MANUAL FOR OBSCURATION CODE WITH SPACE STATION APPLICATIONS

R. J. Marhefka and L. Takacs

THE OHIO STATE UNIVERSITY
ElectroScience Laboratory
Columbus, Ohio

Grant NSG-1498
May 1986

FOR REFERENCE

NOT TO BE TAKEN FROM THIS BOOK

LIBRARY COPY

NOTICE

FOR EARLY DOMESTIC DISSEMINATION

Because of its significant early commercial potential, this information, which has been developed under a U.S. Government program, is being disseminated within the United States in advance of general publication. This information may be duplicated and used by the recipient with the express limitation that it not be published. Release of this information to other domestic parties by the recipient shall be made subject to these limitations.

Foreign release may be made only with prior NASA approval and appropriate export licenses. This legend shall be marked on any reproduction of this information in whole or in part.

Date for general release

May 31, 1988

NASA
National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia 23665
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2.1</td>
<td>EXIT</td>
<td>49</td>
</tr>
<tr>
<td>5.2.2</td>
<td>HELP</td>
<td>50</td>
</tr>
<tr>
<td>5.2.3</td>
<td>SHADOW</td>
<td>52</td>
</tr>
<tr>
<td>5.2.4</td>
<td>SPAWN</td>
<td>54</td>
</tr>
<tr>
<td>5.2.5</td>
<td>SET ANTENNA_LOCATION</td>
<td>56</td>
</tr>
<tr>
<td>5.2.6</td>
<td>SET COORDINATES</td>
<td>58</td>
</tr>
<tr>
<td>5.2.7</td>
<td>SET FILL_CHARACTER</td>
<td>59</td>
</tr>
<tr>
<td>5.2.8</td>
<td>SET INPUT_SET</td>
<td>62</td>
</tr>
<tr>
<td>5.2.9</td>
<td>SET KEYPAD_MODE</td>
<td>63</td>
</tr>
<tr>
<td>5.2.10</td>
<td>SET OUTPUT</td>
<td>64</td>
</tr>
<tr>
<td>5.2.11</td>
<td>SET PATTERN_CUT</td>
<td>66</td>
</tr>
<tr>
<td>5.2.12</td>
<td>SET SCALE_FACTOR</td>
<td>67</td>
</tr>
<tr>
<td>5.2.13</td>
<td>SET UNITS</td>
<td>68</td>
</tr>
<tr>
<td>5.2.14</td>
<td>SET WINDOW</td>
<td>69</td>
</tr>
<tr>
<td>5.2.15</td>
<td>SHOW ANTENNA_LOCATION</td>
<td>71</td>
</tr>
<tr>
<td>5.2.16</td>
<td>SHOW COORDINATES</td>
<td>72</td>
</tr>
<tr>
<td>5.2.17</td>
<td>SHOW FILL_CHARACTER</td>
<td>73</td>
</tr>
<tr>
<td>5.2.18</td>
<td>SHOW INPUT_SET</td>
<td>74</td>
</tr>
<tr>
<td>5.2.19</td>
<td>SHOW KEYPAD_MODE</td>
<td>75</td>
</tr>
<tr>
<td>5.2.20</td>
<td>SHOW OUTPUT</td>
<td>76</td>
</tr>
<tr>
<td>5.2.21</td>
<td>SHOW PATTERN_CUT</td>
<td>77</td>
</tr>
<tr>
<td>5.2.22</td>
<td>SHOW SCALE_FACTOR</td>
<td>78</td>
</tr>
<tr>
<td>5.2.23</td>
<td>SHOW UNITS</td>
<td>79</td>
</tr>
<tr>
<td>5.2.24</td>
<td>SHOW WINDOW</td>
<td>80</td>
</tr>
</tbody>
</table>

6 Interpretation of the Output

7 Examples

7.1 Example 1: A Plate 84
7.2 Example 2: A Different Plate 85
7.3 Example 3: The First Plate Revisited 87
7.4 Example 4: A Non-Interactive Version of Example 1 89
7.5 Example 5: An Elliptic Cylinder 91
7.6 Example 6: Two Elliptic Cylinders 92
7.7 Example 7: A Space Station Model 97
7.8 Example 8: Another Look at the Space Station 102

References

II Code Manual

8 Introduction

9 Code Organization
10 Listings of the Code
10.1 VAX/VMS Subroutines
   10.1.1 MAIN PROGRAM
   10.1.2 SUBROUTINE INIT
   10.1.3 SUBROUTINE INTRAC
   10.1.4 Interactive Service Routines
10.2 Non-VAX/VMS Subroutines
   10.2.1 MAIN PROGRAM (non-interactive)
10.3 Subroutines common to both modes
   10.3.1 SUBROUTINE ABSHIN
   10.3.2 FUNCTION BABS
   10.3.3 BLOCK DATA
   10.3.4 FUNCTION BTAN2
   10.3.5 SUBROUTINE CYLINT
   10.3.6 SUBROUTINE CYLROT
   10.3.7 SUBROUTINE DOCYLS
   10.3.8 SUBROUTINE DOCYL
   10.3.9 SUBROUTINE DOPLAS
   10.3.10 SUBROUTINE DOPLA
   10.3.11 SUBROUTINE GEOM
   10.3.12 SUBROUTINE INITGF
   10.3.13 SUBROUTINE PLAINT
   10.3.14 SUBROUTINE SCAN
   10.3.15 FUNCTION SCANC/SPANC
   10.3.16 SUBROUTINE WRTOUT
   10.3.17 Include file
10.4 Non-FORTAN VAX/VMS source files
   10.4.1 CDU Source file
   10.4.2 Keypad initialization file
11 VAX Implementation
   11.1 Assembling the Code
   11.2 Running the Code
   11.3 Modifying the code
12 Non-VAX Implementation
13 NCAR Plot Program
List of Tables

3.1 Table of non-interactive commands. ........................................... 16
3.2 Table of interactive commands. .................................................. 20
3.3 VT100 keypad for SHADOW interactive commands. ....................... 21
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Geometry showing the projection of the plate edge onto the far zone sphere.</td>
<td>5</td>
</tr>
<tr>
<td>2.2</td>
<td>Geometry showing the projection of a cone frustum onto the far zone sphere.</td>
<td>7</td>
</tr>
<tr>
<td>2.3</td>
<td>Intersection of observation direction vector with plate.</td>
<td>8</td>
</tr>
<tr>
<td>2.4</td>
<td>The geometry for deciding whether a ray does or does not hit the plate.</td>
<td>10</td>
</tr>
<tr>
<td>2.5</td>
<td>Geometry illustrating the hit point on a cone frustum segment.</td>
<td>11</td>
</tr>
<tr>
<td>4.1</td>
<td>Definition of finite cylinder geometry composed of cone frustum segments with elliptic cross section.</td>
<td>26</td>
</tr>
<tr>
<td>4.2</td>
<td>Definition of finite elliptic cylinder geometry.</td>
<td>29</td>
</tr>
<tr>
<td>4.3</td>
<td>Definition of flat plate geometry.</td>
<td>35</td>
</tr>
<tr>
<td>4.4</td>
<td>Definition of rotate-translate coordinate system geometry.</td>
<td>37</td>
</tr>
<tr>
<td>4.5</td>
<td>Definition of source geometry for dipole antennas.</td>
<td>40</td>
</tr>
<tr>
<td>4.6</td>
<td>Definition of source geometry for aperture antennas.</td>
<td>41</td>
</tr>
<tr>
<td>4.7</td>
<td>Definition of volumetric pattern coordinate system.</td>
<td>46</td>
</tr>
<tr>
<td>7.1</td>
<td>Illustration of the Space Station.</td>
<td>115</td>
</tr>
<tr>
<td>7.2</td>
<td>Three-axis view of the Space Station as modeled by the input set.</td>
<td>116</td>
</tr>
<tr>
<td>7.3</td>
<td>NCAR plot showing the shadow map of the space station model.</td>
<td>118</td>
</tr>
</tbody>
</table>
Part I

User's Manual
Chapter 1

Introduction

When siting antennas on large structures, it is desirable to be able to quickly determine the clear line of sight transmission or reception paths for the antennas. If the structure under consideration is a space station, there will be many antennas to consider in an environment composed of a very large and complex array of living and working module, solar panels, and support structures. The antennas will potentially need to communicate with systems anywhere around the near zone of the structure and the complete far zone sphere. In short, a challenging problem.

In order to aid the antenna design engineer in the prediction of the near and far zone antenna patterns, antenna to antenna coupling, and radiation hazard considerations, for high frequency antennas in a complex environment, a couple of user oriented computer codes have been developed: the NEC - Basic Scattering Code (NEC-BSC) [1,2] and the Aircraft Code (NEWAIR) [3]. Both codes are based on the Uniform Geometrical Theory of Diffraction (UTD) [4], which is a high frequency ray optical method with corrections at shadow boundaries. The UTD is ideal for construction of efficient computer codes, such as these, for modeling the scattering from large structures. The NEC-BSC and NEWAIR are complementary codes, that is, the NEC-BSC is used when the antennas are not mounted on a curved surface, and the NEWAIR is used when the antennas are mounted on a curved surface. Both codes use plates to model flat structures, the NEC-BSC presently uses finite elliptic cylinders to model curved surfaces, and the NEWAIR presently uses ellipsoids.

Although the two UTD codes are presently very useful for predicting the performance of antennas in a complex environment, such as a space station, there are a few important consideration that should be taken into account. First, the present versions of these codes were not specifically developed for a space station application. The NEC-BSC was developed for ships and the NEWAIR for aircraft. Second and most importantly, even though they run fast for large size structures in terms of a wavelength, as compared with computer codes using other theories, such as method of moments; a problem with as many structural pieces as a space station can take a very long time to calculate a volumetric pattern. This means that the problem of antenna siting in a large structural environment should be viewed as a multiple stepped procedure to optimize results for minimum time and cost.

The design procedure for antenna siting can be viewed as a three step process, as far as the computer codes are concerned. First, it can generally be assumed that a good antenna location will provide a clear line of sight path between transmitter and receiver over the desired range of operation. This can best be accomplished using a obscuration code, which
is the goal of this computer code and document. This code will provide a volumetric shadow map of the projected shadow of a structure onto the far zone sphere centered at the antenna location. It is very fast running on space station applications and can be run interactively providing nearly immediate answers depending on the overall usage of the computer.

Second, a worst case code could be developed that will predict not only the clear line of sight regions, but will also map out the maximum values of the various field terms, such as the reflected and diffracted lobes. These scattered fields can cause undesired lobes to show up in the region of interest. This type of code will not only provide an answer to the question of where the optimum location for an antenna system should be, but also how it should be oriented at that position and what the gain and side lobe levels would be optimum. It can be designed to run at a little additional time cost over the obscuration code.

The final step would be to run a field prediction code such as the NEC-BSC and NEWAIR codes or their future versions optimized for the space station. This would be the confirmation phase of the design procedure to make sure that no surprises occur in the volumetric patterns. At this stage, it does not matter that the codes take a little longer to run, especially for the wealth of information that they produce. Of course, these results can be used to compare with measured results on scale models to validate the measurements and vice versa.

This document is concerned with the obscuration code, referred to here as “SHADOW”. It has been specifically designed with space station applications in mind. It directly solves for a shadow map by projecting the border of multiple sided flat plates and composite cone frustums of elliptic cross section onto the far zone sphere. It then fills between the borders based on a pixel resolution and window size specified by the user. The definition of the geometry is based on a subset of the command word input system used for the UTD codes. This means that as the engineer proceeds through a design scenario progressing through the different levels of codes, there will be a minimum amount of conversion of input information.

The obscuration code has proven to be so efficient, that it was felt that it could be of great benefit to the design engineer to be able to run it in an interactive mode. Unfortunately, interactive procedures are not generally transportable between different computer systems. Because of the wide availability of DEC VAX computers in the engineering environment, and because of the ease of developing an interactive system on a VAX, the interactive features have been developed using device dependent software for the VAX. The non-interactive and interactive parts of the code have been kept separate, however, so that the code can be run non-interactively without much change.

This document is divided into two parts. Part I is a user manual, that treats the code more or less as a black box device. It is about all that will be need for the average user to get started and obtain results. Chapter 2 describes the method that is used to obtain the shadow. The overall view of the operation of the code is given in Chapter 3. It describes the non-interactive and interactive commands in a qualitative way. A dictionary of all the non-interactive commands needed in the SHADOW code is given in Chapter 4. It gives the details for inputting each command. Chapter 5 provides the details for the interactive commands. The output features are interpreted in Chapter 6. Examples on how to use the code are given in Chapter 7. When first learning how to use the code, it is essential to be able to reproduce some of these examples to be sure that the code is functioning properly.
on your system.

Part II of this document is a code manual. It goes into more specific information about
the coding itself. It is of importance primarily for people implementing the code on a
new system, for debugging errors, or for making changes in how the code operates. An
overview of how the code is organized is given in Chapter 9. A listing of the code is given
in Chapter 10. It is broken up into three parts for the non-interactive, FORTRAN 77
subroutines and into the interactive VAX dependent subroutines. The implementation of
the code on a VAX is given in Chapter 11 and a brief description of implementing the code
on a non-VAX computer is given in Chapter 12. A listing of an NCAR plotting code for
the shadow map is given in Chapter 13.
Chapter 2

Method

The first gauge of the ability of two antenna systems to communicate with one another at high frequencies is to determine if there is a clear line of sight path between them. This can be conveniently represented by a map of the projected shadow on the far zone sphere caused by the structures around an antenna's environment. One method of producing a shadow map is to choose an observation point on the far zone sphere and then determine if anything obscures the path and then move on to the next point. This method is slow, however, because there must be many repeated tests on the same blocking structures for the various observation point making up the shadow map.

In order to have quick turn around for antennas mounted on large structures, it is desirable to use a method that will directly produce the shadow projected onto the far zone sphere. This can be accomplished in a two step process. First the outside boundary of each individual piece making up the structure can be transformed from the $x, y, z$ coordinate system into a sequence of lines in the $\theta$ and $\phi$ pattern coordinate system. The area of the shadow map between the boundary lines for each piece may then be filled by looking at the center location between the lines and a shadow check on that piece of the structure can be performed. This reduces the test on each piece of structure from once every observation point to a few tests every pattern cut line. The calculation time, in general, is reduced by about two orders of magnitude. For example, instead of taking two hours, a map can conservatively be produced in about one minutes or better. These numbers dependent on the geometry, the window size of the map, and the resolution desired.

There are two fundamental types of structural pieces presently available for modeling in this obscuration code, the multiple sided flat plate and the multiple rimmed composite cone frustum of elliptic cross section. More than one plate or cylinder can be specified to build up a complex structure. A plate can be defined by the location of its corners in a reference coordinate system. A cone frustum can be defined by the size of its major and minor radii for each rim making up the composite cylinder.

The boundary of the structures are traced onto the far zone sphere by defining a vector from the source position, $\mathbf{R}_s$, to some position along its outer boundary, $\mathbf{R}_t$, such that

$$\mathbf{R} = \mathbf{R}_t - \mathbf{R}_s.$$  

In the case of the plate, the boundary is defined by some location along its edges, as illustrated in Figure 2.1. This vector can then be transformed into the pattern cut coordinate system, since the pattern may be defined relative to a different set of axes. The vector can
Figure 2.1: Geometry showing the projection of the plate edge onto the far zone sphere.
then be transformed onto the two dimensional far zone sphere by

\[ \theta = \arctan \left( \frac{\hat{\rho} \cdot \hat{R} \cdot \hat{z}}{R} \right) \]

and

\[ \phi = \arctan \left( \frac{\hat{y} \cdot \hat{R} \cdot \hat{z}}{R} \right) \].

The position of the vector along the edge is defined by starting at a corner and then incrementing the edge in steps of \( \delta t \) along the edge. In order to provide the most efficient performance and the best image of the shadow on the map, it is necessary to define \( \delta t \) as a function of the chosen resolution desired for the map, \( \delta \alpha \), the distance, \( R \), from the source to the edge point and the relative position of the projected shadow point with respect to the polar caps, the Greenland effect.

The resolution, \( \delta \alpha \) is chosen to be the minimum of the two specified incremental values of \( \theta \) and \( \phi \). The distance \( R \) is defined as \( R = |\vec{R} - \vec{R}_t| \). Assuming that the resolution increment is small and the distance is relatively large, the value of the edge increment is given by

\[ \delta t = \delta \alpha R \sin \theta. \]

The new edge point then becomes

\[ \vec{R}_{t+1} = \vec{R}_t + \delta t \hat{e}, \]

where \( \hat{e} \) is the edge vector pointing from the first corner to the second corner making up the edge.

The composite cone frustum can be done in the same way as the plate. In fact the end caps can be defined as plates with curved edges and the curved surfaces are added as edges whose corners are the tangent points illustrated in Figure 2.2.

Once a give plate or cylinder outer boundary is transformed onto the shadow map and stored in pixels of the desired resolution, the fill process can begin. The pixel array is considered one row at a time in a scanning operation from the one range of theta embodied in the pixel array to the other. The direction of the scan and the order in which rows are scanned is arbitrary. The fill process is the same for each scan line in the pixel array so that no logical interaction between lines takes place. The process is similar to the way in which a television paints pictures one row at a time on the screen. As the scan proceeds say from left to right, unlit pixels between object boundaries on the line which correspond to regions in the interior of the object are turned on creating an area fill. The decision to light a group of pixels on a given row is not made by testing each pixel individually for obscuration but by making a single test between the pixels which represent boundaries of the projected regions. In this way, only a single test is made to determine whether a whole group of pixels represent the interior or exterior of a region. This is one major key to the sizable reduction of processor time achieved.

The shadow test for a plate is made by first projecting the vector chosen at the midpoint of the scan line, \( \hat{r} \), onto the plane of the plate to find its intersection point, as shown in Figure 2.3, that is

\[ \vec{R}_t = \vec{R}_s - \frac{[\hat{n} \cdot (\vec{R}_s - \vec{C}_t)]\hat{r}}{\hat{n} \cdot \hat{r}}. \]
Figure 2.2: Geometry showing the projection of a cone frustum onto the far zone sphere.
Figure 2.3: Intersection of observation direction vector with plate.
Now, using an idea based on Cauchy's formula from complex variables, that is,

\[
\oint_C f(z) \, dz = \begin{cases} 
0, & \text{no pole in } f(z) \\
2\pi j, & \text{one pole in } f(z)
\end{cases}
\]

the intersection point can be tested to see whether or not it falls within the limits of the plate. This is illustrated in Figure 2.4.

It is easy to show that

\[
\theta_m = \arctan \left[ \frac{[(C_m - R_t) \times (C_{m+1} - R_t)] \cdot \hat{n}}{(C_m - R_t) \cdot (C_{m+1} - R_t)} \right]
\]

which leads to the test, if

\[
\sum_{m=1}^{M} \theta_m = \begin{cases} 
< \pi, & \text{no hit occurs} \\
> \pi, & \text{a hit occurs}
\end{cases}
\]

The end caps of the cone frustum cylinders can be done in the same way, by projecting the hit point in the plane of the end cap. The hit point distance can be tested from the center of the disk to see if it falls within the finite limits of a disk to simplify things a little. The curved surface test is a different matter, but still quite easy to accomplish. A vector on the surface of the cone frustum can be represented as

\[
\vec{R}_c = \vec{R} + \vec{R}_s
\]

or

\[
\vec{R}_c = (R \cos \phi \sin \theta + x_s)^2 \hat{x} + (R \sin \phi \sin \theta + y_s)^2 \hat{y} + (R \cos \theta + z_s)^2 \hat{z}.
\]

The geometry is illustrated in Figure 2.5. The point defined by \( \vec{R}_c \) should satisfy the equation for a cone, that is,

\[
\frac{(R \cos \theta + x_s)^2}{a_j^2} + \frac{(R \sin \phi \sin \theta + y_s)^2}{b_j^2} - \lambda_j^2 (R \cos \theta + z_s) = 0
\]

where

\[
\lambda_j(R \cos \theta + z_s) = \left[ 1 + \frac{1}{a_j} \tan \theta_j (R \cos \theta + z_s - z_j) \right].
\]

The distance \( R \) is unknown in this equation, since we know the direction to the observer, \( \theta \) and \( \phi \), but not the distance to the hit point. We can solve for \( R \), however, from the above equations using

\[
\alpha R^2 + 2\beta R + \gamma = 0,
\]

where

\[
\alpha = \frac{\cos^2 \phi \sin^2 \theta}{a_j^2} + \frac{\sin^2 \phi \sin^2 \theta}{b_j^2} - \frac{\tan^2 \theta_j \cos^2 \theta}{a_j^2},
\]

\[
\beta = \frac{x_s \cos \phi \sin \theta}{a_j^2} + \frac{y_s \sin \phi \sin \theta}{b_j^2} - \tan \theta_j \cos \theta \lambda_j(x_s),
\]

\[
\gamma = \frac{z_s \cos \phi \sin \theta}{a_j^2} + \frac{z_s \sin \phi \sin \theta}{b_j^2}.
\]
Figure 2.4: The geometry for deciding whether a ray does or does not hit the plate.
Figure 2.5: Geometry illustrating the hit point on a cone frustum segment.
and

$$\gamma = \frac{x^2}{a_j^2} + \frac{y^2}{b_j^2} - \lambda_j^2(z_s).$$

If the value of R is real, then the hit point is on the finite cone frustum and therefore the ray from the source to observer is shadowed. If the actual hit point is desired it should be noted that there are two values found from this equation, and that the right hit point can be found from the one representing the shortest distance. If the value of R is imaginary, however, this indicates that the hit point is off the real boundary of the cone frustum and therefore the ray is not shadowed. If R is real, an additional test must be made to decide whether the hit point in between the finite length bounds of the frustum.

The basic theory discussed here is rather straight forward. The implementation, of course, requires a lot of other considerations to be user friendly and as general purpose as possible. The next chapter will go into more detail about how the code interfaces with the operator.
Chapter 3

Principle of Operation

3.1 Overview

The Obscuration Code is intended to be an efficient means of determining the clear line of sight path for an antenna mounted in a complex environment. This code produces a shadow map of the geometry for a given source location. The configuration is defined using a command word system as discussed below. The geometry of the structure is defined by using plates and cylinders. It is thought that the obscuration code is just one step in a total evaluation scheme. The next step would be to either look at a "worst case" map that projects the location of the maximum lobes on to a volumetric map or to calculate the fields using a code like the NEC-BSC. In any case, the real fields should be calculated as the final step whether an intermediate one is used or not. For this reason, the geometry definition is based on the NEC-BSC code method of inputting information.

The obscuration code, however, is a very efficient means of providing a shadow map. It can be run in a matter of minutes or less for a given shadow map. It is, therefore, felt that it can be most efficiently run interactively, that is with the user sitting at a terminal changing antenna locations, looking at the resultant maps, deciding where to try the next antenna location until the desired optimum spot is found to achieve a given performance. For this reason the code has been developed in two pieces. One is a standard FORTRAN 77 part that does the essential shadowing calculations. The second is an interactive part that allows the user to change the source locations and window size without leaving the code. Unfortunately, this second part of the code is by nature device dependent. This part has been written for the DEC VAX series of computers using version 4 of VMS. It uses system handlers for defining the commands discussed in the sections below for the interactive commands and the keypad mode. The keys on the keypad of VT100 or VT200 series terminals can be used to represent the typed commands. This will simplify the use of the code by reducing the amount of typing necessary.

This chapter tries to give a brief overview of the specifics needed to run the code by treating it as a black box. It is intended to just get the user comfortable with the overall philosophy of the obscuration code.
3.2 Modeling the Structures

The building blocks available for the obscuration code are composed of pieces that are an extension of version 2 of the NEC-BSC [1]. Structures can be modeled using multiple sided flat plates and multiple rimmed cone frustum cylinders. The plates can be used individually to model things like solar panels or together to form box like structures to model things like the mast, etc. The cone frustums are a new feature here, and can be handy for modeling living modules, etc. Examples of space station models are given in Chapter 7.

Unlike the NEC-BSC, there are no real restrictions on how these structures are defined. Since the code just looks at each defined piece of the modeled structure individually, casts its shadow, then moves on to the next piece, it does not have to properly account for the wedge angles and other geometrical features needed in field calculations in the NEC-BSC. If one is setting up a model, however, it still might be useful to use the same modeling considerations as the NEC-BSC, such as defining the corners of a plate so the normals point in the region of space in which the source is located. It is assumed that the obscuration code phase of the design procedure will be followed by calculating the fields for the antenna on the structure using a code such as the NEC-BSC.

The number of plates and cylinders that can be used in the models is dictated only by the size of the dimensions implemented in the array for defining the geometry in the code. For convenience, these parameters are located in one file in the code so they only need to be changed in one spot. The details are given in Part II.

More information on how models are to be constructed are given in the section below on the non-interactive commands and in Chapter 4 where these commands are defined in more detail.

3.3 Running the Code

The first step of course is to get the code implemented on your system. The details of how to accomplish this are given in Part II. In order to use the full interactive features of the code, it is necessary to use the code on a DEC VAX. Many of the interactive features use VAX dependent implementations from version 4 of the VMS operating system. The code has been divided into standard FORTRAN 77 files and VAX dependent files, however, so that the code can be used without the interactive features on other systems. A slightly different main program needs to be used as provided in Part II. In addition, the non FORTRAN 77 INCLUDE statement has been used in the non interactive file. Many systems have this feature, so it was left in as a convenience. If the user system does not, it is easy to remove by hard wiring the lines in the appropriate file in place of the INCLUDE statement. Most of the information, here, will assume that the full features of the code will be able to be used.

The first step in using the code is to create a file that contains the basic structure definitions using the non-interactive commands discussed in the next section. The command defining the source location and window size of the shadow map can also be defined in this input file or added and changed in the interactive session. Of course, if you are running non-interactively, then all the data must be input from the input file.

Once the input file has been created or chosen from some stored files, the obscuration
code can be executed. It will read the input file from logical unit #5. An interactive command allows the user to connect the chosen input file to this logical unit number. The code then proceeds to read the input information and produce an output that is sent to the terminal (logical unit #6) representing how the code has interpreted the input. In this process, it is converting all the input into a standard reference coordinate system and into a common set of units which is meters. If there is a typographical error or other error in the input set the code will indicate so and stop execution at that point.

If the code completes the input, it will wait for the next instruction. For example, the output file name can be connect to the logical unit which is #7 for the line printer output and logical unit #10 for the plotter output. The antenna position can now be defined or modified and the desired window changed. The code can than be told to proceed to produce a shadow map.

When the code has completed the shadow calculation, the user can change the source location, the window of the map, or input another structure and run the code again; or he can print the map out. The map is an array of pixels (doubly dimensioned character array) that are in general either a blank representing a clear path or a character representing a blocked path. The character is normally a uniform character such as an “X”. There is an option to tag a particular plate with a character that you define, or the code will letter each plate and cylinder separately. This is useful to determine which plates get in the way or for debugging purposes. More details on this will be given in Chapter 6.

3.4 Non-Interactive Commands

The non-interactive commands needed in this code are a subset and a slight extension of those used in version 2 of the NEC-BSC [1]. The total list of the available non-interactive commands are given in Table 3.1. Only the commands of interest to the obscuration code are defined in this manual. The rest can be found in reference [1] or in later reports and manuals for newer versions of the codes. This section is intended to give the user a brief overview of the specific commands of interest with the details coming in the next Chapter.

The input commands words are intended to make it convenient for the user to define the geometry of the structure without having to define information not needed or repeat information already defined. They are two letter pairs. The rest of the characters on a command word line can be used for comments, since only the first two letters are interpreted.

There is a place in the code to place default data that will be present without a call to the command. This is convenient when a specific resolution sized of the shadow map is desired as a default, for example. The default window can be initial theta angles of 0 to 180 in steps of 2 degrees and initial phi angles of 0 to 360 in steps of 2 degrees by defining the proper variables in this default section. A call to the VF command will over ride this data if it is specified in the input set.

The geometry information is by default assumed to be in meters in a definition coordinate system that is initially the reference coordinate system. The units can be changed using the UN command to either inches or feet or back to meters again. Once the UN command is specified all information after that command is assumed to be in those units unless changed by another call to UN. There is also provision for using any conversion fac-
<table>
<thead>
<tr>
<th>COMMAND</th>
<th>DEFINITION</th>
<th>LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>BP</td>
<td>back or bistatic scatter</td>
<td>[1]</td>
</tr>
<tr>
<td>CC</td>
<td>cone frustum geometry</td>
<td>pg 24</td>
</tr>
<tr>
<td>CE</td>
<td>last or only comment</td>
<td>pg 30 or [1]</td>
</tr>
<tr>
<td>CG</td>
<td>cylinder geometry</td>
<td>pg 27 or [1]</td>
</tr>
<tr>
<td>CM</td>
<td>comment card</td>
<td>pg 30 or [1]</td>
</tr>
<tr>
<td>EN</td>
<td>end execution</td>
<td>pg 30 or [1]</td>
</tr>
<tr>
<td>FM</td>
<td>swept frequencies</td>
<td>[1]</td>
</tr>
<tr>
<td>FR</td>
<td>frequency</td>
<td>[1]</td>
</tr>
<tr>
<td>GP</td>
<td>infinite ground plane</td>
<td>pg 31 or [1]</td>
</tr>
<tr>
<td>GR</td>
<td>range gate</td>
<td>not documented</td>
</tr>
<tr>
<td>LP</td>
<td>line printer output</td>
<td>[1]</td>
</tr>
<tr>
<td>NC</td>
<td>next set of cylinders</td>
<td>pg 32 or [1]</td>
</tr>
<tr>
<td>NG</td>
<td>no ground plane</td>
<td>pg 32 or [1]</td>
</tr>
<tr>
<td>NP</td>
<td>next set of plates</td>
<td>pg 32 or [1]</td>
</tr>
<tr>
<td>NR</td>
<td>next set of receivers</td>
<td>[1]</td>
</tr>
<tr>
<td>NS</td>
<td>next set of sources</td>
<td>pg 32 or [1]</td>
</tr>
<tr>
<td>NX</td>
<td>next problem</td>
<td>pg 32 or [1]</td>
</tr>
<tr>
<td>PD</td>
<td>far zone pattern cut</td>
<td>[1]</td>
</tr>
<tr>
<td>PF</td>
<td>far zone cut (non integer)</td>
<td>not documented</td>
</tr>
<tr>
<td>PG</td>
<td>plate geometry</td>
<td>pg 33 or [1]</td>
</tr>
<tr>
<td>PN</td>
<td>near zone pattern cut</td>
<td>[1]</td>
</tr>
<tr>
<td>PP</td>
<td>plotter output</td>
<td>[1]</td>
</tr>
<tr>
<td>PR</td>
<td>gain or coupling factors</td>
<td>[1]</td>
</tr>
<tr>
<td>RA</td>
<td>receiver array geometry</td>
<td>[1]</td>
</tr>
<tr>
<td>RD</td>
<td>far zone range</td>
<td>[1]</td>
</tr>
<tr>
<td>RG</td>
<td>receiver geometry</td>
<td>[1]</td>
</tr>
<tr>
<td>RM</td>
<td>NEC-MOM receiver input</td>
<td>[1]</td>
</tr>
<tr>
<td>RT</td>
<td>rotate-translate geometry</td>
<td>pg 36 or [1]</td>
</tr>
<tr>
<td>SA</td>
<td>source array geometry</td>
<td>[1]</td>
</tr>
<tr>
<td>SG</td>
<td>source geometry</td>
<td>pg 38 or [1]</td>
</tr>
<tr>
<td>SM</td>
<td>NEC-MOM source input</td>
<td>[1]</td>
</tr>
<tr>
<td>TO</td>
<td>test options</td>
<td>[1]</td>
</tr>
<tr>
<td>UF</td>
<td>model scale factor</td>
<td>pg 42 or [1]</td>
</tr>
<tr>
<td>UN</td>
<td>units of geometry</td>
<td>pg 42 or [1]</td>
</tr>
<tr>
<td>US</td>
<td>units of source size</td>
<td>[1]</td>
</tr>
<tr>
<td>VD</td>
<td>volumetric cut (integer)</td>
<td>not documented</td>
</tr>
<tr>
<td>VF</td>
<td>volumetric far zone cut</td>
<td>pg 44</td>
</tr>
<tr>
<td>VN</td>
<td>volumetric near zone cut</td>
<td>not documented</td>
</tr>
<tr>
<td>VP</td>
<td>volumetric plotter output</td>
<td>not documented</td>
</tr>
<tr>
<td>XQ</td>
<td>execute code</td>
<td>pg 47 or [1]</td>
</tr>
</tbody>
</table>

Table 3.1: Table of non-interactive commands.
tor desired. It is input using the UF command and is a scale factor multiplying times all
the input dimensions in whatever unit have been defined. The code then takes the input
information and changes it internally and stores it in meters, in order to have a uniform
system in which to operate. The input dimensions and the internal dimensions are output
in the feed back print out sent to a file so the user can see what happened. The dimensions
of the source itself, that is length and width not its position, is handled with a default of
wavelengths. This can be changed with the US. The length and width of the source is not
important in this code so it can be ignored here.

The reference coordinate system is really whatever is convenient for the user. The
definition coordinate system is the same as this initial reference system or it can be changed
using the rotate translate command RT. The RT command allows allows the user to relocate
the origin and orientation of the definition coordinate system with respect to the reference
coordinate system. The definition system stays as defined for all subsequent geometry input
until it is changed. The RT command's definition is always referenced to the reference
coordinate system NOT to itself, that is, one does not put in inverse locations and angles
to undo the command, but resets it to the zero position of the origin and the x-axis and
x-axis of the reference coordinate system. Note that all angles are assumed to be input in
degrees. The coordinate axes are input in a uniform way through out the code by treating
the new axes vectors as if they were radial vectors in the system being used. That is the
z-axis is defined using a theta and phi angle relative to the reference coordinate system in
the RT command and likewise the x-axis is treated as a radial vector. The y-axis is defined
by a cross product between the x and z axes. The code checks that the x and z axes were
defined orthogonal to one another. If not an error message will result and the code will
stop.

The geometry commands are the PG command for the plates, the GP command for the
infinite ground plane, the CG command for an elliptic cylinder, and the CC command for
the cone frustum cylinders. The plates are defined by inputting the number and location of
their corners in the definition system. The ground plane is defined as a infinite plane lying
in the x-y plane of the definition coordinate system. The elliptic cylinder definition is base
on the location of its origin and the orientation of its z- and x-axes relative to the definition
coordinate system. In addition, the radius along its cylinder x-axis and the radius along its
cylinder y-axis, along with the z-axis position and angular orientation of its end caps are
needed. The cone frustum's definition is similar except that the number of rims making up
the cylinder need to be specified and the orientation of the rims does not, since they can't
be cut at an angle as in the elliptic cylinder case. For the plates and cylinders the code
automatically adds up the number of calls to the commands and counts that as the number
of plates or cylinders specified. Only one infinite ground plane can be defined.

The location of the sources are specified by their location, type, orientation, and relative
weights using the SG command. Only the location information is important to the obscuration
code. Each source specified is automatically counted and remember as the number of
sources. Unlike the plates and cylinders, the obscuration code only calculates one source at
a time for a shadow map. In non-interactive mode, it does one source at a time producing
a map for each. In interactive mode, it takes the first one as the default source and then
each subsequent one needs to be interactively input. Receivers are not recognized by the
shadow code, so if in reality you are studying a receiver, it must be input as a source not a
receiver for shadowing purposes.

In order to negate already defined commands for the geometry which is automatically increasing their number, a series of commands have been implemented. The plates can be reset to zero using the NP command. The ground plane with NG, the cylinders with NC, the sources with NS, and the entire run can be reset with the NX command.

The code is told to go and execute the interactive mode if it is available, or to go and execute the shadow calculations if the interactive mode has not been implemented using the XQ command. The EN command tells it to exit back to the operating system.

The next section will discuss an overview of the interactive commands and examples of these commands are given in Chapter 7.

3.5 Interactive Commands

The interactive commands provided by the code under VMS are designed to allow easy specification of commonly changed parameters with a syntax which is well-known to users of VMS, the DCL command interpreter syntax. To acquaint the reader with the appearance of these commands, they are summarized below. Detailed descriptions of each command complete with examples can be found in the Chapter 5 on interactive commands. A list of the available interactive commands are given in Table 3.2.

There are interactive commands to allow the user to control the operation of the code or to change or view the geometry. The SHADOW command produces the shadow map. The HELP command gives a descriptions of the commands. The EXIT command exits the user back to the operating system. The SPAWN command allows the use of DCL command while the user is still in the shadow code.

The rest of the commands either allow the user to change the geometry, with the SHOW commands, or see the present status of the geometry, with the SET commands. Most of them have a non-interactive command to which they are at least somewhat associated. The SET UNITS command allows the units of the antenna location to be chosen, similar to the UN command. The SET SCALE.FACTOR command is like the UF command, which allows an arbitrary scale factor for the geometry to be chosen. The SET COORDINATES command allows the definition coordinate system to be change, like in the RT command. The SET ANTENNA command enables the user to interactively specify the antenna location in the definition coordinate system. It is related to the SG command. The SET PATTERN.CUT command allow the user to specify the orientation of the pattern coordinate system in the reference coordinate system. The SET WINDOW command enables the initial, final and incremental angles of the shadow map to be specified. These two commands are related to the VF command.

The next four commands do not have non-interactive commands to which they are related. The SET INPUT command allows the user to specify what file containing the non-interactive commands is to be read. The SET OUTPUT command enables the specification of which output files are to be assigned and their names. The SET FILL.CHARACTER command allows the user to define the symbols that are used for the plate and cylinder shadows. The SET KEYPAD.MODE command enables the VT100 keypad to be used for command definitions as is discussed in the next section, otherwise, the keypad can be used for numerical input. These four commands are discussed much more thoroughly in
3.6 Keypad Use

The definable keypad functions are available for the interactive version of the code only. The keypad definitions are made possible through the use of an integrated VMS screen/terminal management package called SMG. It is a collection of runtime library routines which perform terminal I/O and intercept the special sequences transmitted by the keypad keys. When one of these keys are pressed, the text definition associated with the key is substituted onto the command line. All of this I/O is transparent to the user so that he need only worry about making the initial keypad definitions. For more information about SMG, the reader is referred to the VMS runtime library reference manual.

The keypad definitions are initialized by a text file containing suitable “DEFINE/KEY” commands. The file is called SHADOW.KPD and must reside in the default directory of the user running the code. There is a template file provided with the code which may be customized by the user. The predefined definitions of the VT100 keypad are shown in Table 3.3. Note that the “gold” enables the lower case action in the top row, that is, in most case the “SHOW” operation instead of the “SET” operation.
<table>
<thead>
<tr>
<th>COMMAND</th>
<th>LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXIT</td>
<td>Page 49</td>
</tr>
<tr>
<td>HELP</td>
<td>Page 50</td>
</tr>
<tr>
<td>SPAWN</td>
<td>Page 54</td>
</tr>
<tr>
<td>SET ANTENNA_LOCATION</td>
<td>Page 56</td>
</tr>
<tr>
<td>SET COORDINATES</td>
<td>Page 58</td>
</tr>
<tr>
<td>SET FILL_CHARACTER</td>
<td>Page 59</td>
</tr>
<tr>
<td>SET INPUT_SET</td>
<td>Page 62</td>
</tr>
<tr>
<td>SET KEYPAD_MODE</td>
<td>Page 63</td>
</tr>
<tr>
<td>SET OUTPUT</td>
<td>Page 64</td>
</tr>
<tr>
<td>SET PATTERN_CUT</td>
<td>Page 66</td>
</tr>
<tr>
<td>SET SCALE_FACTOR</td>
<td>Page 67</td>
</tr>
<tr>
<td>SET UNITS</td>
<td>Page 68</td>
</tr>
<tr>
<td>SET WINDOW</td>
<td>Page 69</td>
</tr>
<tr>
<td>SHADOW</td>
<td>Page 52</td>
</tr>
<tr>
<td>SHOW ANTENNA_LOCATION</td>
<td>Page 71</td>
</tr>
<tr>
<td>SHOW COORDINATES</td>
<td>Page 72</td>
</tr>
<tr>
<td>SHOW FILL_CHARACTER</td>
<td>Page 73</td>
</tr>
<tr>
<td>SHOW INPUT_SET</td>
<td>Page 74</td>
</tr>
<tr>
<td>SHOW KEYPAD_MODE</td>
<td>Page 75</td>
</tr>
<tr>
<td>SHOW OUTPUT</td>
<td>Page 76</td>
</tr>
<tr>
<td>SHOW PATTERN_CUT</td>
<td>Page 77</td>
</tr>
<tr>
<td>SHOW SCALE_FACTOR</td>
<td>Page 78</td>
</tr>
<tr>
<td>SHOW UNITS</td>
<td>Page 79</td>
</tr>
<tr>
<td>SHOW WINDOW</td>
<td>Page 80</td>
</tr>
</tbody>
</table>

Table 3.2: Table of interactive commands.
<table>
<thead>
<tr>
<th>PF1</th>
<th>PF2</th>
<th>PF3</th>
<th>PF3 no keypad SET KEYPAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>gold</td>
<td>HELP</td>
<td>SHADOW</td>
<td></td>
</tr>
<tr>
<td>7 show SET OUTPUT</td>
<td>8 show SET INPUT</td>
<td>9 show SET ANTENNA</td>
<td></td>
</tr>
<tr>
<td>4 show SET SCALE</td>
<td>5 show SET UNITS</td>
<td>6 show SET COORD</td>
<td></td>
</tr>
<tr>
<td>1 show SET FILL</td>
<td>2 /cylin FILL /PLATE</td>
<td>3 show FILL /SEQUEN</td>
<td>Enter</td>
</tr>
<tr>
<td>0 SPAWN</td>
<td>EXIT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.3: VT100 keypad for SHADOW interactive commands.
Chapter 4

Non-Interactive Commands

The non interactive commands discussed in this chapter are a subset of the commands used for the NEC-BSC2. The shadow code will recognize the entire set of NEC-BSC2 commands plus a few new ones. The new commands and some of the old ones that are pertinent to this code will be described here. The following sections define in detail each command word and the variables associated with them. This chapter is organized in alphabetical order of the commands. It is intended to be used as a reference for the user. Chapter 7 will give specific examples using this input method.

The method used to input data into the computer is presently based on a command word system. This is especially convenient when more than one problem is to be analyzed during a computer run. The code stores the previous input data such that one need only input that data which has to be changed from the previous execution. Also, there is a default list of data so for any given problem the amount of data that needs to be input has been shortened. The command word options presently available are listed in Table 3.1 on page 16. The colon after the command word is not necessary and is sometimes used just to illustrate the separation between the command word and the space where comments can be inserted.

In this system, all linear dimensions may be specified in either meters, inches, or feet and all angular dimensions are in degrees. All the dimensions are eventually referred to a fixed cartesian coordinate system used as a common reference for the source and scattering structures. There is, however, a geometry definition coordinate system that may be defined using the RT command. This command enables the user to rotate and translate the coordinate system to be used to input any selected data set into the best coordinate system for that particular geometry. Once the RT command is used all the input following the command will be in that rotated and translated coordinate system until the RT command is called again. See below for more details. There is also a separate coordinate system that can be used to define a pattern coordinate system. This is discussed in more detail in terms of the VF command.

It is felt that the maximum usefulness of the computer code can be achieved using it on an interactive computer system. As a consequence, all input data are defined in free format such that the operator need only put commas between the various inputs. This allows the user on an interactive terminal to avoid the problems associated with typing in the field length associated with a fixed format. This method also is useful on batch processing computers. Note that all read statements are made on unit #5, i.e., READ(5,*), where
the "*" symbol refers to free format. Other machines, however, may have different symbols representing free format.

In all the following discussions associated with logical variables a "T" will imply true, and an "F" will imply false. The complete words true and false need not be input since most compilers just consider the first character in determining the state of the logical variable.
4.1 Command CC: Cone Frustum Geometry

This command enables the user to define the geometry of the finite elliptic conical cylinder structures to be considered. The geometry is illustrated in Figure 4.1. One call to this command defines one cylinder. The number of cylinders in the structure are automatically counted by the number of calls to this command.

READ: (XCL(N,MC), N=1,3)

where

XCL(N,MC) This is a doubly dimensioned real variable. It is used to specify the location of the origin of the MCth elliptic cylinder relative to the definition coordinate system. It is input on a single line with the real numbers being the x,y,z coordinates of the origin which correspond to N=1,2,3, respectively.

READ: TCLZ, PCLZ, TCLX, PCLX

where

TCLZ, PCLZ These are real variables. They are input in degrees as spherical angles that define the \( z_c \)-axis of the cylinder coordinate system as if it was a radial vector in the definition coordinate system.

TCLX, PCLX These are real variables. They are input in degrees as spherical angles that define the \( z_c \)-axis of the cylinder coordinate system as if it was a radial vector in the definition coordinate system.

Note that the new \( z_c \)-axis and \( z_c \)-axis must be defined orthogonal to each other. The new \( y_c \)-axis is found from the cross product of the \( x_c \)- and \( z_c \)-axes.

READ: NEC(MC)

where

NEC(MC) This is a dimensioned integer variable which defines the number of edges the conical cylinder has.

READ: AC(NC,MC), BC(NC,MC), ZC(NC,MC)
AC(NC,MC) This is a double dimensioned real variable which defines the radius of the NCth rim on the $x_c$-axis of the MCth elliptic cylinder.

BC(NC,MC) This is a double dimensioned real variable which defines the radius of the NCth rim on the $y_c$-axis of the MCth elliptic cylinder.

ZC(NC,MC) This is a double dimensioned real variable which defines the z position of the NCth rim along the $z_c$-axis of the MCth elliptic cylinder.

Note that the program will keep increasing the number of cylinders in the solution by the number of calls to this command unless the NC or NX commands are called to reinitialize the cylinder geometry. Also, the ellipticity of a conical structure should remain the same for the entire length of that structure. The most positive rim should be defined first until all NC rims are defined in descending order.
Figure 4.1: Definition of finite cylinder geometry composed of cone frustum segments with elliptic cross section.
4.2 Command CG: Cylinder Geometry

This command enables the user to define the geometry of the finite elliptic cylinder structures to be considered. The geometry is illustrated in Figure 4.2. One call to this command defines one cylinder. The number of cylinders in the structure are automatically counted by the number of calls to this command.

READ: \((XCL(N,MC), N=1,3)\)

\[ \text{where } \]

\(XCL(N,MC)\) This is a doubly dimensioned real variable. It is used to specify the location of the origin of the MCth elliptic cylinder relative to the definition coordinate system. It is input on a single line with the real numbers being the \(x,y,z\) coordinates of the origin which correspond to \(N=1,2,3\), respectively.

READ: \(TCLZ, PCLZ, TCLX, PCLX\)

\[ \text{where } \]

\(TCLZ, PCLZ\) These are real variables. They are input in degrees as spherical angles that define the \(z_c\)-axis of the cylinder coordinate system as if it was a radial vector in the definition coordinate system.

\(TCLX, PCLX\) These are real variables. They are input in degrees as spherical angles that define the \(x_c\)-axis of the cylinder coordinate system as if it was a radial vector in the definition coordinate system.

Note that the new \(x_c\)-axis and \(z_c\)-axis must be defined orthogonal to each other. The new \(y_c\)-axis is found from the cross product of the \(x_c\)- and \(z_c\)-axes.

READ: \(AC(1,MC), BC(1,MC)\)

\[ \text{where } \]

\(AC(1,MC)\) This is a double dimensioned real variable which defines the radius of the MCth elliptic cylinder on the \(x_c\)-axis of the cylinder.

\(BC(1,MC)\) This is a double dimensioned real variable which defines the radius of the MCth elliptic cylinder on the \(y_c\)-axis of the cylinder.

READ: \(ZCN, THTN, ZCP, THTP\)
where

**ZCN** This is a real variable that defines the position the center of the most negative end cap on the \( z_c \)-axis of the cylinder.

**THTN** This is a real variable. It is input in degrees and defines the angle the surface of the most negative end cap makes with the positive \( z_c \)-axis in the \( x_c-z_c \) plane.

**ZCP** This is a real variable that defines the position of the center of the most positive end cap on the \( z_c \)-axis of the cylinder.

**THTP** This is a real variable. It is input in degrees and defines the angle the surface of the most positive end cap makes with the positive \( z_c \)-axis in the \( x_c-z_c \) plane.

Note that the program will keep increasing the number of cylinders in the solution by the number of calls to this command unless the NC or NX commands are called to reinitialize the cylinder geometry.
Figure 4.2: Definition of finite elliptic cylinder geometry.
4.3 Command CM: and CE: Comments

These commands enable the user to place comment cards in the input and output data in order to help identify the computer runs for present and future reference.

READ: \((IR(I), I=1,36)\)

where

\(IR(I)\) This is a CHARACTER*2 dimensioned array used to store the command word and comments. Each card should have CM or CE on them followed by an alphanumeric string of characters. The CM command implies that there will be another comment card following it. The last comment card must have the CE command on it. If there is only one comment card the CE command must be used.

Note that it is possible to place comments to the right of all the command words, if desired.

4.4 Command EN: End Program

This command enables the user to terminate the execution of the scattering code.
4.5 Command GP: Ground Plane

This command enables the user to specify an infinite ground plane in the $x_t-y_t$ plane.

---

READ: LSLAB(MPDX)

---

**LSLAB(MPDX)** This is a dimensioned integer variable. It is used to define the type of plate desired as follows:

- **0** = Perfectly conducting metallic plate
- **-3** = Dielectric half space

Note that if LSLAB(MPDX)=0 the code will skip around the READ statement for the dielectric information, therefore, the next line defining the dielectric properties should not be placed in the input data set.

---

READ: ERSLAB(1,MPDX), TESLAB(1,MPDX), URSLAB(1,MPDX), TMSLAB(1,MPDX)

---

**ERSLAB(1,MPDX)** This is a doubly dimensioned variable. It is used to specify the relative dielectric constant of the half space.

**TESLAB(1,MPDX)** This is a doubly dimensioned variable. It is used to specify the dielectric loss tangent if the number is positive or the conductivity if the number is negative of the half space.

**URSLAB(1,MPDX)** This is a doubly dimensioned variable. It is used to specify the relative permeability constant of the half space.

**TMSLAB(1,MPDX)** This is a doubly dimensioned variable. It is used to specify the permeability loss tangent of the half space.
4.6 Command NC: Next Set of Cylinders
This command enables the user to initialize the cylinder data. All of the cylinders are removed from the problem unless they are respecified following this command.

4.7 Command NG: No Ground Plane
This command enables the user to initialize the infinite ground plane. The ground plane is removed from the problem unless it is respecified following this command.

4.8 Command NP: Next Set of Plates
This command enables the user to initialize the plate data. All of the plates are removed from the problem unless they are respecified following this command.

4.9 Command NS: Next Set of Sources
This command enables the user to initialize the source data. All of the sources are removed from the problem unless they are respecified following the command.

4.10 Command NX: Next Problem
This command enables the user to initialize the commands to their default conditions specified in the list at the beginning of the main program.
4.11 Command PG: Plate Geometry

This command enables the user to define the geometry of the flat plate structures to be considered. The geometry is illustrated in Figure 4.3. One call to this command defines one plate. The number of plates in the structure are automatically counted by the number of calls to this command.

READ: MEP(MP), LSLAB(MP)

where

MEP(MP) This is a dimensioned integer variable. It is used to define the number of corners (or edges) on the MPth plate.

LSLAB(MP) This is a dimensioned integer variable. It is used to define the type of plate desired as follows:

1 = Transparent thin dielectric slab
0 = Perfectly conducting metallic plate
-2 = Dielectric covered plate

Note that if LSLAB(MP)=0 the code will skip to the read statements associated with the corners XX(N,ME,MP). Therefore, the information for the different slab layers should not be put in the data list for the perfectly conducting plate.

READ: NSLAB(MP)

where

NSLAB(MP) This is a dimensioned integer variable. It is used to define the number of dielectric layers on the MPth plate.

READ: DSLAB(NS,MP), ERSLAB(NS,MP), TESLAB(NS,MP), URSLAB(NS,MP), TMSLAB(NS,MP)

where

DSLAB(NS,MP) This is a doubly dimensioned variable. It is used to specify the thickness of the NSth layer.

ERSLAB(NS,MP) This is a doubly dimensioned variable. It is used to specify the relative dielectric constant of the NSth layer.

TESLAB(NS,MP) This is a doubly dimensioned variable. It is used to specify the dielectric loss tangent if the number is positive or the conductivity if the number is negative of the NSth layer.
**URSLAB**(NS,MP) This is a doubly dimensioned variable. It is used to specify the relative permeability constant of the NSth layer.

**TMSLAB**(NS,MP) This is a doubly dimensioned variable. It is used to specify the permeability loss tangent of the NSth layer.

Note there will be **NSLAB**(MP) number of lines of the above data.

**READ:** \((XX(N,ME,MP), N=1,3)\)

**XX**(N,ME,MP) This is a triply dimensioned real variable. It is used to specify the location of the MEth corner of the MPth plate. It is input on a single line with the real numbers being the \(x,y,z\) coordinates of the corner, in the specified coordinate system, which corresponds to \(N=1,2,3\), respectively, in the array. For example, the array will contain the following for plate #1 and corner #2 located at \(x=2., y=4., z=6.\):

\[
XX(1,2,1)=2.
XX(2,2,1)=4.
XX(3,2,1)=6.
\]

This data is input as: 2.,4.,6.

This read statement will be called **MEP**(MP) times so that all the corners are defined. As an example, the input data for the flat plate structure given in Figure 4.3, is given by

\[
4,0 
1., 1., 0. 
-1., 1., 0. 
-1.,-1., 0. 
1.,-1., 0. 
\]

:corners and type of plate

:corner #1

:corner #2

:corner #3

:corner #4.

See elsewhere for further details on how to number the corners. Note that the program will keep increasing the number of plates in the solution by the number of calls to this command unless the NP or NX commands are called to reinitialize the plate geometry.
Figure 4.3: Definition of flat plate geometry.
4.12 Command RT: Rotate-Translate Geometry

This command enables the user to translate and/or rotate the coordinate system used to define the input data in order to simplify the specification of the plate, cylinder, and source geometries. The geometry is illustrated in Figure 4.4.

READ: \((TR(N), N=1,3)\)

where

\(TR(N)\) This is a dimensioned real variable. It is used to specify the origin of the new coordinate system to be used to input the data for the source or the scattering structures. It is input on a single line with the real numbers being the \(x,y,z\) coordinates of the new origin which corresponds to \(N=1,2,3\), respectively.

READ: \(THZP, PHZP, THXP, PHXP\)

where

\(THZP, PHZP\) These are real variables. They are input in degrees as spherical angles that define the \(z\)-axis of the new coordinate system as if it was a radial vector in the reference coordinate system.

\(THXP, PHXP\) These are real variables. They are input in degrees as spherical angles that define the \(x\)-axis of the new coordinate system as if it was a radial vector in the reference coordinate system.

The new \(x\)-axis and \(z\)-axis must be defined orthogonal to each other. The new \(y\)-axis is found from the cross product of the \(x\)- and \(z\)-axis. All the subsequent inputs will be made relative to this new coordinate system, which is shown as \(x_t, y_t, z_t\), unless command RT is called again and redefined.
Figure 4.4: Definition of rotate-translate coordinate system geometry.
4.13 Command SG: Source Geometry

This command enables the user to specify the location and type of source to used. The geometry is illustrated in Figure 4.5 and 4.6. One call to this command defines one source. The number of sources in the problem are automatically counted by the number of calls to this command and the SA command.

**READ:** \( (XSS(N,MS), N=1,3) \)

**READ:** \( THSZ, PHSZ, THSX, PHSX \)

**READ:** \( IMS(MS), HS(MS), HAWS(MS) \)

---

**XSS(N,MS)** This is a doubly dimensioned real array which is used to define the \( x,y,z \) location of the \( M \)th element in the definition coordinate system. Again, a single line of data contains the \( x,y,z \) \((N=1,2,3)\) locations.

**THSZ,PHSZ** These are real variables which are used to define the orientation of the \( M \)th element in the definition coordinate system. They are input in degrees as spherical angles that define a radial direction which is parallel to the \( M \)th current flow for a dipole antenna or which is parallel to the length of an aperture antenna.

**THSX,PHSX** These are real variables which are used to define the orientation of the \( M \)th element in the definition coordinate system. They are input in degrees as spherical angles that define a radial direction which is parallel to the \( M \)th elements aperture width or which is parallel to a slot's width. For a dipole antenna, these angles can be made in a convenient direction.

The \( x \)-axis and \( z \)-axis specified by these angles must be defined orthogonal to each other. The \( y \)-axis is found by the cross product of the \( x \)- and \( z \)-axes.

**IMS(MS)** This is an integer array which is used to define the \( M \)th element's source type. The details of the different types of sources are given elsewhere. The designations are defined as follows:
- \( IMS(MS)<0 \) for an electric element
- \( IMS(MS)>0 \) for a magnetic element
| IMS(MS)| = 1 for a uniform current distribution
| IMS(MS)| = 2 for a piece-wise sinusoidal distribution
| IMS(MS)| = 3 for a TE01 cosine current distribution

**HS(MS)** This is a real array which is used to input the length of the MSth element.

**HAWS(MS)** This is a real array which is used to input the width of the MSth element in the case of an aperture antenna. If HAWS(MS) = 0, then it is assumed to be a dipole.

Note that the units of the variable HS(MS) and HAWS(MS) can be specified by the US command. If wavelength is chosen as the units then all the sources must be specified in wavelengths.

---

**READ:** WMS, WPS

__________________ where __________________

**WMS, WPS** These are real variables used to define the excitation associated with the MSth element. The magnitude is given by WMS and the phase in degrees by WPS.

Note that the program will keep increasing the number of sources in the solution by the number of calls to this command unless the NS or NX commands are called to reinitialize the source geometry.
Figure 4.5: Definition of source geometry for dipole antennas.
Figure 4.6: Definition of source geometry for aperture antennas.
4.14 Command UF: Scale Factor

This command enables the user to scale the linear dimensions that follow the command by the factor specified.

READ: UNITF

where

UNITF This is a real variable that is used as a scale factor for all the linear dimensions that follow the command.

4.15 Command UN: Units of Geometry

This command enables the user to specify the units of all the linear dimensions to be input after the command is called. (The exceptions are the source length HS and width HAWS, and receiver length HR and width HAWR, see command US.)

READ: IUNIT

where

IUNIT This is an integer variable that indicates the units for the input data that follows, such that
1 = meters
2 = feet
3 = inches

4.16 Command US: Units of Source

This command enables the user to specify the units of the source length HS and width HAWS or receiver length HR and width HAWR to be input after the command is called. These variables are in the commands SG, SA, RG, and RA.

READ: IUNST

where

IUNST This is an integer variable that indicates the units for the input data HS, HAWS, HR, HAWR that follows, such that if
0 = wavelengths  
1 = meters  
2 = feet  
3 = inches

Note that if the units are specified to be wavelengths for one source it must be wavelengths for all the sources specified.
4.17 Command VF: Far Zone Volumetric Pattern

This command enables the user to define the far zone volumetric pattern coordinate system, the pattern cut, and the angular range that is desired. The geometry is illustrated in Figure 4.7.

**READ: THCZ, PHCZ, THCX, PHCX**

**THCZ, PHCZ** These are real variables. They are input in degrees as spherical angles that define the $z_p$-axis of the pattern coordinate system as if it was a radial vector in the reference coordinate system.

**THCX, PHCX** These are real variables. They are input in degrees as spherical angles that define the $x_p$-axis of the pattern coordinate system as if it was a radial vector in the reference coordinate system.

Note that the new $x_p$-axis and $z_p$-axis must be defined orthogonal to each other. The new $y_p$-axis is found from the cross product of the $x_p$- and $z_p$-axes.

**READ: LCNPAT, TPPD, TPPV, NPV**

**LCNPAT** This is a logical variable that defines the pattern cut desired, such that

- $T$: The theta angle is held fixed while the phi angle is varied. The theta angle will then be incremented and another cut will be calculated.
- $F$: The phi angle is held fixed while the theta angle is varied. The phi angle will then be incremented and another cut will be calculated.

**TPPD** This is a real variable. It defines the starting angle of the "fixed" angle specified by LCNPAT.

**TPPV** This is a real variable. It defines the incremental angle of the "fixed" angle specified by LCNPAT.

**NPV** This is an integer variable. It defines the number of pattern points of the "fixed" angle specified by LCNPAT.

**READ: TPPS, TPPI, NPN**

**TPPS** This is a real variable. It defines the starting angle of the "varying" angle specified by LCNPAT.
TPPI This is a real variable. It defines the incremental angle of the "varying" angle specified by LCNPAT.

NPN This is an integer variable. It defines the number of pattern points of the "varying" angle specified by LCNPAT.
Figure 4.7: Definition of volumetric pattern coordinate system.
4.18 Command XQ: Execute Code

This command is used to execute the code so that the results may be computed. After execution the code returns for another possible command word.
Chapter 5

Interactive Commands

5.1 Overview

Facilities for interactive programs vary greatly from one operating system to the next with little or no standardization between systems. In spite of this, it was felt that the users of this code would benefit immensely from an interactive mode of operation. In order for the code to have interactive capability without an excessive amount of development time, the developers have used many features of the DEC VAX/VMS operating system. Since many engineers presently have access to the DEC VAX, it is felt that this will lead to reasonable transportability of the interactive mode for this code.

This decision has several ramifications for users of the SHADOW code. It means that the commands described in this chapter do not exist on computers that don't run VAX/VMS Version 4.0 or greater. It also means that this code has been separated into two parts, one standard FORTRAN 77 and the other VMS dependent containing the interactive facility, with a slightly different main program for the non-interactive code.

5.2 Command Descriptions

This section describes the interactive SHADOW commands in detail complete with examples for each. The syntax of the interactive commands is that of the Digital Command Language or DCL and for obvious reasons familiarity with the syntax of DCL is assumed throughout this chapter. For details about the utilities used to perform this DCL style command parsing, readers are referred to the VMS documentation concerning the Command Definition Utility or CDU.
EXIT

Causes the program to exit.

<table>
<thead>
<tr>
<th>FORMAT</th>
<th>EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Qualifiers</td>
<td>Defaults</td>
</tr>
<tr>
<td>None.</td>
<td>None.</td>
</tr>
</tbody>
</table>

restrictions
None.

prompts
None.

command parameters
None.

DESCRIPTION
All output files are closed, and control is returned to DCL.

COMMAND QUALIFIERS
None.

EXAMPLES

```
$ RUN SHADOW
SHADOW>
SHADOW> EXIT
$ 
```

This example shows how to exit the program.
HELP
Displays information about SHADOW commands or help text from any other library you specify.

**FORMAT**

<table>
<thead>
<tr>
<th>HELP help-item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Qualifiers</td>
</tr>
<tr>
<td>/LIBRARY [=library-name]</td>
</tr>
</tbody>
</table>

**restrictions**
The indicated help files must exist.

**prompts**
None.

**command parameters**
help-item
The help-item is a keyword which is the item you want help on.

**DESCRIPTION**
The SHADOW help command adheres to the conventions of VMS help libraries in form and content.

**COMMAND QUALIFIERS**

<table>
<thead>
<tr>
<th>/LIBRARY [=library-name]</th>
</tr>
</thead>
<tbody>
<tr>
<td>/NOLIBRARY</td>
</tr>
</tbody>
</table>

Controls whether an alternate help library will be used in the search for topics. This qualifier must appear immediately after the HELP command or it will be interpreted as part of the help-item. If you specify /NOLIBRARY then no library will be searched.

**EXAMPLES**

**SHADOW> HELP SET OUTPUT**

 (SET OUTPUT help message)

Topic? EXIT

(EXIT help message)

Topic? <RETURN>

**SHADOW> HELP/LIBR=HELPLIB LOGOUT**

(LOGOUT help message from the system help library)
The above examples show how to get help about shadow topics and how to access other VMS help libraries with the HELP command.
**SHADOW**

Initiates the obscuration calculation for the current antenna location and input geometry.

**FORMAT**

<table>
<thead>
<tr>
<th>Command Qualifiers</th>
<th>Defaults</th>
</tr>
</thead>
<tbody>
<tr>
<td>None.</td>
<td>None.</td>
</tr>
</tbody>
</table>

**restrictions**

None. Command may be abbreviated “S”.

**prompts**

None.

**command parameters**

None.

**DESCRIPTION**

The commands which alter parameters, such as SET WINDOW and SET ANTENNA do not initiate shadowing calculations automatically. This is to avoid redundant calculations when several parameters are changed at once. Once desired parameters are set, the SHADOW command performs the obscuration calculations and outputs the result.

**COMMAND QUALIFIERS**

None.

**EXAMPLES**

```
SHADOW> SET ANTENNA
Input antenna location in meters: 11,22,32
Antenna in RCS (meters): 11.00000 22.00000 32.00000
Definit system (meters): 11.00000 22.00000 32.00000
SHADOW> SHAD
Working...
SHADOW> SET ANTENNA
Input antenna location in meters: 10,20,30
Antenna in RCS (meters): 10.00000 20.00000 30.00000
Definit system (meters): 10.00000 20.00000 30.00000
SHADOW> S
Working...
SHADOW>
```
The above commands all calculate the projected shadows for two different antenna locations on given input geometry. The results all go into the same output file, because no "SET OUTPUT" command was executed in between "SHADOW" commands.
SPAWN

Creates a subprocess for executing DCL commands without exiting the SHADOW program. This command is useful for executing DCL commands without reinitializing the context of a SHADOW program session.

FORMAT

SPAWN command-string

Command Qualifiers: None.

Defaults: None.

restrictions

A few restrictions are imposed by VMS.

- The RESOURCE_WAIT state must be enabled for the spawning process.
- Requires TMPMBX or PRMMBX user privileges.
- SPAWN does not manage terminal characteristics.

Command may be abbreviated "$ $", where the blank after the $ is necessary.

prompts

None.

command parameters

command-string

Specifies a DCL command string to be executed in the context of the subprocess. SHADOW will wait until the subprocess completes executing. If command-string is blank, the subprocess will prompt for commands repeatedly.

DESCRIPTION

The details of the spawn command are exactly as documented in the DCL dictionary, volume 2 of the VAX/VMS documentation set.

COMMAND QUALIFIERS

None.

EXAMPLES

SHADOW> SPAWN SHOW USERS

VAX/VMS Interactive Users
11-DEC-1985 08:34:18.72
Total number of interactive users = 5

Username    Process Name    PID    Terminal

54
The above spawn command illustrate how DCL commands may be executed without exiting the SHADOW program.
SET ANTENNALOCATION

Determines the location of the source point, or the center of the far-zone sphere for subsequent shadowing calculations.

<table>
<thead>
<tr>
<th>FORMAT</th>
<th>SET ANTENNA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Qualifiers</td>
<td>Defaults</td>
</tr>
<tr>
<td>None.</td>
<td>None.</td>
</tr>
</tbody>
</table>

restrictions

Its recommended that the antenna not be placed in the interior of a cylinder. Unusual results may occur if this is done.

prompts

Input antenna location in meters:
Input antenna location in feet:
Input antenna location in inches:

command parameters

None.

DESCRIPTION

The antenna location consists of the \((x,y,z)\) components of a vector in the current units and definition coordinate system, set by the SET UNITS and the SET COORDINATE commands, respectively. The command does NOT accept the antenna location on the command line, but prompts for it instead. The input syntax for the numbers is that of an unformatted FORTRAN read.

COMMAND QUALIFIERS

None.

EXAMPLES

SHADOW> SET ANTENNA
Input antenna location in meters: 10,20,30
Antenna in RCS (meters): 10.00000 20.00000 30.00000
Definit system (meters): 10.00000 20.00000 30.00000

This example sets the antenna location to 10,20,30. \((x,y,z)\) in the current units, which are meters.

56
EXAMPLES

SHADOW> SET UNIT FEET
SHADOW> SET ANT
Input antenna location in feet : 10, 20, 30
Antenna in RCS (meters): 3.04800 6.09600 9.14400
Definit system (feet ): 10.00000 20.00000 30.00000
SHADOW>

This example shows how the antenna location is interpreted in the units of feet.
SET COORDINATES

Sets up a coordinate transformation to be applied to subsequent geometry.

<table>
<thead>
<tr>
<th>FORMAT</th>
<th>SET COORDINATES</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Command Qualifiers</td>
</tr>
<tr>
<td></td>
<td>None.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>restrictions</th>
<th>The specified coordinate axes must be orthogonal to one another.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>prompts</th>
<th>Please input a translation vector in feet:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Please input THZP, PHZP, THXP, PHXP in degrees:</td>
</tr>
<tr>
<td>command parameters</td>
<td>None.</td>
</tr>
</tbody>
</table>

DESCRIPTION

The antenna location may be specified relative to an alternative coordinate system. This coordinate system is established via the SET COORDINATES command. It does not affect the pattern cut coordinate system.

COMMAND QUALIFIERS

None.

EXAMPLES

SHADOW> SET COORD
Please input a translation vector in feet : 100,200,300
Please input THZP,PHZP,THXP,PHXP in degrees: 0, -64, 265.5, 45
* The following rotations are used for ALL subsequent inputs:
* VRS(1,1)= -0.70711 VRS(1,2)= -0.70711 VRS(1,3)= 0.00000
* VRS(2,1)= 0.70711 VRS(2,2)= -0.70711 VRS(2,3)= 0.00000
* VRS(3,1)= 0.00000 VRS(3,2)= 0.00000 VRS(3,3)= 1.00000

The above example shows how a default coordinate system may be established. The program echoes the established coordinate axes. These may be re-examined at any time with the SHOW COORDINATE command.
**SET FILL_CHARACTER**

Allows selection of the characters used to fill the output. Can be used to highlight particular elements of a geometry.

**FORMAT**

```
SET FILL [tag-character]
```

<table>
<thead>
<tr>
<th>Command Qualifiers</th>
<th>Defaults</th>
</tr>
</thead>
<tbody>
<tr>
<td>/SEQUENTIAL</td>
<td>None.</td>
</tr>
<tr>
<td>/PLATE=(num[,char])</td>
<td>None.</td>
</tr>
<tr>
<td>/CYLINDER=(num[,char])</td>
<td>None.</td>
</tr>
</tbody>
</table>

**restrictions**

None.

**prompts**

None.

**command parameters**

`tag-character`

Is any single ASCII character. If a lowercase letter is desired, enclose the letter in double quotes, i.e. "a". The default is "X".

**DESCRIPTION**

In order to better trace portions of a geometry through the shadowing process, the ability to tag a particular cylinder or plate has been added. The tag setting remains in effect until altered by a subsequent “SET FILL” command. The highlighted plate or cylinder appears in its entirety in the output regardless of its actual position in the hierarchy of obscuration. This allows the user to be absolutely certain of the shadowing caused by the particular highlighted geometry.

There are three tagging modes available. One is sequential tagging. In this mode, the code attempts to assign a unique character in the output to each plate/cylinder in the input. Plates are numbered beginning with “A” and increasing through the ASCII character sequence, and cylinders are treated the same way beginning with “1”.

The second mode causes all parts of the geometry to be shaded with a single specified character such as “X”. In this total obscuration mode, any one part of the input geometry is not easily identified — rather the the total obscuration is presented homogeneously. It is specified using SET FILL without qualifiers. The third mode is the same as the second mode, but with the added feature of one single plate (or cylinder) highlighted with a different character. In this mode the relation of one part of the geometry to the rest is clearly...
visible. This mode can be very helpful when isolating particular parts of a geometry that are shadowing the source.

<table>
<thead>
<tr>
<th>COMMAND QUALIFIERS</th>
<th>/SEQUENTIAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>The /SEQUENTIAL qualifier selects the first mode of obscuration, sequential tagging of the input geometry. This qualifier may not be specified with a tag-character parameter nor with any of the other qualifiers.</td>
<td></td>
</tr>
<tr>
<td>/PLATE=num</td>
<td></td>
</tr>
<tr>
<td>/PLATE=(num, char)</td>
<td></td>
</tr>
<tr>
<td>The /PLATE qualifier selects the third mode of obscuration, homogeneous tagging with highlighting of a particular plate. The num argument is the number of the plate to be tagged. It is a required argument. The char argument is the ASCII character to be used when tagging the plate. It is optional, and defaults to &quot;P&quot; if unspecified. This qualifier may not be specified in combination with other qualifiers. It is mutually exclusive with the /CYLINDER qualifier.</td>
<td></td>
</tr>
<tr>
<td>/CYLINDER=num</td>
<td></td>
</tr>
<tr>
<td>/CYLINDER=(num, char)</td>
<td></td>
</tr>
<tr>
<td>The /CYLINDER qualifier selects the third mode of obscuration, homogeneous tagging with highlighting of a particular cylinder. It works exactly like the /PLATE qualifier. The num argument is the number of the cylinder to be tagged. It is a required argument. The char argument is the ASCII character to be used when tagging the cylinder. It is optional, and defaults to &quot;C&quot; if unspecified. This qualifier may not be specified in combination with other qualifiers. It is mutually exclusive with the /PLATE qualifier.</td>
<td></td>
</tr>
</tbody>
</table>

**EXAMPLES**

SHADOW> set fill
No individual plates/cylinders are tagged
All geometry marked by [X]

SHADOW> set fill $ 
No individual plates/cylinders are tagged
All geometry marked by [$]

SHADOW> set fill * /plate=6
Plate 6 is tagged with [P]
All other geometry tagged with [*]

SHADOW> set fill * /plate=(7,%)  
Plate 7 is tagged with [%]  
All other geometry tagged with [*]

SHADOW> set fill Q /cyl=(2,)$  
Cylinder 2 tagged with [$]  
All other geometry tagged with [Q]

SHADOW> set fill /plate=9 /cyl=4  
%CLI-W-CONFLICT, illegal combination of command elements

SHADOW> set fill Q /cyl=(2,)$ /seq  
%CLI-W-MAXPARM, too many parameters - reenter command with fewer parameters

SHADOW> set fill /seq ! Q /cyl=(2,)$ /seq  
All cylinders/plates sequentially tagged

The above examples are obvious except possibly the last three. They show that the qualifiers are not allowed in combination, that the /SEQUENTIAL qualifier does not allow specification of a fill character, and that the DCL syntax ignores everything after an exclamation point.
## SET INPUT_SET

Reads an input set from a named file

<table>
<thead>
<tr>
<th>FORMAT</th>
<th>SET INPUT_SET filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Qualifiers</td>
<td>Defaults</td>
</tr>
<tr>
<td>None.</td>
<td>None.</td>
</tr>
</tbody>
</table>

### restrictions

The named input file must exist.

### prompts

input set:

### command parameters

filename

The name of the input set. It may be any valid VMS filename, including a logical name. The default filetype is .INP.

### DESCRIPTION

The set output command has the dual role of designating an input file and simultaneously causing that input set to be read and prepared for subsequent shadow commands. The current output files are NOT affected by this command so that several outputs may be concatenated. Normally though, this command would be entered after a SET OUTPUT command.

### COMMAND QUALIFIERS

None.

### EXAMPLES

SHADOW> SET OUT AN5S1  
Plotting file is: USER1:[RJM.NAS]AN5S1.PLT;1  
Printer file is: _NLA0:[FOR007.DAT;  
Input echo file: USER1:[RJM.NAS]AN5S1.LIS;1  
SHADOW> SET INPUT AN5S1  
The current input set is  
USER1:[RJM.NAS]AN5S1.INP;1

The SET OUTPUT command is used to set the output files – the printer output is discarded by default. The input set AN5S1.INP is then read and processed by the SET INPUT command.
SET KEYPAD_MODE

Causes the keypad state to change to non-numeric.

<table>
<thead>
<tr>
<th>FORMAT</th>
<th>SET [NO]KEYPAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Qualifiers</td>
<td>Defaults</td>
</tr>
<tr>
<td>None.</td>
<td>None.</td>
</tr>
</tbody>
</table>

restrictions

None.

prompts

None.

command parameters

None.

DESCRIPTION

The keypad of most DEC terminals can be in one of two states, numeric mode or keypad mode. In numeric mode, the keypad buttons represent the numbers and symbols printed on the keys. In keypad mode, the keys may be defined to provide functions, in much the same way as they do in DCL.

SET KEYPAD enables the defined-key feature of SHADOW, and SET NOKEYPAD returns the keypad to numeric-entry mode.

The keypad definitions are made in a session startup file called SHADOW.KPD; in the current default directory.

EXAMPLES

SHADOW> SET KEYPAD
The keyboard is in keypad mode.

SHADOW> SET NOKEYPAD
The keyboard is not in keypad mode.
**SET OUTPUT**

Determines the names of new output files and closes current output files.

<table>
<thead>
<tr>
<th>FORMAT</th>
<th>SET OUTPUT filename</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Command Qualifiers</td>
</tr>
<tr>
<td></td>
<td>/PLOTTABLE</td>
</tr>
<tr>
<td></td>
<td>/PRINTABLE</td>
</tr>
<tr>
<td></td>
<td>/ECHOING</td>
</tr>
</tbody>
</table>

**restrictions**
The filename must be a valid VMS filename.

**prompts**
.filename:

**command parameters**
filename

The name(s) of the newly created output file(s).

**DESCRIPTION**
There are three different outputs from the shadow program. One is an echo of the input set from the input processor. Another is a line printer output of the shadow map. The third is an output suitable for input to a separate plotting program. The set output command opens these files for the code. The name of the file opened is specified as the filename parameter. The filetypes are set by the command automatically, so that only the filename need be specified.

<table>
<thead>
<tr>
<th>COMMAND QUALIFIERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>/PLOTTABLE</td>
</tr>
<tr>
<td>/NOPLOTTABLE</td>
</tr>
</tbody>
</table>

Causes a plottable output file to be produced. This is the default. Specifying /NOPLOT will override this default.

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>/PRINTABLE</td>
</tr>
<tr>
<td>/NOPRINTABLE</td>
</tr>
</tbody>
</table>

Causes an output file to be produced which is suitable for printing on a standard line printer. /NOPRINT is the default. Specifying /PRINT will override this default.

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>/ECHOING</td>
</tr>
<tr>
<td>/NOECHOING</td>
</tr>
</tbody>
</table>

Causes the input echo to be saved in a file when a new input set is
read. /ECHOING is the default. Specifying /NOECHO will override this default.

EXAMPLES

SHADOW> SET OUT AN5S1
Plotting file is: USER1:[RJM.NAS]AN5S1.PLT;1
Printer file is: _NLAO[:]FOR007.DAT;
Input echo file: USER1:[RJM.NAS]AN5S1.LIS;1
SHADOW> SET OUT AN5S1 /PRINT
Plotting file is: USER1:[RJM.NAS]AN5S1.PLT;2
Printer file is: USER1:[RJM.NAS]AN5S1.PRT;1
Input echo file: USER1:[RJM.NAS]AN5S1.LIS;2
SHADOW> SET OUT AN5S1 /NOPLOT /NOECHO /PRINT
Plotting file is: _NLAO[:]FOR010.DAT;
Printer file is: USER1:[RJM.NAS]AN5S1.PRT;2
Input echo file: _NLAO[:]FOR008.DAT;
SHADOW> SET OUT AN5S1
Plotting file is: USER1:[RJM.NAS]AN5S1.PLT;3
Printer file is: _NLAO[:]FOR007.DAT;
Input echo file: USER1:[RJM.NAS]AN5S1.LIS;3

The above examples show the operation of the SET OUTPUT command. Note that the printer file is not produced by default, and the device _NLAO: (the null device) is where the output is discarded.
**SET PATTERN\_CUT**

Specifies the pattern cut coordinate system.

<table>
<thead>
<tr>
<th>FORMAT</th>
<th>SET PATTERN_CUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Qualifiers</td>
<td>Defaults</td>
</tr>
<tr>
<td>None.</td>
<td>None.</td>
</tr>
</tbody>
</table>

**restrictions**
The specified coordinate axes must be orthogonal.

**prompts**
Please input THZP,PHZP,THXP,PHXP in degrees:

**command parameters**
None.

**DESCRIPTION**
The shadow map window is specified relative to the pattern-cut coordinate system. This system can be changed to facilitate easier specification of this window relative to the blocking object coordinate system.

**COMMAND QUALIFIERS**
None.

**EXAMPLES**

SHADOW> SET PAT
Please input THZP,PHZP,THXP,PHXP in degrees: 0, -64, 265.5, 46, 0, 135
* The following rotations are used for ALL subsequent inputs: *
* VPC(1,1) = -0.70711 VPC(1,2) = -0.70711 VPC(1,3) = 0.00000 *
* VPC(2,1) = 0.70711 VPC(2,2) = -0.70711 VPC(2,3) = 0.00000 *
* VPC(3,1) = 0.00000 VPC(3,2) = 0.00000 VPC(3,3) = 1.00000 *

The pattern-cut coordinate system shown has been set up.
**SET SCALE.FACTOR**

Sets a new value for the uniform scale factor.

<table>
<thead>
<tr>
<th>FORMAT</th>
<th>SET SCALE.FACTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Qualifiers</td>
<td>Defaults</td>
</tr>
<tr>
<td>None.</td>
<td>None.</td>
</tr>
</tbody>
</table>

**restrictions**

The scale factor may not be specified on the command line.

**prompts**

Please input a uniform scale factor:

**command parameters**

None.

**DESCRIPTION**

In order to allow for more flexibility in specifying input, an additional scale factor may be applied to numerical inputs. The default value of this command is 1.

**COMMAND QUALIFIERS**

None.

**EXAMPLES**

```
SHADOW> SET SCALE
Please input a uniform scale factor: 5.5
The uniform scale factor is 5.50000000
```

The uniform scale factor has been changed to 5.5.
SET UNITS

Sets the default units for the entry of numeric values. Allowable units are Meters, Feet, Inches.

<table>
<thead>
<tr>
<th>FORMAT</th>
<th>SET UNITS keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Qualifiers</td>
<td>Defaults</td>
</tr>
<tr>
<td>None.</td>
<td>None.</td>
</tr>
</tbody>
</table>

restrictions

None.

prompts

inches, feet, or meters:

command parameters

Keyword may be one of the following:

- METERS
- FEET
- INCHES

DESCRIPTION

When the antenna location is set, these are the units applied to the specified position. Internal calculations are always done in meters.

COMMAND QUALIFIERS

None.

EXAMPLES

SHADOW> SET UNI FEET

This example sets the default units to feet.
**SET WINDOW**

Sets parameters for windowing of the output.

<table>
<thead>
<tr>
<th>FORMAT</th>
<th>SET WINDOW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Qualifiers</td>
<td>Defaults</td>
</tr>
<tr>
<td>None.</td>
<td>None.</td>
</tr>
</tbody>
</table>

**restrictions**

The maximum span of theta must be less than 180 degrees. The maximum span of phi must be less than 360 degrees. The maximum resolution is a function of the specified range for both theta and phi. None of these parameters is specified on the command line.

**prompts**

Please enter a new range for theta (lower,higher):
Please enter a new THETA resolution in degrees/pixel:
Please enter a new range for phi (lower,higher):
Please enter a new PHI resolution in degrees/pixel:

**command parameters**

None.

**DESCRIPTION**

In order to be more flexible on the presentation of the output, a windowing feature was included so that portions of theta-phi space may be mapped onto a larger output surface. The set window command does this by prompting for the desired range of displayed theta and phi, and the desired levels of resolution. The default window displays the entire range of theta and phi at a resolution of 2 degrees/pixel in both directions.

**COMMAND QUALIFIERS**

None.

**EXAMPLES**

**SHADOW> SET WINDOW**

The current range of theta in degrees is 0.0000000E+00 to 180.0000 with a resolution of 2.000000 degrees/pixel.
The current range of phi in degrees is 0.0000000E+00 to 360.0000 with a resolution of 2.000000 degrees/pixel.
Please enter a new range for theta (lower,higher): 30.40
Please enter a new THETA resolution in degrees/pixel: .6
Please enter a new range for phi (lower,higher): 45.90
Please enter a new PHI resolution in degrees/pixel: .5

The current range of theta in degrees is 30.00000 to 40.00000 with a resolution of 0.5000000 degrees/pixel.
The current range of phi in degrees is 45.00000 to 90.00000 with a resolution of 0.5000000 degrees/pixel.

The set window command above first displays the current window settings (which also happen to be the default settings), then prompts for new values. The new values are then also shown.
**SHOW ANTENNA_LOCATION**
Display the current antenna position.

<table>
<thead>
<tr>
<th>FORMAT</th>
<th>SHOW ANTENNA_LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Qualifiers</td>
<td>Defaults</td>
</tr>
<tr>
<td>None.</td>
<td>None.</td>
</tr>
</tbody>
</table>

| restrictions | None. |

| prompts | None. |

| command parameters | None. |

**DESCRIPTION**
The antenna location is displayed in both the current default units and the Reference Coordinate System.

**COMMAND QUALIFIERS**
None.

**EXAMPLES**

```
SHADOW> SHO ANT
Antenna in RCS (meters):  2.00000   3.00000   4.00000
Definit system (meters):  2.00000   3.00000   4.00000
```

This command displays the current antenna location in both the reference coordinate systems (RCS) and the current default units, which are also meters in this example.
**SHOW COORDINATES**
Displays the default transformation applied to antenna placement commands.

<table>
<thead>
<tr>
<th>FORMAT</th>
<th>SHOW COORDINATES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Qualifiers</td>
<td>Defaults</td>
</tr>
<tr>
<td>None.</td>
<td>None.</td>
</tr>
</tbody>
</table>

**restrictions**
None.

**prompts**
None.

**command parameters**
None.

**DESCRIPTION**
The antenna location is input in terms of an antenna coordinate system. This command displays the orientation of this system.

**COMMAND QUALIFIERS**
None.

**EXAMPLES**

```
SHADOW> SHOW COORD
* VRS(1,1)= 1.00000  VRS(1,2)= 0.00000  VRS(1,3)= 0.00000  *
* VRS(2,1)= 0.00000  VRS(2,2)= 1.00000  VRS(2,3)= 0.00000  *
* VRS(3,1)= 0.00000  VRS(3,2)= 0.00000  VRS(3,3)= 1.00000  *
```

In this example, the antenna coordinate system is coincident with the reference coordinate system.
SHOW FILL CHARACTER
Displays the current output fill modes.

<table>
<thead>
<tr>
<th>FORMAT</th>
<th>SHOW FILL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Qualifiers</td>
<td>Defaults</td>
</tr>
<tr>
<td>None.</td>
<td>None.</td>
</tr>
</tbody>
</table>

restrictions
None.
prompts
None.
command parameters
None.

DESCRIPTION
The output may be generated in one of three modes. For a detailed description of the possible modes, see the SET FILL command.

COMMAND QUALIFIERS
None.

EXAMPLES
SHADOW> SHOW FILL
Plate 6 is tagged with [P]
All other geometry tagged with [*]

In the above example, the sixth plate of the input set is tagged with the ASCII character "P". The SET FILL command has many more examples.
SHOW INPUT_SET

Displays the name of the file from which the current geometry was defined.

<table>
<thead>
<tr>
<th>FORMAT</th>
<th>SHOW INPUT_SET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Qualifiers</td>
<td>Defaults</td>
</tr>
<tr>
<td>None.</td>
<td>None.</td>
</tr>
</tbody>
</table>

restrictions
None.

prompts
None.

custom parameters
None.

DESCRIPTION
The input set is determined with the SET INPUT command. The SHOW INPUT command echoes this input set filename.

COMMAND QUALIFIERS
None.

EXAMPLES

SHADOW> SHOW INPUT
The current input set is
USER1:[RJM.NAS]AN5S1.INP;1
# SHOW KEYPAD\_MODE

Displays the current state of the keyboard.

<table>
<thead>
<tr>
<th>FORMAT</th>
<th>SHOW KEYPAD_MODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Qualifiers</td>
<td>Defaults</td>
</tr>
<tr>
<td>None.</td>
<td>None.</td>
</tr>
</tbody>
</table>

| restrictions | None. |
| prompt | None. |

| command parameters | None. |

**DESCRIPTION**
The keypad of most DEC terminals can be in one of two states, numeric mode or keypad mode. In numeric mode, the keypad buttons represent the numbers and symbols printed on the keys. In keypad mode, the keys may be defined to provide functions, in much the same way as they do in DCL. The keypad definitions are established by a startup file called SHADOW.KPD in the current default directory.

**COMMAND QUALIFIERS**
None.

**EXAMPLES**

```
SHADOW> SHOW KEYPAD
The keyboard is not in keypad mode.
```

The keypad was not in keypad mode in this example.
SHOW OUTPUT
Displays the names of the current output files.

<table>
<thead>
<tr>
<th>FORMAT</th>
<th>SHOW OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Qualifiers</td>
<td>Defaults</td>
</tr>
<tr>
<td>None.</td>
<td>None.</td>
</tr>
</tbody>
</table>

restrictions
None.

prompts
None.

command parameters
None.

DESCRIPTION
There are three possible output files produced by the shadow program. One is for plotting with a separate plotting program and has a filetype of .PLT. The second is a line-printer formatted output with a filetype of .PRT. The third is the input set listing echo, which may be redirected into a file. Its filetype is .LIS.

COMMAND QUALIFIERS
None.

EXAMPLES
SHADOW> SET OUTPUT EXAMPLE3 /PRINT
SHADOW> SHOW OUTPUT
Plotting file is: USER1:[RJM.NAS]EXAMPLE3.PLT;1
Printer file is: USER1:[RJM.NAS]EXAMPLE3.PRT;1
Input echo file: USER1:[RJM.NAS]EXAMPLE3.LIS;1

This example shows how a SET OUTPUT command creates the names shown for output files. See the SET OUTPUT command description for more details.
### SHOW PATTERN_CUT

Displays the pattern-cut coordinate system transformation matrix.

<table>
<thead>
<tr>
<th>FORMAT</th>
<th>SHOW PATTERN_CUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Qualifiers</td>
<td>Defaults</td>
</tr>
<tr>
<td>None.</td>
<td>None.</td>
</tr>
</tbody>
</table>

| restrictions | None. |

| prompts | None. |

| command parameters | None. |

#### DESCRIPTION

The shadow map window is specified relative to the pattern-cut coordinate system. This system can be changed to facilitate easier specification of this window relative to the blocking object's coordinate system, that is, the reference coordinate system. For more information, see the SET PATTERN command on page 66.

#### COMMAND QUALIFIERS

None.

#### EXAMPLES

```
SHADOW> SHOW PATT
* The following rotations are used for ALL subsequent inputs:  *
* VPC(1,1) = -0.70711 VPC(1,2) = -0.70711 VPC(1,3) = 0.00000  *
* VPC(2,1) = 0.70711 VPC(2,2) = -0.70711 VPC(2,3) = 0.00000  *
* VPC(3,1) = 0.00000 VPC(3,2) = 0.00000 VPC(3,3) = 1.00000  *
```

The pattern-cut coordinate system shown has been set up.
SHOW SCALE_FACTOR
Displays the uniform scale factor currently in effect.

<table>
<thead>
<tr>
<th>FORMAT</th>
<th>SHOW SCALE_FACTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Qualifiers</td>
<td>Defaults</td>
</tr>
<tr>
<td>None.</td>
<td>None.</td>
</tr>
</tbody>
</table>

restrictions
None.

prompts
None.

custom parameters
None.

DESCRIPTION
The SET SCALE_FACTOR command can set a uniform scale factor on subsequent antenna inputs. It allows an extra scaling on the inputs.

COMMAND QUALIFIERS
None.

EXAMPLES

SHADOW> SHOW SCALE
The uniform scale factor is 1.00000000

The above scale factor is the default. It has not been changed with SET SCALE.
**SHOW UNITS**

Displays the current units in effect. Valid units are meters, feet, and inches.

<table>
<thead>
<tr>
<th>FORMAT</th>
<th>SHOW UNITS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Command Qualifiers</td>
</tr>
<tr>
<td></td>
<td>None.</td>
</tr>
</tbody>
</table>

| restrictions | None. |
| prompts | None. |
| command parameters | None. |

**DESCRIPTION**

There are three different units in which antenna locations may be specified. This command displays the units currently in effect. The SET UNITS command changes the default units.

**COMMAND QUALIFIERS**

None.

**EXAMPLES**

```
SHADOW> SHOW UNITS
The current units are feet
```
SHOW WINDOW
Displays the current window parameters.

<table>
<thead>
<tr>
<th>FORMAT</th>
<th>SHOW WINDOW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Qualifiers</td>
<td>None.</td>
</tr>
<tr>
<td>Defaults</td>
<td>None.</td>
</tr>
</tbody>
</table>

restrictions
None.

prompts
None.

command parameters
None.

DESCRIPTION
The output can be windowed onto a smaller range of theta or phi, with any desired resolution. The parameters for this windowing are established by the SET WINDOW command.

COMMAND QUALIFIERS
None.

EXAMPLES

SHADOW> SHOW WIND
The current range of theta in degrees is 0.0000000E+00 to 180.0000
with a resolution of 2.000000 degrees/pixel.
The current range of phi in degrees is 0.0000000E+00 to 360.0000
with a resolution of 2.000000 degrees/pixel.

In this case, the window is set to its default range with a resolution of two degrees/pixel.
Chapter 6

Interpretation of the Output

The final product of the obscuration code, SHADOW, is a map of the projected shadow of a defined object onto the far zone sphere with its center at the antenna location. The map is composed of pixels with the size and range specified by the user. The obscuration code provides complete control over the parameters needed to define the map and provides a line printer output or a plottable file that can be used by an external plotting code. This chapter outlines the details of defining, obtaining, and interpreting the shadow map.

For this discussion, the far zone sphere can be viewed as being ironed out into a flat plane, that is, a Mercator's projection with the angle phi along the x axis and the angle theta along the y axis. Using the VF non-interactive command or the SET WINDOW interactive command, the user can choose starting angles, incremental step size which is the resolution of the map, and the total number of steps or pixels for both the theta and phi angles. This, of course, also dictates the stopping angles of the map which it computes. The default is for theta to vary from 0 to 180 degrees in steps of 2 degrees for a total of 91 pixels, and for phi to vary from 0 to 360 in steps of 2 for a total of 181 pixels. The interactive command SET WINDOW allows these parameters to be changed at any time during a session. It asks for the starting and stopping angle and the resolution which is the step size and it computes the number of pixels for each angle. These angles are defined with respect to the pattern coordinate system, which is specified by the first set of angles in the VF command or by the SET PATTERN command. The default is for the pattern coordinate system to be the same as the reference coordinate system.

As discussed in Chapter 2, the code computes the shadow by first projecting the objects border onto the far zone sphere and then filling in between the borders. A pixel is considered to be filled if the border at least passes through more than half the distance to the center of a pixel. It determines this by rounding the theta and phi angles defining the border to the nearest integer with respect to the resolution size of the pixel, which is the step size. This sometimes appears to produce a ragged border around the edges of the shadow if the border is very curved. Note that a straight edged plate projects a shadow that is curved in border. In addition, this is dependent on the coordinate system in which the shadow is viewed. Chapter 7 presents specific examples of these types of maps.

The shadow is represented by an ASCII character being placed in an array corresponding to the integerized theta and phi angles. A clear viewing point is left blank. The choice of the character that is placed in the pixel can be controlled by the user. The default is for an "X" to be used as a fill character. Interactively this can be changed using the SET FILL
command. Noninteractively, these are hard-wired into the source code.

For debugging purposes or so that the user can get a feel for which plates and cylinders are shadowing which regions of space a highlighting feature has been provided. The SET FILL/SEQUENTIAL command tags each plate and cylinder with its own unique fill character. The first plate starts with "A" and each succeeding plate is incremented up by one ASCII character. The first cylinder starts with "1" and each succeeding cylinder is incremented up by one ASCII character. Note that if there are a lot of plates and/or cylinders, the fill characters will eventually get into some of the more seldom used ASCII characters. Also note that in this mode of filling, the code superimposes the latest calculated shadow for a plate or cylinder on top of the shadow map. This means that the character in a pixel for a finished map will represent the last object that the code calculated a shadow for and not the object that is located closest to the observer.

In order to get around the ambiguous behavior of highlighting the plates and cylinders by order of processing rather than by location, the user can instead use the standard fill character for all plates and cylinders and highlight one particular specified object. The command SET FILL/PLATE = (number,character) or SET FILL/CYLINDER = (number,character) will highlight the chosen plate or cylinder against the regular fill character. The plate or cylinder options are mutually exclusive. It represents the shadow of the whole plate or cylinder that is tagged. A non interactive command has not been provided for these fill features. The user can change the fill characters and mode in the INIT subroutine.

The output that the user sees can come in three forms. The first type of output comes from an echo of the command set that is read from the input file on logical unit #5. The output is sent to logical unit #6, which is normally assigned to a default file type of .LIS on a VAX in the interactive mode. An ASCII file of the shadow map is written to logical unit #7, which is normally assigned to a default file type of .PRT on a VAX. A binary file of the shadow map that can be used to transfer information to another code to plot the map is sent to logical unit #10, which is normally assigned to a default file type of .PLT on a VAX.

In the interactive mode, the input set can be opened using the SET OUTPUT command. The output files can be opened and closed using the SET OUTPUT /ECHOING, /PRINTABLE, /PLOTTABLE commands, respectively. In the non-interactive mode, they can be controlled by using system commands, such as ASSIGN on the VAX. In the interactive mode, the output files should generally be set first, so that the code will have the desired information as to where to sent the echo back information. In addition, once the code is run and it is desired to see the results, it is possible to print or plot the results using the SPAWN ("$") command. The files that are desired to be printed or plotted, however, must be closed first, that is, the SET OUTPUT command should be given again reassigning the files to another name, a null device, or the printing device. This will close the files and allow them to be accessed. Of course, it is important to remember to reopen them after the user is finished and wants to run more results. Presently, the echo, printable, and plottable map files will accumulate information until they are closed.

Generally, the code will be used to produce plottable files of the shadow maps with the printable file being used for debug purposes. Plotted maps are small and nicer to look at. Unfortunately, graphical routines are presently system dependent. A plotting code for a NCAR [5], has been provided, however, in Chapter 13. This is one example of how the
data of the shadow map can be plotted. Examples of both the printed and plotted maps are illustrated in the examples of Chapter 7. It should be noted that due to the limited amount of space across the width of a line printer, a printed map will be broken up into widths that will fit onto the width of the paper if it is too wide. The map will come out in as many strips as necessary to produce the whole map. Plotted maps should not have this problem since the individual pixels can be graphed very close together.
Chapter 7

Examples

The following examples are used to illustrate the various features of the SHADOW computer code. Each example is designed to show how a set of non-interactive and interactive commands can be put together to solve a problem. The beginner can use the examples in this chapter to learn more about the code. In addition, these examples can be used to ensure that the code is operating correctly on your system. These examples were run on a DEC VAX 11/780 computer using version 4 of the VMS operating system.

The shadow maps shown here are presented mostly with the line printer output, since this is generally the most transportable. Plotted output would normally be preferred in a design situation. A few examples of this type of output are also given.
7.1 Example 1: A Plate

The first example is a four-cornered plate centered at the origin and situated in the X-Y plane. The antenna is located on the positive $Z$ axis. It was generated with the following input files and commands. The commands were:

$\text{RUN SHADOW}
\text{SHADOW> SET OUT PLAEX1/NOPLOT/PRINT}
\text{SHADOW> SET INP PLAEX}
\text{SHADOW> SET UNI METERS}
\text{SHADOW> SET WIND 90, 180 1.0 0., 360 5.}
\text{SHADOW> SET ANT}
\text{SHADOW> 0,0,8}
\text{SHADOW> SHADOW}
\text{SHADOW> EXIT}$

The input set defining the plate was the following:

CM: SIMPLE PLATE TEST SET
CE: RCS INPUT SET
UN: 1
PG: THE PLATE IS 400 SQUARE-METERS.
4,0
-10.0, +10.0, 0.0
-10.0, -10.0, 0.0
+10.0, -10.0, 0.0
+10.0, +10.0, 0.0
XQ:
EN:
The output this produced was the following:

<table>
<thead>
<tr>
<th>PSI</th>
<th>0.00</th>
<th>20.00</th>
<th>40.00</th>
<th>60.00</th>
<th>80.00</th>
<th>100.00</th>
<th>120.00</th>
<th>140.00</th>
<th>160.00</th>
<th>180.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>REAL (KCE)</td>
<td>(0.0000, 0.0000, 0.0000)</td>
<td>IN METERS</td>
<td>INPUT DAT: USER1: (00N, 00W, 00FLAT, INF)</td>
<td>TEKTA (DEGREES)</td>
<td>00.00</td>
<td>100.00</td>
<td>110.00</td>
<td>120.00</td>
<td>130.00</td>
<td>140.00</td>
</tr>
<tr>
<td>0.00</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>10.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>110.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>120.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>130.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>140.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>150.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>160.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>170.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>180.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>190.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>200.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>210.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>220.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>230.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>240.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>250.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>260.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>270.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>280.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>290.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>300.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>310.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>320.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>330.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>340.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>350.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>360.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>370.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>380.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example 2: A Different Plate

This example is another four-cornered plate, but this time the antenna is located at the origin, and the plate is centered along the positive Y axis and is normal to it.

The commands were:

```
$ RUN SHADOW
SHADOW> SET OUT PLAEX2/NOPLOT/PRINT
SHADOW> SET INP PLAEX2
SHADOW> SET UNI METERS
SHADOW> SET WIND
          0,  180
          2.0
          0., 180
          6.
SHADOW> SET ANT
          0,0,0
SHADOW> SHADOW
SHADOW> EXIT
$ EXIT
```

The input set defining the plate was the following:

```
CM:  SIMPLE PLATE TEST SET
CE:  RCS INPUT SET
UN:
  1
PG:  THE PLATE IS 400 SQUARE-METERS.
  4 0
  -10.0,  8, +10.0
  -10.0,  8, -10.0
  +10.0,  8, -10.0
  +10.0,  8, +10.0
XQ:
EN:
```
The output generated by the code was the following:

```
ANTENNA (RCS) = ( 0.0000, 0.0000, 0.0000 ) IN METERS  INPUT SET: USER1: [RXN.RAS.MAP]PLAXE3.INP;3

<table>
<thead>
<tr>
<th>THETA (DEGREES)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00  20.00  40.00  60.00  80.00  100.00  120.00  140.00  160.00  180.00</td>
</tr>
<tr>
<td>*     *     *     *     *     *     *     *     *     *</td>
</tr>
</tbody>
</table>
```

```

```
```
7.3 Example 3: The First Plate Revisited

The current example is deceptive. Both the input geometry and the source location are identical with the first plate example, but the obscuration output is identical to the second example! A closer of the input sets reveals the the two examples are really the same geometry, but defined in different orientations with respect to the Reference Coordinate System. The third example takes advantage of this fact and uses the SET PATTERN_CUT command to reorient the coordinate system of the antenna. The result is that while the geometry is defined the same as the first example, the output resembles the second example. The commands to generate the example were:

```$ RUN SHADOW
SHADOW> SET OUT PLAEX3/NOPLOT/PRINT
SHADOW> SET INP PLAEX
SHADOW> SET UNI METERS
SHADOW> SET WIND
0,180
2.0
0.,180
5.
SHADOW> SET ANT
0,0,8
SHADOW> SET PATT
90.,+90.,90.,0.
SHADOW> SHADOW
SHADOW> EXIT
$ EXIT
```

The input set defining the plate was the same one used in example one. It is:

```CM: SIMPLE PLATE TEST SET
CE: RCS INPUT SET
UN:
1
PG: THE PLATE IS 400 SQUARE-METERS.
4 0
-10.0, +10.0, 0.0
-10.0, -10.0, 0.0
+10.0, -10.0, 0.0
+10.0, +10.0, 0.0
XQ:
EN:
```
The output generated by the code was the following:

```plaintext
ANTENNA (RCS) = ( 0.0000, 0.0000, 0.0000 ) IN METERS
INPUT NET: ODRS1: [RRM.RAM.NAM]FLUX.REF

<table>
<thead>
<tr>
<th>Theta (Degrees)</th>
<th>0.00</th>
<th>20.00</th>
<th>40.00</th>
<th>60.00</th>
<th>80.00</th>
<th>100.00</th>
<th>120.00</th>
<th>140.00</th>
<th>160.00</th>
<th>180.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>PII</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>45.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>55.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>65.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>75.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>85.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>95.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>105.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>110.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>115.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>120.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>125.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>130.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>135.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>140.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>145.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>150.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>155.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>160.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>165.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>170.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>175.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>180.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

90
7.4 Example 4: A Non-Interactive Version of Example 1

This example illustrates an input set for non-interactive use of the code. The main program has been changed to the non-interactive version and non-interactive subroutines were not linked into the code. The input set is the same as Example 1, except that the source and window have been defined using the SG and VF commands, respectively. Note that these commands can also be used in the interactive mode also to hard wire the antenna location and window as a default case. The output is not shown here because it is identical to that of Example 1.

The input set defining the plate is the following:

CM: SIMPLE PLATE TEST SET
CE: RCS INPUT SET
UN:
1
PG: THE PLATE IS 400 SQUARE-METERS.
4 0
-10.0, +10.0, 0.0
-10.0, -10.0, 0.0
+10.0, -10.0, 0.0
+10.0, +10.0, 0.0
SG: THE SOURCE LOCATION
0.0, 0., 8.
0.0, 0., 90., 0.
-1.0, 5.0.
1.0.
VF: WINDOW SIZE
0.0, 90., 0.
1.0, .2., .91
0.2, 181
XQ: 
EN:
Example 5: An Elliptic Cylinder

This example consists of one elliptic cylinder centered on the origin with its axis directed along the Y axis. Three different source locations are presented with this single example.

The commands were:

```bash
$ RUN SHADOW
SHADOW> SET OUT CYLEX1/HOPLOT/PRINT
SHADOW> SET INP CYLEX1
SHADOW> SET UNI METERS
SHADOW> SET WIND
130,180
0.55555556
0.,360
5.
!
! An overhead view of the cylinder, which is centered on the origin, with radii of 1 and 1, with a length of 1 meter.
!
SHADOW> SET ANT
0,0,4
SHADOW> SHADOW

! A broadside look at the cylinder.
!
SHADOW> SET WIND
45,135
1.0
220.,310
1.25
SHADOW> SET ANT
0,4,0
SHADOW> SHADOW

! Now a look at the same geometry along the axis of the cylinder.
!
SHADOW> SET ANT
4,0,0
SHADOW> SET WIND
45,135
1.0
130.,220
1.25
SHADOW> SHADOW
SHADOW> EXIT
$ EXIT
```

The input set defining the plate was the following:

```plaintext
CM:  SIMPLE AIRCRAFT
CE:  RCS INPUT SET
UN: 
```

92
1
CC: FIRST CYLINDER
0.,0.,0.
90.,0.,0.,0.
2
1.,1., 1.
1.,1., -1.
XQ:
ZN:
The output generated by the code was the following:

| ANTeNA (ACS) = ( 0.0000, 0.0000, 4.0000 ) IN METERS | INPUT SKT: USER1RTLX.BAS,RTX|CSTXX,IMP;4 |
|-----------------------------------------------------|-----------------------------|
| Theta (Degrees)                                     | Prism                      |
| 0.00       5.66 135.11 146.67 147.22 157.78 163.35 168.89 174.44 180.00 |
| 0.00       | XXXXXXXXXXXXXXXXXXXXX     |
| 5.00       | XXXXXXXXXXXX              |
| 10.00      | XXXXXXXXXXXX              |
| 15.00      | XXXXXXXXXXXX              |
| 20.00      | XXXXXXXXXXXX              |
| 25.00      | XXXXXXXXXXXX              |
| 30.00      | XXXXXXXXXXXX              |
| 35.00      | XXXXXXXXXXXX              |
| 40.00      | XXXXXXXXXXXX              |
| 45.00      | XXXXXXXXXXXX              |
| 50.00      | XXXXXXXXXXXX              |
| 55.00      | XXXXXXXXXXXX              |
| 60.00      | XXXXXXXXXXXX              |
| 65.00      | XXXXXXXXXXXX              |
| 70.00      | XXXXXXXXXXXX              |
| 75.00      | XXXXXXXXXXXX              |
| 80.00      | XXXXXXXXXXXX              |
| 85.00      | XXXXXXXXXXXX              |
| 90.00      | XXXXXXXXXXXX              |
| 95.00      | XXXXXXXXXXXX              |
| 100.00     | XXXXXXXXXXXX              |
| 105.00     | XXXXXXXXXXXX              |
| 110.00     | XXXXXXXXXXXX              |
| 115.00     | XXXXXXXXXXXX              |
| 120.00     | XXXXXXXXXXXX              |
| 125.00     | XXXXXXXXXXXX              |
| 130.00     | XXXXXXXXXXXX              |
| 135.00     | XXXXXXXXXXXX              |
| 140.00     | XXXXXXXXXXXX              |
| 145.00     | XXXXXXXXXXXX              |
| 150.00     | XXXXXXXXXXXX              |
| 155.00     | XXXXXXXXXXXX              |
| 160.00     | XXXXXXXXXXXX              |
| 165.00     | XXXXXXXXXXXX              |
| 170.00     | XXXXXXXXXXXX              |
| 175.00     | XXXXXXXXXXXX              |
| 180.00     | XXXXXXXXXXXX              |
| 185.00     | XXXXXXXXXXXX              |
| 190.00     | XXXXXXXXXXXX              |
| 195.00     | XXXXXXXXXXXX              |
| 200.00     | XXXXXXXXXXXX              |
| 205.00     | XXXXXXXXXXXX              |
| 210.00     | XXXXXXXXXXXX              |
| 215.00     | XXXXXXXXXXXX              |
| 220.00     | XXXXXXXXXXXX              |
| 225.00     | XXXXXXXXXXXX              |
| 230.00     | XXXXXXXXXXXX              |
| 235.00     | XXXXXXXXXXXX              |
| 240.00     | XXXXXXXXXXXX              |
| 245.00     | XXXXXXXXXXXX              |
| 250.00     | XXXXXXXXXXXX              |
| 255.00     | XXXXXXXXXXXX              |
| 260.00     | XXXXXXXXXXXX              |
| 265.00     | XXXXXXXXXXXX              |
| 270.00     | XXXXXXXXXXXX              |
| 275.00     | XXXXXXXXXXXX              |
| 280.00     | XXXXXXXXXXXX              |
| 285.00     | XXXXXXXXXXXX              |
| 290.00     | XXXXXXXXXXXX              |
| 295.00     | XXXXXXXXXXXX              |
| 300.00     | XXXXXXXXXXXX              |
| 305.00     | XXXXXXXXXXXX              |
| 310.00     | XXXXXXXXXXXX              |
| 315.00     | XXXXXXXXXXXX              |
| 320.00     | XXXXXXXXXXXX              |
| 325.00     | XXXXXXXXXXXX              |
| 330.00     | XXXXXXXXXXXX              |
| 335.00     | XXXXXXXXXXXX              |
| 340.00     | XXXXXXXXXXXX              |
| 345.00     | XXXXXXXXXXXX              |
| 350.00     | XXXXXXXXXXXX              |
| 355.00     | XXXXXXXXXXXX              |
| 360.00     | XXXXXXXXXXXX              |
Example 6: Two Elliptic Cylinders

This example consists of two elliptic cylinders equidistant from the origin with axes coincident and directed along the Y axis. Three different source locations are presented with this single example.

The commands were:

```
$ RUN SHADOW
SHADOW> SET OUT CYLEX2/HOPLOT/PRINT
SHADOW> SET INP CYLEX2
SHADOW> SET UNI METERS
SHADOW> SET WIND 130,180
0.55555556
0.,360
5.

! An overhead view of the 2 cylinders with radii of 1 and 1,
! with a length of 1 meter each.

SHADOW> SET FILL /CYL=1
SHADOW> SET ANT 0,0,4
SHADOW> SHADOW

SHADOW> SET WIND 45,135
1.0
220.,310
1.25
SHADOW> SET ANT 0,4,0
SHADOW> SHOW FILL
SHADOW> SHADOW

SHADOW> SET ANT 4,0,0
SHADOW> SET WIND 45,135
1.0
130.,220
1.25
SHADOW> SHOW FILL
SHADOW> SHADOW
SHADOW> EXIT
$ EXIT

The input set defining the plate was the following:

CM: SIMPLE AIRCRAFT
CE: RCS INPUT SET
UN:
1
CC: FIRST CYLINDER
0.,-2.,0.
90.,0.,0.,0.
2
1.,1., 1.
1.,1., -1.
CC: SECOND CYLINDER
0.,+2.,0.
90.,0.,0.,0.
2
1.,1., 1.
1.,1., -1.
XQ:
EN:
The output generated by the code was the following:

| PHI  | 0.00 | 5.00 | 10.00 | 15.00 | 20.00 | 25.00 | 30.00 | 35.00 | 40.00 | 45.00 | 50.00 | 55.00 | 60.00 | 65.00 | 70.00 | 75.00 | 80.00 | 85.00 | 90.00 | 95.00 | 100.00 | 105.00 | 110.00 | 115.00 | 120.00 | 125.00 | 130.00 | 135.00 | 140.00 | 145.00 | 150.00 | 155.00 | 160.00 | 165.00 | 170.00 | 175.00 | 180.00 | 185.00 | 190.00 | 195.00 | 200.00 | 205.00 | 210.00 | 215.00 | 220.00 | 225.00 | 230.00 | 235.00 | 240.00 | 245.00 | 250.00 | 255.00 | 260.00 | 265.00 | 270.00 | 275.00 | 280.00 | 285.00 | 290.00 | 295.00 | 300.00 | 305.00 | 310.00 | 315.00 | 320.00 | 325.00 | 330.00 | 335.00 | 340.00 | 345.00 | 350.00 | 355.00 | 360.00 | 365.00 | 370.00 | 375.00 | 380.00 | 385.00 | 390.00 | 395.00 | 400.00 | 405.00 | 410.00 | 415.00 | 420.00 | 425.00 | 430.00 | 435.00 | 440.00 | 445.00 | 450.00 | 455.00 | 460.00 | 465.00 | 470.00 | 475.00 | 480.00 | 485.00 | 490.00 | 495.00 | 500.00 | 505.00 | 510.00 | 515.00 | 520.00 | 525.00 | 530.00 | 535.00 | 540.00 | 545.00 | 550.00 | 555.00 | 560.00 | 565.00 | 570.00 | 575.00 | 580.00 | 585.00 | 590.00 | 595.00 | 600.00 | 605.00 | 610.00 | 615.00 | 620.00 | 625.00 | 630.00 | 635.00 | 640.00 | 645.00 | 650.00 | 655.00 | 660.00 | 665.00 | 670.00 | 675.00 | 680.00 | 685.00 | 690.00 | 695.00 | 700.00 | 705.00 | 710.00 | 715.00 | 720.00 | 725.00 | 730.00 | 735.00 | 740.00 | 745.00 | 750.00 | 755.00 | 760.00 | 765.00 | 770.00 | 775.00 | 780.00 | 785.00 | 790.00 | 795.00 | 800.00 | 805.00 | 810.00 | 815.00 | 820.00 | 825.00 | 830.00 | 835.00 | 840.00 | 845.00 | 850.00 | 855.00 | 860.00 | 865.00 | 870.00 | 875.00 | 880.00 | 885.00 | 890.00 | 895.00 | 900.00 | 905.00 | 910.00 | 915.00 | 920.00 | 925.00 | 930.00 | 935.00 | 940.00 | 945.00 | 950.00 | 955.00 | 960.00 | 965.00 | 970.00 | 975.00 | 980.00 | 985.00 | 990.00 | 995.00 | 1000.00 |
7.7 Example 7: A Space Station Model

This example uses a space station, shown in Figure 7.1, that has been provided by NASA, Langley. The computer model is illustrated in Figure 7.2. It is an demonstrates how to use the windowing and highlighting commands (SET FILL) to effectively show obscuration.

The commands were:

```
$ RUN SHADOW
SHADOW> SET OUT AN6S1 /PRINT/NOECHO
SHADOW> SET INP AN6S1
SHADOW> SET UNI FEET
SHADOW> SET WIND
0,180
2.0
20.,290
2.5
!
! Display ONLY plate 6.
!
SHADOW> SET FILL "" /PLATE=6
SHADOW> SET ANT
    25, 15, 256.5
SHADOW> SHADOW
!
! Now make plate 6 stand out from the crowd.
!
SHADOW> SET FILL "!" /PLATE=(6,$)
SHADOW> SET ANT
    25, 15, 256.5
SHADOW> SHADOW
$ EXIT
```

The input set defining the plate was the following:

```
CM: ********CASE AN6S1********
CM: ********OBSCURATION********
CE:
LP: F
UN: UNITS IN FEET 2
CH: UPPER BOOM
CE:
PG: BOTTOM
  4 0
    4.5 49.5 387.
    4.5 -49.5 387.
    -4.5 -49.5 387.
    -4.5 49.5 387.
PG: +X SIDE
  4 0
    4.5 49.5 396.
```
<table>
<thead>
<tr>
<th>CM: UPPER KEEL</th>
<th>CE:</th>
<th>PG: -Y #1</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.6 -49.5 396.</td>
<td></td>
<td>4 0</td>
</tr>
<tr>
<td>4.5 -49.5 387.</td>
<td></td>
<td>4.5 -4.5 270.</td>
</tr>
<tr>
<td>4.5 -4.5 387.</td>
<td></td>
<td>-4.5 -4.5 387.</td>
</tr>
<tr>
<td>-4.5 -4.5 270.</td>
<td></td>
<td>PG: +Y #1</td>
</tr>
<tr>
<td>4 0</td>
<td></td>
<td>4.5 4.5 270.</td>
</tr>
<tr>
<td>-4.5 4.5 270.</td>
<td></td>
<td>-4.5 4.5 387.</td>
</tr>
<tr>
<td>-4.5 4.5 387.</td>
<td></td>
<td>4.5 4.5 387.</td>
</tr>
<tr>
<td>PC: +Y #1</td>
<td></td>
<td>PC: +X SIDE</td>
</tr>
<tr>
<td>4 0</td>
<td></td>
<td>4.5 4.5 387.</td>
</tr>
<tr>
<td>4.5 -4.5 387.</td>
<td></td>
<td>4.5 -4.5 387.</td>
</tr>
<tr>
<td>-4.5 -4.5 270.</td>
<td></td>
<td>PG: +X SIDE</td>
</tr>
<tr>
<td>4 0</td>
<td></td>
<td>4.5 4.5 387.</td>
</tr>
<tr>
<td>4.5 -4.5 387.</td>
<td></td>
<td>4.5 -4.5 270.</td>
</tr>
<tr>
<td>4.5 4.5 270.</td>
<td></td>
<td>PC: +X SIDE</td>
</tr>
<tr>
<td>CM: LOWER KEEL &amp; EXTENSION</td>
<td>CE:</td>
<td>PG: -Y #1</td>
</tr>
<tr>
<td>4.5 -4.5 261.</td>
<td></td>
<td>4 0</td>
</tr>
<tr>
<td>4.5 -4.5 261.</td>
<td></td>
<td>4.5 -22.5 98.</td>
</tr>
<tr>
<td>4.5 -22.5 98.</td>
<td></td>
<td>4.5 -22.5 0.</td>
</tr>
<tr>
<td>4.5 -22.5 0.</td>
<td></td>
<td>4.5 -13.5 0.</td>
</tr>
<tr>
<td>4.5 -13.5 0.</td>
<td></td>
<td>4.5 13.5 64.</td>
</tr>
<tr>
<td>4.5 13.5 64.</td>
<td></td>
<td>4.5 13.5 0.</td>
</tr>
<tr>
<td>PG: -Y #1</td>
<td></td>
<td>PG: -Y #2</td>
</tr>
<tr>
<td>4 0</td>
<td></td>
<td>4 0</td>
</tr>
<tr>
<td>4.5 -22.5 99.</td>
<td></td>
<td>4.5 -22.5 0.</td>
</tr>
<tr>
<td>4.5 -22.5 99.</td>
<td></td>
<td>4.5 -22.5 0.</td>
</tr>
<tr>
<td>-4.5 -22.5 99.</td>
<td></td>
<td>-4.5 -22.5 0.</td>
</tr>
<tr>
<td>-4.5 -22.5 99.</td>
<td></td>
<td>-4.5 -22.5 99.</td>
</tr>
<tr>
<td>-4.5 -22.5 99.</td>
<td></td>
<td>-4.5 -22.5 99.</td>
</tr>
</tbody>
</table>
PC: -Y #3
4 0
4.5 -4.5 99.
4.5 -4.5 261.
-4.5 -4.5 261.
-4.5 -4.5 99.
PC: +Y #1
4 0
4.5 22.5 0.
-4.5 22.5 0.
-4.5 22.5 99.
4.5 22.5 99.
PC: +Y #2
4 0
4.5 22.5 99.
-4.5 22.5 99.
-4.5 4.5 99.
4.5 4.5 99.
PC: +Y #3
4 0
4.5 4.5 99.
-4.5 4.5 99.
-4.5 4.5 261.
4.5 4.5 261.
CM: NON-ROTATING SECTION
CM: OF SOLAR PANEL BOOM
CE:
PG: BOTTOM
4 0
4.5 49.5 261.
4.5 -49.5 261.
-4.5 -49.5 261.
-4.5 49.5 261.
PG: +X SIDE
4 0
4.5 49.5 270.
4.5 -49.5 270.
4.5 -49.5 261.
4.5 49.5 261.
CM: ROTATING SECTION OF
CM: SOLAR PANEL BOOM
CE:
RT: -Y SIDE
0. -54. 265.5
0. 0. 90. 0.
PG: TOP
4 0
4.5 4.5 4.5
-4.5 4.5 4.5
-4.5 -76.5 4.5
4.5 -76.5 4.5
PC: BOTTOM
4 0
4.5 4.5 -4.5
4.5 -76.5 -4.5
-4.5 -76.5 -4.5
-4.5 4.5 -4.5
PC: +X SIDE
4 0
4.5 4.5 4.5
4.5 -76.5 4.5
4.5 -76.5 -4.5
4.5 4.5 -4.5
PC: -X SIDE
4 0
-4.5 4.5 4.5
-4.5 4.5 -4.5
-4.5 -76.5 -4.5
-4.5 -76.5 4.5
CM: UPPER OUTBOARD SOLAR PANEL
CE:
RT: -Y OUTBOARD
0. -132. 266.5
0. 0. 90. -52.
PG: -X 82X33
4 0
-1. 16.5 89.
-1. 16.5 7.
-1. -16.5 7.
-1. -16.5 89.
PG: UPPER 33
4 0
1. 16.5 89.
-1. 16.5 89.
-1. -16.5 89.
1. -16.5 89.
PG: LOWER 33
4 0
1. 16.5 7.
1. -16.5 7.
-1. -16.5 7.
-1. 16.5 7.
PG: INSIDE 82
4 0
1. 16.5 89.
1. 16.5 7.
-1. 16.5 7.
-1. 16.5 89.
CM: LOWER OUTBOARD SOLAR PANEL
CE:
PG: -X 82X33
4 0
-1. 16.5 -89.
-1. -16.5 -89.
-1. -16.5 -7.
-1. 16.5 -7.
PG: LOWER 33
4 0
1. 16.5 -89.
1. -16.5 -89.
1. -16.5 -89.
1. 16.5 -89.
PG: UPPER 33
4 0
1. 16.5 -89.
1. 16.5 -7.
1. -16.5 -7.
1. -16.5 -7.
PG: INSIDE 82
4 0
1. 16.5 -89.
1. 16.5 -89.
1. -16.5 -7.
1. 16.5 -7.
CM: UPPER INBOARD SOLAR PANEL
CE:
RT: -Y INBOARD
0. -78. 265.5
0. 0. 90. -52.
PG: -X 82X33
4 0
1. 16.5 89.
1. 16.5 7.
1. -16.5 7.
1. -16.5 89.
PG: UPPER 33
4 0
1. 16.5 89.
1. 16.5 89.
1. -16.5 89.
1. -16.5 89.
PG: LOWER 33
4 0
1. 16.5 7.
1. -16.5 7.
1. -16.5 7.
1. 16.5 7.
PG: 82 INSIDE
4 0
1. 16.5 89.
1. 16.5 7.
1. 16.5 7.
1. 16.5 89.
<table>
<thead>
<tr>
<th>PG: 82 OUTSIDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 0</td>
</tr>
<tr>
<td>1. -16.5 -89.</td>
</tr>
<tr>
<td>-1. -16.5 -89.</td>
</tr>
<tr>
<td>-1. -16.5 -7.</td>
</tr>
<tr>
<td>1. -16.5 -7.</td>
</tr>
</tbody>
</table>

CM: LOWER INBOARD SOLAR PANEL

CE:
<table>
<thead>
<tr>
<th>PG: -X 82X33</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 0</td>
</tr>
<tr>
<td>-1. -16.5 -89.</td>
</tr>
<tr>
<td>-1. -16.5 -89.</td>
</tr>
<tr>
<td>-1. -16.5 -7.</td>
</tr>
<tr>
<td>-1. 16.5 -7.</td>
</tr>
</tbody>
</table>

CM: LOWER 33

PC: 4 0
1. 16.5 -89. |
1. -16.5 -89. |
-1. -16.5 -89.|
-1. 16.5 -89. |

CM: UPPER 33

PC: 4 0
1. 16.5 -7. |
-1. 16.5 -7. |
-1. -16.5 -7. |
1. -16.5 -7. |

CM: 82 INSIDE

PC: 4 0
1. 16.5 -89. |
1. 16.5 -89. |
-1. 16.5 -7. |
1. 16.5 -7. |

CM: 82 OUTSIDE

PC: 4 0
1. -16.5 -89. |
1. -16.5 -7. |
-1. -16.5 -7. |
-1. -16.5 -89.|

CM: ROTATING SECTION OF SOLAR PANEL BOOM

CE:
<table>
<thead>
<tr>
<th>RT: +Y SIDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0. 54. 265.5</td>
</tr>
<tr>
<td>0. 0. 90. 0.</td>
</tr>
</tbody>
</table>

PG: TOP

<table>
<thead>
<tr>
<th>4 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.5 -4.5 4.5</td>
</tr>
<tr>
<td>4.5 76.5 4.5</td>
</tr>
<tr>
<td>-4.5 76.5 4.5</td>
</tr>
<tr>
<td>-4.5 -4.5 4.5</td>
</tr>
</tbody>
</table>

PG: BOTTOM
<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0</td>
<td>4.6</td>
</tr>
<tr>
<td>-4.5</td>
<td>-4.5</td>
<td>-4.5</td>
</tr>
<tr>
<td>-4.5</td>
<td>76.5</td>
<td>-4.6</td>
</tr>
<tr>
<td>4.5</td>
<td>76.5</td>
<td>4.5</td>
</tr>
<tr>
<td>PG: +X SIDE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4.5</td>
<td>-4.5</td>
<td>4.5</td>
</tr>
<tr>
<td>4.5</td>
<td>-4.5</td>
<td>-4.5</td>
</tr>
<tr>
<td>4.5</td>
<td>76.5</td>
<td>-4.5</td>
</tr>
<tr>
<td>4.5</td>
<td>76.5</td>
<td>4.5</td>
</tr>
<tr>
<td>PG: +X SIDE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>-4.5</td>
<td>-4.5</td>
<td>4.5</td>
</tr>
<tr>
<td>-4.5</td>
<td>76.5</td>
<td>4.5</td>
</tr>
<tr>
<td>-4.5</td>
<td>76.5</td>
<td>-4.5</td>
</tr>
<tr>
<td>-4.5</td>
<td>-4.5</td>
<td>-4.5</td>
</tr>
<tr>
<td>CM: UPPER OUTBOARD SOLAR PANEL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CE:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RT: +Y OUTBOARD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.</td>
<td>132.</td>
<td>265.5</td>
</tr>
<tr>
<td>0.</td>
<td>0.</td>
<td>90.</td>
</tr>
<tr>
<td>PG: -X 82X33</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>-1.</td>
<td>16.5</td>
<td>89.</td>
</tr>
<tr>
<td>-1.</td>
<td>16.5</td>
<td>7.</td>
</tr>
<tr>
<td>-1.</td>
<td>-16.5</td>
<td>7.</td>
</tr>
<tr>
<td>-1.</td>
<td>-16.5</td>
<td>89.</td>
</tr>
<tr>
<td>PG: UPPER 33</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>16.5</td>
<td>89.</td>
</tr>
<tr>
<td>-1.</td>
<td>16.5</td>
<td>89.</td>
</tr>
<tr>
<td>-1.</td>
<td>-16.5</td>
<td>89.</td>
</tr>
<tr>
<td>1.</td>
<td>-16.5</td>
<td>89.</td>
</tr>
<tr>
<td>PG: LOWER 33</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>16.5</td>
<td>7.</td>
</tr>
<tr>
<td>1.</td>
<td>-16.5</td>
<td>7.</td>
</tr>
<tr>
<td>-1.</td>
<td>-16.5</td>
<td>7.</td>
</tr>
<tr>
<td>-1.</td>
<td>16.5</td>
<td>7.</td>
</tr>
<tr>
<td>PG: INSIDE 82</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>-16.5</td>
<td>89.</td>
</tr>
<tr>
<td>-1.</td>
<td>-16.5</td>
<td>89.</td>
</tr>
<tr>
<td>-1.</td>
<td>-16.5</td>
<td>7.</td>
</tr>
<tr>
<td>1.</td>
<td>-16.5</td>
<td>7.</td>
</tr>
<tr>
<td>CM: LOWER OUTBOARD SOLAR PANEL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CE:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PG: -X 82X33</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>-1.</td>
<td>16.5</td>
<td>-89.</td>
</tr>
</tbody>
</table>
-1. -16.5 -89.
-1. -16.5 -7.
-1. 16.5 -7.
PG: LOWER 33
  4 0
  1. 16.5 -89.
  1. -16.5 -89.
  -1. -16.5 -89.
  -1. 16.5 -89.
PG: UPPER 33
  4 0
  1. 16.5 -7.
  -1. 16.5 -7.
  -1. -16.5 -7.
  1. -16.5 -7.
PG: INSIDE 82
  4 0
  1. -16.5 -89.
  1. -16.5 -7.
  -1. -16.5 -7.
  -1. -16.5 -89.
CM: UPPER INBOARD SOLAR PANEL
CE:
RT: +Y INBOARD
  0. 78. 255.5
  0. 0. 90. -52.
PG: -X 82X33
  4 0
  -1. 16.5 89.
  -1. 16.5 7.
  -1. -16.5 7.
  -1. -16.5 89.
PG: UPPER 33
  4 0
  1. 16.5 89.
  -1. 16.5 89.
  -1. -16.5 89.
  1. -16.5 89.
PG: LOWER 33
  4 0
  1. 16.5 7.
  1. -16.5 7.
  -1. -16.5 7.
  -1. 16.5 7.
PG: 82 OUTSIDE
  4 0
  1. 16.5 89.
  1. 16.5 7.
  -1. 16.5 7.
  -1. 16.5 89.
PG: 82 INSIDE
The output generated by the code was the following:

```
ANTENNA (RCS) = ( 7.8200, 4.6720, 78.1812 ) IN METERS

THETA (DEGREES)

   0.00  20.00  40.00  60.00  80.00 100.00 120.00 140.00 160.00 180.00
  +    +    +    +    +    +    +    +    +    +
  30.00  50.00  70.00  90.00

XQ: EXECUTE CODE
EH: END CODE
```
Figure 7.1: Illustration of the Space Station
Figure 7.2: Three-axis view of the Space Station as modeled by the input set.
7.8 Example 8: Another Look at the Space Station

This example presents a full view of the space station in the previous, except that the output is generated with the NCAR graphics interface. The non-interactive input is the same. The standard fill character procedure is used and a complete window is displayed with two degree resolution in theta and phi. The NCAR plot has been obtained using the plotting code in Chapter 13. The shadow map produced is shown in Figure 7.3.
Figure 7.3: NCAR plot showing the shadow map of the space station model.
References


Part II

Code Manual
Chapter 8

Introduction

The obscuration code SHADOW, is designed to produce a projected shadow map onto the far zone radiation sphere of an antenna in a complex environment. The map is efficiently calculated by directly tracing the outer boundaries of the multisided flat plates and composite cone frustum cylinders onto theta – phi space and then filling between the boundaries along raster lines. The code has been developed to be interactively run on a DEC VAX computer. It can, also, be run non-iteractively on any other computer by simply substituting the small main program and leaving out the interactive subroutines.

Part I of this manual is a user’s guide which treats the code from the users standpoint without much particular details about the coding. Part II, given here, is intended to give some details about the internal workings of the code. It gives more specific information about the coding itself. It is of importance primarily for people implementing the code on a new system, for debugging errors, or for making changes in how the code operates. An overview of how the code is organized is given in Chapter 9. A listing of the code is given in Chapter 10. It is broken up into three parts for the non-interactive, FORTRAN 77 subroutines and into the interactive VAX dependent subroutines. The implementation of the code on a VAX is given in Chapter 11 and a brief description of implementing the code on a non-VAX computer is given in Chapter 12. A listing of an NCAR plotting code for the shadow map is given in Chapter 13.
Chapter 9

Code Organization

The obscuration code SHADOW is designed to produce a projected shadow map onto the far zone radiation sphere of an antenna in a complex environment. The map is efficiently calculated by directly tracing the outer boundaries of the multisided flat plates and composite cone frustum cylinders onto theta-phi space and then filling between the boundaries along raster lines.

The code has been developed with efficiency and ease of use as primary considerations. Often with other similar codes the engineer is not part of a tight interactive design loop. In order to facilitate this capability, while maintaining necessary transportability, the code has been split into two versions so that it can be run in two different modes, interactively or non-interactively depending on the computer being used. In both versions the flow of program control is basically the same. The main program either accepts interactive commands from the terminal and acts on those commands, or reads a different set of non-interactive commands from the input file and processes those. In both cases, the main program loops on input commands and calls appropriate subroutines for the creation and output of the shadow map.

The map creation is broken down into separate phases for each class of geometry being processed. Plates and elliptic cylinders are the two phases currently implemented. Each processing phase works by projecting each member of each class of geometry onto the far-zone sphere. The code implements the shadow map by mapping the far zone sphere in theta-phi space into a rectangular character array. The size of the array and hence the angular resolution of the resulting map is determined by the user at run time. After a member is projected, the far-zone grid (array) is processed in a raster-scan fashion to implement an area fill for the member. In this way every geometric entity is processed and included in the array. After all items of all classes have been processed, the output routines format and display/dump the resulting map. The main program then readies itself to execute yet another command or commands.

The source code is also organized into two groups of files depending upon the desired mode of operation. The code is organized this way so that minimum source modification is necessary in order to run in either interactive (in the case of a VAX computer) or non-interactive modes. The chapter on Non-VAX implementation describes the conversion of the source to non-interactive mode in detail.

Since the map computation and display routines are identical for both modes of operation, the transportability of generated results depends on the numerical behavior of the
target machine an not on implementational differences between the interactive and non-
interactive versions.
Chapter 10

Listings of the Code

This chapter describes the operation of the routines and functions used by the program. Each listing is presented in alphabetical order and is preceded where appropriate by a short explanation of methods used.

10.1 VAX/VMS Subroutines

The following routines are for the interactive implementations of the code. They are used in conjunction with the routines in this chapter that are common to both versions.

MAIN PROGRAM

This is the main routine for the interactive versions of the program. It calls a one-time initialization routine and then executes commands until finished. There is another slightly different main program for the non-interactive code.

```
0001 PROGRAM SHADOW
0002 C////++
0003 C///// This program was written at the ohio state university
0004 C///// electroscience laboratory. any problems or comments
0005 C///// can be referred to:
0006 C///// 1320 KINNICH RD.
0007 C///// COLUMBUS,OHIO 43212
0008 C///// PHONE: (614) 422-6762 OR 422-6848
0009 C///// Beginning of the main routine.
```
Initialize any SHADOW data structures.

CALL INIT

Call the interactive terminal interface. This routine calls all other subroutines.

CALL INTRAC

Finished.

END
SUBROUTINE INIT

0001  C---------------------------
0002  SUBROUTINE INIT
0003  INCLUDE 'SHACON.FOR'
0170  C!!!
0171  C!!! This subroutine initializes the main routine.
0172  C!!! It is meant to be called once, at the start of the program.
0173  C!!!
0174  C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
0175  C!!!
0176  C!!! NOTICE:
0177  C!!! This routine performs actions which do not apply to the
0178  C!!! non-interactive mode of operation. In particular, the variables
0179  C!!! which are initialized here may be reinitialized elsewhere in both
0180  C!!! interactive and non-interactive versions. Altering these
0181  C!!! parameters may or may not achieve the expected results.
0182  C!!!
0183  C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
0184  C!!!
0185  C!!! Initialize variables to their default values.
0186  C!!!
0187  C!!! The lower/higher theta end of the range.
0188  C!!!
0189  THET1 = 0.0  * RPD
0190  THET2 = 180.0 * RPD
0191  C!!!
0192  C!!! The lower/higher phi end of the range.
0193  C!!!
0194  PHI1 = 0.0  * RPD
0195  PHI2 = 360.0 * RPD
0196  C!!!
0197  C!!! The desired theta/phi resolution in units of radians/pixel.
0198  C!!!
0199  RESTH = 2.  * RPD
0200  RESPH = 2.  * RPD
0201  C$$
0202  C$$ Rotate translate default data RT:
0203  C$$
0204  THZP = 0.
0205  PHZP = 0.
0206  THXP = 90.
0207  PHXP = 0.
0208  C$$
0209  TRS( 1 ) = 0.
0210  TRS( 2 ) = 0.
0211  TRS( 3 ) = 0.
0212  C$$
0213  VRS( 1, 1 ) = 1.
0214  VRS( 1, 2 ) = 0.
0215  VRS( 1, 3 ) = 0.
0216  C$$
0217  VRS( 2, 1 ) = 0.
0218  VRS( 2, 2 ) = 1.
0219  VRS( 2, 3 ) = 0.
0220  C$$
0221  VRS( 3, 1 ) = 0.
0222  VRS( 3, 2 ) = 0.
0223  VRS( 3, 3 ) = 1.
0224  C$$
0225  C$$ Units default data UN:, UF:
IUNIT = 1
UNITF = 1.0
UNITN = UNIT( IUNIT )
UNITS = UNITN * UNITF

Pattern cut orientation data VF:

VF( 1, 1 ) = 1.
VF( 1, 2 ) = 0.
VF( 1, 3 ) = 0.
VF( 2, 1 ) = 0.
VF( 2, 2 ) = 1.
VF( 2, 3 ) = 0.
VF( 3, 1 ) = 0.
VF( 3, 2 ) = 0.
VF( 3, 3 ) = 1.

Open some standard input/output files for the VMS support routines.

Units 5 and 6 are reserved for input set reading and echoing by the input set processor. NOTE: This is operating system dependent stuff. This is the natural place for it since it is initialized at the start.

OPEN( UNIT=1, FILE='SYS$INPUT', TYPE='OLD' )
OPEN( UNIT=2, FILE='SYS$OUTPUT', TYPE='UNKNOWN' )

End of program initialization.

RETURN
END
SUBROUTINE INTRAC

This is the interactive commands subroutine called by the main routine. It fields commands typed by the user and executes the appropriate service routines. Also listed here are two I/O function subprograms which are indirectly invoked by INTRAC. They are called GET_INPUT and PUT_OUTPUT.

0001 C-----------------------------------------------------------------------
0002 SUBROUTINE INTRAC
0003 | ++
0005 | FACILITY: INTERACTIVE TERMINAL COMMAND INTERFACE
0006 |
0007 | ABSTRACT: This procedure prompts a terminal for input and parses/distributes
0008 | through CLI routines.
0009 |
0010 | ENVIRONMENT: VAX/VMS Version 4.x
0011 |
0012 | AUTHOR: Laszlo Takacs CREATION DATE: 20-AUG-1986
0013 |
0014 | MODIFIED BY:
0015 |
0016 | 1-001 - Original, LAT 20-AUG-1985
0017 |
0019 | IMPLICIT NONE
0020 | INCLUDE '"$RMSDEF"'
0021 | INCLUDE '"$SMCDEF"'
0022 | INCLUDE 'SHACOM.FOR'
0023 | EXTERNAL
0024 | * COMMAND_TABLES, | ! User-defined com
0025 | + GET_INPUT | ! The I/O routine at the b
0026 |
0027 | INTEGER 4
0028 | * STS,
0029 | * READ_STS,
0030 | * CLI$PRESENT,
0031 | * CLI$DISPATCH,
0032 | * CLI$DCL_PARSE,
0033 | * CLI$GET_VALUE,
0034 | * SMC$LOAD_KEY_DEFS,
0035 | * SMC$CREATE_KEY_TABLE,
0036 | * SMC$DELETE_VIRTUAL_KEYBOARD,
0037 | * SMC$CREATE_VIRTUAL_KEYBOARD
0038 | ! Make a defininition table.
0039 |
0040 | STS = SMC$CREATE_KEY_TABLE( KETTBL )
0041 | IF ( NOT. STS) CALL LBSIGNAL( VAL(STS) )
0042 | ! Load the definitions from the key definition file. Ignore "file not f
0043 | STS = SMC$LOAD_KEY_DEFS( KETTBL, 'SHADOW.FPD' )
0044 | IF ( ( NOT. STS) .AND. (STS .NE. RMS$_FN))
0045 | + CALL LBSIGNAL( VAL(STS) )
0046 | ! Get a handle on SYS$INPUT.
0047 |
0048 | READ_STS = SMC$CREATE_VIRTUAL_KEYBOARD( KBDID )
The main processing loop. Keep reading input until the user types EOF.

DO WHILE ( READ_STS .NE. RMS$_EOF )
  read from input and parse the command.
  READ_STS = CLI$DCL_PARSE(, COMMAND_TABLES,
  GET_INPUT,
  GET_INPUT,
  'SHADOW'
  )
  If the command parse was successful, execute the command-routine.
  IF (.NOT. (.NOT. READ_STS)) CALL CLI$DISPATCH()
END DO

Get rid of the virtual keyboard.

If (.NOT. (.NOT. READ_STS)) CALL LIB$SIGNAL( 'VAL_STS')

Return

RETURN

C-------------
INTEGER*4 FUNCTION GET_INPUT( COMMAND, PROMPT, LENGTH )
C!!! This routine does all the reading for the terminal interface.
It has the same calling format as LIB$GET_INPUT except that options
parameters may not be omitted.
C!!!
C!!!
INCLUDE '"RMSDEF')
INCLUDE 'SHACOM.FOR'
EXTERNAL
CHARACTER*(*)

COMMAND,
PROMPT

INTEGER

LENGTH=2

SMC$READ_COMPOSED_LINE*4

| Read a (composed) line and return the status to CLI$ staff.

GET_INPUT = SMC$READ_COMPOSED_LINE(
  KBDID,
  RETBL,
  COMMAND,
  PROMPT,
  LENGTH
)

IF ( GET_INPUT .EQ. XLOC( SMC$_EOF ) ) GET_INPUT = RMS$_EOF

RETURN

C-------------
INTEGER*4 FUNCTION PUT_OUTPUT ( STRING )
C!!! This routine does all the writing for the terminal interface.
C!!! It has the same calling format as LIB$PUT_OUTPUT.
C!!!

131
INCLUDE 'SHAC04.FOR'

CHARACTER*(*)
STRING
INTEGER*4
LIB$PUT_OUTPUT

! Read a line.
! PUT_OUTPUT = LIB$PUT_OUTPUT ( STRING )
! There should be no errors here. Signal if there are any.
! IF (.NOT. PUT_OUTPUT) CALL LIB$SIGNAL( xval(PUT_OUTPUT) )
! Return.
RETURN
END
Interactive Service Routines

The following routines are used ONLY in the interactive version of the code and are operating system dependent. They provide functions and service routines for the interactive commands.

```
0001 C'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
0002 C
0003 C The system-dependent stuff goes below here.
0004 C
0005 C'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
0006 C
0007 C
0008 C
0009 C FUNCTIONAL DESCRIPTION:
0010 C
0011 C These functions are the action routines invoked by the VERB which
0012 C follows from the on each routine.
0013 C
0014 C CALLING SEQUENCE:
0015 C
0016 C
0017 C
0018 C FORMAL PARAMETERS:
0019 C
0020 C
0021 C
0022 C IMPLICIT INPUTS:
0023 C
0024 C
0025 C IMPLICIT OUTPUTS:
0026 C
0027 C
0028 C FUNCTION SPECIFIC
0029 C
0030 C COMPLETION STATUS:
0031 C
0032 C FUNCTION SPECIFIC
0033 C
0034 C: $SS_NORMAL Success, or
0035 C: fact_status some other status
0036 C
0037 C: SIDE EFFECTS:
0038 C
0039 C
0040 C:--
0041 C: INTEGER FUNCTION SERVICE_ROUTINES
0042 C: IMPLICIT NONE
0043 C: PARAMETER SUCCESS = 1
0044 C: INCLUDE '($SDEF)/NOLIST' Include system status definitions
0435 C: INCLUDE 'SHADOW.FOR/LIST' Include SHADOW common block
0436 C:
0437 C COMMON declarations...
0438 C
0439 C
0440 C: COMMON /PIS/
0441 C: + PI,
0442 C: + TPI,
0443 C: + DPR,
0444 C: + RPD
```
1046 1 C+++ MAXIMUM DIMENSION FOR PLATES
1047 1 INTEGER NPI
1048 1 C+++ MAXIMUM DIMENSION FOR PLATE EDGES
1049 1 INTEGER NEI
1050 1 PARAMETER (NEI=12)
1051 1 C+++ MAXIMUM DIMENSION FOR CYLINDERS
1052 1 INTEGER NCX
1053 1 PARAMETER (NCX=6)
1054 1 C+++ MAXIMUM DIMENSION FOR CYLINDER RIMS
1055 1 INTEGER NRI
1056 1 PARAMETER (NRI=10)
1057 1 C+++ MAXIMUM DIMENSION FOR ROWS (PHI)
1058 1 INTEGER MAWROW
1059 1 PARAMETER (MAWROW=361)
1060 1 C+++ MAXIMUM DIMENSION FOR COLUMNS (THETA)
1061 1 INTEGER MAICOL
1062 1 PARAMETER (MAICOL=181)
1063 1 C!!!
1064 1 COMMON /GESSPLA/
1065 1 + XI (3,NEI,NPI).
1066 1 + V (3,NEI,NPI).
1067 1 + VP (3,NEI,NPI).
1068 1 + VN (3,NPI).
1069 1 + NEP (NPI).
1070 1 + NPI
1071 1 C!!!
1072 1 COMMON /GEDNEL/
1073 1 + AC (NHI,NCX).
1074 1 + BC (NHI,NCX).
1075 1 + ZC (NHI,NCX).
1076 1 + TCR (NHI,NCX).
1077 1 + ICL (3,NCX).
1078 1 + VCL (3,3,NCX).
1079 1 + NEC (NCX).
1080 1 + NCX
1081 1 C!!!
1082 1 COMMON /EDNAG/ VNAG(NEX,NPI)
1083 1 C!!!
1084 1 COMMON /SHADOW/ COLS, ROWS, ANTIEN(3),CTROID(3),
1085 1 + MP, ME, NEITME, WC,
1086 1 + THET1, THET2, PHI, PH2, RESTH, RESPH, ALPH,
1087 1 + UNIT(3), TBS(3), VBS(3,3), UNI, UNITF, UNITS, UN
1088 1 + THZP, PHZP, THSP, PHSP, FILPNM, FILCHM
1089 1 COMMON /SHADOW/ IMPFIL, OUTBUF(MAICOL,MAWROW),
1090 1 + FILCHC,FILCHP,FILCHR
1091 1 C!!!
1092 1 COMMON /PATCUT/ VPC(3,3)
1093 1 C!!!
1094 1 C!!! The first set of declarations is the stuff in /SHADOW/ common bloc
1095 1 C!!!
1096 1 INTEGER
1097 1 + MP, ME, NEITME, WC.
1098 1 C! Plate# /edge# /cyl# variables.
1099 1 + FILPNM, FILCHM,
1100 1 C! Plate and cyl numbers for special filling
1101 1 + COLS,
1102 1 C! The size of the array subsection determined
1103 1 + ROWS
1104 1 C! by internal resolution requirements.
1105 1 REAL
1106 1 + CTROID,
1107 1 C! A geometric center of the object in question.
ANEM,  
A The antenna location in Ref Coord. System.  
THET1,  
A The lower theta end of the range.  
THET2,  
A The higher theta end of the range.  
PH1,  
A The lower phi end of the range.  
PH2,  
A The higher phi end of the range.  
RESTM,  
A The desired theta/phi resolution.  
RESPH,  
A in units of radians/pixel.  
METO,  
A Higher theta and of the range.  
PHI,  
A Higher phi and of the range.  
PH2,  
A Higher phi and of the range.  
PH2,  
A Higher phi and of the range.  
BESTH,  
A Maximum allowed angular excursion.  
CHARACTER  
OBUF*1,  
A The output buffer which is displayed.  
INPFL*63,  
A The filename of the input set.  
FILGHC,  
A special fill character for cylinders  
FILCHP,  
A special fill character for everything else  
FILCHR,  
A special fill character for plates.  
DATA FILGHC, FILCHP, FILCHR / 'C', 'P', 'X' /  
FROM the/PIS/ COMMON block...  
REAL PI, TPI, DPR, RPD  
记者从/CDOPLA/ COMMON block...  
INTECER  
M,  
A Number of edges per plate  
NVI,  
A Total number of plates  
REAL,  
A The array of plate corners  
V,  
A Edge unit vectors  
VP,  
A Edge unit binormals  
VN  
A Unit normal for each plate  
NVI,  
A From the /EDOPLA/ COMMON block...  
C,  
A Elliptic parameter along x-axis  
AC,  
A Elliptic parameter along y-axis  
EC,  
A
1173 1 C! Cylinder endcaps in cyl coord sys
1174 1 + TCR.
1176 1 C! Angle endcap makes with positive z axis
1176 1 + ICL.
1177 1 C! Cyl coord sys origin
1178 1 + VCL
1179 1 C! Definition of cyl coord sys
1180 1 C!
1181 1 INTEGER
1182 1 + IUNIT
1183 1 REAL
1184 1 + UNIT,
1185 1 + UNITS,
1186 1 + UNI,
1187 1 + UNIT,
1188 1 + TRS,
1189 1 + THP, PHIP,
1190 1 + VRS,
1191 1 + VPC,
1192 1 + VMAG
1193 1 DATA UNIT/1..3048,0.025/
1194 1 C!
1195 1 C!!!
1196 1 C!!! The following common block is for VMS$SG# software only.
1197 1 C!!!
1198 1 INTEGER
1199 1 COMMON /TERCON/                   KBID, KEYTBL
1200 1 C!!!-
1201 EXTERNAL
1202 + PUT_OUTPUT, GET_INPUT, | My own $SG-type I/O routines
1203 + CL1#_PRESEN, |
1204 + CL1#_NEGAT, |
1205 + CL1#_LOCRES, | locally present
1206 + CL1#_LOCRED, | locally negated
1207 + CL1#_DEFATL,
1208 + CL1#_ABSE
1209 + CL1#_IVALU
1210 CHARACTER
1211 + P1=80, | Command line variable
1212 + P2=80, |
1213 + UNCHAR=1, | A character
1214 + LIBRARY=64, | Name of the help library is defa
1215 + LABEL(3)=6, | Units label
1216 + "/"meters", 'feet', 'inches'/,
1217 + FILE *50, | Temporary file variable
1218 + PRFIL=50, | Printable file
1219 + LISFIL=50, | Input echo listing
1220 + OFTIL=50, | "Plotable" output file
1221 !
1222 DATA IUNIT/1/
1223
1224 LOGICAL+c
1225 + VALID_INPUT, | A loop control variable
1226 + CL1#_PRESEN, | CL1 interface to get info about
1227 + CL1#_GET_VALUE, | CL1 interface to get info about
1228
1229 REAL+c
1230 + DOT.DZI,XQ(3)
1231 INTEGER+c
1232 + N.NI,NJ,STS, | sordid variables...
1233 + KEYPAD, | Keypad condition flag
1234
1235 !
1236 ! General library routines
The librarian help routine sets the current fill characters being used for plates or cylinders.

```assembly
ENTRY SET_FIL
  IF ( CLI$PRESENT( 'SEQUENTIAL' ) ) THEN
    C1 Reset things to their default state.
    C1
    FILPNM = -1
    FILCMN = -1
    FILCHP = 'P'
    FILHC = 'C'
    FILCHR = 'X'
  C1 To avoid screwing up the text in SCAN, use a character that will not be used by the fill process, like char 7.
  C1 Set a plate up for tagging.
  C1
  ELSEIF ( CLI$PRESENT( 'PLATE' ) ) THEN
    C1 Clear any cylinder tagging residue.
    C1
    FILCMN = 0
    FILHC = 'C'
    C1 Get the master fill character.
  C1
  CALL CLI$GET_VALUE( 'P2', FILCHR )
  C1 Get the qualifier numeral value. STS is being used for the length of the decode.
  C1
  IF ( CLI$GET_VALUE( 'PLATE', P2, STS ) ) THEN
    C1
    DECODE (STS,1,P2,IOSTAT=STS) FILPNM
    ELSE
      STS = -1
    ENDIF
  C1
  C1 Get the fill character for that plate. Use a 'P' if none is given.
  C1
  IF ( STS .NE. 0 ) THEN
    C1
    SET_FIL = %LOC( CLI$IVALU )
  END
  C1
```

I+LIB$SPAVN, I Executee a rubprocerr +LBR$OUfPUT-HELP, 1 The librarian help routine +GHG$SET,KETPAD-MODE, I Screen management package

I I-SET/SHARE routinmr I
I SET_ANT, SET_PAT, SET_KEY, SET_UNI_INCHES, SET_UNI_FEET,
I SHOW_ANT, SHOW_PAT, SHOW_SCA, SHOW_WIN, SHOW_KEY, SHOW_INP,
I SHOW_UNI,
I
I varioun command routinrn I
I EXIT_COMMAND, HELP_COMMAND, DCL_COMMAND, SHADOW_COMM
I
I C1
I C1 This routine sets the current fill characters being used for plates or cylinders.
I C1
ENTRY SET_FIL
  IF ( CLI$PRESENT( 'SEQUENTIAL' ) ) THEN
    C1 Reset things to their default state.
    C1
    FILPNM = -1
    FILCMN = -1
    FILCHP = 'P'
    FILHC = 'C'
    FILCHR = 'X'
  C1 To avoid screwing up the text in SCAN, use a character that will not be used by the fill process, like char 7.
  C1 Set a plate up for tagging.
  C1
  ELSEIF ( CLI$PRESENT( 'PLATE' ) ) THEN
    C1 Clear any cylinder tagging residue.
    C1
    FILCMN = 0
    FILHC = 'C'
    C1 Get the master fill character.
  C1
  CALL CLI$GET_VALUE( 'P2', FILCHR )
  C1 Get the qualifier numeral value. STS is being used for the length of the decode.
  C1
  IF ( CLI$GET_VALUE( 'PLATE', P2, STS ) ) THEN
    C1
    DECODE (STS,1,P2,IOSTAT=STS) FILPNM
    ELSE
      STS = -1
    ENDIF
  C1
  C1 Get the fill character for that plate. Use a 'P' if none is given.
  C1
  IF ( STS .NE. 0 ) THEN
    C1
    SET_FIL = %LOC( CLI$IVALU )
  END
ELSE
  IF (.NOT. CLI$GET_VALUE( 'PLATE', FILCHP ) ) THEN
    FILCHP = 'P'
  ENDIF
ENDIF

C! Set a cylinder up for tagging.

ELSEIF (.NOT. CLI$PRESENT( 'CYLINDER' ) ) THEN

C! Clear any cylinder tagging residue.

  FILPNM = 0
  FILCHP = 'P'

C! Get the master fill character.

  CALL CLI$GET_VALUE( 'P2', FILCHR )

C! Get the qualifier numeral value. STS is being used for the length of
C! and the status of the decode.

  IF ( CLI$GET_VALUE( 'CYLINDER', P2, STS ) ) THEN
    DECODE (STS,1,P2,IOSTAT=STS) FILCHM
  ELSE
    STS = -1
  ENDIF

C! Get the fill character for that cylinder. Use a 'C' if none is given

C! The else here is for a "SET FILL [x]" command.

  IF ( STS .NE. 0 ) THEN
    SET_FIL = ILSC( CLI$IVALU )
  ELSE
    IF (.NOT. CLI$GET_VALUE( 'CYLINDER', FILCHC ) ) THEN
      FILCHC = 'C'
    ENDIF
  ENDIF

C! The else here is for a "SET FILL [x]" command.

C! Get the master fill character.

  CALL CLI$GET_VALUE( 'P2', FILCHR )

C! End of cases.

C!

GINO 3

GOTO 3

FORMAT( I )

C! This routine displays the current fill characters being used for plat
C! or cylinders.

C!

ENTRY SHOW_FIL

C!

C! Assume success only when the SHOW command is being executed.

C!

SHOW_FIL = SUCCESS

C!
CI Examine the plate situation.
3 IF (FILPNM .GT. 0) THEN
   WRITE(2,FMT='("Plate ",I3,' tag with [',A,']")')
   + FILPNM, FILCHP
   WRITE(2,FMT='(" All other geometry tagged with [',A,']")')
   + FILCHR
ENDIF
CI Examine the cylinder situation.
CI IF (FILCNM .GT. 0) THEN
   WRITE(2,FMT='(" Cylinder ",I3,' tag with [',A,']")')
   + FILCNM, FILCHC
   WRITE(2,FMT='(" All other geometry tagged with [',A,']")')
   + FILCHR
ENDIF
CI Check on a no-tag background character situation.
CI IF ( (FILCNM .EQ. 0) .AND. (FILPNM .EQ. 0) ) THEN
   WRITE(2,FMT='(" No individual plates/cylinders are tagged")')
ENDIF
CI Report the sequential numbering case.
CI IF ( (FILCNM .LT. 0) .AND. (FILPNM .LT. 0) ) THEN
   WRITE(2,FMT='(" All cylinders/plates sequentially tagged")')
RETURN
CI This routine sets the antenna location.
ENTRY SET_ANT
   WRITE (2,FMT=('(IX,'"Input antenna location in ",A6,"": ",4')")
   + LABEL(IUNIT)
   READ (1,*) ANTE(1), ANTE(2), ANTE(3)
CI Perform appropriate units conversion here.
DO 3424 N=1,3
   XQ(N)=ANTE(N)
SET_ANT = SUCCESS
END DO
CI This routine displays the current antenna position.
ENTRY SHOW_ANT
DO N11.3
   XQ(N)= (ANTE(1)-TRS(1)) * VRS(N,1) + (ANTE(2)-TRS(2)) * VRS(N,2) + (ANTE(3)-TRS(3)) * VRS(N,3) ) / UNITS
SET_ANT = SUCCESS
END DO
WRITE(2,FMT="('Antenna in RCS (meters): ','2F12.6')") ANTE
WRITE(2,FMT="('Definite system (','A,'): ','2F12.6')")
  + LABEL(IUNIT), A
SHOW_ANT = SUCCESS
RETURN
C! Process a new input set. Inquire about the full name.
C!
C! ENTRY SET-INP
CALL CLI$GET_VALUE( 'P2', FILE )
OPEN ( UNIT=5, FILE=FILE, DEFAULTFILE='.INP', STATUS='OLD')
CALL ASCRIPT
SET_INP = SUCCESS
C!
C! This routine displays the current input set name.
C!
C! ENTRY SHOW-INP
INQUIRE ( UNIT=5, NAME=INPFIL )
TYPE *, 'The current input set is ', INPFIL
SHOW_INP = SUCCESS
RETURN
C!
C! This routine toggles/report keypad mode.
C!
C! ENTRY SET_KEY
IF ( .NOT. CLI$PRESENT( 'KEYPAD_MODE') ) THEN
  KEYPAD = 0
ELSE
  KEYPAD = 1
END IF
SET_KEY = SMG$SET_KEYPAD_MODE( KBDID, KEYPAD )
C!
C! This routine displays the current keypad mode.
C!
C! ENTRY SHOW_KEY
IF ( KEYPAD.EQ. 0 ) THEN
  WRITE(2,*), 'The keyboard is not in keypad mode.'
ELSE
  WRITE(2,*), 'The keyboard is in keypad mode.'
END IF
RETURN
C!
C! Set up a coordinate system.
C!
C! ENTRY SET_COO
C## TRS(N)=LINEAR TRANSLATION OF COORDINATES FROM THE FIXED
C## COORDINATES WHICH IS ORIGINALLY SET UP BY OPERATOR.
C##
 TYPE 3021, LABEL(IUNIT)
3021 FORMAT( 'Please input a translation vector in ','A6,' : ')
accept*, (TRS(N),N=1,3)
DO 3020 N=1,3
3020 TRS(N)=TRS(N)*UNITS
C##
C## THZP,PHZP=ORIENTATION OF THE VRS(3,N) AXIS RELATIVE TO THE
C## FIXED COORDINATE SYSTEM.
C### THXP,PHXP=ORIENTATION OF THE VRS(1,N) AXIS RELATIVE TO THE
C### FIXED COORDINATE SYSTEM.

1469 continue

1470 123 continue

1471    type*, 'Please input THZP,PHZP,THXP,PHXP in degrees:'
1472    accept*, THZP,PHZP,THXP,PHXP
1473
1474    VRS(3,1)=SIN(THZP*RPD)*COS(PHZP*RPD)
1475    VRS(3,2)=SIN(THZP*RPD)*SIN(PHZP*RPD)
1476    VRS(3,3)=COS(THZP*RPD)
1477
1478    VPC(3,1)=SIN(THZP*RPD)*COS(PHZP*RPD)
1479    VPC(3,2)=SIN(THZP*RPD)*SIN(PHZP*RPD)
1480    VPC(3,3)=COS(THZP*RPD)

1481    INSURE VRS(1,N) IS PERPENDICULAR TO VRS(3,N)

1482    DZI=VRS(3,1)*VRS(1,1)+VRS(3,2)*VRS(1,2)+VRS(3,3)*VRS(1,3)
1483    IF(ABS(DZI).GT.0.1) THEN
1484        TYPE*, 'The coordinates are NOT orthogonal - Respecify.'
1485        go to 123
1486    ELSE

1487    VRS(1,1)=VRS(1,1)-VRS(3,1)*DZI
1488    VRS(1,2)=VRS(1,2)-VRS(3,2)*DZI
1489    VRS(1,3)=VRS(1,3)-VRS(3,3)*DZI
1490
1491    DOT=VRS(1,1)*VRS(1,1)+VRS(1,2)*VRS(1,2)+VRS(1,3)*VRS(1,3)
1492    DOT=SQRT(DOT)
1493
1494    VRS(1,1)=VRS(1,1)/DOT
1495    VRS(1,2)=VRS(1,2)/DOT
1496    VRS(1,3)=VRS(1,3)/DOT
1497
1498    VRS(3,1)=VRS(3,1)*VRS(3,1)-VRS(3,2)*VRS(3,2)
1499    VRS(3,2)=VRS(3,2)*VRS(3,2)-VRS(3,3)*VRS(3,3)
1500    VRS(3,3)=VRS(3,3)*VRS(3,3)
1501
1502    END IF

1503 C11 Display the coordinates

1504 C1 ENTRY SHOW_COO

1505 3931 FORMAT(2H *,6X,'The following rotations are used for ALL',
1506     2' subsequent inputs:',THZP,THXP)
1507     DO 3932 N1=1,3
1508     WRITE(6,3933) (NI,NJ,VRS(NI,NJ),NJ=1,3)
1509     3932 WRITE(6,3933) (NI,NJ,VRS(NI,NJ),NJ=1,3)
1509     3933 FORMAT(2H *,1X,3(2X,'VRS(',1L,' ',',',1L,')=',F9.6),1X,3(2X,'VRS(',1L, ',',1L,')=',F9.6),THZP,THXP)
1510     C1 ENTRY SET_PAT

1511 C#### THZP,PHZP=ORIENTATION OF THE VPC(3,N) AXIS RELATIVE TO THE
1512 C#### FIXED COORDINATE SYSTEM.

1513 C#### THXP,PHXP=ORIENTATION OF THE VPC(1,N) AXIS RELATIVE TO THE
1514 C#### FIXED COORDINATE SYSTEM.

1515 C#### continue

1516 123 continue

1517    type*, 'Please input THZP,PHZP,THXP,PHXP in degrees:'
1518    accept*, THZP,PHZP,THXP,PHXP
1519
1520    VPC(3,1)=SIN(THZP*RPD)*COS(PHZP*RPD)
1521    VPC(3,2)=SIN(THZP*RPD)*SIN(PHZP*RPD)
1522    VPC(3,3)=COS(THZP*RPD)
1523
1524    VPC(1,1)=SIN(THXP*RPD)*COS(PHXP*RPD)
1525    VPC(1,2)=SIN(THXP*RPD)*SIN(PHXP*RPD)
1526    VPC(1,3)=COS(THXP*RPD)
VPC(1,3) = COS(THXP*RPD)

Cl

Cl

INSURE VPC(1,N) IS PERPENDICULAR TO VPC(3,N)

DZX=VPC(3,1)*VPC(1,1)+VPC(3,2)*VPC(1,2)+VPC(3,3)*VPC(1,3)

IF(ABS(DZX).GT.0.1) THEN

TYPE*, 'The coordinates are NOT orthogonal - Respecify.'

goto 1234

ELSE

VPC(1,1)=VPC(1,1)-VPC(3,1)*DZX

VPC(1,2)=VPC(1,2)-VPC(3,2)*DZX

VPC(1,3)=VPC(1,3)-VPC(3,3)*DZX

DOT=VPC(1,1)*VPC(1,1)+VPC(1,2)*VPC(1,2)+VPC(1,3)*VPC(1,3)

DOT=SQRT(DOT)

VPC(1,1)=VPC(1,1)/DOT

VPC(1,2)=VPC(1,2)/DOT

VPC(1,3)=VPC(1,3)/DOT

VPC(2,1)=VPC(3,2)*VPC(1,3)-VPC(3,3)*VPC(1,2)

VPC(2,2)=VPC(3,3)*VPC(1,1)-VPC(3,1)*VPC(1,3)

VPC(2,3)=VPC(3,1)*VPC(1,2)-VPC(3,2)*VPC(1,1)

WRITE(6,3931)

END IF

Cl

Cl

Cl re-display the pattern cut system

ENTRY SHOW-PAT

DO NI=1,3

WRITE(6,4933) (NI,NJ,VPC(NI,NJ),NJ=1,3)

END DO

4933 FORMAT(2H *,1X,3(2X,'VPC(',I1,'.',',I1,'=',F9.6),708,1H*)

RETURN

1667

Cl

Cl

C! This routine sets/displays a scale factor.

ENTRY SET_SCA

WRITE (2,*) ' Please input a uniform scale factor:'

READ (1,*) UNITF

UNITS = UNITF * UNITN

C! This entry displays the uniform scale factor.

ENTRY SHOW_SCA

WRITE (2,FMT=(' The uniform scale factor is ',F10.8)) UNITF

RETURN

1602

Cl

Cl

Cl This routine sets the units for the program.

C! IUNIT = Indicator of units used for input data.

C! 1=METERS, 2=FEET, 3=INCHES

ENTRY SET_UNIT_METERS

IUNIT = 1

GOTO 2

ENTRY SET_UNIT_FEET

IUNIT = 2

GOTO 2

ENTRY SET_UNIT_INCHES

IUNIT = 3

2 UNIT = UNIT(IUNIT)

UNIT = UNITN * UNITF

RETURN
This entry shows the current units.

ENTRY SHOW_UNI
WRITE (2,FMT='("The current units are ",A6")') LABEL( IUNIT )
RETURN

ENTRY SET_WIN:

ENTRY SHOW_WIN
RETURN

This routine sets the window.
C1 This routine determines names of output files. Here are the current assignments.
C1

C1 Unit Meaning Default file name
C1 1 interactive input sys$input
C1 2 interactive output sys$output
C1 3 input processor input FILE.INP
C1 4 input processor (echo) output FILE.LIS
C1 5 printable output file FILE.PRT
C1 6 "plot" data output file FILE.PLT

ENTRY SET_OUT
CALL CLI$GET_VALUE('P2', FILE)

C1 Only if /NOPLOT is specified, then discard all output written to unit 7.
C1 The user should always get plottable output by default.

C1 IF ( .NOT. CLI$PRESENT('PLOTABLE') ) THEN
OPEN( UNIT=10, FILE='.'NL:' , STATUS='OLD', FORM='UNFORMATTED' )
ELSE
OPEN( UNIT=10, FILE='.PLT', DEFAULTFILE=FILE, STATUS='NEW', FORM='UNFORMATTED' )
ENDIF

C1 IF /PRINT is not specified, discard all output written to unit 7.
C1 The user only wants to see the line printer if he asks for it.

C1 OPEN( UNIT=7, STATUS='OLD', FILE='.'NL:' )
C1 IF ( CLI$PRESENT( 'PRINTABLE' ) ) THEN
OPEN( UNIT=7, DEFAULTFILE=FILE, STATUS='NEW', FILE='.'PRT' )
ENDIF

C1 IF /NOECHO is specified, the input echo is discarded.
C1 The user should get an echo file by default, just like a .PLT file.

C1 IF ( .NOT. CLI$PRESENT( 'ECHOING' ) ) THEN
OPEN( UNIT=6, FILE='.'NL:' , STATUS='OLD' )
ELSE
OPEN( UNIT=6, FILE='.'LIS', DEFAULTFILE=FILE, STATUS='NEW' )
ENDIF

C1 Now retrieve the full filenames for future reference.

C1 INQUIRE ( UNIT = 10, NAME = OUTFIL )
C1 INQUIRE ( UNIT = 7, NAME = PRTFIL )
C1 INQUIRE ( UNIT = 6, NAME = LISFIL )
SET_OUT = SUCCESS

C1 This routine displays the current output files.

ENTRY SHOW_OUT
TYPE *, 'Plotting file is: ', OUTFIL
TYPE *, 'Printer file is: ', PRTFIL
TYPE *, 'Input echo file: ', LISFIL
SHOW_OUT = SUCCESS
RETURN

C1 This routine stops the program.

ENTRY EXIT_COMMAND
CALL EXIT
This routine services online help requests.

ENTRY HELP_COMMAND
LIBRARY = '
P1 = ' 
CALL CL$GET_VALUE( 'P1', P1 )
CALL CL$GET_VALUE( 'HELP', LIBRARY )
HELP_COMMAND = LIB$OUTPUT(HELP)

+ PUT_OUTPUT,, Help output routine
+ P1,, Help key description
+ LIBRARY,, Help library name
+ GET_INPUT ) The prompting input

RETURN

ENTRY SHADOW_COMMAND

ENTRY DCL_COMMAND

ENTRY DCL_ADD

END
10.2 Non-VAX/VMS Subroutines

The following routines are for the non-interactive implementations of the code. They are used in conjunction with the routines in this chapter that are common to both versions.

MAIN PROGRAM (non-interactive)

This is the main routine to be used with the non-interactive code.

```
0001 PROGRAM SHADOW
0002  C111
0003  C111 THIS COMPUTE CODE WAS WRITTEN AT THE OHIO STATE UNIVERSITY
0004  C111 ELECTROSCIENCE LABORATORY. ANY PROBLEMS OR COMMENTS
0005  C111 CAN BE REFERRED TO:
0006  C111
0007  C111 RONALD J. MARHEFKA OR LASZLO A. TAKACS
0008  C111 ELECTROSCIENCE LABORATORY
0009  C111 1320 KINNEAR RD.
0010  C111 COLUMBUS, OHIO 43212
0011  C111 POSE: (614) 422-6762 OR 422-6848
0012  C111
0013  C111 THIS COMPUTER CODE CALCULATES SHADOWING OF AN ANTENNA
0014  C111 IT SHOULD BE USED IN PLACE OF INTERACTIVE MAIN PROGRAM
0015  C111 WHEN THE SHADOW CODE IS USED ON NON VAX COMPUTERS.
0016  C111
0017  C111
0018  INCLUDE 'SHACOM.FOR'
0019  PARAMETER (NSX=30)
0020  COMPLEX WS
0021  LOGICAL LRET
0022  COMPLEX/XS(NSX),XSS(3,NSX),MSA(2,NSX),MSX,MSPP
0023  C111 Initialize fill tags
0024  C111 FILPNM and FILCNM < 0 is sequential tagging
0025  C111 FILPNM or FILCNM > 0 that object is tagged with
0026  C111 FILCHP or FILCHC
0027  C111 FILPNM = 0 everything tagged with FILCHR
0028  C111
0029  FILPNM=0
0030  FILPNM=0
0031  C111 Initialize fill characters
0032  C111 FILCHP='P'
0033  C111 FILCHC='C'
0034  C111 FILCHR='X'
0035  C111 Initialize return flag
0036  C111 LRET=.TRUE.
0037  C111 Initialize and read command information.
0038  C111 CALL ABSCOM
0039  C111 CONTINUE
0040  C111 Choose a source location from stored positions.
0041  DO 100 MS=1,MSX
0042  DO 100 N=1,3
0043  ANTEM(N)=XSS(N,MS)
0044  DO 100 N=1,3
0045  DO 100 MS=1,MSX
0046  C111 Initialize graphics information.
0047  C111 CALL IDITGF
0048  C111 Calculate shadow of plates.
0049  C111 CALL DOPLAS
0050  C111 Calculate shadow of cylinders.
0051  C111 CALL DCYLS
0052  C111 Write out maps to printer and plotter files.
0053  C111 CALL WRTOUT
0054  CONTINUE
```

146
0217  C111 Read more command information.
0218   CALL ABSCMR
0219  C111 Return to execute next shadow map.
0220   IF(LRET) GO TO 100
0221   STOP
0222  END
10.3 Subroutines common to both modes

The following routines are used by both the interactive and non-interactive implementations of the code. They are written in transportable FORTRAN-77.

SUBROUTINE ABSCIN

This is the input-set processor routine. It reads commands from the input file which define the input geometry.

0001 C-------------------------------------------------------------------
0002            SUBROUTINE    ABSCIN
0003 C-------------------------------------------------------------------
0005 C111 THE NEC - BASIC SCATTERING CODE (NEC-BSC) WAS WRITTEN
0006 C111 AT THE OHIO STATE UNIVERSITY ELECTROSCIENCE LABORATORY.
0006 C111 ANY PROBLEMS OR COMMENTS CAN BE REFERRED TO:
0007 C111
0008 C111 RONALD J. MARHEFKA
0009 C111 ELECTROSCIENCE LABORATORY
0010 C111 1520 KINNEAR RD.
0011 C111 COLUMBUS, OHIO 43212
0012 C111 PHONE: (614) 422-6762
0013 C111
0014 C111 THIS IS A PORTION OF THE MAIN PROGRAM OF THE NEC-BSC.
0016 C111 IT READS IN THE INPUT AND PASSES THE GEOMETRY INFORMATION
0017 C111 TO THE SHADOW CALCULATION PART OF THIS OBSCURATION CODE.
0017 C111 IT READS LOCATIONS OF SOURCES A NUMBER OF FINITE
0018 C111 PLATES AND/OR A SET OF FINITE
0019 C111 ELLIPTIC CYLINDERS AND CONE FRUSTUM SECTIONS.
0020 C111 THE PLATES ARE DEFINED
0021 C111 BY THEIR CORNER LOCATIONS. THEY CAN BE PERFECTLY
0022 C111 CONDUCTING, MULTI LAYERED DIELECTRIC SLABS, OR COATED
0023 C111 METAL PLATES. An INFINITE GROUND PLANE CAN ALSO BE
0024 C111 ADDED. THE CYLINDERS ARE DEFINED BY THEIR ORIGIN,
0025 C111 AXES DIRECTIONS, AND BY THE RADIUS ON THEIR MAJOR
0026 C111 AND MINOR AXES AND THE ENDCAPS AND FRUSTUM RIMS ARE DEFINED BY
0027 C111 THEIR POSITION ON THE CYLINDER AXIS AND THE ANGLE
0028 C111 OF THEIR SURFACES WITH THE CYLINDER AXIS IN THE I-Z
0029 C111 CYLINDER PLANE. THE CYLINDERS MUST BE PERFECTLY
0030 C111 CONDUCTING. AS DIMENSIONED, IT CAN HANDLE 76 PLATES
0031 C111 WITH A MAXIMUM OF 12 CORNERS PER PLATE, WITH 6 LAYERS
0032 C111 OF DIELECTRIC AND 6 CYLINDERS, WITH 10 RIMS
0033 C111 ALSO 30 TRANSMITTING
0034 C111 ELEMENTS AND 30 RECEIVING ELEMENTS CAN BE INPUT.
0035 C111 NOTE THAT THE LIMITS ON THE NUMBER OF PLATES,
0036 C111 CORNERS, LAYERS, CYLINDERS, SOURCES, AND RECEIVERS
0037 C111 ARE ONLY DUE TO THE SIZE OF THE ARRAYS.
0038 C111 THE LINEAR DIMENSIONS ARE INPUT IN METERS UNLESS
0039 C111 SPECIFIED OTHERWISE. THE ANGULAR DIMENSIONS
0040 C111 ARE IN DEGREES.
0041 C111
0042 C111 NOTE THAT COMMENTS ARE INDICATED IN DIFFERENT FORMS:
0043 C111 C111 IMPLIES EXPLANATION OF PROGRAM SECTION
0044 C111 C### IMPLIES DESCRIPTION OF INPUT DATA
0045 C111 C--- IMPLIES COMMAND INPUT SECTION
0046 C111 C+++ IMPLIES BEGINNING OF SUBROUTINE
0047 C111 C+++ IMPLIES SPECIFICATION OF MAXIMUM DIMENSIONS
0048 C111 C111 means lines were not needed for SHADOW program
0049 C111 C111 means lines were not implemented for current version

148
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
NOTE ON VERSION 2.2
1) THE PLATE - CYLINDER TERMS ARE NOT PRESENTLY INCLUDED.
2) THE CYLINDER - CYLINDER INTERACTION TERMS WORK ONLY
FOR PARALLEL CYLINDERS WITH THE PATTERN CUT
PERPENDICULAR TO THE CYLINDER AXES.
NOTE ON VERSION 2.3
RANGE GATING HAS BEEN ADDED IN THE NEAR ZONE
NOTE ON VERSION 2.4
VOLUMETRIC PATTERN CAPABILITY HAS BEEN ADDED
NOTE ON VERSION 2.6
PARAMETER STATEMENTS FOR DIMENSIONS ADDED
NOTE ON VERSION 2.6
ARRAY INDICES CHANGED FOR MORE EFFICIENCY
NOTE ON VERSION 2.6
NOTE ON VERSION 2.6
NOTE ON VERSION 2.6
NOTE ON VERSION 2.6
NOTE ON VERSION 2.6
NOTE ON VERSION 2.6
NOTE ON VERSION 2.6
NOTE ON VERSION 2.6
NOTE ON VERSION 2.6
NOTE ON VERSION 2.6
NOTE ON VERSION 2.6
RANGE CITING HAS BEEN ADDED IN THE NEAR ZONE
VOLUMETRIC PATTERN CAPABILITY HAS BEEN
ADDED
PARAMETER STATEMENTS FOR DIMENSIONS ADDED
ARRAY INDICES CHANGED FOR MORE EFFICIENCY
NOTE ON VERSION 2.6
RANGE GATING HAS BEEN ADDED IN THE NEAR ZONE
NOTE ON VERSION 2.3
RANGE GATING HAS BEEN ADDED IN THE NEAR ZONE
NOTE ON VERSION 2.4
VOLUMETRIC PATTERN CAPABILITY HAS BEEN ADDED
NOTE ON VERSION 2.6
PARAMETER STATEMENTS FOR DIMENSIONS ADDED
NOTE ON VERSION 2.6
ARRAY INDICES CHANGED FOR MORE EFFICIENCY
NOTE ON VERSION 2.6
RANGE GATING HAS BEEN ADDED IN THE NEAR ZONE
NOTE ON VERSION 2.3
RANGE GATING HAS BEEN ADDED IN THE NEAR ZONE
NOTE ON VERSION 2.4
VOLUMETRIC PATTERN CAPABILITY HAS BEEN
ADDED
NOTE ON VERSION 2.6
PARAMETER STATEMENTS FOR DIMENSIONS ADDED
NOTE ON VERSION 2.6
ARRAY INDICES CHANGED FOR MORE EFFICIENCY
COMMON/COMP/CJ.CPJ4
COMMON/FLNX/FPN(HEX, NPL)
COMMON/LPLC/LPLA.LCTL
COMMON/GROUND/LGRD.MPXR
COMMON/OUTPFI/T/PD, PRAD, RANG, LCPAT, LPRAD, LITRAD
COMMON/OUTPH2/RXS, RIX, TYS, TYP, PBS, P2I, LRECT
COMMON/OUTPHV/IVPN, IV, LVLDP
COMMON/TRANS/LSLAB(NPL), LSLAB(NPL), DSLAB(NPL, NPL)
COMMON/LSLAB(NPL, NPL), TESLAB(NPL, NPL), URSLAB(NPL, NPL)
COMMON/LDLAB(NPL, NPL)
DATA L1/, 'METERS', 'FEET', 'INCHES'/
DATA IT/TO, 'PD', 'PC', 'SQ', 'LP', 'PP', 'PQ', 'LQ', 'RT', 'CG'
DATA 'SN', 'BD', 'CM', 'CE', 'BP', 'UN', 'UR', 'FR', 'HX'
DATA 'JH', 'HL', 'JH', 'HL', 'NS', 'FR', 'US', 'PN', 'RG', 'HR'
DATA 'JH', 'HL', 'JH', 'HL', 'NS', 'FR', 'US', 'PN', 'RG', 'HR'
2, 'SA', 'FM', 'RA', 'GR', 'VD', 'VK', 'VP', 'PF', 'VF', 'CC'/
C111 MAX. DIMENSION OF SOURCES, RECEIVERS, CYLINDERS, RIMS, PLATES, EDGES,
C111 LAYERS, AND OBSERVATION POINTS.
C111
C111 SET TIME FLAGS TO ZERO
C111
C111 NOTE: IN SUB. FPCTI THE VARIABLES IVT, PHOT, PHORP, AND VRO
C111 MUST BE DIMENSIONED 2*MPX+1
C111
C111
C111 INITIALIZE DATA TO DEFAULT VALUES.
C111
C111 TEST OUTPUT DEFAULT DATA TO:
C111
C111
C111
C111
C111
C111
LPRAD=.FALSE.
PRAD=0.
IPAD=1

PATTERN DEFAULT DATA PD:, PN:, PF:, VD:, VF:, & VH:
LVOLP=.TRUE.
LPANR=.TRUE.
LINEAR=.FALSE.
LRECT=.FALSE.
LCNPAT=.TRUE.

TPPD=0.
TPPV=2.
TPPS=0.
TPPI=2.
THCZ=0.
THCZ=0.

LPHIZ=.FALSE.
THCZ=0.

VPC(l,l)=1.
VPC(1,2)=0.
VPC(1,3)=0.
VPC(2,0)=0.
VPC(2,2)=1.
VPC(2,3)=0.
VPC(1,3)=0.
VPC(2,3)=0.
VPC(3,3)=1.

XPC(1)=0.
XPC(2)=0.
XPC(3)=0.

BJS=-1.
RIC=0.
TYS=0.
TYI=2.

PZS=0.
PZI=2.
IVPN=3
NPN=181

BACK OR BISTATIC NEAR ZONE SCATTERING DEFAULT DATA BP:
LSCAT=.FALSE.

FREQ=.2997926
LFQC=.FALSE.

FRQC=.2997926
FQCI=0.
NFQC=1

PLATE DEFAULT DATA PC:
LPLA=.FALSE.

MPX=0
NEN(1)=4

LSLAB(1)=0

XX(1,1,1)=1.
XX(2,1,1)=1.
XX(3,1,1)=0.

XX(1,2,1)=-1.
XX(2,2,1)=1.
XX(3,2,1)=0.

XX(1,3,1)=1.
XX(2,3,1)=1.

XX(3,3,1)=1.

GROUNDF PLANE DEFAULT DATA GP:
LCRND=.FALSE.
MPXR=MPX

C### SOURCE DEFAULT DATA SG: ,SA: ,& SM:
LSNP=.FALSE.
MSI=0
MSAT=0
MSA(1,1)=0
MSA(2,1)=0
ISS(1,1)=0.
ISS(2,1)=0.
ISS(3,1)=0.
IMS(1)=1
HS(1)=0.5
HAWS(1)=0.
THSZ=0.
THS=0.
PHSZ=0.

C### RECEIVER DEFAULT DATA RG: ,RA: ,& RM:
LCVCR=.FALSE.
LMNP=.FALSE.
MRX=0
MRAT=0
MRA(1,1)=0
MRA(2,1)=0
ARR(1,1)=0.
ARR(2,1)=0.
ARR(3,1)=0.
IMR(1)=1
HR(1)=0.5
HAWR(1)=0.
THRZ=0.

C### LINE PRINTER DEFAULT DATA LP:
LWRITE=.FALSE.

C### PLOTTER DEFAULT DATA PP: & VP:
DPLOT=.FALSE.
LPLT=.FALSE.
LPPREC=.FALSE.
PPXL=0.
PPYL=3.
PPXB=0.
PPX=360.
PPY=30.
PPB=-40.
PPY=0.
PPY=10.

### ROTATE TRANSLATE DEFAULT DATA RT:

THZP=0.
PHZP=0.
THXP=90.
PHXP=0.
TH(1)=0.
TH(2)=0.
TH(3)=0.
VRT(1,1)=1.
VRT(1,2)=0.
VRT(1,3)=0.
VRT(2,1)=0.
VRT(2,2)=1.
VRT(2,3)=0.
VRT(3,1)=0.
VRT(3,2)=0.
VRT(3,3)=1.

### CYLINDER DEFAULT DATA Co: & Co:

MC=0
LCIL=.FALSE.
MI=0
NC=2
AC(1,1)=1.
BC(1,1)=1.
AC(2,1)=1.
BC(2,1)=1.
ZC(2,1)=-3.
TCR(1,1)=1.570796
ZC(1,1)=3.
TCR(1,1)=1.570796
VCL(1,1,1)=1.
VCL(1,2,1)=0.
VCL(1,3,1)=0.
VCL(2,1,1)=0.
VCL(2,2,1)=1.
VCL(2,3,1)=0.
VCL(3,1,1)=0.
VCL(3,2,1)=0.
VCL(3,3,1)=1.
XCL(1,1)=0.
XCL(2,1)=0.
XCL(3,1)=0.

### UNITS DEFAULT DATA UN: ,UF: & US:

IUNIT=1
UNIT=UNIT(IUNIT)
UNIT=1.
UNIT=UNIT+UNITF
IUNIT=0
IUNIT+IUNIT
GO TO 2999
ENTRY ABSCRE

3000 CONTINUE
WRITE(6,3006)
FORMAT(11,1H*,7E1,1H*)
WRITE(6,3006)
WRITE(6,3006)
FORMAT(11,26SH++)
C!! READ IN VARIOUS COMMAND OPTIONS.
2999 READ(6,3001,END=3004) (IR(I),I=1,36)
0538 3001 FORMAT(38A2)
0537 WRITE(6,3002)
0538 3002 FORMAT(1H, ////////,1X,26(3H***))
0539 WRITE(6,3003) (IR(1), I=1,38)
0540 3003 FORMAT(1X,1H*,2X,38A2,2X,1H*)
0541 C111
0542 C111 CHECK AGAINST STORED OPTIONS
0543 C111
0544 C111
0545 C111 CM: COMMENT CARD
0546 IF(IR(1).EQ.IX(13)) GO TO 3090
0547 C111 CE: LAST COMMENT CARD
0548 IF(IR(1).EQ.IX(14)) GO TO 3000
0549 WRITE(6,3008)
0550 WRITE(6,3009)
0551 C111 TO: TEST DATA GENERATION OPTION.
0552 IF(IR(1).EQ.IX(1)) GO TO 3100
0553 C111 PD: FAR ZONE PATTERN INTEGER ANGLES
0554 IF(IR(1).EQ.IX(2)) GO TO 3200
0555 C111 RD: FAR ZONE RANGE INPUT
0556 IF(IR(1).EQ.IX(12)) GO TO 3250
0557 C111 PG: PLATE GEOMETRY INPUT
0558 IF(IR(1).EQ.IX(3)) GO TO 3300
0559 C111 SG: SOURCE GEOMETRY INPUT
0560 IF(IR(1).EQ.IX(4)) GO TO 3400.
0561 C111 SM: SOURCE NEC OR AMP INPUT
0562 IF(IR(1).EQ.IX(11)) GO TO 3450
0563 C111 LP: LINE PRINTER LISTING OF RESULTS
0564 IF(IR(1).EQ.IX(6)) GO TO 3500
0565 C111 FP: PEN PLOT OF RESULTS
0566 IF(IR(1).EQ.IX(5)) GO TO 3600
0567 C111 GP: INCLUDE INFINITE GROUND PLANE
0568 IF(IR(1).EQ.IX(7)) GO TO 3700
0569 C111 EQ: EXECUTE PROGRAM
0570 IF(IR(1).EQ.IX(8)) GO TO 3800
0571 C111 RT: TRANSLATE AND/OR ROTATE COORDINATES
0572 IF(IR(1).EQ.IX(9)) GO TO 3900
0573 C111 CG: CYLINDER GEOMETRY INPUT
0574 IF(IR(1).EQ.IX(10)) GO TO 4000
0575 C111 CC: CONE GEOMETRY INPUT
0576 IF(IR(1).EQ.IX(40)) GO TO 4000
0577 C111 BP: BACK OR BISTATIC NEAR ZONE SCATTERING
0578 IF(IR(1).EQ.IX(15)) GO TO 5240
0579 C111 UF: SCALE FACTOR FOR INPUT
0580 IF(IR(1).EQ.IX(16)) GO TO 4120
0581 C111 UN: UNITS OF INPUT
0582 IF(IR(1).EQ.IX(18)) GO TO 4100
0583 C111 FR: FREQUENCY
0584 IF(IR(1).EQ.IX(19)) GO TO 4200
0585 C111 NA: HEAT PROBLEM
0586 IF(IR(1).EQ.IX(20)) GO TO 3700
0587 C111 EP: END PROGRAM
0588 IF(IR(1).EQ.IX(21)) GO TO 997
0589 C111 HP: NEXT SET OF PLATES
0590 IF(IR(1).EQ.IX(22)) GO TO 3560
0591 C111 HC: NEXT SET OF CYLINDERS
0592 IF(IR(1).EQ.IX(23)) GO TO 4050
0593 C111 HG: NO GROUND PLANE
0594 IF(IR(1).EQ.IX(24)) GO TO 3760
0595 C111 HS: NEXT SET OF SOURCES
0596 IF(IR(1).EQ.IX(25)) GO TO 3460
0597 C111 FP: POWER RADIATED INPUT
0598 IF(IR(1).EQ.IX(26)) GO TO 3440
0599 C111 US: UNITS OF HS AND HAWS IN SG: . SA: . RC: . & RA:
IF (IR(1).EQ.IT(27)) GO TO 4110
C$88 PN: NEAR ZONE PATTERN DESIRED
IF (IR(1).EQ.IT(28)) GO TO 3260
Ca8$ RC: RECEIVER GEOMETRY
IF (IR(1).EQ.IT(29)) GO TO 4400
C8$$ RM: RECEIVER NEC OR AMP INPUT
IF (IR(1).EQ.IT(30)) GO TO 3496
Cat$ NR: NEXT SET OF RECEIVERS
IF (IR(1).EQ.IT(31)) GO TO 3810
C$$ SA: SOURCE ARRAY GEOMETRY INPUT
IF (IR(1).EQ.IT(32)) GO TO 4260
CaO RA: RECEIVER ARRAY GEOMETRY INPUT
IF (IR(1).EQ.IT(33)) GO TO 4810
Cat$ OR: RANGE INPUT
IF (IR(1).EQ.IT(34)) GO TO 6280
C$88 VD: FAR ZONE VOLUMETRIC PATTERN INTEGER ANGLES
IF (IR(1).EQ.IT(36)) GO TO 9210
C8$8 VN: NEAR ZONE VOLUMETRIC PATTERN
IF (IR(I).EQ.IT(36)) GO TO 3270
CaO VP: VOLUMETRIC DUMP OF RESULTS FOR PLOTTING
IF (IR(1).EQ.IT(37)) GO TO 3660
C$$ PF: FAR ZONE NON INTEGER ANGLES
IF (IR(1).EQ.IT(38)) GO TO 3220
CaSS VF: FAR ZONE VOLUMETRIC PATTERN NON INTEGER ANGLES
IF (IR(1).EQ.IT(39)) GO TO 3230
CS$$ YRITE(6.3021) FORNAT('*** PROGRAM ABORTS!!! COMMAND INPUT IS NOT PART', 2' OF STORED COMMAND LIST ***)
STOP c====.. 3090 CONTINUE
C=== CM: CE: COMMANDS ======
C$$ LO UNT=OUTPUT MAIN PROGRAM DATA ON LINE PRINTER(TRUE OR FALSE)
C$$ LTEST=TEST DATA TO INSURE PROGRAM OPERATION(TRUE OR FALSE)
C$$ LDEBUG=DEBUG DATA OUTPUT OR LINE PRINTER(TRUE OR FALSE)
C$$ LWARN=WARNING DATA OUTPUT OR LINE PRINTER(TRUE OR FALSE)
READ(6,3101) LDEBUG,LTEST,LOUT,LWARN
WRITE(6,3101) LDEBUG,LTEST,LOUT,LWARN
3101 FORMAT(2H*,6X,'LDEBUG=',L3,6X,'LTEST=',L3,6X,'LOUT=',L3
2.6X,'LWARN=',L3,T79,1H+)
3106 WRITE(6,3006)
3107 C### LSLOPE=SLOPE DIFFRACTED FIELD DESIRED (T OR F)
3108 C### LCORNR=CORNER DIFFRACTED FIELD DESIRED (T OR F)
3109 C### LSOR=SOURCE SHADOW ALONE (T OR F)
3112 C### LSOR=ANTENNA SHADOW ALONE (T or F)
3174 READ(6,*) LSLOPE,LCORNR,LSOR
3176 WRITE(6,3102) LSLOPE,LCORNR,LSOR
3102 FORMAT(2H*,6X,'LSLOPE=',L3,6X,'LCORNR=',L3,6X,'LSOR=',L3,
2T79,1H+)
3178 WRITE(6,3006)
3179 IF(LS0R) WRITE(6,3402)
3402 FORMAT(2H*,6X,'SOURCE SHADOW ALONE IS COMPUTED!!!',T79,1H+)
3181 IF(LS0R) WRITE(6,3008)
3182 C###
3183 C### K=1,J=OPTION TO RUN DIRECT RAY TERM:
3184 C### 1=DIRECT FIELD
3185 C### 2=DIRECT FIELD
3186 C### 3=SINGLE REFLECTED FIELD
3187 C### 4=DOUBLE REFLECTED FIELD
3188 C### 5=SINGLE DIFFRACTED FIELD
3189 C### 6=DOUBLE DIFFRACTED FIELD
3190 C### K=2,J=OPTION TO RUN VARIOUS RAY TERMS FOR PLATES:
3191 C### 1=SINGLE REFLECTED FIELD
3192 C### 2=DOUBLE REFLECTED FIELD
3193 C### 3=SINGLE DIFFRACTED FIELD
3194 C### 4=DOUBLE DIFFRACTED FIELD
3195 C### 5=REFLECTED/DIFFRACTED FIELD
3196 C### 6=DIFFRACTED/REFLECTED FIELD
3197 C### 7=REFLECTED/DIFFRACTED FIELD
3198 C### 8=DIFFRACTED/REFLECTED FIELD
3199 C### 9=DIFFRACTED-SCATTERED FIELD FROM TWO PARALLEL CYLINDERS
3200 C### 10=DIFFRACTED-SCATTERED FIELD FROM ENDCAP RIMS
3201 C### 11=DIFFRACTED-SCATTERED FIELD
3202 C### 12=DIFFRACTED-SCATTERED FIELD
3203 C### 13=DIFFRACTED-SCATTERED FIELD FROM TWO PARALLEL CYLINDERS
3204 C### NOTE: NORMALLY LKJ(3,1 TO 6) = .TRUE. THIS COMPUTES ALL FIELD
3205 C### VALUES INCLUDING IDENTIFYING DOUBLE DIFFRACTION PROBLEM AREAS
3206 C### FOR A CONVEX OR CONCAVE PLATE STRUCTURE.
3207 C###
3208 C### K=3,J=OPTION TO RUN VARIOUS RAY TERMS FOR CYLINDER:
3209 C### 1=REFLECTED,TRANSMISSION, AND CREEPING WAVE FIELDS
3210 C### 2=REFLECTED,TRANSMISSION, AND CREEPING WAVE FIELDS
3211 C### 3=REFLECTED,TRANSMISSION, AND CREEPING WAVE FIELDS
3212 C### 4=REFLECTED,TRANSMISSION, AND CREEPING WAVE FIELDS
3213 C### 5=REFLECTED,TRANSMISSION, AND CREEPING WAVE FIELDS
3214 C### 6=REFLECTED,TRANSMISSION, AND CREEPING WAVE FIELDS
3215 C### 7=REFLECTED,TRANSMISSION, AND CREEPING WAVE FIELDS
3216 C### 8=REFLECTED,TRANSMISSION, AND CREEPING WAVE FIELDS
3217 C### 9=REFLECTED,TRANSMISSION, AND CREEPING WAVE FIELDS
3218 C### 10=REFLECTED,TRANSMISSION, AND CREEPING WAVE FIELDS
3219 C### 11=REFLECTED,TRANSMISSION, AND CREEPING WAVE FIELDS
3220 C### 12=REFLECTED,TRANSMISSION, AND CREEPING WAVE FIELDS
3221 C### 13=REFLECTED,TRANSMISSION, AND CREEPING WAVE FIELDS
3222 C### NOTE: NORMALLY LKJ(3,1 TO 6) = .TRUE. THIS COMPUTES ALL FIELD
3223 C### VALUES FOR A FINITE ELLIPTIC CYLINDER.
3224 C###
3225 C### K=4,J=OPTION TO RUN VARIOUS RAY TERMS FOR PLATE-CYLINDER:
3226 C### 1=FIELDS REFLECTED FROM THE PLATES THEN REFLECTED OR
3227 C### 2=FIELDS REFLECTED FROM THE PLATES THEN REFLECTED OR
3228 C### 3=FIELDS REFLECTED FROM THE PLATES THEN REFLECTED OR
3229 C### 4=FIELDS REFLECTED FROM THE PLATES THEN REFLECTED OR
3230 C### 5=FIELDS REFLECTED FROM THE PLATES THEN REFLECTED OR
3231 C### 6=FIELDS REFLECTED FROM THE PLATES THEN REFLECTED OR
3232 C### 7=FIELDS REFLECTED FROM THE PLATES THEN REFLECTED OR
3233 C### 8=FIELDS REFLECTED FROM THE PLATES THEN REFLECTED OR
3234 C### 9=FIELDS REFLECTED FROM THE PLATES THEN REFLECTED OR
3235 C### 10=FIELDS REFLECTED FROM THE PLATES THEN REFLECTED OR
3236 C### 11=FIELDS REFLECTED FROM THE PLATES THEN REFLECTED OR
3237 C### 12=FIELDS REFLECTED FROM THE PLATES THEN REFLECTED OR
3238 C### 13=FIELDS REFLECTED FROM THE PLATES THEN REFLECTED OR
3239 C### NOTE: NORMALLY LKJ(4,1 TO 4) = .TRUE. THIS COMPUTES ALL FIELD
3240 C### VALUES THAT INTERACT BETWEEN THE PLATES AND CYLINDERS.
3241 C###
3242 K=1,J=OPTION TO RUN DIRECT RAY TERM:
3243 JK=JMK(K)
3244 READ(6,*) (LJ(J),J=1,JK)
3245 WRITE(6,3103) K,(LJ(J),J=1,JK)
3246 FORMAT(2H*,T79,1H*,T8,':LJ(J),I,J)=',6L2)
3247 GO TO 3000
3248 C====(==
3249 4100 CONTINUE

156
Cm==

W:= COMMAND ==IS==

c:: Cm=

IUNIT-INDICATOR OF UNITS USED FOR INPUT DATA.

a:

l=METERS

3=INCHES

c::

READ(6,*) IUNIT

UNIT=UNIT(IUNIT)

WRITE(6,4101) LABEL(IUNIT)

4101 FORMAT(2H *,6X,'ALL THE LINEAR DIMENSIONS BELOW ARE'

2,' ASSUMED TO BE IN ',A6,T79,1H*)

GO TO 3000

4120 CONTINUE

C== UP: COMMAND ==

C=: UNIT = SCALE FACTOR FOR GEOMETRY

C:

READ(6,*) UNITF

UNITS=UNITN*UNITF

WRITE(6,4111) UNITF

4111 FORMAT(1H *,C6X,'ALL THE LINEAR Dimensions'

2,' ARE SCALLED A FACTOR OF ',F12.6,T79,1H*)

GO TO 3000

C== IUNST=INDICATOR OF UNITS USED FOR HS AND HAWS IN THE

C:

0=WAVELENGTHS

3=METERS

2=FEET

3=INCHES

C:

NOTE: IF ONE SOURCE IS SPECIFIED IN WAVELENGTHS, THEY ALL

MUST BE IN WAVELENGTHS.

READ(6,*) IUNST

IF(NSI.EQ.0) GO TO 4112

4112 CONTINUE

IF(IUNST.EQ.0.AND.IUNSP.EQ.0) GO TO 4112

WRITE(6,4111)

4111 FORMAT(’ *** PROGRAM ABOARDS IN SOURCE UNITS. ALL UNITS NOT'

2,’ SPECIFIED IN WAVELENGTHS!!! ***’)

STOP

IF(IUNST.EQ.0) GO TO 4114

WRITE(6,4113) LABEL(IUNST)

4113 FORMAT(2H *,6X,'THE SOURCE LENGTH HS AND WIDTH HAWS ARE'

2,' ASSUMED TO BE IN ',A6,T79,1H*)

GO TO 4116

WRITE(6,4116)

4114 FORMAT(2H *,6X,'THE SOURCE LENGTH HS AND WIDTH HAWS ARE'

2,' ASSUMED TO BE IS WAVELENGTHS',T79,1H*)

GO TO 3000

C==

4200 CONTINUE

C== FR: COMMAND ==

C:

FRQG=FREQUENCY IN GIGAHERTZ.

C:

157
LFQC=.FALSE.
NFQC=1
READ(6,+) FRQC
WL=.297926/FRQC
WRITE(6,4201) FRQC
4201 FORMAT(2H *,6X,'FREQUENCY=',F7.3,' GIGAHERTZ',7H*)
WRITE(6,3006)
WRITE(6,4202) WL
4202 FORMAT(2H *,6X,'WAVELENGTH=',F10.6,' METERS',7H*)
GO TO 3000
C======
4260 CONTINUE
C=== FM: COMMAND ======
C###
NFQC=NUMBER OF FREQUENCIES DESIRED
C###
FQC=STARTING FREQUENCY IN GIGAHERTZ
C###
FQCI=INCREMENTAL FREQUENCY CHANGE IN GIGAHERTZ
C###
NOTE: THE SOURCE LENGTH AND WIDTH MUST NOT BE SPECIFIED
C###
IS WAVELENGTH. ALSO ONLY ONE PATTERN LOCATION
C###
CAN BE SPECIFIED.
C###
LFQC=.TRUE.
READ(6,+) NFQC,FQCS,FQCI
WRITE(6,4265) NFQC
4261 FORMAT(2H *,6X,'FREQUENCIES ARE SPECIFIED',7H*)
IF(NFQC.GT.MODX) WRITE(6,3286) NFQC
IF(NFQC.GT.MODX) STOP
WRITE(6,3006)
WRITE(6,4265) NFQC,FQCS
4262 FORMAT(2H *,6X,'STARTING FREQ.=',F10.6,' IN STEPS OF ',F10.6
2,',' GHz.',7H*)
C111 CALCULATE MID-FREQUENCY
FQG=FQCS+0.5*FQCI*(NFQC-1)
WL=.2997925/FRQC
GO TO 3000
C======
3230 CONTINUE
C=== VF: COMMAND ======
C###
FAR ZONE VOLUMETRIC PATTERN NON INTEGER ANGLES
C###
LVOLP=.TRUE.
LFAR=.TRUE.
GO TO 3211
C======
3210 CONTINUE
C=== VF: COMMAND ======
C###
FAR ZONE VOLUMETRIC PATTERN INTEGER ANGLES
C###
LVOLP=.TRUE.
LFAR=.FALSE.
GO TO 3211
C======
3220 CONTINUE
C=== PF: COMMAND ======
C###
FAR ZONE PATTERN NON INTEGER ANGLES
C###
LVOLP=.FALSE.
LFAR=.TRUE.

158
GO TO 3211
C=====
3200 CONTINUE
C== PD: COMMAND ======
C###
C### FAR ZONE PATTERN INTEGER ANGLES
C###
C### THCZ,PHCZ=ORIENTATION OF THE Z AXIS RELATIVE TO THE
C### FIXED COORDINATE SYSTEM.
C###
C### THCZ,PHCZ=ORIENTATION OF THE X AXIS RELATIVE TO
C### FIXED COORDINATE SYSTEM.
C###
C### LVOLP=.FALSE.
C###
3211 LINEAR=.FALSE.
C### FORMAT: *** PROGRAM ABOORTS IN PATTERN CUT SECTION.***
C### 2, 'THE COORDINATES ARE NOT ORTHOGONAL!!! ***')
C### IF(ABS(DZI).GT.0.1) STOP
C### VPC(1,1)=VPC(1,1)-VPC(3,1)*DZI
C### VPC(1,2)=VPC(1,2)-VPC(3,2)*DZI
C### VPC(1,3)=VPC(1,3)-VPC(3,3)*DZI
C### DOT=VPC(1,1)*VPC(1,1)+VPC(1,2)*VPC(1,2)*VPC(1,3)*VPC(1,3)
C### DOT=SQRT(DOT)
C### VPC(1,1)=VPC(1,1)/DOT
C### VPC(1,2)=VPC(1,2)/DOT
C### VPC(1,3)=VPC(1,3)/DOT
C### VPC(2,1)=VPC(2,1)*VPC(1,1)-VPC(2,3)*VPC(2,3)
C### VPC(2,2)=VPC(2,2)*VPC(1,2)-VPC(2,3)*VPC(2,3)
C### VPC(2,3)=VPC(2,3)*VPC(1,3)-VPC(2,3)*VPC(2,3)
C### WRITE(6,3202)
C### FORMAT(2H*.6X,'THE PATTERN AXES ARE AS FOLLOWS:',T79,1H+)
C### DO 3204 NI=1,3
C### WRITE(6,3206)
C### WRITE(6,3206)
C### WRITE(6,3206)
C### IF(LCWPAT=T) TPPD=THF'
C### IF(LCWPAT=F) TPPD=PHP
C### IF(LVOLP) GO TO 3212
C### TPPF=0.
C### NPV=1
C### READ(*,*) LCWPAT,TPPD
C### WRITE(6,3006)
C### IF(.NOT.LCWPAT) WRITE(6,3206) TPPD
C### FORMAT(2H*.6X,'THETA IS BEING VARIED WITH PHI=',F10.6
C### 2,T79,1H+)
IF(LCNPAT) WRITE(6,3207) TPPD
0210    3207 FORMAT(2H *,6X,'PHI IS BEING VARIED WITH THETA= ',F10.6
0212                2,T79,1H*)
0213    WRITE(6,3006)
0214    GO TO 3216
0215
0216    C###
0217    C###  TPDD=START OF VOLUMETRIC PATTERN ANGLE
0218    C###  TPPV=INCREMENT FOR VOLUMETRIC PATTERN ANGLE
0219    C###  NPV=NUMBER OF VOLUMETRIC PATTERN ANGLES
0220    C###
0221    3212 READ(*,*) LCNPAT,TPDD,TPPV,NPV
0222    WRITE(6,3006)
0223    IF(LCNPAT) WRITE(6,3213)
0224    3213 FORMAT(2H *,6X,'FOR THETA ANGLE: ',T79,1H*)
0225    IF(.NOT.LCNPAT) WRITE(6,3214)
0226    3214 FORMAT(2H *,6X,'FOR PHI ANGLE: ',T79,1H*)
0227    WRITE(6,3215) TPDD,TPPV,NPV
0228    3216 FORMAT(2H *,6X,'START= ',F10.6,' STEP= ',F10.6,' NUMBER= ',I4
0229                2,T79,1H*)
0230    WRITE(6,3006)
0231    IF(LCNPAT) WRITE(6,3213)
0232    IF(.NOT.LCNPAT) WRITE(6,3214)
0233    3216 CONTINUE
0234    IF(LFARN) GO TO 3217
0235
0236    C###
0237    C###  IB,IE,IS=BEGIN,END,STEP
0238    C###
0239    3217 CONTINUE
0240    C###
0241    3218 CONTINUE
0242    3219 CONTINUE
0243    C###
0244    C###  TPPL=START OF PATTERN
0245    C###  TPPI=PATTERN INCREMENT
0246    C###  NPNI=NUMBER OF PATTERN POINTS
0247    C###
0248    3219 CONTINUE
0249    3218 CONTINUE
0250    3217 CONTINUE
0251    3216 FORMAT(2H *,6X,'START= ',F10.6,' STEP= ',F10.6,' NUMBER= ',I4
0252                2,T79,1H*)
0253    WRITE(6,3006)
0254    IF(LCNPAT) WRITE(6,3213)
0255    IF(.NOT.LCNPAT) WRITE(6,3214)
0256    3216 CONTINUE
0257    IF(LCNPAT) GO TO 3209
0258    3209 CONTINUE
0259    GO TO 3000
0260
0261    C------
0262    3260 CONTINUE
0263
C=== RC: COWND ==I=-=
cat1 ($((
RANCSrFAR FIELD RANGE DISTANCE
C((((
NOTE IF RANCS IS GREATER THAN OR EQUAL TO 1.E30
C$($
THAN LRANG WILL BE SET FALSE
C$($

LRANG=.TRUE.
READ(6,*) RANCS
IF(RANCS.GT.9.9E29) GO TO 3252
RANG=UNITS*RANCS
WRITE(6,3251) RANCS,LABEL(IUNIT),RANG
3251 FORMAT(2H 'THE FAR FIELD RANGE SPECIFIED IS ',E12.6,
2' IN ',A6,TTQ,1H+,/2H 'THE RANGE SPECIFIED IN METERS'
3,' IS ',E12.6,TTQ,1H+)
GO TO 3000
3252 CONTINUE
LRANG=.FALSE.
RANG=1.
WRITE(6,3253)
3253 FORMAT(2H 'NO FAR FIELD RANGE SPECIFIED.',TTQ,1H+)
GO TO 3000
3254 CONTINUE
C=== VN: COMMAND ======
C$($
NEAR ZONE VOLUMETRIC PATTERN
C$($
LVOLP=.TRUE.
GO TO 3271
3271 L!IEAR=.TRUE.
3270 CONTINUE
C=== PH: COMMAND ======
C$($
XYZ LOCATION OF THE NEAR ZONE PATTERN ORIGIN
C$($
LVOLP=.FALSE.
GO TO 3271
3271 L!IEAR=.TRUE.
READ(6,*) (XPC(N),N=1,3)
WRITE(6,3264) LABEL(IUNIT), (XPC(N),N=1,3)
3264 FORMAT(2H 'PATTERN ORIGIN IN ',A6,'
2 ',XPC(1),',',F8.3 '
2 ',XPC(2),',',F8.3 ',
2 ',XPC(3),',',F8.3,TTQ,1H+)
WRITE(6,3006)
DO 3263 N=1,3
3263 XPC(N)=UNITS*XPC(N)
IF(IUNIT.NE.1) WRITE(6,3264) LABEL(I), (XPC(N),N=1,3)
IF(IUNIT.NE.1) WRITE(6,3006)
WRITE(6,3000)
3261 CONTINUE
THCZ,PHCZ=ORIENTATION OF THE Z-AXIS OF THE PATTERN AXES
RELATIVE TO THE FIXED COORDINATE SYSTEM
THCX,PHCX=ORIENTATION OF THE X-AXIS OF THE PATTERN AXES
RELATIVE TO THE FIXED COORDINATE SYSTEM
READ(6,*) THCZ,PHCZ,THCX,PHCX
VPC(3,1)=SIN(THCZ*RPD)*COS(PHCZ*RPD)
VPC(3,2)=SIN(THCZ*RPD)*SIN(PHCZ*RPD)
VPC(3,3)=COS(THCZ*RPD)
VPC(1,1)=SIN(THCZ*RPD)*COS(PHCZ*RPD)
VPC(1,2)=SIN(THCZ*RPD)*SIN(PHCZ*RPD)
VPC(1,3)=COS(THCZ*RPD)
 insure VPC(1,3) IS PERPENDICULAR TO VPC(3,3)
DZ1=VPC(3,1)+VPC(1,1)+VPC(3,2)+VPC(1,2)+VPC(3,3)+VPC(1,3)
IF(ABS(DZX) > 0.1) WRITE(6,3201)
IF(ABS(DZX) > 0.1) STOP
VPC(1,1) = VPC(1,1) - VPC(3,1) * DZX
VPC(1,2) = VPC(1,2) - VPC(3,2) * DZX
VPC(1,3) = VPC(1,3) - VPC(3,3) * DZX
DOT = VPC(1,1) + VPC(1,2) * VPC(1,2) + VPC(1,3) * VPC(1,3)

**DO** 3261 NIS = 1,3
**IF** (.NOT. LRECT) WRITE(6,3261) RXS, TYS, PZS, LABEL(IUNIT)
**IF** (.NOT. LRECT) WRITE(6,3262) RXI, TYI, PZI, LABEL(IUNIT)
**IF** (.NOT. LRECT) WRITE(6,3267) RXS, TYS, PZS, LABEL(IUNIT)
**IF** (.NOT. LRECT) WRITE(6,3268) RXI, TYI, PZI, LABEL(IUNIT)
**IF** (.NOT. LRECT) WRITE(6,3006)

**C###** LRECT=F, SPHERICAL PATTERN CUT
**C###** LRECT=T, LINEAR PATTERN CUT

**FORMAT(2H*,2X,'START XYZ='*,F10.6,2(',',F10.6),1XX,A8,T79,1H*)

READ(6,+) LRECT
READ(6,+) RXS, TYS, PZS
READ(6,+) RXI, TYI, PZI

**IF** (LRECT) WRITE(6,3261) RXS, TYS, PZS, LABEL(IUNIT)
**IF** (LRECT) WRITE(6,3262) RXI, TYI, PZI, LABEL(IUNIT)
**IF** (.NOT. LRECT) WRITE(6,3267) RXS, TYS, PZS, LABEL(IUNIT)
**IF** (.NOT. LRECT) WRITE(6,3268) RXI, TYI, PZI, LABEL(IUNIT)
**IF** (.NOT. LRECT) WRITE(6,3006)

**READ(6,*) RXS, TYS, PZS
READ(6,*) RXI, TYI, PZI

**IF** (.NOT. LRECT) WRITE(6,3261) RXS, TYS, PZS, LABEL(IUNIT)
**IF** (.NOT. LRECT) WRITE(6,3262) RXI, TYI, PZI, LABEL(IUNIT)
**IF** (.NOT. LRECT) WRITE(6,3267) RXS, TYS, PZS, LABEL(IUNIT)
**IF** (.NOT. LRECT) WRITE(6,3268) RXI, TYI, PZI, LABEL(IUNIT)
**IF** (.NOT. LRECT) WRITE(6,3006)

**READ(6,+) RXS, TYS, PZS
READ(6,+) RXI, TYI, PZI

**IF** (.NOT. LRECT) WRITE(6,3261) RXS, TYS, PZS, LABEL(IUNIT)
**IF** (.NOT. LRECT) WRITE(6,3262) RXI, TYI, PZI, LABEL(IUNIT)
**IF** (.NOT. LRECT) WRITE(6,3267) RXS, TYS, PZS, LABEL(IUNIT)
**IF** (.NOT. LRECT) WRITE(6,3268) RXI, TYI, PZI, LABEL(IUNIT)
**IF** (.NOT. LRECT) WRITE(6,3006)

**READ(6,+) NPN
WRITE(6,3269) NPN
3269 FORMAT(2H0,+5X,'NUMBER OF PATTERN POINTS= ',I4,T79,1H+)
IVPN=3
IF(ABS(PZ1).LT.SMLA) IVPN=-3
IF(LENCT) IVPN=0
GO TO 3276

C###
IVPN=1 FOR R-THETA OR X-Y VARYING
NPV=NUMBER OF R OR X AND NPN=NUMBER OF THETA OR Y
C###
IVPN=2 FOR R-PHI OR X-Z VARYING
NPV=NUMBER OF R OR X AND NPN=NUMBER OF PHI OR Z
C###
IVPN=3 FOR THETA-PHI OR Y-Z VARYING
NPV=NUMBER OF THETA OR Y AND NPN=NUMBER OF PHI OR Z
C###
IF IVPN IS LESS THAN ZERO THE ORDER IS REVERSED
I.E. IVPN=-1 FOR THETA-R OR Y-X VARYING
C###
RMIN=THE MINIMUM DISTANCE FROM TRANSMITTER TO RECEIVER
RMAX=THE MAXIMUM DISTANCE FROM TRANSMITTER TO RECEIVER
THE SG., SG., AND FN: COMMANDS MUST BE SPECIFIED
TO USE THIS OPTION.
LSCAT=.TRUE.
GO TO 3376
CONTINUE
IF(NPN.GT.MODE) STOP
GO TO 3000

6260 CONTINUE
C### CR: COMMAND ======
163 C###
RANGE GATE INPUT
165 C###
READ(6,*) RMIN,RMAX
WRITE(6,6261) RMIN,RMAX,UNIT(UNIT)
6261 FORMAT(2H0,+2X,'RMIN= ',F10.6,'RMAX= ',F10.6,' IN ',A6,T79,1H+)
UNIT=UNITS+RMIN
RMAX=UNITS+RMAX
WRITE(0,6261) RMIN,RMAX,LABEL(1)
GO TO 3300
C=====
3300 CONTINUE
C== PG: COMMAND ======
C$$$ PLATE GEOMETRY INPUT
C$$$ LPLA=. TRUE.
MPX=MPX+1
IF(4MPX.GT.MPD) WRITE(6,5001) MPX
901 FORMAT(’ ***** NUMBER OF PLATES= ’,I3,’ PROGRAM ABORTS’,
2’ SINCE MAX. PLATE DIMENSION IS EXCEEDED. *****’)
IF(4MPX.GT.MPD) STOP
WRITE(6,3301) MPX
3301 FORMAT(2H *,6X,’THIS IS PLATE NO. ’,I3,’ IN THIS ’,
2’ SIMULATION’,779,1H+)
MPX=MPX
WRITE(6,3008)
WRITE(6,3006)
WRITE(6,3009)
C$$$ NCP(MP)=NUMBER OF CORNERS ON THE MP-TH PLATE.
C$$$ LSLAB= 1 IMPLIES TRANSPARENT THIN DIELECTRIC SLAB
C$$$ = 0 IMPLIES METAL PLATE, AND
C$$$ =-2 IMPLIES DIELECTRIC COVERED PLATE ON BOTH SIDES
C$$$ =-4 IMPLIES DIELECTRIC COVERED PLATE ON SIDE OF NORMAL
C$$$ NOTE: IF DIELECTRIC COVERED, ONE MUST READ DIELECTRIC DATA.
C$$$ LSLAB(MP)=NUMBER OF DIELECTRIC LAYERS ON THE MP-TH PLATE.
1210 C$$$ READ(6,*) MCP(MP), LSLAB(MP)
1211 IF(LSLAB(MP).EQ.0) WRITE(6,3392)
1212 3392 FORMAT(2H *,6X,’METAL PLATE USED IN THIS SIMULATION’,779,1H+)
1213 IF(LSLAB(MP).EQ.1) WRITE(6,3393)
1214 3393 FORMAT(2H *,6X,’TRANSPARENT THIN DIELECTRIC LAYER USED IN THIS’,
1215 2’ SIMULATION’,779,1H+)
1216 IF(LSLAB(MP).EQ.-2) WRITE(6,3394)
1217 3394 FORMAT(2H *,6X,’DIELECTRIC COVERED PLATE USED IN THIS’,
1218 2’ SIMULATION’,779,1H+)
1219 WRITE(6,3008)
1220 IF(LSLAB(MP).EQ.0) GO TO 3313
1221 C$$$ NSLAB(MP)=NUMBER OF DIELECTRIC LAYERS ON THE MP PLATE
121222 READ(6,*) NSLAB(MP)
121223 NSS=NSLAB(MP)
121224 IF(NSS.GT.NLX) STOP
121225 WRITE(6,3391)
121226 3391 FORMAT(2H *,13X,’THICKNESS’,2X,’DIELECTRIC’,3X,’LOSS’,4X,
121227 2’PERMITIVITY’,3X,’LOSS’,779,1H+,
121228 32H *,6X,’LAYERS’,2X,’IN METERS’,3X,’CONSTANT’,3X,’TANGENT’,
121229 4X,’CONSTANT’,3X,’TANGENT’,779,1H+,
121230 62H *,6X,’--------’,2X,’--------’,2X,’--------’,2X,’--------’,
121231 62X,‘--------’,2X,’--------’,779,1H+)
121232 C$$$ DSLAB(NS,MP)=THICKNESS OF NS LAYER
121233 C$$$ RSLAB(NS,MP)=RELATIVE DIELECTRIC CONSTANT OF THE NS LAYER
121234 C$$$ TSLAB(NS,MP)=DIELECTRIC LOSS TANGENT OF THE NS LAYER
121235 C$$$ USLAB(NS,MP)=RELATIVE PERMEABILITY CONSTANT OF THE NS LAYER
164
C### TNSLAB(NS,MP)=PERMEABILITY LOSS TANGENT OF THE NS LAYER
1242 C###
1243 DO 3312 NS=1,NSS
1244 READ(*,*) DSLAB(NS,MP),EMLSLAB(NS,MP),TNSLAB(NS,MP),
1245 ZURSLAB(NS,MP),TNSLAB(NS,MP)
1246 DMLSLAB(NS,MP)=DSLAB(NS,MP)+UNITS
1247 3312 WRITE(*,3302) NS,DSLAB(NS,MP),EMLSLAB(NS,MP),TNSLAB(NS,MP),
1248 ZURSLAB(NS,MP),TNSLAB(NS,MP)
1249 3399 FORMAT(2H*.6X,13.4X,F9.4,2X,F10.4,2X,F7.4,2X,F11.4,2X,
1250 2F7.4,T70,1H*)
1251 WRITE(6,3006)
1252 WRITE(6,3006)
1253 WRITE(6,3006)
1254 3313 MEX=MEP(MP)
1255 IF(MEX.GT.MEDX) STOP
1256 DO 6 ME=1,MEX
1257 WRITE(6,3006)
1258 3006 FORMAT(2H*.LX,6X,1A6,4X,2H,.LX,FF*)
1259 WRITE(6,3006)
1260 GO TO 3000
1261 C###
1262 C### XX(N,ME,MP)=COMPONENTS OF CORNER #ME OF PLATE #MP.
1263 C### N=1(X),N=2(Y),N=3(Z). INPUT CORNER DATA AS FOLLOWS:
1264 C### 1.1.0
1265 C### 1.1.0
1266 C### 1.1.0
1267 C### 1.1.0
1268 C### THIS IS THE INPUT FOR A 1 METER SQUARE PLATE.
1269 C### NOTE THAT IF THERE IS MORE THAN ONE PLATE, THEN THE CORNER
1270 C### DATA FOR EACH PLATE WOULD FOLLOW SEQUENTIALLY.
1271 C###
1272 6 CONTINUE
1273 WRITE(6,3006)
1274 WRITE(6,3006)
1275 3302 FORMAT(2H*.6X,13.4X,F9.4,2X,F10.4,2X,F7.4,2X,F11.4,2X,
1276 4H*.6X,13.4X,F9.4,2X,F10.4,2X,F7.4,2X,F11.4,2X,
1277 2H*.6X,13.4X,F9.4,2X,F10.4,2X,F7.4,2X,F11.4,2X,
1278 2H*.6X,13.4X,F9.4,2X,F10.4,2X,F7.4,2X,F11.4,2X,
1279 2H*.6X,13.4X,F9.4,2X,F10.4,2X,F7.4,2X,F11.4,2X,
1280 2H*.6X,13.4X,F9.4,2X,F10.4,2X,F7.4,2X,F11.4,2X,
1281 2H*.6X,13.4X,F9.4,2X,F10.4,2X,F7.4,2X,F11.4,2X,
1282 2H*.6X,13.4X,F9.4,2X,F10.4,2X,F7.4,2X,F11.4,2X,
1283 2H*.6X,13.4X,F9.4,2X,F10.4,2X,F7.4,2X,F11.4,2X,
1284 2H*.6X,13.4X,F9.4,2X,F10.4,2X,F7.4,2X,F11.4,2X,
1285 2H*.6X,13.4X,F9.4,2X,F10.4,2X,F7.4,2X,F11.4,2X,
1286 2H*.6X,13.4X,F9.4,2X,F10.4,2X,F7.4,2X,F11.4,2X,
1287 2H*.6X,13.4X,F9.4,2X,F10.4,2X,F7.4,2X,F11.4,2X,
1288 2H*.6X,13.4X,F9.4,2X,F10.4,2X,F7.4,2X,F11.4,2X,
1289 2H*.6X,13.4X,F9.4,2X,F10.4,2X,F7.4,2X,F11.4,2X,
1290 2H*.6X,13.4X,F9.4,2X,F10.4,2X,F7.4,2X,F11.4,2X,
1291 2H*.6X,13.4X,F9.4,2X,F10.4,2X,F7.4,2X,F11.4,2X,
1292 2H*.6X,13.4X,F9.4,2X,F10.4,2X,F7.4,2X,F11.4,2X,
1293 2H*.6X,13.4X,F9.4,2X,F10.4,2X,F7.4,2X,F11.4,2X,
1294 2H*.6X,13.4X,F9.4,2X,F10.4,2X,F7.4,2X,F11.4,2X,
1295 2H*.6X,13.4X,F9.4,2X,F10.4,2X,F7.4,2X,F11.4,2X,
1296 2H*.6X,13.4X,F9.4,2X,F10.4,2X,F7.4,2X,F11.4,2X,
1297 2H*.6X,13.4X,F9.4,2X,F10.4,2X,F7.4,2X,F11.4,2X,
1298 2H*.6X,13.4X,F9.4,2X,F10.4,2X,F7.4,2X,F11.4,2X,
1299 2H*.6X,13.4X,F9.4,2X,F10.4,2X,F7.4,2X,F11.4,2X,
1300 2H*.6X,13.4X,F9.4,2X,F10.4,2X,F7.4,2X,F11.4,2X,
1301 2H*.6X,13.4X,F9.4,2X,F10.4,2X,F7.4,2X,F11.4,2X,
1302 2H*.6X,13.4X,F9.4,2X,F10.4,2X,F7.4,2X,F11.4,2X,
1303 C###

165
CONTINUE
C ===
SC:
CO.VULID
CSSt
C $$$
MSX=NVMSER
OF
ANTERNA
ELEM.E.
LSMP=.FALSE.
KSX-MSX*
MSXAT=MSAT+MSX
IF(MSXAT.GT.KSDX)
WXITE(0,904)
MSXAT
901 FORMAT('*****
XLWER
OF SOURCES=
',I3.'PROGRAM',
2' ABORTS SINCE MAX. SOURCE DIMENSION IS EXCEEDED. *****')
IF(b1SXAT.GT.EISDX)
5TOP
WRITE(6.3401)
M3X
3401 FORMAT(2H*
*,6X,'THIS IS SOURCE NO. ',I3,' IN THIS',
2' COMPUTATION.',T7P,lH*)
WRITE(0.3003)
bRITE(O,JOO6)
cat0
CjO$
XSS(N,m$:S)=XYZ
LOCATION OF
MS-TM
ANTENNA ELEMENT.
c:
0:
CS$$
IMS(MS)=TYPE
OF LINEAP.
ANTENliA
C$$$.LT.O
ELECTRIC LINEAR ELUENT
C$$$.GT.O:
MAGNETIC LINEAR
EL~NT
C$$$.ABS(IMS)=I
UIIIFOW
CURREIIT
DISTRIBUTION
C$$$.=2:
STA!:DA?.D
DIPOLE CURRENT DISTRIBUTION
C$$$.=3: CAVITY DACKED SLOT
CURRENT DISTRIBUTION (TE01)
C$$$.=3:
	HAYS(blS)=APEXTUF
WIDTH IN WAVELENGTHS (NOTE: IF
C$$$.HAIS(MS)
IS LESS THAN
.l
LAbQDA,
SOURCE IS
C$)$
CONSIDERED
TO
DE
DIPOLE
SOLRCE
C$$$.MS(bIS)=LENGTH
OF LINEAR ELUENT IN WAVELENGTHS
C$$$.THSZ.PBSZ=ORIEllTATION
ANGLES
USED TO DEFINE LINEAR
C$$$.THSX.PHSX=ORIEllTATION
ANGLES
USED
TO
DEFINE APERTURE
PLANE
OR DIPOLE X-AXIS.
C$$$.THSZ.PHSX.THSX.PHSX
READ(6,*)
THSZ.PHSX
READ(6,*)
IMS(MS)
READ(6,*)
WIS.WS
IF(IPlS(bIS).LT.O)
WRITE(6,3411)
It.lS(NS)
3411 FOR.lAT(2H*
*,6X,'THIS IS AN ELECTRIC SOURCE OF TYPE ',I3,T79.1H*)
IF(IMS(MS).GE.O)
WRITE(6,3412)
INS(MS)
3412 FORMAT(2H*
*,6X,'THIS IS A MAGNETIC SOURCE OF TYPE ',I3,T79,1H*)
WRITE(6,3000)
IF(IUl1ST.EQ.O)
GO TO 3114
ulisTs=ulrrT
(IUl1ST)
WRITE(0,3113)
HS(KS)
,HAWS(NS)
,LABEL(IUl1ST)
3413 FOR.lAT(2H*
*,6X,'S0UF.CE
LE:iCTH='
,F10.6,'AID WIDTH='
,F10.6.1X.AO.T7D.iH*)
HS(h!S)=WJSTS*UliITF*HS(bIS)
HAWS
(MS)=UlISTS*UllITF*HAWS(MS)
IF(IU1IST.NE.
1)
YRITE(6.3000)
IF(1UIiST.
liE.
1)
KRITE(~
,3413)
HS(hIS)
,HAWS(MS)
GO TO 3414
3414 IBRITE(6.3416)
HS(MS)
,HAIS(MS)
WRITE(6,3413)
HS(NS).HAWS(NS).LABEL(IU1ST)
3413 FORMAT(2H*
*,6X,'SOURCE LENGTH='
,F10.6,' AND WIDTH='
,F10.6.1X.46,T79,1H*)
HS(NS)=U1STS=U1ITF*HS(NS)
HAWS
(NS)=U1STS=U1ITF*HAWS(NS)
IF(IU1ST.NE.1) WRITE(6,3006)
3006 IF(IU1ST.NE.1) WRITE(6,3413)
HS(NS).HAWS(NS).LABEL(1)
GO TO 3416
3414 WRITE(6,3416)
HS(NS).HAWS(NS)
166
3415 FORMAT(2H *,6X,'SOURCE LENGTH=',F10.6,' AND WIDTH='
3416 2,F10.6,' WAVELENGTHS',T79,1H+)
3417 WRITE(6,3006)
3418 WS(MS)=WS+CEIL(CI+WP*RPD)
3419 WRITE(6,3417) WS,WP
3417 FORMAT(2H *,6X,'THE SOURCE WEIGHT HAS MAGNITUDE='
3414 2,F10.6,' AND PHASE=',F10.6,T79,1H+)
3416 WRITE(6,3006)
3417 WRITE(6,3006)
3417 WRITE(6,3421) LABEL(IUNIT)
3421 FORMAT(2H *,16,'SOURCE=',I7,' INPUT LOCATION IN ',A6,T46,
3420 2,' ACTUAL LOCATION IN METERS',T79,1H+)
3420 WRITE(6,3422)
3422 FORMAT(2H *,16,' ',T46,27(' '),T46,27(' ')
3423 2,T79,1H+)
3423 WRITE(6,3006)
3424 DO 3424 N=1,3
3424 FORMAT(2H *,6X,'THE FOLLOWING SOURCE ALIGNMENT IS USED:'
3425 2,T79,1H*)
3426 DO 3426 N=1,3
3427 FORMAT(2H *,6X,'SOURCE=',I7,' INPUT LOCATION IN ',A6,T46,
3426 2,' ACTUAL LOCATION IN METERS',T79,1H+)
3425 WRITE(6,3422)
3428 FORMAT(2H *,16,' ',T46,27(' '),T46,27(' ')
3429 2,T79,1H+)
3429 WRITE(6,3006)
3430 DO 3430 N=1,3
3430 FORMAT(2H *,6X,'THE FOLLOWING SOURCE ALIGNMENT IS USED:'
3431 2,T79,1H*)
3432 FORMAT(2H *,6X,'SOURCE=',I7,' INPUT LOCATION IN ',A6,T46,
3431 2,' ACTUAL LOCATION IN METERS',T79,1H+)
3430 WRITE(6,3422)
3433 FORMAT(2H *,16,' ',T46,27(' '),T46,27(' ')
3434 2,T79,1H+)
3434 WRITE(6,3006)
FOUlAT(2H 3(9X,'VASS(',II,':',',,II,':',',12,')=',F9.6)

CO TO 3000

C==PI==

CONTINUE

C==

CO

MSW=MSW+1

MSX=MSX+1

WRITE(O,3806) bIS,FISW

FOUIAT(2H 3(9X,'THIS IS SOURCE NO. ',II,': IN THIS',

MSX=MSX+1

F!SAT=FISAT+~.ISAX

MSXAT=MSAT+MSX

IF(MSXAT.CT.bISX)

STOP

WRITE(O,3306) IMS(MS)

XSS(IW,HA)=XYZ LOCATION OF MA-TH ANTENNA ELEMENT.

XSS(IW,MS)=XYZ LOCATION OF MS-TH WEIGHTED CENTER OF THE ARRAY GROUPING.

THE ARRAY ELEMENTS ARE ASSUMED TO HAVE THE SAME LENGTH,

WIDTH, AND ORIENTATION. ALSO, THEY ARE ASSUMED TO BE EITHER ALL MOUNTED OR ALL OFF A PLATE.

IMS(MS)=TYPE OF LINEAR ANTENNA

.LT.0: ELECTRIC LINEAR ELEMENT

.GT.0: MAGNETIC LINEAR ELEMENT

ABS(IW)=1: UNIFORM CURRENT DISTRIBUTION

=2: STANDARD DIPOLE CURRENT DISTRIBUTION

=3: CAVITY BACKED SLOT CURRENT DISTRIBUTION (TE01)

HWS(MS)=APERTURE WIDTH IN WAVELENGTHS (NOTE: IF HWS(MS) IS LESS THAN .1 LAMBDA, SOURCE IS CONSIDERED TO BE DIPOLE SOURCE

HS(MS)=LENGTH OF LINEAR ELEMENT IN WAVELENGTHS

THSZ,PHSZ=ORIENTATION ANGLES USED TO DEFINE LINEAR ELEMENT AXIS.

THSZ,PHSZ=ORIENTATION ANGLES USED TO DEFINE APERTURE PLANE OR DIPOLE X-AXIS.

HMS,WPS=MAGNITUDE AND PHASE OF EXCITATION OF MA-TH ELEMENT.

MS=MSI

MAI=MSAI+1

MAF=MAI+MSAX-1

MSA(1,MS)=MAI

MSA(2,MS)=MAF

DO 3841 MA=MAI,MAF

READ(6,*) (XSS(M,MA),M=1,3)

READ(6,*) THSZ,PHSZ,THSX,PHSX

READ(5,*) IMS(M),HS(M),HWS(M)

IF(IMS(M).LT.0) WRITE(6,3411) IMS(M)
IR(3412) WRITE(6,3412) IMS(MS)
WRITE(6,3006)
IF(IUNST.EQ.0) GO TO 3814
UNSTS=UNIT(IUNST)
WRITE(6,3413) HS(MS),HAWS(MS),LABEL(IUNST)
3814 WRITE(6,3415) HS(MS),HAWS(MS)
WRITE(6,3006)
WS(MS)=(1.,0.)
WSA=0.
ISAI=0.
ISAT=0.
DD 3843 MA=MAI,MAF
READ(6,*) WMS,WPS
WRITE(6,3417) MA,WMS,WPS
3817 FORMAT(2H,*,' SOURCE ',13,' HAS MAGNITUDE='
2,F10.6,' AND PHASE=',F10.6, 'H=')
WS(MA)=WPS*CEXP(CJ*YPS*RPD)
WMSA=WMSA+WS
ISAI=ISAI+WS+ISS(1,MA)
ISAT=ISAT+WS+ISS(2,MA)
XSA=WMSA+WS
ISS(1,MS)=XSA/WMS
ISS(2,MS)=XSA/WMS
ISS(3,MS)=XSA/WMS
WRITE(6,3008)
WRITE(6,3421) LABEL(IUNIT)
WRITE(6,3422)
WRITE(6,3008)
DO 3824 N=1,3
IQ(N)=ISS(N,MS)
DO 3825 N=1,3
ISS(N,MS)=UNITS*(IQ(1)*VRT(1,N)+IQ(2)*VRT(2,N)
+IQ(3)*VRT(3,N))+TR(N)
WRITE(6,3426) MS,(IQ(N),N=1,3),(ISS(N,MS),N=1,3)
DO 3829 MA=MAI,MAF
DO 3827 N=1,3
DO 3828 N=1,3
ISS(N,MA)=UNITS*(IQ(1)*VRT(1,N)+IQ(2)*VRT(2,N)
+IQ(3)*VRT(3,N))+TR(N)
WRITE(6,3426) MA,(IQ(N),N=1,3),(ISS(N,MA),N=1,3)
DO 3831 N=1,3
IQ(1)=SIN(TQR)+COS(PQR)
IQ(2)=SIN(TQR)+SIN(PQR)
IQ(3)=COS(TQR)
DO 3831 N=1,3
DZX=VXSS(i,i,MS)*VXSS(3,1,MS)*VXSS(i,n,MS)*VXSS(3,1,MS)
+VXSS(i,3,MS)*VXSS(3,3,MS)
3831 VXSS(i,i,MS)=IQ(1)*VRT(1,N)+IQ(2)*VRT(2,N)+IQ(3)*VRT(3,N)
3831 VXSS(i,i,MS)=IQ(1)+VRT(1,N)+VR(2,N)+VR(3,N)
3831 VXSS(i,i,MS)=IQ(1)+VR(1,N)+VR(2,N)+VR(3,N)
3831 VXSS(i,i,MS)=IQ(1)+VR(1,N)+VR(2,N)+VR(3,N)
3831 VXSS(i,i,MS)=IQ(1)+VR(1,N)+VR(2,N)+VR(3,N)
3831 VXSS(i,i,MS)=IQ(1)+VR(1,N)+VR(2,N)+VR(3,N)
3832 VXSS(i,i,MS)=IQ(1)+VR(1,N)+VR(2,N)+VR(3,N)
3832 VXSS(i,i,MS)=IQ(1)+VR(1,N)+VR(2,N)+VR(3,N)
3832 VXSS(i,i,MS)=IQ(1)+VR(1,N)+VR(2,N)+VR(3,N)
3832 VXSS(i,i,MS)=IQ(1)+VR(1,N)+VR(2,N)+VR(3,N)
IF(ABS(DZX).GT.0.1) WRITE(6,3436)
1661   IF(ABS(DZX).GT.0.1) STOP
1662   VISS(1,1,MS)=VISS(1,1,MS)-VISS(3,1,MS)*DZX
1663   VISS(1,2,MS)=VISS(1,2,MS)-VISS(3,2,MS)*DZX
1664   VISS(1,3,MS)=VISS(1,3,MS)-VISS(3,3,MS)*DZX
1665   DOT=VISS(1,1,MS)*VISS(1,1,MS)+VISS(1,2,MS)*VISS(1,2,MS)
1666   2*VISS(1,3,MS)=VISS(1,3,MS)
1667   DOT=SQRT(DOT)
1668   VXSS(l,l,b!S)=VXSS(l,l,VS)-VXSS(3,l,KS)*DZX
1669   VXSS(~,~.b!S)QVXSS(~,~,MS)-VXSS(~,~,F!S)*DZX
1670   VXSS(l,3,MS)=VXSS(l,3,MS)-VXSS(3,3,MS)*DZX
1671   VXSS(l,1,MS)=VXSS(l,1,MS)/DOT
1672   VXSS(l,2,MS)=VXSS(l,2,MS)/DOT
1673   VXSS(l,3,MS)=VXSS(l,3,MS)/DOT
1674   WRITE(6,3000)
1675   WRITE(6,3000)
1676   WRITE(6,3000)
1677   DO 3033 NI=1,3
1678   WRITE(6,3000)
1679   3033 WRITE(6,3434) (NI,NJ,MS,VISS(NI,NJ,MS),NJ=1,3)
1680   GO TO 3000
1681   C======
1682   3440 CONTINUE
1683   C== PR: COMMAND =====
1684   C$$
1685   C$$ IPRAD= 1 =NORMALIZATION FOR FAR ZONE AS FOLLOWS
1686   C$$
1687   C$$ PRAD=TOTAL POWER RADIATED IN WATTS.
1688   C$$
1689   C$$ PRAD CAN ALSO BE SPECIFIED AS THE POWER INPUT IN WATTS.
1690   C$$
1691   C$$ NOTE IF PRAD IS LESS THAN OR EQUAL TO 1.E-30
1692   C$$ THAN LPRAD WILL BE SET FALSE
1693   C$$
1694   C$$ LPRAD= TRUE
1695   READ(6,*) IPRAD
1696   IF(IPRAD.GT.4) STOP
1697   GO TO (3444,3446,3446,3447),IPRAD
1698   3444 READ(6,*) PRAD
1699   IF(PRAD.LT.1.1E-30) GO TO 3442
1700   WRITE(6,3441) PRAD
1701   3441 FORMAT(2H TOTAL POWER RADIATED IN WATTS= ',E12.6
1702   2,779,1H+)
1703   GO TO 3000
1704   3442 CONTINUE
1705   LPRAD= FALSE.
1706   PRAD=0.
1707   WRITE(6,3443)
1708   3443 FORMAT(2H NO POWER RADIATED IS SPECIFIED',779,1H*)
1709   GO TO 3000
1710   3445 CONTINUE
1711   C$$
1712   C$$ IPRAD = 2 =MUTUAL IMPEDANCE CALCULATION Z12 = 221
1713   C$$
1714   C$$ CI11 = SOURCE TERMINAL CURRENT (REAL AND IMAGINARY)
1715   C$$
1716   C$$ CI12 = RECEIVER TERMINAL CURRENT (REAL AND IMAGINARY)
1717   C$$
1718   C$$
1719   C$$ Z12 = mutual impedance between source and receiver terminal current
1720   C$$
1721   READ(6,*) CI11,CI22
1722   WRITE(6,3446) CI11,CI22
1723   4446 FORMAT(2H SOURCE TERMINAL CURRENT= ',2E12.6,779,1H*/
1724   2,2H +.6X,'RECEIVER TERMINAL CURRENT= ',2E12.6,779,1H*)
1725   GO TO 3000
1726   3448 CONTINUE
1727   C$$
1728   C$$
1624 C$$$ IPRAD = 3 = COUPLING VIA THE REACTION THEORY
1625 C$$$ THIS GIVES A MODIFIED FRIT'S TRANSMISSION TYPE RESULT
1626 C$$
1627 C$$ PRAD = POWER RADIATED BY THE SOURCE
1628 C$$ PRADR = POWER RADIATED BY THE RECEIVER AS IF IT WERE A SOURCE
1629 C$$
1630 C$$ READ(6,*) PRAD, PRADR
1631寫出(6,4447) PRAD, PRADR
1632 4447 \text{FORMAT}(2*6X,:' SOURCE POWER RADIATED'= ',E12.6,779,1H*)
1633 2,2H *,6X,:' RECEIVER POWER RADIATED'= ',E12.6,779,1H*)
1634 GO TO 3000
1635 3447 CONTINUE
1636 C$$
1637 C$$ IPRAD = 4 = COUPLING BY THE LINVILLE METHOD
1638 C$$
1639 C$$ C11 = SOURCE TERMINAL CURRENT (REAL AND IMAGINARY)
1640 C$$ C122 = RECEIVER TERMINAL CURRENT (REAL AND IMAGINARY)
1641 C$$
1642 C$$ Z11 = SOURCE TERMINAL IMPEDANCE (REAL AND IMAGINARY)
1643 C$$ Z22 = RECEIVER TERMINAL IMPEDANCE (REAL AND IMAGINARY)
1644 C$$
1645 C$$ READ(6,*) C111, C112
1646 C$$ WRITE(6,4445) C111, C112
1647 C$$ READ(6,*) Z11, Z22
1648 C$$ WRITE(6,4446) Z11, Z22
1649 4446 \text{FORMAT}(2*6X,:' SOURCE TERMINAL IMPEDANCE= ',2E12.6,779,1H*)
1650 2,2H *,6X,:' RECEIVER TERMINAL IMPEDANCE= ',2E12.6,779,1H*)
1651 GO TO 3000
1652 C===
1653 4400 CONTINUE
1654 C=== RG: COMMAND
1655 C$$
1656 C$$ MAX = NUMBER OF ANTENNA ELEMENTS.
1657 C$$
1658 C$$ LACVR = TRUE.
1659 LRMF = FALSE.
1660 MXI = MXI + 1
1661 MXAT = MAX + MXI
1662 IF(MXAT .GT. MAX) WRITE(6,4404) MAXAT
1663 4404 \text{FORMAT}(' **** NUMBER OF RECEIVERS= ',I3, ' PROGRAM',
1664 2' ABORTS SINCE MAX. RECEIVER DIMENSION IS EXCEEDED. *****')
1665 IF(MAXAT .GT. MAX) STOP
1666 WRITE(6,4401) MAX
1667 4401 \text{FORMAT}(2*6X,:' THIS IS RECEIVER NO. ',I3, ' IN THIS',
1668 2' COMPUTATION ',T79,1H*)
1669 WRITE(6,3006)
1670 WRITE(6,3006)
1671 C$$
1672 C$$ XRR(U,MR) = XYZ LOCATION OF MR-TH ANTENNA ELEMENT.
1673 C$$
1674 C$$ IMR(MR) = TYPE OF LINEAR ANTENNA
1675 C$$
1676 C$$ LT.O: ELECTRIC LINEAR ELEMENT
1677 C$$
1678 C$$ CT.O: MAGNETIC LINEAR ELEMENT
1679 C$$
1680 C$$ ABSC(IMR)=1: UNIFORM CURRENT DISTRIBUTION
1681 C$$
1682 C$$ =2: STANDARD DIPOLE CURRENT DISTRIBUTION
1683 C$$
1684 C$$ =3: CAVITY BACKED SLOT CURRENT DISTRIBUTION (TE01)
1685 C$$
1686 C$$ HAWR(MR) = APERTURE WIDTH IN WAVELENGTHS (NOTE: IF
1687 C$$ HAWR(MR) IS LESS THAN .1 LAMBDA, RECEIVER IS
1688 C$$ CONSIDERED TO BE DIPOLE RECEIVER
1689 C$$ HR(MR) = LENGTH OF LINEAR ELEMENT IN WAVELENGTHS
1690 C$$
1691 C$$ THXZ, PHXZ = ORIENTATION ANGLES USED TO DEFINE LINEAR
1692 
1693 C$$ ELEMENT AXIS.
171
C488  THRZ, PHRZ = ORIENTATION ANGLES USED TO DEFINE APERTURE
C489  PLANE OR DIPOLE X-AXIS.
C490  AAGLES USED TO DEFINE APERTURE PLANE OR DIPOLE X-AXIS.
C491  C: $$ YI = R = W = MAGNITUDE AND PHASE OF EXCITATION OF
C492  KR-THE FIELD.
C493  MRA = O
C494  MRA(2, N) = O
C495  READ(6, *) (XRR(N, MR), N=1, 3)
C496  READ(6, *) THRZ, PHRZ, THR, PHR
C497  READ(6, *) IMR(MR), HR(NR), HMR(NR)
C498  READ(6, *) WMR, WPR
C499  IF(IMR(MR). LT. 0) WRITE(6, 4411) IMR(MR)
C500  4411 FORMAT(2H *), 6X, 'THIS IS AN ELECTRIC RECEIVER OF TYPE ', I3
C501  2, 'T79, 1H')
C502  IF(IMR(MR). GE. 0) WRITE(6, 4412) IMR(MR)
C503  4412 FORMAT(2H *), 6X, 'THIS IS A MAGNETIC RECEIVER OF TYPE ', I3
C504  2, 'T79, 1H')
C505  WRITE(6, 3006)
C506  IF(IINST.EQ.0) GO TO 4414
C507  IINST=UNIT(IINST)
C508  4413 FORMAT(2H *), 6X, 'RECEIVER LENGTH=', 'F10.6,' AND WIDTH='
C509  2, 'F10.6,' AND WIDTH='
C510  HR(MR)=IINST*UNIT*HR(03)
C511  HMR(MR)=IINST*UNIT*HMR
C512  IF(IINST.NE.1) WRITE(6, 3006)
C513  IF(IINST.NE.1) WRITE(6, 4413) HR(MR), HMR, LABEL(IINST)
C514  GO TO 4416
C515  4414 WRITE(6, 4415) HR(03), HMR
C516  4415 FORMAT(2H *), 6X, 'RECEIVER LENGTH=', 'F10.6,' AND WIDTH='
C517  2, 'F10.6,' AND WIDTH='
C518  WRITE(6, 3006)
C519  WRITE(6, 4416) HR(03), HMR
C520  4416 FORMAT(2H *), 6X, 'RECEIVER WEIGHT HAS MAGNITUDE='
C521  2, 'F10.6,' AND PHASE='
C522  WRITE(6, 3006)
C523  WRITE(6, 4417) WSR, WPR
C524  4417 FORMAT(2H *), 6X, 'THE RECEIVER WEIGHT HAS MAGNITUDE='
C525  2, 'F10.6,' AND PHASE='
C526  WRITE(6, 3006)
C527  WRITE(6, 4418)
C528  4418 FORMAT(2H *), 6X, 'THE RECEIVER LOCATION IN METERS=', 'F10.6,'
C529  2, 'ACTUAL LOCATION IN METERS=', 'F10.6,'
C530  WRITE(6, 3006)
C531  WRITE(6, 4422)
C532  4422 FORMAT(2H *), 6X, 'THE RECEIVER LOCATION IN METERS=', 'F10.6,'
C533  2, 'T79, 1H')
C534  WRITE(6, 3006)
C535  DO 4424 N=1, 3
C536  4424 FORMAT(2H *), 6X, 'THE RECEIVER LOCATION IN METERS=', 'F10.6,'
C537  2, 'T79, 1H')
C538  WRITE(6, 3006)
C539  DO 4425 N=1, 3
C540  4425 FORMAT(2H *), 6X, 'THE RECEIVER LOCATION IN METERS=', 'F10.6,'
C541  2, 'T79, 1H')
C542  WRITE(6, 3006)
C543  WRITE(6, 4426) N=1, 3
C544  4426 FORMAT(2H *), 6X, 'THE RECEIVER LOCATION IN METERS=', 'F10.6,'
C545  2, 'T79, 1H')
C546  WRITE(6, 3006)
C547  WRITE(6, 4427)
C548  4427 FORMAT(2H *), 6X, 'THE RECEIVER LOCATION IN METERS=', 'F10.6,'
C549  2, 'T79, 1H')
C550  WRITE(6, 3006)
C551  WRITE(6, 4428)
C552  4428 FORMAT(2H *), 6X, 'THE RECEIVER LOCATION IN METERS=', 'F10.6,'
C553  2, 'T79, 1H')
C554  WRITE(6, 3006)
C555  WRITE(6, 4429)
C556  4429 FORMAT(2H *), 6X, 'THE RECEIVER LOCATION IN METERS=', 'F10.6,'
C557  2, 'T79, 1H')
C558  WRITE(6, 3006)
C559  WRITE(6, 4430)
C560  4430 FORMAT(2H *), 6X, 'THE RECEIVER LOCATION IN METERS=', 'F10.6,'
C561  2, 'T79, 1H')
C562  WRITE(6, 3006)
C563  WRITE(6, 4431)
C564  4431 FORMAT(2H *), 6X, 'THE RECEIVER LOCATION IN METERS=', 'F10.6,'
PQR = PHRX*RPD

Xp(i) = SIN(TpR)*COS(PqR)

Xq(l) = SIN(TqR)*SIN(PqR)

\[ xq(3) = \cos(q) \]

DO 4431

N1 = 3

4431 VXRR(I,N,~) = XQ(~)*VRT(~,N)+XQ(~)*VRT(~,~)+I~(~)*VRT(~,~)

DZX = VXRR(I,~)~*VXRR(~,~,MR)+VIBR(~.,~)~*VIBR(~.,~,HB)

1+VXRR(1,3,MR)*VXRR(3,3,MR)

IF(ABS(DZX) .GT. 0.1) WRITE(6,4438)

IF(ABS(DZX) .GT. 0.1) STOP

VIRR(I,I,~) = VXRR(~,~,~) - VXBR(~.,MR)*DZX

VXRR(1,1,MR) = VXRR(1,1,~) - VIRR(3,1,MR)*DZX

VXRR(~,~,~) = VXRR(~,~,~) - VIRR(~,~,~)*DZX

DOT = VXRR(~,~,~)*VXRR(~,~,~) + VXRR(~,~,~)*VIRR(~,~,~)

2+VXRR(1,3,MR)*VXRR(3,3,MR)

DOT = SPRT(DOT)

VXRR(I,1,NR) = VXRR(I,1,MR)/DOT

VXRR(I,2,MR) = VXRR(1,2,MR)/DOT

VIRR(~,~,~) = VXRR(I,~,MR)/DOT

VXRR(~,~,~) = VXRR(~,~,~) - VXRR(~,~,~)*VXBR(~,~,~)

VXRR(~,~,~) = VXRR(~,~,~) - VXRR(~,~,~)*VXRR(I,~,MR)

WRITE(8,3008)

WRITE(8,3008)

WRITE(8,4437)

4437 FORMAT(2H *, 6X, 'THE FOLLOWING RECEIVER ALIGNMENT IS USED:

2. T70,1H *)

DO 1433 NR = 1, 3

WRITE(6,3006)

1433 WRITE(6,4434) (NR,MR,VXRR(NR,MR,MR),MR=1,3)

4434 FORMAT(2H *, 11, 3(2X, 'VXRR(',11,',',11,',',11,')=',F9.6)

2,179, IH *)

GO TO 3000

C======

4810 CONTINUE

C== RA: COMMAND ======

C### MAX=NUMBER OF ANTENNA ARRAY GROUPINGS.

C### MRA=NUMBER OF ELEMENTS PER GROUPING.

C### THE ARRAY ELEMENTS ARE ASSUMED TO HAVE THE SAME LENGTH.
C$$
WIDTH, AND ORIENTATION. ALSO, THEY ARE ASSUMED TO BE
EITHER ALL MOUNTED AND OR ALL OFF A PLATE.
C$$
INKR(MR)=TYPE OF LINEAR ANTENNA
C$$
.LT. 0: ELECTRIC LINEAR ELEMENT
C$$
.GT. 0: MAGNETIC LINEAR ELEMENT
C$$
ABS(IMR)=1: UNIFORM CURRENT DISTRIBUTION
C$$
=2: STANDARD DIPOLE CURRENT DISTRIBUTION
C$$
=3: CAVITY BACKED SLOT CURRENT DISTRIBUTION (I904)
C$$
C$$
HAWR(MR)=APERTURE WIDTH IN WAVELENGTHS (NOTE: IF
HAWR(MR) IS LESS THAN .1 LAMBDA, RECEIVER IS
CONSIDERED TO BE DIPOLE RECEIVER
HR(MR)=LENGTH OF LINEAR ELEMENT IN WAVELENGTHS
C$$
C$$
THRX,PHRX=ORIENTATION ANGLES USED TO DEFINE LINEAR
ELEMENT AXIS.
C$$
C$$
THRZ,PHRZ=ORIENTATION ANGLES USED TO DEFINE APERTURE
PLANE OR DIPOLE X-AXIS.
C$$
WIDTH, AND ORIENTATION ANGLES USED TO DEFINE LINEAR
MA-TH ELEMENT.
C$$
MR=KML
MA=MA+MR*XRAT+1
MAP=MA+MRA+1
MRA(1,MRA)=MA
MRA(2,MRA)=MAF
DO 4841 MA=MA,MAF
4841 READ(6,*)(XRM(MA),MA=1,3)
DO 4846 READ(6,*)(THRX,PHRX,THRZ,PHRZ)
4846 READ(6,*)(IMR(MR),HR(MR),HAWR(MR))
IF(IMR(MR).LT.0) WRITE(6,4411) IMR(MR)
IF(IMR(MR).GE.0) WRITE(6,4412) IMR(MR)
WRITE(6,3006)
IF(IUUST.EQ.0) GO TO 4814
UNSTS=UNIT(IUUST)
WRITE(6,4413) HR(MR),HAWR(MR),LABEL(IUUST)
HR(MR)=UNSTS*UNITF*HR(MR)
HAWR(MR)=UNSTS*UNITF*HAWR(MR)
WRITE(6,3006)
IF(IUUST.NE.0) WRITE(6,4413) HR(MR),HAWR(MR),LABEL(1)
4814 WRITE(6,4415) HR(MR),HAWR(MR)
4816 WRITE(6,3006) WR(MR)=(1.,0.)
4817 FORMAT(2H=,'RECEIVER ',3H HAS MAGNITUDE=
2,F9.6,' AND PHASE=',F10.6,T9.9,1H+)
WR(MR)=WR+CEAP(CJ+VPA+RPD)
WR=WR+WR+VPA
XRAX=XRAX*XR+XRR(1,NA)
XRAT=XRAT*WR*XR(2,MA)
XR2=XR2*XR2+XRR(3,MA)
XRR(1,MR)=XRAX/WR
XRR(2,MR)=XRAT/WR
XRR(3,MR)=XR2/WR
WRITE(6,3006)
WRITE(6,3006)
WRITE(6,4421) LABEL(UNIT)
WRITE(6,4422)
WRITE(6,3006)
DO 4824 N=1,3
4824 IQ(N)=IR(N,MR)
DO 4826 N=1,3
4826 XRR(N,MR)=UNITS*(XQ(N)*VRT(N,N)+XQ(N)*VRT(N,N)+XQ(N)*YRT(N,N))*TR(N)
WRITE(6,4426) NR, (XQ(N),N=1,3), (IRR(N,MR),N=1,3)
DO 4829 MA=MAI,MAF
4827 XQ(N)=XBR(N,MA)
4828 XRR(N,MA)=UNITS*(XQ(N)*VBT(N,N)*XQ(N)*VRT(N,N))*IR(N)
YRITE(6,4428) MA, (XQ(N),N=1,3), (XRR(N,MA),N=1,3)
TQR=THRZ*RPD
PQR=PHRZ*RPD
XQ(1)=SIN(TQ%)*COS(PQR)
XQ(2)=SIN(TQR)*COS(PQR)
XQ(3)=COS(TQR)
DO 4831 N=1,3
4831 VXRR(NI,NJ,MR)+XQ(NJ)*YRT(NJ,N)+XQ(N)*VRT(N)*XU(N)*VRT(N)
IF(ABS(DZX).GT.1) STOP
VXRR(NI,NJ,MR)=VXRR(NI,NJ,MR)-VXRR(NI,NJ,MR)*DZX
VXRR(NI,NJ,MR)=VXRR(NI,NJ,MR)-VXRR(NI,NJ,MR)*DZX
VXRR(NI,NJ,MR)=VXRR(NI,NJ,MR)-VXRR(NI,NJ,MR)*DZX
3460 CONTINUE
C====
3460 CONTINUE
C====
3460 CONTINUE
C====
3460 CONTINUE
C====
3460 CONTINUE
C====
3460 CONTINUE
C====
3460 CONTINUE
C====
C$$$
WX=NUMBER
of
ANTENNA
SEGMENTS
ctt:
LSMx=
TRUE.
READ(6,*)
MSX
IF(MSX.GT.MSDX)
WRITE(6,3477)
MSX
3477
FORMAT(****>
NUMBER
of
SEGMENTS=
', I3,
2' PROGRAM
ABORTS
SINCE
MAX.
SOURCE
DIMENSION'
3,' IS EXCEEDED.
*****)
IF(MSX.GT.MSDX)
STOP
WRITE(6,3451)
MSX
3451
FORMAT(2H*,6X,'THERE
ARE
',I3,' SEGMENTS
IN
THIS',
2'
COMPUTATION',',199,1H+))
WRITE(6,3006)
WRITE(6,3006)
CS$$
XS(MS,N)=XYZ
LOCATION
of
MS-TH
ANTENNA
SEGMENT
CS$$
IS(MS)=1=ELECTRIC
LINEAR
ELEMENT
WITH
A
UNIFORM
DISTRIBUTION
CS$$
HS(MS)=LENGTH
OF
LINEAR
ELEMENT
CS$$
THSZ,PHSZ=ORIENTATION
ANGLES
USED
TO
DEFINE
LINEAR
ELEMENT
AXIS.
CS$$
WX,PSZ=REAL
AND
IMAGINARY
CURRENT
WEIGHT.
CS$$
WRITE(6,3458)
LABEL(IU111)
3458
FORMAT(2H*,77, 'MS',T13, 'HS',',A6,T23, 'HS-METERS',
2T41, 'INPUT:
THS,PHS',T6O, 'ACTUAL:
THS,PHS',T79,1H+)
WRITE(6,3459)
3459
FORMAT(2H*,76,3('-'),T12,20('-'),T40,16('-'),T69,
217('-'),T79,1H+)
WRITE(6,3006)
WRITE(6,3006)
DO 3463
MS=I,MSX
READ(6,*)
(XSS(N,M),N=1,3),HS(MS),THSZ,PHSZ
NSA(1,MS)=0
NSA(2,MS)=0
NSA(3,MS)=0
HSZ=HS(MS)
ThSZ=ORIGIN
AL
ANGLES
TO
DEFINE
LINEAR
ELEMENT
AXIS.
WRITE(6,3006)
WRITE(6,3006)
DO 3481 N=1,3
3481
FORMAT(2H*,76,3('-'),T12,20('-'),T40,16('-'),T69,
217('-'),T79,1H+)
WRITE(6,3006)
WRITE(6,3006)
DO 3481 N=1,3
3481
VXSS(N,MS)=XQR(N)
XQR(N)=IQ(1)+VRI(1,H)+VRI(2,H)+VRI(3,H)
IQ(2)=2D(R)+IQ(1)
VRI(3)=VRI(2)+VRI(1)
DO 3481 N=1,3
3481
VXSS(3,N,MS)=XQR(N)
VXSS(1,1,MS)=COS(THSZ)+PHS
VXSS(1,2,MS)=COS(THSZ)+PI(DR)
VXSS(1,3,MS)=SIN(THSZ)+PI(DR)
VXSS(2,1,MS)=SIN(THSZ)+PHS
VXSS(2,2,MS)=COS(THSZ)+PI(DR)
VXSS(2,3,MS)=0.
CONTINUE
WRITE(6,3006)
176
WRITE (6, 3006)
WRITE (6, 3008)
WRITE (6, 3464)
3464 FORMAT (2H ' ', T31, ' SEGMENT COORDINATES ', T70, ' *')
WRITE (6, 3006)
WRITE (6, 3008)
WRITE (6, 3464)
WRITE (6, 3006) LABEL (UNIT)
3466 FORMAT (2H ' ', T7, ' NS', T14, ' INPUT LOCATION IN ', A6, 
' ACTUAL LOCATION IN METERS ', T70, ' *')
WRITE (6, 3457)
WRITE (6, 3006)
3457 FORMAT (2H ', T6, 3 ('' ') ', T13, 26 ('' ') ', T42, 27 ('' ') ', T70, ' *')
WRITE (6, 3006)
DO 3473 MS = 1, HSX
DO 3474 N = 1, 3
XQ (N) = XSS (N, MS)
DO 3476 N = 1, 3
XSS (N, MS) = U%IIS * (XQ (1) * VRT (1, N) * XQ (1)) * VRT (1, N) * TR (N)
WRITE (6, 3470)
MS, (XQ (N) = 1, MS), (XSS (N, MS) = 1, 3)
WRITE (6, 3478) MS, (XQ (N), N = 1, 3), (ISS (N, MS), N = 1, 3)
3476 FORMAT (2H ', T6, 13, T13, F8.3, 2 ('' ') ', F8.3, T42, F8.3, 
F8.3)
3478 FORMAT (2H ', F8.3, T70, 1H *)
CONTINUE
3473 CONTINUE
READ (6, *)
WRITE (6, 3008)
WRITE (6, 3465)
3465 FORMAT (2H ' ', T33, ' CURRENT WEIGHTS ', T70, 1H *, /2H ' ', T7, ' NS ', T18, 
' REAL', T31, ' MAG ', T46, ' PHASE ', T70, 1H *)
WRITE (6, 3466)
3466 FORMAT (2H ', T6, 3 ('' ') ', T17, 6 ('' ') ', T30, 7 ('' ') ', T46, 6 ('' ') , 
T168, 7 ('' ') ', T70, 1H *)
DO 3467 MS = 1, MSX
READ (6, *) WSS, WPS
DO 3468 MS = COMPLI (WSS, WPS)
DO 3469 MS = WSS, WPS, WPS, WPS
WRITE (6, 3460) MS, WSS, WPS, WPS, WPS
WRITE (6, 3460) MS, WSS, WPS, WPS, WPS
WRITE (6, 3008)
3466 CONTINUE
WRITE (6, 3008)
DO TO 3000
C===== CONTINUE
4460 CONTINUE
C== RM: COMMAND ======
4460 RM: COMMAND ======
C### PRADR = TOTAL POWER RADIATED IN WATTS
3441 C###
READ (6, +) PRADR
WRITE (6, 3441) PRADR
WRITE (6, 3006)
C###
5466 C### MAX = NUMBER OF ANTENNA SEGMENTS
3006 C###
READ (6, +) MAX
IF (MAX GT .999) WRITE (6, 4477) MAX
4477 FORMAT ( ' ***** NUMBER OF SEGMENTS= ', I3, 
' 2 PROGRAM ABORTS SINCE MAX. RECEIVER DIMENSION' , 
3, ' IS EXCEEDED. *****')
STOP
WRITE (6, 3461) MAX
WRITE (6, 3006)
WRITE (6, 3006)
C###
LOCATION OF KTH ANTENNA SEGMENT

ELECTRIC LINEAR ELEMENT WITH A UNIFORM DISTRIBUTION

LENGTH OF LINEAR ELEMENT

ORIENTATION ANGLES USED TO DEFINE LINEAR ELEMENT AXIS.

REAL AND IMAGINARY CURRENT WEIGHT.

WRITE(6,4458) LABEL(IUNIT)

FORMAT(2H *.T7,*'HR:'.A0,T23,*'HR:METERS',

INPUT: THR, PHR*, T60, ACTUAL: THR, PHR*, T70, IH*)

WRITE(6,4459)

FORMAT(*.T7,*.T7,*.T7,*.T7,*.T7)

INPUT:

ACTUAL:

THR, PHR'

WRITE(6,3469)

HR(MR)=UNITS*HRQ

THR=QO.

PHR=PHRZ

XQ(l)=SIN(TQ*RPD)*COS(PHQ*RPD)

XQ(2)=SIN(TQ*RPD)*SIN(PHQ*RPD)

XQ(3)=COS(TQ*RPD)

WRITE(6,3484)

WRITE(0,3008)

WRITE(0,3464)

WRITE(6,4466) LABEL(IUNIT)

FORMAT(*.T7,*'HR:'.A0,T23,*'HR:METERS',

INPUT LOCATION IN ',A6,

'ACTUAL LOCATION IN METERS',T70, '*)

WRITE(6,3467)

WRITE(6,3466) LABEL(IUNIT)

FORMAT(*.T7,*'HR:'.A0,T23,*'HR:METERS',

INPUT LOCATION IN ',A6,

'ACTUAL LOCATION IN METERS',T70, '*)

WRITE(6,3467)

WRITE(6,3008)

WRITE(6,3465)

WRITE(0,3008)

WRITE(0,3464)

WRITE(6,4466) LABEL(IUNIT)

FORMAT(*.T7,*'HR:'.A0,T23,*'HR:METERS',

INPUT LOCATION IN ',A6,

'ACTUAL LOCATION IN METERS',T70, '*)

WRITE(6,3467)

WRITE(6,3008)

WRITE(6,3465)

WRITE(0,3008)

WRITE(0,3464)

WRITE(6,3008)

WRITE(6,4466) LABEL(IUNIT)

FORMAT(*.T7,*'HR:'.A0,T23,*'HR:METERS',

INPUT LOCATION IN ',A6,

'ACTUAL LOCATION IN METERS',T70, '*)

WRITE(6,3467)
WRITE(6,3006)
WRITE(6,3006)
WRITE(6,3465)
WRITE(6,3466)
DO 4466 IJIR=1,MRX
READ(6,*),MRX
WR=CMPLX(I6MR,WR)
YIMM=BABS(CMPLX(I6MR,WR))
YPP=DPR*BTA12
WR=CMPLX(I6MR,WR)
WRITE(6,3466)MR,YMR,WF'R,WW,YPP
4466 CONTINUE
WRITE(6,3006)
GO TO 3000
3000 CONTINUE
C== command
C== NR: COMMAND
C### INITIALIZE SOURCE DATA.
C###
LMP=.FALSE.
MSX=0
MSAT=0
WRITE(6,3491)
3491 FORMAT(*,6X,'THE SOURCE DATA IS INITIALIZED.',T70,1H+)
2,2H .*6X.' NO SOURCES ARE PRESENTLY IN THE PROBLEM.
3,779,1H+)
GO TO 3000
3496 CONTINUE
C== command
C== NR: COMMAND
C### INITIALIZE RECEIVER DATA
C###
LACVR=.FALSE.
LWFR=.FALSE.
LNX=0
WRITE(6,3496)
3496 FORMAT(*,6X,'THE RECEIVER DATA IS INITIALIZED. ')
2,779,1H+/2H .*6X.' NO RECEIVERS ARE PRESENTLY IN THE
3,779,1H+)
GO TO 3000
3500 CONTINUE
C### WRITE=TRUE IF LINE PRINTER OUTPUT OF DATA IS DESIRED
C###
READ(5,*) LWRITE
6605 FORMAT(*,6X,'NO LINE PRINTER OUTPUT',T70,1H+)
IF(.NOT.LWRITE) GO TO 3000
WRITE(6,3501)
3501 FORMAT(*,6X,'DATA WILL BE OUTPUT ON LINE PRINTER !11!',
2T70,1H+)
GO TO 3000
3505 CONTINUE
C== command
C== LP: COMMAND
C###
C### VOLUMETRIC DUMP FOR PLOTS
C###
READ(5,*) LVPLT
IF(.NOT.LVPLT) WRITE(6,6606)
IF(.NOT.LVPLT) GO TO 3000
179
 Continue

 C*** LPLY=TRUE IF PEN PLOTTER OUTPUT IS DESIRED
 C***
 C*** READ(6,*) LPLY
 C*** IF (.NOT. LPLY) WRITE(6,6603)
 C*** WRITE(6,6609) FORMAT(2H8X,'NO PEn PLOT DESIRED',179,1H)*
 C*** IF (.NOT.LPLY) GO TO 3000
 C***
 C*** IF LPLY=TRUE READ IN DIMENSIONS
 C***
 C*** LPPREC = TRUE IMPLIES RECTANGULAR PLOT
 C*** PPXL = LENGTH OF X-AXIS (ANGLE AXIS)
 C*** PPYL = LENGTH OF Y-AXIS (OB AXIS)
 C***
 C*** LPPREC = FALSE IMPLIES POLAR PLOT
 C*** PPPL = ANGULAR POSITION OF X-AXIS
 C*** PPPL = RADIUS OF GRID
 C***
 C*** READ(6,*) LPPREC, PPXL, PPYL
 C***
 C*** PPXIB = BEGINNING VALUE OF X-AXIS
 C*** PPXE = END VALUE OF X-AXIS
 C*** PPXS = STEP SIZE OF X-AXIS GRID MARKS
 C***
 C*** READ(6,*) PPXIB, PPXE, PPXS
 C***
 C*** PPYB = BEGINNING VALUE OF Y-AXIS
 C*** PPYE = END VALUE OF Y-AXIS
 C*** PPYS = STEP SIZE OF Y-AXIS GRID MARKS
 C***
 C*** READ(6,*) PPYB, PPYE, PPYS
 C***
 C*** WRITE(6,3603) WRITE(6,3600)
 C***
 C*** WRITE(6,3603) LPPREC, PPXL, PPYL
 C*** WRITE(6,3603) FORMAT(2H8X,'DATA WILL BE OUTPUT FOR A PLOT !!!'
 C*** 2,179,1H)*
 C*** WRITE(6,3603) WRITE(6,3600)
 C***
 C*** WRITE(6,3603) FORMAT(2H8X,'LPPREC= ',L2,6X,'PPXL= ',F10.6,6X,
 C*** 2*PPYL= ',F10.6,179,1H)*
 C*** WRITE(6,3603) PPIB, PPIE, PPIS
 C***
 C*** WRITE(6,3603) FORMAT(2H8X,'PPXIB= ',F10.6,6X,'PPXE= ',F10.6,6X,
 C*** 2*PPXS= ',F10.6,179,1H)*
 C*** WRITE(6,3603) PPXB, PPXE, PPXS
 C***
 C*** WRITE(6,3603) FORMAT(2H8X,'PPYB= ',F10.6,6X,'PPYE= ',F10.6,6X,
 C*** 2*PPYS= ',F10.6,179,1H)*
 C*** WRITE(6,3603) PPXB, PPXE, PPXS
 C***
 C*** IVTYP= TYPE OF RESULTS OUTPUT
 C*** IVTYP=1 ELECTRIC FIELD OUTPUT
 C*** IVTYP=2 MAGNETIC FIELD OUTPUT
 C*** IVTYP=3 BOTH ELECTRIC AND MAGNETIC FIELDS OUTPUT
 C*** COUPLING IS OUTPUT IF RECEIVER IS DEFINED FOR ANY IVTYP
 C***
 C*** IVPOL= POLARIZATION OF RESULTS OUTPUT
 C*** IVPOL=1,2,3 THEN R, THETA, PHI OR X, Y, Z RESPECTIVELY IS OUTPUT
 C*** IVPOL=4 THEN R-THETA OR X-Y ARE OUTPUT
 C*** IVPOL=5 THEN R-PHI OR X-Z ARE OUTPUT
 C*** IVPOL=6 THEN THETA-PHI OR Y-Z ARE OUTPUT
 C*** IVPOL=7 THEN R, THETA AND PHI OR X, Y AND Z ARE OUTPUT
 C*** COUPLING HAS NO POLARIZATION
READ(6,*) IVTYP, IVPOL
WRITE(6,3856) IVTYP, IVPOL
FORMAT(2H *,6X,* IVTYP= '12,' IVPOL= ',12,79,1H+')
GO TO 3000

C=====
3700 CONTINUE
C==== GP: COMMAND =====

C### INFINITE GROUND PLANE EFFECT INCLUDED.

C### LOAD=.TRUE.
DO 3702 N=1,3
XX(N,1,MPDX)=1.E6*(VRT(1,N)*VRT(2,N)+TR(N))
XX(N,2,MPDX)=1.E6*(-VRT(1,N)+VRT(2,N)+TR(N))
XX(N,3,MPDX)=1.E6*(-VRT(1,N)-VRT(2,N)+TR(N))
3702 XX(N,4,WDX)=1.E6*(VRT(1,N)-VRT(2,N))+TR(N)

C### LSLAB=0 IMPLIES METAL PLATE, AND
C### LSLAB=-3 IMPLIES DIELECTRIC HALF SPACE
C### NOTE: IF DIELECTRIC COVERED, ONE MUST READ DIELECTRIC DATA.
C### ERSLAB, TESLAB, USLAB, TMSLAB
READ(6,*) LSLAB(NPDI)
IF(LSLAB(NPDI).EQ.0) WRITE(6,3706)
FORMAT(2H *,6X,'PERFECTLY CONDUCTING',779,1H*)
IF(LSLAB(NPDI).NE.0) WRITE(6,3707)
FORMAT(2H *,6X,'SEMI-INFINITELY THICK DIELECTRIC',790,1H*)
WRITE(0,3701)
3701 FORMAT(2H *,6X,'INFINITE GROUND PLANE INSERTED IN',1' STRUCTURE',11,179,1H*)
WRITE(6,3703)
FORMAT(2H *,6X,'THE ORIGIN IS AT ',F12.6, ', ',F12.6)
WRITE(6,3704)
FORMAT(2H *,6X,'THE NORMAL IS ',F12.6, ', ',F12.6, ', ')
WRITE(6,3705)
FORMAT(2H *,6X,'PERFECTLY CONDUCTING',779,1H*)
WRITE(6,3706)
FORMAT(2H *,6X,'SEMI-INFINITELY THICK DIELECTRIC',790,1H*)
WRITE(6,3707)
FORMAT(2H *,6X,'INFINITE GROUND PLANE INSERTED IN',1' STRUCTURE',11,179,1H*)
WRITE(6,3708)
FORMAT(2H *,6X,'THE ORIGIN IS AT ',F12.6, ', ',F12.6)
WRITE(6,3709)
FORMAT(2H *,6X,'THE NORMAL IS ',F12.6, ', ',F12.6, ', ')
WRITE(6,3710)
FORMAT(2H *,6X,'PERFECTLY CONDUCTING',779,1H*)
GO TO 3000
CSE.
3760 CONTINUE
C==
NG:
CObNABD

cast

INITIALIZE GROUND PLANE DATA.

LCRllD=.FALSE.
BRITE(6,3761)=.FALSE.
GROUND PLANE DATA IS INITIALIZED.
GROUND PLANE IS PRESENTLY IN THE PROBLEM.
GO TO 3000
C==
PIE=
3900 CONTINUE
C==
RT: COMMAND

TR(N)=LINEAR TRANSLATION OF COORDINATES FROM THE FIXED
COORDINATES WHICH IS ORIGINALLY SET UP BY OPERATOR.

READ(6,*) (TR(N),N=1,3)
WRITE(6,3901) LABEL(IUNIT),(TR(N),N=1,3)
FORMAT(2H*,'TR(N)=',F8.3,'; TR(2)=',F8.3,'; TR(3)=',F8.3,T79.1H*)
DO 3920 N=1,3
3920 TR(Il)=TR(N)*UBITS
WRITE(6,3008)
IF(IUNIT.NE.1) WRITE(6,3901) LABEL(I), (TR(N),N=1,3)
ELSE WRITE(6,3008)
WRITE(6,3009)
C<<< THZP.PHPZ=ORIENTATION OF THE VRT(3,N) AXIS RELATIVE TO THE
FIXED COORDINATE SYSTEM.

READ(6,*)THZP,PHPZ,THXP,PHPX
VRT(3,1)=SIN(THZP*RPD)*COS(PHPZ*RPD)
VRT(3,2)=SIN(THZP*RPD)*SIN(PHPZ*RPD)
VRT(3,3)=COS(THZP*RPD)
VRT(1,1)=SIN(THXP*RPD)*COS(PHPX*RPD)
VRT(1,2)=SIN(THXP*RPD)*SIN(PHPX*RPD)
VRTX(1,3)=COS(THXP*RPD)

INSURE VRT(1,N) IS PERPENDICULAR TO VRT(3,N)

IF(ABS(DZX) .GT. 0.1) WRITE(6,3931)
3931 FORMAT(' *** PROGRAM ABORTS IN ROTATE SECTION IN THAT THE',
'2' COORDINATES ARE NOT ORTHOGONAL!!! ***')
IF(ABS(DZX).GT.0.1) STOP
VRT(1,1)=VRT(1,1)-VRT(3,1)*DZX
VRT(1,2)=VRT(1,2)-VRT(3,2)*DZX
VRT(1,3)=VRT(1,3)-VRT(3,3)*DZX
DOT=VRT(1,1)+VRT(1,2)+VRT(1,3)
DOT=SQRT(DOT)
VRT(1,1)=VRT(1,1)/DOT
VRT(1,2)=VRT(1,2)/DOT
VRT(1,3)=VRT(1,3)/DOT
VRT(2,1)=VRT(2,1)-VRT(3,2)*DZX
VRT(2,2)=VRT(2,2)-VRT(3,2)*DZX
VRT(2,3)=VRT(2,3)-VRT(3,2)*DZX
VRT(3,1)=VRT(3,1)-VRT(3,2)*DZX
VRT(3,2)=VRT(3,2)-VRT(3,2)*DZX
VRT(3,3)=VRT(3,3)-VRT(3,2)*DZX
WRITE(6,3931)
THE FOLLOWING ROTATIONS ARE USED FOR ALL:

2' SUBSEQUENT INPUTS: 'T79,'1H'

DO 3932 NI=1,3

WRITE(6,3006) (NI,3933) (NI,NI,VRT(NI,NI),NJ=1,3)

FORMAT(2H *,6X,'THE FOLLOWING ROTATIONS ARE

FOR ALL',1'

SUBSEQUENT INPUTS:',179.

GO TO 3000

C=====

4000 CONTINUE

C== CC: AND CC: COMMAND ======

3931 FORMAT(1H *,6X,'THE FOLLOWING ROTATIONS ARE

FOR ALL',1'

SUBSEQUENT INPUTS:',179.

GO TO 3000

C=====

4000 CONTINUE

C== CC: AND CC: COMMAND ======

C\$\$ CYLINDER GEOMETRY INPUT

C\$\$ ICL(N,MC)=XYZ LOCATION OF THE ORIGIN OF THE MC-TH CYLINDER

C\$\$ READ(5,*) (ICL(N,MC),N=1,3)

DO 6301 N=1,3

6301 ICL(N,MC)=XCL(N,MC)

DO 6302 N=1,3

6302 ICL(N,MC)=UNITS*(IQ(1)+VRT(1,N)+IQ(2)+VRT(2,N))

WRITE(6,6308) LABEL(IUII1)

6308 FORMAT(2H *,TS.'CYLINDER',T17,'INPUT LOCATION IN

ACTUAL LOCATION IN METERS'.T79,lH*)

WRITE(6,3006)

WRITE(6,3006)

WRITE(6,3006)

WRITE(6,3006)

WRITE(6,3006)

C\$\$ TCLZ,PCLZ=ORIENTATION OF THE CYLINDER'S Z-AXIS

C\$\$ TCLX,PCLX=ORIENTATION OF THE CYLINDER'S X-AXIS

READ(6,*) TCLZ,PCLZ,TCLX,PCLX

IQ(1)=SIN(TCLZ*RPD)*COS(PCLZ*RPD)

IQ(2)=SIN(TCLZ*RPD)*SIN(PCLX*RPD)

IQ(3)=COS(TCLZ*RPD)

DO 6303 N=1,3

6303 VCL(3,N,MC)=VCL(1,N,MC)+VCL(3,1,MC)+VCL(1,2,MC)+VCL(3,2,MC)

WRITE(6,6308) LABEL(IUII1)

6308 FORMAT(2H *,TS.'CYLINDER',T17,'INPUT LOCATION IN

ACTUAL LOCATION IN METERS'.T79,lH*)

WRITE(6,3006)

WRITE(6,3006)

WRITE(6,3006)

WRITE(6,3006)

WRITE(6,3006)

C\$\$ TCLZ,PCLZ=ORIENTATION OF THE CYLINDER'S Z-AXIS

C\$\$ TCLX,PCLX=ORIENTATION OF THE CYLINDER'S X-AXIS

READ(6,*) TCLZ,PCLZ,TCLX,PCLX

IQ(1)=SIN(TCLZ*RPD)*COS(PCLZ*RPD)

IQ(2)=SIN(TCLZ*RPD)*SIN(PCLX*RPD)

IQ(3)=COS(TCLZ*RPD)

DO 6303 N=1,3

6303 VCL(3,N,MC)=VCL(1,N,MC)+VCL(3,1,MC)+VCL(1,2,MC)+VCL(3,2,MC)

WRITE(6,6308) LABEL(IUII1)

6308 FORMAT(2H *,TS.'CYLINDER',T17,'INPUT LOCATION IN

ACTUAL LOCATION IN METERS'.T79,lH*)

WRITE(6,3006)

WRITE(6,3006)

WRITE(6,3006)

WRITE(6,3006)

WRITE(6,3006)
\[ \text{DOT=S\textsc{qrt(DOT)}} \]

\[ \text{VCL(1,1,MC)=VCL(1,1,MC)/DOT} \]

\[ \text{VCL(1,2,MC)=VCL(1,2,MC)/DOT} \]

\[ \text{VCL(1,3,MC)=VCL(1,3,MC)/DOT} \]

\[ \text{VCL(2,1,MC)=VCL(3,2,MC)-VCL(1,3,MC)} \]

\[ \text{VCL(2,2,MC)=VCL(3,3,MC)-VCL(1,1,MC)} \]

\[ \text{VCL(2,3,MC)=VCL(3,2,MC)+VCL(1,1,MC)} \]

\[ \text{WRITE(6,6309)} \]

\[ \text{VCL(l,3,b;C)=VCL(l,3,MC)/DOT} \]

\[ \text{VCL(2,3,blC)=VCL(3,3,~)*VCL(l,3,MC)-VCL(3,1,MC)*VCL(1,3,MC)} \]

\[ \text{WRITE(6,6306)} \]

\[ \text{(H1,NJ,VCL(H1,NJ,MC),NJ=1,3)} \]

\[ \text{FORMAT(2H + XI.3(F1,11,..,11,'..',F9.6),779,1H+)} \]

\[ \text{AC=\text{RADIUS OF ELLIPSE ON X CYLINDER AXIS}} \]

\[ \text{BC=\text{RADIUS OF ELLIPSE ON Y CYLINDER AXIS}} \]

\[ \text{ZCN,THTN=\text{MOST NEGATIVE END CAP'S Z COMPONENT}} \]

\[ \text{ZCP,THTP=\text{MOST POSITIVE END CAP'S Z COMPONENT}} \]

\[ \text{AND ANGLE OF SURFACE WITH THE CYLINDER AXIS} \]

\[ \text{ZCN,THTN,ZCP,THTP} \]

\[ \text{AC(1,MC)=AC(1,MC)*UNITS} \]

\[ \text{BC(1,MC)=BC(1,MC)*UNITS} \]

\[ \text{TCH(2,MC)=TTHP*RPD} \]

\[ \text{TCH(1,MC)=TTHP*RPD} \]

\[ \text{WRITE(6,3006)} \]

\[ \text{WRITE(6,3006)} \]

\[ \text{WRITE(6,3006)} \]

\[ \text{WRITE(6,3006)} \]

\[ \text{WRITE(6,3006)} \]

\[ \text{WRITE(6,3006)} \]

\[ \text{WRITE(6,3006)} \]

\[ \text{WRITE(6,3006)} \]

\[ \text{WRITE(6,3006)} \]

\[ \text{WRITE(6,3006)} \]

\[ \text{WRITE(6,3006)} \]

\[ \text{WRITE(6,3006)} \]

\[ \text{WRITE(6,3006)} \]

\[ \text{WRITE(6,3006)} \]

\[ \text{WRITE(6,3006)} \]

\[ \text{WRITE(6,3006)} \]

\[ \text{WRITE(6,3006)} \]

\[ \text{WRITE(6,3006)} \]

\[ \text{WRITE(6,3006)} \]

\[ \text{WRITE(6,3006)} \]

\[ \text{WRITE(6,3006)} \]

\[ \text{WRITE(6,3006)} \]

\[ \text{WRITE(6,3006)} \]

\[ \text{WRITE(6,3006)} \]

\[ \text{WRITE(6,3006)} \]

\[ \text{WRITE(6,3006)} \]

\[ \text{WRITE(6,3006)} \]

\[ \text{WRITE(6,3006)} \]

\[ \text{WRITE(6,3006)} \]

\[ \text{WRITE(6,3006)} \]
WRITE(6,6350) LABEL(IUNIT),ZCP
6360 FORMAT(2H *,6X,'HIST POSITIVE END CAP Z COMPONENT IN ',
216,="",F8.3,T79,1H+)
6363 WRITE(6,3006)
6364 IF(IUNIT.NE.1) WRITE(6,6350) LABEL(1),ZCP(1,MC)
6365 IF(IUNIT.NE.1) WRITE(6,3006)
6366 WRITE(6,6360) THTP
6367 6360 FORMAT(2H *,6X,'ANGLE OF POS. END CAP SURFACE WITH POS.',
6368 2'T CTI. AXIS ','=',F8.3,T79,1H+)
6369 GO TO 3000
6400 CONTINUE
6401 READ(6,6410) NEC(MC)
6402 IF(NEC(MC).GT.MC) STOP
6403 NEC=M C=0
6404 DO 6410 MC=1,NEC
6405 READ(E,+) AC(NC,MC),BC(NC,MC),ZC(NC,MC)
6406 AAD=AC(NC,MC)
6407 BB=BC(NC,MC)
6408 ZCN=ZC(NC,MC)
6409 AC(NC,MC)=AC(NC,MC)*UNITS
6410 BB=BC(NC,MC)
6411 ZC(NC,MC)=ZC(NC,MC)*UNITS
6412 TC(NC,MC)=0.5*PI
6413 WRITE(6,3305) MC,NC,AAD,BB,ZC,AC(NC,MC),BC(NC,MC),ZC(NC,MC)
6414 6410 CONTINUE
6415 GO TO 3000
6416 C=====
6417 4060 CONTINUE
6418 C== NC: COMMAND =====
6419 C###
6500 C### INITIALIZE CYLINDER DATA.
6511 C###
6512 LCTL=.FALSE.
6513 MC=0
6514 WRITE(6,4061)
6515 4061 FORMAT(2H *,6X,'CYLINDER DATA IS INITIALIZED.',T79,1H+)
6516 2,'H *.EX,' NO CYLINDER IS PRESENTLY IN THE PROBLEM.'
6517 3,T79,1H+)
6518 GO TO 3000
6519 C======
6520 997 CONTINUE
6521 C== EN: COMMAND =====
6522 C###
6523 C### END PROGRAM
6524 C###
6525 WRITE(6,3006)
6526 WRITE(6,3006)
6527 WRITE(6,3006)
6528 STOP
6529 C=====
6530 3800 CONTINUE
6531 C== IQ: COMMAND =====
6532 C###
6533 C### EXECUTE PROGRAM
6534 C###
6535 WRITE(6,3006)
6536 WRITE(6,3006)
6537 WRITE(6,3006)
6538 G !!!
6539 C !!! 2. INITIALIZATION SECTION
6540 C !!!
6540 W L=.2997926/FRQC
6541 W K=TPP/WL
6542 WRITE(6,3005)
6543 MPX=M PX
CIII GROUND PLANE IS ANOTHER PLATE IN SOLUTION.

IF(LGND) MPXR=MFI+1

IF(MPXR.GT.MPDX) WRITE(6,001) MPXR

IF(MPXR.GT.MPDX) STOP

IF(.NOT.LGRND) GO TO 3801

LPLA=.TRUE.

IF(MPXR.EQ.MPDX) DO 3801 LPLA=.FALSE.

IF(LPLA) CALL GEOM

CIII MAKE PATTERN INFORMATION MATCH WITH SHADOW CODE

IF(IPN.GT.0) THEN

NVFT=NPV

NVFP=NPV

ELSE

NVFT=NPV

NVFP=NPV

ENDIF

CDL=NVFT

TETH=TYT*RPD

RESTH=TYT*RPD

THE2=TETH+(CDL-1)*RESTH

ROWS=NVFP

PH1=PS1*RPD

RESPH=PS1*RPD

PH2=PH1+(ROWS-1)*RESPH

CIII MAKE SOURCE INFORMATION FOR THE FIRST ONE MATCH WITH

CIII SHADOW CODE

DO 3806 N=1,3

AUTEN(N)=ASS(N,1)

RETURN

END
FUNCTION BABS

This is function BABS. It is used to obtain complex absolute values without runtime numerical errors.

```
0001     C-----------------------------------------------
0002     FUNCTION BABS(Z)
0003     C!!!
0004     C!!! THIS ROUTINE IS USED TO GIVE COMPLEX ABSOLUTE VALUES. IT IS
0005     C!!! USED RATHER THAN STANDARD ROUTINES TO AVOID EXECUTION
0006     C!!! ERRORS.
0007     C!!!
0008     C!!! COMPLEX Z
0009     COMMON/LIMIT,SMLR,SMLT,BIG
0010     X=ABS(REAL(Z))
0011     Y=ABS(AIMAG(Z))
0012     IF(X.LT.SMLT.AND.Y.LT.SMLT) GO TO 10
0013     BABS=CABS(Z)
0014     RETURN
0015     10   BABS=SMLT
0016     RETURN
0017     END
```
This is constant block data.

0001 C-----------------------------------------------------------------------
0002 BLOCK DATA
0003 C!!! LOAD COMMONLY USED DATA INTO COMMON AREA.
0005 C!!!
0006 C!!!
0006 COMMON CJ,CPI4
0007 COMMON/FIS/PI,TPI,DPF,DPI
0008 COMMON/COMP/CJ,CPI4
0009 COMMON/LIMIT/SML,SMLR,SMLT,BIG
0010 DATA PI,TPI,DPF,DPI/3.14159266,6.28318531,67.29577955,
0011 20.0174632926/ 
0012 DATA CJ,CPI4/(0.,1.),(.70710678,-.70710678)/ 
0013 DATA SML,SMLR,SMLT,BIG/1.E-3,1.E-6,1.E-10,1.E30/ 
0014 END
FUNCTION BTAN2

This function is identical to the intrinsic fortran ATAN2 function, except it avoids runtime numerical errors.

```fortran
C-----------------------------------------------
0001  C                   FUNCTION BTAN2(Y,X)
0002  C                   
0003  C       THIS ROUTINE IS USED TO COMPUTE THE ARCTANGENT. IT IS
0004  C       SIMILAR TO ATAN2 EXCEPT IT AVOIDS THE RUN TIME ERRORS.
0005  C                   
0006  C                   
0007  COMMON/PI5/PI,TP1,DFR,BPD
0008  COMMON/LIMIT/SML,SMLT,BIG
0009  IF(ABS(X).GT.SML) GO TO 60
0010  IF(ABS(Y).GT.SML) GO TO 10
0011  BTAN2=0.
0012  RETURN
0013  10  BTAN2=0.6*PI
0014  IF(Y.LT.0.) BTAN2=-BTAN2
0015  RETURN
0016  60  BTAN2=ATAN2(Y,X)
0017  RETURN
0018  END
```
SUBROUTINE CAPINT

This routine is used to determine if a ray strikes an elliptic cylinder endcap.

0001 C-----------------------------------------------
0002 SUBROUTINE CAPINT(XIS,D,DHIT,MD,LHIT,MH)
0003 CIII DOES RAY HIT ENDCAP?
0004 CIII INCLUDE 'SHACOH.FOR'
0005 CIII
0006 DIMENSION XIS(3),D(3),XT(3),XISC(3),DC(3)
0007 LOGICAL LHIT,LDEBUC,LTEST,LWARN
0008 COMMON/LIMIT,LDEBUC,LTEST,LWARN
0009 COMMON/LIMIT,SMR,SMLT,BIG
0010 COMMON/WAVE/WK,WL
0011 LHIT=.FALSE.
0012 DHIT=.FALSE.
0013 CIII STEP THROUGH CYLINDERS
0014 DO 40 MCC=1,MCX
0015 IF(MH.LT.0.0.AND. IABS(MH).NE.MCC) GO TO 40
0016 IF(MH.GT.0.0.AND. MH.EQ.MCC) GO TO 40
0017 CALL CYLROT(D,DC,1,NCC)
0018 CALL CYLROT(XIS,XISC,2,NCC)
0019 CIII STEP THRU ENDCAPS
0020 NECX=NEC(MCC)
0021 DO 40 MD=1,NECX
0022 IF(MD.LT.0.0.AND. IABS(MD).NE.MD) GO TO 40
0023 IF(MD.GT.0.0.AND. MD.EQ.MD) GO TO 40
0024 A=AC(MD,MCC)
0025 B=BC(MD,MCC)
0026 C=ACOS(TCR(MD,MCC))
0027 S=ACOS(TCR(MD,MCC))
0028 AN=XT(1)+XT(2)*XT(2)*XT(3)-ZC(MD,MCC)*SNC
0029 AN=-XT(1)*XT(1)*XT(3)-ZC(MD,MCC)*SNC
0030 DHT=SQRT(AN*AN+B*B+C*C)
0031 IF(DHIT.LT.DHT) GO TO 40
0032 CIII COMPUTE POINT XT, WHERE RAY HITS ENDCAP PLANE
0033 DO 10 N=1,3
0034 XT(N)=XISC(N)-AN*DC(N)/DHT
0035 RHO=XT(1)*XT(1)+XT(2)*XT(2)+XT(3)*XT(3)
0036 RHO=SQRT(RHO)
0037 AE=A/CNC
0038 CIII IS HIT POINT ON ENDCAP?
0039 IF(RHO.GT.AE.AND. RHO.GT.B) GO TO 40
0040 IF(RHO.LT.AE.AND. RHO.LT.B) GO TO 20
0041 VE=BIAN2(A+XT(2),B+XT(1))
0042 CVE=COS(VE)
0043 CVE=SQRT(AE+CVE+CVE+B+B+SVE+SVE)
0044 IF(RHO.GT.B) GO TO 40
0045 CONTINUE
0046 CIII CALCULATE DHT, THE DISTANCE FROM SOURCE TO HIT POINT
0047 DHT=0.
0048 DO 30 N=1,3
0049 DHT=DHT+XT(N)-XISC(N)*XT(N)-XISC(N)
0050 DHT=SQRT(DHT)+SURDWL
0051 IF(LHIT.AND. (DHT.GT.DHIT)) GO TO 40
0052 LHIT=.TRUE.
0053 DHT=DHT
0054 IF(MD.LE.0.0) GO TO 40

190
CALL CTYROT(I1S,IT,-2,MCC)
40 CONTINUE
IF(LTEST) THEN
900 WRITE(6,900)
900 FORMAT(/,' TESTING CAPINT SUBROUTINE')
WRITE(6,*), IIS
WRITE(6,*), D
WRITE(6,*), DHIT,MD,LHIT,MH
ENDIF
RETURN
END
SUBROUTINE CYLINT

This routine is used to determine if a ray strikes an elliptic cylinder.

0001  C-----------------------------------------------
0002  SUBROUTINE CYLINT(XS,D,DHIT,LHIT,LIBDF,MH)
0003  CIII
0004  CIII  DOES RAY HIT CYLINDER?
0005  CIII
0006  INCLUDE 'SHACOM.FOR'
0007  DIMENSION D(3),XS(3),VID(2),BT(4),CTC(2),DC(3),ISC(3)
0008  LOGICAL LHIT,LIBDF,LHT,LHIT,LDEBU,LDEBG,LTEST,LHTEST,LWMAR
0009  COMMON/LPLCT/LPLA,LCTL
0010  COMMON/TEST/LDEBG,LTEST,LWMAR
0011  COMMON/LIMIT/SML,SMLR,SMLT,BIG
0012  COMMON/WAVE/WR,WL
0013  LHIT=.FALSE.
0014  DHIT=0.
0015  IF(.NOT.LCTYL) GO TO 100
0016  CIII  STEP THRU CYLINDERS
0017  DO 60 MCC=1,MCX
0018  IF(MH.LE.0.AND.IABS(MH).EQ.MCC) GO TO 60
0019  IF(MH.GT.0.AND.MH.ME.MCC) GO TO 60
0020  CALL CYLRST(XS,XSC,1,MCC)
0021  CALL CYLRST(D,DC,1,KC)
0022  CIII  DOES RAY HIT CYLINDER SURFACE SECTION?
0023  PHSR=BTAN2(DC(2),DC(1))
0024  CPS=COS(PHSR)
0025  SPS=SSIN(PHSR)
0026  RHOS=SQRT(ISC(1)*ISC(1)+ISC(2)*ISC(2))
0027  CIII  STEP THRU CYLINDER SECTIONS
0028  NECX=NEC(MCC)-1
0029  DO 40 NC=1,NECX
0030  A=AC(NC,MCC)
0031  B=BC(NC,MCC)
0032  CIII  PARAMETERS FOR ELLIPTIC CYLINDER
0033  CTC(1)=COS(TCR(NC,MCC))/SIN(TCR(NC,MCC))
0034  CTC(2)=COS(TCR(NC,MCC))/SIN(TCR(NC,MCC))
0035  CIII  PARAMETERS FOR CONE FRUSTUMS SECTIONS
0036  ZZC=ZC(NC,MCC)-ZC(NC,MCC)
0037  IF(ABS(ZZC).LT.SML*WL) GO TO 40
0038  TNJ=(AC(NC,MCC)-AC(NC,MCC))/ZZC
0039  FL=TNJ*(XSC(NC,MCC)-B*XSC(3))/R.1.
0040  RHOE=SQRT(B*XSC(2)*B*XSC(3))
0041  CIII  CONTINUE
0042  IS SOURCE INSIDE OF INFINITE CYLINDER?
0043  IF(RHOS.LT.AL.AND.RHOS.GT.BL) GO TO 5
0044  IF(RHOS.LT.AL.AND.RHOS.GT.BL) GO TO 30
0045  VE=BTAN2(AL-XSC(2),BL+XSC(1))
0046  CVE=CSSVE(VE)
0047  SVE=SSIN(VE)
0048  RHOS=SQRT(AL+AL+CVE+BL*BL*SVE*SVE)
0049  IF(RHOS.GT.RHOS) GO TO 6
0050  CONTINUE
0051  IS SOURCE INSIDE OF INFINITE CYLINDER SECTIONS?
0052  IF(ISC(3).LT.(ZC(NC,MCC)-ZC(NC,MCC))/A+1) GO TO 40
0053  GO TO 100

192
CONTINUE

C111 FIND COEFFICIENT OF EQUATION TO DETERMINE HIT POINT
AA=A*A
BB=B*B
C111 PARTS FOR ALL ELLIPTIC CROSS SECTION TYPES
CA=DC(1)+DC(1)/AA+DC(2)*DC(2)/BB
CB=ISC(1)+DC(1)/AA+ISC(2)+DC(2)/BB
CC=ISC(1)+ISC(1)/AA+ISC(2)*ISC(2)/BB
C111 PARTS FOR CONE FRUSTUM SECTIONS
CA=CA-TNJ*TNJ*DC(3)*DC(3)/AA
CB=CB-TNJ*FL*DC(3)/A
CC=CC+FL*FL
C111 IS QUADRATIC SOLVABLE IN REAL SPACE?
C111 IF NOT, NO HIT POINT ON CYLINDER SURFACE SECTION
CT=CB*CB-CA*CC
IF(CT.LE.0.) GO TO 40
C111 DETERMINE TWO POSSIBLE HIT DISTANCES
SCT=SQR(CT)
RH=(-CB+SCT)/CA
RN=(-CB-SCT)/CA
C111 NEAREST POSITIVE ONE IS TRUE HIT POINT
IF(RHP.LT.0. AND.RHM.LT.0.) THEN
GO TO 40
ELSE
IF(RHP.LT.0. OR.RHM.LT.0.) THEN
RH=AMAX1(RHP,RHM)
ELSE
RH=AMIN1(RHP,RHM)
ENDIF
C111 IS HIT POINT ON FINITE CYLINDER SECTION?
IF(ZPM.GT.ZC(NC,NC)+XPM*CTC(1) OR.
ZPM.LT.ZC(NC,NC)*XPM*CTC(2)) GO TO 40
C111 DISTANCE FROM SOURCE TO HIT
DHT=RH+SMLR*WL
C111 CHECK FOR NEAREST HIT POINT FOR DIFFERENT SECTIONS
IF(LHIT .AND. (DHT.CT.DHIT)) GO TO 40
LHIT=.TRUE.
DHT=DHT
CONTINUE
C111 CHECK TO SEE IF RAY HITS ENDCAPS
CALL CAPINT(IS,DHT,O,LHT,-MCC)
IF( .NOT.LHT) GO TO 60
IF(LHIT .AND. (DHT.GT.DHIT)) GO TO 60
LHIT=.TRUE.
DHT=DHT
CONTINUE
50 IF( .TEST.) THEN
WRITE(6,900)
900 FORMAT(1'A TESTING CYLINT SUBROUTINE'
WRITE(6,*)
WRITE(6,*)
WRITE(6,*)
WRITE(6,*)
WRITE(6,*)
WRITE(6,*)
999 END
SUBROUTINE CYLROT

This routine performs vector transformations between the various cylinder coordinate systems and the reference coordinate system.

C-----------------------------------------------------------------------
SUBROUTINE CYLROT(XREF,XCYL,A,MCL)
C$$$$
C$$$
C$$
C$$$
INCLUDE 'SHACOM.FOR'
DIMENSION IREF(3),XCYL(3)
IF(N.LT.0) GO TO 100
XC=0.
DO 111 3
XCYL(I)=0.
111 DO 60 3
IF(N.EQ.2) XC=XCL(J,MCL)
XCYL(I)=XCYL(I)+VCL(I,J,MCL)*(XREF(J)-XC)
60 CONTINUE
RETURN
007
0173 DIMENSION IREF(3),XCYL(3)
0174 IF(N.LT.0) GO TO 100
0175 XC=0.
0176 DO 111 3
0177 XCYL(I)=0.
0178 DO 111 3
0179 IF(N.EQ.2) XC=XCL(J,MCL)
0180 XCYL(I)=XCYL(I)+VCL(I,J,MCL)*(XREF(J)-XC)
0181 CONTINUE
0182 RETURN
0183 100 DO 200 3
0184 XREF(I)=0.
0185 DO 150 3
0186 150 XREF(I)=XREF(I)+VCL(I,J,MCL)*XCYL(J)
0187 IF(N.EQ.-2) XREF(I)=XREF(I)+XCL(I,MCL)
0188 CONTINUE
0189 RETURN
0190 END
SUBROUTINE DOCYLS

This procedure determines which mode of mapping has been selected by the user and calls the appropriate cylinder processing routines.

```fortran
0001 C---------------------------------------------------------------------------------
0002 SUBROUTINE DOCYLS
0003 INCLUDE 'SHACOM.FOR'
0169 C!!! This subroutine processes all the cylinders one at a time.
0170 C!!! Do any special cylinders last.
0171 C!!!
0172 C!!! IF ( FICTNM .GT. 0 ) THEN
0173 DO 1 MC=1, MCX
0174 IF ( MC .NE. FICTNM ) CALL DOCTL( MC, FICTHR )
0175 1 CONTINUE
0177 CALL DOCTL( FICTNM, FICTHC )
0178 C!!!
0179 C!!! Fill with a different character for each cylinder.
0180 C!!! ELSEIF ( FICTNM .LT. 0 ) THEN
0181 DO 2 MC=1, MCX
0182 CALL DOCTL( MC, CHAR( MC*ICHAR( '0' ) ) )
0183 2 CONTINUE
0186 C!!!
0188 C!!! Fill with the main background fill character.
0190 C!!! ELSE
0192 DO 3 MC=1, MCX
0193 CALL DOCTL( MC, FICTHR )
0194 3 CONTINUE
0192 ENDIF
0193 RETURN
0194 END
```
SUBROUTINE DOCYL

This routine projects the shadow boundary of a single cylinder onto the far-zone sphere and fills the area of the cylinder with the FILL argument.

0001 C-----------------------------------------------------------------------
0002 SUBROUTINE DOCYL( IC, FILL )
0003 INCLUDE 'SHACOL.FOR'
0169 C!!! This subroutine processes a single cylinder.
0171 C!!!
0172 CHARACTER FILL
0173 INTEGER
0174 + J, K, IC
0176 CI Loop control variables.
0178 + IST,
0179 C! Truncate to integer.
0180 + THETAI, PHI
0182 C! Loop control variables.
0183 + REAL
0185 C! Theta & phi in radians.
0186 CI The parametric loop parameter.
0188 CI Length of a pseudo-side
0189 CI Scratch variable.
0190 + DOT, LSTDOT,
0192 CI Dot product variables
0194 CI! Miscellaneous functions
0196 + XYZ( 3 ),
0198 CI! temporary vector
0200 + XPF( 3 ),
0204 CI Source to edge in ref coords
0208 + XPC( 3 ),
0212 CI Source to edge in pat coords
0216 + RIM( 3 ),
0220 CI Point along cap in cyl coords
0224 CI! Transform the antenna to cyl coords (include a translation).
0228 CI! The two angles where dot is minimum
0232 C!! LOGICAL
0236 + FNDONE
0240 CI! Found one of the zero dots
0244 CI
0248 C!! Loop through endcaps, and incrementally on edges.
0252 C!! Transform the antenna to cyl coords (include a translation).
0256 C!!
0260 CALL CYLROT( ANTEW1, ANCYL, +2, IC )
0264 C!!
0268 C!! Do the endcaps one at a time.
0272 C!!!
0276 DO 200 J=1, NEG(IC)
0280 C!!!
0284 C!!! Loop around the endcap and remember where the dot products are zero
0288 C!!! between the vector looking at the point and the radial vector on

196
Clll the endcap to the point. The cryptic parameters on the loop say:
Clll "Loop from zero to 2*PI in one-degree steps."

DD 300 T=0.0, TPI+(TPI/360.0), (TPI/360.0)
Clll Calculate the dot product and remember the two smallest ones.
Clll

RIM(1) = COS(T) \* AC(J,IC)
RIM(2) = SIN(T) \* BC(J,IC)
RIM(3) = ZC(J,IC)
DGT = RIM(1) \* ( RIM(1) - ANCTL(1) )
RIM(2) \* ( RIM(2) - ANCTL(2) )

Clll
Clll If (the last dot product) \* (this dot product) < 0, then that is
Clll where our dot sign goes through zero.
Clll
Clll

IF ( T .EQ. 0.0 ) THEN
LSTDOT = DOT.
FDONE = .FALSE.
ENDIF
IF ( SIGH(1.0, DOT) \* SIGH(1.0, LSTDOT) .LT. 0.0 ) THEN
IF ( NOT. FDONE ) THEN
DOTMIN(1, J) = T
FDONE = .TRUE.
ELSE
DOTMIN(1, J) = T
ENDIF
ENDIF
LSTDOT = DOT

Clll Calculate theta & phi as we go around the rim.
Clll Transform the rim point into ref. coord., system.
Clll Find vector from source to rim.
Clll

CALL CYLROT( XYZ, RIM, -2, IC )
Clll Convert from the reference coordinate system to the pattern
call coordinate system.
Clll

XPQ(1) = XYZ(1) \- ABTEYN(1)
XPQ(2) = XYZ(2) \- ANTELY(2)
XPQ(3) = XYZ(3) \- ANTELN(3)
XPC(1) = XPQ(1) \+ VPC(1,1) \+ XPQ(2) \+ VPC(1,2) \+ XPQ(3) \+ VPC(1,3)
XPC(2) = XPQ(1) \+ VPC(2,1) \+ XPQ(2) \+ VPC(2,2) \+ XPQ(3) \+ VPC(2,3)
XPC(3) = XPQ(1) \+ VPC(3,1) \+ XPQ(2) \+ VPC(3,2) \+ XPQ(3) \+ VPC(3,3)
XPY = SQRT( XPC(1) \* XPC(1) \+ XPC(2) \* XPC(2) )
THETAR = DTAN2( XPY, XPC(3) )
PHIR = BTAII( XPC(2), XPC(1) )
IF ( PHIR .LT. PHI - 0.6 * RESPH ) PHIR = TPI + PHIR
THETAI = INT( (THETAR - THETI) / RESTH + 0.6 ) + 1
PHII = INT( (PHIR - PHI) / RESPH + 0.6 ) + 1
Clll
Clll Put the character into the output buffer at the proper position.
Clll

197
IF ( (THETA1 .GE. 1) .AND. (THETA1 .LE. COLS) ) THEN
  IF ( (PHII .GE. 1) .AND. (PHII .LE. ROWS) ) THEN
    OUTBUF( THETA1, PHII ) = CHAR(7)
  ENDIF
ENDIF

C!!! Reduplicate a wrapped-around character.
C!!!
IF( (PHII .EQ. 1) .AND. ABS(PHI-PHI-TPI) .LE. RESPH)
THEN
  OUTBUF( THETAI, ROYS ) = CHAR(7)
ENDIF

300 CONTINUE
200 CONTINUE
C!!!
Before rasterizing, connect the "dotmins".
C!!! A sneaky trick is pulled here. Instead of transforming every
C!!! increment of the dotmin points, only the two end points are
C!!! transformed, then theta & phi are calculated for each increment.
C!!! This is valid because the line which connects the two points on
C!!! the rims of the cylinders are straight lines in both RCS and cyl
C!!! coord systems. Note that this gizmo assumes that you are never
C!!! inside of a cylinder, or your dotmins(K,.) probably get crossed
C!!! resulting in an inside-out or bowtie-shaped cylinder.

DO 400 K=1, 2
  DO 600 J=1, NEC(1C)-1
    RIM1(1) = COS( DOTMIN(K,J) ) * AC(J,IC)
    RIM1(2) = SIN( DOTMIN(K,J) ) * BC(J,IC)
    RIM1(3) = ZC(J,IC)
    CALL CYLROT( RIM1, RIM, -1, IC)
    RIM2(1) = COS( DOTMIN(K,J+1) ) * AC(J+1,IC)
    RIM2(2) = SIN( DOTMIN(K,J+1) ) * BC(J+1,IC)
    RIM2(3) = ZC(J+1,IC)
    CALL CYLROT( RIM2, RIM, -2, IC)
  T = 0.0
  IF ( T .GT. 1.0 ) GOTO 600
  MAGNE = SQRT( RIM2(1) - RIM1(1) )**2 +
            RIM2(2) - RIM1(2) )**2 +
            RIM2(3) - RIM1(3) )**2
  T = 0.0
  IF ( T .GT. 1.0 ) GOTO 600
C!!!
C!!! This MAGNE is analogous to the one in DOPLA except it
C!!! works with pseudo-sides, so the name is somewhat misleading.
C!!!
C!!!
C!!!
C!!! These functions compute the theta/phi associated with a given point
C!!! along a cylinder pseudo-edge as a function of T (See DOPLA.)
C!!! The variables XYZ and RIM are re-used for multiple purposes here.
C!!!
C!!!
C!!!
C!!! Find vector from source to rim.
C!!!
C!!! Convert from the reference coordinate system to the pattern
C!!!
C!!!
C!!!
C!!!
C!!! Define the angles representing the projection of the curved sides

198
and do a branch cut test on phi.

THETA1 = INT((THETAR - ZHETL) / RESTH + 0.6) + 1
PHII = INT((PHIR - PHI) / RESPH + 0.6) + 1

Check if angles fall within window.

IF( (PHII .GE. 1) .AND. (PHII .LE. ROWS) ) THEN
OUTBUF( THETAI, PHII ) = CHAR(7)
ENDIF

Reduplicate a wrapped-around character.

IF( (PHII .EQ. 1) .AND. ABS(PHI-PHI1) .LE. RESPH ) THEN
OUTBUF( THETAI, ROWS ) = CHAR(7)
ENDIF

Tell SCAN that this is a CYLINDER by using a "2".

DO 700 PHII = 1, ROWS
CALL SCAN( IC, OUTBUF(1,PHII), PHII, FILL, 2 )
700 CONTINUE

RETURN
END
SUBROUTINE DOPLAS

This routine determines which mapping options the user has selected and calls the appropriate plate processing routines.

C-----------------------------------------------------------------------
SUBROUTINE DOPLAS

CIII This subroutine processes each plate one at a time. The
CIII highlighting logic is contained here.
CIII
CIII Do the plates one at a time, then do the plate that was supposed to
CIII be highlighted last.
CIII
CIII IF (FILPNM.GT.0) THEN
0177 DO 1 MP = 1, MPX
0178 IF (MP.NE.FILPNM) CALL DOPLA(MP, FILCHR)
0179 1 CONTINUE
0180 CALL DOPLA(FILPNM, FILCHP)
0181 CIII Fill with a different character for each plate.
0183 CIII ELSEIF (FILPNM.LT.0) THEN
0185 DO 2 MP = 1, MPX
0186 CALL DOPLA(MP, CHAR(MPIFCHAR('O'))) 
0187 2 CONTINUE
0188 CIII Fill everything with the main background character.
0190 CIII ELSE
0192 DO 3 MP = 1, MPX
0193 CALL DOPLA(MP, FILCHR) 
0194 3 CONTINUE
0195 ENDIF
0196
0197 RETURN
0198 END

200
SUBROUTINE DOPLA

This routine computes the shadow map for a single cylinder by projecting its boundaries onto the far-zone sphere and then filling in its area in the map array.

```
0001 C-----------------------------------------------
0002 SUBROUTINE DOPLA( IP, FILL )
0003   INCLUDE 'SHACON.FOR'
0004 CHARACTER FILL
0005   INTEGER + IP, INT,
0006   CI Truncate to the nearest integer.
0007 + THETAI, PHI1
0008 CI Local indices into char array.
0009 + REAL
0010   T.
0011 + CI Parametric increment parameter.
0012 + THETAR, PHIR, CI Theta & phi in radians.
0013 + MAGE, CI Length of side ME.
0014 + IPY,
0015 + CI temporary variable
0016 + IPQ(3),
0017 CI Source to edge in ref coords
0018 + IP(3),
0019 CI Source to edge in pst coords
0020 + BTAN2, SQRT, ABS
0021 CI Miscellaneous functions.
0022 CII!! CI!!! Loop through incrementally along edges.
0023 CII!! DD 200 ME=1, MEP(IP)
0024 CI!!! NEXTME = MOD(ME, MEP(IP)) + 1
0025 CI!!! MAGE = VNAC(ME, IP)
0026 CI!!! T = 0.0
0027 EQ IF ( T .GT. 1.0 ) GOTO 100
0028 CI!!!
0029 CI!!! These functions compute the theta/phi associated with a given
0030 CI!!! point along an edge between two corners ME and NEXTME as a
0031 CI!!! function of T. T varies from 0 to 1 and is adjusted to keep
0032 CI!!! within a safe and efficient excursion at all times.
0033 CI!!!
0034 CI!!! Convert from the reference coordinate system to the pattern
0035 CI!!! coordinate system
0036 CII!!
0037 CI!!! IP(1)=(XX(1,NEXTME,IP)-XX(1,ME,IP))*T+XX(1,ME,IP)-ANTENN(1)
0038 CI!!! IP(2)=(XX(2,NEXTME,IP)-XX(2,ME,IP))*T+XX(2,ME,IP)-ANTENN(2)
0039 CI!!! IP(3)=(XX(3,NEXTME,IP)-XX(3,ME,IP))*T+XX(3,ME,IP)-ANTENN(3)
0040 CI!!!
0041 CI!!!
0042 CI!!!
0043 CI!!!
0044 CI!!!
0045 CI!!!
0046 CI!!!
0047 CI!!!
0048 CI!!!
0049 CI!!!
0050 CI!!!
0051 CI!!!
0052 CI!!!
0053 CI!!!
0054 CI!!!
0055 CI!!!
0056 CI!!!
0057 CI!!!
0058 CI!!!
0059 CI!!!
0060 CI!!!
0061 CI!!!
0062 CI!!!
0063 CI!!!
0064 CI!!!
0065 CI!!!
0066 CI!!!
0067 CI!!!
0068 CI!!!
0069 CI!!!
0070 CI!!!
0071 CI!!!
0072 CI!!!
0073 CI!!!
0074 CI!!!
0075 CI!!!
0076 CI!!!
0077 CI!!!
0078 CI!!!
0079 CI!!!
0080 CI!!!
0081 CI!!!
0082 CI!!!
0083 CI!!!
0084 CI!!!
0085 CI!!!
0086 CI!!!
0087 CI!!!
0088 CI!!!
0089 CI!!!
0090 CI!!!
0091 CI!!!
0092 CI!!!
0093 CI!!!
0094 CI!!!
0095 CI!!!
0096 CI!!!
0097 CI!!!
0098 CI!!!
0099 CI!!!
0100 CI!!!
0101 CI!!!
0102 CI!!!
0103 CI!!!
0104 CI!!!
0105 CI!!!
0106 CI!!!
0107 CI!!!
0108 CI!!!
0109 CI!!!
0110 CI!!!
0111 CI!!!
0112 CI!!!
0113 CI!!!
0114 CI!!!
0115 CI!!!
0116 CI!!!
0117 CI!!!
0118 CI!!!
0119 CI!!!
0120 CI!!!
0121 CI!!!
0122 CI!!!
0123 CI!!!
0124 CI!!!
0125 CI!!!
0126 CI!!!
0127 CI!!!
0128 CI!!!
0129 CI!!!
0130 CI!!!
0131 CI!!!
0132 CI!!!
0133 CI!!!
0134 CI!!!
0135 CI!!!
0136 CI!!!
0137 CI!!!
0138 CI!!!
0139 CI!!!
0140 CI!!!
0141 CI!!!
0142 CI!!!
0143 CI!!!
0144 CI!!!
0145 CI!!!
0146 CI!!!
0147 CI!!!
0148 CI!!!
0149 CI!!!
0150 CI!!!
0151 CI!!!
0152 CI!!!
0153 CI!!!
0154 CI!!!
0155 CI!!!
0156 CI!!!
0157 CI!!!
0158 CI!!!
0159 CI!!!
0160 CI!!!
0161 CI!!!
0162 CI!!!
0163 CI!!!
0164 CI!!!
0165 CI!!!
0166 CI!!!
0167 CI!!!
0168 CI!!!
0169 CI!!!
0170 CI!!!
0171 CI!!!
0172 CI!!!
0173 CI!!!
0174 CI!!!
0175 CI!!!
0176 CI!!!
0177 CI!!!
0178 CI!!!
0179 CI!!!
0180 CI!!!
0181 CI!!!
0182 CI!!!
0183 CI!!!
0184 CI!!!
0185 CI!!!
0186 CI!!!
0187 CI!!!
0188 CI!!!
0189 CI!!!
0190 CI!!!
0191 CI!!!
0192 CI!!!
0193 CI!!!
0194 CI!!!
0195 CI!!!
0196 CI!!!
0197 CI!!!
0198 CI!!!
0199 CI!!!
0200 CI!!!
0201 DO 200 ME=1, MEP(IP)
0202 NEXTME = MOD(ME, MEP(IP)) + 1
0203 MAGE = VNAC(ME, IP)
0204 T = 0.0
0205 EQ IF ( T .GT. 1.0 ) GOTO 100
0206 CI!!!
0207 CI!!!
0208 CI!!!
0209 CI!!!
0210 CI!!!
0211 CI!!!
0212 CI!!!
0213 CI!!!
0214 CI!!!
0215 CI!!!
0216 CI!!!
0217 CI!!!
0218 CI!!!
0219 CI!!!
0220 CI!!!
0221 CI!!!
```

201
0222    XPY = SQRT( XPC(1)+XPC(1) + XPC(2)+XPC(2) )
0224    CIII Define the angles representing the projection of the curved sides
0226    CIII and do a branch cut test on phi.
0227    CIII
0228    THETAR = BTAN2( XPY, XPC(3) )
0229    PHIR  = BTAN2( XPC(2), XPC(1) )
0230    IF ( PHIR .LT. PHI-0.5*RESPH ) PHIR = TPI + PHIR
0232    CIII Define pixel location and put the a character in the appropriate
0233    CIII spot.
0234    CIII
0235    THETAI = INT( (THETAR - THETI) / RESTH + 0.5 ) + 1
0237    PHII  = INT( (PHIR - PHI) / RESPH + 0.5 ) + 1
0238    CIII
0239    CIII Check if angles fall within window.
0240    CIII
0241    IF ( (THETAI .GE. 1) .AND. (THETAI .LE. COLS) ) THEN
0242    IF ( (PHII .GE. 1) .AND. (PHII .LE. ROWS) ) THEN
0243    OUTBUF( THETAI, PHII ) = CHAR(7)
0244    ENDIF
0245    ENDIF
0246    CIII
0247    CIII Reduplicate a wrapped-around character.
0248    CIII
0249    IF( (PHII .EQ. 1) .AND. (ABS(PHI2-PHI-TPI) .LE. RESPH) THEN
0250    OUTBUF( THETAI, ROWS ) = CHAR(7)
0251    ENDIF
0252    CIII
0253    CIII Put an upper bound on the increment for the case when the line
0254    CIII segment is very short or the distance to the segment is great.
0255    CIII In the degenerate case (on the Z-axis) prevent a potential infinite
0256    CIII loop by putting a lower bound on delta-t (ie by always adding at
0257    CIII least a very small number to T.)
0258    CIII
0259    T = T + MIN( 0.99, (XPY+ALPH/MAGMF + 1.E-7) )
0260    GOTO 50
0261    CONTINUE
0262    CONTINUE
0263    CIII
0264    CIII Now do an area fill on the object just outlined.
0265    CIII Tell SCAN that this is a plate by using a "1".
0266    CIII
0267    DO 300 PHI = 1, ROWS
0268    CALL SCAN( IP, OUTBUF(I.PHI), PHII, FILL, 1 )
0269    CONTINUE
0270    CONTINUE
0271    RETURN
0272    END
SUBROUTINE GEOM

This routine computes necessary geometrical information needed by other routines. It is called before the main command loop.

0001 C-------------------------------------------------------------------
0002 SUBROUTINE GEOM
0003 CI
0004 THIS ROUTINE COMPUTES ALL THE GEOMETRY ASSOCIATED
0005 CI WITH FIXED PLATE STRUCTURE, SUCH AS EDGE UNIT VECTORS.
0006 CI PLATE NORMALS, SHADOWED PLATES, ETC.
0007 CI
0008 INCLUDE 'SHACOU.FOR'
0009 DIMENSION INIT(NEX), III(3), IIN(3), VI(3), IX(3), ISI(3), ISII(3)
0010 DIMENSION XGB(3), XDC(3), VTCP(2), BTCP(4), VUCH(2), BTCH(4), DS(3)
0011 DIMENSION VVO(3), VVH(3), VVB(3), VVR(3), IBI(3), PVTRH1(3)
0012 LOGICAL LSHD, LSTD, LDS, LSTD, LSHD, LSTD
0013 LOGICAL LCGN, LIND, LDEBG, LTEST, LWARN
0014 COMMON/TSTC/LDEBG, LTEST, LWARN
0015 COMMON/LITC/SM, SN, SM, BB, BB
0016 COMMON/WAVE/WT, WL
0017 COMMON/LSKIP/LSTD(NEX)
0018 COMMON/GROUN/LGRID, NPXR
0019 IF(LDEBG) WRITE(6,667)
0020 667 FORMAT(/, ' DEBUGGING GEOM SUBROUTINE')
0021 CI DETERMINATION OF V, VH, AND VP UNIT VECTORS FOR EDGE-FIRED
0022 CI COORDINATE SYSTEM
0023 CI STEP THRU PLATES
0024 DO 100 WP=1, NPXR
0025 M= WP
0026 DO 16 ME=1, BEX
0027 M= M+1
0028 IF(NCIE.CT.FEX) M= M+1
0029 W=0.
0030 CALCULATE EDGE UNIT VECTOR V AND EDGE LENGTH VNAG
0031 DO 10 N=1,3
0032 VN= VN+V(N, ME, WP)-V(N, ME, WP)
0033 VNAG(ME, WP)=SRT(VN)
0034 DO 11 N=1,3
0035 VN= VN+V(N, ME, WP)-V(N, ME, WP)
0036 VNAG(ME, WP)=SRT(VN)
0037 CONTINUE
0038 IF(.NOT.LDEBG) GO TO 991
0039 DO 992 ME=1, BEX
0040 WRITE(6, *) (V(N, ME, WP), N=1, 3)
0041 CONTINUE
0042 CONTINUE
0043 CONTINUE
0044 CONTINUE
0045 CI CALCULATE PLATE UNIT NORMAL VH
0046 VH(1, WP)= 0.
0047 VH(2, WP)= 0.
0048 VH(3, WP)= 0.
0049 DO 22 ME=1, BEX
0050 VH= VH+V(1, ME, WP)*V(2, ME, WP)-V(2, ME, WP)+V(3, ME, WP)
0051 VH= VH+V(1, ME, WP)*V(2, ME, WP)-V(2, ME, WP)
0052 CONTINUE
0053 CONTINUE
0054 CONTINUE
203
DO 20 N=1,3
20 VNM=VNM+VII(N,MP)*VN(N,MP)
VNM=SQRT(VNM)
DO 21 N=1,3
21 VN(N,MP)=VNM(VII(N,MP),N=1,3)
IF(LDBUG) WRITE(6,+) (VN(N,MP),N=1,3)
C111 INSURE THAT ALL PLATES ARE FLAT. OTHERWISE ABORT!!
C111 TAKE DOT PRODUCT OF PLATE NORMAL AND EACH EDGE UNIT VECTOR
DO 120 ME=1,MEX
   DOT=VN(1,MP)+VN(2,MP)+VN(3,MP)
   ADOT=ABS(DOT)
   IF(ADOT.GT.0.01) GO TO 120
   MEX=MEX+1
   IF(MEX.GT.MEX) MEX=1
   WRITE(6,121) MP,MEX,ADOT
121 FORMAT(1 PLATE # ',I2,' IS NOT FLAT! CORNER # ',I2,' HAS ',
2 'PROBLEM,'/ WARP= ',F7.3,' PROGRAM ABORTS IF THE WARP'
3 ' IS GREATER THAN 0.03 ''''
   IF(ADOT.GT.0.03) STOP
120 CONTINUE
C111 CALCULATE UNIT BINORMAL VP WHICH IS IN PLATE PLANE
   AND PERPENDICULAR TO PLATE EDGE
C111 TAKE CROSS PRODUCT OF PLATE NORMAL AND EDGE VECTOR
DO 30 ME=1,MEX
   VP(1,ME,MP)=VN(2,MP)-VN(3,MP)+VN(1,MP)
   VP(2,ME,MP)=VN(3,MP)-VN(1,MP)+VN(2,MP)
   VP(3,ME,MP)=VN(1,MP)-VN(2,MP)+VN(3,MP)
30 IF(.NOT.LDBUG) GO TO 993
993 CONTINUE
DO 994 ME=1,MEX
994 WRITE(6,+) (VP(N,ME,MP),N=1,3)
995 CONTINUE
RETN
END
SUBROUTINE INITGF

This routine is used to initialize graphics each time an output is desired. Here, it zeroes out the previous array information and recalculates parameters based on the user-specified desired resolution.

0001 C-----------------------------------------------------------------------
0002 SUBROUTINE INITGF
0003 INCLUDE 'SHACOM.FOR'
0169 Clll
0170 Clll This subroutine initializes some graphics stuff.
0171 Clll Its function is to initialize things from one plot to the next,
0172 Clll but within the context of a single session.
0173 Clll
0174 INTEGER
0175 + I, J, INT
0176 Clll
0177 Clll Clear the character buffer.
0178 Clll
0179 DO 10 J=1, MAIROW
0180 DO 10 I=1, MAICOL
0181 10 OUTBUF(I, J) = '

0182 Clll
0183 Clll The number of rows & columns needed for internal representation is
0184 Clll calculated from the user-selected (or defaulted) angular ranges of
0185 Clll interest combined with the desired resolution in rad/pixel
0186 Clll
0187 ROWS = INT((PH2 - PH1) / RESPH + 0.6 ) + 1
0188 COLS = INT((THET2 - THET1) / RESTH + 0.6 ) + 1
0189 Clll
0190 Clll Calculate some parameters needed by the dynamic T increment
0191 Clll algorithms. The maximum allowable angular excursion is the
0192 Clll smaller of the number of radians in a single pixel of either theta
0193 Clll or phi.
0194 Clll
0195 ALPH = MIN( RESTH, RESPH )
0196 Clll
0197 RETURN
0198 END

205
SUBROUTINE PLAINT

This routines determines if a given ray strikes a plate.

0001 C
0002 C-------------------------------SUBROUTINE PLAINT(XIS,D,DHIT,MP,HIT)-------------------------------
0003 C1111
0004 C1111 DOES RAY HIT PLATE. IF MP=0 ALL PLATES ARE CHECKED.
0005 C1111 IF MH=MP THEN ONLY MP CHECKED AND SOURCE POSITION
0006 C1111 MOVED TO HIT POSITION IF RAY HITS MP.
0007 C1111 IF MH=MP, THEN ALL PLATES OTHER THAN MP ARE CHECKED.
0008 C1111
0009 C1111
0010 INCLUDE 'SHADOM.FOR.'
0176 DIMENSION IXS(3),D(3),IT(3),PVTX(3)
0177 LOGICAL LHT,LPAL,LCYL,LISTD,LTAN
0178 LOGICAL LGRND,LDEBUG,LIST,LWRN
0179 COMMON/TEST/LDEBUG,LTEST,LWRL
0180 COMMON/LIMIT/UL,LML,ULML,LML,LML
0181 COMMON/LPLC/LPLA,LCYL
0182 COMMON/HITPLT/MPH
0183 COMMON/GROUND/LGRND,MPH
0184 LHIT=.FALSE.
0185 DHIT=0.
0186 IF(.NOT.LPLA) RETURN
0187 C1111 STEP THRU PLATES
0188 DO 60 MP=1,MPH
0189 MP=MP
0190 IF(MP.EQ.MH) GO TO 60
0191 IF(MH.LT.0) MP=IABS(MH)
0192 C1111 IF TOTAL SHADING ALGORITHM IS BEING USED, HAS PLATE MP
0193 C1111 SHADOWED EVERY RAY TESTED?
0194 CXXXX IF(LSTS.AND..DOT.LSTD(MP)) GO TO 60
0195 MEX=MP(HP)
0196 AN=0.
0197 DO 6 H=1,3
0198 5 AN=AN+(IXS(HP)-IX(1,MP))*VX(1,MP)
0199 DO=D(1)+VX(1,MP)+D(2)+VX(2,MP)+D(3)+VX(3,MP)
0200 C1111 DOES RAY PASS THRU PLATE PLANE?
0201 IF(AN+DNU.EQ.0.) GO TO 60
0202 DO 10 H=1,3
0203 C1111 CALCULATE POINT WHERE RAY INTERSECTS PLATE PLANE
0204 10 XT(3)=IXS(3)+AN*D(3)/DN
0205 IF(NP.EQ.MP.AND.LGRND) GO TO 11
0206 DBT=0.
0207 C1111 IS HIT POINT ON PLATE?
0208 DO 30 ME=1,MEX
0209 MNE=ME+1
0210 IF(MNE.GT.MEX) MNE=1
0211 RD=0.
0212 DO 20 H=1,3
0213 20 RD=RD+((IX(H,MN,MP)-IT(H))*((IX(H,MN,MP)-IT(H))+((IX(H,MN,MP)-IT(H)))
0214 2-((IX(3,MN,MP)-IT(3))+(IX(3,MN,MP)-IT(3))+(IX(1,MN,MP)-IT(1)))
0215 CP=CP+VX(3,MP)+((IX(3,MN,MP)-IT(3))+((IX(3,MN,MP)-IT(3)))+(IX(1,MN,MP)-IT(1)))
0216 CP=CP+VX(3,MP)+((IX(1,MN,MP)-IT(1))+(IX(3,MN,MP)-IT(3))+(IX(1,MN,MP)-IT(1)))
0217 DBT=DBT+DBI
0220 DBI=SBAR2(CP,RD)
0221 DBI=DBI+DBI
0222 30 CONTINUE
IF(ABS(DBT).LT.PI) GO TO 60

C111 CALCULATE DISTANCE TO HIT (DHIT=SHORTEST DHT)

DHT=0.
DO 40 N=1,3

40 DHT=DHT+(XI(N)-XIS(N))*(XI(N)-XIS(N))

DHT=SQR(DHT)+SMLR
IF(LHIT.AND.(DHT.GT.DHIT)) GO TO 60

LHIT=.TRUE.
DHT=DHT
MPH=MP
IF(MH.GE.0) GO TO 60
DO 45 N=1,3

45 CI11 MOVE HIT POSITIONS AN INCREMENT TOWARDS SIDE OF PLATE

C111 WHICH SOURCE LIES ON

46 XIS(N)=XI(N)-SIGN(SMLR,AN)*VN(N,MP)

GO TO 61

50 CONTINUE

IF(MH.LT.0) GO TO 61

C111 IF TOTAL SHADOWING ROUTINE IS BEING USED, INDICATE
C111 THAT PLATE MP DOES NOT SHADOW SOURCE

C111 IF (LSTS) LSTD(MP)=.FALSE.

60 CONTINUE

IF(.NOT.LIEST) GO TO 62

WRITE(6,63)
63 FORMAT(/,' TESTING PLAIN SUBROUTINE')

WRITE(6,*) XIS
WRITE(6,*) D
WRITE(6,*) DHIT,MH,LHIT

62 RETURN

END
SUBROUTINE SCAN

This subroutine rasterizes a line in the character buffer according to its shading requirements. It calls routines to determine if a given point is shadowed or not and uses this information to shadow the given geometry. The fill character is used to fill the line in.

```
C-----------------------------------------------------------------------
SUBROUTINE SCAN (OBJ, LINE, PHI1, FILL, TYPE)
C!!!
C!!! This subroutine "rasterizes" a line in the character buffer
C!!! according to its shading requirements.
C!!! The fill character is used to fill the
C!!! line in. The line is declared larger character string in this
C!!! subroutine than in the calling routine. This can be "hardwired"
C!!! if it causes problems on other machines.
C!!!
0011 INCLUDE 'SHCOM.FOR'
0177 CHARACTER=600 LINE
0178 CHARACTER=1 FILL
0179 INTEGER PTR, OBJ
0180 INTEGER LSTPTR
0181 INTEGER SCAN
0182 INTEGER SPARC
0183 INTEGER PHI1
0184 INTEGER TYPE
0185 REAL DHIT
0186 REAL D(3)
0187 REAL DP(3)
0188 REAL XIS(3)
0189 REAL THETA, PHI
0190 LOGICAL EOL
0191 LOGICAL LHIT, LHIT1
0192 COMMON /SCANCHR/ PTR, EOL
0193 C!!!
0194 C!!! Initialize local variables.
0195 C!!!
0196 PTR = 1
0197 LSTPTR = 1
0198 EOL = PTR .GT. COLS
0199 C!!!
0200 C!!! Until the end of the line is scanned do ...
0201 C!!!
0202 100 IF (.NOT. EOL) THEN
0203 C!!!
0204 C!!! Locate the first occurrence of CHAR 7, 8, or EOL.
0205 C!!!
0206 PTR = SCANC (LINE )
0207 C!!!
0208 C!!! If point says it's a miss, update the least-pointer, span, scan.
0209 C!!! fill. Otherwise, fill in the characters between the pointers.
0210 C!!! Define the "source point" as the location of the antenna.
0211 C!!! and see if our plate shades the direction of the midpoint of the
0212 C!!! scan.
0213 C!!!
0214 THETA = (0.5*FLOAT(PTR-LSTPTR)-1.0)*RESTH*THET1
0215 PHI = (PHI1-1)*RESPH*PHI
0216 DP(1) = SIN(THETA)*COS(PHI)
0217 DP(2) = SIN(THETA)*SIN(PHI)
0218 DP(3) = COS(THETA)
0219 D(1) = DP(1)*VPC(1,1) + DP(2)*VPC(2,1) + DP(3)*VPC(3,1)
```
D(2) = DP(1)*VPC(1,2) + DP(2)*VPC(2,2) + DP(3)*VPC(3,2)
D(3) = DP(1)*VPC(1,3) + DP(2)*VPC(2,3) + DP(3)*VPC(3,3)

CIII

This must be done due to the behavior of plaint modifying XIS.

CIII

XIS(1) = ANTIEN(1)
XIS(2) = ANTIEN(2)
XIS(3) = ANTIEN(3)

CIII

Now do a case depending on what type of object we test for

CIII shadowing.

CIII

1 = plate
CIII 2 = elliptic cylinder

CIII

GOTO (1,2) TYPE

CIII

The object is a plate.

CIII

1

CALL PLAINT( XIS, D, DHIT, -OBJ, LHIT )
GOTO 999
CIII

CIII

The object is a cylinder. Test endcaps and cylinder bodies.

CIII

2

CALL CAPIENT( XIS, D, DHIT, 0, LHIT, -OBJ )
IF (.NOT. LHIT) CALL CYLINT(XIS,D,DHIT, LHT, .FALSE., -OBJ)
LHT = LHT OR. LHIT
GOTO 999

CIII

CIII

Take the appropriate action in the buffer.

CIII

999

IF (.NOT. LHIT) THEN
LSTPTR = PTR
PTR = SPANC( LINE )
DO 300 LSTPTR = LSTPTR, PTR-1, 1
LINE( LSTPTR:LSTPTR ) = FILL
300 CONTINUE
LSTPTR = PTR
ELSE
PTR = SPANC( LINE )
DO 400 LSTPTR = LSTPTR, PTR-1, 1
LINE( LSTPTR:LSTPTR ) = FILL
400 CONTINUE
LSTPTR = PTR
ENDIF

CIII

CIII End until

CIII

GOTO 100
CIII

END IF

CIII

RETURN
CIII

END
FUNCTION SCANC/SPANC

These functions are used to scan through the character buffer (map array) and locate/skip certain characters. They return the positions of these characters as their result.

0001 C-----------------------------------------------------------------------
0002 CIIII
0003 CIIII The following functions span/scan characters. That is, they
0004 CIIII return the position of next character in LINE which does or does
0005 CIIII not match the specified character. They also
0006 CIIII terminate the scan/span at the end of the line.
0007 CIIII
0008 INTEGER FUNCTION SCANC(LINE)
0009 INCLUDE 'SHACOM.FOR'
0100 CHARACTER*(*) LINE
0101 INTEGER PTR
0102 LOGICAL EOL
0103 COMMON /SCNCHR/ PTR, EOL
0104 CIIII
0105 CIIII Until a character matching CHARAC is found, advance the pointer.
0106 CIIII
0107 SCANC = PTR
0108 200 IF (.NOT. (EOL .OR. (LINE(SCANC:SCANC) .EQ. CHARAC))) THEN
0109 SCANC = SCANC + 1
0110 EOL = SCANC .GT. COLS
0111 GOTO 200
0112 ENDIF
0113 CIIII
0114 CIIII Until a character NOT matching ASCII 7 is found, advance the
0115 CIIII pointer.
0116 CIIII
0117 SPANC = PTR
0118 200 IF (.NOT. (EOL .OR. (LINE(SPANC:SPANC) .NE. CHARAC))) THEN
0119 SPANC = SPANC + 1
0120 EOL = SPANC .GT. COLS
0121 GOTO 200
0122 ENDIF
0123 CIIII
0124 CIIII Until a character NOT matching ASCII 7 is found, advance the
0125 CIIII pointer.
0126 CIIII
0127 RETURN
0128 END
This subroutine produces formatted and binary output of the shadow map.

SUBROUTINE WRTOUT

C-----------------------------------------------------------------------
VROUTIE
VROUT
INCLUDE 'SHACOM. FOR'
CIII
Clll
Thin
trobroutine
writrr
thr
formatted
output buffer to tho output
Clll
file. Start the output on a new page, and calculate
a header
Clll
Unit 7 is the main (ASCII) output file.
Clll
Initialize the width of the map to be printed.
Clll
Print map.

DO 20 J = 1, ROWS
   WRITE( 7, 100 ) ( ANTEI(N(I), I=1, 3, 1 ), INPFIL
   WRITE( 7, 200 ) ( (RESTH*(I-1) + THET1)*DPR , I= COLI, COLF, 10)
   WRITE( 7, 260 ) ( '*' , I= COLI, COLF, 10)
   DO 50 J = COLI - 1, ROWS
      WRITE( 7, 300 ) ( RESPH*(J-1)+PHI)*DPR
         ( OUTBUF(I,J) , I= COLI, COLF )
   50
   IF(COLF .LT. COLS) THEN
      COLI = COLF
      COLF = COLF + 90
   IF(COLF .GT. COLS) COLF = COLS
   GO TO 20
200
CIII
Have internal parameters available in degrees.
CIII

THET1D = THET1 *DPR
THET2D = THET2 *DPR
RESTHD = RESTH *DPR
PHID = PHI *DPR
PH2D = PH2 *DPR
RESPHD = RESPH *DPR
CIII
CIII
Unit 10 is a generic sort of binary output which can be plotted
CIII
anywhere. Place a little header info at the front of the file.
CIII

CIII
CIII
CIII
CIII
CIII
CIII
CIII

WRITE( 10 ) COLS, THET1D, THET2D, RESTHD
WRITE(10 ) ROWS, PHID, PH2D, RESPHD
CIII
CIII
CIII
CIII
CIII
CIII

DO 10 J = 1, ROWS
      WRITE(10 ) OUTBUF(I,J)
10
C111 Output stuff is complete.
C111
RETURN
C111 Format statements.
C111
100 FORMAT( 'I', 6X, 'ANTENNA (RCS) = ', 2(F8.4, ')'),
  + F8.4, ' IN METERS', 6X, 'INPUT SET: ', A42, / )
200 FORMAT( 'THETA (DEGREES)', /, 9X, 11( 4X, F9.2 ) )
250 FORMAT( 9X, 'PHI', 4X, A, 10( 9X, A ) )
300 FORMAT( 6X, F7.2, 3X, 101A )
END
Include file

This is a listing of the common blocks and parameter statements contained in the single include file for SHADOW. Note that the include file appears in the compiler listing for the interactive service routines.

C**  COMMON declarations...
C**
C**  COMMON /PIS/
C**    + PI,
C**    + DRI,
C**    + RPD
C**
C**  MAXIMUM DIMENSION FOR PLATES
C**    INTEGER NPX  PARAMETER (NPX=76)
C**  MAXIMUM DIMENSION FOR PLATE EDGES
C**    INTEGER NEI  PARAMETER (NEI=12)
C**  MAXIMUM DIMENSION FOR CYLINDERS
C**    INTEGER NCX  PARAMETER (NCX=6)
C**  MAXIMUM DIMENSION FOR CYLINDER RIMS
C**    INTEGER NHX  PARAMETER (NHX=10)
C**  MAXIMUM DIMENSION FOR ROWS (PHI)
C**    INTEGER MAIRW  PARAMETER (MAIRW=361)
C**  MAXIMUM DIMENSION FOR COLUMNS (THETA)
C**    INTEGER MAXCOL  PARAMETER (MAXCOL=181)
C**
C**  COMMON /GEOPLA/
C**    + XX (3,NEI,NPX),
C**    + V  (3,NEI,NPX),
C**    + VP (3,NEI,NPX),
C**    + VN (3,NPX),
C**    + MEP (RP2),
C**    + NPX
C**
C**  COMMON /GEOHEL/
C**    + AC (NHX,NCX),
C**    + BC (NHX,NCX),
C**    + ZC (NHX,NCX),
C**    + TCR (NHX,NCX),
C**    + XCL (3,NCX),
C**    + VCL (3,3,NCX),
C**    + NEC (NCX),
C**    + NCX
C**
C**  COMMON /EDMAG/ VMAG(NEX,NPX)
C**
C**  COMMON /SHADVN/ COLS, ROWS, ANTENN(3), CTGROD(3),
C**    + MP, NEXME, HC, TET1, TET2, PHI, PH2, RESLN, RESPH, ALPH,
C**    + UNIT(3), TRS(3), VEC(3,3), UNITF, UNITS, UNI
C**    + THDP, THDP, THDP, PHIP, FILPHM, FILCNM
C**  COMMON /SHADWC/ IPFIL, OUTFUS(MAXCOL, MAIRW),
C**    + FILLHC, FILLCH, FILLCH
C**
C**  COMMON /PACUT/ VFC(3,3)
C**
213
The first set of declarations is the stuff in /SHADOW/ common block

INTEGER
  + MP, ME, NEXTME, MG,
  + MP5M, FILCHM,
  + MP Plate/edge/cyl# variables.
    + FILPM, FILCPM,
  + MP Plate and cyl numbers for special filling
    + CDLS,
  + MP The size of the array subsection determined
    + ROWS
  + MP by internal resolution requirements.

REAL
  + CTROID.
  + MP A geometric center of the object in question.
    + ANTPN,
  + MP The antenna location in Ref Coord. System.
    + THET1,
  + MP The lower theta end of the range.
    + THET2,
  + MP The higher theta end of the range.
    + PHI,
  + MP The lower phi end of the range.
    + PH12,
  + MP The higher phi end of the range.
    + RESTH,
  + MP The desired theta/phi resolution
    + RESPH,
  + MP in units of radians/pixel.
    + ALPH
  + MP Maximum allowed angular excursion.

CHARACTER
  + OUTBUF*1.
  + MP The output buffer which is displayed.
    + INPFL*63,
  + MP The filename of the input set.
    + FILCHC,
  + MP special fill character for cylinders
    + FILCHP,
  + MP special fill character for everything else
    + FILCHR,
  + MP special fill character for plates
    DATA FILCHC, FILCHP, FILCHR / 'C', 'P', 'X' /

  + DATA FILCHC, FILCHP, FILCHR / 'C', 'P', 'X' /
  + MP From the /PI/ COMMON block...

  + MP REAL PI, TPI, DPR, RPD

  + MP From the /GESP/ COMMON block...

  + MP INTEGER
    + MEP,
  + MP Number of edges per plate
    + MPX
  + MP Total number of plates
    REAL
    + IX,
  + MP The array of plate corners
    + V,
  + MP Edge unit vectors
    + WP,
  + MP Edge unit binormals
    + VN
Unit normal for each plate

From the /GEMEL/ COMMON block...

INTEGER
  +  NEC,
  +  MCI
  +  MC1
C! Number of sections per cylinder
  +  NC1
CI Total number of cylinders
  +  AC,
REAL
  +  BC,
  +  ZC,
C! Elliptic parameter along x-axis
  +  BC,
  +  ZC,
C! Elliptic parameter along y-axis
  +  ZC,

C! Cylinder endcaps in cyl coord sys
  +  TCR,
C! Angle endcap makes with positive z axis
  +  ICL,
C! Cyl coord sys origin
  +  VCL,
C! Definition of cyl coord sys

INTEGER
  +  IUNIT
  +  UNITF,
  +  UNITS,
  +  UNITW,
  +  UNIT,
  +  TBS,
  +  THZP,PHZP,THIP,PHIP,
  +  VAS,
  +  VMAC
DATA UNIT/1..,3048,0.0254/

The following common block is for VNS/SMG$ software only.

COMMON /TERCOM/

KBDID, KETTBL
KBDID, KETTBL

215
10.4 Non-FORTRAN VAX/VMS source files

This section contains listings of the source files used by the interactive code which are not written in fortran. They are used by the interactive interface and are needed only by the VMS utilities.

CDU Source file

This file is the source input for the Command Language Definition Utility (CDU) which defines the available interactive commands.

```plaintext
module command_tables
ident /shacmd 01-000/
/**
 * FACILITY: Shadow
 *
 * ABSTRACT:
 *
 * This is the command language definition source for the SHADOW
 * program. It defines the interactive command interface under
 * the VAX/VMS operating system.
 *
 * AUTHOR: Laszlo Takacs
 *
 * CREATED: 1-NOV-1985
 *
 * MODIFIED BY:
 * 1-000 - Original, AAA 1-NOV-1985
 * 1-001 - Laszlo Takacs 20-DEC-1986
 * Added support for the SET FILL command and rearranged
 * the SET PLATE and SET CYLINDER commands.
 *
 * Show syntax
 *
 * Define syntax showfile syntax routine show_file
 * Define syntax showout syntax routine show_out
 * Define syntax showinp syntax routine show_inp
 * Define syntax showant syntax routine show_ant
 * Define syntax showcoo syntax routine show_coo
 * Define syntax showpat syntax routine show_pat
 * Define syntax showsca syntax routine show_sca
 * Define syntax showwin syntax routine show_win
 * Define syntax showkey syntax routine show_key
 *
 * Set syntax
 *
 * Define syntax setant syntax routine set_ant
 * Define syntax setcoo syntax routine set_coo
 * Define syntax setpat syntax routine set_pat
 * Define syntax setasca syntax routine set_sca
 * Define syntax setwin syntax routine set_win
 * Define syntax setkey syntax routine set_key
 * Define syntax setout syntax routine set_out
 * parameter p1 value( required )
 * parameter p2 value( type=file, required ),
 * prompt="filename"
 * qualifier plottable, default
```

216
Define syntax set_init_syntax routine set_init
parameter p1 value( required )
parameter p2 value( type=file, required )
prompt="input set"

Define syntax set_fill_syntax routine set_fill
parameter p1 value( required )
parameter p2 value( default="f" ), prompt="character"
qualified plate value( required, list ), nonnegative
qualified cylinder value( required, list ), nonnegative
qualified sequential nonnegative, syntax=sequential
disallow any2 ( plate, cylinder, sequential )

Define syntax sequential routine set_fill
parameter p1 value( required )
! noqualifiers

! Define syntax set_pla_syntax routine set_pla
! parameter p1 value( required )
! parameter p2 value( required ), prompt="plate number"
! parameter p3 value( default="P" ), prompt="character"
! qualified all syntax=set_pla_all

! Define syntax set_cyl_syntax routine set_cyl
! parameter p1 value( required )
! parameter p2 value( required ), prompt="cyl number"
! parameter p3 value( default="C" ), prompt="character"
! qualified all syntax=set_cyl_all

Define syntax set_pla_all
parameter p1
parameter p2 value( default="f" )

Define syntax set_uni_syntax
parameter p1 value( required )
parameter p2, value( required, type=units_types ),
prompt="inches, feet, or meters"

Define syntax set_uni_meters_syntax routine set_uni_meters
Define syntax set_uni_inches_syntax routine set_uni_inches
Define syntax set_uni_feet_syntax routine set_uni_feet

! Type definitions.

! Define type units_types
keyword inches, syntax = set_uni_inches_syntax
keyword meters, syntax = set_uni_meters_syntax
keyword feet, syntax = set_uni_feet_syntax

Define type set_types
keyword fill_character, syntax = set_fill_syntax
! keyword plate, syntax = set_pla_syntax
! keyword cylinder, syntax = set_cyl_syntax
keyword output_device, syntax = set_out_syntax
keyword input_set, syntax = set_init_syntax
keyword units, syntax = set_uni_syntax
keyword antenna_location, syntax = set_ant_syntax
keyword coordinates, syntax = set_coo_syntax
keyword pattern_cut, syntax = set_pat_syntax
keyword scale_factor, syntax = set_sca_syntax
keyword window, syntax = set_win_syntax

217
keyword keypad_mode, syntax = set_key_syntax, negatable

Define type show_types
    keyword fill_character, syntax = show_fill_syntax
    keyword plate, syntax = show_fill_syntax
    keyword cylinder, syntax = show_fill_syntax
    keyword output_device, syntax = show_out_syntax
    keyword input_set, syntax = show_inp_syntax
keyword units, syntax = show_uni_syntax
    keyword antenna_location, syntax = show_ant_syntax
keyword coordinates, syntax = show_coo_syntax
keyword pattern_cut, syntax = show_pat_syntax
keyword scale_factor, syntax = show_sca_syntax
keyword window, syntax = show_win_syntax
keyword keypad_mode, syntax = show_key_syntax

! Verb definitions.
!
Define verb set
    parameter pi, value( required, type=set_types ),
        prompt = "Set what"

Define verb show
    parameter pi, value( required, type=show_types ),
        prompt = "Show what"

Define verb help routine help_command
    parameter pi, value( type=rest_of_line )
        qualifier library, label = helplib, default,
        value( default="sys$disk:[]shadow" )

Define verb spawn synonym dcl
synonym $ routine dcl_command
    parameter pi, value( type=rest_of_line )

Define verb exit routine exit_command
Define verb shadow synonym s routine shadow_command

! End of file SHACMD.CLD.
!
|--
Keypad initialization file
This file defines the initial keypad assignments for the interactive program at run time. It may be modified to allow customizing of the keypad interface.

```
I+ I SHADOW.KPD - I
I This file starts up the keypad definitions for the SHADOW program. This is a user-definable file and may be altered.
I I Lassio Takacs, 20-DEC-1986
I+

I Set up the GOLD key.
I
Def/key/noecho PF1 ** /if=default /set=gold
Def/key/noecho PF1 ** /if=gold /set=default

I Help & Shadow.
I
Def/key/term/echo PF2 "Help"
Def/key/term/echo PF3 "Shadow"

I Set up the toggle keypad-mode key.
I
Def/key/term/echo PF4 "Set keypad" /if=default
Def/key/term/echo PF4 "Set No keypad" /if=gold

I Define miscellaneous keys.
I
Def/key/echo/if=default KP7 "Set output" /terminate
Def/key/echo/if=default KP8 "Set input" /terminate
Def/key/echo/if=default KP9 "Set antenna" /terminate
Def/key/echo/if=default MINUS "Set window" /terminate
Def/key/echo/if=default KP4 "Set scale_factor"/terminate
Def/key/echo/if=default KP6 "Set units" /terminate
Def/key/echo/if=default KP6 "Set coordinate"/terminate
Def/key/echo/if=default COMMA "Set pattern" /terminate
Def/key/echo/if=default KP1 "Set fill" /terminate
Def/key/echo/if=default KP2 "Set fill /plate=(1,1)" /terminate
Def/key/echo/if=default KP3 "Set fill /Sequential"/terminate
Def/key/echo/if=default KP0 "Spawn" /terminate

Def/key/echo/if=gold KP7 "Show output" /terminate
Def/key/echo/if=gold KP8 "Show input" /terminate
Def/key/echo/if=gold KP0 "Show antenna" /terminate
Def/key/echo/if=gold MINUS "Show window" /terminate
Def/key/echo/if=gold KP4 "Show scale_factor"/terminate
Def/key/echo/if=gold KP6 "Show units" /terminate
Def/key/echo/if=gold KP6 "Show coordinate"/terminate
Def/key/echo/if=gold COMMA "Show pattern" /terminate
Def/key/echo/if=gold KP1 "Show fill" /terminate
Def/key/echo/if=gold KP2 "Set fill /cylinder=(1,1)" /terminate
Def/key/echo/if=gold KP3 "Show fill" /terminate
Def/key/echo/if=gold KP0 "Spawn" /terminate

I Enter key is same as return. Period is EXIT.
```

219
Deik/key/term/echo PERIOD "Exit"
Deik/key/term/echo ENTER **

<table>
<thead>
<tr>
<th>End of SHADOW.KPD</th>
</tr>
</thead>
</table>
Chapter 11

VAX Implementation

This chapter describes the VAX/VMS implementation of the shadow program. The program has been split into two parts which are not used together. When the computer environment is the VAX/VMS operating system, then the more flexible interactive mode described in this chapter should be used. Assuming that the required files have been properly restored from the distribution medium, there are procedures provided to accomplish assembly of the code with minimum user effort.

11.1 Assembling the Code

On a VAX/VMS computer system, the following files are required to build and use the code. Both the interactive and non-interactive versions of the code can be run in any of the standard VMS ways, that is interactively, in a batch queue mode, or in a DCL subprocess. The actual building of the program takes place by invoking the procedure SHABLD.COM. The resulting executable file SHADOW.EXE can then be run with the RUN command.

SHABLD.COM A DCL command procedure to compile and link the files. This is the main assembly command file.

SHACMD.CLD A VMS Command Language Definition file used define the interactive commands available.

SHACOM.FOR The one include file for the code common blocks. The other include statements that appear in the code reference system libraries.

SHADNI.FOR This contains the alternate code that is to be used when a non-interactive code is desired.

SHADNW.FOR This contains code that is very much dependent on the facilities of VMS and has been separated as such. It is an essential part of the interactive program.

SHADOW.FOR This is the main body of the code and is common to both interactive and non-interactive versions. It is standard FORTRAN-77.

SHADOW.HLB This is the VMS-format help library containing descriptions and examples of interactive commands.
SHADOW.KPD This is an initialization file used by the interactive program to equivalence certain functions to keys of the user’s choice.

SHAPLT.COM This is a DCL command procedure invoking the NCAR graphics plotting software.

SHAPLT.FOR This is the FORTRAN program which reads the output produced by the code and calls appropriate NCAR routines to make a plot.

LABEL.DAT This file is read by the SHAPLT program in order to label the NCAR plots.

11.2 Running the Code

In order to run the code on VMS, the executable file created by the SHABLD procedure is necessary. The program is then run with the dcl RUN command.

A typical interactive session with the program might consist of the following elements in their approximate order of execution.

OUTPUT FILES Establish a set of output files with the SET OUTPUT command. The output files are of three types. Using the qualifiers of the SET OUTPUT command, any desired combination of output files may be generated.

PROCESS AN INPUT Issue a SET INPUT command which reads the geometry from the specified file. In order for the program to process input sets, this command must be issued prior to any mapping commands. This command is usually executed once per session.

DEFINE A WINDOW Using the SET WINDOW command, establish the angular range of interest. When the program begins, the size of the window is set to the full angular extent of the far-zone sphere. By specifying a smaller angular range, the user examines portions of the geometry in greater detail.

DEFINE A SOURCE With the SET ANTENNA command, establish the location of the source. This command is one of the more frequently entered commands. It applies units and coordinate transformations that apply from the set units and set coordinates command.

HIGHLIGHT ITEMS With the SET FILL command, the user may optionally cause parts of the geometry to be marked. This very usefull command may be executed at any time before a SHADOW command.

GENERATE A MAP Cause the generation of a shadow map by issuing a SHADOW command. The shadow command is used after the user has set all desired parameters including the window and the antenna location. Without executing this command, the code does not calculate any shadowing.

REPEAT ANY OF THE ABOVE Perform one or more of the above actions repeatedly to obtain several maps. Most of the commands above may be executed in any order provided that the SHADOW command is executed last.
EXIT Terminate the shadow session with an EXIT command. An acceptable alternate mode of exit is eof, or control-Z.

In order to make life easier by reducing the number of keystrokes required to enter interactive commands, a facility is provided with which the user may associate whole command strings with a single key. When the shadow program begins executing, it loads a set of predefined key definitions from a file. The user may edit this file to customize the keypad definitions to his/her liking. Since the file is loaded automatically, the only restriction on its use is that it must exist in the current process default directory and must be accessible at run time. The details about these interface routines and what they do may be found in the VAX/VMS Runtime Library Reference Manual.

11.3 Modifying the code

Modifications to the source code by the user can be performed, but of course the outcome cannot be predicted beforehand. One predictable user modification is changing the program's PARAMETER statements in the include file SHACOM.FOR. This would be necessary (and sufficient) to allow the program to deal with a greater number of plates or to construct a shadow map with greater resolution than the current maximum.
Chapter 12

Non-VAX Implementation

This chapter discusses how to implement the code on a different computer than a VAX. The obscuration code, SHADOW, has been separated into two main parts. The FORTRAN 77 part, is not VAX dependent and is contained in a file called SHADOW.FOR. Most of the rest of the files are VAX dependent and are used mostly for interactive features. Although, it is possible that other types of machines will have similar routines that will allow interactive manipulation, it is not possible here to suggest how this may be accomplished. It is assumed that the easiest way to use SHADOW on a non-VAX would be to run it in a non-interactive mode.

The main program in the default version of the file SHADOW.FOR is designed to be used with the non-FORTRAN 77 interactive version. A file called SHADNI.FOR contains a main program designed to be used in a non-interactive mode. It is listed in section 10.2. The main programs can be easily exchanged.

Note that the only other part of the code is this part that is non-FORTRAN 77 is the INCLUDE statement. This has been retained because many computer systems support this statement. It is used to include the lines of code in the named file in the spot that it is called as if the lines had been in that spot. It provides a powerful means of putting commonly defined parameters used throughout the code in one place. In this case, it is used to include the file SHACOM.FOR which contains COMMON blocks and PARAMETER statements that define the dimensions of arrays that store the geometry. If it is desired to increase the number of plates, edges per plate, cylinders, or rims per cylinder, etc; they can be changed in one spot. Please see the listing for this file elsewhere in this manual. The INCLUDE statement can be easily removed by hardwiring the contents of the file SHACOM.FOR into the text at the main program and the subroutines ABSCIN, CAPINT, CYLINT, CYLROT, DOCYLS, DOCYL, DOPLAS, DOPLA, GEOM, INITGF, PLAIN, SCAN, SCANC, SPANC, and WRTOUT.

The code can now be compiled, linked, and run. The user communicates with the code through the non-interactive commands. This allows almost the same capability. The only information that does not have a command to change its behavior is the fill options and the input and output file names. The fill options can be accessed through the main program. The listing below has comment lines referring to the place that the fill operations may be changed.

The input and output files can be named using assignments to the logical unit numbers for the given operation. The input file is read on logical unit #5. The echo file is written
on logical unit #6. The printable shadow map is written on logical unit #7. The plottable shadow map is written on logical unit #10. On a VAX the ASSIGN VMS command would be used.

Note that the user can specify more than one source. The non-interactive operation will run a shadow map for each source individually. The receiver will not be counted. If the user wants to look at the shadow map for a receiver, they should be treated in this code as if they are a transmitter (source).
Chapter 13

NCAR Plot Program

The shadow map can be plotted using graphical means. The SHADOW code will write an unformatted file that can be used for interfacing to special purpose plotting programs. It writes this file on logical unit #10. In the interactive mode the file name is specified by using the SET OUTPUT commands /PLOTTABLE option. In the non-interactive mode the file name is specified using an assign statement.

There are many ways to plot the resulting shadow map. Presently, there is little standardization between systems for plotting. This may change with the advent of GKS, but for now, it can not be assumed that different organizations have compatible plotting capabilities. This chapter suggests one possible means to plot the output. It uses the National Center for Atmospherics Research (NCAR) graphics package [5]. It has been tried on The Ohio State University ElectroScience Laboratory’s computer system and NASA Langley Research Center’s computer system, both VAX 11/780s, with almost the same results. It is still not possible, however, to assume that it will run everywhere the same way.

The program is listed for the convenience of possible users, knowing that some conversion may be necessary. The code is written in basic NCAR subroutine calls. Consult your local system information on how to link to your systems NCAR graphics subroutines. In addition, it is not written completely in standard FORTRAN 77. There are a few VAX extensions used, such as some of the options in the OPEN subroutine and some comment lines use the non-standard exclamation point. These changes will be minor.

Note that the plot of the shadow map will have grid lines. There is another option given for a map without grid lines. This can be used by commenting out the call to subroutine GRIDL, and uncommenting the call to subroutine PERIML.

The file name containing the maps to be plotted are placed in the first line of a file named LABEL.DAT. The LABEL.DAT file also contains the header information to be place at the top of the plot for future identification and reference. The code will loop through the specified shadow map file until all the shadow map contained in the file are plotted. A sample version of a LABEL.DAT file is given after the code listing. It shows a shadow map being read off of file FOR010.DAT which contains two shadow maps.
Listing of code to plot shadow map using NCAR:

```
0001 PROGRAM PLOTSU
0002 DIMENSION XDUM(2), YDUM(2), NC(6)
0003 INTEGER COLS, ROWS
0004 CHARACTER*80 LABELS(6), XLAB, YLAB, INF
0005 CHARACTER(*) XFORFU, YFORFU
0006 BYTE BYTE
0007 C
0008 C These are character parameters for the plotting output.
0009 C
0010 PARAMETER ( XFORFU = '(F6.1)' )
0011 PARAMETER ( YFORFU = '(F6.1)' )
0012 C
0013 DATA XLAB /' PHI /
0014 DATA YLAB /' THETA /
0015 DATA NC / 572 /
0016 C
0017 C Read a header from FOROS. Open the file readonly so that other users
0018 C can read it without needing write access to the file.
0019 C
0020 OPEN ( UNIT=5, TYPE='OLD', READONLY )
0021 READ ( 5, FMT='(A)' ) INF
0022 C
0023 C Read the header info from the data file. Open it unformatted.
0024 C
0025 OPEN(UNIT=10,FILE=INF,TYPE='OLD',FMT='UNFORMATTED',READONLY)
0026 C
0027 13 READ(10,END=9999) COLS,THET1D,THET2D,RESTD
0028 READ(10,END=9999) ROWS,PH1D,PH2D,RESPHD
0029 C
0030 ISCX = -2
0031 ISCY = -2
0032 XMIN = PH1D
0033 XMAX = PH2D
0034 YMIN = -THET1D
0035 YMAX = -THET2D
0036 SDX = 4
0037 SNI = 2
0038 SND = 4
0039 SNT = 2
0040 C
0041 C Read the label info for this plot.
0042 C
0043 READ ( 6, * ) LABELS(1)
0044 READ ( 6, * ) SI, ST, SZ, SPRX, SPRY, TRI
0045 READ ( 6, * ) ZTHET, ZPHI, ZTHET, ZPHI
0046 C
0047 C Format the labels for the plot (via internal write statements.)
0048 C
0049 WRITE (LABELS(2), 1100) SI, ST, SZ
0050 WRITE (LABELS(3), 1200) ZTHET, ZPHI, ZTHET, ZPHI
0051 WRITE (LABELS(4), 1300) SPRX, SPRY
0052 WRITE (LABELS(5), 1400) TRI
0053 C
0054 C CALL INFOPLT(2,EDUM,TDUM,XMIN, XMAX, YMIN, YMAX, ISCI,
0055 C * NDX,NDY,ISCY,NDY,XYLAB,5,YLAB,7,
0056 C * 6,LABELS,NC,0,-1,1)
0057 C
0058 C Define a mapping window from data to plot
0059 C
0060 C CALL SET (         
0061 + 0.12, 
0062 + 0.84, 
0063 + 0.12, 
227
```
C Do a linear-linear plot.
C A call to labmod might help the output look nicer.
CALL LABMOD (   
  XREF( XFORMA ), 
  YREF( YFORMA ), 
  LEN( XFORMA ),  
  1, 
  1, 
  0, 
  0, 
  0, 
  0 )
C Put labels on plot
C XMID=(XMIN+XMAX)/2.
YMAX+(6.0-IL)*YDEL
CALL PWRT(XL,YL,%REF(LABELS(IL)),NC(IL),1,0,-1)
C Define the perimeter of the plot with a grid.
CALL GRIDL (   
  NDX, ! Number of MAJOR
  NTX, ! Number of MINOR
  NDY, ! Number of MAJOR
  NTY ) ! Number of MINOR
C Theta and Phi Axis Labels
CALL PWRT(XMID,YBOT,%REF(XLAB),6,1,0,0)
XSID=XMID-6.0*XDEL
CALL PWRT(XSID,YMI,REF(YLAB),7,1,90,0)
IC Use this call if you don't want grid lines.
IC Define the perimeter of the plot.
IC
CALL PERIML (   
  NDX, ! Number of MAJOR
  NTX, ! Number of MINOR
  NDY, ! Number of MAJOR
  NTY ) ! Number of MINOR
XINC = 1.8
YINC = 0.9
ISYM = 1
C Loop on rows then on columns.
DO 10 J = 1, ROWS
  X = RESPHD*(J-1)*PHID
  1
DO 20 I = 1, COLS
READ (10, END=999) BYTE
IF (BYTE .NE. 32 ) THEN
  Y = -(RESTHD*(I-1)+THETID)
  CALL PLSTM(I, Y, XINC, YINC, ISTM)
C Plot the symbol on the page.
C
C CALL PWRT(
  + I, ! X coordinate
  + Y, ! Y coordinate
  + BYTE, ! The character to plot
  + I, ! Write one character
  + 0, ! Use the default size
  + 0, ! Use the default orientat
  + O ) ! Use the default centerin
END IF
20 CONTINUE
10 CONTINUE
C
C "Frame" the NCAR output.
C
999 CALL FRAME
GOTO 13
C
C Close the input file and stop.
C
999 CLOSE ( UNIT=10 )
CLOSE ( UNIT=6 )
STOP 'NCAR/Shadow plot completed.'
C
C Format go down here.
C
1100 FORMAT('ANTENNA LOCATED AT ',2(F7.1,','),F7.1)
1200 FORMAT('ANTENNA ORIENTATION: ',3(F7.1,','),F7.1)
1300 FORMAT('SOLAR PANELS ROTATED ',F7.1)
1400 FORMAT(' THERMAL RADIATORS ROTATED ',F7.1)
END
Listing of sample LABEL.DAT file:

FORO1O.DAT:
  '  SHADOW TEST1 FOR CASE ANES1'
  25. 15. 266.5 0. -52. 0.
  0. 0. 90. 0.
  '  SHADOW TEST2 FOR CASE ANES1'
  25. 15. 266.5 0. -52. 0.
  0. 0. 90. 0.
The Obscuration Code, referred to as SHADOW, is a user-oriented computer code to determine the cast shadow of an antenna in a complex environment onto the far zone sphere. The surrounding structure can be composed of multiple composite cone frustrums and multiple sided flat plates. These structural pieces are ideal for modeling space station configurations. The means of describing the geometry input is compatible with the NEC - Basic Scattering Code. In addition, an interactive mode of operation has been provided for DEC VAX computers.

The first part of this document is a User's Manual designed to give a description of the method used to obtain the shadow map, to provide an overall view of the operation of the computer code, to instruct a user in how to model structures, and to give examples of inputs and outputs. The second part is a Code Manual that details how to set up the interactive and non-interactive modes of the code and provides a listing and brief description of each of the subroutines.