NASA-CR-178,099

NASA-CR-178099
19880014547

NASA Contractor Report 178099

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

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

Contents

List of Tables

List of Figures

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 useage 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 availablity 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 devise dependent software for the VAX. The non-interactive and interactive parts of the code have been kept separate, however, so that the code can be run non-interactively without much change.

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

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

Chapter 2

Method

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

In order to have quick turn around for antennas mounted on large structures, it is desirable to use a method that will directly produce the shadow projected onto the far zone sphere. This can be accomplished in a two step process. First the outside boundary of each individual piece making up the structure can be transformed from the x, y, z coordinate system into a sequence of lines in the θ and ϕ 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, $\bar{R}_s$, to some position along its outer boundary, $\bar{R}_i$, such that

$$\bar{R} = \bar{R}_i - \bar{R}_s.$$

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

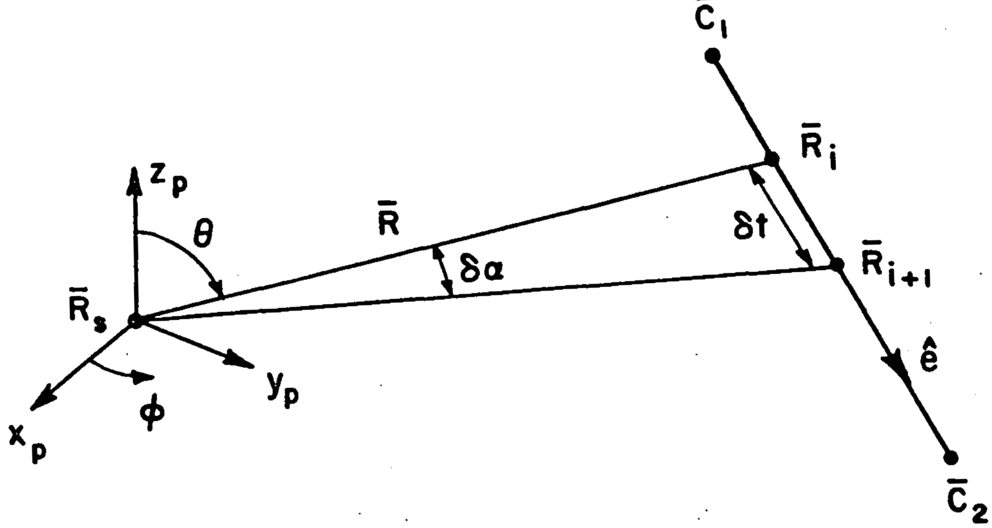


Figure 2.1: Geometry showing the projection of the plate edge onto the far zone sphere.

then be transformed onto the two dimensional far zone sphere by

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

and

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

The position of the vector along the edge is defined by starting at a corner and then incrementing the edge in steps of δt 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 δt 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 θ and ϕ. The distance R is defined as $R = |\bar{R}_s - \bar{R}_i|$. 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

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

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

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

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

The shadow test for a plate is made by first projecting the vector chosen at the 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

$$\bar{R}_t = \bar{R}_s - \frac{[\hat{n}\cdot(\bar{R}_s - \bar{C}_1)]\hat{r}}{\hat{n}\cdot\hat{r}}.$$

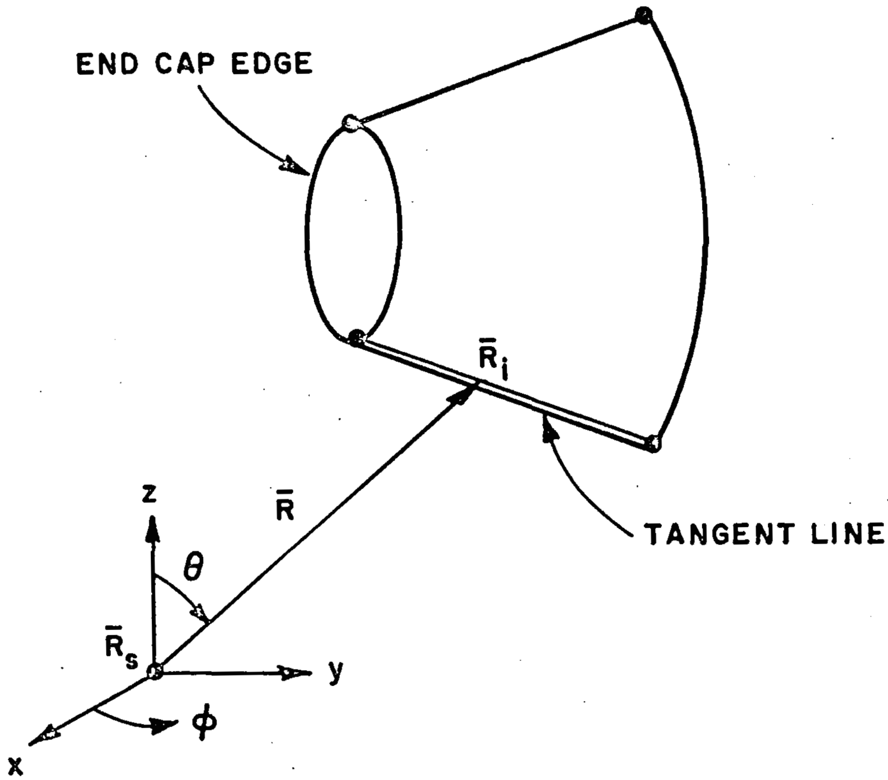


Figure 2.2: Geometry showing the projection of a cone frustum onto the far zone sphere.

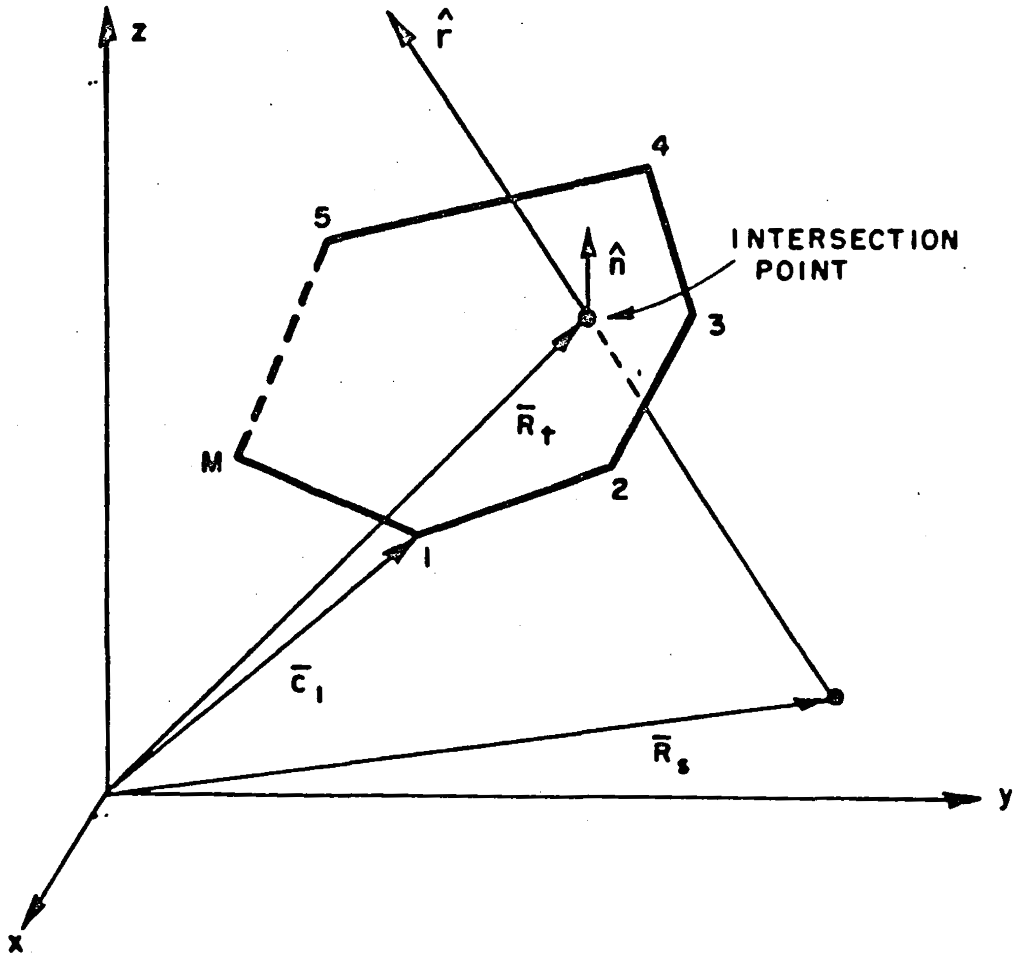


Figure 2.3: Intersection of observation direction vector with plate.

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

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

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

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

or

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

The geometry is illustrated in Figure 2.5. The point defined by $\bar{R}_c$ 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, θ and ϕ, 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

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

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

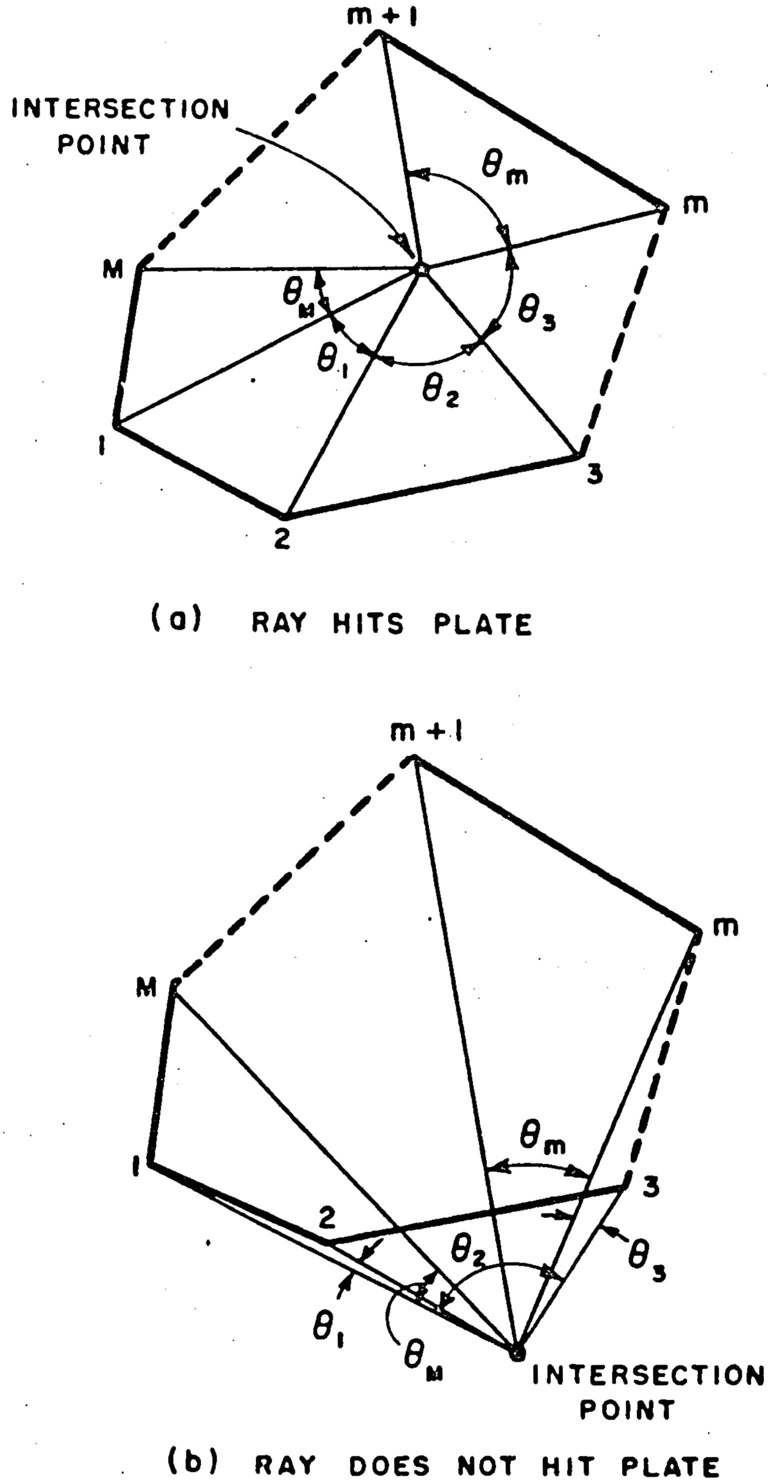


Figure 2.4: The geometry for deciding whether a ray does or does not hit the plate.

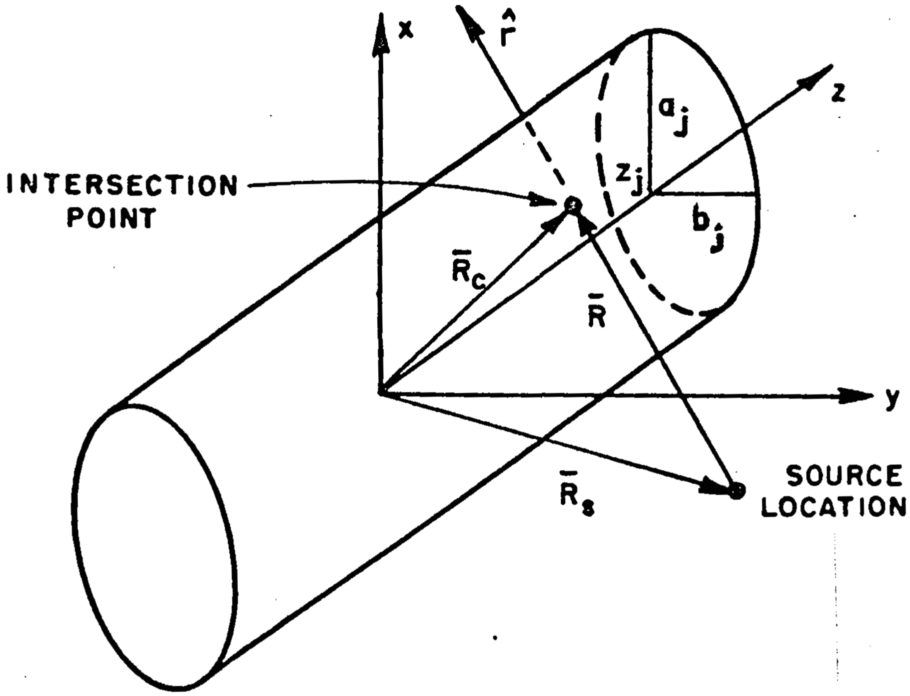


Figure 2.5: Geometry illustrating the hit point on a cone frustum segment.

and

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

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

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

Chapter 3

Principle of Operation

3.1 Overview

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

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

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

3.2 Modeling the Structures

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

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

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

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

3.3 Running the Code

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

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

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

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

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

3.4 Non-Interactive Commands

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

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

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

The geometry information is by default assumed to be in meters in a definition coordinate system that is initially the reference coordinate system. The units can be changed using the UN command to either inches or feet or back to meters again. Once the UN command is specified all information after that command is assumed to be in those units unless changed by another call to UN. There is also provision for using any conversion fac-

COMMAND	DEFINITION	LOCATION
BP	back or bistatic scatter	[1]
CC	cone frustum geometry	pg 24
CE	last or only comment	pg 30 or [1]
CG	cylinder geometry	pg 27 or [1]
CM	comment card	pg 30 or [1]
EN	end execution	pg 30 or [1]
FM	swept frequencies	[1]
FR	frequency	[1]
GP	infinite ground plane	pg 31 or [1]
GR	range gate	not documented
LP	line printer output	[1]
NC	next set of cylinders	pg 32 or [1]
NG	no ground plane	pg 32 or [1]
NP	next set of plates	pg 32 or [1]
NR	next set of receivers	[1]
NS	next set of sources	pg 32 or [1]
NX	next problem	pg 32 or [1]
PD	far zone pattern cut	[1]
PF	far zone cut (non integer)	not documented
PG	plate geometry	pg 33 or [1]
PN	near zone pattern cut	[1]
PP	plotter output	[1]
PR	gain or coupling factors	[1]
RA	receiver array geometry	[1]
RD	far zone range	[1]
RG	receiver geometry	[1]
RM	NEC-MOM receiver input	[1]
RT	rotate-translate geometry	pg 36 or [1]
SA	source array geometry	[1]
SG	source geometry	pg 38 or [1]
SM	NEC-MOM source input	[1]
TO	test options	[1]
UF	model scale factor	pg 42 or [1]
UN	units of geometry	pg 42 or [1]
US	units of source size	[1]
VD	volumetric cut (integer)	not documented
VF	volumetric far zone cut	pg 44
VN	volumetric near zone cut	not documented
VP	volumetric plotter output	not documented
XQ	execute code	pg 47 or [1]

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 descriptions of the commands. The EXIT command exits the user back to the operating system. The SPAWN command allows the use of DCL command while the user is still in the shadow code.

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

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

3.6 Keypad Use

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

The keypad definitions are initialized by a text file containing suitable "DEFINE/KEY" commands. The file is called SHADOW.KPD and must reside in the default directory of the user running the code. There is a template file provided with the code which may be customized by the user. The predefined definitions of the VT100 keypad are shown in Table 3.3. Note that the "gold" enables the lower case action in the top row, that is, in most case the "SHOW" operation instead of the "SET" operation.

COMMAND	LOCATION
EXIT	Page 49
HELP	Page 50
SPAWN	Page 54
SET ANTENNA_LOCATION	Page 56
SET COORDINATES	Page 58
SET FILL_CHARACTER	Page 59
SET INPUT_SET	Page 62
SET KEYPAD_MODE	Page 63
SET OUTPUT	Page 64
SET PATTERN_CUT	Page 66
SET SCALE_FACTOR	Page 67
SET UNITS	Page 68
SET WINDOW	Page 69
SHADOW	Page 52
SHOW ANTENNA_LOCATION	Page 71
SHOW COORDINATES	Page 72
SHOW FILL_CHARACTER	Page 73
SHOW INPUT_SET	Page 74
SHOW KEYPAD_MODE	Page 75
SHOW OUTPUT	Page 76
SHOW PATTERN_CUT	Page 77
SHOW SCALE_FACTOR	Page 78
SHOW UNITS	Page 79
SHOW WINDOW	Page 80

Table 3.2: Table of interactive commands.

<table>
<tr><td>PF1
gold</td><td>PF2
HELP</td><td>PF3
SHADOW</td><td>PF3
nokeypad
SET
KEYPAD</td></tr>
<tr><td>7
show
SET
OUTPUT</td><td>8
show
SET
INPUT</td><td>9
show
SET
ANTENNA</td><td>–
show
SET
WINDOW</td></tr>
<tr><td>4
show
SET
SCALE</td><td>5
show
SET
UNITS</td><td>6
show
SET
COORD</td><td>,
show
SET
PATTERN</td></tr>
<tr><td>1
show
SET
FILL</td><td>2
/cylin
FILL
/PLATE</td><td>3
show
FILL
/SEQUEN</td><td rowspan="2">Enter
RETURN</td></tr>
<tr><td colspan="2">0
SPAWN</td><td>.
EXIT</td></tr>
</table>

Table 3.3: VT100 keypad for SHADOW interactive commands.

Chapter 4

Non-Interactive Commands

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

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

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

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

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

4.1 Command CC: Cone Frustum Geometry

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

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

where

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

READ: TCLZ, PCLZ, TCLX, PCLX

where

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

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

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

READ: 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.

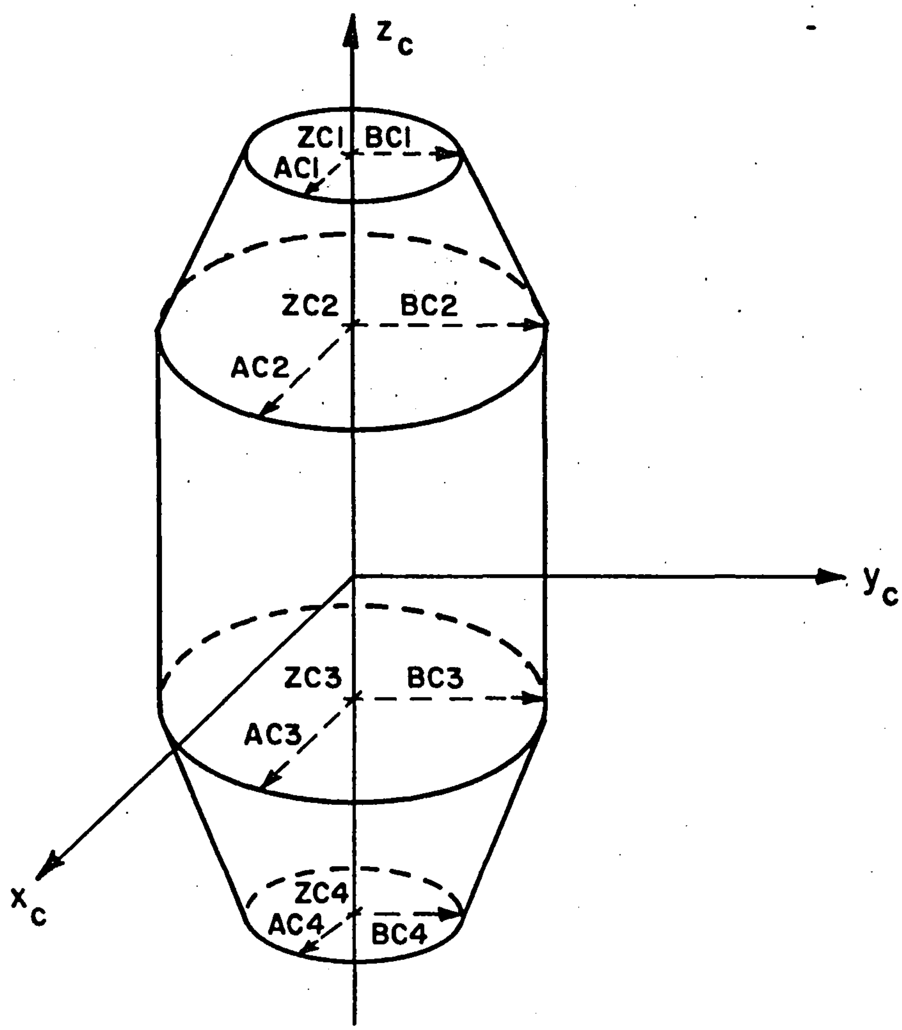


Figure 4.1: Definition of finite cylinder geometry composed of cone frustum segments with elliptic cross section.

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 z_c-axis of the cylinder coordinate system as if it was a radial vector in the definition coordinate system.

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

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

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

where

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

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

READ: ZCN, THTN, ZCP, THTP

——— where ———

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

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

ZCP This is a real variable that defines the position of the center of the most positive end cap on the zc-axis of the cylinder.

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

Note that the program will keep increasing the number of cylinders in the solution by the number of calls to this command unless the NC or NX commands are called to reinitialize the cylinder geometry.

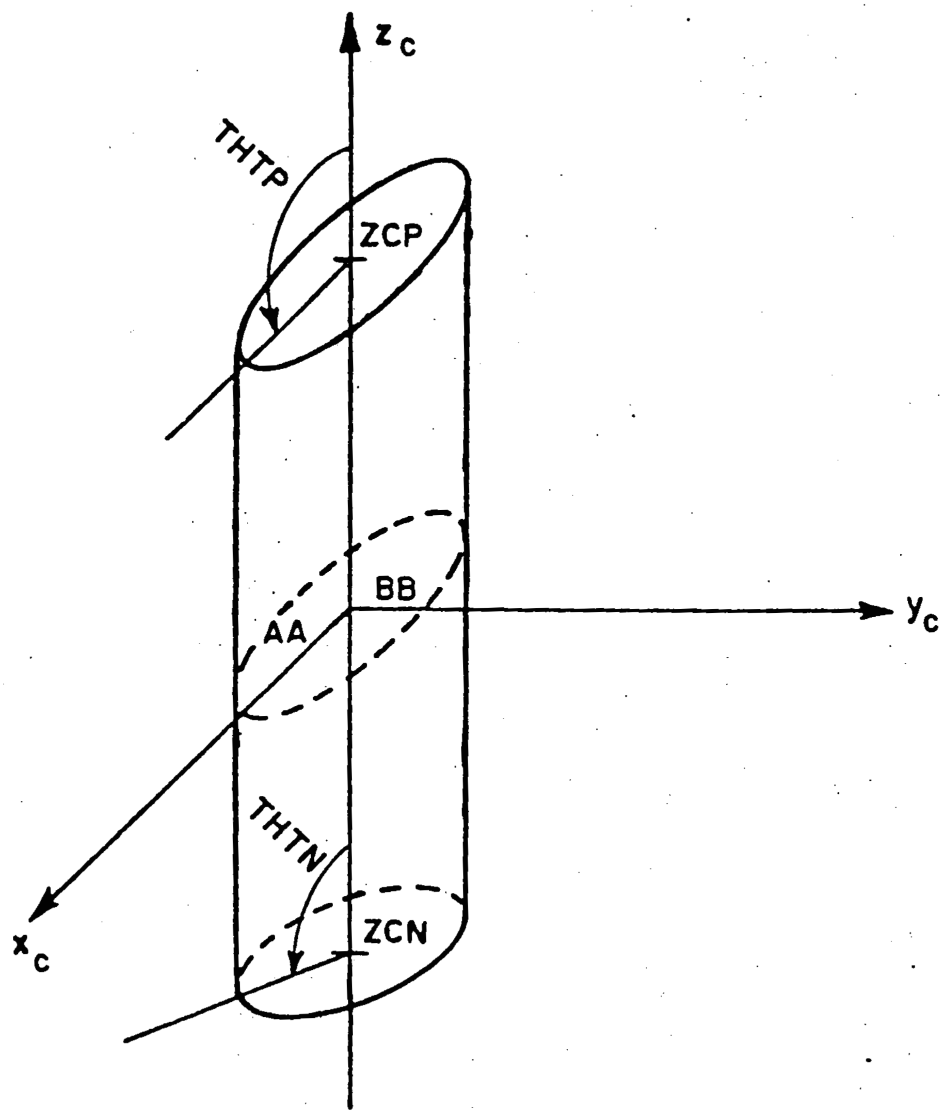


Figure 4.2: Definition of finite elliptic cylinder geometry.

4.3 Command CM: and CE: Comments

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

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

——— where ———

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

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

4.4 Command EN: End Program

This command enables the user to terminate the execution of the scattering code.

4.5 Command GP: Ground Plane

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

READ: LSLAB(MPDX)

——— where ———

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

0 = Perfectly conducting metalic 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)

——— where ———

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 metalic 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)

——— where ———

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

XX(1,2,1)=2.
XX(2,2,1)=4.
XX(3,2,1)=6.

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

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

4,0	:corners and type of plate
1., 1., 0.	:corner #1
-1., 1., 0.	:corner #2
-1.,-1., 0.	:corner #3
1.,-1., 0.	:corner #4.

See elsewhere for further details on how to number the corners. Note that the program will keep increasing the number of plates in the solution by the number of calls to this command unless the NP or NX commands are called to reinitialize the plate geometry.

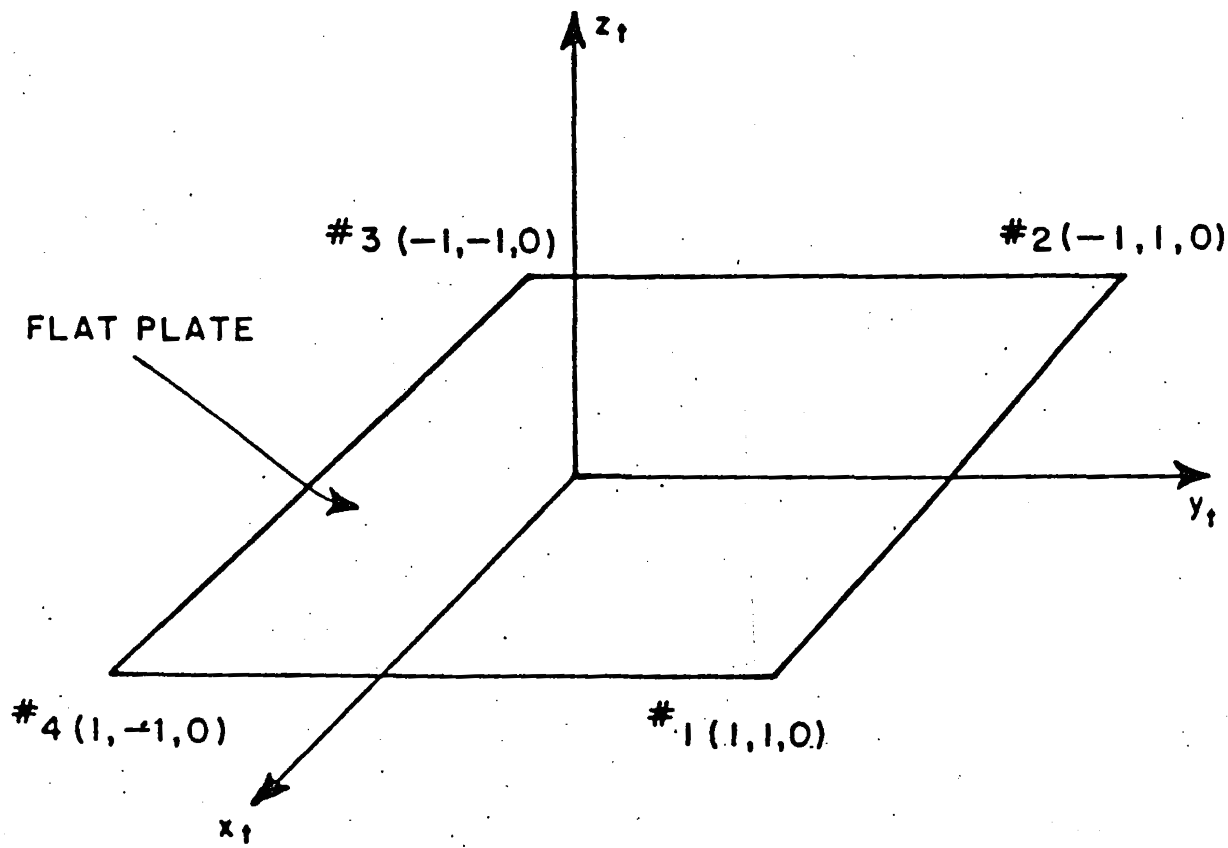


Figure 4.3: Definition of flat plate geometry.

4.12 Command RT: Rotate-Translate Geometry

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

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

where

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

READ: THZP, PHZP, THXP, PHXP

where

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

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

The new x-axis and z-axis must be defined orthogonal to each other. The new y-axis is found from the cross product of the x- and z-axis. All the subsequent inputs will be made relative to this new coordinate system, which is shown as x_t, y_t, z_t, unless command RT is called again and redefined.

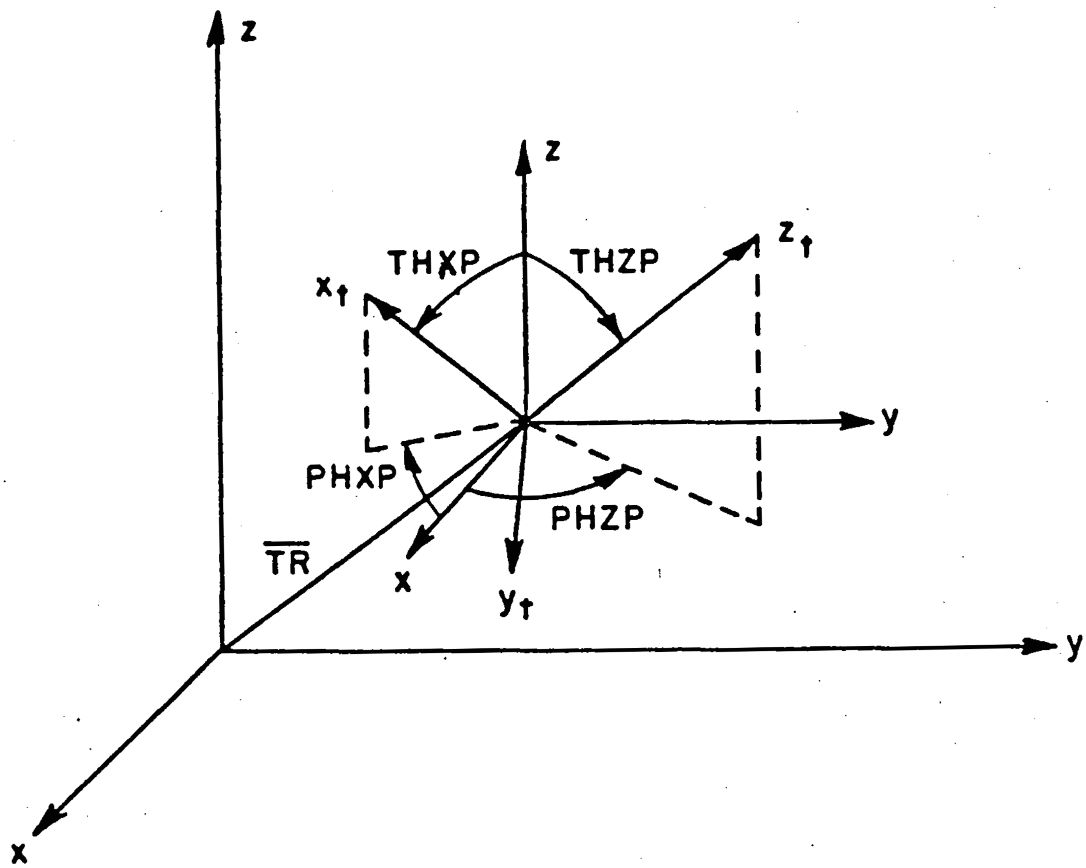


Figure 4.4: Definition of rotate-translate coordinate system geometry.

4.13 Command SG: Source Geometry

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

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

where

XSS(N,MS) This is a doubly dimensioned real array which is used to define the x,y,z location of the MSth 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 MSth element in the definition coordinate system. They are input in degrees as spherical angles that define a radial direction which is parallel to the MSth 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 MSth element in the definition coordinate system. They are input in degrees as spherical angles that define a radial direction which is parallel to the MSth 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 MSth 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.

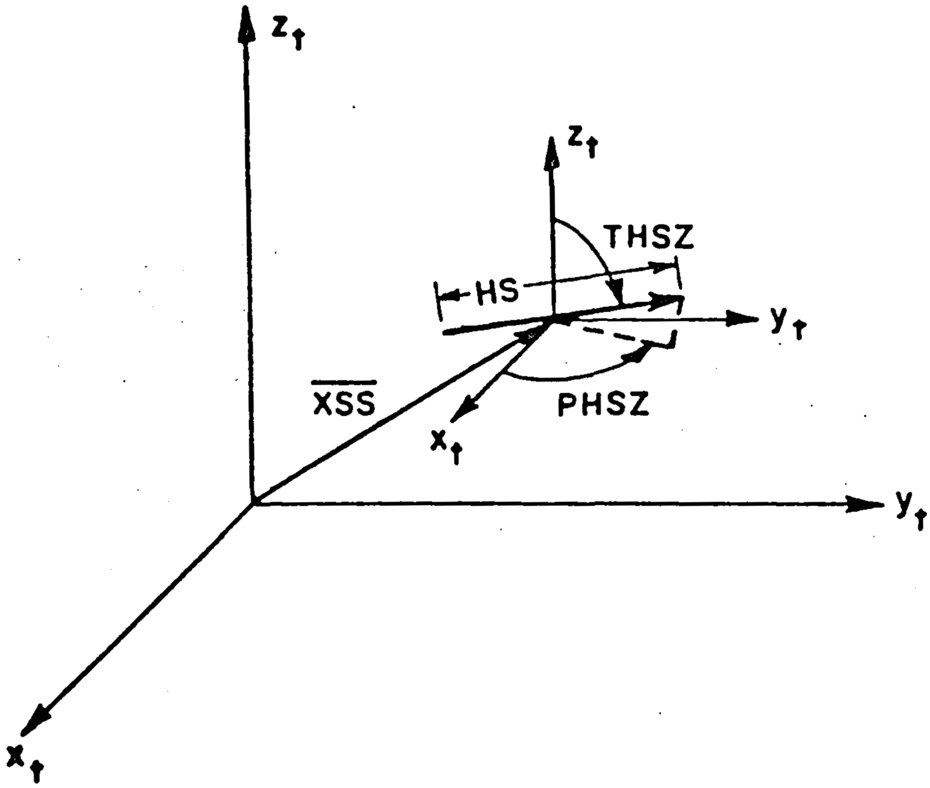


Figure 4.5: Definition of source geometry for dipole antennas.

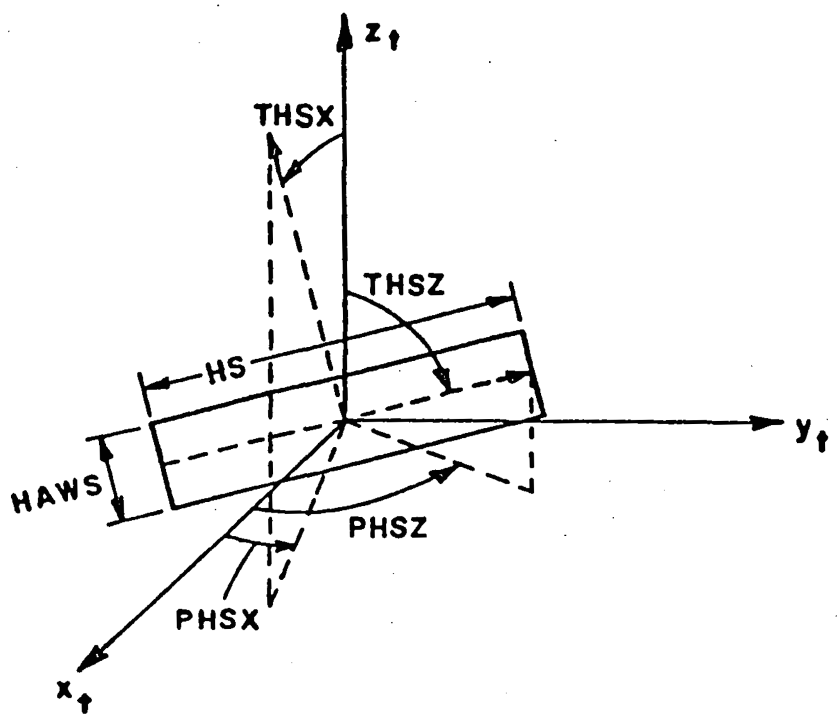


Figure 4.6: Definition of source geometry for aperture antennas.

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

where

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

where

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 a integer variable. It defines the number of pattern points of the "fixed" angle specified by LCNPAT.

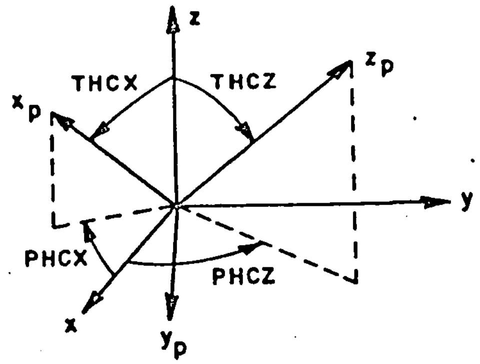
READ: TPPS, TPPI, NPN

where

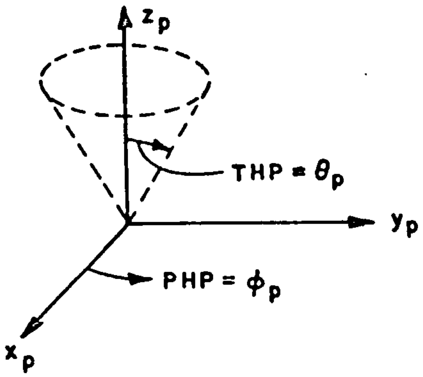
TPPS This is a real variable. It defines the starting angle of the "varying" angle specified by LCNPAT.

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

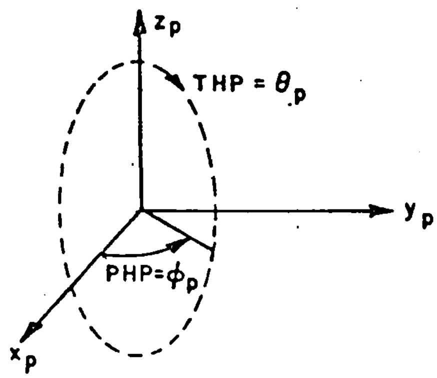
NPN This is a integer variable. It defines the anumber of pattern points of the "varying" angle specified by LCNPAT.



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

EXIT

Command Qualifiers	Defaults
None.	None.

restrictions

None.

prompts

None.

command parameters

None.

DESCRIPTION

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

COMMAND QUALIFIERS

None.

EXAMPLES

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

This example shows how to exit the program.

HELP

Displays information about SHADOW commands or help text from any other library you specify.

FORMAT

`HELP help-item`

Command Qualifiers	Defaults
`/LIBRARY[=library-name]`	`/LIB=SYS$DISK:[]SHADOW`

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)
```

```
Topic? <RETURN>
SHADOW>
```

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

SHADOW

Command Qualifiers	Defaults
None.	None.

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.

- oThe RESOURCE_WAIT state must be enabled for the spawning process.
- oRequires TMPMBX or PRMMBX user privileges.
- oSPAWN does not manage terminal characteristics.

Command may be abreviated "$ ", 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
```

```
 CWP6148      CWP6148          00000891  VT1023:        TTA0:
 DF6148       DF6148           00000AFA  VT1086:        TTC4:
 EHN000       EHN000           000009F8  VT1085:        TTB7:
 LT6199       LT6199           00000AEE  VT1081:        TTA7:
 WE6148       WE6148           00000973  VT1082:        TTB2:
SHADOW> SPAWN
$ SHO TIME
  11-DEC-1985 08:36:13
$ LOG
  LT6199_1      job terminated at 11-DEC-1985 08:36:20.90
SHADOW>
```

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

```
SET ANTENNA
```

Command Qualifiers	Defaults
None.	None.

restrictions

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

prompts

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

command parameters

None.

DESCRIPTION

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

COMMAND QUALIFIERS

None.

EXAMPLES

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

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

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	SET COORDINATES	
	Command Qualifiers	Defaults
	None.	None.

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.

COMMAND QUALIFIERS

None.

EXAMPLES

```
SHADOW> SET COOR
Please input a translation vector in feet   : 100,200,300
Please input THZP,PHZP,THXP,PHXP in degrees: 0, -54, 265.5, 45
*     The following rotations are used for ALL subsequent inputs:        *
*    VRS(1,1)= -0.70711  VRS(1,2)= -0.70711  VRS(1,3)=  0.00000          *
*    VRS(2,1)=  0.70711  VRS(2,2)= -0.70711  VRS(2,3)=  0.00000          *
*    VRS(3,1)=  0.00000  VRS(3,2)=  0.00000  VRS(3,3)=  1.00000          *
```

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

SET FILL_CHARACTER

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

FORMAT

`SET FILL [tag-character]`

Command Qualifiers	Defaults
/SEQUENTIAL	None.
/PLATE=(num[,char])	None.
/CYLINDER=(num[,char])	None.

restrictions

None.

prompts

None.

command parameters

`tag-character`

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

DESCRIPTION

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

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

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

COMMAND QUALIFIERS

/SEQUENTIAL
/SEQUENTIAL

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

FORMAT

```
SET INPUT_SET filename
```

Command Qualifiers	Defaults
None.	None.

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.

FORMAT

SET [NO]KEYPAD

Command Qualifiers	Defaults
None.	None.

restrictions

None.

prompts

None.

command parameters

None.

DESCRIPTION

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

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

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

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
```

Command Qualifiers	Defaults
/PLOTTABLE	/PLOTTABLE
/PRINTABLE	/NOPRINTABLE
/ECHOING	/ECHOING

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`
`/NOECHOING`

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: _NLA0:[]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: _NLA0:[]FOR010.DAT;
Printer file is: USER1:[RJM.NAS]AN5S1.PRT;2
Input echo file: _NLA0:[]FOR006.DAT;
SHADOW> SET OUT AN5S1
Plotting file is: USER1:[RJM.NAS]AN5S1.PLT;3
Printer file is: _NLA0:[]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.

FORMAT

SET PATTERN_CUT

Command Qualifiers	Defaults
None.	None.

restrictions

The specified coordinate axes must be orthogonal.

prompts

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

command parameters

None.

DESCRIPTION

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

COMMAND QUALIFIERS

None.

EXAMPLES

```
SHADOW> SET PAT
Please input THZP,PHZP,THXP,PHXP in degrees: 0, -54, 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

SET SCALE_FACTOR

Command Qualifiers	Defaults
None.	None.

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.

FORMAT

SET UNITS keyword

Command Qualifiers	Defaults
None.	None.

restrictions

None.

prompts

_inches, feet, or meters:

command parameters

Keyword may be one of the following:

- o METERS
- o FEET
- o 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

`SET WINDOW`

Command Qualifiers	Defaults
None.	None.

restrictions

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

prompts

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

command parameters

None.

DESCRIPTION

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

COMMAND QUALIFIERS

None.

EXAMPLES

```
SHADOW> SET WINDOW
The current range of theta in degrees is   0.0000000E+00 to    180.0000
with a resolution of    2.000000     degrees/pixel.
The current range of phi in degrees is   0.0000000E+00 to    360.0000
with a resolution of    2.000000     degrees/pixel.
Please enter a new range for theta (lower,higher): 30,40
Please enter a new THETA resolution in degrees/pixel: .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   0.5000000     degrees/pixel.
The current range of phi in degrees is    45.00000     to    90.00000
with a resolution of   0.5000000     degrees/pixel.
```

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

SHOW ANTENNA_LOCATION

Display the current antenna position.

FORMAT

SHOW ANTENNA_LOCATION

Command Qualifiers	Defaults
None.	None.

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

Command Qualifiers	Defaults
None.	None.

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.

FORMAT

`SHOW FILL`

Command Qualifiers	Defaults
None.	None.

restrictions

None.

prompts

None.

command parameters

None.

DESCRIPTION

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

COMMAND QUALIFIERS

None.

EXAMPLES

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

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

SHOW INPUT_SET

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

FORMAT

SHOW INPUT_SET

Command Qualifiers	Defaults
None.	None.

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.

FORMAT

`SHOW KEYPAD_MODE`

Command Qualifiers	Defaults
None.	None.

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

SHOW OUTPUT

Command Qualifiers	Defaults
None.	None.

restrictions

None.

prompts

None.

command parameters

None.

DESCRIPTION

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

COMMAND QUALIFIERS

None.

EXAMPLES

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

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

SHOW PATTERN_CUT

Displays the pattern-cut coordinate system transformation matrix.

FORMAT

SHOW PATTERN_CUT

Command Qualifiers	Defaults
None.	None.

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 coordinat system. For more information, see the SET PATTERN command on page 66.

COMMAND QUALIFIERS

None.

EXAMPLES

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

The pattern-cut coordinate system shown has been set up.

SHOW SCALE_FACTOR

Displays the uniform scale factor currently in effect.

FORMAT

SHOW SCALE_FACTOR

Command Qualifiers	Defaults
None.	None.

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

SHOW UNITS

Command Qualifiers	Defaults
None.	None.

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.

FORMAT

`SHOW WINDOW`

Command Qualifiers	Defaults
None.	None.

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. Interactivly this can be changed using the SET FILL command. Noninteractivly, 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 uniquie fill character. The first plate starts with "A" and each succeeding plate is incremented up by one ASCII character. The first cylinder starts with "1" and each succeeding cylinder is incremented up by one ASCII character. Note that if there are a lot of plates and/or cylinders, the fill characters will eventually get into some of the more seldom used ASCII characters. Also note that in this mode of filling, the code superimposes the latest calculated shadow for a plate or cylinder on top of the shadow map. This means that the character in a pixel for a finished map will represent the last object that the code calculated a shadow for and not the object that is located closest to the observer.

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

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

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

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

Chapter 7

Examples

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

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

7.1 Example 1: A Plate

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

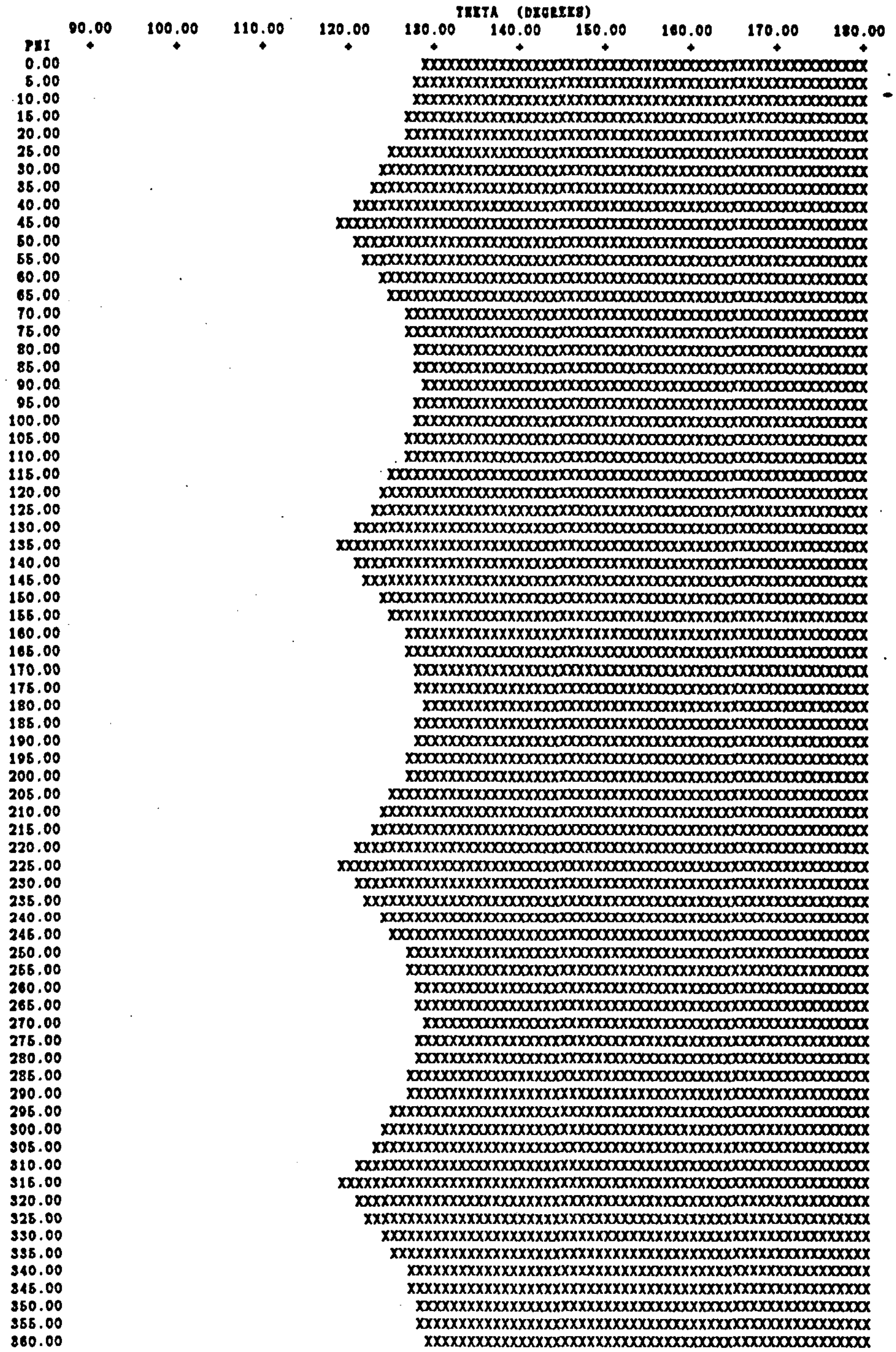
```
$ RUN SHADOW
SHADOW> SET OUT  PLAEX1/NOPLOT/PRINT
SHADOW> SET INP  PLAEX
SHADOW> SET UNI  METERS
SHADOW> SET WIND
        90, 180
        1.0
        0., 360
        5.
SHADOW> SET ANT
SHADOW> 0,0,8
SHADOW> SHADOW
SHADOW> EXIT
$
```

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:

ANTENNA (RCS) = (0.0000, 0.0000, 8.0000) IN METERS INPUT SET: USER1:[RJM.NAS.MAN]PLAEX.INP;5



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:

```
ANTENNA (RCS) = (  0.0000,  0.0000,  0.0000  ) IN METERS     INPUT SET: USER1:[RJM.NAS.MAN]PLAEX2.INP;2

                                                   THETA  (DEGREES)
        0.00     20.00     40.00     60.00     80.00     100.00    120.00    140.00    160.00    180.00
PSI      +         +         +         +         +         +         +         +         +         +
  0.00
  5.00
 10.00
 15.00
 20.00
 25.00
 30.00
 35.00
 40.00                            XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 45.00                           XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 50.00                          XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 55.00                         XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 60.00                        XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 65.00                       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 70.00                       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 75.00                       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 80.00                      XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 85.00                      XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 90.00                      XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 95.00                      XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
100.00                      XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
105.00                       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
110.00                       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
115.00                        XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
120.00                        XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
125.00                         XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
130.00                          XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
135.00                           XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
140.00                            XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
145.00
150.00
155.00
160.00
165.00
170.00
175.00
180.00
```

7.3 Example 3: The First Plate Revisited

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

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

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

```
CM:   SIMPLE PLATE TEST SET
CE:   RCS INPUT SET
UN:
1
PG: THE PLATE IS 400 SQUARE-METERS.
4 0
-10.0, +10.0, 0.0
-10.0, -10.0, 0.0
+10.0, -10.0, 0.0
+10.0, +10.0, 0.0
XQ:
EN:
```

The output generated by the code was the following:

```
ANTENNA (RCS) = (  0.0000,  0.0000,  8.0000  ) IN METERS     INPUT SET: USER1:[RJM.NAS.MAN]PLAEX.INP;5

                                                    THETA  (DEGREES)
       0.00      20.00     40.00     60.00     80.00    100.00    120.00    140.00    160.00    180.00
PHI      +         +         +         +         +         +         +         +         +         +
  0.00
  5.00
 10.00
 15.00
 20.00
 25.00
 30.00
 35.00
 40.00                            XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 45.00                           XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 50.00                          XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 55.00                         XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 60.00                        XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 65.00                       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 70.00                       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 75.00                       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 80.00                      XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 85.00                      XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 90.00                      XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 95.00                      XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
100.00                      XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
105.00                       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
110.00                       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
115.00                        XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
120.00                        XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
125.00                         XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
130.00                          XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
135.00                           XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
140.00                            XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
145.00
150.00
155.00
160.00
165.00
170.00
175.00
180.00
```

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.
T,0.,2.,91
0.,2.,181
XQ:
EN:
```

7.5 Example 5: An Elliptic Cylinder

This example is 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  CYLEX1/NOPLOT/PRINT
SHADOW> SET INP  CYLEX1
SHADOW> SET UNI  METERS
SHADOW> SET WIND
130,180
0.55555556
0.,360
5.
!
! An overhead view of the cylinder, which is centered on the origin,
! with radii of 1 and 1, with a 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
```

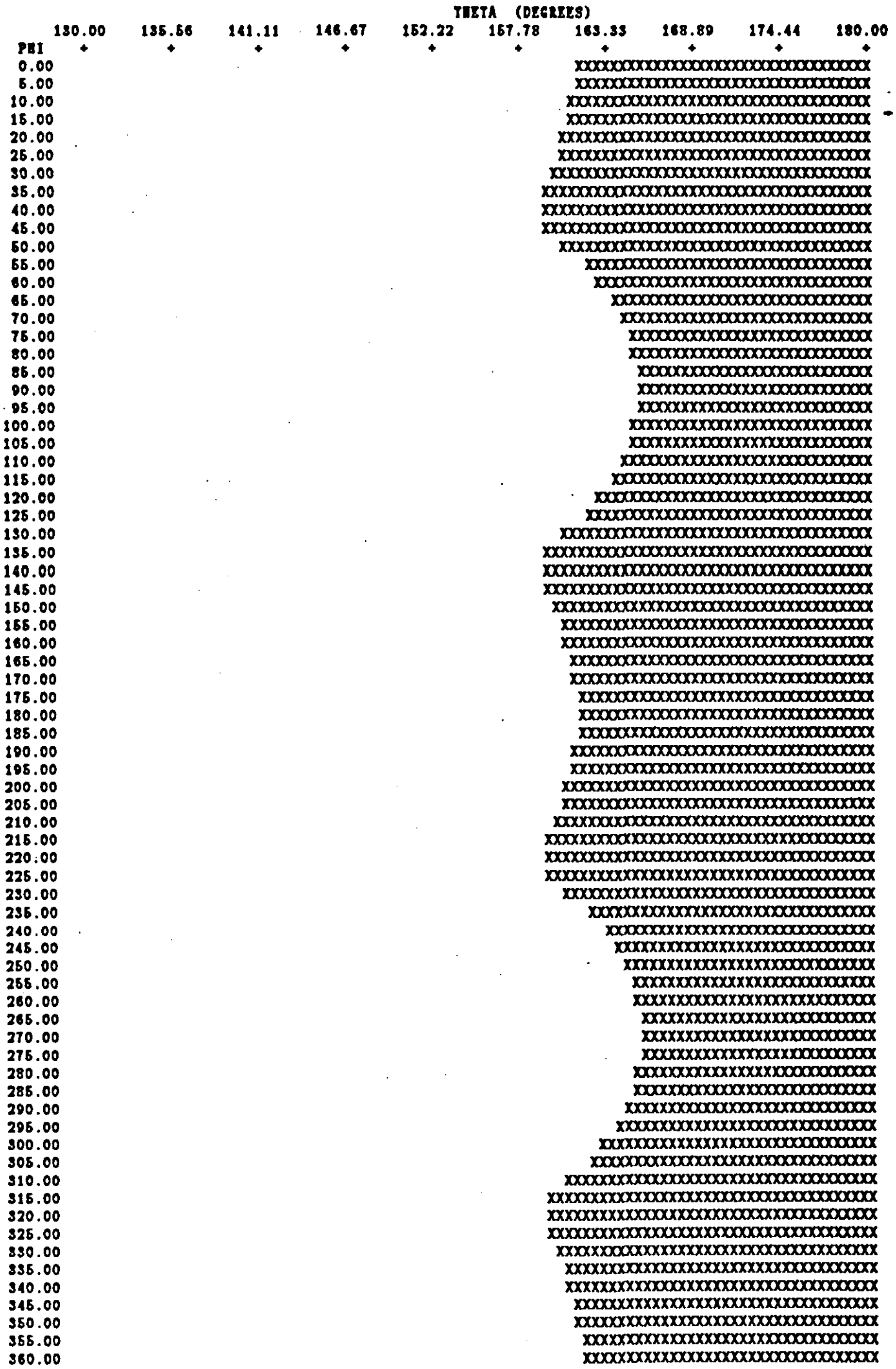
The input set defining the plate was the following:

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

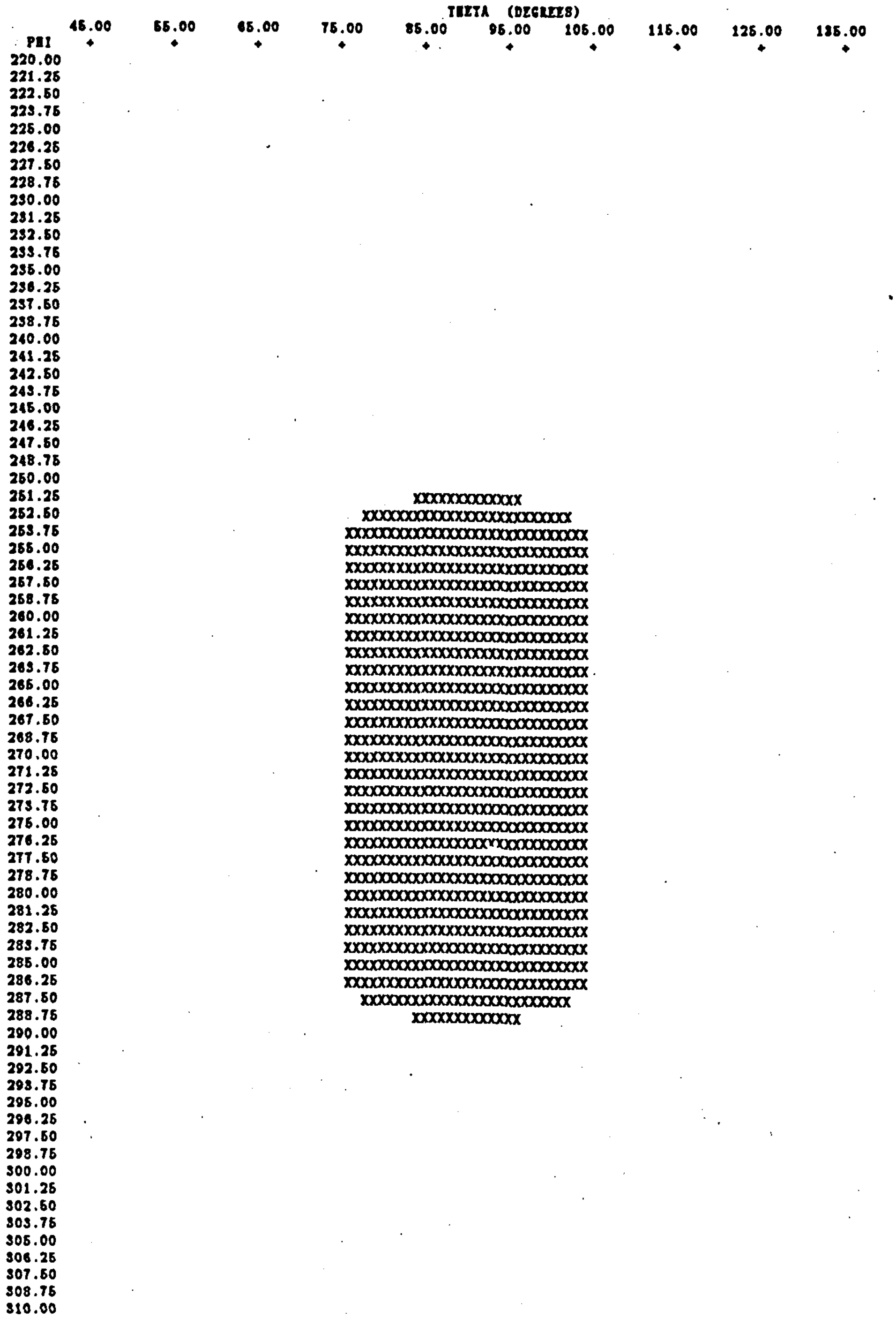
```
1
CC: FIRST CYLINDER
0.,0.,0.
90.,0.,0.,0.
2
1.,1.,  1.
1.,1., -1.
XQ:
EN:
```

The output generated by the code was the following:

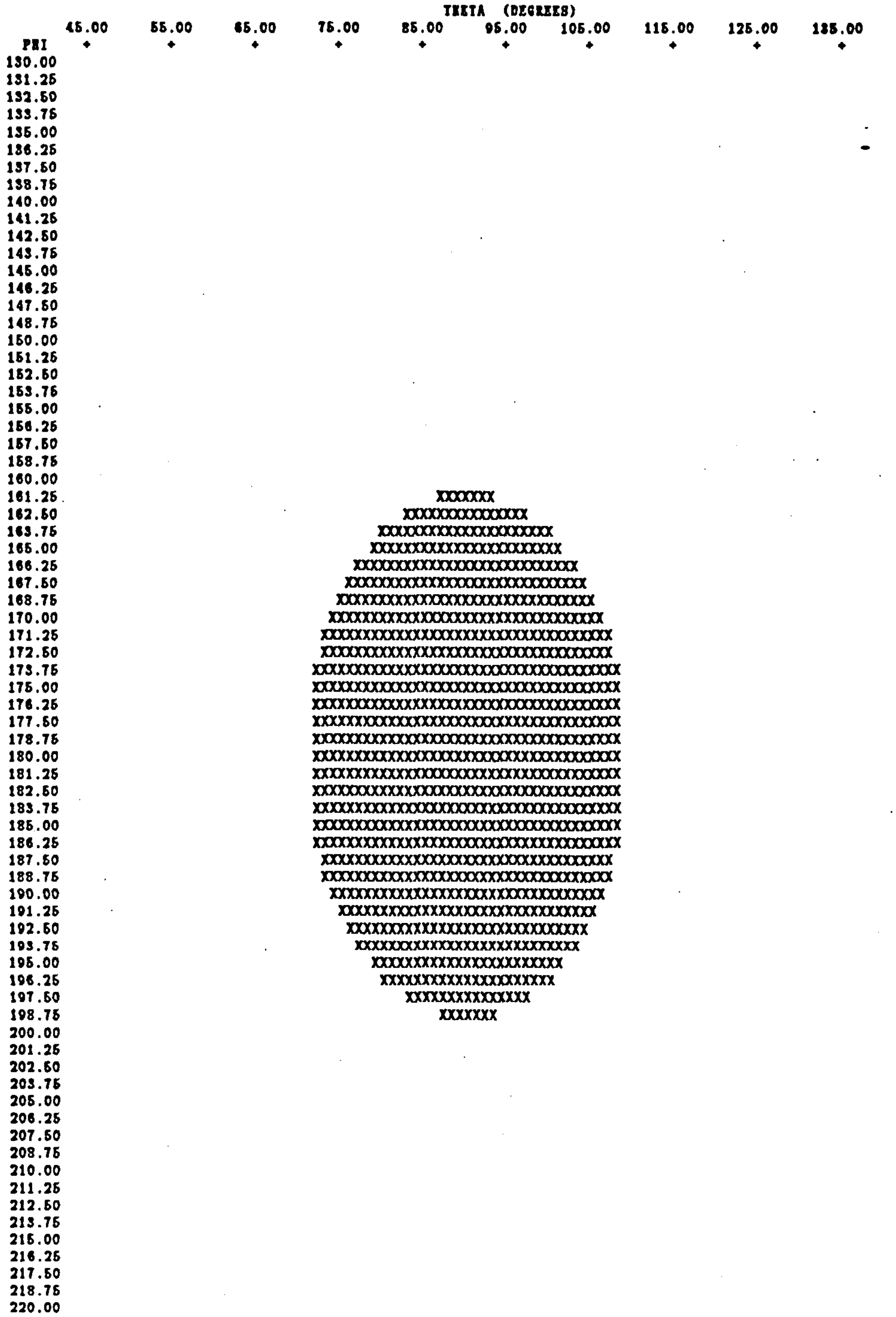
```
ANTENNA (RCS) = (  0.0000,  0.0000,  4.0000  ) IN METERS     INPUT SET: USER1:[RJM.NAS.MAN]CYLEX1.INP;4
```



ANTENNA (RCS) = (0.0000, 4.0000, 0.0000) IN METERS INPUT SET: USER1:[RJM.NAS.MAN]CYLEX1.INP;4



ANTENNA (RCS) = (4.0000, 0.0000, 0.0000) IN METERS   INPUT SET: USER1:[RJM.NAS.MAN]CYLEX1.INP;4



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/NOPLOT/PRINT
SHADOW> SET INP  CYLEX2
SHADOW> SET UNI  METERS
SHADOW> SET WIND
130,180
0.55555556
0.,360
5.
!
! An overhead view of the 2 cylinders with radii of 1 and 1,
! with a length of 1 meter each.
!
SHADOW> SET FILL  /CYL=1
SHADOW> SET ANT
0,0,4
SHADOW> SHADOW
!
SHADOW> SET WIND
45,135
1.0
220.,310
1.25
SHADOW> SET ANT
0,4,0
SHADOW> SHOW FILL
SHADOW> SHADOW
!
SHADOW> SET ANT
4,0,0
SHADOW> SET WIND
45,135
1.0
130.,220
1.25
SHADOW> SHOW FILL
SHADOW> SHADOW
SHADOW> EXIT
$ EXIT
```

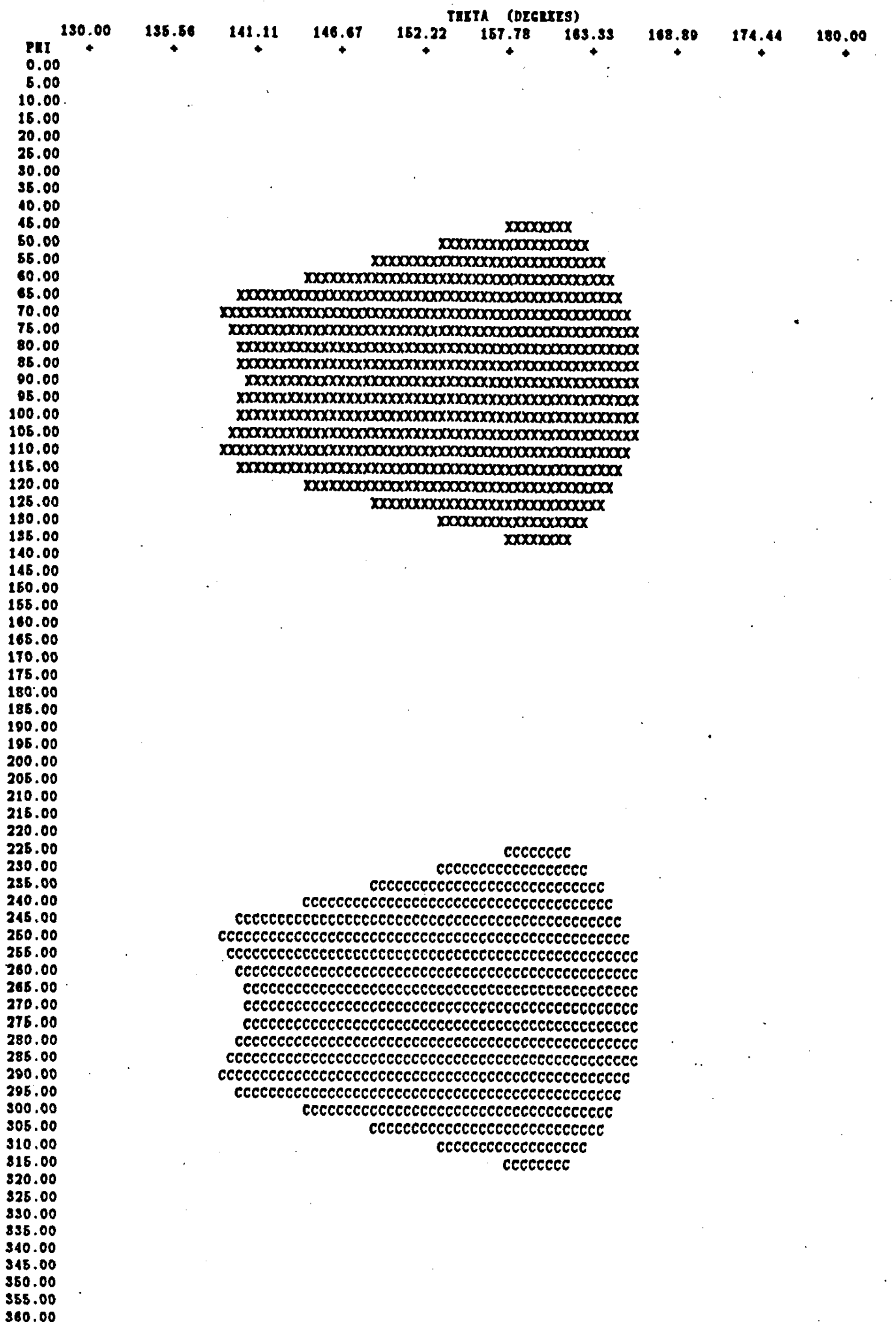
The input set defining the plate was the following:

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

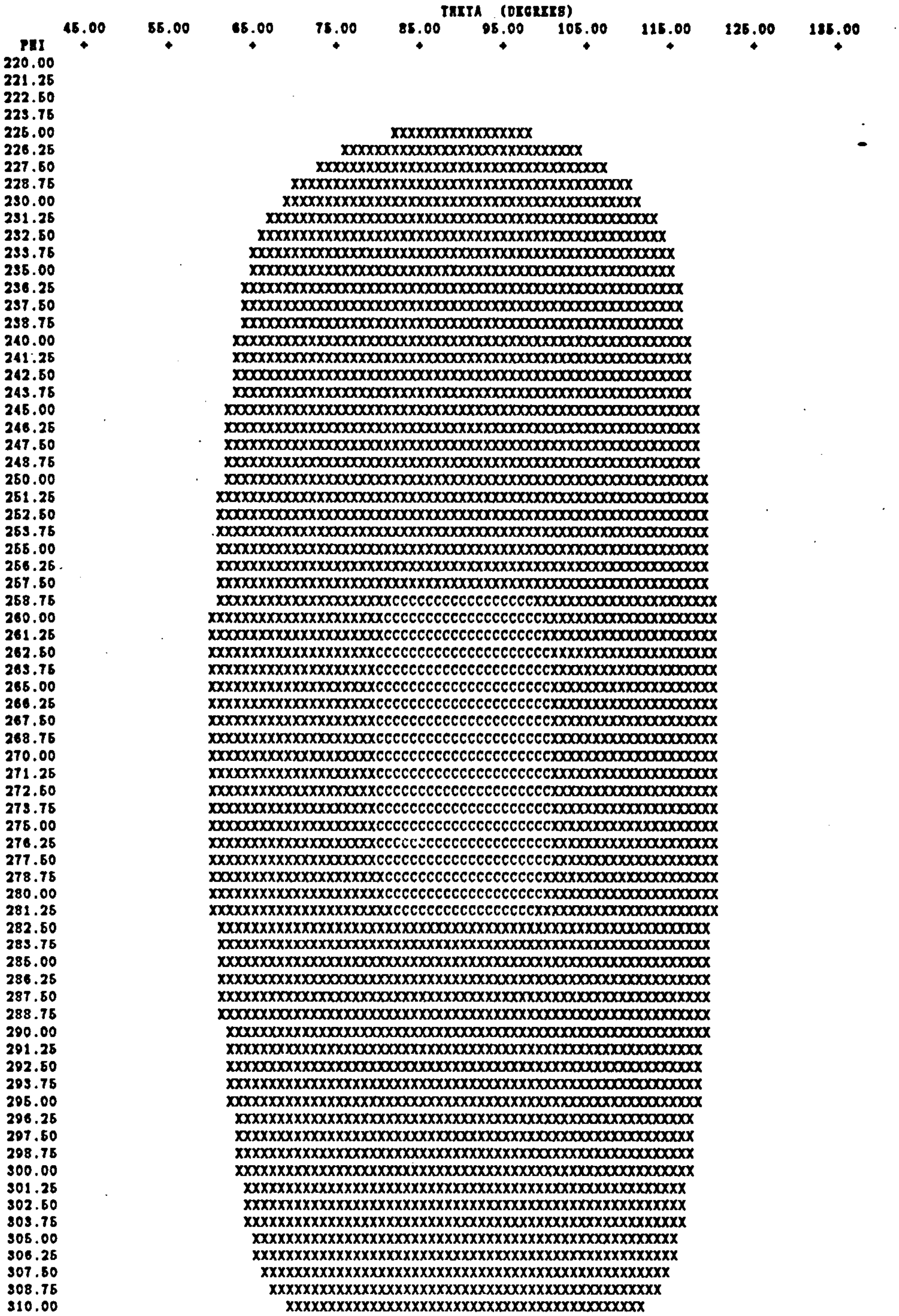
```
1
CC: FIRST CYLINDER
0.,-2.,0.
90.,0.,0.,0.
2
1.,1.,  1.
1.,1., -1.
CC: SECOND CYLINDER
0.,+2.,0.
90.,0.,0.,0.
2
1.,1.,  1.
1.,1., -1.
XQ:
EN:
```

The output generated by the code was the following:

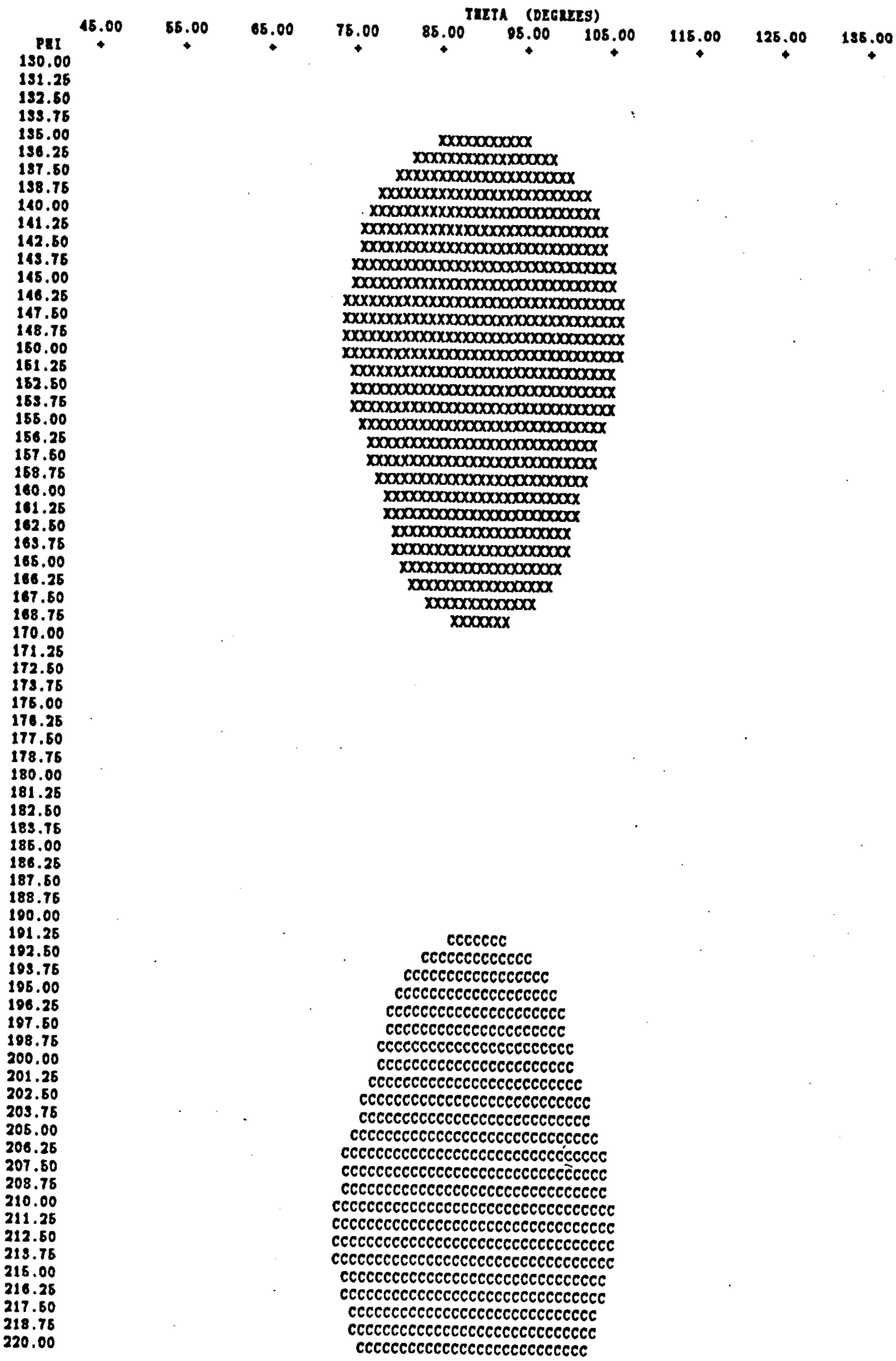
```
ANTENNA (RCS) = (  0.0000,  0.0000,  4.0000  ) IN METERS     INPUT SET: USER1:[RJM.NAS.MAN]CYLEX2.INP;2
```



ANTENNA (RCS) = (0.0000, 4.0000, 0.0000) IN METERS   INPUT SET: USER1:[RJN.NAS.MAN]CYLEX2.INP;2



ANTENNA (RCS) = (4.0000, 0.0000, 0.0000) IN METERS   INPUT SET: USER1:[RJM.NAS.MAN]CYLEX2.INP;2



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 effectivly show obscuration.

The commands were:

```
$ RUN SHADOW
SHADOW> SET OUT  AN5S1 /PRINT/NOECHO
SHADOW> SET INP  AN5S1
SHADOW> SET UNI  FEET
SHADOW> SET WIND
0,180
2.0
20.,290
2.5
!
! Display ONLY plate 6.
!
SHADOW> SET FILL " " /PLATE=6
SHADOW> SET ANT
        25, 15, 256.5
SHADOW> SHADOW
!
! Now make plate 6 stand out from the crowd.
!
SHADOW> SET FILL "!" /PLATE=(6,$)
SHADOW> SET ANT
        25, 15, 256.5
SHADOW> SHADOW
$ EXIT
```

The input set defining the plate was the following:

```
CM: *********CASE AN5S1**********
CM: ********OBSCURATION**********
CE:
LP:
F
UN: UNITS IN FEET
2
CM: 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.
PG: +X SIDE
4 0
4.5 4.5 387.
4.5 -4.5 387.
4.5 -4.5 270.
4.5 4.5 270.
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 99.
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.
```

```
PG: -Y #3
4 0
4.5 -4.5 99.
4.5 -4.5 261.
-4.5 -4.5 261.
-4.5 -4.5 99.
PG: +Y #1
4 0
4.5 22.5 0.
-4.5 22.5 0.
-4.5 22.5 99.
4.5 22.5 99.
PG: +Y #2
4 0
4.5 22.5 99.
-4.5 22.5 99.
-4.5 4.5 99.
4.5 4.5 99.
PG: +Y #3
4 0
4.5 4.5 99.
-4.5 4.5 99.
-4.5 4.5 261.
4.5 4.5 261.
CM: NON-ROTATING SECTION
CM: OF SOLAR PANEL BOOM
CE:
PG: BOTTOM
4 0
4.5 49.5 261.
4.5 -49.5 261.
-4.5 -49.5 261.
-4.5 49.5 261.
PG: +X SIDE
4 0
4.5 49.5 270.
4.5 -49.5 270.
4.5 -49.5 261.
4.5 49.5 261.
CM: ROTATING SECTION OF
CM: SOLAR PANEL BOOM
CE:
RT: -Y SIDE
0. -54. 265.5
0. 0. 90. 0.
PG: TOP
4 0
 4.5   4.5 4.5
-4.5   4.5 4.5
-4.5 -76.5 4.5
 4.5 -76.5 4.5
```

```
PG: 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. 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 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
CE:
RT: -Y INBOARD
0. -78. 265.5
0. 0. 90. -52.
PG: -X 82X33
4 0
-1. 16.5 89.
-1. 16.5 7.
-1. -16.5 7.
-1. -16.5 89.
PG: UPPER 33
4 0
1. 16.5 89.
-1. 16.5 89.
-1. -16.5 89.
1. -16.5 89.
PG: LOWER 33
4 0
1. 16.5 7.
1. -16.5 7.
-1. -16.5 7.
-1. 16.5 7.
PG: 82 INSIDE
4 0
1. 16.5 89.
1. 16.5 7.
-1. 16.5 7.
-1. 16.5 89.
```

```
PG: 82 OUTSIDE
4 0
1. -16.5 89.
-1. -16.5 89.
-1. -16.5 7.
1. -16.5 7.
CM: LOWER INBOARD 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: 82 INSIDE
4 0
1. 16.5 -89.
-1. 16.5 -89.
-1. 16.5 -7.
1. 16.5 -7.
PG: 82 OUTSIDE
4 0
1. -16.5 -89.
1. -16.5 -7.
-1. -16.5 -7.
-1. -16.5 -89.
CM: ROTATING SECTION OF
CM: SOLAR PANEL BOOM
CE:
RT: +Y SIDE
0. 54. 265.5
0. 0. 90. 0.
PG: TOP
4 0
4.5 -4.5 4.5
4.5 76.5 4.5
-4.5 76.5 4.5
-4.5 -4.5 4.5
PG: BOTTOM
```

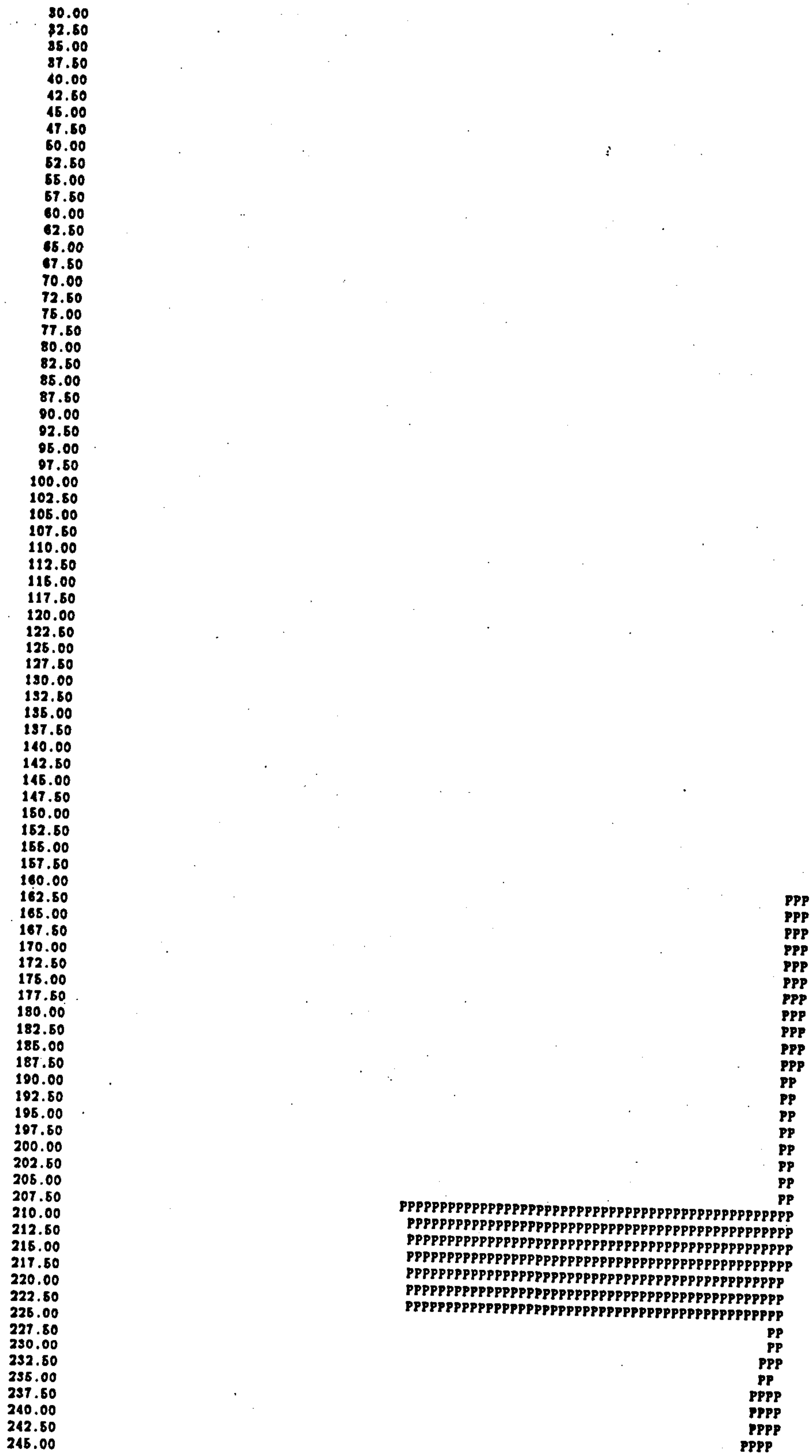
```
4 0
4.5 -4.5 -4.5
-4.5 -4.5 -4.5
-4.5 76.5 -4.5
4.5 76.5 -4.5
PG: +X SIDE
4 0
4.5 -4.5 4.5
4.5 -4.5 -4.5
4.5 76.5 -4.5
4.5 76.5 4.5
PG: -X SIDE
4 0
-4.5 -4.5 4.5
-4.5 76.5 4.5
-4.5 76.5 -4.5
-4.5 -4.5 -4.5
CM: UPPER OUTBOARD SOLAR PANEL
CE:
RT: +Y OUTBOARD
0. 132. 265.5
0. 0. 90. -52.
PG: -X 82X33
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.
```

```
-1. -16.5 -89.
-1. -16.5 -7.
-1. 16.5 -7.
PG: LOWER 33
4 0
1. 16.5 -89.
1. -16.5 -89.
-1. -16.5 -89.
-1. 16.5 -89.
PG: UPPER 33
4 0
1. 16.5 -7.
-1. 16.5 -7.
-1. -16.5 -7.
1. -16.5 -7.
PG: INSIDE 82
4 0
1. -16.5 -89.
1. -16.5 -7.
-1. -16.5 -7.
-1. -16.5 -89.
CM: UPPER INBOARD SOLAR PANEL
CE:
RT: +Y INBOARD
0. 78. 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 OUTSIDE
4 0
1. 16.5 89.
1. 16.5 7.
-1. 16.5 7.
-1. 16.5 89.
PG: 82 INSIDE
```

```
4 0
1. -16.5 89.
-1. -16.5 89.
-1. -16.5 7.
1. -16.5 7.
CM: LOWER INBOARD 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: 82 OUTSIDE
4 0
1. 16.5 -89.
-1. 16.5 -89.
-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.
PP:
T
T 8.186 4.87
180. -180. -30.
-40 40. 4.
XQ: EXECUTE CODE
EN: END CODE
```

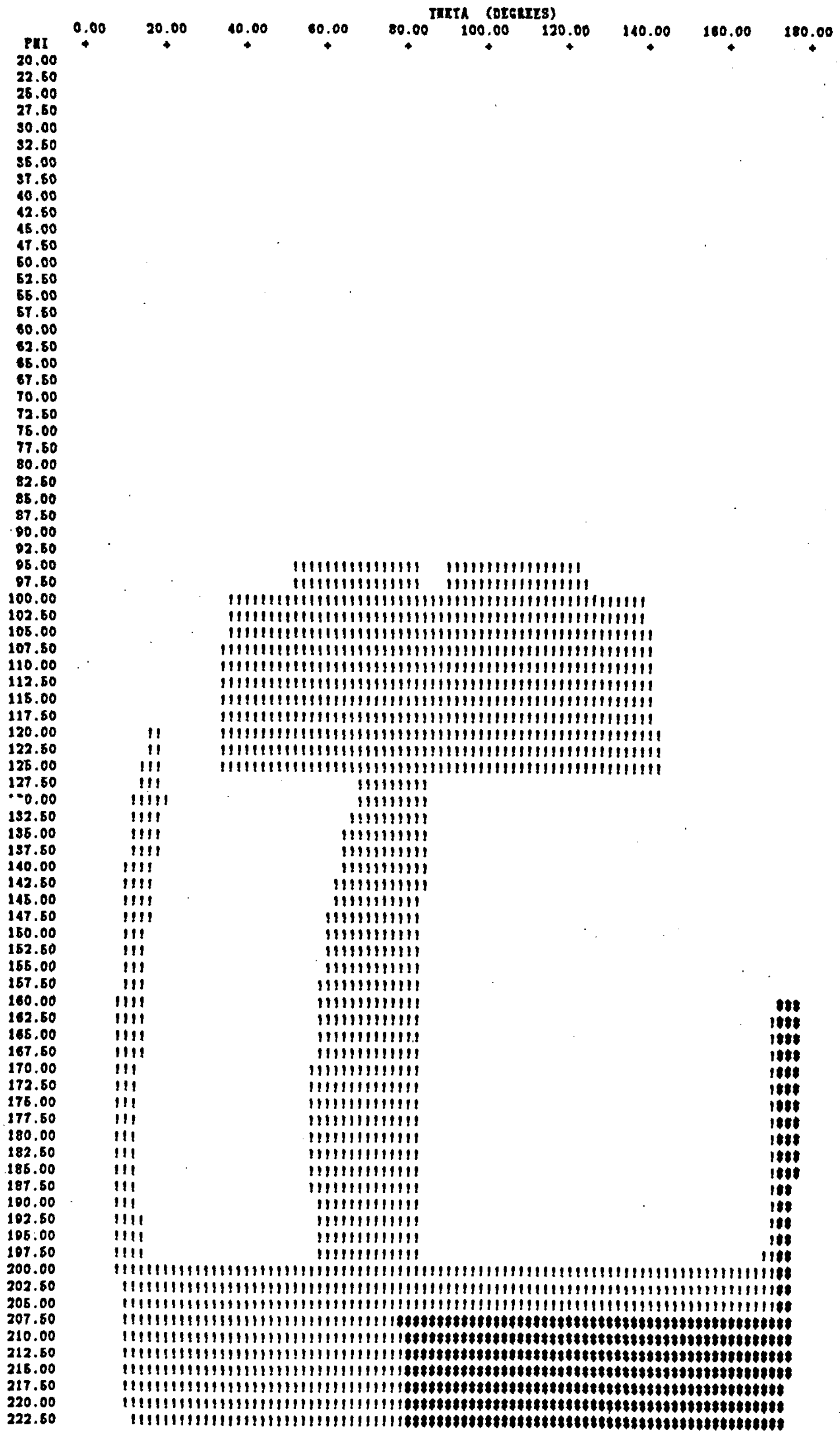
The output generated by the code was the following:

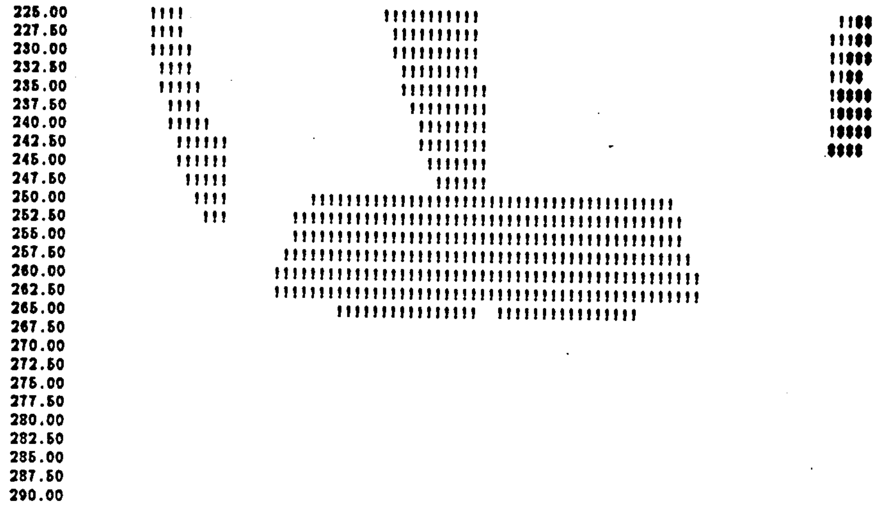
```
ANTENNA (RCS) = (  7.6200,  4.5720, 78.1812  ) IN METERS     INPUT SET: USER1:[RJN.NAS.MAN]AN581.INP;1

                                                   THETA  (DEGREES)
         0.00      20.00     40.00     60.00     80.00    100.00    120.00    140.00    160.00    180.00
  PHI     +         +         +         +         +         +         +         +         +         +
 20.00
 22.50
 25.00
 27.50
```


30.00
32.50
35.00
37.50
40.00
42.50
45.00
47.50
50.00
52.50
55.00
57.50
60.00
62.50
65.00
67.50
70.00
72.50
75.00
77.50
80.00
82.50
85.00
87.50
90.00
92.50
95.00
97.50
100.00
102.50
105.00
107.50
110.00
112.50
115.00
117.50
120.00
122.50
125.00
127.50
130.00
132.50
135.00
137.50
140.00
142.50
145.00
147.50
150.00
152.50
155.00
157.50
160.00
162.50
165.00
167.50
170.00
172.50
175.00
177.50
180.00
182.50
185.00
187.50
190.00
192.50
195.00
197.50
200.00
202.50
205.00
207.50
210.00
212.50
215.00
217.50
220.00
222.50
225.00
227.50
230.00
232.50
235.00
237.50
240.00
242.50
245.00

247.50
250.00
252.50
255.00
257.50
260.00
262.50
265.00
267.50
270.00
272.50
275.00
277.50
280.00
282.50
285.00
287.50
290.00

ANTENNA (RCS) = (7.6200, 4.5720, 78.1812) IN METERS INPUT SET: USER1:[RJM.NAS.MAN]AN581.INP;1




225.00
227.50
230.00
232.50
235.00
237.50
240.00
242.50
245.00
247.50
250.00
252.50
255.00
257.50
260.00
262.50
265.00
267.50
270.00
272.50
275.00
277.50
280.00
282.50
285.00
287.50
290.00

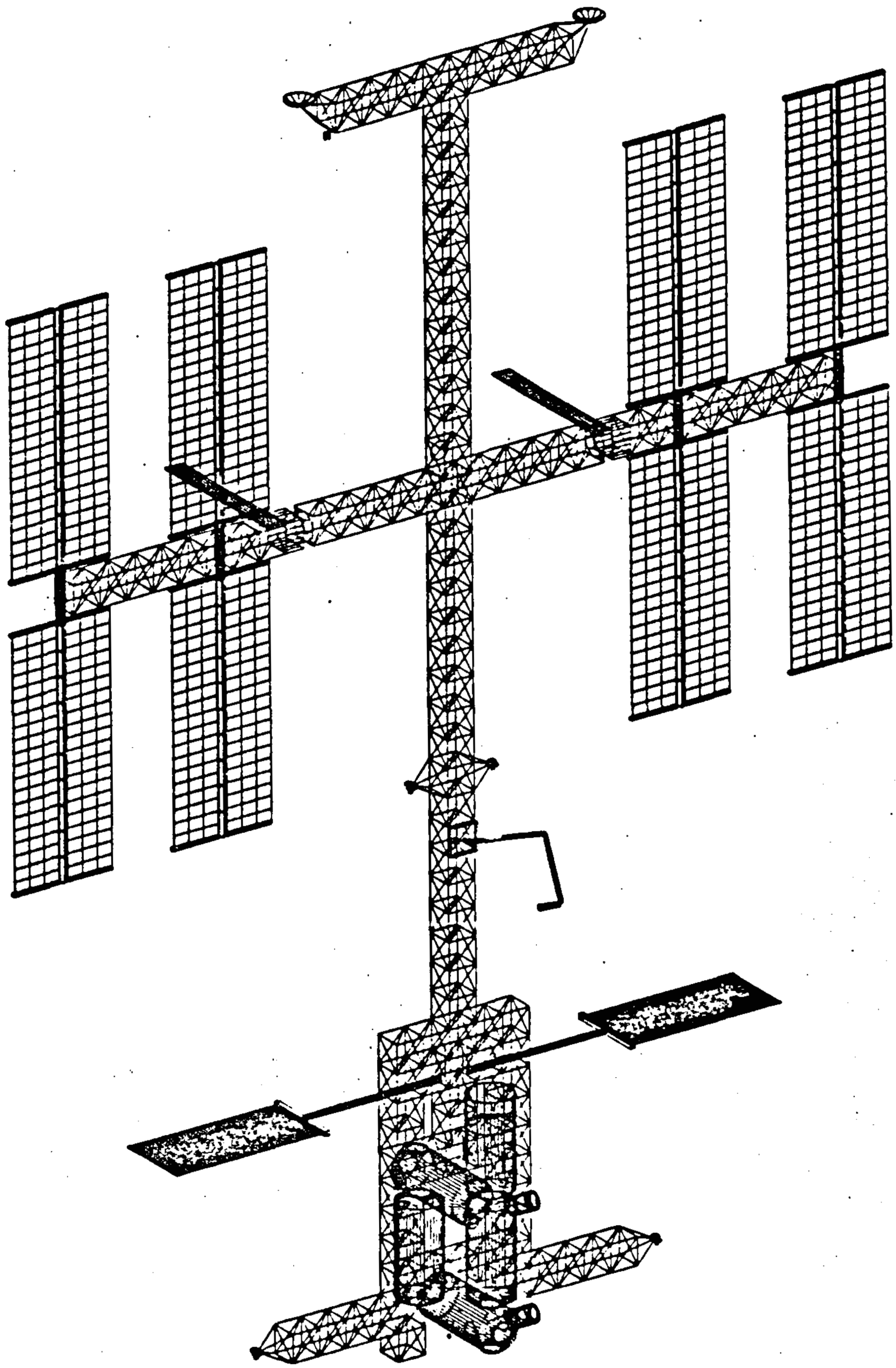


Figure 7.1: Illustration of the Space Station

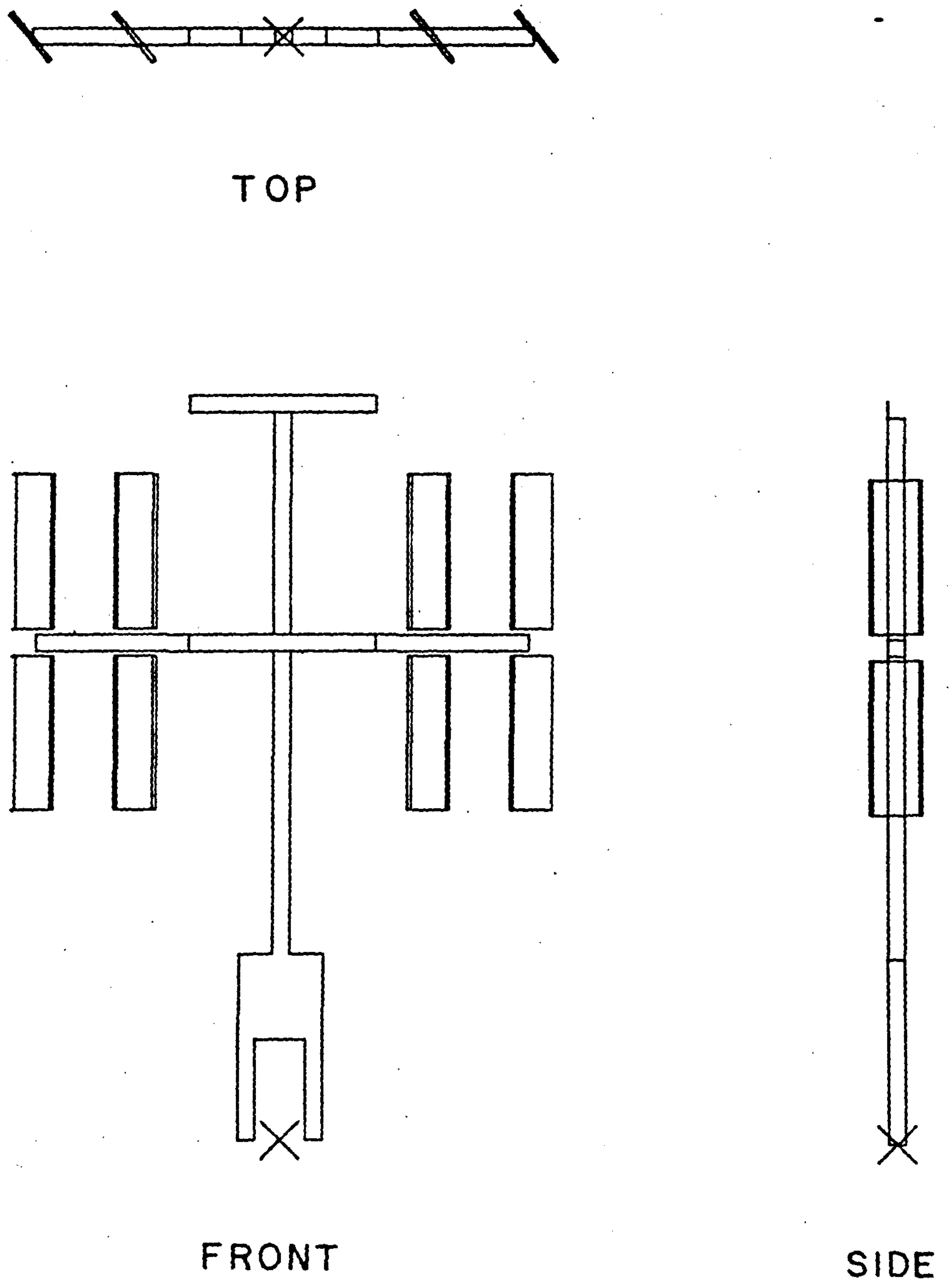


Figure 7.2: Three-axis view of the Space Station as modeled by the input set.

7.8 Example 8: Another Look at the Space Station

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

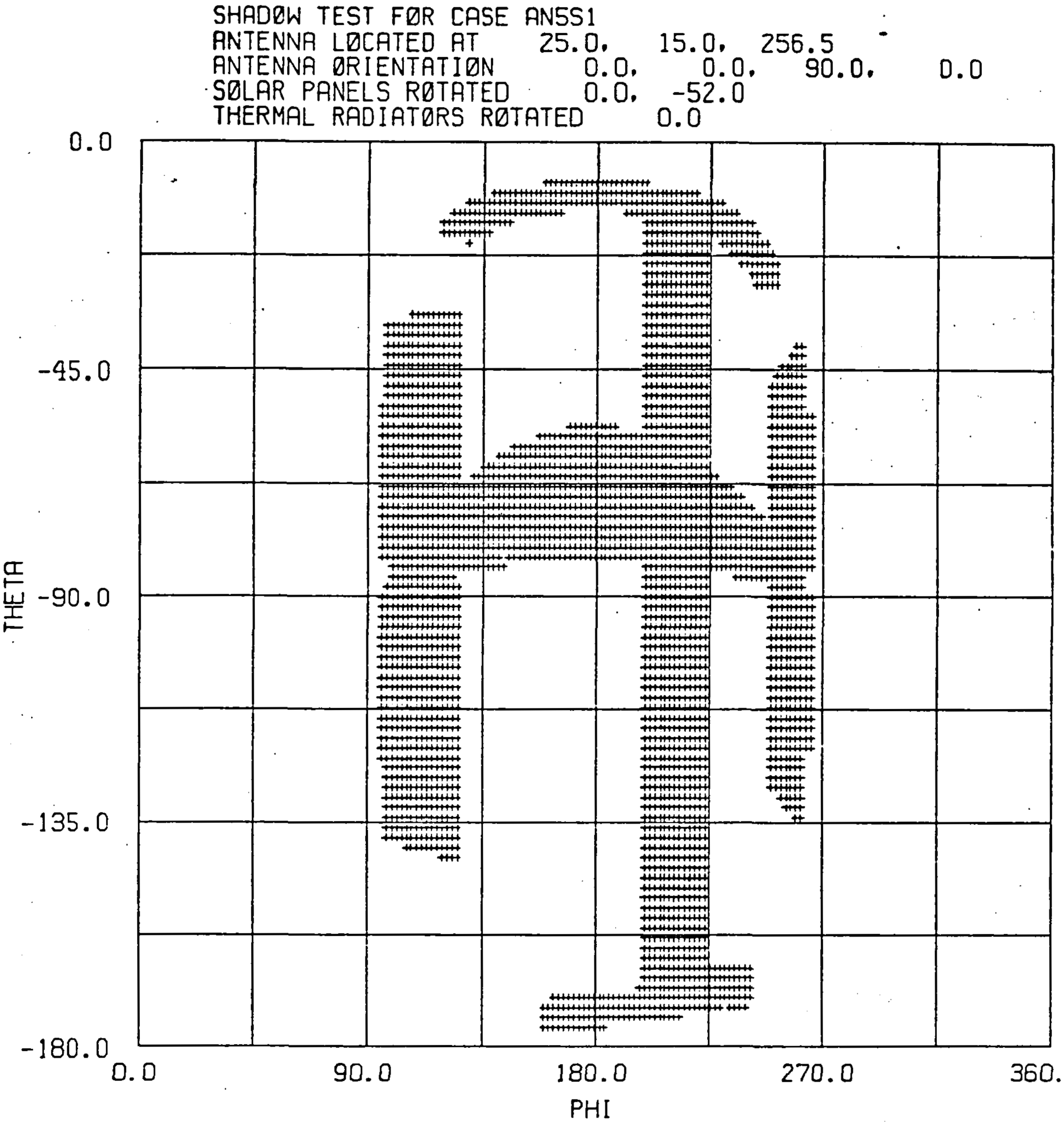


Figure 7.3: NCAR plot showing the shadow map of the space station model.

References

[1] R. J. Marhefka and W. D. Burnside, "Numerical Electromagnetic Code - Basic Scattering Code, NEC-BSC (Version 2), Part I: User's Manual," Technical Report 712242-14, December 1982, The Ohio State University ElectroScience Laboratory, Department of Electrical Engineering; prepared under Contract No. N00123-79-C-1469 for Naval Regional Contracting Office.

[2] R. J. Marhefka, "Numerical Electromagnetic Code - Basic Scattering Code, NEC-BSC (Version 2), Part II: Code Manual," Technical Report 712242-15, December 1982, The Ohio State University ElectroScience Laboratory, Department of Electrical Engineering; prepared under Contract No. N00123-79-C-1469 for Naval Regional Contracting Office.

[3] H. H. Chung and W. D. Burnside, "General 3D Airborne Antenna Radiation Pattern Code User's Manual," Technical Report 711679-10, July 1982, The Ohio State University ElectroScience Laboratory, Department of Electrical Engineering; prepared under Contract No. F30602-79-C-0068 for Air Force Systems Command.

[4] R. G. Kouyoumjian and P. H. Pathak, "A Uniform Geometrical Theory of Diffraction for an Edge in a Perfectly Conducting Surface," Proc. IEEE, Vol. 62, pp. 1448–1461, November 1974.

[5] G. R. McArthur, editor, "The System Plot Package," NCAR Technical Note - 162+IA, January 1981, National Center for Atmospheric Research, Boulder, Colorado.

Part II

Code Manual

Chapter 8

Introduction

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

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

Chapter 9

Code Organization

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

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

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

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

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

Chapter 10

Listings of the Code

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

10.1 VAX/VMS Subroutines

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

MAIN PROGRAM

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

```
      PROGRAM SHADOW
C!!!++
C!!! This program was written at the ohio state university
C!!! electroscience laboratory. any problems or comments
C!!! can be referred to:
C!!!
C!!!          LASZLO  A. TAKACS  OR  RONALD  J. MARHEFKA
C!!!          ELECTROSCIENCE LABORATORY
C!!!          1320 KINNEAR RD.
C!!!          COLUMBUS,OHIO 43212
C!!!          PHONE: (614) 422-5752 OR 422-5848
C!!!
C!!! This program provides a printer output of the geometrical
C!!! shadow boundries of a structure of plates and cylinders input
C!!! as valid input sets to the numerical code.
C!!!
C!!! This program was written 15-JUN-1984.
C!!! The latest modification occurred 18-DEC-1985.
C!!!--
C!!!
C!!! Beginning of the main routine.
```

```
C!!!  Initialize any SHADOW data structures.
C!!!
      CALL INIT
C!!!
C!!!  Call the interactive terminal interface.  This routine calls all
C!!!  other subroutines.
C!!!
      CALL INTRAC
C!!!
C!!!  Finished.
C!!!
      END
```

SUBROUTINE INIT

```

C---------------------------------------------------------------------
      SUBROUTINE INIT
      INCLUDE  'SHACOM.FOR'
C!!!
C!!! This subroutine initializes the main routine.
C!!! It is meant to be called once, at the start of the program.
C!!!
C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C!!!                                                                   !
C!!! NOTICE:                                                           !
C!!!  This routine performs actions which do not apply to the          !
C!!!  non-interactive mode of operation.  In particular, the variables !
C!!!  which are intialized here may be reinitialized elsewhere in both !
C!!!  interactive and non-interactive versions.  Altering these        !
C!!!  parameters may or may not achieve the expected results.          !
C!!!                                                                   !
C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C!!!
C!!! Initialize variables to their default values.
C!!!
C!!! The lower/higher theta end of the range.
C!!!
      THET1   =        0.0          * RPD
      THET2   =      180.0          * RPD
C!!!
C!!! The lower/higher phi end of the range.
C!!!
      PH1     =        0.0          * RPD
      PH2     =      360.0          * RPD
C!!!
C!!! The desired theta/phi resolution in units of radians/pixel.
C!!!
      RESTH   =      2.       * RPD
      RESPH   =      2.       * RPD
C$$$
C$$$ Rotate translate default data RT:
C$$$
      THZP    =   0.
      PHZP    =   0.
      THXP    =  90.
      PHXP    =   0.

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

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

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

      VRS( 3, 1 ) = 0.
      VRS( 3, 2 ) = 0.
      VRS( 3, 3 ) = 1.
C$$$
C$$$ Units default data UN:, UF:
```

```
C$$$
      IUNIT   = 1
      UNITF   = 1.0
      UNITN   = UNIT( IUNIT )
      UNITS   = UNITN * UNITF
C$$$
C$$$ Pattern cut orientation data VF:
C$$$
      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.
C!!!
C!!! Open some standard input/output files for the VMS support routines.
C!!! Units 5 and 6 are reserved for input set reading and echoing by the
C!!! input set processor.  NOTE:  This is operating system dependent
C!!! stuff.  This is the natural place for it since it is intialized
C!!! at the start.
C!!!
      OPEN( UNIT=1,FILE='SYS$INPUT',TYPE='OLD' )
      OPEN( UNIT=2,FILE='SYS$OUTPUT',TYPE='UNKNOWN' )
C!!!
C!!! End of program intialization.
C!!!
      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.

```
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-1985
!
! MODIFIED BY:
!
! 1-001 -  Original,  LAT 20-AUG-1985
!--
        IMPLICIT        NONE
        INCLUDE         '($RMSDEF)'
        INCLUDE         '($SMGDEF)'
        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$LOAD_KEY_DEFS,
     +                  SMG$CREATE_KEY_TABLE,
     +                  SMG$DELETE_VIRTUAL_KEYBOARD,
     +                  SMG$CREATE_VIRTUAL_KEYBOARD
!
! Make a key definiton table.
!
        STS = SMG$CREATE_KEY_TABLE( KEYTBL )
         IF (.NOT. STS) CALL LIB$SIGNAL( %VAL(STS) )
!
! Load the definitions from the key definition file.  Ignore "file not f
!
        STS = SMG$LOAD_KEY_DEFS( KEYTBL, 'SHADOW.KPD' )
         IF ((.NOT. STS) .AND. (STS .NE. RMS$_FNF))
     +                          CALL LIB$SIGNAL( %VAL(STS) )
!
! Get a handle on SYS$INPUT.
!
        READ_STS = SMG$CREATE_VIRTUAL_KEYBOARD( KBDID )
```

```
!
! The main processing loop.  Keep reading input until the user types EOF
!
      DO WHILE ( READ_STS .NE. RMS$_EOF )
!
! Read from input and parse the command.
!
      READ_STS = CLI$DCL_PARSE(,
     +                                                COMMAND_TABLES,
     +                                                GET_INPUT,
     +                                                GET_INPUT,
     +                                                'SHADOW> '      )
!
! If the command parse was successful, execute the command-routine.
!
      IF ( .NOT. (.NOT. READ_STS) ) CALL CLI$DISPATCH()
!
      END DO
!
! Get rid of the virtual keyboard.
!
      STS = SMG$DELETE_VIRTUAL_KEYBOARD( KBDID )
       IF ( .NOT. STS ) CALL LIB$SIGNAL( %VAL(STS) )
!
! Return
!
      RETURN
      END
C-----------------------------------------------------------------------
      INTEGER*4       FUNCTION   GET_INPUT( COMMAND, PROMPT, LENGTH )
C!!!
C!!! This routine does all the reading for the terminal interface.
C!!! It has the same calling format as LIB$GET_INPUT except that optiona
C!!! parameters may not be omitted.
C!!!
      INCLUDE   '($RMSDEF)'
      INCLUDE          'SHACOM.FOR'
      EXTERNAL
     +        SMG$_EOF                                ! The linker finds
      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$READ_COMPOSED_LINE (
     +                                                KBDID,
     +                                                KEYTBL,
     +                                                COMMAND,
     +                                                PROMPT,
     +                                                LENGTH  )
      IF ( GET_INPUT .EQ. %LOC( SMG$_EOF ) ) GET_INPUT = RMS$_EOF

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

```
        INCLUDE         'SHACOM.FOR'
        CHARACTER*(*)
        +                                       STRING
        INTEGER*4
        +                                       LIB$PUT_OUTPUT
!
! Read a line.
!
        PUT_OUTPUT = LIB$PUT_OUTPUT ( STRING )
!
! There should be no errors here.  Signal if there are any.
!
        IF (.NOT. PUT_OUTPUT) CALL LIB$SIGNAL( %val(PUT_OUTPUT) )
!
! Return.
!
        RETURN
        END
```

Interactive Service Routines

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

```
      C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
      C!
      C! The system-dependent stuff goes below here.
      C!
      C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
      C!
      C!++
      C!
      C! FUNCTIONAL DESCRIPTION:
      C!
      C!    These functions are the action routines invoked by the VERB which
      C!    follows from the on each routine.
      C!
      C! CALLING SEQUENCE:
      C!
      C!    ret-status.wlc.ds = routine ( )
      C!
      C! FORMAL PARAMETERS:
      C!
      C!    NONE
      C!
      C! IMPLICIT INPUTS:
      C!
      C!    FUNCTION SPECIFIC
      C!
      C! IMPLICIT OUTPUTS:
      C!
      C!    FUNCTION SPECIFIC
      C!
      C! COMPLETION STATUS:
      C!
      C!    FUNCTION SPECIFIC
      C!
      C!     SS$_NORMAL      Success, or
      C!    fac$_status      some other status
      C!
      C! SIDE EFFECTS:
      C!
      C!    VARIBLE
      C!--
            INTEGER          FUNCTION        SERVICE_ROUTINES
            IMPLICIT         NONE
            PARAMETER        SUCCESS = 1
            INCLUDE '($SSDEF)/NOLIST'        ! Include system status defintions
            INCLUDE 'SHACOM.FOR/LIST'        ! Include SHADOW common block
  1   C!!!
  1   C!!!  COMMON declerations...
  1   C!!!
  1         COMMON /PIS/
  1      +          PI,
  1      +          TPI,
  1      +          DPR,
  1      +          RPD
  1   C!!!
```

```
1 C+++  MAXIMUM DIMENSION FOR PLATES
1       INTEGER          NPX
1       PARAMETER (NPX=75)
1 C+++  MAXIMUM DIMENSION FOR PLATE EDGES
1       INTEGER          NEX
1       PARAMETER (NEX=12)
1 C+++  MAXIMUM DIMENSION FOR CYLINDERS
1       INTEGER          NCX
1       PARAMETER (NCX=5)
1 C+++  MAXIMUM DIMENSION FOR CYLINDER RIMS
1       INTEGER          NNX
1       PARAMETER (NNX=10)
1 C+++  MAXIMUM DIMENSION FOR ROWS (PHI)
1       INTEGER          MAXROW
1       PARAMETER (MAXROW=361)
1 C+++  MAXIMUM DIMENSION FOR COLUMNS (THETA)
1       INTEGER          MAXCOL
1       PARAMETER (MAXCOL=181)
1 C!!!
1       COMMON /GEOPLA/
1      +        XX      (3,NEX,NPX),
1      +        V       (3,NEX,NPX),
1      +        VP      (3,NEX,NPX),
1      +        VN      (3,NPX),
1      +        MEP     (NPX),
1      +        MPX
1 C!!!
1       COMMON /GEOMEL/
1      +        AC      (NNX,NCX),
1      +        BC      (NNX,NCX),
1      +        ZC      (NNX,NCX),
1      +        TCR     (NNX,NCX),
1      +        XCL     (3,NCX),
1      +        VCL     (3,3,NCX),
1      +        NEC     (NCX),
1      +        MCX
1 C!!!
1       COMMON /EDMAG/ VMAG(NEX,NPX)
1 C!!!
1       COMMON /SHADWN/ COLS, ROWS, ANTENN(3),CTROID(3),
1      +                        MP,ME,NEXTME,MC,
1      +                        THET1,THET2,PH1,PH2,RESTH,RESPH,ALPH,
1      +                        UNIT(3),TRS(3),VRS(3,3),IUNIT,UNITF,UNITS,UN
1      +                        THZP,PHZP,THXP,PHXP,FILPNM,FILCNM
1       COMMON /SHADWC/ INPFIL,OUTBUF(MAXCOL,MAXROW),
1      +                FILCHC,FILCHP,FILCHR
1 C!!!
1       COMMON /PATCUT/ VPC(3,3)
1 C!!!
1 C!!!  The first set of declarations is the stuff in /SHADOW/ common bloc
1 C!!!
1       INTEGER
1      +        MP, ME, NEXTME, MC,
1 C! Plate#/edge#/cyl# variables.
1      +  FILPNM, FILCNM,
1 C! Plate and cyl numbers for special filling
1      +        COLS,
1 C! The size of the array subsection determined
1      +        ROWS
1 C! by internal resolution requirements.
1
1       REAL
1      +        CTROID,
1 C! A geometric center of the object in question.
```

```
1     +          ANTENN,
1 C! The antenna location in Ref Coord. System.
1     +          THET1,
1 C! The lower theta end of the range.
1     +          THET2,
1 C! The higher theta end of the range.
1     +          PH1,
1 C! The lower phi end of the range.
1     +          PH2,
1 C! The higher phi end of the range.
1     +          RESTH,
1 C! The desired theta/phi resolution
1     +          RESPH,
1 C! in units of radians/pixel.
1     +          ALPH
1 C! Maximum allowed angular excursion.
1
1        CHARACTER
1     +          OUTBUF*1,
1 C! The output buffer which is displayed.
1     +          INPFIL*63,
1 C! The filename of the input set.
1     + FILCHC,
1 C! special fill character for cylinders
1     + FILCHP,
1 C! special fill character for everything else
1     + FILCHR
1 C! special fill character for plates.
1        DATA    FILCHC, FILCHP, FILCHR / 'C', 'P', 'X' /
1 C!!!
1 C!!!  From the /PIS/ COMMON block...
1 C!!!
1        REAL PI, TPI, DPR, RPD
1 C!!!
1 C!!!  From the /GEOPLA/ COMMON block...
1 C!!!
1        INTEGER
1     +          MEP,
1 C! Number of edges per plate
1     +          MPX
1 C! Total number of plates
1        REAL
1     +          XX,
1 C! The array of plate corners
1     +          V,
1 C! Edge unit vectors
1     +          VP,
1 C! Edge unit binormals
1     +          VN
1 C! Unit normal for each plate
1 C!!!
1 C!!!  From the /GEOMEL/ COMMON block...
1 C!!!
1        INTEGER
1     +          NEC,
1 C! Number of sections per cylinder
1     +          MCX
1 C! Total number of cylinders
1        REAL
1     +          AC,
1 C! Elliptic parameter along x-axis
1     +          BC,
1 C! Elliptic parameter along y-axis
1     +          ZC,
```

```
1 C! Cylinder endcaps in cyl coord sys
1      +          TCR,
1 C! Angle endcap makes with positive z axis
1      +          XCL,
1 C! Cyl coord sys origin
1      +          VCL
1 C! Definition of cyl coord sys
1 C!
1         INTEGER
1      +                    IUNIT
1         REAL
1      +                    UNITF,
1      +                    UNITS,
1      +                    UNITN,
1      +                    UNIT,
1      +                    TRS,
1      +                    THZP,PHZP,THXP,PHXP,
1      +                    VRS,
1      +  VPC,
1      +  VMAG
1         DATA UNIT/1.,.3048,0.0254/
1 C!
1 C!!!+
1 C!!! The following common block is for VMS/SMG$ software only.
1 C!!!
1         INTEGER                                    KBDID, KEYTBL
1         COMMON  /TERCOM/                           KBDID, KEYTBL
1 C!!!-
          EXTERNAL
     +          PUT_OUTPUT, GET_INPUT,  ! My own $SMG-type I/O routines
     +          CLI$_PRESENT,           !
     +  CLI$_NEGATED,           !
     +  CLI$_LOCPRES,           ! locally present
     +  CLI$_LOCNEG,                    ! locally negated
     +  CLI$_DEFAULTED,
     +  CLI$_ABSENT,
     +          CLI$_IVVALU

          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
     +                  /'meters', 'feet  ', 'inches'/,
     +          FILE  *50,              ! Temproary file variable
     +          PRTFIL*50,              ! Printable file
     +          LISFIL*50,              ! Input echo listing
     +          OUTFIL*50               ! "Plottable" output file
!
          DATA IUNIT/1/

          LOGICAL*4
     +          VALID_INPUT,            ! A loop control variable
     +          CLI$PRESENT,            ! CLI interface to get info about
     +          CLI$GET_VALUE           ! CLI interface to get info about

          REAL*4
     +          DOT,DZX,XQ(3)
          INTEGER*4
     +          N,NI,NJ,STS,            ! sordid variables...
     +  KEYPAD,                         ! Keypad condition flag
!
! General library routines
```

```
!
     +        LIB$SPAWN,                ! Executes a subprocess
     +        LBR$OUTPUT_HELP,          ! The librarian help routine
     +        SMG$SET_KEYPAD_MODE,      ! Screen management package
!
! "SET/SHOW" routines
!
     +        SET_ANT,                  SET_OUT,                SET_COO,
     +        SET_PAT,                  SET_SCA,                SET_WIN,
     +        SET_KEY,                  SET_INP,                SET_UNI_ME
     +        SET_UNI_INCHES,           SET_UNI_FEET,
     +        SHOW_ANT,                 SHOW_OUT,               SHOW_COO,
     +        SHOW_PAT,                 SHOW_SCA,               SHOW_WIN,
     +        SHOW_KEY,                 SHOW_INP,
     +        SHOW_UNI,
!
! various command routines
!
     +        EXIT_COMMAND,   HELP_COMMAND,   DCL_COMMAND,    SHADOW_COM

C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C!
C! This routine sets the current fill characters being used for plates
C! or cylinders.
C!
      ENTRY SET_FIL
       IF       ( CLI$PRESENT( 'SEQUENTIAL' ) )  THEN
C!
C! Reset things to their default state.
C!
        FILPNM = -1
        FILCNM = -1
        FILCHP = 'P'
        FILCHC = 'C'
        FILCHR = 'X'
C!
C! To avoid screwing up the test in SCAN, use a character that will
C! not be used by the fill process, like char 7.
C! Set a plate up for tagging.
C!
       ELSEIF   ( CLI$PRESENT( 'PLATE' ) )  THEN
C!
C! Clear any cylinder tagging residue.
C!
        FILCNM = 0
        FILCHC = 'C'
C!
C! Get the master fill character.
C!
        CALL CLI$GET_VALUE( 'P2', FILCHR )
C!
C! Get the qualifier numeral value.  STS is being used for the length of
C!  and the status of the decode.
C!
        IF ( CLI$GET_VALUE( 'PLATE', P2, STS ) ) THEN
         DECODE (STS,1,P2,IOSTAT=STS) FILPNM
        ELSE
         STS = -1
        ENDIF
C!
C! Get the fill character for that plate.  Use a 'P' if none is given.
C!
        IF ( STS .NE. 0 ) THEN
         SET_FIL = %LOC( CLI$_IVVALU )
```

```
          ELSE
           IF ( .NOT. CLI$GET_VALUE('PLATE',FILCHP) ) THEN
            FILCHP = 'P'
           ENDIF
          ENDIF
C!
C! Set a cylinder up for tagging.
C!
         ELSEIF   ( CLI$PRESENT( 'CYLINDER' ) )  THEN
C!
C! Clear any cylinder tagging residue.
C!
          FILPNM = 0
          FILCHP = 'P'
C!
C! Get the master fill character.
C!
          CALL CLI$GET_VALUE( 'P2', FILCHR )
C!
C! Get the qualifier numeral value.  STS is being used for the length of
C!  and the status of the decode.
C!
          IF ( CLI$GET_VALUE( 'CYLINDER', P2, STS ) ) THEN
           DECODE (STS,1,P2,IOSTAT=STS) FILCNM
          ELSE
           STS = -1
          ENDIF
C!
C! Get the fill character for that cylinder.  Use a 'C' if none is given
C!
          IF ( STS .NE. 0 ) THEN
           SET_FIL = %LOC( CLI$_IVVALU )
          ELSE
           IF ( .NOT. CLI$GET_VALUE( 'CYLINDER', FILCHC ) ) THEN
            FILCHC = 'C'
           ENDIF
          ENDIF
C!
C! The else here is for a "SET FILL [x]" command.
C!
         ELSE
C!
C! Get the master fill character.
C!
          CALL CLI$GET_VALUE( 'P2', FILCHR )
C!
C! End of cases.
C!
         ENDIF

         GOTO 3
 1      FORMAT( I )

C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C!
C! This routine displays the current fill characters being used for plat
C! or cylinders.
C!
        ENTRY SHOW_FIL
C!
C! Assume success only when the SHOW command is being executed.
C!
         SHOW_FIL = SUCCESS
C!
```

```
C! Examine the plate situation.
C!
 3      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
C!
C! Examine the cylinder situation.
C!
        IF ( FILCNM .GT. 0 ) THEN
         WRITE(2,FMT='('' Cylinder '',I3,'' tagged with ['',A,'']'')')
     +                    FILCNM, FILCHC
         WRITE(2,FMT='('' All other geometry tagged with ['',A,'']'')')
     +                    FILCHR
        ENDIF
C!
C! Check on a no-tag backgroung character situation.
C!
        IF ( (FILCNM .EQ. 0) .AND. (FILPNM .EQ. 0) ) THEN
         WRITE(2,FMT='('' No individual plates/cyliders are tagged'')')
         WRITE(2,FMT='('' All geometry marked by ['',A,'']'')') FILCHR
        ENDIF
C!
C! Report the sequential numbering case.
C!
        IF ( ( FILCNM .LT. 0 ) .AND. ( FILPNM .LT. 0 ) )
     +  WRITE(2,FMT='('' All cylinders/plates sequentially tagged'')')
C!
       RETURN


C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C!
C! This routine sets the antenna location.
C!
       ENTRY   SET_ANT
        WRITE (2,FMT='(1X,''Input antenna location in '',A6,'': '',$)')
     +.              LABEL(IUNIT)
        READ  (1,*) ANTENN(1), ANTENN(2), ANTENN(3)
C!
C! Perform appropriate units conversion here.
C!
       DO 3424 N=1,3
 3424  XQ(N)=ANTENN(N)

       DO 3425 N=1,3
 3425  ANTENN(N)=UNITS*
     +    ( XQ(1)*VRS(1,N) + XQ(2)*VRS(2,N) + XQ(3)*VRS(3,N) ) + TRS(N)

        SET_ANT  = SUCCESS
C!
C! This routine displays the current antenna position.
C!
       ENTRY   SHOW_ANT
C!
C! Transform the antenna back
C!
               DO N=1,3
                XQ(N) = ( (ANTENN(1)-TRS(1)) * VRS(N,1) +
     +                    (ANTENN(2)-TRS(2)) * VRS(N,2) +
     +                    (ANTENN(3)-TRS(3)) * VRS(N,3) ) / UNITS
       END DO
C!
```

```
        WRITE(2,FMT='('' Antenna in RCS (meters): '',3F12.5)') ANTENN
        WRITE(2,FMT='('' Definit system ('',A,''): '',3F12.5)')
     +        LABEL(IUNIT), XQ
        SHOW_ANT = SUCCESS
       RETURN

C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C!
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!
C! This routine displays the current input set name.
C!
       ENTRY   SHOW_INP
        INQUIRE        ( UNIT=5, NAME=INPFIL )
        TYPE *, 'The current input set is ', INPFIL
        SHOW_INP = SUCCESS
       RETURN

C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C!
C! This routine toggles/report keypad mode.
C!
       ENTRY   SET_KEY
        IF ( .NOT. CLI$PRESENT( 'KEYPAD_MODE' ) ) THEN
       KEYPAD = 0
        ELSE
         KEYPAD = 1
        END IF
        SET_KEY = SMG$SET_KEYPAD_MODE( KBDID, KEYPAD )
C!
C! This routine displays the current keypad mode.
C!
       ENTRY SHOW_KEY
        IF ( KEYPAD .EQ. 0 )   THEN
         WRITE(2,*) 'The keyboard is not in keypad mode.'
        ELSE
         WRITE(2,*) 'The keyboard is in keypad mode.'
        END IF
       RETURN

C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C!
C! Set up a coordinate system.
C!
       ENTRY   SET_COO
C$$$
C$$$  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$$$
C$$$  THZP,PHZP=ORIENTATION OF THE VRS(3,N) AXIS RELATIVE TO THE
C$$$  FIXED COORDINATE SYSTEM.
C$$$
```

```
C$$$  THXP,PHXP=ORIENTATION OF THE VRS(1,N) AXIS RELATIVE TO THE
C$$$  FIXED COORDINATE SYSTEM.
C$$$
123   continue
      type*,'Please input THZP,PHZP,THXP,PHXP in degrees:'
      accept*, THZP,PHZP,THXP,PHXP
      VRS(3,1)=SIN(THZP*RPD)*COS(PHZP*RPD)
      VRS(3,2)=SIN(THZP*RPD)*SIN(PHZP*RPD)
      VRS(3,3)=COS(THZP*RPD)
      VRS(1,1)=SIN(THXP*RPD)*COS(PHXP*RPD)
      VRS(1,2)=SIN(THXP*RPD)*SIN(PHXP*RPD)
      VRS(1,3)=COS(THXP*RPD)
C!!!  INSURE VRS(1,N) IS PERPENDICULAR TO VRS(3,N)
      DZX=VRS(3,1)*VRS(1,1)+VRS(3,2)*VRS(1,2)+VRS(3,3)*VRS(1,3)
      IF(ABS(DZX).GT.0.1) THEN
       TYPE*, 'The coordinates are NOT orthogonal - Respecify.'
       goto 123
      ELSE
       VRS(1,1)=VRS(1,1)-VRS(3,1)*DZX
       VRS(1,2)=VRS(1,2)-VRS(3,2)*DZX
       VRS(1,3)=VRS(1,3)-VRS(3,3)*DZX
       DOT=VRS(1,1)*VRS(1,1)+VRS(1,2)*VRS(1,2)+VRS(1,3)*VRS(1,3)
       DOT=SQRT(DOT)
       VRS(1,1)=VRS(1,1)/DOT
       VRS(1,2)=VRS(1,2)/DOT
       VRS(1,3)=VRS(1,3)/DOT
       VRS(2,1)=VRS(3,2)*VRS(1,3)-VRS(3,3)*VRS(1,2)
       VRS(2,2)=VRS(3,3)*VRS(1,1)-VRS(3,1)*VRS(1,3)
       VRS(2,3)=VRS(3,1)*VRS(1,2)-VRS(3,2)*VRS(1,1)
       WRITE(6,3931)
      END IF
C!
C! Display the coordinates
C!
      ENTRY SHOW_COO
C!
3931   FORMAT(2H *,5X,'The following rotations are used for ALL',
     2' subsequent inputs:',T79,1H*)
       DO 3932 NI=1,3
3932    WRITE(6,3933) (NI,NJ,VRS(NI,NJ),NJ=1,3)
3933   FORMAT(2H *,1X,3(2X,'VRS(',I1,',',I1,')=',F9.5),T79,1H*)
C!
      RETURN

C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C!
C! set up pattern cut coordinate system
C!
      ENTRY   SET_PAT
C$$$
C$$$  THZP,PHZP=ORIENTATION OF THE VPC(3,N) AXIS RELATIVE TO THE
C$$$  FIXED COORDINATE SYSTEM.
C$$$
C$$$  THXP,PHXP=ORIENTATION OF THE VPC(1,N) AXIS RELATIVE TO THE
C$$$  FIXED COORDINATE SYSTEM.
C$$$
1234  continue
      type*,'Please input THZP,PHZP,THXP,PHXP in degrees:'
      accept*, THZP,PHZP,THXP,PHXP
      VPC(3,1)=SIN(THZP*RPD)*COS(PHZP*RPD)
      VPC(3,2)=SIN(THZP*RPD)*SIN(PHZP*RPD)
      VPC(3,3)=COS(THZP*RPD)
      VPC(1,1)=SIN(THXP*RPD)*COS(PHXP*RPD)
      VPC(1,2)=SIN(THXP*RPD)*SIN(PHXP*RPD)
```

```
      VPC(1,3)=COS(THXP*RPD)
C!!!  INSURE VPC(1,N) IS PERPENDICULAR TO VPC(3,N)
      DZX=VPC(3,1)*VPC(1,1)+VPC(3,2)*VPC(1,2)+VPC(3,3)*VPC(1,3)
      IF(ABS(DZX).GT.0.1) THEN
       TYPE*, 'The coordinates are NOT orthogonal - Respecify.'
       goto 1234
      ELSE
       VPC(1,1)=VPC(1,1)-VPC(3,1)*DZX
       VPC(1,2)=VPC(1,2)-VPC(3,2)*DZX
       VPC(1,3)=VPC(1,3)-VPC(3,3)*DZX
       DOT=VPC(1,1)*VPC(1,1)+VPC(1,2)*VPC(1,2)+VPC(1,3)*VPC(1,3)
       DOT=SQRT(DOT)
       VPC(1,1)=VPC(1,1)/DOT
       VPC(1,2)=VPC(1,2)/DOT
       VPC(1,3)=VPC(1,3)/DOT
       VPC(2,1)=VPC(3,2)*VPC(1,3)-VPC(3,3)*VPC(1,2)
       VPC(2,2)=VPC(3,3)*VPC(1,1)-VPC(3,1)*VPC(1,3)
       VPC(2,3)=VPC(3,1)*VPC(1,2)-VPC(3,2)*VPC(1,1)
       WRITE(6,3931)
      END IF
C!
C! re-display the pattern cut system
C!
      ENTRY SHOW_PAT
       DO NI=1,3
        WRITE(6,4933) (NI,NJ,VPC(NI,NJ),NJ=1,3)
       END DO
4933   FORMAT(2H *,1X,3(2X,'VPC(',I1,',',I1,')=',F9.5),T79,1H*)
      RETURN

C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C!
C! This routine sets/displays a scale factor.
C!
      ENTRY SET_SCA
               WRITE (2,*) ' Please input a uniform scale factor:'
       READ  (1,*)  UNITF
       UNITS = UNITN * UNITF
C!
C! This entry displays the uniform scale factor.
C!
      ENTRY SHOW_SCA
       WRITE (2,FMT='('' The uniform scale factor is '',F10.8)') UNITF
      RETURN


C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C!
C! This routine sets the units for the program.
C!
C! IUNIT = Indicator of units used for input data.
C!
C!           1=METERS, 2=FEET, 3=INCHES
C!
      ENTRY          SET_UNI_METERS
         IUNIT = 1
         GOTO 2
      ENTRY          SET_UNI_FEET
         IUNIT = 2
         GOTO 2
      ENTRY          SET_UNI_INCHES
         IUNIT = 3
 2     UNITN = UNIT( IUNIT )
       UNITS = UNITN * UNITF
        RETURN
```

```
C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C!
C! This entry shows the current units.
C!
      ENTRY SHOW_UNI
         WRITE (2,FMT='('' The current units are '',A6)') LABEL( IUNIT )
      RETURN

C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C!
C! This routine sets the window.
C!
      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 ', phi*dpr,
     +                 ' to ',PH2*DPR
        TYPE*,'with a resolution of ',RESPH*DPR,' degrees/pixel.'
!
        TYPE*,'Please enter a new range for theta (lower,higher):'
        ACCEPT*, THET1,THET2
        THET1 = THET1 * RPD
        THET2 = THET2 * RPD
!
        TYPE*,'Please enter a new THETA resolution in degrees/pixel:'
        ACCEPT*, RESTH
        RESTH = RESTH * RPD
!
        TYPE*,'Please enter a new range for phi (lower,higher):'
        ACCEPT*, PH1,PH2
        PH1 = 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 + 0.5 ) + 1
          COLS   = INT( (THET2 - THET1) / RESTH + 0.5 ) + 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
C!
C! Show the window parameters
C!
       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 ', phi*dpr,
     +                 ' to ',PH2*DPR
        TYPE*,'with a resolution of ',RESPH*DPR,' degrees/pixel.'
       RETURN

C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
C!
C! This routine determines names of output files.  Here are the current
C! assignments.
C!
C!    Unit                 Meaning                          Default file n
C!     1               interactive input                       sys$input
C!     2               interactive output                      sys$output
C!     5               input processor input                   FILE.INP
C!     6               input processor (echo) output           FILE.LIS
C!     7               printable output file                   FILE.PRT
C!    10              "plot" data output file                  FILE.PLT
C!
      ENTRY   SET_OUT
       CALL CLI$GET_VALUE( 'P2', FILE )
C!
C! Only if /NOPLOT is specified, then discard all output written to unit
C! The user should always get ploftable output by default.
C!
      IF ( .NOT. CLI$PRESENT('PLOTTABLE') ) THEN
         OPEN( UNIT=10, FILE='_NL:', STATUS='OLD', FORM='UNFORMATTED' )
      ELSE
         OPEN( UNIT=10, FILE='.PLT', DEFAULTFILE=FILE, STATUS='NEW',
     +                            FORM='UNFORMATTED' )
      ENDIF
C!
C! If /PRINT is not specified, discard all output written to unit 7.
C! The user only wants to see the line printer if he asks for it.
C!
      OPEN( UNIT=7, STATUS='OLD', FILE='_NL:' )
      IF ( CLI$PRESENT( 'PRINTABLE' ) ) THEN
       OPEN( UNIT=7, DEFAULTFILE=FILE, STATUS='NEW', FILE='.PRT' )
      ENDIF
C!
C! If /NOECHO is specified, the input echo is discarded.
C! The user should get an echo file by default, just like a .PLT file.
C!
      IF ( .NOT. CLI$PRESENT( 'ECHOING' ) ) THEN
       OPEN( UNIT=6, FILE='_NL:', STATUS='OLD'  )
      ELSE
       OPEN( UNIT=6, FILE='.LIS', DEFAULTFILE=FILE, STATUS='NEW' )
      ENDIF
C!
C! Now retreive the full filenames for future reference.
C!
       INQUIRE        ( UNIT = 10,  NAME = OUTFIL )
       INQUIRE        ( UNIT =  7,  NAME = PRTFIL )
       INQUIRE        ( UNIT =  6,  NAME = LISFIL )
       SET_OUT = SUCCESS
C!
C! This routine displays the current output files.
C!
      ENTRY   SHOW_OUT
       TYPE *, 'Plotting file is: ', OUTFIL
       TYPE *, 'Printer file is: ', PRTFIL
       TYPE *, 'Input echo file: ', LISFIL
       SHOW_OUT = SUCCESS
      RETURN

C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C!
C! This routine stops the program.
C!
      ENTRY   EXIT_COMMAND
       CALL   EXIT
```

```
      RETURN

C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C!
C! This routine services online help requests.
C!
      ENTRY   HELP_COMMAND
       LIBRARY = ' '
       P1 = ' '
       CALL CLI$GET_VALUE( 'P1', P1 )
       CALL CLI$GET_VALUE( 'HELPLIB', LIBRARY )
       HELP_COMMAND = LBR$OUTPUT_HELP(
     +       PUT_OUTPUT,,                          ! Help output rout
     +       P1,                                   ! Help key descrip
     +       LIBRARY,,                             ! Help library nam
     +       GET_INPUT       )                     ! The prompting in
      RETURN

C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C!
C! This routine calls the routines which do the shadowing.
C!
      ENTRY   SHADOW_COMMAND
       TYPE*, 'Working...'                 ! Type an informational me
       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!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C!
C! This routine executes a DCL command as a subprocess.  Add a test for
C! better behavior with blank P1s.
C!
      ENTRY   DCL_COMMAND
       CALL CLI$GET_VALUE( 'P1', P1 )
      IF ( P1 .EQ. ' ' ) THEN
         DCL_COMMAND = LIB$SPAWN()
      ELSE
         DCL_COMMAND = LIB$SPAWN( P1 )
      ENDIF
      RETURN
C!
C! End of action routines.
C!
      END
```

10.2 Non-VAX/VMS Subroutines

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

MAIN PROGRAM (non-interactive)

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

```
      PROGRAM SHADOW
C!!!
C!!!  THIS COMPUTE CODE WAS WRITTEN AT THE OHIO STATE UNIVERSITY
C!!!  ELECTROSCIENCE LABORATORY.  ANY PROBLEMS OR COMMENTS
C!!!  CAN BE REFERRED TO:
C!!!
C!!!  RONALD J. MARHEFKA OR LASZLO A. TAKACS
C!!!  ELECTROSCIENCE LABORATORY
C!!!  1320 KINNEAR RD.
C!!!  COLUMBUS, OHIO 43212
C!!!  PONE: (614) 422-5752 OR 422-5848
C!!!
C!!!  THIS COMPUTER CODE CALCULATES SHADOWING OF AN ANTENNA
C!!!  USING THE NEC-BSC INPUTS NON-INTERACTIVELY.
C!!!  IT SHOULD BE USED IN PLACE OF INTERACTIVE MAIN PROGRAM
C!!!  WHEN THE SHADOW CODE IS USED ON NON VAX COMPUTERS.
C!!!
      INCLUDE 'SHACOM.FOR'
      PARAMETER (NSX=30)
      COMPLEX WS
      LOGICAL LRET
      COMMON/SORARY/WS(NSX),XSS(3,NSX),MSA(2,NSX),MSX,MSP,MSPP
C!!!  Initialize fill tags
C!!!  FILPNM and FILCNM < 0 is sequential tagging
C!!!  FILPNM or FILCNM > 0 that object is tagged with
C!!!    FILCHP or FILCHC
C!!!  FILPNM or FILCNM = 0 everything tagged with FILCHR
      FILPNM=0
      FILCNM=0
C!!!  Initialize fill characters
      FILCHP='P'
      FILCHC='C'
      FILCHR='X'
C!!!  Initialize return flag
              LRET=.TRUE.
C!!!  Initialize and read command information.
      CALL ABSCIN
100   CONTINUE
C!!!  Choose a source location from stored positions.
      DO 1200 MS=1,MSX
      DO 1000 N=1,3
1000  ANTENN(N)=XSS(N,MS)
C!!!  Initialize graphics information.
      CALL INITGF
C!!!  Calculate shadow of plates.
      CALL DOPLAS
C!!!  Calculate shadow of cylinders.
      CALL DOCYLS
C!!!  Write out maps to printer and plotter files.
      CALL WRTOUT
1200  CONTINUE
```

```
C!!! Read more command information.
      CALL ABSCRE
C!!! Return to execute next shadow map.
      IF(LRET) GO TO 100
      STOP
      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.

```
C----------------------------------------------------------------------
      SUBROUTINE      ABSCIN
C!!!
C!!!  THE NEC - BASIC SCATTERING CODE (NEC-BSC) WAS WRITTEN
C!!!  AT THE OHIO STATE UNIVERSITY ELECTROSCIENCE LABORATORY.
C!!!  ANY PROBLEMS OR COMMENTS CAN BE REFERRED TO:
C!!!
C!!!          RONALD J. MARHEFKA
C!!!          ELECTROSCIENCE LABORATORY
C!!!          1320 KINNEAR RD.
C!!!          COLUMBUS,OHIO 43212
C!!!          PHONE: (614) 422-5752
C!!!
C!!!  THIS IS A PORTION OF THE MAIN PROGRAM OF THE NEC-BSC.
C!!!  IT READS IN THE INPUT AND PASSES THE GEOMETRY INFORMATION
C!!!  TO THE SHADOW CALCULATION PART OF THIS OBSCURATION CODE.
C!!!  IT READS LOCATIONS OF SOURCES A NUMBER OF FINITE
C!!!  PLATES AND/OR A SET OF FINITE
C!!!  ELLIPTIC CYLINDERS AND CONE FRUSTUM SECTIONS.
C!!!  THE PLATES ARE DEFINED
C!!!  BY THEIR CORNER LOCATIONS.  THEY CAN BE PERFECTLY
C!!!  CONDUCTING, MULTI LAYERED DIELECTRIC SLABS, OR COATED
C!!!  METAL PLATES.  AN INFINITE GROUND PLANE CAN ALSO BE
C!!!  ADDED.  THE CYLINDERS ARE DEFINED BY THEIR ORIGIN,
C!!!  AXES DIRECTIONS, AND BY THE RADIUS ON THEIR MAJOR
C!!!  AND MINOR AXES AND THE ENDCAPS AND FRUSTUM RIMS ARE DEFINED BY
C!!!  THEIR POSITION ON THE CYLINDER AXIS AND THE ANGLE
C!!!  OF THEIR SURFACES WITH THE CYLINDER AXIS IN THE X-Z
C!!!  CYLINDER PLANE.  THE CYLINDERS MUST BE PERFECTLY
C!!!  CONDUCTING.  AS DIMENSIONED, IT CAN HANDLE 75 PLATES
C!!!  WITH A MAXIMUM OF 12 CORNERS PER PLATE, WITH 5 LAYERS
C!!!  OF DIELECTRIC AND 5 CYLINDERS, WITH 10 RIMS
C!!!  ALSO 30 TRANSMITTING
C!!!  ELEMENTS AND 30 RECEIVING ELEMENTS CAN BE INPUT.
C!!!  NOTE THAT THE LIMITS ON THE NUMBER OF PLATES,
C!!!  CORNERS, LAYERS, CYLINDERS, SOURCES, AND RECEIVERS
C!!!  ARE ONLY DUE TO THE SIZE OF THE ARRAYS.
C!!!  THE LINEAR DIMENSIONS ARE INPUT IN METERS UNLESS
C!!!  SPECIFIED OTHERWISE.  THE ANGULAR DIMENSIONS
C!!!  ARE IN DEGREES.
C!!!
C!!!  NOTE THAT COMMENTS ARE INDICATED IN DIFFERENT FORMS:
C!!!          C!!! IMPLIES EXPLANATION OF PROGRAM SECTION
C!!!          C$$$ IMPLIES DESCRIPTION OF INPUT DATA
C!!!          C=== IMPLIES COMMAND INPUT SECTION
C!!!          C--- IMPLIES BEGINNING OF SUBROUTINE
C!!!          C+++ IMPLIES SPECIFICATION OF MAXIMUM DIMENSIONS
C!!!          CXXX means lines were not needed for SHADOW program
C!!!          CFFF means lines were not implemented for current version
C!!!
```

```
C!!! NEC-BSC VERSION 2.6-1 ( UPDATED 8/16/85 )
C!!!
C!!!        MAJOR VERSION CHANGES ARE DENOTED BY THE FIRST DIGIT
C!!!        MINOR CHANGES IN CAPABILITY ARE DENOTED BY THE DECIMAL
C!!!        POINTS AND MINOR CHANGES THAT DO NOT NEED ADDED
C!!!        DOCUMENTATION ARE SHOWN AFTER THE DASH.
C!!!
C!!! NOTE ON VERSION 2.2
C!!!  1) THE PLATE - CYLINDER TERMS ARE NOT PRESENTLY INCLUDED.
C!!!  2) THE CYLINDER - CYLINDER INTERACTION TERMS WORK ONLY
C!!!     FOR PARALLEL CYLINDERS WITH THE PATTERN CUT
C!!!     PERPENDICULAR TO THE CYLINDER AXES.
C!!!
C!!! NOTE ON VERSION 2.3
C!!!  RANGE GATING HAS BEEN ADDED IN THE NEAR ZONE
C!!!
C!!! NOTE ON VERSION 2.4
C!!!  VOLUMETRIC PATTERN CAPABILITY HAS BEEN ADDED
C!!!
C!!! NOTE ON VERSION 2.5
C!!!  PARAMETER STATEMENTS FOR DIMENSIONS ADDED
C!!!  ARRAY INDICES CHANGED FOR MORE EFFICIENCY
C!!!
C!!! NOTE OF VERSION 2.6
C!!!  CONE FRUSTUM INPUT ADDED
C!!!
C+++
C+++ SPECIFICATION OF MAXIMUM DIMENSION SIZES
C+++
C+++ MAXIMUM DIMENSION FOR OBSERVATION POINTS
      PARAMETER (NOX=1801)
C+++ MAXIMUM DIMENSION FOR PLATE DIELECTRIC LAYERS
      PARAMETER (NLX=5)
C+++ MAXIMUM DIMENSION FOR SOURCES
      PARAMETER (NSX=30)
C+++ MAXIMUM DIMENSION FOR RECEIVERS
      PARAMETER (NRX=30)
C+++
      INCLUDE 'SHACOM.FOR'
      COMPLEX CJ,CPI4,WS,WR
      COMPLEX CI11,CI22,Z11,Z22
      CHARACTER*2 IT(40),IR(36),LABEL(3)*6
      CHARACTER RUNDAT*9,RUNTIM*8
      DIMENSION IMS(NSX),HS(NSX),HAWS(NSX),VXSS(3,3,NSX)
      DIMENSION IMR(NRX),HR(NRX),HAWR(NRX),VXRR(3,3,NRX)
      DIMENSION XRR(3,NRX),VXRP(3,3,NRX)
      DIMENSION XPC(3),VRT(3,3),TR(3)
      DIMENSION JMX(4),DR(3),DT(3),DP(3),RDR(3)
      DIMENSION XQR(3),XQ(3)
      LOGICAL LKJ(4,6),LFQG,LWARN,LSCAT,LPPREC
      LOGICAL LSOR,LOUT,LSRFC,LSURF,LSHD,LCYL,LPLA
      LOGICAL LIHD,LDEBUG,LTEST,LSLOPE,LCORNR,LDC
      LOGICAL LWRITE,LPLT,LGRND,LSMP,LRMP,LPRAD,LRANG,LCNPAT
      LOGICAL LNEAR,LRCVR,LRECT,LVOLP,LVPLT,LFARN
      COMMON/SORDAT/IM,H,HAW,FACTOR
      COMMON/SORARY/WS(NSX),XSS(3,NSX),MSA(2,NSX),MSX,MSP,MSPP
      COMMON/TEST/LDEBUG,LTEST,LWARN
      COMMON/SORINF/XS(3),VXS(3,3)
      COMMON/IMAINF/XI(3,NPX,NPX),VXI(3,3,NPX)
      COMMON/RECINF/WR(NRX),IMRP,HRPP,HAWRP,VXR(3,3),MR
      COMMON/RECARY/XRP(3,NRX),MRA(2,NRX),DRP(3),DTP(3),DPP(3)
      COMMON/LIMIT/SML,SMLR,SMLT,BIG
      COMMON/DIR/RD(3),D(3),LNEAR,LRCVR
      COMMON/WAVE/WK,WL
```

```
      COMMON/COMP/CJ,CPI4
      COMMON/FNANG/FNP(NEX,NPX)
      COMMON/LPLCY/LPLA,LCYL
      COMMON/GROUND/LGRND,MPXR
      COMMON/OUTPFZ/TPPD,PRAD,RANG,LCNPAT,LPRAD,LRANG
      COMMON/OUTPNZ/RXS,RXI,TYS,TYI,PZS,PZI,LRECT
      COMMON/OUTPNV/IVPN,IV,LVOLP
      COMMON/TRANDT/LSLAB(NPX),NSLAB(NPX),DSLAB(NLX,NPX)
     2,ERSLAB(NLX,NPX),TESLAB(NLX,NPX),URSLAB(NLX,NPX)
     3,TMSLAB(NLX,NPX)
      DATA LABEL/'METERS','FEET  ','INCHES'/
      DATA IT/'TO','PD','PG','SG','LP','PP','GP','XQ','RT','CG'
     2,'SM','RD','CM','CE','BP','UF','RM','UN','FR','NX'
     3,'EN','NP','NC','NG','NS','PR','US','PN','RG','NR'
     4,'SA','FM','RA','GR','VD','VN','VP','PF','VF','CC'/
C!!!  MAX. DIMENSION OF SOURCES,RECEIVERS,CYLINDERS,RIMS,PLATES,EDGES,
C!!!  LAYERS, AND OBSERVATION POINTS.
      MSDX=NSX
      MRDX=NRX
      MCDX=NCX
      NCDX=NNX
      MPDX=NPX
      MEDX=NEX
      MLDX=NLX
      MODX=NOX
C!!!  NOTE: IN SUB. RFPTCL THE VARIABLES IVD,PHOR,PHORP,AND VRO
C!!!          MUST BE DIMENSIONED 2*MPDX+1
C!!!
C!!!  SET TIME FLAGS TO ZERO
      IATIM=0
      IBTIM=0
      ICTIM=0
      GO TO 2701
 2700 CONTINUE
      WRITE(6,3006)
      WRITE(6,3005)
 2701 CONTINUE
C!!!  INITIALIZE DATA TO DEFAULT VALUES.
C$$$  TEST OUTPUT DEFAULT DATA TO:
      LDEBUG=.FALSE.
      LTEST=.FALSE.
      LOUT=.FALSE.
      LWARN=.TRUE.
      LSLOPE=.TRUE.
      LCORNR=.TRUE.
      LSOR=.FALSE.
      JMX(1)=1
      JMX(2)=6
      JMX(3)=5
      JMX(4)=4
      DO 2705 J=1,6
      DO 2705 K=1,4
      LKJ(K,J)=.FALSE.
      IF(J.LE.JMX(K)) LKJ(K,J)=.TRUE.
 2705 CONTINUE
      LKJ(3,4)=.FALSE.
      LKJ(3,5)=.FALSE.
C$$$  FAR ZONE RANGE DEFAULT DATA RD:
      LRANG=.FALSE.
      RANG=1.
C$$$  RANGE GATE DATA GR:
      RMIN=SMLT
      RMAX=BIG
C$$$  POWER RADIATED DEFAULT DATA PR:
```

```
      LPRAD=.FALSE.
      PRAD=0.
      IPRAD=1
C$$$  PATTERN DEFAULT DATA PD:, PN:, PF:, VD:, VF:, & VN:
      LVOLP=.TRUE.
      LFARN=.TRUE.
      LNEAR=.FALSE.
      LRECT=.FALSE.
      LCNPAT=.TRUE.
      TPPD=0.
      TPPV=2.
      TPPS=0.
      TPPI=2.
      THCZ=0.
      PHCZ=0.
      THCX=90.
      PHCX=0.
      VPC(1,1)=1.
      VPC(1,2)=0.
      VPC(1,3)=0.
      VPC(2,1)=0.
      VPC(2,2)=1.
      VPC(2,3)=0.
      VPC(1,3)=0.
      VPC(2,3)=0.
      VPC(3,3)=1.
      XPC(1)=0.
      XPC(2)=0.
      XPC(3)=0.
      RXS=1.
      RXI=0.
      TYS=0.
      TYI=2.
      PZS=0.
      PZI=2.
      IVPN=3
      NPN=181
      NPV=91
C$$$  BACK OR BISTATIC NEAR ZONE SCATTERING DEFAULT DATA BP:
      LSCAT=.FALSE.
C$$$  FREQUENCY DEFAULT DATA FR: & FM:
      FRQG=.2997925
      LFQG=.FALSE.
      FQGS=.2997925
      FQGI=0.
      NFQG=1
C$$$  PLATE DEFAULT DATA PG:
      LPLA=.FALSE.
      MPX=0
      MEP(1)=4
      LSLAB(1)=0
      XX(1,1,1)=1.
      XX(2,1,1)=1.
      XX(3,1,1)=0.
      XX(1,2,1)=-1.
      XX(2,2,1)=1.
      XX(3,2,1)=0.
      XX(1,3,1)=-1.
      XX(2,3,1)=-1.
      XX(3,3,1)=0.
      XX(1,4,1)=1.
      XX(2,4,1)=-1.
      XX(3,4,1)=0.
C$$$  GROUND PLANE DEFAULT DATA GP:
```

```
      LGRND=.FALSE.
      MPXR=MPX
C$$$  SOURCE DEFAULT DATA SG: ,SA: ,& SM:
      LSMP=.FALSE.
      MSX=0
      MSAT=0
      MSA(1,1)=0
      MSA(2,1)=0
      XSS(1,1)=0.
      XSS(2,1)=0.
      XSS(3,1)=0.
      IMS(1)=-1
      HS(1)=0.5
      HAWS(1)=0.
      THSZ=0.
      PHSZ=0.
      THSX=90.
      PHSX=0.
      VXSS(1,1,1)=1.
      VXSS(1,2,1)=0.
      VXSS(1,3,1)=0.
      VXSS(2,1,1)=0.
      VXSS(2,2,1)=1.
      VXSS(2,3,1)=0.
      VXSS(3,1,1)=0.
      VXSS(3,2,1)=0.
      VXSS(3,3,1)=1.
      WS(1)=(1.,0.)
C$$$  RECEIVER DEFAULT DATA RG: ,RA: ,& RM:
      LRCVR=.FALSE.
      LRMP=.FALSE.
      MRX=0
      MRAT=0
      MRA(1,1)=0
      MRA(2,1)=0
      XRR(1,1)=0.
      XRR(2,1)=0.
      XRR(3,1)=0.
      IMR(1)=-1
      HR(1)=0.5
      HAWR(1)=0.
      THRZ=0.
      PHRZ=0.
      THRX=90.
      PHRX=0.
      VXRR(1,1,1)=1.
      VXRR(1,2,1)=0.
      VXRR(1,3,1)=0.
      VXRR(2,1,1)=0.
      VXRR(2,2,1)=1.
      VXRR(2,3,1)=0.
      VXRR(3,1,1)=0.
      VXRR(3,2,1)=0.
      VXRR(3,3,1)=1.
      WR(1)=(0.,0.)
C$$$  LINE PRINTER DEFAULT DATA LP:
      LWRITE=.FALSE.
C$$$  PLOTTER DEFAULT DATA PP: & VP:
      LVPLT=.FALSE.
      LPLT=.FALSE.
      LPPREC=.FALSE.
      PPXL=0.
      PPYL=3.
      PPXB=0.
```

```
      PPXE=360.
      PPXS=30.
      PPYB=-40.
      PPYE=0.
      PPYS=10.
C$$$  ROTATE TRANSLATE DEFAULT DATA RT:
      THZP=0.
      PHZP=0.
      THXP=90.
      PHXP=0.
      TR(1)=0.
      TR(2)=0.
      TR(3)=0.
      VRT(1,1)=1.
      VRT(1,2)=0.
      VRT(1,3)=0.
      VRT(2,1)=0.
      VRT(2,2)=1.
      VRT(2,3)=0.
      VRT(3,1)=0.
      VRT(3,2)=0.
      VRT(3,3)=1.
C$$$  CYLINDER DEFAULT DATA CG: & CC:
      MDC=0
      LCYL=.FALSE.
      MCX=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:
      IUNIT=1
      UNITN=UNIT(IUNIT)
      UNITF=1.
      UNITS=UNITN*UNITF
      IUNST=0
      IUNSP=IUNST
      GO TO 2999
      ENTRY ABSCRE
 3000 CONTINUE
      WRITE(6,3006)
 3006 FORMAT(1X,1H*,76X,1H*)
      WRITE(6,3006)
      WRITE(6,3005)
 3005 FORMAT(1X,26(3H***))
C!!!  READ IN VARIOUS COMMAND OPTIONS.
 2999 READ(5,3001,END=3004) (IR(I),I=1,36)
```

```
3001  FORMAT(36A2)
      WRITE(6,3002)
3002  FORMAT(1H ,//////,1X,26(3H***))
      WRITE(6,3006)
      WRITE(6,3003) (IR(I),I=1,36)
3003  FORMAT(1X,1H*,2X,36A2,2X,1H*)
C!!!
C!!!  CHECK AGAINST STORED OPTIONS
C!!!
C$$$ CM: COMMENT CARD
      IF(IR(1).EQ.IT(13)) GO TO 3090
C$$$ CE: LAST COMMENT CARD
      IF(IR(1).EQ.IT(14)) GO TO 3000
      WRITE(6,3006)
      WRITE(6,3006)
C$$$ TO: TEST DATA GENERATION OPTION.
      IF(IR(1).EQ.IT(1)) GO TO 3100
C$$$ PD: FAR ZONE PATTERN INTEGER ANGLES
      IF(IR(1).EQ.IT(2)) GO TO 3200
C$$$ RD: FAR ZONE RANGE INPUT
      IF(IR(1).EQ.IT(12)) GO TO 3250
C$$$ PG: PLATE GEOMETRY INPUT
      IF(IR(1).EQ.IT(3)) GO TO 3300
C$$$ SG: SOURCE GEOMETRY INPUT
      IF(IR(1).EQ.IT(4)) GO TO 3400
C$$$ SM: SOURCE NEC OR AMP INPUT
      IF(IR(1).EQ.IT(11)) GO TO 3450
C$$$ LP: LINE PRINTER LISTING OF RESULTS
      IF(IR(1).EQ.IT(5)) GO TO 3500
C$$$ PP: PEN PLOT OF RESULTS
      IF(IR(1).EQ.IT(6)) GO TO 3600
C$$$ GP: INCLUDE INFINITE GROUND PLANE
      IF(IR(1).EQ.IT(7)) GO TO 3700
C$$$ XQ: EXECUTE PROGRAM
      IF(IR(1).EQ.IT(8)) GO TO 3800
C$$$ RT: TRANSLATE AND/OR ROTATE COORDINATES
      IF(IR(1).EQ.IT(9)) GO TO 3900
C$$$ CG: CYLINDER GEOMETRY INPUT
      IF(IR(1).EQ.IT(10)) GO TO 4000
C$$$ CC: CONE GEOMETRY INPUT
      IF(IR(1).EQ.IT(40)) GO TO 4000
C$$$ BP: BACK OR BISTATIC NEAR ZONE SCATTERING
      IF(IR(1).EQ.IT(15)) GO TO 5240
C$$$ UF: SCALE FACTOR FOR INPUT
      IF(IR(1).EQ.IT(16)) GO TO 4120
C$$$ UN: UNITS OF INPUT
      IF(IR(1).EQ.IT(18)) GO TO 4100
C$$$ FR: FREQUENCY
      IF(IR(1).EQ.IT(19)) GO TO 4200
C$$$ NX: NEXT PROBLEM
      IF(IR(1).EQ.IT(20)) GO TO 2700
C$$$ EN: END PROGRAM
      IF(IR(1).EQ.IT(21)) GO TO 997
C$$$ NP: NEXT SET OF PLATES
      IF(IR(1).EQ.IT(22)) GO TO 3350
C$$$ NC: NEXT SET OF CYLINDERS
      IF(IR(1).EQ.IT(23)) GO TO 4050
C$$$ NG: NO GROUND PLANE
      IF(IR(1).EQ.IT(24)) GO TO 3750
C$$$ NS: NEXT SET OF SOURCES
      IF(IR(1).EQ.IT(25)) GO TO 3490
C$$$ PR: POWER RADIATED INPUT
      IF(IR(1).EQ.IT(26)) GO TO 3440
C$$$ US: UNITS OF HS AND HAWS IN SG: , SA: , RG: ,& RA:
```

```
      IF(IR(1).EQ.IT(27)) GO TO 4110
C$$$ PN: NEAR ZONE PATTERN DESIRED
      IF(IR(1).EQ.IT(28)) GO TO 3260
C$$$ RG: RECEIVER GEOMETRY INPUT
      IF(IR(1).EQ.IT(29)) GO TO 4400
C$$$ RM: RECEIVER NEC OR AMP INPUT
      IF(IR(1).EQ.IT(17)) GO TO 4450
C$$$ NR: NEXT SET OF RECEIVERS
      IF(IR(1).EQ.IT(30)) GO TO 3495
C$$$ SA: SOURCE ARRAY GEOMETRY INPUT
      IF(IR(1).EQ.IT(31)) GO TO 3810
C$$$ FM: MULTIPLE FREQUENCY INPUT
      IF(IR(1).EQ.IT(32)) GO TO 4250
C$$$ RA: RECEIVER ARRAY GEOMETRY INPUT
      IF(IR(1).EQ.IT(33)) GO TO 4810
C$$$ GR: RANGE GATE INPUT
      IF(IR(1).EQ.IT(34)) GO TO 5260
C$$$ VD: FAR ZONE VOLUMETRIC PATTERN INTEGER ANGLES
      IF(IR(1).EQ.IT(35)) GO TO 3210
C$$$ VN: NEAR ZONE VOLUMETRIC PATTERN
      IF(IR(1).EQ.IT(36)) GO TO 3270
C$$$ VP: VOLUMETRIC DUMP OF RESULTS FOR PLOTTING
      IF(IR(1).EQ.IT(37)) GO TO 3650
C$$$ PF: FAR ZONE NON INTEGER ANGLES
      IF(IR(1).EQ.IT(38)) GO TO 3220
C$$$ VF: FAR ZONE VOLUMETRIC PATTERN NON INTEGER ANGLES
      IF(IR(1).EQ.IT(39)) GO TO 3230
C$$$
      WRITE(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$$$
C$$$  IR(I)=CM: OR CE: FOLLOWED BY AN ALPHANUMERIC STRING OF
C$$$  CHARACTERS.  THE CM: COMMAND IMPLIES THAT THERE WILL BE
C$$$  ANOTHER COMMENT CARD FOLLOWING IT.  THE LAST COMMENT CARD
C$$$  MUST HAVE THE CE: COMMAND ON IT.  IF THERE IS ONLY ONE
C$$$  COMMENT CARD THE CE: COMMAND SHOULD BE USED.
C$$$
      READ(5,3001) (IR(I),I=1,36)
      WRITE(6,3003) (IR(I),I=1,36)
      IF(IR(1).EQ.IT(14)) GO TO 3000
      IF(IR(1).EQ.IT(13)) GO TO 3090
      WRITE(6,3091)
3091  FORMAT(' ***   PROGRAM ABORTS!!!   CE: COMMAND MUST BE',
     2' USED TO END COMMENTS.   ***')
      STOP
C======
3100  CONTINUE
C===  TO:     COMMAND         ======
C$$$
C$$$  LDEBUG=DEBUG DATA OUTPUT ON LINE PRINTER(TRUE OR FALSE)
C$$$
C$$$  LTEST=TEST DATA TO INSURE PROGRAM OPERATION(TRUE OR FALSE)
C$$$
C$$$  LOUT=OUTPUT MAIN PROGRAM DATA ON LINE PRINTER(TRUE OR FALSE)
C$$$
C$$$  LWARN=WARNING DATA OUTPUT ON LINE PRINTER(TRUE OR FALSE)
C$$$
      READ(5,*) LDEBUG,LTEST,LOUT,LWARN
      WRITE(6,3101) LDEBUG,LTEST,LOUT,LWARN
```

```
3101  FORMAT(2H *,5X,'LDEBUG= ',L3,5X,'LTEST= ',L3,5X,'LOUT=',L3
     2,5X,'LWARN=',L3,T79,1H*)
      WRITE(6,3006)
C$$$
C$$$  LSLOPE=SLOPE DIFFRACTED FIELD DESIRED (T OR F)
C$$$
C$$$  LCORNR=CORNER DIFFRACTED FIELD DESIRED (T OR F)
C$$$
C$$$  LSOR=ANTENNA SHADOW ALONE(TRUE OR FALSE)
C$$$
      READ(5,*) LSLOPE,LCORNR,LSOR
      WRITE(6,3102) LSLOPE,LCORNR,LSOR
3102  FORMAT(2H *,5X,'LSLOPE= ',L3,5X,'LCORNR= ',L3,5X,'LSOR= ',L3,
     2T79,1H*)
      WRITE(6,3006)
      IF(LSOR) WRITE(6,3402)
3402  FORMAT(2H *,5X,'SOURCE SHADOW ALONE IS COMPUTED!!!',T79,1H*)
      IF(LSOR) WRITE(6,3006)
C$$$
C$$$  K=1,J=OPTION TO RUN DIRECT RAY TERM:
C$$$ 1=DIRECT FIELD
C$$$  NOTE: NORMALLY LKJ(1,1)=.TRUE. THIS COMPUTES THE INCIDENT FIELD.
C$$$
C$$$  K=2,J=OPTION TO RUN VARIOUS RAY TERMS FOR PLATES:
C$$$ 1=SINGLE REFLECTED FIELD
C$$$ 2=DOUBLE REFLECTED FIELD
C$$$ 3=SINGLE DIFFRACTED FIELD
C$$$ 4=REFLECTED/DIFFRACTED FIELD
C$$$ 5=DIFFRACTED/REFLECTED FIELD
C$$$ 6=DOUBLE DIFFRACTION INDENTIFICATION
C$$$  NOTE: NORMALLY LKJ(2,1 TO 6)=.TRUE. THIS COMPUTES ALL FIELD
C$$$  VALUES INCLUDING IDENTIFING DOUBLE DIFFRACTION PROBLEM AREAS
C$$$  FOR A CONVEX OR CONCAVE PLATE STRUCTURE.
C$$$
C$$$  K=3,J=OPTION TO RUN VARIOUS RAY TERMS FOR CYLINDER:
C$$$ 1=REFLECTED,TRANSITION,AND CREEPING WAVE FIELDS
C$$$ 2=SINGLE REFLECTED FIELDS FROM ENDCAPS
C$$$ 3=SINGLE DIFFACTED FIELDS FROM ENDCAP RIMS
C$$$ 4=REFLECTED-SCATTERED FIELDS FROM TWO PARALLEL CYLINDERS
C$$$ 5=DIFFRACTED-SCATTERED FIELDS FROM TWO PARALLEL CYLINDERS
C$$$  NOTE: NORMALLY LKJ(3,1 TO 5)=.TRUE. THIS COMPUTES ALL FIELD
C$$$  VALUES FOR A FINITE ELLIPTIC CYLINDER.
C$$$
C$$$  K=4,J=OPTION TO RUN VARIOUS RAY TERMS FOR
C$$$  PLATE-CYLINDER INTERACTIONS:
C$$$ 1=FIELDS REFLECTED FROM THE PLATES THEN REFLECTED OR
C$$$  DIFFRACTED FROM THE CYLINDER
C$$$ 2=FIELDS REFLECTED OR DIFFRACTED FROM THE CYLINDER THEN
C$$$  REFLECTED FROM THE PLATES
C$$$ 3=FIELDS REFLECTED FROM THE CYLINDER THEN DIFFRACTED
C$$$  FROM THE PLATES
C$$$ 4=FIELDS DIFFRACTED FROM THE PLATES THEN REFLECTED
C$$$  FROM THE CYLINDER
C$$$  NOTE: NORMALLY LKJ(4,1 TO 4)=.TRUE. THIS COMPUTES ALL FIELD
C$$$  VALUES THAT INTERACT BETWEEN THE PLATES AND CYLINDERS.
C$$$
      DO 3104 K=1,4
      JK=JMX(K)
      READ(5,*) (LKJ(K,J),J=1,JK)
3104  WRITE(6,3103) K,(LKJ(K,J),J=1,JK)
3103  FORMAT(2H *,T79,1H*,T8,'LKJ(',I1,',J)= ',6L2)
      GO TO 3000
C=====
4100  CONTINUE
```

```
C===  UN:     COMMAND          ======
C$$$
C$$$  IUNIT=INDICATOR OF UNITS USED FOR INPUT DATA.
C$$$          1=METERS
C$$$          2=FEET
C$$$          3=INCHES
C$$$
      READ(5,*) IUNIT
      UNITN=UNIT(IUNIT)
      UNITS=UNITN*UNITF
      WRITE(6,4101) LABEL(IUNIT)
4101  FORMAT(2H *,5X,'ALL THE LINEAR DIMENSIONS BELOW ARE'
     2,' ASSUMED TO BE IN ',A6,T79,1H*)
      GO TO 3000
C======
4120  CONTINUE
C===  UF:     COMMAND ======
C$$$
C$$$  UNITF = SCALE FACTOR FOR GEOMETRY
C$$$
      READ(5,*) UNITF
      UNITS=UNITN*UNITF
      WRITE(6,4121) UNITF
4121  FORMAT(2H *,5X,'ALL THE LINEAR DIMENSIONS BELOW ARE SCALED BY'
     2,' A FACTOR OF ',F12.5,T79,1H*)
      GO TO 3000
C======
4110  CONTINUE
C===  US:     COMMAND          ======
C$$$
C$$$  IUNST=INDICATOR OF UNITS USED FOR HS AND HAWS IN THE
C$$$  SG: COMMAND.
C$$$          0=WAVELENGTHS
C$$$          1=METERS
C$$$          2=FEET
C$$$          3=INCHES
C$$$
C$$$  NOTE: IF ONE SOURCE IS SPECIFIED IN WAVELENGTHS, THEY ALL
C$$$           MUST BE IN WAVELENGTHS.
      READ(5,*) IUNST
      IF(MSX.EQ.0) GO TO 4112
      IF(IUNST.EQ.0.AND.IUNSP.EQ.0) GO TO 4112
      IF(IUNST.NE.0.AND.IUNSP.NE.0) GO TO 4112
      WRITE(6,4111)
4111  FORMAT(' *** PROGRAM ABORTS IN SOURCE UNITS. ALL UNITS NOT'
     2,' SPECIFIED IN WAVELENGTHS!!! ***')
      STOP
4112  CONTINUE
      IF(IUNST.EQ.0) GO TO 4114
      WRITE(6,4113) LABEL(IUNST)
4113  FORMAT(2H *,5X,'THE SOURCE LENGTH HS AND WIDTH HAWS ARE'
     2,' ASSUMED TO BE IN ',A6,T79,1H*)
      GO TO 4116
4114  WRITE(6,4115)
4115  FORMAT(2H *,5X,'THE SOURCE LENGTH HS AND WIDTH HAWS ARE'
     2,' ASSUMED TO BE IN WAVELENGTHS',T79,1H*)
4116  IUNSP=IUNST
      GO TO 3000
C======
4200  CONTINUE
C===  FR:     COMMAND          ======
C$$$
C$$$  FRQG=FREQUENCY IN GIGAHERTZ.
C$$$
```

```
      LFQG=.FALSE.
      NFQG=1
      READ(5,*) FRQG
      WL=.2997925/FRQG
      WRITE(6,4201) FRQG
 4201 FORMAT(2H *,5X,'FREQUENCY= ',F7.3,' GIGAHERTZ',T79,1H*)
      WRITE(6,3006)
      WRITE(6,4202) WL
 4202 FORMAT(2H *,5X,'WAVELENGTH= ',F10.6,' METERS',T79,1H*)
      GO TO 3000
C======
 4250 CONTINUE
C=== FM:     COMMAND          ======
C$$$
C$$$ NFQG=NUMBER OF FREQUENCIES DESIRED
C$$$
C$$$ FQGS=STARTING FREQUENCY IN GIGAHERTZ
C$$$
C$$$ FQGI=INCREMENTAL FREQUENCY CHANGE IN GIGAHERTZ
C$$$
C$$$ NOTE:   THE SOURCE LENGTH AND WIDTH MUST NOT BE SPECIFIED
C$$$         IN WAVELENGTHS.  ALSO ONLY ONE PATTERN LOCATION
C$$$         CAN BE SPECIFIED.
C$$$
      LFQG=.TRUE.
      READ(5,*) NFQG,FQGS,FQGI
      WRITE(6,4251) NFQG
 4251 FORMAT(2H *,5X,I3,' FREQUENCIES ARE SPECIFIED',T79,1H*)
      IF(NFQG.GT.MODX) WRITE(6,3266) NFQG
      IF(NFQG.GT.MODX) STOP
      WRITE(6,3006)
      WRITE(6,4252) FQGS,FQGI
 4252 FORMAT(2H *,5X,'STARTING FREQ.= ',F10.5,' IN STEPS OF ',F10.5
     2,' GHZ.',T79,1H*)
C!!! CALCULATE MID-FREQUENCY
      FRQG=FQGS+0.5*FQGI*(NFQG-1)
      WL=.2997925/FRQG
      GO TO 3000
C======
 3230 CONTINUE
C=== VF:     COMMAND ======
C$$$
C$$$ FAR ZONE VOLUMETRIC PATTERN NON INTEGER ANGLES
C$$$
      LVOLP=.TRUE.
      LFARN=.TRUE.
      GO TO 3211
C======
 3210 CONTINUE
C=== VD:     COMMAND ======
C$$$
C$$$ FAR ZONE VOLUMETRIC PATTERN INTEGER ANGLES
C$$$
      LVOLP=.TRUE.
      LFARN=.FALSE.
      GO TO 3211
C======
 3220 CONTINUE

C=== PF:     COMMAND ======
C$$$
C$$$ FAR ZONE PATTERN NON INTEGER ANGLES
C$$$
      LVOLP=.FALSE.
      LFARN=.TRUE.
```

```
      GO TO 3211
C======
 3200 CONTINUE
C===  PD:     COMMAND        ======
C$$$
C$$$  FAR ZONE PATTERN INTEGER ANGLES
C$$$
C$$$  THCZ,PHCZ=ORIENTATION OF THE Z AXIS RELATIVE TO THE
C$$$  FIXED COORDINATE SYSTEM.
C$$$
C$$$  THCX,PHCX=ORIENTATION OF THE X AXIS RELATIVE TO THE
C$$$  FIXED COORDINATE SYSTEM
C$$$
      LVOLP=.FALSE.
 3211 LNEAR=.FALSE.
      READ(5,*) THCZ,PHCZ,THCX,PHCX
      VPC(3,1)=SIN(THCZ*RPD)*COS(PHCZ*RPD)
      VPC(3,2)=SIN(THCZ*RPD)*SIN(PHCZ*RPD)
      VPC(3,3)=COS(THCZ*RPD)
      VPC(1,1)=SIN(THCX*RPD)*COS(PHCX*RPD)
      VPC(1,2)=SIN(THCX*RPD)*SIN(PHCX*RPD)
      VPC(1,3)=COS(THCX*RPD)
C!!!  INSURE VPC(1,N) IS PERPENDICULAR TO VPX(3,N)
      DZX=VPC(3,1)*VPC(1,1)+VPC(3,2)*VPC(1,2)+VPC(3,3)*VPC(1,3)
      IF(ABS(DZX).GT.0.1) WRITE(6,3201)
 3201 FORMAT(' *** PROGRAM ABORTS IN PATTERN CUT SECTION.'
     2,' THE COORDINATES ARE NOT ORTHOGONAL!!! ***')
      IF(ABS(DZX).GT.0.1) STOP
      VPC(1,1)=VPC(1,1)-VPC(3,1)*DZX
      VPC(1,2)=VPC(1,2)-VPC(3,2)*DZX
      VPC(1,3)=VPC(1,3)-VPC(3,3)*DZX
      DOT=VPC(1,1)*VPC(1,1)+VPC(1,2)*VPC(1,2)+VPC(1,3)*VPC(1,3)
      DOT=SQRT(DOT)
      VPC(1,1)=VPC(1,1)/DOT
      VPC(1,2)=VPC(1,2)/DOT
      VPC(1,3)=VPC(1,3)/DOT
      VPC(2,1)=VPC(3,2)*VPC(1,3)-VPC(3,3)*VPC(1,2)
      VPC(2,2)=VPC(3,3)*VPC(1,1)-VPC(3,1)*VPC(1,3)
      VPC(2,3)=VPC(3,1)*VPC(1,2)-VPC(3,2)*VPC(1,1)
      WRITE(6,3202)
 3202 FORMAT(2H *,5X,'THE PATTERN AXES ARE AS FOLLOWS:',T79,1H*)
      DO 3204 NI=1,3
      WRITE(6,3006)
 3204 WRITE(6,3205) (NI,NJ,VPC(NI,NJ),NJ=1,3)
 3205 FORMAT(2H *,1X,3(2X,'VPC(',I1,',',I1,')=',F9.5),T79,1H*)
      DO 3203 N=1,3
 3203 XPC(N)=0.
C$$$
C$$$  LCNPAT=IS PATTERN CONIC CUT(T OR F)?
C$$$  T=THETA CUT(CONIC CUT)
C$$$  F=PHI CUT(PHI CONSTANT)
C$$$
C$$$  TPPD=PATTERN ANGLE THAT IS CONSTANT
C$$$  IF LCNPAT=T: TPPD=THP CONSTANT
C$$$  IF LCNPAT=F: TPPD=PHP CONSTANT
C$$$
      IF(LVOLP) GO TO 3212
      TPPV=0.
      NPV=1
      READ(5,*) LCNPAT,TPPD
      WRITE(6,3006)
      IF(.NOT.LCNPAT) WRITE(6,3206) TPPD
 3206 FORMAT(2H *,5X,'THETA IS BEING VARIED WITH PHI= ',F10.5
     2,T79,1H*)
```

```
      IF(LCNPAT) WRITE(6,3207) TPPD
3207  FORMAT(2H *,5X,'PHI IS BEING VARIED WITH THETA= ',F10.5
     2,T79,1H*)
      WRITE(6,3006)
      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$$$
3212  READ(5,*) LCNPAT,TPPD,TPPV,NPV
      WRITE(6,3006)
      IF(LCNPAT) WRITE(6,3213)
3213  FORMAT(2H *,5X,'FOR THETA ANGLE:',T79,1H*)
      IF(.NOT.LCNPAT) WRITE(6,3214)
3214  FORMAT(2H *,5X,'FOR PHI ANGLE:',T79,1H*)
      WRITE(6,3215) TPPD,TPPV,NPV
3215  FORMAT(2H *,5X,'START= ',F10.5,' STEP= 'F10.5,' NUMBER= ',I4
     2,T79,1H*)
      WRITE(6,3006)
      IF(LCNPAT) WRITE(6,3214)
      IF(.NOT.LCNPAT) WRITE(6,3213)
3216  CONTINUE
      IF(LFARN) GO TO 3217
C$$$
C$$$  IB,IE,IS=BEGIN,END,STEP
C$$$
      READ(5,*) IB,IE,IS
      IF(IB.LT.0) IB=0
      IF(IE.GT.360) IE=360
      IF(IS.LE.0) IS=1
      TPPS=IB
      TPPI=IS
      NPN=(IE-IB)/IS+1
      WRITE(6,3208) IB,IE,IS
3208  FORMAT(2H *,5X,'THE RANGE OF PATTERN ANGLE INDICES FOR THIS'
     2,' RUN ARE: ',I3,2(',',I3),T79,1H*)
      GO TO 3218
3217  CONTINUE
C$$$
C$$$  TPPS=START OF PATTERN
C$$$  TPPI=PATTERN INCREMENT
C$$$  NPN=NUMBER OF PATTERN POINTS
C$$$
      READ(5,*) TPPS,TPPI,NPN
      WRITE(6,3215) TPPS,TPPI,NPN
3218  CONTINUE
      RXS=1.
      RXI=0.
      TYS=TPPD
      TYI=TPPV
      PZS=TPPS
      PZI=TPPI
      IVPN=3
      IF(LCNPAT) GO TO 3209
      TYS=TPPS
      TYI=TPPI
      PZS=TPPD
      PZI=TPPV
      IVPN=-3
3209  CONTINUE
      GO TO 3000
C======
3250  CONTINUE
```

```
C===  RD:     COMMAND          ======
C$$$
C$$$  RANGS=FAR FIELD RANGE DISTANCE
C$$$
C$$$  NOTE IF RANGS IS GREATER THAN OR EQUAL TO 1.E30
C$$$  THAN LRANG WILL BE SET FALSE
C$$$
      LRANG=.TRUE.
      READ(5,*) RANGS
      IF(RANGS.GT.9.9E29) GO TO 3252
      RANG=UNITS*RANGS
      WRITE(6,3251) RANGS,LABEL(IUNIT),RANG
3251  FORMAT(2H *,5X,'THE FAR FIELD RANGE SPECIFIED IS ',E12.6,
     2' IN ',A6,T79,1H*,/2H *,5X,'THE RANGE SPECIFIED IN METERS'
     3,' IS ',E12.6,T79,1H*)
      GO TO 3000
3252  CONTINUE
      LRANG=.FALSE.
      RANG=1.
      WRITE(6,3253)
3253  FORMAT(2H *,5X,'NO FAR FIELD RANGE SPECIFIED.',T79,1H*)
      GO TO 3000
C======
3270  CONTINUE
C===  VN:     COMMAND ======
C$$$
C$$$  NEAR ZONE VOLUMETRIC PATTERN
C$$$
      LVOLP=.TRUE.
      GO TO 3271
C======
3260  CONTINUE
C===  PN:     COMMAND          ======
C$$$
C$$$  XPC(N)=XYZ LOCATION OF THE NEAR ZONE PATTERN ORIGIN
C$$$
      LVOLP=.FALSE.
3271  LNEAR=.TRUE.
      READ(5,*) (XPC(N),N=1,3)
      WRITE(6,3254) LABEL(IUNIT),(XPC(N),N=1,3)
3254  FORMAT(2H *,1X,'PATTERN ORIGIN IN ',A6,':  XPC(1)=',F8.3
     2,' XPC(2)=',F8.3,' XPC(3)=',F8.3,T79,1H*)
      WRITE(6,3006)
      DO 3263 N=1,3
3263  XPC(N)=UNITS*XPC(N)
      IF(IUNIT.NE.1) WRITE(6,3254) LABEL(1),(XPC(N),N=1,3)
      IF(IUNIT.NE.1) WRITE(6,3006)
      WRITE(6,3006)
C$$$
C$$$  THCZ,PHCZ=ORIENTATION OF THE Z-AXIS OF THE PATTERN AXES
C$$$  RELATIVE TO THE FIXED COORDINATE SYSTEM
C$$$
C$$$  THCX,PHCX=ORIENTATION OF THE X-AXIS OF THE PATTERN AXES
C$$$  RELATIVE TO THE FIXED COORDINATE SYSTEM
C$$$
      READ(5,*) THCZ,PHCZ,THCX,PHCX
      VPC(3,1)=SIN(THCZ*RPD)*COS(PHCZ*RPD)
      VPC(3,2)=SIN(THCZ*RPD)*SIN(PHCZ*RPD)
      VPC(3,3)=COS(THCZ*RPD)
      VPC(1,1)=SIN(THCX*RPD)*COS(PHCX*RPD)
      VPC(1,2)=SIN(THCX*RPD)*SIN(PHCX*RPD)
      VPC(1,3)=COS(THCX*RPD)
C!!!  INSURE VPC(1,N) IS PERPENDICULAR TO VPC(3,N)
      DZX=VPC(3,1)*VPC(1,1)+VPC(3,2)*VPC(1,2)+VPC(3,3)*VPC(1,3)
```

```
      IF(ABS(DZX).GT.0.1) WRITE(6,3201)
      IF(ABS(DZX).GT.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 3264 NI=1,3
      WRITE(6,3006)
 3264 WRITE(6,3205) (NI,NJ,VPC(NI,NJ),NJ=1,3)
      WRITE(6,3006)
      WRITE(6,3006)
C$$$
C$$$  LRECT=F, SPHERICAL PATTERN CUT
C$$$  LRECT=T, LINEAR PATTERN CUT
C$$$
C$$$  RXS,TYS,PZS=STARTING LOCATION OF PATTERN
C$$$          LRECT=F: RADIAL,THETA,PHI
C$$$          LRECT=T: X,Y,Z
C$$$
C$$$  RXI,TYI,PZI=SIZE OF INCREMENTAL STEPS
C$$$          LRECT=F: RADIAL,THETA,PHI
C$$$          LRECT=T: X,Y,Z
C$$$
      READ(5,*) LRECT
      READ(5,*) RXS,TYS,PZS
      READ(5,*) RXI,TYI,PZI
      IF(LRECT) WRITE(6,3261) RXS,TYS,PZS,LABEL(IUNIT)
 3261 FORMAT(2H *,2X,'STARTING XYZ=',F10.5,2(',',F10.5),1X,A6
     2,T79,1H*)
      IF(LRECT) WRITE(6,3262) RXI,TYI,PZI,LABEL(IUNIT)
 3262 FORMAT(2H *,2X,'STEP XYZ=',F10.5,2(',',F10.5),1X,A6,T79,1H*)
      IF(.NOT.LRECT) WRITE(6,3267) RXS,TYS,PZS,LABEL(IUNIT)
 3267 FORMAT(2H *,2X,'STARTING R,THETA,PHI=',F10.5
     2,2(',',F10.5),1X,A6,' AND DEG.',T79,1H*)
      IF(.NOT.LRECT) WRITE(6,3268) RXI,TYI,PZI,LABEL(IUNIT)
 3268 FORMAT(2H *,2X,'STEP R,THETA,PHI=',F10.5,2(',',F10.5),1X,A6
     2,' AND DEG.',T79,1H*)
      WRITE(6,3006)
      RXS=UNITS*RXS
      RXI=UNITS*RXI
      IF(.NOT.LRECT) GO TO 3265
      TYS=UNITS*TYS
      PZS=UNITS*PZS
      TYI=UNITS*TYI
      PZI=UNITS*PZI
 3265 CONTINUE
      IF(LRECT.AND.IUNIT.NE.1) WRITE(6,3261) RXS,TYS,PZS,LABEL(1)
      IF(LRECT.AND.IUNIT.NE.1) WRITE(6,3262) RXI,TYI,PZI,LABEL(1)
      IF(.NOT.LRECT.AND.IUNIT.NE.1) WRITE(6,3267) RXS,TYS,PZS,LABEL(1)
      IF(.NOT.LRECT.AND.IUNIT.NE.1) WRITE(6,3268) RXI,TYI,PZI,LABEL(1)
      IF(.NOT.LRECT.AND.IUNIT.NE.1) WRITE(6,3006)
      IF(LVOLP) GO TO 3272
C$$$
C$$$  NPN=NUMBER OF PATTERN POINTS
C$$$
      READ(5,*) NPN
```

```
      WRITE(6,3269) NPN
 3269 FORMAT(2H *,5X,'NUMBER OF PATTERN POINTS= ',I4,T79,1H*)
      IVPN=3
      IF(ABS(PZI).LT.SMLR) IVPN=-3
      IF(LRECT) IVPN=0
      GO TO 3276
C$$$
C$$$  IVPN=1 FOR R-THETA OR X-Y VARYING
C$$$  NPV=NUMBER OF R OR X AND NPN=NUMBER OF THETA OR Y
C$$$
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$$$
C$$$  IVPN=3 FOR THETA-PHI OR Y-Z VARYING
C$$$  NPV=NUMBER OF THETA OR Y AND NPN=NUMBER OF PHI OR Z
C$$$
C$$$  IF IVPN IS LESS THAN ZERO THE ORDER IS REVERSED
C$$$  I.E. IVPN=-1 FOR THETA-R OR Y-X VARYING
C$$$
 3272 READ(5,*) IVPN,NPV,NPN
      IF(IVPN.EQ.1) WRITE(6,3273) NPV,NPN
      IF(IVPN.EQ.-1) WRITE(6,3273) NPN,NPV
 3273 FORMAT(2H *,5X,'NUMBER OF POINTS FOR R OR X= ',I4
     2,' AND THETA OR Y= ',I4)
      IF(IVPN.EQ.2) WRITE(6,3274) NPV,NPN
      IF(IVPN.EQ.-2) WRITE(6,3274) NPN,NPV
 3274 FORMAT(2H *,5X,'NUMBER OF POINTS FOR R OR X= ',I4
     2,' AND PHI OR Z= ',I4)
      IF(IVPN.EQ.3) WRITE(6,3275) NPV,NPN
      IF(IVPN.EQ.-3) WRITE(6,3275) NPN,NPV
 3275 FORMAT(2H *,5X,'NUMBER OF POINTS FOR THETA OR Y= ',I4
     2,' AND PHI OR Z= ',I4)
 3276 CONTINUE
      IF(NPN.GT.MODX) WRITE(6,3266) NPN
 3266 FORMAT(' ***** NUMBER OF POINTS= ',I3,' PROGRAM ABORTS'
     2,'PATTERN STORAGE DIMENSION IS EXCEEDED *****')
      IF(NPN.GT.MODX) STOP
      GO TO 3000
C======
 5240 CONTINUE
C===  BP:     COMMAND ======
C$$$
C$$$  BACK OR BISTATIC NEAR ZONE SCATTERING
C$$$
C$$$  THE SG:, RG:, AND PN: COMMANDS MUST BE SPECIFIED
C$$$  TO USE THIS OPTION.
      LSCAT=.TRUE.
      GO TO 3000
C======
 5260 CONTINUE
C===  GR:     COMMAND ======
C$$$
C$$$  RANGE GATE INPUT
C$$$
C$$$  RMIN=THE MINIMUM DISTANCE FROM TRANSMITTER TO RECEIVER
C$$$  RMAX=THE MAXIMUM DISTANCE FROM TRANSMITTER TO RECEIVER
C$$$
C$$$  THE PN: COMMAND MUST BE USED
C$$$
      READ(5,*) RMIN,RMAX
      WRITE(6,5261) RMIN,RMAX,LABEL(IUNIT)
 5261 FORMAT(2H *,2X,'RMIN= ',F10.5,'RMAX= ',F10.5,' IN ',A6,T79,1H*)
      RMIN=UNITS*RMIN
      RMAX=UNITS*RMAX
```

```
      WRITE(6,5261) RMIN,RMAX,LABEL(1)
      GO TO 3000
C======
3300  CONTINUE
C===  PG:      COMMAND          ======
C$$$
C$$$  PLATE GEOMETRY INPUT
C$$$
      LPLA=.TRUE.
      MPX=MPX+1
      IF(MPX.GT.MPDX) WRITE(6,901) MPX
901   FORMAT(' *****   NUMBER OF PLATES= ',I3,'  PROGRAM ABORTS',
     2' SINCE MAX. PLATE DIMENSION IS EXCEEDED.   *****')
      IF(MPX.GT.MPDX) STOP
      WRITE(6,3301) MPX
3301  FORMAT(2H *,5X,'THIS IS PLATE NO. ',I3,' IN THIS ',
     2'SIMULATION.',T79,1H*)
      MP=MPX
      WRITE(6,3006)
      WRITE(6,3006)
      WRITE(6,3006)
C$$$
C$$$  MEP(MP)=NUMBER OF CORNERS ON THE MP-TH PLATE.
C$$$

C$$$  LSLAB= 1  IMPLIES TRANSPARENT THIN DIELECTRIC SLAB
C$$$       = 0  IMPLIES METAL PLATE, AND
C$$$       =-2  IMPLIES DIELECTRIC COVERED PLATE ON BOTH SIDES
C$$$       =-4  IMPLIES DIELECTRIC COVERED PLATE ON SIDE OF NORMAL
C$$$
C$$$  NOTE:  IF DIELECTRIC COVERED, ONE MUST READ DIELECTRIC DATA.
C$$$
C$$$
      READ(5,*) MEP(MP), LSLAB(MP)
      IF(LSLAB(MP).EQ.0) WRITE(6,3392)
3392  FORMAT(2H *,5X,'METAL PLATE USED IN THIS SIMULATION',T79,1H*)
      IF(LSLAB(MP).EQ.1) WRITE(6,3393)
3393  FORMAT(2H *,5X,'TRANSPARENT THIN DIELECTRIC LAYER USED IN THIS',
     2'SIMULATION',T79,1H*)
      IF(LSLAB(MP).EQ.-2) WRITE(6,3394)
3394  FORMAT(2H *,5X,'DIELECTRIC COVERED PLATE USED IN THIS',
     2' SIMULATION',T79,1H*)
      WRITE(6,3006)
      IF(LSLAB(MP).EQ.0) GO TO 3313
C$$$
C$$$  NSLAB(MP)=NUMBER OF DIELECTRIC LAYERS ON THE MP PLATE
C$$$
      READ(5,*) NSLAB(MP)
      NSS=NSLAB(MP)
      IF(NSS.GT.MLDX) STOP
      WRITE(6,3391)
3391  FORMAT(2H *,13X,'THICKNESS',2X,'DIELECTRIC',3X,'LOSS',4X,
     2'PERMITIVITY',3X,'LOSS',T79,1H*,/,
     32H *,5X,'LAYER#',2X,'IN METERS',3X,'CONSTANT',3X,'TANGENT',
     44X,'CONSTANT',3X,'TANGENT',T79,1H*,/,
     52H *,5X,'------',2X,'---------',2X,'----------',2X,'-------',
     62X,'-----------',2X,'-------',T79,1H*)
C$$$
C$$$  DSLAB(NS,MP)=THICKNESS OF NS LAYER
C$$$
C$$$  ERSLAB(NS,MP)=RELATIVE DIELECTRIC CONSTANT OF THE NS LAYER
C$$$
C$$$  TESLAB(NS,MP)=DIELECTRIC LOSS TANGENT OF THE NS LAYER
C$$$
C$$$  URSLAB(NS,MP)=RELATIVE PERMEABILITY CONSTANT OF THE NS LAYER
```

```
C$$$
C$$$  TMSLAB(NS,MP)=PERMEABILITY LOSS TANGENT OF THE NS LAYER
C$$$
      DO 3312 NS=1,NSS
      READ(5,*) DSLAB(NS,MP),ERSLAB(NS,MP),TESLAB(NS,MP),
     2URSLAB(NS,MP),TMSLAB(NS,MP)
      DSLAB(NS,MP)=DSLAB(NS,MP)*UNITS
3312  WRITE(6,3399) NS,DSLAB(NS,MP),ERSLAB(NS,MP),TESLAB(NS,MP),
     2URSLAB(NS,MP),TMSLAB(NS,MP)
3399  FORMAT(2H *,6X,I3,4X,F9.4,2X,F10.4,2X,F7.4,2X,F11.4,2X,
     2F7.4,T79,1H*)
      WRITE(6,3006)
      WRITE(6,3006)
      WRITE(6,3006)
3313  MEX=MEP(MP)
      IF(MEX.GT.MEDX) WRITE(6,903) MP,MEX
903   FORMAT(' *****  PLATE #',I3,'  HAS ',I3,' EDGES.',
     2' PROGRAM ABORTS SINCE MAX. EDGE DIMENSION IS EXCEEDED.'
     3,' *****')
      IF(MEX.GT.MEDX) STOP
      DO 5 ME=1,MEX
C$$$
C$$$  XX(N,ME,MP)=X,Y,Z COMPONENTS OF CORNER #ME OF PLATE #MP.
C$$$  N=1(X),N=2(Y),N=3(Z). INPUT CORNER DATA AS FOLLOWS:
C$$$ 1.,1.,0.
C$$$  -1.,1.,0.
C$$$  -1.,-1.,0.
C$$$ 1.,-1.,0.
C$$$  THIS IS THE INPUT FOR A 2 METER SQUARE PLATE.
C$$$  NOTE THAT IF THERE IS MORE THAN ONE PLATE, THEN THE CORNER
C$$$  DATA FOR EACH PLATE WOULD FOLLOW SEQUENTIALLY.
C$$$
      READ(5,*) (XX(N,ME,MP),N=1,3)
5     CONTINUE
      WRITE(6,3302) LABEL(IUNIT)
3302  FORMAT(2H *,2X,'PLATE#',2X,'CORNER#',3X,'INPUT LOCATION IN ',
     2A6,4X,'ACTUAL LOCATION IN METERS',T79,1H*)
      WRITE(6,3303)
3303  FORMAT(2H *,2X,'------',2X,'-------'
     2,2(2X,2('-------------')),T79,1H*)
      DO 3304 ME=1,MEX
      WRITE(6,3006)
      DO 3310 N=1,3
3310  XQ(N)=XX(N,ME,MP)
      DO 3311 N=1,3
3311  XX(N,ME,MP)=UNITS*(XQ(1)*VRT(1,N)+XQ(2)*VRT(2,N)
     2+XQ(3)*VRT(3,N))+TR(N)
      WRITE(6,3305) MP,ME,(XQ(N),N=1,3),(XX(N,ME,MP),N=1,3)
3305  FORMAT(2H *,4X,I3,6X,I2,2X,2(2X,F8.3,2(',',F8.3)),T79,1H*)
3304  CONTINUE
      GO TO 3000
C======
3350  CONTINUE
C===  NP:     COMMAND          ======
C$$$
C$$$  INITIALIZE PLATE DATA.
C$$$
      LPLA=.FALSE.
      MPX=0
      WRITE(6,3351)
3351  FORMAT(2H *,5X,' THE PLATE DATA IS INITIALIZED. ',T79,1H*/
     2,2H *,5X,' NO PLATES ARE PRESENTLY IN THE PROBLEM. ',T79,1H*)
      GO TO 3000
C======
```

```
 3400 CONTINUE
C===  SG:     COMMAND         ======
C$$$
C$$$  MSX=NUMBER OF ANTENNA ELEMENTS.
C$$$
      LSMP=.FALSE.
      MSX=MSX+1
      MSXAT=MSAT+MSX
      IF(MSXAT.GT.MSDX) WRITE(6,904) MSXAT
  904 FORMAT(' *****   NUMBER OF SOURCES= ',I3,'  PROGRAM',
     2' ABORTS SINCE MAX. SOURCE DIMENSION IS EXCEEDED.   *****')
      IF(MSXAT.GT.MSDX) STOP
      WRITE(6,3401) MSX
 3401 FORMAT(2H *,5X,'THIS IS SOURCE NO. ',I3,' IN THIS',
     2' COMPUTATION.',T79,1H*)
      WRITE(6,3006)
      WRITE(6,3006)
C$$$
C$$$  XSS(N,MS)=XYZ LOCATION OF MS-TH ANTENNA ELEMENT.
C$$$
C$$$  IMS(MS)=TYPE OF LINEAR ANTENNA
C$$$          .LT.0: ELECTRIC LINEAR ELEMENT
C$$$          .GT.0: MAGNETIC LINEAR ELEMENT
C$$$  ABS(IMS)=1: UNIFORM CURRENT DISTRIBUTION
C$$$          =2: STANDARD DIPOLE CURRENT DISTRIBUTION
C$$$          =3: CAVITY BACKED SLOT CURRENT DISTRIBUTION (TE01)
C$$$
C$$$  HAWS(MS)=APERTURE WIDTH IN WAVELENGTHS   (NOTE: IF
C$$$  HAWS(MS) IS LESS THAN .1 LAMBDA, SOURCE IS
C$$$  CONSIDERED TO BE DIPOLE SOURCE
C$$$  HS(MS)=LENGTH OF LINEAR ELEMENT IN WAVELENGTHS
C$$$
C$$$  THSZ,PHSZ=ORIENTATION ANGLES USED TO DEFINE LINEAR
C$$$  ELEMENT AXIS.
C$$$
C$$$  THSX,PHSX=ORIENTATION ANGLES USED TO DEFINE APERTURE
C$$$  PLANE OR DIPOLE X-AXIS.
C$$$
C$$$  WMS,WPS=MAGNITUDE AND PHASE OF EXCITATION OF
C$$$  MS-TH ELEMENT.
C$$$
      MS=MSX
      MSA(1,MS)=0
      MSA(2,MS)=0
      READ(5,*) (XSS(N,MS),N=1,3)
      READ(5,*) THSZ,PHSZ,THSX,PHSX
      READ(5,*) IMS(MS),HS(MS),HAWS(MS)
      READ(5,*) WMS,WPS
      IF(IMS(MS).LT.0) WRITE(6,3411) IMS(MS)
 3411 FORMAT(2H *,5X,'THIS IS AN ELECTRIC SOURCE OF TYPE ',I3,T79,1H*)
      IF(IMS(MS).GE.0) WRITE(6,3412) IMS(MS)
 3412 FORMAT(2H *,5X,'THIS IS A MAGNETIC SOURCE OF TYPE ',I3,T79,1H*)
      WRITE(6,3006)
      IF(IUNST.EQ.0) GO TO 3414
      UNSTS=UNIT(IUNST)

      WRITE(6,3413) HS(MS),HAWS(MS),LABEL(IUNST)
 3413 FORMAT(2H *,5X,'SOURCE LENGTH=',F10.5,' AND WIDTH='
     2,F10.5,1X,A6,T79,1H*)
      HS(MS)=UNSTS*UNITF*HS(MS)
      HAWS(MS)=UNSTS*UNITF*HAWS(MS)
      IF(IUNST.NE.1) WRITE(6,3006)
      IF(IUNST.NE.1) WRITE(6,3413) HS(MS),HAWS(MS),LABEL(1)
      GO TO 3416
 3414 WRITE(6,3415) HS(MS),HAWS(MS)
```

```
 3415 FORMAT(2H *,5X,'SOURCE LENGTH=',F10.5,' AND WIDTH='
     2,F10.5,' WAVELENGTHS',T79,1H*)
 3416 WRITE(6,3006)
      WS(MS)=WMS*CEXP(CJ*WPS*RPD)
      WRITE(6,3417) WMS,WPS
 3417 FORMAT(2H *,5X,'THE SOURCE WEIGHT HAS MAGNITUDE='
     2,F10.5,' AND PHASE=',F10.5,T79,1H*)
      WRITE(6,3006)
      WRITE(6,3006)
      WRITE(6,3421) LABEL(IUNIT)
 3421 FORMAT(2H *,T6,'SOURCE#',T17,'INPUT LOCATION IN ',A6,T46,
     2'ACTUAL LOCATION IN METERS',T79,1H*)
      WRITE(6,3422)
 3422 FORMAT(2H *,T6,7('-'),T16,27('-'),T45,27('-'),
     2T79,1H*)
      WRITE(6,3006)
      DO 3424 N=1,3
 3424 XQ(N)=XSS(N,MS)
      DO 3425 N=1,3
 3425 XSS(N,MS)=UNITS*(XQ(1)*VRT(1,N)+XQ(2)*VRT(2,N)
     2+XQ(3)*VRT(3,N))+TR(N)
      WRITE(6,3426) MS,(XQ(N),N=1,3),(XSS(N,MS),N=1,3)
 3426 FORMAT(2H *,T8,I3,T15,F8.3,2(',',F8.3),T44,F8.3,2(',',F8.3)
     2,T79,1H*)
      TQR=THSZ*RPD
      PQR=PHSZ*RPD
      XQ(1)=SIN(TQR)*COS(PQR)
      XQ(2)=SIN(TQR)*SIN(PQR)
      XQ(3)=COS(TQR)
      DO 3431 N=1,3
 3431 VXSS(3,N,MS)=XQ(1)*VRT(1,N)+XQ(2)*VRT(2,N)+XQ(3)*VRT(3,N)
      TQR=THSX*RPD
      PQR=PHSX*RPD
      XQ(1)=SIN(TQR)*COS(PQR)
      XQ(2)=SIN(TQR)*SIN(PQR)
      XQ(3)=COS(TQR)
      DO 3432 N=1,3
 3432 VXSS(1,N,MS)=XQ(1)*VRT(1,N)+XQ(2)*VRT(2,N)+XQ(3)*VRT(3,N)
      DZX=VXSS(1,1,MS)*VXSS(3,1,MS)+VXSS(1,2,MS)*VXSS(3,2,MS)
     2+VXSS(1,3,MS)*VXSS(3,3,MS)
      IF(ABS(DZX).GT.0.1) WRITE(6,3436)
 3436 FORMAT(' *** PROGRAM ABORTS IN SOURCE SECTION IN THAT THE',
     2' COORDINATES ARE NOT ORTHOGONAL !!! ***')
      IF(ABS(DZX).GT.0.1) STOP
      VXSS(1,1,MS)=VXSS(1,1,MS)-VXSS(3,1,MS)*DZX
      VXSS(1,2,MS)=VXSS(1,2,MS)-VXSS(3,2,MS)*DZX
      VXSS(1,3,MS)=VXSS(1,3,MS)-VXSS(3,3,MS)*DZX
      DOT=VXSS(1,1,MS)*VXSS(1,1,MS)+VXSS(1,2,MS)*VXSS(1,2,MS)
     2+VXSS(1,3,MS)*VXSS(1,3,MS)
      DOT=SQRT(DOT)
      VXSS(1,1,MS)=VXSS(1,1,MS)/DOT
      VXSS(1,2,MS)=VXSS(1,2,MS)/DOT
      VXSS(1,3,MS)=VXSS(1,3,MS)/DOT
      VXSS(2,1,MS)=VXSS(3,2,MS)*VXSS(1,3,MS)-VXSS(3,3,MS)*VXSS(1,2,MS)
      VXSS(2,2,MS)=VXSS(3,3,MS)*VXSS(1,1,MS)-VXSS(3,1,MS)*VXSS(1,3,MS)
      VXSS(2,3,MS)=VXSS(3,1,MS)*VXSS(1,2,MS)-VXSS(3,2,MS)*VXSS(1,1,MS)
      WRITE(6,3006)
      WRITE(6,3006)
      WRITE(6,3437)
 3437 FORMAT(2H *,5X,'THE FOLLOWING SOURCE ALIGNMENT IS USED:'
     2,T79,1H*)
      DO 3433 NI=1,3
      WRITE(6,3006)
 3433 WRITE(6,3434) (NI,NJ,MS,VXSS(NI,NJ,MS),NJ=1,3)
```

```
3434  FORMAT(2H *,1X,3(2X,'VXSS(',I1,',',I1,',',I2,')=',F9.5)
     2,T79,1H*)
      GO TO 3000
C======
3810  CONTINUE
C===  SA:     COMMAND         ======
C$$$
C$$$  MSX=NUMBER OF ANTENNA ARRAY GROUPINGS.
C$$$
C$$$  MSAX=NUMBER OF ELEMENTS PER GROUPING.
C$$$
      LSMP=.FALSE.
      MSX=MSX+1
      READ(5,*) MSAX
      MSAT=MSAT+MSAX
      MSXAT=MSAT+MSX
      IF(MSXAT.GT.MSDX) WRITE(6,904) MSXAT
      IF(MSXAT.GT.MSDX) STOP
      WRITE(6,3805) MSX,MSAX
3805  FORMAT(2H *,5X,'THIS IS SOURCE NO. ',I3,' IN THIS',
     2' COMPUTATION.',T79,1H*/2H *,5X,'THERE ARE ',
     3I3,' SOURCES ARRAYED TOGETHER.',T79,1H*)
      WRITE(6,3006)
      WRITE(6,3006)
C$$$
C$$$  XSS(N,MA)=XYZ LOCATION OF MA-TH ANTENNA ELEMENT.
C$$$
C$$$  XSS(N,MS)=XYZ LOCATION OF MS-TH WEIGHTED CENTER OF THE
C$$$          ARRAY GROUPING.
C$$$
C$$$  THE ARRAY ELEMENTS ARE ASSUMED TO HAVE THE SAME LENGTH,
C$$$  WIDTH, AND ORIENTATION.  ALSO, THEY ARE ASSUMED TO BE
C$$$  EITHER ALL MOUNTED AND OR ALL OFF A PLATE.
C$$$  IMS(MS)=TYPE OF LINEAR ANTENNA
C$$$          .LT.0: ELECTRIC LINEAR ELEMENT
C$$$          .GT.0: MAGNETIC LINEAR ELEMENT
C$$$  ABS(IMS)=1: UNIFORM CURRENT DISTRIBUTION
C$$$          =2: STANDARD DIPOLE CURRENT DISTRIBUTION
C$$$          =3: CAVITY BACKED SLOT CURRENT DISTRIBUTION (TE01)
C$$$
C$$$  HAWS(MS)=APERTURE WIDTH IN WAVELENGTHS   (NOTE: IF
C$$$  HAWS(MS) IS LESS THAN .1 LAMBDA, SOURCE IS
C$$$  CONSIDERED TO BE DIPOLE SOURCE
C$$$  HS(MS)=LENGTH OF LINEAR ELEMENT IN WAVELENGTHS
C$$$
C$$$  THSZ,PHSZ=ORIENTATION ANGLES USED TO DEFINE LINEAR
C$$$  ELEMENT AXIS.
C$$$
C$$$  THSX,PHSX=ORIENTATION ANGLES USED TO DEFINE APERTURE
C$$$  PLANE OR DIPOLE X-AXIS.
C$$$
C$$$  WMS,WPS=MAGNITUDE AND PHASE OF EXCITATION OF
C$$$  MA-TH ELEMENT.
C$$$
      MS=MSX
      MAI=MSDX-MSAT+1
      MAF=MAI+MSAX-1
      MSA(1,MS)=MAI
      MSA(2,MS)=MAF
      DO 3841 MA=MAI,MAF
3841  READ(5,*) (XSS(N,MA),N=1,3)
      READ(5,*) THSZ,PHSZ,THSX,PHSX
      READ(5,*) IMS(MS),HS(MS),HAWS(MS)
      IF(IMS(MS).LT.0) WRITE(6,3411) IMS(MS)
```

```
      IF(IMS(MS).GE.0) WRITE(6,3412) IMS(MS)
      WRITE(6,3006)
      IF(IUNST.EQ.0) GO TO 3814
      UNSTS=UNIT(IUNST)
      WRITE(6,3413) HS(MS),HAWS(MS),LABEL(IUNST)
      HS(MS)=UNSTS*UNITF*HS(MS)
      HAWS(MS)=UNSTS*UNITF*HAWS(MS)
      IF(IUNST.NE.1) WRITE(6,3006)
      IF(IUNST.NE.1) WRITE(6,3413) HS(MS),HAWS(MS),LABEL(1)
      GO TO 3816
 3814 WRITE(6,3415) HS(MS),HAWS(MS)
 3816 WRITE(6,3006)
      WS(MS)=(1.,0.)
      WMSA=0.
      XSAX=0.
      XSAY=0.
      XSAZ=0.
      DO 3843 MA=MAI,MAF
      READ(5,*) WMS,WPS
      WRITE(6,3817) MA,WMS,WPS
 3817 FORMAT(2H *,5X,'SOURCE ',I3,' HAS MAGNITUDE='
     2,F10.5,' AND PHASE=',F10.5,T79,1H*)
      WS(MA)=WMS*CEXP(CJ*WPS*RPD)
      WMSA=WMSA+WMS
      XSAX=XSAX+WMS*XSS(1,MA)
      XSAY=XSAY+WMS*XSS(2,MA)
 3843 XSAZ=XSAZ+WMS*XSS(3,MA)
      XSS(1,MS)=XSAX/WMSA
      XSS(2,MS)=XSAY/WMSA
      XSS(3,MS)=XSAZ/WMSA
      WRITE(6,3006)
      WRITE(6,3006)
      WRITE(6,3421) LABEL(IUNIT)
      WRITE(6,3422)
      WRITE(6,3006)
      DO 3824 N=1,3
 3824 XQ(N)=XSS(N,MS)
      DO 3825 N=1,3
 3825 XSS(N,MS)=UNITS*(XQ(1)*VRT(1,N)+XQ(2)*VRT(2,N)
     2+XQ(3)*VRT(3,N))+TR(N)
      WRITE(6,3426) MS,(XQ(N),N=1,3),(XSS(N,MS),N=1,3)
      DO 3829 MA=MAI,MAF
      DO 3827 N=1,3
 3827 XQ(N)=XSS(N,MA)
      DO 3828 N=1,3
 3828 XSS(N,MA)=UNITS*(XQ(1)*VRT(1,N)+XQ(2)*VRT(2,N)
     2+XQ(3)*VRT(3,N))+TR(N)
 3829 WRITE(6,3426) MA,(XQ(N),N=1,3),(XSS(N,MA),N=1,3)
      TQR=THSZ*RPD
      PQR=PHSZ*RPD
      XQ(1)=SIN(TQR)*COS(PQR)
      XQ(2)=SIN(TQR)*SIN(PQR)
      XQ(3)=COS(TQR)
      DO 3831 N=1,3
 3831 VXSS(3,N,MS)=XQ(1)*VRT(1,N)+XQ(2)*VRT(2,N)+XQ(3)*VRT(3,N)
      TQR=THSX*RPD
      PQR=PHSX*RPD
      XQ(1)=SIN(TQR)*COS(PQR)
      XQ(2)=SIN(TQR)*SIN(PQR)
      XQ(3)=COS(TQR)
      DO 3832 N=1,3
 3832 VXSS(1,N,MS)=XQ(1)*VRT(1,N)+XQ(2)*VRT(2,N)+XQ(3)*VRT(3,N)
      DZX=VXSS(1,1,MS)*VXSS(3,1,MS)+VXSS(1,2,MS)*VXSS(3,2,MS)
     2+VXSS(1,3,MS)*VXSS(3,3,MS)
```

```
      IF(ABS(DZX).GT.0.1) WRITE(6,3436)
      IF(ABS(DZX).GT.0.1) STOP
      VXSS(1,1,MS)=VXSS(1,1,MS)-VXSS(3,1,MS)*DZX
      VXSS(1,2,MS)=VXSS(1,2,MS)-VXSS(3,2,MS)*DZX
      VXSS(1,3,MS)=VXSS(1,3,MS)-VXSS(3,3,MS)*DZX
      DOT=VXSS(1,1,MS)*VXSS(1,1,MS)+VXSS(1,2,MS)*VXSS(1,2,MS)
     2+VXSS(1,3,MS)*VXSS(1,3,MS)
      DOT=SQRT(DOT)
      VXSS(1,1,MS)=VXSS(1,1,MS)/DOT
      VXSS(1,2,MS)=VXSS(1,2,MS)/DOT
      VXSS(1,3,MS)=VXSS(1,3,MS)/DOT
      VXSS(2,1,MS)=VXSS(3,2,MS)*VXSS(1,3,MS)-VXSS(3,3,MS)*VXSS(1,2,MS)
      VXSS(2,2,MS)=VXSS(3,3,MS)*VXSS(1,1,MS)-VXSS(3,1,MS)*VXSS(1,3,MS)
      VXSS(2,3,MS)=VXSS(3,1,MS)*VXSS(1,2,MS)-VXSS(3,2,MS)*VXSS(1,1,MS)
      WRITE(6,3006)
      WRITE(6,3006)
      WRITE(6,3437)
      DO 3833 NI=1,3
      WRITE(6,3006)
 3833 WRITE(6,3434) (NI,NJ,MS,VXSS(NI,NJ,MS),NJ=1,3)
      GO TO 3000
C======
 3440 CONTINUE
C===  PR:     COMMAND        ======
C$$$
C$$$  IPRAD= 1 =NORMALIZATION FOR FAR ZONE AS FOLLOWS
C$$$
C$$$  PRAD=TOTAL POWER RADIATED IN WATTS.
C$$$
C$$$  PRAD CAN ALSO BE SPECIFIED AS THE POWER INPUT IN WATTS.
C$$$
C$$$  NOTE IF PRAD IS LESS THAN OR EQUAL TO 1.E-30
C$$$  THAN LPRAD WILL BE SET FALSE
C$$$
      LPRAD=.TRUE.
      READ(5,*) IPRAD
      IF(IPRAD.GT.4) STOP
      GO TO (3444,3445,3446,3447),IPRAD
 3444 READ(5,*) PRAD
      IF(PRAD.LT.1.1E-30) GO TO 3442
      WRITE(6,3441) PRAD
 3441 FORMAT(2H *,5X,'TOTAL POWER RADIATED IN WATTS= ',E12.6
     2,T79,1H*)
      GO TO 3000
 3442 CONTINUE
      LPRAD=.FALSE.
      PRAD=0.
      WRITE(6,3443)
 3443 FORMAT(2H *,5X,'NO POWER RADIATED IS SPECIFIED',T79,1H*)
      GO TO 3000
 3445 CONTINUE
C$$$
C$$$  IPRAD = 2 =MUTUAL IMPEDANDCE CALCULATION Z12 = Z21
C$$$
C$$$  CI11 = SOURCE TERMINAL CURRENT (REAL AND IMAGINARY)
C$$$  CI22 = RECEIVER TERMINAL CURRENT (REAL AND IMAGINARY)
C$$$
      READ(5,*) CI11,CI22
      WRITE(6,4445) CI11,CI22
 4445 FORMAT(2H *,5X,' SOURCE TERMINAL CURRENT= ',2E12.6,T79,1H*/
     2,2H *,5X,'RECEIVER TERMINAL CURRENT= ',2E12.6,T79,1H*)
      GO TO 3000
 3446 CONTINUE
C$$$
```

```
C$$$  IPRAD = 3 =COUPLING VIA THE REACTION THEORY
C$$$  THIS GIVES A MODIFIED FRII'S TRANSMISSION TYPE RESULT
C$$$
C$$$  PRAD = POWER RADIATED BY THE SOURCE
C$$$  PRADR = POWER RADIATED BY THE RECEIVER AS IF IT WERE A SOURCE
C$$$
      READ(5,*) PRAD,PRADR
      WRITE(6,4447) PRAD,PRADR
4447  FORMAT(2H *,5X,' SOURCE POWER RADIATED= ',E12.6,T79,1H*/
     2,2H *,5X,'RECEIVER POWER RADIATED= ',E12.6,T79,1H*)
      GO TO 3000
3447  CONTINUE
C$$$
C$$$  IPRAD= 4 =COUPLING BY THE LINVILLE METHOD
C$$$
C$$$  CI11 = SOURCE TERMINAL CURRENT (REAL AND IMAGINARY)
C$$$  CI22 = RECEIVER TERMINAL CURRENT (REAL AND IMAGINARY)
C$$$
C$$$  Z11 = SOURCE TERMINAL IMPEDANCE (REAL AND IMAGINARY)
C$$$  Z22 = RECEIVER TERMINAL IMPEDANCE (REAL AND IMAGINARY)
C$$$
      READ(5,*) CI11,CI22
      WRITE(6,4445) CI11,CI22
      READ(5,*) Z11,Z22
      WRITE(6,4446) Z11,Z22
4446  FORMAT(2H *,5X,' SOURCE TERMINAL IMPEDANCE= ',2E12.6,T79,1H*/
     2,2H *,5X,'RECEIVER TERMINAL IMPEDANCE= ',2E12.6,T79,1H*)
      GO TO 3000
C======
4400  CONTINUE
C===  RG:     COMMAND         ======
C$$$
C$$$  MRX=NUMBER OF ANTENNA ELEMENTS.
C$$$
      LRCVR=.TRUE.
      LRMP=.FALSE.
      MRX=MRX+1
      MRXAT=MRAT+MRX
      IF(MRXAT.GT.MRDX) WRITE(6,4404) MRXAT
4404  FORMAT(' *****   NUMBER OF RECEIVERS= ',I3,'  PROGRAM',
     2' ABORTS SINCE MAX. RECEIVER DIMENSION IS EXCEEDED.   *****')
      IF(MRXAT.GT.MRDX) STOP
      WRITE(6,4401) MRX
4401  FORMAT(2H *,5X,'THIS IS RECEIVER NO. ',I3,' IN THIS',
     2' COMPUTATION.',T79,1H*)
      WRITE(6,3006)
      WRITE(6,3006)
C$$$
C$$$  XRR(N,MR)=XYZ LOCATION OF MR-TH ANTENNA ELEMENT.
C$$$
C$$$  IMR(MR)=TYPE OF LINEAR ANTENNA
C$$$          .LT.0: ELECTRIC LINEAR ELEMENT
C$$$          .GT.0: MAGNETIC LINEAR ELEMENT
C$$$  ABS(IMR)=1: UNIFORM CURRENT DISTRIBUTION
C$$$          =2: STANDARD DIPOLE CURRENT DISTRIBUTION
C$$$          =3: CAVITY BACKED SLOT CURRENT DISTRIBUTION (TE01)
C$$$
C$$$  HAWR(MR)=APERTURE WIDTH IN WAVELENGTHS   (NOTE: IF
C$$$  HAWR(MR) IS LESS THAN .1 LAMBDA, RECEIVER IS
C$$$  CONSIDERED TO BE DIPOLE RECEIVER
C$$$  HR(MR)=LENGTH OF LINEAR ELEMENT IN WAVELENGTHS
C$$$
C$$$  THRZ,PHRZ=ORIENTATION ANGLES USED TO DEFINE LINEAR
C$$$  ELEMENT AXIS.
```

```
C$$$
C$$$  THRX,PHRX=ORIENTATION ANGLES USED TO DEFINE APERTURE
C$$$  PLANE OR DIPOLE X-AXIS.
C$$$
C$$$  WMR,WPR=MAGNITUDE AND PHASE OF EXCITATION OF
C$$$  MR-TH ELEMENT.
C$$$
      MR=MRX
      MRA(1,MR)=0
      MRA(2,MR)=0
      READ(5,*) (XRR(N,MR),N=1,3)
      READ(5,*) THRZ,PHRZ,THRX,PHRX
      READ(5,*) IMR(MR),HR(MR),HAWR(MR)
      READ(5,*) WMR,WPR
      IF(IMR(MR).LT.0) WRITE(6,4411) IMR(MR)
 4411 FORMAT(2H *,5X,'THIS IS AN ELECTRIC RECEIVER OF TYPE ',I3
     2,T79,1H*)
      IF(IMR(MR).GE.0) WRITE(6,4412) IMR(MR)
 4412 FORMAT(2H *,5X,'THIS IS A MAGNETIC RECEIVER OF TYPE ',I3
     2,T79,1H*)
      WRITE(6,3006)

      IF(IUNST.EQ.0) GO TO 4414
      UNSTS=UNIT(IUNST)
      WRITE(6,4413) HR(MR),HAWR(MR),LABEL(IUNST)
 4413 FORMAT(2H *,5X,'RECEIVER LENGTH=',F10.5,' AND WIDTH='
     2,F10.5,1X,A6,T79,1H*)
      HR(MR)=UNSTS*UNITF*HR(MR)
      HAWR(MR)=UNSTS*UNITF*HAWR(MR)
      IF(IUNST.NE.1) WRITE(6,3006)
      IF(IUNST.NE.1) WRITE(6,4413) HR(MR),HAWR(MR),LABEL(1)
      GO TO 4416
 4414 WRITE(6,4415) HR(MR),HAWR(MR)
 4415 FORMAT(2H *,5X,'RECEIVER LENGTH=',F10.5,' AND WIDTH='
     2,F10.5,' WAVELENGTHS',T79,1H*)
 4416 WRITE(6,3006)
      WR(MR)=WMR*CEXP(CJ*WPR*RPD)
      WRITE(6,4417) WMR,WPR
 4417 FORMAT(2H *,5X,'THE RECEIVER WEIGHT HAS MAGNITUDE='
     2,F10.5,' AND PHASE=',F10.5,T79,1H*)
      WRITE(6,3006)
      WRITE(6,3006)
      WRITE(6,4421) LABEL(IUNIT)
 4421 FORMAT(2H *,T6,'RECEIVER#',T17,'INPUT LOCATION IN ',A6,T46,
     2'ACTUAL LOCATION IN METERS',T79,1H*)
      WRITE(6,4422)
 4422 FORMAT(2H *,T6,7('-'),T16,27('-'),T45,27('-'),
     2T79,1H*)
      WRITE(6,3006)
      DO 4424 N=1,3
 4424 XQ(N)=XRR(N,MR)
      DO 4425 N=1,3
 4425 XRR(N,MR)=UNITS*(XQ(1)*VRT(1,N)+XQ(2)*VRT(2,N)
     2+XQ(3)*VRT(3,N))+TR(N)
      WRITE(6,4426) MR,(XQ(N),N=1,3),(XRR(N,MR),N=1,3)
 4426 FORMAT(2H *,T8,I3,T15,F8.3,2(',',F8.3),T44,F8.3,2(',',F8.3)
     2,T79,1H*)
      TQR=THRZ*RPD
      PQR=PHRZ*RPD
      XQ(1)=SIN(TQR)*COS(PQR)
      XQ(2)=SIN(TQR)*SIN(PQR)
      XQ(3)=COS(TQR)
      DO 4431 N=1,3
 4431 VXRR(3,N,MR)=XQ(1)*VRT(1,N)+XQ(2)*VRT(2,N)+XQ(3)*VRT(3,N)
      TQR=THRX*RPD
```

```
      PQR=PHRX*RPD
      XQ(1)=SIN(TQR)*COS(PQR)
      XQ(2)=SIN(TQR)*SIN(PQR)
      XQ(3)=COS(TQR)
      DO 4432 N=1,3
 4432 VXRR(1,N,MR)=XQ(1)*VRT(1,N)+XQ(2)*VRT(2,N)+XQ(3)*VRT(3,N)
      DZX=VXRR(1,1,MR)*VXRR(3,1,MR)+VXRR(1,2,MR)*VXRR(3,2,MR)
     2+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
      VXRR(1,1,MR)=VXRR(1,1,MR)-VXRR(3,1,MR)*DZX
      VXRR(1,2,MR)=VXRR(1,2,MR)-VXRR(3,2,MR)*DZX
      VXRR(1,3,MR)=VXRR(1,3,MR)-VXRR(3,3,MR)*DZX
      DOT=VXRR(1,1,MR)*VXRR(1,1,MR)+VXRR(1,2,MR)*VXRR(1,2,MR)
     2+VXRR(1,3,MR)*VXRR(1,3,MR)
      DOT=SQRT(DOT)
      VXRR(1,1,MR)=VXRR(1,1,MR)/DOT
      VXRR(1,2,MR)=VXRR(1,2,MR)/DOT
      VXRR(1,3,MR)=VXRR(1,3,MR)/DOT
      VXRR(2,1,MR)=VXRR(3,2,MR)*VXRR(1,3,MR)-VXRR(3,3,MR)*VXRR(1,2,MR)
      VXRR(2,2,MR)=VXRR(3,3,MR)*VXRR(1,1,MR)-VXRR(3,1,MR)*VXRR(1,3,MR)
      VXRR(2,3,MR)=VXRR(3,1,MR)*VXRR(1,2,MR)-VXRR(3,2,MR)*VXRR(1,1,MR)
      WRITE(6,3006)
      WRITE(6,3006)
      WRITE(6,4437)
 4437 FORMAT(2H *,5X,'THE FOLLOWING RECEIVER ALIGNMENT IS USED:'
     2,T79,1H*)
      DO 4433 NI=1,3
      WRITE(6,3006)
 4433 WRITE(6,4434) (NI,NJ,MR,VXRR(NI,NJ,MR),NJ=1,3)
 4434 FORMAT(2H *,1X,3(2X,'VXRR(',I1,',',I1,',',I2,')=',F9.5)
     2,T79,1H*)
      GO TO 3000
C======
 4810 CONTINUE
C===  RA:     COMMAND         ======
C$$$
C$$$  MRX=NUMBER OF ANTENNA ARRAY GROUPINGS.
C$$$
C$$$  MRAX=NUMBER OF ELEMENTS PER GROUPING.
C$$$
      LRCVR=.TRUE.
      LRMP=.FALSE.
      MRX=MRX+1
      READ(5,*) MRAX
      MRAT=MRAT+MRAX
      MRXAT=MRAT+MRX
      IF(MRXAT.GT.MRDX) WRITE(6,4404) MRXAT
      IF(MRXAT.GT.MRDX) STOP
      WRITE(6,4805) MRX,MRAX
 4805 FORMAT(2H *,5X,'THIS IS RECEIVER NO. ',I3,' IN THIS',
     2' COMPUTATION.',T79,1H*/2H *,5X,'THERE ARE ',
     3I3,' RECEIVERS ARRAYED TOGETHER.',T79,1H*)
      WRITE(6,3006)
      WRITE(6,3006)
C$$$
C$$$  XRR(N,MA)=XYZ LOCATION OF MA-TH ANTENNA ELEMENT.
C$$$
C$$$  XRR(N,MR)=XYZ LOCATION OF MR-TH WEIGHTED CENTER OF THE
C$$$          ARRAY GROUPING.
C$$$
C$$$  THE ARRAY ELEMENTS ARE ASSUMED TO HAVE THE SAME LENGTH.
```

```
C$$$ WIDTH, AND ORIENTATION.  ALSO, THEY ARE ASSUMED TO BE
C$$$ EITHER ALL MOUNTED AND OR ALL OFF A PLATE.
C$$$ IMR(MR)=TYPE OF LINEAR ANTENNA
C$$$         .LT.0: ELECTRIC LINEAR ELEMENT
C$$$         .GT.0: MAGNETIC LINEAR ELEMENT
C$$$ ABS(IMR)=1: UNIFORM CURRENT DISTRIBUTION
C$$$         =2: STANDARD DIPOLE CURRENT DISTRIBUTION
C$$$         =3: CAVITY BACKED SLOT CURRENT DISTRIBUTION (TE01)
C$$$
C$$$ HAWR(MR)=APERTURE WIDTH IN WAVELENGTHS   (NOTE: IF
C$$$ HAWR(MR) IS LESS THAN .1 LAMBDA, RECEIVER IS
C$$$ CONSIDERED TO BE DIPOLE RECEIVER
C$$$ HR(MR)=LENGTH OF LINEAR ELEMENT IN WAVELENGTHS
C$$$
C$$$ THRZ,PHRZ=ORIENTATION ANGLES USED TO DEFINE LINEAR
C$$$ ELEMENT AXIS.
C$$$
C$$$ THRX,PHRX=ORIENTATION ANGLES USED TO DEFINE APERTURE
C$$$ PLANE OR DIPOLE X-AXIS.
C$$$
C$$$ WMR,WPR=MAGNITUDE AND PHASE OF EXCITATION OF
C$$$ MA-TH ELEMENT.
C$$$
      MR=MRX
      MAI=MRDX-MRAT+1
      MAF=MAI+MRAX-1
      MRA(1,MR)=MAI
      MRA(2,MR)=MAF
      DO 4841 MA=MAI,MAF
 4841 READ(5,*) (XRR(N,MA),N=1,3)
      READ(5,*) THRZ,PHRZ,THRX,PHRX
      READ(5,*) IMR(MR),HR(MR),HAWR(MR)
      IF(IMR(MR).LT.0) WRITE(6,4411) IMR(MR)
      IF(IMR(MR).GE.0) WRITE(6,4412) IMR(MR)

      WRITE(6,3006)
      IF(IUNST.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(IUNST.NE.1) WRITE(6,3006)
      IF(IUNST.NE.1) WRITE(6,4413) HR(MR),HAWR(MR),LABEL(1)
      GO TO 4816
 4814 WRITE(6,4415) HR(MR),HAWR(MR)
 4816 WRITE(6,3006)
      WR(MR)=(1.,0.)
      WMRA=0.
      XRAX=0.
      XRAY=0.
      XRAZ=0.
      DO 4843 MA=MAI,MAF
      READ(5,*) WMR,WPR
      WRITE(6,4817) MA,WMR,WPR
 4817 FORMAT(2H *,5X,'RECEIVER ',I3,' HAS MAGNITUDE='
     2,F10.5,' AND PHASE=',F10.5,T79,1H*)
      WR(MA)=WMR*CEXP(CJ*WPR*RPD)
      WMRA=WMRA+WMR
      XRAX=XRAX+WMR*XRR(1,MA)
      XRAY=XRAY+WMR*XRR(2,MA)
 4843 XRAZ=XRAZ+WMR*XRR(3,MA)
      XRR(1,MR)=XRAX/WMRA
      XRR(2,MR)=XRAY/WMRA
      XRR(3,MR)=XRAZ/WMRA
      WRITE(6,3006)
```

```
      WRITE(6,3006)
      WRITE(6,4421) LABEL(IUNIT)
      WRITE(6,4422)
      WRITE(6,3006)
      DO 4824 N=1,3
 4824 XQ(N)=XRR(N,MR)
      DO 4825 N=1,3
 4825 XRR(N,MR)=UNITS*(XQ(1)*VRT(1,N)+XQ(2)*VRT(2,N)
     2+XQ(3)*VRT(3,N))+TR(N)
      WRITE(6,4426) MR,(XQ(N),N=1,3),(XRR(N,MR),N=1,3)
      DO 4829 MA=MAI,MAF
      DO 4827 N=1,3
 4827 XQ(N)=XRR(N,MA)
      DO 4828 N=1,3
 4828 XRR(N,MA)=UNITS*(XQ(1)*VRT(1,N)+XQ(2)*VRT(2,N)
     2+XQ(3)*VRT(3,N))+TR(N)
 4829 WRITE(6,4426) MA,(XQ(N),N=1,3),(XRR(N,MA),N=1,3)
      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(3,N,MR)=XQ(1)*VRT(1,N)+XQ(2)*VRT(2,N)+XQ(3)*VRT(3,N)
      TQR=THRX*RPD
      PQR=PHRX*RPD
      XQ(1)=SIN(TQR)*COS(PQR)
      XQ(2)=SIN(TQR)*SIN(PQR)
      XQ(3)=COS(TQR)
      DO 4832 N=1,3
 4832 VXRR(1,N,MR)=XQ(1)*VRT(1,N)+XQ(2)*VRT(2,N)+XQ(3)*VRT(3,N)
      DZX=VXRR(1,1,MR)*VXRR(3,1,MR)+VXRR(1,2,MR)*VXRR(3,2,MR)
     2+VXRR(1,3,MR)*VXRR(3,3,MR)
      IF(ABS(DZX).GT.0.1) WRITE(6,4436)
      IF(ABS(DZX).GT.0.1) STOP
      VXRR(1,1,MR)=VXRR(1,1,MR)-VXRR(3,1,MR)*DZX
      VXRR(1,2,MR)=VXRR(1,2,MR)-VXRR(3,2,MR)*DZX
      VXRR(1,3,MR)=VXRR(1,3,MR)-VXRR(3,3,MR)*DZX
      DOT=VXRR(1,1,MR)*VXRR(1,1,MR)+VXRR(1,2,MR)*VXRR(1,2,MR)
     2+VXRR(1,3,MR)*VXRR(1,3,MR)
      DOT=SQRT(DOT)
      VXRR(1,1,MR)=VXRR(1,1,MR)/DOT
      VXRR(1,2,MR)=VXRR(1,2,MR)/DOT
      VXRR(1,3,MR)=VXRR(1,3,MR)/DOT
      VXRR(2,1,MR)=VXRR(3,2,MR)*VXRR(1,3,MR)-VXRR(3,3,MR)*VXRR(1,2,MR)
      VXRR(2,2,MR)=VXRR(3,3,MR)*VXRR(1,1,MR)-VXRR(3,1,MR)*VXRR(1,3,MR)
      VXRR(2,3,MR)=VXRR(3,1,MR)*VXRR(1,2,MR)-VXRR(3,2,MR)*VXRR(1,1,MR)
      WRITE(6,3006)
      WRITE(6,3006)
      WRITE(6,4437)
      DO 4833 NI=1,3
      WRITE(6,3006)
 4833 WRITE(6,4434) (NI,NJ,MR,VXRR(NI,NJ,MR),NJ=1,3)
      GO TO 3000
C======
 3450 CONTINUE
C===  SM:     COMMAND          ======
C$$$
C$$$  PRAD=TOTAL POWER RADIATED IN WATTS
C$$$
      LPRAD=.TRUE.
      READ(5,*) PRAD
      WRITE(6,3441) PRAD
      WRITE(6,3006)
```

```
C$$$
C$$$  MSX=NUMBER OF ANTENNA SEGMENTS
C$$$
      LSMP=.TRUE.
      READ(5,*) MSX
      IF(MSX.GT.MSDX) WRITE(6,3477) MSX
 3477 FORMAT(' *****   NUMBER OF SEGMENTS= ',I3,
     2' PROGRAM ABORTS SINCE MAX. SOURCE DIMENSION'
     3,' IS EXCEEDED.   *****')
      IF(MSX.GT.MSDX) STOP
      WRITE(6,3451) MSX
 3451 FORMAT(2H *,5X,'THERE ARE ',I3,' SEGMENTS IN THIS',
     2' COMPUTATION.',T79,1H*)
      WRITE(6,3006)
      WRITE(6,3006)
C$$$
C$$$  XS(MS,N)=XYZ LOCATION OF MS-TH ANTENNA SEGMENT
C$$$
C$$$  IMS(MS)=-1=ELECTRIC LINEAR ELEMENT WITH A UNIFORM DISTRIBUTION
C$$$
C$$$  HS(MS)=LENGTH OF LINEAR ELEMENT
C$$$
C$$$  THSZ,PHSZ=ORIENTATION ANGLES USED TO DEFINE
C$$$  LINEAR ELEMENT AXIS.
C$$$
C$$$  WMS,WPS=REAL AND IMAGINARY CURRENT WEIGHT.
C$$$
      WRITE(6,3458) LABEL(IUNIT)
 3458 FORMAT(2H *,T7,'MS',T13,'HS:',A6,T23,'HS:METERS',
     2T41,'INPUT: THS,PHS',T60,'ACTUAL: THS,PHS',T79,1H*)
      WRITE(6,3459)
 3459 FORMAT(2H *,T6,3('-'),T12,20('-'),T40,16('-'),T59,
     217('-'),T79,1H*)
      WRITE(6,3006)
      DO 3463 MS=1,MSX
      READ(5,*) (XSS(N,MS),N=1,3),HS(MS),THSZ,PHSZ
      MSA(1,MS)=0
      MSA(2,MS)=0
      IMS(MS)=-1
      HAWS(MS)=0.
      HSQ=HS(MS)
      HS(MS)=UNITS*HSQ
      TQ=90.-THSZ
      PQ=PHSZ
      XQ(1)=SIN(TQ*RPD)*COS(PQ*RPD)
      XQ(2)=SIN(TQ*RPD)*SIN(PQ*RPD)
      XQ(3)=COS(TQ*RPD)
      DO 3481 N=1,3
 3481 XQR(N)=XQ(1)*VRT(1,N)+XQ(2)*VRT(2,N)+XQ(3)*VRT(3,N)
      THSZ=DPR*BTAN2(SQRT(XQR(1)*XQR(1)+XQR(2)*XQR(2)),XQR(3))
      PHSZ=DPR*BTAN2(XQR(2),XQR(1))
      WRITE(6,3464) MS,HSQ,HS(MS),TQ,PQ,THSZ,PHSZ
 3464 FORMAT(2H *,T6,I3,3X,2(2X,F8.4),5X,2(2X,F8.3,',',F8.3)
     2,T79,1H*)
      DO 3484 N=1,3
 3484 VXSS(3,N,MS)=XQR(N)

      VXSS(1,1,MS)=COS(THSZ*RPD)*COS(PHSZ*RPD)
      VXSS(1,2,MS)=COS(THSZ*RPD)*SIN(PHSZ*RPD)
      VXSS(1,3,MS)=-SIN(THSZ*RPD)
      VXSS(2,1,MS)=-SIN(PHSZ*RPD)
      VXSS(2,2,MS)=COS(PHSZ*RPD)
      VXSS(2,3,MS)=0.
 3463 CONTINUE
      WRITE(6,3006)
```

```
      WRITE(6,3006)
      WRITE(6,3006)
      WRITE(6,3454)
 3454 FORMAT(2H *,T31,'SEGMENT COORDINATES',T79,'*')
      WRITE(6,3006)
      WRITE(6,3006)
      WRITE(6,3456) LABEL(IUNIT)
 3456 FORMAT(2H *,T7,'MS',T14,'INPUT LOCATION IN ',A6,
     2T43,'ACTUAL LOCATION IN METERS',T79,'*')
      WRITE(6,3457)
 3457 FORMAT(2H *,T6,3('-'),T13,26('-'),T42,27('-'),T79,'*')
      WRITE(6,3006)
      DO 3473 MS=1,MSX
      DO 3474 N=1,3
 3474 XQ(N)=XSS(N,MS)
      DO 3475 N=1,3
 3475 XSS(N,MS)=UNITS*(XQ(1)*VRT(1,N)+XQ(2)*VRT(2,N)
     2+XQ(3)*VRT(3,N))+TR(N)
      WRITE(6,3476) MS,(XQ(N),N=1,3),(XSS(N,MS),N=1,3)
 3476 FORMAT(2H *,T6,I3,T13,F8.3,2(',',F8.3),T42,F8.3,
     22(',',F8.3),T79,1H*)
 3473 CONTINUE
      WRITE(6,3006)
      WRITE(6,3006)
      WRITE(6,3485)
 3485 FORMAT(2H *,T33,'CURRENT WEIGHTS',T79,1H*,/2H *,T7,'MS',T18,
     2'REAL',T31,'IMAG.',T46,'MAG.',T57,'PHASE',T79,1H*)
      WRITE(6,3486)
 3486 FORMAT(2H *,T6,3('-'),T17,6('-'),T30,7('-'),T45,6('-'),
     2T56,7('-'),T79,1H*)
      DO 3465 MS=1,MSX
      READ(5,*) WMS,WPS
      WS(MS)=CMPLX(WMS,WPS)
      WMM=BABS(CMPLX(WMS,WPS))
      WPP=DPR*BTAN2(WPS,WMS)
      WRITE(6,3466) MS,WMS,WPS,WMM,WPP
 3466 FORMAT(2H *,T6,I3,5X,E11.4,2X,E11.4,4X,E11.4,2X,F8.3,T79,1H*)
 3465 CONTINUE
      WRITE(6,3006)
      GO TO 3000
C======
 4450 CONTINUE
C===  RM:     COMMAND          ======
C$$$
C$$$  PRADR=TOTAL POWER RADIATED IN WATTS
C$$$
      READ(5,*) PRADR
      WRITE(6,3441) PRADR
      WRITE(6,3006)
C$$$
C$$$  MRX=NUMBER OF ANTENNA SEGMENTS
C$$$
      LRCVR=.TRUE.
      LRMP=.TRUE.
      READ(5,*) MRX
      IF(MRX.GT.MRDX) WRITE(6,4477) MRX
 4477 FORMAT(' *****   NUMBER OF SEGMENTS= ',I3,
     2' PROGRAM ABORTS SINCE MAX. RECEIVER DIMENSION'
     3,' IS EXCEEDED.   *****')
      IF(MRX.GT.MRDX) STOP
      WRITE(6,3451) MRX
      WRITE(6,3006)
      WRITE(6,3006)
C$$$
```

```
C$$$  XRR(N,MR)=XYZ LOCATION OF MR-TH ANTENNA SEGMENT
C$$$
C$$$  IMR(MR)=-1=ELECTRIC LINEAR ELEMENT WITH A UNIFORM DISTRIBUTION
C$$$
C$$$  HR(MR)=LENGTH OF LINEAR ELEMENT
C$$$
C$$$  THRZ,PHRZ=ORIENTATION ANGLES USED TO DEFINE
C$$$  LINEAR ELEMENT AXIS.
C$$$
C$$$  WMR,WPR=REAL AND IMAGINARY CURRENT WEIGHT.
C$$$
      WRITE(6,4458) LABEL(IUNIT)
 4458 FORMAT(2H *,T7,'MR',T13,'HR:',A6,T23,'HR:METERS',
     2T41,'INPUT: THR,PHR',T60,'ACTUAL: THR,PHR',T79,1H*)
      WRITE(6,3459)
 4459 FORMAT(2H *,T6,3('-'),T12,20('-'),T40,16('-'),T59,
     217('-'),T79,1H*)
      WRITE(6,3006)
      DO 4463 MR=1,MRX
      READ(5,*) (XRR(N,MR),N=1,3),HR(MR),THRZ,PHRZ
      MRA(1,MR)=0
      MRA(2,MR)=0
      IMR(MR)=-1
      HAWR(MR)=0.
      HRQ=HR(MR)
      HR(MR)=UNITS*HRQ
      TQ=90.-THRZ
      PQ=PHRZ
      XQ(1)=SIN(TQ*RPD)*COS(PQ*RPD)
      XQ(2)=SIN(TQ*RPD)*SIN(PQ*RPD)
      XQ(3)=COS(TQ*RPD)
      DO 4481 N=1,3
 4481 XQR(N)=XQ(1)*VRT(1,N)+XQ(2)*VRT(2,N)+XQ(3)*VRT(3,N)
      THRZ=DPR*BTAN2(SQRT(XQR(1)*XQR(1)+XQR(2)*XQR(2)),XQR(3))
      PHRZ=DPR*BTAN2(XQR(2),XQR(1))
      WRITE(6,3464) MR,HRQ,HR(MR),TQ,PQ,THRZ,PHRZ
      DO 4484 N=1,3
 4484 VXRR(3,N,MR)=XQR(N)
      VXRR(1,1,MR)=COS(THRZ*RPD)*COS(PHRZ*RPD)
      VXRR(1,2,MR)=COS(THRZ*RPD)*SIN(PHRZ*RPD)
      VXRR(1,3,MR)=-SIN(THRZ*RPD)
      VXRR(2,1,MR)=-SIN(PHRZ*RPD)
      VXRR(2,2,MR)=COS(PHRZ*RPD)
      VXRR(2,3,MR)=0.
 4463 CONTINUE
      WRITE(6,3006)
      WRITE(6,3006)
      WRITE(6,3006)
      WRITE(6,3454)
 3    WRITE(6,3006)
      WRITE(6,3006)
      WRITE(6,4456) LABEL(IUNIT)
 4456 FORMAT(2H *,T7,'MR',T14,'INPUT LOCATION IN ',A6,
     2T43,'ACTUAL LOCATION IN METERS',T79,'*')
      WRITE(6,3457)
      WRITE(6,3006)
      DO 4473 MR=1,MRX
      DO 4474 N=1,3
 4474 XQ(N)=XRR(N,MR)
      DO 4475 N=1,3
 4475 XRR(N,MR)=UNITS*(XQ(1)*VRT(1,N)+XQ(2)*VRT(2,N)
     2+XQ(3)*VRT(3,N))+TR(N)
      WRITE(6,3476) MR,(XQ(N),N=1,3),(XRR(N,MR),N=1,3)
 4473 CONTINUE
```

```
      WRITE(6,3006)
      WRITE(6,3006)
      WRITE(6,3485)
      WRITE(6,3486)
      DO 4465 MR=1,MRX
      READ(5,*) WMR,WPR
      WR(MR)=CMPLX(WMR,WPR)
      WMM=BABS(CMPLX(WMR,WPR))
      WPP=DPR*BTAN2(WPR,WMR)
      WRITE(6,3466) MR,WMR,WPR,WMM,WPP
 4465 CONTINUE
      WRITE(6,3006)
      GO TO 3000
C======
 3490 CONTINUE
C===  NS:     COMMAND          ======
C$$$
C$$$  INITIALIZE SOURCE DATA.
C$$$
      LSMP=.FALSE.
      MSX=0
      MSAT=0
      WRITE(6,3491)
 3491 FORMAT(2H *,5X,' THE SOURCE DATA IS INITIALIZED. ',T79,1H*/
     2,2H *,5X,' NO SOURCES ARE PRESENTLY IN THE PROBLEM. '
     3,T79,1H*)
      GO TO 3000
C======
 3495 CONTINUE
C===  NR:     COMMAND          ======
C$$$
C$$$  INITIALIZE RECEIVER DATA
C$$$
      LRCVR=.FALSE.
      LRMP=.FALSE.
      MRX=0
      WRITE(6,3496)
 3496 FORMAT(2H *,5X,' THE RECEIVER DATA IS INITIALIZED. '
     2,T79,1H*/2H *,5X,' NO RECEIVERS ARE PRESENTLY IN THE'
     3,' PROBLEM. ',T79,1H*)
      GO TO 3000
C======
 3500 CONTINUE
C===  LP:     COMMAND          ======
C$$$
C$$$  LWRITE=TRUE IF LINE PRINTER OUTPUT OF DATA IS DESIRED
C$$$
      READ(5,*) LWRITE
      IF(.NOT.LWRITE) WRITE(6,5505)
 5505 FORMAT(2H *,5X,'NO LINE PRINTER OUTPUT',T79,1H*)
      IF(.NOT.LWRITE) GO TO 3000
      WRITE(6,3501)
 3501 FORMAT(2H *,5X,' DATA WILL BE OUTPUT ON LINE PRINTER !!!',
     2T79,1H*)
      GO TO 3000
C======
 3650 CONTINUE
C===  VP:     COMMAND ======
C$$$
C$$$  VOLUMETRIC DUMP FOR PLOTS
C$$$
      READ(5,*) LVPLT
      IF(.NOT.LVPLT) WRITE(6,6606)
      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$$$
      READ(5,*) LPLT
      IF(.NOT.LPLT) WRITE(6,6606)
 6606 FORMAT(2H *,5X,'NO PEN PLOT DESIRED',T79,1H*)
      IF(.NOT.LPLT) GO TO 3000
C$$$
C$$$ IF LPLT=TRUE READ IN DIMENSIONS
C$$$
C$$$ LPPREC = TRUE IMPLIES RECTANGULAR PLOT
C$$$ PPXL = LENGTH OF X-AXIS (ANGLE AXIS)
C$$$ PPYL = LENGTH OF Y-AXIS (DB AXIS)
C$$$
C$$$ LPPREC = FALSE IMPLIES POLAR PLOT
C$$$ PPXL = ANGULAR POSITION OF X-AXIS
C$$$ PPYL = RADIUS OF GRID
C$$$
 3651 READ(5,*) LPPREC,PPXL,PPYL
C$$$
C$$$ PPXB = BEGINNING VALUE OF X-AXIS
C$$$ PPXE = END VALUE OF X-AXIS
C$$$ PPXS = STEP SIZE OF X-AXIS GRID MARKS
C$$$
      READ(5,*) PPXB,PPXE,PPXS
C$$$
C$$$ PPYB = BEGINNING VALUE OF Y-AXIS
C$$$ PPYE = END VALUE OF Y-AXIS
C$$$ PPYS = STEP SIZE OF Y-AXIS GRID MARKS
C$$$
      READ(5,*) PPYB,PPYE,PPYS
      WRITE(6,3602)
 3602 FORMAT(2H *,5X,'DATA WILL BE OUTPUT FOR A PLOT !!!'
     2,T79,1H*)
      WRITE(6,3006)
      WRITE(6,3603) LPPREC,PPXL,PPYL
 3603 FORMAT(2H *,5X,' LPPREC= ',L2,5X,'PPXL= ',F10.5,5X,
     2'PPYL= ',F10.5,T79,1H*)
      WRITE(6,3604) PPXB,PPXE,PPXS
 3604 FORMAT(2H *,5X,' PPXB= ',F10.5,5X,'PPXE= ',F10.5,5X,
     2'PPXS= ',F10.5,T79,1H*)
      WRITE(6,3605) PPYB,PPYE,PPYS
 3605 FORMAT(2H *,5X,' PPYB= ',F10.5,5X,'PPYE= ',F10.5,5X,
     2'PPYS= ',F10.5,T79,1H*)
      IF(LPLT) GO TO 3000
      WRITE(6,3006)

C$$$
C$$$ IVTYP=TYPE OF RESULTS OUTPUT
C$$$ IVTYP=1 ELECTRIC FIELD OUTPUT
C$$$ IVTYP=2 MAGNETIC FIELD OUTPUT
C$$$ IVTYP=3 BOTH ELECTRIC AND MAGNETIC FIELDS OUTPUT
C$$$ COUPLING IS OUTPUT IF RECEIVER IS DEFINED FOR ANY IVTYP
C$$$
C$$$ IVPOL=POLARIZATION OF RESULTS OUTPUT
C$$$ IVPOL=1,2,3 THEN R,THETA,PHI OR X,Y,Z RESPECTIVELY IS OUTPUT
C$$$ IVPOL=4 THEN R-THETA OR X-Y ARE OUTPUT
C$$$ IVPOL=5 THEN R-PHI OR X-Z ARE OUTPUT
C$$$ IVPOL=6 THEN THETA-PHI OR Y-Z ARE OUTPUT
C$$$ IVPOL=7 THEN R, THETA AND PHI OR X, Y AND Z ARE OUTPUT
C$$$ COUPLING HAS NO POLARIZATION
```

```
C$$$
      READ(5,*) IVTYP,IVPOL
      WRITE(6,3655) IVTYP,IVPOL
 3655 FORMAT(2H *,5X,' IVTYP= 'I2,' IVPOL= ',I2,T79,1H*)
      GO TO 3000
C======
 3700 CONTINUE
C===  GP:     COMMAND          ======
C$$$
C$$$  INFINITE GROUND PLANE EFFECT INCLUDED.
C$$$
      LGRND=.TRUE.
      DO 3702 N=1,3
      XX(N,1,MPDX)=1.E5*(VRT(1,N)+VRT(2,N))+TR(N)
      XX(N,2,MPDX)=1.E5*(-VRT(1,N)+VRT(2,N))+TR(N)
      XX(N,3,MPDX)=1.E5*(-VRT(1,N)-VRT(2,N))+TR(N)
 3702 XX(N,4,MPDX)=1.E5*(VRT(1,N)-VRT(2,N))+TR(N)
C$$$
C$$$  LSLAB= 0  IMPLIES METAL PLATE, AND
C$$$        =-3  IMPLIES DIELECTRIC HALF SPACE
C$$$
C$$$  NOTE:  IF DIELECTRIC COVERED, ONE MUST READ DIELECTRIC DATA.
C$$$
      READ(5,*) LSLAB(MPDX)
      IF(LSLAB(MPDX).EQ.0) WRITE(6,3706)
 3706 FORMAT(2H *,5X,'PERFECTLY CONDUCTING',T79,1H*)
      IF(LSLAB(MPDX).NE.0) WRITE(6,3707)
 3707 FORMAT(2H *,5X,'SEMI-INFINITELY THICK DIELECTRIC',T79,1H*)
      WRITE(6,3701)
 3701 FORMAT(2H *,5X,'INFINITE GROUND PLANE INSERTED IN',
     2' STRUCTURE !!!',T79,1H*)
      WRITE(6,3006)
      WRITE(6,3703) (TR(N),N=1,3)
 3703 FORMAT(2H *,5X,'THE ORIGIN IS AT ',F12.6,',',F12.6
     2,',',F12.6,' METERS',T79,1H*)
      WRITE(6,3006)
      WRITE(6,3704) (VRT(3,N),N=1,3)
 3704 FORMAT(2H *,5X,'THE NORMAL IS ',F12.6,',',F12.6,','
     2,F12.6,T79,1H*)
      IF(LSLAB(MPDX).EQ.0) GO TO 3000
      NSLAB(1)=1
      DSLAB(1,MPDX)=0.
      LSLAB(MPDX)=-3
C$$$
C$$$  ERSLAB(1,MPDX)=RELATIVE DIELECTRIC CONSTANT
C$$$
C$$$  TESLAB(1,MPDX)=DIELECTRIC LOSS TANGENT
C$$$
C$$$  URSLAB(1,MPDX)=RELATIVE PERMEABILITY CONSTANT
C$$$
C$$$  TMSLAB(1,MPDX)=PERMEABILITY LOSS TANGENT
C$$$
      READ(5,*) ERSLAB(1,MPDX),TESLAB(1,MPDX)
     2,URSLAB(1,MPDX),TMSLAB(1,MPDX)
      WRITE(6,3006)
      WRITE(6,3708)
 3708 FORMAT(2H *,5X,'DIELECTRIC',3X,'LOSS',4X,
     2'PERMITIVITY',3X,'LOSS',T79,1H*,/,
     32H *,6X,'CONSTANT',3X,'TANGENT',
     44X,'CONSTANT',3X,'TANGENT',T79,1H*,/,
     52H *,5X,'----------',2X,'-------',
     62X,'-----------',2X,'-------',T79,1H*)
      WRITE(6,3709) ERSLAB(1,MPDX),TESLAB(1,MPDX)
     2,URSLAB(1,MPDX),TMSLAB(1,MPDX)
```

```
3709  FORMAT(2H *,6X,F10.4,2X,F7.4,2X,F11.4,2X,
     2F7.4,T79,1H*)
      GO TO 3000
C======
3750  CONTINUE
C===  NG:     COMMAND        ======
C$$$
C$$$  INITIALIZE GROUND PLANE DATA.
C$$$
      LGRND=.FALSE.
      WRITE(6,3751)
3751  FORMAT(2H *,5X,' GROUND PLANE DATA IS INITIALIZED. ',T79,1H*/
     2,2H *,5X,' NO GROUND PLANE IS PRESENTLY IN THE PROBLEM. '
     3,T79,1H*)
      GO TO 3000
C======
3900  CONTINUE
C===  RT:     COMMAND        ======
C$$$
C$$$  TR(N)=LINEAR TRANSLATION OF COORDINATES FROM THE FIXED
C$$$  COORDINATES WHICH IS ORIGINALLY SET UP BY OPERATOR.
C$$$
      READ(5,*) (TR(N),N=1,3)
      WRITE(6,3901) LABEL(IUNIT),(TR(N),N=1,3)
3901  FORMAT(2H *,5X,'TRANSLATION IN ',A6,':  TR(1)=',F8.3,
     2'  TR(2)=',F8.3,'  TR(3)=',F8.3,T79,1H*)
      DO 3920 N=1,3
3920  TR(N)=TR(N)*UNITS
      WRITE(6,3006)
      IF(IUNIT.NE.1) WRITE(6,3901) LABEL(1),(TR(N),N=1,3)
      IF(IUNIT.NE.1) WRITE(6,3006)
      WRITE(6,3006)
C$$$
C$$$  THZP,PHZP=ORIENTATION OF THE VRT(3,N) AXIS RELATIVE TO THE
C$$$  FIXED COORDINATE SYSTEM.
C$$$
C$$$  THXP,PHXP=ORIENTATION OF THE VRT(1,N) AXIS RELATIVE TO THE
C$$$  FIXED COORDINATE SYSTEM.
C$$$
      READ(5,*) 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,N) IS PERPENDICULAR TO VRT(3,N)
      DZX=VRT(3,1)*VRT(1,1)+VRT(3,2)*VRT(1,2)+VRT(3,3)*VRT(1,3)
      IF(ABS(DZX).GT.0.1) WRITE(6,3903)
3903  FORMAT(' *** PROGRAM ABORTS IN ROTATE SECTION IN THAT THE',
     2' COORDINATES ARE NOT ORTHOGONAL!!! ***')
      IF(ABS(DZX).GT.0.1) STOP
      VRT(1,1)=VRT(1,1)-VRT(3,1)*DZX
      VRT(1,2)=VRT(1,2)-VRT(3,2)*DZX
      VRT(1,3)=VRT(1,3)-VRT(3,3)*DZX
      DOT=VRT(1,1)*VRT(1,1)+VRT(1,2)*VRT(1,2)+VRT(1,3)*VRT(1,3)
      DOT=SQRT(DOT)
      VRT(1,1)=VRT(1,1)/DOT
      VRT(1,2)=VRT(1,2)/DOT
      VRT(1,3)=VRT(1,3)/DOT
      VRT(2,1)=VRT(3,2)*VRT(1,3)-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)
```

```
3931  FORMAT(2H *,5X,'THE FOLLOWING ROTATIONS ARE USED FOR ALL',
     2' SUBSEQUENT INPUTS:',T79,1H*)
      DO 3932 NI=1,3
      WRITE(6,3006)
3932  WRITE(6,3933) (NI,NJ,VRT(NI,NJ),NJ=1,3)
3933  FORMAT(2H *,1X,3(2X,'VRT(',I1,',',I1,')=',F9.5),T79,1H*)
      GO TO 3000
C======
4000  CONTINUE
C===  CG: AND CC:     COMMAND        ======
C$$$
C$$$  CYLINDER GEOMETRY INPUT
C$$$
      LCYL=.TRUE.
      MCX=MCX+1
      MC=MCX
      IF(MCX.GT.MCDX) WRITE(6,6311) MCX
6311  FORMAT(' *****   NUMBER OF CYLINDERS= ',I3,'PROGRAM',
     2' ABORTS SINCE MAX. CYLINDER DIMENSION IS EXCEEDED.   *****')
      IF(MCX.GT.MCDX) STOP
C$$$
C$$$  XCL(N,MC)=XYZ LOCATION OF THE ORIGIN OF THE MC-TH CYLINDER
C$$$
      READ(5,*) (XCL(N,MC),N=1,3)
      DO 6301 N=1,3
6301  XQ(N)=XCL(N,MC)
      DO 6302 N=1,3
6302  XCL(N,MC)=UNITS*(XQ(1)*VRT(1,N)+XQ(2)*VRT(2,N)
     2+XQ(3)*VRT(3,N))+TR(N)
      WRITE(6,6308) LABEL(IUNIT)
6308  FORMAT(2H *,T5,'CYLINDER#',T17,'INPUT LOCATION IN ',A6
     2,T46,'ACTUAL LOCATION IN METERS',T79,1H*)
      WRITE(6,3422)
      WRITE(6,3006)
      WRITE(6,3426) MC,(XQ(N),N=1,3),(XCL(N,MC),N=1,3)
      WRITE(6,3006)
      WRITE(6,3006)
C$$$
C$$$  TCLZ,PCLZ=ORIENTATION OF THE CYLINDER AXIS
C$$$
C$$$  TCLX,PCLX=ORIENTATION OF THE CYLINDER'S X-AXIS
C$$$
      READ(5,*) TCLZ,PCLZ,TCLX,PCLX
      XQ(1)=SIN(TCLZ*RPD)*COS(PCLZ*RPD)
      XQ(2)=SIN(TCLZ*RPD)*SIN(PCLZ*RPD)
      XQ(3)=COS(TCLZ*RPD)
      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)
      XQ(1)=SIN(TCLX*RPD)*COS(PCLX*RPD)
      XQ(2)=SIN(TCLX*RPD)*SIN(PCLX*RPD)
      XQ(3)=COS(TCLX*RPD)
      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)
      DZX=VCL(1,1,MC)*VCL(3,1,MC)+VCL(1,2,MC)*VCL(3,2,MC)
     2+VCL(1,3,MC)*VCL(3,3,MC)
      IF(ABS(DZX).GT.0.1) WRITE(6,6305)
6305  FORMAT(' *** PROGRAM ABORTS THE COORDINATES ARE NOT',
     2' ORTHOGONAL!!! ***')
      IF(ABS(DZX).GT.0.1) STOP
      VCL(1,1,MC)=VCL(1,1,MC)-VCL(3,1,MC)*DZX
      VCL(1,2,MC)=VCL(1,2,MC)-VCL(3,2,MC)*DZX
      VCL(1,3,MC)=VCL(1,3,MC)-VCL(3,3,MC)*DZX
      DOT=VCL(1,1,MC)*VCL(1,1,MC)+VCL(1,2,MC)*VCL(1,2,MC)
     2+VCL(1,3,MC)*VCL(1,3,MC)
```

```
      DOT=SQRT(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(3,2,MC)*VCL(1,3,MC)-VCL(3,3,MC)*VCL(1,2,MC)
      VCL(2,2,MC)=VCL(3,3,MC)*VCL(1,1,MC)-VCL(3,1,MC)*VCL(1,3,MC)
      VCL(2,3,MC)=VCL(3,1,MC)*VCL(1,2,MC)-VCL(3,2,MC)*VCL(1,1,MC)
      WRITE(6,6309)
 6309 FORMAT(2H *,5X,'THE FOLLOWING CYLINDER ALIGNMENT IS USED:'
     2,T79,1H*)
      DO 6306 NI=1,3
      WRITE(6,3006)
 6306 WRITE(6,6307) (NI,NJ,VCL(NI,NJ,MC),NJ=1,3)
 6307 FORMAT(2H *,1X,3(2X,'VCL(',I1,',',I1,')=',F9.5),T79,1H*)
C$$$
C$$$  AC=RADIUS OF ELLIPSE ON X CYLINDER AXIS
C$$$  BC=RADIUS OF ELLIPSE ON Y CYLINDER AXIS
C$$$
C$$$  ZCN,THTN=MOST NEGATIVE ENDCAP'S Z COMPONENT
C$$$  AND ANGLE OF SURFACE WITH THE CYLINDER AXIS
C$$$  ZCP,THTP=MOST POSITIVE ENDCAP'S Z COMPONENT
C$$$  AND ANGLE OF SURFACE WITH THE CYLINDER AXIS
C$$$
      IF(IR(1).EQ.IT(40)) GO TO 6400
      NEC(MC)=2
      READ(5,*) AC(1,MC),BC(1,MC)
      READ(5,*) ZCN,THTN,ZCP,THTP
      AAO=AC(1,MC)
      BBO=BC(1,MC)
      AC(1,MC)=AC(1,MC)*UNITS
      BC(1,MC)=BC(1,MC)*UNITS
      AC(2,MC)=AC(1,MC)
      BC(2,MC)=BC(1,MC)
      ZC(2,MC)=ZCN*UNITS
      TCR(2,MC)=THTN*RPD
      ZC(1,MC)=ZCP*UNITS
      TCR(1,MC)=THTP*RPD
      WRITE(6,3006)
              WRITE(6,3006)
      WRITE(6,6310) LABEL(IUNIT),AAO
 6310 FORMAT(2H *,5X,'X AXIS DIMENSION IN ',
     2A6,'=',F8.3,T79,1H*)
      WRITE(6,3006)
      IF(IUNIT.NE.1) WRITE(6,6310) LABEL(1),AC(1,MC)

      IF(IUNIT.NE.1) WRITE(6,3006)
      WRITE(6,3006)
      WRITE(6,6320) LABEL(IUNIT),BBO
 6320 FORMAT(2H *,5X,'Y AXIS DIMENSION IN ',
     2A6,'=',F8.3,T79,1H*)
      WRITE(6,3006)
      IF(IUNIT.NE.1) WRITE(6,6320) LABEL(1),BC(1,MC)
      IF(IUNIT.NE.1) WRITE(6,3006)
      WRITE(6,3006)
      WRITE(6,6330) LABEL(IUNIT),ZCN
 6330 FORMAT(2H *,5X,'MOST NEGATIVE END CAP Z COMPONENT IN ',
     2A6,'=',F8.3,T79,1H*)
      WRITE(6,3006)
      IF(IUNIT.NE.1) WRITE(6,6330) LABEL(1),ZC(2,MC)
      IF(IUNIT.NE.1) WRITE(6,3006)
      WRITE(6,6340) THTN
 6340 FORMAT(2H *,5X,'ANGLE OF NEG. END CAP SURFACE WITH NEG.',
     2' CYL. AXIS ','=',F8.3,T79,1H*)
      WRITE(6,3006)
      WRITE(6,3006)
```

```
      WRITE(6,6350) LABEL(IUNIT),ZCP
 6350 FORMAT(2H *,5X,'MOST POSITIVE END CAP Z COMPONENT IN ',
     2A6,'=',F8.3,T79,1H*)
      WRITE(6,3006)
      IF(IUNIT.NE.1) WRITE(6,6350) LABEL(1),ZC(1,MC)
      IF(IUNIT.NE.1) WRITE(6,3005)
      WRITE(6,6360) THTP
 6360 FORMAT(2H *,5X,'ANGLE OF POS. END CAP SURFACE WITH POS.',
     2' CYL. AXIS ','=',F8.3,T79,1H*)
      GO TO 3000
 6400 CONTINUE
      READ(5,*) NEC(MC)
      IF(NEC(MC).GT.NCDX) STOP
      NECX=NEC(MC)
      DO 6410 NC=1,NECX
      READ(5,*) AC(NC,MC),BC(NC,MC),ZC(NC,MC)
      AAO=AC(NC,MC)
      BBO=BC(NC,MC)
      ZCN=ZC(NC,MC)
      AC(NC,MC)=AC(NC,MC)*UNITS
      BC(NC,MC)=BC(NC,MC)*UNITS
      ZC(NC,MC)=ZC(NC,MC)*UNITS
      TCR(NC,MC)=0.5*PI
      WRITE(6,3305) MC,NC,AAO,BBO,ZCN,AC(NC,MC),BC(NC,MC),ZC(NC,MC)
 6410 CONTINUE
      GO TO 3000
C======
 4050 CONTINUE
C===  NC:     COMMAND          ======
C$$$
C$$$  INITIALIZE CYLINDER DATA.
C$$$
      LCYL=.FALSE.
      MCX=0
      WRITE(6,4051)
 4051 FORMAT(2H *,5X,' CYLINDER DATA IS INITIALIZED. ',T79,1H*/
     2,2H *,5X,' NO CYLINDER IS PRESENTLY IN THE PROBLEM. '
     3,T79,1H*)
      GO TO 3000
C======
 997  CONTINUE
C===  EN:     COMMAND          ======
C$$$
C$$$  END PROGRAM
C$$$
      WRITE(6,3006)
      WRITE(6,3006)
      WRITE(6,3005)
      STOP
C======
 3800 CONTINUE
C===  XQ:     COMMAND          ======
C$$$
C$$$  EXECUTE PROGRAM
C$$$
      WRITE(6,3006)
      WRITE(6,3006)
C!!!
C!!! 2. INITIALIZATION SECTION
C!!!
      WL=.2997925/FRQG
      WK=TPI/WL
      WRITE(6,3005)
      MPXR=MPX
```

```
C!!!  GROUND PLANE IS ANOTHER PLATE IN SOLUTION.
      IF(LGRND) MPXR=MPX+1
      IF(MPXR.GT.MPDX) WRITE(6,901) MPXR
      IF(MPXR.GT.MPDX) STOP
      IF(.NOT.LGRND) GO TO 3801
      LPLA=.TRUE.
      MEP(MPXR)=4
      DO 3802 I=1,4
      DO 3802 N=1,3
 3802 XX(N,I,MPXR)=XX(N,I,MPDX)
      LSLAB(MPXR)=LSLAB(MPDX)
      DSLAB(1,MPXR)=DSLAB(1,MPDX)
      ERSLAB(1,MPXR)=ERSLAB(1,MPDX)
      TESLAB(1,MPXR)=TESLAB(1,MPDX)
      URSLAB(1,MPXR)=URSLAB(1,MPDX)
      TMSLAB(1,MPXR)=TMSLAB(1,MPDX)
 3801 CONTINUE
      IF(MPXR.EQ.0) LPLA=.FALSE.
      IF(LPLA) CALL GEOM
C!!!  MAKE PATTERN INFORMATION MATCH WITH SHADOW CODE
      IF(IVPN.GT.0) THEN
        NVFT=NPV
        NVFP=NPN
      ELSE
        NVFT=NPN
        NVFP=NPV
      ENDIF
      COLS=NVFT
      THET1=TYS*RPD
      RESTH=TYI*RPD
      THET2=THET1+(COLS-1)*RESTH
      ROWS=NVFP
      PH1=PZS*RPD
      RESPH=PZI*RPD
      PH2=PH1+(ROWS-1)*RESPH
C!!!  MAKE SOURCE INFORMATION FOR THE FIRST ONE MATCH WITH
C!!!  SHADOW CODE
      DO 3806 N=1,3
 3806 ANTENN(N)=XSS(N,1)
      RETURN
      END
```

FUNCTION BABS

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

```
C-----------------------------------------------------------------------
      FUNCTION BABS(Z)
C!!!
C!!!  THIS ROUTINE IS USED TO GIVE COMPLEX ABSOLUTE VALUES. IT IS
C!!!  USED RATHER THAN STANDARD ROUTINES TO AVOID EXECUTION
C!!!  ERRORS.
C!!!
      COMPLEX Z
      COMMON/LIMIT/SML,SMLR,SMLT,BIG
      X=ABS(REAL(Z))
      Y=ABS(AIMAG(Z))
      IF(X.LT.SMLT.AND.Y.LT.SMLT) GO TO 10
      BABS=CABS(Z)
      RETURN
   10 BABS=SMLT
      RETURN
      END
```

BLOCK DATA

This is contant block data.

```
C-----------------------------------------------------------------------
      BLOCK DATA
C!!!
C!!!  LOAD COMMONLY USED DATA INTO COMMON AREA.
C!!!
      COMPLEX CJ,CPI4
      COMMON/PIS/PI,TPI,DPR,RPD
      COMMON/COMP/CJ,CPI4
      COMMON/LIMIT/SML,SMLR,SMLT,BIG
      DATA PI,TPI,DPR,RPD/3.14159265,6.28318531,57.2957795,
     20.0174532925/
      DATA CJ,CPI4/(0.,1.),(.70710678,-.70710678)/
      DATA SML,SMLR,SMLT,BIG/1.E-3,1.E-5,1.E-10,1.E30/
      END
```

FUNCTION BTAN2

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

```
C-----------------------------------------------------------------------
      FUNCTION BTAN2(Y,X)
C!!!
C!!! THIS ROUTINE IS USED TO COMPUTE THE ARCTANGENT. IT IS
C!!! SIMILAR TO ATAN2 EXCEPT IT AVOIDS THE RUN TIME ERRORS.
C!!!
      COMMON/PIS/PI,TPI,DPR,RPD
      COMMON/LIMIT/SML,SMLR,SMLT,BIG
      IF(ABS(X).GT.SMLT) GO TO 50
      IF(ABS(Y).GT.SMLT) GO TO 10
      BTAN2=0.
      RETURN
10    BTAN2=0.5*PI
      IF(Y.LT.0.) BTAN2=-BTAN2
      RETURN
50    BTAN2=ATAN2(Y,X)
      RETURN
      END
```

SUBROUTINE CAPINT

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

```
C-----------------------------------------------------------------------
      SUBROUTINE CAPINT(XIS,D,DHIT,MD,LHIT,MH)
C!!!
C!!!  DOES RAY HIT ENDCAP?
C!!!
      INCLUDE 'SHACOM.FOR'
      DIMENSION XIS(3),D(3),XT(3),XISC(3),DC(3)
      LOGICAL LHIT,LDEBUG,LTEST,LWARN
      COMMON/TEST/LDEBUG,LTEST,LWARN
      COMMON/LIMIT/SML,SMLR,SMLT,BIG
      COMMON/WAVE/WK,WL
      LHIT=.FALSE.
      DHIT=0.
C!!!  STEP THROUGH CYLINDERS
      DO 40 MCC=1,MCX
      IF(MH.LT.0 .AND. IABS(MH).NE.MCC) GO TO 40
      IF(MH.GT.0 .AND. MH.EQ.MCC) GO TO 40
      CALL CYLROT(D,DC,1,MCC)
      CALL CYLROT(XIS,XISC,2,MCC)
C!!!  STEP THRU ENDCAPS
      NECX=NEC(MCC)
      DO 40 MN=1,NECX
      IF(MD.LT.0 .AND. IABS(MD).NE.MN) GO TO 40
      IF(MD.GT.0 .AND. MD.EQ.MN) GO TO 40
      A=AC(MN,MCC)
      B=BC(MN,MCC)
      CNC=COS(TCR(MN,MCC))
      SNC=SIN(TCR(MN,MCC))
      AN=-XISC(1)*CNC+(XISC(3)-ZC(MN,MCC))*SNC
      DN=-CNC*DC(1)+SNC*DC(3)
C!!!  DOES RAY HIT ENDCAP PLANE?
      IF(AN*DN.GE.0.) GO TO 40
C!!!  COMPUTE POINT XT, WHERE RAY HITS ENDCAP PLANE
      DO 10 N=1,3
10    XT(N)=XISC(N)-AN*DC(N)/DN
      RHOT=XT(1)*XT(1)+XT(2)*XT(2)
     2+(XT(3)-ZC(MN,MCC))*(XT(3)-ZC(MN,MCC))
      RHOT=SQRT(RHOT)
      AE=A/SNC
C!!!  IS HIT POINT ON ENDCAP?
      IF(RHOT.GT.AE .AND. RHOT.GT.B) GO TO 40
      IF(RHOT.LT.AE .AND. RHOT.LT.B) GO TO 20
      VE=BTAN2(A*XT(2),B*XT(1))
      CVE=COS(VE)
      SVE=SIN(VE)
      RHO=SQRT(AE*AE*CVE*CVE+B*B*SVE*SVE)
      IF(RHOT.GT.RHO) GO TO 40
20    CONTINUE
C!!!  CALCULATE DHT, THE DISTANCE FROM SOURCE TO HIT POINT
      DHT=0.
      DO 30 N=1,3
30    DHT=DHT+(XT(N)-XISC(N))*(XT(N)-XISC(N))
      DHT=SQRT(DHT)+SMLR*WL
      IF(LHIT .AND. (DHT.GT.DHIT)) GO TO 40
      LHIT=.TRUE.
      DHIT=DHT
      IF(MD.LE.0) GO TO 40
```

```
      CALL CYLROT(XIS,XT,-2,MCC)
40    CONTINUE
      IF(LTEST) THEN
            WRITE(6,900)
900         FORMAT(/,' TESTING CAPINT SUBROUTINE')
            WRITE(6,*) XIS
            WRITE(6,*) D
            WRITE(6,*) DHIT,MD,LHIT,MH
      ENDIF
      RETURN
      END
```

SUBROUTINE CYLINT

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

```
C----------------------------------------------------------------------
      SUBROUTINE CYLINT(XS,D,DHIT,LHIT,LBDF,MH)
C!!!
C!!!  DOES RAY HIT CYLINDER?
C!!!
      INCLUDE 'SHACOM.FOR'
      DIMENSION D(3),XS(3),VTD(2),BTD(4),CTC(2),DC(3),XSC(3)
      LOGICAL LHIT,LBDF,LPLA,LCYL,LDEBUG,LTEST,LHT,LWARN
      COMMON/LPLCY/LPLA,LCYL
      COMMON/TEST/LDEBUG,LTEST,LWARN
      COMMON/LIMIT/SML,SMLR,SMLT,BIG
      COMMON/WAVE/WK,WL
      LHIT=.FALSE.
      DHIT=0.
      IF(.NOT.LCYL) GO TO 100
C!!!  STEP THRU CYLINDETS
      DO 50 MCC=1,MCX
      IF(MH.LT.0 .AND. IABS(MH).NE.MCC) GO TO 50
      IF(MH.GT.0 .AND. MH.EQ.MCC) GO TO 50
      CALL CYLROT(XS,XSC,2,MCC)
      CALL CYLROT(D,DC,1,MCC)
C!!!  DOES RAY HIT CYLINDER SURFACE SECTION?
      PHSR=BTAN2(DC(2),DC(1))
      CPS=COS(PHSR)
      SPS=SIN(PHSR)
      RHOS=SQRT(XSC(1)*XSC(1)+XSC(2)*XSC(2))
C!!!  STEP THRU CYLINDER SECTIONS
      NECX=NEC(MCC)-1
      DO 40 NC=1,NECX
      NCP=NC+1
      A=AC(NC,MCC)
      B=BC(NC,MCC)
C!!!  PARAMETER FOR ELLIPTIC CYLINSER
      CTC(1)=COS(TCR(NC,MCC))/SIN(TCR(NC,MCC))
      CTC(2)=COS(TCR(NCP,MCC))/SIN(TCR(NCP,MCC))
C!!!  PARAMETERS FOR CONE FRUSTUMS SECTION
      ZZC=ZC(NCP,MCC)-ZC(NC,MCC)
      IF(ABS(ZZC).LT.SML*WL) GO TO 40
      TNJ=(AC(NCP,MCC)-AC(NC,MCC))/ZZC
      FL=TNJ*(XSC(3)-ZC(NC,MCC))/A+1.
C!!!  RADII AT SOURCE LOCATION
      AL=A*FL
      BL=B*FL
C!!!  IS SOURCE INSIDE OF INFINITE CYLINDER?
      IF(RHOS.GT.AL .AND. RHOS.GT.BL) GO TO 5
      IF(RHOS.LT.AL .AND. RHOS.LT.BL) GO TO 30
      VE=BTAN2(AL*XSC(2),BL*XSC(1))
      CVE=COS(VE)
      SVE=SIN(VE)
      RHOE=SQRT(AL*AL*CVE*CVE+BL*BL*SVE*SVE)
      IF(RHOS.GT.RHOE) GO TO 5
30    CONTINUE
C!!!  IS SOURCE INSIDE OF FINITE CYLINDER SECTION?
      IF(XSC(3).GT.(ZC(NC,MCC)+XSC(1)*CTC(1))) GO TO 40
      IF(XSC(3).LT.(ZC(NCP,MCC)+XSC(1)*CTC(2))) GO TO 40
      LHIT=.TRUE.
      GO TO 100
```

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

SUBROUTINE CYLROT

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

```
C----------------------------------------------------------------------
      SUBROUTINE CYLROT(XREF,XCYL,N,MCL)
C$$$
C$$$  ROTATES AND OR TRANSLATES VECTORS IN OR OUT OF THE
C$$$  CYLINDER COORDINATE SYSTEMS
C$$$
      INCLUDE 'SHACOM.FOR'
      DIMENSION XREF(3),XCYL(3)
      IF(N.LT.0) GO TO 100
      XC=0.
      DO 50 I=1,3
      XCYL(I)=0.
      DO 50 J=1,3
      IF(N.EQ.2) XC=XCL(J,MCL)
      XCYL(I)=XCYL(I)+VCL(I,J,MCL)*(XREF(J)-XC)
50    CONTINUE
      RETURN
100   DO 200 I=1,3
      XREF(I)=0.
      DO 150 J=1,3
150   XREF(I)=XREF(I)+VCL(J,I,MCL)*XCYL(J)
      IF(N.EQ.-2) XREF(I)=XREF(I)+XCL(I,MCL)
200   CONTINUE
      RETURN
      END
```

SUBROUTINE DOCYLS

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

```
C-----------------------------------------------------------------------
      SUBROUTINE DOCYLS
      INCLUDE 'SHACOM.FOR'
C!!!
C!!!  This subroutine processes all the cylinders one at a time.
C!!!  Do any special cylinders last.
C!!!
       IF     ( FILCNM .GT. 0 ) THEN
        DO 1 MC=1, MCX
         IF ( MC .NE. FILCNM ) CALL DOCYL( MC, FILCHR )
 1      CONTINUE
        CALL DOCYL( FILCNM, FILCHC )
C!!!
C!!! Fill with a different character for each cylinder.
C!!!
       ELSEIF ( FILCNM .LT. 0 ) THEN
        DO 2 MC=1, MCX
         CALL DOCYL( MC, CHAR( MC+ICHAR( '0' ) ) )
 2      CONTINUE
C!!!
C!!! Fill with the main background fill character.
C!!!
       ELSE
        DO 3 MC=1, MCX
         CALL DOCYL( MC, FILCHR )
 3      CONTINUE
       ENDIF
      RETURN
      END
```

SUBROUTINE DOCYL

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

```
C-----------------------------------------------------------------------
      SUBROUTINE DOCYL( IC, FILL )
      INCLUDE 'SHACOM.FOR'
C!!!
C!!!  This subroutine processes a single cylinder.
C!!!
      CHARACTER FILL
      INTEGER
     +        J, K, IC
C! Loop control variables.
     +        INT,
C! Truncate to integer.
     +        THETAI, PHII

      REAL
     +        THETAR, PHIR,
C! Theta & phi in radians.
     +        T,
C! The parametric loop parameter.
     +        MAGME,
C! Length of a psuedo-side
     +        XPY,
C! Scratch variable.
     +        DOT, LSIDOT,
C! Dot product variables
     +        BTAN2, SQRT, ABS,
C! Miscellaneous functions
     +        XYZ( 3 ),
C! temporary vector
     +  XPQ( 3 ),
C! Source to edge in ref coords
     +  XPC( 3 ),
C! Source to edge in pat coords
     +        RIM( 3 ),
C! Point along cap in cyl coords
     +        RIM1( 3 ),
C! Use for dotmin points
     +        RIM2( 3 ), ANCYL( 3 ),
C! Antenna location in cylinder coords
     +        DOTMIN( 2, 10 )
C! The two angles where dot is minimum
      LOGICAL
     +        FNDONE
C! Found one of the zero dots
C!
C!!! Loop through endcaps, and incrementally on edges.
C!!! Transform the antenna to cyl coords (include a translation).
C!!!
        CALL CYLROT( ANTENN, ANCYL, +2, IC )
C!!!
C!!! Do the endcaps one at a time.
C!!!
        DO 200 J=1, NEC(IC)
C!!!
C!!! Loop around the endcap and remember where the dot products are zero
C!!! between the vector looking at the point and the radial vector on
```

```
C!!! the endcap to the point.  The cryptic parameters on the loop say:
C!!!    "Loop from zero to 2*PI in one-degree steps."
C!!!
              DO 300 T=0.0, TPI+(TPI/360.0), (TPI/360.0)
C!!!
C!!! Calculate the dot product and remember the two smallest ones.
C!!!
          RIM( 1 ) = COS(T) * AC(J,IC)
          RIM( 2 ) = SIN(T) * BC(J,IC)
          RIM( 3 ) =          ZC(J,IC)
          DOT = RIM(1) * ( RIM(1) - ANCYL(1) )
     +              + RIM(2) * ( RIM(2) - ANCYL(2) )
C!!!
C!!! If (the last dot product) * (this dot product) < 0, then that is
C!!! where our dot sign goes through zero.
C!!!
          IF ( T .EQ. 0.0 ) THEN
           LSTDOT = DOT
           FNDONE = .FALSE.
          ENDIF
            IF ( SIGN(1.0,DOT) * SIGN(1.0,LSTDOT) .LT. 0.0 ) THEN
           IF( .NOT. FNDONE ) THEN
            DOTMIN( 1, J ) = T
              FNDONE = .TRUE.
           ELSE
            DOTMIN( 2, J ) = T
           ENDIF
          ENDIF
            LSTDOT = DOT
C!!!
C!!! Calculate theta & phi as we go around the rim.
C!!! Transform the rim point into ref. coord.system.
C!!! Find vector from source to rim.
C!!!
            CALL CYLROT( XYZ, RIM, -2, IC )
C!!!
C!!!  Convert from the reference coordinate system to the pattern
C!!!  coordinate system.
C!!!
          XPQ(1) = XYZ(1) - ANTENN(1)
          XPQ(2) = XYZ(2) - ANTENN(2)
          XPQ(3) = XYZ(3) - ANTENN(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) )
C!!!
C!!! Calculate angles representing border of rim and do branch test
C!!! on the phi angle.
C!!!
          THETAR = BTAN2( XPY, XPC(3) )
          PHIR = BTAN2( XPC(2), XPC(1) )
          IF ( PHIR .LT. PH1-0.5*RESPH ) PHIR = TPI + PHIR
C!!!
C!!! Define pixel location.
C!!!
              THETAI = INT( (THETAR - THET1) / RESTH + 0.5 ) + 1
              PHII = INT( (PHIR - PH1) / RESPH + 0.5 ) + 1
C!!!
C!!! Put the character into the output buffer at the proper position.
C!!! Test if indices fall within specified window.
C!!!
```

```
          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!!!
C!!! Reduplicate a wrapped-around character.
C!!!
        IF( (PHII .EQ. 1) .AND. ABS(PH2-PH1-TPI) .LE. RESPH) THEN
         OUTBUF( THETAI, ROWS ) = CHAR(7)
        ENDIF
300   CONTINUE
200    CONTINUE
C!!!
C!!! Before rasterizing, connect the "dotmins".
C!!! A sneakey trick is pulled here.  Instead of transforming every
C!!! increment of the dotmin points, only the two end points are
C!!! transformed, then theta & phi are calculated for each increment.
C!!! This is valid because the line which connects the two points on
C!!! the rims of the cylinders are straigt lines in both RCS and cyl
C!!! coord systems.  Note that this gizmo assumes that you are never
C!!! inside of a cylinder, or your dotmins(K,) probably get crossed
C!!! resulting in an inside-out or bowtie-shaped cylinder.
C!!!
        DO 400 K=1, 2
         DO 500 J=1, NEC(IC)-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, -2, 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 )
C!!!
C!!! This MAGME is analogous to the one in DOPLA except it
C!!! works with psudeo-sides, so the name is somewhat misleading.
C!!!
          MAGME = SQRT(
     +               ( RIM2(1) - RIM1(1) )**2 +
     +               ( RIM2(2) - RIM1(2) )**2 +
     +               ( RIM2(3) - RIM1(3) )**2 )
          T = 0.0
50        IF ( T .GT. 1.0 ) GOTO 600
C!!!
C!!!
C!!! These functions compute the theta/phi associated with a given point
C!!!  along a cylinder psuedo-edge as a function of T (See DOPLA.)
C!!! The variables XYZ and RIM are re-used for multiple purposes here.
C!!!
C!!!  Find vector from source to rim.
C!!!  Convert from the reference coordinate system to the pattern
C!!!  coordinate system
C!!!
         XPQ(1) = ( RIM2(1)-RIM1(1) )*T + RIM1(1)-ANTENN(1)
         XPQ(2) = ( RIM2(2)-RIM1(2) )*T + RIM1(2)-ANTENN(2)
         XPQ(3) = ( RIM2(3)-RIM1(3) )*T + RIM1(3)-ANTENN(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))
C!!!
C!!! Define the angles representing the projection of the curved sides
```

```
C!!! and do a branch cut test on phi.
C!!!
          THETAR = BTAN2( XPY, XPC(3) )
          PHIR = BTAN2( XPC(2), XPC(1) )
          IF ( PHIR .LT. PH1-0.5*RESPH ) PHIR = TPI + PHIR
C!!!
C!!! Define pixel location and put character in appropriate spot.
C!!!
           THETAI = INT( (THETAR - THET1) / RESTH + 0.5 ) + 1
           PHII = INT( (PHIR - PH1) / RESPH + 0.5 ) + 1
C!!!
C!!! Check if angles fall within window.
C!!!
            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!!!
C!!! Reduplicate a wrapped-around character.
C!!!
          IF( (PHII .EQ. 1) .AND. ABS(PH2-PH1-TPI) .LE. RESPH) THEN
           OUTBUF( THETAI, ROWS ) = CHAR(7)
          ENDIF
            T = T + MIN( 0.99, (XPY*ALPH/MAGNE + 1.E-7) )
         GOTO 50
 600       CONTINUE
 500      CONTINUE
 400     CONTINUE
C!!!
C!!! Now do an area fill on the object just outlined.
C!!! Tell SCAN that this is a CYLINDER by using a "2".
C!!!
          DO 700 PHII = 1, ROWS
           CALL SCAN( IC, OUTBUF(1,PHII), PHII, FILL, 2 )
 700      CONTINUE

        RETURN
        END
```

SUBROUTINE DOPLAS

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

```
C---------------------------------------------------------------------
      SUBROUTINE DOPLAS
      INCLUDE 'SHACOM.FOR'
C!!!
C!!! This subroutine processes each plate one at a time.  The
C!!! highlighting logic is contained here.
C!!!
C!!! Do the plates one at a time, then do the plate that was supposed to
C!!! be highlighted last.
C!!!
       IF     ( FILPNM .GT. 0 ) THEN
        DO 1 MP = 1, MPX
         IF ( MP .NE. FILPNM ) CALL DOPLA( MP, FILCHR )
 1      CONTINUE
        CALL  DOPLA( FILPNM, FILCHP )
C!!!
C!!! Fill with a different character for each plate.
C!!!
       ELSEIF ( FILPNM .LT. 0 ) THEN
        DO 2 MP = 1, MPX
         CALL DOPLA( MP, CHAR( MP+ICHAR( '@' ) ) )
 2      CONTINUE
C!!!
C!!! Fill everything with the main background character.
C!!!
       ELSE
        DO 3 MP = 1, MPX
         CALL DOPLA( MP, FILCHR )
 3      CONTINUE
       ENDIF

      RETURN
      END
```

SUBROUTINE DOPLA

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

```
C------------------------------------------------------------------------
        SUBROUTINE DOPLA( IP, FILL )
        INCLUDE 'SHACOM.FOR'
        CHARACTER FILL
        INTEGER
       + IP, INT,
C! Truncate to the nearest integer.

       +        THETAI, PHII
C! Local indicies into char array.

        REAL
       +        T,
C! Parametric increment parameter.

       +        THETAR, PHIR,
C!Theta & phi in radians.

       +        MAGME,
C! Length of side ME.

       +        XPY,
C! temporary variable

       + XPQ(3),
C! Source to edge in ref coords

       + XPC(3),
C! Source to edge in pat coords

       +        BTAN2, SQRT, ABS
C! Miscellaneous functions.
C!!!
C!!! Loop through incrementaly along edges.
C!!!
         DO 200 ME=1, MEP( IP )
                NEXTME = MOD( ME, MEP(IP) ) + 1
           MAGME  = VMAG( ME, IP )
           T = 0.0
50         IF ( T .GT. 1.0 ) GOTO 100
C!!!
C!!! These functions compute the theta/phi associated with a given
C!!! point along an edge between two corners ME and NEXTME as a
C!!! function of T.  T varies from 0 to 1 and is adjusted to keep
C!!! within a safe and efficient excursion at all times.
C!!!
C!!!  Convert from the reference coordinate system to the pattern
C!!!  coordinate system
C!!!
          XPQ(1)=(XX(1,NEXTME,IP)-XX(1,ME,IP))*T+XX(1,ME,IP)-ANTENN(1)
          XPQ(2)=(XX(2,NEXTME,IP)-XX(2,ME,IP))*T+XX(2,ME,IP)-ANTENN(2)
          XPQ(3)=(XX(3,NEXTME,IP)-XX(3,ME,IP))*T+XX(3,ME,IP)-ANTENN(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) )
C!!!
C!!! Define the angles representing the projection of the curved sides
C!!! and do a branch cut test on phi.
C!!!
        THETAR = BTAN2( XPY,    XPC(3) )
        PHIR   = BTAN2( XPC(2), XPC(1) )

          IF ( PHIR .LT. PH1-0.5*RESPH ) PHIR = TPI + PHIR
C!!!
C!!! Define pixel location and put the a character in the appropriate
C!!! spot.
C!!!
         THETAI = INT( (THETAR - THET1) / RESTH + 0.5 ) + 1
         PHII   = INT( (PHIR - PH1)     / RESPH + 0.5 ) + 1
C!!!
C!!! Check if angles fall within window.
C!!!
          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!!!
C!!! Reduplicate a wrapped-around character.
C!!!
        IF( (PHII .EQ. 1) .AND. ABS(PH2-PH1-TPI) .LE. RESPH) THEN
         OUTBUF( THETAI, ROWS ) = CHAR(7)
        ENDIF
C!!!
C!!! Put an upper bound on the increment for the case when the line
C!!! segment is very short or the distance to the segment is great.
C!!! In the degenerate case (on the Z-axis) prevent a potential infinite
C!!! loop by putting a lower bound on delta-t (ie by always adding at
C!!! least a very small number to T.)
C!!!
          T = T + MIN( 0.99, (XPY*ALPH/MAGME + 1.E-7) )
          GOTO 50
100      CONTINUE
200     CONTINUE
C!!!
C!!! Now do an area fill on the object just outlined.
C!!! Tell SCAN that this is a plate by using a "1".
C!!!
        DO 300 PHII = 1, ROWS
         CALL SCAN( IP, OUTBUF(1,PHII), PHII, 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.

```
C-----------------------------------------------------------------------
      SUBROUTINE GEOM
C!!!
C!!!         THIS ROUTINE COMPUTES ALL THE GEOMETRY ASSOCIATED
C!!!  WITH FIXED PLATE STRUCTURE,SUCH AS EDGE UNIT VECTORS,
C!!!  PLATE NORMALS,SHADOWED PLATES,ETC.
C!!!
      INCLUDE 'SHACOM.FOR'
      DIMENSION IHIT(NEX),XII(3),XIN(3),VI(3),XC(3),XSI(3),XSII(3)
      DIMENSION XOB(3),XDC(3),VTCP(2),BTCP(4),VTCN(2),BTCN(4),DS(3)
      DIMENSION VVO(3),VVN(3),VVB(3),VVH(3),XBT(3),PVTRN1(3)
      LOGICAL LSURF,LNPL,LTRN1
      LOGICAL LSHD,LSTD,LSTS,LCTD,LHCT,LHIT
      LOGICAL LGRND,LIHD,LDEBUG,LTEST,LWARN
      COMMON/TEST/LDEBUG,LTEST,LWARN
      COMMON/LIMIT/SML,SMLR,SMLT,BIG
      COMMON/WAVE/WK,WL
      COMMON/LSHDP/LSTS,LSTD(NEX)
      COMMON/GROUND/LGRND,MPXR
      IF(LDEBUG) WRITE(6,667)
667   FORMAT(/,' DEBUGGING GEOM SUBROUTINE')
C!!!  DETERMINAION OF V,VN,AND VP UNIT VECTORS FOR EDGE-FIXED
C!!!  COORDINATE SYSTEM
C!!!  STEP THRU PLATES
      DO 100 MP=1,MPXR
      MEX=MEP(MP)
C!!!  STEP THRU EDGES
      DO 15 ME=1,MEX
      MME=ME+1
      IF(MME.GT.MEX) MME=1
      VM=0.
C!!!  CALCULATE EDGE UNIT VECTOR V AND EDGE LENGTH VMAG
      DO 10 N=1,3
      V(N,ME,MP)=XX(N,MME,MP)-XX(N,ME,MP)
10    VM=VM+V(N,ME,MP)*V(N,ME,MP)
      VMAG(ME,MP)=SQRT(VM)
      DO 11 N=1,3
11    V(N,ME,MP)=V(N,ME,MP)/VMAG(ME,MP)
15    CONTINUE
      IF(.NOT.LDEBUG) GO TO 991
      DO 992 ME=1,MEX
      WRITE(6,*) (V(N,ME,MP),N=1,3)
992   CONTINUE
991   CONTINUE
C!!!  CALCULATE PLATE UNIT NORMAL VN
      VN(1,MP)=0.
      VN(2,MP)=0.
      VN(3,MP)=0.
      DO 22 ME=1,MEX
      MV=ME+1
      IF(MV.GT.MEX) MV=1
      VN(1,MP)=VN(1,MP)+V(2,ME,MP)*V(3,MV,MP)-V(2,MV,MP)*V(3,ME,MP)
      VN(2,MP)=VN(2,MP)+V(3,ME,MP)*V(1,MV,MP)-V(3,MV,MP)*V(1,ME,MP)
      VN(3,MP)=VN(3,MP)+V(1,ME,MP)*V(2,MV,MP)-V(1,MV,MP)*V(2,ME,MP)
22    CONTINUE
      VNM=0.
```

```
      DO 20 N=1,3
   20 VNM=VNM+VN(N,MP)*VN(N,MP)
      VNM=SQRT(VNM)
      DO 21 N=1,3
   21 VN(N,MP)=VN(N,MP)/VNM
      IF(LDEBUG) WRITE(6,*) (VN(N,MP),N=1,3)
C!!!  INSURE THAT ALL PLATES ARE FLAT. OTHERWISE ABORT!
C!!!  TAKE DOT PRODUCT OF PLATE NORMAL AND EACH EDGE UNIT VECTOR
      DO 120 ME=1,MEX
      DOT=VN(1,MP)*V(1,ME,MP)+VN(2,MP)*V(2,ME,MP)+VN(3,MP)*V(3,ME,MP)
      ADOT=ABS(DOT)
      IF(ADOT.LT.0.01) GO TO 120
      MEE=ME+1
      IF(MEE.GT.MEX) MEE=1
      WRITE(6,121) MP,MEE,ADOT
  121 FORMAT(' PLATE # ',I2,' IS NOT FLAT! CORNER # ',I2,' HAS ',
     2'PROBLEM.'/' WARP= ',F7.3,' PROGRAM ABORTS IF THE WARP'
     3' IS GREATER THAN 0.03 ******')
      IF(ADOT.GT.0.03) STOP
  120 CONTINUE
C!!!  CALCULATE UNIT BINORMAL VP WHICH IS IN PLATE PLANE
C!!!  AND PERPENDICULAR TO PLATE EDGE
C!!!  TAKE CROSS PRODUCT OF PLATE NORMAL AND EDGE VECTOR
      DO 30 ME=1,MEX
      VP(1,ME,MP)=VN(2,MP)*V(3,ME,MP)-VN(3,MP)*V(2,ME,MP)
      VP(2,ME,MP)=VN(3,MP)*V(1,ME,MP)-VN(1,MP)*V(3,ME,MP)
   30 VP(3,ME,MP)=VN(1,MP)*V(2,ME,MP)-VN(2,MP)*V(1,ME,MP)
      IF(.NOT.LDEBUG) GO TO 993
      DO 994 ME=1,MEX
  994 WRITE(6,*) (VP(N,ME,MP),N=1,3)
  993 CONTINUE
  100 CONTINUE
      RETURN
      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.

```
C----------------------------------------------------------------------
      SUBROUTINE INITGF
      INCLUDE  'SHACOM.FOR'
C!!!
C!!!  This subroutine initializes some graphics stuff.
C!!!  Its function is to initialize things from one plot to the next,
C!!!  but within the context of a single session.
C!!!
       INTEGER
     +        I, J, INT
C!!!
C!!!  Clear the character buffer.
C!!!
      DO 10 J=1, MAXROW
       DO 10 I=1, MAXCOL
10      OUTBUF( I, J ) = ' '
C!!!
C!!! The number of rows & columns needed for internal representation is
C!!! calculated from the user-selected (or defaulted) angular ranges of
C!!! interest combined with the desired resolution in rads/pixel
C!!!
      ROWS   = INT( (PH2   - PH1)   / RESPH + 0.5 ) + 1
      COLS   = INT( (THET2 - THET1) / RESTH + 0.5 ) + 1
C!!!
C!!! Calculate some parameters needed by the dynamic T increment
C!!! algorithms.  The maximum allowable angular excursion is the
C!!! smaller of the number of radians in a single pixel of either theta
C!!! or phi.
C!!!
      ALPH = MIN( RESTH, RESPH )
C!!!
      RETURN
      END
```

SUBROUTINE PLAINT

This routines determines if a given ray strikes a plate.

```
C
C----------------------------------------------------------------------
      SUBROUTINE PLAINT(XIS,D,DHIT,MH,LHIT)
C!!!
C!!!  DOES RAY HIT PLATE.IF MH=0 ALL PLATES ARE CHECKED.
C!!!  IF MH=-MP THEN ONLY MP CHECKED AND SOURCE POSITION
C!!!  MOVED TO HIT POSITION IF RAY HITS MP.
C!!!  IF MH=MP,THEN ALL PLATES OTHER THAN MP ARE CHECKED.
C!!!
      INCLUDE 'SHACOM.FOR'
      DIMENSION XIS(3),D(3),XT(3),PVTRN(3)
      LOGICAL LHIT,LPLA,LCYL,LSTS,LSTD,LTRN
      LOGICAL LGRND,LDEBUG,LTEST,LWARN
      COMMON/TEST/LDEBUG,LTEST,LWARN
      COMMON/LIMIT/SML,SMLR,SMLT,BIG
      COMMON/LPLCY/LPLA,LCYL
      COMMON/HITPLT/MPH
      COMMON/GROUND/LGRND,MPXR
      LHIT=.FALSE.
      DHIT=0.
      IF(.NOT.LPLA) RETURN
C!!!  STEP THRU PLATES
      DO 60 MPP=1,MPXR
      MP=MPP
      IF(MP.EQ.MH) GO TO 60
      IF(MH.LT.0) MP=IABS(MH)
C!!!  IF TOTAL SHADOWING ALGORITHM IS BEING USED, HAS PLATE MP
C!!!  SHADOWED EVERY RAY TESTED?
C%%%%%        IF(LSTS.AND..NOT.LSTD(MP)) GO TO 60
      MEX=MEP(MP)
      AN=0.
      DO 5 N=1,3
5     AN=AN+(XIS(N)-XX(N,1,MP))*VN(N,MP)
      DN=D(1)*VN(1,MP)+D(2)*VN(2,MP)+D(3)*VN(3,MP)
C!!!  DOES RAY PASS THRU PLATE PLANE?
      IF(AN*DN.GE.0.) GO TO 50
      DO 10 N=1,3
C!!!  CALCULATE POINT WHERE RAY INTERSECTS PLATE PLANE
10    XT(N)=XIS(N)-AN*D(N)/DN
      IF(MP.EQ.MPXR.AND.LGRND) GO TO 11
      DBT=0.
C!!!  IS HIT POINT ON PLATE?
      DO 30 ME=1,MEX
      MME=ME+1
      IF(MME.GT.MEX) MME=1
      RD=0.
      DO 20 N=1,3
20    RD=RD+(XX(N,ME,MP)-XT(N))*(XX(N,MME,MP)-XT(N))
      CP=VN(1,MP)*((XX(2,ME,MP)-XT(2))*(XX(3,MME,MP)-XT(3))
     2-(XX(3,ME,MP)-XT(3))*(XX(2,MME,MP)-XT(2)))
      CP=CP+VN(2,MP)*((XX(3,ME,MP)-XT(3))*(XX(1,MME,MP)-XT(1))
     2-(XX(1,ME,MP)-XT(1))*(XX(3,MME,MP)-XT(3)))
      CP=CP+VN(3,MP)*((XX(1,ME,MP)-XT(1))*(XX(2,MME,MP)-XT(2))
     2-(XX(2,ME,MP)-XT(2))*(XX(1,MME,MP)-XT(1)))
      DBI=BTAN2(CP,RD)
      DBT=DBT+DBI
30    CONTINUE
```

```
      IF(ABS(DBT).LT.PI) GO TO 50
C!!!  CALCULATE DISTANCE TO HIT (DHIT=SHORTEST DHT)
11    DHT=0.
      DO 40 N=1,3
40    DHT=DHT+(XT(N)-XIS(N))*(XT(N)-XIS(N))
      DHT=SQRT(DHT)+SMLR
      IF(LHIT.AND.(DHT.GT.DHIT)) GO TO 60
      LHIT=.TRUE.
      DHIT=DHT
      MPH=MP
      IF(MH.GE.0) GO TO 60
      DO 45 N=1,3
C!!!  MOVE HIT POSITION AN INCREMENT TOWARDS SIDE OF PLATE
C!!!  WHICH SOURCE LIES ON
45    XIS(N)=XT(N)-SIGN(SMLR,AN)*VN(N,MP)
      GO TO 61
50    CONTINUE
      IF(MH.LT.0) GO TO 61
C!!!  IF TOTAL SHADOWING ROUTINE IS BEING USED, INDICATE
C!!!  THAT PLATE MP DOES NOT SHADOW SOURCE
C%%%%%        IF(LSTS) LSTD(MP)=.FALSE.
60    CONTINUE
61    IF(.NOT.LTEST) GO TO 62
      WRITE(6,63)
63    FORMAT(/,' TESTING PLAINT SUBROUTINE')
      WRITE(6,*) XIS
      WRITE(6,*) D
      WRITE(6,*) DHIT,MH,LHIT
62    RETURN
      END
```

SUBROUTINE SCAN

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

```
C----------------------------------------------------------------------
      SUBROUTINE SCAN ( OBJ, LINE, PHII, FILL, TYPE )
C!!!
C!!! This subroutine "rasterizes" a line in the character buffer
C!!! according to its shading requirements.
C!!! The fill character is used to fill the
C!!! line in.  The line is declared larger character string in this
C!!! subroutine than in the calling routine.  This can be "hardwired"
C!!! if it causes problems on other machines.
C!!!
      INCLUDE 'SHACOM.FOR'
      CHARACTER*500   LINE
      CHARACTER*1     FILL
      INTEGER                 PTR, OBJ
      INTEGER                 LSTPTR
      INTEGER                 SCANC
      INTEGER                 SPANC
      INTEGER                 PHII
      INTEGER                 TYPE
      REAL            DHIT
      REAL            D(3)
      REAL            DP(3)
      REAL            XIS(3)
      REAL            THETA,PHI
      LOGICAL                 EOL
      LOGICAL                 LHIT, LHIT1
      COMMON  /SCNCMN/ PTR, EOL
C!!!
C!!! Initialize local variables.
C!!!
      PTR = 1
      LSTPTR = 1
      EOL = PTR .GT. COLS
C!!!
C!!! Until the end of the line is scanned do ...
C!!!
100   IF (.NOT. EOL) THEN
C!!!
C!!!  Locate the first occurence of CHAR 7, 8, or EOL.
C!!!
       PTR = SCANC( LINE )
C!!!
C!!! If plaint says it's a miss, update the last-pointer, span, scan,
C!!! fill.  Otherwise, fill in the characters between the pointers.
C!!! Define the "source point" AS the location of the antenna,
C!!! and see if our plate shadows the direction of the midpoint of the
C!!! scan.
C!!!
          THETA = (0.5*FLOAT(PTR+LSTPTR)-1.0)*RESTH+THET1
          PHI   = (PHII-1)*RESPH+PH1
        DP(1) = SIN(THETA)*COS(PHI)
        DP(2) = SIN(THETA)*SIN(PHI)
        DP(3) = COS(THETA)
        D(1)  = DP(1)*VPC(1,1) + DP(2)*VPC(2,1) + DP(3)*VPC(3,1)
```

```
      D(2)  = DP(1)*VPC(1,2) + DP(2)*VPC(2,2) + DP(3)*VPC(3,2)
      D(3)  = DP(1)*VPC(1,3) + DP(2)*VPC(2,3) + DP(3)*VPC(3,3)
C!!!
C!!! This must be done due to the behavior of plaint modifying XIS.
C!!!
      XIS(1) = ANTENN(1)
      XIS(2) = ANTENN(2)
      XIS(3) = ANTENN(3)
C!!!
C!!! Now do a case depending on what type of object we test for
C!!! shadowing.
C!!!
C!!! 1 = plate
C!!! 2 = elliptic cylinder
C!!!
      GOTO (1,2) TYPE
C!!!
C!!! The object is a plate.
C!!!
1     CALL PLAINT( XIS, D, DHIT, -OBJ, LHIT )
      GOTO 999
C!!!
C!!! The object is a cylinder.  Test endcaps and cylinder bodies.
C!!!
2     CALL CAPINT( XIS, D, DHIT, 0, LHIT1, -OBJ )
      IF (.NOT. LHIT1) CALL CYLINT(XIS,D,DHIT, LHIT, .FALSE., -OBJ)
      LHIT = LHIT .OR. LHIT1
      GOTO 999
C!!!
C!!! Take the appropriate action in the buffer.
C!!!
999   IF ( .NOT. LHIT ) THEN
         LSTPTR = PTR
       PTR = SPANC( LINE )
       DO 300 LSTPTR = LSTPTR, PTR-1, 1
          LINE( LSTPTR:LSTPTR ) = FILL
300      CONTINUE
       LSTPTR = PTR
      ELSE
       PTR = SPANC( LINE )
       DO 400 LSTPTR = LSTPTR, PTR-1, 1
          LINE( LSTPTR:LSTPTR ) = FILL
400      CONTINUE
       LSTPTR = PTR
      ENDIF
C!!!
C!!! End UNTIL
C!!!
      GOTO    100
      END IF
C!!!
      RETURN
      END
```

FUNCTION SCANC/SPANC

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

```
C---------------------------------------------------------------------
C!!!
C!!! The following functions span/scan characters.  That is, they
C!!! return the position of next character in LINE which does or does
C!!! not match the specficied character. They also
C!!! terminate the scan/span at the end of the line.
C!!!
      INTEGER                  FUNCTION SCANC( LINE )
      INCLUDE                  'SHACOM.FOR'
      CHARACTER*(*)   LINE
      INTEGER                  PTR
      LOGICAL                  EOL
      COMMON  /SCNCMN/ PTR, EOL
C!!!
C!!! Until a character matching CHARAC is found, advance the pointer.
C!!!
      SCANC = PTR
200     IF (.NOT. (EOL .OR.
     +                         ( LINE(SCANC:SCANC) .EQ. CHAR(7) ) )) THEN
        SCANC = SCANC + 1
        EOL = SCANC .GT. COLS
        GOTO 200
      ENDIF
C!!!
C!!! End UNITL
C!!!
      RETURN
      END
C!!!
C---------------------------------------------------------------------
C!!!
      INTEGER                  FUNCTION SPANC( LINE )
      INCLUDE                  'SHACOM.FOR'
      CHARACTER*(*)   LINE
      INTEGER                  PTR
      LOGICAL                  EOL
      COMMON  /SCNCMN/ PTR, EOL
C!!!
C!!! Until a character NOT matching ASCII 7 is found, advance the
C!!! pointer.
C!!!
      SPANC = PTR
200     IF (.NOT. (EOL .OR.
     +                         ( LINE(SPANC:SPANC) .NE. CHAR(7) ) )) THEN
        SPANC = SPANC + 1
        EOL = SPANC .GT. COLS
        GOTO 200
      ENDIF
C!!!
C!!! End UNITL
C!!!
      RETURN
      END
```

SUBROUTINE WRTOUT

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

```
C----------------------------------------------------------------------
      SUBROUTINE WRTOUT
      INCLUDE        'SHACOM.FOR'
      INTEGER        I, J, COLI, COLF
C!!!
C!!! This subroutine writes the formatted output buffer to the output
C!!! file.  Start the output on a new page, and calculate a header
C!!! based on the specified pixel resolution.
C!!!
C!!!  Unit 7 is the main (ASCII) output file.
C!!!
C!!! Initilize the width of the map to be printed.
C!!!
      COLI = 1
      COLF = 91
      IF(COLF .GT. COLS) COLF = COLS
C!!!
C!!! Print map.
C!!!
20    WRITE( 7, 100 ) ( ANTENN(I), I=1, 3, 1 ), INPFIL
      WRITE( 7, 200 ) ( (RESTH*(I-1) + THET1)*DPR , I= COLI, COLF, 10)
      WRITE( 7, 250 ) ( '+', I= COLI, COLF, 10)
      DO 50 J = 1, ROWS
50    WRITE( 7, 300 ) ( RESPH*(J-1) + PH1 )*DPR,
     +                      ( OUTBUF(I,J), I= COLI, COLF )
C!!!
C!!! If the map does not fit on the line printer width,
C!!! then split it onto another set of pages.
C!!!
      IF(COLF .LT. COLS) THEN
        COLI = COLF
        COLF = COLF + 90
        IF(COLF .GT. COLS) COLF = COLS
        GO TO 20
      ENDIF
C!!!
C!!! Have internal parameters available in degrees.
C!!!
      THET1D = THET1  *DPR
      THET2D = THET2  *DPR
      RESTHD = RESTH  *DPR
      PH1D   = PH1    *DPR
      PH2D   = PH2    *DPR
      RESPHD = RESPH  *DPR
C!!!
C!!! Unit 10 is a generic sort of binary output which can be plotted
C!!! anywhere.  Place a little header info at the front of the file.
C!!!
      WRITE( 10 ) COLS, THET1D, THET2D, RESTHD
      WRITE( 10 ) ROWS, PH1D,   PH2D,   RESPHD
C!!!
C!!! Dump only that part of the buffer which pertains to this plot.
C!!!
      DO 10 J = 1, ROWS
       DO 10 I = 1, COLS
10      WRITE (10) OUTBUF( I, J )
C!!!
```

```
C!!! Output stuff is complete.
C!!!
      RETURN
C!!!
C!!! Format statements.
C!!!
100   FORMAT( '1', 8X, 'ANTENNA (RCS) = ', '(', 2(F8.4, ','),
     +        F8.4, '  ) IN METERS', 5X, 'INPUT SET: ', A42, / )
200   FORMAT( T60, 'THETA  (DEGREES)',/, 9X, 11( 4X, F6.2) )
250   FORMAT( 9X, 'PHI', 4X, A, 10( 9X, A) )
300   FORMAT( 6X, F7.2, 3X, 101A )
      END
```

Include file

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

```
C!!!
C!!!  COMMON declerations...
C!!!
      COMMON /PIS/
     +        PI,
     +        TPI,
     +        DPR,
     +        RPD
C!!!
C+++  MAXIMUM DIMENSION FOR PLATES
      INTEGER            NPX
      PARAMETER (NPX=75)
C+++  MAXIMUM DIMENSION FOR PLATE EDGES
      INTEGER            NEX
      PARAMETER (NEX=12)
C+++  MAXIMUM DIMENSION FOR CYLINDERS
      INTEGER            NCX
      PARAMETER (NCX=5)
C+++  MAXIMUM DIMENSION FOR CYLINDER RIMS
      INTEGER            NNX
      PARAMETER (NNX=10)
C+++  MAXIMUM DIMENSION FOR ROWS (PHI)
      INTEGER            MAXROW
      PARAMETER (MAXROW=361)
C+++  MAXIMUM DIMENSION FOR COLUMNS (THETA)
      INTEGER            MAXCOL
      PARAMETER (MAXCOL=181)
C!!!
      COMMON /GEOPLA/
     +        XX      (3,NEX,NPX),
     +        V       (3,NEX,NPX),
     +        VP      (3,NEX,NPX),
     +        VN      (3,NPX),
     +        MEP     (NPX),
     +        MPX
C!!!
      COMMON /GEOMEL/
     +        AC      (NNX,NCX),
     +        BC      (NNX,NCX),
     +        ZC      (NNX,NCX),
     +        TCR     (NNX,NCX),
     +        XCL     (3,NCX),
     +        VCL     (3,3,NCX),
     +        NEC     (NCX),
     +        MCX
C!!!
      COMMON /EDMAG/ VMAG(NEX,NPX)
C!!!
      COMMON /SHADWN/ COLS, ROWS, ANTENN(3),CTROID(3),
     +                      MP,ME,NEXTME,MC,
     +                      THET1,THET2,PH1,PH2,RESTH,RESPH,ALPH,
     +                      UNIT(3),TRS(3),VRS(3,3),IUNIT,UNITF,UNITS,UN
     +                      THZP,PHZP,THXP,PHXP,FILPNM,FILCNM
      COMMON /SHADWC/ INPFIL,OUTBUF(MAXCOL,MAXROW),
     +                FILCHC,FILCHP,FILCHR
C!!!
      COMMON /PATCUT/ VPC(3,3)
C!!!
```

```
C!!!  The first set of declarations is the stuff in /SHADOW/ common bloc
C!!!
      INTEGER
     +        MP, ME, NEXTME, MC,
C! Plate#/edge#/cyl# variables.
     + FILPNM, FILCNM,
C! Plate and cyl numbers for special filling
     +        COLS,
C! The size of the array subsection determined
     +        ROWS
C! by internal resolution requirements.

      REAL
     +        CTROID,
C! A geometric center of the object in question.
     +        ANTENN,
C! The antenna location in Ref Coord. System.
     +        THET1,
C! The lower theta end of the range.
     +        THET2,
C! The higher theta end of the range.
     +        PH1,
C! The lower phi end of the range.
     +        PH2,
C! The higher phi end of the range.
     +        RESTH,
C! The desired theta/phi resolution
     +        RESPH,
C! in units of radians/pixel.
     +        ALPH
C! Maximum allowed angular excursion.

      CHARACTER
     +        OUTBUF*1,
C! The output buffer which is displayed.
     +        INPFIL*63,
C! The filename of the input set.
     + FILCHC,
C! special fill character for cylinders
     + FILCHP,
C! special fill character for everything else
     + FILCHR
C! special fill character for plates
      DATA    FILCHC, FILCHP, FILCHR / 'C', 'P', 'X' /
C!!!
C!!!  From the /PIS/ COMMON block...
C!!!
      REAL PI, TPI, DPR, RPD
C!!!
C!!!  From the /GEOPLA/ COMMON block...
C!!!
      INTEGER
     +        MEP,
C! Number of edges per plate
     +        MPX
C! Total number of plates
      REAL
     +        XX,
C! The array of plate corners
     +        V,
C! Edge unit vectors
     +        VP,
C! Edge unit binormals
     +        VN
```

```
C! Unit normal for each plate
C!!!
C!!!  From the /GEOMEL/ COMMON block...
C!!!
      INTEGER
     +          NEC,
C! Number of sections per cylinder
     +          MCX
C! Total number of cylinders
      REAL
     +          AC,
C! Elliptic parameter along x-axis
     +          BC,
C! Elliptic parameter along y-axis
     +          ZC,
C! Cylinder endcaps in cyl coord sys
     +          TCR,
C! Angle endcap makes with positive z axis
     +          XCL,
C! Cyl coord sys origin
     +          VCL
C! Definition of cyl coord sys
C!
      INTEGER
     +                    IUNIT
      REAL
     +                    UNITF,
     +                    UNITS,
     +                    UNITN,
     +                    UNIT,
     +                    TRS,
     +                    THZP,PHZP,THXP,PHXP,
     +                    VRS,
     +    VPC,
     +    VMAG
      DATA UNIT/1.,.3048,0.0254/
C!
C!!!+
C!!! The following common block is for VMS/SMG$ software only.
C!!!
      INTEGER                                  KBDID, KEYTBL
      COMMON  /TERCOM/                         KBDID, KEYTBL
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 Definiton Utility (CDU) which defines the available interactive commands.

```
!++
!
! File: SHACMD.CLD Edit: AAA1001
!
MODULE COMMAND_TABLES
IDENT /SHACMD 01-001/
!+
! FACILITY: Shadow
!
! ABSTRACT:
!
!  This is the command language definiton source for the SHADOW
!  program.  It defines the interactive command interface under
!  the VAX/VMS operating system.
!
! AUTHOR: Laszlo Takacs
!
! CREATED: 1-NOV-1985
!
! MODIFIED BY:
!  1-000 - Original.  AAA  1-NOV-1985
!  1-001 - Laszlo Takacs  20-DEC-1985
!  Added support for the SET FILL command and rearranged
!  the SET PLATE and SET CYLINDER commands.
!-
!
! Show syntax
!
Define syntax  show_fil_syntax      routine show_fil
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_ant_syntax      routine show_ant
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_ant_syntax routine set_ant
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=$file, required ),
prompt="filename"
qualifier plottable, default
```

```
qualifier printable, batch
qualifier echoing, default

Define syntax set_inp_syntax routine set_inp
parameter p1 value( required )
parameter p2 value( type=$file, required ),
prompt="input set"

Define syntax set_fil_syntax routine set_fil
parameter p1 value(required)
parameter p2 value(default="X"),prompt="character"
qualifier plate value(required,list), nonnegatable
qualifier cylinder value(required,list), nonnegatable
qualifier sequential nonnegatable, syntax=sequential
disallow any2 ( plate, cylinder, sequential )

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

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

!Define syntax set_cyl_syntax          routine set_cyl
!  parameter p1 value( required )
!  parameter p2 value( required ), prompt="cyl number"
!  parameter p3 value( default="C" ), prompt="character"
!  qualifier all     syntax=set_placyl_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
!
! Type definitons.
!
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_inp_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
```

```
keyword keypad_mode,    syntax = set_key_syntax, negatable

Define type show_types
   keyword fill_character,    syntax = show_fil_syntax
 !  keyword plate,    syntax = show_fil_syntax
 !  keyword cylinder,    syntax = show_fil_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 p1, value( required, type=set_types ),
prompt = "Set what"

Define verb show
  parameter p1, value( required, type=show_types ),
prompt = "Show what"

Define verb help routine help_command
      parameter p1, 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 p1, value( type=$rest_of_line )

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

!
! End of file SHACMD.CLD.
!
!--
```

Keypad initialization file

This file defines the initial keypad assignments for the interactive program at run time. It may be modified to allow customizing of the keypad interface.

```
!+
!  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"/termina
Def/key/echo/if=default  KP5  "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=(1,X) "
Def/key/echo/if=default  KP3    "Set fill /Sequential"/ter
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"/termin
Def/key/echo/if=gold     KP5  "Show units" /terminate
Def/key/echo/if=gold     KP6  "Show coordinate"/terminate
Def/key/echo/if=gold     COMMA  "Show pattern" /terminate
Def/key/echo/if=gold     KP1   "Show fill" /terminate
Def/key/echo/if=gold     KP2   "Set  fill /cylinder=(1,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.
```

```
!
Def/key/term/echo PERIOD "Exit"
Def/key/term/echo ENTER   ""
!
! End of SHADOW.KPD
!-
```

Chapter 11

VAX Implementation

This chapter describes the VAX/VMS implemetation 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 seperated 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 initalization file used by the interactive program to equivalence certain functions to keys of the user's choice.

SHAPLT.COM This is a DCL command procedure invoking the NCAR graphics plotting software.

SHAPLT.FOR This is the FORTRAN program which reads the output produced by the code and calls appropriate NCAR routines to make a plot.

LABEL.DAT This file is read by the SHAPLT program in order to label the NCAR plots.

11.2 Running the Code

In order to run the code on VMS, the executable file created by the SHABLD procedure is necessary. The program is then run with the dcl RUN command.

A typical interactive session with the program might consist of the following elements in their approximate order of execution.

OUTPUT FILES Establish a set of output files with the SET OUTPUT command. The output files are of three types. Using the qualifiers of the SET OUTPUT command, any desired combination of output files may be generated.

PROCESS AN INPUT Issue a SET INPUT command which reads the geometry from the specified file. In order for the program to process input sets, this command must be issued prior to any mapping commands. This command is usually executed once per session.

DEFINE A WINDOW Using the SET WINDOW command, establish the angular range of interest. When the program begins, the size of the window is set to the full angular extent of the far-zone sphere. By specifying a smaller angular range, the user examines portions of the geometry in greater detail.

DEFINE A SOURCE With the SET ANTENNA command, establish the location of the source. This command is one of the more frequently entered commands. It applies units and coordinate transformations that apply from the set units and set coordinates command.

HIGHLIGHT ITEMS With the SET FILL command, the user may optionally cause parts of the geometry to be marked. This very usefull command may be executed at any time before a SHADOW command.

GENERATE A MAP Cause the generation of a shadow map by issuing a SHADOW command. The shadow command is used after the user has set all desired parameters including the window and the antenna location. Without executing this command, the code does not calculate any shadowing.

REPEAT ANY OF THE ABOVE Perform one or more of the above actions repeatedly to obtain several maps. Most of the commands above may be executed in any order provided that the SHADOW command is executed last.

EXIT Terminate the shadow session with an EXIT command. An acceptable alternate mode of exit is eof, or control-Z.

In order to make life easier by reducing the number of keystrokes required to enter interactive commands, a facility is provided with which the user may associate whole command strings with a single key. When the shadow program begins executing, it loads a set of predefined key definitions from a file. The user may edit this file to customize the keypad definitons to his/her liking. Since the file is loaded automaticly, 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 acsumed that the easiest way to use SHADOW on a non-VAX would be to run it in a non-interactive mode.

The main program in the default version of the file SHADOW.FOR is designed to be used with the non-FORTRAN 77 interactive version. A file called SHADNI.FOR contains a main program designed to be used in a non-interactive mode. It is listed in section 10.2. The main programs can be easily exchanged.

Note that the only other part of the code is this part that is non-FORTRAN 77 is the INCLUDE statement. This has been retained because many computer systems support this statement. It is used to include the lines of code in the named file in the spot that it is called as if the lines had been in that spot. It provides a powerful means of putting commonly defined parameters used throughout the code in one place. In this case, it is used to include the file SHACOM.FOR which contains COMMON blocks and PARAMETER statements that define the dimensions of arrays that store the geometry. If it is desired to increase the number of plates, edges per plate, cylinders, or rims per cylinder, etc; they can be changed in one spot. Please see the listing for this file elsewhere in this manual. The INCLUDE statement can be easily removed by hardwiring the contents of the file SHACOM.FOR into the text at the main program and the subroutines ABSCIN, CAPINT, CYLINT, CYLROT, DOCYLS, DOCYL, DOPLAS, DOPLA, GEOM, INITGF, PLAINT, 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 a 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:

```
      PROGRAM PLTOSU
      DIMENSION XDUM(2), YDUM(2), NC(5)
      INTEGER COLS, ROWS
      CHARACTER*80 LABELS(5), XLAB, YLAB, INF
      CHARACTER*(*)   XFORMA, YFORMA
      BYTE    BYTE
C
C These are character parameters for the plotting output.
C
      PARAMETER                       ( XFORMA = '(F6.1)' )
      PARAMETER                       ( YFORMA = '(F6.1)' )
C
        DATA XLAB /' PHI '/
        DATA YLAB /' THETA '/
        DATA NC   / 5*72 /
C
C Read a header from FOR005.  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', READONLY )
        READ ( 5, FMT='(A)' ) INF
C
C Read the header info from the data file.  Open it unformatted.
C
      OPEN(UNIT=10,FILE=INF,TYPE='OLD',FORM='UNFORMATTED',READONLY)

 13   READ(10,END=9999) COLS,THET1D,THET2D,RESTHD
      READ(10,END=9999) ROWS,PH1D,PH2D,RESPHD

        ISCX = -2
        ISCY = -2
      XMIN = PH1D
      XMAX = PH2D
      YMIN = -THET2D
      YMAX = -THET1D
        NDX = 4
        NTX = 2
        NDY = 4
        NTY = 2
C
C Read the label info for this plot.
C
        READ ( 5, * ) LABELS(1)
        READ ( 5, * ) SX, SY, SZ, SPRX, SPRY, TRX
        READ ( 5, * ) ZTHET, ZPHI, XTHET, XPHI
C
C Format the labels for the plot (via internal write statements.)
C
        WRITE (LABELS(2), 1100) SX, SY, SZ
        WRITE (LABELS(3), 1200) ZTHET, ZPHI, XTHET, XPHI
        WRITE (LABELS(4), 1300) SPRX, SPRY
        WRITE (LABELS(5), 1400) TRX
C
C         CALL INFOPLT(2,XDUM,YDUM,XMIN, XMAX,YMIN,YMAX,ISCX,
C     *               NDX,NTX,ISCY,NDY,NTY,XLAB,5,YLAB,7,
C     *               5,LABELS,NC,0,-1,1)
C
C Define a mapping window from data to plot
C
      CALL SET (
     +                        0.12,
     +                        0.84,
     +                        0.12,
```

```
     +                       0.84,
     +                       XMIN,
     +                       XMAX,
     +                       YMIN,
     +                       YMAX,
     +                       1 )     ! Do a linear-linear plot.
C
C A call to labmod might help the output look nicer.
C
      CALL LABMOD (
     +                       %REF( XFORMA ),
     +                       %REF( YFORMA ),
     +                        LEN( XFORMA ),
     +                        LEN( YFORMA ),
     +                       1,
     +                       1,
     +                       0,
     +                       0,
     +                       0              )
C
C Put labels on plot
C
              XMID=0.5*(XMIN+XMAX)
              YMID=0.5*(YMIN+YMAX)
              XDEL=(XMAX-XMIN)/36.
              YDEL=(YMAX-YMIN)/36.
              XL=XMIN+0.5*XDEL
      DO 100 IL=1,5
              YL=YMAX+(6-IL)*YDEL
100   CALL PWRIT(XL,YL,%REF(LABELS(IL)),NC(IL),1,0,-1)
C
C Define the perimeter of the plot wit a grid.
C
      CALL GRIDL (
     +                       NDX,                     ! Number of MAJOR
     +                       NTX,                     ! Number of MINOR
     +                       NDY,                     ! Number of MAJOR
     +                       NTY )                    ! Number of MINOR
C
C     Theta and Phi Axis Labels
C
              YBOT=YMIN-2.5*YDEL
      CALL PWRIT(XMID,YBOT,%REF(XLAB),5,1,0,0)
              XSID=XMIN-5.0*XDEL
      CALL PWRIT(XSID,YMID,%REF(YLAB),7,1,90,0)
!C
!C Use this call if you don't want grid lines.
!C Define the perimeter of the plot.
!C
!     CALL PERIML (
!     +                      NDX,                      ! Number of MAJOR
!     +                      NTX,                      ! Number of MINOR
!     +                      NDY,                      ! Number of MAJOR
!     +                      NTY )                     ! Number of MINOR
C
        XINC = 1.8
        YINC = 0.9
        ISYM =  1
C
C Loop on rows then on columns.
C
      DO 10 J = 1, ROWS
       X = RESPHD*(J-1)+PHID
```

```
         DO 20 I = 1, COLS
          READ ( 10, END=999 ) BYTE
          IF ( BYTE .NE. 32 ) THEN
          Y = -(RESTHD*(I-1)+THET1D)
C
C            CALL PLTSYM( X, Y, XINC, YINC, ISYM )
C
C Plot the symbol on the page.
C
           CALL PWRIT(
     +                                X,       ! X coordinate
     +                                Y,       ! Y coordinate
     +                                BYTE,    ! The character to plot
     +                                1,       ! Write one character
     +                                0,       ! Use the default size
     +                                0,       ! Use the default orientat
     +                                0 )      ! Use the default centerin
          END IF
 20        CONTINUE
 10       CONTINUE
C
C "Frame" the NCAR output.
C
 999      CALL FRAME
          GOTO 13
C
C Close the input file and stop.
C
 9999     CLOSE ( UNIT=10 )
        CLOSE ( UNIT=5 )
          STOP  'NCAR/Shadow plot completed.'
C
C Formats go down here.
C
 1100     FORMAT ('    ANTENNA LOCATED AT ',2(F7.1,','),F7.1)
 1200     FORMAT ('    ANTENNA ORIENTATION: ',3(F7.1,','),F7.1)
 1300     FORMAT ('    SOLAR PANELS ROTATED ',F7.1,',',F7.1)
 1400     FORMAT ('    THERMAL RADIATORS ROTATED ',F7.1)
          END
```

Listing of sample LABEL.DAT file:

```
FOR010.DAT;
'    SHADOW TEST1  FOR CASE AN5S1'
25. 15. 256.5 0. -52. 0.
0. 0. 90. 0.
'    SHADOW TEST2 FOR CASE AN5S1'
25. 15. 256.5 0. -52. 0.
0. 0. 90. 0.
```

Standard Bibliographic Page

1. Report No. NASA CR-178099	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle MANUAL FOR OBSCURATION CODE WITH SPACE STATION APPLICATIONS		5. Report Date May 1986
		6. Performing Organization Code
7. Author(s) R. J. Marhefka and L. Takacs		8. Performing Organization Report No. 716199-7
9. Performing Organization Name and Address The Ohio State University ElectroScience Laboratory Department of Electrical Engineering 1320 Kinnear Road Columbus, Ohio 43212		10. Work Unit No.
		11. Contract or Grant No. NSG-1498
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546		13. Type of Report and Period Covered Contractor Report
		14. Sponsoring Agency Code 482-54-23-15

15. Supplementary Notes

Langley Technical Monitor: E. M. Bracalente

16. Abstract

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.

17. Key Words (Suggested by Authors(s)) Obscuration computer code Shadow map Plate models Cone frustrum models Space Station applications	18. Distribution Statement ~~FEDD Distribution~~ Subject Category 32

19. Security Classif.(of this report) Unclassified	20. Security Classif.(of this page) Unclassified	21. No. of Pages 237	22. Price

Available: NASA's Industrial Applications Centers