2)
   466             CALL DH4DP(0.0D00, RVLxB2H2, RVLxC2H2, RVLxH4H2, RVLxD2)
   467             RVLxA2Y2 = RVLxB2H2 / RVCxHB(3)
   468             RVLxB2Y2 = -2.0D00 * RVLxA2Y2 * RVCxHB(2)
   469             RVLxC2Y2 = RVLxA2Y2 * RVCxHB(2) * RVCxHB(2)
   470             CALL DH4DP(RVLxA2Y2, RVLxB2Y2, RVLxC2Y2, RVLxH4Y2, RVLxD2
   471             RVSxDXHB = RVLxH4X1*RVCxB5(2) + RVLxH4X2*RVCxB5(3) +
   472           #RVLxH4H1*RVCxB6(1)
   473             RVSxDXHB = RVSxDXHB + RVLxH4H2*RVCxB6(2) + RVLxH4Y2*RVCxB6(3
   474             RVSxDXHB = -RVSxDXHB / RVCxV1
   475             RVSxDYHB = RVLxH4X1*RVCxE5(2) + RVLxH4X2*RVCxE5(3) +
   476           #RVLxH4H1*RVCxE6(1)
   477             RVSxDYHB = RVSxDYHB + RVLxH4H2*RVCxE6(2) + RVLxH4Y2*RVCxE6(3
   478             RVSxDYHB = RVSxDYHB / RVCxV2
   479             RETURN
   480             END


H4TLT  Local Symbols

Name                          Class    Type              Size

RVSXDYHB. . . . . . . . . . . param
RVSXDXHB. . . . . . . . . . . param
IVSXSN. . . . . . . . . . . . param
```

H4TLT Local Symbols

Name	Class	Type	Size
RVLXH4H1	local	REAL*8	8
RVLXBB1	local	REAL*8	8
RVLXH4H2	local	REAL*8	8
RVLXBB2	local	REAL*8	8
RVLXD2	local	REAL*8	8
RVLXA2X1	local	REAL*8	8
RVLXA2X2	local	REAL*8	8
RVLXB2X1	local	REAL*8	8
RVLXA2Y2	local	REAL*8	8
RVLXB2X2	local	REAL*8	8
RVLXB2Y2	local	REAL*8	8
RVLXC2X2	local	REAL*8	8
RVLXC2Y2	local	REAL*8	8
RVLXH4X1	local	REAL*8	8
RVLXH4X2	local	REAL*8	8
RVLXH4Y2	local	REAL*8	8
RVLXA2H1	local	REAL*8	8
RVLXB2H1	local	REAL*8	8
RVLXB2H2	local	REAL*8	8
RVLXC2H2	local	REAL*8	8
ITCXA	IONO3	INTEGER*4	4
RVCXA2	VAR4	REAL*8	8
RVCXB2	VAR4	REAL*8	8
RVCXC2	VAR4	REAL*8	8
RVCXHT4	VAR4	REAL*8	8
RVCXHB	IONO1	REAL*8	24
RVCXFB	IONO1	REAL*8	24
RVCXHT3	VAR4	REAL*8	8
RVCXB5	IONO2	REAL*8	24
RVCXB6	IONO2	REAL*8	24
RVCXE5	IONO2	REAL*8	24
RVCXE6	IONO2	REAL*8	24
RVCXV1	IONO3	REAL*8	8
RVCXV2	IONO3	REAL*8	8
RVCXALPH	IONO1	REAL*8	8
RVCXBETA	IONO1	REAL*8	8
RVCXET1	IONO1	REAL*8	8
RVCXET2	IONO1	REAL*8	8
RVCXA5	IONO2	REAL*8	24
RVCXA6	IONO2	REAL*8	24
RTCXA	IONO3	REAL*8	8
RTCXB	IONO3	REAL*8	8

```
Line#  Source Line

  483          SUBROUTINE DH4DP(RVSxA2D, RVSxB2D, RVSxC2D, RVSxH4D, RVSxD2)
  484  C
  485  C-------------------------------------------------------------
  486  C
  487  C       DH4DP  -- SUBROUTINE TO CALCULATE THE DERIVATIVE OF THE
  488  C                 BOUNDARY HEIGHT W.R.T. A PARTICULAR PARAMETER.
  489  C
  490  C
  491  C       CALLED BY:      H4TLT
  492  C
  493  C-------------------------------------------------------------
  494  C
  495  C       AUTHOR:         MICHAEL H. REILLY & ERIC L. STROBEL
  496  C
  497  C       DATE:           09/30/87
  498  C
  499  C       VERSION:        1.0
  500  C
  501  C-------------------------------------------------------------
  502  C
  503  C       REVISED:        09/30/87 -- V1.0  Initial revision.
  504  C
  505  C-------------------------------------------------------------
  506  C
  507  C       USES:           RVSxA,B,C2D    The partials of the para-
  508  C                                      bolic profile parameters w.r.t.
  509  C                                      the parameter of interest.
  510  C
  511  C                       RVSxD2         The discriminant of the para-
  512  C                                      bolic parameters.
  513  C
  514  C
  515  C       To calculate the partial derivative of H4 with respect to
  516  C         the chosen parameter.  For further details, see the
  517  C         RADAR-C report referred to in the documentation.
  518  C
  519  C
  520  C       RETURNS:        RVSxH4D          The aforementioned partial
  521  C                                      derivative.
  522  C
  523  C-------------------------------------------------------------
  524  C
  525          REAL*8 RVSxA2D, RVSxB2D, RVSxC2D, RVSxH4D, RVSxD2, RVCxHT4
  526          REAL*8 RVCxA2, RVCxB2, RVCxC2, RVCxHT3
  527  C
  528          COMMON /VAR4/ RVCxA2,RVCxB2,RVCxC2,RVCxHT3,RVCxHT4
  529  C
  530          RVSxH4D = (-RVCxHT4 + RVCxC2/RVSxD2) * RVSxA2D
  531          RVSxH4D = RVSxH4D - (1.0D00 + RVCxB2/RVSxD2)*RVSxB2D/2.0D00
  532          RVSxH4D = (RVSxH4D + (RVCxA2/RVSxD2)*RVSxC2D)/RVCxA2
```

```
 Line#  Source Line

   533          RETURN
   534          END
```

DH4DP Local Symbols

Name	Class	Type	Size
RVSXD2.	param		
RVSXH4D	param		
RVSXC2D	param		
RVSXB2D	param		
RVSXA2D	param		
RVCXC2.	VAR4	REAL*8	8
RVCXHT3	VAR4	REAL*8	8
RVCXHT4	VAR4	REAL*8	8
RVCXA2.	VAR4	REAL*8	8
RVCXB2.	VAR4	REAL*8	8

```
Line#  Source Line

  537          SUBROUTINE H1LTLT(RVSxDXHB, RVSxDYHB)
  538  C
  539  C-----------------------------------------------------------------
  540  C
  541  C       H1LTLT  -- SUBROUTINE TO CALCULATE BOUNDARY TILT FOR
  542  C                    LINEAR VALLEY - LINEAR F1 BOUNDARY
  543  C
  544  C       CALLED BY:      TILTS
  545  C
  546  C       CALLS:          DH1LDP
  547  C
  548  C-----------------------------------------------------------------
  549  C
  550  C       AUTHOR:         MICHAEL H. REILLY & ERIC L. STROBEL
  551  C
  552  C       DATE:           09/30/87
  553  C
  554  C       VERSION:        1.0
  555  C
  556  C-----------------------------------------------------------------
  557  C
  558  C       REVISED:        09/30/87 -- V1.0.  Initial revision.
  559  C
  560  C-----------------------------------------------------------------
  561  C
  562  C       USES:           The parameters from the COMMON blocks.
  563  C
  564  C
  565  C       To calculate the rate of change of the boundary in the
  566  C         locally horizontal directions.  The boundary here is
  567  C         the boundary between the linear valley profile and the
  568  C         linear F1 profile.  The routine DH1LDP is just
  569  C         d (H1L) / d (parameter).  For further details, see the
  570  C         RADAR-C report referred to in the documentation.
  571  C
  572  C
  573  C       RETURNS:        RVSxDXHB          (d/dx) Height of Boundary.
  574  C
  575  C                       RVSxDYHB          (d/dy) Height of Boundary.
  576  C
  577  C-----------------------------------------------------------------
  578  C
  579          INTEGER ITCxA
  580  C
  581          REAL*8 RVSxDXHB, RVSxDYHB, RVLxBB1, RVCxB0, RVCxHU, RVCxHL
  582          REAL*8 RVLxHUXE, RVCxHB(3), RVCxFB(3),RVLxYSX1, RVCxH2
  583          REAL*8 RVLxBB2, RVCxYS, RVLxA0XE, RVCxA0, RVLxB0XE, RVLxH1XE
  584          REAL*8 RVLxS1X1, RVLxH1X1, RVLxS1H1, RVLxH1H1, RVLxHUX2
  585          REAL*8 RVLxA0X2, RVLxB0X2, RVLxS1X2, RVLxH1X2, RVLxA0H2
  586          REAL*8 RVLxB0H2, RVLxS1H2, RVLxH1H2, RVLxHUY2, RVLxA0Y2
```

```
Line#  Source Line

  587           REAL*8 RVLxB0Y2, RVLxS1Y2, RVLxH1Y2, RVCxB5(3), RVCxB6(3)
  588           REAL*8 RVCxE5(3), RVCxE6(3), RVCxV1, RVCxV2, RVCxALPH
  589           REAL*8 RVCxBETA, RVCxET1, RVCxET2, RVCxA5(3), RVCxA6(3)
  590           REAL*8 RVCxF40, RVCxF65, RVCxK1, RVCxXL, RVCxXU, RVCxHB1
  591           REAL*8 RTCxA, RTCxB, RTCxC, RTCxD, RTCxE
  592   C
  593           COMMON /IONO1/ RVCxALPH,RVCxBETA,RVCxET1,RVCxET2,RVCxFB,RVCxHB
  594           COMMON /IONO2/ RVCxA5,RVCxA6,RVCxB5,RVCxB6,RVCxE5,RVCxE6
  595           COMMON /IONO3/ RVCxV1, RVCxV2, RTCxA, ITCxA, RTCxB
  596           COMMON /TEMP1/ RVCxF40, RVCxF65, RVCxK1, RVCxHL
  597           COMMON /VAR1/ RVCxXL, RVCxXU, RVCxHU, RVCxA0, RVCxB0
  598           COMMON /VAR2/ RVCxHB1,RVCxH2,RVCxYS,RTCxC,RTCxD,RTCxE
  599   C
  600           RVLxBB1 = RVCxB0 / (RVCxHU - RVCxHL)
  601           RVLxHUXE = 0.9604D00 * RVCxHB(3) * RVCxHB(3) / (2.0D00 *
  602         #RVCxFB(3) * (RVCxHB(2) - RVCxHU))
  603           RVLxYSX1 = RVCxHB(3) * RVCxHB(3) / (2.0D00 * RVCxFB(3) *
  604         #(RVCxHB(2) - RVCxH2))
  605           RVLxBB2 = RVCxFB(2) / (RVCxYS * RVCxYS)
  606           RVLxA0XE = RVCxA0/RVCxFB(1) + RVCxHL*RVLxBB1*RVLxHUXE
  607           RVLxB0XE = RVCxB0/RVCxFB(1) - RVLxBB1*RVLxHUXE
  608           CALL DH1LDP(RVLxA0XE, 0.0D00, RVLxB0XE, 0.0D00, RVLxH1XE)
  609           RVLxS1X1 = 1.0D00 / RVCxYS
  610           IF (RVCxYS.NE.1.0D00) RVLxS1X1 = RVLxS1X1 - RVLxBB2*RVLxYSX1
  611           CALL DH1LDP(0.0D00,0.0D00,0.0D00, RVLxS1X1, RVLxH1X1)
  612           RVLxS1H1 = 0.0D00
  613           IF (RVCxYS.NE.1.0D00) RVLxS1H1 = 0.75D00 * RVLxBB2
  614           CALL DH1LDP(0.0D00,1.0D00,0.0D00, RVLxS1H1, RVLxH1H1)
  615           RVLxHUX2 = -RVLxHUXE * RVCxFB(1) / RVCxFB(3)
  616           RVLxA0X2 = RVCxHL * RVLxBB1 * RVLxHUX2
  617           RVLxB0X2 = -RVLxBB1 * RVLxHUX2
  618           RVLxS1X2 = 0.0D00
  619           IF (RVCxYS.NE.1.0D00) RVLxS1X2 = RVLxBB2*RVLxYSX1*RVCxFB(2)
  620         #/RVCxFB(3)
  621           CALL DH1LDP(RVLxA0X2, 0.0D00, RVLxB0X2, RVLxS1X2, RVLxH1X2)
  622           RVLxA0H2 = RVCxHL * RVLxBB1
  623           RVLxB0H2 = -RVLxBB1
  624           RVLxS1H2 = 0.0D00
  625           IF (RVCxYS.NE.1.0D00) RVLxS1H2 = -RVLxBB2
  626           CALL DH1LDP(RVLxA0H2, 0.0D00, RVLxB0H2, RVLxS1H2, RVLxH1H2)
  627           RVLxHUY2 = -(RVCxHB(2) - RVCxHU) / RVCxHB(3)
  628           RVLxA0Y2 = RVCxHL * RVLxBB1 * RVLxHUY2
  629           RVLxB0Y2 = -RVLxBB1 * RVLxHUY2
  630           RVLxS1Y2 = 0.0D00
  631           IF (RVCxYS.NE.1.0D00) RVLxS1Y2 = RVLxBB2*(RVCxHB(2)-RVCxH2
  632         #/RVCxHB(3)
  633           CALL DH1LDP(RVLxA0Y2, 0.0D00, RVLxB0Y2, RVLxS1Y2, RVLxH1Y2)
  634           RVSxDXHB = RVLxH1XE*RVCxB5(1) + RVLxH1X1*RVCxB5(2)
  635         #+ RVLxH1X2*RVCxB5(3)
  636           RVSxDXHB = RVSxDXHB + RVLxH1H1*RVCxB6(1) + RVLxH1H2*RVCxB6(2)
```

```
 Line#  Source Line

    637        #+ RVLxH1Y2*RVCxB6(3)
    638           RVSxDXHB = -RVSxDXHB / RVCxV1
    639           RVSxDYHB = RVLxH1XE*RVCxE5(1) + RVLxH1X1*RVCxE5(2)
    640        #+ RVLxH1X2*RVCxE5(3)
    641           RVSxDYHB = RVSxDYHB + RVLxH1H1*RVCxE6(1) + RVLxH1H2*RVCxE6(2)
    642        #+ RVLxH1Y2*RVCxE6(3)
    643           RVSxDYHB = RVSxDYHB / RVCxV2
    644           RETURN
    645           END


H1LTLT  Local Symbols

Name                            Class    Type              Size

RVSXDYHB. . . . . . . . . . . . param
RVSXDXHB. . . . . . . . . . . . param
RVLXYSX1. . . . . . . . . . . . local    REAL*8               8
RVLXBB1 . . . . . . . . . . . . local    REAL*8               8
RVLXBB2 . . . . . . . . . . . . local    REAL*8               8
RVLXHUXE. . . . . . . . . . . . local    REAL*8               8
RVLXA0X2. . . . . . . . . . . . local    REAL*8               8
RVLXA0Y2. . . . . . . . . . . . local    REAL*8               8
RVLXB0X2. . . . . . . . . . . . local    REAL*8               8
RVLXB0Y2. . . . . . . . . . . . local    REAL*8               8
RVLXS1H1. . . . . . . . . . . . local    REAL*8               8
RVLXS1H2. . . . . . . . . . . . local    REAL*8               8
RVLXH1X1. . . . . . . . . . . . local    REAL*8               8
RVLXH1X2. . . . . . . . . . . . local    REAL*8               8
RVLXH1Y2. . . . . . . . . . . . local    REAL*8               8
RVLXS1X1. . . . . . . . . . . . local    REAL*8               8
RVLXS1X2. . . . . . . . . . . . local    REAL*8               8
RVLXA0XE. . . . . . . . . . . . local    REAL*8               8
RVLXS1Y2. . . . . . . . . . . . local    REAL*8               8
RVLXB0XE. . . . . . . . . . . . local    REAL*8               8
RVLXH1XE. . . . . . . . . . . . local    REAL*8               8
RVLXHUX2. . . . . . . . . . . . local    REAL*8               8
RVLXHUY2. . . . . . . . . . . . local    REAL*8               8
RVLXA0H2. . . . . . . . . . . . local    REAL*8               8
RVLXB0H2. . . . . . . . . . . . local    REAL*8               8
RVLXH1H1. . . . . . . . . . . . local    REAL*8               8
RVLXH1H2. . . . . . . . . . . . local    REAL*8               8
RTCXE . . . . . . . . . . . . . VAR2     REAL*8               8
ITCXA . . . . . . . . . . . . . IONO3    INTEGER*4            4
RVCXB0. . . . . . . . . . . . . VAR1     REAL*8               8
RVCXHU. . . . . . . . . . . . . VAR1     REAL*8               8
RVCXHL. . . . . . . . . . . . . TEMP1    REAL*8               8
RVCXHB. . . . . . . . . . . . . IONO1    REAL*8              24
RVCXFB. . . . . . . . . . . . . IONO1    REAL*8              24
RVCXH2. . . . . . . . . . . . . VAR2     REAL*8               8
```

H1LTLT Local Symbols

Name	Class	Type	Size
RVCXYS	VAR2	REAL*8	8
RVCXA0	VAR1	REAL*8	8
RVCXB5	IONO2	REAL*8	24
RVCXB6	IONO2	REAL*8	24
RVCXE5	IONO2	REAL*8	24
RVCXE6	IONO2	REAL*8	24
RVCXV1	IONO3	REAL*8	8
RVCXV2	IONO3	REAL*8	8
RVCXALPH	IONO1	REAL*8	8
RVCXBETA	IONO1	REAL*8	8
RVCXET1	IONO1	REAL*8	8
RVCXET2	IONO1	REAL*8	8
RVCXA5	IONO2	REAL*8	24
RVCXA6	IONO2	REAL*8	24
RVCXF40	TEMP1	REAL*8	8
RVCXF65	TEMP1	REAL*8	8
RVCXK1	TEMP1	REAL*8	8
RVCXXL	VAR1	REAL*8	8
RVCXXU	VAR1	REAL*8	8
RVCXHB1	VAR2	REAL*8	8
RTCXA	IONO3	REAL*8	8
RTCXB	IONO3	REAL*8	8
RTCXC	VAR2	REAL*8	8
RTCXD	VAR2	REAL*8	8

```
Line#  Source Line

  648          SUBROUTINE DH1LDP(RVSxA0P, RVSxH1D, RVSxB0P, RVSxS1P, RVSxDD)
  649  C
  650  C-----------------------------------------------------------------
  651  C
  652  C       DH1LDP  -- SUBROUTINE TO CALCULATE THE DERIVATIVE OF THE
  653  C                  BOUNDARY HEIGHT W.R.T. A PARTICULAR PARAMETER.
  654  C
  655  C
  656  C       CALLED BY:      H1LTLT
  657  C
  658  C-----------------------------------------------------------------
  659  C
  660  C       AUTHOR:         MICHAEL H. REILLY & ERIC L. STROBEL
  661  C
  662  C       DATE:           09/30/87
  663  C
  664  C       VERSION:        1.0
  665  C
  666  C-----------------------------------------------------------------
  667  C
  668  C       REVISED:        09/30/87 -- V1.0.  Initial revision.
  669  C
  670  C-----------------------------------------------------------------
  671  C
  672  C       USES:           RVSxA,B0P      The partials of the linear
  673  C                                      valley profile parameters w.r.t.
  674  C                                      the parameter of interest.
  675  C
  676  C                       RVSxH1D,S1P    The partials of the linear
  677  C                                      F1 profile parameters w.r.t.
  678  C                                      the parameter of interest.
  679  C
  680  C
  681  C       To calculate the partial derivative of H1L with respect to
  682  C         the chosen parameter.  For further details, see the
  683  C         RADAR-C report referred to in the documentation.
  684  C
  685  C
  686  C       RETURNS:        RVSxDD          The aforementioned partial
  687  C                                       derivative.
  688  C
  689  C-----------------------------------------------------------------
  690  C
  691          REAL*8 RVSxA0P, RVSxH1D, RVSxB0P, RVSxS1P, RVSxDD, RTLxL
  692          REAL*8 RVCxSL1, RVCxB0, RVCxHB1, RVCxA0, RVCxXL, RVCxXU
  693          REAL*8 RVCxHU, RVCxH2, RVCxYS, RTCxA, RTCxB
  694  C
  695          COMMON /VAR1/ RVCxXL, RVCxXU, RVCxHU, RVCxA0, RVCxB0
  696          COMMON /VAR2/ RVCxHB1,RVCxH2,RVCxYS,RVCxSL1,RTCxA,RTCxB
  697  C
```

```
 Line#  Source Line

   698          RTLx1 = RVCxSL1 - RVCxB0
   699          RVSxDD = RTLx1 * (RVSxA0P + 0.75D00*RVCxSL1*RVSxH1D)
   700          RVSxDD = RVSxDD + (RVCxHB1*RVCxSL1 + RVCxA0) * RVSxB0P
   701          RVSxDD = (RVSxDD - (RVCxHB1*RVCxB0 + RVCxA0)*RVSxS1P)
   702       #/ (RTLx1 * RTLx1)
   703          RETURN
   704          END
```

DH1LDP Local Symbols

Name	Class	Type	Size
RVSXDD.	param		
RVSXS1P	param		
RVSXB0P	param		
RVSXH1D	param		
RVSXA0P	param		
RTLX1	local	REAL*8	8
RVCXSL1	VAR2	REAL*8	8
RVCXB0.	VAR1	REAL*8	8
RVCXHB1	VAR2	REAL*8	8
RVCXA0.	VAR1	REAL*8	8
RVCXXL.	VAR1	REAL*8	8
RVCXXU.	VAR1	REAL*8	8
RVCXHU.	VAR1	REAL*8	8
RVCXH2.	VAR2	REAL*8	8
RVCXYS.	VAR2	REAL*8	8
RTCXA	VAR2	REAL*8	8
RTCXB	VAR2	REAL*8	8

```
Line#  Source Line

  707          SUBROUTINE H2TLT(RVSxDXHB, RVSxDYHB)
  708  C
  709  C-------------------------------------------------------------
  710  C
  711  C       H2TLT  -- SUBROUTINE TO CALCULATE BOUNDARY TILT FOR
  712  C                    LINEAR F1 - F2 BOUNDARY
  713  C
  714  C       CALLED BY:      TILTS
  715  C
  716  C-------------------------------------------------------------
  717  C
  718  C       AUTHOR:         MICHAEL H. REILLY & ERIC L. STROBEL
  719  C
  720  C       DATE:           09/30/87
  721  C
  722  C       VERSION:        1.0
  723  C
  724  C-------------------------------------------------------------
  725  C
  726  C       REVISED:        09/30/87 -- V1.0.  Initial revision.
  727  C
  728  C-------------------------------------------------------------
  729  C
  730  C       USES:           The parameters from the COMMON blocks.
  731  C
  732  C
  733  C       To calculate the rate of change of the boundary in the
  734  C         locally horizontal directions.  The boundary here is
  735  C         the boundary between the linear F1 profile and the
  736  C         parabolic F2 profile.  For further details, see the
  737  C         RADAR-C report referred to in the documentation.
  738  C
  739  C
  740  C       RETURNS:        RVSxDXHB          (d/dx) Height of Boundary.
  741  C
  742  C                       RVSxDYHB          (d/dy) Height of Boundary.
  743  C
  744  C-------------------------------------------------------------
  745  C
  746          INTEGER ITCxA
  747  C
  748          REAL*8 RVSxDXHB, RVSxDYHB, RVLxH2X1, RVCxHB(3), RVCxFB(3)
  749          REAL*8 RVCxH2, RVLxH2X2, RVLxH2H2, RVLxH2Y2, RVCxB5(3)
  750          REAL*8 RVCxB6(3), RVCxE5(3), RVCxE6(3), RVCxV1, RVCxV2
  751          REAL*8 RVCxALPH, RVCxBETA, RVCxET1, RVCxET2, RVCxA5(3)
  752          REAL*8 RVCxA6(3), RVCxHB1, RTCxA, RTCxB, RTCxC, RTCxD
  753          REAL*8 RTCxE, RTCxF
  754  C
  755          COMMON /IONO1/ RVCxALPH,RVCxBETA,RVCxET1,RVCxET2,RVCxFB,RVCxHB
  756          COMMON /IONO2/ RVCxA5,RVCxA6,RVCxB5,RVCxB6,RVCxE5,RVCxE6
```

```
 Line#  Source Line

   757           COMMON /IONO3/ RVCxV1, RVCxV2, RTCxA, ITCxA, RTCxB
   758           COMMON /VAR2/ RVCxHB1,RVCxH2,RTCxC,RTCxD,RTCxE,RTCxF
   759  C
   760           RVLxH2X1 = (RVCxHB(3) * RVCxHB(3))/(2.0D00 * RVCxFB(3) *
   761         #(RVCxHB(2) - RVCxH2))
   762           RVLxH2X2 = -RVLxH2X1 * RVCxFB(2) / RVCxFB(3)
   763           RVLxH2H2 = 1.0D00
   764           RVLxH2Y2 = -(RVCxHB(2) - RVCxH2) / RVCxHB(3)
   765           RVSxDXHB = RVLxH2X1*RVCxB5(2) + RVLxH2X2 * RVCxB5(3)
   766           RVSxDXHB = RVSxDXHB + RVLxH2H2*RVCxB6(2) + RVLxH2Y2*RVCxB6(3)
   767           RVSxDXHB = -RVSxDXHB / RVCxV1
   768           RVSxDYHB = RVLxH2X1*RVCxE5(2) + RVLxH2X2 * RVCxE5(3)
   769           RVSxDYHB = RVSxDYHB + RVLxH2H2*RVCxE6(2) + RVLxH2Y2*RVCxE6(3)
   770           RVSxDYHB = RVSxDYHB / RVCxV2
   771           RETURN
   772           END
```

H2TLT Local Symbols

Name	Class	Type	Size
RVSXDYHB.	param		
RVSXDXHB.	param		
RVLXH2H2.	local	REAL*8	8
RVLXH2X1.	local	REAL*8	8
RVLXH2X2.	local	REAL*8	8
RVLXH2Y2.	local	REAL*8	8
ITCXA	IONO3	INTEGER*4	4
RVCXHB.	IONO1	REAL*8	24
RVCXFB.	IONO1	REAL*8	24
RVCXH2.	VAR2	REAL*8	8
RVCXB5.	IONO2	REAL*8	24
RVCXB6.	IONO2	REAL*8	24
RVCXE5.	IONO2	REAL*8	24
RVCXE6.	IONO2	REAL*8	24
RVCXV1.	IONO3	REAL*8	8
RVCXV2.	IONO3	REAL*8	8
RVCXALPH.	IONO1	REAL*8	8
RVCXBETA.	IONO1	REAL*8	8
RVCXET1	IONO1	REAL*8	8
RVCXET2	IONO1	REAL*8	8
RVCXA5.	IONO2	REAL*8	24
RVCXA6.	IONO2	REAL*8	24
RVCXHB1	VAR2	REAL*8	8
RTCXA	IONO3	REAL*8	8
RTCXB	IONO3	REAL*8	8
RTCXC	VAR2	REAL*8	8
RTCXD	VAR2	REAL*8	8
RTCXE	VAR2	REAL*8	8

H2TLT Local Symbols

Name	Class	Type	Size
RTCXF	VAR2	REAL*8	8

Global Symbols

Name	Class	Type	Size
DH1LDP	FSUBRT	***	***
DH1PDP	FSUBRT	***	***
DH4DP	FSUBRT	***	***
H1LTLT	FSUBRT	***	***
H1PTLT	FSUBRT	***	***
H2TLT	FSUBRT	***	***
H4TLT	FSUBRT	***	***
H5TLT	FSUBRT	***	***
HUTLT	FSUBRT	***	***
INTSIGN	extern	INTEGER*4	***
IONO1	common	***	80
IONO2	common	***	144
IONO3	common	***	36
PRAM	common	***	32
RAID	common	***	4
TEMP1	common	***	32
TILTS	FSUBRT	***	***
VAR1	common	***	40
VAR2	common	***	48
VAR3	common	***	40
VAR4	common	***	40

Code size = 13f3 (5107)
Data size = 007a (122)

```
Line#  Source Line

     1          SUBROUTINE NEWCS(IFSxG)
     2  C
     3  C-----------------------------------------------------------------
     4  C
     5  C   NEWCS -- SUBROUTINE TO CALCULATE NEW CX, CY, CZ VALUES
     6  C
     7  C         CALLED BY:      RAYSUB
     8  C
     9  C         CALLS:          FREESP
    10  C
    11  C-----------------------------------------------------------------
    12  C
    13  C         AUTHOR:         MICHAEL H. REILLY & ERIC L. STROBEL
    14  C
    15  C         DATE:           09/17/86
    16  C
    17  C         VERSION:        2.0
    18  C
    19  C-----------------------------------------------------------------
    20  C
    21  C         REVISED:        07/25/86 -- INITIAL REVISION.  TRANSLATED
    22  C                         FROM TEKTRONIX BASIC TO VAX FORTRAN BY
    23  C                         ERIC L. STROBEL.
    24  C
    25  C                         07/30/86 -- V1.1.  Change over to use of
    26  C                         REAL*8 precision in the calculations.
    27  C
    28  C                         09/17/86 -- V2.0.  Got rid of the use of
    29  C                         approximate rotation matrix in calc. of
    30  C                         the new c-values.
    31  C
    32  C-----------------------------------------------------------------
    33  C
    34  C        USES:    IFSxG            A FLAG
    35  C                 IVCxSX,SY,SZ     OLD SIGN VARIABLES
    36  C                 RVCxXF,YF,ZF     FINAL INTERVAL COORDINATES
    37  C                 RVCxLA1          PREVIOUS POINT'S LAT
    38  C                 RVCxLO1          PREVIOUS POINT'S LON
    39  C                 RVCxLAI          LAT OF BEGINNING OF INTERVAL
    40  C                 RVCxLOI          LON OF BEGINNING OF INTERVAL
    41  C                 RVCxALPH         X
    42  C                 RVCxBETA         X-IONOSPHERIC PARAMETERS
    43  C                 RVCxET1          X
    44  C                 RVCxET2          X
    45  C                 RVCxCX,CY,CZ     OLD C-VALUES
    46  C
    47  C         TO CALCULATE THE NEW C AND S VALUES.
    48  C
    49  C         RETURNS:
    50  C                 IVCxSX,SY,SZ     THE NEW SIGN VALUES
```

```
Line#  Source Line

   51  C                RVCxCX,CY,CZ    THE NEW C VALUES
   52  C
   53  C-----------------------------------------------------------------
   54  C
   55            INTEGER IFSxG, IVCxSX, IVCxSY, IVCxSZ
   56  C
   57            REAL*8 RVCxCX, RVCxCY, RVCxCZ, RVCxXF, RVCxYF, RVCxZF
   58            REAL*8 RVCxH5, RVCxLA1, RVCxLO1, RVCxHBOT
   59            REAL*8 RVCxALPH, RVCxBETA, RVCxET1, RVCxET2, RVLxI(5), RVLxJ(5),
   60            REAL*8 RVLxK(5), RTLxC5, RTLxC6, RTLxC7, RTLxA2, RTLxA3
   61            REAL*8 RTLxA4, RTLxQ5, RTLxQ6, RTLxQ7, RTLxD6, RVCxD6
   62            REAL*8 RTCxA, RTCxB, RTCxC, RTCxD
   63  C
   64            COMMON /MORE/ IVCxSX, IVCxSY, IVCxSZ, RVCxCX, RVCxCY, RVCxCZ
   65            COMMON /OTHER/ RVCxLA1,RVCxLO1,RVCxHBOT,RTCxA,RTCxB,RTCxC
   66            COMMON /END/ RVCxXF, RVCxYF, RVCxZF, RVCxH5
   67            COMMON /IONO1/ RVCxALPH,RVCxBETA,RVCxET1,RVCxET2
   68            COMMON /GORP/ RVCxD6, RTCxD
   69  C
   70  10000     RTLxC5 = RVCxCX
   71            RTLxC6 = RVCxCY
   72            RTLxC7 = RVCxCZ
   73            RVLxI(4) = DSIN(RVCxLA1)
   74            RVLxI(5) = DCOS(RVCxLA1)
   75  C
   76  C         A routine to concoct the rotation matrix for the
   77  C           calculation of the new C values.
   78  C
   79            CALL FREESP(RVLxI, RVLxJ, RVLxK)
   80  C
   81  C         Recall that the flag IFSxG can take on values of 3, 4, & 5,
   82  C            corresponding to the occurrance of reflections in the
   83  C            z, x, & y directions, respectively.
   84  C
   85            IF (IFSxG.EQ.4) THEN
   86                RTLxA4 = 0.0D00
   87            ELSE
   88                RTLxA4 = IVCxSX*DSQRT(RTLxC5 - RVCxET1*RVCxXF)
   89            ENDIF
   90            IF (IFSxG.EQ.5) THEN
   91                RTLxA2 = 0.0D00
   92            ELSE
   93                RTLxA2 = IVCxSY*DSQRT(RTLxC6 - RVCxET2*RVCxYF)
   94            ENDIF
   95            IF (IFSxG.EQ.3) THEN
   96                RTLxA3 = 0.0D00
   97            ELSE
   98                RTLxA3 = IVCxSZ*DSQRT(RVCxBETA*RVCxZF*RVCxZF -
   99         *RVCxALPH*RVCxZF + RTLxC7)
  100            ENDIF
```

```
Line#  Source Line

  101  C
  102           RTLxQ5 = RVLxI(1)*RTLxA4 + RVLxI(2)*RTLxA2 + RVLxI(3)*RTLxA3
  103           RTLxQ6 = RVLxJ(1)*RTLxA4 + RVLxJ(2)*RTLxA2 + RVLxJ(3)*RTLxA3
  104           RTLxQ7 = RVLxK(1)*RTLxA4 + RVLxK(2)*RTLxA2 + RVLxK(3)*RTLxA3
  105           RTLxD6 = DSQRT(RVCxXF*RVCxXF + RVCxYF*RVCxYF + RVCxZF*RVCxZF)
  106           RVCxD6 = RTLxD6
  107           IF (IFSxG.NE.4) GO TO 10414
  108  C
  109  C        Begin handling for an x-reflection.
  110  C
  111           IF (INTSIGN(RTLxQ5).EQ.IVCxSX) GO TO 10410
  112           IVCxSX = -IVCxSX
  113           GO TO 10416
  114  C
  115  C        A limit is set to ensure that the calculation doesn't take
  116  C          an infinite time to creep up to a reflection point, so
  117  C          when the limit is reached, a reflection is forced.
  118  C
  119  10410    IF (RTLxD6.GT.1.0D-04) GO TO 10416
  120           IVCxSX = -IVCxSX
  121           RTLxQ5 = 0.0D00
  122           GO TO 10416
  123  10414    IVCxSX = INTSIGN(RTLxQ5)
  124           GO TO 10419
  125  C
  126  C        The x-reflection is complete, so handle updating the values
  127  C          in the y and z directions.
  128  C
  129  10416    IVCxSY = INTSIGN(RTLxQ6)
  130           IVCxSZ = INTSIGN(RTLxQ7)
  131           GO TO 10446
  132  10419    IF (IFSxG.NE.5) GO TO 10428
  133  C
  134  C        Begin handling for a y-reflection.
  135  C
  136           IF (INTSIGN(RTLxQ6).EQ.IVCxSY) GO TO 10424
  137           IVCxSY = -IVCxSY
  138           GO TO 10430
  139  C
  140  C        Again a limit is set to ensure that the calculation doesn't
  141  C          take an infinite time to creep up to a reflection point,
  142  C          so when the limit is reached, a reflection is forced.
  143  C
  144  10424    IF (RTLxD6.GT.1.0D-04) GO TO 10430
  145           IVCxSY = -IVCxSY
  146           RTLxQ6 = 0.0D00
  147           GO TO 10430
  148  10428    IVCxSY = INTSIGN(RTLxQ6)
  149           GO TO 10433
  150  C
```

```
Line#  Source Line

  151  C       The y-reflection is complete, so handle updating the values
  152  C         in the x and z directions.
  153  C
  154  10430   IVCxSX = INTSIGN(RTLxQ5)
  155          IVCxSZ = INTSIGN(RTLxQ7)
  156          GO TO 10446
  157  10433   IF (IFSxG.NE.3) GO TO 10442
  158  C
  159  C       Begin handling for a z-reflection.
  160  C
  161          IF (INTSIGN(RTLxQ7).EQ.IVCxSZ) GO TO 10438
  162  10436   IVCxSZ = -IVCxSZ
  163          GO TO 10444
  164  C
  165  C       Again a limit is set to ensure that the calculation doesn t
  166  C         take an infinite time to creep up to a reflection point.
  167  C         so when the limit is reached, a reflection is forced.
  168  C
  169  10438   IF (RTLxD6.GT.1.0D-04) GO TO 10444
  170          IVCxSZ = -IVCxSZ
  171          RTLxQ7 = 0.0D00
  172          GO TO 10444
  173  10442   IVCxSZ = INTSIGN(RTLxQ7)
  174          GO TO 10446
  175  C
  176  C       The z-reflection is complete, so handle updating the values
  177  C         in the x and y directions.
  178  C
  179  10444   IVCxSX = INTSIGN(RTLxQ5)
  180          IVCxSY = INTSIGN(RTLxQ6)
  181  10446   RVCxCX = RTLxQ5 * RTLxQ5
  182          RVCxCY = RTLxQ6 * RTLxQ6
  183          RVCxCZ = RTLxQ7 * RTLxQ7
  184  C
  185  C       IFSxG = 6 which means that the ray is outside the bounds
  186  C         of the ionosphere, headed down, so presumably the program
  187  C         is handling an earth bounce event.  Therefore, a specular
  188  C         reflection is forced.
  189  C
  190          IF (IFSxG.EQ.6.AND.RVCxH5.LT.RVCxHBOT) IVCxSZ = -IVCxSZ
  191          RETURN
  192          END


EWCS  Local Symbols

ame                          Class    Type              Size

FSXG . . . . . . . . . . . . param
TLXC5. . . . . . . . . . . . local    REAL*8               8
```

NEWCS Local Symbols

Name	Class	Type	Size
RTLXC6	local	REAL*8	8
RTLXD6	local	REAL*8	8
RTLXC7	local	REAL*8	8
RTLXQ5	local	REAL*8	8
RTLXQ6	local	REAL*8	8
RTLXQ7	local	REAL*8	8
RVLXI	local	REAL*8	40
RVLXJ	local	REAL*8	40
RVLXK	local	REAL*8	40
RTLXA2	local	REAL*8	8
RTLXA3	local	REAL*8	8
RTLXA4	local	REAL*8	8
IVCXSX	MORE	INTEGER*4	4
IVCXSY	MORE	INTEGER*4	4
IVCXSZ	MORE	INTEGER*4	4
RVCXCX	MORE	REAL*8	8
RVCXCY	MORE	REAL*8	8
RVCXCZ	MORE	REAL*8	8
RVCXXF	END	REAL*8	8
RVCXYF	END	REAL*8	8
RVCXZF	END	REAL*8	8
RVCXH5	END	REAL*8	8
RVCXLA1	OTHER	REAL*8	8
RVCXLO1	OTHER	REAL*8	8
RVCXHBOT	OTHER	REAL*8	8
RVCXALPH	IONO1	REAL*8	8
RVCXBETA	IONO1	REAL*8	8
RVCXET1	IONO1	REAL*8	8
RVCXET2	IONO1	REAL*8	8
RVCXD6	GORP	REAL*8	8
RTCXA	OTHER	REAL*8	8
RTCXB	OTHER	REAL*8	8
RTCXC	OTHER	REAL*8	8
RTCXD	GORP	REAL*8	8

```
Line#  Source Line

  194          SUBROUTINE TRIANG
  195  C
  196  C-----------------------------------------------------------------
  197  C
  198  C   TRIANG -- SUBROUTINE TO CHECK IF A LOCATION IS WITHIN A
  199  C                TRIANGLE OF S.C.P.'S
  200  C
  201  C         CALLED BY:       IONOPAR
  202  C
  203  C-----------------------------------------------------------------
  204  C
  205  C         AUTHOR:          MICHAEL H. REILLY & ERIC L. STROBEL
  206  C
  207  C         DATE:            02/08/88
  208  C
  209  C         VERSION:         2.2
  210  C
  211  C-----------------------------------------------------------------
  212  C
  213  C         REVISED:         07/25/86 -- INITIAL REVISION.  TRANSLATED
  214  C                          FROM TEKTRONIX BASIC TO VAX FORTRAN BY
  215  C                          ERIC L. STROBEL.
  216  C
  217  C                          07/30/86 -- V1.1.  Change over to use of
  218  C                          REAL*8 precision in the calculations.
  219  C
  220  C                          10/10/86 -- V2.0.  Changed to conform to
  221  C                          the change over to RAYTRACE as a subrout.
  222  C
  223  C                          09/01/87 -- V2.1.  Slight changes to speed
  224  C                          things up a bit.
  225  C
  226  C                          02/08/88 -- V2.2.  More streamlining of INTs
  227  C                          and NINTs.  Logical variable added to help
  228  C                          correctly handle the (possibly) worrisome
  229  C                          case of the grid at the geographic north pole.
  230  C
  231  C-----------------------------------------------------------------
  232  C
  233  C        USES:    IVSxSCT1,2,3     THE THREE S.C.P.'S THAT MAKE THE
  234  C                                            TRIANGLE
  235  C                 RVCxX,Y,Z        INTERMEDIATE COORDINATES
  236  C                 RVCxU,V,W(900)   THE S.C.P. COORDINATES
  237  C
  238  C
  239  C         TO DETERMINE WHETHER OR NOT A LOCATION IS WITHIN A
  240  C             TRIANGLE OF S.C.P.'S.  First, the locations  & array
  241  C             indices) of the four surrounding grid points are
  242  C             obtained, and then the potential triangles (involving
  243  C             three of the corners of this surrounding rectangle
```

```
Line#  Source Line

  244  C             are checked in an attempt to make sure that the point
  245  C             of interest is as far within a triangle as possible.
  246  C             The three grid points that make up this triangle are
  247  C             reported.  These are the points from which the
  248  C             interpolation is based.  Recall the following
  249  C             definitions: RVxGRID(1) = lat spacing
  250  C                                (2) = lon spacing
  251  C                                (3) = starting lat
  252  C                                (4) = starting lon
  253  C                                (5) = # in lat
  254  C                                (6) = # in lon.
  255  C
  256  C
  257  C         RETURNS:
  258  C                 IFSxN4           FLAG (=1 IF NOT INTERIOR TO A
  259  C                                       TRIANGLE OF S.C.P.'S)
  260  C
  261  C-----------------------------------------------------------------
  262  C
  263           INTEGER IVLxT1, IVLxT2, IVLxT3, IVLxT4, IVCxSCT1
  264           INTEGER IVCxSCT2, IVCxSCT3, IVCxSCS, IVLxNLN, IVLxILN
  265  C
  266           REAL*8 RVLxLAIN, RVLxLOIN, RVCxLAI, RVxGRID(6), RVCxLOI
  267           REAL*8 RVLxP1, RVLxP2, RVCxXI, RVCxYI, RVCxZI
  268  C
  269           LOGICAL LVLxPOLE
  270  C
  271           COMMON /MAINDAT/ RVxGRID
  272           COMMON /START/ RVCxXI, RVCxYI, RVCxZI, RVCxLAI, RVCxLOI
  273           COMMON /TEMP2/ IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3
  274  C
  275           RVLxLAIN = (RVCxLAI - RVxGRID(3)) / RVxGRID(1)
  276           RVLxLOIN = (RVCxLOI - RVxGRID(4)) / RVxGRID(2)
  277           IVLxILN = INT(RVLxLAIN)
  278           RVLxP1 = RVLxLAIN - IVLxILN
  279           RVLxP2 = RVLxLOIN - INT(RVLxLOIN)
  280           IVLxNLN = NINT(RVxGRID(6))
  281           IVLxT1 = IVLxILN * IVLxNLN + INT(RVLxLOIN) - 1
  282           IVLxT2 = IVLxT1 + 1
  283           IVLxT3 = IVLxT1 + IVLxNLN
  284           IVLxT4 = IVLxT3 + 1
  285           LVLxPOLE = NINT(RVxGRID(3) + (IVLxILN + 1) * RVxGRID(1)).EQ.90
  286           IF (RVLxP1.LT.0.5.OR.LVLxPOLE) THEN
  287               IVCxSCT1 = IVLxT1
  288               IVCxSCT2 = IVLxT2
  289               IF (RVLxP2.LT.0.5D00) THEN
  290                   IVCxSCT3 = IVLxT3
  291               ELSE
  292                   IVCxSCT3 = IVLxT4
  293               ENDIF
```

```
 Line#  Source Line

   294          ELSE
   295              IVCxSCT2 = IVLxT3
   296              IVCxSCT3 = IVLxT4
   297              IF (RVLxP2.LT.0.5D00) THEN
   298                  IVCxSCT1 = IVLxT1
   299              ELSE
   300                  IVCxSCT1 = IVLxT2
   301              ENDIF
   302          ENDIF
   303          RETURN
   304          END


TRIANG  Local Symbols

Name                          Class    Type                 Size

IVLXT1. . . . . . . . . . . . local    INTEGER*4               4
IVLXT2. . . . . . . . . . . . local    INTEGER*4               4
IVLXT3. . . . . . . . . . . . local    INTEGER*4               4
IVLXT4. . . . . . . . . . . . local    INTEGER*4               4
RVLXP1. . . . . . . . . . . . local    REAL*8                  8
RVLXP2. . . . . . . . . . . . local    REAL*8                  8
IVLXILN . . . . . . . . . . . local    INTEGER*4               4
IVLXNLN . . . . . . . . . . . local    INTEGER*4               4
RVLXLAIN. . . . . . . . . . . local    REAL*8                  8
LVLXPOLE. . . . . . . . . . . local    LOGICAL*4               4
RVLXLOIN. . . . . . . . . . . local    REAL*8                  8
IVCXSCT1. . . . . . . . . . . TEMP2    INTEGER*4               4
IVCXSCT2. . . . . . . . . . . TEMP2    INTEGER*4               4
IVCXSCT3. . . . . . . . . . . TEMP2    INTEGER*4               4
IVCXSCS . . . . . . . . . . . TEMP2    INTEGER*4               4
RVCXLAI . . . . . . . . . . . START    REAL*8                  8
RVXGRID . . . . . . . . . . . MAINDAT REAL*8                  48
RVCXLOI . . . . . . . . . . . START    REAL*8                  8
RVCXXI. . . . . . . . . . . . START    REAL*8                  8
RVCXYI. . . . . . . . . . . . START    REAL*8                  8
RVCXZI. . . . . . . . . . . . START    REAL*8                  8
```

```
Line#  Source Line

  307          SUBROUTINE ACCFSP(RVSxH0, RVSxS4, IFSxG)
  308  C
  309  C-----------------------------------------------------------------
  310  C
  311  C    ACCFSP -- SUBROUTINE TO SPEED UP CALCULATION OF FREE SPACE
  312  C                  PROPAGATION
  313  C
  314  C          CALLED BY: ENDPT
  315  C
  316  C-----------------------------------------------------------------
  317  C
  318  C          AUTHOR:           MICHAEL H. REILLY & ERIC L. STROBEL
  319  C
  320  C          DATE:             03/18/88
  321  C
  322  C          VERSION:          2.2
  323  C
  324  C-----------------------------------------------------------------
  325  C
  326  C          REVISED:          07/25/86 -- INITIAL REVISION.  TRANSLATED
  327  C                            FROM TEKTRONIX BASIC TO VAX FORTRAN BY
  328  C                            ERIC L. STROBEL.
  329  C
  330  C                            07/30/86 -- V1.1.  Change over to use of
  331  C                            REAL*8 precision in the calculations.
  332  C
  333  C                            09/01/87 -- V2.0.  Changed to allow ray to
  334  C                            be above the top of the ionosphere.
  335  C
  336  C                            12/08/87 -- V2.1.  A change has been made
  337  C                            to make sure that the ray stops at the
  338  C                            desired point.
  339  C
  340  C                            03/18/88 -- V2.2.  Minor changes made to
  341  C                            accommodate the new usage of angular range.
  342  C
  343  C-----------------------------------------------------------------
  344  C
  345  C        USES:    IVCxSX,SY,SZ     SIGN VARIABLES
  346  C                 RPCxRE           EARTH RADIUS
  347  C                 RVSxH0           CURRENT HEIGHT
  348  C                 RVCxCX,CY,CZ     C-VALUES
  349  C
  350  C          TO OBTAIN THE ENDPOINT OF A RAYPATH INCREMENT THAT IS
  351  C                   THROUGH FREESPACE.
  352  C
  353  C          RETURNS:
  354  C                 RVCxXF,YF,ZF     THE ENDPOINT COORDINATES
  355  C
  356  C-----------------------------------------------------------------
```

```
Line#  Source Line

  357  C
  358            INTEGER IVCxSX, IVCxSY, IVCxSZ, IVLxSSS, IVLxJ, IVLxI
  359            INTEGER IFSxG
  360  C
  361            REAL*8 RPCxRE, RVSxH0, RVCxCX, RVCxCY, RVCxCZ, RVLxHGT
  362            REAL*8 RVCxXF, RVCxYF, RVCxZF, RPCxPI, RTLxD, RVCxHCT
  363            REAL*8 RPCxDTR, RVCxH5, RTLxQ1, RTLxQ2, RTLxQ3, RPCxHTP
  364            REAL*8 RTLxT1, RTLxC, RTLxC2, RTLxRR(4), RVLxS,VLxRF
  365            REAL*8 RTLxL1,RTLxL2, RTLxL4,RTLxL5, RVCxHB,RTLxD1,RTLxD2
  366            REAL*8 RTLxD3, RVSxS4, RTLxZZZ, RVCxALIM, RTLxT2, RTLxT3
  367            REAL*8 RTLxRP, RTLxTTH, RTLxTHE, RTLxDEL, RTLxDP, RTLxS4
  368            REAL*8 RTLxSPR
  369  C
  370            COMMON /MORE/ IVCxSX, IVCxSY, IVCxSZ, RVCxCX, RVCxCY, RVCxCZ
  371            COMMON /PRAM/ RPCxPI, RPCxDTR, RPCxRE, RPCxHTP
  372            COMMON /OTHER/ RTLxL1,RTLxL2,RVCxHB,RTLxL4,RTLxL5,RVCxHCT
  373            COMMON /END/ RVCxXF, RVCxYF, RVCxZF, RVCxH5
  374            COMMON /GORP/ RTLxZZZ, RVCxALIM
  375  C
  376            IVLxI = 0
  377  C
  378  C         Calculate the ray direction cosines.
  379  C
  380            RTLxQ1 = IVCxSX*DSQRT(RVCxCX)
  381            RTLxQ2 = IVCxSY*DSQRT(RVCxCY)
  382            RTLxQ3 = IVCxSZ*DSQRT(RVCxCZ)
  383  C
  384  C         The calculation is the solution of the quadratic equation
  385  C            resulting from a straight line piercing a sphere.  First
  386  C            the radii of the pertinent spheres are obtained, based on
  387  C            whether the freespace propagation is above or below the
  388  C            bounds of the model ionosphere.  Based on the discriminant
  389  C            the proper solution is chosen.  The linear distance covered
  390  C            is calculated (RVLxS) and using the direction cosines, the
  391  C            final coordinate values are obtained.
  392  C
  393            RTLxT1 = RVSxH0 + RPCxRE
  394            RTLxC =  RTLxT1*RTLxQ3
  395            RTLxC2 = RTLxC*RTLxC
  396            IVLxSSS = 1
  397            IF (RVCxH5.GE.RPCxHTP) THEN
  398                IVLxJ = 3
  399                RTLxRR(3) = RPCxHTP
  400                RTLxRR(4) = RVCxHCT
  401            ELSE
  402                IVLxJ = 1
  403                RTLxRR(1) = 0.0D00
  404                RTLxRR(2) = RVCxHB
  405            ENDIF
  406            IF (IVCxSZ.LT.0) GO TO 10000
```

```
Line#  Source Line

  407   9000    IVLxJ = IVLxJ + 1
  408           IVLxSSS = -1
  409  10000    RTLxD = RTLxC2 + (RTLxRR(IVLxJ) - RVSxH0)*(RTLxT1 +
  410       #RTLxRR(IVLxJ) + RPCxRE)
  411           IF (RTLxD.LT.0.0D00) THEN
  412               PRINT *,' Ray passes through a minimum of altitude.'
  413               IFSxG = 7
  414               GO TO 9000
  415           ENDIF
  416           RVLxS = -RTLxC  - IVLxSSS * DSQRT(RTLxD)
  417  11000    RVCxXF = RTLxQ1*RVLxS
  418           RVCxYF = RTLxQ2*RVLxS
  419           RVCxZF = RTLxQ3*RVLxS
  420  C
  421  C        DONE 12/8.  After some preliminary calculations, a check
  422  C          is made to see if the straight line endpoint is beyond
  423  C          the range cutoff.  If so, the distance traveled along
  424  C          the line is backed off by an appropriate amount and the
  425  C          endpoint coordinates are recalculated.
  426  C
  427           RTLxT2 = RVCxZF + RTLxT1
  428           RTLxT3 = RVCxXF * RVCxXF + RVCxYF * RVCxYF
  429           RTLxRP = DSQRT(RTLxT3 + RTLxT2*RTLxT2)
  430           RTLxTTH = RVCxZF / DSQRT(RTLxT3)
  431           RTLxTTH = DATAN(RTLxTTH)
  432           RTLxTHE = RPCxPI / 2.0 + RTLxTTH
  433           RTLxDEL = RVLxS * DSIN(RTLxTHE) / RTLxRP
  434           RTLxDEL = DASIN(RTLxDEL)
  435           IF ((RVSxS4 + RTLxDEL).GE.RVCxALIM) THEN
  436               RTLxDP = RVCxALIM - RVSxS4
  437               RTLxSPR = RVLxS * RTLxT1 * DSIN(RTLxDP)
  438               RTLxSPR = RTLxSPR / (RTLxRP * DSIN(RTLxDEL - RTLxDP)
  439       # + RTLxT1 * DSIN(RTLxDP))
  440               RVCxXF = RTLxQ1*RTLxSPR
  441               RVCxYF = RTLxQ2*RTLxSPR
  442               RVCxZF = RTLxQ3*RTLxSPR
  443               RTLxT2 = RVCxZF + RTLxT1
  444               RTLxT3 = RVCxXF * RVCxXF + RVCxYF * RVCxYF
  445               RTLxRP = DSQRT(RTLxT3 + RTLxT2*RTLxT2)
  446               IF (DABS(RTLxRP-RPCxRE).GT.1.0D-02) THEN
  447                   IFSxG = 7
  448               ENDIF
  449           ENDIF
  450  C
  451           RETURN
  452           END
```

ACCFSP Local Symbols

Name	Class	Type	Size
IFSXG	param		
RVSXS4	param		
RVSXH0	param		
RTLXQ1	local	REAL*8	8
IVLXI	local	INTEGER*4	4
RTLXC	local	REAL*8	8
RTLXQ2	local	REAL*8	8
IVLXJ	local	INTEGER*4	4
RTLXQ3	local	REAL*8	8
RTLXD	local	REAL*8	8
RTLXT1	local	REAL*8	8
RTLXT2	local	REAL*8	8
RTLXT3	local	REAL*8	8
RTLXDP	local	REAL*8	8
RTLXDEL	local	REAL*8	8
RVLXS	local	REAL*8	8
RTLXTHE	local	REAL*8	8
RTLXRP	local	REAL*8	8
RTLXRR	local	REAL*8	32
RTLXTTH	local	REAL*8	8
IVLXSSS	local	INTEGER*4	4
RTLXSPR	local	REAL*8	8
RTLXC2	local	REAL*8	8
IVCXSX	MORE	INTEGER*4	4
IVCXSY	MORE	INTEGER*4	4
IVCXSZ	MORE	INTEGER*4	4
PCXRE	PRAM	REAL*8	8
VCXCX	MORE	REAL*8	8
VCXCY	MORE	REAL*8	8
VCXCZ	MORE	REAL*8	8
VCXXF	END	REAL*8	8
VCXYF	END	REAL*8	8
VCXZF	END	REAL*8	8
PCXPI	PRAM	REAL*8	8
VCXHCT	OTHER	REAL*8	8
PCXDTR	PRAM	REAL*8	8
VCXH5	END	REAL*8	8
PCXHTP	PRAM	REAL*8	8
TLXL1	OTHER	REAL*8	8
TLXL2	OTHER	REAL*8	8
TLXL4	OTHER	REAL*8	8
TLXL5	OTHER	REAL*8	8
VCXHB	OTHER	REAL*8	8
TLXZZZ	GORP	REAL*8	8
VCXALIM	GORP	REAL*8	8

```
Line#  Source Line

  455          SUBROUTINE FREESP(RVSxI, RVSxJ, RVSxK)
  456  C
  457  C-----------------------------------------------------------------
  458  C
  459  C    FREESP -- SUBROUTINE TO OBTAIN SOME VALUES USED TO COMPUTE
  460  C                 NEW C-VALUES IN THE CASE OF FREESPACE
  461  C                 PROPAGATION.
  462  C
  463  C          CALLED BY: NEWCS
  464  C
  465  C-----------------------------------------------------------------
  466  C
  467  C          AUTHOR:           MICHAEL H. REILLY
  468  C
  469  C          DATE:             07/30/86
  470  C
  471  C          VERSION:          1.1
  472  C
  473  C-----------------------------------------------------------------
  474  C
  475  C          REVISED:          07/25/86 -- INITIAL REVISION.  TRANSLATED
  476  C                            FROM TEKTRONIX BASIC TO VAX FORTRAN BY
  477  C                            ERIC L. STROBEL.
  478  C
  479  C                            07/30/86 -- V1.1.  Change over to use of
  480  C                            REAL*8 precision in the calculations.
  481  C
  482  C-----------------------------------------------------------------
  483  C
  484  C         USES:    RVCxLO1          PREVIOUS POINT'S LON
  485  C                  RVCxLAI          LAT OF BEGINNING OF INTERVAL
  486  C                  RVCxLOI          LON OF BEGINNING OF INTERVAL
  487  C                  RVSxI(5)         X
  488  C                  RVSxJ(5)         X-ARRAYS OF INTERMEDIATE VALUES
  489  C                  RVSxK(5)         X
  490  C
  491  C          TO CALCULATE SOME MORE INTERMEDIATE VALUES INVOLVED IN
  492  C                   COMPUTING THE C-VALUES FOR FREESPACE
  493  C                   PROPAGATION
  494  C
  495  C          RETURNS:
  496  C                  RVSxI(5)         X
  497  C                  RVSxJ(5)         X-ARRAYS OF INTERMEDIATE VALUES
  498  C                  RVSxK(5)         X
  499  C
  500  C-----------------------------------------------------------------
  501  C
  502          REAL*8 RVSxI(5), RVSxJ(5), RVSxK(5), RVCxXI, RVCxYI, RVCxZI
  503          REAL*8 RVCxLAI, RVCxLOI, RVCxL1, RVCxLO1, RTCxA, RTCxB
  504          REAL*8 RTCxC, RTCxD
```

```
Line#  Source Line

   505  C
   506           COMMON /START/ RVCxXI, RVCxYI, RVCxZI, RVCxLAI, RVCxLOI
   507           COMMON /OTHER/ RVCxL1,RVCxLO1,RTCxA,RTCxB,RTCxC,RTCxD
   508  C
   509           RVSxJ(4) = DSIN(RVCxLAI)
   510           RVSxJ(5) = DCOS(RVCxLAI)
   511           RVSxK(4) = DSIN(RVCxLOI - RVCxLO1)
   512           RVSxK(5) = DCOS(RVCxLOI - RVCxLO1)
   513           RVSxI(1) = RVSxI(4)*RVSxJ(4)*RVSxK(5) + RVSxI(5)*RVSxJ(5)
   514           RVSxI(2) = RVSxJ(4)*RVSxK(4)
   515           RVSxI(3) = RVSxI(5)*RVSxJ(4)*RVSxK(5) - RVSxI(4)*RVSxJ(5)
   516           RVSxJ(1) = -RVSxI(4)*RVSxK(4)
   517           RVSxJ(2) = RVSxK(5)
   518           RVSxJ(3) = -RVSxI(5)*RVSxK(4)
   519           RVSxK(1) = RVSxI(4)*RVSxJ(5)*RVSxK(5) - RVSxI(5)*RVSxJ(4)
   520           RVSxK(2) = RVSxJ(5)*RVSxK(4)
   521           RVSxK(3) = RVSxI(5)*RVSxJ(5)*RVSxK(5) + RVSxI(4)*RVSxJ(4)
   522           RETURN
   523           END


FREESP  Local Symbols

Name                            Class    Type                Size

RVSXK . . . . . . . . . . . . . param
RVSXJ . . . . . . . . . . . . . param
RVSXI . . . . . . . . . . . . . param
RVCXXI. . . . . . . . . . . . . START    REAL*8                 8
RVCXYI. . . . . . . . . . . . . START    REAL*8                 8
RVCXZI. . . . . . . . . . . . . START    REAL*8                 8
RVCXLAI . . . . . . . . . . . . START    REAL*8                 8
RVCXLOI . . . . . . . . . . . . START    REAL*8                 8
RVCXL1. . . . . . . . . . . . . OTHER    REAL*8                 8
RVCXLO1 . . . . . . . . . . . . OTHER    REAL*8                 8
RTCXA . . . . . . . . . . . . . OTHER    REAL*8                 8
RTCXB . . . . . . . . . . . . . OTHER    REAL*8                 8
RTCXC . . . . . . . . . . . . . OTHER    REAL*8                 8
RTCXD . . . . . . . . . . . . . OTHER    REAL*8                 8
```

```
Line#  Source Line

  526          SUBROUTINE TIMES(RVSxBUN, IVSxNB, RVSxLON0)
  527  C
  528  C-------------------------------------------------------------
  529  C
  530  C   TIMES -- SUBROUTINE TO CONVERT GIVEN TIME INTO THE TIME
  531  C                VARIABLES NEEDED LATER
  532  C
  533  C        CALLED BY: RAYSUB
  534  C
  535  C-------------------------------------------------------------
  536  C
  537  C        AUTHOR:          ERIC L. STROBEL
  538  C
  539  C        DATE:            03/18/88
  540  C
  541  C        VERSION:         2.0
  542  C
  543  C-------------------------------------------------------------
  544  C
  545  C        REVISED:         08/07/86 -- INITIAL REVISION.
  546  C
  547  C                         10/10/86 -- V1.1.  Changed to conform to
  548  C                         the status of RAYTRACE as a subroutine.
  549  C
  550  C                         03/18/88 -- V2.0.  Vastly rewritten to give
  551  C                         a better idea of the mid-hop longitude, and
  552  C                         to better handle being off by factors of
  553  C                         (2 * pi) in the mid-hop longitude.
  554  C
  555  C-------------------------------------------------------------
  556  C
  557  C       USES:   RVSxBUN(11,15)          ARRAY OF BOUNCE PT. DATA
  558  C               IVSxNB                  BOUNCE COUNT
  559  C               RVSxLON0                LONG. OF LAUNCH PT.
  560  C               ICLxM(11)               NUMBER OF DAYS IN EACH MO.
  561  C               IVCxTIM(5)              DATE AND TIME ARRAY
  562  C
  563  C        TO CALCULATE THE DECIMAL YEAR VALUE AND THE TIME SINCE
  564  C                MIDNIGHT IN RADIANS
  565  C
  566  C        RETURNS:
  567  C               RVCxYR          DECIMAL YEAR
  568  C               RVCxTIM         TIME SINCE MIDNIGHT IN RADIANS
  569  C
  570  C-------------------------------------------------------------
  571  C
  572          REAL*8 RVCxYR, RVCxTIM, RVSxBUN(11,15), RPCxPI, RPCxDTOR
  573          REAL*8 RVCxA, RTLxLON, RVSxLON0, RTCxA, RTCxB, RTLxLO
  574          REAL*8 RTLxL1, RTLxDEL
  575  C
```

```
Line#  Source Line

   576           INTEGER ICLxM(11), IVCxTIM(5), ITLxA, ITLxB, IVSxNB
   577           INTEGER IVCxSSN, IFCxGND, IVLxNR
   578   C
   579           COMMON /PRAM/ RPCxPI, RPCxDTOR, RTCxA, RTCxB
   580           COMMON /LPARM/ IVCxSSN, IVCxTIM, RVCxYR, RVCxA, IFCxGND,
   581         *RVCxTIM
   582   C
   583           DATA ICLxM/31,28,31,30,31,30,31,31,30,31,30/
   584   C
   585   10000   ITLxA = IVCxTIM(2) - 1
   586           IVLxNR = 1
   587           ITLxB = IVCxTIM(3)
   588           DO 10100 I = 1, ITLxA
   589               ITLxB = ITLxB + ICLxM(I)
   590   10100   CONTINUE
   591           RVCxYR = IVCxTIM(1) - ITLxB/365.0D00 - 1980.0D00
   592           RVCxTIM = IVCxTIM(4) + IVCxTIM(5)/60.0D00
   593   C
   594   C       Compute the longitude of the midpoint of a hop, for the
   595   C         absorption calculation.
   596   C
   597           IF (IVSxNB.GT.1) THEN
   598               RTLxL0 = RVSxBUN(IVSxNB-1,2)
   599           ELSE
   600               RTLxL0 = RVSxLON0
   601           ENDIF
   602           RTLxL1 = RVSxBUN(IVSxNB,2)
   603           RTLxDEL = RTLxL1 - RTLxL0
   604           IF (DABS(RTLxDEL).GT.RPCxPI) RTLxDEL = RTLxDEL-INTSIGN(RTLxDEL
   605         *(2.0D00 * RPCxPI)
   606           RTLxLON = RTLxL0 + RTLxDEL/2.0D00
   607   C
   608           RVCxTIM = RTLxLON + 15.0D00 * RPCxDTOR * RVCxTIM
   609   10200   IF (RVCxTIM.GE.(2.0D00*RPCxPI)) THEN
   610               RVCxTIM = RVCxTIM - 2.0D00*RPCxPI
   611               GO TO 10200
   612           ELSE IF (RVCxTIM.LT.0.0D00) THEN
   613               RVCxTIM = RVCxTIM + 2.0D00*RPCxPI
   614               GO TO 10200
   615           ENDIF
   616           RETURN
   617           END


TIMES  Local Symbols

Name                          Class    Type              Size

RVSXLON0. . . . . . . . . . . param
IVSXNB. . . . . . . . . . . . param
```

TIMES Local Symbols

Name	Class	Type	Size
RVSXBUN	param		
ICLXM	local	INTEGER*4	44
ITLXA	local	INTEGER*4	4
ITLXB	local	INTEGER*4	4
RTLXL0	local	REAL*8	8
RTLXL1	local	REAL*8	8
I	local	INTEGER*4	4
RTLXDEL	local	REAL*8	8
IVLXNR	local	INTEGER*4	4
RTLXLON	local	REAL*8	8
RVCXYR	LPARM	REAL*8	8
RVCXTIM	LPARM	REAL*8	8
RPCXPI	PRAM	REAL*8	8
RPCXDTOR	PRAM	REAL*8	8
RVCXA	LPARM	REAL*8	8
RTCXA	PRAM	REAL*8	8
RTCXB	PRAM	REAL*8	8
IVCXTIM	LPARM	INTEGER*4	20
IVCXSSN	LPARM	INTEGER*4	4
IFCXGND	LPARM	INTEGER*4	4

```
Line#  Source Line

  620          SUBROUTINE LOSS(RVSxBOU,RVSxBUN,IVSxNB,RVSxLAT0,RVSxLON0
  621         *,RVSxEL0,IFSxEND)
  622   C
  623   C-----------------------------------------------------------------
  624   C
  625   C    LOSS -- SUBROUTINE TO EVALUATE THE LOSSES ALONG THE RAY'S
  626   C                 PATH
  627   C
  628   C         CALLED BY: RAYSUB
  629   C
  630   C-----------------------------------------------------------------
  631   C
  632   C         AUTHOR:          ERIC L. STROBEL
  633   C
  634   C         DATE:            09/01/87
  635   C
  636   C         VERSION:         3.0
  637   C
  638   C-----------------------------------------------------------------
  639   C
  640   C         REVISED:         08/07/86 -- INITIAL REVISION.
  641   C
  642   C                          09/05/86 -- V2.0.  ADD HORIZON EFFECTS.
  643   C                          BUG FIX.  COSMETIC CHANGES.
  644   C
  645   C                          10/10/86 -- V2.1.  Changed to conform to
  646   C                          the new status of RAYTRACE as a subrout.
  647   C                          Also trimmed some of the commented out
  648   C                          code.
  649   C
  650   C                          04/09/87 -- V2.2.  Corrected error in
  651   C                          calculation of geometric loss.
  652   C
  653   C                          09/01/87 -- V3.0.  Changed to comment out
  654   C                          absorption loss calculation in anticipation
  655   C                          of a better routine.  Also, the kludgy
  656   C                          "excess loss" has been commented out.  The
  657   C                          focusing calculation has been generalized
  658   C                          and some corrections have been made.
  659   C
  660   C-----------------------------------------------------------------
  661   C
  662   C        USES:    RVSxBOU(11,15)          BOUNCE PT. ARRAY
  663   C                 RVSxBUN(4,3,11)         RAY BUNDLE ARRAY
  664   C                 IVSxNB                  BOUNCE COUNT
  665   C                 RVSxLAT0                LAT OF LAUNCH POINT
  666   C                 RVSxLON0                LON OF LAUNCH POINT
  667   C                 IFSxEND                 FLAGS THE LOSS COMP. FOR
  668   C                                            END OF THE RAY
  669   C                 RVCxFSQ                 WAVE FREQUENCY SQUARED
```

```
Line#  Source Line

  670  C                IVCxSSN                    SUNSPOT NUMBER
  671  C                IVCxTIM(5)                 DATE AND TIME ARRAY
  672  C                RVCxYEAR                   DECIMAL YEAR
  673  C                RVCxA100                   ANGLE OF INCIDENCE AT
  674  C                                              100 KM ALTITUDE
  675  C                IFCxGND                    TYPE OF REFLECTION SURF.
  676  C                RVCxTIM                    TIME SINCE MIDNIGHT IN
  677  C                                              RADIANS
  678  C
  679  C         TO CALCULATE THE SIGNAL LOSS CONTRIBUTIONS FROM:
  680  C                 ABSORPTION
  681  C                 REFLECTION
  682  C                 OTHER PROCESSES
  683  C                 GEOMETRIC FOCUSING/DEFOCUSING
  684  C
  685  C         RETURNS:
  686  C                 RVCxLOSA                   ABSORPTION LOSS
  687  C                 RVCxLOSR                   REFLECTION LOSS
  688  C                 RVCxLOSX                   EXCESS LOSS
  689  C                 RVCxLOSG                   GEOMETRIC LOSS
  690  C
  691  C-----------------------------------------------------------------
  692  C
  693           REAL*8 RTLxK1, RVLxMOT, RVCxYEAR, RVLxDEC, RVLxLAT
  694           REAL*8 RVSxBUN(4,3,11), RVSxLAT0, RVSxLON0, RVLxZEN
  695           REAL*8 RVLxZ12, RVCxA100, RVLxW, RVLxM, RVLxK, RTLxA
  696           REAL*8 RTLxB, RTLxC, RTLxDEN, RVCxLOSA, RTLxRE, RTLxIM
  697           REAL*8 RTLxARG, RVCxLOSR, RCTxA2, RCTxB2, RCTxC2, RVLxELEV
  698           REAL*8 RCTxA3, RCTxB3, RCTxC3, RTLxGC, RTLxGP, RVLxLON
  699           REAL*8 RTLxGLA, RTLxGLO, RTLxDLA, RTLxDLO, RVLxMC
  700           REAL*8 RVLxGML, RVCxLOSX, RCLxAU, RTLxR, RTLxS(3,3)
  701           REAL*8 RTLxV1(3), RTLxV2(3), RTLxDE, RTLxCT(3),RTLxCV(3)
  702           REAL*8 RVLxAREA, RTLxT(3), RTLxCRS(3), RTLxMAG1, RTLxMAG2
  703           REAL*8 RVCxLOSG, RPCxPI, RPCxDTOR, RPCxRE, RVSxBCU(11,15)
  704           REAL*8 RVCxII, RVCxJJ, RVCxKK, RVCxFSQ, RVLxFRE
  705           REAL*8 RVCxTIM, RVLxHOR  RVCxCY, RVCxCZ, RVSxEL0
  706           REAL*8 RVCxSEZTX(3,3), RTCx1, RTCx2, RTCx3, RTCx4, RTCx5
  707           REAL*8 RTCx6, RTCx7, RTCx8, RVCxHMIN, RVCxH5, RVCxCX
  708           REAL*8 RTCxA, RTCxB, RTCxC
  709  C
  710           COMPLEX*16 CTLxNSQ, CTLxCSQ, CTLxSIN, CTLxSQT
  711           COMPLEX*16 CVLxRH, CVLxRV
  712  C
  713           INTEGER IVCxSX, IVCxSY, IVCxSZ
  714           INTEGER IVSxNB, IVCxTIM(5), IFCxGND, IVCxSSN, IFSxEND
  715  C
  716           COMMON /MISC/ RVCxSEZTX,RTCx1,RTCx2,RTCx3,RTCx4,RTCx5,
  717         #RVCxHMIN
  718           COMMON /MORE/ IVCxSX IVCxSY,IVCxSZ,RVCxCX,RVCxCY,RVCxCZ
  719           COMMON /END/ RTCx6,RTCx7,RTCx8,RVCxH5
```

```
Line#  Source Line

  720           COMMON /LOCAL/ RVLxELEV
  721           COMMON /PRAM/ RPCxPI, RPCxDTOR, RPCxRE, RTCxC
  722           COMMON /OTHER/ RVCxII,RVCxJJ,RVCxKK,RVCxFSQ,RTCxA,RTCxB
  723           COMMON /LPARM/ IVCxSSN,IVCxTIM,RVCxYEAR,RVCxA100,IFCxGND
  724          *,RVCxTIM
  725           COMMON /LOSSES/ RVCxLOSA, RVCxLOSR, RVCxLOSX, RVCxLOSG
  726  C
  727  C    ABSORPTION LOSS CALC (ICLxI IS A TEMPORARY VBL. TO DENOTE
  728  C                                  WHICH RAY IS BEING DONE)
  729  C
  730  C10000   IF (IFSxEND.EQ.1.AND.IVSxNB.EQ.1.AND.RVCxHMIN.GT.80.0)
  731  C       #GO TO 20000
  732  C         IF (IFSxEND.EQ.1.AND.IVSxNB.GT.1.AND.RVCxH5.LT.60.0)
  733  C       #GO TO 20000
  734  C         RTLxK1 = 0.0D00
  735  C         RVLxMOT = (( RVCxYEAR - JIDINT(RVCxYEAR))*365.0D00 + 16.0D00)
  736  C       *30.41667D00
  737  C         RVLxDEC = 0.398D00*DSIN(RPCxPI*(RVLxMOT - 3.17D00)/6.0D00)
  738  C         RVLxDEC = DASIN(RVLxDEC)
  739  C         RVLxLAT = ((2.0D00*IVSxNB - 1.0D00)/(2.0D00*IVSxNB)) *
  740  C       *(RVSxBOU(IVSxNB,1) - RVSxLAT0) + RVSxLAT0
  741  C         RVLxZEN = DSIN(RVLxLAT)*DSIN(RVLxDEC) - DCOS(RVLxLAT)*DCOS(RVLx
  742  C       *DEC) * DCOS(RVCxTIM)
  743  C         RVLxZEN = DACOS(RVLxZEN)
  744  C         RVLxZ12 = RVLxLAT - RVLxDEC
  745  C         RVLxDEC = RVLxDEC / RPCxDTOR
  746  C         RVLxLAT = RVLxLAT / RPCxDTOR
  747  C         RVLxZEN = RVLxZEN / RPCxDTOR
  748  C         RVLxZ12 = RVLxZ12 / RPCxDTOR
  749  C         RVCxA100 = RVCxA100 / RPCxDTOR
  750  C         IF (RVLxLAT.GE.30.0D00) THEN
  751  C             IF (IVCxTIM(2).EQ.11) THEN
  752  C                 RTLxK1 = 0.0083D00
  753  C             ELSE IF (IVCxTIM(2).EQ.12) THEN
  754  C                 RTLxK1 = 0.0282D00
  755  C             ELSE IF (IVCxTIM(2).EQ.1) THEN
  756  C                 RTLxK1 = 0.0269D00
  757  C             ELSE IF (IVCxTIM(2).EQ.2) THEN
  758  C                 RTLxK1 = 0.0089D00
  759  C             ENDIF
  760  C         ENDIF
  761  C         RVLxW = 30.0D00 - DABS(RVLxLAT - 60.0D00)
  762  C         RVLxW = 1.0D00 - RTLxK1*RVLxW
  763  C         RVLxM = 2.25D00 - 0.032D00*RVLxLAT
  764  C         IF (RVLxZEN.GT.90.0D00) THEN
  765  C             RVLxK = 0.01D00
  766  C         ELSE
  767  C             RTLxA = DCOSD(0.393D00*RVLxZEN)
  768  C             RTLxB = DCOSD(0.893D00*RVLxZ12)
  769  C             RTLxC = DCOSD(RVLxZ12)
```

```
Line#  Source Line

  770  C            RTLxC = RTLxC ** RVLxM
  771  C            RVLxK = (1.0D00 + 0.005D00*IVCxSSN) * RTLxC * RTLxA / RTLxE
  772  C         ENDIF
  773               RVLxFRE = DSQRT(RVCxFSQ)
  774  C         RTLxDEN = 10.2D00 + (RVLxFRE + 1.4D00)*(RVLxFRE + 1.4D00)
  775  C         RTLxDEN = RTLxDEN * DCOSD(RVCxA100)
  776  C         RVCxLOSA = RVCxLOSA + (286.0D00*(1.0D00+0.0087D00*RVLxLAT)*
  777  C      *RVLxW*RVLxK / RTLxDEN)
  778  C
  779  C    REFLECTION LOSS CALCULATION, a Fresnel coefficient calculation.
  780  C       For details, see the Radio Science paper referred to in the
  781  C       documentation.
  782  C
  783  20000    IF (IVSxNB-1.LE.0) GO TO 40000
  784               IF (IFCxGND.EQ.1) THEN
  785                   RTLxRE = 7.0D00
  786                   RTLxIM = -0.005D00 * 18000.0D00 / RVLxFRE
  787                   CTLxNSQ = DCMPLX(RTLxRE, RTLxIM)
  788               ELSE
  789                   RTLxRE = 80.0D00
  790                   RTLxIM = -5.0D00 * 18000.0D00 / RVLxFRE
  791                   CTLxNSQ = DCMPLX(RTLxRE, RTLxIM)
  792               ENDIF
  793               CTLxCSQ = DCMPLX(DCOS(RVLxELEV)*DCOS(RVLxELEV))
  794               CTLxSIN = DCMPLX(DSIN(RVLxELEV))
  795               CTLxSQT = CDSQRT(CTLxNSQ - CTLxCSQ)
  796               CVLxRH = (CTLxSIN - CTLxSQT)/(CTLxSIN + CTLxSQT)
  797               CVLxRV = (CTLxNSQ*CTLxSIN-CTLxSQT)/(CTLxNSQ*CTLxSIN+CTLxSQT
  798               RTLxARG = (CDABS(CVLxRV)*CDABS(CVLxRV) + CDABS(CVLxRH)*CDABS
  799            *CVLxRH)) / 2.0D00
  800               RVCxLOSR = RVCxLOSR + DABS(10.0D00*DLOG10(RTLxARG))
  801  C
  802  C    EXCESS LOSS CALCULATION
  803  C
  804  C30000   RCTxA2 = 11.18959D00
  805  C        RCTxB2 = -0.02916595D00
  806  C        RCTxC2 = -8.5401D-04
  807  C        RCTxA3 = -70.74865D00
  808  C        RCTxB3 = -0.04418982D00
  809  C        RCTxC3 = 0.002169971D00
  810  C        RTLxGC = RCTxA2 + RVCxYEAR*(RCTxB2 + RVCxYEAR*RCTxC2)
  811  C        RTLxGP = RCTxA3 + RVCxYEAR*(RCTxB3 + RVCxYEAR*RCTxC3
  812  C        RVLxLON = ((2.0D00*IVSxNB - 1.0D00)/(2.0D00*IVSxNB)   *
  813  C      *(RVSxBOU(IVSxNB,2) - RVSxLON0) + RVSxLON0
  814  C        RTLxGLA = (90.0D00 - RTLxGC) * RPCxDTOR
  815  C        RTLxGLO = RTLxGP * RPCxDTOR
  816  C        RTLxDLA = RTLxGLA - RVLxLAT*RPCxDTOR
  817  C        RTLxDLO = RTLxGLO - RVLxLON
  818  C        RVLxMC = DCOS(RTLxDLA) * DCOS(RTLxDLO)
  819  C        RVLxMC = DACOS(RVLxMC)
```

```
Line#  Source Line

  820  C        RVLxGML = (RPCxPI/2.0D00 - RVLxMC) / RPCxDTOR
  821  C        IF (RVLxGML.GT.60.0D00) THEN
  822  C            RVCxLOSX = 13.8D00
  823  C        ELSE
  824  C            RVCxLOSX = 9.8D00
  825  C        ENDIF
  826  C
  827  C    GEOMETRIC LOSS CALCULATION
  828  C
  829  40000    RCLxAU = 1.903858875D-09*DCOS(RVSxEL0)
  830  C
  831  C        RCLxAU is the cross-sectional area of a bundle of rays
  832  C          bounded by a square defined by azimuth and elevation
  833  C          deviations of 0.0025 degrees from one corner.  This
  834  C          cross-section is taken at a distance of 1 km from the
  835  C          starting point of the problem.  Only three rays are
  836  C          necessary to define the area, the area may be taken
  837  C          as represented by the vector cross product of the
  838  C          vector from the 'corner' point to one of the other
  839  C          points, with the other similarly taken vector.  This
  840  C          is in fact how the calculation is done.
  841  C
  842           DO 40100 I = 1,3
  843  C
  844  C        RVSxBUN contains the endpoint locations for the rays in
  845  C          the bundle.
  846  C
  847             RTLxR = RPCxRE - RVSxBUN(I,3,IVSxNB)
  848  C
  849  C        The RTLxS are the GEC coordinate values for the endpoints
  850  C          of the three rays that define the area.
  851  C
  852             RTLxS(I,1) = RTLxR*DCOS(RVSxBUN(I,1,IVSxNB))*DCOS(RVSxBUN
  853       #(I,2,IVSxNB))
  854             RTLxS(I,2) = RTLxR*DCOS(RVSxBUN(I,1,IVSxNB))*DSIN(RVSxBUN
  855       #(I,2,IVSxNB))
  856             RTLxS(I,3) = RTLxR*DSIN(RVSxBUN(I,1,IVSxNB))
  857  40100    CONTINUE
  858  C
  859  C        These next two quantities are the vectors (in GEC components
  860  C          representing the deviation of two of the points from the
  861  C          third (the 'corner').
  862  C
  863           DO 40200 I = 1,3
  864             RTLxV1(I) = RTLxS(1,I) - RTLxS(2,I)
  865             RTLxV2(I) = RTLxS(3,I) - RTLxS(2,I)
  866  40200    CONTINUE
  867           RTLxDE = SQRT(RVCxCX+RVCxCY+RVCxCZ)
  868  C
  869  C        This next vector is the ray direction in SEC components.
```

```
Line#  Source Line

  870  C
  871           RTLxCT(1) = IVCxSX * DSQRT(RVCxCX) / RTLxDE
  872           RTLxCT(2) = IVCxSY * DSQRT(RVCxCY) / RTLxDE
  873           RTLxCT(3) = IVCxSZ * DSQRT(RVCxCZ) / RTLxDE
  874  C
  875  C        Next, the ray direction vector is transformed into GEC
  876  C          coordinates.
  877  C
  878           DO 40300 I = 1,3
  879               RTLxCV(I) = 0.0
  880               DO 40250 J = 1,3
  881                   RTLxCV(I) = RTLxCV(I) + RVCxSEZTX(I,J)*RTLxCT(J)
  882  40250        CONTINUE
  883  40300    CONTINUE
  884           RVLxAREA = 0.0
  885  C
  886  C        Now, the triple vector product is calculated which will
  887  C          give the cross-sectional area of the ray bundle in the
  888  C          plane defined by the ray direction vector.
  889  C
  890           DO 40400 I = 1,3
  891               RTLxT(I) = RTLxV2(MOD(I,3)+1)*RTLxCV(MOD(I+1,3)+1)
  892       # - RTLxV2(MOD(I+1,3)+1)*RTLxCV(MOD(I,3)+1)
  893               RVLxAREA = RVLxAREA + RTLxT(I)*RTLxV1(I)
  894  40400    CONTINUE
  895  C
  896  C        A calculation now begins to obtain the angle between
  897  C          the ray direction and the surface determined by the
  898  C          cutoff / earth bounce condition.
  899  C
  900           RVLxAREA = DABS(RVLxAREA)
  901           RTLxMAG1 = 0.0
  902           RTLxMAG2 = 0.0
  903           DO 40500 I = 1,3
  904               RTLxCRS(I) = RTLxV1(MOD(I,3)+1)*RTLxV2(MOD(I+1,3)+1
  905       # - RTLxV1(MOD(I+1,3)+1)*RTLxV2(MOD(I,3)+1)
  906               RTLxMAG1 = RTLxMAG1 + RTLxCRS(I)*RTLxCRS(I)
  907               RTLxMAG2 = RTLxMAG2 + RTLxCV(I)*RTLxCV(I)
  908  40500    CONTINUE
  909           RTLxMAG1 = DSQRT(RTLxMAG1)
  910           RTLxMAG2 = DSQRT(RTLxMAG2)
  911           RVLxELEV = RVLxAREA / (RTLxMAG1*RTLxMAG2)
  912           RVLxELEV = DACOS(RVLxELEV)
  913           RVLxELEV = (RPCxPI/2.0) - RVLxELEV
  914  C
  915  C        The ordinary geometric loss calculation.
  916  C
  917           RVCxLOSG = 60.0D00 + 10.0D00 * DLOG10(RVLxAREA/RCLxAU
  918  C
  919  C        If a ray came into the earth at a sufficiently small angle
```

```
Line#  Source Line

   920  C          then the empirical horizon focusing calculation is done.
   921  C          The result of this is then compared to the normal focusing
   922  C          result and the greater loss is used.
   923  C
   924             IF (ABS(RVLxELEV/RPCxDTOR).LT.1.0D00) THEN
   925                 RVLxHOR = 60.012D00 + 20.0D00*DLOG10(DBLE(IVSxNB))/3.0D00
   926         *-10.0D00*DLOG10(DCOS(RVLxELEV)) + 10.0D00*DLOG10(DSIN(RVSxBOU(
   927         *IVSxNB,4))) - 10.0D00*DLOG10(RVLxFRE)/3.0D00
   928                 IF (RVLxHOR.GT.RVCxLOSG) RVCxLOSG = RVLxHOR
   929             ENDIF
   930             RETURN
   931             END
```

LOSS Local Symbols

Name	Class	Type	Size
IFSXEND	param		
RVSXEL0	param		
RVSXLON0	param		
RVSXLAT0	param		
IVSXNB	param		
RVSXBUN	param		
RVSXBOU	param		
I	local	INTEGER*4	4
J	local	INTEGER*4	4
RCLXAU	local	REAL*8	8
RTLXMAG1	local	REAL*8	8
RTLXMAG2	local	REAL*8	8
RTLXV1	local	REAL*8	24
RTLXV2	local	REAL*8	24
RTLXDE	local	REAL*8	8
CVLXRH	local	COMPLEX*16	16
RTLXR	local	REAL*8	8
RTLXS	local	REAL*8	72
RTLXT	local	REAL*8	24
RTLXIM	local	REAL*8	8
RTLXCT	local	REAL*8	24
RTLXRE	local	REAL*8	8
CTLXCSQ	local	COMPLEX*16	16
RTLXCV	local	REAL*8	24
RTLXARG	local	REAL*8	8
CVLXRV	local	COMPLEX*16	16
CTLXSIN	local	COMPLEX*16	16
RVLXAREA	local	REAL*8	8
RVLXFRE	local	REAL*8	8
CTLXNSQ	local	COMPLEX*16	16
RTLXCRS	local	REAL*8	24
CTLXSQT	local	COMPLEX*16	16

LOSS Local Symbols

Name	Class	Type	Size
RVLXHOR	local	REAL*8	8
RVCXLOSG	LOSSES	REAL*8	8
RPCXPI	PRAM	REAL*8	8
RPCXDTOR	PRAM	REAL*8	8
RPCXRE	PRAM	REAL*8	8
RVCXII	OTHER	REAL*8	8
RVCXJJ	OTHER	REAL*8	8
RVCXKK	OTHER	REAL*8	8
RVCXFSQ	OTHER	REAL*8	8
RVCXTIM	LPARM	REAL*8	8
RVCXCY	MORE	REAL*8	8
RVCXCZ	MORE	REAL*8	8
RVCXSEZTX	MISC	REAL*8	72
RTCX1	MISC	REAL*8	8
RTCX2	MISC	REAL*8	8
RVCXYEAR	LPARM	REAL*8	8
RTCX3	MISC	REAL*8	8
RTCX4	MISC	REAL*8	8
RTCX5	MISC	REAL*8	8
RTCX6	END	REAL*8	8
RTCX7	END	REAL*8	8
RTCX8	END	REAL*8	8
RVCXA100	LPARM	REAL*8	8
RVCXHMIN	MISC	REAL*8	8
RVCXH5	END	REAL*8	8
RVCXCX	MORE	REAL*8	8
RTCXA	OTHER	REAL*8	8
RTCXB	OTHER	REAL*8	8
RTCXC	PRAM	REAL*8	8
RVCXLOSA	LOSSES	REAL*8	8
RVCXLOSR	LOSSES	REAL*8	8
IVCXSX	MORE	INTEGER*4	4
IVCXSY	MORE	INTEGER*4	4
IVCXSZ	MORE	INTEGER*4	4
RVLXELEV	LOCAL	REAL*8	8
IVCXTIM	LPARM	INTEGER*4	20
IFCXGND	LPARM	INTEGER*4	4
IVCXSSN	LPARM	INTEGER*4	4
RVCXLOSX	LOSSES	REAL*8	8

```
Line#  Source Line

   934          SUBROUTINE GCDEV(IVSxNB,RVSxAZ,RVSxLA0,RVSxLO0,RVSxBOU,RVSxDEV)
   935  C
   936  C-----------------------------------------------------------------
   937  C
   938  C    GCDEV -- SUBROUTINE TO EVALUATE THE DEVIATION OF A RAY FROM
   939  C                  A GREAT CIRCLE PATH
   940  C
   941  C         CALLED BY: RAYSUB
   942  C
   943  C-----------------------------------------------------------------
   944  C
   945  C         AUTHOR:           ERIC L. STROBEL
   946  C
   947  C         DATE:             10/10/86
   948  C
   949  C         VERSION:          2.1
   950  C
   951  C-----------------------------------------------------------------
   952  C
   953  C         REVISED:          09/05/86 -- INITIAL REVISION.
   954  C
   955  C                           09/17/86 -- V2.0.  Practically a complete
   956  C                           rewrite.  Now does the spherical trig OK
   957  C                           even in the special cases... I think.
   958  C
   959  C                           10/10/86 -- V2.1.  Changed to conform to
   960  C                           the new status of RAYTRACE as a subrout.
   961  C
   962  C-----------------------------------------------------------------
   963  C
   964  C        USES:    RVSxBOU(11,15)          BOUNCE PT. ARRAY
   965  C                 IVSxNB                  BOUNCE COUNT
   966  C                 RVSxAZ                  LAUNCH AZIMUTH
   967  C                 RVSxLA0                 LAUNCH LATITUDE
   968  C                 RVSxLO0                 LAUNCH LONGITUDE
   969  C
   970  C         TO CALCULATE THE DEVIATION OF THE RAYPATH AWAY FROM ITS
   971  C                  EXPECTED LANDING POINT (IF IT HAD FOLLOWED A
   972  C                  GREAT CIRCLE PATH).
   973  C
   974  C         RETURNS:
   975  C                 RVSxDEV                 DEVIATION IN KM
   976  C
   977  C-----------------------------------------------------------------
   978  C
   979          INTEGER IVSxNB, ITLxS, ITLxC
   980  C
   981          REAL*8 RVSxDEV, RVSxBOU(11,15), RVSxAZ
   982          REAL*8 RVSxLA0, RVSxLO0, RPCxPI, RPCxRE, RPCxDTR
   983          REAL*8 RVLxR, RTLx1, RVLxLAP, RVLxLOP, RVLxSDL
```

```
Line#  Source Line

   984            REAL*8 RVLxCDL, RVLxDL, RTLxA, RTLxB, RTLxD, RTCxA
   985  C
   986            COMMON /PRAM/ RPCxPI, RPCxDTR, RPCxRE, RTCxA
   987  C
   988  C         First, do the spherical trig to get the great circle
   989  C           landing point (i.e., where the ray would have landed,
   990  C           given the range that it went, if it had stayed on the
   991  C           great circle path at the original azimuth).
   992  C
   993            RVLxR = RVSxBOU(IVSxNB, 4)
   994            RTLx1 = DSIN(RVSxLA0)*DCOS(RVLxR) + DCOS(RVSxLA0)*
   995           *DSIN(RVLxR)*DCOS(RVSxAZ)
   996            RVLxLAP = DASIN(RTLx1)
   997            RVLxSDL = DSIN(RVSxAZ)*DSIN(RVLxR)/DCOS(RVLxLAP)
   998            RVLxCDL = (DCOS(RVLxR)-DSIN(RVSxLA0)*DSIN(RVLxLAP))/
   999           *(DCOS(RVSxLA0)*DCOS(RVLxLAP))
  1000            RVLxDL = DASIN(RVLxSDL)
  1001            ITLxS = INTSIGN(RVLxSDL)
  1002            ITLxC = INTSIGN(RVLxCDL)
  1003            IF (ITLxC.LT.0) RVLxDL = ITLxS*RPCxPI - RVLxDL
  1004            IF (ITLxS.EQ.0.AND.ITLxC.LT.0) RVLxDL = RPCxPI
  1005            RVLxLOP = RVSxLO0 + RVLxDL
  1006  C
  1007  C         Now, calculate the great circle distance between the actual
  1008  C           landing point and the great circle landing point.
  1009  C
  1010            RTLxA = RVSxBOU(IVSxNB, 1)
  1011            RTLxB = RVSxBOU(IVSxNB, 2)
  1012            RTLxD = RTLxB - RVLxLOP
  1013            RVSxDEV = DSIN(RTLxA)*DSIN(RVLxLAP) + DCOS(RTLxA)*
  1014           #DCOS(RVLxLAP)*DCOS(RTLxD)
  1015            IF ((RVSxDEV - 1.0D00).GT.0.0D00) RVSxDEV = 1.0D00
  1016            IF ((RVSxDEV + 1.0D00).LT.0.0D00) RVSxDEV = -1.0D00
  1017            RVSxDEV = RPCxRE*DACOS(RVSxDEV)
  1018            RETURN
  1019            END


GCDEV  Local Symbols

Name                          Class    Type                Size

RVSXDEV . . . . . . . . . . . param
RVSXBOU . . . . . . . . . . . param
RVSXLO0 . . . . . . . . . . . param
RVSXLA0 . . . . . . . . . . . param
RVSXAZ. . . . . . . . . . . . param
IVSXNB. . . . . . . . . . . . param
ITLXC . . . . . . . . . . . . local    INTEGER*4             4
RTLXA . . . . . . . . . . . . local    REAL*8                8
```

GCDEV Local Symbols

Name	Class	Type	Size
RTLXB	local	REAL*8	8
RTLXD	local	REAL*8	8
ITLXS	local	INTEGER*4	4
RVLXDL	local	REAL*8	8
RVLXR	local	REAL*8	8
RVLXCDL	local	REAL*8	8
RVLXLAP	local	REAL*8	8
RVLXSDL	local	REAL*8	8
RVLXLOP	local	REAL*8	8
RTLX1	local	REAL*8	8
RPCXPI	PRAM	REAL*8	8
RPCXRE	PRAM	REAL*8	8
RPCXDTR	PRAM	REAL*8	8
RTCXA	PRAM	REAL*8	8

```
Line#  Source Line

 1022          SUBROUTINE ANRANG(RVSxAZ, RVSxLA1, RVSxLO1, RVSxLA0, RVSxLO0
 1023       #, RVSxR)
 1024  C
 1025  C-----------------------------------------------------------------
 1026  C
 1027  C        ANRANG  -- SUBROUTINE TO COMPUTE THE ANGULAR RANGE BETWEEN
 1028  C                   TWO POINTS
 1029  C
 1030  C
 1031  C        CALLED BY:  RAYSUB
 1032  C
 1033  C-----------------------------------------------------------------
 1034  C
 1035  C        AUTHOR:           ERIC L. STROBEL
 1036  C
 1037  C        DATE:             03/18/88
 1038  C
 1039  C        VERSION:          2.0
 1040  C
 1041  C-----------------------------------------------------------------
 1042  C
 1043  C        REVISED:          05/01/87 -- V1.0.  Initial revision.
 1044  C
 1045  C                          03/18/88 -- V2.0.  Rewritten to handle using
 1046  C                          the correct azimuth between the launch point
 1047  C                          and the current point, as opposed to using the
 1048  C                          launch azimuth.  Also, now correctly
 1049  C                          handles azimuths of 0 & 180 degrees, as
 1050  C                          well as angular ranges > 180 degrees.
 1051  C
 1052  C-----------------------------------------------------------------
 1053  C
 1054  C        USES:             RVSxAZ        Azimuth.
 1055  C
 1056  C                          RVSxLA1 \
 1057  C                          RVSxLO1 |_    The endpoints of the path.
 1058  C                          RVSxLA0 |
 1059  C                          RVSxLO0 /
 1060  C
 1061  C
 1062  C        To calculate the angular range along the great circle path
 1063  C          between the two points.
 1064  C
 1065  C
 1066  C        RETURNS:          RVSxR         The angular range.
 1067  C
 1068  C-----------------------------------------------------------------
 1069  C
 1070          REAL*8 RTLx1, RVSxAZ, RVSxLA1, RVSxLO1, RVSxLA0, RVLxTHC
 1071          REAL*8 RVSxLO0, RVSxR, RVLxCR, RPCxPI, RVLxPIO2, RVLxTH1
```

```
Line#  Source Line

 1072          REAL*8 RTCxA, RTCxB, RTCxC, RVLxS0, RVLxC0, RVLxS1, RVLxC1
 1073          REAL*8 RVLxS2, RVLxC2, RVLxTAD, RVLxAC1, RVLxAC2, RVLxTAN
 1074          REAL*8 RVLxAC, RVLxD1, RVLxD2
 1075  C
 1076          INTEGER IVLxSS, IVLxSC
 1077  C
 1078          COMMON /PRAM/ RPCxPI, RTCxA, RTCxB, RTCxC
 1079  C
 1080          RVLxPIO2 = RPCxPI/2.0D00
 1081          RTLx1 = RVSxLO1 - RVSxLO0
 1082          RVLxTH0 = RVLxPIO2 - RVSxLA0
 1083          RVLxTH1 = RVLxPIO2 - RVSxLA1
 1084  C
 1085          RVLxS0 = DSIN(RVLxTH0)
 1086          RVLxC0 = DCOS(RVLxTH0)
 1087          RVLxS1 = DSIN(RVLxTH1)
 1088          RVLxC1 = DCOS(RVLxTH1)
 1089          RVLxS2 = DSIN(RTLx1)
 1090          RVLxC2 = DCOS(RTLx1)
 1091  C
 1092          RVLxCR = RVLxC2*RVLxS0*RVLxS1 + RVLxC0*RVLxC1
 1093          RVLxTAD = RVLxC1 - RVLxCR*RVLxC0
 1094  C
 1095  C        We're trying to push spherical trig to yield correct values
 1096  C          even when the sides of the triangle are greater than 180 deg.
 1097  C          Here, the correct azimuth between the endpoints of the range
 1098  C          is being calculated.  In this way, if the range is greater
 1099  C          than 180 degrees, we can get enough information to determine
 1100  C          the correct value.
 1101  C
 1102          IF (RVLxTAD.EQ.0.0) THEN
 1103              RVLxAC1 = RVLxPIO2
 1104              RVLxAC2 = 3.0 * RVLxPIO2
 1105          ELSE IF (RVLxS2.EQ.0.0) THEN
 1106              RVLxAC1 = 0.0
 1107              RVLxAC2 = RPCxPI
 1108          ELSE
 1109              RVLxTAN = RVLxS0*RVLxS1*RVLxS2
 1110              RVLxAC = DATAN(RVLxTAN/RVLxTAD)
 1111              IF (RVLxAC.LT.0.0) RVLxAC = RVLxAC + RPCxPI
 1112              RVLxAC1 = RVLxAC
 1113              RVLxAC2 = RVLxAC + RPCxPI
 1114          ENDIF
 1115          RVLxD1 = DABS(RVSxAZ - RVLxAC1)
 1116          RVLxD2 = DABS(RVSxAZ - RVLxAC2)
 1117          IF (RVLxD1.LT.RVLxD2) THEN
 1118              RVSxAZ = RVLxAC1
 1119          ELSE
 1120              RVSxAZ = RVLxAC2
 1121          ENDIF
```

```
 Line#  Source Line

  1122  C
  1123  C       Now calculate the angular range.
  1124  C
  1125          IF (RVSxAZ.EQ.0.0.OR.RVSxAZ.EQ.RPCxPI) THEN
  1126              RVSxR = (RVLxTH0 - RVLxC2*RVLxTH1)*DCOS(RVSxAZ)
  1127          ELSE
  1128              IVLxSS = INTSIGN(RVLxS2*RVLxS1/DSIN(RVSxAZ))
  1129              RVSxR = DACOS(RVLxCR)
  1130          ENDIF
  1131  C
  1132  C       Range will be negative if greater than 130 degrees.
  1133  C         Therefore the following line will give the desired
  1134  C         value.
  1135  C
  1136          IF (RVSxR.LT.0.0) RVSxR = 2.0D00*RPCxPI - RVSxR
  1137  C
  1138          RETURN
  1139          END


ANRANG  Local Symbols

Name                            Class    Type                Size

RVSXR . . . . . . . . . . . . . param
RVSXLO0 . . . . . . . . . . . . param
RVSXLA0 . . . . . . . . . . . . param
RVSXLO1 . . . . . . . . . . . . param
RVSXLA1 . . . . . . . . . . . . param
RVSXAZ. . . . . . . . . . . . . param
RVLXC1. . . . . . . . . . . . . local    REAL*8                 8
RVLXD1. . . . . . . . . . . . . local    REAL*8                 8
RVLXAC1 . . . . . . . . . . . . local    REAL*8                 8
RVLXC2. . . . . . . . . . . . . local    REAL*8                 8
RVLXD2. . . . . . . . . . . . . local    REAL*8                 8
RVLXAC2 . . . . . . . . . . . . local    REAL*8                 8
RVLXS0. . . . . . . . . . . . . local    REAL*8                 8
RVLXAC. . . . . . . . . . . . . local    REAL*8                 8
RVLXS1. . . . . . . . . . . . . local    REAL*8                 8
RVLXS2. . . . . . . . . . . . . local    REAL*8                 8
RVLXTH0 . . . . . . . . . . . . local    REAL*8                 8
RVLXTH1 . . . . . . . . . . . . local    REAL*8                 8
RVLXCR. . . . . . . . . . . . . local    REAL*8                 8
RVLXTAD . . . . . . . . . . . . local    REAL*8                 8
RVLXPIO2. . . . . . . . . . . . local    REAL*8                 8
IVLXSS. . . . . . . . . . . . . local    INTEGER*4              4
RVLXTAN . . . . . . . . . . . . local    REAL*8                 8
RTLX1 . . . . . . . . . . . . . local    REAL*8                 8
RVLXC0. . . . . . . . . . . . . local    REAL*8                 8
RPCXPI. . . . . . . . . . . . . PRAM     REAL*8                 8
```

ANRANG Local Symbols

Name	Class	Type	Size
RTCXA	PRAM	REAL*8	8
RTCXB	PRAM	REAL*8	8
RTCXC	PRAM	REAL*8	8

```
Line#  Source Line

 1142          SUBROUTINE INBOX(RVSxLAI,RVSxLOI,RVSxLAS,RVSxLOS
 1143       #,RVSxLAE,RVSxLOE,IVSxOUT)
 1144  C
 1145  C-----------------------------------------------------------------
 1146  C
 1147  C        INBOX  -- SUBROUTINE TO DETERMINE WHETHER A GIVEN PT. IS
 1148  C                  WITHIN THE IONOSPHERIC SPECIFICATION GRID.
 1149  C
 1150  C
 1151  C        CALLED BY:       IONOPAR
 1152  C
 1153  C-----------------------------------------------------------------
 1154  C
 1155  C        AUTHOR:          ERIC L. STROBEL
 1156  C
 1157  C        DATE:            03/01/87
 1158  C
 1159  C        VERSION:         1.0
 1160  C
 1161  C-----------------------------------------------------------------
 1162  C
 1163  C        REVISED:         03/01/87 -- V1.0.  Initial revision.
 1164  C
 1165  C-----------------------------------------------------------------
 1166  C
 1167  C        USES:            RVSxLAI, LOI     The coord. of the pt. of
 1168  C                                          interest.
 1169  C
 1170  C                         RVSxLAS \
 1171  C                         RVSxLOS  |__     The starting and ending
 1172  C                         RVSxLAE  |       (SW & NE) corners of the
 1173  C                         RVSxLOE /        specification grid.
 1174  C
 1175  C
 1176  C        To determine whether or not the point of interest lies
 1177  C          within the grid over which the ionosphere is specified.
 1178  C          If it is outside, then the program will use a spherically
 1179  C          symmetric ionosphere having the parameters of the nearest
 1180  C          grid point, otherwise the program will do the interpolation.
 1181  C
 1182  C
 1183  C        RETURNS:         IVSxOUT          A flag specifying whether
 1184  C                                          the pt. of interest is in or
 1185  C                                          out of the grid. (0 = in,
 1186  C                                          1 = out)
 1187  C
 1188  C-----------------------------------------------------------------
 1189  C
 1190          REAL*8 RVSxLAI, RVSxLOI, RVSxLAS, RVSxLOS
 1191          REAL*8 RVSxLAE, RVSxLOE
```

```
Line#  Source Line

 1192           REAL*8 RTLx1, RTLx2, RTLx3, RTLx4
 1193  C
 1194           INTEGER IVSxOUT
 1195  C
 1196  C
 1197           RTLx1 = MIN(RVSxLOS,RVSxLOE)
 1198           RTLx2 = MAX(RVSxLOS,RVSxLOE)
 1199           RTLx3 = MIN(RVSxLAS,RVSxLAE)
 1200           RTLx4 = MAX(RVSxLAS,RVSxLAE)
 1201           IF ((((RVSxLAI-RTLx4).LE.0.0D00).AND.
 1202          #((RVSxLAI-RTLx3).GE.0.0D00)).AND.
 1203          #(((RVSxLOI-RTLx2).LE.0.0D00).AND.
 1204          #((RVSxLOI-RTLx1).GE.0.0D00))) THEN
 1205               IVSxOUT = 0
 1206           ELSE
 1207               IVSxOUT = 1
 1208           ENDIF
 1209           RETURN
 1210           END


INBOX  Local Symbols

Name                             Class    Type              Size

IVSXOUT . . . . . . . . . . . . param
RVSXLOE . . . . . . . . . . . . param
RVSXLAE . . . . . . . . . . . . param
RVSXLOS . . . . . . . . . . . . param
RVSXLAS . . . . . . . . . . . . param
RVSXLOI . . . . . . . . . . . . param
RVSXLAI . . . . . . . . . . . . param
RTLX1 . . . . . . . . . . . . . local    REAL*8              8
RTLX2 . . . . . . . . . . . . . local    REAL*8              8
RTLX3 . . . . . . . . . . . . . local    REAL*8              8
RTLX4 . . . . . . . . . . . . . local    REAL*8              8
```

```
 Line#  Source Line

  1213            FUNCTION INTSIGN(R)
  1214  C
  1215  C          GIVES 1,-1,OR 0 AS THE SIGN OF A REAL*8 NUMBER.  This
  1216  C              mimics the SIGN function available in most BASICs.
  1217  C
  1218            REAL*8 R
  1219  C
  1220            INTEGER INTSIGN
  1221  C
  1222            IF (R.EQ.0.0D00) THEN
  1223                INTSIGN = 0
  1224                RETURN
  1225            ENDIF
  1226            INTSIGN = IDNINT(R/DABS(R))
  1227            RETURN
  1228            END
```

INTSIGN Local Symbols

Name	Class	Type	Size
R	param		
INTSIGN	param		

Global Symbols

Name	Class	Type	Size
ACCFSP	FSUBRT	***	***
ANRANG	FSUBRT	***	***
END	common	***	32
FREESP	FSUBRT	***	***
GCDEV	FSUBRT	***	***
GORP	common	***	16
INBOX	FSUBRT	***	***
INTSIGN	FFUNCT	INTEGER*4	***
IONO1	common	***	32
LOCAL	common	***	8
LOSS	FSUBRT	***	***
LOSSES	common	***	32
LPARM	common	***	52
MAINDAT	common	***	48
MISC	common	***	120
MORE	common	***	36
NEWCS	FSUBRT	***	***
OTHER	common	***	48
PRAM	common	***	32
START	common	***	40

Global Symbols

Name	Class	Type	Size
TEMP2	common	***	16
TIMES	FSUBRT	***	***
TRIANG.	FSUBRT	***	***

Code size = 1f47 (8007)
Data size = 0116 (278)