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

NOTICE

FOR EARLY DOMESTIC DISSEMINATION

Because of its significant early commercial potential, this
information, which has been developed under a U.S. Gov-
ernment 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 ex-
press limitation that it not be published.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 ap-
proval 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
## Contents

1 User's Manual  
   1 Introduction  
   2 Method  
   3 Principle of Operation  
      3.1 Overview  
      3.2 Modeling the Structures  
      3.3 Running the Code  
      3.4 Non-Interactive Commands  
      3.5 Interactive Commands  
      3.6 Keypad Use  
   4 Non-Interactive Commands  
      4.1 Command CC: Cone Frustum Geometry  
      4.2 Command CG: Cylinder Geometry  
      4.3 Command CM: and CE: Comments  
      4.4 Command EN: End Program  
      4.5 Command GP: Ground Plane  
      4.6 Command NC: Next Set of Cylinders  
      4.7 Command NG: No Ground Plane  
      4.8 Command NP: Next Set of Plates  
      4.9 Command NS: Next Set of Sources  
      4.10 Command NX: Next Problem  
      4.11 Command PG: Plate Geometry  
      4.12 Command RT: Rotate-Translate Geometry  
      4.13 Command SG: Source Geometry  
      4.14 Command UF: Scale Factor  
      4.15 Command UN: Units of Geometry  
      4.16 Command US: Units of Source  
      4.17 Command VF: Far Zone Volumetric Pattern  
      4.18 Command XQ: Execute Code  
   5 Interactive Commands  
      5.1 Overview  
      5.2 Command Descriptions
10 Listings of the Code

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1</td>
<td>VAX/VMS Subroutines</td>
<td>126</td>
</tr>
<tr>
<td>10.1.1</td>
<td>MAIN PROGRAM</td>
<td>126</td>
</tr>
<tr>
<td>10.1.2</td>
<td>SUBROUTINE INIT</td>
<td>128</td>
</tr>
<tr>
<td>10.1.3</td>
<td>SUBROUTINE INTRAC</td>
<td>130</td>
</tr>
<tr>
<td>10.1.4</td>
<td>Interactive Service Routines</td>
<td>133</td>
</tr>
<tr>
<td>10.2</td>
<td>Non-VAX/VMS Subroutines</td>
<td>146</td>
</tr>
<tr>
<td>10.2.1</td>
<td>MAIN PROGRAM (non-interactive)</td>
<td>146</td>
</tr>
<tr>
<td>10.3</td>
<td>Subroutines common to both modes</td>
<td>148</td>
</tr>
<tr>
<td>10.3.1</td>
<td>SUBROUTINE ABSCLIN</td>
<td>148</td>
</tr>
<tr>
<td>10.3.2</td>
<td>FUNCTION BABS</td>
<td>187</td>
</tr>
<tr>
<td>10.3.3</td>
<td>BLOCK DATA</td>
<td>188</td>
</tr>
<tr>
<td>10.3.4</td>
<td>FUNCTION BTAN2</td>
<td>189</td>
</tr>
<tr>
<td>10.3.5</td>
<td>SUBROUTINE CAPINT</td>
<td>190</td>
</tr>
<tr>
<td>10.3.6</td>
<td>SUBROUTINE CYLINT</td>
<td>192</td>
</tr>
<tr>
<td>10.3.7</td>
<td>SUBROUTINE CYLROT</td>
<td>194</td>
</tr>
<tr>
<td>10.3.8</td>
<td>SUBROUTINE DOCYLS</td>
<td>195</td>
</tr>
<tr>
<td>10.3.9</td>
<td>SUBROUTINE DOCYL</td>
<td>196</td>
</tr>
<tr>
<td>10.3.10</td>
<td>SUBROUTINE DOPLAS</td>
<td>200</td>
</tr>
<tr>
<td>10.3.11</td>
<td>SUBROUTINE DOPLA</td>
<td>201</td>
</tr>
<tr>
<td>10.3.12</td>
<td>SUBROUTINE GEOM</td>
<td>203</td>
</tr>
<tr>
<td>10.3.13</td>
<td>SUBROUTINE INITGF</td>
<td>205</td>
</tr>
<tr>
<td>10.3.14</td>
<td>SUBROUTINE PLAINT</td>
<td>206</td>
</tr>
<tr>
<td>10.3.15</td>
<td>SUBROUTINE SCAN</td>
<td>208</td>
</tr>
<tr>
<td>10.3.16</td>
<td>FUNCTION SCANC/SPANC</td>
<td>210</td>
</tr>
<tr>
<td>10.3.17</td>
<td>SUBROUTINE WRTOUT</td>
<td>211</td>
</tr>
<tr>
<td>10.3.18</td>
<td>Include file</td>
<td>213</td>
</tr>
<tr>
<td>10.4</td>
<td>Non-FORTRAN VAX/VMS source files</td>
<td>216</td>
</tr>
<tr>
<td>10.4.1</td>
<td>CDU Source file</td>
<td>216</td>
</tr>
<tr>
<td>10.4.2</td>
<td>Keypad initialization file</td>
<td>219</td>
</tr>
</tbody>
</table>

11 VAX Implementation

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.1</td>
<td>Assembling the Code</td>
<td>221</td>
</tr>
<tr>
<td>11.2</td>
<td>Running the Code</td>
<td>222</td>
</tr>
<tr>
<td>11.3</td>
<td>Modifying the code</td>
<td>223</td>
</tr>
</tbody>
</table>

12 Non-VAX Implementation

13 NCAR Plot Program

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>NCAR Plot Program</td>
</tr>
</tbody>
</table>

iii
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

2.1 Geometry showing the projection of the plate edge onto the far zone sphere. 5
2.2 Geometry showing the projection of a cone frustum onto the far zone sphere. 7
2.3 Intersection of observation direction vector with plate. 8
2.4 The geometry for deciding whether a ray does or does not hit the plate. 10
2.5 Geometry illustrating the hit point on a cone frustum segment. 11

4.1 Definition of finite cylinder geometry composed of cone frustum segments with elliptic cross section. 26
4.2 Definition of finite elliptic cylinder geometry. 29
4.3 Definition of flat plate geometry. 35
4.4 Definition of rotate-translate coordinate system geometry. 37
4.5 Definition of source geometry for dipole antennas. 40
4.6 Definition of source geometry for aperture antennas. 41
4.7 Definition of volumetric pattern coordinate system. 46

7.1 Illustration of the Space Station 115
7.2 Three-axis view of the Space Station as modeled by the input set. 116
7.3 NCAR plot showing the shadow map of the space station model. 118
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 design 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.
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, \( \vec{R}_s \), to some position along its outer boundary, \( \vec{R}_t \), such that

\[
\vec{R} = \vec{R}_t - \vec{R}_s.
\]

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

$$\theta = \arctan \left( \hat{\rho} \cdot \hat{z} \cdot R \right)$$

and

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

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 $R = |\bar{R}_s - \bar{R}_e|$. 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

$$\bar{R}_{i+1} = \bar{R}_i + \delta \hat{t},$$

where $\hat{t}$ 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 mid point 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

$$\bar{R}_t = \bar{R}_s - \frac{[\hat{n} \cdot (\bar{R}_s - \bar{C}_t)]\hat{r}}{\hat{n} \cdot \hat{r}}.$$
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

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

which leads to the test, if

$$\left| \sum_{m=1}^{M} \theta_m \right| = \begin{cases} < \pi, & \text{no hit occurs} \\ > \pi, & \text{a hit occurs} \end{cases}$$

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

$$\hat{R}_c = \hat{R} + \hat{R}_s$$

or

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

The geometry is illustrated in Figure 2.5. The point defined by \( \hat{R}_c \) 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

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

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

$$\gamma = \frac{z_s \cos \phi \sin \theta}{a_j^2} + \frac{y_s \sin \phi \sin \theta}{b_j^2} - \tan \theta_j \cos \theta \lambda_j(z_s).$$
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
\[ \gamma = \frac{x^2}{a_j^2} + \frac{y_j^2}{b_j^2} - \lambda_j^2(z_s). \]

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 is between the finite length bounds of the frustum.

The basic theory discussed here is rather straightforward. 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 then 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 z-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 description 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 allows 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 3.3: VT100 keypad for SHADOW interactive commands.

<table>
<thead>
<tr>
<th>PF1</th>
<th>PF2</th>
<th>PF3</th>
<th>PF3</th>
</tr>
</thead>
<tbody>
<tr>
<td>gold</td>
<td>HELP</td>
<td>SHADOW</td>
<td>no keypad SET KEYPAD</td>
</tr>
<tr>
<td>7</td>
<td>show</td>
<td>SET OUTPUT</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>show</td>
<td>SET SCALE</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>show</td>
<td>SET FILL</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>SPAWN</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 \( x_c \)-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 \( x_c \)-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: \( NEC(MC) \)

where

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

---

READ: \( AC(NC,MC), BC(NC,MC), ZC(NC,MC) \)

where
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) 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 zc-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 xc-axis of the cylinder coordinate system as if it was a radial vector in the definition coordinate system.

Note that the new zc-axis and zc-axis must be defined orthogonal to each other. The new yc-axis is found from the cross product of the zc- and zc-axes.

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

AC(1,MC) This is a double dimensioned real variable which defines the radius of the MCth elliptic cylinder on the xc-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 yc-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: \( \text{IR(I), I}=1,36 \) (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

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4.0</td>
<td>:corners and type of plate</td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>:corner #1</td>
<td></td>
</tr>
<tr>
<td>-1.</td>
<td>:corner #2</td>
<td></td>
</tr>
<tr>
<td>-1.</td>
<td>:corner #3</td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>:corner #4</td>
<td></td>
</tr>
</tbody>
</table>

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:** \((\text{TR}(N), N=1,3)\)

---

**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:** \(\text{THZP, PHZP, THXP, PHXP}\)

---

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

where

\(XSS(N,MS)\) 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.

---

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

where

\(THSZ,PHSZ\) 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.

---

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

where

\(IMS(MS)\) 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.
a. Definition of pattern coordinate system.

b. Conic pattern cut, LCNPAT=TRUE., TPPD=THP.

c. Constant Phi pattern cut, LCNPAT=FALSE., TPPD=PHP.

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.

### 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.

### 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.

<table>
<thead>
<tr>
<th>FORMAT</th>
<th>HELP help-item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Qualifiers</td>
<td>Defaults</td>
</tr>
<tr>
<td>/LIBRARY[=library-name]</td>
<td>/LIB=SYS$DISK:[]SHADOW</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

/LIBRARY[=library-name]

/NOLIBRARY

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          Defaults
None.                       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 ANTENNA LOCATION**

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

**FORMAT**

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

**restrictions**

It is 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.
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.

### 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

The specified coordinate axes must be orthogonal to one another.

### prompts

Please input a translation vector in feet:

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

### command parameters

None.

### 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.

### EXAMPLES

**SHADOW> SET COOR**

Please input a translation vector in feet: 100, 200.300

Please input THZP, PHZP, THXP, PHXP in degrees: 0, -54, 266.5, 45

* The following rotations are used for ALL subsequent inputs: *

<table>
<thead>
<tr>
<th>Rotation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>VRS(1,1)</td>
<td>-0.70711</td>
</tr>
<tr>
<td>VRS(1,2)</td>
<td>-0.70711</td>
</tr>
<tr>
<td>VRS(2,2)</td>
<td>-0.70711</td>
</tr>
<tr>
<td>VRS(2,3)</td>
<td>0.00000</td>
</tr>
<tr>
<td>VRS(3,1)</td>
<td>0.00000</td>
</tr>
<tr>
<td>VRS(3,2)</td>
<td>0.00000</td>
</tr>
<tr>
<td>VRS(3,3)</td>
<td>1.00000</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>FORMAT</th>
<th>SET FILL [tag-character]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Qualifiers</td>
<td>Defaults</td>
</tr>
<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

59
visible. This mode can be very helpful when isolating particular parts of a geometry that are shadowing the source.

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>QUALIFIERS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>/SEQUENTIAL</td>
</tr>
<tr>
<td></td>
<td>/SEQUENTIAL</td>
</tr>
</tbody>
</table>

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.

/PLATE=num  
/PLATE=(num,char)

The /PLATE qualifier selects the third mode of obscuration, homogenous 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 "P" if unspecified.

This qualifier may not be specified in combination with other qualifiers. It is mutually exclusive with the /CYLINDER qualifier.

/CYLINDER=num  
/CYLINDER=(num,char)

The /CYLINDER qualifier selects the third mode of obscuration, homogenous 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 "C" if unspecified.

This qualifier may not be specified in combination with other qualifiers. It is mutually exclusive with the /PLATE qualifier.

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: _NLAO:[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>None.</td>
</tr>
<tr>
<td>Defaults</td>
<td>None.</td>
</tr>
</tbody>
</table>

restrictions | None.

prompts

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.

| COMMAND QUALIFIERS | None. |

**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.

FORMAT

SET OUTPUT filename

<table>
<thead>
<tr>
<th>Command Qualifiers</th>
<th>Defaults</th>
</tr>
</thead>
<tbody>
<tr>
<td>/PLOTTABLE</td>
<td>/PLOTTABLE</td>
</tr>
<tr>
<td>/PRINTABLE</td>
<td>/NOPRINTABLE</td>
</tr>
<tr>
<td>/ECHOING</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.

COMMAND QUALIFIERS

/PLOTTABLE
/NOPLOTTABLE

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

/PRINTABLE
/NOPRINTABLE

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.

/ECHOING
/NECHOING

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:[FOR006.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 NLA0: (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

```plaintext
SHADOW> SET PAT
Please input THZP,PHZP,THXP,PHXP in degrees: 0, -64, 265.5, 45, 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.

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
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.

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
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 0.00000000E+00 to 180.0000
with a resolution of 2.000000 degrees/pixel.
The current range of phi in degrees is 0.00000000E+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: .5
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 $0.6000000$ degrees/pixel.
The current range of phi in degrees is $46.00000$ 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.

**FORMAT**

SHOW COORDINATES

<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.

**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> SHO 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>

<table>
<thead>
<tr>
<th>restrictions</th>
<th>None.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>prompts</th>
<th>None.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>command parameters</th>
<th>None.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>DESCRIPTION</th>
<th>The output may be generated in one of three modes. For a detailed description of the possible modes, see the SET FILL command.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>COMMAND QUALIFIERS</th>
<th>None.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>EXAMPLES</th>
</tr>
</thead>
</table>

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.

command 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.
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. 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.

#### FORMAT

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

<table>
<thead>
<tr>
<th>restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>None.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>prompts</th>
</tr>
</thead>
<tbody>
<tr>
<td>None.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>command parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>None.</td>
</tr>
</tbody>
</table>

#### 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.

77
### 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. |

| command 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.

FORMAT

<table>
<thead>
<tr>
<th>SHOW UNITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Qualifiers</td>
</tr>
<tr>
<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>Defaults</td>
</tr>
<tr>
<td>None.</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 output files 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:

```
$ RUN SHADOW
SHADOW> SET OUT PLAEX1/NOPLOT/PRINT
SHADOW> SET INP PLAEX
SHADOW> SET UNI METERS
SHADOW> SET WIND
   90, 180
   1.0
   0., 360
   5.
SHADOW> SET ANT
SHADOW> 0,0,8
SHADOW> SHADOW
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>Antenna (RCE) = ( 0.0000, 0.0000, 0.0000 ) in Meters</th>
<th>Input Ext: UZ1E; (RJ.M.RAB.MAI)PLAX. INF,S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theta (Degrees)</td>
<td>90.00 100.00 110.00 120.00 130.00 140.00 150.00 160.00 170.00 180.00</td>
</tr>
<tr>
<td>PSI</td>
<td>*     *     *     *     *     *     *     *     *     *</td>
</tr>
<tr>
<td>0.00</td>
<td>X</td>
</tr>
<tr>
<td>5.00</td>
<td>X</td>
</tr>
<tr>
<td>10.00</td>
<td>X</td>
</tr>
<tr>
<td>15.00</td>
<td>X</td>
</tr>
<tr>
<td>20.00</td>
<td>X</td>
</tr>
<tr>
<td>25.00</td>
<td>X</td>
</tr>
<tr>
<td>30.00</td>
<td>X</td>
</tr>
<tr>
<td>35.00</td>
<td>X</td>
</tr>
<tr>
<td>40.00</td>
<td>X</td>
</tr>
<tr>
<td>45.00</td>
<td>X</td>
</tr>
<tr>
<td>50.00</td>
<td>X</td>
</tr>
<tr>
<td>55.00</td>
<td>X</td>
</tr>
<tr>
<td>60.00</td>
<td>X</td>
</tr>
<tr>
<td>65.00</td>
<td>X</td>
</tr>
<tr>
<td>70.00</td>
<td>X</td>
</tr>
<tr>
<td>75.00</td>
<td>X</td>
</tr>
<tr>
<td>80.00</td>
<td>X</td>
</tr>
<tr>
<td>85.00</td>
<td>X</td>
</tr>
<tr>
<td>90.00</td>
<td>X</td>
</tr>
<tr>
<td>95.00</td>
<td>X</td>
</tr>
<tr>
<td>100.00</td>
<td>X</td>
</tr>
<tr>
<td>105.00</td>
<td>X</td>
</tr>
<tr>
<td>110.00</td>
<td>X</td>
</tr>
<tr>
<td>115.00</td>
<td>X</td>
</tr>
<tr>
<td>120.00</td>
<td>X</td>
</tr>
<tr>
<td>125.00</td>
<td>X</td>
</tr>
<tr>
<td>130.00</td>
<td>X</td>
</tr>
<tr>
<td>135.00</td>
<td>X</td>
</tr>
<tr>
<td>140.00</td>
<td>X</td>
</tr>
<tr>
<td>145.00</td>
<td>X</td>
</tr>
<tr>
<td>150.00</td>
<td>X</td>
</tr>
<tr>
<td>155.00</td>
<td>X</td>
</tr>
<tr>
<td>160.00</td>
<td>X</td>
</tr>
<tr>
<td>165.00</td>
<td>X</td>
</tr>
<tr>
<td>170.00</td>
<td>X</td>
</tr>
<tr>
<td>175.00</td>
<td>X</td>
</tr>
<tr>
<td>180.00</td>
<td>X</td>
</tr>
<tr>
<td>185.00</td>
<td>X</td>
</tr>
<tr>
<td>190.00</td>
<td>X</td>
</tr>
<tr>
<td>195.00</td>
<td>X</td>
</tr>
<tr>
<td>200.00</td>
<td>X</td>
</tr>
<tr>
<td>205.00</td>
<td>X</td>
</tr>
<tr>
<td>210.00</td>
<td>X</td>
</tr>
<tr>
<td>215.00</td>
<td>X</td>
</tr>
<tr>
<td>220.00</td>
<td>X</td>
</tr>
<tr>
<td>225.00</td>
<td>X</td>
</tr>
<tr>
<td>230.00</td>
<td>X</td>
</tr>
<tr>
<td>235.00</td>
<td>X</td>
</tr>
<tr>
<td>240.00</td>
<td>X</td>
</tr>
<tr>
<td>245.00</td>
<td>X</td>
</tr>
<tr>
<td>250.00</td>
<td>X</td>
</tr>
<tr>
<td>255.00</td>
<td>X</td>
</tr>
<tr>
<td>260.00</td>
<td>X</td>
</tr>
<tr>
<td>265.00</td>
<td>X</td>
</tr>
<tr>
<td>270.00</td>
<td>X</td>
</tr>
<tr>
<td>275.00</td>
<td>X</td>
</tr>
<tr>
<td>280.00</td>
<td>X</td>
</tr>
<tr>
<td>285.00</td>
<td>X</td>
</tr>
<tr>
<td>290.00</td>
<td>X</td>
</tr>
<tr>
<td>295.00</td>
<td>X</td>
</tr>
<tr>
<td>300.00</td>
<td>X</td>
</tr>
<tr>
<td>305.00</td>
<td>X</td>
</tr>
<tr>
<td>310.00</td>
<td>X</td>
</tr>
<tr>
<td>315.00</td>
<td>X</td>
</tr>
<tr>
<td>320.00</td>
<td>X</td>
</tr>
<tr>
<td>325.00</td>
<td>X</td>
</tr>
<tr>
<td>330.00</td>
<td>X</td>
</tr>
<tr>
<td>335.00</td>
<td>X</td>
</tr>
<tr>
<td>340.00</td>
<td>X</td>
</tr>
<tr>
<td>345.00</td>
<td>X</td>
</tr>
<tr>
<td>350.00</td>
<td>X</td>
</tr>
<tr>
<td>355.00</td>
<td>X</td>
</tr>
<tr>
<td>360.00</td>
<td>X</td>
</tr>
<tr>
<td>365.00</td>
<td>X</td>
</tr>
<tr>
<td>370.00</td>
<td>X</td>
</tr>
<tr>
<td>375.00</td>
<td>X</td>
</tr>
<tr>
<td>380.00</td>
<td>X</td>
</tr>
</tbody>
</table>

86
7.2 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
   5.
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:

\[
\text{ANTENNA (RCS) } = (\ 0.0000, \ 0.0000, \ 0.0000 \ ) \ \text{IN METERS} \\
\text{INPUT SET: USER1:} \ \text{[B3M.NAS.MAN]PLAX33.INP} \\
\theta \text{ (DEGREES)} \\
\begin{array}{cccccccccc}
\text{PHI} & \text{0.00} & \text{5.00} & \text{10.00} & \text{15.00} & \text{20.00} & \text{25.00} & \text{30.00} & \text{35.00} & \text{40.00} \\
\text{0.00} & * & * & * & * & * & * & * & * & * \\
\text{5.00} & * & * & * & * & * & * & * & * & * \\
\text{10.00} & * & * & * & * & * & * & * & * & * \\
\text{15.00} & * & * & * & * & * & * & * & * & * \\
\text{20.00} & * & * & * & * & * & * & * & * & * \\
\text{25.00} & * & * & * & * & * & * & * & * & * \\
\text{30.00} & * & * & * & * & * & * & * & * & * \\
\text{35.00} & * & * & * & * & * & * & * & * & * \\
\text{40.00} & * & * & * & * & * & * & * & * & * \\
\end{array}
\]
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:

```plaintext
$ RUN SHADOW
SHADOW> SET OUT PLAEX3/HOPLOT/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:

```plaintext
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:

\[
\text{\textbf{ANTENNA (RCS) = ( 0.0000, 0.0000, 0.0000 ) IN METERS \quad INPUT file: USERS1:ceans.DMAT432FLEX.INP;6}}
\]

\[
\begin{array}{cccccccccccc}
\text{PHI} & 0.00 & 20.00 & 40.00 & 60.00 & 80.00 & 100.00 & 120.00 & 140.00 & 160.00 & 180.00 \\
\text{THETA (DEGREES)} & \ast & \ast & \ast & \ast & \ast & \ast & \ast & \ast \ast & \ast & \ast \\
0.00 & & & & & & & & & & \\
5.00 & & & & & & & & & & \\
10.00 & & & & & & & & & & \\
15.00 & & & & & & & & & & \\
20.00 & & & & & & & & & & \\
25.00 & & & & & & & & & & \\
30.00 & & & & & & & & & & \\
35.00 & & & & & & & & & & \\
40.00 & & & & & & & & & & \\
45.00 & & & & & & & & & & \\
50.00 & & & & & & & & & & \\
55.00 & & & & & & & & & & \\
60.00 & & & & & & & & & & \\
65.00 & & & & & & & & & & \\
70.00 & & & & & & & & & & \\
75.00 & & & & & & & & & & \\
80.00 & & & & & & & & & & \\
85.00 & & & & & & & & & & \\
90.00 & & & & & & & & & & \\
95.00 & & & & & & & & & & \\
100.00 & & & & & & & & & & \\
105.00 & & & & & & & & & & \\
110.00 & & & & & & & & & & \\
115.00 & & & & & & & & & & \\
120.00 & & & & & & & & & & \\
125.00 & & & & & & & & & & \\
130.00 & & & & & & & & & & \\
135.00 & & & & & & & & & & \\
140.00 & & & & & & & & & & \\
145.00 & & & & & & & & & & \\
150.00 & & & & & & & & & & \\
155.00 & & & & & & & & & & \\
160.00 & & & & & & & & & & \\
165.00 & & & & & & & & & & \\
170.00 & & & & & & & & & & \\
175.00 & & & & & & & & & & \\
180.00 & & & & & & & & & &
\end{array}
\]
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 define 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.,8.
  0.,0.,90.,0.
  -1.,0.5,0.
  1.,0.
VF: WINDOW SIZE
  0.,0.,90.,0.
  7,0.,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:

$ RUN SHADOW
SHADOW> SET OUT CYLEXI/HOPLOT/PRINT
SHADOW> SET INP CYLEXI
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:

CM: SIMPLE AIRCRAFT
CE: RCS INPUT SET
UN:
CC: FIRST CYLINDER
0.0.0.
g0.0.0.0.
2
1.1.1.1.
1.1.-1.
XQ:
ZN:
The output generated by the code was the following:

```
<table>
<thead>
<tr>
<th>ANTEENA (ACS) = ( 0.0000, 0.0000, 4.0000 ) IN METERS</th>
<th>INPUT SET: USER1;[EISN;NAR,MAH]CFLNEX1,1NP;4</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSI</td>
<td>13.00</td>
</tr>
<tr>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>5.00</td>
<td></td>
</tr>
<tr>
<td>10.00</td>
<td></td>
</tr>
<tr>
<td>15.00</td>
<td></td>
</tr>
<tr>
<td>20.00</td>
<td></td>
</tr>
<tr>
<td>25.00</td>
<td></td>
</tr>
<tr>
<td>30.00</td>
<td></td>
</tr>
<tr>
<td>35.00</td>
<td></td>
</tr>
<tr>
<td>40.00</td>
<td></td>
</tr>
<tr>
<td>45.00</td>
<td></td>
</tr>
<tr>
<td>50.00</td>
<td></td>
</tr>
<tr>
<td>55.00</td>
<td></td>
</tr>
<tr>
<td>60.00</td>
<td></td>
</tr>
<tr>
<td>65.00</td>
<td></td>
</tr>
<tr>
<td>70.00</td>
<td></td>
</tr>
<tr>
<td>75.00</td>
<td></td>
</tr>
<tr>
<td>80.00</td>
<td></td>
</tr>
<tr>
<td>85.00</td>
<td></td>
</tr>
<tr>
<td>90.00</td>
<td></td>
</tr>
<tr>
<td>95.00</td>
<td></td>
</tr>
<tr>
<td>100.00</td>
<td></td>
</tr>
<tr>
<td>105.00</td>
<td></td>
</tr>
<tr>
<td>110.00</td>
<td></td>
</tr>
<tr>
<td>115.00</td>
<td></td>
</tr>
<tr>
<td>120.00</td>
<td></td>
</tr>
<tr>
<td>125.00</td>
<td></td>
</tr>
<tr>
<td>130.00</td>
<td></td>
</tr>
<tr>
<td>135.00</td>
<td></td>
</tr>
<tr>
<td>140.00</td>
<td></td>
</tr>
<tr>
<td>145.00</td>
<td></td>
</tr>
<tr>
<td>150.00</td>
<td></td>
</tr>
<tr>
<td>155.00</td>
<td></td>
</tr>
<tr>
<td>160.00</td>
<td></td>
</tr>
<tr>
<td>165.00</td>
<td></td>
</tr>
<tr>
<td>170.00</td>
<td></td>
</tr>
<tr>
<td>175.00</td>
<td></td>
</tr>
<tr>
<td>180.00</td>
<td></td>
</tr>
<tr>
<td>185.00</td>
<td></td>
</tr>
<tr>
<td>190.00</td>
<td></td>
</tr>
<tr>
<td>195.00</td>
<td></td>
</tr>
<tr>
<td>200.00</td>
<td></td>
</tr>
<tr>
<td>205.00</td>
<td></td>
</tr>
<tr>
<td>210.00</td>
<td></td>
</tr>
<tr>
<td>215.00</td>
<td></td>
</tr>
<tr>
<td>220.00</td>
<td></td>
</tr>
<tr>
<td>225.00</td>
<td></td>
</tr>
<tr>
<td>230.00</td>
<td></td>
</tr>
<tr>
<td>235.00</td>
<td></td>
</tr>
<tr>
<td>240.00</td>
<td></td>
</tr>
<tr>
<td>245.00</td>
<td></td>
</tr>
<tr>
<td>250.00</td>
<td></td>
</tr>
<tr>
<td>255.00</td>
<td></td>
</tr>
<tr>
<td>260.00</td>
<td></td>
</tr>
<tr>
<td>265.00</td>
<td></td>
</tr>
<tr>
<td>270.00</td>
<td></td>
</tr>
<tr>
<td>275.00</td>
<td></td>
</tr>
<tr>
<td>280.00</td>
<td></td>
</tr>
<tr>
<td>285.00</td>
<td></td>
</tr>
<tr>
<td>290.00</td>
<td></td>
</tr>
<tr>
<td>295.00</td>
<td></td>
</tr>
<tr>
<td>300.00</td>
<td></td>
</tr>
<tr>
<td>305.00</td>
<td></td>
</tr>
<tr>
<td>310.00</td>
<td></td>
</tr>
<tr>
<td>315.00</td>
<td></td>
</tr>
<tr>
<td>320.00</td>
<td></td>
</tr>
<tr>
<td>325.00</td>
<td></td>
</tr>
<tr>
<td>330.00</td>
<td></td>
</tr>
<tr>
<td>335.00</td>
<td></td>
</tr>
<tr>
<td>340.00</td>
<td></td>
</tr>
<tr>
<td>345.00</td>
<td></td>
</tr>
<tr>
<td>350.00</td>
<td></td>
</tr>
<tr>
<td>355.00</td>
<td></td>
</tr>
<tr>
<td>360.00</td>
<td></td>
</tr>
</tbody>
</table>
```

94
7.6 Example 6: Two Elliptic Cylinders

This example is 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/NOPLDT/PRINT
SHADOW> SET INP CYLEX2
SHADOW> SET UNI METERS
SHADOW> SET WIND 130,180
0.56555556
0.,360
S.
!
! 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 ANS1 /PRINT/NOECHO
SHADOW> SET INP ANS1
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:

```
CH: ****************CASE ANS1**********
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.
```
4.5 -49.5 396.
4.5 -49.5 387.
4.5 49.5 387.

CM: UPPER KEEL
CE:
PG: -Y #1
4 0
4.5 -4.5 270.
4.5 -4.5 387.
-4.5 -4.5 387.
-4.5 -4.5 270.
PG: +Y #1
4 0
4.5 4.5 270.
-4.5 4.5 270.
-4.5 4.5 387.
4.5 4.5 387.

CM: LOWER KEEL & EXTENSION
CE:
PG: +X SIDE
12 0
4.5 22.5 0.
4.5 22.5 99.
4.5 4.5 99.
4.5 4.5 261.
4.5 -4.5 261.
4.5 -4.5 99.
4.5 -22.5 98.
4.5 -22.5 0.
4.5 -13.5 0.
4.5 -13.5 54.
4.5 13.5 54.
4.5 13.5 0.
PG: -Y #1
4 0
4.5 -22.5 0.
4.5 -22.5 99.
-4.5 -22.5 99.
-4.5 -22.5 0.
PG: -Y #2
4 0
4.5 -22.5 99.
4.5 -4.5 99.
-4.5 -4.5 99.
-4.5 -22.5 99.
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:
PC: BOTTOM
4 0
4.5 49.5 261.
4.5 -49.5 261.
-4.5 -49.5 261.
-4.5 49.5 261.
PC: +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.
PC: 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
PG: +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
PG: -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 -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 -89.
-1. 16.5 -7.
1. 16.5 -7.
CM: UPPER INBOARD SOLAR PANEL
GE:
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.

106
```
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
PG: +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
PG: -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
CM: UPPER OUTBOARD SOLAR PANEL
CE:
RT: +Y OUTBOARD
0. 132. 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: INSIDE 82
4 0
1. -16.5 89.
-1. -16.5 89.
-1. -16.5 7.
1. -16.5 7.
CM: LOWER OUTBOARD SOLAR PANEL
CE:
PG: -X 82X33
4 0
-1. 16.5 -89.
```
PC: LOWER 33
4 0
1. 16.5 -89.
1. -16.5 -89.
1. -16.6 -89.
1. -16.6 -89.

PC: UPPER 33
4 0
1. 16.5 -7.
1. 16.5 -7.
1. -16.5 -7.
1. -16.5 -7.

PC: INSIDE 82
4 0
1. -16.5 -89.
1. -16.5 -89.
1. -16.5 -89.
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 7.

PG: 82 INSIDE
The output generated by the code was the following:

```
ANTENNA (ECS) = ( 7.4200, 4.8720, 78.1812 ) IN METERS
INPUT SET: USERS1: [BJJN.MAB.MAN]ANG91.INP; 1
THETA (DEGREES)

0.00  20.00  40.00  60.00  80.00  100.00  120.00  140.00  160.00  180.00
18.00  38.00  58.00  78.00  98.00

PP
T
8.186 4.87
-40 40. 4.
XQ: EXECUTE CODE
EN: 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.
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-iteratively 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!!! electroscope laboratory. any problems or comments
0005 C!!! can be referred to:
0006 C!!!
0007 C!!! LASZLO A. TAKACS OR RONALD J. MARHEFKA
0008 C!!! ELECTROSCIENCE LABORATORY
0009 C!!! 1520 KINNEAR RD.
0010 C!!! COLUMBUS, OHIO 43212
0011 C!!! PHONE: (614) 422-6762 OR 422-5848
0012 C!!!
0013 C!!! This program provides a printer output of the geometrical
0014 C!!! shadow boundaries of a structure of plates and cylinders input
0015 C!!! as valid input sets to the numerical code.
0016 C!!!
0017 C!!! This program was written 15-JUN-1984.
0019 C!!!
0020 C!!!
0021 C!!! Beginning of the main routine.
```
0022 C111 Initialize any SHADOW data structures.
0023 C111
0024 C111 CALL INIT
0026 C111 Call the interactive terminal interface. This routine calls all
0027 C111 other subroutines.
0028 C111
0029 C111 CALL INTRAC
0030 C111
0031 C111 Finished.
0032 C111
0033 C111 END
SUBROUTINE INIT

0001  SUBROUTINE INIT
0002  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  C!!! THET1 = 0.0 * RPD
0190  C!!! THET2 = 180.0 * RPD
0191  C!!!
0192  C!!! The lower/higher phi end of the range.
0193  C!!!
0194  C!!! PH1 = 0.0 * RPD
0195  C!!! PH2 = 360.0 * RPD
0196  C!!!
0197  C!!! The desired theta/phi resolution in units of radians/pixel.
0198  C!!!
0199  C!!! RESTH = 2. * RPD
0200  C!!! RESPH = 2. * RPD
0201  C###
0202  C### Rotate translate default data RT:
0203  C###
0204  C###
0205  C###
0206  C###
0207  C###
0208  C###
0209  C###
0210  C###
0211  C###
0212  C###
0213  C###
0214  C###
0215  C###
0216  C###
0217  C###
0218  C###
0219  C###
0220  C###
0221  C###
0222  C###
0223  C###
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:

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

```fortran
C-----------------------------------------------------------------------
SUBROUTINE  INTRAC

| FACILITY:  INTERACTIVE TERMINAL COMMAND INTERFACE
| ABSTRACT:  This procedure prompts a terminal for input and parses/dispatches through CLI$ routines.
| ENVIRONMENT:  VAX/VMS Version 4.x
| AUTHOR:  Laszlo Takacs  CREATION DATE:  20-AUG-1986
| MODIFIED BY:

IMPLICIT  NONE
INCLUDE  '(RMSDEF)'
INCLUDE  '(SMCDEF)'
INCLUDE  'SHACOM.FOR'
EXTERNAL
COMMAND_TABLES,  ! User-defined com
GET_INPUT  ! The I/O routine at the b

INTEGER*4  STS,
READ_STS,
CLI$PRESENT,
CLI$DISPATCH,
CLI$DCL_PARSE,
CLI$GET_VALUE,
SMG$LOAD_KEY_DECLS,
SMG$CREATE_KEY_TABLE,
SMG$DELETE_VIRTUAL_KEYBOARD,
SMG$CREATE_VIRTUAL_KEYBOARD

COMMAND_TABLES

0001 C-----------------------------------------------------------------------
0002 SUBROUTINE  INTRAC
0003 |
0004 |**
0005 | FACILITY:  INTERACTIVE TERMINAL COMMAND INTERFACE
0006 |
0007 | ABSTRACT:
0008 |
0009 | ENVIRONMENT:  VAX/VMS Version 4.x
0010 |
0011 | AUTHOR:  Laszlo Takacs  CREATION DATE:  20-AUG-1986
0012 |
0013 | MODIFIED BY:
0014 |
0015 | 1-001 - Original,  LAT 20-AUG-1986
0016 |
0017 |
0018 | IMPLICIT  NONE
0019 | INCLUDE  '(RMSDEF)'
0020 | INCLUDE  '(SMCDEF)'
0021 | INCLUDE  'SHACOM.FOR'
0022 |
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 | SMG$LOAD_KEY_DECLS,
0035 | SMG$CREATE_KEY_TABLE,
0036 | SMG$DELETE_VIRTUAL_KEYBOARD,
0037 | SMG$CREATE_VIRTUAL_KEYBOARD
0038 |
0039 | CMake a key definiton table.
0040 |
0041 |
0042 |
0043 |
0044 |
0045 |
0046 |
0047 |
0048 |
0049 |
0050 |
0051 |
0052 |
0053 |
0054 |
0055 |
0056 |
0057 |
0058 |
0059 |
0060 |
0061 |
0062 |
0063 |
0064 |
0065 |
0066 |
0067 |
0068 |
0069 |

130
```
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 LIBISIGNAL( %VAL(STS) )

RETURN

END

-----------------------------------------------------------------------

INTEGER*4 FUNCTION GET_INPUT( COMMAND, PROMPT, LENGTH )

This routine does all the reading for the terminal interface.

It has the same calling format as LIBIGET_INPUT except that options

parameters may not be omitted.

INCLUDE '('$(RMSDEF)'

INCLUDE 'SHACOM.FOR'

EXTERNAL

CHARACTER(*) COMMAND,

PROMPT

INTEGER LENGTH*2

SMG$READ_COMPOSED_LINE*4

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

GET_INPUT = SMG$READ_COMPOSED_LINE ( KBDID,

RETTBL,

COMMAND,

PROMPT,

LENGTH )

IF ( GET_INPUT .EQ. $LOC( SMG$_EOF ) ) GET_INPUT = RMS$_EOF

RETURN

END

-----------------------------------------------------------------------

INTEGER*4 FUNCTION PUT_OUTPUT ( STRING )

This routine does all the writing for the terminal interface.

It has the same calling format as LIBIPUT_OUTPUT.

-----------------------------------------------------------------------
INCLUDE 'SHACON.FOR'

0173 CHARACTER(*) STRING
0174 INTEGER*4
0175 INTEGER*4
0176 * LIB$PUT_OUTPUT
0177 !
0178 ! Read a line.
0179 !
0180 PUT_OUTPUT = LIB$PUT_OUTPUT ( STRING )
0181 !
0182 ! There should be no errors here. Signal if there are any.
0183 !
0184 IF (.NOT. PUT_OUTPUT) CALL LIB$SIGNAL( %val(PUT_OUTPUT) )
0185 !
0186 ! Return.
0187 !
0188 RETURN
0189 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|...........................................................................................................
0002 C|
0003 C| The system-dependent stuff goes below here.
0004 C|
0005 C|...........................................................................................................
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:
0016 C| 0016 C| ret-status.wlc.de = routine ( )
0017 C|
0018 C| FORMAL PARAMETERS:
0019 C|
0020 C| NONE
0021 C|
0022 C| IMPLICIT INPUTS:
0023 C|
0024 C| FUNCTION SPECIFIC
0025 C|
0026 C| IMPLICIT OUTPUTS:
0027 C|
0028 C| FUNCTION SPECIFIC
0029 C|
0030 C| COMPLETION STATUS:
0031 C|
0032 C| FUNCTION SPECIFIC
0033 C|
0034 C| SS$NORMAL Success, or
0035 C| fact_status some other status
0036 C|
0037 C| SIDE EFFECTS:
0038 C|
0039 C| VARIABLE
0040 C|--
0041 C| INTEGER FUNCTION SERVICE_ROUTINES
0042 C| IMPLICIT NONE
0043 C| PARAMETER SUCCESS = 1
0044 C| INCLUDE '($SSDEF)/NOLIST' Include system status definitions
1035 C| INCLUDE 'SHADOW.FOR/LIST' Include SHADOW common block
1036 1 C| COMMON declarations...
1037 1 C|COMMON /PIS/
1038 1 C| + PI,
1039 1 C| + TPI,
1040 1 C| + DPR,
1041 1 C| + RPD
1042 1 C|--------
```
C+++ MAXIMUM DIMENSION FOR PLATES
1046 1 INTEGER NPI
1047 1 PARAMETER (NPI=76)
1048 1 C+++ MAXIMUM DIMENSION FOR PLATE EDGES
1049 1 INTEGER NEX
1050 1 PARAMETER (NE=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 NHX
1056 1 PARAMETER (NHX=10)
1057 1 C+++ MAXIMUM DIMENSION FOR ROWS (PHI)
1058 1 INTEGER MAROW
1059 1 PARAMETER (MAROW=361)
1060 1 C+++ MAXIMUM DIMENSION FOR COLUMNS (THETA)
1061 1 INTEGER MAXCOL
1062 1 PARAMETER (MAXCOL=181)
1063 1 C
1064 1 COMMON /GEPLA/
1065 1 + XX (3,NEX,NPI),
1066 1 + V (3,NEX,NPI),
1067 1 + VP (3,NEX,NPI),
1068 1 + VN (3,NPI),
1069 1 + NEP (NPI),
1070 1 + NPI
1071 1 C
1072 1 COMMON /GEMHEL/
1073 1 + AC (NHX,NCX),
1074 1 + BC (NHX,NCX),
1075 1 + ZC (NHX,NCX),
1076 1 + TCR (NHX,NCX),
1077 1 + XCL (3,NCX),
1078 1 + VCL (3,3,NCX),
1079 1 + NEC (NCX),
1080 1 + NCX
1081 1 C
1082 1 COMMON /EDMAG/ VMAG(NEX,NPI)
1083 1 C
1084 1 COMMON /SHADOW/ COLS, ROWS, ANIENH(3),CTROID(3),
1085 1 + MP,ME,NEITME,MC,
1086 1 + THE1,THE2,PHI,PK2,RESTH,RESPH,ALPH,
1087 1 + UNIT(3),TRS(3),VTS(3,3),UNITF,UNIT,UNITF,UNIT,
1088 1 + THZP,PHZH,THZH,PHZH,FIPLNM,FIPLCN
1089 1 COMMON /SHADOWC/ INPFIL,OUTBUF(MAXCOL,MAROW),
1090 1 + FILCMC,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, MC,
1098 1 C Plate# /edge# /cyl# variables.
1099 1 + FILPNM, FILPNM,
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.
1100 1 + ANTEHN,
1110 1 CI The antenna location in Ref Coord. System.
1111 1 + TET1,
1112 1 CI The lower theta end of the range.
1113 1 + TET2,
1114 1 CI The higher theta end of the range.
1115 1 + PHI,
1116 1 CI The lower phi end of the range.
1117 1 + PH2,
1118 1 CI The higher phi end of the range.
1119 1 + RESTH,
1120 1 CI The desired theta/phi resolution
1121 1 + RESPH,
1122 1 CI in units of radians/pixal.
1123 1 + ALPH
1124 1 CI Maximum allowed angular excursion.
1125 1
1126 1 CHARACTER
1127 1 + OUTBUF=1,
1128 1 CI The output buffer which is displayed.
1129 1 + INPBUF=63,
1130 1 CI The filename of the input set.
1131 1 + FILNG,
1132 1 CI special fill character for cylinders
1133 1 + FILCH1,
1134 1 CI special fill character for everything else
1135 1 + FILCH2
1136 1 CI special fill character for plates.
1137 1 DATA FILCH1, FILCH2, FILCHR / 'C', 'P', 'I' /
1138 1
1139 1 CI!!! From the /PIS/ COMMON block...
1140 1 CI!!!
1141 1 REAL PI, TPI, DPR, RPD
1142 1 CI!!! From the /COSPLA/ COMMON block...
1143 1 CI!!!
1144 1 CI!!!
1145 1 INTEGER
1146 1 + MEL,
1147 1 CI Number of edges per plate
1148 1 + MX
1149 1 CI Total number of plates
1150 1 REAL
1151 1 + IX,
1152 1 CI The array of plate corners
1153 1 + V,
1154 1 CI Edge unit vectors
1155 1 + VP,
1156 1 CI Edge unit binormals
1157 1 + VN
1158 1 CI Unit normal for each plate
1159 1 CI!!! From the /GEODEL/ COMMON block...
1160 1 CI!!!
1161 1 CI!!!
1162 1 INTEGER
1163 1 + MEC,
1164 1 CI Number of sections per cylinder
1165 1 + MXI
1166 1 CI Total number of cylinders
1167 1 REAL
1168 1 + AC,
1169 1 CI Elliptic parameter along x-axis
1170 1 + BC,
1171 1 CI Elliptic parameter along y-axis
1172 1 + EC,
Cylinder endcaps in cyl coord sys

Cyl coord sys origin

Definition of cyl coord sys

INTEGER

REAL

REAL

INTEGER

COMMON /TERCON/ KBDID, KEYTBL

EXTERNAL

PUT_OUTPUT, GET_INPUT, ! My own $SG-type I/O routines

CLI$_PRESEN T, !

CLI$-_NEGATED, !

CLI$-_LOCRES, ! locally present

CLI$-_LOCNEGED, ! locally negated

CLI$-_DEFAULTED,

CLI$-_ABSENT,

CLI$-_IVALEU

CHARACTER

P1$=80, ! Command line variable

P2$=80,

UNCHAR=1, ! A character

LIBRARY=64, ! Name of the help library is defa

LABEL(3)=6 ! Units label

FILE=*50, ! Temporary file variable

PRFIL=50, ! Printable file

LISFIL=50, ! Input echo listing

OUTFIL=50 ! "Plottable" output file

DATA IUNIT/I/

LOGICAL=6

VALID_INPUT, ! A loop control variable

CLI$PRESEN T, ! CLI interface to get info about

CLI$-_GET_VALUE ! CLI interface to get info about

REAL=4

DOT,DEI,XQ(3)

INTEGER=4

N,NI,NJ,STS, ! sordid variables...

KEYPAD, ! Keypad condition flag

! General library routines
†+ LIB$SPAVN,  ! Executes a subprocess
   + LBNXOUTPUT_HELP,  ! The librarian help routine
   + SMG$SET_KEYPAD_MODE,  ! Screen management package

   † "SET/SHOW" routines
   †
   + SET抑え,  SET_OUT,  SET_COD,  !
   + SET_pat,  SET_SCA,  SET_WIN,  !
   + SET_key,  SET_Inp,  SET_UNI_ME  !
   + SET_uni_inches,  SET_uni_feet,  !
   + SHOW_ant,  SHOW_OUT,  SHOW_COD,  !
   + SHOW_pat,  SHOW_SCA,  SHOW_WIN,  !
   + SHOW_key,  SHOW_inp,  !
   + SHOW_uni,  !

   † various command routines
   †
   + EXIT_command,  HELP_command,  DCL_command,  SHADOW_command
   +

   ENTRY SET_fil
   IF ( CLI$PRESENT('SEQUENTIAL') ) THEN
   C1
   FILPNM = -1
   FILCNM = -1
   FILCHP = 'P'
   FILCHC = 'C'
   FILCHR = 'X'

   C1
   To avoid screwing up the text in SCAN, use a character that will
   C1 not be used by the fill process, like char 7.
   C1 Set a plate up for tagging.
   C1
   ELSEIF ( CLI$PRESENT('PLATE') ) THEN
   C1
   FILCNM = 0
   FILCHC = 'C'

   C1
   Get the master fill character.
   C1
   CALL CLI$GET_VALUE('P2', FILCHR)
   C1
   Get the qualifier numeral value.  STS is being used for the length of
   C1 and the status of the decode.
   C1
   IF ( CLI$GET_VALUE('PLATE', P2, STS ) ) THEN
   C1
   DECODE (STS,P1,ISTAT=STS) FILPNM
   C1
   ELSE
   C1
   STS = -1
   C1
   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 = $LOC( CLI$IVvalu )
ELSE IF (.NOT. CLI$GET_VALUE('PLATE', FILCHP)) THEN
    FILCHP = 'P'
ENDIF
ENDIF

C1 Set a cylinder up for tagging.

ELSEIF (.NOT. CLI$PRESENT('CYLINDER')) THEN
    C1 Clear any cylinder tagging residue.
    FILPNM = 0
    FILCHP = 'P'
    C1 Get the master fill character.
    C1 CALL CLI$GET_VALUE('P2', FILCHR)
    C1 Get the qualifier numeral value. STS is being used for the length of
    C1 and the status of the decode.
    C1 IF (CLI$GET_VALUE('CYLINDER', P2, STS)) THEN
        DECODE (STS,1,P2,ISTAT=STS) FILCNM
    ELSE
        STS = -1
    ENDIF
    C1 Get the fill character for that cylinder. Use a 'C' if none is given
    C1 IF (STS .NE. 0) THEN
        SET_FIL = ILSQ( CLI$_IVALU )
        ELSE
            IF (.NOT. CLI$GET_VALUE('CYLINDER', FILCHC)) THEN
                FILCHC = 'C'
            ENDIF
        ENDIF
    C1 The else here is for a "SET FILL [x]" command.
    C1 ELSE
    C1 Get the master fill character.
    C1 CALL CLI$GET_VALUE('P2', FILCHR)
    C1 End of cases.
    C1 ENDIF
    C1 GOTO 3
    1 FORMAT( I )
    C1 This routine displays the current fill characters being used for plat
    C1 or cylinders.
    C1 ENTRY SHOW_FIL
    C1 Assume success only when the SHOW command is being executed.
    C1 SHOW_FIL = SUCCESS

C1
CI Examine the plate situation.

IF (FILPNM.GT.0) THEN
  WRITE(2,FMT=('"Plate ",I3," is tagged with ",A,"')
  + FILPNM, FILCHP
  WRITE(2,FMT=('" All other geometry tagged with ",A,"')
  + FILCHR
ENDIF

CI Examine the cylinder situation.

IF (FILCNM.EQ.0) THEN
  WRITE(2,FMT=('"Cylinder ",I3," tagged with ",A,"')
  + FILCNM, FILCHC
  WRITE(2,FMT=('" All other geometry tagged with ",A,"')
  + FILCHR
ENDIF

CI Check on a no-tag background character situation.

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.

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=('"Input antenna location in ",A6," : ",A")
  + LABEL(IUN1T)
  READ (1,*), ANTENN(1), ANTENN(2), ANTENN(3)
ENTRY

CI Perform appropriate units conversion here.

DO 3424 N=1,3
  XQ(N)=ANTENN(N)
SET-ANT SUCCESS
END DO

CI This routine displays the current antenna position.

ENTRY SHOW_ANT

DO 1111 N=1,3
  XQ(N) = ( (ANTENN(N)-TRS(N)) + VRS(N) ) / UNITS
SET_ANT = SUCCESS
END DO

CI Transform the antenna back

DO 1111 N=1,3
  IQ(N) = ( (ANTENN(N)-TRS(N)) + VRS(N) ) / UNITS
END DO
WRITE(2,FMT=('"Antenna in RCS (meters): ",3F12.6') ANTEH
WRITE(2,FMT=('"Definit system ("A,"'): ",3F12.6')
+ LABEL(IUNIT), AN
SHOW_ANT = SUCCESS
RETURN
C! Process a new input set. Inquire about the full name.
C!
ENTRY SET_INP
CALL CLI$GET_VALUE( 'P2', FILE )
OPEN ( UNIT=5, FILE=FILE, DEFAULTFILE='.INP', STATUS='OLD')
CALL ABSCIN
SET_INP = SUCCESS
C!
ENTRY SHOW_INP
INQUIRE ( UNIT=6, NAME=INPFIL )
TYPE *, 'The current input set is ', INPFIL
SHOW_INP = SUCCESS
RETURN
C!
ENTRY SHOW_KEY
IF ( .NOT. CLI$PRESENT( 'KEYPAD_MODE' ) ) THEN
KEYPAD = 0
ELSE
KEYPAD = 1
ENDIF
SET_KEY = SM$SET_KEYPAD_MODE( KBDID, KEYPAD )
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.'
ENDIF
RETURN
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 3921,LABEL(IUNIT)
3921 FORMAT( 'Please input a translation vector in ',A6,':' )
accept*, (TRS(N),N=1,3)
DO 3920 N=1,3
3920 TRS(N)=TRS(N)+UNITS
C##
THZP,PHZP=ORIENTATION OF THE VHS(3,9) AXIS RELATIVE TO THE
C## FIXED COORDINATE SYSTEM.
THXP,PHXP=ORIENTATION OF THE VRS(1,N) AXIS RELATIVE TO THE FIXED COORDINATE SYSTEM.

continue

type*, 'Please input THXP,PHXP in degrees:'

accept*, THXP,PHXP

VPC(3,1)=SIN(THXP*RDP)*COS(PHXP*RDP)
VPC(3,2)=SIN(THXP*RDP)*SIN(PHXP*RDP)
VPC(3,3)=COS(THXP*RDP)
VPC(1,1)=SIN(THXP*RDP)*COS(PHXP*RDP)
VPC(1,2)=SIN(THXP*RDP)*SIN(PHXP*RDP)
VPC(1,3)=COS(THXP*RDP)

THZP,PHZP=ORIENTATION OF THE VRS(3,N) AXIS RELATIVE TO THE FIXED COORDINATE SYSTEM.

continue

type*, 'Please input THZP,PHZP in degrees:'

accept*, THZP,PHZP

VPC(3,1)=SIN(THZP*RDP)*COS(PHZP*RDP)
VPC(3,2)=SIN(THZP*RDP)*SIN(PHZP*RDP)
VPC(3,3)=COS(THZP*RDP)
VPC(1,1)=SIN(THZP*RDP)*COS(PHZP*RDP)
VPC(1,2)=SIN(THZP*RDP)*SIN(PHZP*RDP)
VPC(1,3)=COS(THZP*RDP)

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

DXY=VPC(1,1)*VPC(1,1)+VPC(1,2)*VPC(1,2)+VPC(1,3)*VPC(1,3)
DXY=SQRT(DXY)
VPC(1,1)=VPC(1,1)/DXY
VPC(1,2)=VPC(1,2)/DXY
VPC(1,3)=VPC(1,3)/DXY

ENTRY SET-PAT
THZP,PHZP=ORIENTATION OF THE VPC(3,N) AXIS RELATIVE TO THE FIXED COORDINATE SYSTEM.
THXP,PHXP=ORIENTATION OF THE VPC(1,N) AXIS RELATIVE TO THE FIXED COORDINATE SYSTEM.

ENTRY SHOW-COO

C1 Display the coordinates
ENTRY SHOW_COO

FORMAT(2H*,IX,'The following rotations are used for ALL',
2' subsequent inputs:','T70,1H*)
DO 3932 NI=1,3
WRITE(6,3933) (NI,NJ,VPC(NI,NJ),NJ=1,3)
FORMAT(2H*,IX,3(2X,'VPC(',II,',',II,')=',F9.6),1H*)
VPC(1,3) = \cos(\theta \times \text{RPD})

VPC(1,I) is PERPENDICULAR TO VPC(3,I)

DZX = VPC(3,I) \times VPC(1,1) + VPC(3,2) \times VPC(1,2) + VPC(3,3) \times VPC(1,3)

IF (ABS(DZX) < 0.01) THEN
  TYPE*, 'The coordinates are NOT orthogonal - Respecify.'
goto 1234
ELSE
  VPC(1,1) = VPC(1,1) - VPC(3,1) \times DZX
  VPC(1,2) = VPC(1,2) - VPC(3,2) \times DZX
  VPC(1,3) = VPC(1,3) - VPC(3,3) \times DZX
  VPC(1,1) = VPC(1,1) / DOT
  VPC(1,2) = VPC(1,2) / DOT
  VPC(1,3) = VPC(1,3) / DOT
END IF

WRITE(6,3931)

C

ENTRY SHOW-PAT
DO N=1,3
  WRITE(6,4935) (NI,NJ,VPC(NI,NJ),NJ=1,3)
END DO
4935 FORMAT(2H*,I3,2X,'VPC(1,','I1,2X,'I1,2X,':',F9.6),779,1H*)
RETURN

C

ENTRY SHOW_SC

ENTRY SET_SC

WRITE (2,+) 'Please input a uniform scale factor:'
READ (1,+) UNIT
UNITS = UNIT * UNIT
RETURN

C

ENTRY SET_UNI_METERS
IUNIT = 1
GOTO 2

ENTRY SET_UNI_FEET
IUNIT = 2
GOTO 2

ENTRY SET_UNI_INCHES
IUNIT = 3
2 UNIT = UNIT(IUNIT)
RETURN
This entry shows the current units.

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

ENTRY SET-WIN
VALID_INPUT = .FALSE.
DO WHILE (.NOT. VALID_INPUT )
  TYPE*, 'The current range of theta in degrees is ', THET1*DPR,
           ' to ', THET2*DPR
  TYPE*, 'with a resolution of ', RESTH*DPR, ' degrees/pixel.'
  TYPE*, 'The current range of phi in degrees is ', PH1*DPR,
           ' to ', PH2*DPR
  TYPE*, 'with a resolution of ', RESPH*DPR, ' degrees/pixel.'
  TYPE*, 'Please enter a new range for theta (lower,higher):'
  ACCEPT*, THE1,THET1
  THE1 = THE1 * RPD
  TYPE*, 'Please enter a new range for phi (lower,higher):'
  ACCEPT*, PH1,PH2
  "HI = PH1 * RPD
  PH2 = PH2 * RPD
  TYPE*, 'Please enter a new PHI resolution in degrees/pixel:'
  ACCEPT*, RESPH
  RESPH = RESPH * RPD
  ROWS = INT( (PH2 - PH1) / RESPH + 0.6 ) + 1
  COLS = INT( (THET2 - THET1) / RESTH + 0.6 ) + 1
  VALID_INPUT = (.NOT. (ROWS.GT.MAXROW) ) .OR.
           (.NOT. (COLS.GT.MAXCOL) )
  IF (.NOT. VALID_INPUT ) WRITE(2,*)
       ' Insufficient dimensions for specified resolution.'
  END DO
ENTRY SHOW-WIN
    TYPE*, 'The current range of theta in degrees is ', THET1*DPR,
           ' to ', THET2*DPR
    TYPE*, 'with a resolution of ', RESTH*DPR, ' degrees/pixel.'
    TYPE*, 'The current range of phi in degrees is ', PH1*DPR,
           ' to ', PH2*DPR
    TYPE*, 'with a resolution of ', RESPH*DPR, ' degrees/pixel.'
RETURN

Show the window parameters
This routine determines names of output files. Here are the current assignments.

<table>
<thead>
<tr>
<th>Unit</th>
<th>Meaning</th>
<th>Default file name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>interactive input</td>
<td>sys$in</td>
</tr>
<tr>
<td>2</td>
<td>interactive output</td>
<td>sys$out</td>
</tr>
<tr>
<td>5</td>
<td>input processor input</td>
<td>FILE.INP</td>
</tr>
<tr>
<td>6</td>
<td>input processor (echo) output</td>
<td>FILE.LIS</td>
</tr>
<tr>
<td>7</td>
<td>printable output file</td>
<td>FILE.PRT</td>
</tr>
<tr>
<td>10</td>
<td>&quot;plot&quot; data output file</td>
<td>FILE.PLT</td>
</tr>
</tbody>
</table>

ENTRY SET_OUT

CALL CL1$GET_VALUE('P2', FILE)

Only if /NOPLOT is specified, then discard all output written to unit 1.
The user should always get plottable output by default.

ENTRY SHOW-OUT

ENTRY EXIT_COMMAND

CALL EXIT
RETURN

C%%%%%%%%%%%%%%%%%%%%%% This routine services online help requests.
C  
C ENTRY HELP_COMMAND
CALL CLIGET_VALUE('P1', P1)
CALL CLIGET_VALUE('HELPLIB', 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

C%%%%%%%%%%%%%%%%%%%%%% This routine calls the routines which do the shadowing.
C  
C ENTRY SHADOW_COMMAND
TYPE*, 'Working...' ! Type an informational message
CALL INITGF ! Initialize next plot
CALL DOPLAS ! Draw the plates
CALL DOCYLS ! Draw the cylinders
CALL WRTOUT ! Write the output buffer
SHADOW_COMMAND = SUCCESS ! Return a normal
RETURN

C%%%%%%%%%%%%%%%%%%%%%% This routine executes a DCL command as a subprocess. Add a test for
C better behavior with blank P1s.
C  
C ENTRY DCL_COMMAND
CALL CLIGET_VALUE('P1', P1)
IF ( P1 .EQ. ' ' ) THEN
  DCL_COMMAND = LIB$SPAWN()
ELSE
  DCL_COMMAND = LIB$SPAWN( P1 )
ENDIF
RETURN

C End of action routines.
C  
C END

145
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 CIII
0003 CIII THIS COMPUTER CODE WAS WRITTEN AT THE OHIO STATE UNIVERSITY
0004 CIII ELECTROSCIENCE LABORATORY. ANY PROBLEMS OR COMMENTS
0005 CIII CAN BE REFERRED TO:
0006 CIII
0007 CIII RONALD J. MARHEFKA OR LASZLO A. TAKACS
0008 CIII ELECTROSCIENCE LABORATORY
0009 CIII 1320 KINNEAR RD.
0010 CIII COLUMBUS, OHIO 43212
0011 CIII POSE: (614) 422-6762 OR 422-6848
0012 CIII
0013 CIII THIS COMPUTER CODE CALCULATES SHADOWING OF AN ANTENNA
0014 CIII USING THE REC-BSC INPUTS NON-INTERACTIVELY.
0015 CIII IT SHOULD BE USED IN PLACE OF INTERACTIVE MAIN PROGRAM
0016 CIII WHEN THE SHADOW CODE IS USED ON NON VAX COMPUTERS.
0017 CIII
0018 INCLUDE 'SHACOM.FOR'
0019 PARAMETER (NSX=30)
0020 CIII COMPLEX WS
0021 CIII LOGICAL LRET
0022 CIII COMMON/STORARY/WS(NSX),XSS(3,NSX),NSA(2,NSX),MSA,MSX,MSPP
0023 CIII Initialize fill tags
0024 CIII FILPNM and FILCNUM < 0 is sequential tagging
0025 CIII FILPNM or FILCNUM > 0 that object is tagged with
0026 CIII FILCHR or FILCHC
0027 CIII FILPNM or FILCNUM = 0 everything tagged with FILCHR
0028 CIII FILPNM=0
0029 CIII FILCNUM=0
0030 CIII Initialize fill characters
0031 CIII FILCHR='P'
0032 CIII FILCHR='C'
0033 CIII FILCHR='X'
0034 CIII Initialize return flag
0035 CIII LRET=.TRUE.
0036 CIII Initialize and read command information.
0037 CALL ABSCII
0038 CIII CONTINUE
0039 CIII Choose a source location from stored positions.
0040 CIII DO 1200 MS=1,MSX
0041 CIII DO 1000 N=1,3
0042 CIII ANS(N)=XSS(N,MS)
0043 CIII Initialize graphics information.
0044 CALL INITGF
0045 CIII Calculate shadow of plates.
0046 CALL DOPLAS
0047 CIII Calculate shadow of cylinders.
0048 CALL DOCYLS
0049 CIII Write out maps to printer and plotter files.
0050 CALL WRTOUT
0051 CIII CONTINUE
```
0217  C111 Read more command information.
0218      CALL ABSCRE
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 C SUBROUTINE   ABSCIN
0003 C-------------------------------------------------------------------
0004 C111 THIS IS THE INPUT-SET PROCESSOR ROUTINE. IT READS COMMANDS FROM THE INPUT FILE WHICH DEFINE THE INPUT GEOMETRY.
0005 C111 THE NEC - BASIC SCATTERING CODE (NEC-BSC) WAS WRITTEN
0006 C111 AT THE OHIO STATE UNIVERSITY ELECTROSCIENCE LABORATORY.
0007 C111 ANY PROBLEMS OR COMMENTS CAN BE REFERRED TO:
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.
0015 C111 IT READS IN THE INPUT AND PASSES THE GEOMETRY INFORMATION
0016 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 RMS 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 C### means lines were not needed for SHADOW program
0049 C111 C### means lines were not implemented for current version
0050 C111

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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
CI
C++
C++
C++
C++
C++
C++
C++
C++
0280 COMMON/CMP/CJ,CPIA
0281 COMMON/FWANG/FPN(HEX,NPX)
0282 COMMON/LPLCY/LPLA,LCTL
0283 COMMON/GROUND/LGRUD,MPTR
0284 COMMON/OUTPZ/TPPD,PRAD,RANG,LCPAT,LPRA,LRANG
0285 COMMON/OUTPBZ/RSI,RTS,TRY,SYM,PSI,PZI,RECT
0286 COMMON/OUTPHV/IVPH,IVWLPL
0287 COMMON/TAU/LSLAB(NPX),MSLAB(NPX),DSLAB(NXL,NPX)
0288 2,ENSLAB(NXL,NPX),TSLAB(NXL,NPX),URSLAB(NXL,NPX)
0289 3,TMSLAB(NXL,NPX)
0290 DATA IY/TO/,PD/,PC/,SL/,LP/,PP/,GP/,XQ/,RT/,CG/
0291 2,SM/,RD/,CH/,CE/,BP/,UF/,BN/,UN/,FR/,NX/
0292 3,EN/,MP/,HC/,NG/,WS/,FR/,US/,PH/,RG/,HR/
0293 4,SA/,FM/,RA/,GR/,VD/,DN/,FP/,VF/,CC/
0294 C111 MAX. DIMENSION OF S,RECEIVERS,CYLINDERS,AIMS,PLATES,EDGES,
0295 C111 LAYERS, AND OBSERVATION POINTS.
0296 C111 NOTE: IN SUB. NPFTCL THE VARIABLES IVD,PHOR,PHOP,AND VRO
0297 C111 MUST BE DIMENSIONED 2+MDX+1
0298 C111
0299 C111 SET TIME FLAGS TO ZERO
0300 IATIM=0
0310 IBTIM=0
0311 ICTIM=0
0312 GO TO 2700
0313 2700 CONTINUE
0314 WRITE(6,3006)
0315 WRITE(6,3005)
0316 2701 CONTINUE
0317 C111 INITIALIZE DATA TO DEFAULT VALUES.
0318 C### TEST OUTPUT DEFAULT DATA TO:
0319 LDEBUG=.FALSE.
0320 LTEST=.FALSE.
0321 LOUT=.FALSE.
0322 LWARN=.TRUE.
0323 LSLOPE=.TRUE.
0324 LDIR=.TRUE.
0325 LOP=.FALSE.
0326 JMX(1)=1
0327 JMX(2)=6
0328 JMX(3)=6
0329 JMX(4)=4
0330 DO 2706 J=1,6
0331 DO 2706 K=1,4
0332 LKJ(K,J)=.FALSE.
0333 JMX(K,J) LE JMX(K,J)=.TRUE.
0334 2705 CONTINUE
0335 LKJ(K,J)=.FALSE.
0336 LKJ(K,J)=.FALSE.
0337 C### FAR ZONE RANGE DEFAULT DATA RD:
0338 LRANG=.FALSE.
0339 RANG=1.
0340 C### RANGE GATE DATA GR:
0341 RNIN=SMIL
0342 RNIN=SMIL
0343 C### POWER RADIATED DEFAULT DATA PR:
LPRAD=.FALSE.
PRAD=0.
IPAD=1

C### PATTERN DEFAULT DATA PD:, PN:, PF:, VD:, VF:, & VH:

LVOLP=.TRUE.
LPARK=.TRUE.
LNEAR=.FALSE.
LRECT=.FALSE.
LCPAT=.TRUE.

TPPD=0.
TPPF=0.
TPPI=2.

THCZ=0.

THCZ=90.

LRVZ=0.

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

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

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

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

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

RXS=I.
RXI=0.
TYS=0.
TYI=2.

PZS=0.
PZI=0.

NPN=181
NPV=91

C### BACK OR BISTATIC NEAR ZONE SCATTERING DEFAULT DATA BP:

LSCAT=.FALSE.

C### FREQUENCY DEFAULT DATA FR: & FM:

FRQC=.997926
LFQC=.FALSE.

C### PLATE DEFAULT DATA PG:

LPLA=.FALSE.

MPX=0
NEXP(1)=4

LSLAB(1)=0

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

C### GROUND PLANE DEFAULT DATA GP:

C###
LCRND=.FALSE.
MXR=MPX
C### SOURCE DEFAULT DATA SG: SA: & SM:
LSNP=.FALSE.
MSI=0
MSAT=0
MSA(1,1)=0
MS(2,1)=0
ISS(1,1)=0.
ISS(2,1)=0.
ISS(3,1)=0.
IMS(i)=-1
HS(l)=0.
HAWS(l)=0.
THSZ=0.
PHSZ=0.
THSX=0.
PHSX=0.
VXSS(i,i,l)=0.
VXSS(1,2,1)=0.
VXSS(1,3,1)=0.
VXSS(2,2,1)=0.
VXSS(2,3,1)=0.
VXSS(3,3,1)=0.
WS(l)=(0,0,0)
C### RECEIVER DEFAULT DATA RG: RA: & RM:
LRCVR=.FALSE.
LRNP=.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)=0.6
HAWR(l)=0.
THRZ=0.
PHRZ=0.
THRX=0.
PHRX=0.
VARR(1,1,1)=1.
VARR(1,2,1)=0.
VARR(1,3,1)=0.
VARR(2,1,1)=0.
VARR(2,2,1)=1.
VARR(2,3,1)=0.
VARR(3,3,1)=0.
WR(1)=(0,0,0)
C### LINE PRINTER DEFAULT DATA LP:
WRITE=.FALSE.
C### PLOTTER DEFAULT DATA PP: VP:
LVPLT=.FALSE.
LPLT=.FALSE.
LPREC=.FALSE.
PPXL=0.
PPYL=3.
PPXB=0.
PPXE=360.
PPXS=30.
PPYB=-40.
PPYE=0.
PPYS=10.

C $$ ROTATE TRANSLATE DEFAULT DATA RT: 
THZP=0.
PHZP=0.
THXP=90.
PHXP=0.
TR(1)=0.
TR(2)=0.
TR(3)=0.
VRT(1,1)=1.
VRT(1,2)=0.
VRT(1,3)=0.
VRT(2,1)=0.
VRT(2,2)=0.
VRT(2,3)=1.
VRT(3,1)=0.
VRT(3,2)=0.
VRT(3,3)=1.

C $$ CYLINDER DEFAULT DATA CD: & CC:
MDC=0
LCYL=.FALSE.
MCJ=0
NEC(1)=2
AC(1,1)=1.
BC(1,1)=1.
AC(2,1)=1.
BC(2,1)=1.
ZC(2,1)=-3.
TCR(2,1)=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.

C $$ UNITS DEFAULT DATA UN: & UF: & US:
UNIT=1
UNIT=UNIT(IUNIT)
UNIT=1.
UNIT=UNIT+UNITF
UNIT=0
IUNIT=IUNIT
GO TO 2999
ENTRY ABSCRE
CONTINUE
WRITE(6,3006)
FORMAT(IX,26(3H***))
READ IN VARIOUS COMMAND OPTIONS.
READ(6,3001,END=3004)
(IR(I),I=1,36)
FORMAT(38A2)
WRITE(6,3002)
FORMAT(1H //////////, /X, 26(3H***))
WRITE(6,3008)
WRITE(6,3003) (IR(I),I=1,36)
WRITE(6,3004)
WRITE(6,3005) (IR(I),I=1,38)
FORMAT(lX, lH*, 2X, 36A2, 2X, lH*)
WRITE(6,3006)
WRITE(6,3008)
WRITE(6,3009)

CM: COMMENT CARD
IF(IR(1).EQ.IT(1)) GO TO 3090
CE: LAST COMMENT CARD
IF(IR(1).EQ.IT(14)) GO TO 3000
WRITE(6,3008)
WRITE(6,3009)
WRITE(6,3005)
WRITE(6,3006)
WRITE(6,3004)
WRITE(6,3003) (IR(I),I=1,36)

TO: TEST DATA GENERATION OPTION.
IF(IR(1).EQ.IT(1)) GO TO 3100
PD: FAR ZONE PATTERN INTEGER ANGLES
IF(IR(1).EQ.IT(2)) GO TO 3200
RE: FAR ZONE RANGE INPUT
IF(IR(1).EQ.IT(12)) GO TO 3250
PC: PLATE GEOMETRY INPUT
IF(IR(1).EQ.IT(3)) GO TO 3300
SG: SOURCE GEOMETRY INPUT
IF(IR(1).EQ.IT(4)) GO TO 3400.
SH: SOURCE NEC OR AMP INPUT
IF(IR(1).EQ.IT(15)) GO TO 3450
LP: LINE PRINTER LISTING OF RESULTS
IF(IR(1).EQ.IT(6)) GO TO 3600
PP: PEN PLOT OF RESULTS
IF(IR(1).EQ.IT(6)) GO TO 3600
GP: INCLUDE INFINITE GROUND PLANE
IF(IR(1).EQ.IT(7)) GO TO 3700
EQ: EXECUTE PROGRAM
IF(IR(1).EQ.IT(8)) GO TO 3800
RT: TRANSLATE AND/OR ROTATE COORDINATES
IF(IR(1).EQ.IT(9)) GO TO 3900
CG: CYLINDER GEOMETRY INPUT
IF(IR(1).EQ.IT(10)) GO TO 4000
CC: CONE GEOMETRY INPUT
IF(IR(1).EQ.IT(11)) GO TO 4000
BP: BACK OR BISTATIC NEAR ZONE SCATTERING
IF(IR(1).EQ.IT(15)) GO TO 5240
UF: SCALE FACTOR FOR INPUT
IF(IR(1).EQ.IT(16)) GO TO 4120
UN: UNITS OF INPUT
IF(IR(1).EQ.IT(18)) GO TO 4100
FR: FREQUENCY
IF(IR(1).EQ.IT(19)) GO TO 4200
HA: HEAT PROBLEM
IF(IR(1).EQ.IT(20)) GO TO 3700
EX: END PROGRAM
IF(IR(1).EQ.IT(21)) GO TO 997
NP: NEXT SET OF PLATES
IF(IR(1).EQ.IT(22)) GO TO 3360
HC: NEXT SET OF CYLINDERS
IF(IR(1).EQ.IT(23)) GO TO 4050
HG: NO GROUND PLANE
IF(IR(1).EQ.IT(24)) GO TO 3760
HS: NEXT SET OF SOURCES
IF(IR(1).EQ.IT(25)) GO TO 3450
PR: POWER RADIATED INPUT
IF(IR(1).EQ.IT(26)) GO TO 3440
US: UNITS OF HS AND HAWS IN SC, 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$$ RM: RECEIVER NEC OR AMP INPUT
IF(IR(l).EQ.IT(l7)) GO TO 4460
Cat$ NR: NEXT SET OF RECEIVERS
IF(IR(1).EQ.IT(30)) GO TO 3496
CaO SA: SOURCE ARRAY GEOMETRY INPUT
IF(IR(1).EQ.IT(31)) GO TO 3810
Ca$$ FF: MULTIPLE FREQUENCY 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$88 VD: FAR ZONE VOLUMETRIC PATTERN INTEGER ANGLES
IF(IR(1).EQ.IT(36)) GO TO 9210
C8$ VN: NEAR ZONE VOLUMETRIC PATTERN
IF(IR(1).EQ.IT(36)) GO TO 3270
CaO VP: VOLUMETRIC DUMP OF RESULTS FOR PLOTTING
IF(IR(1).EQ.IT(37)) GO TO 3660
C$$ PF: FAR ZONE NON INTEGER ANGLES
IF(IR(1).EQ.IT(38)) GO TO 3220
CaO VF: FAR ZONE VOLUMETRIC PATTERN NON INTEGER ANGLES
IF(IR(I).EQ.IT(39)) GO TO 3230
CS$$ YRITE(6,3021)
3021 FORMAT(' *** PROGRAM ABORTS!!! COMMAND INPUT IS NOT PART',
2' OF STORED COMMAND LIST ***')
3004 STOP
C=====
3090 CONTINUE
C=== CM: CE: COMMANDS ======
C$$ LDEBUG=DEBUG DATA OUTPUT ON LINE PRINTER(TRUE OR FALSE)
C$$ LTEST=TEST DATA TO INSURE PROGRAM OPERATION(TRUE OR FALSE)
C$$ LOUT=OUTPUT MAIN PROGRAM DATA ON LINE PRINTER(TRUE OR FALSE)
C$$ LWARN=WARNING DATA OUTPUT ON LINE PRINTER(TRUE OR FALSE)
READ(6,3101) LDEBUG,LTEST,LOUT,LWARN
WRITE(6,3101) LDEBUG,LTEST,LOUT,LWARN

FORMAT(2H *,6X,'LDEBuc= ',L3,6X,'LTEST= ',L3,6X,'LOUT= ',L3
2,6X,'LWARN= ',L3,T79,1H+)
WRITE(6,3006)

C###

Lslope=SLOPE DIFFRACTED FIELD DESIRED (T OR F)
LCORNR= CORNER DIFFRACTED FIELD DESIRED (T OR F)
LSOR= SORANTE SHADOW ALONE (TRUE OR FALSE)

READ(6,*) Lslope,LCORNR,LSOR
WRITE(6,3102) Lslope,LCORNR,LSOR

FORMAT(2H *,6X,'LSLOPE= ',L3,6X,'LCORNR= ',L3,6X,'LSOR= ',L3,
2,6X,'LVAR!I= ',L3,T79,1H*)

3102 FORMAT(2H *,6X,'LSLOPE=SLOPE DIFFRACTED FIELD DESIRED (T OR F)
LCORNR= CORNER DIFFRACTED FIELD DESIRED (T OR F)
LCORNR= CORNER DIFFRACTED FIELD DESIRED (T OR F)
LSOR= SORANTE SHADOW ALONE (TRUE OR FALSE)

READ(6,*) Lslope,LCORNR,LSOR
WRITE(6,3006)
IF(LS0R) WRITE(6,3008)

IF(LS0R)

K=1,J=OPTION TO RUN DIRECT RAY TERM:

K=2,J=OPTION TO RUN VARIOUS RAY TERMS FOR PLATES:

K=3,J=OPTION TO RUN VARIOUS RAY TERMS FOR Cylinder:

K=4,J=OPTION TO RUN VARIOUS RAY TERMS FOR PLATE-CYLINDER INTERACTIONS:

DO 3104 K=1,4
JK=JK(K)
READ(5,*) (LKJ(K,J),J=1,JK)
WRITE(6,3103) K,(LKJ(K,J),J=1,JK)
FORMAT(2H *,T79,1H*,T8,'LKJ(',I1,'',',J)= ',6L2)
GO TO 3000

C###

156
C*** UN: COMMAND
C*** IUUNIT=INDICATOR OF UNITS USED FOR INPUT DATA.
C*** 1=METERS
C*** 2=FEET
C*** 3=INCHES
C*** READ(6,*) IUUNIT
C*** UNITS=UNIT(IUUNIT)
C*** WRITE(6,4101) LABEL(IUUNIT)
C*** 4101 FORMAT(2H*.6X,'ALL THE LINEAR DIMENSIONS BELOW ARE'
C*** 2,' ASSUMED TO BE IN ',A6,T79,1H*)
C*** GO TO 3000
C***
C*** CONTINUE
C*** 4120 CONTINUE
C*** UF: COMMAND
C*** C*** IUUNIT = SCALE FACTOR FOR GEOMETRY
C*** C*** READ(6,*) UNITF
C*** UNITS=UNITN*UNITF
C*** WRITE(6,4111) UNITF
C*** 4111 FORMAT(1H*,CX,'ALL THE LINEAR DIMENSIONS'
C*** 2,' BELOW ARE SCALED BY ',F12.6,T79,1H*)
C*** GO TO 3000
C***
C*** CONTINUE
C*** 4110 CONTINUE
C*** US: COMMAND
C*** C*** IUINST=INDICATOR OF UNITS USED FOR HS AND HAWS IN THE
C*** C*** EG: COMMAND.
C*** C*** IUSP: WAVELENGTHS
C*** C*** 1=METERS
C*** C*** 2=FEET
C*** C*** 3=INCHES
C*** C*** NOTE: IF ONE SOURCE IS SPECIFIED IN WAVELENGTHS, THEY ALL
C*** C*** MUST BE IN WAVELENGTHS.
C*** C*** READ(6,*) IUINST
C*** IF(NSI.EQ.0) GO TO 4112
C*** IF(IUINST.EQ.0 .AND. IUWSP.EQ.0) GO TO 4112
C*** IF(IUINST.NE.0 .AND. IUWSP.NE.0) GO TO 4112
C*** WRITE(6,4111)
C*** 4111 FORMAT(' *** PROGRAM ABORTS IN SOURCE UNITS. ALL UNITS NOT'
C*** 2,' SPECIFIED IN WAVELENGTHS!!! ***')
C*** STOP
C***
C*** CONTINUE
C*** 4112 CONTINUE
C*** IF(IUINST.EQ.0) GO TO 4114
C*** WRITE(6,4113) LABEL(IUINST)
C*** 4113 FORMAT(2H*.5X,'THE SOURCE LENGTH HS AND WIDTH HAWS ARE'
C*** 2,' ASSUMED TO BE IN ',A6,T79,1H*)
C*** GO TO 4116
C*** 4114 WRITE(6,4116)
C*** 4116 IUWSP=IUINST
C*** GO TO 3000
C***
C*** CONTINUE
C*** 4200 CONTINUE
C*** FR: COMMAND
C*** C*** FREQ=FREQUENCY IN GIGAHERTZ.
LFQC=.FALSE.
NFQC=1
READ(6,*) FRQQ
WL=.2997926/FRQC
WRITE(6,4201) FRQQ
WRITE(6,4202) WL
GO TO 3000
C=====
4260 CONTINUE
C=== FM: COMMAND =====
C$$$ NFQC=NUMBER OF FREQUENCIES DESIRED
C$$$ CQCS=STARTING FREQUENCY IN GIGAHERTZ
C$$$ FQCI=INCREMENTAL FREQUENCY CHANGE IN GIGAHERTZ
C$$ NOTE: THE SOURCE LENGTH AND WIDTH MUST NOT BE SPECIFIED
IN WAVELENGTHS. ALSO ONLY ONE PATTERN LOCATION CAN BE SPECIFIED.
LFQC=.TRUE.
READ(6,*) NFQC,FQCS,FQCI
WRITE(6,4251) NFQC
FRQC=FQCS+0.6*FQCI*(NFQC-1)
WL=.2997926/FRQC
GO TO 3000
C=====
3230 CONTINUE
C=== VF: COMMAND =====
C$$$ FAR ZONE VOLUMETRIC PATTERN NON INTEGER ANGLES
C$$$ LVOLP=.TRUE.
C$$$ LFAR=.TRUE.
C$$$ GO TO 3211
C=====
3210 CONTINUE
C=== VD: COMMAND =====
C$$$ FAR ZONE VOLUMETRIC PATTERN INTEGER ANGLES
C$$$ LVOLP=.TRUE.
C$$$ LFARN=.FALSE.
C$$$ GO TO 3211
C=====
3220 CONTINUE
C=== PF: COMMAND =====
C$$$ FAR ZONE PATTERN NON INTEGER ANGLES
C$$$ LVOLP=.FALSE.
C$$$ LFARN=.TRUE.
GO TO 3211
C-------
3200 CONTINUE
C-------
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*** THC1.PHC1=ORIENTATION OF THE X AXIS RELATIVE TO THE
C*** FIXED COORDINATE SYSTEM.
C***
LVOLP=.FALSE.
3211 LINEAR=.FALSE.
READ(5,*) THCZ,PHCZ,THCI,PHCI
VPC(3,1)=SIN(THCZ*RPD)*COS(PHCZ*RPD)
VPC(3,2)=SIN(THCZ*RPD)*SIN(PHCZ*RPD)
VPC(3,3)=COS(THCZ*RPD)
WC(1,1)=SIN(THCI*RPD)*COS(PHCI*RPD)
VPC(1,2)=SIN(THCI*RPD)*SIN(PHCI*RPD)
VPC(1,3)=COS(THCI*RPD)
DZX=WC(1,1)*WC(1,1)+WC(1,2)*WC(1,2)*WC(1,3)*WC(1,3)
IF(ABS(DZX).LT.0.1) WRITE(6,3201)
3201 FORMAT(*** PROGRAM ABORTS IN PATTERN CUT SECTION. ***)
2, . THE COORDINATES ARE NOT ORTHOGONAL!!! ***
IF(ABS(DZX).LT.0.1) STOP
VPC(1,1)=VPC(1,1)-WC(3,1)*DZX
VPC(1,2)=VPC(1,2)-WC(3,2)*DZX
VPC(1,3)=VPC(1,3)-WC(3,3)*DZX
DOT=WC(1,1)*WC(1,1)+WC(1,2)*WC(1,2)*WC(1,3)*WC(1,3)
DOR=SQRT(DOT)
WC(1,1)=WC(1,1)/DOR
VPC(1,2)=VPC(1,2)/DOT
WC(1,3)=VPC(1,3)/DOT
WC(2,1)=WC(3,1)*WC(1,1)-VPC(3,1)*WC(1,3)
WC(2,2)=WC(3,2)*WC(1,2)-VPC(3,2)*WC(1,3)
WC(2,3)=WC(3,3)*WC(1,3)-VPC(3,3)*WC(1,3)
WRITE(6,3202)
3202 FORMAT(2H * ,6X, 'THE PATTERN AXES ARE AS FOLLOWS: ',T90,1H*)
DO 3204 N1=1,3
WRITE(6,3206) (NI,NJ,VPC(NI,NJ),NJ+1,NJ)
3206 FORMAT(2H * ,1X,3(2X,'VPC(',',',',',')=',F10.6),T90,1H*)
DO 3203 N=1,3
XPC(N)=O.
3203 XPC(N)=O.
3204 WRITE(6,3206) (NI,NJ,VPC(NI,NJ),NJ+1,NJ)
3206 FORMAT(2H * ,1X,3(2X,'VPC(',',',',',')=',F10.6),T90,1H*)
DO 3203 N=1,3
3203 XPC(N)=O.
3204 WRITE(6,3206) (NI,NJ,VPC(NI,NJ),NJ+1,NJ)
3206 FORMAT(2H * ,1X,3(2X,'VPC(',',',',',')=',F10.6),T90,1H*)
DO 3203 N=1,3
3203 XPC(N)=O.
3204 WRITE(6,3206) (NI,NJ,VPC(NI,NJ),NJ+1,NJ)
3206 FORMAT(2H * ,1X,3(2X,'VPC(',',',',',')=',F10.6),T90,1H*)
DO 3203 N=1,3
3203 XPC(N)=O.
3204 WRITE(6,3206) (NI,NJ,VPC(NI,NJ),NJ+1,NJ)
3206 FORMAT(2H * ,1X,3(2X,'VPC(',',',',',')=',F10.6),T90,1H*)
DO 3203 N=1,3
3203 XPC(N)=O.
3204 WRITE(6,3206) (NI,NJ,VPC(NI,NJ),NJ+1,NJ)
3206 FORMAT(2H * ,1X,3(2X,'VPC(',',',',',')=',F10.6),T90,1H*)
DO 3203 N=1,3
3203 XPC(N)=O.
3204 WRITE(6,3206) (NI,NJ,VPC(NI,NJ),NJ+1,NJ)
3206 FORMAT(2H * ,1X,3(2X,'VPC(',',',',',')=',F10.6),T90,1H*)
DO 3203 N=1,3
3203 XPC(N)=O.
3204 WRITE(6,3206) (NI,NJ,VPC(NI,NJ),NJ+1,NJ)
3206 FORMAT(2H * ,1X,3(2X,'VPC(',',',',',')=',F10.6),T90,1H*)
DO 3203 N=1,3
3203 XPC(N)=O.
3204 WRITE(6,3206) (NI,NJ,VPC(NI,NJ),NJ+1,NJ)
3206 FORMAT(2H * ,1X,3(2X,'VPC(',',',',',')=',F10.6),T90,1H*)
DO 3203 N=1,3
3203 XPC(N)=O.
3204 WRITE(6,3206) (NI,NJ,VPC(NI,NJ),NJ+1,NJ)
3206 FORMAT(2H * ,1X,3(2X,'VPC(',',',',',')=',F10.6),T90,1H*)
DO 3203 N=1,3
3203 XPC(N)=O.
IF(LCPAT) WRITE(6,3207) TPPD
3207 FORMAT(2H +5X,'PHI IS BEING VARIED WITH THETA= ',F10.6)
            2.779,1H+)
WRITE(6,3006)
GO TO 3216

3208 IF(LCPAT) WRITE(6,3207) TPPD
3209 GO TO 3216

C###
C### TPPD=START OF VOLUMETRIC PATTERN ANGLE
C### TPPV=INCREMENT FOR VOLUMETRIC PATTERN ANGLE
C### NPV=NUMBER OF VOLUMETRIC PATTERN ANGLES
C###
C### 3212 READ(6,*) LCPAT,TPPD,TPPV,NPV
3213 WRITE(6,3008)
3214 IF(.NOT.LCPAT) WRITE(6,3214)
3215 WRITE(6,3216) TPPD,TPPV,NPV
3216 FORMAT(2H +5X,'FOR THETA ANGLE = ',F10.6,' STEP = ',F10.6,' NUMBER = ',I4
            2.779,1H+)
WRITE(6,3006)
IF(LCPAT) WRITE(6,3216)
IF(.NOT.LCPAT) WRITE(6,3213)
GO TO 3216
CONTINUE
3218 IF(LCPAT) GO TO 3217

C###
C### IB,IE,IS=BEGIN,END,STEP
C###
3217 CONTINUE
3218 CONTINUE
3219 CONTINUE
3220 READ(6,*) IB,IE,IS
3221 IF(IE.LT.0) IE=0
3222 IF(IS.LT.0) IS=1
3223 TPPD=IB
3224 TPPI=IS
3225 NPV=(IE-IB)/IS+1
3226 WRITE(6,3208) IB,IE,IS
3227 FORMAT(2H +5X,'THE RANGE OF PATTERN ANGLE INDICES FOR THIS'
            2,' RUN ARE: ',I3,2('(','),I3),T79,1H+)
WRITE(6,3218)
GO TO 3218

C###
C### TPPS=START OF PATTERN
C### TPPI=PATTERN INCREMENT
C### NPV=NUMBER OF PATTERN POINTS
C###
3228 CONTINUE
3229 CONTINUE
3230 CONTINUE
3231 CONTINUE
RIS=1.
RI=0.
TYS=TPPD
TYI=TPPV
PZS=TPPS
PZI=TPPI
IVPH=3
IF(LCPAT) GO TO 3209
3209 GO TO 3209
CONTINUE
GO TO 3000
CONTINUE

160
C==
RD:
COWND
=I===
cat1
C($(
RANCSrFAR
FIELD RANGE DISTANCE
cats
Cl((
NOTE IF RANCS IS GREATER THAN OR EQUAL TO 1.12.6.
1997
3,' IS ',E12.6,T79,1H+)
1998
GO TO 3000
1000
3262 CONTINUE
1001
LRANG=.FALSE.
1002
RANG=1.
1003
WRITE(6,3253)
1004
3263 FORMAT(2H*,'NO FAR FIELD RANGE SPECIFIED.',T79,1H*)
1005
GO TO 3000
1006
C===
1007
3270 CONTINUE
1008
C== VN:  COMMAND ======
1009
C===
1010
C===
NEAR ZONE VOLUMETRIC PATTERN
1011
C===
1012
LVOLP=.TRUE.
1013
GO TO 3271
1014
C===
1015
3260 CONTINUE
1016
C== PH:  COMMAND ======
1017
C===
1018
C===
XPC(N)=XYZ LOCATION OF THE NEAR ZONE PATTERN ORIGIN
1019
C===
1020
C===
1021
3271
LVOLP=.FALSE.
1022
3270
LINEAR=.TRUE.
1023
READ(6,*) (XPC(N),N=1,3)
1024
WRITE(6,3264) LABEL(IUNIT), (XPC(N),N=1,3)
1025
3264 FORMAT(2H*,1X,'PATTERN ORIGIN IN ',A6,' :
1026
XPC(1)=',F8.3,
1027
XPC(2)=',F8.3,
1028
XPC(3)=',F8.3,T79,1H+)
1029
WRITE(6,3006)
1030
DO 3263 N=1,3
1031
3263
WRITE(6,3006)
1032
C===
1033
C===
THCZ,PHCZ=ORIENTATION OF THE Z-AXIS OF THE PATTERN AXIS
1034
C===
1035
C===
1036
C===
THCZ,PHCZ=ORIENTATION OF THE X-AXIS OF THE PATTERN AXES
1037
C===
1038
C===
1039
READ(6,*) THCZ,PHCZ,THCI,PHCI
1040
VPC(3,1)=SIN(THCZ*RPD)*COS(PHCZ*RPD)
1041
VPC(3,2)=SIN(THCZ*RPD)*SIN(PHCZ*RPD)
1042
VPC(3,3)=COS(THCZ*RPD)
1043
VPC(1,1)=SIN(THCI*RPD)*COS(PHCI*RPD)
1044
VPC(1,2)=SIN(THCI*RPD)*SIN(PHCI*RPD)
1045
VPC(1,3)=COS(THCI*RPD)
1046
C!!! INSURE VPC(1,3) IS PERPENDICULAR TO VPC(3,3)
1047
DE1=VPC(3,1)+VPC(1,1)+VPC(3,2)+VPC(1,2)+VPC(3,3)+VPC(1,3)

161
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(1,1) + 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(6,3202)
DO 3261 N = 1,3
WRITE(6,3205) (NI, NJ, VPC(NI, NJ), NI = 1, 3)
WRITE(6,3206)
WRITE(6,3208)
WRITE(6,3209)
LRECT = F, SPHERICAL PATTERN CUT
LRECT = T, LINEAR PATTERN CUT
RIS, TYS, PZS = STARTING LOCATION OF PATTERN
LRECT = F: RADIAL, THETA, PHI
LRECT = T: X, Y, Z
RIS, TYS, PZS = SIZE OF INCREMENTAL STEPS
LRECT = F: RADIAL, THETA, PHI
LRECT = T: X, Y, Z
READ(*, *) LRECT
READ(*, *) RIS, TYS, PZS
READ(*, *) RXI, TYI, PZI
IF(LRECT) WRITE(6,3201) RIS, TYS, PZS, LABEL(IUNIT)
FORMAT(2H *, 2X, 'STARTING XYZ=', F10.5, 2(', ', F10.5), 1X, A6
2.779, 1H*)
IF(LRECT) WRITE(6,3202) RXI, TYI, PZI, LABEL(IUNIT)
FORMAT(2H *, 2X, 'STEP XYZ=', F10.5, 2(', ', F10.5), 1X, A6, T7.9, 1H*)
IF(.NOT. LRECT) WRITE(6,3207) RIS, TYS, PZS, LABEL(IUNIT)
IF(.NOT. LRECT) WRITE(6,3207) RIS, TYS, PZS, LABEL(IUNIT)
FORMAT(2H *, 2X, 'STARTING R, THETA, PHI=', F10.5, 2(', ', F10.5), 1X, A6
2.2, 'AND DEG.', T7.9, 1H*)
IF(.NOT. LRECT) WRITE(6,3208) RXI, TYI, PZI, LABEL(IUNIT)
IF(., NOT. LRECT) WRITE(6,3208) RXI, TYI, PZI, LABEL(IUNIT)
FORMAT(2H *, 2X, 'STEP R, THETA, PHI=', F10.5, 2(', ', F10.5), 1X, A6
2.2, 'AND DEG.', T7.9, 1H*)
WRITE(6,3209)
RIS = UNITS * RIS
RXI = UNITS * RXI
TYS = UNITS * TYS
PZS = UNITS * PZS
TYI = UNITS * TYI
PZI = UNITS * PZI
CONTINUE
IF(LRECT AND IUNIT NE. 1) WRITE(6,3261) RIS, TYS, PZS, LABEL(IUNIT)
IF(LRECT AND IUNIT NE. 1) WRITE(6,3261) RIS, TYS, PZS, LABEL(IUNIT)
IF(., NOT. LRECT) GO TO 3265
WRITE(6,3265)
TYS = UNITS * TYS
PZS = UNITS * PZS
TYI = UNITS * TYI
PZI = UNITS * PZI
CONTINUE
IUNIT = NUMBER OF PATTERN POINTS
READ(*, *)ertain
WRITE(6,3269) NPN
3269 FORMAT(2H*,'NUMBER OF PATTERN POINTS=' ,I4,T70,1H+)  
IVPN=3
IF(ABS(PZ1).LT.SMLA) IVPN=-3
IF(IAECT) IVPN=0
GO TO 3276
C###
IVPN=1 FOR R-THETA OR X-Y VARYING
C### NPV=NUMBER OF R OR X AND NPN=NUMBER OF THETA OR Y
C###
IVPN=2 FOR R-PHI OR X-Z VARYING
C### NPV=NUMBER OF R OR X AND NPN=NUMBER OF PHI OR Z
C###
IVPN=3 FOR THETA-PHI OR Y-Z VARYING
C### 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###
3272 READ(6,*) IVPN,IVP,NPN  
3273 FORMAT(2H*,'NUMBER OF POINTS FOR R OR X= ',I4,2,')
2. AND THETA OR Y= ',I4)
3274 FORMAT(2H*,'NUMBER OF POINTS FOR R OR X= ',I4,2,')
2. AND PHI OR Z= ',I4)
3275 FORMAT(2H*,'NUMBER OF POINTS FOR THETA OR Y= ',I4,2,')
2. AND PHI OR Z= ',I4)
3276 CONTINUE
3266 FORMAT(' ***** NUMBER OF POINTS=' ,I3,' PROGRAM ABORTS'
2.'PATTERN STORAGE DIMENSION IS EXCEEDED *****')
3266 IF(NPN.GT.MODX) STOP
GO TO 3000
C=====
E240 CONTINUE
C== BP: COMMAND =====
C###
BACK OR BISTATIC NEAR ZONE SCATTERING
C###
THE SC.; RG.; AND PN: COMMANDS MUST BE SPECIFIED
C### TO USE THIS OPTION.
LSCAT=.TRUE.
GO TO 3000
C=====
E260 CONTINUE
C== GR: COMMAND =====
C###
RANGE GATE INPUT
C###
RMIN=THE MINIMUM DISTANCE FROM TRANSMITTER TO RECEIVER
C### RMAX=THE MAXIMUM DISTANCE FROM TRANSMITTER TO RECEIVER
C###
THE PN: COMMAND MUST BE USED
C###
READ(6,*). RMH,RMA
WRITE(6,3261) RMH,RMA,LABEL(UNIT)
3261 FORMAT(2H*,'RMH= ',I3,'RMAX= ',F10.5,'RMH= ',F10.5,' IN ',A6,T70,1H+)
RMH=UNITS*RMIN
RMA=UNITS*RMAX

163
WRITE(0,6261) RMIN, RMAX, LABEL(1)
GO TO 3000
C=====
3300 CONTINUE
C== PC: COMMAND ======
C### PLATE GEOMETRY INPUT
C###
1184 LPLA = .TRUE.
1185 MPX = MPX + 1
1186 IF(MPX.GT.MPDX) WRITE(6,001) MPX
1187 001 FORMAT(5X, '***** NUMBER OF PLATES=', 'I3', ' PROGRAM ABORTS',
1188 1X, ' SINCE MAX. PLATE DIMENSION IS EXCEEDED. *****')
1189 IF(MPX.GT.MPDX) STOP
1190 WRITE(6,3301) MPX
1191 3301 FORMAT(2H4, 'THIS IS PLATE NO. ', 'I3', ' IN THIS ',
1192 2H SIMULATION', 'I79, 1H+')
1193 MPX = MPX
1194 WRITE(6,3006)
1195 WRITE(6,3006)
1196 WRITE(6,3009)
1197 C### MCP(MP)=NUMBER OF CORNERS ON THE MP-TH PLATE.
1198 C###
1200 C### LSLAB= 1 IMPLIES TRANSPARENT THIN DIELECTRIC SLAB
1201 C### = 0 IMPLIES METAL PLATE, AND
1202 C### =-2 IMPLIES DIELECTRIC COVERED PLATE ON BOTH SIDES
1203 C### =-4 IMPLIES DIELECTRIC COVERED PLATE ON SIDE OF NORMAL
1204 C###
1205 C### NOTE: IF DIELECTRIC COVERED, ONE MUST READ DIELECTRIC DATA.
1206 C###
1207 C###
1208 READ(6,*) MCP(MP), LSLAB(MP)
1209 IF(LSLAB(MP).EQ.0) WRITE(6,3392)
1210 3392 FORMAT(2H4, 'METAL PLATE USED IN THIS SIMULATION', 'I79, 1H+')
1211 IF(LSLAB(MP).EQ.1) WRITE(6,3393)
1212 3393 FORMAT(2H4, 'TRANSPARENT THIN DIELECTRIC LAYER USED IN THIS',
1213 2H SIMULATION', 'I79, 1H+')
1214 IF(LSLAB(MP).EQ.-2) WRITE(6,3394)
1215 3394 FORMAT(2H4, 'DIELECTRIC COVERED PLATE USED IN THIS',
1216 2H SIMULATION', 'I79, 1H+')
1217 WRITE(6,3006)
1218 IF(LSLAB(MP).EQ.0) GO TO 3313
1219 C###
1220 C### NSLAB(MP)=NUMBER OF DIELECTRIC LAYERS ON THE MP PLATE
1221 C###
1222 READ(6,*) NSLAB(MP)
1223 RSL=NSLAB(MP)
1224 IF(RSL.GT.MLNX) STOP
1225 WRITE(6,3391)
1226 3391 FORMAT(2H4, 'THICKNESS', '2X', 'DIELECTRIC', '3X', 'LOSS', '4X',
1227 2H PERMITIVITY', '3X', 'LOSS', 'I79, 1H+.,',
1228 5H, 'LAYER', '2X', 'IN METERS', '3X', 'CONSTANT', '3X', 'TANGENT',
1229 4H, 'CONSTANT', '3X', 'TANGENT', 'I79, 1H+.,',
1230 6H, '---------', '2X', '--------', '2X', '--------', '2X', '--------',
1231 62X, '---------', '2X', '--------', 'I79, 1H+.)
1232 C###
1233 C### DSLAB(NS,MP)=THICKNESS OF NS LAYER
1234 C###
1235 ERSLAB(NS,MP)=RELATIVE DIELECTRIC CONSTANT OF THE NS LAYER
1236 C###
1237 TESLAB(NS,MP)=DIELECTRIC LOSS TANGENT OF THE NS LAYER
1238 C###
1239 URSLAB(NS,MP)=RELATIVE PERMEABILITY CONSTANT OF THE NS LAYER

164
C$$$  TMBSLAB(NS,MP)*PERMEABILITY LOSS TANGENT OF THE NS LAYER
C$$
DO 3312 NS=1,NS
READ(6,*) DLSLAB(NS,MP),ERSLAB(NS,MP),TESLAB(NS,MP),
2URSLAB(NS,MP),TMBSLAB(NS,MP)
DLSLAB(NS,MP)=DLSLAB(NS,MP)+UNITS
3312 WRITE(6,3302) NS,DLSLAB(NS,MP),ERSLAB(NS,MP),TESLAB(NS,MP),
2URSLAB(NS,MP),TMBSLAB(NS,MP)
3399 FORMAT(2H *,6X,13,4X,9F4,4,2X,10F4,2X,7F4,2X,11F4,2X.
2P7.4,T79,1H*)
WRITE(6,3006)
WRITE(6,3006)
WRITE(6,3006)
3313 ME=MEP(MP)
IF(ME.GT.MEX) STOP
DO 6 ME=1,MEX
C$$
XX(N,KE,KP)=X,T,Z
COMPONENTS OF CORNER #:HE
OF P*nw.
C$$
INPUT CORNER DATA AS FOLLOWS:

1. -1.,0.,
2. -1.,1.,0.
3. 1.,-1.,0.
4. 1.,1.,0.

NOTE IF THERE IS MORE THAN ONE PLATE, THEN
DATA FOR EACH PLATE WOULD FOLLOW
SEQUENTIALLY.

3302 FORMAT(2H *,13,'PLATE #',13,'CORNER #',4X,'INPUT LOCATION IN
',1A6,4X,'ACTUAL LOCATION IN METERS',779,1H*)
WRITE(6,3303)
3303 FORMAT(2H *,2X,'-----',2X,'-----
',2,1(2~,2(*-------------#
,T7QB1H*)
DO 3304 ME=1,MEX
DO 3310 N=1,3
3310 IQ(N)=II(N,ME,MP)
DO 3311 N=1,3
3311 IX(N,ME,MP)=UNITS*IX(N)+VBT(1,N)*IQ(1)+VBT(2,N)
2*IX(N)+VBT(3,N)*TR(N)
WRITE(6,3306) MP,ME,IX(N),N=1,3,(IX(N,ME,MP),N=1,3)
3306 FORMAT(2H *,4X,13,6X,12,2X,(21,X8.3),T79,1H*)
3304 CONTINUE
3300 GO TO 3000
C=====
3360 CONTINUE
C== NP: COMMAND =======
C$$
LPLA=.FALSE.
MPI=0
WRITE(6,3351)
3351 FORMAT(2H *,6X,'THE PLATE DATA IS INITIALIZED',79,T79,1H*)
3352 IF(ME.GT.MEX) STOP
GO TO 3000
C======
165
CONTINUE

C===

CST

C$$$

MSX

C$$$

LSMP=.FALSE.

C$$$

MSX=MSX+1

C$$$

IF(3401.GT.KSMX) WRITE(6,904) MSX

C$$$

FORMAT('***** NUMBER OF SOURCES= ',13,' PROGRAM',

C$$$

2' ABORTS SINCE MAX. SOURCE DIMENSION IS EXCEEDED. *****')

C$$$

IF(3401.GT.KDSX) STOP

C$$$

WRITE(6,3401) MSX

C$$$

FORMAT(2H*.5X,'THIS IS SOURCE NO. ',13,' IN THIS',

C$$$

2' COMPUTATION.',779,1H*)

C$$$

WRITE(6,3000)

C$$$

WRITE(6,3000)

C$$$

XSS(N,MS)=XYZ LOCATION OF MS-TH ANTENNA ELEMENT.

C$$$

IMS(MS)=TYPE OF LINEAR ANTENNA

C$$$

. LT: ELECTRIC LINEAR ELEMENT

C$$$

. GT: MAGNETIC LINEAR ELEMENT

C$$$

ABS(IMS)=1: UNIFORM CURRENT DISTRIBUTION

C$$$

=2: STANDARD DIPOLE CURRENT DISTRIBUTION

C$$$

=3: CAVITY BACKED SLOT CURRENT DISTRIBUTION (TDF)

C$$$

HAYS(MS)=APERTURE WIDTH IN WAVELENGTHS (NOTE: IF

C$$$

HAYS(MS) IS LESS THAN .1 LAMBDA, SOURCE IS

C$$$

CONSIDERED TO BE DIPOLE SOURCE

C$$$

HS(MS)=LENGTH OF LINEAR ELEMENT IN WAVELENGTHS

C$$$

THSZ.PHSZ=ORIENTATION ANGLES USED TO DEFINE LINEAR

C$$$

ELEMENT AXIS.

C$$$

THSZ.PHSZ=ORIENTATION ANGLES USED TO DEFINE APERATURE

C$$$

PLANE OR DIPOLE X-AXIS.

C$$$

WMS,WPS=MAGNITUDE AND PHASE OF EXCITATION OF

C$$$

MS-TH ELEMENT.

C$$$

MS=MSX

C$$$

MSA(1,NS)=0

C$$$

MSA(2,NS)=0

C$$$

READ(6,*) (XSS(N,MS),N=1,3)

C$$$

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

C$$$

READ(6,*) IMS(MS),HS(MS),HAWS(MS)

C$$$

READ(6,*) WMS,WPS

C$$$

IF(1MS(MS).LT.0) WRITE(6,3411) IMS(MS)

C$$$

formatsch=.6X,'THIS IS AN ELECTRIC SOURCE OF TYPE ',13,T79,1H*)

C$$$

IF(1MS(MS).GE.0) WRITE(6,3412) IMS(MS)

C$$$

formatsch=.6X,'THIS IS A MAGNETIC SOURCE OF TYPE ',13,T79,1H*)

C$$$

WRITE(6,3000)

C$$$

IF(IUI1ST.EQ.0) GO TO 3114

C$$$

ULISTS=ULISTS(IUI1ST)

C$$$

WRITE(6,3413) HS(MS),HAWS(MS),LABEL(IUI1ST)

C$$$

formatsch=.6X,'SOURCE LENGTH=',F10.6,' AND WIDTH='

C$$$

2,F10.6,14,T79,1H*)

C$$$

HS(MS)=ULISTS+ULISTS+HS(MS)

C$$$

HAWS(MS)=ULISTS+ULISTS+HAWS(MS)

C$$$

IF(IUI1ST.NE.1) WRITE(6,3000)

C$$$

IF(IUI1ST.NE.1) WRITE(6,3413) HS(MS),HAWS(MS),LABEL(1)

C$$$

GO TO 3416

C$$$

WRITE(6,3414) HS(MS),HAWS(MS)

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)=WSM+CELP(C1+WSP+RPD)
3419 WRITE(6,3417) WS,WS
3417 FORMAT(2H*,6X,'THE SOURCE WEIGHT HAS MAGNITUDE='
3419 2,F10.6,' AND PHASE=',F10.6,T79,1H*)
3420 WRITE(6,3422)
3421 FORMAT(2H*,T6,'SOURCE=',T77,'INPUT LOCATION IN ',A6,T46,
3422 '2'ACTUAL LOCATION IN METERS',T79,1H*)
3423 WRITE(6,3422)
3424 YRITE(6,3006)
3425 FORMAT(2H*,6X,'THE FOLLOWING SOURCE ALIGNMENT IS USED:'
3426 T79,1H*)
3427 WRITE(6,3434)
3428 DO 3431
3429 I=1,3
3430 VXSS(I)=XQ(I)*VBI(I,MS)*XQ(I)*VRT(I,MS)*XQ(I)*VRT(I,MS)
3431 TQR=THSX*RPD
3432 PQR=PHSX*RPD
3433 XQ(I)=SIN(TQR)*COS(PQR)
3434 XQ(I)=COS(TQR)*SIN(PQR)
3435 XQ(I)=SIN(TQR)*COS(PQR)
3436 DO 3531
3437 I=1,3
3438 VXSS(I)=XQ(I)*VBI(I,MS)*XQ(I)*VRT(I,MS)*XQ(I)*VRT(I,MS)
3439 TQR=THSX*RPD
3440 PQR=PHSX*RPD
3441 XQ(I)=SIN(TQR)*COS(PQR)
3442 XQ(I)=COS(TQR)*SIN(PQR)
3443 XQ(I)=SIN(TQR)*COS(PQR)
3444 DO 3531
3445 I=1,3
3446 VXSS(I)=XQ(I)*VBI(I,MS)*XQ(I)*VRT(I,MS)*XQ(I)*VRT(I,MS)
3447 DOT=VXSS(I,I)*VXSS(I,3,MS)*VXSS(I,1,MS)
3448 VXSS(I,1,MS)=VXSS(I,1,MS)/DOT
3449 VXSS(I,2,MS)=VXSS(I,2,MS)/DOT
3450 VXSS(I,3,MS)=VXSS(I,3,MS)/DOT
3451 VXSS(I,1,MS)=VXSS(I,1,MS)*VXSS(I,1,MS)-VXSS(I,1,MS)*VXSS(I,1,MS)
3452 VXSS(I,2,MS)=VXSS(I,2,MS)*VXSS(I,2,MS)-VXSS(I,2,MS)*VXSS(I,2,MS)
3453 VXSS(I,3,MS)=VXSS(I,3,MS)*VXSS(I,3,MS)-VXSS(I,3,MS)*VXSS(I,3,MS)
3454 WRITE(6,3006)
3455 WRITE(6,3006)
3456 WRITE(6,3434)
FOULAT(2H,*.lX.3(9X,'VXSS(',Il,.'',Il,.'',Il,.')=',F~.6)

CO TO 3000

CONTINUE

SA: ERROR

CO=NUMBER OF ANTENNA ARRAY GROUPINGS.

MSAX=NUMBER OF ELEMENTS PER GROUPING.

LSMP=FALSE.

MSI=MSI+1

READ(6,+) MSAX

MSAT=MSAT+MSAX

IF(MSAT.GT.MSAX) WRITE(6,204) MSAT

IF(MSAT.GT.MSAX) STOP

WRITE(6,3805) MSI,MSAX

3805 F0ULAT(2H,*.6X,'THIS IS SOURCE NO. ',3S,' IN THIS',

2' COMPUTATION ',T79,1H+/2H*.6X,'THERE ARE ',

313,' SOURCES ARRAYED TOGETHER, ',T79,1H+)

WRITE(6,3006)

WRITE(6,3006)

MS(bIS)=TYPE OF LINEAR ANTEIIIIA

.H.T.O: ELECTRIC LINEAR ELEMENT

.G.T.O: MAGNETIC LINEAR ELEMENT

ABS(HMS)=1: UNIFORM CURRENT DISTRIBUTION

=2: STANDARD DIPOLC CURRENT DISTRIBUTION

=3: CAVITY BACKED SLOT CURRENT DISTRIBUTION (TE01)

HAMS(HS)=APERTURE WIDTH IN WAVELENGTHS (NOTE: IF

HAMS(HS) IS LESS THAN .1 LAMBD, SOURCE IS

CONSIDERED TO BE DIPOLC SOURCE

HS(HS)=LENGTH OF LINEAR ELEMENT IN WAVELENGTHS

THSZ,PHSZ=ORIENTATIOII ANGLES USED TO DEFINE LINEAR

ELE'EIIT AXIS.

THSZ.PHSZ=ORIENTATIOII ANGLES USED TO DEFINE APERTURE

PLANE OR DIPOLC X-AXIS.

WMS,WPS=MAGNITUDE AND PHASE OF EXCITATION OF

MA-TH ELEMENT.

MS=MSI

MAI=MSDI-MSAT+1

MAF=MAI+MSAX-1

MSA(1,MS)=MAI

MSA(2,MS)=MAF

DO 3811 MA=MAI,MAF

READ(6,+) (ISSN(MA),H=1,3)

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

READ(5,+) IMS(HS),HAMS(HS),HAMS(HS)

IF(IMSHS LT 0) WRITE(6,3411) IMS(HS)
IF(IMS(MS).GE.0) WRITE(6,3412) IMS(MS)
1497   WRITE(6,3006)
1498   IF(UIUNST.EQ.0) GO TO 3814
1499   UNSTS=UNIT(UIUNST)
1500   WRITE(6,3413) HS(MS),HAWS(MS),LABEL(UIUNST)
1501   IF(HS(MS).EQ.UNSTS+UNIT+HWS(MS))
1502   HAWS(MS)=UNSTS+UNIT+HWS(MS)
1503   IF(UIUNST.NE.1) WRITE(6,3006)
1504   WRITE(6,3414) HS(MS),HAWS(MS),LABEL(1)
1505   GO TO 3816
1506   WRITE(6,3415) HS(MS),HAWS(MS)
1507   WRITE(6,3006)
1508   WS(MS)=(1.,0.)
1509   WSAA=0.
1510   IASAI=0.
1511   ISAI=0.
1512   ISAZ=0.
1513   DO 3843 MA=MAI,MAF
1514   READ(6,*) WMS,WPS
1515   WRITE(6,3817) MA,WMS,WPS
1516   FORMAT(2H,*,' SOURCE ',I3,' HAS MAGNITUDE='
1517   2,F10.6,' AND PHASE=' F10.6, 'H')
1518   WS(MA)=WMS*CEXP(CJ*WPS*RPD)
1519   WMA=WMSA=WMS
1520   IASAI=IISAI=IWSA=IIS(1,MA)
1521   ISAI=ISAY=WMSA=IIS(2,MA)
1522   IIS(1,MS)=IASAI/WMSA
1523   IIS(2,MS)=ISAI/WMSA
1524   IIS(3,MS)=ISAZ/WMSA
1525   WRITE(6,3006)
1526   WRITE(6,3006)
1527   WRITE(6,3006)
1528   WRITE(6,3421) LABEL(UIUNIT)
1529   WRITE(6,3422)
1530   WRITE(6,3006)
1531   DO 3824 N=1,3
1532   3824  IQ(N)=ISS(N,MS)
1533   DO 3825 N=1,3
1534   3825  IQ(N)=ISS(N,MS)=UNITS*(IQ(1)+VR1(1,N)+IQ(2)+VR1(2,N)
1535   2*IQ(3)+VR1(3,N))+TR(N)
1536   WRITE(6,3426) MS,(IQ(N),N=1,3),(ISS(N,MS),N=1,3)
1537   DO 3829 MA=MAI,MAF
1538   DO 3827 N=1,3
1539   3827  IQ(N)=ISS(N,MA)
1540   DO 3828 N=1,3
1541   3828  IQ(N)=ISS(N,MA)=UNITS*(IQ(1)+VR1(1,N)+IQ(2)+VR1(2,N)
1542   2*IQ(3)+VR1(3,N))+TR(N)
1543   WRITE(6,3426) MA,(IQ(N),N=1,3),(ISS(N,MA),N=1,3)
1544   TQR=THSZ*RPD
1545   PQR=PHSZ*RPD
1546   IQ(1)=SIN(TQR)*COS(PQR)
1547   IQ(2)=SIN(TQR)*SIN(PQR)
1548   IQ(3)=COS(TQR)
1549   DO 3831 N=1,3
1550   3831  VXSS(3,N,MS)=IQ(1)+VR1(1,N)+IQ(2)*VR1(2,N)+IQ(3)*VR1(3,N)
1551   TQR=THSZ*RPD
1552   PQR=PHSZ*RPD
1553   IQ(1)=SIN(TQR)*COS(PQR)
1554   IQ(2)=SIN(TQR)*SIN(PQR)
1555   IQ(3)=COS(TQR)
1556   DO 3832 N=1,3
1557   3832  VXSS(1,N,MS)=IQ(1)+VR1(1,N)+IQ(2)+VR1(2,N)+IQ(3)+VR1(3,N)
1558   DX=VXSS(1,1,MS)+VXSS(3,1,MS)+VXSS(3,2,MS)+VXSS(3,3,MS)
1559   2*VXSS(1,3,MS)=VXSS(3,3,MS)
IF(A(3).GT.0.1) WRITE(6,3436)
IF(A(3).GT.0.1) STOP
\* LITE(0,3436)
V\*SS(1,1,MS)\*V\*SS(1,1,MS)-V\*SS(3,1,MS)+DZX
V\*SS(1,2,MS)\*V\*SS(1,2,MS)-V\*SS(3,2,MS)+DZX
V\*SS(1,3,MS)\*V\*SS(1,3,MS)-V\*SS(3,3,MS)+DZX
DOT=V\*SS(1,1,MS)\*V\*SS(1,1,MS)\*V\*SS(1,2,MS)\*V\*SS(1,2,MS)
DOT=\*RT(DOT)
V\*SS(1,1,lVlS)=V\*SS(1,1,MS)-V\*SS(3,1,KS)*DZX
V\*SS(1,3,MS)=V\*SS(1,3,MS)-V\*SS(3,3,MS)*DZX
DOT=V\*SS(1,1,MS)\*V\*SS(1,1,MS)+V\*SS(1,1,MS)\*V\*SS(1,2,NS)
2+vxss(l,3,b!S)*vxss(l,3,b!S)
DOT=\*RT(DOT)
V\*SS(l,l,MS)=V\*SS(l,l,\*tS)/DOT
V\*SS(l,2,MS)=V\*SS(1,2,MS)/DOT
V\*SS(l,3,MS)=V\*SS(1,3,MS)/DOT
V\*SS(2,1,MS)\*V\*SS(3,2,MS)\*V\*SS(1,3,MS)-V\*SS(3,3,MS)\*V\*SS(1,2,MS)
V\*SS(2,2,MS)\*V\*SS(3,3,MS)\*V\*SS(1,1,MS)-V\*SS(3,1,MS)\*V\*SS(1,3,MS)
V\*SS(2,3,MS)\*V\*SS(3,1,MS)\*V\*SS(1,2,MS)-V\*SS(3,2,MS)\*V\*SS(1,1,MS)
WRITE(6,3000)
WRITE(6,3000)
WRITE(6,3437)
DO 3833 HI=1,3
WRITE(6,3000)
WRITE(6,3000)
WRITE(6,3000)
WRITE(6,3000)
WRITE(6,3000)
WRITE(6,3000)
WRITE(6,3000)
WRITE(6,3434)
(HI,HJ,MS,V\*SS(HI,HJ,MS),HJ=1,3)
DO TO 3000
C===
3440 CONTINUE
C== PR: COMMAND ======
3444 C$$ IPRAD= 1 =NORMALIZATION FOR FAR ZONE AS follows
3445 C$$ TOTOL POWER RADIATED IN WATTS.
3446 C$$ PRAD CAN ALSO BE SPECIFIED AS THE POWER INPUT IN WATTS.
3447 C$$ NOTE IF PRAD IS LESS THAN OR EQUAL TO 1.E-30
3448 C$$ THAN LPRAD WILL BE SET FALSE
3449 C$$ IPRAD= TRUE
3450 IF(IPRAD.GT.4) STOP
3451 GO TO (3444,3446,3446,3447),IPRAD
3452 READ(6,*),IPRAD
3453 IF(IPRAD.GT.1.1E-30) GO TO 3442
3454 WRITE(6,3444) IPRAD
3455 FORMAT(2H TOTAL POWER RADIATED IN WATTS= ',12.6
3456 2,2H 2,779,1H+)
3457 GO TO 3000
3458 CONTINUE
3459 LPRAD= FALSE.
3460 PRAD=0.
3461 WRITE(6,3445)
3462 FORMAT(2H NO POWER RADIATED IS SPECIFIED',779,1H+)
3463 GO TO 3000
3464 CONTINUE
3465 IPRAD= 2 =MUTUAL IMPEDANCE CALCULATION Z12 = 221
3466 C$$ IPRAD = 2 =MUTUAL IMPEDANCE CALCULATION Z12 = 221
3467 C$$ CI1L = SOURCE TERMINAL CURRENT (REAL AND IMAGINARY)
3468 C$$ CI22 = RECEIVER TERMINAL CURRENT (REAL AND IMAGINARY)
3469 C$$ READ(6,*),CI1L,CI22
3470 WRITE(6,3446) CI1L,CI22
3471 4446 FORMAT(2H SOURCE TERMINAL CURRENT= ',2E12.6,779,1H+)
3472 2.2H +.6X, 'RECEIVER TERMINAL CURRENT= ',2E12.6,779,1H+)
3473 GO TO 3000
3474 CONTINUE
C## IPRAD = 3 - COUPLING VIA THE REACTION THEORY
1626 C## THIS GIVES A MODIFIED FRIS'S TRANSMISSION TYPE RESULT
1628 C##
1629 C## PRAD = POWER RADIATED BY THE SOURCE
1630 C## PRADR = POWER RADIATED BY THE RECEIVER AS IF IT WERE A SOURCE
1632 C##
1633 C## READ(6,+) PRAD,PRADR
1634 WRITE(6,4447) PRAD,PRADR
1635 4447 FORMAT(2H +,6X,' SOURCE POWER RADIATED= ',E12.6,779,1H+/
1636 2,6H +,6X,' RECEIVER POWER RADIATED= ',E12.6,779,1H+)
1637 GO TO 3000
1638
1639 3447 CONTINUE
1640 C##
1641 C## IPRAD = 4 - COUPLING BY THE LINVILLE METHOD
1642 C##
1643 C## CI11 = SOURCE TERMINAL CURRENT (REAL AND IMAGINARY)
1644 C## CI22 = RECIIVER TERMINAL CURRENT (REAL AND IMAGINARY)
1645 C##
1646 C## ZI1 = SOURCE TERMINAL IMPEDANCE (REAL AND IMAGINARY)
1647 C## ZI2 = RECEIVER TERMINAL IMPEDANCE (REAL AND IMAGINARY)
1648 C##
1649 C## READ(6,+) CI11,CI22
1650 WRITE(6,4445) CI11,CI22
1651 4445 FORMAT(6H CI11,CI22)
1652 C##
1653 C## READ(6,+) ZI1,ZI22
1654 WRITE(6,4446) ZI1,ZI22
1655 4446 FORMAT(6H ZI1,ZI22)
1656 C##
1657 C## C### MAX=NUMBER OF ANTENNA ELEMENTS.
1658 C##
1659 C## LACVR=. TRUE.
1660 C## LAMF=. FALSE.
1661 C## MAX=MRX+1
1662 C## XIAT=MRAT+MRX
1663 C##
1664 C## 4404 FORMAT(2H ***** NUMBER OF RECEIVERS= ',I3,PROGRAM',
1665 C## 2' ABOATS SINCE MAX. RECEIVER DIMENSION IS EXCEEDED. *****)
1666 C## IF(XIAT.GT.MRAT) STOP
1667 C## WRITE(6,4401) XIAT
1668 C## 4401 FORMAT(6H THIS IS RECEIVER NO. ',I3,' IN THIS',
1669 C## 2' COMPUTATION.' ,779,1H+)
1670 C## WRITE(6,3007)
1671 C## WRITE(6,3008)
1672 C##
1673 C##
1674 C## XBR(U,MR)=XYZ LOCATION OF MR-TH ANTENNA ELEMENT.
1675 C##
1676 C## IMR(MR)=TYPE OF LINEAR ANTENNA
1677 C##
1678 C## .LT.0: ELECTRIC LINEAR ELEMENT
1679 C## .GT.0: MAGNETIC LINEAR ELEMENT
1680 C##
1681 C## ABS(IMR)=1: UNIFORM CURRENT DISTRIBUTION
1682 C## =2: STANDARD DIPOLE CURRENT DISTRIBUTION
1683 C## =3: CAVITY BACKED SLOT CURRENT DISTRIBUTION (TE01)
1684 C##
1685 C## HAWR(MR)=APERTURE WIDTH IN WAVELENGTHS (NOTE: IF
1686 C## HAHR(MR) IS LESS THAN .1 LAMBD, RECEIVER IS
1687 C## CONSIDERED TO BE DIPOLE RECEIVER
1688 C## HR(MR)=LENGTH OF LINEAR ELEMENT IN WAVELENGTHS
1689 C##
1690 C## TH33,PH33=ORIENTATION ANGLES USED TO DEFINE LINEAR
1691 C## ELEMENT AXIS.
THRZ, PHX = ORIENTATION ANGLES USED TO DEFINE
PLANE OR DIPOLE X-AXIS.

WNR, WPR = MAGNITUDE AND PHASE OF EXCITATION OF
NR-TH ELEMENT.

WNR = WPR

IF(MR .LT. 0) WRITE(6, 4411) IMR(MR)
4411 FORMAT(2H *, 6X, 'THIS IS AN ELECTRIC RECEIVER OF TYPE ', I3
2, T79, 1H+)

IF(MR .GE. 0) WRITE(6, 4412) IMR(MR)
4412 FORMAT(2H *, 6X, 'THIS IS A MAGNETIC RECEIVER OF TYPE ', I3
2, T79, 1H+)

WRITE(6, 3002)

WRITE(0, 3000)
WRITE(6, 4421)
4421 FORMAT(2H *, 6X, 'RECEIVER LOCATION IN KILOMETERS', T79, 1H*)
WRITE(6, 4422)
4422 FORMAT(2H *, 7.6(' '), T16, 27(' '), T45, 27(' '),
2 T79, 1H+)
WRITE(6, 3006)
DO 4424 N=1, 3
4424 IQ(N)=IRR(N, MR) DO 4425 N=1, 3
4425 DO 4426 N=1, 3
4426 IF(MR=N) THEN
4427 DO 4426 N=1, 3
4428 IF(REC(N) .NE. REC(MR)) THEN
4429 DO 4426 N=1, 3
4430 IF(MR=N) THEN
4431 WRITE(0, 3000)
PQR = PHRX * RPD

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

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

xq(3) = COS(rq~)

DO 4431

N1=1,3

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

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

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

IF(ABS(DZX).GT.0.1) WRITE(6,4436)

4436 FORMAT(' *** PROGRAM ABORTS IN RECEIVER SECTION IN THAT THE',

2 ' COORDINATES ARE NOT ORTHOGONAL !!! ***')

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

VRR(I,1,MR) = VXRR(1,1,MR) - VXRR(1,2,MR) - VXRR(1,3,MR)

VRR(1,2,MR) = VXRR(1,2,MR) - VXRR(2,1,MR) - VXRR(2,2,MR) - VXRR(2,3,MR)

VRR(1,3,MR) = VXRR(1,3,MR) - VXRR(3,1,MR) - VXRR(3,2,MR) - VXRR(3,3,MR)

DOT = VXRR(1,1,MR) + VXRR(1,2,MR) + VXRR(1,3,MR)

2 * VXRR(1,3,MR) * VXRR(1,5,MR)

DOT = SQRT(DOT)

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

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

VRR(1,3,MR) = VXRR(1,3,MR) / DOT

VRR(2,1,MR) = VXRR(2,1,MR) + VXRR(2,3,MR) - VXRR(1,1,MR) - VXRR(3,1,MR)

VRR(2,2,MR) = VXRR(2,2,MR) + VXRR(2,3,MR) - VXRR(1,1,MR) - VXRR(3,2,MR)

VRR(2,3,MR) = VXRR(2,3,MR) + VXRR(3,1,MR) - VXRR(1,2,MR) - VXRR(3,3,MR)

WRITE(S,3006)

WRITE(S,3006)

WRITE(S,3006)

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

2, T79,1H+)

DO 4433 M1=1,3

WRITE(S,3006)

4433 WRITE(S,4434) (HI,HJ,MR,VRR(HI,HJ,MR),MJ=1,5)

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

2, T79,1H+)

GO TO 3000

C=====

4810 CONTINUE

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

C$$$

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

C$$$

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

C$$$

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.
INR(MR)=TYPE OF LINEAR ANTENNA
LT: 0: ELECTRIC LINEAR ELEMENT
GT: 0: MAGNETIC LINEAR ELEMENT
ABS(IMR)=1: UNIFORM CURRENT DISTRIBUTION
=2: STANDARD DIPOLE CURRENT DISTRIBUTION
=3: CAVITY BACKED SLOT CURRENT DISTRIBUTION (TE01)

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

TRX,PHR=ORIENTATION ANGLES USED TO DEFINE LINEAR
ELEMENT AXIS.

TRX,PHRX=ORIENTATION ANGLES USED TO DEFINE APERTURE
PLANE OR DIPOLE X-AXIS.

WPR=MAGNITUDE AND PHASE OF EXCITATION OF
MA-TH ELEMENT.

MR=WG1
MA=WGA*XRA*XRAT+1
MAF=MA+MRA-1
MRA(1,MR)=MAI
MRA(2,MR)=NAF
DO 4841 MA=MAI,MAF
4841 READ(6,*) (XRR(N,MA),N=1,3)
READ(6,*) TRX,PHR,TRX,PHRX
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(6,3005)
IF(IUINST.EQ.0) GO TO 4814
UNSTS=UNIT(IUNST)
WRITE(6,4413) HR(MR),HAWR(MR),LABEL(IUNST)
HR(MR)=UNSTS*UNITF*HR(MR)
HAWR(MR)=UNSTS*UNITF*HAWR(MR)
IF(IUINST.NE.1) WRITE(6,3006)
IF(IUINST.NE.1) WRITE(6,4413) HR(MR),HAWR(MR),LABEL(1)
GO TO 4818
4814 WRITE(6,4415) HR(MR),HAWR(MR)
4816 WRITE(6,3006)
WR(MR)=(1.,0.)
WRA=O.
XRAX=0.
XRAT=0.
XR2=0.
DO 4843 MA=MAI,MAF
4843 WRITE(6,3007) MA,WR,WF
READ(6,*) WPR
WRITE(6,4817) MA,WR,WPR
4817 FORMAT(2H *,6X,'RECEIVER ',13,' HAS MAGNITUDE='
2,F10.6,' AND PHASE=',F10.6,F10.6,1H+)
WR(MA)=WRA+CEXP(CJ*WPR/WPR)
WRA=WRA+WRR
XRAX=XRAX+WRR*XR(1,MA)
XRAT=XRAT+WRR*XR(2,MA)
XR2=XR2+WRR*XR(3,MA)
XR(1,MA)=XRAX/WRA
XR(2,MA)=XRAT/WRA
XR(3,MA)=XR2/WRA
WRITE(6,3006)
! ELIE (6, 4421) LABEL (UNIT)
WRITE (6, 4422)
WRITE (6, 4426)
DO 4824 N = 1, 3
4824 IQ (N) = IRR (N, MR)
DO 4826 N = 1, 3
4826 XRR (N, MR) = UNITS * (XQ (N) * VRT (N) + TR (N))
WRITE (6, 4426) N, (XQ (N), N = 1, 3), (IRR (N, MR), N = 1, 3)
DO 4829 M = NAI, MAF
DO 4827 N = 1, 3
4827 IQ (N) = IRR (N, MA)
DO 4828 N = 1, 3
4828 XRR (N, MA) = UNITS * (XQ (N) * VRT (N) * TR (N))
WRITE (6, 4426) MA, (XQ (N), N = 1, 3), (IRR (N, MA), N = 1, 3)
DO 4829 MA = MIA, MAF
DO 4827 N = 1, 3
4827 IQ (N) = IRR (N, MA)
WRITE (6, 4428)
TQR = THRZ * RPD
PQR = PHRZ * RPD
XQ (1) = SIN (TQR) * COS (PQR)
XQ (2) = SIN (TQR) * SIN (PQR)
XQ (3) = COS (TQR)
DO 4831 N = 1, 3
4831 VXRR (N, MR) = XQ (1) * VRT (N) * VRT (N) + VRT (N) * VXRR (N, MR)
TQR = THRZ * RPD
PQR = PHRZ * RPD
XQ (1) = SIN (TQR) * COS (PQR)
XQ (2) = SIN (TQR) * SIN (PQR)
XQ (3) = COS (TQR)
DO 4831 N = 1, 3
4831 VXRR (1, MR) = VXRR (1, MR) - VXRR (3, MR) * DZX
VXRR (2, MR) = VXRR (2, MR) - VXRR (3, MR) * DZX
VXRR (3, MR) = VXRR (3, MR) - VXRR (1, MR) * DZX
DO 4832 N = 1, 3
4832 VXRR (1, MR) = VXRR (1, MR) - VXRR (3, MR) * DZX
VXRR (2, MR) = VXRR (2, MR) - VXRR (3, MR) * DZX
VXRR (3, MR) = VXRR (3, MR) - VXRR (1, MR) * DZX
DO 4833 N = 1, 3
4833 WRITE (6, 4434) (NH, NJ, MR, VXRR (NH, NJ, MR), KJ = 1, 3)
DO TO 3000
C = CONTINUE
C = SN: COMMAND
C = PRAD = TOTAL POWER RADIATED IN WATTS
C = LPRA = TRUE.
C = READ (6, *) PRAD
C = WRITE (6, 3441) PRAD
C = WRITE (6, 3006)

175
C## MSX=NUMBER OF ANTENNA SEGMENTS
C## LSNP=.TRUE.
C## READ(6,+), MSX
C## IF(MSX.GT.MSDX) WRITE(6,3477) MSX
C## 3477 FORMAT(5,,,,,, 'NUMER OF SEGMENTS= ',3,
C## 2,' PROGRAM ABORTS SINCE MAX. SOURCE DIMENSION'
C## 3,' IS EXCEEDED. '******')
C## IF(MSX.GT.MSDX) STOP
C## WRITE(6,3451) MSX
C## 3451 FORMAT(2H *,6X,'THERE ARE ',3,' SEGMENTS IN THIS',
C## 2', COMPUTATION.',T79,1H+)
C## WRITE(6,3006)
C## WRITE(6,3006)
C## $$$$ XS(MS,N)=XYZ LOCATION OF MS-TH ANTENNA SEGMENT
C## $$$$ IMS(NS)=ELECTRIC LINEAR ELEMENT WITH A UNIFORM DISTRIBUTION
C## $$$$ HS(NS)=LENGTH OF LINEAR ELEMENT
C## $$$$ THS,Z,PHS=ORIENTATION ANGLES USED TO DEFINE
C## $$$$ LINEAR ELEMENT AXIS.
C## $$$$ WMS,WPS=REAL AND IMAGINARY CURRENT WEIGHT.
C## $$$$ WRITE(6,3458) LABEL(IWAIT)
C## 3458 FORMAT(2H*,77,'MS'.T13.'HS'.T50,'A6,T23,'HS-METERS',
C## 2T41,'INPUT: THS,PHS',T60,'ACTUAL: THS,PHS',T79,1H+)
C## WRITE(6,3459)
C## 3459 FORMAT(2H *,76,3(''),T12,20(''),T60,1H+)
C## 217('..'),T79,1H+)
C## WRITE(6,3006)
C## DD 3463 MS=1,MSI
C## READ(6,+),(XSS(N,MS),N=1,3),HS(NS),THS,Z,PHS
C## NSA(1,NS)=0
C## NSA(2,NS)=0
C## IMS(NS)=1
C## HAYS(NS)=0.
C## HS=HS(NS)
C## HS(NS)=UNITS*HSP
C## TQ=.90.-THSZ
C## PQ=PHSZ
C## XQ(1)=SIN(TQ*RDP)*COS(PQ*RDP)
C## XQ(2)=SIN(TQ*RDP)*SIN(PQ*RDP)
C## XQ(3)=COS(TQ*RDP)
C## DD 3481 N=1,3
C## 3481 FORMAT(2H *,76,3(''),T12,20(''),T60,1H+)
C## VQ=VQ(XQ)
C## XQ(1)=XQR(1)+VQ(1)*VQ(2)+VQ(2)*VQ(3)
C## 3493 TDSZ=PR+BTAN2(SQRT(XQR(1)+XQR(1)+XQR(2)+XQR(2)),XQR(3))
C## 3494 PHZ=PR+BTAN2(XQR(2),XQR(3))
C## WRITE(6,3464) MS,HS,HS(NS),TQ,PQ,THS,Z,PHS
C## 3464 FORMAT(2H *,76,3(''),T12,20(''),T60,1H+)
C## VQ(3)=VQ(XQ)
C## VXSS(3,H,MS)=XQR(3)
C## 2,179,1H+)
C## DD 3481 N=1,3
C## 3481 FORMAT(2H *,76,3(''),T12,20(''),T60,1H+)
C## VQ(3)=VQ(XQ)
C## VXSS(3,H,MS)=XQR(3)
C## 2000 VSSS(1,1,MS)=COS(THSZ*RDP)+COS(PHSZ*RDP)
C## 2001 VSSS(1,2,MS)=COS(THSZ*RDP)*SIN(PHSZ*RDP)
C## 2002 VSSS(1,3,MS)=-SIN(THSZ*RDP)
C## 2003 VSSS(2,1,MS)=-SIN(PHSZ*RDP)
C## 2004 VSSS(2,2,MS)=COS(PHSZ*RDP)
C## 2005 VSSS(2,3,MS)=.0.
C## DD 3463 CONTINUE
C## WRITE(6,3006)
2008 WRITE(6,3006)
2009 WRITE(6,3008)
2010 WRITE(6,3484)
2011 3464 FORMAT(2H *,T31,'SEGMENT COORDINATES',T79,*)
2012 WRITE(6,3006)
2013 WRITE(6,3008)
2014 WRITE(6,3466) LABEL(IUNIT)
2015 3466 FORMAT(2H *,T7,'NS',T14,'INPUT LOCATION IN ',A6,
2016 2T43,'ACTUAL LOCATION IN METERS',T79,*)
2017 WRITE(6,3457)
2018 3457 FORMAT(2H *,T6.3(*-'),T13.26(*-'),T42.27(*-'),T79,*)
2019 WRITE(6,3006)
2020 DO 3473 MS=1,HSX
2021 DO 3474 N=1,3
2022 3474 FORMAT(I3=ISS(N,MS))
2023 DO 3475 N=1,3
2024 3475 ISS(N,MS)=UNITS*I1Q(I1)+VRH(I1,N)*VRH(N)+WKS(N)
2025 2+1Q(3)+VRH(3,N)+TR(N)
2026 WRITE(6,3476) MS,(ISS(N,MS),N=1,3), (ISS(N,MS),N=1,3)
2027 3476 FORMAT(2H *,T6.13,T13,F6.3,2(' ',',F6.3),T42,F6.3,
2028 22(' ',',F6.3),T79,1H+)
2029 3473 CONTINUE
2030 WRITE(6,3008)
2031 WRITE(6,3008)
2032 WRITE(6,3465)
2033 3465 FORMAT(2H *,T33,'CURRENT WEIGHTS',T79,1H+/,2H *,T7,'NS',T18,
2034 2'REAL',T31,'IMAG',',T46,'MAC',',T67,'PHASE',T79,1H+)
2035 WRITE(6,3466)
2036 3466 FORMAT(2H *,T6.3(*-'),T17.6(*-'),T30.7(*-'),T46.6(*-'),
2037 2T6.7(*-'),T79,1H+)
2038 DO 3465 MS=1,MSI
2039 READ(5,+) WKS,WPS
2040 WS(NS)=CMPLX(WMS,WPS)
2041 WPS=PRAD(WMS,WPS)
2042 WPP=DIR*BAH2(WPS,WPS)
2043 WRITE(6,3466) MS,WKS,WPS,WPS,WPP
2045 3465 CONTINUE
2046 WRITE(6,3008)
2047 GO TO 3000
2048 C=====
2049 4460 CONTINUE
2050 C== RM: COMMAND ======
2051 C###
2052 C### PRADR=TOTAL POWER RADIATED IN WATTS
2053 C###
2054 READ(5,+) PRADR
2055 WRITE(6,3441) PRADR
2056 WRITE(6,3006)
2057 C###
2058 C### MAX=NUMBER OF ANTENNA SEGMENTS
2059 C###
2060 C### LACVR=.TRUE.
2061 LACV=.TRUE.
2062 READ(5,+) MAX
2063 IF(MAX.GT.MDI) WRITE(6,4477) MAX
2064 4477 FORMAT(' ***** NUMBER OF SEGMENTS= ',I3,
2065 2' PROGRAM ABORTS SINCE MAX. RECEIVER DIMENSION'
2066 3,' IS EXCEEDED. ******')
2067 IF(MAX.GT.MDI) STOP
2068 WRITE(6,3461) MAX
2069 WRITE(6,3006)
2070 WRITE(6,3006)
2071 C###
LOCATION OF K-th 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(6,4460) LABEL(IUNIT)

FORMAT(2H *.T7,'HR',T13,'HR:','A6,T23,'HR:','T79,1H+)

WRITE(6,3459)

FORMAT(2H *.T7,'HR',T13,'HR:','A6,T23,'HR:','T79,1H+)

WRITE(6,3469)

WRITE(6,3476) LOC(5,1,MR)

WRITE(6,3486)

WRITE(6,3476) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)

WRITE(6,3486) LOC(5,1,MR)
WRITE(6,3006)
WRITE(6,3006)
WRITE(6,3465)
WRITE(6,3486)
DO 4466 IJR.I=1,MRX
READ(6,*), WMR, WPR
WMR=CMPLX(WMR, WPR)
WPP=CMPLX(WPR, WMR)
WRITE(6,3486) MR, WMR, WPR, WPP
WRITE(6,3466)
WRITE(6,3006)
GO TO 3000
C=.=r==
3490 CONTINUE
C=NS:
COMMAND ===
C$$:
INITIALIZE SOURCE DATA.
C$$:
LUMP=.FALSE.
MSX=0
MSAT=0
WRITE(6,3491)
3491 FORMAT(1H*).6X,' THE SOURCE DATA IS INITIALIZED.',177,1H*/
2,1H *.6X,' NO SOURCES ARE PRESENTLY IN THE PROBLEM.'
3,1H,1H*)
GO TO 3000
C==r==
3496 CONTINUE
C== NR: COMMAND ===
C$$:
LACVR=.FALSE.
LMPG=.FALSE.
MRX=0
WRITE(6,3496)
3496 FORMAT(1H*).6X,' THE RECEIVER DATA IS INITIALIZED.'
2,1H,1H*/2H *.6X,' NO RECEIVERS ARE PRESENTLY IN THE'
3,' PROBLEM.',177,1H*)
GO TO 3000
C==r==
3500 CONTINUE
C== LP: COMMAND ===
C$$:
WRITE=TRUE IF LINE PRINTER OUTPUT OF DATA IS DESIRED
C$$:
READ(5,*) LWRITE
2184 IF(.NOT.LWRITE) WRITE(6,6605)
2185 6605 FORMAT(1H*,6X,' NO LINE PRINTER OUTPUT',177,1H*)
2186 IF(.NOT.LWRITE) GO TO 3000
2187 WRITE(6,3501)
2188 3501 FORMAT(1H*,6X,' DATA WILL BE OUTPUT ON LINE PRINTER 111',
2189 2TT0,1H*)
2190 GO TO 3000
C==r==
3500 CONTINUE
C== VP: COMMAND ===
C$$:
VOLUMETRIC DUMP FOR PLOTS
C$$:
READ(6,*) LVPLT
2198 IF(.NOT.LVPLT) WRITE(6,6605)
2199 IF(.NOT.LVPLT) GO TO 3000
GO TO 3651
C=======
3600 CONTINUE
C== PP: COMMAND
C===
C### LPLT=TRUE IF PEN PLOTTER OUTPUT IS DESIRED
C###
2006 C###
2007 READ(5,*) LPLT
2008 IF(.NOT.LPLT) WRITE(6,6806)
2009 6806 FORMAT(2H16,6X,'NO PEN PLOT DESIRED',779,1H+)
2010 IF(.NOT.LPLT) GO TO 3000
C###
2011 C###
2012 C### IF LPLT=TRUE READ IN DIMENSIONS
2013 C###
2014 C### LPREC = TRUE IMPLIES RECTANGULAR PLOT
2015 C###
2016 C### PPXL = LENGTH OF X-AXIS (ANGLE AXIS)
2017 C###
2018 C### PPYL = LENGTH OF Y-AXIS (DB AXIS)
2019 C###
2020 C### LPREC = FALSE IMPLIES POLAR PLOT
2021 C###
2022 C### PPVL = ANGULAR POSITION OF X-AXIS
2023 C###
2024 C### PPYL = RADIUS OF GRID
2025 C###
2026 3051 READ(6,*) LPREC,PPXL,PPYL
2027 C###
2028 C### PPXB = BEGINNING VALUE OF X-AXIS
2029 C###
2030 C### PPYE = END VALUE OF X-AXIS
2031 C###
2032 C### PPXS = STEP SIZE OF X-AXIS GRID MARKS
2033 C###
2034 READ(6,*) PPXB,PPXE,PPXS
2035 3602 FORMAT(2H16,6X,'DATA WILL BE OUTPUT FOR A PLOT!!'
2036 1779,1H+),
2037 1800,6X)
2038 WRITE(6,3006)
2039 3903 FORMAT(2H16,6X,'LPPREC,PPI',PPYL,PPYL)
2040 4904 FORMAT(2H16,6X,'LPPREC=',L2,6X,'PPXL=',F10.6,6X,
2041 2'PPYL=',F10.6,1779,1H+),
2042 WRITE(6,3004) PPXB,PPXE,PPXS
2043 3905 FORMAT(2H16,6X,'PPXB=',F10.6,6X,'PPYE=',F10.6,6X,
2044 2'PPXE=',F10.6,1779,1H+),
2045 WRITE(6,3005) PPXB,PPYE,PPYS
2046 3906 FORMAT(2H16,6X,'PPYA=',F10.6,6X,'PPYE=',F10.6,6X,
2047 2'PPYS=',F10.6,1779,1H+),
2048 IF(LPLT) GO TO 3000
2049 WRITE(6,3006)
2050 C###
2051 C### IVTYP=TYPE OF RESULTS OUTPUT
2052 C###
2053 C### IVTYP=1 ELECTRIC FIELD OUTPUT
2054 C###
2055 C### IVTYP=2 MAGNETIC FIELD OUTPUT
2056 C###
2057 C### IVTYP=3 BOTH ELECTRIC AND MAGNETIC FIELDS OUTPUT
2058 C### COUPLING IS OUTPUT IF RECEIVER IS DEFINED FOR ANY IVTYP
2059 C###
2060 C###
2061 C### IVPOL=POLARIZATION OF RESULTS OUTPUT
2062 C###
2063 C### IVPOL=1,2,3 THEN R,THETA,PHI OR X,Y,Z RESPECTIVELY IS OUTPUT
2064 C###
2065 C### IVPOL=4 THEN R-THETA OR X-Y ARE OUTPUT
2066 C###
2067 C### IVPOL=6 THEN R-PHI OR X-Z ARE OUTPUT
2068 C###
2069 C### IVPOL=6 THEN THETA-PHI OR Y-Z ARE OUTPUT
2070 C###
2071 C### IVPOL=7 THEN R, THETA AND PHI OR X, Y AND Z ARE OUTPUT
2072 C###
2073 C### COUPLING HAS NO POLARIZATION
2074 C###
READ(6,*) IVTTP, IVPOL
WRITE(6,3666) IVTTP, IVPOL
GO TO 3000
C-------
3700 CONTINUE
C==n
GP: CONMAID
C$SS
INFINITE CROUND PLANE EFFECT INCLUDED.
L=1. TRUE.
DO 3702 N=1,3
XX(N,1,MPDX)=1.E6*(VRT(1,N)*VRT(2,N)+TR(N)
XX(N,2,MPDX)=1.E6*(-VRT(1,N)*VRT(2,N)+TR(N)
XX(N,3,MPDX)=1.E6*(-VRT(1,N)+VRT(2,N)+TR(N
3702 XX(N,4,MPDX)=1.E6*(VRT(1,N)-VRT(2,N)+TR(N
C$SS
LSLAB= 0 IMPLIES METAL PLATE, AND
=3 IMPLIES DIELECTRIC HALF SPACE
C$SS
NOTE: IF DIELECTRIC COVERED, ONE MUST READ DIELECTRIC DATA.
C$SS
READ(6,*) LSLAB(NPDI)
IF(LSLAB(NPDI).EQ.0) WRITE(6,3708)
3708 FORMAT(2H6X,'PERFECTLY CONDUCTING',T79,1H*)
IF(LSLAB(NPDI).NE.0) WRITE(6,3707)
3707 FORMAT(2H6X,'SEMI-INFINITELY THICK DIELECTRIC',T79,1H*)
WRITE(6,3701)
3701 FORMAT(2H6X,'INFINITE CROUND PLANE INSERTED IN',
2' STRUCTURE 111',T79,1H*)
WRITE(6,3006)
3006 FORMAT(2H6X,'THE ORIGIN IS AT ',F12.6,' ',F12.6
2, ', F12.6,' METERS',T79,1H*)
WRITE(6,3008)
3008 FORMAT(2H6X,'THE NORMAL IS ',F12.6,' ',F12.6,' ',
2.F12.6,T79,1H*)
IF(LSLAB(NPDI).EQ.0) GO TO 3000
3004 N=1
3005 N=1
3006 N=1
3007 C$SS
ERSLAB(1,NPDI)=RELATIVE DIELECTRIC CONSTANT
C$SS
TESLAB(1,NPDI)=DIELECTRIC LOSS TANGENT
C$SS
URSLAB(1,NPDI)=RELATIVE PERMEABILITY CONSTANT
C$SS
TMNLAB(1,NPDI)=RELATIVE PERMUTABILITY LOSS TANGENT
C$SS
READ(6,*) ERSLAB(1,NPDI),TESLAB(1,NPDI)
2,URSLAB(1,NPDI),(TMSLAB(1,NPDI)
WRITE(6,3006)
WRITE(6,3708)
3708 FORMAT(2H6X,'DIELECTRIC',3X,'LOSS',4X,
2'PERMUTIVITY',3X,'LOSS',T79,1H*,/.
32H +,6X,'CONSTANT',3X,'TANGENT',
44X,'CONSTANT',3X,'TANGENT',T79,1H*,/.
E2H +,6X,'---------',2X,'---------',T79,1H*)
WRITE(6,3709) ERSLAB(1,NPDI),TESLAB(1,NPDI)
2,URSLAB(1,NPDI),(TMSLAB(1,NPDI)
181
3709 FORMAT(2H5.,6X,F10.4,2X,F7.4,2X,F11.4,2X,2F7.4,179,1H*)
3760 CONTINUE
3761 FORMAT(2H5.,6X,F10.4,2X,F7.4,2X,F11.4,2X,2F7.4,179,1H*)

GO TO 3000

CSE.===

GO:

CObNABD===---

cast

C$$$

INITIALIZE GROUlID PLANE DATA.

cast

LCRllD=.FALSE.

BRITE(6,3761)

3761 FORMAT(2H5.,6X,F10.4,2X,F7.4,2X,F11.4,2X,2F7.4,179,1H*)

GROUND PLANE DATA IS INITIALIZED. ' .T78,1H= /

3741 3.T79,1H=)

GO TO 3000

C===

3000 CONTINUE

C== PI.

C$$$

TR(N)=LINEAR TRANSLATION OF COORDINATES FROM THE FIXED

C$$$

COORDINATES WHICH IS ORIGINALLY SET UP BY OPERATOR.

C$$$

READ(6,*),(TR(N),N=1,3)

WRITE(6,3901) LABEL(IUNIT),(TR(N),N=1,3)

3901 FORMAT(2H5.,6X,'TRANSLATION IN ',A8,

2' 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)

C$$$

READ(6,*),(THZP,PHZP,THXP,PHXP)

VRT(3,1)=SIN(THZP*RPD)*COS(PHZP*RPD)

VRT(3,2)=SIN(THZP*RPD)*SIN(PHZP*RPD)

VRT(3,3)=COS(THZP*RPD)

VRT(1,1)=SIN(THXP*RPD)*COS(PHXP*RPD)

VRT(1,2)=SIN(THXP*RPD)*SIN(PHXP*RPD)

VRT(1,3)=COS(THXP*RPD)

C$$$

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

3931 FORMAT(2H5.,6X,'*** PROGRAM ABORTS IN ROTATE SECTION IN THAT THE',

2' COORDINATES ARE NOT ORTHOGONAL!!! ***')

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

3931 FORMAT(2H5.,6X,'*** 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

3932 DOT=VRT(1,1)*VRT(1,1)+VRT(1,2)*VRT(1,2)+VRT(1,3)*VRT(1,3)

3934 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(3,2)*VRT(1,1)-VRT(3,3)*VRT(1,2)

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

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

WRITE(6,3931)
THE FOLLOWING ROTATIONS ARE USED FOR ALL

SUBSEQUENT INPUTS:

DO 3932 NI=1,3

WRITE(6,3006)

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

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

FOR ALL SUBSEQUENT INPUTS:',779,1H*)

GO TO 3000

C=====

4000 CONTINUE

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

C### CYLINDER GEOMETRY INPUT

C###

LCTL=.TRUE.

MCX=MCI+1

NC=MCX

IF(MCI.GT.NCDX) WRITE(6,6311) NCI

6311 FORMAT(' *** NUMBER OF CYLINDERS= ',I3,'PROGRAM',

2' ABORTS SINCE MAX. CYLINDER DIMENSION IS EXCEEDED. *****'1

IF(MCI.GT.NCDX) STOP

C###

ICL(N,MC)=XTZ LOCATION OF THE ORIGIN OF THE NC-TH CYLINDER

C###

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

DO 6301 N=1,3

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

DO 6302 N=1,3

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

2*IQ(3)+VRT(3,N)+TR(N)

WRITE(6,6308) LABEL(IUNIT)

6308 FORMAT(2H*,'CYLINDER',T17,'INPUT LOCATION IN ',A6

1,146,'ACTUAL LOCATION IN METERS'.T79,1H*)

WRITE(6,3411)

WRITE(6,3006)

WRITE(6,3420)

MC,(XI(N),N=1,3),(ICL(N,MC),N=1,3)

WRITE(6,3006)

WRITE(6,3006)

C### ICLZ,PCLZ=ORIENTATION OF THE CYLINDER

AXIS

C###

C### TCLZ,PCLZ=ORIENTATION OF THE CYLINDER'S X-AXIS

C###

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

TCLI=SQRT((TCLZ*RPD)**2+(TCLI*RPD)**2)

XQ(I)=TCLI*SQRT((TCLX*RPD)**2+(TCLY*RPD)**2)

DO 6303 N=1,3

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

2*XQ(3)+VRT(3,N)+TR(N)

WRITE(6,6308) LABEL(IUNIT)

DO 6304 N=1,3

6304 VCL(1,N,MC)=XQ(1)*VRT(1,N)+XQ(2)*VRT(2,N)+XQ(3)*VRT(3,N)

2*XQ(1)+VRT(1,N)+TR(N)

IF(ABS(DZX).LT.1) WRITE(6,6306)

6306 FORMAT(' *** PROGRAM ABORTS THE COORDINATES ARE NOT',

2' ORTHOGONAL!!! ***')

IF(ABS(DZX).GT.0.1) STOP
DOT=SRT(DOT)

VCL(1,1,MC)=VCL(1,1,MC)/DOT
VCL(1,2,MC)=VCL(1,2,MC)/DOT
VCL(1,3,MC)=VCL(1,3,MC)/DOT
VCL(2,1,MC)=VCL(2,2,MC)-VCL(1,1,MC)*VCL(3,3,MC)*VCL(1,1,MC)
VCL(2,2,MC)=VCL(2,2,MC)-VCL(1,1,MC)*VCL(3,3,MC)*VCL(1,1,MC)
VCL(2,3,MC)=VCL(2,3,MC)-VCL(1,1,MC)*VCL(3,3,MC)*VCL(1,1,MC)

WRITE(6,6309)

6309 FORMAT(2H +,6X, 'THE FOLLOWING CYLINDER ALIGNMENT IS USED:

2,779,1H+)

DO 6308 NI=1,3

WRITE(6,3008)

6308 WRITE(6,6307) (HI, NJ, VCL(HI, NJ, MC), NJ=1,3)

6307 FORMAT(2H +, 11, 3(2X, 'VCL(', HI, ',', NJ, ')=', F9.6), T79, 1H+)}

470 C###

471 C### AC=RADIUS OF ELLIPSE ON X CYLINDER AXIS

472 C### BC=RADIUS OF ELLIPSE ON Y CYLINDER AXIS

473 C###

474 C### ZCN,THTN=MOST NEGATIVE ENDCAP'S Z COMPONENT

476 C### AND ANGLE OF SURFACE WITH THE CYLINDER AXIS

478 C### AND ANGLE OF SURFACE WITH THE CYLINDER AXIS

6310 FORMAT(2H +, 6X, 'X AXIS DIMENSION IN ',

2497 2A6='=', F8.3, T79, 1H+)

WRITE(6,3006)

6310 WRITE(6,6310) LABEL(IUNIT), AAC

6310 IF(IUNIT.NE.1) WRITE(6,6310) LABEL(1), AC(1,MC)

6310 IF(IUNIT.NE.1) WRITE(6,3006)

6310 WRITE(6,6310) LABEL(IUNIT),BBB

6320 FORMAT(2H +, 6X, 'Y AXIS DIMENSION IN ',

2497 2A6='=', F8.3, T79, 1H+)

WRITE(6,3006)

6320 IF(IUNIT.NE.1) WRITE(6,6320) LABEL(1), BC(1,MC)

6320 IF(IUNIT.NE.1) WRITE(6,3006)

6320 WRITE(6,6320) LABEL(IUNIT), ZCH

6330 FORMAT(2H +, 6X, 'MOST NEGATIVE ENDCAP Z COMPONENT IN ',

2497 2A6='=', F8.3, T79, 1H+)

WRITE(6,3006)

6330 IF(IUNIT.NE.1) WRITE(6,6330) LABEL(1), ZC(2,MC)

6330 IF(IUNIT.NE.1) WRITE(6,3006)

6330 WRITE(6,6330) LABEL(IUNIT), THTN

6340 FORMAT(2H +, 6X, 'ANGLE OF NEG. ENDCAP SURFACE WITH NEG.',

2497 2A6='=', F8.3, T79, 1H+)

WRITE(6,3006)

6340 WRITE(6,3006)
2620 WRITE(8,6350) LABEL(IUNIT),ZCP
2621 6350 FORMAT(2H +,6I,'MOST POSITIVE END CAP Z COMPONENT IN ',
2622 2H,=",F8.3,T79,1H+"
2623 WRITE(8,3006)
2624 IF(IUNIT.NE.1) WRITE(6,6350) LABEL(1),ZC(1,MC)
2625 IF(IUNIT.NE.1) WRITE(6,3006)
2626 WRITE(6,6350) THTP
2627 6350 FORMAT(2H +,6I,'ANGLE OF POS. END CAP SURFACE WITH POS.',
2628 2H'CYL. AXIS '='F8.3,T79,1H+)
2629 GO TO 3000
2630 6400 CONTINUE
2631 READ(6,*) NEC(MC)
2632 IF(NEC(MC) .GT. MCXD) STOP
2633 NEC=NEC(MC)
2634 DO 6410 MC=1,NECI
2635 READ(6,*) AC(NC,MC),BC(NC,MC),ZC(NC,MC)
2636 AAD=AC(NC,MC)
2637 BBD=BC(NC,MC)
2638 ZCH=ZC(NC,MC)
2639 AC(NC,MC)=AC(NC,MC)*UNITS
2640 BC(NC,MC)=BC(NC,MC)*UNITS
2641 ZC(NC,MC)=ZC(NC,MC)*UNITS
2642 TCR(NC,MC)=0.5*PI
2643 WRITE(6,3305) MC,NC,AAD,BBD,ZCH,AC(NC,MC),BC(NC,MC),ZC(NC,MC)
2644 6410 CONTINUE
2645 GO TO 3000
2646 C===
2647 4050 CONTINUE
2648 C=== NC: COMMAND ======
2649 C###
2650 C### INITIALIZE CYLINDER DATA.
2651 C###
2652 LCTL=.FALSE.
2653 MCX=0
2654 WRITE(6,4061)
2655 4061 FORMAT(2H *,6I,'CYLINDER DATA IS INITIALIZED. ',79,1H+)
2656 2H *,6I,'NO CYLINDER IS PRESENTLY IN THE PROBLEM. '
2657 3H,1H+)
2658 GO TO 3000
2659 C===
2660 997 CONTINUE
2661 C=== EN: COMMAND ======
2662 C###
2663 C### END PROGRAM
2664 C###
2665 WRITE(6,3006)
2666 WRITE(6,3006)
2667 WRITE(6,3006)
2668 STOP
2669 C===
2670 3800 CONTINUE
2671 C=== X: COMMAND ======
2672 C###
2673 C### EXECUTE PROGRAM
2674 C###
2675 WRITE(6,3006)
2676 WRITE(6,3006)
2677 C### 2. INITIALIZATION SECTION
2678 C###
2679 W1=.2997926/FRQC
2680 WX=TP/WL
2681 WRITE(6,3008)
2682 MPX=MXP

185
CIII GROUND PLANE IS ANOTHER PLATE IN SOLUTION.

2584 IF(LGRND) MPXR=MPY+1
2586 IF(MPXR.GT.MPDX) WRITE(6,001) MPXR
2587 IF(MPXR.GT.MPDX) STOP
2588 IF(.NOT.LGRND) GO TO 3801
2589 LPLA=.TRUE.
2590 MFP(MPXR)=4
2591 DO 3802 I=1,4
2592 DO 3802 N=1,3
2593 3802 IX(N,I,MPXR)=IX(N,I,MPDX)
2594 LSLAB(MPXR)=LSLAB(MPDX)
2595 DSLAB(1,MPXR)=DSLAB(1,MPDX)
2596 ESLAB(1,MPXR)=ESLAB(1,MPDX)
2597 TESLAB(1,MPXR)=TESLAB(1,MPDX)
2598 URSLAB(1,MPXR)=URSLAB(1,MPDX)
2599 TLSLAB(1,MPXR)=TLSLAB(1,MPDX)
2600 CONTINUE
2601 IF(MPXR.EQ.0) LPLA=.FALSE.
2602 IF(LPLA) CALL GEOM
2603 CIII MAKE PATTERN INFORMATION MATCH WITH SHADOW CODE
2604 IF(IVPN.GT.0) THEN
2605 NVFT=NVY
2606 NVFP=NPY
2607 ELSE
2608 NVFT=NVY
2609 NVFP=NPY
2610 ENDIF
2611 COLS=NVFT
2612 THE1=TY1*RPD
2613 RESTH=TY1*RPD
2614 THE2=THE1+(COLS-1)*RESTH
2615 ROWS=NVFP
2616 PHI1=PZ1*RPD
2617 RESPH=PZ1*RPD
2618 PHI2=PHI1+(ROWS-1)*RESPH
2619 CIII MAKE SOURCE INFORMATION FOR THE FIRST ONE MATCH WITH
2620 CIII SHADOW CODE
2621 DO 3806 N=1,3
2622 3806 ANTE(N)=ASS(N,1)
2623 RETURN
2624 END
FUNCTION BABS

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

```
0001  COMMON/LIMIT,SMLR,SMLT,BIG
0002    X=ABS(REAL(Z))
0003    Y=ABS(AIMAG(Z))
0004    IF(X.LT.SMLT.AND.Y.LT.SMLT) GO TO 10
0005    BABS=CABS(Z)
0006    RETURN
0007   10  BABS=SMLT
0008    RETURN
0009    END
```
This is constant block data.

```plaintext
0001  C-----------------------------------------------
0002  BLOCK DATA
0003  C!!! LOAD COMMONLY USED DATA INTO COMMON AREA.
0004  C!!!
0005
0006  COMPLEX CJ,CPI4
0007  COMMON/FIS/PI,TPI,DPR,RPD
0008  COMMON/COMP/CJ,CPI4
0009  COMMON/LIMIT/SML,SMLR,SMLT,BIG
0010  DATA PI,TPI,DPR,RPD/3.14159266,6.28318531,67.29577956
0011    20.0174632928/
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                      \ This routine is used to compute the arctangent. It is
0003 C                      \ similar to ATAN2 except it avoids the run time errors.
0004 C                      \---------------------------------------------------------------
0005 COMMON/PIS/PI,TP1,DPR,BPD
0006 COMMON/LIMIT/SML,SMLT,BIG
0007 IF(ABS(X).GT.SMLT) GO TO 60
0008 IF(ABS(Y).GT.SMLT) GO TO 10
0009 BTAN2=0.
0010 RETURN
0011 10 BTAN2=0.5*PI
0012 RETURN
0013 60 BTAN2=ATAN2(Y,X)
0014 RETURN
0015 END
```

189
SUBROUTINE CAPINT

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

003   C  INCLUDE 'SHACOH.FOR'
013   LOCAL LHIT,LDBUG,LMET,LWARN
016   COMMON/LIMIT/SM,SM,L,SLIT,BIG
018   COMMON/WAVE/WK, WL
017    LHIT=.FALSE.
018    DHT=.D0.
019    C  STEP THROUGH CYLINDERS
018    DO 40 MCC=l,MCX
019     IF(MH.LT.0 .AND. ABS(MH).NE.MCC) GO TO 40
019     IF(MH.GT.0 .AND. MH.EQ.NCC) GO TO 40
020     CALL CYLROT(D,DC,l,NCC)
021     CALL CYLROT(XIS,XISC,2,MCC)
022    C  STEP THRU ENDCAPS
022    NECX=NECX(NCC)
022    DO 40 MCC=l,MCX
023     IF(MD.LT.0 .AND. ABS(MD).NE.MN) GO TO 40
023     IF(MD.GT.0 .AND. MD.EQ.MN) GO TO 40
024     A=AC(MN,MCC)
024     B=BC(MN,MCC)
025     CMC=COS(TCR(MC,MCC))
025     SMC=SQRT(1-RC(RC,MC,MCC))
026     ALL=-XISC(1)*CMC*(XISC(3)-ZC(M,MCC))*SMC
027     DLL=-CMC*DC(1)+SMC*DC(3)
026     C DOES RAY HIT END CAP PLANE?
027     IF(AN*DN.LE.O.) GO TO 40
028     RHOT=XT(1)*XT(1)*XT(2)*XT(2)
029     RHOT=SR(WH,XT(3)-ZC(M,MCC))*(XT(3)-ZC(M,MCC))
029     RHOT=SQRT(RHOT)
030     AE*A/SQRT(B)*B
030     IS HIT POINT ON ENDCAP?
031     IF(RHOT.GT.AE .AND. RHOH.GT.B) GO TO 40
031     VE=BIAN2(A*XT(2),B*XT(1))
032     CVE=COS(VE)
032     SVE=SIN(VE)
033     RHOT=SQRT(AE+AE+CVE+CVE+B+B+SVE+SVE)
034     IF(RHOT.GT.RHOH) GO TO 40
035     CONTINUE
036     C CALCULATE DHT, THE DISTANCE FROM SOURCE TO HIT POINT
037     DHT=O.
038     DO 30 N=1,3
039     DHT=DHT+(XT(N)-XISC(N))*(XT(N)-XISC(N))
039     DHT=SQRT(DHT)+SQRT(WL)
040     IF(LHIT .AND. (DHT.GT.DHIT)) GO TO 40
041     LHIT=.TRUE.
042     DHT=DHT
CALL CTIBOT(IIS,IT,-2,MCC)

CONTINUE

IF(LTEST) THEN

WRITE(6,000)

FORMAT(,' TESTING CAPINT SUBROUTINE')

WRITE(6,*) IIS

WRITE(6,*) D

WRITE(6,*) DHIT,MD,LHIT,MD

ENDIF

RETURN

END
SUBROUTINE CYLINT

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

DOES RAY HIT CYLINDER?

DO 60 MCC=1,MCX
   IF(MH.LT.0 .AND. IABS(MH).LT.MCC) GO TO 60
   CALL CYLROT(XS,XSC,~,MCC)
   CALL CYLROI(D,DC,1,KC)
   STEP THRU CYLINDER SECTIONS
   NECX=NEC(MCC)-1
   NC=1,NECX
   A=AC(nc,MCX)
   B=BC(nc,MCX)
   AT A PARAMETER FOR ELLIPTIC CYLINDER
   CTC(I)=COS(RC(i),MCX))/SIN(RC(i),MCX))
   CTC(2)=COS(RC(i),MCX))/SIN(RC(i),MCX))
   AT C PARAMETERS FOR CONE FRUSTUMS SECTION
   ZZC=ZC(MCZ,MCX)-ZC(KC,MCX)
   IF(ABS(ZZC).LT.SML*WL) GO TO 40
   TNJ=(AC(1),MCZ)-AC(KC,MCX))/ZZC
   FL=TAN2(AL*XSC(2),BL*XSC(1))
   CV=100(AL+XSC(2),BL+XSC(1))
   GVE=COS(VH)
   SVE=SIN(VH)
   RHOE=SQR(AL+XSC(2),BL+XSC(1))
   IF(RHOE.GT.RHOE) GO TO 6
   CONTINUE
   IS SOURCE INSIDE OF INFINITE CYLINDER?
   IF(RHOS.LT.AL .AND. RHOS.GT.BL) GO TO 5
   IF(RHOS.LT.BL .AND. RHOS.GT.LT) GO TO 30
   Ve=TAN2(AL+XSC(2),BL+XSC(1))
   CVE=COS(VH)
   SVE=SIN(VH)
   RHOE=SQR(AL+XSC(2),BL+XSC(1))
   IF(RHOS.GT.RHOE) GO TO 6
   CONTINUE
   IS SOURCE INSIDE OF INFINITE CYLINDER SECTION?
   IF(XSC(3).LT.(ZC(MCZ,MCX)+XSC(I)*CTC(I))) GO TO 40
   IF(XSC(3).LT.(ZC(KC,MCX)+XSC(I)*CTC(2))) GO TO 40
   LHIT=.TRUE.
   GOTO 100
CONTINUE

CIII FIND COEFFICIENT OF EQUATION TO DETERMINE HIT POINT
AA=A*A
BB=B*B

CIII 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

CIII 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

CIII IS QUADRATIC SOLVABLE IN REAL SPACE?

CIII IF NOT, NO HIT POINT ON CYLINDER SURFACE SECTION
CT=CB*CB-CA*CC
IF(CT.LE.0.) GO TO 40

CIII DETERMINE TWO POSSIBLE HIT DISTANCES
SCT=SQRAT(CT)
RHP=(-CB+SCT)/CA
RHN=(-CB-SCT)/CA

CIII NEAREST POSITIVE ONE IS TRUE HIT POINT
IF(RHP.LT.0. .AND.RHN.LT.0.) THEN
GO TO 40
ELSE
IF(RHP.LT.0. .OR.RHN.LT.0.) THEN
RH=AMAX1(RHP,RHN)
ELSE
RH=AMIN1(RHP,RHN)
ENDIF

ENDIF

CIII IF(DHT.LE.0.) GO TO 40

CIII DETERMINE HIT DISTANCE FROM SOURCE TO HIT POINT
DHT=RH+SMR*WL

CIII IS HIT POINT ON FINITE CYLINDER SECTION?
IF(ZPM.GT.ZC(NC,MCC)+XPM*CTC(1) .OR.
ZPM.LT.ZC(NCP,MCC)*XPM*CTC(2)) GO TO 40

CIII DISTANCE FROM SOURCE TO HIT POINT
DHT=RH+SMR*WL

CIII CHECK FOR NEAREST HIT POINT FOR DIFFERENT SECTIONS
IF(LHIT .AND. (DHT.GT.DHIT)) GO TO 40
LHIT=.TRUE.

DHIT=DHT
CONTINUE

CIII CHECK TO SEE IF RAY HITS ENDCAPS
CALL CAPINT(IS,D,DHT,O,LHIT,-MCC)
IF(.NOT.LHIT) GO TO 50
IF(LHIT .AND. (DHT.GT.DHIT)) GO TO 50
LHIT=.TRUE.

DHIT=DHT
CONTINUE

IF(LHIT) THEN
WRITE(6,900)
FORMAT(//'TESTING CYLINT SUBROUTINE')
WRITE(6,'(I5)') IS
WRITE(6,'(E10.3)') D
WRITE(6,'(E10.3)') DHIT,LHIT,LEDF, MH
ENDIF
RETURN
END
SUBROUTINE CYLROT

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

0001  C-----------------------------------------------
0002  SUBROUTINE CYLROT(XREF,XCYL,N,MCL)
0003  C$$$$
0004  C$$$$ Rotates and or translates vectors in or out of the
0005  C$$$$ CYLINDER COORDINATE SYSTEMS
0006  C$$$$
0007  INCLUDE 'SHACOM.FOR'
0173  DIMENSION IREF(3),XCYL(3)
0174  IF(N.LT.0) GO TO 100
0175  XC=0.
0176  DO 50 I=1,3
0177  XCYL(I)=0.
0178  DO 10 J=1,3
0179  IF(N.EQ.2) XC=XCL(J,MCL)
0180  XCYL(I)=XCYL(I)+VCL(I,J,MCL)*(XREF(J)-XC)
0181  50  CONTINUE
0182  RETURN
0183  100 DO 200 I=1,3
0184  XREF(I)=0.
0185  DO 160 J=1,3
0186  160 XREF(I)=XREF(I)+VCL(J,I,MCL)*XCYL(J)
0187  IF(N.EQ.-2) XREF(I)=XREF(I)+XCL(I,MCL)
0188  200 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.

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.
0172 C!!!
0173 IF ( FILCNM .GT. 0 ) THEN
0174 DO 1 MC=1, MCX
0175 IF ( MC .NE. FILCNM ) CALL DOCTL( MC, FILCHR )
0176 1 CONTINUE
0177 CALL DOCTL( FILCNM, FILCHR )
0178 C!!! Fill with a different character for each cylinder.
0179 C!!! ELSEIF ( FILCNM .LT. 0 ) THEN
0180 DO 2 MC=1, MCX
0181 CALL DOCTL( MC, CHAR( MC+ICHAR( '0' ) ) )
0182 2 CONTINUE
0183 C!!! Fill with the main background fill character.
0184 C!!! ELSE
0185 DO 3 MC=1, MCX
0186 CALL DOCTL( MC, FILCHR )
0187 3 CONTINUE
0188 C!!! ENDIF
0190 RETURN
0192 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.

```fortran
0001  C-----------------------------------------------------------------------
0002  SUBROUTINE DOCYL( IC, FILL )
0003  INCLUDE 'SHACOM.FOR'
0169  C!!! This subroutine processes a single cylinder.
0171  C!!!
0172  CHARACTER FILL
0173  INTEGER
0174    +  J, K, IC
0176  C! Loop control variables.
0178    +  IBT,
0179  C! Truncate to integer.
0180    +  THETAI, PHII
0190  C!! REAL
0191    +  THETAR, PHIR,
0192  C! Theta & phi in radians.
0193    +  T,
0195  C! The parametric loop parameter.
0196    +  MAGTE,
0198  C! Length of a pseudo-side
0187    +  IPY,
0188  C! Scratch variable.
0189    +  DOT, LSTDOT,
0190  C! Dot product variables
0191    +  BTAN2, SQRT, ABS,
0192  C! Miscellaneous functions
0192  C! temporary vector
0194  C! Source to edge in ref coords
0195    +  XPQ( 3 ),
0196  C! Source to edge in pat coords
0197    +  XPC( 3 ),
0198  C! Point along cap in cyl coords
0199    +  RIM( 3 ),
0200  C! Use for dotmin points
0202  C! Antenna location in cylinder coords
0204    +  DOTMIN( 2, 10 )
0206  C! The two angles where dot is minimum
0207    +  LOGICAL
0208  C! Found one of the zero dots
0210  C!!
0211  C!!! Loop through endcaps, and incrementally on edges.
0212  C!!! Transform the antenna to cyl coords (include a translation).
0213  C!!!
0214  CALL CYLROT( ANTEHH, ANCYL, +2, IC )
0216  C!!!
0216  C!!! Do the endcaps one at a time.
0217  C!!!
0218  DO 200 J=1, NEC(IC)
0219  C!!
0220  C!! Loop around the endcap and remember where the dot products are zero
0221  C!!! between the vector looking at the point and the radial vector on
```
CII the endcap to the point. The cryptic parameters on the loop say:
"Loop from zero to 2*PI in one-degree steps."

DO 300 T=0.0, TPI+(TPI/360.0), (TPI/360.0)

CII Calculate the dot product and remember the two smallest ones.

RIM(1) = COS(J) * AC(1,IC)
RIM(2) = SIN(J) * BC(1,IC)
RIM(3) = ZC(1,IC)
DOT = RIM(1) * ( RIM(1) - ANCT(1) )
+ RIM(2) * ( RIM(2) - ANCT(2) )

CII If (the last dot product) * (this dot product) < 0, then that is

where our dot sign goes through zero.

Calculate theta & phi as we go around the rim.

Transform the rim point into ref. coord. system.

Find vector from source to rim.

CALL CYLROT( XYZ, RIM, -2, IC )

Convert the reference coordinate system to the pattern

coordinate system.

XPQ(1) = XYZ(1) - ABTEYN(1)
XPQ(2) = XYZ(2) - ABTEYN(2)
XPQ(3) = XYZ(3) - ABTEYN(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) )

CII Calculate angles representing border of rim and do branch test

on the phi angle.

THETAR = BTAN2( XPY, XPC(3) )
PHIR = BTAN2( XPC(2), XPC(1) )
IF ( PHIR .LT. PHI-0.5*RESPH ) PHIR = TPI + PHIR

Define pixel location.

THETA1 = INT( (THETAR - THET1) / RESTH + 0.6 ) + 1
PHI1 = INT( (PHIR - PHI1) / RESPH + 0.6 ) + 1

Put the character into the output buffer at the proper position.

Test if indices fall within specified window.
IF ( (THETAI .GE. 1) .AND. (THETAI .LE. COLS) ) THEN
  IF ( (PHII .GE. 1) .AND. (PHII .LE. ROWS) ) THEN
    OUTBUF( THETAI, PHII ) = CHAR(7)
  ENDIF
ENDIF

C   Reduplicate a wrapped-around character.

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

300 CONTINUE
200 CONTINUE

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

DO 400 K=1, 2
  DO 600 J=1, NEC(1C)-1
    RIM( 1 ) = COS( DOTMIN(K,J) ) * AC(J,IC)
    RIM( 2 ) = SIN( DOTMIN(K,J) ) * BC(J,IC)
    RIM( 3 ) = ZC(J,IC)
    CALL CYLROT( RIM1, RIM, -1, IC )
    RIM( 1 ) = COS( DOTMIN(K,J+1) ) * AC(J+1,IC)
    RIM( 2 ) = SIN( DOTMIN(K,J+1) ) * BC(J+1,IC)
    RIM( 3 ) = ZC(J+1,IC)
    CALL CYLROT( RIM2, RIM, -2, IC )
  400
  600
C   This MAGNE is analogous to the one in DOPLA except it works with pseudo-sides, so the name is somewhat misleading.
  MAGNE = SQRT(
    ( RIM(1) - RIM1(1) )**2 +
    ( RIM(2) - RIM1(2) )**2 +
    ( RIM(3) - RIM1(3) )**2 )
  T = 0.0
  50 IF ( T .GT. 1.0 ) GO TO 600
C   Find vector from source to rim.
C   These functions compute the theta/phi associated with a given point 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   Convert from the reference coordinate system to the pattern coordinate system
  XPQ(1) = ( RIM2(1)-RIM1(1) )*T + RIM1(1)-ANTEM1(1)
  XPQ(2) = ( RIM2(2)-RIM1(2) )*T + RIM1(2)-ANTEM1(2)
  XPQ(3) = ( RIM2(3)-RIM1(3) )*T + RIM1(3)-ANTEM1(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)**2 + XPC(2)**2 + XPC(3)**2 )
C   Define the angles representing the projection of the curved sides
and do a branch cut test on phi.

THETAR = BTAN2( XPY, IPC(3) )
PHIR = BTAN2( XPC(2), IPC(1) )

IF ( PHIR .LT. PHI-0.5*RESPH ) PHIR = TPI + PHIR

Define pixel location and put character in appropriate spot.

THETA1 = INT( (THETAR - ZHET1) / RESTH + 0.5 ) + 1
PHII = INT( (PHIR - PHI) / RESPH + 0.5 ) + 1

Check if angles fall within window.

IF ( (THETAI .GE. 1) .AND. (THETAI .LE. COLS) ) THEN
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(PHI2-PHI1-TP1) .LE. RESPH ) THEN
OUTBUF( THETAI, ROWS ) = CHAR(7)
ENDIF

How do an area fill on the object just outlined.

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

DO 700 PHII = 1, ROWS
CALL SCAN( IC, OUTBUF1 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.

```fortran
0001  C-----------------------------------------------------------------------
0002   SUBROUTINE DOPLAS
0003      INCLUDE 'SHACON.FOR'
0169   C!!!
0170   C!!! This subroutine processes each plate one at a time. The
0171   C!!! highlighting logic is contained here.
0172   C!!!
0173   C!!! Do the plates one at a time, then do the plate that was supposed to
0174   C!!! be highlighted last.
0175   C!!!
0176   IF ( FILPNM .GT. 0 ) THEN
0177       DO 1 MP = 1, NPI
0178           IF ( MP .NE. FILPNM ) CALL DOPLA( MP, FILCHR )
0179           1 CONTINUE
0180           CALL DOPLA( FILPNM, FILCHP )
0181   C!!!
0182   C!!! Fill with a different character for each plate.
0183   C!!!
0184   ELSEIF ( FILPNM .LT. 0 ) THEN
0185       DO 2 MP = 1, NPI
0186           CALL DOPLA( MP, CHAR( MP+ICHAR( '0' ) )
0187           2 CONTINUE
0188   C!!!
0189   C!!! Fill everything with the main background character.
0190   C!!!
0191   ELSE
0192       DO 3 MP = 1, NPI
0193           CALL DOPLA( MP, FILCHR )
0194           3 CONTINUE
0195   ENDIF
0196   END
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
0006       + IP, INT,
0007     CI Truncates to the nearest integer.
0008
0009     + THETAI, PHI
0010     CI Local indices into char array.
0011
0012     REAL
0013       + T,
0014     CI Parametric increment parameter.
0015
0016     + THETAR, PHIR,
0017     CITheta & phi in radians.
0018
0019     + MAGNE,
0020     CI Length of side NE.
0021
0022     + IPY,
0023     CI temporary variable
0024
0025     + IPQ(3),
0026     CI Source to edge in ref coords
0027
0028     + IPQ(3),
0029     CI Source to edge in pat coords
0030
0031     + BTAN2, SQR, ABS
0032     CI Miscellaneous functions.
0033
0034     CI!!
0035     CI!! Loop through incrementally along edges.
0036
0037     DO 200 ME=1, MEP( IP )
0038       NEXTME = MOD( ME, MEP(IP) ) + 1
0039     MAGNE = VAGC( ME, IP )
0040     T = 0.0
0041
0042     EQ  IF ( T .GT. 1.0 ) GOTO 100
0043
0044     CI!!
0045     CI!! These functions compute the theta/phi associated with a given
0046     CI!! point along an edge between two corners ME and NEXTME as a
0047     CI!! function of T. T varies from 0 to 1 and is adjusted to keep
0048     CI!! within a safe and efficient excursion at all times.
0049
0050     CI!!
0051     CI!! Convert from the reference coordinate system to the pattern
0052     CI!! coordinate system
0053
0054     CI!!
0055
0056     IPQ(1)=(XX(1,NEXTME,IP)-XX(1,ME,IP))*T+XX(1,ME,IP)-ANTENN(1)
0057     IPQ(2)=(XX(2,NEXTME,IP)-XX(2,ME,IP))*T+XX(2,ME,IP)-ANTENN(2)
0058     IPQ(3)=(XX(3,NEXTME,IP)-XX(3,ME,IP))*T+XX(3,ME,IP)-ANTENN(3)
0059
0060     CI!!
\( \text{XPy} = \sqrt{XPC(1)^2 + XPC(2)^2 + XPC(3)^2} \)

CIIII Define the angles representing the projection of the curved sides

\[ \text{THETAR} = \text{TAN2}( \text{XPy}, \ XPC(3) ) \]
\[ \text{PHIR} = \text{TAN2}( \text{XPC}(2), \ XPC(1) ) \]

IF ( PHIR .LT. PHI - 0.5 * RESPH ) PHIR = TPI + PHIR

CIIII Define pixel location and put the a character in the appropriate spot.

\[ \text{THETAI} = \text{INT}((\text{THETAR} - \text{THETI}) / \text{RETH} + 0.5) + 1 \]
\[ \text{PHII} = \text{INT}((\text{PHIR} - \text{PHI}) / \text{RESPH} + 0.5) + 1 \]

CIII Check if angles fall within window.

IF ( (THETAI .GE. 1) .AND. (THETAI .LE. COLS) ) THEN
  IF ( (PHII .GE. 1) .AND. (PHII .LE. ROWS) ) THEN
    \text{OUTBUF}( \text{THETAI}, \text{PHII} ) = \text{CHAR(7)}
  ENDIF
ENDIF

CIII Reduplicate a wrapped-around character.

IF ( (PHII .EQ. 1) .AND. \( |\text{PHII} - \text{PHI} - \text{TPI}| \) .LE. RESPH) THEN
  \text{OUTBUF}( \text{THETAI}, \text{ROWS} ) = \text{CHAR(7)}
ENDIF

CIII Put an upper bound on the increment for the case when the line segment is very short or the distance to the segment is great.

CIII In the degenerate case (on the z-axis) prevent a potential infinite CIII loop by putting a lower bound on \( \Delta t \) (ie by always adding at least a very small number to \( t \).)

\[ T = T + \text{MIN}(0.01, (\text{XPy} + \text{ALPH}/\text{MAGME} + 1.0) ) \]

GOTO 50

100 CONTINUE

CIII Now do an area fill on the object just outlined.

CIII Tell \text{SCAN} that this is a plate by using a "1".

DO 300 PHII = 1, ROWS
  CALL \text{SCAN}( \text{IP}, \text{OUTBUF(1,PHII)}, \text{PHII}, \text{FILL, 1} )
300 CONTINUE

RETURN

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 C
0004 CI This routine computes all the geometry associated
0005 CII with fixed plate structure, such as edge unit vectors.
0006 CII PLATE NORMALS, SHADOWED PLATES, ETC.
0007 CII
0008 INCLUDE 'SHACON.FOR'
0174 DIMENSION INIT(NEX), III(3), JIN(3), VI(3), IC(3), ISI(3), ISII(3)
0176 DIMENSION IOG(3), IDC(3), VTCP(2), BTC(4), VTC(2), BTCH(4), DS(3)
0178 DIMENSION VDO(3), VVR(3), VBR(3), VDR(3), PVRTNI(3)
0177 LOGICAL LSURF, LNPL, LTRM1
0178 LOGICAL LSHD, LST, LSTD, LSH, LHIT
0179 LOGICAL LGRED, LSHD, LDEBUG, LTEST, LWARN
0180 COMMON/TEST/LDEBUG, LTEST, LWARN
0181 COMMON/LIMIT/SHX, SHY, SHN, SHT, SHF
0182 COMMON/WAVE/V, WL
0183 COMMON/LSHID/LS, LSTD(NEX)
0184 COMMON/GROUND/LGRID, NPXR
0185 IF(LDEBUG) WRITE(6,667)
0186 667 FORMAT( 'DEBUGGING GEOM SUBROUTINE' )
0187 CII DETERMINATION OF V, VB, AND VP UNIT VECTORS FOR EDGE-FIXED
0188 CII COORDINATE SYSTEM
0189 CII STEP THRU PLATES
0190 DO 100 NP=1, NPXR
0191 100 M=NP+1
0192 CII STEP THRU EDGES
0193 DO 190 KE=1, KE
0194 KE=KE+1
0195 IF( (KE.GT.NE1).AND.ME.EQ.1 ) GO TO 200
0196 VN=0.
0197 CII CALCULATE EDGE UNIT VECTOR V AND EDGE LENGTH VNAG
0198 DO 10 N=1, 3
0199 10 VN=W-V(N,ME,MP)+V(N,ME,MP)*V(N,ME,MP)
0200 10 VN=W-V(N,ME,MP)+V(N,ME,MP)
0201 VNAG=SQRT(VN)
0202 DO 11 N=1, 3
0203 11 VNAG=VNAG/V(N,ME,MP)
0204 15 CONTINUE
0205 IF(.NOT.LDEBUG) GO TO 891
0206 DO 992 ME=1, ME
0207 WRITE(6,*) ( V(N,ME,MP), N=1, 3)
0208 992 CONTINUE
0209 991 CONTINUE
0210 CII CALCULATE PLATE UNIT NORMAL V
0211 VN(1,MP)=0.
0212 VN(2,MP)=0.
0213 VN(3,MP)=0.
0214 DO 22 ME=1, ME
0215 ME=ME+1
0216 IF( (ME.EQ.NE1).AND.ME.EQ.1 ) VN=1
0217 VN(1,MP)=VN(1,MP)+V(2,ME,MP)*V(3,ME,MP)-V(2,ME,MP)*V(1,ME,MP)
0218 VN(2,MP)=VN(2,MP)+V(3,ME,MP)*V(1,ME,MP)-V(3,ME,MP)*V(1,ME,MP)
0219 VN(3,MP)=VN(3,MP)+V(1,ME,MP)*V(2,ME,MP)-V(1,ME,MP)*V(2,ME,MP)
0220 22 CONTINUE
0221 VBN=0.
DO 20 N=1,3
20 VNM=VNM+VII(N,M)*VN(M,N)
21 DO 26 N=1,3
26 IF(LDBUG) WRITE(6,*) (V(N,M),N=1,3)
27 C!!! INSURE THAT ALL PLATES ARE FLAT. OTHERWISE ABORT!
28 C!!! TAKE DOT PRODUCT OF PLATE NORMAL AND EACH EDGE UNIT VECTOR
29 DO 30 ME=1,MEX
30 DOT=VII(1,M)*VN(1,ME)+VII(2,M)*VN(2,ME)+VII(3,M)*VN(3,ME)
31 ADOT=ABS(DOT)
32 IF(ADOT.LT.0.01) GO TO 120
33 NME=M+1
34 IF(NME.GT.MEX) NME=1
35 WRITE(6,121) M,ME,ADOT
36 121 FORMAT(' PLATE # ',I2,' IS NOT FLAT! CORNER # ',I2,' HAS ',
37 'PROBLEM. IF WARP > .073, ABORT PROGRAM')
38 3' IS GREATER THAN 0.03 STOP
39 IF(ADOT.GT.0.03) STOP
40 120 CONTINUE
41 C!!! CALCULATE UNIT BINORMAL VP WHICH IS IN PLATE PLANE
42 C!!! AND PERPENDICULAR TO PLATE EDGE
43 C!!! TAKE CROSS PRODUCT OF PLATE NORMAL AND EDGE VECTOR
44 DO 30 MEX=1,3
45 VP(1,ME,M)=VII(1,M)*VN(2,ME)-VII(2,M)*VN(1,ME)
46 VP(2,ME,M)=VII(2,M)*VN(3,ME)-VII(3,M)*VN(2,ME)
47 VP(3,ME,M)=VII(3,M)*VN(1,ME)-VII(1,M)*VN(3,ME)
48 IF(.NOT.LDBUG) GO TO 993
49 993 WRITE(6,*) (VP(N,ME,M),N=1,3)
50 DO 994 ME=1,MEX
51 994 CONTINUE
52 CONTINUE
53 CONTINUE
54 RETURN
55 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 INTEG
0175   + I, J, INT
0176 CIII
0177 CIII Clear the character buffer.
0178 CIII
0179 DO 10 J=1, MAIROW
0180 DO 10 I=1, MAICOL
0181 10 OUTBUF(I, J) = ' ' 
0182 CIII
0183 CIII The number of rows & columns needed for internal representation is
0184 CIII calculated from the user-selected (or defaulted) angular ranges of
0185 CIII interest combined with the desired resolution in rads/pixel
0186 CIII
0187 ROWS = INT((PH2 - PH1) / RESPH + 0.6) + 1
0188 COLS = INT((THET2 - THET1) / RESTH + 0.6) + 1
0189 CIII
0190 CIII Calculate some parameters needed by the dynamic T increment
0191 CIII algorithms. The maximum allowable angular excursion is the
0192 CIII smaller of the number of radians in a single pixel of either theta
0193 CIII or phi.
0194 CIII
0195 ALPH = MIN(RESTH, RESPH)
0196 CIII
0197 RETURN
0198 END
SUBROUTINE PLAINT

This routine determines if a given ray strikes a plate.

0001 C
0002 C-------------------------------------------------------------
0003 C SUBROUTINE PLAINT(XIS,D,DHIT,MH,LHIT)
0004 C111
0005 C111 DOES RAY HIT PLATE IF MH=0 ALL PLATES ARE CHECKED.
0006 C111 IF MH='#P THEN ONLY MP CHECKED AND SOURCE POSITION
0007 C111 MOVED TO HIT POSITION IF RAY HITS MP.
0008 C111 IF MH=MP THEN ALL PLATES OTHER THAN MP ARE CHECKED.
0009 C111
0010 INCLUDE 'SHACOM.FOR'
0011 DIMENSION XIS(3),D(3),XT(3),PVT(3)
0012 LOGICAL LHIT,LPLA,LCYL,LSTD,LTRN
0013 LOGICAL LSPAC,LDEBUG,LTEST,LVAR
0014 Cobil/LIMIT/SPIL,SMLR,SMLT,BIG
0015 CobilPLCILPLA.LGYL
0016 CobilMON/HITPLI/bIPH
0017 CobilPlO11/CRIOUL/LCRHID
0018 LHIT=.FALSE.
0019 DHIT=O.
0020 IF(.NOT.LPLA) RETURN
0021 C111 STEP THRU PLATES
0022 DO 60 MPP=1,MPAR
0023 MP=MP
0024 IF(MP.EQ.MH) GO TO 60
0025 IF(MH.LT.0) MPP=ABS(MH)
0026 C111 IF TOTAL SHADOWING ALGORITHM IS BEING USED, HAS PLATE MP
0027 C111 SHADOWED EVERY RAY TESTED?
0028 IF(LSTS.AND..11OT.LSTD(MP)) GO TO 60
0029 MEX=MMP(MP)
0030 AN=O.
0031 DO 6 N=1,3
0032 M=A*M+((XIS(N)-XX(N,1,MP))*VH(N,MP)
0033 C111 DO 60 MPP=1,MPAR
0034 MP=MP
0035 IF(MP.EQ.MH) GO TO 60
0036 IF(MH.LT.0) MPP=ABS(MH)
0037 C111 DOES RAY PASS THRU PLATE PLANE?
0038 IF(AN.EQ.DE.O.) GO TO 60
0039 DO 10 N=1,3
0040 C111 CALCULATE POINT WHERE RAY INTERSECTS PLATE PLANE
0041 DO 10 N=1,3
0042 M=AN+((XIS(N)-XX(N,1,MP))*VH(N,MP)
0043 C111 DO 60 MPP=1,MPAR
0044 MP=MP
0045 IF(MP.EQ.MH) GO TO 60
0046 IF(MH.LT.0) MPP=ABS(MH)
0047 C111 IS HIT POINT ON PLATE?
0048 DO 30 ME=1,MEX
0049 MME=M+1
0050 IF(MME.GT.MEX) MME=1
0051 RD=O.
0052 DO 20 N=1,3
0053 RD=RD+((XX(N,ME,MP)-XT(3)+((XX(N,ME,MP)-XT(3))))
0054 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3))
0055 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3)))
0056 2*(XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3))
0057 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3)))
0058 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3)))
0059 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3)))
0060 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3)))
0061 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3))
0062 DO 30 M=ME-1,MME
0063 MME=M+1
0064 IF(MME.GT.MEX) MME=1
0065 RD=O.
0066 DO 20 N=1,3
0067 RD=RD+((XX(N,ME,MP)-XT(3)+((XX(N,ME,MP)-XT(3)))
0068 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3))
0069 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3)))
0070 2*(XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3))
0071 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3)))
0072 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3)))
0073 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3)))
0074 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3)))
0075 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3))
0076 DO 30 M=ME-1,MME
0077 MME=M+1
0078 IF(MME.GT.MEX) MME=1
0079 RD=O.
0080 DO 20 N=1,3
0081 RD=RD+((XX(N,ME,MP)-XT(3)+((XX(N,ME,MP)-XT(3)))
0082 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3))
0083 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3)))
0084 2*(XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3))
0085 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3)))
0086 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3)))
0087 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3)))
0088 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3)))
0089 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3)))
0090 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3))
0091 DO 30 M=ME-1,MME
0092 MME=M+1
0093 IF(MME.GT.MEX) MME=1
0094 RD=O.
0095 DO 20 N=1,3
0096 RD=RD+((XX(N,ME,MP)-XT(3)+((XX(N,ME,MP)-XT(3)))
0097 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3))
0098 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3)))
0099 2*(XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3))
0100 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3)))
0101 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3)))
0102 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3)))
0103 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3)))
0104 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3))
0105 DO 30 M=ME-1,MME
0106 MME=M+1
0107 IF(MME.GT.MEX) MME=1
0108 RD=O.
0109 DO 20 N=1,3
0110 RD=RD+((XX(N,ME,MP)-XT(3)+((XX(N,ME,MP)-XT(3)))
0111 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3))
0112 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3)))
0113 2*(XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3))
0114 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3)))
0115 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3)))
0116 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3)))
0117 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3)))
0118 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3)))
0119 CP=CP+((XX(N,ME,MP)-XT(3))+(XX(N,ME,MP)-XT(3)))
0120 DBI=STAR2(CP,RD)
0121 DBI=DBI+DBI
0122 30 CONTINUE
IF(ABS(DBT).LT.PI) GO TO 60
  11 DHT=0.
  20 DO 40 N=1,3
  40 DHT=DHT+(XI(N)-XIS(N))*((XI(N)-XIS(N)))
  220 DHT=SQR(DHT)+SMLR
  220 IF(LHIT.AND.(DHT.GT.DHIT)) GO TO 60
  230 LHIT=.TRUE.
  231 DHT=DHT
  232 MPH=MP
  233 IF(MPH.GE.0) GO TO 60
  234 DO 45 N=1,3
  45 XIS(N)=XT(N)-SIGN(SMLR,AN)*VN(N,MP)
  236 GO TO 61
  239 50 CONTINUE
  240 IF(MPH.LT.0) GO TO 61
  241 C111 IF TOTAL SHADOWING ROUTINE IS BEING USED, INDICATE
  242 C111 THAT PLATE MP DOES NOT SHADOW SOURCE
  243 C117 IF(LSTS) LSTD(MP)=.FALSE.
  244 60 CONTINUE
  245 61 IF(.NOT.LTEST) GO TO 62
  246 WRITE(6,63)
  247 63 FORMAT(/' TESTING PLAIN SUBROUTINE'/)
  248 WRITE(6,63) XIS
  249 WRITE(6,63) D
  250 WRITE(6,63) DHIT,MP,LHIT
  251 62 RETURN
  252 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.

0001 C---------------------------------------------
0002     SUBROUTINE SCAN ( OBJ, LINE, PHI, FILL, TYPE )
0003 C!!!
0004 C!!! This subroutine "rasterizes" a line in the character buffer
0005 C!!! according to its shading requirements.
0006 C!!! The fill character is used to fill the
0007 C!!! line in. The line is declared larger character string in this
0008 C!!! subroutine than in the calling routine. This can be "hardwired"
0009 C!!! if it causes problems on other machines.
0010 C!!!
0011 INCLUDE 'SHCOM. FOR'
0012 CHARACTER*600 LINE
0013 CHARACTER*1 FILL
0014 INTEGER PTR, OBJ
0015 INTEGER LSTPTR
0016 INTEGER SCANC
0017 INTEGER SPARC
0018 INTEGER PHI
0019 INTEGER TYPE
0020 REAL DHIT
0021 REAL D(3)
0022 REAL DP(3)
0023 REAL XIS(3)
0024 REAL THETA, PHI
0025 LOGICAL EOL
0026 LOGICAL LHIT, LHI1
0027 COMMON /SCACHN/ PTR, EOL
0028 C!!!
0029 C!!! Initialize local variables.
0030 C!!!
0031 PTR = 1
0032 LSTPTR = 1
0033 EOL = PTR .GT. COLS
0034 C!!!
0035 C!!! Until the end of the line is scanned do ... 
0036 C!!!
0037 100 IF (.NOT. EOL) THEN
0038 C!!!
0039 C!!! Locate the first occurrence of CHR 7, 8, or EOL.
0040 C!!!
0041 PTR = SCANC( LINE )
0042 C!!!
0043 C!!! If point says it's a miss, update the last-pointer, scan, scan,
0044 C!!! fill. Otherwise, fill in the characters between the pointers.
0045 C!!! Define the "source point" as the location of the antenna,
0046 C!!! and see if our plate shadows the direction of the midpoint of the
0047 C!!! scan.
0048 C!!!
0049 THETA = (0.5*FLOAT(PTR-LSTPTR)-1.0)*RESTH+THET1
0050 PHI = (PHII-1)*BESP+PHI
0051 DP(1) = SIN(THETA)*COS(PHI)
0052 DP(2) = SIN(THETA)*SIN(PHI)
0053 DP(3) = COS(THETA)
0054 D(1) = DP(1)*VPC(1,1) + DP(2)*VPC(2,1) + DP(3)*VPC(3,1)

208
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)

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

XIS(1) = ANTENN(1)
XIS(2) = ANTENN(2)
XIS(3) = ANTENN(3)

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

The object is a plate.

GOTO (1,2) TYPE

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

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

CALL CAPI(XIS, D, DHIT, 0, LHIT, -OBJ)
IF (.NOT. LHIT) THEN CALL CYLINT(XIS, D, DHIT, LHIT, .FALSE., -OBJ)
LHIT = LHIT .OR. LHIT1
GOTO 999

Take the appropriate action in the buffer.

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

ENDUNTIL
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 C
0003 C!!! The following functions span/scan characters. That is, they
0004 C!!! return the position of next character in LINE which does or does
0005 C!!! not match the specified character. They also
0006 C!!! terminate the scan/span at the end of the line.
0007 C!!!
0008 C!!! INTEGER FUNCTION SCANC( LINE )
0009 INCLUDE 'SHACOM.FOR'
0176 CHARACTER(*) LINE
0176 INTEGER PTR
0177 LOGICAL EOL
0178 COMMON /SCNCMS/ PTR, EOL
0179 C!!!
0180 C!!! Until a character matching CHARAC is found, advance the pointer.
0181 C!!!
0182 SCANC = PTR
0183 200 IF (.NOT. (EOL .OR.
0184 + ( LINE(SCANC:SCANC) .EQ. CHAR(7) ) )) THEN
0185 SCANC = SCANC + 1
0186 EOL = SCANC .GT. COLS
0187 GOTO 200
0188 ENDIF
0189 C!!!
0190 C!!! End UNTIL
0191 C!!!
0192 RETURN
0193 END
0001 C!!!
0002 C-----------------------------------------------------------------------------------
0004 C!!! INTEGER FUNCTION SPANC( LINE )
0005 INCLUDE 'SHACOM.FOR'
0171 CHARACTER(*) LINE
0172 INTEGER PTR
0173 LOGICAL EOL
0174 COMMON /SCNCMS/ PTR, EOL
0176 C!!!
0176 C!!! Until a character NOT matching ASCII 7 is found, advance the
0177 C!!! pointer.
0178 C!!!
0179 SPANC = PTR
0180 200 IF (.NOT. (EOL .OR.
0181 + ( LINE(SPANC:SPANC) .NE. CHAR(7) ) )) THEN
0182 SPANC = SPANC + 1
0183 EOL = SPANC .GT. COLS
0184 GOTO 200
0185 ENDIF
0186 C!!!
0187 C!!! End UNTIL
0188 C!!!
0189 RETURN
0190 END
```
SUBROUTINE WRTOUT

This subroutine produces formatted and binary output of the shadow map.

0001 C-----------------------------------------------
0002 SUBROUTINE WRTOUT
0003 INCLUDE 'SHACOM.FOR'
0004 INTEGER I, J, COLI, COLF
0010 C!!!
0011 C!!! This subroutine writes the formatted output buffer to the output
0012 C!!! file. Start the output on a new page, and calculate a header
0013 C!!! based on the specified pixel resolution.
0014 C!!!
0015 C!!! Unit 7 is the main (ASCII) output file.
0016 C!!!
0017 C!!! Initialize the width of the map to be printed.
0018 C!!!
0019 COLI = 1
0020 COLF = 91
0021 IF(COLF .GT. COLS) COLF = COLS
0022 C!!!
0023 C!!! Print map.
0024 C!!!
0025 20 WRITE( 7, 100 ) ( ANTETN(I), I=1, 3, 1 ), INPFL
0026 20 WRITE( 7, 200 ) ( (RESTH*(I-1) + THET1)*DPR , I= COLI, COLF, 10)
0027 20 WRITE( 7, 200 ) ( ' ', I= COLI, COLF, 10)
0028 50 WRITE( 7, 300 ) ( RESPH*(J-1) + PHI )*DPR,
0029 + ( OUTBUF(I,J), I= COLI, COLF )
0030 C!!!
0031 C!!! If the map does not fit on the line printer width,
0032 C!!! then split it onto another set of pages.
0033 C!!!
0034 IF(COLF .LT. COLS) THEN
0035 COLI = COLF
0036 COLF = COLF + 90
0037 IF(COLF .GT. COLS) COLF = COLS
0038 GO TO 20
0040 ENDIF
0041 C!!!
0042 C!!! Have internal parameters available in degrees.
0043 C!!!
0044 THET1D = THET1 *DPR
0045 THET2D = THET2 *DPR
0046 RESTHD = RESTH *DPR
0047 PHID = PHI *DPR
0048 PH2D = PH2 *DPR
0049 RESPHD = RESPH *DPR
0050 C!!!
0051 C!!! Unit 10 is a generic sort of binary output which can be plotted
0052 C!!! anywhere. Place a little header info at the front of the file.
0053 C!!!
0054 WRITE( 10 ) COLS, THET1D, THET2D, RESTHD
0055 WRITE( 10 ) ROWS, PHID, PH2D, RESPHD
0056 C!!!
0057 C!!! Dump only that part of the buffer which pertains to this plot.
0058 C!!!
0059 DO 10 J = 1, ROWS
0060 DO 10 I = 1, COLS
0061 10 WRITE( 10 ) OUTBUF( I, J )
0062 C!!!

211
C Clll Output stuff is complete.
C
RETURN

C Clll Format statements.

100 FORMAT ('1', 6X, 'ANTENNA (RCS) = ', 2(F8.4, ','),
          + F8.4, ',') IN METERS', 6X, 'INPUT SET:', A42, / )
200 FORMAT ( THETA (DEGREES)', /, 9X, 11( 4X, F8.2) )
250 FORMAT ( PHI', 4X, A, 10( 9X, A ) )
300 FORMAT ( 6X, F7.3, 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.

C111 COMMON declarations...
C111 COMMON /PIS/
  *  PI, TPI, DPR, RPD
C111 MAXIMUM DIMENSION FOR PLATES
  INTEGER NPI
  PARAMETER (NPI=76)
C111 MAXIMUM DIMENSION FOR PLATE EDGES
  INTEGER NEX
  PARAMETER (NEX=12)
C111 MAXIMUM DIMENSION FOR CYLINDERS
  INTEGER NCI
  PARAMETER (NCI=6)
C111 MAXIMUM DIMENSION FOR CYLINDER RIMS
  INTEGER NHX
  PARAMETER (NHX=10)
C111 MAXIMUM DIMENSION FOR ROWS (PHI)
  INTEGER MAXROW
  PARAMETER (MAXROW=361)
C111 MAXIMUM DIMENSION FOR COLUMNS (THETA)
  INTEGER MAXCOL
  PARAMETER (MAXCOL=181)
C111 COMMON /GEOPLA/
  + XX (3,NEX,NPX),
  + V (3,NEX,NPX),
  + VP (3,NEX,NPX),
  + VN (3,NPX),
  + NEX (NPX),
C111 COMMON /GENEEL/
  + AC (NHX,NCI),
  + BC (NHX,NCI),
  + ZC (NHX,NCI),
  + TCR (NHX,NCI),
  + ICL (3,NCI),
  + VCL (3,3,NCI),
  + NEC (NCI),
  + NCI
C111 COMMON /EDMAG/ WMAG(NEX,NPX)
C111 COMMON /SHADW/ COLS, ROWS, ANTENN(3),CTROID(3),
  + MP,ME,NEXTM,NC,
  + THET1,THET2,PHI,PH2,RES1,RES2,ALPH,
  + UNIT(3),THP(3),VRC(3,3),UNITF,UNITW,UNITF,
  + THXP,THYP,THZP,PHXP,PHYP,PHZP,MPN,MC,
C111 COMMON /SHADWC/ IPPFIL,DOUTBUF(MAXCOL,MAROW),
  + FILCHC,FILCPR,FILCHR
C111 COMMON /PACTCUT/ VFC(3,3)
C111
The first set of declarations is the stuff in /SHADOW/ common block

```
INTEGER  
  MP, ME, NEXTME, MC,  
C1 Plate/edge#/cyl# variables.  
  + FILPHM, FILCHM,  
C1 Plate and cyl numbers for special filling  
  + CDLS,  
C1 The size of the array subsection determined  
  + ROWS  
C1 by internal resolution requirements.  
  
REAL  
  + CTROID.  
C1 A geometric center of the object in question.  
  + ANTELL,  
C1 The antenna location in Ref Coord. System.  
  + THE1,  
C1 The lower theta end of the range.  
  + THE2,  
C1 The higher theta end of the range.  
  + PH1,  
C1 The lower phi end of the range.  
  + PH2,  
C1 The higher phi end of the range.  
  + RESTH,  
C1 The desired theta/phi resolution  
  + REPH,  
C1 in units of radians/pixel.  
  + ALPH  
C1 Maximum allowed angular excursion.  
  
CHARACTER  
  + GUTBUF*1,  
C1 The output buffer which is displayed.  
  + INPFIL*63,  
C1 The filename of the input set.  
  + FILCH,  
C1 special fill character for cylinders  
  + FILCHP,  
C1 special fill character for everything else  
  + FILCHR,  
C1 special fill character for plates  
  DATA FILCH, FILCHP, FILCHR / 'C', 'P', 'X' /  
C111  
C111 From the /PIS/ COMMON block...  
C111  
REAL PI, TPI, DPR, RPD  
C111  
C111 From the /GESPLA/ COMMON block...  
C111  
  INTEGER  
  + MEP,  
C1 Number of edges per plate  
  + MPX  
C1 Total number of plates  
  REAL  
  + MPY,  
C1 The array of plate corners  
  + VX,  
C1 Edge unit vectors  
  + WY,  
C1 Edge unit binormals  
  + VY  
```
CI Unit normal for each plate
CIII
C!!! From the /GEOMEL/ COMMON block...
C!!!
   INTEGER
   +   NCI,
   C! Number of sections per cylinder
   +   MCX
   C! Total number of cylinders
   REAL
   +   MCX,
   C! Elliptic parameter along x-axis
   +   MCY
   C! Elliptic parameter along y-axis
   +   MCZ,
   C! Cylinder endcaps in cyl coord sys
   +   MCX,
   C! Angle endcap makes with positive x axis
   +   MCX,
   C! Cyl coord sys origin
   +   MCX,
   C! Definition of cyl coord sys
   C!
   INTEGER
   +   IUNIT
   REAL
   +   UNIT,
   +   UNITF,
   +   UNITS,
   +   UNITW,
   +   UNIT,
   +   TAZ,
   +   TPZ,PHZ,THP,PHP,
   +   VAS,
   +   VM4
   DATA UNIT/1.,.3048.0.0264/
C!
C!!! The following common block is for VMS/SMG$ software only.
C!!!
   INTEGER COMMON /TERCOM/
   +   KBDID, KEYBL
C!!!-
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
+++  
! File: SHACMD.CLD Edit: AAA1001  

MODULE COMMAND_TABLES
IDENT /SHACMD 01-001/  
/+  
! 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-1986  
!  
! MODIFIED BY:  
! 1-000 - Original. AAA 1-NOV-1986  
! 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 show_file_syntax routine show_file  
Define syntax show_out_syntax routine show_out  
Define syntax show_inp_syntax routine show_inp  
Define syntax show_uni_syntax routine show_uni  
Define syntax show_ent_syntax routine show_ent  
Define syntax show_coo_syntax routine show_coo  
Define syntax show_pat_syntax routine show_pat  
Define syntax show_sca_syntax routine show_sca  
Define syntax show_win_syntax routine show_win  
Define syntax show_key_syntax routine show_key  
!  
! Set syntax  
!  
Define syntax set_ent_syntax routine set_ent  
Define syntax set_coo_syntax routine set_coo  
Define syntax set_pat_syntax routine set_pat  
Define syntax set_sca_syntax routine set_sca  
Define syntax set_win_syntax routine set_win  
Define syntax set_key_syntax routine set_key  
Define syntax set_out_syntax routine set_out  
parameter p1 value( required )  
parameter p2 value( type=filename, required ),  
prompt="filename"  
qualifier plottable, default
```

216
qualifier printable, batch
qualifier echoing, default

Define syntax set_insp_syntax routine set_insp
parameter p1 value( required )
parameter p2 value( type=file, required ),
prompt=“input set”

Define syntax set_pla_syntax routine set_pla
parameter p1 value( required )
parameter p2 value( required, list),
parameter p3 value( default="F" ), prompt=“character”
qualifier all syntax=set_placyl_all

Define syntax set_cyl_syntax routine set_cyl
parameter p1 value( required )
parameter p2 value( required, list),
parameter p3 value( default="C" ), prompt=“character”
qualifier all syntax=set_cyl_all

Define syntax set_placyl_all
parameter p1
parameter p2 value( default="X" )

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

<table>
<thead>
<tr>
<th>Type definitions.</th>
</tr>
</thead>
</table>

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_fil_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_insp_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
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 definitons.
!
Define verb set
  parameter pl, value( required, type=set_types ),
  prompt = "Set what"

Define verb show
  parameter pl, 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 $ 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.

```plaintext
/*
| SHADOW.KPD -
| This file starts up the keypad definitions for the SHADOW program. This is a user-definable file and may be altered.
|
| Laszlo Takacs, 20-DEC-1985
|
| Set up the GOLD key.
| Def/key/noecho PF1 ** /if=default /set=gold
Def/key/noecho PF1 ** /if=gold   /set=default

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

| Set up the toggle keypad-mode key.
| Def/key/term/echo PF4 "Set keypad" /if=default
Def/key/term/echo PF4 "Set nokeypad" /if=gold

| Define miscellaneous keys.
| Def/key/echo/if=default KP7 "Set output 
Def/key/echo/if=default KP8 "Set input 
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 "
Def/key/echo/if=default KP2 "Set fill /plate=(i,X) 
Def/key/echo/if=default KP3 "Set fill /Sequential"/term
Def/key/echo/if=default KP0 "Spawn"

| Def/key/echo/if=gold KP7 "Show output" /terminate
Def/key/echo/if=gold KP8 "Show input" /terminate
Def/key/echo/if=gold KP9 "Show antenna" /terminate
Def/key/echo/if=gold MINUS "Show window" /terminate
Def/key/echo/if=gold KP4 "Show scale_factor"/term
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=(i,X) 
Def/key/echo/if=gold KP3 "Show fill" /terminate
Def/key/echo/if=gold KP0 "Spawn "

| Enter key is same as return. Period is EXIT.

219
```
Def/keY/term/echo PERIOD "Exit"
Def/key/term/echo ENTER **

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

220
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 useful 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, DOCYL, 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 system 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:

```fortran
PROGRAM PLOTSU
DIMENSION XDUM(2), YDUM(2), NC(6)
INTEGER COLS, ROWS
CHARACTER*80 LABELS(6), XLAB, YLAB, INF
CHARACTER(*) XFORU, YFORU
BYTE BTTE

C These are character parameters for the plotting output.
C
PARAMETER (XFORMU = 'F6.1')
PARAMETER (YFORMU = 'F6.1')

DATA XLAB // PHI //
DATA YLAB // THETA //
DATA NC // 6*72 //

C Read a header from FOROS. Open the file readonly so that other users
C can read it without needing write access to the file.
C
OPEN (UNIT=5, TYPE='OLD', READING )
READ ( 5, FMT='A') INF

C Read the header info from the data file. Open it unformatted.
C
OPEN(UNIT=10,FILE=INF,TYPE='OLD',FORM='UNFORMATTED',READING)
READ(10,END=9999) COLS,THETID,THETID2,RESTHD
READ(10,END=9999) ROWS,PHID,PHID2,RESPHD

ISCI = -2
ISCY = -2
XMIN = PHID
XMAX = PHID2
TMIN = -THETID
TMAX = -THETID2
BDI = 4
NI = 2
NDY = 4
BYY = 2

C Read the label info for this plot.
C
READ ( 6, *) LABELS(1)
READ ( 6, *) SI, ST, SZ, SPRX, SPRY, TRX
READ ( 6, *) ZTHEI, ZPHI, ZTHEI, ZPHI

C Format the labels for the plot (via internal write statements.)
C
WRITE (LABELS(2), 1100) SI, ST, SZ
WRITE (LABELS(3), 1100) ZTHEI, ZPHI, ZTHEI, ZPHI
WRITE (LABELS(4), 1300) SPRX, SPRY
WRITE (LABELS(5), 1400) TRX

CALL INFOPLT(2, XDUM, YDUM, XMIN, XMAX, YMIN, YMAX, ISCI,
       5, LABELS, NC, 0, -1, 1)

C Define a mapping window from data to plot
C
CALL SET ( 0.12, 0.54, 0.12, 0.54 )
```

227
! Do a linear-linear plot.

C A call to labmod might help the output look nicer.

CALL LABMOD

+ %REF( XFORMA ),
+ %REF( YFORMA ),
+ LEM( XFORMA ),
+ LEM( YFORMA ),
+ 1,
+ 1,
+ 0,
+ 0,
+ 0,

C Put labels on plot

XMID=0.6*(XMIN+XMAX)
YMID=0.6*(YMIN+YMAX)
XDEL=(XMAX-XMIN)/36.
YDEL=(YMAX-YMIN)/36.
XL=XMIN+0.5*XDEL
DO 100 IL=1,6
YL=YMAX+(6-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

CALL PWRT(XMID,YBOT,%REF(XLAB),6,1,0,0)
XSID=XMID-6.0*XDEL
CALL PWRT(XSID,YMID,%REF(XLAB),7,1,90,0)

C Use this call if you don't want grid lines.

C Define the perimeter of the plot.

CALL PERIML

XINC = 1.8
YINC = 0.9
ISYM = 1

C Loop on rows then on columns.

DO 10 J = 1, ROWS
I = RESPHD*(J-1)+PHID
100
DO 20 I = 1, COLS
READ (10, END=999) BYTE
IF (BYTE .NE. 32) THEN
  Y = -(RESTH(I)-((I-1)*THE1D))
  CALL PLSTM( I, Y, XINC, YINC, ISTM )
  CALL PWRIT( I, Y, BYTE, 1, XINC, YINC, ISTM )
  CALL PYRIT( Y, 0. )
END IF
20 CONTINUE
10 CONTINUE
C Formstr go down here.
1100 FORMAT ('ANTENNA LOCATED AT ',A,F7.1,A,F7.1)
1200 FORMAT ('ANTENNA ORIENTATION: ',A,F7.1,A,F7.1)
1300 FORMAT ('SOLAR PANELS ROTATED ',A,F7.1,A,F7.1)
1400 FORMAT ('THERMAL RADIATORS ROTATED ',A,F7.1)
END
Listing of sample LABEL.DAT file:

FORO1O.DAT:
   ' SHADOW TEST1 FOR CASE AN6S1'
  26. 15. 266.6 0.  -52. 0.
  0. 0. 90. 0.
   ' SHADOW TEST2 FOR CASE AN6S1'
  25. 15. 266.6 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.

Available: NASA's Industrial Applications Centers